AN14139

Optimizing Performance on MCX N-Series MCUs

Rev. 1.0 — 20 January 2024 Application note

Document information

Information	Content
Keywords	AN14139, MCX, performance, cache, MCXN54x, MCXN94x
Abstract	This application note explains the features of MCX N-series devices that can affect system performance.

1 Introduction

In embedded systems, resources are often limited and getting the best possible performance out of those resources can be critical. Although high performance and low power can seem contradictory, entering a low-power mode after completing a task quickly can reduce system power consumption. Therefore, almost any system can benefit from efforts to improve performance.

Increasing performance for an embedded system can be a complicated task. Often the nuances of the inner workings of the architecture and features can impact the system. In addition, every system can have different performance goals. For instance, one system can prioritize CPU performance, while another can prioritize optimizing throughput for communication ports like Ethernet or USB.

This application note explains the features of MCX N-series devices that can affect system performance. The document is not a step-by-step guide on optimizing an application as there are no hard rules that work for all cases. By explaining key architectural and system and module features, this document allows you to make informed decisions for your system hardware and software.

2 MCX N-series architecture overview

The system architecture is one of the biggest factors in the overall system performance. How the different blocks fit together also has an impact on some of the module-level features. So, the first step to understanding how to optimize system performance is understanding the architecture from a high level.

Figure 1 shows a simplified block diagram of the MCX N94x family device. This family is selected because it shows the superset for the performance features that are discussed in further sections. The other MCX N-series devices do not have the same features, but in general, the overall architecture is largely the same.

