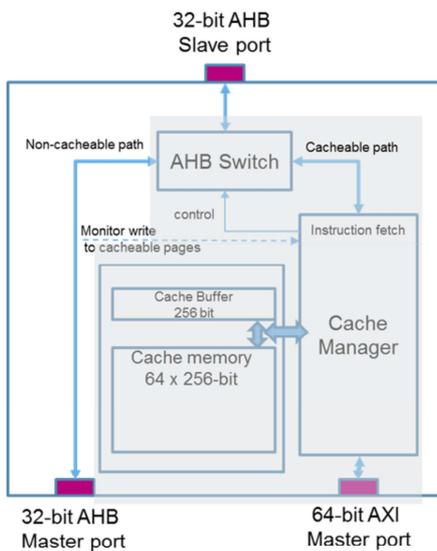


STM32H7x5 and STM32H7x7 Lines ART Accelerator

ART™ (adaptive real-time) accelerator
Revision 1.0



Hello, and welcome to this presentation about the STM32H7x5 and STM32H7x7 dual-core lines ART™ (adaptive real-time) accelerator.



- STM32H7x5 and STM32H7x7 ART features
 - 64 cache lines of 256 bits to speed up instruction fetch accesses of the Cortex-M4 core to D1-domain memories
 - 64-bit AXI master port to load selected code into an embedded cache

Application benefits

- Code is instantly available to Cortex-M4 core
- Avoid latency due to memory wait states

