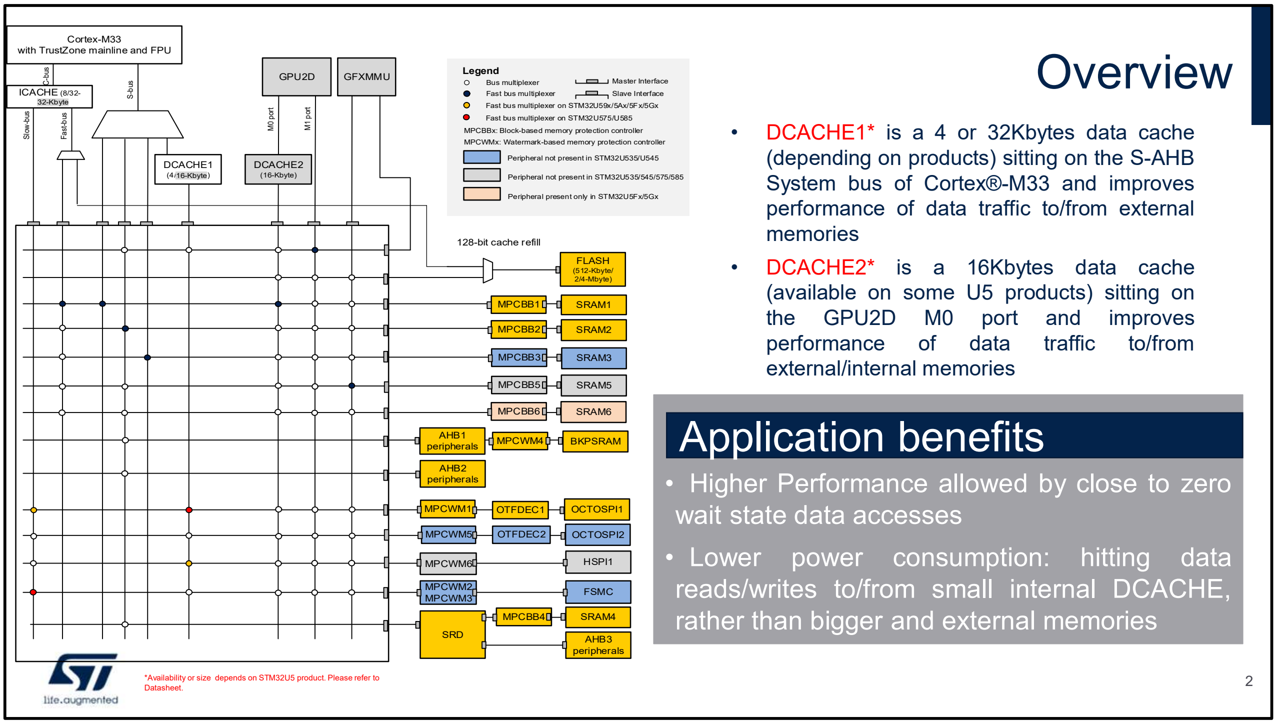
STM32U5

DCACHE

Hello, and welcome to this presentation of the DCACHE module, which is embedded in all products of the STM32U5 microcontroller family.



The STM32U5 Series embed up to two data caches :

