C71x DSP Corepac Technical Reference Manual

User's Guide

Literature Number: SPRUIQ3
April 2020

Contents

Copyright © 2020, Texas Instruments Incorporated

Copyright © 2020, Texas Instruments Incorporated

List of Figures

List of Tables

Copyright © 2020, Texas Instruments Incorporated

Copyright © 2020, Texas Instruments Incorporated

Copyright © 2020, Texas Instruments Incorporated

Copyright © 2020, Texas Instruments Incorporated

1 Memory System and Caches

This section describes the Corepac memory system, which consists of one or more instances of the L1 Program Cache (L1P), L1 Program Memory Controller (PMC), L1 Data Cache (L1D), L1 Data Memory Controller (DMC), L2 Unified Cache (L2), and the Unified Cache Controller (UMC). Architectural features that have changed substantially from previous Corepac generations have been summarized. This list also highlights important features of the memory system. Internal and external interfaces have been described. The error correction feature has been described at the end of this section. Separate sections are included for each of the memory controllers for detailed description of their architectures. Additional sections describe the interface protocol and cache coherence.

1.1 Introduction

The Corepac memory system is the first generation of the caches and memory controller system for the TMS320C7x fixed and floating point DSP. It supports 1 to 2 C7x CPU cores, each with its private L1 caches and controllers, Memory Management Unit (CMMU) and a streaming engine. The L2 cache controller is shared by the cores. The Corepac memory system can provide bandwidth of up to 2048-bits of data per cycles, which is an 8x bandwidth improvement over previous generations. The L1D can sustain 512 bits of data to the CPU every cycle, while the L2 can provide 1024 bits of data to the streaming engine every cycle. The L1 and L2 controllers have the ability to queue up multiple transactions out to the next level of memory, and can handle out of order data return. The L1P controller supports branch exit prediction from the CPU and can queue up multiple prefetch misses to L2.

The Corepac memory system has full soft error correction (ECC) on its data and TAG rams. A novel ECC scheme implemented in the C7x covers many pipeline and interface registers in addition to memories. This results in an improvement in the FIT (Failure in Time) rate from soft error.

This memory system supports full memory coherency, where all the internal data caches and memories (L1D, L2) are kept coherent to each other and to external caches and memories (MSMC, L3, DDR). The L2 controller keeps L1D cache to the next level of caches (L2, L3, and so forth) and to DMAs to L2 SRAM. The L2 controller also keeps the streaming engine coherent by snooping out the line from the L1D cache in response to streaming engine reads. External coherency, which includes the L2 cache being coherent to the L3 cache and DDR/external.

This memory system supports virtual memory, and includes as part of it address translation, uTLBs, L2 page table walk, and L1P cache invalidates.

The shared L2 controller supports 1 streaming engine with two streams. These two streams are kept coherent to the L1D caches, and have a pipelined, high bandwidth interface to L2 in the 1-core configuration.

The L1D cache is backed up by a victim cache, has a larger cache line size (128 bytes), and implements aggressive write merging. New features include look-up table, histogram, and atomic accesses. Cache changes in the L1P include higher associativity (4 way), and a larger cache line size (64 bytes). The L2 cache also features higher associativity (8 way).

Figure 1 is a high-level view of the C7x Corepac cluster.

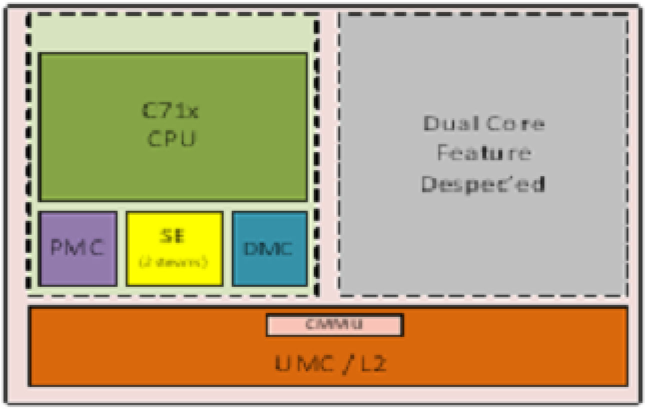


Figure 1. Memory System – High Level View

Copyright © 2020, Texas Instruments Incorporated

Figure 2. Memory System - Interface Details

Figure 2 shows the interfaces between the various blocks. The C71x Corepac cluster consists of:

- 1 C71x CPU
- 1 L1 data cache controller (DMC), with its private configurable L1D memory – L1D cache can be a maximum size of 32KB, and total L1 memory (cache + SRAM) maximum of 48KB
- 1 L1 program cache controller (PMC), with its private 32KB L1P cache
- 1 streaming engine (SE), with two streams
- L2 unified cache controller (UMC), with a configurable unified L2 cache and SRAM up to 2M bytes
- 1 Corepac memory management unit (CMMU)

The C71x Corepac memory system, consisting of L1D, L1P, and L2, has significant architectural and micro-architectural changes. The rest of this section describes those changes. Each of the memory controllers has its own section with more details, with a separate section on coherence.

Table 1. Memory System Configurations

Parameter	Value
No. of DSP cores	1 core only
Datapath Width	512 bits
Cache line Size	1024 bits
L2 Banking	4-banks LS Banked
L1D RAM Size	48KB
L2 RAM Size	512KB
L2 Banking	LS Banking
L2 RAM Latency	1-cycle only
Maximum L2 Cache	512KB

Copyright © 2020, Texas Instruments Incorporated

1.1.1 Memory Map

Table 2. Memory Map

PHYSICAL ADDRESS										
[43:29]	**[28:27]**			**[26:24]**			**[23:0]**			
15 bits	**2 bits**			**4 bits**			**24 bits**			
MSMC ID	**MSMC REGION**			**COREPAC ID**			**COREPAC INTERNAL MEMORY**			
	[28:27]	**Name**	**Size**	**[26:24]**	**Name**	**Size**	**[23:20]**	**Name**	**Size**	**[23:0]**
	00	COREPAC	256 MB	000	COREPAC 0	16 MB	1111	L1D 1	1 MB (64 KB)	F0_0000 - FF_FFFF
							1110	L1D 0	1 MB (64 KB)	E0_0000 - EF_FFFF
							1000	L2	4 MB (1 MB)	80_0000 - 8F_FFFF
							0xxx	Reserved	8 MB	00_0000 - 7F_FFFF
	01	MSMC INT	768 MB							
	10									
	11									

1.1.2 Interfaces

The following interfaces exist in the C71x CorePac subsystem:

1. CPU-DMC:
   - 512-bit vector read and 512-bit vector write
   - 64-bit scalar and 64-bit scalar write
2. CPU-PMC: 512-bit Read
3. DMC-UMC: 512-bit Read, 512-bit Write interfaces, that can do cache transactions, snoop, L1DSRAM DMA, and external MMR accesses. Each interface can handle two dataphase transactions.
4. PMC-UMC: 512-bit Read, which can be either 1 or 2 dataphases
5. SE-UMC: 512-bit Read, which can be either 1 or 2 dataphases
6. UMC-MSMC: 512 bit-Read and 512-bit Write, with snoop and DMA transactions overlapped. Transactions can be either 1 or 2 dataphases.
7. MMU-UMC: Page table walks from L2.
8. PMC-MMU: uTLB miss to MMU

1.1.3 Features and Functional Blocks

This section summarizes the features of the C7x CorePac. Figure 3 shows a pictorial view of the features implemented in each controller.

Figure 3. Memory System Features

Copyright © 2020, Texas Instruments Incorporated

1.1.3.1 Program Cache Controller (PMC) Features

The following are the features supported by the PMC controller:

- L1P cache
  - 32KB L1P cache: Size is not configurable
  - 4-way set associative
  - 64-byte cache line size
  - Virtually indexed and virtually tagged (48-bit virtual address)
- L1P SRAM
  - No support for L1P SRAM, either from CPU or DMA
- Prefetch
  - Auto prefetching on L1P misses from L2
  - Capability to queue multiple fetch packet requests (64 bytes) to UMC to enable deeper prefetch in program pipeline.
- Error detection (ECC)
  - Parity protection on data and tag RAMs. 1-bit error detection
  - Data RAM parity protection is on instruction width granularity (1 parity bit every 32 bits)
  - Auto-invalidate and re-fetch on errors in TAG RAM.
  - Aligns to Keystone 3 safety architecture provided testing and mechanism for ECC detection logic
  - Aligns to Keystone 3 safety architecture for reporting of detected parity and ECC faults from both error injection and functional access
- Software-initiated coherence operations
  - Single-cycle cache invalidate with support for three modes: All cache lines, MMU page table base 0, and MMU page table base 1
- Virtual memory
  - Virtual to physical addressing on misses
  - uTLB to handle address translation and for code protection
- Debug
  - Pass/Fail status is returned to on all reads
  - Events are exported out for interesting conditions that can be used for debug and performance measurements.
  - Cache tag view is visible through the ECR registers, and lets the user access the contents of the TAG RAMs
- Extended Control register (ECR – previously MMR) access
  - L1P ECR registers are accessible from the CPU through a non-pipelined interface that looks similar to the CPU-ERI interface present on previous Corepac generations. These registers are not memory mapped, and instead are mapped to a MOVC CPU instruction.

1.1.3.2 L1 Data Cache Controller (DMC) Features

The following features are supported on the L1D controller:

- L1D cache
  - 32KB cache, configurable down to 8KB cache
  - Dual datapath (DP0+DP1) support similar to C6x
  - Datapath configuration supported (64 bit + 512 bit)
  - 512-bit vector datapath with 128-byte cache line size
  - Read allocate cache
  - Support for write-back and write-through modes (write-through mode is not supported in the first generation, and functions like write-back)

Copyright © 2020, Texas Instruments Incorporated

  - 16-entry victim cache
  - Physically indexed, physically tagged
  - Hit under miss
  - Posted write miss support
  - Write merging on all outstanding write transactions inside L1D
  - FENCE operation on outstanding transactions, with support for two types of transactions (‘colors’)
  - Auto-flush and idle-flush
- L1D SRAM
  - Support for DMA for CPU reads and writes to L1D SRAM. The amount of SRAM available depends on the total L1D memory (SOC Tie-off) and L1D cache size.
- Lookup table and histogram
  - Capability to support up to 16 parallel table lookup
  - Histogram support
  - CPU LUTINIT instruction can update L1D with lookup table data, with support for a mode to provide duplication of data within L1D
  - Support for both signed and unsigned bin and signed weights for histogram
  - Saturation and non-saturation mode for all combinations of bin type and weights for histogram
- LUTWR in L1D SRAM
  - This feature adds the capability to service multiple parallel stores to different addresses in L1D RAM.
  - L1D SRAM is dynamically configured into multiple regions/ways based on degree of parallelism required.
- Bandwidth
  - 64 + 512-bit CPU load / store bandwidth
  - 1024-bit L1D RAM bandwidth
  - 16 × 64-bit wide banks
- Error detection (ECC)
  - Full ECC on data RAMs: 1-bit error correction and 2-bit error detection
  - Parity-only protection on TAG RAMs
  - Provides ECC syndrome on writes and victims out to L2.
  - Receives ECC syndromes with read data from L2, and does detection and correction before presenting this data to the CPU.
  - Full ECC on victim cache
  - Read-modify-Write support to prevent parity corruption on partial line writes
  - L2-L1D I/F correction for Read-Response data pipeline ECC protection
  - Aligns to Keystone 3 safety architecture for testing and mechanism for ECC detection logic
  - Aligns to Keystone 3 safety architecture for reporting of detected parity and ECC faults from both error injection and functional access
- Emulation and debug capabilities
  - DAS codes are returned on reads to indicate the level of cache that the data was read from.
  - Events are exported out for interesting conditions that can be used for debug and performance measurements.
  - Events are exported out for transactions qualified with the interest attribute. This can be used to filter out transactions based on user-programmable criteria.
  - Cache TAG view is visible through the ECR registers, and lets the user access the contents of the TAG RAMs.
  - Hardware watch point (HWWP) and command interest (cinterest) matching compare for HWWP criterias, and interest address matching generation

Copyright © 2020, Texas Instruments Incorporated

  - CPU initiated load / store command interest generation
- Atomic
  - Compare and swap
  - Swap
- Coherence
  - Full MESI support in both main and victim cache. The shared state is not used by the cache controllers, but that is a micro-architectural detail not visible to the user.
  - Support for global cache coherence operations
  - Snoops and cache maintenance operation support from L2
  - Snoops for L2 SRAM, MSMC SRAM, and external (DDR) addresses
  - Full TAG-RAM comparisons on snoop and cache maintenance operations
- Virtual memory support
  - Support for wider (40-bit) physical address
- Extended Control register (ECR – previously MMR) access
  - L1D ECR registers are accessible from the CPU through a non-pipelined interface that is similar to the CPU-ERI interface present on previous Corepac generations. These registers are not memory mapped, and instead are mapped to a MOVC CPU instruction.
- L2 address aliasing
  - Support for VCOP address aliasing mode
  - Support for aliasing for multiple separate buffers, equivalent to the VCOP – IBUFAH, IBUFAL, IBUFBH, IBUFBL buffers
  - Out of range and ownership check for all buffers

1.1.3.3 L2 Unified Cache Controller (UMC) Features

Features implemented in the L2 unified cache controller:

- L2 cache:
  - 8-way set associative
  - Support for cache sizes: 64KB, 128KB, 256KB, 512KB, and 1MB
  - The number of sets scales with the cache size. The number of ways remains static.
  - Random replacement
  - 128-byte cache line size
  - Write-allocate cache
  - Physically indexed, physically tagged (40-bit physical address)
  - 4x Banked TAG RAMs, which allows for four independent split pipelines
  - Supports 2 × 64-byte streams from one streaming engine
  - External MMR and MDMA accesses on an unified interface to MSMC
  - Caches MMU page tables
  - Cache pre-warming from SE and MSMC
  - Default is that L2 does not allocate msmc_sram. This matches the C66x implementation. The ECR register L2CFG has a bit that can be programmed to allow msmc_sram to be allocated in L2.
- L2 SRAM
  - 4 × 512-bit physical banks with 4 virtual banks each. Each has independent access control.
  - Security firewall on L2 SRAM accesses
  - DMA access to L2 SRAM on merged MSMC I/F
- Error detection (ECC)
  - ECC detection and correction is done on a 256-bit granularity
  - Full ECC support for both TAG and data RAMs: 1-bit error correction and 2-bit error detection for

Copyright © 2020, Texas Instruments Incorporated

both

  - Provides ECC syndrome on writes and victims out to MSMC
  - Read-Modify-Writes on DMA/DRU writes to keep parity valid and updated
  - ECC correction and generation of multiple parity bits to L1P and streaming engine
  - Auto-scrub to prevent accumulation of 1-bit errors, and to refresh parity
  - Parity clear / invalidate on reset can be disabled through tie-off
  - Datapath and pipeline protection
  - Aligns to Keystone 3 safety architecture provided testing and mechanism for ECC detection logic.
  - Aligns to Keystone 3 safety architecture for reporting of detected parity and ECC faults from both error injection and functional access.
- Debug
  - Access codes are returned on reads to indicate the level of cache that the data was read from.
  - Bus error codes are returned to indicate pass / fail status of debug reads and writes.
  - Events are exported out for interesting conditions that can be used for debug and performance measurements.
  - Events are exported out for transactions qualified with the interest attribute. This can be used to filter out transactions based on user-programmable criteria.
  - Cache Tag View is visible through the ECR registers, and lets the user access the contents of the TAG RAMs.
- Coherence
  - Full coherence between L1D cache, 2 streams, L2 SRAM, MSMC SRAM, and external (DDR)
  - Snoops for L2 SRAM, MSMC SRAM, and external (DDR) addresses
  - Full MESI support. The shared state is not used by the cache controllers, but that is a micro-architectural detail not visible to the user.
  - User coherence commands from streaming engine
  - Support for global coherence operations
- Extended Control register (ECR – previously MMR) access
  - L2 ECR registers are accessible from the CPU through a non-pipelined interface that is similar to the CPU-ERI interface present on previous Corepac generations. These registers are not memory mapped, and instead are mapped to a MOVC CPU instruction.
- L2 address aliasing
  - Support for VCOP address aliasing mode
  - Support for aliasing for multiple separate buffers, equivalent to the VCOP – IBUFAH, IBUFAL, IBUFBH, IBUFBL buffers
  - Out of range and ownership check for all buffers with error logging

1.1.3.4 Caches and Memory System Features

These features are applied across the three levels of caches.

- Security and firewall
  - The C7x architecture provides two security states, each with its associated memory attributes. This is aligned with the SoC security and firewall architecture. The secure / non-secure attribute accompanies the transaction presented by the CPU to the cache controllers.
    - Secure state: When the CPU is in a secure state, indicated by the csecure attribute on its transactions, the cache controllers allow the CPU to access secure and non-secure memory locations.
    - Non-secure state: When the CPU is in a non-secure state, indicated by the csecure attribute on its transactions, the cache controllers allow the CPU to access non-secure memory locations, but prevents it from accessing secure memory locations.
  - The secure firewall IP is present at two places:

Copyright © 2020, Texas Instruments Incorporated

    - UMC - MSMC Interface: to protect transactions initiated by the C7x that go external.
    - UMC - L2RSAM: to protect accesses to L2 SRAM space.
- Privilege
  - The C7x architecture provides six privilege and execution levels for full isolation and protection:
    - Secure root supervisor
    - Secure root user
    - Non-secure root supervisor
    - Non-secure root user
    - Non-secure guest supervisor
    - Non-secure guest user
  - This information is presented by the CPU as attributes with every transaction, and is passed along with transactions that miss the cache, to the next level of cache or memory. On all cache misses, the response comes back with an error status on a privilege violation, and that is recorded as an exception.
    - In the case of PMC, some of these attributes are held with the tags to be able to invalidate cached lines from the L1P cache based on the privilege.
    - The DMC and UMC cache controllers do not hold this information with its tags.
- C7x CPU and caches powerdown modes
- Safety
  - The details of safety, error detection, and correction for all three levels of caches are described in Section 1.1.3.6.

1.1.3.5 Changes from Previous Corepac Generation (C66x+)

1.1.3.5.1 Program Cache Controller (PMC)

Table 3. Program Memory Controller Feature Changes from Previous Corepac (C66x)

Feature	Sub-Feature	C64x / C66x	Corepac (C71x)
L1P Cache	Associativity	Direct Mapped (1-Way)	4-Way
	Cache Size	Programmable (4KB-32KB)	Fixed (32KB)
	Cache LIne Size	16bytes	32bytes
	Replacement Policy	N/A	FIFO
	Indexing		Virtually Indexed, Virtually Tagged
	Interface	1-dataphase L2 miss return	2-dataphases L2 miss return with implicit pre-fetch
L1P SRAM		Support for up to 1MB	No Support for L1PSRAM
Prefetch		Maximum 3	Branch prediction enables continuous linear prefetch Auto-prefetch of next 64byte on every fetch
Branch Prediction		No Support	CPU Support coupled with PMC pre-fetch and flush
ECC		Parity only on Data	Full ECC - Parity on Data with restartable interrupts in CPU Parity on TAG RAMs with auto invalidate and re-fetch
Virtual Memory		No Support	Virtual to Physical address translation on misses uTLB for code protection

Copyright © 2020, Texas Instruments Incorporated

Table 3. Program Memory Controller Feature Changes from Previous Corepac (C66x) (continued)

Feature	Sub-Feature	C64x / C66x	Corepac (C71x)
L1D Cache	Associativity Main Cache	2-Way	Multi Level Cache: Main 1-way cache + Victim Cache
	Victim Cache	Not Applicable	Fully Associative, 8 or 16 Cache Lines
	Cache Size	Programmable (4KB-32KB)	Programmable (8KB-32KB)
	Cache Line Size	64bytes	128 Bytes
	Replacement Policy	LRU	N/A
	Indexing	Not Applicable	Physically Indexed, Physically Tagged
	Load Miss Behavior	Stall on Miss	non-blocking cache (Support hit under miss)
	Aggressive Write Merging	Write Miss merging only to tail entry of write miss buffer	Write merging on all outstanding writes, until the write miss buffer needs to be flushed
	S/W Prefetch	Not supported	S/W controlled prefetch to L1D Cache
L1D SRAM		Only CPU and DMA access to flat SRAM	CPU and DMA access still supported
			Table Look-Up and Histogram Support
Bandwidth		64-bit Load/Store	512-bit Load/Store (configurable to 256 bits)
		256-bit L1D Bandwidth	1024-bit L1D Bandwidth
		8 x 32-bit banks	16 x 64-bit banks
ECC		No ECC Support (ECC support in Turing)	Read-Modify-Write on all partial stores. Full ECC - ECC on Data RAMs. Parity only on TAG RAMs
Atomic Operations		Not Supported	Compare and Swap
Coherence		Only coherent with L2 SRAM	Coherent to L2 SRAM, MSMC SRAM, DDR
		Limited MEI support	Full MESI support
Virtual Memory		32-bit addressing	Support for 40-bit physical addressing
Architecture	UMC Clock Frequency	C64x: CPU_CLK / 2 C66x: CPU_CLK (UMC)	CPU_ CLK
	External Clocks	C64x: Master & Slave DMA (CPU_CLK/2, CPU_CLK/3, CPU_CLK/4) C66x: Master DMA (CPU_CLK, CPU_CLK /2) + Slave DMA (CPU_CLK/2, CPU_CLK/3, CPU_CLK/4)	CPU_CLK
	L1 Support	1 L1D + 1L1P	1 L1D + 1L1P
	Streaming Engine Support	No support	Supports 1Streaming engine, each with 2 512bit streams
	External Interfaces	256bit Master DMA (MDMA) + 64bit / 128bit slave DMA (SDMA)	Merged 512bit MDMA+SDMA I/F
L2 Cache	Associativity	4-Way	8-Way
	Max Cache Size	256KB / 1MB	256 KB / 512 KB / 1MB
	Replacement Policy	Pseudo-Tree LRU	Random
L2 SRAM	Banking	2 256bit banks, but single source access every cycle	4 512bit banks, with 4 different sources being able to access every cycle
	Memory Protection	Controlled by (Memory Protection) MPPA MMR registers	Controlled by MMU Page Table Attributes
	Security	Controlled by (Memory Protection) MPPA MMR registers	Controlled by Security Firewall in addition to MMU Page Table Attributes

Copyright © 2020, Texas Instruments Incorporated

Table 3. Program Memory Controller Feature Changes from Previous Corepac (C66x) (continued)

Feature	Sub-Feature	C64x / C66x	Corepac (C71x)
Configuration	Static Configurations	Expanded configurations over size, non power-of-two memories, banking, base address, etc.	Restricted configurations over size only
	Control Registers	Memory Mapped Registers accessible to the CPU through its Load/Store pipe	Extended Control Registers accessible to the CPU through its MOVC instructions. Not memory mapped
	Internal Config Bus	VBUSP bus driven by DMC/EMC	Broadcast CPU-ERI bus
Prefetch		Handled by XMC for DDR and MSMC_SRAM regions	Included in UMC for L2_SRAM in addition to DDR and MSMC_SRAM regions
ECC		Full ECC on Data, but with limitations	Full ECC - 2-bit Detect, 1-bit Correct ECC on TAG and Data RAMs
Coherence		Limited L2SRAM - L1D Cache coherence	Full Memory Coherence between L1D caches, L2 SRAM, MSMC SRAM, DDR
		Limited MEI Support	Full MESI Support
		Snoop writes to L1D cache for DMA writes to L2 SRAM	No Snoop writes. Snoop Invalidate instead

Copyright © 2020, Texas Instruments Incorporated

1.1.3.6 Safety, Error Detection, and Correction (ECC)

ECC is a key feature which impacts the entire memory system architecture.Figure 4 shows the dataflow for ECC and the responsibilities of each block.