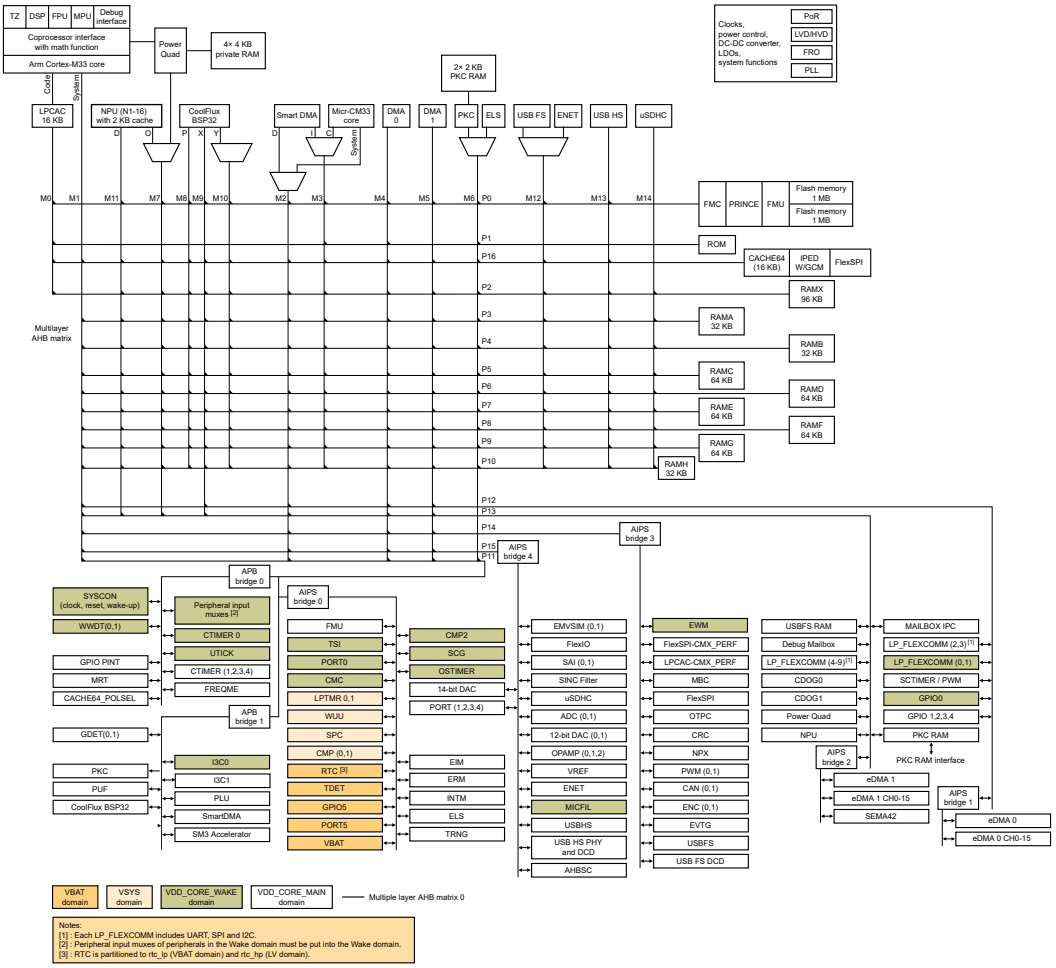

Figure 1. Bus matrix block diagram

2.1 Core buses on MCX

The Arm Cortex M33 core uses a pseudo-Harvard architecture with several memory-mapped buses:

- Code - The code bus is used to access addresses `0x0000_0000-0x1FFF_FFFF`. As the name implies, it is used for instructions; however, data can also be accessed on this bus.
- System - The system bus is used for all accesses to addresses between `0x2000_0000-0xDFFF_FFFF` and `0xE010_0000-0xFFFF_FFFF`.
- Private peripheral bus - The private peripheral bus (PPB) is mapped to addresses `0xE004_0000-0xE00FFFFF`.

The performance is the same for both the code and system bus. NXP recommends using the code bus for instructions and the system bus for data. Using both the code and system bus allows the core to access instructions and data in parallel.

2.2 MCX N-series memory map

To maximize code bus usage by applications, key memory regions have been included in the MCX system memory map at addresses below `0x2000_0000`. Whenever possible, code bus regions should be used for storing instructions.

Table 1 shows a simplified memory map with regions for the non-secure code bus on the MCX Nx4x devices. These memory regions include an aliased region for the optional FlexSPI controller. Normally the FlexSPI region is in the system bus portion of the memory map. The aliased region has been added so that the memory is also available on the code bus. This allows for the most efficient performance when executing code from external memory and enables the use of the LPCAC for the FlexSPI.

Table 1. Simplified MCX Nx4x code bus memory map

Start address	End address	Description	Size
0x0000_0000	0x001F_FFFF	Program Flash	2 MB
0x0300_0000	0x0303_FFFF	ROM	256 KB
0x0400_0000	0x0401_7FFF	RAMX	96 KB
0x0800_0000	0x0FFF_FFFF	FlexSPI alias	128 MB

Note: *The boot ROM does support booting from external flash on the FlexSPI. However, the ROM expects the initial PC in the vector table for a FlexSPI boot image in the FlexSPI system bus address range. The check is only done on the PC in the vector table. The image itself can be linked to run from the aliased FlexSPI code bus region.*

The on-chip flash memory is only accessible on the code bus. Typically, the flash is primarily used to store instructions. If a large amount of data stored in a flash is accessed regularly, it can be beneficial to copy the data to internal RAM.

On-chip RAMs are instantiated as multiple blocks where the RAMX block is mapped to the code bus, and RAMA-RAMH are mapped to the system bus. RAMX is used for storing RAM code, and RAMA-RAMH are intended for data storage.

Note: *The number and size of the RAM blocks vary depending on the specific N-series device.*

3 MCX memories and caches

The following sections discuss how the usage and configuration of memories and caches can impact performance.

3.1 Flash

MCX N-series devices include up to 2 Mbytes of on-chip flash memory. The flash is the primary location for code and non-volatile data.

3.1.1 Wait states

Usually, the flash access time requires adding wait states to the on-chip flash access. The `FMU_FCTRL[RWSC]` value configures the wait states for the flash. Table 2 shows an example of minimum wait state values for different voltage and frequency configurations. To see the wait state requirements of the flash for your MCX device, refer to the chip-specific information section of the FMU chapter in the device reference manual.

By default, the flash wait states are configured with a value that supports the maximum frequency of the device. For the device shown in Table 2, the flash is configured for three wait states by default. If a lower frequency is used, the wait state value must be reconfigured to improve performance. For example, if the maximum frequency for the system is 100 MHz, RWSC can be reconfigured for two wait states instead of the default three.