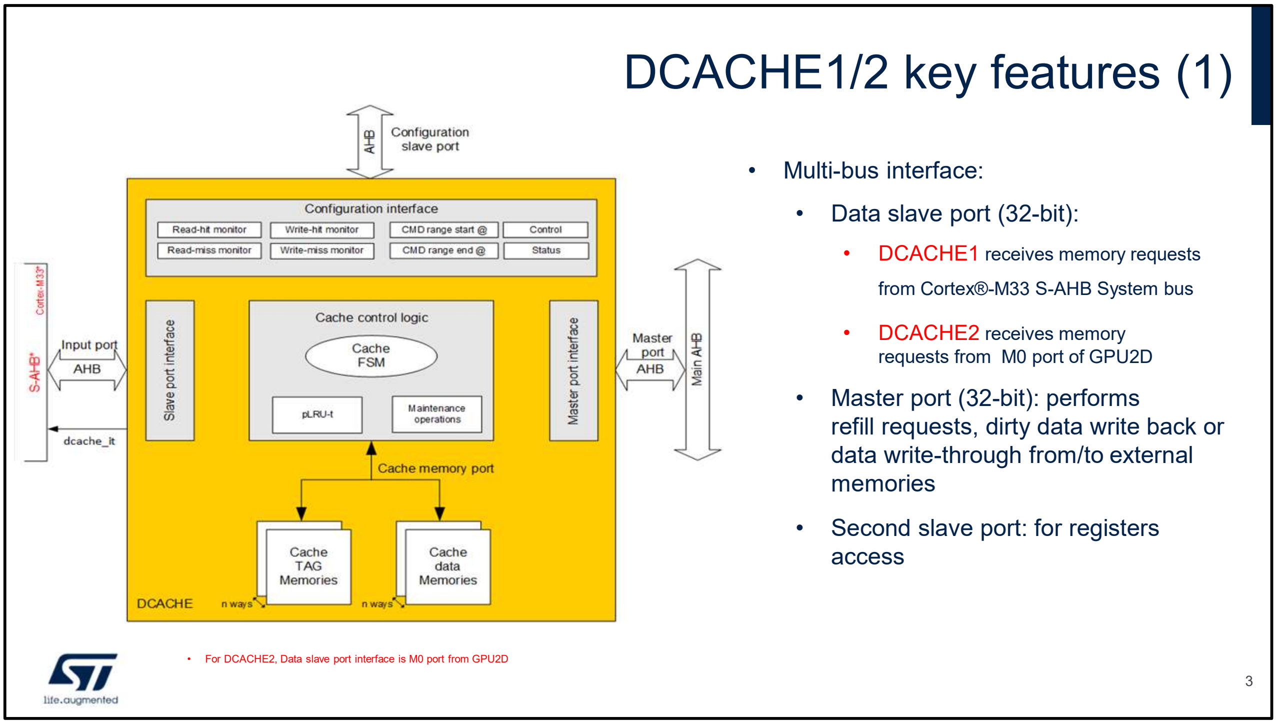
- DCACHE1 is a 4 or 32 Kilobyte data cache (depending on products)
- DCACHE2 is a 16 Kilobyte data cache (available on some U5 products).

DCACHE1 is placed on Cortex-M33 S-AHB bus and caches only the external RAM memory region (OCTOSPIs, HSPI, and FSMC).
Indeed, by placing a bus matrix demultiplexing node in front of DCACHE1, S-AHB bus memory requests addressing SRAM region or peripherals region, are routed directly to the main AHB bus matrix, and the DCACHE1 is bypassed.
The concurrence between DCACHE1 accesses to external memories and core accesses to internal SRAMs also improves

the overall performance of the microcontroller.
In the figure, the DCACHE1 master bus used to access external memories, is completely independent of the bus used to access internal SRAMs.

DCACHE2 is placed on the AHB bus driven by the port M0 of GPU2D, and caches all the memory regions accessed by it.
The best performance is achieved by caching only external memories and by bypassing DCACHE2 for internal memories.

Both DCACHEs autonomously handle cache line reloads, cache line evictions, and write stores to external memories. Performance is achieved through the two following features: hit-under-miss support and critical-word-first refill policy. These data caches help to reduce the microcontroller's power consumption by accessing data in their internal memory, rather than in the main memories, which are larger and therefore consume more power.