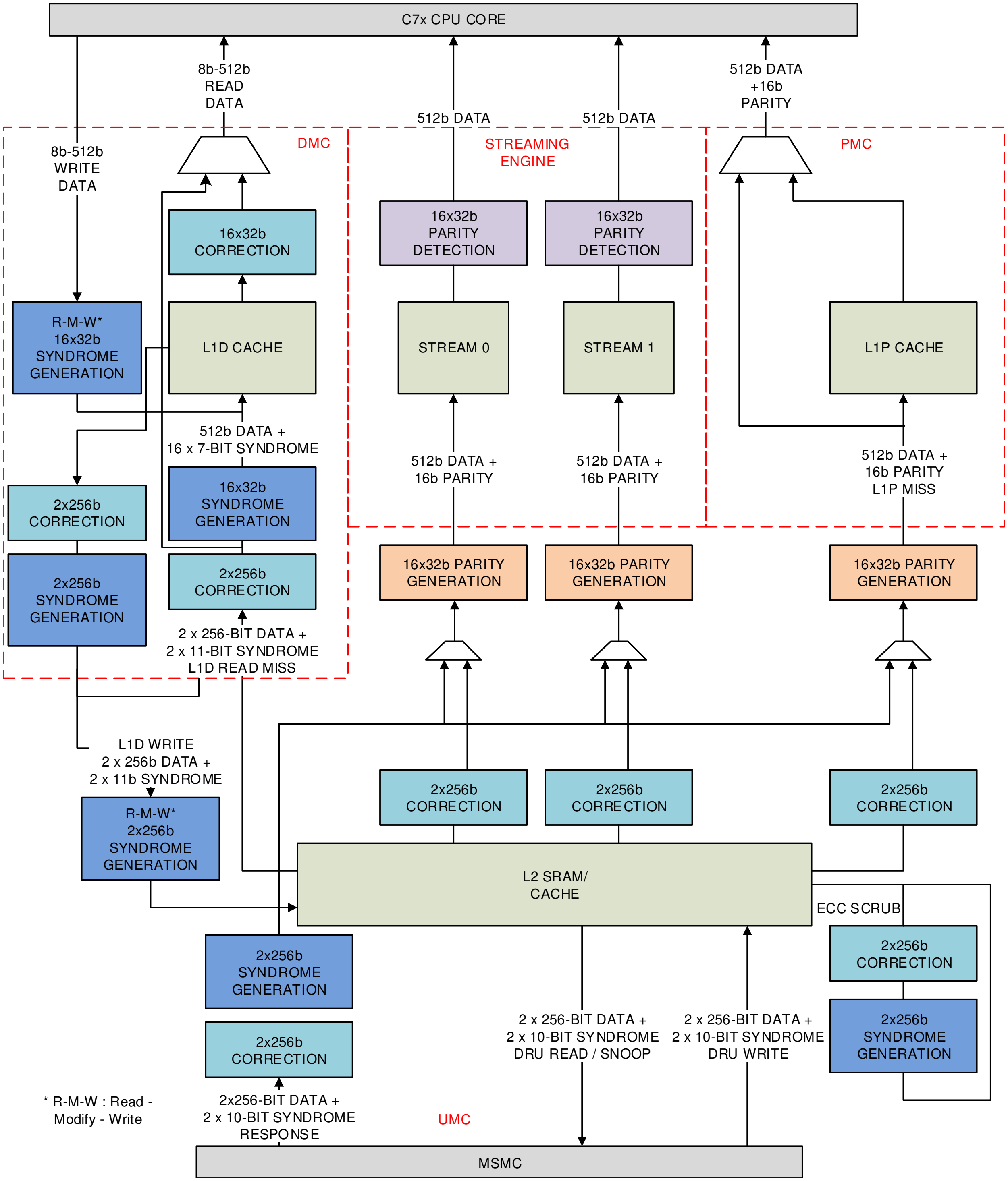

C7x CPU CORE
8b-512b READ DATA
512b DATA
512b DATA
512b DATA +16b PARITY
DMC
STREAMING ENGINE
PMC
8b-512b WRITE DATA
16x32b CORRECTION
16x32b PARITY DETECTION
16x32b PARITY DETECTION
R-M-W* 16x32b SYNDROME GENERATION
L1D CACHE
STREAM 0
STREAM 1
L1P CACHE
512b DATA + 16 x 7-BIT SYNDROME
512b DATA + 16b PARITY
512b DATA + 16b PARITY
512b DATA + 16b PARITY L1P MISS
2x256b CORRECTION
16x32b SYNDROME GENERATION
2x256b SYNDROME GENERATION
2x256b CORRECTION
16x32b PARITY GENERATION
16x32b PARITY GENERATION
16x32b PARITY GENERATION
2 x 256-BIT DATA + 2 x 11-BIT SYNDROME L1D READ MISS
L1D WRITE 2 x 256b DATA + 2 x 11b SYNDROME
R-M-W* 2x256b SYNDROME GENERATION
2x256b CORRECTION
2x256b CORRECTION
2x256b CORRECTION
L2 SRAM/ CACHE
ECC SCRUB
2x256b SYNDROME GENERATION
2x256b CORRECTION
2x256b CORRECTION
2x256b SYNDROME GENERATION
2 x 256-BIT DATA + 2 x 10-BIT SYNDROME DRU READ / SNOOP
2 x 256-BIT DATA + 2 x 10-BIT SYNDROME DRU WRITE
* R-M-W : Read - Modify - Write
2x256-BIT DATA + 2 x 10-BIT SYNDROME RESPONSE
UMC
MSMC

Figure 4. Memory System ECC Dataflow

Copyright © 2020, Texas Instruments Incorporated

Along with the data RAMs, the pipeline registers on the ECC datapath are also protected. That is done by passing along the ECC parity syndrome for the data, with detection and correction done when that data leaves the memory system (CPU and MSMC). The one exception to this is the L2-L1P interface, where the ECC protocol changes from correction to detect only. There, L2 does the correction, and passes along the per-instruction parity bits to PMC. PMC, then, passes along the parity information to the CPU, and the CPU does the parity check and takes necessary actions. The ECC details for each block are part of that memory controller description chapter.

The key features of this approach are:

- ECC detection and correction on memory read out of SRAMs (not TAG) is done only in these cases:
  - Data is consumed by the CPU.
  - Data is accessed from the last level cache.
  - Data width over which the ECC syndrome is computer changes.
- The ECC detection and correction is delayed to the point at which one of these conditions is true.
- The ECC syndrome is carried through the system, and in that way provides protection on all the pieces of logic that it is carried over; these include interface, pipeline, and datapath registers.
- This approach protects not just the SRAMs, but also a number of registers, including the CPU registers.

Figure 4 shows the ECC protection for data. The specific actions taken by each controller vary and are described here.

- PMC
  - On cache hits in L1P, the parity information from the L1P data RAMs is forwarded to the CPU, and the CPU does the parity detection. This ensures that the datapath from PMC to CPU is also ECC protected. Following this, the CPU sends down an flush indication to the PMC, which invalidates the offending line from its cache and re-fetches it from L2.
  - On cache misses to L2, the L2 controller does the ECC detection and correction. In case of an ECC error, that information is forwarded to PMC along with the parity bits. PMC invalidates that line from its cache (does not allocate) and forwards that information to the CPU.
- DMC
  - Cache read hits in L1D, and reads from L1D SRAM.
    - 1-bit ECC errors are corrected. An exception may be raised, and the appropriate information recorded. The error counter increments.
    - 2-bit ECC errors are detected and an exception is raised with the appropriate information recorded. An error status is also sent to the CPU with the read data. The error counter increments.
  - Cache write hits in L1D, and writes to L1D SRAM.
    - The DMC controller goes through a read-modify-write cycle to ensure that the correct ECC information is generated and stored with data.
  - Misses from L2
    - L2 forwards the ECC information to L1D. The DMC controller does ECC detection and correction on this data, and re-generates the ECC information on the granularity that it needs.
    - 1-bit ECC errors are corrected. An exception may be raised, and the appropriate information recorded. The error counter increments.
    - 2-bit ECC errors are detected and an exception is raised with the appropriate information. The line is not allocated in the cache. The error counter increments.
- UMC
  - Data and DMA: On cache read hits in L2, or on reads from L2 SRAM.
    - The ECC information is forwarded to DMC / MSMC. No ECC detection or correction is done here. This enables ECC protection on the datapath from UMC to DMC / MSMC in addition to just the data RAMs.
  - Code / CMMU page table walk: On cache read hits in L2, or on reads from L2 SRAM.
    - 1-bit ECC errors are corrected. An exception may be raised, and the appropriate information

Copyright © 2020, Texas Instruments Incorporated

recorded. The error counter increments.

- 2-bit ECC errors are detected and an exception is raised with the appropriate information. An error status is also sent to the PMC / CMMU with the read data. The error counter increments.
- In addition, parity information is generated at the granularity that PMC / CMMU requires.

– On cache write hits in L2, or on writes to L2 SRAM.

- The UMC controller goes through a read-modify-write cycle to ensure that the correct ECC information is generated and stored with data.

– On misses to MSMC / DDR

- The MSMC controller sends the ECC syndrome along with the read response and UMC stores it with data in its cache. UMC does not any ECC detection or correction at this point.

Figure 4 does not show the ECC protection for tags. The actions takes for ECC errors in the TAG RAMs are as follows:

- PMC
  - 1-bit parity error in the TAG RAM is detected, and the controller invalidates the line and re-fetches it. This is done inside the PMC hardware without any interaction by the CPU.
- DMC
  - 1-bit ECC errors in the TAG RAM are detected. An exception may be raised, and the appropriate information recorded. The error counter increments.
  - The current implementation of the C7x does not support correction or 2 or more bit detection for the TAG RAMs.
- UMC
  - 1-bit ECC errors in the TAG RAM are corrected. An exception may be raised, and the appropriate information recorded. The error counter increments.
  - 2-bit ECC errors in the TAG RAM are non-correctable, and the controller takes an exception with the appropriate information recorded in ECR registers. The error counter increments.

1.1.3.6.1 Address Pipeline Protection

NOTE: Address bits in the pipeline or the SRAM are not protected on this generation of the C7x.

Factor in the address bits of the SRAM location being accesses, into the syndrome recorded in the SRAM. On a read from the SRAM, the contribution of the address bits in the syndrome is removed, before the data and the syndrome are forwarded to the ECC detection and correction logic. In the event of an error, where 1 or 2 of the address pipeline register output are inverted, the memory row read is different from what was intended. After removing the contribution of the address from the syndrome, the syndrome does not match the data, and results in an error detected.

1.1.3.6.2 Fragmented MDMA Read Response

In some cases, reads from the C7x to MSMC results in fragmented responses. There responses are merged in UMC and presented to the requestor as a complete dataphase. Each of the fragmented response from MSMC comes with its own syndrome, but these various syndromes are not merged. Instead, in these cases, UMC generates a new syndrome from the merged data. This was done for PPA reasons. This behavior occurs on a special signalling from MSMC (dtrace).

1.1.3.6.3 Read Response from MSMC

This read response from MSMC follows two paths inside the UMC pipeline:

- For L2 cache allocates, this data goes through the UMC pipeline. The data and syndrome from MSMC is written to the cache without any modification.
- There exists a bypass path for the response from MSMC, such that it is forwarded to the requestor without having to go through the entire pipeline. This reduces latency on reads. This is handled differently depending on the requestor.

Copyright © 2020, Texas Instruments Incorporated

- DMC Reads: UMC fowards the data and syndrome without any modification to DMC. DMC does the detection/correction on the data, generates a new syndrome (for each 32 bits if data), and commits that to its cache
- PMC, SE, MMU Reads: UMC does the detection/correction of the response data and generates the appropriate parity bits off the corrected data.

1.1.3.6.4 Snoop Response from DMC

The snoop response from DMC is forwarded by UMC for the requestor (MSMC, SE, MMU) without any modification. DMC does the detection/correction of that data. On the case of SE and MMU, UMC generated the parity bits from this data. This data is not re-corrected in UMC.

1.1.3.6.5 Gappy Write Data from DMC and MSMC

DMC stores and MSMC DMA Writes that have gappy data initiate a Read-Modify-Write cycle in UMC. UMC reads the data from its SRAM/cache, applies detection/correction on it, merges the new (gappy) data, generates a new syndrome, and writes the merged data and newly generated syndrome to its SRAM/cache. The data from DMC/MSMC does not go through detection/correction in UMC.

1.1.3.6.6 Non-Gappy Write Data from DMC and MSMC

In the case of non-gappy write stores from DMC and DMC writes from MSMC, the data and syndrome is written to the L2 SRAM/cache without any modification.

1.1.3.6.7 Error Injection

The primary reason for supporting error injection in all memories is to be able to test customer code following an exception / interrupt, and to test the detection and correction hardware.

1.1.4 Events – Exceptions and Interrupts

All exceptions and interrupts are referred to as events in this document. There are two types of events that C7x memory system will take: synchronous and asynchronous.

1.1.4.1 Synchronous Events

These events accompany the read or write response to the requestor, and are usually signalled through the read response status (dstatus) or the write response status (wstatus). There is no separate event pin for these events. The requestor is responsible for responding to these events. The memory system does not log any information regarding these events after it has responded with the status.

Table 4 lists all such events. On interfaces where no read or write status bus exists, this event is signalled through the asynchronous event mechanism.

NOTE: The DMC / CPU do not record write status, and thus, UMC may not drive the write status up to DMC. This is a micro-architectural decision. For writes that terminate in L2, UMC takes the exception and logs the error. For writes terminating in MSMC, MSMC does the error logging.

Table 4. Synchronous Events

Category	Name	Details	Read / Write Status
Address Error	CPU Read Address Error	CPU Read address was outside the valid L1D or L2 address space. CPU Read was mapped to a region allocated for L1D or L2 cache.	The controller drives the appropriate error code on the rstatus field of the read response back to the requestor.
Protection Error	ECR Store Access Error	ECR Write Access with insufficient privileges ECR Write to a Write Only register ot register field	The controller drives the appropriate error code on the sstatus field of the write response back to the requestor.

Copyright © 2020, Texas Instruments Incorporated

Table 4. Synchronous Events (continued)

Category	Name	Details	Read / Write Status
Protection Error	ECR Load Access Error	ECR Read Access with insufficient privileges ECR Read to a Read Only register or register field	The controller drives the appropriate error code on the rstatus field of the read response back to the requestor.
Address Error	DMA Store Address Error	DMA store address was outside the valid L1D or L2 address space. DMA store was mapped to a region allocated for L1D or L2 cache.	The controller drives the appropriate error code on the sstatus field of the write response to back to the requestor.
Address Error	DMA Read Address Error	DMA Read address was outside the valid L1D or L2 address space. DMA Read was mapped to a region allocated for L1D or L2 cache.	The controller drives the appropriate error code on the rstatus field of the read response back to the requestor.
Command Error	LUT Command Error	LUT command error indicating an illegal LUT transaction (DMC only)	The controller drives the appropriate error code on the rstatus field of the read response back to the requestor.
Firewall Error	DMA Write to L2SRAM Firewall Error	CPU Store failed L2SRAM Firewall check	UMC sends a bad Write Status (sstatus) to MSMC / Requesting Core.
ECC	Non-Correctable ECC Error for CPU Fetch / SE Load	Non-Correctable Data Error encountered on ECC syndrome conversion for data to PMC, SE.	UMC indicates Data Error on the Read Status (dstatus) to the requestor.
ECC	Non-Correctable ECC Error for CPU Data Load	Non-Correctable Data Error encountered on ECC syndrome conversion for data load from DMC	DMC indicates Data Error on the Read Status (dstatus) to the requestor.

1.1.4.2 Asynchronous Events

These events do not have an accompanying read or write response. These are signalled through event pins. Table 5 lists all such events. The C7x memory system signals three types of asynchronous events, as shown in Table 5. Events from the three controllers are combined into a single set of events. Each event is independently controlled.

Table 5. Asynchronous Events

Category	Name	Details	Severity / User Response
MEMSYS_ADDERR_ EV – DMC_ADDRERR_EV – UMC_ADDRERR_EV	Addressing Error	DMA / CPU Store address was outside the valid L2 space. DMA / CPU Store was mapped to a region allocated for cache. (DMC only) Invalid LUT transaction	Transaction is dropped, and may point to software errors.
MEMSYS_CACHEERR_ EV – UMC_CACHEERR_EV	Cache Allocation Error	Write Allocate was dropped due to a bad read status on the read response. Allocate response came with an error for transactions that do not send a response back to the requestor (example: pre-warm)	This could point to either a software / configuration error or an error at the endpoint memory / IP.
MEMSYS_FWFAIL_EV – UMC_FWFAIL_EV [1:0]	Firewall Error	CPU / DMC Store failed L2SRAM Firewall check	This could point to a software configuration error or an attempt at circumventing security. Firewall Error Logging is described in the Keystone 3 Firewall IP Specification document.

Copyright © 2020, Texas Instruments Incorporated

NOTE: The individual events are OR'ed into a single event from the memory system. For example, MEMSYS_ECCCOR_EV = PMC_ECCCOR_EV | DMC_ECCCOR_EV | UMC_ECCCOR_EV.

After signalling an event, each of the memory controllers log the event in the ECR registers. The core is then required to read through all the ECR registers related to that particular event to acquire details of the event. After processing the event, the core is also required to clear the fault by writing to the ECR Clear register. A number of ECR registers are provided to record details of the error, and for the core to clear the fault. Table 6 lists the various ECR registers provided.

Table 6. Event Logging and Processing Registers

Category	Name	Details
DMC_ADDRERR_EV	DMC_ADDRERR_EAR	Addressing Error Event Address Register
	DMC_ADDRERR_ESR	Addressing Error Event Status Register
	DMC_ADDRERR_ECR	Addressing Error Event Clear Register
	DMC_CMDERR_EAR	LUT Command Error Address Register
	DMC_CMDERR_ESR	LIUT Command Error Status Register
	DMC_CMDERR_ECR	LUT Command Error Clear Register
UMC_ADDRERR_EV	UMC_ADDRERR_EAR	Addressing Error Event Address Register
	UMC_ADDRERR_ESR	Addressing Error Event Status Register
	UMC_ADDRERR_ECR	Addressing Error Event Clear Register
UMC_ALLOCERR_EV	UMC_ALLOCERR_EAR	Cache Allocation Error Event Address Register
	UMC_ALLOCERR_ESR	Cache Allocation Error Event Status Register
	UMC_ALLOCERR_ECR	Cache Allocation Error Event Clear Register
UMC_FWFAIL_EV	Firewall Error Logging is described in the Keystone 3 Firewall IP Spec	

The format of these registers is described below.

Table 7. Event Address Register

REG NAME	63							0	REG ID
EAR	ADDRESS								TBD
	R, +0								
REG NAME	63	P+1	P	M+1	M	N+1	N	0	REG ID
ESR	Reserved		BITPOS		Reserved		Error Type		TBD
	R, +0								

Table 8. Event Status Register Bits

Field	Description	
ERROR_TYPE	N Bits	iDetails in each of the arch specs. Includes control status bits such as: Memory where error occured (example: Tag RAM, Data RAM) Requestor (CPU, ECR, DMA)
BITPOS	M Bits	Position of the error for 1-bit soft error

Table 9. Event Clear Register

REG NAME	63	1	0	REG ID
ECR	Reserved		CIR	TBD
	R, +0		W, +0	

Copyright © 2020, Texas Instruments Incorporated

1.1.5 Debug Events and Stalls

The memory system exposes a limited number of events and stalls for functional and performance debug purposes. These can be pulse (typically used for events), or level (typically used for stalls). They are further classified as generic, where they are driven for all transactions, and interest qualified, where they are driven only for transactions accompanied by the Interest bit. PMC debug events do not have an interest qualifier.

