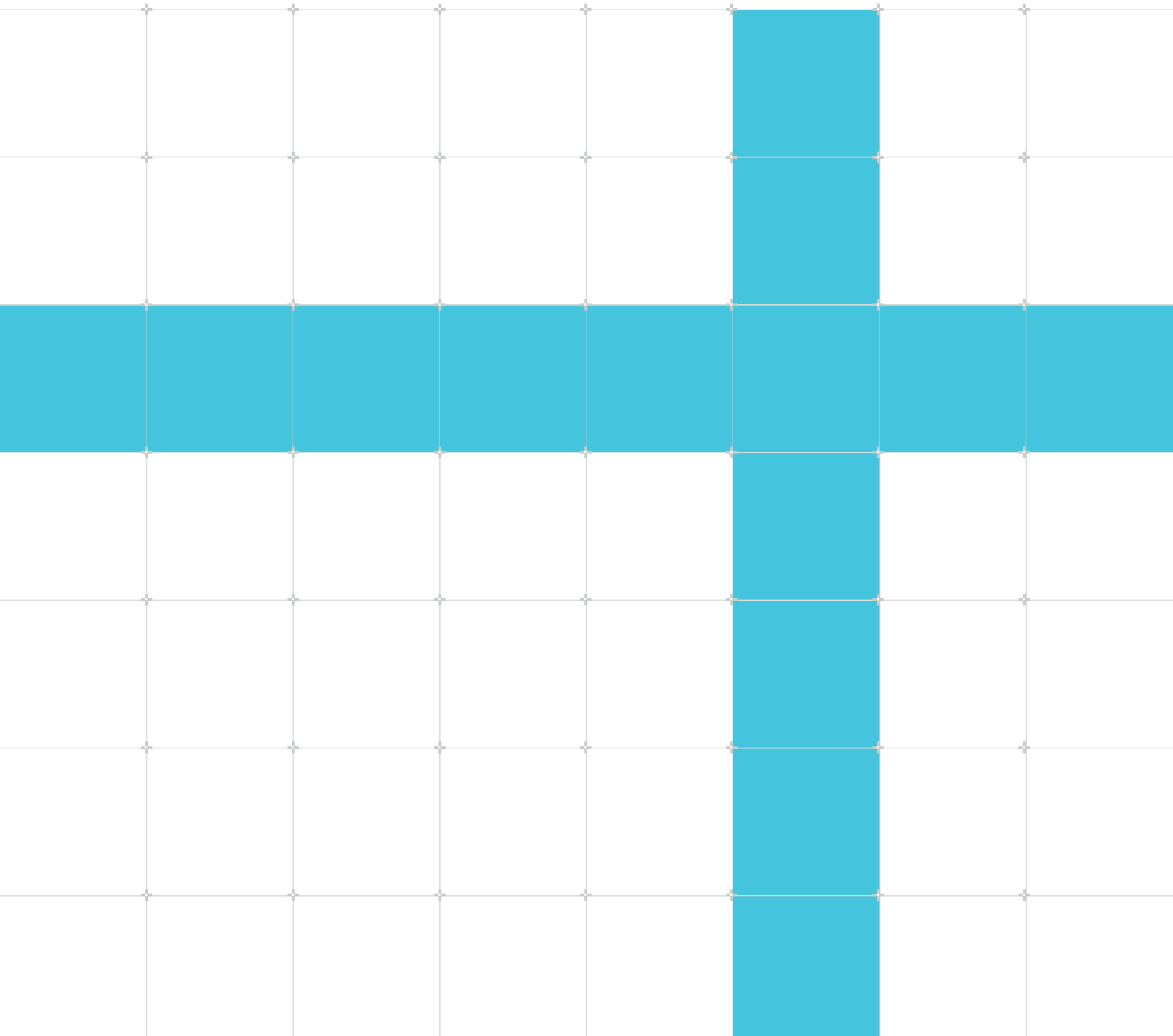
Arm® MMU L1 System Memory Management Unit

Revision r0p2

Technical Reference Manual

Non-Confidential

Issue 05
107957_0002_05_en



Arm® MMU L1 System Memory Management Unit Technical Reference Manual

This document (107957_0002_05_en) was issued on 2025-08-14. There might be a later issue at https://developer.arm.com/documentation/107957

The product revision is r0p2.

See also: Proprietary notice | Product and document information | Useful resources

Start reading

If you prefer, you can skip to the start of the content.

Intended audience

This document is written for system designers, system integrators, and programmers who are designing or programming a System-on-Chip (SoC) that uses MMU L1.

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.

Previous issues of this document included language that can be offensive. We have replaced this language. See Revision history on page 225.

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

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.

Contents

1. Overview of MMU L1

MMU L1 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 L1 translation tables in memory.

MMU L1 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. MMU L1 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. MMU L1 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 MMU L1 can define memory attributes for disabled and bypassed stages of translation.

MMU L1 uses inputs from the requester to identify a context. Configuration tables in memory define how MMU L1 is to translate each context, such as which translation tables to use.

MMU L1 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.

MMU L1 contains the following key components:

Translation Buffer Units (TBUs)
: The TBUs use a TLB to cache translation tables.

A Translation Control Unit (TCU)
: The TCU controls and manages address translations.

Distributed Translation Interface (DTI)
: Interconnect components that connect multiple TBUs to the TCU.

1.1 Compliance

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.

MMU L1 complies with, or implements, the specifications that we refer to in the following list:

Arm architecture

MMU L1 implements parts of the Arm®v8.5 Virtual Memory System Architecture (VMSA), as the *Arm® Architecture Reference Manual for A-profile architecture* defines. The *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2* describes the parts of VMSA that apply to MMU L1.

SMMU architecture

MMU L1 implements the SMMUv3.2 architecture, as the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2* defines.

AMBA Distributed Translation Interface protocol

MMU L1 implements the AMBA® Distributed Translation Interface (DTI) protocol, as the *AMBA® DTI Protocol Specification* defines. MMU L1 conveys DTI messages using an AXI5-Stream interface. The *AMBA® AXI-Stream Protocol Specification* defines the AXI5-Stream protocol.

AMBA ACE5-Lite and AMBA AXI5 protocol

MMU L1 complies with the AXI5 protocol, as the *AMBA® AXI Protocol Specification* defines.

AMBA APB protocol

MMU L1 complies with the AMBA APB5 protocol, as the *AMBA® APB Protocol Specification* defines.

AMBA LPI Q-Channel protocol

MMU L1 complies with the AMBA Low-Power Interface (LPI) Q-Channel, as the *AMBA® Low Power Interface Specification* defines.

1.2 Supported features

MMU L1 has features which are described in terms of compliance with architectures, support for interfaces, support for high-performance translation, support for flexible integration, and trace debugging.

MMU L1 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)
- Support for Page Request Interface (PRI), as SMMUv3.2 defines. PRI is an optional PCIe ATS extension that enables support for unpinned memory in PCIe.
- Support for Memory System Resource Partitioning and Monitoring (MPAM). This enables supervisory software such as an OS or Hypervisor to associate unique identifiers with each VM or application. Memory system components use the identifiers for allocation of resources to each VM or application.
- Support for Secure Exception Level 2 (Secure-EL2)
- Requesters can be stalled while a processor handles translation faults, enabling software support for on-demand paging
- Configuration tables in memory
- Queues in memory perform MMU L1 management. There is no requirement to stall a processor when it accesses MMU L1.
- A Performance Monitoring Unit (PMU) in each TBU and TCU that enables you to investigate MMU L1 performance
- Reliability, Availability, and Serviceability (RAS) features for RAM corruption detection and correction

Support for AMBA interfaces

- 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 AXI5 Distributed Virtual Memory (DVM) TCU table walk interface that enables Armv8.5 processors to perform shared TLB invalidate operations without accessing MMU L1 directly
- An AXI5 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 requesters 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
- WAKEUP signaling on all interfaces, including AXI5, DTI, and APB interfaces
- Support for AXI5 atomic transactions
- Support to integrate a Generic Interrupt Controller (GIC), 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 requesters being translated

- Communication between the TBU and the TCU over the AXI5-Stream protocol 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 Translation Lookaside Buffers (TLBs) Walk Caches (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 requesters 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

- Use an Arm CoreSight™ ELA-600 Embedded Logic Analyzer to debug MMU L1

1.3 Interfaces

MMU L1 uses various interfaces to communicate across multiple protocols.

Both the Transaction Control Unit (TCU) and Transaction Buffer Unit (TBU) support the following common interfaces:

- Clocks and resets
- Distributed Translation Interface (DTI)
- Tie-offs
- Interrupts

- Performance Monitoring Unit (PMU) snapshot
- Test and debug
- Low Power Interface (LPI) clock gating
- LPI powerdown

The TCU also supports the following interfaces:

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

The TBU also supports the following interfaces:

- Transaction buffer completer (TBS)
- Transaction buffer requester (TBM)

1.4 Configurable options

MMU L1 is highly configurable and provides configuration options for each of the main components.

Although not a comprehensive list of parameters for the TCU, you can configure the following parameters:

- 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 requesters

Although not a comprehensive list of parameters for the TBU, you can configure the following parameters:

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

For a comprehensive list of TCU and TBU parameters, see Configuration parameters and methodology.

Note

Depths are specified as a discrete number of entries.

You can also configure the DTI interconnect components to meet your system requirements. See Distributed Translation Interface overview, DTI interconnect interfaces, and Configuration parameters and methodology.

1.5 Product documentation and design flow

MMU L1 documentation relates to the design flow.

Documentation

The MMU L1 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 MMU L1. 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 MMU L1, then contact:

- The implementer to determine:
  - The build configuration of the implementation
  - The integration, if any, that was performed before implementing MMU L1
- 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 MMU L1 into an SoC. There is also a description of 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 document that is only available to licensees.

Design flow

MMU L1 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 MMU L1, and tests the required application software. Each process is separate, and can include implementation and integration choices that affect the behavior and features of MMU L1.

The operation of the final MMU L1 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 MMU L1 by tying inputs to specific values. These configurations affect the start-up behavior before any software configuration is made.

Software configuration

The programmer configures MMU L1 by programming particular values into registers. This configuration affects the behavior of MMU L1.

The TCU and TBU have configurable options, see Configurable options. To configure different types of configuration parameters, see Configuration parameters and methodology. MMU L1 complies with several architectures and protocols, see Compliance.

2. Constraints and limitations of use

Certain usage constraints and limitations apply to MMU L1.

SMMUv3.2 implementation constraints

MMU L1 implements the SMMUv3.2 architecture. However there are some optional features therefore, MMU L1 does not implement all the SMMUv3.2 architecture features. See SMMUv3.2 implementation constraints.

AMBA implementation constraints

AMBA® constraints include, but are not limited to, AXI5 feature support, interface attribute handling, transaction types and transactions that can or cannot cause a translation fault. See AMBA implementation constraints.

MPAM constraints

Certain Memory System Resource Partitioning and Monitoring (MPAM) registers are implemented. See MPAM constraints.

2.1 SMMUv3.2 implementation constraints

MMU L1 implements the SMMUv3.2 architecture. However there are some optional features therefore, MMU L1 does not implement all the SMMUv3.2 architecture features.

The optional SMMUv3.2 architecture features to implement are:

- ID register architectural values
- Non-implemented commands and events
- **IMPLEMENTATION DEFINED** fields
- Non-implemented registers

2.1.1 ID register architectural values

MMU L1 contains ID register architectural values.

The following table describes the architectural values for MMU L1 from the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2* that the SMMUv3.2 ID registers expose.

Table 2-1: Architectural values for the SMMUv3.2 ID registers

Register	Field	Value	Description
SMMU_IDR0	S2P	1	Stage 2 translation is supported.
SMMU_IDR0	S1P	1	Stage 1 translation is supported.
SMMU_IDR0	TTF	11	AArch64 and AArch32 translation is supported.
SMMU_IDR0	COHACC	sup_cohacc	Coherent accesses are supported, system configuration option.

Register	Field	Value	Description
SMMU_IDR0	BTM	sup_btm	Broadcast TLB maintenance is supported, system configuration option.
SMMU_IDR0	HTTU[1:0]	{sup_httu, 0b0}	Access and dirty flag update are supported, system configuration option.
SMMU_IDR0	DORMHINT	0	Dormant hint is not supported.
SMMU_IDR0	Hyp	1	EL2-E2H is supported.
SMMU_IDR0	ATS	1	ATS is supported.
SMMU_IDR0	NS1ATS	0	Split-stage (stage 1-only) is supported.
SMMU_IDR0	ASID16	1	16-bit ASID is supported.
SMMU_IDR0	MSI	1	Message Signaled Interrupts (MSIs) are supported.
SMMU_IDR0	SEV	sup_sev	Send event is supported, system configuration option.
SMMU_IDR0	ATOS	0	ATOS is not supported.
SMMU_IDR0	PRI	1	PRI is supported.
SMMU_IDR0	VMW	1	VMID wildcard matching is supported.
SMMU_IDR0	VMID16	1	16-bit VMIDs are supported.
SMMU_IDR0	CD2L	1	2-level context descriptor tables are supported.
SMMU_IDR0	VATOS	0	Virtual ATOS is not supported.
SMMU_IDR0	TTENDIAN	0b00	Mixed-endian translation walks are supported.
SMMU_IDR0	STALL_MODEL	{0b0, SMMU_S_CR0.NSSTALLD}	Stall and terminate models that are supported unless the Secure world disables Non-secure stalling.
SMMU_IDR0	TERM_MODEL	0	Terminating a transaction with **RAZ/WI** is supported.
SMMU_IDR0	ST_LEVEL	01	2-level stream table is supported.
SMMU_IDR1	SIDSIZE	32	32-bit StreamIDs are supported.
SMMU_IDR1	SSIDSIZE	20	20-bit SubstreamIDs are supported.
SMMU_IDR1	PRIQS	0b10011	2^{19} PRI queue entries are supported.
SMMU_IDR1	EVENTQS	0b10011	2^{19} Event queue entries are supported.
SMMU_IDR1	CMDQS	0b10011	2^{19} Command queue entries are supported.
SMMU_IDR1	ATTR_PERMS_OVR	1	Incoming permission attributes can be overridden.
SMMU_IDR1	ATTR_TYPES_OVR	1	Incoming memory attributes can be overridden.
SMMU_IDR1	REL	0	N/A, not fixed base addresses.
SMMU_IDR1	QUEUES_PRESET	0	Not fixed queue base addresses.
SMMU_IDR1	TABLES_PRESET	0	Not fixed table base addresses.
SMMU_IDR2	BA_VATOS	0	N/A, VATOS is not supported.
SMMU_IDR3	HAD	1	Hierarchical attribute disable is supported
SMMU_IDR3	PBHA	1	Page-based hardware attributes are supported.
SMMU_IDR3	XNX	1	EL0/EL1 stage 2 execute control is supported.
SMMU_IDR3	PPS	1	PASID, when present always used in PRI auto-generated response.
SMMU_IDR3	MPAM	1	MPAM is supported.
SMMU_IDR3	FWB	1	S2 control of memory type is supported.
SMMU_IDR3	STT	1	Small translation tables are supported.
SMMU_IDR3	RIL	1	Range-based invalidation and Level hint are supported.
SMMU_IDR3	BBML	2	Break before Make level 2 is supported.

Register	Field	Value	Description
SMMU_IDR4	IMPDEF	0	No **IMPLEMENTATION DEFINED** features.
SMMU_IDR5	OAS	sup_oas	Output address size, system configuration option.
SMMU_IDR5	GRAN4K	1	4K translation granule is supported.
SMMU_IDR5	GRAN16K	1	16K translation granule is supported.
SMMU_IDR5	GRAN64K	1	64K translation granule is supported.
SMMU_IDR5	VAX	01	Virtual addresses of 52 bits per CD.TTBx (Context Descriptor Translation Table Base T0 or T1) are supported.
SMMU_IDR5	STALL_MAX	`TCUCFG_XLATE_SLOTS`	Maximum number of outstanding stalled transactions.
SMMU_IIDR	Implementor	`0x43B`	Arm implementation.
SMMU_IIDR	Revision	MAX(p_level, ecorevnum)	Where p_level is: **2** For p2
SMMU_IIDR	Variant	0	Product variant, or major revision is r0.
SMMU_IIDR	ProductID	`0x48A`	MMU L1 TCU ID.
SMMU_AIDR	ArchMinorRev	2	Architectural minor revision is 2.
SMMU_AIDR	ArchMajorRev	0	Architectural minor revision is SMMUv3.
SMMU_S_IDR0	MSI	1	Secure MSIs are supported.
SMMU_S_IDR0	STALL_MODEL	`0b00`	Stall and terminate model is supported.
SMMU_S_IDR1	S_SIDSIZE	32	32-bit Secure StreamIDs are supported.
SMMU_S_IDR1	SEL2	1	Secure EL2 is supported.
SMMU_S_IDR1	SECURE_IMPL	1	2 Security states are implemented.
SMMU_S_IDR2	BA_S_VATOS	0	N/A, VATOS is not supported.
SMMU_S_IDR3	SAMS	1	Secure ATS maintenance is not implemented.
SMMU_S_IDR4	IMPDEF	0	No **IMPLEMENTATION DEFINED** features.

The following table shows the architectural options for the MPAM ID registers.

Table 2-2: Architectural options for the MPAM ID registers

Register	Field	Value	Description
SMMU_MPAMIDR	PMG_MAX	1	Maximum PMG value that is permitted to be used in Non-secure state.
SMMU_MPAMIDR	PARTID_MAX	$2^{\texttt{TCUCFG\_PARTID\_WIDTH}} - 1$	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
SMMU_S_MPAMIDR	PMG_MAX	1	Maximum permitted PMG value for use in Secure state.
SMMU_S_MPAMIDR	PARTID_MAX	$2^{\texttt{TCUCFG\_PARTID\_WIDTH}} - 1$	Maximum permitted PARTID value for use in Secure state.

2.1.2 Non-implemented commands and events

MMU L1 does not implement all the Events and Commands that the architecture defines. In the Event queue, specific events are not generated, and in the Command queue, specific commands are accepted but silently ignored.

Event queue

MMU L1 does not generate the following events:

- F_UUT
- F_TLB_CONFLICT
- F_CFG_CONFLICT
- E_PAGE_REQUEST
- IMPDEF_EVENTn

Command queue

MMU L1 accepts the following commands but silently ignores them:

- 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 the CMD_ATC_INV command in the Secure Command queue, it results in a Secure Command queue error with reason code CERROR_ILL.

2.1.3 Non-implemented registers

MMU L1 does not implement several optional registers and Performance Monitor Counter Group (PMCG) registers that the architecture defines.

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

- SMMU_IDR4
- SMMU_S_IDR4
- SMMU_STATUSR
- SMMU_GATOS_*
- SMMU_S_GATOS_*
- SMMU_VATOS_*
- SMMU_S_VATOS_*

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

- SMMU_PMCG_IRQ_CFG0

- SMMU_PMCG_IRQ_CFG1
- SMMU_PMCG_IRQ_CFG2
- SMMU_PMCG_IRQ_STATUS
- SMMU_PMCG_GMPAM
- SMMU_PMCG_MPAMIDR
- SMMU_PMCG_S_MPAMIDR

2.1.4 IMPLEMENTATION DEFINED fields

Unless otherwise specified, **IMPLEMENTATION DEFINED** fields in structures that MMU L1 generates are 0.

IMPLEMENTATION DEFINED fields in structures that MMU L1 reads are ignored, unless otherwise specified.

2.2 AMBA implementation constraints

There are several AMBA implementation constraints in MMU L1.

These constraints include, but are not limited to, AXI5 feature support, interface attribute handling, transaction types, and transactions that can or can not cause a translation fault:

AXI5 feature support
A feature list that shows which AXI5 features MMU L1 supports. See AXI5 feature support.

SLVERR and DECERR
How the TCU Queue and Table Walk (QTW) interface treats SLVERR and DECERR signals. See SLVERR and DECERR.

Attribute Handling
Constraints on how MMU L1 handles untranslated (non-ATS) transactions, fully-translated (full ATS) transactions, and partially-translated (Split-stage ATS) transactions. See Attribute handling.

AxCACHE encodings
MMU L1 only uses specific AxCACHE encodings. See AxCACHE encodings.

Completer interface attribute handling
The TBU TBS interface converts the incoming AXI5 attributes into Armv8 attributes. See Completer interface attribute handling.

Requester interface attribute handling
MMU L1 handles requester interface attributes through normalization. See Requester interface attribute handling.

AxREGION

The TCU QTW/DVM interface drives this signal to 0. The TBM interface drives this signal with the value received on the TBS interface. See AxREGION.

DVM interface

The Distributed Virtual Memory (DVM) interface supports DVM operations and DVM complete messages. See DVM interface.

Internally terminated transactions

Transactions that are terminated inside the MMU L1 TBU return with all RUSER and BUSER bits as zero. See Internally terminated transactions.

Transaction types

MMU L1 supports several special transaction types, distinguished by a nonzero encoding of AxSNOOP. See Transaction types.

Transactions that can cause a translation fault

Certain transactions can result in a translation fault. There is also certain behavior associated with such transactions. See Transactions that can cause a translation fault.

Transactions that do not cause a translation fault

Certain transactions cannot result in a translation fault, and certain behavior is associated with such transactions. See Transactions that do not cause a translation fault.

2.2.1 AXI5 feature support

MMU L1 supports many AXI5 features.

The following table shows the AXI5 features that MMU L1 supports. The table also shows the specific version of the *AMBA® AXI Protocol Specification*, to which the particular feature was first added.

Table 2-3: AXI5 feature support

Specification issue	Interface properties	TBU - TBS	TBU - TBM	TCU
E	DVM_v8	False	False	True
E	Multi_Copy_Atomicity	True	True	True
E	Ordered_Write_Observation	True	True	False
F	Atomic_Transactions	True	True	True
F	Cache_Stash_Transactions	True	True	False
F	Check_Type	False	False	False
F	Coherency_Connection_Signals	False	False	True
F	DeAllocation_Transactions	True	True	False
F	DVM_v8.1	False	False	True
F	Loopback_Signals	True	True	False
F	NSAccess_Identifiers	False	False	False

Specification issue	Interface properties	TBU - TBS	TBU - TBM	TCU
F	Persist_CMO	True	True	False
F	Poison	True	True	True
F	QoS_Accept	False	False	False
F	Trace_Signals	False	False	False
F	Untranslated_Transactions	True (V3)	False 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 L1 does not require these signals and you can ignore them.	False
F	Wakeup_Signals	True	True	True
G	CMO_On_Read	True	True	True
G	CMO_On_Write	False	False	False
G	MPAM_Support	False	True	True
G	Read_Data_Chunking	True	True	False
G	Read_Interleaving_Disabled	False	False	False
G	Unique_ID_Support	True	True	True
H	Consistent_DECERR	False	False	False
H	DVM_Message_Support	False	False	Completer
H	DVM_v8.4	False	False	True
H	Exclusive_Accesses	True	True	True Set to True because the AxLOCK signal is present. However the TCU does not generate exclusives and internally ties this signal to 0. The interfaces need to be prepared to connect the AxLOCK.
H	Max_Transaction_Bytes	4096	4096	64
H	MTE_Support	Basic	Basic	False
H	Prefetch_Transaction	False	False	False
H	Regular_Transactions_Only	False	False	False
H	Shareable_Transactions	True	True	True
H	Write_Plus_CMO	False	False	False
H	WriteZero_Transaction	False	False	False
J	Busy_support	False	False	False
J	DVM_v9.2	False	False	False
J	InvalidateHint_Transaction	True	True	False

Specification issue	Interface properties	TBU - TBS	TBU - TBM	TCU
J	PBHA_Support	False	True	True The TCU drives the PBHA signals, AxPBHA on the AxUSER bits of the QTW interface.
J	RME_Support	False	False	False
J	Shareable_Cache_Support	False	False	False
J	UnstashTranslation_Transaction	True	False	False
J	WriteDeferrable_Transaction	False	False	False

2.2.2 SLVERR and DECERR

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 response type is propagated to the TBS interface.

2.2.3 Attribute handling

MMU L1 handles untranslated or non-ATS transactions, fully-translated or full ATS transactions, and partially-translated or Split-stage ATS transactions.

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

Untranslated or non-ATS transaction

The SMMU applies attributes that a combination of the input attributes, Stream Table Entry (STE) overrides, and translation table descriptors determine.

Fully-translated or 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 or Split-stage ATS transaction

The ATS translation response from the TCU to the PCIe Root Port 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 more information on these transactions and their attributes, see the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2*.

2.2.3.1 AxCACHE encodings

MMU L1 only generates specific AxCACHE encodings.

Where there are multiple legal values for AxCACHE as the *AMBA® AXI Protocol Specification* describes, the canonical, non-bracketed, one is used. Therefore, the following table shows the ARCACHE and AWCACHE encodings used for the AMBA Memory Type.

Table 2-4: 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

2.2.3.2 Completer interface attribute handling

The TBU TBS interface converts the incoming AXI5 attributes into Armv8 attributes.

The following table shows the completer interface attribute handling in MMU L1. The following terms are used in the table:

NSH

Non-shareable (`0b00`)

SH

Shareable (`0b10`)

SYS

System (`0b11`)

MMU L1 uses the `0b10` encoding for Shareable on AXI as recommended in Domain signaling in the *AMBA® AXI Protocol Specification*.

Table 2-5: Completer interface attribute handling

AXI5 AxCACHE	AXI5 AxDOMAIN	Armv8 Memory Type	Armv8 Shareability	Description
Device Non-bufferable	SYS	Device-nGnRnE	OSH	-
Device Bufferable	SYS	Device-nGnRE	OSH	-

AXI5 AxCACHE	AXI5 AxDOMAIN	Armv8 Memory Type	Armv8 Shareability	Description
Normal Non-cacheable Bufferable, Normal Non-cacheable Non-bufferable,	NSH/SH/SYS	Normal-iNC-oNC	OSH	Normal Non-cacheable Non-bufferable, is a deprecated AxCACHE type and is handled in the same way as Normal Non-cacheable Bufferable.
Write-Through No-Allocate, Write-Through Read-Allocate, Write-Through Write-Allocate, Write-Through Read and Write-Allocate	NSH/SH	Normal-iNC-oNC	OSH	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/SH	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.

Completer interface AxPROT handling

There are several rules that apply when the normalization step occurs in MMU L1. For example, when instruction writes are converted into data writes, transactions that are already translated are fixed to Privileged Data, transactions for ATST flows are treated as Unprivileged Data and speculative operations are treated as Unprivileged over DTI.

The following describes each rule the normalization step applies:

- Instruction writes and InvalidateHint are converted into data writes. This rule includes Atomic operations. For more information on Instruction writes and InvalidateHint, see Transaction types.
  - AWPROT_M[2] = 0
- The transactions for ATST flows are treated as Unprivileged Data
  - If AxMMUFLOW_S = ATST, then:
    - ARPROT_M[2:0] = {0b0, ARPROT_S[1], 0b0} and
    - AWPROT_M[2:0] = {0b0, AWPROT_S[1], 0b0}


Note

When AxMMUVALID is LOW, AxMMUFLOW is not applicable, so there is no ATST flow.

- Speculative operations are treated as Unprivileged over DTI.
  - If AWSNOOP_S inside {StashOnceShare, StashOnceUnique, InvalidateHint}, then DTI_REQ.PnU = 0 because speculative operations have speculative permission settings over DTI. This applies to StashTranslation transactions, however UnstashTranslation transactions never issue DTI requests.


Note

In most cases this means that the downstream AXI transfer is also unprivileged. However, if the DTI_RSP.PRIVCFG = 2'b00 (use-incoming) or 2'b11 (privileged), it might result in AxPROT_M[0] = 1. Therefore, the incoming attribute must be stored in the TBU.

- Transactions with physical addresses that do not need MMU translation are fixed to Privileged Data
  - If AxMMUVALID_S = 0, then:
    - ARPROT_M[2:0] = {`0b0` ARPROT_S[1], `0b1`} and
    - AWPROT_M[2:0] = {`0b0` AWPROT_S[1], `0b1`}
- Non-secure streams are always treated as Non-secure.
  - If AxMMUSECSID_S = 0, then AxPROT_M[1] = 1

2.2.3.3 Requester interface attribute handling

MMU L1 handles requester interface attributes through normalization.

Normalization

Both AMBA requester interfaces, that is the TBU TBM interface and the TCU QTW interface, carry normalized attributes using the standard 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, System 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 AXI5 signals. The following terms are used in the table:

NSH

Non-shareable (`0b00`)

SH

Shareable (`0b10`)

SYS

System (`0b11`)

Exclusive access transactions in AXI have certain restrictions on the memory type, see Exclusive access restrictions in the *AMBA® AXI Protocol Specification*.

When an exclusive access transaction is translated into an Inner Shareable or Outer Shareable transaction, MMU L1 transforms the transaction to non-exclusive. This is the recommended behavior as specified in Treatment of AMBA Exclusives from client devices in the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2*. Regardless of shareability, MMU L1 extends this transformation to all exclusive access transactions that are translated into a Normal Inner Write-Back Outer Write-Back transaction.

Table 2-6: Armv8 transaction types translated into AMBA AXI5 signals

Armv8 Memory Type	AxCACHE (TBM, QTW)	AxDOMAIN (TBM, QTW)	AxLOCK (TBM)	AxUSER outer cacheable bit (TBM)
Device-nGnRnE	Device Non-bufferable	SYS	TBS AxLOCK value	0
Device-GRE, Device-nGRE, Device-nGnRE	Device Bufferable	SYS	TBS AxLOCK value	0
Normal-iNC-oNC, Normal-iWT-oNC, Normal-iWB-oNC	Normal Non-cacheable Bufferable	SYS	TBS AxLOCK value	0

Armv8 Memory Type	AxCACHE (TBM, QTW)	AxDOMAIN (TBM, QTW)	AxLOCK (TBM)	AxUSER outer cacheable bit (TBM)
Normal-iNC-oWT, Normal-iWT-oWT, Normal-iWB-oWT, Normal-iNC-oWB, Normal-iWT-oWB	Normal Non-cacheable Bufferable	SYS	TBS AxLOCK value	1
Normal-iWB-oWB	Write-Back No-Allocate/Write-Back Read-Allocate /Write-Back Write-Allocate Write-Back Read and Write-Allocate, depending on the Armv8 outer allocate hints.	AxBURST = FIXED: NSH See the following section on Fixed burst termination. AxBURST != FIXED: NSH or SH (0b10), depending on the Armv8 Shareability.	0	1

Fixed burst termination

If a transaction with AxBURST == FIXED becomes Shareable as a result of the translation process, MMU L1 terminates the transaction.

MMU L1 internally terminates successfully translated transactions that result in an AxBURST = FIXED and AxDOMAIN = SH combination. This termination results in a SLVERR on the TBS interface.

These transactions are not pushed to the downstream interface and therefore MMU L1 does not apply normalization rules to them. When AxMMUVALID = 0, MMU L1 cannot receive FIXED SH bursts on the TBS side because of AXI protocol restrictions. Attribute conversions cannot result in FIXED SH bursts downstream.

2.2.4 AxREGION

The AxREGION signals on TCU QTW are driven as 0. On TBM, they reflect the values of the corresponding TBS transaction.

2.2.5 DVM interface

The Distributed Virtual Memory (DVM) interface supports DVM operations and DVM complete messages.

Supported DVM operations

In response to an ACSNOOP request, the CRRESP signal 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 messages

DVM complete messages are presented with:

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

2.2.6 Internally terminated transactions

Transactions that are terminated inside the MMU L1 TBU return with all RUSER and BUSER bits as zero.

Terminated transactions are not issued downstream but are terminated after the TCU response has arrived. Termination can occur when the following occur:

- A translation response from the TCU is a non-stalling fault response.
- A translation response from the TCU is missing permissions, or it is of a non-compatible memory type.
- Transactions that have FIXED burst type and have successful translations but result in SH shareability after attribute merging.
- An AxMMUVALID_S = 0 transaction is outside the output address space.

Translation faults can cause transactions to be terminated. When a translation fault causes a transaction to terminate, the TBU generates a SLVERR or TRANSFAULT response on the BRESP or RRESP signals.

2.2.7 Transaction types

MMU L1 supports several special transaction types, distinguished by a nonzero encoding of AxSNOOP.

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

The following transactions, with ARSNOOP = `0b0000` and AWSNOOP = `0b00000`, require both read and write permissions on the appropriate privilege level:

- AtomicLoad
- AtomicSwap
- AtomicCompare
- AtomicStore

An ordinary read or ordinary write is one with ARSNOOP = `0b0000` and AWSNOOP = `0b00000`, that is, depending on AxDOMAIN, and whether it is a read or a write, one of the following:

- ReadNoSnoop
- ReadOnce
- WriteNoSnoop
- WriteUniquePtl

MMU L1 always treats the following transactions as Normal Write-Back:

- ReadOnceCleanInvalid (ROCI)
- ReadOnceMakeInvalid (ROMI)
- WriteUniquePtlStash
- WriteUniqueFullStash
- StashOnceShared
- StashOnceUnique

Although AXI allows ReadOnceCleanInvalid (ROCI), ReadOnceMakeInvalid (ROMI), StashOnceShared, StashOnceUnique, WriteUniquePtlStash, and WriteUniqueFullStash transactions to be Non-cacheable (NC), this is not recommended in the *AMBA® AXI Protocol Specification*. LTI only allows writeback for the corresponding transaction types. Although MMU L1 does not implement LTI, it is restricted to only allow writeback to provide consistency with LTI.

For compatibility, MMU L1 does not support making these transaction types NC either and always considers such transactions as Write-Back (WB) Allocate. You must not generate cacheability as Non-cacheable or Write-Through for these transaction types on an AXI interface to be compatible with the SMMU architecture. Therefore, MMU L1 can generate an architecturally incorrect output with WB, instead of downgrading. MMU L1 also converts all Write-Through (WT) transactions to NC therefore, this limitation also applies to the same transaction types issued as WT. We do not recommend using WT transactions.

AXI allows InvalidateHint Cache Maintenance Operations (CMO) to be either Data or Instruction access. Although MMU L1 does not implement LTI, for compatibility it always allows only Data access for the corresponding Destructive Hint Cache Maintenance Operation (DHCMO) transaction type. For compatibility, MMU L1 always treats incoming InvalidateHint transactions as Data access, regardless of AWPROT_S[2] and outputs all InvalidaHint transactions as Data AWPROT_M[2]=`0b0`.