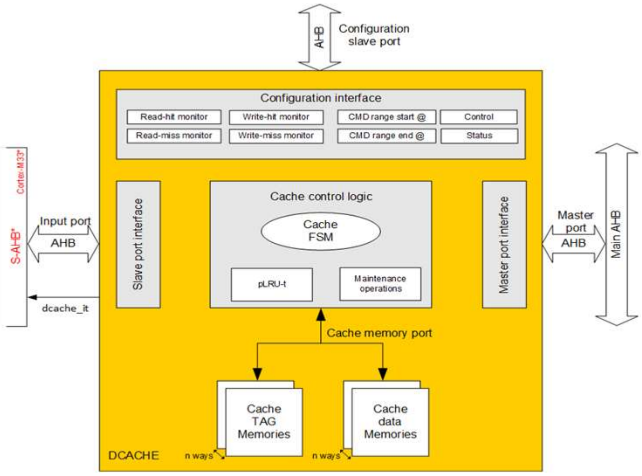


The multi-bus interface minimizes potential conflicts regarding memory traffic:

- The 32-bit data slave port, receives instruction and data memory requests from the Cortex®-M33 S-AHB System bus, for DCACHE1, and from M0 port of GPU2D for DCACHE2
- The 32-bit master port performs refill of missing requests from memories, dirty data write-back or data write-through to external memories. These memories are external FLASH & RAMs, accessed through OctoSPIs, HSPI1 and FMC controllers.
- The second slave port is used for registers access.

When an external memory access is marked as non-cacheable by the MPU, DCACHE1 is bypassed. The request is forwarded unchanged to the external memory on the DCACHE1 master port, in the same clock cycle.

DCACHE1/2 key features (2)



- For DCACHE2, Fata slave port interface is M0 port from GPU2D

Close to zero wait-state data access performance in any external memories:

- 0 wait-state on cache hit
- Hit-under-miss capability
- Critical-word-first refill policy, minimal processor stalls on cache miss
- Hit ratio improved by 2-way set-associative architecture and PLRU-t replacement policy
- Optimal cache line refill thanks to AHB WRAP bursts (aligned on cache line size)
- Write-back and write-through policies
- Byte, half-word and word write operations

life.augmented

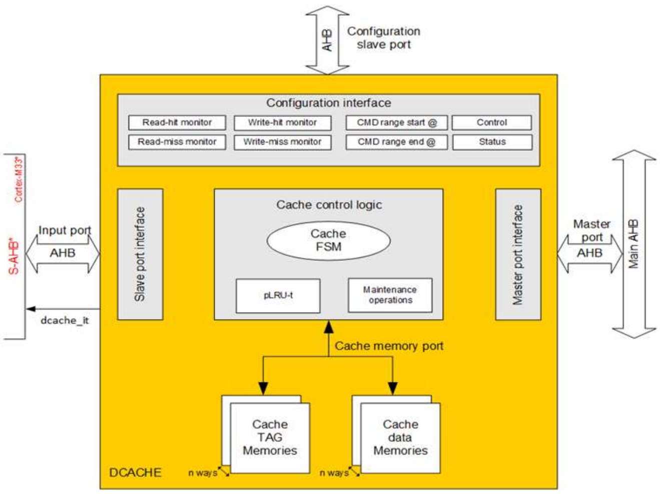
4

DCACHE1/2 offers close to zero wait states data read/write access performance, due to:
- Zero wait-state on cache hit
- Hit-under-miss capability, that allows to serve new processor requests while a line refill (due to a previous cache miss) is still ongoing;
- And critical-word-first refill policy, which minimizes processor stalls on cache miss.

The hit ratio is improved by:
- The 2-way set-associative architecture and
- The pseudo-least-recently-used, based on binary tree (or pLRU-t) replacement policy. This algorithm is a good tradeoff between hardware complexity and performance.

Cache lines are transferred with the critical word first, by implementing the WRAP4 AHB transaction ordering, in order to deliver the data requested by the processor’s pipeline first. Write-back and write-through policies are supported, selection depends on the MPU setting for the addressed data region. DCACHE1/2 supports all data sizes: byte, halfword and word.