Table 10. PMC Debug Events and Stalls

No.	Category	Event	Description	Generic / Interest Qualified	Level / Pulse
0	Cache Event	l1p_cache_hit	L1P Cache Hit	Generic	Pulse
1	Cache Event	l1p_cache_miss	L1P Cache Miss	Generic	Pulse
2	Cache Event	l1p_glbinv	Global Invalidate initiated	Generic	Pulse
3	Cache Event	l1p_tbl0_inv	Table 0 Invalidate in progress	Generic	Pulse
4	Cache Event	l1p_tbl1_inv	Tabel 1 Invalidate in progress	Generic	Pulse
5	Cache Event	l1p_cache_hit_conflist	Hit in L1P cache, but a pending miss to the same set and way	Generic	Pulse
6	Cache Event	l1p_multi_hit_inv	Multiple Hits in L1P cache and all except one are invalidated	Generic	Pulse
7	Cache Event	l1p_way_inv	Invalidate in response to a data error	Generic	Pulse
8	Unused				
9	Unused				
10	Unused				
11	Unused				

Table 11. DMC Debug Events and Stalls

No.	Category	Event	Description	Generic / Interest Qualified	Level / Pulse
0	Coherence stall	l1d_stall_coh	DP0 : Access Stall Because of conflict with Coherence operation.	Generic	Level
			L1D is stalled due to coherence transaction.	Generic	Level
			DP1 : Access Stall Because of conflict with Coherence operation.	Generic	Level
1	Bank conflict	l1d_stall_bcft	DP0 : CPU-STQ bank Conflict.	Generic	Level
			DP0 : CPU access having bank conflict.	Generic	Level
			DP1 : CPU-STQ bank Conflict.	Generic	Level
			DP1 : CPU access having bank conflict.	Generic	Level
			L1D is stalled due to a bank conflicted with datapaths.	Generic	Level
			L1D is stalled due to a bank confict between any CPU access and a STQ	Generic	Level

Copyright © 2020, Texas Instruments Incorporated

Table 11. DMC Debug Events and Stalls (continued)

No.	Category	Event	Description	Generic / Interest Qualified	Level / Pulse
2	Stall on Cmd Drain	l1d_stall_cmd	L1D is stalled for a read-miss waiting on Draining of Write miss entry.	Generic	Level
			L1D is stalled for a read-miss waiting on Draining of Victim.	Generic	Level
			L1D is stalled for a Write-miss waiting on Draining of Write Miss Buffer Entry.	Generic	Level
			L1D is stalled for a read-miss waiting on Completion of internal coherence updates (global invalidate etc).	Generic	Level
3	Non-availability of blocking	l1d_stall_blocking	L1D is stalled due to non-availability of blocking Command Credit.	Generic	Level
			L1D is stalled due to non-availability of non-blocking Command Credit	Generic	Level
4	Non-availability non-blocking	l1d_stall_non_blocking	L1D is stalled due to non-availability of blocking Data Credit.	Generic	Level
			L1D is stalled due to non-availability of non-blocking Data Credit.	Generic	Level
5	Cache	l1d_dp0_hit_wmb	DP0 : Write Miss Buffer Hit-L1D.	Generic	Pulse
6	Cache	l1d_dp0_hit_main	DP0 : DP0 : Hit-L1D – Main Cache/SRAM.	Generic	Pulse
7	Cache	l1d_dp0_hit_vct	P0 : Hit-L1D – Victim Cache.	Generic	Pulse
8	Cache	l1d_dp0_rnw	DP0 : Read-Not-Write.	Generic	Pulse
9	Cache	l1d_dp1_hit_wmb	DP1 : Write Miss Buffer Hit-L1D.	Generic	Pulse
10	Cache	l1d_dp1_hit_main	DP1 : DP0 : Hit-L1D – Main Cache/SRAM.	Generic	Pulse
11	Cache	l1d_dp1_hit_vct	DP1 : Hit-L1D – Victim Cache.	Generic	Pulse
12	Cache	l1d_dp1_rnw	DP1 : Read-Not-Write.	Generic	Pulse
13	Cache	l1d_dp0_hit_wmb_int	DP0 : Write Miss Buffer Hit-L1D.	interest	Pulse
14	Cache	l1d_dp0_hit_main_int	DP0 : DP0 : Hit-L1D – Main Cache/SRAM.	interest	Pulse
15	Cache	l1d_dp0_hit_vct_int	P0 : Hit-L1D – Victim Cache.	interest	Pulse
16	Cache	l1d_dp0_rnw_int	DP0 : Read-Not-Write.	interest	Pulse
17	Cache	l1d_dp1_hit_wmb_int	DP1 : Write Miss Buffer Hit-L1D.	interest	Pulse
18	Cache	l1d_dp1_hit_main_int	DP1 : DP0 : Hit-L1D – Main Cache/SRAM.	interest	Pulse
19	Cache	l1d_dp1_hit_vct_int	DP1 : Hit-L1D – Victim Cache.	interest	Pulse
20	Cache	l1d_dp1_rnw_int	DP1 : Read-Not-Write.	interest	Pulse
21	Unused				
22	Unused				
23	Unused				

Table 12. UMC Debug Events and Stalls

No.	Category	Event	Description	Generic / Interest Qualified	Level / Pulse
0	Global	l2_cache_glb_reset	L2 Cache Global Reset is in progress	Generic	Level
1	Global	l2_snp_glb_reset	L2 Snoop Cache Global Reset is in progress	Generic	Level

Copyright © 2020, Texas Instruments Incorporated

Table 12. UMC Debug Events and Stalls (continued)

No.	Category	Event	Description	Generic / Interest Qualified	Level / Pulse
2	Stall	l2_l1p_stall_arb	L1P transaction present but is stalled in arbitration	Generic	Level
3	Stall	l2_l1d_stall_arb	L1D transaction present but is stalled in arbitration	Generic	Level
4	Stall	l2_se_stall_arb	SE transaction present but is stalled in arbitration	Generic	Level
5	Stall	l2_cmmu_stall_arb	CMMU transaction present but is stalled in arbitration	Generic	Level
6	Stall	l2_msmc_stall_arb	MSMC transaction present but is stalled in arbitration	Generic	Level
7	Stall	l2_int_mdma_cr_stall	Internal MDMA credit stall	Generic	Level
8	Stall	l2_int_snp_cr_stall	Internal SNOOP credit stall	Generic	Level
9	Stall	l2_int_l1d_resp_cr_stall	Internal DMC RESPONSE credit stall	Generic	Level
10	Stall	l2_int_dma_resp_cr_stall	Internal SDMA RESPONSE credit stall	Generic	Level
11	Arb Event	l2_l1p_won_arb	L1P transaction won arbitration	Generic	Pulse
12	Arb Event	l2_l1d_won_arb	L1D transaction won arbitration	Generic	Pulse
13	Arb Event	l2_se_won_arb	SE transaction won arbitration	Generic	Pulse
14	Arb Event	l2_cmmu_won_arb	CMMU transaction won arbitration	Generic	Pulse
15	Arb Event	l2_msmc_dma_won_arb	MSMC transaction won arbitration	Generic	Pulse
16	Cache Event	l2_cache_hit	Hit L2 Cache for Load/Store/Snoop/Warm	Generic	Pulse
17	Cache Event	l2_cache_miss_evict	Miss L2 Cache with silent or no eviction for Load/Store/Snoop/Warm	Generic	Pulse
18	Cache Event	l2_cache_miss_victim_wb	Miss L2 Cache with victim writeback for Load/Store/Snoop/Warm	Generic	Pulse
19	Cache Event	l2_cache_cmo_miss	Miss L2 Cache for CMO/Global Coherence	Generic	Pulse
20	Cache Event	l2_cache_cmo_hit_no_wb	Hit L2 Cache with no victim writeback oreviction for CMO/Global Coherence	Generic	Pulse
21	Cache Event	l2_cache_cmo_hit_wb	Hit L2 Cache with victim writeback for CMO/Global Coherence	Generic	Pulse
22	Coherence Event	l2_snp_hit	Hit L2 Snoop Cache	Generic	Pulse
23	Coherence Event	l2_snp_l1d_miss	Snooped L1D but missed L1D cache	Generic	Pulse
24	Arb Event	l2_l1d_won_arb	L1D transaction won arbitration	Interest Qualified	Pulse
25	Arb Event	l2_se_won_arb	SE transaction won arbitration	Interest Qualified	Pulse
26	Arb Event	l2_cmmu_won_arb	CMMU transaction won arbitration	Interest Qualified	Pulse
27	Arb Event	l2_msmc_dma_won_arb	MSMC transaction won arbitration	Interest Qualified	Pulse
28	Cache Event	l2_cache_hit	Hit L2 Cache for Load/Store/Snoop/Warm	Interest Qualified	Pulse
29	Cache Event	l2_cache_miss_evict	Miss L2 Cache with silent or no eviction for Load/Store/Snoop/Warm	Interest Qualified	Pulse
30	Cache Event	l2_cache_miss_victim_wb	Miss L2 Cache with victim writeback for Load/Store/Snoop/Warm	Interest Qualified	Pulse
31	Cache Event	l2_cache_cmo_miss	Miss L2 Cache for CMO/Global Coherence	Interest Qualified	Pulse

Copyright © 2020, Texas Instruments Incorporated

Table 12. UMC Debug Events and Stalls (continued)

No.	Category	Event	Description	Generic / Interest Qualified	Level / Pulse
32	Cache Event	l2_cache_cmo_hit_no_wb	Hit L2 Cache with no victim writeback oreviction for CMO/Global Coherence	Interest Qualified	Pulse
33	Cache Event	l2_cache_cmo_hit_wb	Hit L2 Cache with victim writeback for CMO/Global Coherence	Interest Qualified	Pulse
34	Coherence Event	l2_snp_hit	Hit L2 Snoop Cache	Interest Qualified	Pulse
35	Coherence Event	l2_snp_l1d_miss	Snooped L1D but missed L1D cache	Interest Qualified	Pulse
36	Unused				
37	Unused				
38	Unused				
39	Unused				

1.1.6 Cache Tag View Debug

Cache tag view access provides an ECR-based functional path to read tags directly from the caches for debug. This is different from the C66x implementation, which uses the pbist data path to implement this feature.

1.1.6.1 Cache Tag View Micro-Architecture

This feature requires one ECR register to be mapped for each IP. For the memory system, there are three such registers:

- L1PCTAG (0x244)
- L1DCTAG (0x206)
- L2CTAG (0x286)

This is the protocol followed:

1. The CPU issues an ECR READ to this register, with the following attributes:
   - ecr_caddress: address for the CTAG register
   - ecr_index: set, way, and victim cache information

Table 13. ECR Index for Cache Tag View

INDEX	DETAILS	IP Specific Details
[15:14]	Unused	00
[13:10]	Way / Victim Cache Entry	L1P • [13:12]: 00 • [11:10]: Way (4 ways) L1D • [10]: Main/Victim cache • 1: Victim Cache • 0: Main Cache [L2 • [13]: 0 • [12:10]: Way (8 ways)

Copyright © 2020, Texas Instruments Incorporated

Table 13. ECR Index for Cache Tag View (continued)

INDEX	DETAILS	IP Specific Details
[9:0]	Cache Set	L1P • [9:7]: 000 • [6:0]: Set L1D • [9:8]: 00 • [7:0]: Set when main cache is selected • [3:0]: Victim cache entry when victim cache is selected L2 • [9:0]: Set

- ecr_cpriv: Priv attribute
- ecr_csecure: Secure attribute
- ecr_cdir: ‘1’ for READ

2. This read initiates a read to the tag and other cache-related memory RAMs.
   - Each IP arbitrates the cache_tag read with other pipeline transactions.
   - Decide the priority of cache tag read access. In all the memory controllers, because the arbitration unit for the tag RAM is usually accessed from various masters simultaneously, the controllers might need an internal register to hold the Cache Tag address (set, way, and so forth) while it is stalled during arbitration. This is an internal micro-architectural register specific to that IP’s implantation.
3. Mux out, tag, csecure, cpriv, valid and all other attributes specific to the block, using the way+bank information and store this in the <CACHE>CTAG ECR register.
4. Compare csecure, cpriv fields that came in with the ECR access and <CACHE>CTAG ECR register. If they match, ecr_pmc_rstatus = 0 (Pass), else ecr_pmc_rstatus = 1 (Fail).
5. Assert ecr_pmc_cready (if deasserted) and ecr_pmc_ack. Drive the contents of the <CACHE>CTAG ECR register on the ecr_rdata bus.

1.1.6.2 Cache Tag View Security and Permissions

Each memory controller compares the privilege and secure attributes of the held tag to the requesting ECR transaction to detect if access is allowed. For memory controllers which do not cache these attributes, a more strict check is done based on the ECR register permissions. Hierarchical access permissions are not defined (such as access for root supervisor to access non-secure user tags) . All three ECR registers are read only. Non-secure debug accesses are allowed to these registers when the cache line is not secure.

1.1.6.3 Cache Tag View Register Formats

The contents of the cache are held in the ECR registers and are described below.

Table 14. L1PCTAG Register Details

REG NAME	63	52	51	50	49	48	13	12	0	REG ID
L1PCTAG	Reserved		VALID	TABLE BASE	CSECURE	TAG		Reserved		0x244
	R, +0		R, +0	R, +0	R, +0	R, +0	R, +0	R, +0		

Table 15. L1PCTAG Field Descriptions

Bit	Field	Description
12-0	Reserved	Reads return 0
48-13	TAG	Tag for cached line
49	CSECURE	Secure bit for cached line

Copyright © 2020, Texas Instruments Incorporated

Table 15. L1PCTAG Field Descriptions (continued)

Bit	Field	Description
50	TABLE BASE	Privilege bits for cached line
51	VALID	Line is present in the cache
63-52	Reserved	Reads return 0

Table 16. L1DCTAG Register Details

REG NAME	63	52	51	50	49	48	7	6	0	REG ID
L1DCTAG	Reserved		MESI		CSECURE	TAG		Reserved		0x206
	R, +0		R, +0		R, +0	R, +0		R, +0		

Table 17. L1DCTAG Field Descriptions

Bit	Field	Description
6-0	Reserved	Reads return 0
48-7	TAG	[48:13]: Valid TAG bits when L1D Main Cache is being read and the L1D Main Cache is 8K[1] [48:14]: Valid TAG bits when L1D Main Cache is being read and the L1D Main Cache is 16K [48:15]: Valid TAG bits when L1D Main Cache is being read and the L1D Main Cache is 32K [48:7]: Valid TAG bits when L1D Victim Cache is being read[2]
49	CSECURE	0 : Cache line is non-secure 1 : Cache line is secure
51-50	MESI	MESI bits for cached line 00 : Line in not present in the cache 01: Line is present in the cache in the Shared state 10: Line is present in the cache in the Exclusive state 11: Line is present in the cache in the Modifixed state
63-52	Reserved	Reads return 0

[1] Bits [48:44] return 0.

[2] Bits [12:7] return 0 when L1D main cache is being read.

Table 18. L2CTAG Register Details

REG NAME	63	53	51	50	49	48	12	11	0	REG ID
L2CTAG	Reserved		MESI		CSECURE	TAG		Reserved		0x286
	R, +0		R, +0		R, +0	R, +0		R, +0		

Table 19. L2CTAG Field Descriptions

Bit	Field	Description
11-0	Reserved	Reads return 0
48-12	TAG	[48:11]: Valid TAG bits when L2 Cache is 32K [48:12]: Valid TAG bits when L2 Cache is 64K [48:13]: Valid TAG bits when L2 Cache is 128K [48:14]: Valid TAG bits when L2 Cache is 256K [48:15]: Valid TAG bits when L2 Cache is 512K [48:16]: Valid TAG bits when L2 Cache is 1024K
49	CSECURE	Secure bit for cached line
51-50	MESI	MESI bits for cached line 00 : Line in not present in the cache 01: Line is present in the cache in the Shared state 10: Line is present in the cache in the Exclusive state 11: Line is present in the cache in the Modified state

Copyright © 2020, Texas Instruments Incorporated

Table 19. L2CTAG Field Descriptions (continued)

Bit	Field	Description
63-52	Reserved	Reads return 0

NOTE: Cache sizes: the cache sizes are software programmable, and changing them impacts the total number of sets. When the set requested is larger than the current cache size, the sets wrap around, and the tag information for that (aliased) set is recorded.

NOTE: L1D victim cache: the L1D victim cache and Write Miss Buffer (WMB) share a physical memory. The WMB is not tracked through this feature.

1.1.7 ECR Registers

The MMR (Memory Mapper registers) have been renamed to ECR (Extended CPU registers), but are functionally similar in the context of the C7x memory system. They are used to configure the behavior of various functions, and record information useful to the user. The ECR registers are described in detail in each IP Specification Document.

1.1.8 Aliased Address Mode

The C7x L2 memory supports an aliased addressing mode.

Two sets of buffers are supported:

- Working buffer
- Image buffer

The working buffer (WBUF) is the primary location for the lookup tables and other relatively long lived data buffers. The L2 controller does not natively support the lookup table feature. That is supported by the L1 data controller. The WBUF is the location in the L2 memory which holds the data for the tables, which is then brought into the L1 data memory, or by the streaming engine.

In the context of this feature, accesses from CPU refer to both from L1D and from the streaming engine (SE). The SE accesses are handled by the L2 memory controller (UMC), while CPU accesses are handled by the L1D data memory controller (DMC). SDMA accesses are handled by the UMC.

The image buffers (IBUF) are the primary location for the time-slot based data buffers. These buffers allow ping-pong ownership/accesses by the SDMA and C7x CPU core. On a time-slot basis, each of the respective image buffers can be assigned ownership to either SDMA or the C7x CPU core. Four such buffers are supported:

- IBUFLA : Image Buffer Low - copy A
- IBUFLB : Image Buffer Low - copy B
- IBUFHA : Image Buffer High - copy A
- IBUFHA : Image Buffer High - copy B

A and B are ping-pong copies. The L and H memory address regions are intended to allow for double read and write bandwidth. The pipelined and multi-bank architecture of the C7x L2 memory natively supports this parallel access to the buffers, and thus unlike the VCOP, no extra hardware is required for supporting these buffers.

All these buffers together span the entire L2 SRAM memory range. The start of every buffer is required to be TBD. A further requirement is that this memory block start from a 128KB aligned address. Table 20 shows the L2 memory map for this mode.

Copyright © 2020, Texas Instruments Incorporated

Table 20. C7x L2 Memory Map for Aliased Addressing

L2 ADDR	BUFFER	SIZE	L2 ADDR[16]
END OF L2SRAM	WBUF	32KB - MAX	Vector Working Buffer, 32KB - MAX MAX = (4MB - 128KB)
82_0000			
81_0080	IBUFHB	32KB	1 - Image buffer high copy B, 32KB
81_0000	IBUFLB	32KB	1 - Image buffer low copy B, 32KB
80_0080	IBUFHA	32KB	0 - Image buffer high copy A, 32KB
80_0000	IBUFLA	32KB	0 - Image buffer low copy A, 32KB

NOTE: Buffer WBUF extends from the end of IBUFHB to the end of L2 SRAM.

Figure 5. Non-Aliased and Aliased Views

The aliased view is usually leveraged in tandem, by both the CPU core and the SDMA engine, which allows the user to use the IBUFLA/B as aliases by using one virtual address, namely the IBUFA address only; the core / L2 controller then determines whether the user is looking at the A copy or the B copy of the buffer based on the buffer switch CPU_MSW_CTL / L2_MSW_CTL.

1.1.8.1 Memory Map Control Register

The buffer ownership and address aliasing feature is supported in hardware. The actual implementation may be divided across the CPU/L1D and L2 controller. The ECR registers below are used by the C7x CPU and L2 memory controller for the ownership and address aliasing control.

The buffer ownership checks are done only when aliasing is enabled.

The memory-map control register is shown in Table 21 and described in Table 22. There are two copies of this register, one in the CPU - CPU_MEMMAP and the second in the UMC controller - L2_MEMMAP.

NOTE: In all instances where a copy of the ECR register exists in both the CPU and UMC, the user is responsible for programming them with the same values.

Table 21. Memory Map Control Register (L2_MEMMAP / CPU_MEMMAP)

REG NAME	63	5	4	3	1	0	REG ID
L2_MEMMAP / CPU_MEMMAP	RESERVED		LCL_SDMA_ALIAS	RESERVED		CPU_ALIAS	0x297 / cpu tbd
	R,+0		RW,+0	R,+0		RW,+0	

Copyright © 2020, Texas Instruments Incorporated

Table 22. L2/CPU Memory-Map Control Register (L2_MEMMAP / CPU_MEMMAP) Field Descriptions

Bit	Field	Description
63-5	Reserved	Reserved
4	LCL_SDMA_ALIAS	Local SDMA View 0 = Local SDMA views full memory map 1 = Local SDMA views aliased memory map. In this mode, SDMA views IBUFLA and IBUFLB at the same address, and views IBUFHA and IBUFHB at the same address. In this mode, only one IBUFLA or IBUFHLB is owned by the system, and only one IBUFHA or IBUFHB is owned by the system. Software must poll for updated value to ensure that the mode change has taken effect.
3-1	Reserved	Reserved
0	CPU_ALIAS	CPU View 0 = CPU views full memory map CPU views aliased memory map. In this mode, the CPU views IBUFLA and IBUFLB at the same address, and views IBUFHA and IBUFHB at the same address. In this mode, only one IBUFLA or IBUFHLB is owned by the cpu, and only one IBUFHA or IBUFHB is owned by the cpu. Software must poll for updated value to ensure that the mode change has taken effect.

1.1.8.2 Memory Switch Control Register

The memory switch control register (L2_MSW_CTL / CPU_MSW_CTL) is shown in Table 23 and described in Table 24.

Table 23. L2/CPU Memory Switch Control Register (L2_MSW_CTL / CPU_MSW

REG NAME	63	17	16	15	13	12	11	9	8	7	5	4	3	1	0	REG ID
L2_MSWCTL / CPU_MSWCTL	RESERVED		WBUF	RESERVED		IBUFHB	RESERVED		IBUFLB	RESERVED		IBUFHA	RESERVED		IBUFLA	0x298 / cpu tbd
	R,+0		RW,+0	R,+0		RW,+0	R,+0		RW,+0	R,+0		RW,+0	R,+0		RW,+0	

Table 24. L2/CPU Memory Switch Control Registers (L2_MSW_CTL / CPU_MSW_CTL) Field Descriptions

Bit	Field	Description
63-17	Reserved	Reserved
16	WBUF	Working buffer ownership 0 = DMA owned 1 = CPU owned
15-13	Reserved	Reserved
12	IBUFHB	Image Buffer High-B ownership 0 = DMA owned 1 = CPU owned
11-9	Reserved	Reserved
8	IBUFLB	Image Buffer Low-B ownership 0 = DMA owned 1 = CPU owned
7-5	Reserved	Reserved
4	IBUFHA	Image Buffer High-A ownership 0 = DMA owned 1 = CPU owned
3-1	Reserved	Reserved
0	IBUFLA	Image Buffer Low-A ownership 0 = DMA owned 1 = CPU owned

Copyright © 2020, Texas Instruments Incorporated

1.1.8.3 Memory Map Base Addresses

NOTE: This feature, which provides the ability to change the starting address of each buffer, is not supported in the first version of the C7x. Instead, the starting address is fixed to the base of the L2 memory.

The base addressed for the IBUF and WBUF buffers are not fixed, and are configurable through the ECR registers. This is different from EVE, where the base addresses are fixed. Some assumptions have been made:

- The size of each of the IBUF buffers is fixed at 32KB.
- The WBUF buffer extends to the end of the L2 SRAM address range, and can, thus, vary from 64KB to (4MB - 128KB).
- The base addresses are aligned at a 32KB boundary.

This may be further modified based on architectural simplifications:

- Separate control for the four IBUF base address may not be required, and it may be sufficient to have a fixed relative offset between the four addresses, or for them to be contiguous.
- The Low and High halves of the buffers may be assumed to be contiguous.
- The base address for WBUF may not be required to be configurable, and can be static. If static, it must be communicated to the compiler.

The base addresses are programmed by ECR registers and are described in Table 25, Table 26, and Table 27.

Table 25. IBUFA Memory Base Address Register (L2_MEMMAP_IBUFA / CPU_MEMMAP_IBUFA)

REG NAME	63	47	46	32	31	15	14	0	REG ID
L2_MEMMAP_IBUFA / CPU_MEMMAP_IBUFA	IBUFHA_BASE		RESERVED		IBUFLA_BASE		RESERVED		0x299 / cpu tbd
	RW,+0		R,+0		RW,+0		R,+0		

Table 26. IBUFB Memory Base Address Register (L2_MEMMAP_IBUFB / CPU_MEMMAP_IBUFB)

REG NAME	63	47	46	32	31	15	14	0	REG ID
L2_MEMMAP_IBUFB / CPU_MEMMAP_IBUFB	IBUFHB_BASE		RESERVED		IBUFLB_BASE		RESERVED		0x29A / cpu tbd
	RW,+0		R,+0		RW,+0		R,+0		

Table 27. WBUF Memory Base Address Register (L2_MEMMAP_WBUF / CPU_MEMMAP_WBUF)

REG NAME	63	32	31	15	14	0	REG ID
L2_MEMMAP_WBUF / CPU_MEMMAP_WBUF	RESERVED		WBUF_BASE		RESERVED		0x29B / cpu tbd
	R,+0		RW,+0		R,+0		

Copyright © 2020, Texas Instruments Incorporated

1.1.9 Memory Switch Error Status Register

Both the CPU and L2 controller check for ownership of the buffer before committing the access. If the requestor does not have ownership of that buffer, an error is triggered and an exception or event taken. In addition, the address range is also checked for out of bound accesses. This is done on loads and stores in both the CPU and the L2 controller.

The memory switch error status register (L2_MSW_ERR / CPU_MSW_ERR) is shown in Table 28 and described in Table 29.

Table 28. Memory Switch Error Status Register

REG NAME	63	8	7	6	5	4	3	2	1	0	REG ID
L2_MSW_ERR_STAT / CPU_MSW_ERR_STAT	RSVD		ST_WBUF_ERR	LD_WBUF_ERR	ST_IBUF_ERR	LD_IBUF_ERR	RSVD	DMA-_ERR	CPU_ERR	RSVD	0x29C / cpu tbd
	R,+0		R,+0	R,+0	R,+0	R,+0	R,+0	R,+0	R,+0	R,+0	

Table 29. Memory Switch Error Status Register (L2_MSW_ERR / CPU_MSW_ERR) Field Descriptions

Bit	Field	Description
63-8	Reserved	Reserved
7	ST_WBUF_ERR	Store to WBUF is outside address range
6	LD_WBUF_ERR	Load to WBUF is outside address range
5	ST_IBUF_ERR	Store to IBUF is outside address range
4	LD_IBUF_ERR	Load to IBUF is outside address range
3	Reserved	Reserved
2	DMA_ERR	DMA initiated buffer ownership error
1	CPU_ERR	CPU initiated buffer ownership error
0	Reserved	Reserved

1.1.9.1 Memory Switch Error Clear Register

The error status is cleared by writing a 1 to the error clear register. The memory switch error clear register (L2_MSW_ERR_CLR / CPU_MWS_ERR_CLR) is shown in Table 30 and described in Table 31.