Table 2. FMU_FCTRL[RWSC] minimum values based on frequency and mode

Voltage mode	Overdrive (1.2 V)	Normal voltage (1.1 V)	Mid voltage (1.0 V)	FMU_FCTRL[RWSC]
Frequency	101 MHz to 150 MHz	--	--	0011 (three wait states)
	65 MHz to 100 MHz	65 MHz to 100 MHz	--	0010 (two wait states)
	37 MHz to 64 MHz	37 MHz to 64 MHz	25 MHz to 50 MHz	0001 (one wait state)
	<= 36 MHz	<= 36 MHz	<= 24 MHz	0000 (single cycle access)

Note:

The flash memory controller includes features to minimize the exposure of the wait states when accessing flash. These features are described in the following sections. However, the flash wait states must always be configured to at least the minimum value based on the current voltage and frequency conditions.

The SDK clock drivers do not automatically change flash wait states. When changing the voltage or bus frequency for the system, the application must ensure that the flash wait states are set to a valid and optimized value.

3.1.2 Flash memory controller (FMC)

The FMC manages accesses performed by the bus managers of the system to the flash memory. The FMC accelerates flash memory transfers to allow program code execution at a higher clock frequency than flash memory.

The FMC provides two separate mechanisms for accelerating read operations to the flash memory:

- 128-bit prefetch buffer, which can prefetch the next 128-bit flash memory location.
- 64-byte cache is organized as a one set, four-way associative cache with 128-bit (or 16-byte) size entries.

Note: *The flash memory module (FMU) directly manages erase and program cycles. Because these cycles bypass the FMC, its cache and buffers have no visibility to flash, erase, and program operations. Software is required to maintain memory coherence when any segment of the flash cache is programmed. For example, all buffer data associated with the reprogrammed flash must be invalidated.*

The speculation logic is tuned to work with the flash cache enabled. The speculation logic assumes that the data is moved to the cache when any access hits the speculation buffer. The speculation buffer immediately requests the next sequential flash phrase. If the flash cache is disabled, the speculation logic still moves to the next sequential flash phrase. If additional data within the first flash phrase is accessed again, then it must be read directly from the flash (cache is disabled and a speculation buffer has moved to the next phrase). For best performance, enable the flash cache whenever the speculation logic is enabled.

3.1.2.1 Flash prefetch buffer

When speculative reads are enabled, the FMC immediately requests the next sequential address after a read completes. The next 128-bit memory location is read. The speculative prefetch mechanism improves performance by reducing or even eliminating wait states when accessing sequential code and/or data.

The FMC provides invalidation control for the prefetch buffer but the `NVM_CTRL` register of SYSCON is used to enable and configure speculative prefetching. While the `DIS_DATA_SPEC` and `DIS_FLASH_SPEC` bits of `NVM_CTRL` are cleared by default, the operation of these bits interacts with the `DIS_MBECC_ERR_DATA` and `DIS_MBECC_ERR_INST` bits. `DIS_MBECC_ERR_DATA` is set by default, which disables the flash speculation even though `DIS_FLASH_SPEC` is cleared. For best performance, the `SYSCON->NVM_CTRL[DIS_MBECC_ERR_DATA]` bit must be cleared early in the startup code.

3.1.2.2 Flash cache

Cache memory stores already-fetched data. This code is immediately available for repeated execution without any wait states. The FMC provides controls for flash replacement algorithm, lock per way, and invalidation per way. The ways are numbered from 0 to 3 and the sets are numbered from 0 to 3. The cache supports the least recently used (LRU) replacement algorithm per set across all 4 ways. The `NVM_CTRL` register of SYSCON is used to enable/disable the cache and set other configurations. The flash cache is enabled by default.

3.1.3 Internal flash data throughput

Figure 2 shows internal flash data throughput measurements for MCX Nx4x. The setup used for the measurement:

- The system clock is 150 MHz.
- The measurements show the effect of FMC acceleration features, so the LPCAC is turned off for all testing.
- The flash wait states (`FMU_FCTRL[RWSC]`) is set to three.

