SMMU feature introduction

### About the MMU-500

The MMU-500 is a system-level Memory Management Unit (MMU) that translates an input address to an output address, based on address mapping and memory attribute information available in the MMU-500 internal registers and translation tables.

An address translation from an input address to an output address is described as a stage of address translation.

The MMU-500 supports the translation table formats defined by the ARM architecture, ARMv7 and ARMv8, and 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 output IPA, and then translate that IPA to a PA. The MMU-500 performs translation table walks for each stage of the translation.

Address translation can span over two stages, namely stage 1 and stage 2. Address translation can require multiple translation table lookups. Each translation table lookup is described as a level of address lookup. Each level of stage 1 translation might require additional stage 2 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 defined by the stage 1 translation.

A stage of address translation can be disabled or bypassed, and the MMU-500 can define memory attributes for disabled and bypassed stages of translation.

The MMU-500 uses inputs from the requesting master to identify a context. This context tells the MMU-500 what resources to use for the translation, including the translation tables to use.

For the stage 1 translations that are typically associated with application and OS-level operation, the VA range can be split into two subranges, translated by Translation Table Base registers, TTBR0 and TTBR1, each with associated translation tables and control registers.

Stage 1 translations are supported for both Secure and Non-secure translation contexts. Usually, the appropriate OS:

* Defines the translation tables, in memory, for the stage 1 translations for its security state.
* Programs the MMU-500 to configure stage 1 translations, and then enables the translations.

Stage 2 translations are supported only for Non-secure translation contexts. The typical usage model for two stages of address translation is as follows:

* The Non-secure operating system defines the stage 1 address translations for application-level and OS-level operation. It does this assuming it is mapping from the Vas used by the processors to PAs in the physical memory system. However, it actually maps VAs to IPAs.
* The hypervisor defines the stage 2 address translations that map the IPAs to PAs. It does this as part of its virtualization of one or more Non-secure guest operating systems.

The MMU-500 can cache the result of a translation table lookup in a Translation Lookaside Buffer (TLB) that means the MMU-500 also supports TLB maintenance operations.

The MMU-500 has the following key components:

* ***Translation Buffer Unit* (TBU)** The TBU contains a TLB that caches page tables. The MMU-500 implements a TBU for each connected master, and the TBU is designed, so that it is local to the master.
* ***Translation Control Unit* (TCU)** Controls and manages the address translations. The MMU-500 implements a single TCU.
* **Interconnect** Connects multiple TBUs to the TCU.

The Figure 1-1 shows the MMU-500 block diagram.

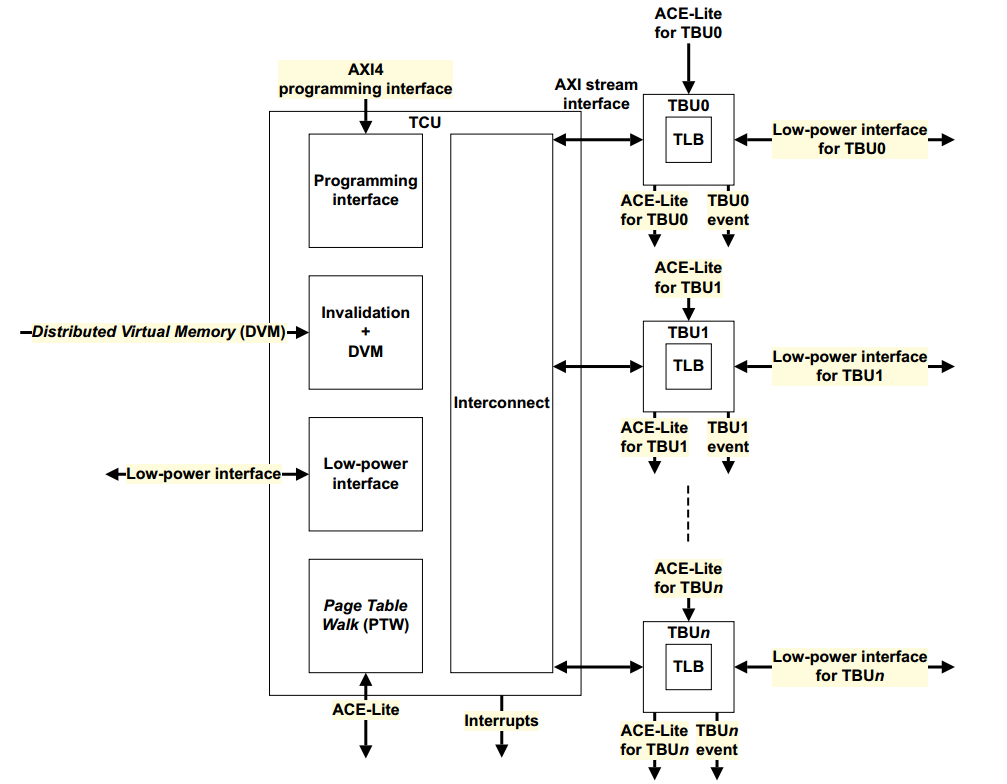


Figure 1-1

### Feature list

The MMU-500 provides the following features:

* Address virtualization to other masters in an ARM processor based system and other bus masters in the system.
* Support for the following translations:
* Stage 1.
* Stage 2.
* Stage 1 followed by stage 2.
* Programmable *Quality of Service* (QoS).
* Distributed translation support for up to 32 TBUs.
* Translation support for 32-bit to 49-bit virtual address ranges and 48-bit physical address ranges.
* Multiple transaction contexts to apply to address translations for specific streams of transactions. Supports up to 128 configurable contexts and programmable page size. The MMU-500 maps each context by using an input stream ID from the master device that requires address translation.
* Translation support for the following:
* Stage 1 ARMv7 VMSA.
* Stage 1 and Stage 2 ARMv8 AArch32.
* Stage 1 and Stage 2 ARMv8 AArch64 with 4KB and 64KB granules.
* Stage 1 followed by stage 2 translations.
* No page size restrictions. All page sizes are supported apart from the 16KB page granule defined by ARMv8 architecture.
* Arbitration of PTW requests from different TBUs by using the programmed QoS value.
* Page table walk cache for storing intermediate page table walk data.
* Page table entry cache in the TLB.
* Support for TLB Hit-Under-Miss (HUM).
* Configurable PTW depth using parallel PTWs.
* TLB invalidation through the AMBA 4 DVM signaling or register programming.
* Translation and protection check support including TrustZone® extension support.
* Fault handling, logging, and signaling that includes demand paging and support for the stall model.
* One AMBA slave interface that supports ACE-Lite per TBU for connecting the bus master device that requires address translations.
* One AMBA master interface for master device transactions or PTWs that support ACE-Lite and DVM.
* An AXI4 interface for programming.
* Page table entry cache in the TLB at two levels, namely:
* Macro TLB.
* Micro TLB.
* The TLB at two levels and the walk cache RAMs support single bit error detection and invalidation on error detection. The context disambiguation *Multi-FIFO* (MFIFO) RAM supports single bit error detection and correction.
* Debug and performance-monitoring events.
* A prefetch buffer to prefetch the next 4K or 64K leaf page entry to reduce latency.
* An IPA2PA cache to speed up stage 1 followed by stage 2 translations.
* Support for 256 outstanding transactions for each TBU master interface.
* Support for priority elevation as part of the QoS scheme.