Table 30. Memory Switch Error Clear Register

REG NAME	63	8	7	6	5	4	3	2	1	0	REG ID
L2_MSW_ERR_CLR / CPU_MSW_ERR_CLR	RSVD		ST_WBUF_CLR	LD_WBUF_CLR	ST_IBUF_CLR	LD_IBUF_CLR	RSVD	DMA_CLR	CPU_CLR	RSVD	0x29D / cpu tbd
	R,+0		R,+0	R,+0	R,+0	R,+0	R,+0	R,+0	R,+0	R,+0	

Table 31. Memory Switch Error Clear Register (L2_MSW_ERR_CLR / CPU_MSW_ERR_CLR) Field Descriptions

Bit	Field	Description
63-8	Reserved	Reserved
7	ST_WBUF_CLR	Writing a 1 clears error condition ST_WBUF_ERR

Copyright © 2020, Texas Instruments Incorporated

Table 31. Memory Switch Error Clear Register (L2_MSW_ERR_CLR / CPU_MSW_ERR_CLR) Field Descriptions (continued)

Bit	Field	Description
6	LD_WBUF_CLR	Writing a 1 clears error condition LD_WBUF_ERR
5	ST_IBUF_CLR	Writing a 1 clears error condition ST_IBUF_ERR
4	LD_IBUF_CLR	Writing a 1 clears error condition LD_WBUF_ERR
3	Reserved	Reserved
2	DMA_CLR	Writing a 1 clears error condition DMA_ERR
1	CPU_CLR	Writing a 1 clears error condition CPU_ERR
0	Reserved	Reserved

1.1.9.2 Memory Switch Error Address Register

The memory switch error address regiser (L2_MSW_ERRADDR / CPU_MSW_ERRADDR) records the physical address of the offending transaction. This is not the aliased address.

Table 32.

REG NAME	63	44	43	0	REG ID
L2_MSW_ERRADDR / CPU_MSW_ERRADDR	RESERVED		ADDR		0x29E / cpu tbd
	R,+0		R,+0		

1.1.9.3 Implementation (Micro-Architectural Detail)

The implementation of this feature is split across the L1D memory controller (DMC) and the L2 memory controller (UMC). DMC handles the load and store accesses, while UMC handles streaming engine reads and SDMA read and writes.

These are the actions taken by the CPU:

- The CPU does the virtual to physical address translation of the address presented to it. It also checks for any permission violations based on the page table attributes.
- If aliasing is enabled, that is, if CPU_ALIAS from the register CPU_MEMMAP is ‘1’, then the CPU performs the address aliasing. This is described in **Table 2-53**.
- The CPU then performs the ownership and address range checks. In case of an error, the transaction is killed and the corresponding error event is signalled. The appropriate response is returned to the requestor.

Similar actions are performed by the L2 memory controller for the DMA-initiated accesses. In the non-aliased mode, the address is not modified. For both the CPU and DMA accesses, an extra step in the L2 controller is the firewall check, which is performed on all accesses to the L2 memory. VCOP Aliasing for loads and stores is done in DMC, and has some exceptions as described below. VCOP Aliasing for streaming engine loads and DMAs is done in UMC, and follows the tables below, except for the WBUF behavior, which is also described below.

Some usage restrictions are listed here:

- For the address being aliased, the virtual address should be equal to the physical address. This must be restricted through MMU programmation.
- WBUF ownership is ignored by both UMC and DMC. The error bits reflect out of bound operations, and are captured in the addressing error register.
- For CPU access to DMC, on a non-aligned access to the last line in some configurations, aliasing may not be as expected. TI recommends that the last line of the buffer not be used for this reason. Table 33 shows the actual behavior. SE and DMAs are not affected by this.
- In Table 34, for invalid ownership programming, DMC does not take an exception. In the conditions

Copyright © 2020, Texas Instruments Incorporated

marked as ERR, DMC does not alias and does not take an exception. Thus, the logging registers do not record load / store vcop errors.

Table 33. DMC VCOP Aliasing Behavior in Some Configurations

	IBUFLA	IBUFHA	IBUFLB	IBUFHB	DMC Action
Owned	CPU	CPU	DMA	DMA	No issue
	DMA	DMA	CPU	CPU	No issue
	DMA	CPU	CPU	DMA	
[2]	[1]CPU	DMA	CPU	DMA	

[1] On a non-aligned access to the last line in IBUFLA, where the line spills into IBUFHA, both lines are aliased.

[2] On a non-aligned access to the last line in IBUFLA, where the line spills into IBUFHA, both lines are not aliased.

Table 34 shows the CPU view and actions in the aliased mode, while Table 35 shows the SDMA view and actions in the aliased mode.

Table 34. CPU Actions in Aliased Mode

	OWNERSHIP				CPU ADDR[16] to L2 ADDR [16] MAPPING			
					80_0000-80_7FFF	80_8000-80_FFFF	81_0000-81-7FFF	81_8000-81_FFFF
CPU_ALIAS	**IBUFLA**	**IBUFHA**	**IBUFLB**	**IBUFHB**	**0**	**0**	**1**	**1**
1	0	0	0	0	ERR	ERR	Reserved	
1	0	0	0	1	ERR	1		
1	0	0	1	0	1	ERR		
1	0	0	1	1	1	1		
1	0	1	0	X	ERR	0		
1	0	1	1	X	1	0		
1	1	0	X	0	0	ERR		
1	1	0	X	1	0	1		
1	1	1	X	X	0	0		

Table 35. DMA Actions in Aliased Mode

	OWNERSHIP				DMA ADDR[16] to L2 ADDR [16] MAPPING			
					80_0000-80_7FFF	80_8000-80_FFFF	81_0000-81-7FFF	81_8000-81_FFFF
DMA_ALIAS	**IBUFLA**	**IBUFHA**	**IBUFLB**	**IBUFHB**	**0**	**0**	**1**	**1**
1	0	0	X	X	0	0	Reserved	
1	0	1	X	0	0	1		
1	0	1	X	1	0	ERR		
1	1	0	0	X	1	0		
1	1	0	1	X	ERR	0		
1	1	1	0	0	1	1		
1	1	1	0	1	1	ERR		
1	1	1	1	0	ERR	1		
1	1	1	1	1	ERR	ERR		

Copyright © 2020, Texas Instruments Incorporated

2 Program Memory Controller (PMC)

The Corepac2 L1P cache controller supports a fixed cache size of 32KB.

The purpose of the Level 1 program cache (L1P) is to maximize performance of the code execution. L1P cache is necessary to facilitate fetching program code at a fast clock rate to maintain a large system memory. The cache is responsible for hiding the latency associated with executing code from the slower system memory.

2.1 Program Memory Controller Features

The following are the features supported by the PMC controller:

- L1P cache
  - 32KB L1P cache – size is not configurable
  - 4-way set associative
  - 64-byte cache line size
  - Virtually indexed and virtually tagged (48-bit virtual address)
  - L1P SRAM mode is not supported either from the CPU or DMA
- Prefetch
  - Auto prefetching on L1P misses from L2
  - Capability to queue multiple fetch packet requests (64 bytes) to UMC to enable deeper prefetch in the program pipeline.
- Error detection (ECC)
  - Parity protection on data and TAG RAMs. 1-bit error detection.
  - Data RAM parity protection is on instruction width granularity (1 parity bit every 32 bits)
  - Auto-invalidate and re-fetch on errors in TAG RAM
  - Aligns to safety architecture provided testing and mechanism for ECC detection logic
  - Aligns to safety architecture for reporting of detected parity and ECC faults from both error injection and functional access
- Software-initiated coherence operations
  - Single-cycle cache invalidate with support for three modes – all cache lines, MMU page table base 0, and MMU page table base 1
- Virtual memory
  - Virtual to physical addressing on misses
  - uTLB to handle address translation and for code protection
- Debug
  - Pass/Fail status is returned to on all reads.
  - Events are exported out for interesting conditions that can be used for debug and performance measurements.
  - Cache Tag View is visible through the ECR registers, and lets the user access the contents of the TAG RAMs.
- Extended Control register (ECR – previously MMR) access

2.2 Program Memory Controller Interfaces

Figure 6 shows the main sub-modules in PMC.

Copyright © 2020, Texas Instruments Incorporated

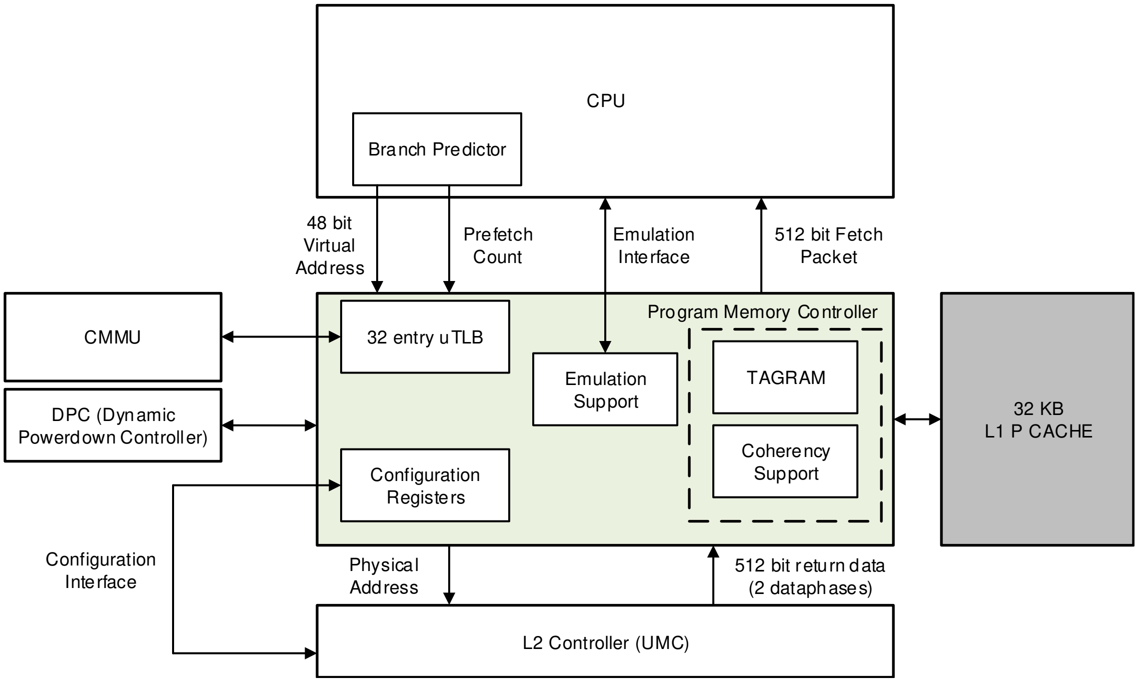


Figure 6. L1P Controller Block Diagram

2.2.1 CPU-PMC Interface

There are two types interfaces from the CPU to PMC:

- Functional / Fetch – The interface is used for program fetches, and consists of the following, as a minimum:
  - 512-bit CPU read path
  - CPU to PMC fetch address
  - Any required additional handshaking
- Emulation – The interface is used for EDI accesses, and consists of the following, as a minimum:
  - CPU to PMC fetch address
  - Signalling used for cache invalidate
  - Any required additional handshaking

2.2.2 PMC-UMC Interface

The UMC to PMC interface is based on the VBUSM.C interface protocols, and consists of the following, at a minimum:

- 512-bit PMC read path
- PMC to UMC fetch address
- Any required additional handshaking

2.2.3 PMC-CMMU Interface

The PMC (uTLB) to MMU interface consists of the following, at a minimum:

- Virtual page address
- Translated physical address and page attributes
- Any required additional handshaking

Copyright © 2020, Texas Instruments Incorporated

2.3 L1P Memory

PMC supports 32KB cache only. PMC does not support SRAM mode.

2.4 PMC ECR Register

The PMC ECR register is described in Table 36. This register is directly accessible by the CPU and is not memory mapped, but still resides physically inside L1P.

Table 36. L1P Registers Summary

Type of Instruction	Instruction	Description
Cache Tag View	L1PCTAG	“Cache Tag View Debug” section in the L2 Caches Intro spec

2.5 L1P Configuration Inputs

No additional configuration inputs, such as region, adjust, and so forth, are required for PMC.

2.6 L1P Virtualization Support

2.6.1 L1P Address Decoding

No address decoding is required in PMC. This is taken care of by the virtualization support in PMC.

2.6.2 MicroTLB Integration

A MicroTLB is integrated into PMC.

2.7 L1P Cache Architecture

The Corepac2 L1P cache is a virtually indexed and virtually tagged cache. The L1P cache is a 4-way cache, meaning that every physical memory location in the system has four possible locations in the cache where it may reside. When the CPU attempts to fetch a piece of code, L1P must check whether the requested address resides in the L1P cache. To do so, the 48-bit virtual address provided by the CPU is partitioned into three fields (tag, set, and offset), as shown in Figure 7.

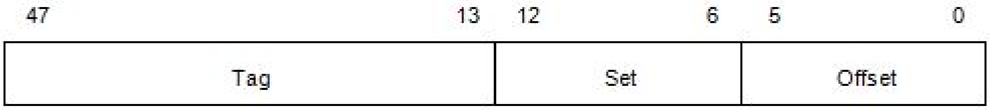


Figure 7. L1P Cache Organization

The offset of 6 bits accounts for the fact that an L1P line size is 64 bytes. The cache control logic ignores bits 0 through 5 of the address. The set field indicates the L1P cache line address where the data would reside, if present in cache. The width of the set field is 7 bits. L1P uses the set field to look up and check the tag for any already-cached data from that address. The valid bit is then used to indicate whether the entry held in cache is valid or not. The tag field is the upper portion of the address that is used along with the set field to identify the true physical location of the data element.

On a program fetch, if the tag matches and the corresponding valid bit is set, then it is a “hit,” and the data is read directly from the L1P cache location and returned to the CPU. Otherwise, it is a “miss” and the request is sent on to the L2 controller for the data to be fetched from its location in the system. Misses may or may not directly result in CPU stalls. The CPU cannot write data to L1P

Replacement and allocation policy: The L1P cache operates as a 4-way cache. This means that each location in system memory can reside in four locations in the L1P cache. L1P cache follows the FIFO replacement policy.

Copyright © 2020, Texas Instruments Incorporated

Each newly cached line replaces the first line allocated to a specific set. Since The L1P controller implements a read-allocate cache. This means that the L1P fetches a complete line of 64 bytes on a read miss.

L1P coherence support: Corepac2 supports full hardware coherency. L1P cache is virtually indexed and virtually tagged and there are no writes to L1P Cache. Thus the coherence support required from the L1P cache controller is minimal.

2.8 Program-Initiated Cache Invalidate Operation

Cache Invalidate operations synchronize L1P with the system between major events, such as a task switch, or change to memory protection settings. Thus, cache invalidate operations are viewed as synchronous with respect to other program activity.

This is initiated by a CPU transaction. There are three flavors, and the CPU-PMC interface has signalling to indicate which type of cache invalidate is being asked for.

- Full L1P cache – all lines in the cache are invalidated
- Cache lines associated with MMU Page Table Base 0
- Cache lines associated with MMU Page Table Base 1

2.9 L1P Error Detection Support

The PMC supports error detection on program fetches. On a fetch, PMC forwards all parity bits of the line to CPU. The CPU does the actual parity check. The following sections define the operation of the error detection support.

2.9.1 DATA RAM

16 parity bits are stored in the data RAM for each 512-bit fetch packet (on instruction width granularity) and are generated by the L2 cache controller. The L2 cache controller provides the 16 parity bits.

2.9.2 TAG RAM

PMC checks parity on every access to the TAG RAM. In case of a parity mismatch, PMC invalidates and re-fetches the cache line.

2.9.3 Functional Test for EDC Logic

PMC provides testing mechanisms for the EDC detection and correction logics.

2.9.4 Error Capture and Reporting

PMC reports detected parity and EDC faults both from scrubbing and functional access.

2.10 L1P Emulation Support

PMC supports software breakpoints through EDI accesses from the CPU, which invalidate lines from the L1P cache.

2.11 L1P Trace and Events Support

Details of the events are in the Caches Introduction Architectural Specification Document.

2.12 L1P Cache Powerdown Support

Refer to the Corepac Powerdown Chapter on L1P Powerdown Support.

Copyright © 2020, Texas Instruments Incorporated

3 L1 Data Cache

The purpose of the L1D memory and cache is to maximize performance of the data processing. The cache is necessary to facilitate reading and writing data at the full CPU clock rate, while still having a large system memory. It is the cache's responsibility to hide much of the latency associated with reading from and writing to the slower system memory.

In the C7x processor, the L1 data cache is a non-blocking cache with a maximum of 32Kbyte of cache supported. The L1 data cache also provides support for victim cache. Figure 8 shows the top level view of the C7x with the L1 data cache controller highlighted.

Figure 8. L1 Data Cache Controller Instantiation in C7x

3.1 Features

The C7x Level 1 data cache (L1D) controller (DMC) supports the following features:

- L1D cache
  - 32KB cache, configurable down to 8KB cache
  - Dual datapath (DP0+DP1) support similar to C6x
  - Datapath configuration supported (64 bit+ 512 bit)
  - 512-bit vector datapath with 128 byte cache line size
  - Read allocate cache
  - Support for write-back and write-through modes (write-through mode is not supported in the first generation, and functions like write-back)
  - 16-entry victim cache
  - Physically indexed, physically tagged
  - Hit under miss
  - Posted write miss support
  - Write merging on all outstanding write transactions inside L1D
  - FENCE operation on outstanding transactions, with support for two types of store tags
  - Auto-flush and idle-flush
- L1D SRAM
  - Support for DMA for CPU reads and writes to L1D SRAM.
- Lookup table and histogram
  - Capability to support up to 16 parallel table lookup
  - Histogram support
  - CPU LUTINIT instruction can update L1D with lookup table data, with support for a mode to provide duplication of data within L1D.
  - Support for both signed and unsigned bin and signed weights for histogram
  - Saturation and non-saturation mode for all combinations of bin type and weights for histogram
- LUTWR in L1D SRAM

Copyright © 2020, Texas Instruments Incorporated

  - This feature adds the capability to service multiple parallel stores to different addresses in L1D RAM.
  - L1D SRAM is dynamically configured into multiple regions and ways based on the degree of parallelism required.
- Bandwidth
  - 64 + 512-bit CPU load / store bandwidth
  - 1024-bit L1D RAM bandwidth
  - 16 × 64-bit wide banks
- Error detection (ECC)
  - Full ECC on data RAMs: 1-bit error correction and 2-bit error detection
  - Parity-only protection on TAG RAMs
  - Provide ECC syndrome on writes and victims out to L2
  - Receives ECC syndromes with read data from L2, and does detection and correction before presenting this data to the CPU
  - Full ECC on victim cache
  - Read-modify-Write support to prevent parity corruption on partial line writes
  - L2-L1D I/F correction for Read-Response data pipeline ECC protection
  - Aligns to Keystone 3 safety architecture provided testing and mechanism for ECC detection logic.
  - Aligns to Keystone 3 safety architecture for reporting of detected parity and ECC faults from both error injection and functional access.
- Emulation and debug capabilities
  - DAS codes are returned on reads to indicate the level of cache that the data was read from.
  - Events are exported out for interesting conditions that can be used for debug and performance measurements.
  - Events are exported out for transactions qualified with the interest attribute. This can be used to filter out transactions based on user-programmable criteria.
  - Cache Tag View is visible through the ECR registers, and lets the user access the contents of the TAG RAMs
  - Hardware watch point (HWWP) and command interest (cinterest) matching compare for HWWP criteria’s, and interest address matching generation.
  - CPU-initiated load / store command interest generation
- Atomic
  - Compare and swap
  - Swap
- Coherence
  - Fully MESI support in both main and victim cache. The shared state is not used by the cache controllers, but that is a micro-architectural detail not visible to the user.
  - Support for global cache coherence operations
  - Snoops and cache maintenance operation support from L2
  - Snoops for L2 SRAM, MSMC SRAM, and external (DDR) addresses
  - Full TAG-RAM comparisons on snoop and cache maintenance operations
- Virtual memory support
  - Support for wider (40-bit) physical address
- Extended Control register (ECR – previously MMR) access
  - L1D ECR registers are accessible from the CPU through a non-pipelined interface that is similar to the CPU-ERI interface present on previous Corepac generations. These registers are not memory mapped, and instead are mapped to a MOVC CPU instruction.
- L2 address aliasing

Copyright © 2020, Texas Instruments Incorporated

- – Support for VCOP address aliasing mode
- – Support for aliasing for multiple separate buffers – IBUFAH, IBUFAL, IBUFBH, IBUFBL buffers
- – Out of range and ownership check for all buffers

3.2 Register Summary

DMC configuration registers are accessed through the internal configuration interface. From the CPU view, these are not memory mapped, but, instead, are mapped to CPU register space, and accessed by CPU instructions. The complete set of ECR registers in DMC are described in Table 49 and Table 50.

3.3 L1 Data Cache Functionality

3.3.1 L1D Configurations

A series of boundary signals configure the L1 data cache (L1D). These signals may be configured by tying them to fixed values, by eFuses or by another mechanism external to the memory system.

L1D can support up to 48KB of total memory. Maximum cache supported by L1D is 32KB, remainder of the attached memory can be use in lookup table/SRAM with specific CPU instructions.

L1D always stores data in L1D SRAM as little endian. The CPU can either be in little or Big Endian. L1D has the logic to rotate for endianness (on element size boundary as specified with CPU transaction) and data alignment. Load return to CPU is always in little endian.

Table 37. L1D Configuration Inputs

Configuration Tieoff	Description
dmc_region_sz[2:0]	Size of L1D attached SRAM used for both L1D Cache and Lookup Tables
crouteid[11:0]	Routeid of L2 / C7x
msmc_id[14:0]	L2/C7x Physical Address bits [43:29] -- Identifies the location of MSMC in the SOC memory map
msmc_region[1:0]	L2/C7x Physical Address bits [28:28] -- Identifies the Corepac address region in the MSMC memory map.
corepac_id[3:0]	L2/C7x Physical Address bits [27:24] -- Identifies the unique corepac / C7x cluster
dmc_baseaddr[3:0]	L2/C7x Physical Address bits [23:20] -- Identifies the 1M Aligned location for L1D SRAM

3.4 L1D Memory

DMC can support up to a maximum of 32KB of cache and up to 48KB total physical memory. Total available physical memory is divided into 2 sections. Section 1 is used to implement cache functionality is 32KB. Section 2 (the remainder of the attached physical memory) is available for look-up table, histogram, and SRAM access mode, and is 16kB. The minimum L1D cache size is 8KB. There is no support for a cache of zero size. These details are described in Table 40.

3.5 L1D Cache

The C7x L1D includes a non-blocking cache controller, which has a main cache and a small fully associative victim cache. Victim cache services read and write hit directly with no performance overhead as the main cache.

The cache controller design supports a range of cache sizes, from 8KB through 32KB. However, a given device may implement less than 32K of L1D RAM. The L1D cache controller 'clamps' requested L1D cache modes to the available L1D memory size.

The L1D cache converts L1D memory to cache, starting at the highest L1D memory address in L1D RAM and working downwards. This computation takes into account the exact size of L1D memory as specified by dmc_sz.