DCACHE1:2 key features (3)



- Performance monitoring:
  - Read hit + read miss counters
  - Write hit + write miss counters
- Power consumption reduced by accessing DCACHE rather than external memories
- TrustZone® security support only for DCACHE1
- Maintenance operations to manage cache coherency by software:
  - Full invalidate,
  - Invalidate/ clean-and-invalidate/clean a programmable address range
- Error management (error flag, with optional interrupt raising): detection of errors returned to master port requests initiated by DCACHE (line eviction or clean).

- For DCACHE2, Fata slave port interface is M0 port from GPU2D

life.augmented

5

DCACHE1/2 implements performance counters: two 32-bit hit counters, one for read and one for write transactions, and two 16-bit miss counters, one for read and one for write transactions.
This performance monitoring allows to analyze and optimize data placement in accordance with cacheability and write-back/write-through policy to achieve the most performant data traffic.
Power consumption is reduced: most data accesses are performed to/from internal cache memory rather to/from bigger and external main memories.
A dedicated Secure-bit in TAG RAM of each cache line, prevents non secure requests from hitting secure DCACHE1 entries.
As the GPU2D traffic is non-secure, DCACHE2 does not

support TrustZone.
Software cache coherency is performed through maintenance operations controlled by memory-mapped registers.
These are:

- The full cache invalidation operation, which is a fast command
- The invalidate, clean-and-invalidate and clean operations that are related to a programmable address range.

The data cache is automatically invalidated after a reset.
The address range maintenance operations, are typically used to maintain the coherency of buffers shared by DMA channels and the processor core.
These commands are not interruptible, with an end of operation raising a specific flag and possibly an interrupt.
An error flag and possibly an interrupt, are raised whenever a bus error is returned to the master port of DCACHE, when the request is initiated by DCACHE itself: either a line eviction or a clean operation.
When the master port forwards a request received on the slave port, DCACHE simply forwards the AHB response received on the master port back to the processor.

SUMMARY

Cache line size*	16 or 32 bytes	
Cache size*	4 or 16 KB	
Organization*	2-way or 4-way set associative	
Write & allocate policies	Write through no write allocate	Write-back write allocate
Maintenance operations	Global	Per address range
Clean		✓
Clean & invalidate		✓
Invalidate	✓	✓

- Depends on product. Please refer to the datasheet

life.augmented

This table summarizes the characteristics of the data cache (depending on the product):