	System clock [MHz] 150	Instruction	SRAM-A code execution		
			Data size [B]	Clocks [-]	Throughput [MB/s]
Flash cache OFF Flash prefetch buffer OFF Not recommended ✗	Flash - linear data	LDR 4B	160	202	118,81
	Flash - non-linear data	LDR.W 4B	160	198	121,21
	Flash - linear data	LDM 16B	160	202	118,81
	Flash - linear data	LDM 32B	320	377	127,32
Flash cache ON Flash prefetch buffer OFF Not recommended ✗	Flash + cache 64B - linear data	LDR 4B	160	82	292,68
	Flash + cache 64B - non-linear data	LDR.W 4B	160	198	121,21
	Flash + cache 64B - linear data	LDM 16B	160	82	292,68
	Flash + cache 64B - linear data	LDM 32B	320	158	303,80
Flash cache ON Flash prefetch buffer ON Recommended ✓	Flash + cache 64B + speculation - linear data	LDR 4B	160	62	387,10
	Flash + cache 64B + speculation- non-linear data	LDR.W 4B	160	312	76,92
	Flash + cache 64B + speculation - linear data	LDM 16B	160	62	387,10
	Flash + cache 64B + speculation - linear data	LDM 32B	320	120	400,00
Flash cache OFF Flash prefetch buffer ON Not recommended ✗	Flash + speculation - linear data	LDR 4B	160	67	358,21
	Flash + speculation- non-linear data	LDR.W 4B	160	319	75,24
	Flash + speculation - linear data	LDM 16B	160	67	358,21
	Flash + speculation - linear data	LDM 32B	320	131	366,41

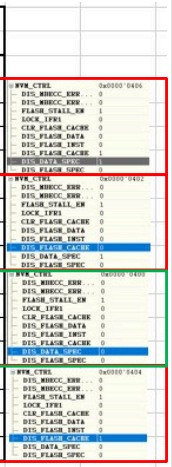

Figure 2. Internal flash data throughput measurements for MCX Nx4x

3.2 On-chip RAM

MCX N-series devices include multiple blocks of on-chip RAM. The number and size of the RAM blocks vary depending on the specific part number and device configuration. Typically, RAMs are accessible with no wait states, but ECC and auto clock gating can increase the RAM access time.

RAMX (up to 96 KB) is connected to the CM33 code buses. RAMX is the preferred RAM block to use for code storage.

RAMA, which is always four 8 KB banks (32 kB total), is the preferred RAM block to use for data retention. The RAMA banks can be retained in device low-power modes. It can optionally be powered from VBAT using `LDO_RAM`. To optimize power consumption, RAMA is split into four banks, where the low-power mode and VBAT retention for each bank is individually programmable. The VBAT module controls the low-power configurations (`LDO_RAM` enable/disable and bank retention).

The other RAM blocks and partitions (other than RAMA) all have independent power switches that can be turned on/off depending on the RAM needs of the application. `CMC_SRAMDIS0` can be used to completely power gate a RAM partition (applies for all power modes). `CMC_SRAMRET0` can be used to turn off the periphery of RAM partitions while retaining the contents of those RAMs during low-power modes.

Note: *Although the RAMA-RAMH blocks are contiguous in the system memory map, each block uses a different physical AHB port. This means that misaligned or burst accesses across the boundary from one RAM block to another are not allowed.*

3.2.1 RAM ECC

RAMA supports software configurable ECC (enabled by default). Each 8 KB RAM bank supports 32+7 ECC, which provides one-bit correction and two-bit detection capability.

ECC is also supported for the other RAM blocks. To determine which RAM blocks have ECC enabled by default, refer to the device reference manual.

Note: *For blocks other than RAMA, ECC is implemented by repurposing upper RAM blocks to provide ECC data for lower blocks. Therefore, enabling ECC can also affect the overall number of RAM blocks that are accessible on the device.*

Because the ECC is implemented as 32+7, ECC implements a read-modify-write mechanism to support 16-bit and 8-bit writes. 16-bit and 8-bit writes to an ECC-enabled RAM take two additional clocks. Reads have no penalty because the ECC code only changes on a write. For best performance, variables in ECC memory must be 32-bit. If 16-bit or 8-bit data types are necessary, then consider storing them in a RAM where ECC is not going to be used.