Copyright © 2020, Texas Instruments Incorporated

3.5.1 Non-Blocking Loads (Hit Under Miss)

Unlike the previous generations' L1D cache controller, the C7x L1D cache controller is non-blocking, that is, L1D does not block the operation of the CPU in case of a read miss.

In C6x, the L1D cache controller operated in lock step with the CPU. Pipe stages E1 to E5 were implemented in L1D. In case of a load-miss, L1D stalled the CPU from advancing further because the data for a particular transactions must be returned by the E5 pipe stage in order.

In C6x, the L1D had a build-in optimization of pipe forwarding where, in case of outstanding misses, L1D internally advanced its pipeline and issued additional load-miss to L2 to save on the miss penalty. However, this optimization yielded only marginal benefits, as the CPU stopped issuing further instructions after the initial stall caused by the primary load-miss.

In C7x, the CPU maintains a scoreboard to keep track of outstanding memory load operations. This structure has an additional benefit that it allows out of order return from the memory system.

The memory system can continue servicing additional requests from CPU even in the presence of outstanding misses to a higher level of cache.

The C7x CPU has 2 modes of operation: protected and un-protected mode. As scoreboard functionality is present in both modes, this behavior is transparent to L1D. With each transaction, the CPU sends a transaction ID, which is used to track a particular transaction. L1D returns the same ID along with the load data, when it is available.

To implement non-blocking loads, L1D maintains a Miss Status Handling register (MSHR) buffer to keep track of outstanding loads. This MSHR buffer is used to detect secondary misses and is looked up in parallel with TAG-RAM reads. MSHR is also used to store transaction information required to return the appropriate data back to the CPU in parallel with storing a cache line in the L1D cache.

The first miss to a cache line called a primary miss. Subsequent misses to any of the bytes to the same line or needing the same resources (set conflict) as outstanding transactions are referred to as secondary misses. Secondary misses need in-flight miss resources. The L1D cache behavior in the presence of a secondary miss is defined in Table 38.

Table 38. Secondary Miss Behavior

Primary Miss	Secondary Miss	Conflict-Type	Action
Write	Write	Same set	L1D pipe is not locked
			Misses are send out to L2 or Held in Write-Miss Buffer
Write	Write	Same address	L1D pipe is not locked
			L1D supports aggressive Write Merging to all outstanding writes entires
Write	Read	Same set	L1D pipe is not locked
			Read-Miss is serviced
Write	Read	Same address	L1D pipe is not locked
			Read flushes the same address write-miss entry from the write miss buffer inside L1D and then goes out as read-miss
Read	Read	Same set	L1D pipe is not locked
			Send subsequent reads marked as (NoAllocate) and send out to Higher Level Cache
Read	Read	Same address	L1D Pipe is locked. Secondary Load to the same address will result in an Internal L1D stall
Read	Write	Same set	L1D pipe is not locked
			Write Miss goes out
Read	Write	Same address	L1D pipe is locked till outstanding read-miss is serviced

To ensure that all outstanding transactions have completed, the CPU must issue a memory fence operation. This operation also flushes any outstanding stores which are held in the write miss buffer.

Copyright © 2020, Texas Instruments Incorporated

3.5.2 L1D Cache Controller

L1D has two caches; both are physically indexed and physically tagged.

- Main cache
- Victim cache

3.5.2.1 Main Cache

The main cache is non-blocking cache. The 44-bit physical address provided by the CPU is partitioned into the following fields, as shown in Figure 9.

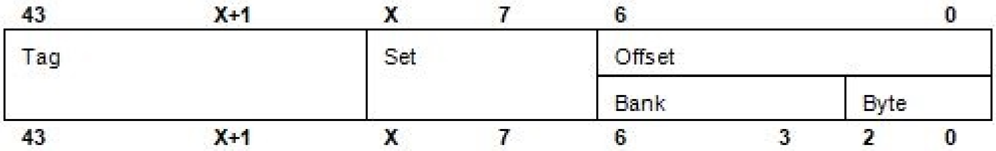


Figure 9. Data Access Address Organization (Main Cache)

The offset of 7 bits accounts for the fact that an L1D line size is 128 bytes. Bits 0 through 6 of the address (the byte and bank fields) are ignored by the cache control logic. (Bits 0 through 6 only determine what bank and what bytes within a bank to access, and thus are irrelevant to the cache's tag compare logic.)

The Set field indicates the L1D main cache line address where the data would reside, were it to be cached. The width of the Set field depends on the amount of L1D configured as cache, as defined in Table 40. The DMC uses the Set field to look up and check the tags for any already-cached data from that address as well as the Valid bit, which indicates whether the address in the tag represents a valid address held in cache.

The Tag field is the upper portion of the address that identifies the true physical location of the data element. The cache compares the tag to the stored tags to determine if the line is present in main cache.

On reads, if one of the tags matches and it’s corresponding valid bit is set, then it is a "hit" and the data cache returns data to the CPU directly from the L1D main cache.

The CPU can also write data through the DMC. When the CPU performs a store, the DMC performs the same tag comparison as it does for reads. If a valid matching tag is found and corresponding MESI state of line is Exclusive or Modified, then the write is a “hit” and the DMC writes data directly into the L1D cache location. Otherwise, the write is a “miss”.

Table 39. Data Access Address Set Field Width

L1DCFG.L1DMODE Setting	Amount of L1D Cache	‘X’ Bit Position	Description
000b	Reserved. Maps to 8K		
001b	Reserved. Maps to 8K		
010b	8K	12	64 L!D sets (64 cache lines)
011b	16K	13	128 L1D sets (128 cache lines
100b	32K	14	256 L1D sets (256 cache lines)
101b	Reserved. Maps to 32K		
110b	Reserved. Maps to 32K		
111b	“Maximal Cache”. Maps to 32K		

In general, a larger value of L1DMODE specifies a larger cache size, up to the size of the implemented L1D memory. The maximum L1D cache size is the smaller of largest power of 2 that fits in L1D RAM size, and 32K. The actual range of L1D cache modes is constrained by the size of L1D Region 0. The L1D cache can be no larger than 16K when L1D Region 0 is only 16K in size. Thus, the L1D maps the encoding 011b through 111b to 16K cache on devices whose L1D Region 1 is only 16K. As a result of this policy, programs wanting no more than a certain amount of cache should program the value corresponding to this upper bound. Programs desiring as much cache as possible should program 111b into L1DMODE. Details of these configurations are described in Table 40.

Copyright © 2020, Texas Instruments Incorporated

Table 40. Configured L1D Cache and SRAM

Total L1D Memory Size	L1DCFG.L1DMOD Setting	Amount of L1D Cache	Amount of L1D SRAM
48K	000b, 001b	8K	40K
	010b	8K	40K
	011b	16K	32K
	100b	32K	16K
	101b, 110b, 111b	32K	16K

3.5.2.2 Victim Cache

In C6x, the L1D data cache is 2-way associative and the Data RAM read was set up in the E3 pipe stage. To support full ECC detection and correction without increasing DSP load latency, it was necessary to move Data RAM up by a pipe stage to E2 in the C7x DMC.

In the E2 stage, the way information of a particular cache line is not known. One option is to read both ways and multiplex the required data when the result of the tag comparison is complete in E3. This option increases the dynamic power significantly, as twice the number of required bits are read.

For the C7x, this required moving to a direct-mapped L1D main cache structure with a victim cache.

The following list provides the basic features of the L1D victim cache:

- 16 cache line fully associative cache
- Total size of 0.5 KBytes for 512-bit configuration
- Tags of victim cache store 34 bits of a 40-bit physical address.
- Victim cache can directly service read and write hits for DSP transactions.
- Victim cache tag comparison is done in parallel with the main cache.
- Zero overhead in the case when a particular transaction is serviced from the victim cache instead of the main cache. Read latency is the same for both caches inside L1D.
- On hits to the victim cache, no swapping of entry takes place between the main and victim cache.
- Victim cache supports atomic operations.
- Full ECC (SECDED) protection on stored cache lines
- Read-Modify-Write pipeline implemented to avoid parity corruption on partial writes.
- Tracking the MESI state of the cache lines to support full hardware coherency

3.5.3 L1D Cache Replacement and Allocation Strategy

3.5.3.1 Main Cache

The L1D main cache is direct mapped in all cache configurations. Thus each location in system memory can reside in only one location of the main cache, as specified by set-bits on a given physical address.

The L1D main cache is a read-allocate-only cache. The L1D cache fetches a complete line on a read miss. The L1D cache does not allocate in the case of a write miss.

A L1D write miss happens when either a line is not present in the L1D main cache, the victim cache, or is present in a shared state. If the line is present in a shared state, a write access to that line invalidates the line in the main cache, and a write-miss is sent to the write-miss handling logic to be serviced. This implementation of C7x does not support the shared state.

L1D is a write-back cache with support for write-through functionality. L1D processes write hits directly within L1D (Write-hit for L1D is defined as when a store happens to a cached line in an exclusive or modified state). It does not pass the update to L2 or the rest of the memory system immediately.

L1D writes back dirty lines when evicting them due to allocation on the same set, or when the cache receives a request to write back a line due to snoops or coherence operations.

Copyright © 2020, Texas Instruments Incorporated

3.5.3.2 Victim Cache

The victim cache is allocated only in the case of a line eviction from the main cache. If a load miss evicts a cache line, that cache line moves to the victim cache.

The victim cache supports MESI to provide full coherency.

The victim cache also services coherence transactions coming from L2 either for coherency or maintenance operations. The victim cache implements a FIFO replacement policy to evict lines to L2. The details of the replacement policy are a micro-architectural detail which are not covered in this document. Table 41 summarizes the operation of the main and victim cache.

Table 41. L1D Main and Victim Cache Behavior

Transaction Type	Main Cache	Victim Cache	Allocation	Swap Entries Between Main and Victim Cache	Comments
read	Hit	X	N/A	No	Hit in Main Cache entry not present in Victim Cache
read	Miss	Hit	No	No	Read Hit serviced from Victim Cache with data returned to CPU in E5 pipe stage.
read	Miss	Miss	Main	N/A	Evicted line is moved to Victim Cache.
write	Hit	X	N/A	N/A	Hit in Main Cache entry not present in Victim Cache
write	Miss	Hit	No	No	Write hit serviced directly from Victim Cache
write	Miss	Miss	No	No	Write Miss serviced by Write-Miss Buffer

3.5.4 L1D MESI Support and Cache Coherence Protocol

C7x supports full hardware coherency. The L1 data cache tracks the full MESI state of lines cached. L1D has the following states for a given cache line.

- Invalid – The cache line is not present in the L1D cache.
- Shared – The cache line is present in the L1D cache in a clean state and can only be read.
- Exclusive – The cache line is present in the L1D cache in a clean state and can be read or written.
- Modified – The cache line is present in the L1D cache and has been modified after being allocated in the exclusive state.

On a load miss, L1D always requests a cache line from L2 in a shared state. L2 has the authority to automatically upgrade the requested line and send it back to L1D with exclusive permissions.

This automatic upgrade policy prevents the degradation due to back-to-back load store transactions to the same address. In the case of a write hit to a line in the shared state, L1D invalidates the line and sends the write miss for servicing by the write miss buffer.

A write hit to a line in the exclusive state changes its state to modified. All lines in the exclusive and shared state are silently dropped by L1D on evictions.

L1D also supports snoops and cache maintenance operations from L2. L1D does a full tag comparison for incoming snoops with both the main cache tags and the victim cache tags to service the request and send an appropriate response back to L2.

NOTE: This implementation of C7x does not support the shared state. On a read miss from L1D, L2 always upgrades the line to exclusive.

3.5.5 L1D Posted-Write Support

L1D is a read allocate cache which supports both write back and write through modes for handling write misses. Unlike prior generations of C6x DSP, L1D does not flush write misses out to L2; instead, L1D supports a posted write feature in C7x, where write-misses are held inside L1D until any of the following conditions are met.

- Outstanding load-miss to the matching address already present in the write miss buffer

Copyright © 2020, Texas Instruments Incorporated

- DSP-initiated fence operation to flush all outstanding store operations
- Eviction from the write miss buffer to allocate a new entry
- Buffer is full and space is needed for a new entry

The posted-write feature is an optimization in L1D to reduce store traffic from L1D to L2 during a stream of write stores.

This optimization is more important for write-through mode, required by an OpenMP application. In write-through mode, all writes, irrespective of whether they hit or miss in L1D, are sent out to L2. In this case, the ability to hold write-misses inside the write-miss buffer allows L1D to do write merging on several store transactions to the same address, and send a single write-miss out when any of the above conditions are met. The current implementation of C7x does not support write-through mode, and this does not apply.

The number of outstanding write misses is limited by the size of the victim cache.

3.5.5.1 Auto-Flush

DMC uses its victim cache for write-miss caching as well. In case of a write miss, instead of sending the write miss out to L2, DMC allocates the write-miss entry into one of the slots in the victim cache. Storing a write-miss entry into the victim cache allows subsequent stores to the same address to be merged, thus reducing the write-miss traffic out to L2. All 16 entries of victim cache can be used to store write-misses, and DMC can merge subsequent writes on any of these 16 entries. When a new line (cache line or write-miss entry) must be allocated to the victim cache, DMC uses a pseudo-FIFO scheme. However, if there is any entry in the victim cache which holds a write-miss entry where a full line is available (all 128 bytes have been written) or a valid half line is available (any of the lower or upper 64 bytes are available with no bytes written in the other half) then the DMC tries to evict that write miss entry first, bypassing the pseudo-FIFO scheme.

The intention here is that because all of the 128 or 64 bytes of a write miss entry have been written, the opportunity of further stores to the same line is reduced. This feature also helps to prevent victim cache thrashing when an entire buffer is being written out from the CPU. With the above scheme, DMC does not evict all 16 lines out of victim cache due to a stream of stores from the CPU. As DMC detects the availability of a full or half write miss line inside victim cache, DMC evicts that write-miss entry first and allocates a new cache line/write-miss in its place, bypassing the FIFO scheme. DMC drains all the write miss entries present in the victim cache when the CPU requests a fence operation.

To optimize the delay from when fence operation is initiated by the CPU and when all stores are committed to memory system, DMC hardware implements an IDLE flush feature. DMC continuously monitors the CPU-DMC interface and its internal pipeline. If DMC’s internal pipelines are empty and the CPU interface doesn’t show any activity for a certain number of cycles, then DMC internally initiates a flush operation and starts to drain the write miss entries stored in the victim cache. In case of any activity from the CPU, this auto-flush operation is cancelled and DMC resumes its normal operation.

3.5.6 L1D Aggressive Write-Merging

The L1D write miss buffer is implemented using the victim cache hardware. On an L1D store miss, an entry is allocated to the victim cache.

The address of store is stored in the victim cache tag RAM. Partial line data are stored in data RAM, and byte enables are tracked for each entry.

An extra bit is needed to identify cached lines from write miss entries in victim cache. Victim cache structure for store writes provides the ability to compare the address of a new incoming store transaction with all outstanding stores in full associative mode.

If a new store hits an outstanding write present in the victim cache, the write transaction is merged on this entry. Data and byte enable information in the victim cache are accordingly updated.

This is an improvement from previous C6x generation L1D behavior. In C6x, write merging was only supported to the last entry in the write miss buffer, even if there were multiple outstanding write misses present in L1D.

With the above approach, L1D can now merge new stores to any outstanding write miss in L1D and reduce the effective write traffic out to L2.

Copyright © 2020, Texas Instruments Incorporated

Aggressive write merging, along with support for posted writes, improves the store data handling capability of L1D in comparison with C6x.

3.5.7 ECC Support

While scaling down in technology, the sensitivity of RAM and even flops to a transient fault is increasing and there is a need to protect against failures. In Corepac2, L1D provides full ECC protection for the data RAM and victim cache, for which L1D implements the SECDED (single-error correction and double error detection) scheme. The L1D TAG RAMs are parity protected, with single-error detection only.

In C7x, parity information is transmitted along with data from one point to another, but each level does not correct and detect the data being transmitted. Instead, the last end point before data is consumed uses the parity information to detect and correct any errors in the received data.

This feature provides protection on all the pipe stages involved in transferring data to L1D.

L2 parity block size is 256 bits, and along with read responses L2 provides parity for each 256-bit block. L1D would do ECC check and correction on the read response data before sending it to the DSP.

Also, while sending out victims to L2, the L1D sends parity information along with victim data to provide ECC protection on the way back.

3.5.8 Read-Modify-Write Support

In C7x, L1D has 16 × 64 bit banks to provide a total data RAM bandwidth of 1024 bits/cycle. The parity block size for L1D is 32 bits. Thus, each physical bank stores two parity blocks. On a CPU store, L1D calculates parity on a 32-bit boundary and updates the parity information in the parity RAM.

Because the CPU can do stores not aligned to a 32-bit boundary, L1D is not able to calculate parity information for such partials writes, as the value of existing data is needed to calculate parity for 32-bit block of data.

In these cases, the memory system usually must invalidate the stored parity information, as it is no longer valid. This results in loss of ECC protection on these data blocks. For a level 1 memory system, the frequency of partial writes is much more frequent, to avoid the loss of protection. The L1D does read-modify-writes on all CPU stores.

Doing a read-modify-write requires double the bandwidth from the memory system to data RAM, as each store must first read what is already written. Additionally, in a pipelined memory system, this leads to data ordering issues.

In C7x, L1D overcomes these issues by having 16 independent banks providing overall bandwidth of 1024 bits per cycle, which is twice the CPU-L1D load/store bandwidth.

Further, L1D detects partial parity block updates in a given CPU store and issues Read transactions for only those banks. This optimization reduces the number of banks that need a full read-modify write cycle.

To resolve data ordering issues due to implementing read-modify-write behavior without increasing the load latency, L1D implements a full associative look-up for any new load transactions with all outstanding stores, and implements a forwarding path in case of a hit.

This parallel comparison enables L1D to implement Read-Modify-Write without any performance penalty. Figure 10 gives a high-level view of the L1D pipeline to support Read-Modify-Write functionality.

Copyright © 2020, Texas Instruments Incorporated

Figure 10. L1D Read-Modify-Write Pipeline

3.5.9 ECC Protection for TAG RAMs

The L1D TAG rams have parity-only protection and can only detect 1-bit errors.

3.5.10 UMC Interaction

L1D uses the VBUSMC protocol to interact with UMC.

3.5.11 Software Pre-fetch

Software prefetch may be implemented by the CPU, but is presented to DMC as a regular load. There is no separate prefetch opcode or transaction type.

3.5.12 Fence Operation

The L1D provides support for the CPU to fence (flush all outstanding) stores to an end point. Outstanding stores include stores in flight, stores pending to L2, and the stores from the Write Miss buffer. L1D tracks the byte count on the write status back from the endpoint to detect completion of the stores. CPU writes can be tagged with a control bit called ‘store tag’ or ‘color’, which is used by the L1D to provide support for flush store priority. This allows the CPU to chose which stores should be flushed first based on the store tag.

To start a fence operation, the CPU asserts a drain signal indicating to the L1D that it should begin flushing the outstanding stores. The L1D starts flushing the outstanding stores, giving priority to the stores tagged with the same value as the priority signal supplied by the CPU. When all of the stores for a specific tag have landed at the end point, the DMC asserts the idle bit for that tag, indicating to the CPU that the stores for that tag are now complete.

The L1D continues to drain the stores until either all of the stores have landed, or the drain signal from the CPU is de-asserted.

3.5.13 Atomic Accesses on Cacheable Write Back Memory Space

In C7x, DMC provides support for the following atomic instructions:

- Compare and swap
- Swap

Atomic accesses are supported to these address spaces:

- L1D SRAM
- External (non - l1dsram), with page attributes set to Cacheable WriteBack

The data sizes are limited to word and double-word. The address must be aligned to the data size.

All atomic operations are ‘load and store’ operations, because the old value of the specified location is returned to the CPU and a new value is written to the specified memory location.

Copyright © 2020, Texas Instruments Incorporated

DMC reports an error in the cases listed below. In all these cases, it zeros out the data and returns a bad read status to the CPU. It also takes an exception and logs the error in the L1DADDR* registers.

- Command Error – Invalid size: Not word or double word
- Command Error – External address space, but with page attributes not supported for atomic accesses
- Addressing Error – L1D SRAM with an address error (out of address range, SRAM under cache, and so forth)

Support for atomic operation takes advantage of the read-modify-write pipeline in both the main and the victim cache.

The ‘C’ code below highlights the basic behavior of a compare and swap operation.

```
int compare_and_swamp (int*reg, int oldval, int newval)
{
  int old_reg_val = *reg
  if (old_reg_val == oldval)
    *reg = newval
  return old_reg_val
```

This ‘C’ function highlights the basic behavior of a compare and swap function. Two sets of values are provided along with a memory location. If the contents of the memory location are the same as the specified value, then the memory location is updated to the new value and the old value is returned to the CPU. If the value is not the same, then the old value is still returned but no update is done to the contents of the specified memory location.

L1D ensures that all of the above operations are executed atomically.

3.5.14 Lookup Table and Histogram Support

The L1D provides support for the lookup table and histogram operations with the following features.

- Maximum of 16 parallel lookups are supported
- Number of parallel tables required can be controlled by the CPU. Supported values are 1, 2, 4, 8, and 16.
- Table element sizes of byte, half-word, and word are supported.
- The L1D provides support for multiple sets of parallel lookup tables, each located at a different base address.
- Look up table data is ECC protected.
- Support for interpolated table lookup to get 2 or 4 successive elements from the lookup table.
- Table entries are supported in packed format, for example, in case of a 8 bit element table, 8 table elements are stored in single L1D bank, which is 64 bits wide.
- Maximum lookup data/ cycle is limited to 512 bits per cycle.
- Non-aligned addressing mode is supported.
- CPU provides 16 maximum possible 16-bit addresses to L1D.
- CPU provides 16 × 16-bit data values or 8 × 32-bit values for weighted histogram increments.

3.5.14.1 Look-up Tables

Table 42 shows all the possible combination of supported table types with the number of consecutive elements returned for interpolation. Table 43 shows how the data elements for a look-up-table are placed in the L1D memory banks.

Copyright © 2020, Texas Instruments Incorporated

Table 42. Items and Table Types

<table>
<tr><th rowspan="2">Table Type</th><th rowspan="2" colspan="2">Number of Items Returned per Table Lookup</th><th colspan="5">Number of Parallel Tables[1]</th></tr>
<tr><th>1</th><th>2</th><th>4</th><th>8</th><th>16</th></tr>
<tr><td rowspan="4">Word</td><td colspan="2">1</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td colspan="2">2</td><td>X</td><td>X</td><td>X</td><td>X</td><td>NS</td></tr>
<tr><td colspan="2">4</td><td>X</td><td>X</td><td>X</td><td>NS</td><td>NS</td></tr>
<tr><td colspan="2">8</td><td>X</td><td>X</td><td>NS</td><td>NS</td><td>NS</td></tr>
<tr><td rowspan="4">Half-Word</td><td colspan="2">1</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td colspan="2">2</td><td>X</td><td>X</td><td>X</td><td>X</td><td>NS</td></tr>
<tr><td colspan="2">4</td><td>X</td><td>X</td><td>X</td><td>Possible</td><td>NS</td></tr>
<tr><td colspan="2">8</td><td>X</td><td>X</td><td>Possible</td><td>NS</td><td>NS</td></tr>
<tr><td rowspan="4">Byte</td><td colspan="2">1</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td colspan="2">2</td><td>X</td><td>X</td><td>X</td><td>X</td><td>NS</td></tr>
<tr><td colspan="2">4</td><td>X</td><td>X</td><td>X</td><td>Possible</td><td>NS</td></tr>
<tr><td colspan="2">8</td><td>X</td><td>X</td><td>Possible</td><td>Possible</td><td>NS</td></tr>
<tr><td>Color Legend for Max No. of Bits per Lookup Returned to CPU</td><td>512</td><td>256</td><td>128</td><td>64</td><td>32</td><td>16</td><td>8</td></tr>
</table>

[1] NS: Not Supported

NOTE: Returning 8 elements per look-up table might not be a use case, and is likely to be removed from L1D functionality.