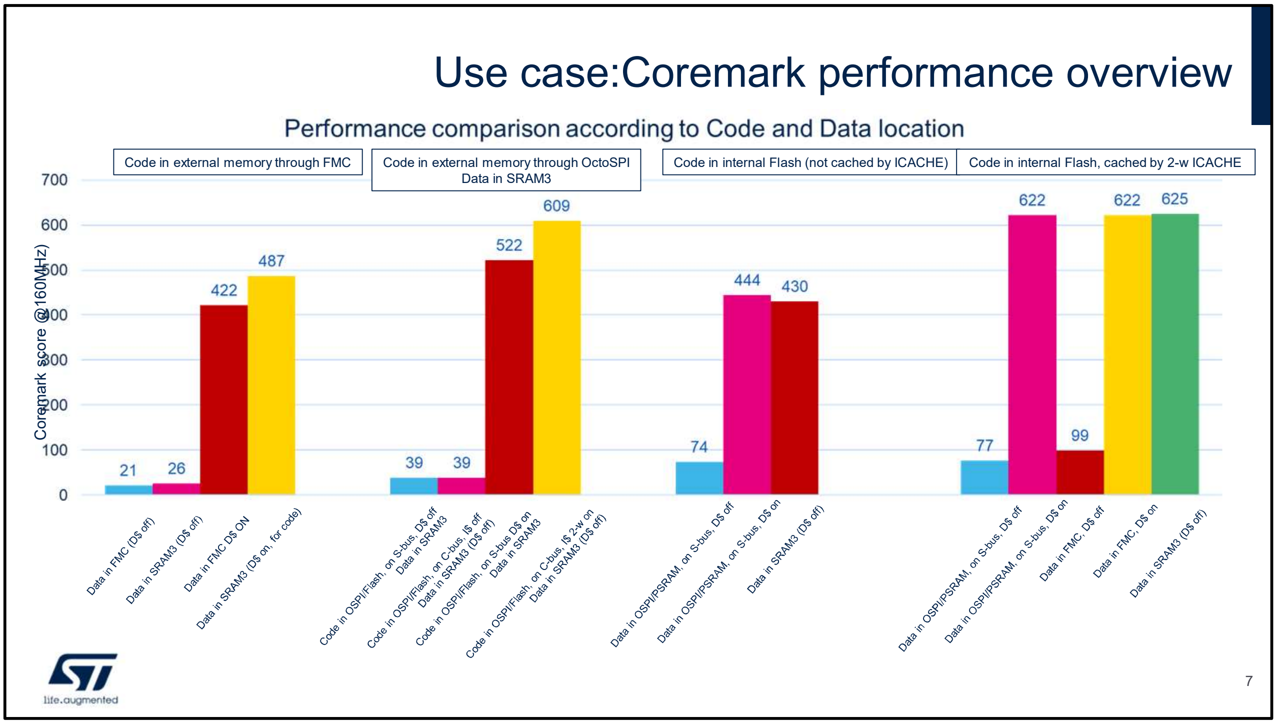
- 16 or 32-byte cache line size, transferred using a burst of four or eight words
- 4 or 16-KiloByte cache
- 2-way or 4-way set associative.

The data cache implements the following <u>write and allocate</u> policies:

- Write-through no write allocate. When a store miss occurs, DCACHE1/2 is bypassed, the data is directly written to memory
- Write-back write allocate. When a store miss occurs, the cacheline is acquired from memory, updated with data received from the processor. The resulting cacheline is then written to the data cache, with the dirty bit set.

The supported maintenance operations are:

- Invalidate: global and per address range
- Clean and invalidate, per address range
- And Clean, per address range.



This slide demonstrates the influence of data cache and instruction cache on performance, expressed in Coremark, when the processor core frequency is 160 MHz.
Four scenarios are described, for which the location of data and instructions varies as well as their cacheability in ICACHE and DCACHE1.

<u>In the first case, code is in external memory, accessible through the FMC on S-AHB bus, ICACHE is not involved</u>
Lowest performance is obtained when code and data are accessed through the FMC, not cached by DCACHE1, and therefore code and data are multiplexed on the S-AHB bus . This is slightly better when data is in the SRAM3, but still transferred on the S-AHB which is also used to fetch code from the FMC.

When DCACHE1 is enabled, the performance increases a lot, but is still limited by code and data sharing, both on the S-AHB bus and in the DCACHE1.
The best performance, which is 487 coremark, is achieved when data is in the SRAM3: code and data still share S-AHB bus, but the DCACHE1 is dedicated to code storage only.
Code accessible through the FMC could advantageously be cached by ICACHE, through ICACHE address remapping, which would split code and data traffics between C-AHB and S-AHB buses.

In the second case, code is in external memory, accessible through the OctoSPI on the S-AHB bus or on the C-AHB bus
Performance is low when data is in the SRAM3 and code is in the OctoSPI External Flash, accessible through the S-AHB bus and not cached by DCACHE1.
Performance is low also when code is in the OctoSPI External Flash accessible through the C-AHB bus and not cached by ICACHE.
In both cases, the score in coremarks is only 39.
The performance is drastically increased when the code accessible through the S-AHB bus is cached by the DCACHE1, but the S-AHB bus remains a bottleneck, because it is used to transport both code and data.
Almost optimal performance is achieved, when the code accessible through the C-AHB bus, is cached by ICACHE. This requires the implementation of address remapping in the ICACHE. In this case, code is transferred over the C-AHB bus while data is transferred over the S-AHB bus.