3.2.2 RAM auto clock gating

To reduce power consumption, the RAM blocks support an auto clock gating feature. When auto clock gating is enabled, the clock to the RAM block is automatically gated off if the block is not accessed for 16 bus clocks. If the clock is off, there is a one-bus cycle delay for the next access to the RAM block.

The auto clock gating feature is configurable on a per RAM block basis. The `AUTOCLKGATEOVERRIDE` and `AUTOCLKGATEOVERRIDEC` registers of the SYSCON configure the auto clock gating function. To determine which, if any, RAM blocks have auto clock gating enabled by default, refer to the device reference manual. The auto clock gating feature should be disabled for RAM blocks that are used for code/data sections that require time-critical or deterministic execution.

3.3 LPCAC

The 16 KB low-power cache controller (LPCAC) is connected to the code bus of the primary M33 core (CPU0). The content of this cache is only visible to CPU0. The LPCAC can be used to cache M33 access to the program flash (`0x0000_0000-0x001F_FFFF` and `0x1000_0000-0x101F_FFFF`) and FlexSPI (`0x0800_0000-0x0FFF_FFFF` and `0x1800_0000-0x1FFF_FFFF`) code bus memory regions.

The LPCAC chapter in the device reference manual provides the functional description of the cache, but the LPCAC Control (`LPCAC_CTRL`) register of the SYSCON is used to control the operation of the cache. Because the LPCAC is on CPU0's code bus, it is mostly intended for caching instructions. It can be used for data but only supports a single cacheable, write-through mode that is used for all memory regions when enabled (no address regions with different cache policies and no write-back/copy-back mode).

The LPCAC is disabled by default. For best performance, NXP recommends enabling it by clearing `SYSCON->LPCAC_CTRL[DIS_LPCAC]`.

3.4 FlexSPI subsystem

Some MCX N-series devices include a FlexSPI subsystem supporting Octal and Quad SPI memory devices. The FlexSPI is primarily intended for execute-in-place code execution from off-chip SPI NOR flash memory. The FlexSPI also supports external serial RAM expansion.

The FlexSPI subsystem also includes a 16 KB cache with a CACHE64 AHB-cache controller.

3.4.1 CACHE64

The 16 KB cache controller (CACHE64) of FlexSPI is interfaced directly with the FlexSPI memory controller. As the CACHE64 is integrated into the controller, access to the FlexSPI from any manager can be cached, provided the corresponding address is configured as cacheable.

The CACHE64 policy select module (CACHE64_POLSEL) is used to define the cache policy (non-cacheable, write-thru cacheable, or write-back cacheable) for up to three regions in the FlexSPI. On a FlexSPI access CACHE64_POLSEL determines the cache policy to use for that address, and then passes the information to CACHE64_CTRL.

There are multiple FlexSPI regions in the system memory map. But these regions are remapped to create a single contiguous 512 Mbyte region (0x8000_0000-0x9FFF_FFFF), where accesses to some system regions are aliased into a single region for the cache. Table 3 shows how the FlexSPI system regions are mapped to the CACHE64_CTRL and CACHE64_POLSEL.

Table 3. FlexSPI memory map region mapping to CACHE64_CTRL and CACHE64_POLSEL

System start address	System end address	Size	CACHE64 start address	CACHE64 end address	Access
0x0800_0000	0x0FFF_FFFF	128 MB	0x8000_0000	0x87FF_FFFF	Non-secure
0x1800_0000	0x1FFF_FFFF	128 MB	0x8000_0000	0x87FF_FFFF	Secure
0x8000_0000	0x8FFF_FFFF	256 MB	0x8000_0000	0x8FFF_FFFF	Non-secure
0x9000_0000	0x9FFF_FFFF	256 MB	0x8000_0000	0x8FFF_FFFF	Secure
0xA000_0000	0xAFFF_FFFF	256 MB	0x9000_0000	0x9FFF_FFFF	Non-secure
0xB000_0000	0xBFFF_FFFF	256 MB	0x9000_0000	0x9FFF_FFFF	Secure

The CACHE64_CTRL defaults to disabled. If external memory on FlexSPI is being used, then NXP recommends enabling CACHE64:

1. Configure the three CACHE64 regions as needed using the CACHE64_POLSEL registers.
2. Write 0x8500_0003 to the CACHE64_CTRL0->CCR. This value requests a full invalidation of the cache and enables the cache and write buffer in a single operation.