Table 43. Look-up Table Data Layout in L1D Memory (512-Bit Mode)

<table>
<tr><td colspan="16">16 Parallel Tables - Element Size "Byte"</td></tr>
<tr><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td></tr>
<tr><td>P[F] .. P[8]</td><td>O[F] .. O[8]</td><td>N[F] .. N[8]</td><td>M[F] .. M[8]</td><td>L[F] .. L[8]</td><td>K[F] .. K[8]</td><td>J[F] .. J[8]</td><td>I[F] .. I[8]</td><td>H[F] .. H[8]</td><td>G[F] .. G[8]</td><td>F[F] .. F[8]</td><td>E[F] .. E[8]</td><td>D[F] .. D[8]</td><td>C[F] .. C[8]</td><td>B[F] .. B[8]</td><td>A[F] .. A[8]</td></tr>
<tr><td>P[7] … P[0]</td><td>O[7] … O[0]</td><td>N[7] … N[0]</td><td>M[7] … M[0</td><td>L[7] … L[0]</td><td>K[7] … K[0]</td><td>J[7] … J[0]</td><td>I[7] … I[0]</td><td>H[7] … H[0]</td><td>G[7] … G[0]</td><td>F[7] … F[0]</td><td>E[7] … E[0]</td><td>D[7] … D[0]</td><td>C[7] … C[0]</td><td>B[7] … B[0]</td><td>A[7] … A[0]</td></tr>
<tr><td colspan="16">8 Parallel Tables - Element Size "Byte"</td></tr>
<tr><td colspan="2">… … …</td><td colspan="2">… … …</td><td colspan="2">… … …</td><td colspan="2">… … …</td><td colspan="2">… … …</td><td colspan="2">… … …</td><td colspan="2">… … …</td><td colspan="2">… … …</td></tr>
<tr><td colspan="2">H[1F] .. H[10]</td><td colspan="2">G[1F] .. G[10]</td><td colspan="2">F[1F] .. F[10]</td><td colspan="2">E[1F] .. E[10]</td><td colspan="2">D[1F] .. D[10]</td><td colspan="2">C[1F] .. C[10]</td><td colspan="2">B[1F] .. B[10]</td><td colspan="2">A[1F] .. A[10]</td></tr>
<tr><td colspan="2">H[F] … H[0]</td><td colspan="2">G[F] … G[0]</td><td colspan="2">F[F] … F[0]</td><td colspan="2">E[F] … E[0]</td><td colspan="2">D[F] … D[0]</td><td colspan="2">C[F] … C[0]</td><td colspan="2">B[F] … B[0]</td><td colspan="2">A[F] … A[0]</td></tr>
<tr><td colspan="16">4 Parallel Tables - Element Size "Byte"</td></tr>
<tr><td colspan="4">… … …</td><td colspan="4">… … …</td><td colspan="4">… … …</td><td colspan="4">… … …</td></tr>
<tr><td colspan="4">D[3F] .. D[20]</td><td colspan="4">C[3F] .. C[20]</td><td colspan="4">B[3F] .. B[20]</td><td colspan="4">A[3F] .. A[20]</td></tr>
<tr><td colspan="4">D[1F] … D[0]</td><td colspan="4">C[1F] … C[0]</td><td colspan="4">B[1F] … B[0]</td><td colspan="4">A[1F] … A[0]</td></tr>
<tr><td colspan="16">2 Parallel Tables - Element Size "Byte"</td></tr>
<tr><td colspan="8">… … …</td><td colspan="8">… … …</td></tr>
<tr><td colspan="8">B[7F] … B[40]</td><td colspan="8">A[7F] … A[40]</td></tr>
<tr><td colspan="8">B[3F] … B[0]</td><td colspan="8">A[3F] … A[0]</td></tr>
<tr><td colspan="16">1 Table - Element Size "Byte"</td></tr>
<tr><td colspan="16">… … …</td></tr>
<tr><td colspan="16">A[FF] … A[80]</td></tr>
<tr><td colspan="16">A[7F] … A[0]</td></tr>
<tr><td colspan="16">16 Parallel Tables - Element Size "Half Word"</td></tr>
<tr><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td><td>… … …</td></tr>
<tr><td>P[7] .. P[4]</td><td>O[7] .. O[4]</td><td>N[7] .. N[4]</td><td>M[7] .. M[4]</td><td>L[7] .. L[4]</td><td>K[7] .. K[4]</td><td>J[7] .. J[4]</td><td>I[7] .. I[4]</td><td>H[7] .. H[4]</td><td>G[7] .. G[4]</td><td>F[7] .. F[4]</td><td>E[7] .. E[4]</td><td>D[7] .. D[4]</td><td>C[7] .. C[4]</td><td>B[7] .. B[4]</td><td>A[7] .. A[4]</td></tr>
<tr><td>P[3] … P[0]</td><td>O[3] … O[0]</td><td>N[3] … N[0]</td><td>M[3] … M[0</td><td>L[3] … L[0]</td><td>K[3] … K[0]</td><td>J[3] … J[0]</td><td>I[3] … I[0]</td><td>H[3] … H[0]</td><td>G[3] … G[0]</td><td>F[3] … F[0]</td><td>E[3] … E[0]</td><td>D[3] … D[0]</td><td>C[3] … C[0]</td><td>B[3] … B[0]</td><td>A[3] … A[0]</td></tr>
</table>

Copyright © 2020, Texas Instruments Incorporated

Table 43. Look-up Table Data Layout in L1D Memory (512-Bit Mode) (continued)

8 Parallel Tables - Element Size "Half Word"							
… … …	… … …	… … …	… … …	… … …	… … …	… … …	… … …
H[F] .. H[8]	G[F] .. G[8]	F[F] .. F[8]	E[F] .. E[8]	D[F] .. D[8]	C[F] .. C[8]	B[F] .. B[8]	A[F] .. A[8]
H[7] … H[0]	G[7] … G[0]	F[7] … F[0]	E[7] … E[0]	D[7] … D[0]	C[7] … C[0]	B[7] … B[0]	A[7] … A[0]
4 Parallel Tables - Element Size "Half Word"							
… … …		… … …		… … …		… … …	
D[1F] .. D[10]		C[1F] .. C[10]		B[1F] .. B[10]		A[1F] .. A[10]	
D[F] … D[0]		C[F] … C[0]		B[F] … B[0]		A[F] … A[0]	
2 Parallel Tables - Element Size "Half Word"							
… … …				… … …			
B[3F] … B[20]				A[3F] … A[20]			
B[1F] … B[0]				A[1F] … A[0]			
1 Table - Element Size "Half Word"							
… … …							
A[7F] … A[40]							
A[3F] … A[0]							

Table 44 specifies the dimensions of a look-up table supported by C7x with different configured sizes of look-up table physical memory.

Copyright © 2020, Texas Instruments Incorporated

Table 44. Look-up Table Size and Configuration Supported (512-Bit Mode)

Size of LUT Region (Kbytes)	No of Parallel Tables	Element Size	Size of Each Table (Kilobytes)	Max Number of Elements Per Table		Size of LUT Region (Kbytes)	No of Parallel Tables	Element Size	Size of Each Table (Kilobytes)	Max Number of Elements Per Table
8	16	Byte	0.50	512		24	16	Byte	1.50	1536
8	16	Half Word	0.50	256		24	16	Half Word	1.50	768
8	16	Word	0.50	128		24	16	Word	1.50	384
8	8	Byte	1.00	1024		24	8	Byte	3.00	3072
8	8	Half Word	1.00	512		24	8	Half Word	3.00	1536
8	8	Word	1.00	256		24	8	Word	3.00	768
8	4	Byte	2.00	2048		24	4	Byte	6.00	6144
8	4	Half Word	2.00	1024		24	4	Half Word	6.00	3072
8	4	Word	2.00	512		24	4	Word	6.00	1536
8	2	Byte	4.00	4096		24	2	Byte	12.00	12288
8	2	Half Word	4.00	2048		24	2	Half Word	12.00	6144
8	2	Word	4.00	1024		24	2	Word	12.00	3072
8	1	Byte	8.00	8192		24	1	Byte	24.00	24576
8	1	Half Word	8.00	4096		24	1	Half Word	24.00	12288
8	1	Word	8.00	2048		24	1	Word	24.00	6144
16	16	Byte	1.00	1024		32	16	Byte	2.00	2048
16	16	Half Word	1.00	512		32	16	Half Word	2.00	1024
16	16	Word	1.00	256		32	16	Word	2.00	512
16	8	Byte	2.00	2048		32	8	Byte	4.00	4096
16	8	Half Word	2.00	1024		32	8	Half Word	4.00	2048
16	8	Word	2.00	512		32	8	Word	4.00	1024
16	4	Byte	4.00	4096		32	4	Byte	8.00	8192
16	4	Half Word	4.00	2048		32	4	Half Word	8.00	4096
16	4	Word	4.00	1024		32	4	Word	8.00	2048
16	2	Byte	8.00	8192		32	2	Byte	16.00	16384
16	2	Half Word	8.00	4096		32	2	Half Word	16.00	8192
16	2	Word	8.00	2048		32	2	Word	16.00	4096
16	1	Byte	16.00	16384		32	1	Byte	32.00	32768
16	1	Half Word	16.00	8192		32	1	Half Word	32.00	16384
16	1	Word	16.00	4096		32	1	Word	32.00	8192

Copyright © 2020, Texas Instruments Incorporated

Table 45. Data Organization for Byte Look-up Table and CPU Return Muxing (512-Bit Mode)

S.No	Column	Lookup Table Organization																															
1	32 bits	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
2	Bank Number	F		E		D		C		B		A		9		8		7		6		5		4		3		2		1		0	
3	Table Structure	F		E		D		C		B		A		9		8		7		6		5		4		3		2		1		0	
4	Words	P1	P0	O1	O0	N1	N0	M1	M0	L1	L0	K1	K0	J1	J0	I1	I0	H1	H0	G1	G0	F1	F0	E1	E0	D1	D0	C1	C0	B1	B0	A1	A0
5	512 CPU Return	Element Size = 1 Byte																15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
6		No of Tables = 16																P1	O1	N1	M1	L1	K1	J1	I1	H1	G1	F1	E1	D1	C1	B1	A1
7	Return Data Highlighting Muxing	Total No of Lookup bits Returned per Lookup = 128																P0	O0	N0	M0	L0	K0	J0	I0	H0	G0	F0	E0	D0	C0	B0	A0
S.No	Column	Lookup Table Organization																															
1	32 bits	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
2	Bank Number	F		E		D		C		B		A		9		8		7		6		5		4		3		2		1		0	
3	Table Structure	7				6				5				4				3				2				1				0			
4	Words	P1	P0	O1	O0	N1	N0	M1	M0	L1	L0	K1	K0	J1	J0	I1	I0	H1	H0	G1	G0	F1	F0	E1	E0	D1	D0	C1	C0	B1	B0	A1	A0

Copyright © 2020, Texas Instruments Incorporated

Table 45. Data Organization for Byte Look-up Table and CPU Return Muxing (512-Bit Mode) (continued)

S.No	Column	Lookup Table Organization																															
5	512 CPU Return	Element Size = 1 Byte																7		6		5		4		3		2		1		0	
6		No of Tables = 8																O1	OO	M1	M0	K1	K0	I1	I0	G1	G0	E1	E0	C1	C0	A1	A0
7	Return Data Highlighing Muxing	Total No of Lookup bits Returned per Lookup = 64																P1	P0	N1	N0	L1	L0	J1	J0	H1	H0	F1	F0	D1	D0	B1	B0
S.No	Column	Lookup Table Organization																															
1	32 bits	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
2	Bank Number	F		E		D		C		B		A		9		8		7		6		5		4		3		2		1		0	
3	Table Structure	4								3								2								1							
4	Words	P1	P0	O1	O0	N1	N0	M1	M0	L1	L0	K1	K0	J1	J0	I1	I0	H1	H0	G1	G0	F1	F0	E1	E0	D1	D0	C1	C0	B1	B0	A1	A0
5	512 CPU Return	Element Size = 1 Byte																3				2				1				0			
6		No of Tables = 4																N1	N0	M1	M0	J1	J0	I1	I0	F1	F0	E1	E0	B1	B0	A1	A0
7	Return Data Highlighing Muxing	Total No of Lookup bits Returned per Lookup = 32																P1	P0	O1	O0	L1	L0	K1	K0	H1	H0	G1	G0	D1	D0	C1	C0
S.No	Column	Lookup Table Organization																															

Copyright © 2020, Texas Instruments Incorporated

Table 45. Data Organization for Byte Look-up Table and CPU Return Muxing (512-Bit Mode) (continued)

S.No	Column	Lookup Table Organization																															
1	32 bits	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
2	Bank Number	F		E		D		C		B		A		9		8		7		6		5		4		3		2		1		0	
3	Table Structure	1																0															
4	Words	P1	P0	O1	O0	N1	N0	M1	M0	L1	L0	K1	K0	J1	J0	I1	I0	H1	H0	G1	G0	F1	F0	E1	E0	D1	D0	C1	C0	B1	B0	A1	A0
5	512 CPU Return	Element Size = 1 Byte																1								0							
6		No of Tables = 2																L1	L0	K1	K0	J1	J0	I1	I0	D1	D0	C1	C0	B1	B0	A1	A0
7	Return Data Highlighting Muxing	Total No of Lookup bits Returned per Lookup = 16																P1	P0	O1	O0	N1	N0	M1	M0	H1	H0	G1	G0	F1	F0	E1	E0
S.No	Column	Lookup Table Organization																															
1	32 bits	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
2	Bank Number	F		E		D		C		B		A		9		8		7		6		5		4		3		2		1		0	
3	Table Structure	0																															
4	Words	P1	P0	O1	O0	N1	N0	M1	M0	L1	L0	K1	K0	J1	J0	I1	I0	H1	H0	G1	G0	F1	F0	E1	E0	D1	D0	C1	C0	B1	B0	A1	A0

Copyright © 2020, Texas Instruments Incorporated

Table 45. Data Organization for Byte Look-up Table and CPU Return Muxing (512-Bit Mode) (continued)

S.No	Column	Lookup Table Organization																
5	512 CPU Return	Element Size = 1 Byte	0															
6		No of Tables = 1	H1	H0	G1	G0	F1	F0	E1	E0	D1	D0	C1	C0	B1	B0	A1	A0
7	Return Data Highlighing Muxing	Total No of Lookup bits Returned per Lookup = 8	P1	P0	O1	O0	N1	N0	M1	M0	L1	L0	K1	K0	J1	J0	I1	I0

Copyright © 2020, Texas Instruments Incorporated

3.5.14.2 Histogram Support

The C7x DMC supports byte, half-word, and word histograms. The number of bins supported are limited by the amount of RAM allocated to the lookup table/histogram partition.

DMC also provides support for multi-threaded histograms to speed up histogram calculations. The multithreading allows speed up of histogram calculation by providing more memory for histogram calculations.

Support is provided for weighted histograms, where different increment values for each histogram bin can be specified. Table 46 provides details on the attributes of each supported histogram configuration.

Table 46. Supported Histogram Configuration and Sizes (512-Bit Mode)

LUT/Histogram Partition Size (KBytes)	No. of Histogram Threads	Histogram Bin Size	Histogram Thread Size (Kilobytes)	Max Number of Bins Per Histogram Thread
8	16	Byte	0.50	512
8	16	Half-word	0.50	256
8	16	Word	0.50	128
8	8	Byte	1.00	1024
8	8	Half-word	1.00	512
8	8	Word	1.00	256
8	4	Byte	2.00	2048
8	4	Half-word	2.00	1024
8	4	Word	2.00	512
8	2	Byte	4.00	4096
8	2	Half-word	4.00	2048
8	2	Word	4.00	1024
8	1	Byte	8.00	8192
8	1	Half-word	8.00	4096
8	1	Word	8.00	2048
16	16	Byte	1.00	1024
16	16	Half-word	1.00	512
16	16	Word	1.00	256
16	8	Byte	2.00	2048
16	8	Half-word	2.00	1024
16	8	Word	2.00	512
16	4	Byte	4.00	4096
16	4	Half-word	4.00	2048
16	4	Word	4.00	1024
16	2	Byte	8.00	8192
16	2	Half-word	8.00	4096
16	2	Word	8.00	2048
16	1	Byte	16.00	16384
16	1	Half-word	16.00	8192
16	1	Word	16.00	4096
24	16	Byte	1.50	1536
24	16	Half-word	1.50	768
24	16	Word	1.50	384
24	8	Byte	3.00	3072
24	8	Half-word	3.00	1536
24	8	Word	3.00	768
24	4	Byte	6.00	6144

Copyright © 2020, Texas Instruments Incorporated

Table 46. Supported Histogram Configuration and Sizes (512-Bit Mode) (continued)

LUT/Histogram Partition Size (KBytes)	No. of Histogram Threads	Histogram Bin Size	Histogram Thread Size (Kilobytes)	Max Number of Bins Per Histogram Thread
24	4	Half-word	6.00	3072
24	4	Word	6.00	1536
24	2	Byte	12.00	12288
24	2	Half-word	12.00	6144
24	2	Word	12.00	3072
24	1	Byte	24.00	24576
24	1	Half-word	24.00	12288
24	1	Word	24.00	6144
32	16	Byte	2.00	2048
32	16	Half-word	2.00	1024
32	16	Word	2.00	512
32	8	Byte	4.00	4096
32	8	Half-word	4.00	2048
32	8	Word	4.00	1024
32	4	Byte	8.00	8192
32	4	Half-word	8.00	4096
32	4	Word	8.00	2048
32	2	Byte	16.00	16384
32	2	Half-word	16.00	8192
32	2	Word	16.00	4096
32	1	Byte	32.00	32768
32	1	Half-word	32.00	16384
32	1	Word	32.00	8192

3.5.14.3 Histogram Organization in L1D Banks

Table 47 provides details on how histograms are stored in L1D banks for each of the multi-threaded configurations. For example, in the case of a 4-way multi-threaded histogram, bank 0 to bank3 store Thread0, bank 4 to bank 7 store Thread 1, bank 8 to bank B store Thread 2, and bank C to bank F store Thread 3.

Table 47. Histogram Organization in L1D Physical Memory for (512-Bit Mode)

<table>
<tr><th>Lane Size (Bits)</th><th>64</th><th>64</th><th>64</th><th>64</th><th>64</th><th>64</th><th>64</th><th>64</th><th>64</th><th>64</th><th>64</th><th>64</th><th>64</th><th>64</th><th>64</th><th>64</th></tr>
<tr><th>L1D Banks</th><th>F</th><th>E</th><th>D</th><th>C</th><th>B</th><th>A</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>16 Thread</td><td>ThF</td><td>ThE</td><td>ThD</td><td>ThC</td><td>ThB</td><td>ThA</td><td>Th9</td><td>Th8</td><td>Th7</td><td>Th6</td><td>Th5</td><td>Th4</td><td>Th3</td><td>Th2</td><td>Th1</td><td>Th0</td></tr>
<tr><td>8 Thread</td><td colspan="2">Th7</td><td colspan="2">Th6</td><td colspan="2">Th5</td><td colspan="2">Th4</td><td colspan="2">Th3</td><td colspan="2">Th2</td><td colspan="2">Th1</td><td colspan="2">Th0</td></tr>
<tr><td>4 Thread</td><td colspan="4">Th3</td><td colspan="4">Th2</td><td colspan="4">Th1</td><td colspan="4">Th0</td></tr>
<tr><td>2 Thread</td><td colspan="8">Th1</td><td colspan="8">Th0</td></tr>
<tr><td>1 Thread</td><td colspan="16">Th0</td></tr>
</table>

Copyright © 2020, Texas Instruments Incorporated

NOTE: At the end of a histogram loop, the user can use look-up table access mode to add all of the partial histograms collected in multi-threaded histogram mode to get the final histogram.

3.5.14.4 Permission Checks and Protection

In case of a Read access exceeding the look-up table physical address range, L1D zeros out the read data and returns an Address Error to the CPU. In case of a store access exceeding the look-up table physical address range, L1D issues an exception along with capturing failure information in MPFxR registers. In case of a command error, L1D issues an exception along with capturing failure information in MPFxR registers.

3.5.15 Vector Byte Lane Predicate Support

C7x L1D does not treat aligned and non-aligned access as separate transactions. All the DSP instructions are treated to be non-aligned. DSP no longer masks the lower bits of an address to align it to the transaction size.

The CPU provides write byte enables (byte lane predicate) to indicate if a particular byte must be written or not. The CPU to L1D write data bus is 512/256 bits wide (64/32 bytes); the CPU can choose to write any combination of 64/32 bytes by using write predicates along with a store transaction.

L1D always stores data in L1D SRAM as little endian; in case of Big Endian mode, L1D does a rotation store data on the element-type boundary, as specified by the store transaction.

3.6 DMC Registers

The C7x memory system provides a set of control registers to govern the operation of the L1D cache and provide error reporting capability. These registers allow programs to change cache modes and manually initiate cache coherence operations.

In C7x, these registers are directly accessible by the CPU and are not memory mapped, but they still reside physically inside L1D. Table 49 describes these registers.

3.6.1 L1D Cache Configuration Registers

The L1D allows selecting the size of the L1D cache at run time. Programs select the size of the L1D cache by writing the requested mode to the L1DMODE field in the L1DCFG memory mapped Control registers.

3.6.2 Global Cache Coherence Operation

Global cache operations exist for the purpose of synchronizing L1D with the system between major events, such as a task switch, L1D mode change, or change to memory protection settings. Thus, programs view global cache operations as synchronous.

Upon receipt of a global cache command, the DMC may stall the CPU while it processes the request, as needed. The DMC must take care that all 'in flight' activity (such as victim writebacks or L1D write buffer draining) does not negatively interfere with the global cache operation or otherwise cause data loss.

Details of the Global Coherence configuration registers are in Table 48 and in Table 50.

Table 48. Global Coherence Cache Operation

Cache Operation	Register Usage	Address	Effect on L1D Cache
L1D Global Writeback	L1WB	0x202	Writes back all updated lines in L1D to external. Leaves all of L1D clean.
L1D Global Writeback with Invalidate	L1WBINV	0x203	Writes back all updated lines in L1D . Leaves all of L1D invalid.
L1D Invalidate	L1INV	0x204	Invalidates the entire contents of L1D. Discards all updated lines.

Copyright © 2020, Texas Instruments Incorporated

3.6.3 Exception Status Registers

To allow programs to diagnose a fault after an exception occurs, the DMC implements error logging registers, which are described in Table 50.

Two types of errors are recorded:

- Addressing errors captured in L1DADDREE registers: address outside allowable SRAM space, SRAM under cache, LUT base address outside implemented SRAM, and all atomic errors.
- LUT command errors captured in L1DCMDEE registers: when a LUT store transaction has a command error, the DMC captures the transaction information (error type, error flag) in the command error registers and generates an exception. The address is not captured because it is not relevant in the context of LUT. For LUT load transactions that have a command error, the DMC zeros the data and returns an address error in the rstatus.