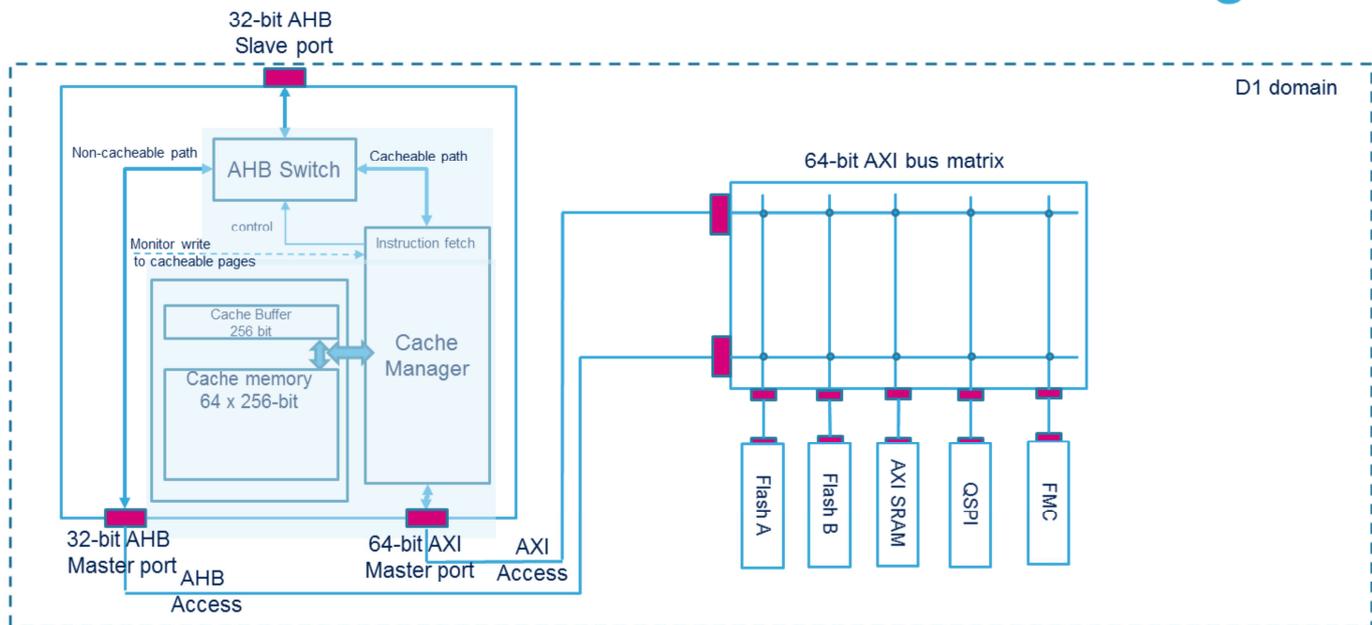


The ART™ (adaptive real-time) accelerator block speeds up instruction fetch accesses of the Cortex-M4 core to D1-domain internal memories (Flash A memory, Flash B memory, and AXI SRAM) and to D1-domain external memories attached via the Quad-SPI controller (QSPI) and Flash memory controller (FMC).

The acceleration is achieved by loading selected code into an embedded cache and making it instantly available to the Cortex-M4 core, thus avoiding latency due to memory wait states.

ART Block diagram

3



This figure shows the block schematic and the environment of the ART accelerator:

Instruction fetch accesses can be qualified as cacheable or non-cacheable.

The cache memory is organized in 64 lines of 256 bits each. Each line contains eight 32-bit segments. Each segment can contain one 32-bit or two 16-bit instruction codes. The cache buffer is an extra line of 256 bits keeping a copy of one of the cache lines.

The Cache manager checks whether the incoming memory instruction fetch access is cacheable and controls the AHB switch accordingly. For a cacheable access, it allows the instruction fetch from cache in case of a cache hit and it manages the cache memory line refill (instruction code load) in case of a cache miss. For non-cacheable memory accesses, the instruction is directly fetched from memory.

- 32-bit AHB slave port for interfacing with the D2 domain
- 32-bit AHB master port for non-cacheable memory accesses
- 64-bit AXI master port for code load from memory to cache
- Fully-associative cache
- Least recently used replacement policy
- Programmable cacheable page with content consistency checker



ART™ accelerator features:

- 32-bit AHB slave port to interface with the D2 domain
- 32-bit AHB master port for non-cacheable memory accesses
- 64-bit AXI master port to load the code from memory to cache
- 64 cache lines of 256 bits
- Fully-associative cache
- Programmable cacheable page
- Cache content consistency checker

- Instruction fetch accesses can be (depending on MPU attributes):
 - Cacheable
 - Non-cacheable
- Instruction fetch follows:
 - Cacheable access path when access attributes are set as cacheable
 - Master AHB port, for non-cacheable accesses
- The cache is organized as
 - A memory block: 64 lines of 256 bits each
 - A cache buffer: 256 bits
 - It allows fetching instruction code by the core while the corresponding buffer source line is in refill progress.



The instruction fetch accesses can be qualified as cacheable or non-cacheable depending on MPU attributes.

For cacheable accesses, the instruction fetch follows the cacheable access path. Non-cacheable accesses are directed to the master AHB port, following the non-cacheable access path. The appropriate access path is selected with the AHB switch controlled by the cache manager.

The cache memory is organized in 64 lines of 256 bits each. Every line contains eight 32-bit segments. Each segment can contain one 32-bit or two 16-bit instruction codes. The cache buffer is an extra line of 256 bits keeping a copy of one of cache lines (buffer source line). It allows fetching instruction code by the core while the corresponding buffer source line is in refill progress.

- To ensure cache consistency, all cache lines are invalidated if one of the following operations is detected:
 - A write access modifies a piece of data in the cacheable page
 - The software modifies the cacheable page index in the ART accelerator's control register.
 - A data transfer error occurs during the cache buffer and/or cache memory line refill



The following cases can cause the code in the cache to become inconsistent with the code on the corresponding address of the addressed memory:

- A write access modifies a piece of data in the cacheable page of the addressed memory.
- The software modifies the cacheable page index in the ART accelerator's control register.
- A data transfer error occurs during the cache buffer and/or cache memory line refill.

The device incorporates a mechanism preventing software corruption due to the cache inconsistency. It detects the above cases and flags all the cache memory lines and the cache buffer as invalid. As a consequence, the next cacheable accesses result in a cache miss and so the cache line refills with new data.