3.4.2 FlexSPI controller prefetch buffer

In addition to CACHE64, the FlexSPI controller has a prefetch buffer within it that can also help to increase system performance. When the external FlexSPI memory is accessed, the FlexSPI fetches a prefetch buffer worth of data sequentially from the requested address. When the next access is sequential or close to the original address, the data is likely to be stored in the prefetch buffer. Therefore, the FlexSPI controller can retrieve the data without initiating a new access to the external memory. On MCX, the FlexSPI has a total of 1 KB memory for prefetch. The 1 KB buffer can be divided into 8 individual prefetch buffers with configurable sizes where each buffer is assigned to a specific bus manager.

The FlexSPI controller on MCX has a new feature called prefetch buffer enhancement. This feature can be useful for increasing performance if a given manager frequently accesses memory at multiple FlexSPI address areas. To illustrate, suppose that the CPU is carrying out task A at a particular address. After that, it switches to task B at a different address. A new prefetch operation begins at the address pertaining to task B. When the CPU switches back to task A, the prefetch buffer is flushed again as the buffer now holds task B addresses. This results in task A addresses missing in the prefetch buffer. The buffer then starts prefetching at the task A address again. If task A and task B switch frequently, FLEXSPI can re-read previously stored items in the prefetch buffer due to frequent buffer flushing.

The prefetch buffer enhancement feature helps with task switching, by allowing multiple prefetch buffers to be assigned to a single manager/manager ID. When prefetch buffer enhancement is enabled, an address range is added to differentiate between multiple prefetch buffers assigned to the same manager ID. In this example, you can assign two buffers to the main CPU, with one mapped to task A address range, and another mapped to task B. This way both tasks can be stored in the prefetch buffer and decrease the number of times the external memory is accessed.

4 System bus access and arbitration

The multilayer AHB matrix is the primary bus interconnector for the microcontroller. It manages connections between bus managers, subordinate ports, and arbitrates access conflicts.

4.1 AHB accesses

Simultaneous accesses from different managers is allowed if they access different subordinate ports. Careful planning of the memory usage by managers in a system can yield a significant increase in the overall system performance.

For example, here is a possible system memory configuration:

- Main M33 core (CPU0) - Instructions in flash and core-only data and stack in RAMA-RAMC
- Secondary Micr-M33 core (CPU1) – Instructions in SRAMX and core-only data and stack in RAMD
- USB - Data buffers in RAME

This memory configuration allows all three of the managers to run the bus cycles they need with little interference from other managers. Occasionally, one of the cores may need to access the USB buffers. However, outside these accesses, the managers can run in parallel.

4.2 AHB arbitration

If multiple managers attempt access to the same subordinate port at the same time, then arbitration is required. The `AHBMATPRIO` register of the SYSCON is where the programmable priorities for each of the manager ports can be configured. Managers are assigned a priority value between zero and three with three being the highest priority. If two ports have the same priority, then the lowest port number is given priority.

The `SYSCON_AHBMATPRIO` should be configured according to the requirements of the system. For example, if the DMA is being used to read a SPI receive buffer, it might make sense to give the DMA higher priority than the primary core. This can help to avoid SPI receive buffer overflows in a heavily loaded system.

There are some manager ports that are shared between two managers. Where the port is shared, only one of the managers can have an active access at a time. The priority between two managers sharing a port uses a fixed arbitration scheme. To determine which ports can be shared and the priority used for those ports, refer to the Memory chapter in your device reference manual.

Note: *Arbitration only happens when there is more than one pending request to access a subordinate port.*

If a low-priority manager makes a request to an idle subordinate port, then the low-priority manager gets to start its bus cycle. If a higher-priority manager requests access to the bus while a low-priority manager is already using it, the low-priority manager must finish its bus cycle before the high-priority manager gains access to the bus. For fixed-length bursts, the transfer boundary is at the end of the bus cycle.