The error logging registers only store enough information for one fault. Generally, the hardware records the information about the first fault and generates an exception only for that fault.

DMC holds the fault information until software clears it by writing to 1 to the clean register. DMC does nothing if software writes 0 to the clear register. The DMC ignores the value written to bits 31:1 of these clear registers.

3.6.4 Error Status Reporting

DMC reports errors to the CPU and UMC through rstatus. Because multiple errors may be detected, DMC implements a priority scheme when reporting the rstatus. DMC does not generate all error encodings, but passes along any error code that it received from the L2 controller. The following is the priority list:

- Read exclusive error
- Timeout error
- Data error
- Protection error
- Address error
- Unsupported address mode

3.6.5 ECC Error and Status Register

DMC aligns to the Keystone 3 safety architecture for reporting of detected parity and ECC errors.

3.7 Interfaces

The DMC has following interfaces:

- L1D memory interface, for reading / writing memory
- L2 interface for servicing cache line fills, writebacks, partial line writes, non-cacheable data accesses, and for snoop commands and DMA reads and writes to L1D SRAM
- CPU data interface, for servicing load, store requests and table lookups
- Configuration interface, for accessing registers implemented inside L1D
- Hardware watch point (HWWP) interface

3.7.1 DMC to L1D RAM Interface

The interface from the DMC to L1D RAM facilitates both reads from and writes to the L1D memory. CPU load store accesses, CPU lookup table accesses, snoops from L2, line fills from L2, and write backs all use this interface to get access to L1D RAM.

3.7.2 L1D RAM Interface

DMC supports a variety of memory accesses internally from different sources:

- CPU datapath 0/1: 64 + 512-bit read (reads due to loads and reads due to partial stores for read-modify-write operation)

Copyright © 2020, Texas Instruments Incorporated

- CPU datapath 0/1: 64 + 512-bit write
- CPU lookup table/histogram: 512-bit read. 512-bit write divided for up to a maximum of 16 different addresses
- Internal cache operation: 1024-bit read (victim reads)
- Snoop read operation: 1024-bit read
- L2 512-bit line fills

To support the concurrency requirements between write stream and CPU read stream [max 512 bits] and to support 1K bit internal cache operations, the DMC has a 1024-bit read path and a 1024-bit write path to the L1D memory. These paths provide address, byte stores, and read-no-write signals for each of the 64-bit banks.

To support Read-Modify-Write operations, CPU writes are not committed inside L1D at the same pipe stage that reads are issued. Instead, all partial writes cause a read to the L1D data RAM and after correcting the read data, store data is merged on it and committed back to the L1D RAM. In C7x L1D, the pipeline for read and writes are decoupled. In this Specification, read stream refers to transactions to L1D RAM through DMC's Read pipeline, and Write stream refers to transactions to L1D RAM through DMC's store pipeline. Both Read and Write streams still arbitrate in E2 to get bank access.

The DMC takes responsibility for sorting through the separate requestors that desire access to the L1D memory, generating necessary addresses, and enabling and setting up the necessary routing. The L1D RAM interface must, at the minimum, support the following.

- CPU load and store must be able to specify the exact set of bytes accessed. This feature is also required to support write lane predicates.
- Concurrent access (Read or Write) from both CPU read and CPU write posted streams when there are no bank conflicts. There might be some additional restrictions on this muxing.
- 1024-bit / 512-bit aligned single-cycle read for internal cache operations. Internal cache operation are never concurrent with any other operation.
- Concurrent access of CPU Write stream from store unit and 512 / 256-bit L2 line fills.
- In addition to Data Read and Write, L1D interface must also support writing and reading of parity information stored in L1D banks. Parity block size for L1D is 32 bits. 32-bit parity block size requires 7 bits of parity information to implement SECDED. 6 bits of parity + 1 bit of All parity for double error detect. Because the L1D bank size is 64 bits, each bank contains 2 parity blocks. Total width of each bank including parity information is 78 bits. Also, the L1D SRAM bank must support wrenz (bit write enable control) to control writing parity of these two blocks individually.

If there are no bank conflicts between CPU Read and CPU Write accesses, DMC attempts to provide access to L1D memory with 100% efficiency. Implicit in this requirement is that L1D doesn't support wait states. DMC may insert at most a one cycle stall due to a bank conflict between read and write streams. Each bank in L1D memory must allow for arbitrary byte enables and bit write enables.

Each must support at least 1 read or 1 write per cycle. Neither L1D nor memory attached to it must support simultaneous read and write to a single bank.

3.7.3 Error Detection

L1D internally implements SECDED scheme to protect its data and RAMs, and parity only for TAG RAMs. DMC aligns to the Keystone 3 safety architecture for reporting detected parity and ECC errors.

3.7.4 DMC to UMC Interface

The L1D to L2 interface follows the VBUSMC protocol. This interface to the L2 controller consists of a 512-bit read data bus and a 512-bit write data bus. The specific set of addition signals are provided as defined in the VBUSMC protocol.

3.7.5 DMC to CPU Interface

The DMC processes up to 2 Read or Write data accesses from the CPU every cycle. This section discusses how the DMC acts upon these requests.

Copyright © 2020, Texas Instruments Incorporated

3.7.5.1 Data Format

- Load store format
- Lookup table and histogram format

3.7.5.2 Non-Aligned Access

All load/store and lookup table access from the CPU are considered non-aligned by L1D. L1D supports non-aligned access of any size up to a maximum of 512 bits.

3.7.5.3 Ordering Between the Two Datapaths

Ordering between 2 datapaths (DP0 and DP1) is the same as in C6x. L1D always gives priority to loads over stores transactions, and DP0 gives priority over DP1. Support for crosspath and the impact of crosspath on priority is not yet defined.

3.7.5.4 Pipeline Advancing

Similar to the C6x, in C7x L1D either accepts both DP0 or DP1 transactions or stalls both transactions in the E1 pipestage. Both pipes advance together inside L1D, and there might be some serialization done inside the controller to resolve conflicts such as bank stalls.

3.7.5.5 Banking

The C7x, L1D provides a 64-bit by 16-bank structure that is similar to the banking structure provided by C6x, but the total bandwidth of L1D RAM has increased 5x from 128 bits per cycle to (64 + 512) bits per cycle.

The C7x L1D’s banking mechanism has the following features:

- L1D memory is divided into 16 × 64-bit wide banks.
- The banks are interleaved based on the low-order bits of the address.
- Two CPU accesses, a read from CPU, and write from store – unit can cause a bank conflict.

3.7.6 DMC ECR Interface

DMC supports an ERI-esque interface to allow access to all implemented registers directly by the CPU. These registers are not memory mapped, and instead are mapped to a MOVC CPU instruction.

3.7.7 Hardware Watch Point (HWWP)

The hardware watch point functionality, as described in the Thinman Specification Document, is implemented across the CPU and DMC. The interface is described in the CPU-DMC Protocol Spec.

Copyright © 2020, Texas Instruments Incorporated

3.8 Memory-Mapped ECR Registers

Table 49. DMC Memory-Mapped ECR Registers Summary

REGID	Register	Description	PERMISSIONS															
			GU	GS	RU	RS	SRU	SRS	SDBG	NSDBG	GU	GS	RU	RS	SRU	SRS	SDBG	NSDBG
			Write								Read							
0x200	L1DCFG	L1D Cache Mode Configuration	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1
0x202	L1DWB	L1D Global Writeback	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x203	L1DWBINV	L1D Global Writeback with Invalidate	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x204	L1DINV	L1D Global Invalidate	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1
0x206	L1DCTAG	L1D Cache tag View	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0x211	L1DADDREEA	L1D Addressing Error Event Address	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x212	L1DADDREES	L1D Addressing Error Event Status	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x213	L1DADDREER	L1D Addressing Error Event Reset	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1
0x214	L1DCMDEEA	L1D CMD Error Event Address	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x215	L1DCMDEES	L1D CMD Error Event Status	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x216	L1DCMDEER	L1D CMD Error Event Reset	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1

Copyright © 2020, Texas Instruments Incorporated

Table 50. DMC Memory-Mapped ECR Registers Details

REGID	REGISTER	Register Fields	MSB	LSB	Field Access	Field Reset	Register Description
0x200	L1DCFG	RSVD	63	3	R	0	
		L1DMODE	2	0	RW	7	L1D Cache size
0x202	L1DWB	RSVD	63	1	R	0	
		WB	0	0	RW	0	Set to start the writeback operation. Will get cleared when the writeback is complete
0x203	L1DWBINV	RSVD	63	1	R	0	
		WBINV	0	0	RW	0	Set to start the writeback operation. Will get cleared when the writeback is complete
0x204	L1DINV	RSVD	63	1	R	0	
		INV	0	0	RW	0	Set to start the writeback operation. Will get cleared when the writeback is complete
0x206	L1DCTAG	RSVD	63	52	R	0	
		MESI	51	50	R	0	MESI Bits includes valid and dirty bits
		SECURE	49	49	R	0	
		TAG	48	7	R	0	L1D Tag. Bits [48:44] will return 0
		RSVD	6	0	R	0	
0x211	L1DADDREEA	RSVD	63	44	R	0	
		EADDR	43	0	R	0	Transaction address for which error was detected
0x212	L1DADDREES	RSVD	63	10	R	0	
		RQSTR	8	8	R	0	Requestor of the transaction for which error was detected [CPU,DMA]
		RSVD	7	6	R	0	
		ERROR_TYPE	5	4	R	0	Type of Error - Address outside allowable SRAM space, SRAM under Cache, LUT base address outside implemented SRAM, Atomic errors
		RSVD	3	1	R	0	Bank in which error was detected [0 - F]
		ERROR_FLAG	0	0	R	0	Indicates that an error has occurred and has not been cleared yet. Clear occurs on a write to L1DADDREER
0x213	L1DADDREER	RSVD	63	1	R	0	
		ERROR_RESET	0	0	W		
0x214	L1DCMDEEA	RSVD	63	44	R	0	
		EADDR	43	0	R	0	Transaction address for which error was detected

Copyright © 2020, Texas Instruments Incorporated

Table 50. DMC Memory-Mapped ECR Registers Details (continued)

REGID	REGISTER	Register Fields	MSB	LSB	Field Access	Field Reset	Register Description
0x215	L1DCMDEES	RSVD	63	6	R	0	
		ERROR_TYPE	5	4	R	0	CAS Operation failed, LUT Operation Failed (invalid size or cfg)
		RSVD	3	1	R	0	
		ERROR_FLAG	0	0	R	0	Indicates that an error has occurred and has not been cleared yet. Clear occurs on a write to L1DCOREDER
0x216	L1DCMDEER	RSVD	63	1	R	0	
		ERROR_RESET	0	0	W		

Copyright © 2020, Texas Instruments Incorporated

4 Unified Memory Controller

The unified memory controller is the L2 cache controller in the CorePac. The CorePac memory system is the next generation caches and memory controller system for the TMC320C7x fixed and floating point DSP. It can support one C7x CPU core, with private L1 caches and controllers, a streaming engine, and an unified L2 cache and memory controller.

The unified memory controller contains the interfaces to the L2 memory, the L1D, the L1P, streaming engines, and the MSMC. The interface to MSMC allows the UMC and CPU to communicate with peripherals and external memory. The UMC can provide bandwidth of up to 2048 bits of data per cycles, which is an 8× bandwidth improvement over previous generations. The UMC can queue up multiple transactions out to the next level of memory, and can handle out of order data return. The UMC has full soft error correction (ECC) on its data and TAG rams. The UMC supports full memory coherency, where all the internal caches and memories (L1, L2) are kept coherent to each other and external caches and memories (MSMC, L3, DDR). This memory system support virtual memory, and includes as part of it address translation, uTLBs, L2 page table walk, and L1P cache invalidates. The shared L2 controller can support one streaming engines, each with two streams. The streaming engines are kept coherent to the L1D cache, and have a pipeline, high-bandwidth interface to L2. The L2 cache also features higher associativity (8-ways).

Figure 11 shows the CorePac, and the UMC in relation to it.

Figure 11. UMC/L2 Instantiation in CorePac

4.1 Features

CorePac UMC has a number of new features compared to previous generations of CorePac. These features are listed here.

- L2 memory
  - Virtual and physical banked L2 memory, shared by L2 SRAM and L2 cache. Support for 4 and 2 virtual banked memories, with independent access control for each bank.
    - 4 × 512-bit physical banks with 2 virtual banks each
    - 4 × 512-bit physical banks with 4 virtual banks each. This configuration is not supported for the first version of the C7x.
  - Firewall on L2 SRAM accesses
  - DMA access to L2 SRAM on merged MSMC I/F
  - L2 sizes supported from 64KB to 2MB
- Virtual memory
  - Support for wider physical address
  - Support for MMU to read page table from L2
  - Support for DVM message from MMU
- ECR interface
  - Previously memory-mapped registers are now ECR (Extended Control register) and directly accessible by the CPU through a new ECR interface.

Copyright © 2020, Texas Instruments Incorporated

  - Each CPU has a separate ECR interface to access internal UMC config registers.
- Level 2 cache
  - 8-way set associative
  - Support for cache sizes: 64KB to 1MB
  - Random replacement policy
  - 128-byte cache line size
  - Read and Write-allocate cache
  - Support for write-back and write-through modes (write-through mode is not supported in the first generation, and functions similar to write-back)
  - Physically indexed, physically tagged (44-bit physical address)
  - 4x banked TAG RAMs, which allows for four independent split pipelines
  - Supports 2 × 64-byte streams from one streaming engine
  - Config/MMR and MDMA accesses on an unified interface to MSMC
  - Support for allowing msmc_sram region to be cached in L2
  - Default is that L2 does not allocate msmc_sram. This matches the C66x implementation. The ECR register L2CFG has a bit that can be programmed to allow msmc_sram to be allocated in L2
- Coherence
  - Full coherence between L1D cache, 2 streams, L2 SRAM, MSMC SRAM, and external (DDR)
  - Snoops for L2 SRAM, MSMC SRAM, and external (DDR) addresses
  - Full MESI support. The shared state is not used by the cache controllers, but that is a micro-architectural detail that is not visible to the user.
  - User coherence commands from the streaming engine.
  - Support for global coherence operations
- Error detection and correction
- (EDC / ECC)
  - Full ECC support for both TAG and data RAMs: 1-bit error correction and 2-bit error detection for both
  - Provide ECC syndrome on writes and victims out to MSMC
  - Full ECC on data read from MSMC
  - Read-Modify-Writes on DMA stores to keep parity always valid
  - Auto-scrub to prevent accumulation of 1-bit errors, and to refresh parity
  - Clear parity on reset
- Emulation
  - DAS codes are returned on reads on the DTRACE bus to indicate the level of cache that the data was read from.
  - DAS codes are returned to indicate pass / fail status of emulation writes on the STRACE bus.

4.2 Interfaces

Figure 12 shows interfaces to UMC and the major functional blocks.

Copyright © 2020, Texas Instruments Incorporated

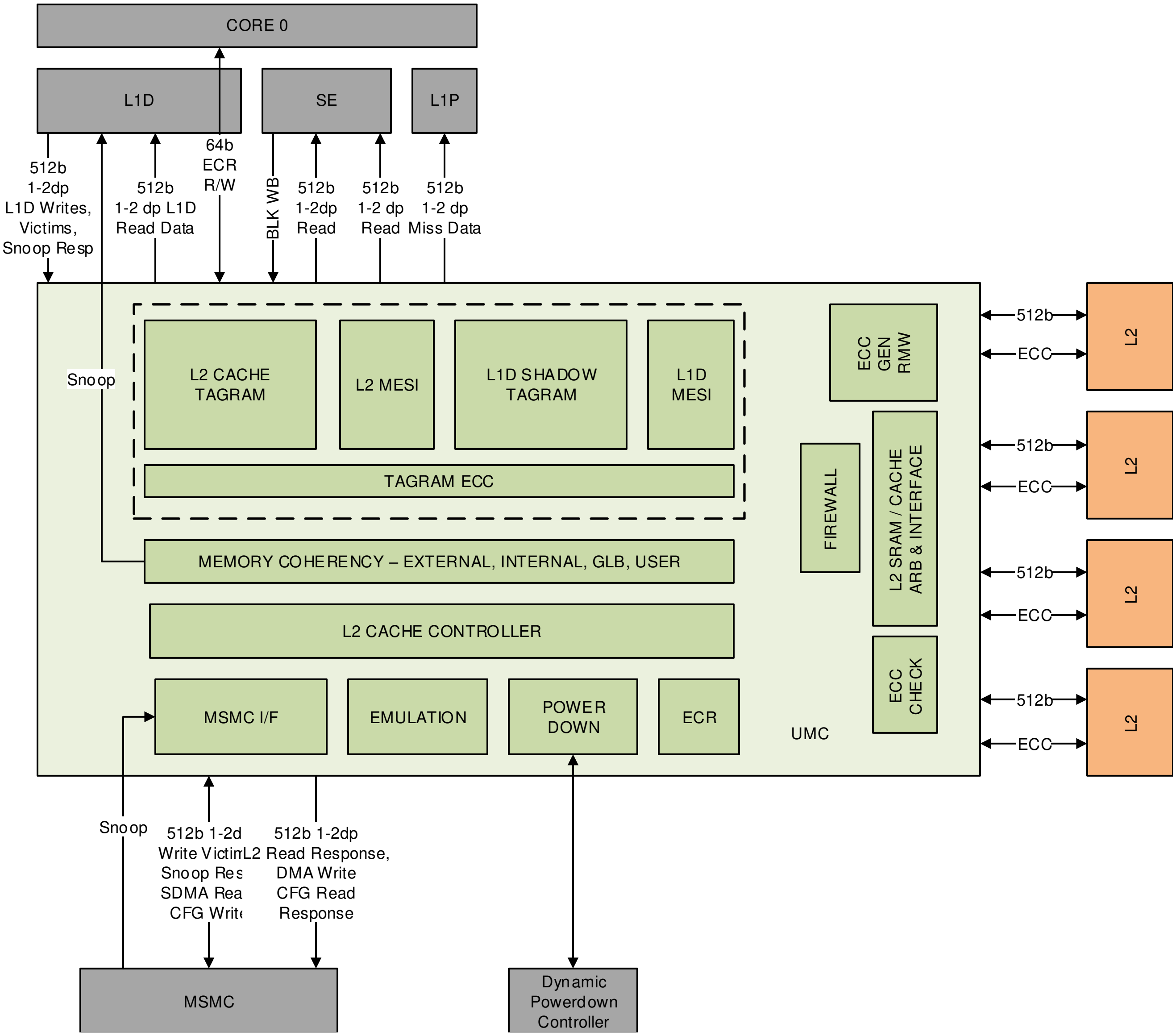


Figure 12. UMC Interfaces and Block Diagram

The UMC has six requestor ports: one PMC, one DMC, two SE ports (one streaming engine), internal ECR interface from CPU, and the MSMC. This list describes the various interfaces to UMC.

- The DMC interface has separate 512-bit read and write paths. This interface can also be used for snooping from the L1D cache. Each read transaction can be either one or two data phases.
- The PMC interface consists of a 512-bit read-only path (L1P fetch only). Each read transaction can be either one or two data phases.
- Two SE interfaces, which are 512-bit read-only. Each read transaction can be either one or two data phases. These commands are also used as part of the user block coherence functionality.
- The MSMC interface consists of separate 512-bit read and write paths. These interfaces are also used for snoop commands, read/write accesses to L2 SRAM, and read/write accesses to L1D SRAM. Each read transaction can be either one or two data phases.
- The internal ECR interface from the CPU is a 64-bit non-pipelined interface, used for config accesses to UMC ECR registers.
- Four 512-bit wide L2 memory ports, which allows parallel access to four 512-bit wide L2 memory banks.

Copyright © 2020, Texas Instruments Incorporated

NOTE: Figure 12 shows an optional second CPU core, which is not supported in the first version of the C7x.

4.2.1 UMC to DMC Interface

The UMC to DMC interface consists of the following:

- 512-bit DMC read path
- 512-bit DMC write path
- DMC to UMC signals:
  - Read/Write/Victim address
  - Address and secure cache line evicted to victim buffer
  - Address and secure cache line evicted from victim buffer
  - Two tag update interfaces to indicate a clean line which was evicted from the victim buffer
  - Byte enables
  - Read/write indicator
  - MMU page table attributes/privilege/secure level indicators
  - Snoop response
  - L1D cache-mode signals, including size, size change on, global coherence on, and global coherence type
- UMC to DMC signals
  - Snoop signaling
  - Response on reads and writes

4.2.2 UMC to PMC Interface

The UMC to PMC interface consists of the following:

- 512-bit PMC read path
- PMC to UMC fetch address
- Any required additional handshaking

4.2.3 UMC to SE Interface

The UMC to SE interface consists of the following:

- 512-bit SE read path
- SE to UMC fetch address
- SE to UMC user block coherence indicators

4.2.4 UMC to MSMC Interface

The MSMC to UMC interface carries multiple types of transactions:

- Master DMA (MDMA) – Includes cache allocates, victims, long distance writes, and non-cacheable reads. These originate from UMC.
- External config (ECFG) – These are read/write accesses to memory-mapped registers which are outside the CorePac. These originate from UMC.
- DMA – These originate from MSMC and are transactions that can transfer data between different CorePacs, CorePac and DDR, or CorePac and some other memory on the SoC. These are created by the EDMA or DRU, and can be to either L2 SRAM or L1D SRAM.
- Snoop – These originate from the MSMC, and are in response to a transaction from some other core or IP, which needs data to be snooped out of CorePac.
- Cache warm – MSMC can originate transactions that UMC uses to allocate a line from external to its

Copyright © 2020, Texas Instruments Incorporated

cache.

The UMC to MSMC interface consists of the following:

- 512-bit MSMC read path
- 512-bit MSMC write path
- MSMC to UMC signals
  - Address
  - Byte enables
  - Read/write indicator
  - MMU page table attributes/privilege/secure level indicators
  - Snoop transactions
  - DMA transactions
  - Cache warm transactions
- UMC to MSMC signals
  - Snoop response
  - Address
  - Byte enables
  - Read/write indicator
  - MMU page table attributes/privilege/secure level indicators

4.2.5 UMC ECR Interface

Memory-mapped registers on previous generations have been replaced by Extended Control registers (ECR), and are mapped to the MOVC CPU instruction. These are directly accessed by the CPU through a new interface called the ECR interface, which is described in the CorePac integration and CPU specs.

The UMC ECR path allows for 64-bit read/write access to the UMC’s control registers. For configuration reads, the UMC samples the contents of the register and holds it for the duration of the access.

The UMC ECR interface consists of the following, at a minimum:

- 64-bit ECR read path
- 64-bit ECR write path
- Address
- Privilege/secure level indicators
- Index, used for the cache tag view feature
- Any required additional handshaking

4.2.6 UMC to MMU Interface

- 64-bit read path
- Address
- Any required additional handshaking

4.2.7 UMC to L2 Interface

This interface assumes a 4 virtual bank, 4 physical bank RTA L2 memory, with each bank 512 bits wide.

- 512-bit read datapath
- 512-bit write datapath
- Address
- Byte-enables
- Memory enable indicator
- Read/Write indicators

Copyright © 2020, Texas Instruments Incorporated

- Virtual bank select

4.3 Register Summary

UMC configuration registers are accessed through the internal configuration interface. From the CPU view, these are not memory mapped, but, instead, are mapped to CPU register space, and accessed by CPU instructions.