In the third case, code is stored in the internal Flash, not cached by ICACHE
When data is stored in OctoSPI PSRAM, the performance is

low when data is not cached.
When data is cached in the DCACHE1, the performance increases a lot and becomes even better than having the data in SRAM3: 444 Coremark instead of 430.

In the last case, code is in internal Flash, cached by ICACHE configured in 2-way set associative mode.
Performance is low when data stored in OctoSPI PSRAM is non cacheable and a bit better when data is stored in an SRAM accessible through the FMC but still non cacheable.
In both cases, marking the address range containing this data as cacheable, leads to the same good performance of 622 coremark.
The best performance is obtained by having data in SRAM3 and instructions in cacheable internal flash.

DCACHE errors and interrupts

Interrupt vector	Interrupt event	Event Flag	Interrupt Enable bit	Interrupt Clear bit	Description
DCACHE	Functional Error	DCACHE_SR[ERRF]	DCACHE_IER[ERRIE]	DCACHE_FCR[CERRF]	Error on master port transaction initiated by DCACHE itself (eviction or clean operation)
	End of Busy State	DCACHE_SR[BSYENDF]	DCACHE_IER[BSYENDIE]	DCACHE_FCR[CBSYENDF]	When cache-busy state is finished, at the end of the cache full invalidate operation
	End of Cache Operation	DCACHE_SR[CMDENDF]	DCACHE_IER[CMDENDIE]	DCACHE_FCR[CCMDENDF]	When command-busy state is finished, at the end of the cache range operation (invalidate and/or clean)

DCACHE does not manage AHB bus errors on Master port transactions that result from a data request on the slave port (that received the initial Core S-AHB bus transaction), but propagates these errors back to the Slave port

life.augmented

8

All DCACHE interrupt sources raise the same and unique interrupt signal, dcache_it, and then use the same interrupt vector.

The three sources of DCACHE global interrupt are:

- Error detection on data request initiated by the DCACHE itself, either for dirty cache line eviction or clean operations, which sets the ERRF bit in the DCACHE status register
- End of full Invalidate operation, which sets the BSYENDF bit in the DCACHE status register
- End of cache range maintenance operation (invalidate, clean & invalidate, or clean), which sets the CMDENDF bit in the DCACHE status register.

DCACHE also propagates all AHB bus errors (such as security issues, address decoding issues) from master port back to the S-AHB slave port. A typical case is an erroneous refill request, initiated by an initial data request, that misses in cache.

Low-power modes

Low power mode	State of the DCACHE
Run	Active
Sleep	Active
Stop	Frozen, DCACHE registers content is kept Option: a dedicated control bit in Power Controller to power-down DCACHE in Stop mode
Standby	Powered-down The peripheral must be reinitialized after exiting Standby mode

When disabled, DCACHE1/2 are bypassed and internal TAG and Data memories are not accessed

- Almost no power consumption in DCACHE, with the drawback that each data is read from/written to the more power consuming main (external) memory

life.augmented

At product level, using the DCACHE1/2 reduces the power consumption by loading/storing data from/to the internal DCACHE most of the time, rather than from the bigger and then more power consuming main memories. This reduction is even much higher, since the cached main memories are external.
As a result, the DCACHE1/2 and their masters have the same state in the various low power modes.
When the microcontroller is in stop mode, the user can decide to power-down the DCACHEs, which may require a complete clean and invalidate maintenance operation.
When DCACHE is disabled, it is bypassed. The system bus input requests are just forwarded to the master port.

Thank you

life.augmented

In addition to this presentation, you can refer to the following presentations:

- Instruction cache
- Security
- FMC
- OCTOSPI.