5 Multi-core considerations

Some MCX N-series devices are dual-core devices with two Arm Cortex-M33 cores. The primary core (CPU0) includes TrustZone-M, floating point unit (FPU), DSP, and memory protection unit (MPU). The secondary core

(CPU1) is a micro-CM33, which does not include the TrustZone-M, FPU, or DSP. The functionality available for each core must be considered when deciding what tasks can be offloaded to the secondary core.

The device always boots using the primary core. To configure CPU1 and release it from reset for dual-core functionality, perform the following steps:

1. Optionally, copy the CPU1 application to the target memory address.
2. Write `SYSCON->CPBOOT` with the CPU1 VTOR address.
3. Clear the `SYSCON->CPUCTRL[CPU1RSTEN]` bit to release CPU1 from reset. Make sure to keep the `CPU_CLKEN` set when writing the `CPUCTRL` register.

The system memory usage must be considered when using CPU1. CPU1 does not include an LPCAC like CPU0. CPU1 can still benefit from the acceleration features within the FMC, but CPU1 does not reach its maximum performance when executing from internal flash. If CPU1 is mostly offloading CPU0, the maximum performance for CPU1 might not be required for the system. If CPU0 is executing from its LPCAC, then CPU1 can access the flash without conflict. So, it is possible to have both cores using internal flash code addresses entirely or largely without creating bus arbitration delays.

If maximum performance for CPU1 is needed, then RAMX is the recommended memory to use for CPU1 code. This does assume that CPU0 is not using RAMX, in which case another location can be used. For devices that include FlexSPI, external flash is another option for the CPU1 code location. CPU1 can use `CACHE64` included in the FlexSPI subsystem and have mostly zero wait state execution.

6 Summary

- Identify the system priorities. Some optimizations help increase overall performance, but many optimization options create a trade-off where performance is gained in one area and lost in another. Clear optimization goals are a must.
- Plan data movements and code locations in advance. Not all memory addresses are created equal. Be aware of the bus ports that are used for each access.
- When available, consider offloading some tasks to CPU1.
- Take advantage of the flash acceleration features built into the flash memory controller (FMC). Clear the `SYSCON->NVM_CTRL[DIS_MBECC_ERR_DATA]` bit to enable the flash speculation buffer fully.
- Use the LPCAC cache. The cache hits are as fast as storing code/data in the on-chip SRAMs.
- Use the SRAMX block for storing critical code. Use other RAM blocks for data and stack.
- Avoid 16-bit and 8-bit writes to ECC-enabled RAMs. The read-modify-write operation required to maintain the correct ECC value adds clocks to the access.
- Enable auto clock gating to all RAM blocks for the best power consumption. Disable auto-clock gating to RAMs storing critical code/data or if determinism is required.
- If external memory on the FlexSPI is used for code, use the aliased memory regions on the code bus for accessing instructions.
- If using external memory on the FlexSPI, then use the CACHE64 controller. The cache hits are as fast as storing code/data in the on-chip SRAMs.
- Use code optimizations wisely. Compilers usually offer a choice of optimizing for speed or size. Optimizing for speed is often the best option for performance, but that is not always the case. If optimizing for size allows to fit functions more easily in the cache, then performance might be best using size optimizations. Experiment with the switches that are available to find the optimal compiler settings.
- Parallelism is the best way to increase overall system performance. Take advantage of the multilayer AHB matrix and its ability to have concurrent, non-blocking transfers.
- When moving large blocks of data, use the DMA. The DMA can transfer data more efficiently than the cores often. Using the DMA also frees up the cores to perform other tasks (more parallelism).
- Do not forget to look at the multilayer AHB matrix arbitration settings. Some experimentation can be needed to find the best configuration.

7 Revision history

Table 4 summarizes the revisions done to this document.

Revision history

Document ID	Revision date	Description
AN14139 v.1.0	20 January 2024	Initial public release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

For more information, please visit: https://www.nxp.com

Date of release: 20 January 2024
Document identifier: AN14139