Note

To maintain compatibility with the SMMU architecture, you must not generate InvalidateHint transaction as Instruction access on an AXI interface.

This limitation only applies to the AXI interfaces. If the InvalidateHint transaction becomes an Instruction access as a result of the translation process, then internally, the permission checking is carried out accordingly.

2.2.7.1 Transactions that can cause a translation fault

In an MMU L1 system, some transactions can result in a translation fault. There is also certain behavior associated with such transactions.

MMU L1 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.

MMU L1 treats the following transactions as ordinary writes when calculating translation faults:

- WriteUniquePtlStash
- WriteUniqueFullStash
- WriteUniqueFull

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

The following transactions do not have a memory type:

- CleanShared
- CleanInvalid
- MakeInvalid
- CleanSharedPersist

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.

MMU L1 treats transactions that pass the translation fault check as follows:

MakeInvalid transactions

MMU L1 converts MakeInvalid transactions to CleanInvalid transactions, unless the translation also grants write permission and Destructive Read Enable (DRE) permission.

ReadOnceMakeInvalid and ReadOnceCleanInvalid transactions

MMU L1 converts ReadOnceMakeInvalid (ROMI) and ReadOnceCleanInvalid (ROCI) transactions into ordinary reads if the final transaction attributes are not Normal Write-Back or not Shareable.

In the case of ROMI if this conversion does not take place, MMU L1 checks the write and DRE permissions. Unless both permissions are granted the ROMI is converted into a ROCI transaction.

WriteUniquePtlStash and WriteUniqueFullStash transactions

MMU L1 converts WriteUniquePtlStash and WriteUniqueFullStash transactions to ordinary write transactions if either:

- The translation does 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.

WriteUniqueFull transactions

If the final transaction shareability is not Shareable, MMU L1 converts the transaction to an ordinary write.

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

2.2.7.2 Transactions that do not cause a translation fault

In an MMU L1 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
- InvalidateHint
- UnstashTranslation

If any of these transactions, except the UnstashTranslation, require a translation request to the TCU, the TBU issues a speculative translation request on the DTI interconnect.

Note

UnstashTranslation is a hint that the translation with the relevant transaction address and StreamID must be deallocated from the Main and Micro Translation Lookaside Buffers (TLBs). Therefore, an UnstashTransalation transaction can never result in a translation request.

StashOnceShared and StashOnceUnique transactions

StashOnceShared and StashOnceUnique transactions are terminated in the TBU, with a BRESP value of OKAY, when any of the following cases apply:

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

MMU L1 always generates a value of OKAY value when a StashOnceShared or a StashOnceUnique transaction is terminated in the TBU. This behavior applies even when a StreamDisabled or GlobalDisabled translation response causes the transaction to be terminated.

InvalidateHint transactions

InvalidateHint transactions are terminated with BRESP value of OKAY unless they meet a specific set of requirements.

An InvalidateHint transaction is never terminated if AWMMUVALID_S == 0, or if StreamBypass or GlobalBypass is enabled.

Additionally, an InvalidateHint transaction is never terminated if AWMMUVALID_S == 1 and:

- Either its final instruction or data attribute is an instruction access, and:
  - Its outgoing AxPROT Non-secure bit, calculated from the TCU security configuration, indicates Secure or,
  - Its Secure Stream Identifier (SECSID) attribute is Non-secure or,
  - Its SECSID attribute is Secure, and Non-secure instruction read permission is granted.
- Or, its final instruction or data attribute is data.

The InvalidateHint transaction is not terminated if it is not a global or stream bypassing transaction, and also has the following permissions:

- Write permission at the appropriate privilege level AND
- Destructive Read Enable (DRE) permission AND
- IF the final instruction or data attribute indicates an instruction access the transaction requires execute permission at the appropriate privilege level
- ELSE IF the final instruction or a data access attribute indicates data the transaction requires read permission at the appropriate privilege level.

The InvalidateHint transaction is terminated in all other scenarios.

StashTranslation and UnstashTranslation

MMU L1 never propagates StashTranslation or UnstashTranslation transactions downstream, and uses StashTranslation only to prefetch Main TLB and MicroTLB contents. UnstashTranslation never creates a DTI request. MMU L1 always terminates StashTranslation and UnstashTranslation 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. UnstashTranslation is not speculative in this scenario.

Use a StashTranslation transaction to prefetch translations

A StashTranslation transaction can be used to prefetch translations into the Main TLB or MicroTLB of MMU L1. However, for any subsequent transactions to take advantage of these prefetched translations into the Main TLB or MicroTLB, they must use the same StreamID and SECSID as the original prefetch.

The StreamID and SECSID identify a translation context. Using a different StreamID or SECSID for a subsequent transaction means that this subsequent transaction uses a different translation context to the translation that has been prefetched into the TLB, and might lead to a TLB miss.

StashTranslation and direct indexing

When `TBUCFG_DIRECT_IDX` = 1, StashTranslation transactions always miss in uTLB and perform a lookup in Main TLB (MTLB).

MicroTLB lookup and MicroTLB miss PMU event counters do not record these lookups and misses.

2.3 MPAM constraints

MMU L1 implements Memory System Resource Partitioning and Monitoring (MPAM) with some constraints. Certain MPAM registers are implemented. Registers that are not implemented are not described.

For more information, see the *Arm® Memory System Resource Partitioning and Monitoring (MPAM) System Component Specification*.

MPAM cache maximum capacity partitioning manages the following:

- TBU MTLB
- TCU configuration cache
- Walk cache

No other mechanism from the *Arm® Memory System Resource Partitioning and Monitoring (MPAM) System Component Specification*, is implemented.

2.3.1 TCU MPAM

MMU L1 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 2-7: TCU MPAM registers implemented

Register	Field	Value	Description
MPAMF_IDR_LO (0x0000, Shared)	PARTID_MAX	1, 63, 511	Set to $2^{\texttt{TCUCFG\_PARTID\_WIDTH}} - 1$
MPAMF_IDR_LO (0x0000, Shared)	PMG_MAX	1	2 Non-secure Performance Monitoring groups supported for each PARTID
MPAMF_IDR_LO (0x0000, Shared)	HAS_CCAP_PART	1	Supports cache maximum capacity partitioning
MPAMF_IDR_LO (0x0000, Shared)	HAS_CPOR_PART	0	Cache portion partitioning not supported
MPAMF_IDR_LO (0x0000, Shared)	HAS_MBW_PART	0	Memory bandwidth partitioning not supported
MPAMF_IDR_LO (0x0000, Shared)	HAS_PRI_PART	0	Priority partitioning not supported
MPAMF_IDR_LO (0x0000, Shared)	EXT	1	EXTended MPAMF_IDR
MPAMF_IDR_LO (0x0000, Shared)	HAS_IMPL_IDR	0	Does not have **IMPLEMENTATION SPECIFIC** partitioning features
MPAMF_IDR_LO (0x0000, Shared)	HAS_MSMON	1	Supports performance monitoring by matching a combination of PARTID and PMG
MPAMF_IDR_LO (0x0000, Shared)	HAS_PARTID_NRW	0	Does not have MPAMF_PARTID_NRW_IDR, MPAMCFG_INTPARTID, or intPARTID mapping support
MPAMF_IDR_HI (0x0004, Shared)	HAS_RIS	1	Has Resource Instance Selector
MPAMF_IDR_HI (0x0004, Shared)	NO_IMPL_PART	1	There are no **IMPLEMENTATION DEFINED** resource controls that MPAMF_IMPL_IDR defines

Register	Field	Value	Description
MPAMF_IDR_HI (0x0004, Shared)	NO_IMPL_MSMON	1	There are no **IMPLEMENTATION DEFINED** resource monitors that MPAMF_IMPL_IDR defines
MPAMF_IDR_HI (0x0004, Shared)	HAS_EXTD_ESR	0	MPAMF_ESR is 64-bits. Not relevant because HAS_ESR is 0.
MPAMF_IDR_HI (0x0004, Shared)	HAS_ESR	0	MPAMF_ESR and MPAMF_ECR are not implemented
MPAMF_IDR_HI (0x0004, Shared)	RIS_MAX	1	Maximum RIS value used in the Memory-System Component (MSC). Set to 1.
MPAMF_SIDR (0x0008, S-Only)	S_PARTID_MAX	1, 63, 511	Set to $2^{\texttt{TCUCFG\_PARTID\_WIDTH}} - 1$
MPAMF_SIDR (0x0008, S-Only)	S_PMG_MAX	1	2 Secure Performance Monitoring groups supported per PARTID
MPAMF_MSMON_IDR (0x0080, Shared)	MSMON_CSU	1	Performance monitor supported for Cache Storage Usage by PARTID and PMG
MPAMF_MSMON_IDR (0x0080, Shared)	MSMON_MBWU	0	No performance monitor for memory bandwidth usage by PARTID and PMG
MPAMF_MSMON_IDR (0x0080, Shared)	HAS_LOCAL_CAPT_EVNT	1	Has the local capture event generator and the MSMON_CAPT_EVNT register
MPAMF_CCAP_IDR (0x0038, Shared)	CMAX_WD	8	256 fractions are supported
MPAMF_CSUMON_IDR (0x0018 Shared)	NUM_MON	4	4 monitoring counters are implemented
MPAMF_CSUMON_IDR (0x0018 Shared)	HAS_CAPTURE	1	Has an MSMON_CSU_CAPTURE register for every MSMON_CSU and supports the capture event behavior
MPAMF_IIDR (0x0018, Shared)	All fields	All fields valid	Implementation ID Register
MPAMF_AIDR (0x0020, Shared)	ArchMajorRev	0x1	MPAM architecture v1.1
MPAMF_AIDR (0x0020, Shared)	ArchMinorRev	0x1	MPAM architecture v1.1
MPAMCFG_PART_SEL (0x0100, Shared)	PARTID_SEL	Bits [(TCUCFG_PARTID_WIDTH – 1):0] valid	Can select up to 512 partitions to configure, based on TCUCFG_PARTID_WIDTH
MPAMCFG_PART_SEL (0x0100, Shared)	RIS	Bits [27:24] valid	Resource Instance Selector
MPAMCFG_CMAX (0x0108, Banked)	CMAX	Bits [15:8] valid	Can choose up to 256 fractions
MSMON_CFG_MON_SEL (0x0800 Banked)	MON_SEL	Bits [1:0] valid	Selects the monitor to configure
MSMON_CFG_MON_SEL (0x0800 Banked)	RIS	Bits [27:24] valid	Resource Instance Selector
MSMON_CFG_CSU_FLT (0x0810, Banked)	PARTID	Bits [(TCUCFG_PARTID_WIDTH – 1):0] valid	Can select up to 512 partitions to configure, based on TCUCFG_PARTID_WIDTH
MSMON_CFG_CSU_FLT (0x0810, Banked)	PMG	0	Can select up to PMG number 1

Register	Field	Value	Description
MSMON_CFG_CSU_CTL (0x0818, Banked)	EN	Valid field	The monitor instance is enabled or disabled to collect information
MSMON_CFG_CSU_CTL (0x0818, Banked)	CAPT_EVNT	3'b111	Capture occurs when a MSMON_CAPT_EVNT register is written. All other values are not supported.
MSMON_CFG_CSU_CTL (0x0818, Banked)	CAPT_RESET	**RES0**	There is no reason to ever reset a CSU monitor
MSMON_CFG_CSU_CTL (0x0818, Banked)	OFLOW_STATUS	**RES0**	Overflow is not possible for a CSU monitor
MSMON_CFG_CSU_CTL (0x0818, Banked)	OFLOW_INTR	**RES0**	This MPAM implementation does not support OFLOW_INTR
MSMON_CFG_CSU_CTL (0x0818, Banked)	OFLOW_FRZ	**RES0**	Overflow is not possible for a CSU monitor
MSMON_CFG_CSU_CTL (0x0818, Banked)	SUBTYPE	**RES0**	This field is reserved for future use
MSMON_CSU (0x0840, Banked)	All fields	All fields valid	Cache storage usage value
MSMON_CSU_CAPTURE (0x0848, Banked)	All fields	All fields valid	Capture cache storage usage
MSMON_CAPT_EVNT (0x0808, Banked)	All fields	All fields valid	Capture event

2.3.2 TBU MPAM

MMU L1 enables you to implement TBU Memory System Resource Partitioning and Monitoring (MPAM).

However, the following constraints apply:

- When `TBUCFG_MTLB_DEPTH` == 0, the MTLB is not present
- When `TBUCFG_DIRECT_IDX` == 1, Direct Indexing is present but does not have MPAM controls
- When `TBUCFG_MTLB_PARTS` > 1, the MTLB 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 base offsets for the register frames are as follows:

Non-secure register frame (MPAMF_BASE_ns)

`0x3000`

Secure register frame (MPAMF_BASE_s)

`0xB000`

Note

All registers in the following table are marked whether they exist in the Secure (S) or the Non-secure (NS) register frames. Banked registers can be selected by another register to set the values for the selected resource.

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

Table 2-8: TBU MPAM ID registers implemented

Register	Field	Value	Description
MPAMF_IDR_LO (0x0000, Shared)	PARTID_MAX	1, 63, 511	Set to $2^{\texttt{TBUCFG\_PARTID\_WIDTH}} - 1$
MPAMF_IDR_LO (0x0000, Shared)	PMG_MAX	1	2 Non-secure Performance Monitoring groups supported for each PARTID
MPAMF_IDR_LO (0x0000, Shared)	HAS_CCAP_PART	1	Supports cache maximum capacity partitioning
MPAMF_IDR_LO (0x0000, Shared)	HAS_CPOR_PART	0	Cache portion partitioning not supported
MPAMF_IDR_LO (0x0000, Shared)	HAS_MBW_PART	0	Memory bandwidth partitioning not supported
MPAMF_IDR_LO (0x0000, Shared)	HAS_PRI_PART	0	Priority partitioning not supported
MPAMF_IDR_LO (0x0000, Shared)	EXT	1	EXTended MPAMF_IDR is present
MPAMF_IDR_LO (0x0000, Shared)	HAS_IMPL_IDR	0	Does not have **IMPLEMENTATION DEFINED** partitioning features
MPAMF_IDR_LO (0x0000, Shared)	HAS_MSMON	1	Supports performance monitoring by matching a combination of PARTID and Performance Monitoring Group (PMG)
MPAMF_IDR_LO (0x0000, Shared)	HAS_PARTID_NRW	0	Does not have MPAMF_PARTID_NRW_IDR, MPAMCFG_INTPARTID, or intPARTID mapping support.
MPAMF_IDR_HI (0x0004, Shared)	HAS_RIS	0	Does not have Resource Instance Selector
MPAMF_IDR_HI (0x0004, Shared)	HAS_EXTD_ESR	0	MPAMF_ESR is 64-bits. Not relevant because HAS_ESR is 0.
MPAMF_IDR_HI (0x0004, Shared)	HAS_ESR	0	MPAMF_ESR and MPAMF_ECR are not implemented
MPAMF_IDR_HI (0x0004, Shared)	HAS_ERR_MSI	0	No MSI support is implemented
MPAMF_IDR_HI (0x0004, Shared)	HAS_ENDIS	0	PARTID enable and disable not supported
MPAMF_IDR_HI (0x0004, Shared)	HAS	0	MPAMCFG_DIS.NFU is not implemented
MPAMF_SIDR (0x0008, S-Only)	S_PARTID_MAX	1, 63, 511	Set to $2^{\texttt{TBUCFG\_PARTID\_WIDTH}} - 1$
MPAMF_SIDR (0x0008, S-Only)	S_PMG_MAX	1	2 Secure Performance Monitoring groups supported per PART ID.
MPAMF_IIDR (0x0018, Shared)	All fields	All fields valid	Implementation ID Register

Register	Field	Value	Description
MPAMF_AIDR (0x0020, Shared)	ArchMajorRev	0x1	MPAM architecture v1.1
MPAMF_AIDR (0x0020, Shared)	ArchMinorRev	0x1	MPAM architecture v1.1
MPAMF_CCAP_IDR (0x0038, Shared)	CMAX_WD	8	8 fractions are supported
MPAMF_CCAP_IDR (0x0038, Shared)	CASSOC_WD	0	MPAMCFG_CASSOC is not implemented
MPAMF_CCAP_IDR (0x0038, Shared)	HAS_CASSOC	0	MPAMCFG_CASSOC is not implemented
MPAMF_CCAP_IDR (0x0038, Shared)	HAS_CMIN	0	MPAMCFG_CMIN is not implemented
MPAMF_CCAP_IDR (0x0038, Shared)	NO_CMAX	0	MPAMCFG_CMAX is implemented
MPAMF_CCAP_IDR (0x0038, Shared)	HAS_CMAX_SOFTLIM	0	MPAMCFG_CMAX has no SOFTLIM field
MPAMF_MSMON_IDR (0x0080, Shared)	MSMON_CSU	1	Performance monitor is supported for Cache Storage Usage by PARTID and PMG
MPAMF_MSMON_IDR (0x0080, Shared)	MSMON_MBWU	0	No performance monitor for Memory Bandwidth Usage by PARTID and PMG
MPAMF_MSMON_IDR (0x0080, Shared)	HAS_OFLOW	0	No overflow status register is implemented
MPAMF_MSMON_IDR (0x0080, Shared)	HAS_OFLOW	0	No MSI for overflows is implemented
MPAMF_MSMON_IDR (0x0080, Shared)	NO_HW_OFLOW	1	MPAM monitor overflow interrupt is not supported
MPAMF_MSMON_IDR (0x0080, Shared)	HAS_LOCAL_CAPT_EVNT	1	Has the local capture event generator and the MSMON_CAPT_EVNT register
MPAMF_CSUMON_IDR (0x0088 Shared)	NUM_MON	4	4 monitoring counters are implemented
MPAMF_CSUMON_IDR (0x0088 Shared)	HAS_OFLOW_CAPT	0	No support for MSMON_CFG_CSU_CTL.OFLOW_CAPT
MPAMF_CSUMON_IDR (0x0088 Shared)	HAS_CEVNT_OFLW	0	No support for MSMON_CFG_CSU_CTL.CEVNT_OFLW
MPAMF_CSUMON_IDR (0x0088 Shared)	HAS_OFSR	0	No overflow status bitmap is implemented
MPAMF_CSUMON_IDR (0x0088 Shared)	HAS_OFLOW_LNKG	0	No support for CSU overflow linkage
MPAMF_CSUMON_IDR (0x0088 Shared)	HAS_XCL	0	MSMON_CFG_CSU_FLT does not implement the XCL field
MPAMF_CSUMON_IDR (0x0088 Shared)	CSU_RO	1	MSMON_CSU is read-only
MPAMF_CSUMON_IDR (0x0088 Shared)	HAS_CAPTURE	1	Has an MSMON_CSU_CAPTURE register for every MSMON_CSU and supports the capture event behavior

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

Table 2-9: TBU MPAM configuration registers implemented

Register	Field	Value when resource is present	Description
MPAMCFG_PART_SEL (0x0100, Shared)	PARTID_SEL	Bits[(TBUCFG_PARTID_WIDTH – 1):0] valid	Can select up to 512 partitions to configure, based on TBUCFG_PARTID_WIDTH
MPAMCFG_PART_SEL (0x0100, Shared)	RIS	Not used	Resource Instance Selector
MPAMCFG_CMAX (0x0108, Banked)	CMAX	Bits [15:8] valid	Can choose up to 256 fractions

The following table shows the TBU MSMON configuration registers that are implemented.

Table 2-10: TBU MSMON configuration registers implemented

Register	Field	Value when resource is present	Description
MSMON_CFG_MON_SEL (0x0800, Banked)	MON_SEL	Bits [1:0] valid	Selects the monitor to configure. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
MSMON_CFG_MON_SEL (0x0800, Banked)	RIS	Not used	Resource Instance Selector When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
MSMON_CAPT_EVNT (0x0808, Banked)	All fields Bit [10]: Secure and Non-secure Bit [1]: Secure	All fields valid	Capture event. 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	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.
MSMON_CFG_CSU_FLT (0x0810, Banked)	PMG	0 or 1, only bit [16] used	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	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.

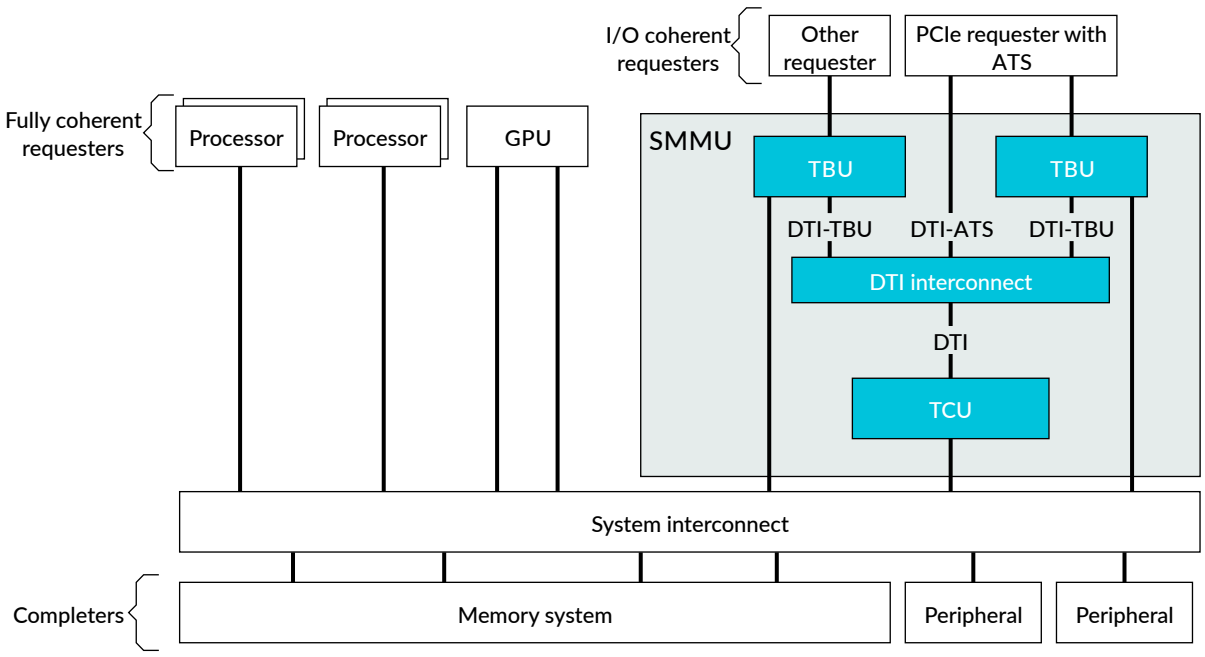
Register	Field	Value when resource is present	Description
MSMON_CFG_CSU_CTL (0x0818, Banked)	CAPT_EVNT	0b111	Capture occurs when a MSMON_CAPT_EVNT register is written. All other values are not supported. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
MSMON_CFG_CSU_CTL (0x0818, Banked)	CAPT_RESET	**RES0**	There is no reason to ever reset a Cache Storage Usage (CSU) monitor. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
MSMON_CFG_CSU_CTL (0x0818, Banked)	OFLOW_STATUS	**RES0**	Overflow is not possible for a CSU monitor. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
MSMON_CFG_CSU_CTL (0x0818, Banked)	OFLOW_INTR	**RES0**	This MPAM implementation does not support OFLOW_INTR. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
MSMON_CFG_CSU_CTL (0x0818, Banked)	OFLOW_FRZ	**RES0**	Overflow is not possible for a CSU monitor. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
MSMON_CFG_CSU_CTL (0x0818, Banked)	SUBTYPE	**RES0**	This field is reserved for future use. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.
MSMON_CFG_CSU_CTL (0x0818, Banked)	MATCH_PMG	Valid	You can control using the PMG for the monitor capture
MSMON_CFG_CSU_CTL (0x0818, Banked)	MATCH_PARTID	Valid	You can control using the PARTID for the monitor capture
MSMON_CFG_CSU_CTL (0x0818, Banked)	TYPE	0x43	Fixed value for the CSU monitor
MSMON_CSU (0x0840, Banked)	All fields	All fields valid	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	Capture cache storage usage. When direct indexing or direct partitioning is enabled, this field does not reflect any meaningful value.

3. Functional description

The major functional blocks of MMU L1 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 MMU L1.

Figure 3-1: MMU L1 example system



MMU L1 contains the following key components:

Translation Buffer Unit (TBU)

The TBU contains Translation Lookaside Buffers (TLBs) that cache address translations. MMU L1 implements a TBU that can be connected to a single requester or multiple requesters through an interconnect. It is also possible to connect multiple TBUs to a single requester to improve performance. These TBUs are local to the corresponding requester.

Translation Control Unit (TCU)

The TCU controls and manages the address translations. In MMU L1-based systems, the *AMBA® DTI Protocol Specification* defines the standard for communicating with the TCU.

DTI interconnect

The DTI interconnect connects multiple TBUs, and PCIe Root Ports, to the TCU.

When an MMU L1 TBU receives a transaction on the TBS 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 Queue and Table Walk (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 performed. Sometimes no walks are required. When the TBU receives a translation from the TCU, it stores the translation in its TLBs unless the DO_NOT_CACHE field in DTI_TBU_TRANS_FAULT is set to 1. For more information, see the *AMBA® DTI Protocol Specification*.

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

For more information about the following, see the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2*:

- Translation
- How software communicates with the TCU

3.1 MMU L1 Interfaces

MMU L1 includes interfaces for each of the Transaction Control Unit (TCU), Transaction Buffer Unit (TBU), and Distributed Translation Interface (DTI) interconnect components.

TCU

The MMU L1 TCU includes several requester and completer interfaces. For example, TCU Queue and Table Walk and Distributed Virtual Memory interface, TCU PROG interface, TCU DTI interface, TCU SYSCO signaling, TCU tie-off signals, and TCU ELA observation interface.

TBU

Each MMU L1 TBU includes several requester and completer interfaces. For example, TBU TBS interface, TBU TBM interface, TBU low power interfaces, TBU DTI interface, TBU interrupt interface, TBU tie-off signals, and TBU ELA observation interface.

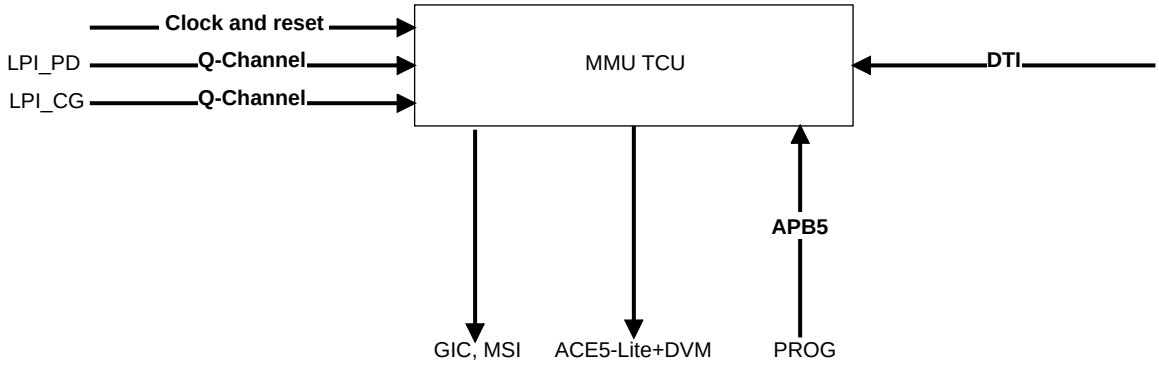
DTI

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

3.1.1 TCU interfaces

The MMU L1 TCU includes several requester and completer interfaces.

The following figure shows the key TCU interfaces.



3.1.1.1 TCU Queue and Table Walk and Distributed Virtual Memory interface

The Queue and Table Walk and Distributed Virtual Memory (QTW and DVM) interface is an AXI5 +DVM requester interface.

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

- ReadNoSnoop
- WriteNoSnoop
- ReadOnce, if coherency is supported by tying the sup_cohacc signal HIGH
- WriteUnique, if coherency is supported by tying the sup_cohacc signal HIGH
- DVM Complete
- AtomicCompare transactions, if HTTU is supported by tying the sup_httu signal HIGH

The QTW and 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 Configuration parameters and methodology.

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] = `0b00`. If `TCU_ID_WIDTH` > 4, bits {`TCU_ID_WIDTH` − 1 :4} are 0.	`0b00`

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

The following table shows the possible values of awid_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] = 0b01. If TCU_ID_WIDTH > 4, bits {TCU_ID_WIDTH − 1:4} are 0.	0b00
Event Queue write	Bits[3:2] = 0b10. If TCU_ID_WIDTH > 4, bits {TCU_ID_WIDTH − 1:4} are 0.	0b00
MSI write	Bits[3:2] = 0b11. If TCU_ID_WIDTH > 4, bits {TCU_ID_WIDTH − 1:4} are 0.	0b00
HTTU Write	Indicates the page table walk slot requesting the Hardware Translation Table Update (HTTU) write	0b11

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® APB5 completer interface. This interface enables software to program the MMU L1 internal registers and read the Performance Monitoring Unit (PMU) registers and the Debug registers. The 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, or **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 *AMBA® APB Protocol Specification*.

3.1.1.3 TCU low power interfaces

These Q-Channel completer interfaces manage Low Power Interface (LPI) powerdown and clock gating for the TCU.

TCU LPI_PD interface

This Q-Channel completer interface manages LPI powerdown for the TCU. For LPI powerdown signals see TCU LPI_PD interface signals.

TCU LPI_CG interface

This Q-Channel completer interface enables LPI clock gating for the TCU. For LPI clock gating signals see TCU LPI_CG interface signals.

For detailed specification information on LPIs, see the *AMBA® Low Power Interface Specification*.

3.1.1.4 TCU DTI interface

The Distributed Translation Interface (DTI) manages communication between the TBUs and the TCU, using the DTI protocol. MMU L1 implements AXI5-Stream interfaces for conveying DTI messages.

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

Requester interface

One AXI5-Stream requester interface.

Completer interface

One AXI5-Stream completer interface.

For the AXI5-Stream interfaces, the Wakeup_Signal property is enabled and the Check_Type property is not enabled.

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

See Distributed Translation Interface overview for an overview of the DTI and TCU DTI interface signals for information on the DTI interface signals.

3.1.1.5 TCU SYSCO signaling

MMU L1 provides a hardware system coherency interface. This requester 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 signal is ignored. See TCU SYSCO interface signals.

3.1.1.6 TCU tie-off signals and operating parameters

You can use the TCU tie-off signals to initialize various operating parameters on exit from reset state.

For more information, see TCU tie-off signals.

3.1.1.7 TCU ELA observation interface

This Embedded Logic Analyzer (ELA) observation requester 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 Translation Control Unit debug configuration parameters.

For more information about the interface signals, see TCU observation interfaces.

3.1.2 TBU interfaces

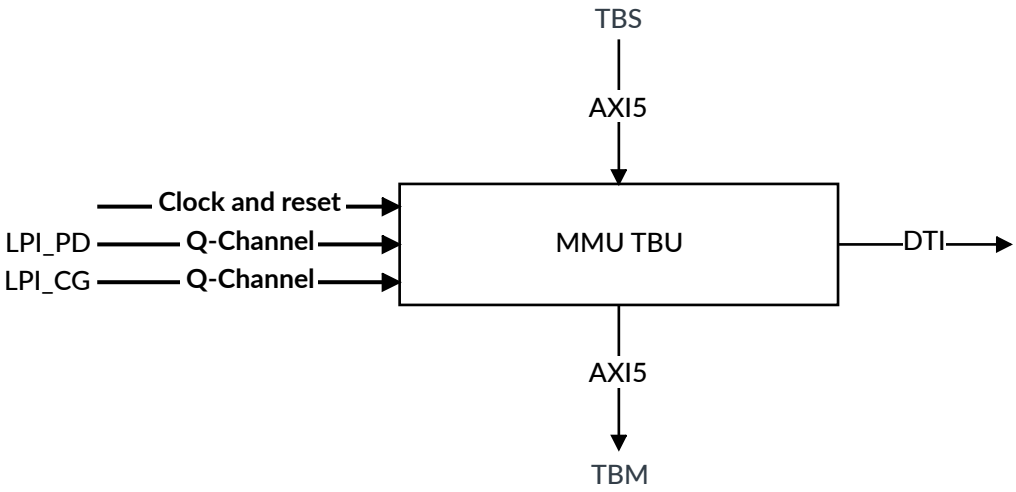
Each MMU L1 TBU includes several requester and completer interfaces.


Note

In the MMU-700 System Memory Management Unit, TBUs used ACE-Lite interfaces. In issue J of the *AMBA® AXI Protocol Specification*, ACE-Lite functionality has been incorporated into AXI functionality. Therefore, new designs use an AXI5 interface. For more information, see Appendix B1 of the *AMBA® AXI Protocol Specification*.

The following figure shows the TBU interfaces.

Figure 3-2: TBU interfaces



3.1.2.1 TBU TBS interface

The transaction completer interface, TBS, is an AXI5 interface on which the TBU receives incoming memory accesses. This interface supports a 64-bit address width and the `TBUCFG_DATA_WIDTH` parameter defines the data width.

The interface implements optional signals to support several AXI5 extensions. For more information, see AMBA implementation constraints.

The TBU receives incoming untranslated, partially translated, or fully translated memory accesses on the AXI5 completer interface. See TBU transaction completer interface, completer interface signals.

For information on MMU L1 error response behavior, see Error responses.

3.1.2.2 TBU TBM interface

The transaction requester interface, TBM, is an AXI5 interface on which the 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.

The TBM interface supports a 52-bit address width, and the `TBUCFG_DATA_WIDTH` parameter defines the data width. For more information, see Translation Buffer Unit I/O configuration parameters.

This interface can issue read and write transactions until the outstanding transaction limit is reached. MMU L1 provides parameters that permit you to configure:

- The outstanding read transactions limit

- The outstanding write transactions limit

The interface implements optional signals to support several AXI5 extensions. For more information, see AMBA implementation constraints.

When receiving an SLVERR or DECERR response to a downstream transaction, the TBM interface propagates the same response to the TBS interface. For more information on errors, see Error responses.

3.1.2.3 TBU low power interfaces

These Q-Channel completer interfaces manage Low Power Interface (LPI) powerdown and clock gating for the TBU.

TBU LPI_PD interface

This Q-Channel completer interface manages LPI powerdown for the TBU. For LPI powerdown signals see TBU LPI_PD interface signals.

TBU LPI_CG interface

This Q-Channel completer interface enables LPI clock gating for the TBU. For LPI clock gating signals see TBU LPI_CG interface signals.

For more information on LPIs, see the *AMBA® Low Power Interface Specification*.

3.1.2.4 TBU DTI interface

The TBU DTI interface enables the TBU to request translations from the TCU. This interface uses the DTI-TBUv2 protocol for communication between the TBU and the TCU and between PCIe Root Ports and the TCU. The TCU includes a completer (TCU) DTI interface and each TBU includes a requester (TBU) DTI interface.

To permit bidirectional communication, each DTI interface includes:

Requester interface

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

Completer interface

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

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

3.1.2.5 TBU interrupt interface

The interrupt interface provides global, per-context, and performance interrupts. These interrupts are wired interrupts rather than Message Signaled Interrupts (MSIs).

3.1.2.6 TBU tie-off signals

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

3.1.2.7 TBU ELA observation interface

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

When `TBUCFG_USE_ELA_DEBUG` is 0, these signals are tied to 0. See Translation Buffer Unit debug configuration parameters.

For more information about the interface signals, see:

- TBU observation interfaces

3.1.3 DTI interconnect interfaces

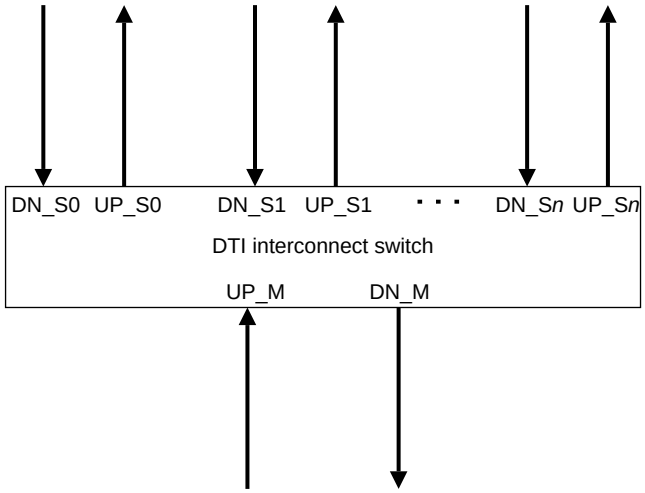
The DTI interconnect includes interfaces for each of the switch, sizer, and register slice components.

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 3-3: DTI interconnect switch interfaces



The following table provides more information about the switch interfaces.

Table 3-3: DTI interconnect switch interfaces

Interface	Data flow	Facing	Description
DN_S	Downstream	TBU side	TCU downstream interface. One DN_Sn interface is present for each TBU interface.
UP_S	Upstream	TBU side	TCU upstream interface. One UP_Sn interface is present for each TBU interface.
DN_M	Downstream	TCU side	TBU downstream interface
UP_M	Upstream	TCU side	TBU upstream interface

Note

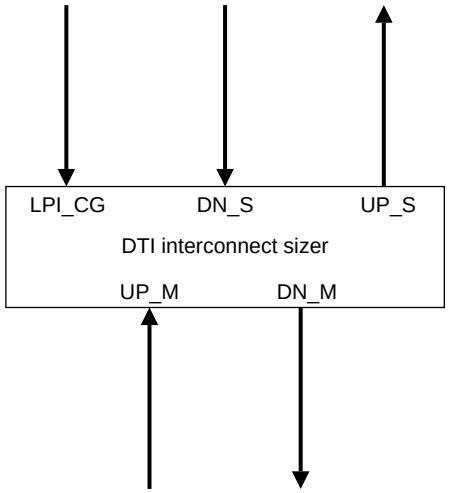
The interconnect switch does not store any data, and therefore does not require a Q-Channel clock-gating interface.

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 3-4: DTI interconnect sizer interfaces



The following table provides more information about the sizer interfaces.

Table 3-4: DTI interconnect sizer interfaces

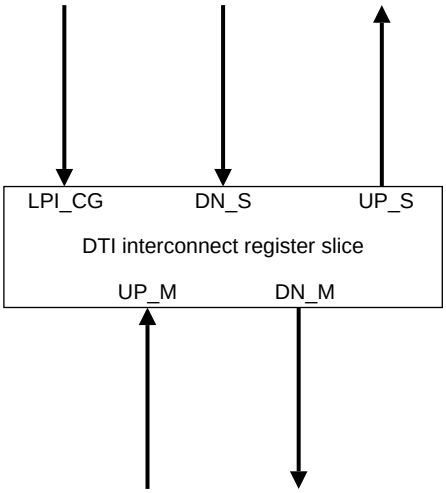
Interface	Data flow	Facing	Description
LPI_CG	-	Facing the clock controller	Clock gating interface
DN_S	Downstream	TBU side	TCU downstream interface
UP_S	Upstream	TBU side	TCU upstream interface
DN_M	Downstream	TCU side	TBU downstream interface
UP_M	Upstream	TCU side	TBU upstream interface

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 3-5: DTI interconnect register slice interfaces



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

Table 3-5: Register slice interfaces and descriptions

Interface	Data flow	Facing	Description
LPI_CG	-	Facing the clock controller	Clock gating interface
DN_S	Downstream	TBU side	TCU downstream interface
UP_S	Upstream	TBU side	TCU upstream interface
DN_M	Downstream	TCU side	TBU downstream interface
UP_M	Upstream	TCU side	TBU upstream interface

4. Operation of MMU L1

The MMU L1 components operate together to provide the functionality of the MMU L1 System Memory Management Unit.

4.1 Distributed Translation Interface overview

In an MMU L1-based system, the AMBA® Distributed Translation Interface (DTI) protocol defines the standard for communicating with a TCU.

The *AMBA® DTI Protocol Specification* includes both:

DTI-TBU protocol
: For communication between a TBU and a TCU

DTI-ATS protocol
: For communication between a PCIe Root Port and a TCU

The DTI protocol is a point-to-point protocol. Each channel consists of a link, a DTI requester, and a DTI completer. The DTI requesters in the respective protocols are:

TBU
: In the DTI-TBU protocol

PCIe Root Port
: In the DTI-ATS protocol

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

DTI requesters and completers 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 requester uses a DTI_TBU_CONDIS_REQ or a DTI_ATS_CONDIS_REQ message to initiate a connection handshake. If the requester provides a TID value that is greater than the maximum supported TID that `TCUCFG_NUM_TBU` defines, the completer sends a Connect Deny message. The TBU uses the TOK_INV_GNT field to grant invalidation tokens.

The TBU or Root Port grant 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.

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

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

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

4.2 Performance Monitoring Unit

MMU L1 includes a Performance Monitoring Unit (PMU) for the TCU and a PMU for each TBU. The PMU events and counters indicate the runtime performance of MMU L1.

The PMU includes logic to gather various statistics on the operation of MMU L1 during runtime, using events and counters. The SMMUv3.2 architecture defines some events and the microarchitectural defines other events that are specific to the MMU L1 implementation.

4.2.1 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 or within a particular StreamID range 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 L1 TCU performance events.

Table 4-1: 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 the clock Q-Channel enters the Q_STOPPED state are not counted.
Transaction	0x1	Yes	Counts the translation requests that a DTI-TCU or DTI-ATS requester receive
TLB miss caused by incoming transaction or translation request	0x2	Yes	Counts translation requests where the translation walks new translation table entries

Event	Event ID	SMMU_PMCG_SMR0 filterable	Description
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, but not page requests, that originate from a DTI-ATS requester

The following table shows the architecturally defined MMU L1 TBU performance events.

Table 4-2: 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 the clock Q-Channel enters the Q_STOPPED state are not counted.
Transaction	0x1	Yes	Counts transactions that are received on the TBS 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*.

4.2.2 MMU L1 TCU events

You can configure the MMU L1 Performance Monitoring Unit (PMU) 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, or within a particular StreamID range, 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 MMU L1 TCU performance events.

Table 4-3: MMU L1 TCU performance events

Event	Event ID	SMMU_PMCG_SMR0 filterable	Description
S1L0WC lookup	0x80	Yes	Counts translation requests that access S1L0WC entries in the walk cache.
S1L0WC miss	0x81	Yes	Counts translation requests that access S1L0WC entries in the walk cache and do not result in a hit.

Event	Event ID	SMMU_PMCG_SMR0 filterable	Description
S1L1WC lookup	0x82	Yes	Counts translation requests that access S1L1WC entries in the walk cache.
S1L1WC miss	0x83	Yes	Counts translation requests that access S1L1WC entries in the walk cache and do not result in a hit.
S1L2WC lookup	0x84	Yes	Counts translation requests that access S1L2WC entries in the walk cache.
S1L2WC miss	0x85	Yes	Counts translation requests that access S1L2WC entries in the walk cache and do not result in a hit.
S1L3WC lookup	0x86	Yes	Counts translation requests that access S1L3WC entries in the walk cache.
S1L3WC miss	0x87	Yes	Counts translation requests that access S1L3WC entries in the walk cache and do not result in a hit.
S2L0WC lookup	0x88	Yes	Counts translation requests that access S2L0WC entries in the walk cache.
S2L0WC miss	0x89	Yes	Counts translation requests that access S2L0WC entries in the walk cache and do not result in a hit.
S2L1WC lookup	0x8A	Yes	Counts translation requests that access S2L1WC entries in the walk cache.
S2L1WC miss	0x8B	Yes	Counts translation requests that access S2L1WC entries in the walk cache and do not result in a hit.
S2L2WC lookup	0x8C	Yes	Counts translation requests that access S2L2WC entries in the walk cache.
S2L2WC miss	0x8D	Yes	Counts translation requests that access S2L2WC entries in the walk cache and do not result in a hit.
S2L3WC lookup	0x8E	Yes	Counts translation requests that access S2L3WC entries in the walk cache.
S2L3WC miss	0x8F	Yes	Counts translation requests that access S2L3WC entries in the walk cache and do not result in a hit.
WC read	0x90	Yes	Counts reads from the entries in the walk cache RAMs, excluding reads that invalidation requests cause. 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. 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.

A single DTI translation request might correspond to multiple translation request events where a translation results in a stall fault event and is restarted. For example, if the Event queue is full

a translation results in a stall fault event and the translation is retried when an Event queue slot becomes available.

4.2.3 MMU L1 TBU events

You can configure the MMU L1 PMU 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, or within a particular StreamID range, are counted.

The 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 4-4: MMU L1 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. 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 translation 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.
Translation request	0x88	Yes	Counts translation requests, including both speculative and non-speculative requests.
Write data uses write data buffer	0x89	Yes	Counts transactions with write data that is stored in the write data buffer.
Write data bypasses write data buffer	0x8A	Yes	Counts transactions with write data that bypass the write data buffer.

Event	Event ID	SMMU_PMCG_SMR0 filterable	Description
MakeInvalid downgrade	0x8B	Yes	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.
Stash fail	0x8C	Yes	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. • A StashOnceShared or StashOnceUnique transaction, that is terminated because of a StreamDisable or GlobalDisable translation response, does not cause this event to count. For more information, see Transactions that do not cause a translation fault.
Fixed Burst Termination	0x8D	Yes	Fixed Burst terminated in TBU. Counts when the TBU issues an abort response for a fixed burst because its domain after translation is determined to be shareable.
InvalidateHint transaction failed	0x8E	Yes	Counts when the TBU aborts an InvalidateHint transaction because of an attribute or permission mismatch with the transaction type.

4.3 Main TLB direct indexing and main TLB direct partitioning

Main Translation Lookaside Buffer (TLB) direct indexing can help your system to meet real-time translation requirements by enabling management of Main TLB (MTLB) entries from outside the TBU.


Note

If you use the Main TLB direct indexing or Main TLB direct partitioning features, Memory System Resource Partitioning and Monitoring (MPAM) is not valid. The MPAM bits still progress to the TBM (Requester) for use in other system components, for example the interconnect. However, MMU L1 ignores any values programmed to the MPAM registers.

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

- You can guarantee that prefetched entries are not overwritten.
- The MTLB can be partitioned into different sets of entries that different streams use.

When direct indexing is enabled for a TBU:

- Lookups and updates to the MTLB use the mtlbidx field as the index.
- Updates to the MTLB use the way that mtlbway specifies.
- Lookups to the MTLB operate on all ways simultaneously.
- All invalidates walk the entire Main TLB, including invalidates by VA and IPA.

- For lookups different page sizes are stored separately and the hitmap optimizes the lookups. Therefore, each lookup is performed once, rather than trying different indexes for each of the in-use translation sizes in turn. The MTLB hitmap bits are therefore ignored.

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 must be provided externally, and the other bits are generated internally.

If an MTLB is partitioned:

- The MTLB size is divided into a number of equal segments defined by `TBUCFG_MTLB_PARTS`
- The mtlbpart field defines the $\log_2$(`TBUCFG_MTLB_PARTS`) most significant index bits

To maintain system performance, we recommend that you disable DVM invalidation on TBUs which have direct indexing enabled. Disable DVM invalidation by setting the appropriate TCU_NODE_CTRLn.DIS_DVM bit. For more information see TBU TBS User signals.

4.4 RAS implementation

Reliability, Availability, and Serviceability (RAS) features enable SRAM corruption to be detected and corrected, optionally generating interrupts into the system. RAS features are always implemented for the TCU.

In MMU L1, the following RAS error detection, or RAS error detection and correction implementation scenarios apply:

- All TCU RAMs support RAS implementation
- TBU MTLB RAMs support RAS implementation if `TBUCFG_TLB_RAS_SUPPORT` = 1
- TBU Write Data Buffer RAM does not support any kind of error detection or correction

MMU L1 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 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.

The following table shows which errors and interrupts are triggered in each error scenario. In the table, the following abbreviations apply:

CE
: Corrected Error

CRI
: Critical Error Interrupt

DE
Deferred Error

DED
Double Error Detect

ERI
Error Recovery Interrupt

FHI
Fault Handling Interrupt

SEC
Single Error Correct

SECDED
Single Error Correct Double Error Detect

UC
Uncontainable error

UE
Uncorrected Error

Table 4-5: RAM RAS error sources

RAM name	Error correction and detection mechanism	RAS error triggered	RAS interrupts triggered
TLB MTLB PLIM	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TLB MTLB PCNT	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TLB MTLB REPL	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TLB MTLB TAGS	SED	CE	FHI
	DED	CE	FHI
TLB 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	UE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU HTTU RAM	SEC	UE	FHI

RAM name	Error correction and detection mechanism	RAS error triggered	RAS interrupts triggered
	DED	UE (UC)	FHI, ERI, and CRI
TMU WCB MWC PLIM	SEC	UE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU WCB MWC PCNT	SEC	UE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU WCB MWC REPL	SEC	UE	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

4.5 Quality of Service

For TCU you can assign a priority level to every DTI node, whether it is a TBU or a PCIe Root Port. The priority level is applied to every translation from that Distributed Translation Interface (DTI) node.

The TCU uses this priority level to:

- Arbitrate between translations that are waiting in the translation request buffer when translation requester 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. For the TBU, the AxQOS value received on the TBS interface is forwarded to the TBM interface.

4.6 Distributed Virtual Memory messages

The QTW/DVM interface supports Distributed Virtual Memory (DVM) messages. MMU L1 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.
- 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 AXI 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

4.7 TCU transaction handling

The transaction width, burst length, and transfer size that the MMU L1 TCU supports depend on the transaction type.

The following table shows the TCU support for read transactions.

Table 4-6: 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	Configuration slot number	0b01
Stream table or Context Descriptor table lookup	512-bit	Configuration slot number	0b01
Translation table lookup	64-bit	PTW slot number	0b10
Command queue read	128-bit	All 0	0b00
DVM Complete	-	Bit[2] is 1 and all other bits are 0	0b00

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 4-7: TCU support for write transactions

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

There can be a maximum of 5 outstanding writes at one time, which includes:

- 1 event queue write
- 1 PRI queue write
- 1 MSI write
- 2 HTTUs

All read and write transactions are aligned to the transaction width, except for HTTU writes, which are aligned to half of the transaction size. They are aligned to half of the transaction width because HTTU writes contain two 8-byte data values, but access only one 8-byte memory location. The transaction width includes both data values, but the transaction is aligned to the width of the

memory location, which is half of the transaction size. Other transactions that the TCU issues convey data that is equal to the width of the memory location being accessed.

4.8 TCU prefetch

The MMU L1 TCU can prefetch translations on a per-context basis to improve translation performance for real-time requesters 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 L1 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 base address of the previous page, which is the current address, minus the page size 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:

- 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.
- 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 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] = `0b11`, 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] = `0b10`, because a TBU receives an AtomicLoad, AtomicSwap, or AtomicCompare transaction.

If the original translation request returns one of 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.

4.9 Error responses

The AXI Protocol Specification defines several error response types.

The error response types include:

- External AXI completer error, SLVERR, and external AXI decode error
- DECERR
- TRANSFAULT

MMU L1 error response behavior depends on the interface. For example, the TCU interface treats SLVERR and DECERR identically, as an abort. It is not possible to receive a TRANSFAULT response or a PREFETCHED response type on the TCU interface.

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. Only the TBS interface generates TRANSFAULT responses, the TBM interface does not receive TRANSFAULT responses. Therefore the TBU can generate TRANSFAULT responses.

4.10 Conversion between AXI and Armv8 attributes

The SMMUv3 architecture defines attributes in terms of the Armv8 architecture.

See the *Arm® Architecture Reference Manual for A-profile architecture*. The MMU L1 components are therefore required to perform conversion between AXI and Armv8 attributes.

The TBU must convert:

- AXI attributes to Armv8 attributes when it receives transactions on the Transaction Completer (TBS) interface
- Armv8 attributes to AXI attributes when it sends transactions on the Transaction Requester (TBM) interface

The TCU must convert Armv8 attributes to AXI attributes when it outputs transactions on the QTW/DVM interface.

4.10.1 Completer interface memory type attribute handling

The AxCACHE and AxDOMAIN signals contain the memory attributes that apply to the MMU L1 TBU TBS interface.

Cache Maintenance Operations (CMOs) for example, CleanShared, CleanInvalid, CleanSharedPersist, MakeInvalid, and InvalidateHint do not have a memory type. Therefore the SMMU ignores the input memory type for these CMOs. For more information on Memory types and Shareability for Cache Maintenance Operations, see the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2*.

MMU L1 always treats some special transaction types as Normal Write-Back, see Transaction types. For more information on the AXI to Armv8 attribute conversions that the TBU TBS interface performs, see Completer interface attribute handling.

4.10.2 Requester interface memory type attribute handling

The AxCACHE and AxDOMAIN signals contain the MMU L1 memory attributes that apply to the TBM and the QTW/DVM interfaces.

For more information on the Armv8 to AXI attribute conversions that the TBU TBM interface performs, see Requester interface attribute handling.

4.11 AXI USER bits that MMU L1 TBU TBM and TCU QTW/DVM define

The TBU TBM interface AxUSER signals, aruser_m and awuser_m, contain 1 more bit than the `TBUCFG_AxUSER_WIDTH` parameters define. This extra bit is output in the higher-order bits of the aruser_m and awuser_m signals. The TCU QTW/DVM interface AxUSER signals are 4 bits wide.

The following table shows the MMU L1-defined TBU aruser_m and awuser_m bits, where w represents the AXI USER bus width that `TBUCFG_AxUSER_WIDTH` defines. See Translation Buffer Unit I/O configuration parameters.

Table 4-8: MMU L1-defined TBU aruser_m and awuser_m bits

Bit position	Value	Description
AxUSER[w]	Outer Cacheable	See Requester interface attribute handling
AxUSER[w−1:0]	Regular AxUSER signals	-

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 L1-defined TCU aruser_qtw and awuser_qtw bits.

Table 4-9: MMU L1-defined TCU aruser_qtw and awuser_qtw bits

Bit position	Value	Description
AxUSER[3:0]	**IMPLEMENTATION DEFINED** Page Based Hardware Attribute (PBHA) bits.	For more information, see Page Based Hardware Attribute in MMU L1 and Calculate Page Based Hardware Attribute in SMMUs in the *Arm® MMU L1 System Memory Management Unit Configuration and Integration Manual*.

4.11.1 TBU TBS User signals

The User signals convey extra information when direct indexing is enabled for the TBU by setting `TBUCFG_DIRECT_IDX` = 1 or when an MTLB is partitioned by setting `TBUCFG_MTLB_PARTS` ≥ 1.

When direct indexing is enabled (`TBUCFG_DIRECT_IDX` = 1), or when an MTLB is partitioned (`TBUCFG_MTLB_PARTS` > 1), 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.

The wuser_s signals are forwarded to the wuser_m signals, and the ruser_m and buser_m signals are forwarded to the ruser_s and buser_s signals respectively. MMU L1 does not modify the WUSER, RUSER, and BUSER signals.

The following table shows the extended bits in the order MSB first for MTLB partitioning.

Table 4-10: Extended aruser_s and awuser_s bits for MTLB partitioning

Field name	Width	Description
mtlbpart	$\log_2$(`TBUCFG_MTLB_PARTS`)	MTLB partition
-	`TBUCFG_AWUSER_WIDTH` for awuser_s. `TBUCFG_ARUSER_WIDTH` for aruser_s.	Regular AxUSER signals

The following table shows the extended bits in the order MSB first for direct indexing.

Table 4-11: Extended aruser_s and awuser_s bits for direct indexing

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
-	`TBUCFG_AWUSER_WIDTH` for awuser_s. `TBUCFG_ARUSER_WIDTH` for aruser_s.	Regular AxUSER signals

4.12 Page Based Hardware Attribute in MMU L1

The Arm® architecture defines that 4 bits in both stage 1 and stage 2 leaf page table entry formats are reserved for software use. These bits are called Page Based Hardware Attribute (PBHA) bits.

The TBU TBM interface uses the AxPBHA signals, AWPBHA and ARPBHA, that were added in *AMBA® AXI Protocol Specification* for PBHA signaling.

Armv8.5 and SMMUv3.2 define a mechanism where software can declare that it does not require the PBHA bits, on a per bit basis.

See the following:

- Section D8.7.2 Page Based Hardware attributes in the *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 PBHA mechanism hands over control of those bits to **IMPLEMENTATION DEFINED** hardware purposes.

For more information about PBHA bits, see the *Arm® MMU L1 System Memory Management Unit Configuration and Integration Manual*.

4.13 AXI5 transaction types

MMU L1 supports a number of special transaction types, distinguished by a nonzero encoding of AxSNOOP.

Unless otherwise specified, transactions are propagated on TBM with the same transaction type that was presented on TBS. An ordinary read is one with ARSNOOP = `0b0000`. That is ReadNoSnoop, ReadOnce, WriteNoSnoop, or WriteUnique depending on AxDOMAIN and whether it is a read or a write. For atomics, AWSNOOP = `0b00000`.

MMU L1 supports AXI5 read and write transactions that use the read and write channels. For more information about AXI5 transactions, see AXI5 read transaction support and AXI5 write transaction support.

4.13.1 AXI5 read transaction support

Supported transactions (Supported = Yes) are handled by MMU L1 as expected based on SMMU and AMBA® architectures.

In cases when MMU L1 receives unsupported transactions (Supported = No), the SMMU behavior is **UNDEFINED**. Upstream requesters should never send these to MMU L1, as their respective property sets are either not supported by MMU L1 or not supported by AXI5.

The following table describes all transaction types encoded with ARSNOOP signals. The table uses these abbreviations:

NSH

Non-shareable (`0b00`)

SH

Shareable (`0b01` or `0b10`)

SYS

System (`0b11`)

Table 4-12: AXI5 read transaction support

ARSNOOP	AWDOMAIN	Opcode	Property	Supported
`0b0000`	NSH, SYS	ReadNoSnoop	-	Yes
`0b0000`	SH	ReadOnce	Shareable_Transactions	Yes
`0b0001`	SH	ReadShared Requester needs to transform this to ReadNoSnoop	Shareable_Transactions Shareable_Cache_Support	No
`0b0010`	SH	ReadClean Requester needs to transform this to ReadNoSnoop	Shareable_Transactions Shareable_Cache_Support	No
`0b0011`	-	Reserved	-	-
`0b0100`	SH	ReadOnceCleanInvalid	Shareable_Transactions DeAllocation_Transactions	Yes
`0b0101`	SH	ReadOnceMakeInvalid	Shareable_Transactions DeAllocation_Transactions	Yes
`0b0110`	-	Reserved	-	-
`0b0111`	-	Reserved	-	-
`0b1000`	NSH, SH	CleanShared	CMO_On_Read	Yes
`0b1001`	NSH, SH	CleanInvalid	CMO_On_Read	Yes
`0b1010`	NSH, SH	CleanSharedPersist	CMO_On_Read Persist_CMO	Yes
`0b1011`	-	Reserved	-	-
`0b1100`	-	Reserved	-	-
`0b1101`	NSH, SH, SYS	MakeInvalid	CMO_On_Read	Yes
`0b1110`	SH	DVMComplete	DVM_Message_Support	No
`0b1111`	-	Reserved	-	No

4.13.2 AXI5 write transaction support

Supported transactions (Supported = Yes) are handled by MMU L1 as expected based on SMMU and AMBA® architectures.

In cases when MMU L1 receives unsupported transactions (Supported = No), the SMMU behavior is **UNDEFINED**. Upstream requesters should never send these to MMU L1, as their respective property sets are either not supported by MMU L1 or not supported by AXI5.

The following table describes all transaction types encoded with AWSNOOP signals and uses these abbreviations:

NSH
: Non-shareable (`0b00`)

SH
: Shareable (`0b01` or `0b10`)

SYS
: System (`0b11`)

Table 4-13: AXI5 write transaction support

AWSNOOP	AWDOMAIN	Opcode	Property	Supported
`0b00000`	NSH, SYS	WriteNoSnoop	-	Yes
`0b00000`	SH	WriteUniquePtl	Shareable_Transactions	Yes
`0b00000`	NSH, SH, SYS	Atomic	Atomic_Transactions	Yes
`0b00001`	NSH	WriteNoSnoopFull Requesters need to transform this to WriteNoSnoop	Shareable_Cache_Support	No
`0b00001`	SH	WriteUniqueFull	Shareable_Transactions	Yes
`0b00010`	-	Reserved	-	No
`0b00011`	SH	WriteBackFull Requesters need to transform this to WriteNoSnoop	Shareable_Transactions Shareable_Cache_Support	No
`0b00100`	-	Reserved	-	No
`0b00101`	SH	WriteEvictFull Requesters need to drop this request, because it is not supported. See AXI5 feature support and the *AMBA® AXI Protocol Specification*.	Shareable_Transactions Shareable_Cache_Support	No
`0b00110`	NSH, SH	CMO	CMO_On_Write	No
`0b00111`	NSH, SH, SYS	WriteZero	WriteZero_Transaction	No
`0b01000`	SH	WriteUniquePtlStash	Shareable_Transactions Cache_Stash_Transactions	Yes
`0b01001`	SH	WriteUniqueFullStash	Shareable_Transactions Cache_Stash_Transactions	Yes

AWSNOOP	AWDOMAIN	Opcode	Property	Supported
0b01010	NSH, SH	WritePtlCMO	Write_Plus_CMO	No
0b01011	NSH, SH	WriteFullCMO	Write_Plus_CMO	No
0b01100	NSH, SH	StashOnceShared	Cache_Stash_Transactions	Yes
0b01101	NSH, SH	StashOnceUnique	Cache_Stash_Transactions	Yes
0b01110	NSH, SH, SYS	StashTranslation	Untranslated_Transactions Cache_Stash_Transactions	Yes
0b01111	NSH, SH	Prefetch	Prefetch_Transaction	No
0b10000	SYS	WriteDeferrable	WriteDeferrable_Transaction	No
0b10001	NSH, SH, SYS	UnstashTranslation	UnstashTranslation_Transaction	Yes
0b10010	NSH, SH	InvalidateHint	InvalidateHint_Transaction	Yes
0b10011 to 0b11111	-	Reserved	-	No

5. Configuration parameters and methodology

In MMU L1 you can parameterize the SystemVerilog delivered Translation Buffer Unit (TBU), Translation Control Unit (TCU), and Bidirectional AXI-Stream (BAS) components. Use the `generate` script to configure these components.

For more information on how to configure the components, see `generate` script in the *Arm® MMU L1 System Memory Management Unit Configuration and Integration Manual*.

There are several versions of the switch component, part of the BAS components that Arm delivers. These versions accommodate different numbers of completer interfaces.

You can configure different types of configuration parameters:

- Translation Control Unit I/O configuration parameters
- Translation Control Unit buffer configuration parameters
- Translation Control Unit debug configuration parameters
- Translation Buffer Unit configuration parameters
- Translation Buffer Unit buffer configuration parameters
- Translation Buffer Unit register slice configuration parameters
- Translation Buffer Unit I/O configuration parameters
- Translation Buffer Unit debug configuration parameters

5.1 Translation Control Unit I/O configuration parameters

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


Tip

For more detailed descriptions of the TCU I/O configuration parameters, see the *Arm® MMU L1 System Memory Management Unit Configuration and Integration Manual*.

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

Table 5-1: TCU I/O configuration parameters

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

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.

5.2 Translation Control Unit buffer configuration parameters

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

For more detailed descriptions of the TCU buffer configuration parameters, see the *Arm® MMU L1 System Memory Management Unit Configuration and Integration Manual*.

The following table contains the TCU buffer configuration parameters.

Table 5-2: 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 (TCUCFG_WC_DEPTH/TCUCFG_WC_BANKS)/TCUCFG_WC_WAYS) must be > 1
TCUCFG_WC_BANKS	1, 2, 4	Number of banks in walk cache (TCUCFG_WC_DEPTH/TCUCFG_WC_BANKS)/TCUCFG_WC_WAYS) must be > 1
TCUCFG_WC_WAYS	4, 8, 16	Number of ways in walk cache (TCUCFG_WC_DEPTH/TCUCFG_WC_BANKS)/TCUCFG_WC_WAYS) must be > 1
TCUCFG_NUM_TBU	14, 62	Maximum number of DTI requesters, that is, DTI-TBU and DTI-ATS requesters, that the TCU supports. The value is two less than 16/64 to better fit into system memory maps.
TCUCFG_XLATE_SLOTS	4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096	Total permitted translation requests from all DTI requesters. 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.	Number of parallel configuration table walks This value must not be greater than TCUCFG_PTW_SLOTS

Parameter name	Values	Description
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	0-8	Number of DTI-ATS requesters. TCUCFG_NUM_TBU is the total number of DTI-TBU and DTI-ATS requesters. TCUCFG_DTI_ATS is the total number of DTI-ATS requesters.
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. It is mandatory to set TCUCFG_PARTID_WIDTH to be equal to the TBUCFG_PARTID_WIDTH value for the TBU. See Translation Buffer Unit buffer configuration parameters
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. Configuring this parameter to a value exceeding the number of page table walk slots, TCUCFG_PTW_SLOTS, does not increase performance.
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 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. See the *Arm® MMU L1 System Memory Management Unit Configuration and Integration Manual*.
TCUCFG_SLOTRAM_TYPE	0, 1, 2	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. **2** 2 × one port, that is, banked configuration If you set TCUCFG_SLOTRAM_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. See the *Arm® MMU L1 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.

5.3 Translation Control Unit debug configuration parameters

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


Tip

For more detailed descriptions of the TCU buffer configuration parameters, see the *Arm® MMU L1 System Memory Management Unit Configuration and Integration Manual*.

The following table shows the TCU debug configuration parameters:

Table 5-3: 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 values according to the TCU observation interfaces.

5.4 Translation Buffer Unit configuration parameters

You can configure the AXI5 Translation Buffer Unit (TBU) parameters.

For more detailed descriptions of the TBU configuration parameters, see the *Arm® MMU L1 System Memory Management Unit Configuration and Integration Manual*.

The following table shows the TBU configuration parameters.

Table 5-4: TBU configuration parameters

Parameter name	Values	Description
TBUCFG_SID_WIDTH	8, 16, 20, 24	Stream ID width
TBUCFG_SSID_WIDTH	1, 8, 20	Substream ID width
TBUCFG_DIRECT_IDX	0, 1	Direct indexing. Must be 0 if TBUCFG_MTLB_DEPTH = 0.
TBUCFG_MTLB_PARTS	1, 2, 4, 8, 16,	Number of main TLB partitions. Must be 1 if TBUCFG_MTLB_DEPTH = 0. Must be 1 if TBUCFG_DIRECT_IDX = 1.
TBUCFG_WBUF_DEPTH	0, 8, 16, 32, 64, 128, 256, 512, 1024, 2048	The TBUCFG_WBUF_DEPTH parameter defines the maximum number of beats that the TBU write data buffer can store. The number of entries available is TBUCFG_WBUF_DEPTH. The Write Data Buffer stores write data for write transactions for which the TBU has issued translation requests to the TCU. This storage is useful when the upstream requester cannot issue transactions on the AW channel until data for previous transactions is accepted on the W channel. The Write Data Buffer also enables transactions with different AXI IDs to be output on the TBM interface in a different order than the order that they arrive at the TBS interface. The reordering enables translations to be provided when they are ready. Set TBUCFG_WBUF_DEPTH to 0 to leave the write data buffer unimplemented. For example, you might not implement the write data buffer if most TBU transactions in your implementation are reads.

Parameter name	Values	Description
TBUCFG_ROT_DEPTH	4, 8, 16, 32, 64, 128, 256, 512	The maximum number of outstanding read transactions that the TBU supports. We recommend that this parameter matches the read acceptance capability of the interconnect interface that is connected to the TBU TBM interface.
TBUCFG_WOT_DEPTH	4, 8, 16, 32, 64, 128, 256, 512	The maximum number of outstanding write transactions that the TBU supports. We recommend that this parameter matches the write acceptance capability of the interconnect interface that is connected to the TBU TBM interface.

5.5 Translation Buffer Unit buffer configuration parameters

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


Tip

For more detailed descriptions of the TBU buffer configuration parameters, see the *Arm® MMU L1 System Memory Management Unit Configuration and Integration Manual*.

The following table shows the TBU buffer configuration parameters.

Table 5-5: TBU buffer configuration parameters

Parameter name	Values	Description
TBUCFG_XLATE_SLOTS	2, 4, 8, 16, 32, 64	Number of translation slots, controlling the Hit-Under-Miss capability of the TBU. This number controls the number of transactions the TBU can perform translations simultaneously. The TBU issues translation requests to the TCU for each transaction that is not within the same page as another transaction that already has a translation request.
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 We recommend that the value of TBUCFG_MTLB_LKP_SLOTS is not greater than the TBUCFG_XLATE_SLOTS parameter value.
TBUCFG_UTLB_DEPTH	4, 8, 12, 16, 32, 64	Micro TLB depth, in entries

Parameter name	Values	Description
TBUCFG_MTLB_DEPTH	0, 32, 64, 128, 256, 512, 1024, 2048, 4096	Main TLB depth, in entries
TBUCFG_PMU_COUNTERS	4, 8, 16, 32	Number of PMU counters
TBUCFG_PARTID_WIDTH	1, 6, 9	Width of PARTID supported that is used for MPAM partitioning of the MTLB in the TBU. The number of MPAM partitions that the MTLB supports is $2^{\text{TBUCFG\_PARTID\_WIDTH}}$. This formula means that a TBUCFG_PARTID_WIDTH value of: • 1 supports up to 2 partitions • 6 supports up to 64 partitions • 9 supports up to 512 partitions It is not necessary for MPAM partitions to be equally sized, and the proportion of a cache that each partition can occupy is programmable at runtime using the MPAMCFG_CMAX registers. The MPAMCFG_CMAX registers enable each partition to be sized with an 8-bit value, which provides a precision level of 1/256 of the total capacity. Set the TBUCFG_PARTID_WIDTH value according to the number of MPAM partitions that is required. It is mandatory to set TBUCFG_PARTID_WIDTH to be equal to the TCUCFG_PARTID_WIDTH value. See Translation Control Unit buffer configuration parameters. 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_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. Set TBUCFG_CACHERAM_TYPE to 0 only for a depth of 8, where cache RAM is implemented as flops. For higher values, set TBUCFG_CACHERAM_TYPE to 1 to conserve area. There is no performance gain by setting TBUCFG_CACHERAM_TYPE to 0 for higher cache depths.

5.6 Translation Buffer Unit register slice configuration parameters

You can configure the Translation Buffer Unit (TBU) register slice as fully registered, registered on the forward path only, registered on the reverse path only, or bypass the register slice completely.


Tip

For more detailed descriptions of the register slice configuration parameters, see the *Arm® MMU L1 System Memory Management Unit Configuration and Integration Manual*.

The following table shows the TBU register slice configuration parameters.

Table 5-6: TBU register slice configuration parameters

Parameter name	Values
TBUCFG_SI_AR_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.
TBUCFG_SI_R_HNDSHK_MODE	Same values as TBUCFG_SI_AR_HNDSHK_MODE
TBUCFG_SI_AW_HNDSHK_MODE	Same values as TBUCFG_SI_AR_HNDSHK_MODE
TBUCFG_SI_W_HNDSHK_MODE	Same values as TBUCFG_SI_AR_HNDSHK_MODE
TBUCFG_SI_B_HNDSHK_MODE	Same values as TBUCFG_SI_AR_HNDSHK_MODE
TBUCFG_MI_AR_HNDSHK_MODE	Same values as TBUCFG_SI_AR_HNDSHK_MODE
TBUCFG_MI_R_HNDSHK_MODE	For TBUCFG_MI_R_HNDSHK_MODE, registering on the forward path is always enabled. If REV is selected, it is treated as FULL, and if BP is selected, it is treated as FWD.
TBUCFG_MI_AW_HNDSHK_MODE	Same values as TBUCFG_SI_AR_HNDSHK_MODE
TBUCFG_MI_W_HNDSHK_MODE	Same values as TBUCFG_SI_AR_HNDSHK_MODE
TBUCFG_MI_B_HNDSHK_MODE	For TBUCFG_MI_R_HNDSHK_MODE, registering on the forward path is always enabled. If REV is selected, it is treated as FULL, and if BP is selected, it is treated as FWD.
TBUCFG_DTI_HNDSHK_MODE	Same values as TBUCFG_SI_AR_HNDSHK_MODE.

5.7 Translation Buffer Unit I/O configuration parameters

You can configure the AXI5 Translation Buffer Unit (TBU) I/O.

For more detailed descriptions of the Translation Buffer Unit I/O configuration parameters, see the *Arm® MMU L1 System Memory Management Unit Configuration and Integration Manual*.

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

Table 5-7: TBU I/O configuration parameters

Interface and module name	Parameter name	Values	Description
TBS, TBM	`TBUCFG_ID_WIDTH`	1-32	AXI ID width for the requester and completer interfaces
TBS, TBM	`TBUCFG_DATA_WIDTH`	64, 128, 256, 512	AXI data width for the requester and completer interfaces

TBS, TBM	`TBUCFG_ARUSER_WIDTH`
	`TBUCFG_AWUSER_WIDTH`
	`TBUCFG_RUSER_WIDTH`
	`TBUCFG_WUSER_WIDTH`
	`TBUCFG_BUSER_WIDTH`
		1-128	AXI USER bus widths
			
			The widths of the TBU aruser_s, awuser_s, aruser_m, and awuser_m signals are wider than this value, because the TBU adds extra information to these signals.

TBS, TBM	`TBUCFG_STASH_SUPPORT`	0-1	Include stash ID signals To include stash ID signals for the TBS and TBM interfaces, set `TBUCFG_STASH_SUPPORT` to 1. If the system supports the AXI5 Cache_Stash_Transactions extension, set `TBUCFG_STASH_SUPPORT` to 1. If you set `TBUCFG_STASH_SUPPORT` to 0, the stash signals are still present in the AXI interface, but they are not connected inside the TBU.
TBS, TBM	`TBUCFG_LOOP_WIDTH`	1-8	AXI loopback signal width The `TBUCFG_LOOP_WIDTH` configuration parameter determines the width of the AWLOOP, BLOOP, ARLOOP, and RLOOP signals on the TBS and TBM interfaces of the TBU. You can optionally use the loopback signals for transaction tracking. See the *AMBA® AXI Protocol Specification*.

5.8 Translation Buffer Unit debug configuration parameters

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


Tip

For more detailed descriptions of the TBU debug configuration parameters, see the *Arm® MMU L1 System Memory Management Unit Configuration and Integration Manual*.

The following table shows the TBU debug configuration parameters.

Table 5-8: 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 the TBU observation interfaces.
`TBUCFG_TLB_RAS_SUPPORT`	0, 1	Enable RAS support on all RAMs in the TLB. The write buffer has no RAS support. **0** RAMs are not protected. **1** RAMs have Error Correction Code (ECC).

6. Debug capability

The MMU L1 provides debug functionality using the CoreSight™ ELA-600 Embedded Logic Analyzer.

For more information about debug capability, see the *Arm® MMU L1 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 MMU L1.

Configuration options

For TCU configuration options, see Translation Control Unit debug configuration parameters.

For TBU configuration options, see Translation Buffer Unit debug configuration parameters.

Signals

For TCU observation signals, see TCU observation interfaces.

For TBU observation signals, see TBU observation interfaces.

7. Programmers model

The programmers model describes the MMU L1 memory map and registers.

The following information applies to the MMU L1 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 L1 registers are accessed using the PROG APB5 completer interface on the TCU, and cannot be accessed directly through any other completer interfaces. Some registers are 64 bits, but the PROG APB5 interface is 32 bits wide. 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 programmers model contains separate TBU and TCU regions for internal control, RAS, and identification registers. Writes to unmapped or reserved registers are ignored, and reads Should-Be-Zero (SBZ).

Non-secure accesses to Secure registers are **RAZ/WI**. The MMU L1 implements the identification register scheme that the SMMUv3 architecture defines.

MMU L1 also contains non-implemented registers. For more information on non-implemented registers, see Non-implemented registers.

7.1 SMMUv3 registers

MMU L1 implements many of the SMMU architectural registers.

See SMMU architectural registers.

MMU L1 implements an SMMUv3 Performance Monitor Counter Group (PMCG) in the TCU and in each TBU. See SMMUv3 Performance Monitor Counter Group registers.

7.1.1 SMMU architectural registers

MMU L1 implements many of the SMMU architectural registers.

The *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2* define these registers. All writeable register fields reset to 0 unless the SMMU architecture specifies otherwise.

The following table shows the SMMUv3 architectural registers that MMU L1 implements.

Table 7-1: 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
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

Register	Name	Description
SMMU_S_GERROR_IRQ_CFG0 - SMMU_S_GERROR_IRQ_CFG2	Secure Global Error IRQ Configuration registers	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 feature Identification Register	Provides information about the features that the SMMU implementation supports
SMMU_IDR5	SMMU feature Identification Register	Provides information about the features that the SMMU implementation supports
SMMU_IIDR	Implementation Identification Register	Provides implementer, part, and revision information for the SMMU implementation
SMMU_AIDR	Architecture Identification Register	Identifies the SMMU architecture version to which the implementation conforms
SMMU_CR0	Non-secure global Control Register 0	Provides the controls for the global configuration of the Non-secure SMMU
SMMU_CR0ACK	Non-secure global Control Register 0 update Acknowledge register	Provides acknowledgment of completion of updates to SMMU_CR0
SMMU_CR1	Non-secure global Control Register 1	Provides the controls for Non-secure table and queue access attributes
SMMU_CR2	Non-secure global Control Register 2	Provides the controls for the configuration of the global Non-secure features
SMMU_GBPA	Non-secure Global Bypass Attribute register	Controls the global bypass attributes that are used for transactions from Non-secure streams when the MMU is disabled
SMMU_IRQ_CTRL	Non-secure Interrupt Control register	Provides IRQ enable flags for edge-triggered wired outputs, if implemented, and MSI writes, if implemented
SMMU_IRQ_CTRLACK	Non-secure Interrupt Control register update Acknowledge register	Provides acknowledgment of the completion of updates to SMMU_IRQ_CTRL
SMMU_GERROR	Non-secure Global Error status register	Provides information about Non-secure global programming interface errors
SMMU_GERRORN	Non-secure Global Error acknowledgment register	Contains the acknowledgment fields for SMMU_GERROR errors

Register	Name	Description
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

7.1.2 SMMUv3 Performance Monitor Counter Group registers

MMU L1 implements an SMMUv3 Performance Monitor Counter Group (PMCG) in the TCU and in each TBU.

The following table lists the registers that MMU L1 implements in each PMCG.

Table 7-2: SMMUv3 PMCG registers

Register	Name	Description
SMMU_PMCG_EVCNTR0 - SMMU_PMCG_EVCNTR63	SMMU PMCG Event Counter registers	Contains the values of the event counters The maximum supported number of registers is 64 depending on the `TCUCFG_PMU_COUNTERS` parameter value setting.
SMMU_PMCG_EVTYPER0 - SMMU_PMCG_EVTYPER63	SMMU PMCG Event Type configuration registers	Configures the events that the corresponding counter counters The maximum supported number of registers is 64 depending on the `TCUCFG_PMU_COUNTERS` parameter value setting.
SMMU_PMCG_SVR0 - SMMU_PMCG_SVR63	SMMU PMCG Shadow Value Registers	Contains the shadow value of the corresponding event counters The maximum supported number of registers is 64 depending on the `TCUCFG_PMU_COUNTERS` parameter value setting.
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

Register	Name	Description
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

7.2 Main MMU L1 memory map

The main MMU L1 memory map includes the TCU and all TBUs, and the maximum number of implemented TBUs.


Note

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

The following table shows the full memory map.

Table 7-3: Main MMU L1 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

The TBU number is assigned according to the TID value provided by the TBU in its initial DTI connection request. The registers for that TBU should be accessed according to the address range for its TID, as described in the preceding table.

7.2.1 TCU memory map

The TCU memory map contains various categories of registers.

The TCU implementation defined registers include the following:

- TCU microarchitectural registers for controlling microarchitectural features
- TCU system discovery registers
- TCU RAS registers
- TCU PMU registers

The following registers are also included:

- TCU PIU integration registers.
- 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 L1 TCU memory map.

Table 7-4: MMU L1 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 MMU L1 allocates the TCU implementation defined PMCG and MPAM registers to regions of the TCU address space. Other regions are reserved.

Table 7-5: TCU PMCG, RAS, and MPAM register allocation to regions of TCU address space

Address range	Description
0x00FD0-0x00FFC	SMMU ID registers
0x02000-0x02FFC	Performance Monitor, page 0
0x03000-0x03FFC	MPAM Non-secure registers
0x08E00-0x08E78	Microarchitectural registers, System discovery registers, and 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

7.3 MMU L1 register summaries

The register summary describes the MMU L1 registers and some key characteristics.

7.3.1 TCU identification register summary

The MMU L1 contains TCU identification registers.

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

Table 7-6: TCU identification register summary

Offset	Name	Type	Description
0x00FFC	SMMU_CIDR3	RO	TCU component and peripheral ID registers. 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	

7.3.2 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 7-7: TCU and TBU PMU identification register summary

Offset	Name	Type	Description
0x02FB8	SMMU_PMCG_PMAUTHSTATUS	RO	TCU PMU registers TBU performance monitor unit registers
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	

7.3.3 TCU Reliability, Availability, and Serviceability register summary

The MMU L1 contains TCU Reliability, Availability, and Serviceability (RAS) registers.

The following table shows the TCU RAS registers in offset order from the base memory address. The TCU RAS registers are 64-bits wide. The top 32 bits are Reserved and are omitted in this document for clarity.

Table 7-8: TCU RAS register summary

Offset	Name	Type	Width	Description
0x08E80	TCU_ERRFR	RO, Secure	32-bit	TCU_ERRFR register
0x08E88	TCU_ERRCTLR	RW, Secure	32-bit	TCU_ERRCTLR register
0x08E90	TCU_ERRSTATUS	RW, Secure	32-bit	TCU_ERRSTATUS register
0x08EC0	TC_ERRGEN	RW, Secure	32-bit	TCU_ERRGEN register

7.3.4 TCU microarchitectural register summary

The MMU L1 contains TCU microarchitectural registers.

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

Table 7-9: TCU microarchitectural register summary

Offset	Name	Type	Width	Description
0x08E00	TCU_CTRL	RW	32-bit	TCU_CTRL register
0x08E04	TCU_QOS	RW	32-bit	TCU_QOS register
0x08E08	TCU_CFG	RO	32-bit	TCU_CFG register
0x08E10	TCU_STATUS	RO	32-bit	TCU_STATUS register
0x08E18	TCU_SCR	RW, Secure	32-bit	TCU_SCR register
0x09000-0x093FC	TCU_NODE_CTRLn	RW	32-bit	TCU_NODE_CTRLn register
0x09400-0x097FC	TCU_NODE_STATUSn	RO	32-bit	TCU_NODE_STATUSn register
0x09800-0x0981C	TCU_WC_SxLy_CMAX	RW	32-bit	TCU_WC_SxLy_CMAX registers

7.3.5 TCU system discovery register summary

The MMU L1 contains TCU system discovery registers.

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

Table 7-10: TCU system discovery register summary

Offset	Name	Type	Width	Description
0x08E34	TCU_SYSDISC0	RO	32-bit	TCUCFG_WC_DEPTH
0x08E38	TCU_SYSDISC1	RO	32-bit	TCUCFG_CC_DEPTH
0x08E3C	TCU_SYSDISC2	RO	32-bit	TCUCFG_WC_WAYS
0x08E40	TCU_SYSDISC3	RO	32-bit	TCUCFG_WC_BANKS
0x08E44	TCU_SYSDISC4	RO	32-bit	TCUCFG_XLATE_SLOTS
0x08E48	TCU_SYSDISC5	RO	32-bit	TCUCFG_PTW_SLOTS
0x08E4C	TCU_SYSDISC6	RO	32-bit	TCUCFG_CTW_SLOTS
0x08E50	TCU_SYSDISC7	RO	32-bit	TCUCFG_CC_IDXGEN_MODE
0x08E54	TCU_SYSDISC8	RO	32-bit	TCUCFG_DTI_ATS
0x08E58	TCU_SYSDISC9	RO	32-bit	TCUCFG_NUM_TBU
0x08E5C	TCU_SYSDISC10	RO	32-bit	TCUCFG_PMU_COUNTERS
0x08E60	TCU_SYSDISC11	RO	32-bit	TCUCFG_PARTID_WIDTH
0x08E64	TCU_SYSDISC12	RO	32-bit	TCUCFG_HZU_DEPTH
0x08E68	TCU_SYSDISC13	RO	32-bit	TCUCFG_PREFETCH_SUPPORTED
0x08E6C	TCU_SYSDISC14	RO	32-bit	TCUCFG_DATARAM_TYPE
0x08E70	TCU_SYSDISC15	RO	32-bit	TCUCFG_SLOTRAM_TYPE

Offset	Name	Type	Width	Description
0x08E74	TCU_SYSDISC16	RO	32-bit	TCUCFG_CACHERAM_TYPE
0x08E78	TCU_SYSDISC17	RO	32-bit	TCUCFG_QTW_DATA_WIDTH

7.3.6 TCU integration register summary

The MMU L1 contains TCU integration registers.

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

Table 7-11: TCU integration register summary

Offset	Name	Type	Width	Description
0x08E20	ITEN	RW	32-bit	ITEN register for the TCU
0x08E24	ITOP_PIU	RW	32-bit	ITOP register for the TCU Programmer Interface Unit
0x08E2C	ITOP_TMU	RW	32-bit	ITOP register for the TCU Translation Management Unit
0x08E30	ITIN_TMU	RO	32-bit	ITIN register for the TCU Translation Management Unit

7.4 TCU registers

The MMU L1 Translation Control Unit (TCU) registers comprise of several groups of registers.

These register groups are:

TCU component and peripheral ID registers

MMU L1 implements an SMMUv3 Performance Monitor Counter Group (PMCG) in the TCU and in each TBU, see TCU component and peripheral ID registers.

TCU PMU registers

The TCU PMU registers are on a separate 64KB page therefore a guest OS can page it for direct access.

TCU RAS registers

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

TCU microarchitectural registers

Set the TCU microarchitectural registers at boot time to optimize TCU behavior for your system.

TCU system discovery registers

The TCU system discovery registers discover components in the system.

TCU PIU integration registers

Programmer Interface Unit (PIU) integration registers, see TCU PIU integration registers

TCU TMU integration registers

MMU L1 also contains TCU TMU integration registers.

7.4.1 TCU component and peripheral ID registers

There are various TCU component and peripheral ID registers in MMU L1.

These registers are explained in SMMUv3 Performance Monitor Counter Group registers.

7.4.2 TCU PMU registers

There are various Performance Monitor Unit (PMU) registers in MMU L1. The Performance Monitor counter registers, on a separate 64KB page, enable it to be paged for direct access from a Guest OS.

There are several categories of TCU PMU registers:

Implemented and non-implemented registers

MMU L1 implements an SMMUv3 Performance Monitor Counter Group (PMCG) in the TCU and in each TBU, see SMMUv3 Performance Monitor Counter Group registers and Non-implemented registers.

Events

Events are translation requests that correspond to a translation slot allocation.

SMMU_PMCG_CFGR fields

For more information on SMMU_PMCG_CFGR fields, see SMMUv3 Performance Monitor Counter Group registers.

SMMU_PMCG_CEID{0-1} registers

The TCU SMMU_PMCG_CEID{0-1} registers indicate the architectural events that the TCU supports.

PMU ID registers

TMC PMU ID registers only appear in Performance Monitor Page 0.

7.4.2.1 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 for TCU and TBU, see:

- MMU L1 TCU events
- MMU L1 TBU events

The following events are also counted for prefetch accesses:

0x80-0x90
Walk cache events.

0x92-0x94
Configuration cache events.

0xC0-0xC8
RAS events.

7.4.2.2 SMMU_PMCG_CFGR fields

An MMU L1 implementation assumes fixed values for SMMU_PMCG_CFGR, and these values define behavioral aspects of the implementation.

For information about the SMMU_PMCG_CFGR fields values, see SMMU_PMCG_CFGR fields.

7.4.2.3 TCU SMMU_PMCG_CEID{0-1} registers

The SMMU_PMCG_CEID{0-1} registers indicate the architectural events that the TCU supports. We describe them as 64-bit registers, but they are accessed 32 bits at a time through the 32-bit PROG interface.

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

Table 7-12: TCU SMMU_PMCG_CEID{0-1} register values

Address	Name	Reset value
0x02E20	SMMU_PMCG_CEID0	0x0000007F
0x02E28	SMMU_PMCG_CEID1	0x00000000

7.4.2.4 TMC PMU ID registers

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

The following table shows the TMC PMU ID registers.

Table 7-13: TMC 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
0x02FEC	SMMU_PMCG_PIDR3, Peripheral ID3	[3:0]	0x00	CMOD
0x02FE8	SMMU_PMCG_PIDR2, Peripheral ID2	[7:4]	0x00 for r0	REVISION, major revision
0x02FE8	SMMU_PMCG_PIDR2, Peripheral ID2	[3]	1	JEDEC-assigned value for DES always used
0x02FE8	SMMU_PMCG_PIDR2, Peripheral ID2	[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
0x02FE4	SMMU_PMCG_PIDR1, Peripheral ID1	[3:0]	0x4	PART_1: bits [11:8] of the Part number
0x02FE0	SMMU_PMCG_PIDR0, Peripheral ID0	[7:0]	0x8A	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
0x02FD0	SMMU_PMCG_PIDR4, Peripheral ID4	[3:0]	0x4	DES_2: JEP106 Designer continuation code
0x02FB8	SMMU_PMCG_PMAUTHSTATUS	[7:0]	0x00	No authentication interface is implemented

Implement the PMDEVARCH and PMDEVTYPE registers according to the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2*.

7.4.3 TCU RAS registers

The Reliability, Availability, and Serviceability (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**.

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

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

7.4.3.1 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

TCU Reliability, Availability, and Serviceability register summary

Address offset

`0x08E80`

Type

S, RO

Bit descriptions

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

Figure 7-1: TCU_ERRFR register bit assignments

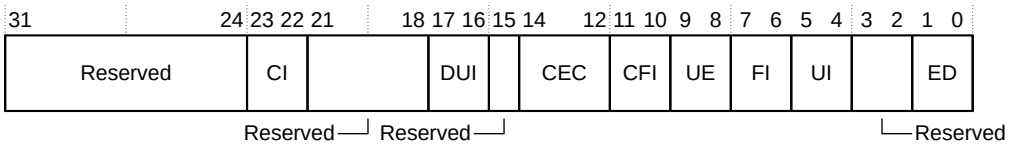


Table 7-14: TCU_ERRFR register bit descriptions

Bits	Name	Description	Value
[31:24]	-	Reserved	-
[23:22]	CI	Critical Error Interrupt is always enabled	`0b01`
[21:18]	-	Reserved	-
[17:16]	DUI	Does not support this feature	`0b00`
[15]	-	Reserved	-
[14:12]	CEC	Does not implement the standard corrected error counter model	`0b000`
[11:10]	CFI	Does not support this feature	`0b00`
[9:8]	UE	In-band error signaling feature is always enabled	`0b01`
[7:6]	FI	Fault handling interrupt is controllable	`0b10`
[5:4]	UI	Error Recovery Interrupt always enabled for UE	`0b01`
[3:2]	-	Reserved	-
[1:0]	ED	Error detection is always enabled	`0b01`

7.4.3.2 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

TCU Reliability, Availability, and Serviceability register summary

Address offset

`0x08E88`

Type

S, RW

Reset value

8

Bit descriptions

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

Figure 7-2: TCU_ERRCTLR register bit assignments

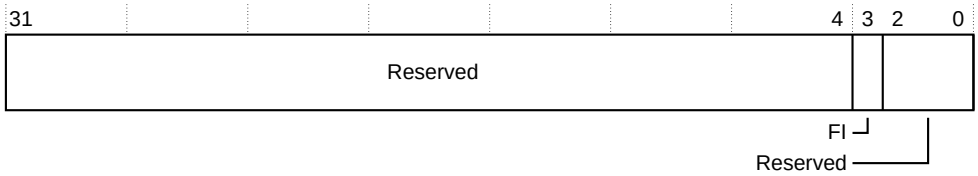


Table 7-15: TCU_ERRCTLR register bit descriptions

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

7.4.3.3 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

TCU Reliability, Availability, and Serviceability register summary

Address offset

`0x08E90`

Type

Secure, RW

Reset value

0

Bit descriptions

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

Figure 7-3: TCU_ERRSTATUS register bit assignments

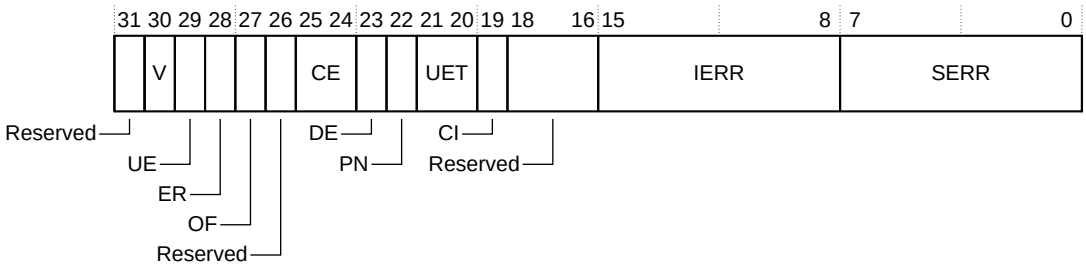


Table 7-16: TCU_ERRSTATUS register bit descriptions

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 1 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 1 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 requester 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. **0b00** No correctable errors recorded **0b10** At least 1 Corrected error recorded Other values are Reserved. This field is cleared by writing `0b11` to it. If OF is set and not being cleared, the write is ignored. A write of any value other than `0b11` is ignored.

Bits	Name	Description
[23]	DE	Deferred error, or errors. The possible values of this bit are: **0** No errors were deferred **1** At least 1 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.
[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

Bits	Name	Description
[15:8]	IERR	implementation defined error code. When SERR≠0, this field indicates the source of the error: **0x12** PIU CMD RPOISON **0x11** TMU CCB MCC DATA **0x10** TMU CCB MCC TAGS **0x0F** TMU WCB MWC DATA **0x0E** TMU WCB MWC TAGS **0x0D** TMU CCB MCC REPL **0x0C** TMU CCB MCC PCNT **0x0B** TMU CCB MCC PLIM **0x0A** TMU WCB MWC REPL **0x09** TMU WCB MWC PCNT **0x08** TMU WCB MWC PLIM **0x07-0x06** Reserved **0x05** TMU HTTU RAM **0x04** TMU TWB WMB SCRATCH **0x03** TMU TWB WMB WLK STATUS **0x02** TMU TWB WMB LKP STATUS **0x01** TMU HZU PTR **0x00** 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.

7.4.3.4 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 TCU_ERRSTATUS.

When this register is updated, TCU_ERRSTATUS 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 == `0b00` or CE == `0b10`):
  - ERRGEN.UE is set and CE is legal
  - ERRGEN.DE is set and CE is legal
  - ERRGEN.CE is set to `0b10`
- One of the following is true:
  - UET == `0b00`
  - UET == `0b11` and UE == 1

All other fields can take any legal value. Any other write is considered invalid and the SMMU behavior is **UNPREDICTABLE**. If there is a valid error record write, then MMU L1 generates the appropriate interrupt or interrupts.

Configurations

The TCU_ERRGEN register is available in all configurations.

Attributes

The TCU_ERRGEN register attributes are as follows:

Width

32-bit

Functional group

TCU Reliability, Availability, and Serviceability register summary

Address offset

`0x08EC0`

Type

Secure, RW

Reset value

0

Bit descriptions

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

Figure 7-4: TCU_ERRGEN register bit assignments

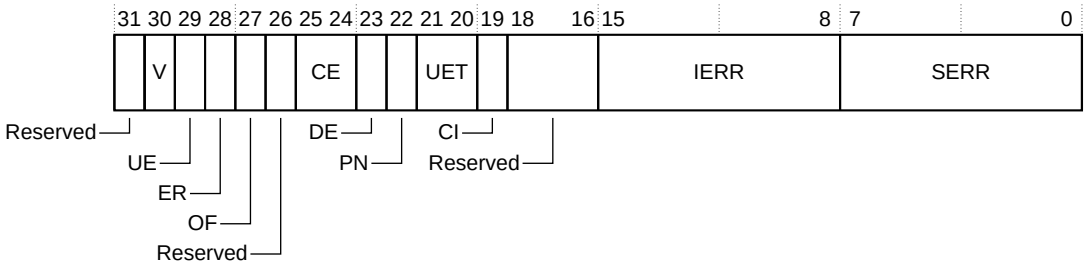


Table 7-17: TCU_ERRGEN register bit descriptions

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 1 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 1 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, or External abort, is reported. **1** The node to the requester 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. **0b00** No correctable errors recorded **0b10** At least 1 Corrected error recorded Other values are Reserved. This field is cleared by writing `0b11` to it. If OF is set and not being cleared, the write is ignored. A write of any value other than `0b11` is ignored.

Bits	Name	Description
[23]	DE	Deferred error, or errors. The possible values of this bit are: **0** No errors were deferred **1** At least 1 error was not corrected and deferred This error is raised when wpoison is set in TBU. 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.
[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

Bits	Name	Description
[15:8]	IERR	Implementation defined error code. When SERR≠0, this field indicates the source of the error: **0x12** PIU CMD RPOISON **0x11** TMU CCB MCC DATA **0x10** TMU CCB MCC TAGS **0x0F** TMU WCB MWC DATA **0x0E** TMU WCB MWC TAGS **0x0D** TMU CCB MCC REPL **0x0C** TMU CCB MCC PCNT **0x0B** TMU CCB MCC PLIM **0x0A** TMU WCB MWC REPL **0x09** TMU WCB MWC PCNT **0x08** TMU WCB MWC PLIM **0x07-0x06** Reserved **0x05** TMU HTTU RAM **0x04** TMU TWB WMB SCRATCH **0x03** TMU TWB WMB WLK STATUS **0x02** TMU TWB WMB LKP STATUS **0x01** TMU HZU PTR **0x00** 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.

7.4.4 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 TCU_SCR is Secure-only. Non-secure access to this register is Read-As-Zero (**RAZ**)/Write-Ignored (**WI**).

The TCU_SCR.NS_UARCH bit controls Non-secure access to the 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 TCU_CTRL, TCU_QOS, TCU_NODE_CTRLn, and TCU_WC_SxLy_CMAX 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.

7.4.4.1 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

TCU microarchitectural register summary

Address offset

0x08E00

Type

RW

Reset value

0

Bit descriptions

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

Figure 7-5: TCU_CTRL register bit assignments

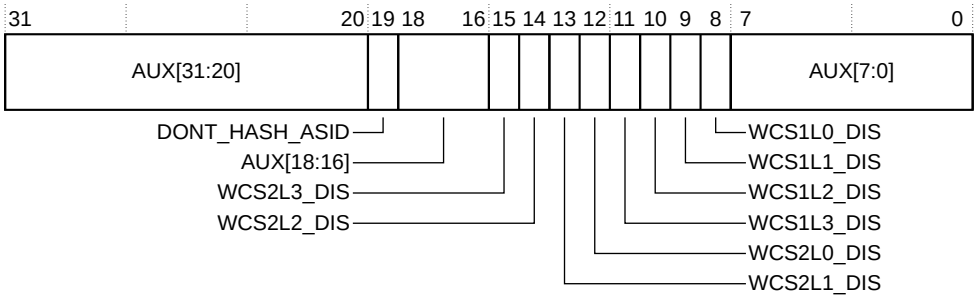


Table 7-18: TCU_CTRL register bit descriptions

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	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].
[13]	WCS2L1_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].
[12]	WCS2L0_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].
[11]	WCS1L3_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].
[10]	WCS1L2_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].
[9]	WCS1L1_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].
[8]	WCS1L0_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].
[7:0]	AUX[7:0]	Reads the value written, but has no other effect

7.4.4.2 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

TCU microarchitectural register summary

Address offset

`0x08E04`

Type

RW

Reset value

0

Bit descriptions

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

Figure 7-6: TCU_QOS register bit assignments

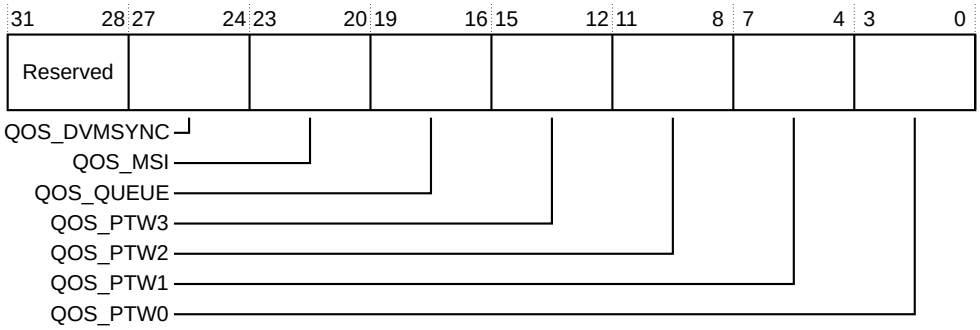


Table 7-19: TCU_QOS register bit descriptions

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

7.4.4.3 TCU_CFG register

This section describes the TCU Configuration Information register.

Configurations

This register is available in all configurations.

Attributes

The TCU_CFG register attributes are as follows:

Width

32-bit

Functional group

TCU microarchitectural register summary

Address offset

`0x08E08`

Type

RO

Bit descriptions

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

Figure 7-7: TCU_CFG register bit assignments

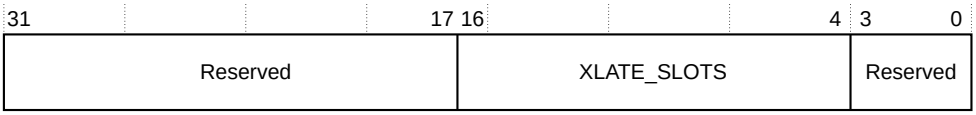


Table 7-20: TCU_CFG register bit descriptions

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 Translation Control Unit buffer configuration parameters.
[3:0]	-	Reserved

7.4.4.4 TCU_STATUS register

This section describes the TCU Status Information register.

Configurations

This register is available in all configurations.

Attributes

The TCU_STATUS register attributes are as follows:

Width

32-bit

Functional group

TCU microarchitectural register summary

Address offset

`0x08E10`

Type

RO

Reset value

0

Bit descriptions

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

Figure 7-8: TCU_STATUS register bit assignments

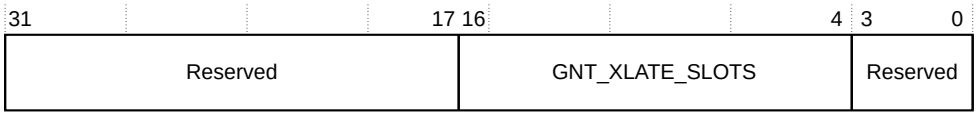


Table 7-21: TCU_STATUS register bit descriptions

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

7.4.4.5 TCU_NODE_CTRLn register

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

Each DTI requester 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 Translation Control Unit buffer configuration parameters.

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.PRI_LEVEL)


Note

When the priority level is established, these are translated into QoS values by selection of the appropriate TCU_QOS.QOS_PTW* field, 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_CTRLn register is available in all configurations.

Attributes

The TCU_NODE_CTRLn register attributes are as follows:

Width

32-bit

Functional group

TCU microarchitectural register summary

Address offset

`0x09000-0x093FC`

Type

RW

Reset value

0

Bit descriptions

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

Figure 7-9: TCU_NODE_CTRLn register bit assignments

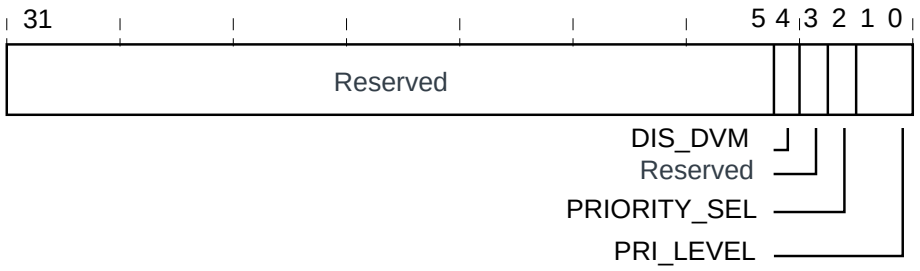


Table 7-22: TCU_NODE_CTRLn register bit descriptions

Bits	Name	Description
[31:5]	_	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	Set this to 0 to use the DTI node ID (default). The priority of all translation requests from the DTI requester are given the priority that the PRI_LEVEL field specifies.
[1:0]	PRI_LEVEL	Default Priority level for this DTI requester. Translation requests from a node with a higher priority level are normally progressed before translation requests from a node with a lower priority level.

7.4.4.6 TCU_NODE_STATUSn register

The TCU_NODE_STATUSn register provides the status for each node, similar to the TCU_NODE_CTRLn register. Each node has a single status register.

Configurations

The TCU_NODE_STATUSn register is available in all configurations.

Attributes

The TCU_NODE_STATUSn register attributes are as follows:

Width

32-bit

Functional group

TCU microarchitectural register summary

Address offset

0x09400-0x097FC

Type

RO

Bit descriptions

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

Figure 7-10: TCU_NODE_STATUS register bit assignments

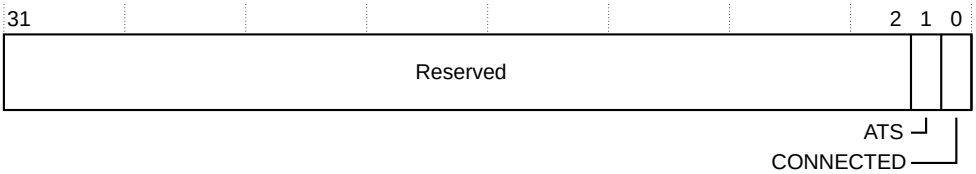


Table 7-23: TCU_NODE_STATUSn register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved
[1]	ATS	Indicates whether the node implements ATS: **0** The node is a TBU and connects using DTI-TBU **1** The node is a PCIe Root Port supporting ATS and connects 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 the 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.

7.4.4.7 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

TCU microarchitectural register summary

Address offset

`0x08E18`

Type

Secure, RW

Reset value

sec_override signal. See TCU tie-off signals.

Bit descriptions

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

Figure 7-11: TCU_SCR register bit assignments

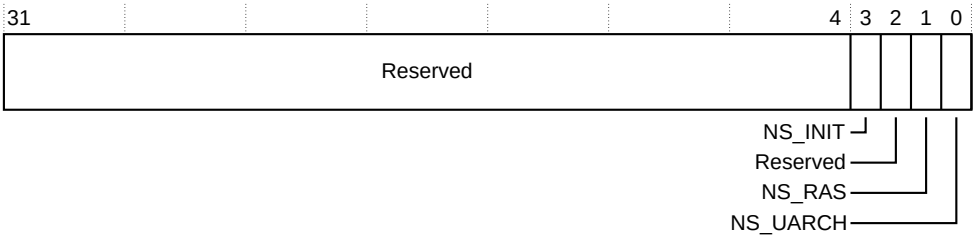


Table 7-24: TCU_SCR register bit descriptions

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 signal sets the reset value of this bit. See TCU tie-off signals.

Bits	Name	Description
[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 • 0x09000-0x0981C The sec_override input sets the reset value of this bit. See TCU tie-off signals. If you use Secure translation, we recommend that software does not set this bit.

7.4.4.8 TCU_WC_SxLy_CMAX registers

TCU_WC_SxLy_CMAX registers enable you to set maximum capacities for the TCU walk cache RAMs, for each 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® Memory System Resource Partitioning and Monitoring (MPAM) System Component Specification* defines. These registers are readable and writable.

The following table describes the TCU_WC_SxLy_CMAX registers.

Table 7-25: TCU_WC_SxLy_CMAX registers

Address	Name	Field	Position	Reset	Meaning
0x09800	TCU_WC_S1L0_CMAX	CMAX	[15:0]	0x0000_FF00	Maximum capacity for TCU Walk Cache stage 1 level 0
0x09804	TCU_WC_S1L1_CMAX	CMAX	[15:0]	0x0000_FF00	Maximum capacity for TCU Walk Cache stage 1 level 1
0x09808	TCU_WC_S1L2_CMAX	CMAX	[15:0]	0x0000_FF00	Maximum capacity for TCU Walk Cache stage 1 level 2
0x0980C	TCU_WC_S1L3_CMAX	CMAX	[15:0]	0x0000_FF00	Maximum capacity for TCU Walk Cache stage 1 level 3
0x09810	TCU_WC_S2L0_CMAX	CMAX	[15:0]	0x0000_FF00	Maximum capacity for TCU Walk Cache stage 2 level 0
0x09814	TCU_WC_S2L1_CMAX	CMAX	[15:0]	0x0000_FF00	Maximum capacity for TCU Walk Cache stage 2 level 1
0x09818	TCU_WC_S2L2_CMAX	CMAX	[15:0]	0x0000_FF00	Maximum capacity for TCU Walk Cache stage 2 level 2
0x0981C	TCU_WC_S2L3_CMAX	CMAX	[15:0]	0x0000_FF00	Maximum capacity for TCU Walk Cache stage 2 level 3

The implementation defines how many bits are used. For MMU L1, the number of bits is 8. As the value represents a fixed-point binary fraction, the MSB of those 16 bits is significant. Only bits [15:8] have any impact.

7.4.5 TCU system discovery registers

The System Discovery (SYSDISC) registers contain important hardware configuration-related information that might be relevant for software.

7.4.5.1 TCU_SYSDISC0

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

TCU system discovery register summary

Address offset

`0x08E34`

Type

RO

Reset value

`TCUCFG_WC_DEPTH`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-12: TCU_SYSDISC0 register bit assignments

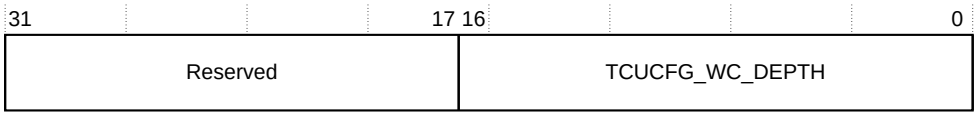


Table 7-26: TCU_SYSDISC0 register bit descriptions

Bits	Name	Description
[31:17]	-	Reserved
[16:0]	TCUCFG_WC_DEPTH	The read data reflects the chosen parameter value, for example: • `0x0_0008` : 8 • ... • `0x1_0000` : 65536

7.4.5.2 TCU_SYSDISC1

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

TCU system discovery register summary

Address offset

`0x08E38`

Type

RO

Reset value

`TCUCFG_CC_DEPTH`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-13: TCU_SYSDISC1 register bit assignments

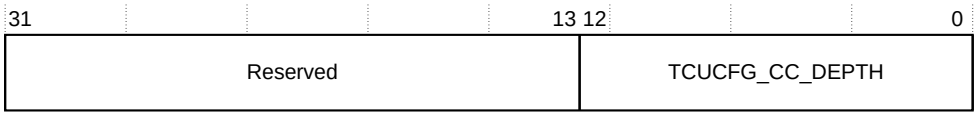


Table 7-27: TCU_SYSDISC1 register bit descriptions

Bits	Name	Description
[31:13]	-	Reserved
[12:0]	TCUCFG_CC_DEPTH	The read data reflects the chosen parameter value, for example: • `0x0004` : 4 • ... • `0x1000` : 4096

7.4.5.3 TCU_SYSDISC2

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

TCU system discovery register summary

Address offset

`0x08E3C`

Type

RO

Reset value

`TCUCFG_WC_WAYS`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-14: TCU_SYSDISC2 register bit assignments

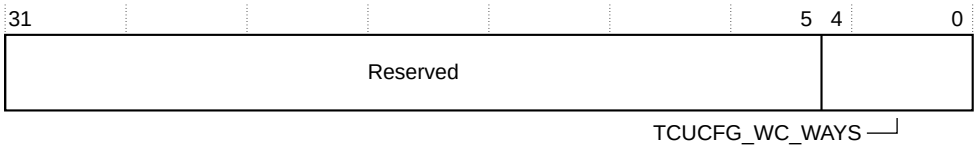


Table 7-28: TCU_SYSDISC2 register bit descriptions

Bits	Name	Description
[31:5]	-	Reserved
[4:0]	TCUCFG_WC_WAYS	The read data reflects the chosen parameter value, for example: • `0x04` : 4 • ... • `0x10` : 16

7.4.5.4 TCU_SYSDISC3

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

TCU system discovery register summary

Address offset

`0x08E40`

Type

RO

Reset value

`TCUCFG_WC_BANKS`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-15: TCU_SYSDISC3 register bit assignments

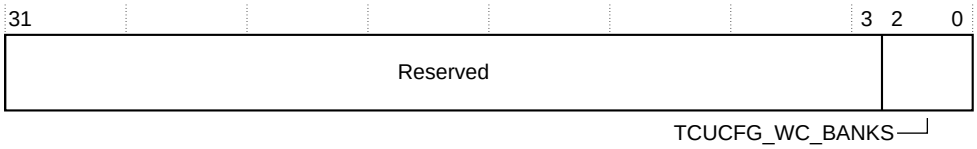


Table 7-29: TCU_SYSDISC3 register bit descriptions

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

7.4.5.5 TCU_SYSDISC4

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

TCU system discovery register summary

Address offset

`0x08E44`

Type

RO

Reset value

`TCUCFG_XLATE_SLOTS`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-16: TCU_SYSDISC4 register bit assignments

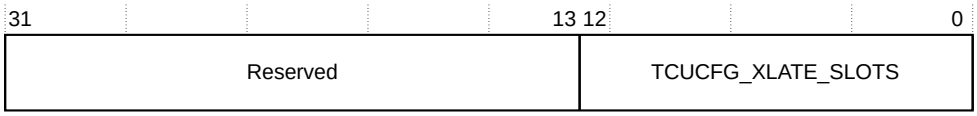


Table 7-30: TCU_SYSDISC4 register bit descriptions

Bits	Name	Description
[31:13]	-	Reserved
[12:0]	TCUCFG_XLATE_SLOTS	The read data reflects the chosen parameter value, for example: • `0x0004` : 4 • ... • `0x1000` : 4096

7.4.5.6 TCU_SYSDISC5

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

TCU system discovery register summary

Address offset

`0x08E48`

Type

RO

Reset value

`TCUCFG_PTW_SLOTS`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-17: TCU_SYSDISC5 register bit assignments

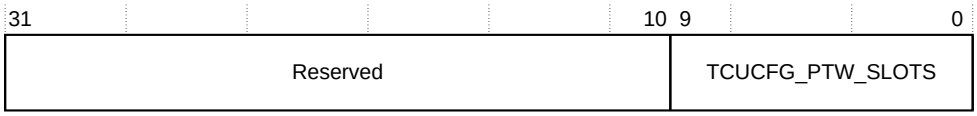


Table 7-31: TCU_SYSDISC5 register bit descriptions

Bits	Name	Description
[31:10]	-	Reserved
[9:0]	TCUCFG_PTW_SLOTS	The read data reflects the chosen parameter value, for example: • `0x002` : 2 • ... • `0x200` : 512

7.4.5.7 TCU_SYSDISC6

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

TCU system discovery register summary

Address offset

`0x08E4C`

Type

RO

Reset value

`TCUCFG_CTW_SLOTS`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-18: TCU_SYSDISC6 register bit assignments

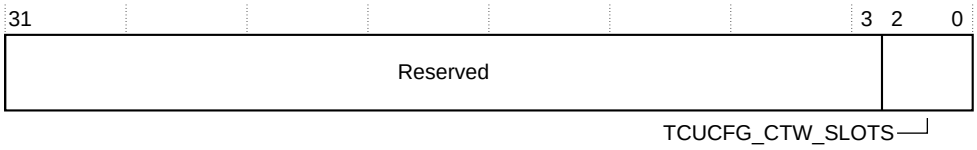


Table 7-32: TCU_SYSDISC6 register bit descriptions

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

7.4.5.8 TCU_SYSDISC7

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

TCU system discovery register summary

Address offset

`0x08E50`

Type

RO

Reset value

`TCUCFG_CC_IDXGEN_MODE`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-19: TCU_SYSDISC7 register bit assignments

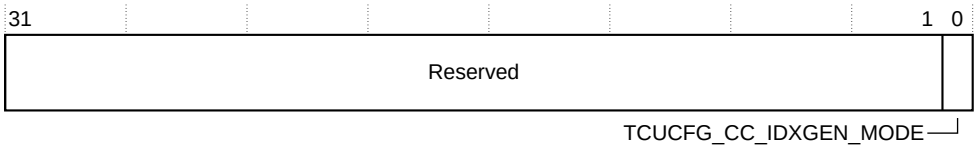


Table 7-33: TCU_SYSDISC7 register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved
[0]	TCUCFG_CC_IDXGEN_MODE	The read data reflects the chosen parameter value: • `0b0` : 0 • `0b1` : 1

7.4.5.9 TCU_SYSDISC8

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

TCU system discovery register summary

Address offset

`0x08E54`

Type

RO

Reset value

`TCUCFG_DTI_ATS`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-20: TCU_SYSDISC8 register bit assignments

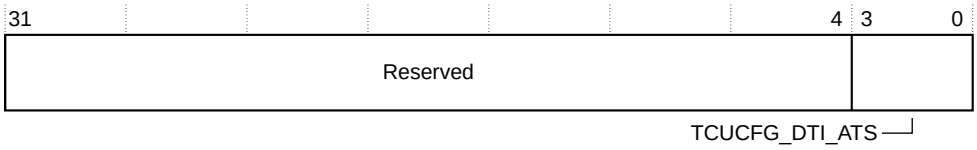


Table 7-34: TCU_SYSDISC8 register bit descriptions

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

7.4.5.10 TCU_SYSDISC9

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

TCU system discovery register summary

Address offset

`0x08E58`

Type

RO

Reset value

`TCUCFG_NUM_TBU`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-21: TCU_SYSDISC9 register bit assignments

Table 7-35: 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: • `0x01` : 1 • ... • `0x3E` : 62 Acceptable values are 14 and 62.

7.4.5.11 TCU_SYSDISC10

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

TCU system discovery register summary

Address offset

`0x08E5C`

Type

RO

Reset value

`TCUCFG_PMU_COUNTERS`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-22: TCU_SYSDISC10 register bit assignments

Table 7-36: TCU_SYSDISC10 register bit descriptions

Bits	Name	Description
[31:6]	-	Reserved
[5:0]	TCUCFG_PMU_COUNTERS	The read data reflects the chosen parameter value, for example: • `0x04` : 4 • ... • `0x20` : 32

7.4.5.12 TCU_SYSDISC11

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

TCU system discovery register summary

Address offset

`0x08E60`

Type

RO

Reset value

`TCUCFG_PARTID_WIDTH`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-23: TCU_SYSDISC11 register bit assignments

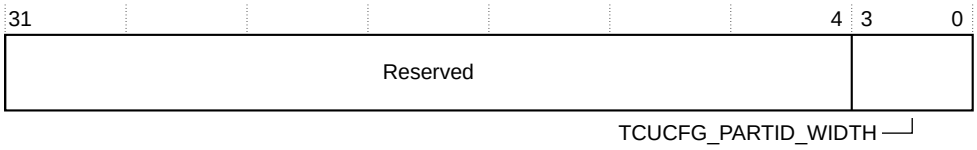


Table 7-37: TCU_SYSDISC11 register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved
[3:0]	TCUCFG_PARTID_WIDTH	The read data reflects the chosen parameter value, for example: • `0b0001` : 1 • ... • `0b1001` : 9

7.4.5.13 TCU_SYSDISC12

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

TCU system discovery register summary

Address offset

`0x08E64`

Type

RO

Reset value

`TCUCFG_HZU_DEPTH`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-24: TCU_SYSDISC12 register bit assignments

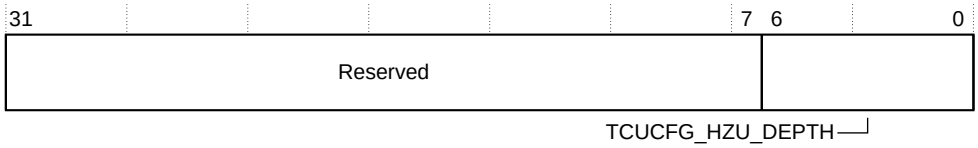


Table 7-38: TCU_SYSDISC12 register bit descriptions

Bits	Name	Description
[31:7]	-	Reserved
[6:0]	TCUCFG_HZU_DEPTH	The read data reflects the chosen parameter value, for example: • `0x02` : 2 • ... • `0x40` : 64

7.4.5.14 TCU_SYSDISC13

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

TCU system discovery register summary

Address offset

`0x08E68`

Type

RO

Reset value

`TCUCFG_PREFETCH_SUPPORTED`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-25: TCU_SYSDISC13 register bit assignments

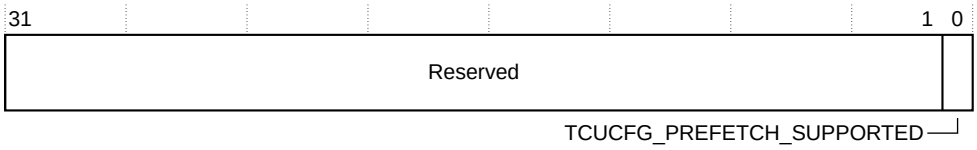


Table 7-39: TCU_SYSDISC13 register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved
[0]	TCUCFG_PREFETCH_SUPPORTED	The read data reflects the chosen parameter value: • `0b0` : 0 • `0b1` : 1

7.4.5.15 TCU_SYSDISC14

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

TCU system discovery register summary

Address offset

`0x08E6C`

Type

RO

Reset value

`TCUCFG_DATARAM_TYPE`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-26: TCU_SYSDISC14 register bit assignments

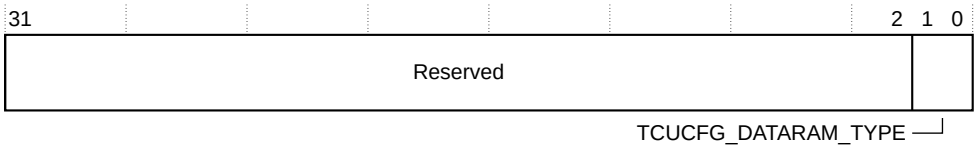


Table 7-40: TCU_SYSDISC14 register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved
[1:0]	TCUCFG_DATARAM_TYPE	The read data reflects the chosen parameter value, for example: • `0b00` : 0 • ... • `0b10` : 2

7.4.5.16 TCU_SYSDISC15

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

TCU system discovery register summary

Address offset

`0x08E70`

Type

RO

Reset value

`TCUCFG_SLOTRAM_TYPE`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-27: TCU_SYSDISC15 register bit assignments

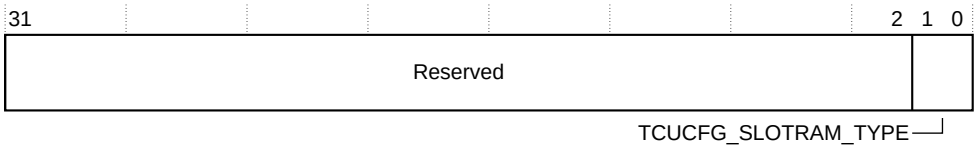


Table 7-41: TCU_SYSDISC15 register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved
[1:0]	TCUCFG_SLOTRAM_TYPE	The read data reflects the chosen parameter value, for example: • `0b00` : 0 • ... • `0b10` : 2

7.4.5.17 TCU_SYSDISC16

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

TCU system discovery register summary

Address offset

`0x08E74`

Type

RO

Reset value

`TCUCFG_CACHERAM_TYPE`. See Translation Control Unit buffer configuration parameters.

Bit descriptions

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

Figure 7-28: TCU_SYSDISC16 register bit assignments

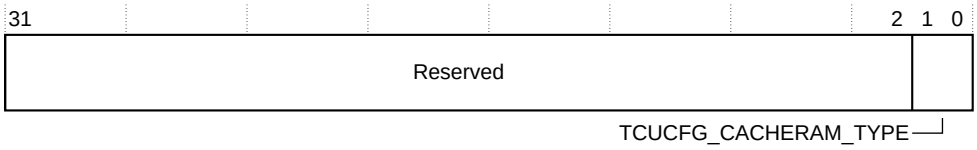


Table 7-42: TCU_SYSDISC16 register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved
[1:0]	TCUCFG_CACHERAM_TYPE	The read data reflects the chosen parameter value: • `0b00` : 0 • `0b01` : 1

7.4.5.18 TCU_SYSDISC17

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

TCU system discovery register summary

Address offset

`0x08E78`

Type

RO

Reset value

`TCUCFG_QTW_DATA_WIDTH`. See Translation Control Unit I/O configuration parameters.

Bit descriptions

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

Figure 7-29: TCU_SYSDISC17 register bit assignments

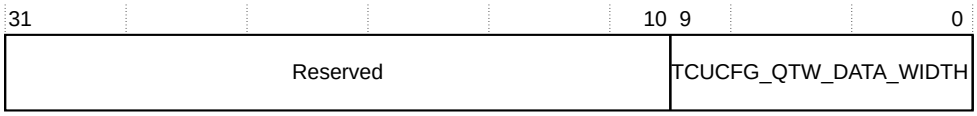


Table 7-43: TCU_SYSDISC17 register bit descriptions

Bits	Name	Description
[31:10]	-	Reserved
[9:0]	TCUCFG_QTW_DATA_WIDTH	The read data reflects the chosen parameter value, for example: • `0x040` : 64 • ... • `0x200` : 512

7.4.6 TCU PIU integration registers

MMU L1 also contains Programmer Interface Unit (PIU) integration registers.

There are two PIU integration registers:

- ITEN register for the TCU
- ITOP register for the TCU Programmer Interface Unit

7.4.6.1 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

TCU integration register summary

Address offset

`0x08E20`

Type

RW

Reset value

0

Bit descriptions

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

Figure 7-30: ITEN register bit assignments

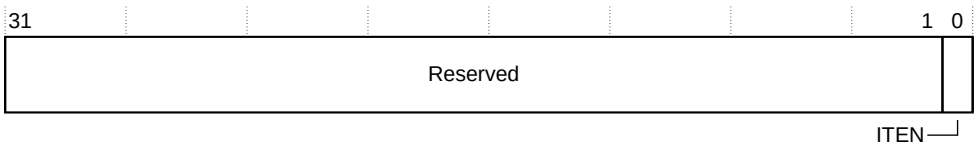


Table 7-44: ITEN register bit descriptions

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

7.4.6.2 ITOP register for the TCU Programmer Interface Unit

This register is 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

TCU integration register summary

Address offset

`0x08E24`

Type

RW

Reset value

0

Bit descriptions

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

Figure 7-31: ITOP register bit assignments

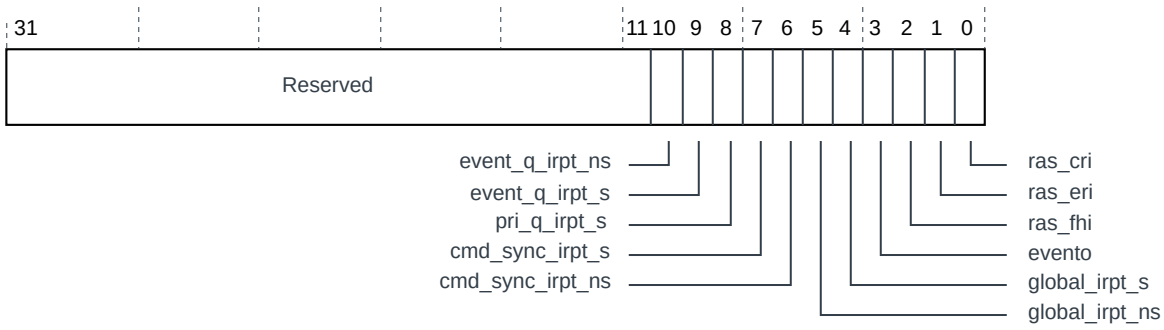


Table 7-45: ITOP register bit descriptions

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

Bits	Name	Description
[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
[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

7.4.7 TCU TMU integration registers

MMU L1 also contains TCU Translation Management Unit (TMU) integration registers.

- ITOP register for the TCU Translation Management Unit
- ITIN register for the TCU Translation Management Unit

7.4.7.1 ITOP register for the TCU Translation Management Unit

This register is 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

TCU integration register summary

Address offset

0x08E2C

Type

RW

Reset value

0

Bit descriptions

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

Figure 7-32: ITOP register bit assignments

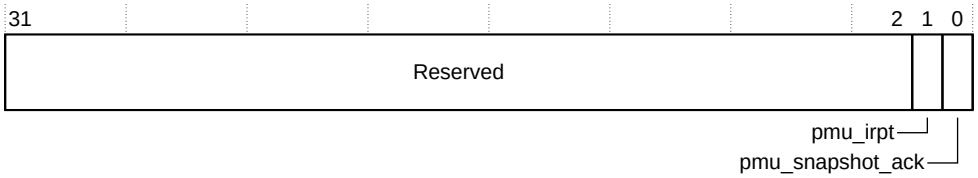


Table 7-46: ITOP register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved, Should-Be-Zero
[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.

7.4.7.2 ITIN register for the TCU Translation Management Unit

This register is 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

TCU integration register summary

Address offset

0x08E30

Type

RO

Reset value

0

Bit descriptions

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

Figure 7-33: ITIN register bit assignments

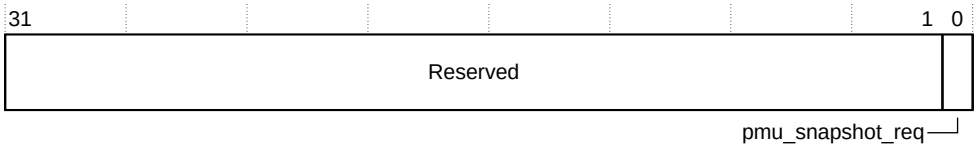


Table 7-47: ITIN register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved, Should-Be-Zero
[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.

7.5 TBU registers

The MMU L1 Translation Buffer Unit (TBU) registers comprise of several groups of registers.

The groups of registers are:

- **IMPLEMENTATION DEFINED** registers for controlling microarchitectural features
- RAS control registers
- Performance Monitor control registers
- Performance Monitor counter registers. This register group is on a separate 64KB page so a guest OS can page it for direct access.
- Memory System Resource Partitioning and Monitoring (MPAM) registers

The following table shows how the regions of the register map are allocated. Other regions are reserved.

Table 7-48: TBU register ranges

Address range	Usage
0x08E00 – 0x08E7C	Microarchitectural features for MMU L1
0x08E80 – 0x08EFC	TBU Reliability, Availability, and Serviceability registers
0x00FD0 – 0x00FFC	TBU identification registers
0x09000 - 0x09FFC	TBU System Discovery 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

7.5.1 Microarchitectural features for MMU L1

The microarchitectural registers are intended to be set at boot time to optimize the behavior of the TBU for specific systems. The recommended settings for most systems are the default settings.

The TBU_SCR register is Secure-only. Non-secure access to this register is **RAZ/WI**.

Non-secure access to TBU_CTRL register when TBU_SCR.NS_UARCH=0 is **RAZ/WI**.

The following table lists the TBU Microarchitectural control registers:

Table 7-49: TBU Microarchitectural control registers

Offset	Name	Type	Width	Description
0x08E00	TBU_CTRL	RW	32-bit	TBU_CTRL, TBU Control register
0x08E18	TBU_SCR	RW, Secure	32-bit	TBU_SCR, TBU Secure Control register
0x08E20	ITEN	RW	32-bit	ITEN, TBU Integration Enable register
0x08E24	ITOP_TBU	RW	32-bit	ITOP, TBU Integration Output register
0x08E28	ITIN_TBU	RO	32-bit	ITIN, TBU Integration Input register

7.5.1.1 TBU_CTRL, TBU Control register

The TBU_CTRL register disables MMU L1 TBU features. In most systems it can be left unchanged, but these controls can be used in some systems to work around issues.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

Microarchitectural features for MMU L1

Address offset

0x08E00

Type

RW

Reset value

0

Bit descriptions

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

Figure 7-34: TBU_CTRL register bit assignments

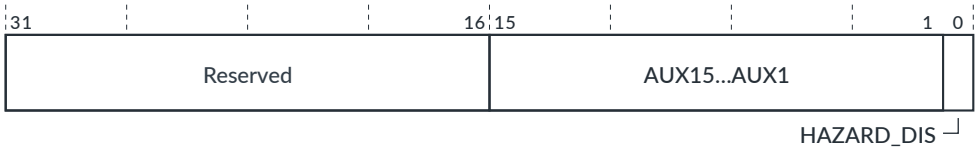


Table 7-50: TBU_CTRL register bit descriptions

Bits	Name	Description
[31:16]	-	Reserved
[15:1]	AUX15...AUX1	Reserved. If writing other bits of this register, ensure that you write the current value for those bits. For example, by performing a read-modify-write sequence.
[0]	HAZARD_DIS	Disables hazarding between translation requests from transactions in the same page. 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. Once you set this bit, only use a hard reset to clear this bit.

7.5.1.2 TBU_SCR, TBU Secure Control register

The TBU_SCR is the Secure control register for the MMU L1 TBU.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

Microarchitectural features for MMU L1

Address offset

`0x08E18`

Type

RW, Secure

Reset value

sec_override signal

Bit descriptions

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

Figure 7-35: TBU_SCR register bit assignments

Table 7-51: TBU_SCR register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved
[1]	NS_RAS	Non-secure register access permitted for RAS registers. When this bit is 0, Non-secure writes to the `0x08E80-0x08EC0` register addresses are ignored, and Non-secure reads return zero. The sec_override input signal is the reset value of this bit.
[0]	NS_UARCH	Non-secure register access permitted for microarchitectural registers. When this bit is 0, Non-secure writes to the `0x08E00-0x08E7C` register addresses are ignored, and Non-secure reads return zero. The sec_override input signal is the reset value of this bit. We recommend that software does not set this bit if Secure translation might be used.

7.5.2 TBU identification registers

The following table shows the identification (ID) registers.

Table 7-52: TBU ID registers

Address	Name	Field	Value	Meaning
`0x00FFC`	SMMU_CIDR3, Component ID3	[7:0]	`0xB1`	Preamble
`0x00FF8`	SMMU_CIDR2, Component ID2	[7:0]	`0x05`	Preamble

Address	Name	Field	Value	Meaning
0x00FF4	SMMU_CIDR1, Component ID1	[7:0]	0xF0	Preamble
0x00FF0	SMMU_CIDR0, Component ID0	[7:0]	0x0D	Preamble
0x00FEC	SMMU_PIDR3, Peripheral ID3	[7:4]	MAX(p_level, ecorevnum)	REVAND (Minor revision, where p_level is 2 for p2)
0x00FEC	SMMU_PIDR3, Peripheral ID3	[3:0]	0x00	CMOD
0x00FE8	SMMU_PIDR2, Peripheral ID2	[7:4]	0x00	REVISION (Major revision)
0x00FE8	SMMU_PIDR2, Peripheral ID2	[3]	1	JEDEC-assigned value for DES always used
0x00FE8	SMMU_PIDR2, Peripheral ID2	[2:0]	3	DES_1: bits [6:4] bits of the JEP106 Designer code
0x00FE4	SMMU_PIDR1, Peripheral ID1	[7:4]	0xB	DES_0: bits [3:0] of the JEP106 Designer code
0x00FE4	SMMU_PIDR1, Peripheral ID1	[3:0]	0x4	PART_1: bits [11:8] of the Part number
0x00FE0	SMMU_PIDR0, Peripheral ID0	[7:0]	0x89	PART_0: bits [7:0] of the Part number
0x00FDC	SMMU_PIDR7, Peripheral ID7	-	**RES0**	Reserved
0x00FD8	SMMU_PIDR6, Peripheral ID6	-	**RES0**	Reserved
0x00FD4	SMMU_PIDR5, Peripheral ID5	-	**RES0**	Reserved
0x00FD0	SMMU_PIDR4, Peripheral ID4	[7:4]	0x0	SIZE = 4KB
0x00FD0	SMMU_PIDR4, Peripheral ID4	[3:0]	0x4	DES_2: JEP106 Designer continuation code

7.5.2.1 ITEN, TBU Integration Enable register

This register is the enable integration mode register for the MMU L1 TBU. When you enable integration mode, the values of certain TBU input pins are visible in the ITIN register.

The values that MMU L1 writes to the TBU ITOP register control the values of certain TBU output pins. When you control these output pins system integrators can integrate the SMMU into the system and perform basic connectivity checks.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

Microarchitectural features for MMU L1

Address offset

0x08E20

Type

RW

Reset value

0

Bit descriptions

The following figure and table show the TBU ITEN register bit assignments and bit descriptions.

Figure 7-36: ITEN register bit assignments

Table 7-53: ITEN for TBU register bit descriptions

Bits	Name	Reset	Description
[31:1]	-	-	Reserved
[0]	ITEN	0	**0** Integration mode disabled **1** Integration mode enabled When integration mode is enabled, a power down request is denied because power down is not expected during integration testing.

7.5.2.2 ITOP, TBU Integration Output register

The TBU Integration Output (ITOP) register is the integration output register for the TBU.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

Microarchitectural features for MMU L1

Address offset

0x08E24

Type

RW

Reset value

0

Bit descriptions

The following figure and table show the TBU ITOP register bit assignments and bit descriptions.

Figure 7-37: ITOP register bit assignments

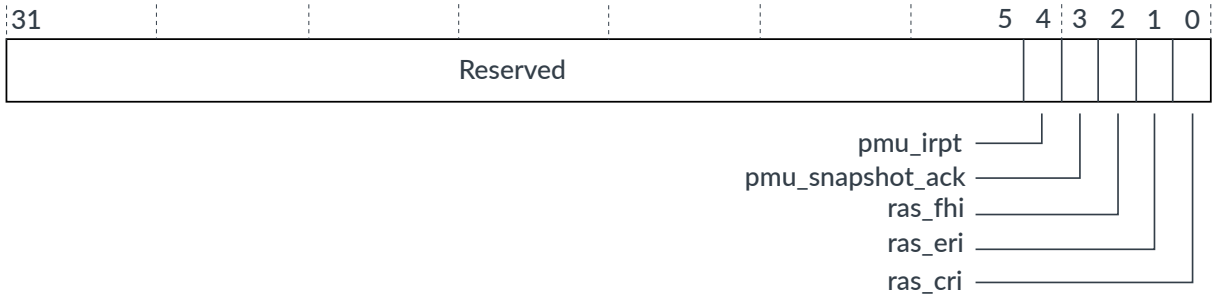


Table 7-54: ITOP for TBU register bit descriptions

Bits	Name	Reset	Description
[31:5]	-	-	Reserved, SBZ
[4]	pmu_irpt	0	When ITEN.ITEN == 0, SBZ. When ITEN.ITEN == 1, the value driven to pmu_irpt
[3]	pmu_snapshot_ack	0	When ITEN.ITEN == 0, SBZ. When ITEN.ITEN == 1, the value driven to pmu_snapshot_ack.
[2]	ras_fhi	0	When ITEN.ITEN == 0, SBZ. When ITEN.ITEN == 1, the value driven to ras_fhi
[1]	ras_eri	0	When ITEN.ITEN == 0, SBZ. When ITEN.ITEN == 1, the value driven to ras_eri
[0]	ras_cri	0	When ITEN.ITEN == 0, SBZ. When ITEN.ITEN == 1, the value driven to ras_cri

7.5.2.3 ITIN, TBU Integration Input register

The TBU Integration Input (ITIN) register is the input register for the MMU L1 TBU.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

Microarchitectural features for MMU L1

Address offset

0x08E28

Type

RO

Reset value

0

Bit descriptions

The following figure and table show the TBU ITIN register bit assignments and bit descriptions.

Figure 7-38: ITIN register bit assignments

Table 7-55: ITIN for TBU register bit descriptions

Bits	Name	Reset	Description
[31:1]	Reserved	-	SBZ
[0]	pmu_snapshot_req	0	When ITEN.ITEN == 0, SBZ. When ITEN.ITEN == 1, reflects pmu_snapshot_req

7.5.3 TBU Reliability, Availability, and Serviceability registers

The Reliability, Availability, and Serviceability (RAS) registers in the MMU L1 TBU implement the RAS Extension registers, Single record format.

The TBU RAS registers are 64 bits wide. The top 32 bits are Reserved and are omitted in this document for clarity.

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

The following table lists the RAS control registers.

Table 7-56: TBU RAS registers

Offset	Name	Type	Width	Description
0x08E80	TBU_ERRFR	S, RO	32-bit	TBU_ERRFR, TBU Error Feature register
0x08E88	TBU_ERRCTLR	S, RW	32-bit	TBU_ERRCTRL, TBU Error Control register
0x08E90	TBU_ERRSTATUS	S, RW	32-bit	TBU_ERRSTATUS, TBU Error Record Primary Syndrome register
0x08EC0	TBU_ERRGEN	S, RW	32-bit	TBU_ERRGEN, TBU Error Generation register

RAS error reporting

The TBU supports CE, and UE (UC) errors.

An interrupt is raised on the following:

ras_fhi

If TBU_ERRCTLR.FI is set, and a CE, or UE (UC) is recorded

ras_eri

If a UE (UC) is recorded

ras_cri

If a UE (UC) is recorded

7.5.3.1 TBU_ERRFR, TBU Error Feature register

The MMU L1 TBU_ERRFR register shows the TBU error features.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

TBU Reliability, Availability, and Serviceability registers

Address offset

0x08E80

Type

RO, Secure

Reset value

0x00400091

Bit descriptions

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

Figure 7-39: TBU_ERRFR register bit assignments

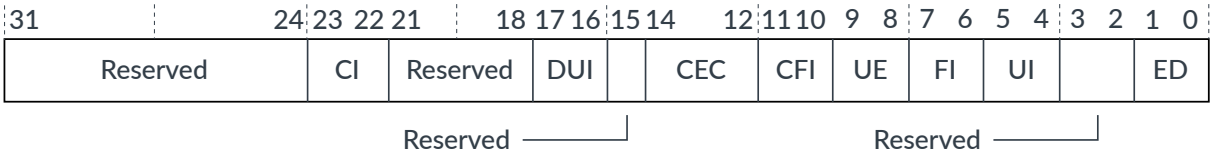


Table 7-57: TBU_ERRFR register bit descriptions

Bits	Name	Description
[31:24]	-	Reserved
[23:22]	CI	Critical Error Interrupt is always enabled, value is `0b01`
[21:18]	-	Reserved
[17:16]	DUI	Does not support feature, value is `0b00`
[15]	-	Reserved
[14:12]	CEC	Does not implement the standard corrected error counter model, value is `0b000`
[11:10]	CFI	Does not support feature, value is `0b00`
[9:8]	UE	In-band error signaling feature is not enabled, value is `0b00`
[7:6]	FI	Fault handling interrupt is controllable, value is `0b10`
[5:4]	UI	Error Recovery Interrupt always enabled for UE, value is `0b01`
[3:2]	-	Reserved
[1:0]	ED	Error detection is always enabled, value is `0b01`

7.5.3.2 TBU_ERRCTRL, TBU Error Control register

The MMU L1 TBU_ERRCTRL register enables fault handling interrupts.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

TBU Reliability, Availability, and Serviceability registers

Address offset

`0x08E88`

Type

RW, Secure

Reset value

0x00000008

Bit descriptions

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

Figure 7-40: TBU_ERRCTRL register bit assignments

Table 7-58: TBU_ERRCTRL register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved
[3]	FI	Fault handling interrupt enable. Reset is 1.
[2:0]	-	Reserved

7.5.3.3 TBU_ERRSTATUS, TBU Error Record Primary Syndrome register

Use the TBU Error Record Primary Syndrome register to find out whether different types of error have occurred on the MMU L1 TBU.

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

For information about how to clear this register, see Clearing ERRSTATUS.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

TBU Reliability, Availability, and Serviceability registers

Address offset

0x08E90

Type

RW, Secure

Reset value

0

Bit descriptions

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

Figure 7-41: TBU_ERRSTATUS register bit assignments

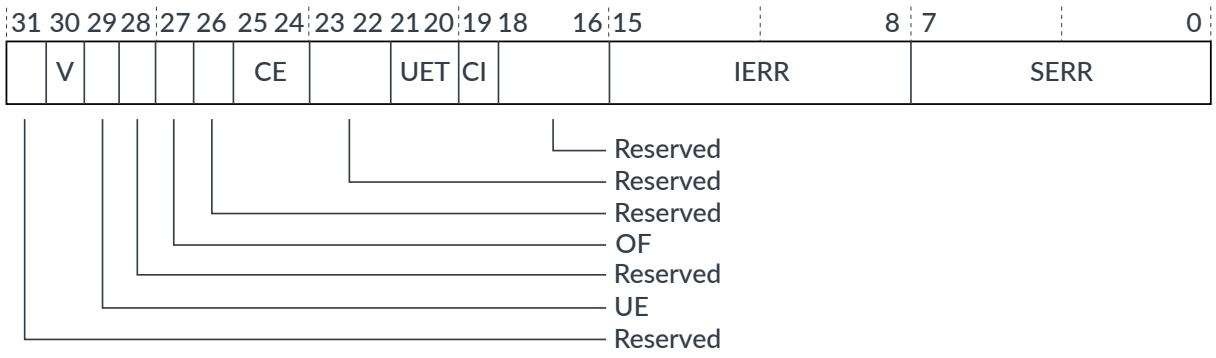


Table 7-59: TBU_ERRSTATUS register bit descriptions

Bits	Name	Description
[31]	-	Reserved
[30]	V	Status Register valid. The possible values of this bit are: **0** ERRSTATUS not valid. **1** ERRSTATUS valid. At least one error has been recorded. This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields, see Clearing ERRSTATUS. This bit resets to zero on a reset. Reset is 0.
[29]	UE	Uncorrected error(s). The possible values of this bit are: **0** No errors that could neither be corrected nor deferred. **1** At least one error detected that has neither been corrected nor deferred. This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields, see Clearing ERRSTATUS. Reset is 0.

Bits	Name	Description
[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 Clearing ERRSTATUS. Reset is 0.
[26]	-	Reserved
[25:24]	CE	Correctable Error(s). **0b00** No correctable errors recorded. **0b10** At least one Corrected error recorded. Other values are Reserved. This field is read/write `0b11` to clear. Clearing depends on other ERRSTATUS fields, see Clearing ERRSTATUS. Reset is `0b00`.
[23:22]	-	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: **0b00** Uncorrected error, Uncontainable error (UC). Writes to this field are ignored. Reset is `0b00`.
[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. Reset is `0`.
[18:16]	-	Reserved

Bits	Name	Description
[15:8]	IERR	Implementation defined error code. This field indicates the source of the error: **0x15-0x05** Reserved **0x04** TLB MTLB DATA **0x03** TLB MTLB TAGS **0x02** TLB MTLB REPL **0x01** TLB MTLB PCNT **0x00** TLB MTLB PLIM Writes to this field are ignored. Reset is `0x00`.
[7:0]	SERR	Error code. This gives 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/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. Reset is `0x00`.

7.5.3.3.1 Clearing ERRSTATUS

Software can clear the TBU_ERRSTATUS, TBU Error Record Primary Syndrome register by writing one(s) to fields which are set.

A write to the TBU_ERRSTATUS is ignored if all of the following is true:

- Any of the V, UE, OF, CE, or DE fields are nonzero before the write.
- The write does not clear the nonzero V, UE, OF, CE, or DE fields to zero by writing ones to the applicable field or fields. CE must be cleared by writing `0b11` to the field.

If a valid clearing write reaches TBU_ERRSTATUS on the same cycle as a new error, the new record is applied as if no previous error existed (TBU_ERRSTATUS.V = 0).

7.5.3.4 TBU_ERRGEN, TBU Error Generation register

The MMU L1 TBU_ERRGEN register enables you to test software for when a RAS error occurs.

The field locations are same as the TBU_ERRSTATUS.

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

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

- TBU_ERRGEN.V is set
- At least one of the following is true, CE is legal if CE == `0b00` or CE == `0b10`:
  - TBU_ERRGEN.UE is set and CE is legal
  - TBU_ERRGEN.DE is set and CE is legal, DE is not supported in the MMU L1 TBU
  - TBU_ERRGEN.CE is set to `0b10`
- UET must be `0b00`

All other fields can take any legal value. Any other write is considered invalid and the SMMU behavior is **UNPREDICTABLE**. If there is a valid error record write, then MMU L1 generates the appropriate interrupt or interrupts.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

TBU Reliability, Availability, and Serviceability registers

Address offset

`0x08EC0`

Type

RW, Secure

Reset value

0

Bit descriptions

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

Figure 7-42: TBU_ERRGEN register bit assignments

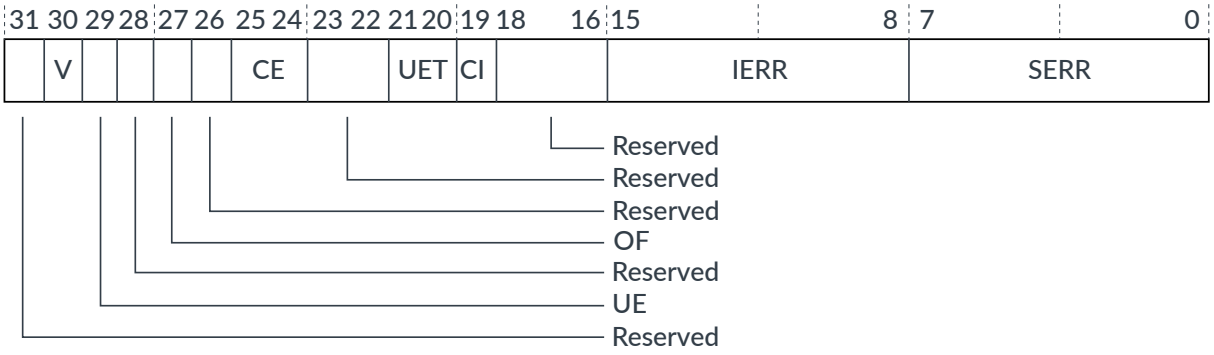


Table 7-60: TBU_ERRGEN register bit descriptions

Bits	Name	Description
[31]	-	Reserved
[30]	V	Status Register valid. The possible values of this bit are: **0** ERRSTATUS not valid. **1** ERRSTATUS valid. At least 1 error has been recorded. This field is read/write-one-to-clear. Clearing depends on other TBU_ERRSTATUS register fields, see Clearing ERRSTATUS. This bit resets to zero on a reset. Reset is 0.
[29]	UE	Uncorrected error(s). The possible values of this bit are: **0** No errors that could neither be corrected nor deferred. **1** At least 1 error detected that has neither been corrected nor deferred. This field is read/write-one-to-clear. Clearing depends on other TBU_ERRSTATUS register fields, see Clearing ERRSTATUS. Reset is 0.
[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 TBU_ERRSTATUS register fields, see Clearing ERRSTATUS. Reset is 0.
[26]	-	Reserved
[25:24]	CE	Correctable Error(s). **0b00** No correctable errors recorded. **0b10** At least one Corrected error recorded. Other values are Reserved. This field is read/write `0b11` to clear. Clearing depends on other TBU_ERRSTATUS register fields, see Clearing ERRSTATUS. Reset is `0b00`.
[23:22]	-	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: **0b00** Uncorrected error, Uncontainable error (UC). Writes to this field are ignored. Reset is `0b00`.
[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. Reset is 0.
[18:16]	-	Reserved

Bits	Name	Description
[15:8]	IERR	Implementation defined error code. This field indicates the source of the error: **0x15-0x05** Reserved **0x04** TLB MTLB DATA **0x03** TLB MTLB TAGS **0x02** TLB MTLB REPL **0x01** TLB MTLB PCNT **0x00** TLB MTLB PLIM Writes to this field are ignored. Reset is `0x00`.
[7:0]	SERR	Error code. This gives 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/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. Reset is `0x00`.

7.5.4 TBU System Discovery registers

The System Discovery (SYSDISC) registers contain important hardware configuration-related information that might be relevant for software. Each register contains a single parameter that is located at the lowest bits of the registers. Gaps in this address range are **RAZ/WI**.

The following table shows the TBU System Discovery registers.

Table 7-61: TBU System Discovery registers

Address	Name	Size	Access	Description
`0x09000`	TBU_SYSDISC0	32	S, RO	TBUCFG_MTLB_DEPTH
`0x09004`	TBU_SYSDISC1	32	S, RO	TBUCFG_UTLB_DEPTH
`0x09008`	-	32	S, RO	Reserved
`0x0900C`	-	32	S, RO	Reserved

Address	Name	Size	Access	Description
0x09010	TBU_SYSDISC4	32	S, RO	TBUCFG_XLATE_SLOTS
0x09014	TBU_SYSDISC5	32	S, RO	TBUCFG_PMU_COUNTERS
0x09018	TBU_SYSDISC6	32	S, RO	TBUCFG_SID_WIDTH
0x0901C	TBU_SYSDISC7	32	S, RO	TBUCFG_SSID_WIDTH
0x09020	TBU_SYSDISC8	32	S, RO	TBUCFG_DIRECT_IDX
0x09024	TBU_SYSDISC9	32	S, RO	TBUCFG_MTLB_PARTS
0x09028	-	32	S, RO	Reserved
0x0902C	TBU_SYSDISC11	32	S, RO	TBUCFG_PARTID_WIDTH
0x09030	-	32	S, RO	Reserved
0x09034	-	32	S, RO	Reserved
0x09038	TBU_SYSDISC14	32	S, RO	TBUCFG_CACHERAM_TYPE
0x0903C	TBU_SYSDISC15	32	S, RO	TBUCFG_MTLB_LKP_SLOTS
0x09040	-	32	S, RO	Reserved
0x09044	-	32	S, RO	Reserved
0x09048	-	32	S, RO	Reserved
0x0904C	TBU_SYSDISC19	32	S, RO	TBUCFG_USE_ELA_DEBUG
0x09050	TBU_SYSDISC20	32	S, RO	TBUCFG_STASH_SUPPORT
0x09054	TBU_SYSDISC21	32	S, RO	TBUCFG_TLB_RAS_SUPPORT

7.5.4.1 TBU_SYSDISC0

The TBU system discovery registers discover components in MMU L1.

Configurations

This register is available in all configurations.

Attributes

Its characteristics are:

Width

32-bit

Address offset

0x09000

Type

RO

Reset value

TBUCFG_MTLB_DEPTH. For more information on the reset value, see Translation Buffer Unit buffer configuration parameters.

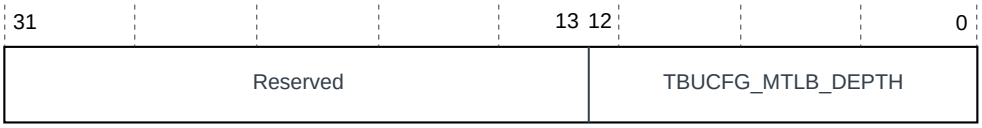
Constraints

None.

Bit descriptions

The following figure shows the TBU_SYSDISC0 register bit assignments.

Figure 7-43: TBU_SYSDISC0 register bit assignments



The following table shows the TBU_SYSDISC0 register bit descriptions.

Table 7-62: TBU_SYSDISC0 register bit descriptions

Bits	Name	Description
[31:13]	Reserved	-
[12:0]	TBUCFG_MTLB_DEPTH	The read data reflects the chosen parameter value, for example: • `0x0000` : 0 • ... • `0x1000` : 4096

7.5.4.2 TBU_SYSDISC1

The TBU system discovery registers discover components in MMU L1.

Configurations

This register is available in all configurations.

Attributes

Its characteristics are:

Width

32-bit

Address offset

`0x09004`

Type

RO

Reset value

`TBUCFG_UTLB_DEPTH`. For more information on the reset value, see Translation Buffer Unit buffer configuration parameters.

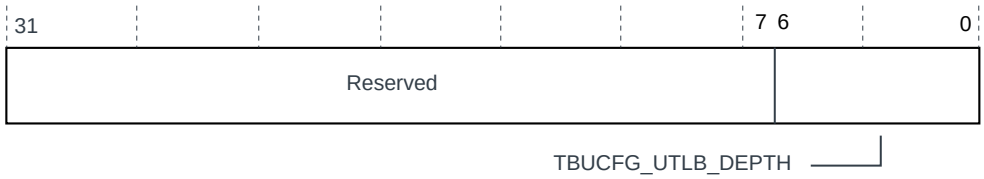
Constraints

None.

Bit descriptions

The following figure shows the TBU_SYSDISC1 register bit assignments.

Figure 7-44: TBU_SYSDISC1 register bit assignments



The following table shows the TBU_SYSDISC1 register bit descriptions.

Table 7-63: 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: • `0x04` : 4 • ... • `0x40` : 64

7.5.4.3 TBU_SYSDISC4

The TBU system discovery registers discover components in MMU L1.

Configurations

This register is available in all configurations.

Attributes

Its characteristics are:

Width

32-bit

Address offset

`0x09010`

Type

RO

Reset value

`TBUCFG_XLATE_SLOTS`. For more information on the reset value, see Translation Buffer Unit buffer configuration parameters.

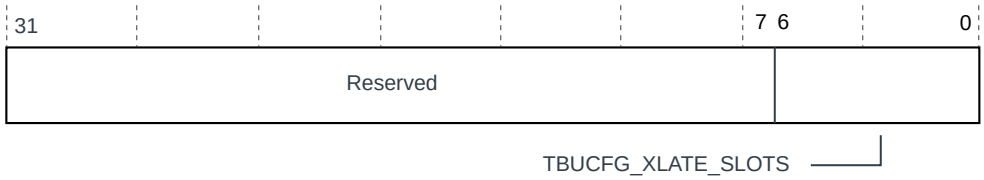
Constraints

None.

Bit descriptions

The following figure shows the TBU_SYSDISC4 register bit assignments.

Figure 7-45: TBU_SYSDISC4 register bit assignments



The following table shows the TBU_SYSDISC4 register bit descriptions.

Table 7-64: TBU_SYSDISC4 register bit descriptions

Bits	Name	Description
[31:7]	Reserved	-
[6:0]	TBUCFG_XLATE_SLOTS	The read data reflects the chosen parameter value, for example: • `0x02` : 2 • ... • `0x40` : 64

7.5.4.4 TBU_SYSDISC5

The TBU system discovery registers discover components in MMU L1.

Configurations

This register is available in all configurations.

Attributes

Its characteristics are:

Width

32-bit

Address offset

`0x09014`

Type

RO

Reset value

`TBUCFG_PMU_COUNTERS`. For more information on the reset value, see Translation Buffer Unit buffer configuration parameters.

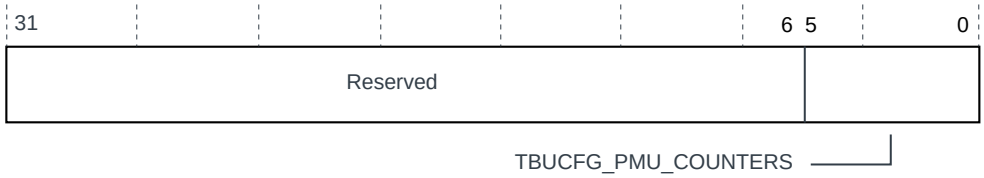
Constraints

None.

Bit descriptions

The following figure shows the TBU_SYSDISC5 register bit assignments.

Figure 7-46: TBU_SYSDISC5 register bit assignments



The following table shows the TBU_SYSDISC5 register bit descriptions.

Table 7-65: 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: • `0x04` : 4 • ... • `0x20` : 32

7.5.4.5 TBU_SYSDISC6

The TBU system discovery registers discover components in MMU L1.

Configurations

This register is available in all configurations.

Attributes

Its characteristics are:

Width

32-bit

Address offset

`0x09018`

Type

RO

Reset value

`TBUCFG_SID_WIDTH`. For more information on the reset value, see Translation Buffer Unit configuration parameters.

Constraints

None.

Bit descriptions

The following figure shows the TBU_SYSDISC6 register bit assignments.

Figure 7-47: TBU_SYSDISC6 register bit assignments

The following table shows the TBU_SYSDISC6 register bit descriptions.

Table 7-66: 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: • `0x08` : 8 • ... • `0x18` : 24

7.5.4.6 TBU_SYSDISC7

The TBU system discovery registers discover components in MMU L1.

Configurations

This register is available in all configurations.

Attributes

Its characteristics are:

Width

32-bit

Address offset

0x0901C

Type

RO

Reset value

TBUCFG_SSID_WIDTH. For more information on the reset value, see Translation Buffer Unit configuration parameters.

Constraints

None.

Bit descriptions

The following figure shows the TBU_SYSDISC7 register bit assignments.

Figure 7-48: TBU_SYSDISC7 register bit assignments

The following table shows the TBU_SYSDISC7 register bit descriptions.

Table 7-67: 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: • 0x01 : 1 • ... • 0x14 : 20

7.5.4.7 TBU_SYSDISC8

The TBU system discovery registers discover components in MMU L1.

Configurations

This register is available in all configurations.

Attributes

Its characteristics are:

Width

32-bit

Address offset

0x09020

Type

RO

Reset value

TBUCFG_DIRECT_IDX. For more information on the reset value, see Translation Buffer Unit configuration parameters.

Constraints

None.

Bit descriptions

The following figure shows the TBU_SYSDISC8 register bit assignments.

Figure 7-49: TBU_SYSDISC8 register bit assignments

The following table shows the TBU_SYSDISC8 register bit descriptions.

Table 7-68: TBU_SYSDISC8 register bit descriptions

Bits	Name	Description
[31:1]	Reserved	-
[0]	TBUCFG_DIRECT_IDX	The read data reflects the chosen parameter value: • 0b0 : 0 • 0b1 : 1

7.5.4.8 TBU_SYSDISC9

The TBU system discovery registers discover components in MMU L1.

Configurations

This register is available in all configurations.

Attributes

Its characteristics are:

Width

32-bit

Address offset

`0x09024`

Type

RO

Reset value

`TBUCFG_MTLB_PARTS`. For more information on the reset value, see Translation Buffer Unit configuration parameters.

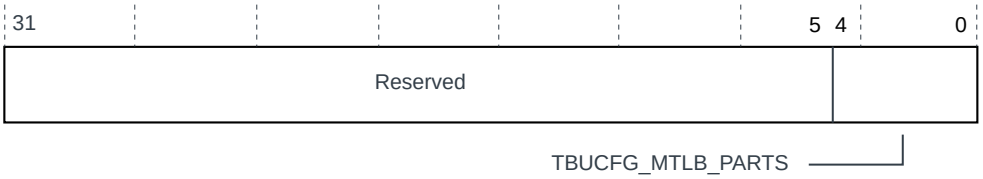
Constraints

None.

Bit descriptions

The following figure shows the TBU_SYSDISC9 register bit assignments.

Figure 7-50: TBU_SYSDISC9 register bit assignments



The following table shows the TBU_SYSDISC9 register bit descriptions.

Table 7-69: 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: • `0x01` : 1 • ... • `0x10` : 16

7.5.4.9 TBU_SYSDISC11

The TBU system discovery registers discover components in MMU L1.

Configurations

This register is available in all configurations.

Attributes

Its characteristics are:

Width

32-bit

Address offset

`0x0902C`

Type

RO

Reset value

`TBUCFG_PARTID_WIDTH`. For more information on the reset value, see Translation Buffer Unit buffer configuration parameters.

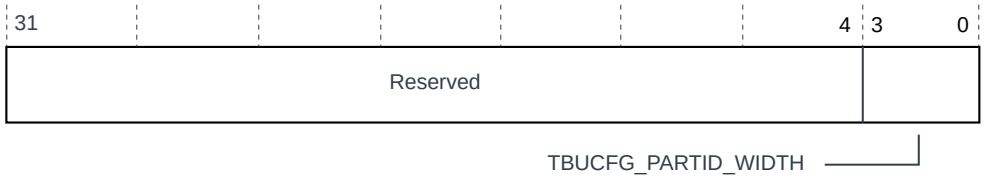
Constraints

None.

Bit descriptions

The following figure shows the TBU_SYSDISC11 register bit assignments.

Figure 7-51: TBU_SYSDISC11 register bit assignments



The following table shows the TBU_SYSDISC11 register bit descriptions.

Table 7-70: 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: • `0x1` : 1 • ... • `0x9` : 9

7.5.4.10 TBU_SYSDISC14

The TBU system discovery registers discover components in MMU L1.

Configurations

This register is available in all configurations.

Attributes

Its characteristics are:

Width

32-bit

Address offset

0x09038

Type

RO

Reset value

TBUCFG_CACHERAM_TYPE. For more information on the reset value, see Translation Buffer Unit buffer configuration parameters.

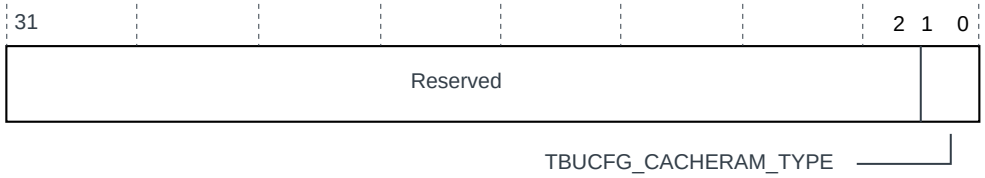
Constraints

None.

Bit descriptions

The following figure shows the TBU_SYSDISC14 register bit assignments.

Figure 7-52: TBU_SYSDISC14 register bit assignments



The following table shows the TBU_SYSDISC14 register bit descriptions.

Table 7-71: TBU_SYSDISC14 register bit descriptions

Bits	Name	Description
[31:2]	Reserved	-

Bits	Name	Description
[1:0]	TBUCFG_CACHERAM_TYPE	The read data reflects the chosen parameter value, for example: • `0b00` : 0 • ... • `0b01` : 1

7.5.4.11 TBU_SYSDISC15

The TBU system discovery registers discover components in MMU L1.

Configurations

This register is available in all configurations.

Attributes

Its characteristics are:

Width

32-bit

Address offset

`0x0903C`

Type

RO

Reset value

`TBUCFG_MTLB_LKP_SLOTS`. For more information on the reset value, see Translation Buffer Unit buffer configuration parameters.

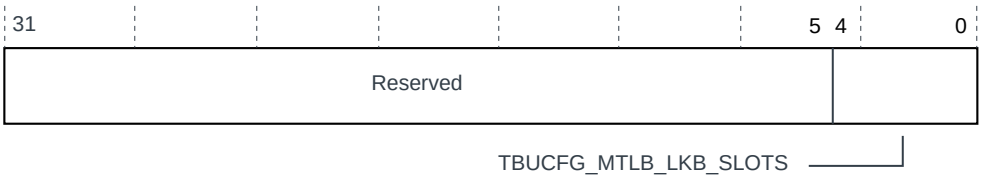
Constraints

None.

Bit descriptions

The following figure shows the TBU_SYSDISC15 register bit assignments.

Figure 7-53: TBU_SYSDISC15 register bit assignments



The following table shows the TBU_SYSDISC15 register bit descriptions.

Table 7-72: TBU_SYSDISC15 register bit descriptions

Bits	Name	Description
[31:5]	Reserved	-
[4:0]	TBUCFG_MTLB_LKP_SLOTS	The read data reflects the chosen parameter value, for example: • `0x02` : 2 • ... • `0x1C` : 28

7.5.4.12 TBU_SYSDISC19

The TBU system discovery registers discover components in MMU L1.

Configurations

This register is available in all configurations.

Attributes

Its characteristics are:

Width

32-bit

Address offset

`0x0904C`

Type

RO

Reset value

`TBUCFG_USE_ELA_DEBUG`. For more information on the reset value, see Translation Buffer Unit debug configuration parameters

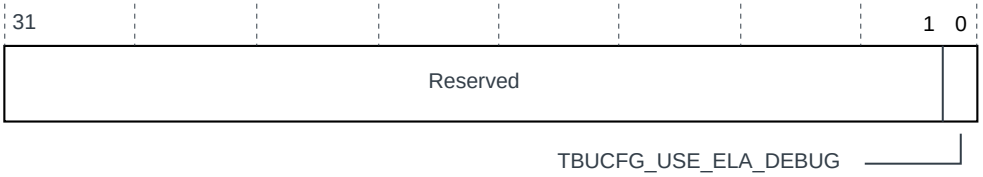
Constraints

None.

Bit descriptions

The following figure shows the TBU_SYSDISC19 register bit assignments.

Figure 7-54: TBU_SYSDISC19 register bit assignments



The following table shows the TBU_SYSDISC19 register bit descriptions.

Table 7-73: TBU_SYSDISC19 register bit descriptions

Bits	Name	Description
[31:1]	Reserved	-
[0]	TBUCFG_USE_ELA_DEBUG	The read data reflects the chosen parameter value: • 0b0 : 0 • 0b1 : 1

7.5.4.13 TBU_SYSDISC20

The TBU system discovery registers discover components in MMU L1.

Configurations

This register is available in all configurations.

Attributes

Its characteristics are:

Width

32-bit

Address offset

0x09050

Type

RO

Reset value

TBUCFG_STASH_SUPPORT. For more information on the reset value, see Translation Buffer Unit I/O configuration parameters.

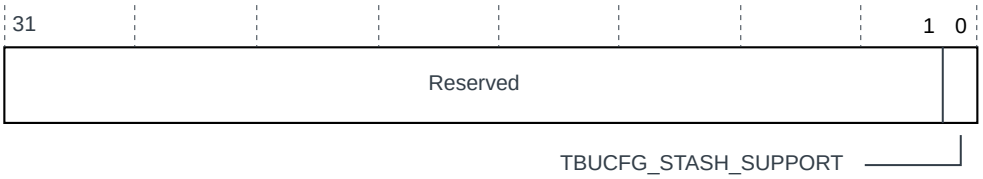
Constraints

None.

Bit descriptions

The following figure shows the TBU_SYSDISC20 register bit assignments.

Figure 7-55: TBU_SYSDISC20 register bit assignments



The following table shows the TBU_SYSDISC20 register bit descriptions.

Table 7-74: TBU_SYSDISC20 register bit descriptions

Bits	Name	Description
[31:1]	Reserved	-
[0]	TBUCFG_STASH_SUPPORT	The read data reflects the chosen parameter value: • `0b0` : 0 • `0b1` : 1

7.5.4.14 TBU_SYSDISC21

The TBU system discovery registers discover components in MMU L1.

Configurations

This register is available in all configurations.

Attributes

Its characteristics are:

Width

32-bit

Address offset

`0x09054`

Type

RO

Reset value

`TBUCFG_TLB_RAS_SUPPORT`. For more information on the reset value, see Translation Buffer Unit debug configuration parameters.

Constraints

None.

Bit descriptions

The following figure shows the TBU_SYSDISC21 register bit assignments.

Figure 7-56: TBU_SYSDISC21 register bit assignments

The following table shows the TBU_SYSDISC21 register bit descriptions.

Table 7-75: TBU_SYSDISC21 register bit descriptions

Bits	Name	Description
[31:1]	Reserved	-
[0]	TBUCFG_TLB_RAS_SUPPORT	The read data reflects the chosen parameter value: • 0b0 : 0 • 0b1 : 1

7.5.5 TBU performance monitor unit registers

The TBU Performance Monitor Unit (PMU) registers follow the register layout that the Performance Monitor Extension section describes.

For more information, see the Performance Monitor Extension section of the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2*.

The TCU and TBU support the same PMCG registers. SMMU_PMCG_SMR0 is 24 bits in size, because the TBU uses 24-bit StreamIDs architecturally, even though a static tie-off sets 8 or 16 bits.

7.5.6 TBU performance monitor events

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

For information on **IMPLEMENTATION DEFINED** events, see MMU L1 TBU events. For information on architectural events that are implemented, see SMMUv3 architectural performance events.

7.5.6.1 SMMU_PMCG_CFGR fields

The MMU L1 uses SMMU_PMCG_CFGR fields.

The following table shows the values that the SMMU_PMCG_CFGR fields contain.

Table 7-76: SMMU_PMCG_CFGR register bits

Field	Default value	Description for default value
FILTER_PARTID_PMG	0	This Performance Monitor Counter Group (PMCG) cannot filter events by PARTID or PMG. This is only present in SMMU v3.3, see *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2*.
MPAM	0	Memory System Resource Partitioning and Monitoring (MPAM) is not supported by the PMCG.

Field	Default value	Description for default value
SID_FILTER_TYPE	1	A single StreamID filter applies to all PMCG counters.
CAPTURE	1	The 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.
SIZE	31	The counter group implements 32-bit counters.
NCTR	`TBUCFG_PMU_COUNTERS` – 1	The counter group includes `TBUCFG_PMU_COUNTERS` counters.

7.5.6.2 TBU SMMU_PMCG_CEID{0-1} registers

The SMMU_PMCG_CEID{0-1} registers indicate the architectural events that a TBU supports. 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 TBU SMMU_PMCG_CEID{0-1} registers.

Table 7-77: TBU SMMU_PMCG_CEID register values

Name	Offset	Value
SMMU_PMCG_CEID0	`0x02E20`	`0x00000087`
SMMU_PMCG_CEID1	`0x02E28`	`0x00000000`

7.5.6.3 SMMU_PMCG_IIDR and SMMU_PMCG_AIDR registers

MMU L1 contains SMMU_PMCG_IIDR and SMMU_PMCG_AIDR registers.

The PMU Implementer and Architecture registers contain all the fields 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 SMMU_PMCG_AIDR and IIDR register values.

Table 7-78: SMMU_PMCG_AIDR and IIDR register values

Name	Offset	Value
SMMU_PMCG_IIDR	`0x02E08`	`0x4890043B`
SMMU_PMCG_AIDR	`0x02E70`	`0x00000002`

7.5.6.4 TBU 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 register values.

Table 7-79: TBU PMU ID register values

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
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 2 for p2
SMMU_PMCG_PIDR3, Peripheral ID3	-	[3:0]	0x00	CMOD
SMMU_PMCG_PIDR2, Peripheral ID2	0x02FE8	[7:4]	0x00	REVISION, major revision
SMMU_PMCG_PIDR2, Peripheral ID2	-	[3]	1	JEDEC-assigned value for DES always used
SMMU_PMCG_PIDR2, Peripheral ID2	-	[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
SMMU_PMCG_PIDR1, Peripheral ID1	-	[3:0]	0x4	PART_1: bits [11:8] of the Part number
SMMU_PMCG_PIDR0, Peripheral ID0	0x02FE0	[7:0]	0x89	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
SMMU_PMCG_PIDR4, Peripheral ID4	-	[3:0]	0x4	DES_2: JEP106 Designer continuation code
SMMU_PMCG_PMAUTHSTATUS	0x02FB8	[7:0]	0x00	No authentication interface is implemented

Appendix A TCU signal descriptions

The Translation Control Unit (TCU) in MMU L1 uses signals that can be either inputs or outputs.

MMU L1 external signal names use the following convention:

_s
Secure signal, for example event_q_irpt_s

_ns
Non-secure signal, for example event_q_irpt_ns

For more information about Secure and Non-secure signals, see the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2*.

A.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.2 TCU QTW and DVM interface signals

The TCU Queue and Table Walk (QTW) and Distributed Virtual Memory (DVM) interface signals are based on the AMBA® AXI5 signals.

Signal definitions

Table A-2: TCU QTW and 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;

Signal	Direction	Description
araddr_qtw	Output	Read address. Width is 52-bit.
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. The width of this signal is always 11 bits, but bits[9:1] that define the `PARTID` are affected by the `TCUCFG_PARTID_WIDTH` parameter. When `TCUCFG_PARTID_WIDTH` is less than 9, the non-used bits are set to 0.
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;
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. The width of this signal is always 11 bits, but bits[9:1] that define the PARTID are affected by the `TCUCFG_PARTID_WIDTH` parameter. When `TCUCFG_PARTID_WIDTH` is less than 9, the non-used bits are set to 0.
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.
rid_qtw	Input	Read data ID. Width is QTW_ID_WIDTH-bit. QTW_ID_WIDTH is calculated as follows: • ((log2(TCUCFG_PTW_SLOTS) + 2) > 4) ? (log2(TCUCFG_PTW_SLOTS) + 2) : 4;
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.
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.
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: • ((log2(TCUCFG_PTW_SLOTS) + 2) > 4) ? (log2(TCUCFG_PTW_SLOTS) + 2) : 4;
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.

A.3 TCU programming interface signals

The TCU programming interface signals are based on the AMBA® APB5 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 Translation Control Unit buffer configuration parameters.

Signal	Direction	Description
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 1-bit.
pwdata_prog	Input	Write data. Width is 1-bit.
pstrb_prog	Input	Write data strobe. Width is 1-bit.
pslverr_prog	Output	Error response. Width is 1-bit.
prdata_prog	Output	Read data. Width is 1-bit.
pready_prog	Output	Transfer ready. Width is 1-bit.
pwakeup_prog	Input	Interface wakeup. Width is 1-bit.

A.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. Width is 1-bit. This output transitions: **HIGH** To indicate that the requester is requesting to enter the coherency domain. **LOW** To indicate that the requester is requesting to exit the coherency domain.
syscoack_qtw	Input	System coherency acknowledge. Width is 1-bit. This input transitions to the same level as syscoreq_qtw when the request to enter or exit the coherency domain is complete.

A.5 TCU PMU snapshot interface signals

The following table shows the TCU PMU snapshot interface signals. This interface is asynchronous.

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. Width is 1-bit. Connect to the debug infrastructure of your SoC.

Signal	Direction	Description
pmusnapshot_ack	Output	PMU snapshot acknowledge. The TCU uses this signal to acknowledge that the PMU snapshot has occurred. Width is 1-bit. This signal is LOW after reset. Connect to the debug infrastructure of your SoC.

A.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.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.8 TCU DTI interface signals

The TCU DTI interface enables the TCU to respond to translation requests from the TBU. This interface uses the DTI-TBU protocol for communication between the TCU and the TBU.

The TCU includes a completer DTI interface and each TBU includes a requester DTI interface. To permit bidirectional communication, each DTI interface includes one AXI5-Stream transmitter interface and one AXI5-Stream receiver interface.

Signal definitions

Table A-8: TCU DTI interface signals

Signal	Direction	Description
tvalid_dti_dn	Input	Flow control signal. Width is 1-bit.
tready_dti_dn	Output	Flow control signal. Width is 1-bit.
tdata_dti_dn	Input	Message data signal. Width is 160-bit.
tid_dti_dn	Input	Identifies the requester that initiated the message. The width is 4-bit or 6-bit, and is calculated as follows: • If `TCUCFG_NUM_TBU` is 62, the width of tid_dti_dn is 6-bit. • Otherwise, the width of tid_dti_dn is 4-bit.
tlast_dti_dn	Input	Indicates the last cycle of a message. Width is 1-bit.
tkeep_dti_dn	Input	Indicates valid bytes. Width is 20-bit.
tvalid_dti_up	Output	Flow control signal. Width is 1-bit.
tready_dti_up	Input	Flow control signal. Width is 1-bit.
tdata_dti_up	Output	Message data signal. Width is 160-bit.
tdest_dti_up	Output	Identifies the requester that is receiving the message. The width is 4-bit or 6-bit, and is calculated as follows: • 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 Translation Control Unit buffer configuration parameters.
tlast_dti_up	Output	Indicates the last cycle of a message. Width is 1-bit.
tkeep_dti_up	Output	Indicates valid bytes. Width is 20-bit.
twakeup_dti_up	Output	Wakeup signal. Width is 1-bit.
twakeup_dti_dn	Input	Wakeup signal. Width is 1-bit.

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

For more information on the DTI-TBU protocol and the DTI-ATS protocol messages, see the *AMBA® DTI Protocol Specification*.

A.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 or DVM interface and the dedicated MSI delivery interface:

- Secure and Non-secure Event queue
- SYNC complete commands
- Global interrupts
- Page Request Interface (PRI) queue interrupt

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.
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. Width is 1-bit. The TCU_ERRCTLR.FI can also enable or disable ras_fhi. MMU L1 cannot output this interrupt as an MSI.
ras_eri	Output	Error recovery RAS interrupt for an uncontained error. Width is 1-bit. MMU L1 cannot output this interrupt as an MSI.
ras_cri	Output	Critical error interrupt for an uncontainable uncorrected error. Width is 1-bit. MMU L1 cannot output this interrupt as an MSI.
pmu_irpt	Output	The pmu_irpt signal asserts a PMU interrupt. Width is 1-bit. MMU L1 cannot output this interrupt as an MSI.
pri_q_irpt_ns	Output	Asserts a Page Request Interface (PRI) queue interrupt. Width is 1-bit.

A.10 TCU Message Signaled Interrupt interface signals

The TCU Message Signaled Interrupt (MSI) interface uses the signals in the following table to send MSIs.

The MSI follows the *AMBA® AXI-Stream Protocol Specification* protocol (with Wakeup_Signal enabled and Check_Type not enabled).

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

Signal	Direction	Description	Connection information
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

A.11 TCU event interface signals

The TCU event interface signal is an event output to connect to processors.

Signal definitions

Table A-11: TCU event interface signals

Signal	Direction	Description
evento	Output	The evento signal is asserted for 1 cycle to indicate an event that enables processors to wake up from the Wait For Event (WFE) low-power state. Width is 1-bit. 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. For example, some processors have an eventi input to connect directly to the evento output from the MMU. Other processors, including DSU-based systems, have a request and acknowledgement handshake mechanism that requires the evento signal from the MMU to be converted and uses the eventiack, eventireq, eventoack, and eventoreq signals.

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 L1 generates, which uses the older event mechanism.

In both mechanisms, in the signal names *i* represents events that are inputs to a particular component and *o* represents events that are outputs from a particular component.

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.

MMU L1 has an event output and therefore only has the evento signal. The processor has an input interface to receive the event from the MMU L1, 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 an MMU L1 to a DSU, the only signals to consider are the MMU L1 evento signal and the DSU eventiack and eventireq signals. Some processors have an eventi input instead.

You can use the SoC-600 Event Pulse to Event adapter. For more information see Event Pulse to Event adapter in the *Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual*. To use the Event Pulse to Event adapter from the SoC-600 product, 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® CMN-600AE Event Interface Connection Application Note*.

For more information, see your processor or DSU documentation.

A.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 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.

Signal	Direction	Description
sup_oas[2:0]	Input	Output address size supported. Width is 3-bit. The encodings for this input are: **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.
sec_override	Input	When HIGH, certain registers are accessible to Non-secure accesses from reset, as the TCU_SCR settings describe. Width is 1-bit.
ecorevnum[3:0]	Input	Tie this signal to 0 unless directed otherwise by Arm. Width is 4-bit.
msi_addr[51:0]	Input	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.
tcu_sid[31:0]	Input	Used as the DeviceID for TCU-generated MSIs. Width is 32-bit. 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.
sup_httu	Input	Width is 1-bit. **0** When set to 0, sup_httu indicates that the AXI5 interface that is connected to the system cannot support atomics. The TCU cannot perform Hardware Translation Table Update (HTTU) transactions. **1** When set to 1, sup_httu indicates that the AXI5 interface that is connected to the system can support atomics. The TCU uses atomic transactions to perform HTTU. The impact of sup_httu on SMMU_IDR0.HTTU is: **sup_httu is 0b0** SMMU_IDR0.HTTU is `0b00` **sup_httu is 0b0** SMMU_IDR0.HTTU is `0b10`

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.13 TCU ELA debug signals

The MMU L1 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. Width is 1-bit. When `TCUCFG_USE_ELA_DEBUG` is: **0** The SMMU ignores the value of the ela_enable signal. **1** ela_enable acts as a clock enable for the TCU ELA observation interface. If you require ELA debug, drive ela_enable HIGH. If you do not require ELA debug, drive ela_enable LOW to reduce the dynamic power consumption of the SMMU.

Appendix B TBU signal descriptions

The Translation Buffer Unit (TBU) in MMU L1 uses signals that can be either inputs or outputs.

MMU L1 external signal names use the following conventions:

_s
: These signals belong to the completer interface

_m
: These signals belong to the requester interface

For more information about Secure and Non-secure signals, see the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2*.

B.1 TBU clock and reset signals

The TBU has a single clock and reset input because it contains only a single clock and power domain.

MMU L1 includes a reset synchronizer. This signal can be asserted and deasserted asynchronously.

Signal definitions

Table B-1: TBU clock and reset signals

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

B.2 TBU LPI_PD interface signals

This Q-Channel completer interface manages LPI power down for the TBU. This interface is asynchronous.

Signal definitions

Table B-2: 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.

B.3 TBU LPI_CG interface signals

This Q-Channel completer interface manages LPI clock gating for the TBU. This interface is asynchronous.

Signal definitions

Table B-3: 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.

B.4 TBU DTI interface signals

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 completer DTI interface and each TBU includes a requester DTI interface. To permit bidirectional communication, each DTI interface includes one AXI5-Stream transmitter interface and one AXI5-Stream receiver interface.

Signal definitions

Table B-4: 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.

B.5 TBU transaction completer interface, completer interface signals

The transaction completer interface is an AXI5 interface on which the TBU receives incoming untranslated memory accesses. This interface supports a 64-bit address width. The completer 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. The interface implements optional signals to support the AXI5 extensions. For more information on AXI5 extensions, see AXI5 feature support.

B.5.1 TBU TBS interface, AR-channel signals

The MMU L1 Translation Buffer Unit (TBU) transaction completer interface, TBS, uses signals for the read address. These signals are the AR-channel signals.

Signal definitions

Table B-5: TBU TBS AR-channel interface signals

Signal	Direction	Description
araddr_s	Input	Read address. Width is 64-bit.
arburst_s	Input	Burst type. Width is 2-bit.
arcache_s	Input	Memory type. Width is 4-bit.
archunken_s	Input	Read data chunking enable. Width is 1-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 Translation Buffer Unit I/O configuration parameters.
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_s. The width of arloop_s is `TBUCFG_LOOP_WIDTH`-bit. For information about how to set the `TBUCFG_LOOP_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
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. The width of armmussid_s is `TBUCFG_SSID_WIDTH`-bit. For information about how to set the `TBUCFG_SSID_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
armmusid_s	Input	The armmuvalid_s qualifies the validity of these signals. The width of armmusid_s is `TBUCFG_SID_WIDTH`-bit. For information about how to set the `TBUCFG_SID_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
armmussidv_s	Input	The armmuvalid_s qualifies the validity of this signal. Width is 1-bit.
armmusecsid_s	Input	The armmuvalid_s qualifies the validity of this signal. Width is 1-bit.

Signal	Direction	Description
armmuvalid_s	Input	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.
artagop_s	Input	Read request tag operation. Width is 2-bit.
aruser_s	Input	Read address (AR) channel User signal. For more information on aruser_s, see TBU TBS User signals. If TBUCFG_DIRECT_IDX==1: • TBUCFG_ARUSER_WIDTH + $\log_2$(TBUCFG_MTLB_DEPTH) Else: • TBUCFG_ARUSER_WIDTH + $\log_2$(TBUCFG_MTLB_PARTS). TBUCFG_MTLB_PARTS still works even when it is 1, and $\log_2$(TBUCFG_MTLB_PARTS)=0. For information on how to configure these parameters, see Translation Buffer Unit I/O configuration parameters.
arvalid_s	Input	Read address valid. Width is 1-bit.

B.5.2 TBU TBS R-channel interface signals

The MMU L1 Translation Buffer Unit (TBU) transaction completer interface, TBS, uses signals for the read data, which are called the R-channel signals.

Signal definitions

Table B-6: TBU TBS R-channel interface signals

Signal	Direction	Description
rchunknum_s	Input	Read data chunk number. CHUNKNUM_WIDTH is calculated in the following way: • If TBUCFG_DATA_WIDTH is 128, then CHUNKNUM_WIDTH is 8-bit. • If TBUCFG_DATA_WIDTH is 256, then CHUNKNUM_WIDTH is 7-bit. • If TBUCFG_DATA_WIDTH is 512, then CHUNKNUM_WIDTH is 6-bit. • If TBUCFG_DATA_WIDTH is 1024, then CHUNKNUM_WIDTH is 5-bit. • Otherwise, CHUNKNUM_WIDTH is 1-bit.
rchunkstrb_s	Input	Read data chunk strobe. CHUNKSTRB_WIDTH is calculated in the following way: • If TBUCFG_DATA_WIDTH is 64, then CHUNKSTRB_WIDTH is 1-bit. • Otherwise, CHUNKSTRB_WIDTH is (TBUCFG_DATA_WIDTH / 128)-bit. For information about how to set the TBUCFG_DATA_WIDTH parameter, see Translation Buffer Unit I/O configuration parameters.

Signal	Direction	Description
rchunkv_s	Input	Valid signal of rchunknum_s and rchunkstrb_s. Width is 1-bit.
rdata_s	Input	Read data. The width of rdata_s is `TBUCFG_DATA_WIDTH`-bit. For information about how to set the `TBUCFG_DATA_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
rid_s	Input	Read ID. The width of rid_s is `TBUCFG_ID_WIDTH`-bit. For information about how to set the `TBUCFG_ID_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
ridunq_s	Input	Read data channel unique ID indicator, active-HIGH. Width is 1-bit.
rlast_s	Input	Read last. Width is 1-bit.
rloop_s	Input	Loopback value for a read response. Reflects back arloop_m. The width of rloop_s is `TBUCFG_LOOP_WIDTH`-bit. For information about how to set the `TBUCFG_LOOP_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
rpoison_s	Input	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 Translation Buffer Unit I/O configuration parameters.
rready_s	Output	Read ready. Width is 1-bit.
rresp_s	Input	Read response. Width is 3-bit.
rtag_s	Input	The tag that is associated with read data. There is a 4-bit tag per 128 bits of data, with a minimum of 4 bits. The width of rtag_s is ceil(`TBUCFG_DATA_WIDTH` / 128) × 4-bit. For information about how to set the `TBUCFG_DATA_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
ruser_s	Input	Read data (R) channel User signal. The width of ruser_s is `TBUCFG_RUSER_WIDTH`-bit. For information about how to set the `TBUCFG_RUSER_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
rvalid_s	Input	Read valid. Width is 1-bit.

B.5.3 TBU TBS AW-channel interface signals

The MMU L1 Translation Buffer Unit (TBU) transaction completer interface, TBS, uses signals for the write address, which are called the AW-channel signals.

Signal definitions

Table B-7: TBU TBS AW-channel interface signals

Signal	Direction	Description
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.

Signal	Direction	Description
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 Translation Buffer Unit I/O configuration parameters.
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	SubstreamID of the originating transaction of the originating transaction. The width of awmmussid_s is TBUCFG_SSID_WIDTH-bit. The AXI5 Untranslated_Transactions extension define this signal. The awmmuvalid_s qualifies the validity of this signal. For information about how to set the TBUCFG_SSID_WIDTH parameter, see Translation Buffer Unit configuration parameters.
awmmusid_s	Input	StreamID of the originating transaction. The width of awmmusid_s is TBUCFG_SID_WIDTH-bit. The AXI5 Untranslated_Transactions extension define this signal. The awmmuvalid_s qualifies the validity of this signal. For information about how to set the TBUCFG_SID_WIDTH parameter, see Translation Buffer Unit configuration parameters.
awmmussidv_s	Input	The awmmuvalid_s qualifies the validity of this signal. Width is 1-bit.
awmmusecsid_s	Input	The awmmuvalid_s qualifies the validity of this signal. Width is 1-bit.
awmmuvalid_s	Input	Width is 1-bit.
awtagop_s	Input	Write request tag operation. Width is 2-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. Calculate the width of awuser_s as follows: If TBUCFG_DIRECT_IDX==1: • TBUCFG_AWUSER_WIDTH + $\log_2$(TBUCFG_MTLB_DEPTH) Else: • TBUCFG_AWUSER_WIDTH + $\log_2$(TBUCFG_MTLB_PARTS). TBUCFG_MTLB_PARTS can be 1 when log2(TBUCFG_MTLB_PARTS)=0. For information about how to set these parameters, see Translation Buffer Unit configuration parameters.
awsnoop_s	Input	Transaction type of write transaction awsnoop_s. Width is 5-bit.

Signal	Direction	Description
awstashnid_s	Input	The AXI5 Cache_Stash_Transactions extension defines this signal. If TBUCFG_STASH_SUPPORT = 0, this signal is ignored. Width is 11-bit. For information about how to set the TBUCFG_STASH_SUPPORT parameter, see Translation Buffer Unit I/O configuration parameters.
awstashniden_s	Input	The AXI5 Cache_Stash_Transactions extension defines this signal. If TBUCFG_STASH_SUPPORT = 0, this signal is ignored. Width is 1-bit. For information about how to set the TBUCFG_STASH_SUPPORT parameter, see Translation Buffer Unit I/O configuration parameters.
awstashlpid_s	Input	The AXI5 Cache_Stash_Transactions extension defines this signal. If TBUCFG_STASH_SUPPORT = 0, this signal is ignored. Width is 5-bit. For information about how to set the TBUCFG_STASH_SUPPORT parameter, see Translation Buffer Unit I/O configuration parameters.
awstashlpiden_s	Input	The AXI5 Cache_Stash_Transactions extension defines this signal. If TBUCFG_STASH_SUPPORT = 0, this signal is ignored. Width is 1-bit. For information about how to set the TBUCFG_STASH_SUPPORT parameter, see Translation Buffer Unit I/O configuration parameters.
awidunq_s	Input	Write address channel unique ID indicator, active-HIGH. Width is 1-bit.
awloop_s	Input	Loopback value for a write transaction. Reflected back on bloop_s. The width of awloop_s is TBUCFG_LOOP_WIDTH-bit. For information about how to set the TBUCFG_LOOP_WIDTH parameter, see Translation Buffer Unit I/O configuration parameters.

B.5.4 TBU TBS interface W-channel signals

The MMU L1 Translation Buffer Unit (TBU) transaction completer interface, TBS, uses signals for the write data, which are called the W-channel signals.

Signal definitions

Table B-8: TBU TBS W-channel interface signals

Signal	Direction	Description
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 Translation Buffer Unit I/O configuration parameters.
wlast_s	Input	Write last. Width is 1-bit.

Signal	Direction	Description
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 Translation Buffer Unit I/O configuration parameters.
wready_s	Output	Write ready. Width is 1-bit.
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 Translation Buffer Unit I/O configuration parameters.
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 Translation Buffer Unit I/O configuration parameters.
wtag_s	Input	The tag that is associated with write data. There is a 4-bit tag per 128 bits of data, with a minimum of 4 bits. The width of wtag_s is ceil(TBUCFG_DATA_WIDTH / 128) × 4-bit. For information about how to set the TBUCFG_DATA_WIDTH parameter, see Translation Buffer Unit I/O configuration parameters.
wtagupdate_s	Input	Indicates which tags must be written to memory in an Update operation. The width of wtagupdate_s is ceil(TBUCFG_DATA_WIDTH / 128) For information about how to set the TBUCFG_DATA_WIDTH parameter, see Translation Buffer Unit I/O configuration parameters.
wvalid_s	Input	Write valid. Width is 1-bit.

B.5.5 TBU TBS B-channel interface signals

The MMU L1 Translation Buffer Unit (TBU) transaction completer interface, TBS, uses signals for the write response, which are called the B-channel signals.

Signal definitions

Table B-9: TBU TBS B-channel interface signals

Signal	Direction	Description
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 Translation Buffer Unit I/O configuration parameters.
bidunq_s	Output	Write response channel unique ID indicator, active-HIGH. Width is 1-bit.
bloop_s	Output	Loopback value for a write response. Reflects back awloop_s. The width of bloop_s is TBUCFG_LOOP_WIDTH-bit. For information about how to set the TBUCFG_LOOP_WIDTH parameter, see Translation Buffer Unit I/O configuration parameters.
bready_s	Input	Response ready. Width is 1-bit.
bresp_s	Output	Write response. Width is 3-bit.

Signal	Direction	Description
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 Translation Buffer Unit I/O configuration parameters.
bvalid_s	Output	Write response valid. Width is 1-bit.

B.5.6 TBU TBS miscellaneous awakeup_s signal

The MMU L1 Translation Buffer Unit (TBU) transaction completer interface, TBS, contains a miscellaneous awakeup_s signal.

Signal definitions

Table B-10: TBU TBS miscellaneous awakeup_s signal

Signal	Direction	Description
awakeup_s	Input	Wakeup signal. Width is 1-bit.

B.6 TBU transaction requester interface, requester interface signals

The transaction requester interface, TBM, is an AXI5 interface on which the 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.

The TBM interface can issue read and write transactions until the outstanding transaction limit is reached. The MMU L1 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

B.6.1 TBU TBM interface, AR-channel signals

The MMU L1 Translation Buffer Unit (TBU) transaction requester interface, TBM, uses signals for the read address, which are called the AR-channel signals.

Signal definitions

Table B-11: TBU TBM AR-channel interface signals

Signal	Direction	Description
araddr_m	Output	Read address. Width is 52-bit.

Signal	Direction	Description
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 Translation Buffer Unit I/O configuration parameters
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_m. The width of arloop_m is `TBUCFG_LOOP_WIDTH`-bit. For information about how to set the `TBUCFG_LOOP_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
armmusid_m	Output	Indicates the StreamID of the originating transaction. The width of armmusid_m is `TBUCFG_SID_WIDTH`-bit. For information about how to set the `TBUCFG_SID_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters The armmuvalid_m qualifies the validity of these signals.
armmusecsid_m	Output	The armmuvalid_m qualifies the validity of this signal. Width is 1-bit.
armmuvalid_m	Output	Width is 1-bit.
arpbha_m	Output	Page based hardware attributes for reads. Width is 4-bit. The meaning of the bits is **IMPLEMENTATION DEFINED** because they are set in the page tables for the current transaction.
armpam_m	Output	Read address channel MPAM information. Width is 11-bit. The width of this signal is always 11 bits, but bits[9:1] that define the PARTID are affected by the `TBUCFG_PARTID_WIDTH` parameter. When `TBUCFG_PARTID_WIDTH` < 9, the non-used bits are set to 0. See Translation Buffer Unit buffer configuration parameters
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.
artagop_m	Output	Read request tag operation. Width is 2-bit.
aruser_m	Output	Read address (AR) channel User signal. The width of aruser_m is (`TBUCFG_ARUSER_WIDTH` + 1)-bit. See AXI USER bits that MMU L1 TBU TBM and TCU QTW/DVM define. For information about how to set the `TBUCFG_ARUSER_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters
arvalid_m	Output	Read address valid. Width is 1-bit.

B.6.2 TBU TBM R-channel interface signals

The MMU L1 Translation Buffer Unit (TBU) transaction requester interface, TBM, uses signals for the read data, which are called the R-channel signals.

Signal definitions

Table B-12: TBU TBM R-channel interface signals

Signal	Direction	Description
rchunknum_m	Input	Read data chunk number. Width is `CHUNKNUM_WIDTH` and can be 1-bit, 5-bit, 6-bit, 7-bit, or 8-bit. `CHUNKNUM_WIDTH` is calculated as follows: • If `TBUCFG_DATA_WIDTH` is 128, then `CHUNKNUM_WIDTH` is 8-bit. • If `TBUCFG_DATA_WIDTH` is 256, then `CHUNKNUM_WIDTH` is 7-bit. • If `TBUCFG_DATA_WIDTH` is 512, then `CHUNKNUM_WIDTH` is 6-bit. • If `TBUCFG_DATA_WIDTH` is 1024, then `CHUNKNUM_WIDTH` is 5-bit. • Otherwise, `CHUNKNUM_WIDTH` is 1-bit. For information about how to set these parameters, see Translation Buffer Unit I/O configuration parameters.
rchunkstrb_m	Input	Read data chunk strobe. `CHUNKSTRB_WIDTH` is calculated as follows: • If `TBUCFG_DATA_WIDTH` is 64, then `CHUNKSTRB_WIDTH` is 1-bit. • Otherwise, `CHUNKSTRB_WIDTH` is (`TBUCFG_DATA_WIDTH` / 128)-bit.
rchunkv_m	Input	Valid signal of rchunknum_m and rchunkstrb_m. Width is 1-bit.
rdata_m	Input	Read data. The width of the rdata_m signal is `TBUCFG_DATA_WIDTH`-bit. For information about how to set the `TBUCFG_DATA_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
rid_m	Input	Read ID. The width of the rid_m signal is `TBUCFG_ID_WIDTH`-bit. For information about how to set the `TBUCFG_ID_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
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. Reflects back arloop_m. The width of the rloop_m signal is `TBUCFG_LOOP_WIDTH`-bit. For information about how to set the `TBUCFG_LOOP_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
rpoison_m	Input	Indicates that the read data in this transfer has been corrupted. The width of rpoison_m signal is (`TBUCFG_DATA_WIDTH` / 64)-bit. For information about how to set the `TBUCFG_DATA_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
rready_m	Output	Read ready. Width is 1-bit.
rresp_m	Input	Read response. Width is 2-bit.

Signal	Direction	Description
rtag_m	Input	The tag that is associated with read data. There is a 4-bit tag per 128 bits of data, with a minimum of 4 bits. The width of the rtag_m signal is ceil(`TBUCFG_DATA_WIDTH` / 128) × 4-bit. For information about how to set the `TBUCFG_DATA_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
ruser_m	Input	Read data (R) channel User signal. The width of the ruser_m signal is `TBUCFG_RUSER_WIDTH`-bit. For information about how to set the `TBUCFG_RUSER_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
rvalid_m	Input	Read valid. Width is 1-bit.

B.6.3 TBU TBM AW-channel interface signals

The MMU L1 Translation Buffer Unit (TBU) transaction requester interface, TBM, uses signals for the write address, which are called the AW-channel signals.

Signal definitions

Table B-13: TBU TBM AW-channel interface signals

Signal	Direction	Description
awaddr_m	Output	Write address. Width is 52-bit.
awatop_m	Output	Atomic operation. Width is 6-bit.
awburst_m	Output	Burst type. Width 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 on how to set the `TBUCFG_ID_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
awlen_m	Output	Burst length. Width is 8-bit.
awlock_m	Output	Lock type. Width is 1-bit.
awmmusid_m	Output	This signal indicates the StreamID of the originating transaction. The Generic Interrupt Controller (GIC) uses this signal to determine the DeviceID of MSIs that originate from upstream requesters. The width of awmmusid_m is `TBUCFG_SID_WIDTH`-bit. For information on how to set the `TBUCFG_SID_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters. The awmmuvalid_m qualifies the validity of this signal.
awmmusecsid_m	Output	This signal indicates the StreamID of the originating transaction. The Generic Interrupt Controller (GIC) uses this signal to determine the DeviceID of MSIs that originate from upstream requesters. Width is 1-bit. The awmmuvalid_m qualifies the validity of this signal.

Signal	Direction	Description
awmmuvalid_m	Output	This signal indicates the StreamID of the originating transaction. The Generic Interrupt Controller (GIC) uses this signal to determine the DeviceID of MSIs that originate from upstream requesters. Width is 1-bit.
awpbha_m	Output	Page based hardware attributes for writes. The meaning of the bits is **IMPLEMENTATION DEFINED** as they are set in the page tables for the current transaction. Width is 4-bit.
awtagop_m	Output	Write request tag operation. Width is 2-bit.
awprot_m	Output	Protection type. Width is 3-bit.
awqos_m	Output	QoS. Width is 4-bit.
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 these signals. Width is 11-bit. See the *AMBA® AXI Protocol Specification*.
awstashniden_m	Output	If `TBUCFG_STASH_SUPPORT` = 0, this signal is ignored. Width is 1-bit.
awstashlpid_m	Output	If `TBUCFG_STASH_SUPPORT` = 0, this signal is ignored. Width is 5-bit.
awstashlpiden_m	Output	If `TBUCFG_STASH_SUPPORT` = 0, this signal is ignored. 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. Reflected back on bloop_m. The width of awloop_m is `TBUCFG_LOOP_WIDTH`-bit. For information on how to set the `TBUCFG_LOOP_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
awmpam_m	Output	Write address channel MPAM information. Width is 11-bit. The width of this signal is always 11 bits, but bits[9:1] that define the PARTID are affected by the `TBUCFG_PARTID_WIDTH` parameter. When `TBUCFG_PARTID_WIDTH` < 9, the non-used bits are set to 0. See Translation Buffer Unit buffer configuration parameters.
awsnoop_m	Output	Transaction type of write transaction. Width is 5-bit.
awuser_m	Output	Write address (AW) channel User signal. The width of awuser_m is (`TBUCFG_AWUSER_WIDTH` + 1)-bit. See AXI USER bits that MMU L1 TBU TBM and TCU QTW/DVM define. For information on how to set the `TBUCFG_AWUSER_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
awvalid_m	Output	Write address valid. Width is 1-bit.

B.6.4 TBU TBM interface W-channel signals

The MMU L1 Translation Buffer Unit (TBU) transaction requester interface, TBM, uses signals for the write data, which are called the W-channel signals.

Signal definitions

Table B-14: TBU TBM W-channel interface signals

Signal	Direction	Description
wdata_m	Output	Write data. The width of wdata_m is `TBUCFG_DATA_WIDTH`-bit. For information on how to set the `TBUCFG_DATA_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
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 on how to set the `TBUCFG_DATA_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
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 on how to set the `TBUCFG_DATA_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
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 on how to set the `TBUCFG_WUSER_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
wtag_m	Output	The tag that is associated with write data. There is a 4-bit tag per 128 bits of data, with a minimum of 4 bits. The width of wtag_m ceil(`TBUCFG_DATA_WIDTH`/128) × 4-bit. For information on how to set the `TBUCFG_DATA_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
wtagupdate_m	Output	Indicates which tags must be written to memory in an Update operation. Width is ceil(`TBUCFG_DATA_WIDTH`/128) For information on how to set the `TBUCFG_DATA_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.

B.6.5 TBU TBM interface, B-channel signals

The MMU L1 Translation Buffer Unit (TBU) transaction requester interface, TBM, uses signals for the write response, which are called the B-channel signals.

Signal definitions

Table B-15: TBU TBM B-channel interface signals

Signal	Direction	Description
bidunq_m	Input	Write response channel unique ID indicator, active-HIGH. Width is 1-bit.
bid_m	Input	Response ID. The width of bid_m is `TBUCFG_ID_WIDTH`-bit. For information about how to set the `TBUCFG_ID_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
bloop_m	Input	Loopback value for a write response. Reflects back awloop_m. The width of bloop_m is `TBUCFG_LOOP_WIDTH`-bit. For information about how to set the `TBUCFG_LOOP_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
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. For information about how to set the `TBUCFG_BUSER_WIDTH` parameter, see Translation Buffer Unit I/O configuration parameters.
bvalid_m	Input	Write response valid. Width is 1-bit.

B.6.6 TBU TBM miscellaneous awakeup_m signal

The MMU L1 Translation Buffer Unit (TBU) transaction requester interface, TBM, contains a miscellaneous awakeup_m signal.

Signal definitions

Table B-16: TBU TBM miscellaneous awakeup_m signal

Signal	Direction	Description
awakeup_m	Input	Wakeup signal. Width is 1-bit.

B.7 TBU interrupt interface signals

The TBU interrupt signals are edge-triggered. The interrupt controller must detect the rising edge of these signals.

The MMU L1 TBU cannot output these interrupts as Message Signaled Interrupts (MSIs). These signals must be connected to an interrupt controller.

Signal definitions

Table B-17: TBU interrupt interface 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 for counter overflow. Width is 1-bit.

B.8 TBU tie-off interface signals

The TBU tie-off signals are sampled between exiting reset and the LPI_PD interface first entering the Q_RUN state.

The value of these signals must not change when the LPI_PD interface is in the Q_STOPPED or Q_EXIT state for the first time after exiting reset. In the rare cases when modifying these tie-off signals is necessary, we recommend that you only implement the changes while the SMMU is in reset.

Signal definitions

Table B-18: TBU tie-off interface 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 Translation Buffer Unit configuration parameters.
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 Translation Buffer Unit configuration parameters.
max_tok_trans	Input	Indicates the number of DTI translation tokens to request when connecting to the TCU, minus 1. The width of max_tok_trans is $\log_2$(`TBUCFG_XLATE_SLOTS`)-bit. See Translation Buffer Unit buffer configuration parameters.

Signal	Direction	Description
pcie_mode	Input	You must tie this signal HIGH when the TBU is connected to a PCIe interface. Width is 1-bit. 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 DT_TBU_TRANS_RESP message is ignored. For SMMUv3, PCIe managers do not support this feature.
sec_override	Input	When HIGH, some registers are accessible to Non-secure accesses from reset, as the TBU_SCR settings describe. Width is 1-bit.
ecorevnum	Input	Tie this signal to 0 unless we recommend otherwise. Width is 4-bit.
utlb_roundrobin	Input	Defines the MicroTLB entry replacement policy. Width is 1-bit. **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.

B.9 TBU PMU snapshot interface signals

This interface initiates a PMU snapshot. It is a 4-phase handshake. Both signals are LOW after reset. A snapshot occurs on the rising edge of pmusnapshot_req. Its function is equivalent to writing the value 1 to SMMU_PMCG_CAPR.CAPTURE. The interface is asynchronous.

When registers synchronize the pmusnapshot_req signal is sampled. Therefore, pmusnapshot_ack is a registered output.

Signal definitions

Table B-19: 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. Width is 1-bit. Connect pmusnapshot_req to the debug infrastructure of your SoC.
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. Width is 1-bit. Connect pmusnapshot_ack to the debug infrastructure of your SoC.

B.10 TBU Design For Test interface signals

The TBU uses Design For Test (DFT) signals.

Signal definitions

Table B-20: TBU DFT interface 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. Width is 1-bit. To disable reset, set this signal HIGH during scan shift.
dftramhold	Input	Preserve RAM state. Width is 1-bit. To preserve the state of the RAMs and their connected registers, set this signal HIGH during scan shift.
MBISTRESETN	Input	MBIST mode reset. Width is 1-bit. 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 MBIST test request. Width is 1-bit.
MBISTREQ	Input	This signal is encoded as follows: **0** Normal operation **1** Enable MBIST testing

B.11 TBU Embedded Logic Analyzer debug signals

The TBU includes Embedded Logic Analyzer (ELA) debug signals.

Signal definitions

Table B-21: TBU ELA debug interface signals

Signal	Direction	Description
ela_enable	Input	ela_enable is an asynchronous port when `TBUCFG_USE_ELA_DEBUG` is 0, the SMMU ignores the value of the signal. Width is 1-bit. When `TBUCFG_USE_ELA_DEBUG` is 1, ela_enable acts like a clock enable for the TBU ELA observation interface. If you require ELA debug, drive ela_enable HIGH. If you do not require ELA debug, drive ela_enable LOW to reduce the dynamic power consumption of the SMMU.
Signal group 0 signals		
signalgrp0	Output	Width is 128-bit
sigqual0	Output	Width is 4-bit
sigclken0	Output	Width is 1-bit
Signal group 1 signals		
signalgrp1	Output	Width is 128-bit
sigqual1	Output	Width is 4-bit
sigclken1	Output	Width is 1-bit
Signal group 2 signals		
signalgrp2	Output	Width is 128-bit
sigqual2	Output	Width is 4-bit
sigclken2	Output	Width is 1-bit
Signal group 3 signals		
signalgrp3	Output	Width is 128-bit
sigqual3	Output	Width is 4-bit
sigclken3	Output	Width is 1-bit
Signal group 4 signals		
signalgrp4	Output	Width is 128-bit
sigqual4	Output	Width is 4-bit
sigclken4	Output	Width is 1-bit

Appendix C TCU observation interfaces

Use the TCU observation interfaces, SIGNALGRP<n>, SIGQUAL<n>, and SIGCLKEN<n> signals 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 C-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
TCU	`TCUCFG_QTW_DATA_WIDTH` == 256	0, 1, 2, 3, 4, 5, 6, 10, 11	10
TCU	`TCUCFG_QTW_DATA_WIDTH` == 512	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	12

Some buses, if configured to be larger than the 128-bit signal group width, are spread across multiple groups. The MMU L1 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 L1 interface, to when the signal is observable on the ELA observation interface.

The following table shows the signal group output ports that are valid for the TCU.

Table C-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]	`0b0`, (tready_dti_dn AND tvalid_dti_dn), tready_dti_dn, tvalid_dti_dn	1
1	[127:0]	tdata_dti_up[127:0]	`0b0`, (tready_dti_up AND tvalid_dti_up), tready_dti_up, tvalid_dti_up	1
2	[127:124]	Unused	-	-
2	[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
2	[117:86]	tdata_dti_dn[159:128]	tvalid_dti_up, (tready_dti_up AND tvalid_dti_up), tvalid_dti_dn, (tready_dti_dn AND tvalid_dti_dn)	0
2	[85:66]	tkeep_dti_dn	tvalid_dti_up, (tready_dti_up AND tvalid_dti_up), tvalid_dti_dn, (tready_dti_dn AND tvalid_dti_dn)	0
2	[65]	tlast_dti_dn	tvalid_dti_up, (tready_dti_up AND tvalid_dti_up), tvalid_dti_dn, (tready_dti_dn AND tvalid_dti_dn)	0

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n>	Number of cycles of delay
2	[64]	twakeup_dti_dn	tvalid_dti_up, (tready_dti_up AND tvalid_dti_up), tvalid_dti_dn, (tready_dti_dn AND tvalid_dti_dn)	0
2	[63]	tready_dti_dn	tvalid_dti_up, (tready_dti_up AND tvalid_dti_up), tvalid_dti_dn, (tready_dti_dn AND tvalid_dti_dn)	0
2	[62]	tvalid_dti_dn	tvalid_dti_up, (tready_dti_up AND tvalid_dti_up), tvalid_dti_dn, (tready_dti_dn AND tvalid_dti_dn)	0
2	[61:56]	tdest_dti_up	tvalid_dti_up, (tready_dti_up AND tvalid_dti_up), tvalid_dti_dn, (tready_dti_dn AND tvalid_dti_dn)	0
2	[55:24]	tdata_dti_up[159:128]	tvalid_dti_up, (tready_dti_up AND tvalid_dti_up), tvalid_dti_dn, (tready_dti_dn AND tvalid_dti_dn)	0
2	[23:4]	tkeep_dti_up	tvalid_dti_up, (tready_dti_up AND tvalid_dti_up), tvalid_dti_dn, (tready_dti_dn AND tvalid_dti_dn)	0
2	[3]	tlast_dti_up	tvalid_dti_up, (tready_dti_up AND tvalid_dti_up), tvalid_dti_dn, (tready_dti_dn AND tvalid_dti_dn)	0
2	[2]	twakeup_dti_up	tvalid_dti_up, (tready_dti_up AND tvalid_dti_up), tvalid_dti_dn, (tready_dti_dn AND tvalid_dti_dn)	0
2	[1]	tready_dti_up	tvalid_dti_up, (tready_dti_up AND tvalid_dti_up), tvalid_dti_dn, (tready_dti_dn AND tvalid_dti_dn)	0
2	[0]	tvalid_dti_up	tvalid_dti_up, (tready_dti_up AND tvalid_dti_up), tvalid_dti_dn, (tready_dti_dn AND tvalid_dti_dn)	0
3	[127]	-	Unused	-
3	[126]	ridunq_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (arready_qtw AND arvalid_qtw)	1
3	[125:118]	rpoison_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (arready_qtw AND arvalid_qtw)	1
3	[117]	rlast_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (arready_qtw AND arvalid_qtw)	1
3	[116:106]	rid_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (arready_qtw AND arvalid_qtw)	1
3	[105]	rready_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (arready_qtw AND arvalid_qtw)	1
3	[104]	rvalid_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (arready_qtw AND arvalid_qtw)	1
3	[103:93]	arid_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (arready_qtw AND arvalid_qtw)	1
3	[92]	aridunq_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (arready_qtw AND arvalid_qtw)	1
3	[91:81]	armpam_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (arready_qtw AND arvalid_qtw)	1
3	[80:79]	ardomain_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw (arready_qtw AND arvalid_qtw)	1
3	[78:75]	aruser_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw (arready_qtw AND arvalid_qtw)	1
3	[74:71]	arqos_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (arready_qtw AND arvalid_qtw)	1

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n>	Number of cycles of delay
3	[70:67]	arcache_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (aready_qtw AND arvalid_qtw)	1
3	[66:65]	arburst_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (aready_qtw AND arvalid_qtw)	1
3	[64:62]	arsize_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (aready_qtw AND arvalid_qtw)	1
3	[61:54]	arlen_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (aready_qtw AND arvalid_qtw)	1
3	[53:2]	araddr_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (aready_qtw AND arvalid_qtw)	1
3	[1]	arready_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (aready_qtw AND arvalid_qtw)	1
3	[0]	arvalid_qtw	rvalid_qtw, (rready_qtw AND rvalid_qtw), arvalid_qtw, (aready_qtw AND arvalid_qtw)	1
4	[127]	unused	-	-
4	[126]	crready_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[125]	crvalid_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[124:114]	bid_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[113]	bidunq_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[112]	bready_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[111]	bvalid_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[110]	awakeup_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[109:99]	awid_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[98:93]	awatop_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[92]	awidunq_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[91:81]	awmpam_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[80:79]	awdomain_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[78:75]	awuser_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[74:71]	awqos_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[70:67]	awcache_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n>	Number of cycles of delay
4	[66:65]	awburst_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[64:62]	awsize_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[61:54]	awlen_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[53:2]	awaddr_qtw	0b0, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[1]	awready_qtw	0b00, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
4	[0]	awvalid_qtw	0b00, (crready_qtw AND crvalid_qtw), (bready_qtw AND bvalid_qtw), (awready_qtw AND awvalid_qtw)	1
5	[127]	syscoack_qtw	0b00 , (wready_qtw AND wvalid_qtw), (acready_qtw AND acvalid_qtw)	1
5	[126]	syscoack_qtw	0b00 , (wready_qtw AND wvalid_qtw), (acready_qtw AND acvalid_qtw)	1
5	[125:62]	syscoreq_qtw	0b00 , (wready_qtw AND wvalid_qtw), (acready_qtw AND acvalid_qtw)	1
5	[61]	wlast_qtw	0b00 , (wready_qtw AND wvalid_qtw), (acready_qtw AND acvalid_qtw)	1
5	[60]	wready_qtw	0b00 , (wready_qtw AND wvalid_qtw), (acready_qtw AND acvalid_qtw)	1
5	[59]	wvalid_qtw	0b00 , (wready_qtw AND wvalid_qtw), (acready_qtw AND acvalid_qtw)	1
5	[58]	acwakeup_qtw	0b00 , (wready_qtw AND wvalid_qtw), (acready_qtw AND acvalid_qtw)	1
5	[57:6]	acaddr_qtw	0b00 , (wready_qtw AND wvalid_qtw), (acready_qtw AND acvalid_qtw)	1
5	[5:2]	acvmidext_qtw	0b00 , (wready_qtw AND wvalid_qtw), (acready_qtw AND acvalid_qtw)	1
5	[1]	acready_qtw	0b00 , (wready_qtw AND wvalid_qtw), (acready_qtw AND acvalid_qtw)	1
5	[0]	acvalid_qtw	0b00 , (wready_qtw AND wvalid_qtw), (acready_qtw AND acvalid_qtw)	1
6	[127:0]	rdata_qtw[127:0]	0b0, (rready_qtw AND rvalid_qtw), rready_qtw, rvalid_qtw	1
7	[127:0]	rdata_qtw[255:128]	0b0, (rready_qtw AND rvalid_qtw), rready_qtw, rvalid_qtw	2
8	[127:0]	rdata_qtw[383:256]	0b0, (rready_qtw AND rvalid_qtw), rready_qtw, rvalid_qtw	3
9	[127:0]	rdata_qtw[511:384]	0b0, (rready_qtw AND rvalid_qtw), rready_qtw, rvalid_qtw	4
10	[127:0]	wdata_qtw_demuxed[127:0]	0b0, (rready_qtw AND rvalid_qtw), rready_qtw, rvalid_qtw 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 Configuration parameters and methodology.	1
11	[127:0]	wdata_qtw_demuxed[255:128]	0b0, (rready_qtw AND rvalid_qtw), rready_qtw, rvalid_qtw	2

Appendix D TBU observation interfaces

The TBU observation interfaces, SIGNALGRP<n>, SIGQUAL<n>, and SIGCLKEN<n> signals 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 TBU.

Table D-1: Number of signal groups per module for the TBU

Component	Parameter	Enabled signal groups	Total
TBU	-	0, 1, 2, 3, 4	5

Some buses, if configured larger than the 128-bit signal group width, are spread across multiple groups. MMU L1 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 following table indicates the number of cycles, from when the signal is observable on an MMU L1 interface, to when the signal is observable on the ELA observation interface.

Table D-2: TBU observation interface signals

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n>4'b{MSB..LSB}	Number of cycles of delay
0	[127:124]	Reserved	-	-
0	[123]	awuser_oc (outer cacheable)	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
0	[122]	awidunq_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
0	[121:116]	awatop_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
0	[115:114]	awdomain_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
0	[113:110]	awqos_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
0	[109:107]	awprot_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
0	[106:103]	awcache_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
0	[102:101]	awburst_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
0	[100:98]	awsize_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
0	[97:90]	awlen_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
0	[89:86]	awregion_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
0	[85:54]	awid_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
0	[53:2]	awaddr_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n>4'b{MSB..LSB}	Number of cycles of delay
0	[1]	awready_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
0	[0]	awvalid_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
1	[127:116]	Reserved	-	-
1	[115:105]	awmpam_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	2
1	[104:81]	awmmusid_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	2
1	[80]	awmmusecsid_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	2
1	[79:72]	awawloop_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	2
1	[71]	awstashlpiden_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	2
1	[70:66]	awstashlpid_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	2
1	[65]	awstashniden_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
1	[64:54]	awstashnid_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
1	[53:2]	awaddr_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
1	[1]	awready_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
1	[0]	awvalid_m	0b0, (awready_m AND awvalid_m), awready_m, awvalid_m	1
2	[1271:113]	Reserved	-	-
2	[112:61]	araddr_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
2	[60:59]	ardomain_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
2	[58:55]	arqos_qtw	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
2	[54:52]	arprot_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
2	[51:48]	arcache_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
2	[47:46]	arburst_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
2	[45:43]	arsize_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
2	[42:35]	arlen_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
2	[34]	aruser_oc (outer cacheable)	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
2	[33:2]	arid_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
2	[1]	arready_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
2	[0]	arvalid_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
3	[127:113]	Reserved	-	-
3	[112:111]	artagop_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
3	[110]	armmuvalid_mw	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
3	[109]	archunken_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
3	[108:98]	armpam_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
3	[97:90]	arloop_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
3	[89]	aridunq_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
3	[88:65]	armmusid_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
3	[64]	armmusecsid_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
3	[63:60]	arregion_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
3	[59:58]	ardomain_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1
3	[57:54]	arqos_m	0b0, (arready_m AND arvalid_m), arready_m, arvalid_m	1

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n>4'b{MSB..LSB}	Number of cycles of delay
3	[53:51]	arprot_m	0b0, (aready_m AND arvalid_m), aready_m, arvalid_m	1
3	[50:47]	arcache_m	0b0, (aready_m AND arvalid_m), aready_m, arvalid_m	1
3	[46:45]	arburst_m	0b0, (aready_m AND arvalid_m), aready_m, arvalid_m	1
3	[44:42]	arsize_m	0b0, (aready_m AND arvalid_m), aready_m, arvalid_m	1
3	[41:34]	arlen_m	0b0, (aready_m AND arvalid_m), aready_m, arvalid_m	1
3	[33:2]	arid_m	0b0, (aready_m AND arvalid_m), aready_m, arvalid_m	1
3	[1]	aready_m	0b0, (aready_m AND arvalid_m), aready_m, arvalid_m	1
3	[0]	arvalid_m	0b0, (aready_m AND arvalid_m), aready_m, arvalid_m	1
4	[127:94]	Reserved	-	-
4	[93]	wlast_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	1
4	[92]	wready_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	1
4	[91]	wvalid_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	1
4	[90:83]	bloop_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	1
4	[82]	bidunq_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	1
4	[81:80]	bresp_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	1
4	[79:48]	bid_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	1
4	[47]	bready_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	1
4	[46]	bvalid_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	1
4	[45]	ridunq_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	1
4	[44:37]	rloop_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	1
4	[36]	rlast_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	2
4	[35:34]	rresp_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	3
4	[33:2]	rid_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	4
4	[1]	rready_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	1
4	[0]	rvalid_m	0b0, (wready_m AND wvalid_m), (bready_m AND bvalid_m), (rready_m AND rvalid_m)	2

Appendix E Software initialization examples for MMU L1

Software must initialize MMU L1 before you can use it. MMU L1 supports Secure and Non-secure translation worlds.

You can initialize Non-secure translation. The procedures for initializing Secure translation are similar, and require you to access the corresponding MMU L1 Secure registers. However, we do 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 L1 initialization, see the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2*.

E.1 Allocate the command queue

MMU L1 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. See the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2*.

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. See the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2*.

3. Set the queue read index in SMMU_CMDQ_CONS and the queue write index in SMMU_CMDQ_PROD to 0.

Setting the queue read index and the queue write index to the same value indicates that the queue is empty.

E.2 Allocate the event queue

MMU L1 uses the Event queue to signal events. Software must allocate memory for the Event queue and configure the appropriate registers in the MMU.

About this task

To allocate the Event queue, ensure that your software performs the following steps:

Procedure

1. Allocate memory for the Event queue.
2. Configure the Event queue size and base address by writing to the SMMU_EVENTQ_BASE register.

The queue size can affect how many bits of the SMMU_EVENTQ_CONS and SMMU_EVENTQ_PROD indices are writeable. It is therefore important that you perform this step before writing to SMMU_EVENTQ_CONS and SMMU_EVENTQ_PROD.

3. Set the queue read index in SMMU_EVENTQ_CONS and the queue write index in SMMU_EVENTQ_PROD to 0.

Setting the queue read index and the queue write index to the same value indicates that the queue is empty.

E.3 Configure 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.

E.4 Initialize 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.

E.5 Initialize 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.

E.6 Invalidate TLBs and configuration caches

Before use, MMU L1 TLBs and configuration cache structures must be invalidated by issuing commands to the Command queue. When powered on, MMU L1 invalidates TLBs and configuration cache structures automatically.

About this task

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.

You can only issue commands to invalidate Secure TLB entries through the Secure Command queue. For a system that implements 2 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 the sec_override signal or TCU_SCR.

Procedure

1. To invalidate TLB entries for Non-secure EL1 contexts, issue CMD_TLBI_NSNH_ALL
2. To invalidate TLB entries for EL2 contexts, issue CMD_TLBI_EL2_ALL
3. To invalidate TLB entries for EL3 contexts, issue CMD_TLBI_EL3_ALL
4. To invalidate TLB entries for Secure EL1 contexts, issue CMD_TLBI_NH_ALL

Next steps

1. To invalidate both the TCU configuration cache and the TBU combined configuration cache and TLB, issue the CMD_CFGI_ALL command.
2. To force all previous commands to complete, issue CMD_SYNC.
3. To invalidate all configuration caches and TLB entries for all translation regimes and Security states, ensure that Secure software:
   a. Sets SMMU_S_INIT.INV_ALL to 1. The SMMU sets SMMU_S_INIT.INV_ALL to 0 after the invalidation completes.
   b. 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*.

E.7 Create 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.

About this task

To create a CD, ensure that your software performs the following steps:

Procedure

1. Allocate 64 bytes of memory for the CD.
2. Configure the CD fields according to the information in the following table.

The following table shows how to configure the CD.

Table E-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
IR0	Cacheability attribute to use for translation table walks to TTB0:
OR0	Cacheability attribute to use for translation table walks to TTB0: **0** Non-cacheable **1** Write-Back Cacheable, Read-Allocate Write-Allocate. **10** Write-Through Cacheable, Read-Allocate.
SH0	Shareability of translation table walks to TTB0: **0** Non-shareable **1** 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
V	Valid CD. This field must be set to 1

E.8 Create 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.

About this task

To create an STE, ensure that your software performs the following steps:

Procedure

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:

   0b000
   No traffic can pass through the MMU. MMU L1 returns an abort.

   0b100
   Stage 1 and Stage 2 bypass.

   0b101
   Stage 1 translation Stage 2 bypass.

   0b110
   Stage 1 bypass Stage 2 translation.

   0b111
   Stage 1 and Stage 2 translation.

Next steps

- If Stage 1 translation is enabled, you can configure 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.

- 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.S2ENDI
  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
  `0b00`: Non-cacheable.

STE.S2OR0

`0b00`: Non-cacheable.

STE.S2SH0

`0b00`: Non-cacheable.

STE.S2VMID

Contains the VMID associated with these translations.

E.9 Enable the SMMU

Software can enable MMU L1 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.

Proprietary Notice

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

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

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

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

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant

export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in Arm documents.

Product status

All products and services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

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

Product revision status

The r0p2 identifier indicates the revision status of the product described in this manual, where:

`rx` Identifies the major revision of the product.
`py` Identifies the minor revision or modification status of the product.

Revision history

These sections can help you understand how the document has changed over time.

Document release information

The Document history table gives the issue number and the released date for each released issue of this document.

Document history

Issue	Date	Confidentiality	Change
0002-05	14 August 2025	Non-Confidential	Second release for r0p2 EAC
0002-04	25 April 2025	Confidential	First release for r0p2 EAC
0001-03	14 June 2024	Confidential	First release for r0p1 EAC
0000-02	22 February 2024	Confidential	First release for r0p0 LAC
0000-01	5 September 2023	Confidential	First release for r0p0 BET

Change history

The Change history tables describe the technical changes between released issues of this document in reverse order. Issue numbers match the revision history in Document release information on page 225.

Table 2: Differences between issue 0002-04 and issue 0002-05

Change	Location
Changed the product name to Arm® MMU L1 System Memory Management Unit.	Throughout the document
Corrected the maximum number of outstanding writes. Added information about the alignment of HTTU write transactions.	TCU transaction handling
Corrected DEFAULT_PRIORITY to PRI_LEVEL.	TCU_NODE_CTRLn register
Added the reset value.	TCU_WC_SxLy_CMAX registers
Corrected TCUCFG_NUM_PMU_COUNTERS to TCUCFG_PMU_COUNTERS.	TCU_SYSDISC10
Added AUX15...AUX1 to the figure.	TBU_CTRL, TBU Control register
Reinstated the TCU DTI interface signals content.	TCU DTI interface signals

Table 3: Differences between issue 0001-03 and issue 0002-04

Change	Location
Updated the description field for the SMMU_IIDR.Revision field, where p_level is 2 for p2.	ID register architectural values
Changed register widths from 64-bit to 32-bit. Clarified that the top 32 bits are Reserved.	TCU Reliability, Availability, and Serviceability register summary
Updated the description field for SMMU_PMCG_PIDR3, Peripheral ID3 register where p_level is 2 for p2.	TMC PMU ID registers
Changed register width from 64-bit to 32-bit.	TCU_ERRGEN register
Changed register widths from 64-bit to 32-bit. Clarified that the top 32 bits are Reserved.	TBU Reliability, Availability, and Serviceability registers
Updated the description field for SMMU_PMCG_PIDR3, Peripheral ID3 register where p_level is 2 for p2.	TBU PMU ID registers

Table 4: Differences between issue 0000-02 and issue 0001-03

Change	Location
Removed Access Control System (ACS) from the list of features supported by the SMMUv3.2 architecture.	Supported features
Updated Stash to StashOnceShare, StashOnceUnique for AWSNOOP_S inside. Updated Data or Privileged to Privileged Data. Updated section on Completer interface AxPROT handling.	Completer interface attribute handling
Added new text on how AXI allows InvalidateHint Cache Maintenance Operations (CMO) to be either Data or Instruction access.	Transaction types
Updated TCU MPAM table layout.	TCU MPAM
Added another constraint to implement TBU MPAM. Updated TBU MPAM table layout.	TBU MPAM
Updated the DTI interconnect switch interfaces and descriptions.	DTI interconnect switch interfaces

Change	Location
Updated the DTI interconnect sizer interfaces and descriptions.	DTI interconnect sizer interfaces
Updated the DTI register slice interfaces and descriptions.	DTI interconnect register slice interfaces
Added new event id `0x8E` and updated the description for event id `0x8C`.	MMU L1 TBU events
Removed redundant text.	Requester interface memory type attribute handling
Updated tables MMU L1-defined TBU aruser_m and awuser_m bits and MMU L1-defined TCU aruser_qtw and awuser_qtw bits.	AXI USER bits that MMU L1 TBU TBM and TCU QTW/DVM define
Updated ARSNOOP table headings and values.	AXI5 read transaction support
Updated AWSNOOP table headings and values.	AXI5 write transaction support
Updated the descriptions for the `TBUCFG_ROT_DEPTH` and `TBUCFG_WOT_DEPTH` parameters.	Translation Buffer Unit configuration parameters
Updated text on legal values and valid error records.	TCU_ERRGEN register
Updated text on legal values and valid error records.	TBU_ERRGEN, TBU Error Generation register
Updated the description for the aruser_s signal.	TBU TBS interface, AR-channel signals
Updated the description for the awuser_s signal.	TBU TBS AW-channel interface signals

Table 5: Differences between issue 0000-01 and issue 0000-02

Change	Location
General improvements to wording and structure.	Throughout the document
Updated the SMMU_IIDR.ProductID field description to MMU L1 TCU ID.	ID register architectural values
Added SMMU_S_IDR4 and SMMU_S_VATOS_* as optional registers that are not implemented. Added (PMCG) registers SMMU_PMCG_IRQ_STATUS, SMMU_PMCG_GMPAM, SMMU_PMCG_MPAMIDR, and SMMU_PMCG_S_MPAMIDR as registers that are not implemented and are also **RAZ/WI**.	Non-implemented registers
Updated values from Y and N, to True and False. Added note on Read_Interleaving_Disabled parameter.	AXI5 feature support
Updated the conditions when an InvalidHint is not terminated.	Transactions that do not cause a translation fault
Updated the TCU MPAMCFG_PART_SEL implemented register fields PARTID_SEL and RIS from Banked to Shared.	TCU MPAM
Updated the TBU MPAMCFG_PART_SEL implemented configuration register fields PARTID_SEL and RIS from Banked to Shared.	TBU MPAM
Added new event Write data bypasses write data buffer, changed DCMO downgrade to MakeInvalid downgrade, changed value for Fixed Burst Termination to `0x8D` and event description.	MMU L1 TBU events
Added a new topic on TBU TBS User signals.	TBU TBS User signals

Change	Location
Updated the maximum value for the parameter `TBUCFG_SSID_WIDTH` to 20.	Translation Buffer Unit configuration parameters
Updated values for the parameter `TBUCFG_PMU_COUNTERS` to 4, 8, 16, and 32.	Translation Buffer Unit buffer configuration parameters
Added the Performance Monitor Counter Group (PMCG) registers SMMU_PMCG_GMPAM, SMMU_PMCG_IRQ_STATUS, SMMU_PMCG_MPAMIDR, and SMMU_PMCG_S_MPAMIDR as registers that MMU L1 does not implement.	Programmers model
Removed duplicate table row.	SMMUv3 Performance Monitor Counter Group registers
Added the bit assignments and bit diagrams for the TBU system discovery registers. Bit assignment numbers updated.	TBU_SYSDISC0 to TBU_SYSDISC21

Table 6: Issue 0000-01

Change	Location
First release	-

Conventions

The following subsections describe conventions used in Arm documents.

Glossary

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

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

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

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.

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

We recommend the following. If you do not follow these recommendations your system might not work.

Your system requires the following. If you do not follow these requirements your system will not work.

You are at risk of causing permanent damage to your system or your equipment, or of harming yourself.

This information is important and needs your attention.

This information might help you perform a task in an easier, better, or faster way.

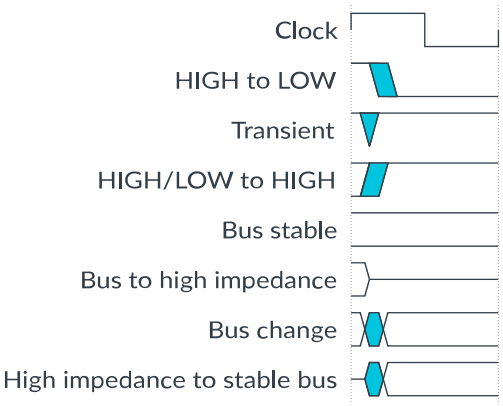
This information reminds you of something important relating to the current content.

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: 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.

Useful resources

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

Arm documents are available on developer.arm.com/documentation.

Confidential documents are only available to licensees, when logged in. Each document link in the tables below provides direct access to the online version of the document.

Arm product resources	Document ID	Confidentiality
Arm® CMN-600AE Event Interface Connection Application Note	ARM051-799564642-325	Non-Confidential
Arm® CoreLink™ ADB-400 AMBA® Domain Bridge User Guide	DUI 0615	Confidential
Arm® CoreLink™ GIC-700 Generic Interrupt Controller Technical Reference Manual	101516	Non-Confidential
Arm® CoreLink™ LPD-500 Low Power Distributor Integration and Implementation Manual	100362	Confidential
Arm® CoreLink™ LPD-500 Low Power Distributor Technical Reference Manual	100361	Non-Confidential
Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Configuration and Integration Manual	101089	Confidential
Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual	101088	Non-Confidential
Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual	100806	Non-Confidential
Arm® MMU L1 System Memory Management Unit Configuration and Integration Manual	107958	Confidential
Arm® MMU L1 System Memory Management Unit Release Note	109310	Confidential

Arm architecture and specifications	Document ID	Confidentiality
AMBA® APB Protocol Specification	IHI 0024E	Non-Confidential
AMBA® AXI Protocol Specification	IHI 0022J	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 0068D	Non-Confidential
Arm® Architecture Reference Manual for A-profile architecture	DDI 0487J.a	Non-Confidential
Arm® Base System Architecture 1.0C Platform Design Document	DEN 0094C	Non-Confidential
Arm® Memory System Resource Partitioning and Monitoring (MPAM) System Component Specification	IHI 0099	Non-Confidential
Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1, and 3.2	IHI 0070F.a	Non-Confidential