Copyright © 2020, Texas Instruments Incorporated

Table 51. UMC ECR Registers Summary

REGID	Register	Description	PERMISSIONS															
			GU	GS	RU	RS	SU	SS	SDBG	NSDBG	GU	GS	RU	RS	SU	SS	SDBG	NSDBG
			Write								Read							
0x280	L2CFG	L2 Cache Mode Configuration	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1
0x281	L2CC	L2 Cache Control	0	1	0	1	0	1	0	0	1	1	1	1	1	1	1	1
0x282	L2WB	L2 Global Writeback	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x283	L2WBINV	L2 Global Writeback with Invalidate	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x284	L2INV	L2 Global Invalidate	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1
0x286	L2CTAG	L2 Cache tag View	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0x287	L2EDCFG	L2 EDC Configuration Control	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1
0x291	L2ADDREEA	L2 Addressing Error Event Address	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x292	L2ADDREES	L2 Addressing Error Event Status	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x293	L2ADDREER	L2 Addressing Error Event Reset	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1
0x294	L2ALLOCEEA	L2 Cache Allocation Error Event Address	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x295	L2ALLOCEES	L2 Cache Allocation Error Event Status	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Copyright © 2020, Texas Instruments Incorporated

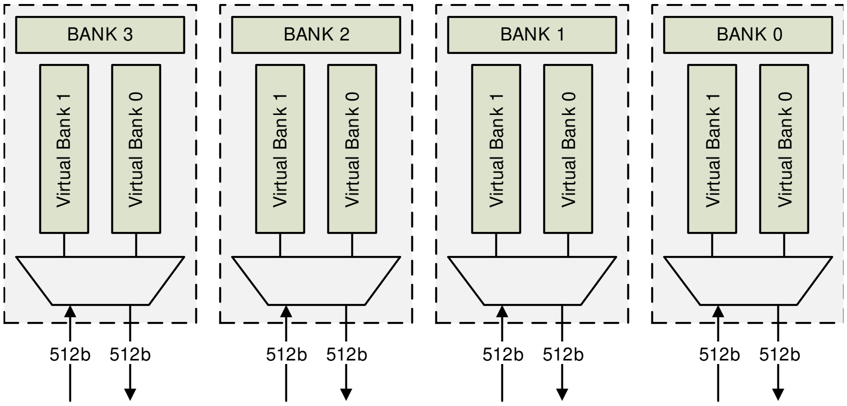
Table 51. UMC ECR Registers Summary (continued)

REGID	Register	Description	PERMISSIONS															
			GU	GS	RU	RS	SU	SS	SDBG	NSDBG	GU	GS	RU	RS	SU	SS	SDBG	NSDBG
			Write								Read							
0x296	L2ALLOCEER	L2 Cache Allocation Error Event Reset	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1
0x297	L2MEMMAP	L2 Memory Map Control	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x298	L2MSWCTL	L2 Memory Switch Control	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x299	L2MEMMAPIBUFA	L2 IBUFA Memory Base Address	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x29A	L2MEMMAPIBUFB	L2 IBUFB Memory Base Address	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x29B	L2MEMMAPWBUF	L2 WBUF Memory Base Address	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x29C	L2MSWERRSTAT	L2 Memory Switch Error Status	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x29D	L2MSWERRCLR	L2 Memory Switch Error Clear	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1
0x29E	L2MSWERRADDR	L2 Memory Switch Error Address	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x29F	L2EDTST	L2 EDC Scrub Test Control	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1

Copyright © 2020, Texas Instruments Incorporated

4.4 L2 Memory

The UMC has four 512-bit wide memory ports, referred to as UMC Memory Access Port (UMAP) ports. It has 2 virtual banks, shown in Figure 13.



-

Figure 13. L2 SRAM Bank Interface - 2 Virtual Banks

Each L2 SRAM interface supports one new access per UMC cycle if the required memory banks behind the SRAM are ready. Accesses may be pipelined over multiple UMC cycles, allowing high-latency memories to be used. Each of the virtual banks has its own ready latency. Each interface looks at the availability of each virtual port, not just the entire bank.

4.4.1 L2 SRAM Protocol

The UMC L2 SRAM protocol caters to a memory connected directly to UMC. The UMC presents address and read/write indication on the UMAP boundary, and after a fixed latency, expects the transaction to be complete. UMC is able to control the four banks independently. Each of the banks has two or four virtual banks. Accesses to these virtual banks are issued serially.

Consecutive requests to the same virtual bank result in a “bank conflict” if the attached memory has greater than 1 cycle pipeline latency. The second request is delayed until the first request completes. Consecutive requests to different virtual banks can proceed without delay, regardless of the memory’s latency. The UMC expects the memory to return read data after the programmed access latency. Two different types of latencies are supported: pipeline latency and access latency. Pipeline latency is the number of cycles that UMC has to wait before it can access the same virtual bank again. Access latency is the number of cycles that it takes for the memory to present data to UMC, after the read command has been presented. CorePac supports latencies from 1 to 6 for both pipeline and access latencies. 1 and 2 cycle access latencies are also referred to as “0 wait-state” and “1 wait-state,” respectively.

4.4.2 UMAP Banking

The CorePac UMC implements a banking scheme that is different in some ways from prior CorePac generations. The primary difference is in the number of parallel accesses. Previous generations allowed a single access on each of the UMAP ports. CorePac allows each of the four banks of the L2 memory to be accessed independently. This results in up to a maximum of four L2 access every cycle. The organization of addresses in these memories is also different, as shown in and Table 52. shows 4 banks, each of which has 4 virtual banks, indicated by VB followed by the bank number. This is the organization for an SRAM address. The organization for a cache address is shown in Table 54. Each virtual bank is 64 bytes wide. Thus, a row in each bank can hold four times that, which is 256 bytes.

Table 52. L2 SRAM Organization - 2 Virtual Banks

Bank 3		Bank 2		Bank 1		Bank 0	
V1	**V0**	**V1**	**V0**	**V1**	**V0**	**V1**	**V0**
0x3C0	0x380	0x340	0x300	0x2C0	0x280	0x240	0x200
0x1C0	0x180	0x140	0x0100	0x0C0	0x080	0x040	0x000

Copyright © 2020, Texas Instruments Incorporated

4.4.3 L2 Memory Map

MSMC decodes and routes accesses to L2 SRAM with 16MB granularity. UMC further refines the address decoding, restricting accesses to the size configured for the UMAP.

The inputs umc_baddr[3:0] and umc_sz[3:0] determine the address range that the UMAP maps into. The umc_sz[3:0] input determines the size of the UMAP, and umc_baddr[3:0] determines the starting address within CorePac's internal address space, down to 1MB granularity.

This generation of Corepac restricts the UMAP address range to 000_0080_0000 - 000_00BF_FFFF, by restricting umc_baddr to 0100b, and umc_sz to no larger than 4MB. The UMAP adjust feature, as specified by umc_adj[7:0], is not supported in CorePac. Half power-of-two UMAP sizes are supported.

The configuration inputs umc_sz sets the size of the UMAP. The UMAP may be set to a size in the range 64K to 4086K, as allowed by Table 53. describes how system integrators must configure UMAPs that have no memory attached.

The umc_sz input defines the precise address map for each UMAP. Accesses outside the exact memory range specified by these inputs are subject to multiple constraints, as defined in Section 4.4.4. An address is within the implemented UMAP memory range if it falls within the exact address range specific in this table.

Table 53. Valid UMAP Sizes

umc_sz[3:0]	UMAP Memory Size	UMAP Start Address	UMAP End Address
0000b	0K bytes	Not Supported	
00001	64K bytes	000_0080_0000	000_0080_FFFF
0010b	96K bytes	000_0080_0000	000_0081_7FFF
0011b	128K bytes	000_0080_0000	000_0081_FFFF
0100b	192K bytes	000_0080_0000	000_0082_FFFF
0101b	256K bytes	000_0080_0000	000_0083_FFFF
0110b	384K bytes	000_0080_0000	000_0085_FFFF
0111b	512K bytes	000_0080_0000	000_0087_FFFF
1000b	768K bytes	000_0080_0000	000_008B_FFFF
1001b	1024K bytes	000_0080_0000	000_008F_FFFF
1010b	1536K bytes	000_0080_0000	000_0097_FFFF
1011b	2048K bytes	000_0080_0000	000_009F_FFFF
1100b-1111b	RESERVED		

4.4.4 Accesses Outside Implemented L2 Memory

UMC always decodes the UMAP address ranges exactly, down to the size specified for each UMAP, and listed in Table 53. MSMC, PMC, and DMC, however, route accesses to UMC based on a coarse 1MB address decoding.

MSMC accesses that target a UMAP, but which are outside the allowed size of that UMAP, are considered invalid. Also, accesses to addresses inside CorePac's address space arriving from DMC and PMC, but which are outside both UMAP's address spaces, are invalid. The UMC responds as follows to invalid accesses, based on the requestor:

- All invalid accesses: Reads return garbage, writes get dropped.
- Functional accesses from DMC, PMC:
  - UMC does not signal a memory protection fault.
  - The requestor ultimately signals a protection fault: DMC for data fetches, the CPU for program fetches.
- DMA accesses from MSMC:
  - UMC signals memory protection fault.
  - UMC returns zero permissions and address error.

Copyright © 2020, Texas Instruments Incorporated

- EDI reads from DMC:
  - UMC returns DAS code NA on the DTACE bus.
  - UMC does not signal a memory protection fault.
- EDI writes from DMC:
  - UMC records the error status in a memory-mapped ECR register.

4.4.5 L2 SRAM Clean on Reset

After reset, UMC clears its entire L2 SRAM. Zeroes are written to every bit in the memory. This is done to clear the ECC syndrome for all data in the L2SRAM. It is not possible to disable this feature.

4.5 ECR Interface

Unlike previous CorePac generations, the configuration registers are not memory mapped. They are treated as part of the CPU control space and accesses through ISA instructions. The CPU decodes these instructions and sends out accesses to the memory system and other CorePac configuration registers on the Internal Config ECR interfaces. This interface is similar to the current ERI interface, that the CPU uses to access the ECM memory-mapped registers in CorePac.

Each CPU has an independent ECR interface to the UMC. The UMC is responsible for correctly arbitrating and sequencing transactions on these two interfaces.

Accesses to the UMC's ECR registers at any time shall not cause UMC to behave abnormally—that is, inconsistent with the behaviors defined in this specification. This specification expects that configuration accesses are independent from all other traffic in UMC. If a configuration access must arbitrate for a shared resource in UMC, the configuration accesses are considered higher priority than any other access to UMC.

4.6 Virtual Memory

UMC supports virtual memory. Virtual address are 48 bits wide, while physical addresses are 44 bits wide. The addresses presented to UMC from both L1 and MSMC are physical addresses and are thus 44 bits wide. L2 cache is physically tagged and mapped, and works on 44 bits of address. The MMU and uTLB's are located in L1P and CPU.

4.6.1 MMU Page Table Miss

On a MMU page table miss, it reads the page table entry from the L2. This interface and functionality has been described in the MMU Architectural Specification Document.

4.6.2 MMU DVM Message

The first implementation of the C7x does not support DVM messaging.

4.7 Level 2 Cache

The L2 cache is a read- and write-allocate, eight-way set associative cache. The UMC maps its entire L2 memory map as SRAM. The user must program a memory-mapped register in UMC to enable L2 cache. The L2 controller supports 32KB, 64KB, 128KB, 256KB, 512KB, and 1MB cache sizes. The cache is a 8-way set-associative cache. The L2 cache is physically tagged and physically mapped. All L2 memory beyond 1MB is SRAM.

The address presented to the L2 cache controller is always a physical address, not the virtual address. CorePac supports a 44-bit physical address.

4.7.1 L2 Cache Organization

To track the line state of the L2 cache, a eight-way tag RAM is included. The address organization within the L2 tags is a function of the partitioning performed between cache and RAM, controlled through the L2CFG.L2MODE control register bits, as described in Section 4.7.10.1. Table 54 outlines this function for the various sizes of cache supported. L2 cache is physically indexed and tagged.

Copyright © 2020, Texas Instruments Incorporated

Table 54. L2 Cache Address Organization

43	X+1	X	7	6	0
Tag		Set		Offset	

The offset of 7 bits accounts for the fact that an L2 line size is 128 bytes. The cache control logic ignores this portion of the address. The Set field indicates the L2 cache line address where the data would reside within in each way, were it to be cached. The width of the Set field depends on the amount of L2 configured as cache, as defined in Table 55. The UMC uses the Set field to look up and check the tags in each way for any already-cached data. It also looks up the Valid bit, which indicates whether the contents of the line are considered valid for purposes of a tag compare.

The set field bits are extracted from the physical address, which implies that the L2 cache is physically tagged.

The size of the Set and Tag fields are dictated by the L2 cache configuration, as described in Table 55.

Table 55. L2MODE Description

L2CFG.L2MODE Setting[1]	Amount of L2 Cache	'X' Bit Position	Description
000b	0K	N/A	L2 is all RAM
001b	32K	11	32 L2 cache lines
010b	64K	12	64 L2 cache lines
011b	128K	13	128 L2 cache lines
100b	256K	14	256 L2 cache lines
101b	512K	15	512 L2 cache lines
110b	1024K	16	1024 L2 cache lines
111b	"Maximal Cache." Maps to 1024K on CorePac. This may map to 2048K cache mode on a future Corepac.		

[1] In general, a larger value of 'L2MODE' specifies a larger cache size, up to the size of the implemented L2 memory in UMAP. The maximum actual L2 cache size is the smaller of "largest power of 2 that fits in L2 RAM size" and 1024K. The UMC must clamp the L2 cache size to this maximum based on the configured size of UMAP.

The Tag field is the upper portion of the address that identifies the true physical location of the cache line. The cache compares the Tag field for a given address to the stored tag in all eight ways of the L2 cache. The tag field bits are part of the physical address, because the L2 cache is physically addressed.

If any of the tags match and the cached data is valid, then the access is a "hit," and the element is read directly from or written directly to the L2 cache location. Otherwise it is a "miss," and L2 fetches a complete line from its system memory location. On read misses, the data is passed directly to the appropriate L1 cache as part of the fetch. On write misses, the L2 merges the write with the fetched line.

This allocate behavior of the cache is controlled by the MMU page table entry. Options are available for allowing for read and write allocate independently.

Because the contents of the L2 can be modified, it must be capable of updating the data in its true physical location. The L2 cache is a write-back and write-through cache. This behavior is configurable and programmed by the MMU page tables. Write back indicates that the cache writes out updates only when it needs to. Write-through indicates that the cache writes out updates on every write coming from L1D. The UMC evicts data from the L2 cache, writing it back to its proper location in system memory. This occurs when a new L2 line replaces one that has been modified, or when the L2 controller is told by the CPU (through the streaming engine) to write back modified data. In the event of an eviction or writeback, the data is sent to its location in system memory via the MSMC.

The C7x does not support write-through mode. The write-through mode behaves similar to write-back mode.

Copyright © 2020, Texas Instruments Incorporated

4.7.2 L2 Cache Banking

The L2 cache starts from the top of the L2 memory, and goes in the direction of lower addresses. The set to bank mapping is as below for the four and two virtual banking configurations.

4.7.2.1 Two Virtual Banking

The “V” indicates virtual bank in Table 56.

- Sets [n]: Bank 0
- Sets [n+1]: Bank 1
- Sets [n+2]: Bank 2
- Sets [n+3]: Bank 3

All eight ways of each set are contained in the same physical bank.

Table 56. L2 Cache Banking - 2 Virtual Banks

Bank 3		**Bank 2**		**Bank 1**		**Bank 0**	
V1	**V0**	**V1**	**V0**	**V1**	**V0**	**V1**	**V0**
512b	**512b**	**512b**	**512b**	**512b**	**512b**	**512b**	**512b**
Set 3 Way 0		Set 2 Way 0		Set 1 Way 0		Set 0 Way 0	
Set 7 Way 0		Set 6 Way 0		Set 5 Way 0		Set 4 Way 0	
Set 3 Way 1		Set 2 Way 1		Set 1 Way 1		Set 0 Way 1	
Set 7 Way 1		Set 6 Way 1		Set 5 Way 1		Set 4 Way 1	
Set 3 Way 2		Set 2 Way 2		Set 1 Way 2		Set 0 Way 2	
Set 7 Way 2		Set 6 Way 2		Set 5 Way 2		Set 4 Way 2	
Set 3 Way 3		Set 2 Way 3		Set 1 Way 3		Set 0 Way 3	
Set 7 Way 3		Set 6 Way 3		Set 5 Way 3		Set 4 Way 3	
.....		.....		.....		.....	
Set n+3 Way [0-7]		Set n+2 Way [0-7]		Set n+1 Way [0-7]		Set n+0 Way [0-7]	
Set n+7 Way [0-7]		Set n+6 Way [0-7]		Set n+5 Way [0-7]		Set n+4 Way [0-7]	
.....		.....		.....		.....	
0x3C0	0x380	0x340	0x300	0x2C0	0x280	0x240	0x200
0x1C0	0x180	0x140	0x0100	0x0C0	0x080	0x040	0x000

4.7.3 Replacement and Allocation Strategy, and Cache Inclusivity

The L2 cache operates with a fixed eight-way set associativity in all cache modes. This means that each location in system memory can reside in any one of eight possible locations in the L2 cache.

The L2 controller implements a read- and write-allocate cache. This means that the L2 fetches a complete line of 128 bytes on any L1 or streaming engine miss for a cacheable location, regardless of whether it is a read or a write. The replacement strategy is random replacement. This is different from CorePac1 (C6x), which had a four-way LRU cache.

L2 cache is not inclusive of the L1D and L1P caches. This has multiple implications, as listed below.

- L1D victims can miss L2 cache, and do not get allocated in L2 cache.
- A clean eviction from either L1D or L1P cache does not cause L2 to evict the line from its cache.

Copyright © 2020, Texas Instruments Incorporated

- L2 victim does not snoop or invalidate the line from either of the L1 caches.

However, coherency is maintained in all situations. Memory coherency is described in a separate chapter, but to summarize:

- Read or Writes from MSMC snoop or change the state of both the L2 and the L1D copy of that cache line.
- A coherence maintenance transaction from the streaming engine that causes L2 to writeback a line from its cache snoops the line from the L1D cache. The appearance of inclusivity is created, but this is primarily to honor the coherency request.

NOTE: A configuration is planned which lets the user control the way allocation among the two cores. This is controlled by a bit in the L2CACHE ECR register (not yet defined). This configuration forces the cache allocation logic to allocate ways 0 - 3 to core 0 and ways 4 -8 to core 1. This feature is not supported in the first version of the L2 controller.

4.7.4 Response Errors

For read allocates that return from MSMC with an error, this causes the line to not be allocated in L2. However, it stays allocated in the snoop shadow RAM, and can cause an extra snoop later to L1D, if L1D had indicated on the read that it intended to cache this line.

On write allocates that respond with a data error, UMC drops the write allocate.

4.7.5 L1D Victims Policy

L1D victim writebacks do not trigger line allocations in L2. L1D victims are written directly to external memory if they miss L2.

They do update L2’s dirty status as needed on hits.

In the case where a L1D read miss causes both a L2 and a L1D victim, these two victims may not be merged, and could go out to external as two separate victims. L1D has a victim buffer where it moves all dirty lines to, on a read miss that replaces a dirty line from its cache. In that case, the dirty line does not get written back to L2, and stays in the L1D victim cache. There exists signalling between L1D and L2 to track these lines.

4.7.6 L2 Writeback Victims and Evictions

There are three instances in which L2 cache lines are evicted or written back:

1. L2 evicts clean lines when it needs to replace that line for an allocate.
   - The clean line is dropped. The L1 caches are not snooped or invalidated, and neither is any eviction notice sent down to MSMC.
   - At the end of this eviction, this line stays valid in the MSMC snoop filter, which is expected, because the line may still be valid in the L1D cache.
   - A small performance impact of this occurs if the line was not present in the L1D cache, and is evicted from the L2 cache. Because the line stays valid in the MSMC snoop filter, MSMC snoops this line from L2. On the first such snoop, where L2 indicates that it does not have the line, it is removed from the MSMC snoop filter.
2. L2 writebacks lines when it must replace a dirty line for an allocate.
   - The dirty line gets written back.
   - To main coherency, this writeback is sent as a WriteClean to MSMC. This indicates that although L2 is evicting the line, this line may be present in the L1D cache. This is a result of L1D and L2 not being inclusive. This can cause a future L1D writeback of this line, if the line was indeed valid and dirty in the L1D cache.
3. L2 coherence writeback. These writebacks can happen as a result of a cache mode change, global coherence operation, or a single cache line coherence transaction from the streaming engine. In this case, the user wants to clean this line from all caches by forcing a coherency operation. Coherency is maintained similar to the second instance described above.

Copyright © 2020, Texas Instruments Incorporated

4.7.7 Allocation of MSMC_SRAM

The default behavior is for L2 to not allocate from MSMC_SRAM. This matches the C66x behavior. The L2 CFG register has a bit, L2CFG.M3CACHE, which can be used to change this behavior. L2CFG.M3CACHE is set to 0 on reset.

- Default: MSMC_SRAM is never cached in L2.
- L2CFG.M3CACHE = 0: UMC does not allocate from MSMC_SRAM.
- L2CFG.M3CACHE = 1: UMC looks at cmemtype and ccinner to decide whether to allocate from MSMC SRAM in L2.

4.7.8 CPU and DMA/IDMA Accesses to L2 Addresses Configured as Cache

The UMC detects accesses to L2 address configured as cache from any requestor and treats them as invalid accesses. Writes are dropped, and trigger an exception. Reads return a read status indicating address error. This behavior is different from the C66x, where such accesses were allowed and used as a mechanism to scrub the parity RAM.

4.7.9 Reset Considerations

In response to a hard reset, the UMC switches the L2 cache to All-RAM mode. While coming out of reset, the UMC clears all its RAMs.

4.7.10 Program-Initiated Cache Controls

In addition to the L2 cache and RAM access, the controller also provides access to several memory-mapped control registers that the CPU can access to configure the L2 operating mode.

To eliminate a large number of potential lock-up conditions in the system, the L2 control registers are only accessible by the CPU. This prevents the DMA, MSMC, host processors (through HPI, PCI, or other), and other CPUs in a multi-processing environment from accessing these registers. These registers are accessible through 64-bit reads and writes on a separate internal config interface, which is initiated by the CPU.

User/program-initiated cache control operations fall into three categories, covered in the following sections:

- Cache size and operating mode controls: Section 4.7.10.1 lists the register that controls the size of the L2 cache.
- User-initiated global coherence operations: Section 4.7.11 – these operations allow programs to manually move data out of the cache.
- User-initiated block/single cache line coherence operations: Section 4.7.12 – these operations allow programs to manually move data out of the cache.

4.7.10.1 Cache Configuration Registers

An ECR register controls the operating modes of L2 cache, as shown in Table 57.

Table 57. L2 Cache Configuration Register

Register Name	ECR REG ID	Lives In	Action
L2CFG	0x280	UMC	Sets the amount of L2 memory that acts as cache. Controls cache reset/invalidate on enabling cache (for retention)

4.7.10.2 L2 Mode Change Operations

The L2MODE field in the L2CFG control register controls the size of L2 cache. Programs set the amount of L2 memory mapped as L2 cache by writing the desired cache mode to this field. Table 58 illustrates the valid settings for L2MODE.

Copyright © 2020, Texas Instruments Incorporated

Table 58. Cache Size Specified by L2CFG.L2MODE

L2CFG.L2MODE setting	Amount of L2 Cache
000b	0K
001b	32K
010b	64K
011b	128K
100b	256K
101b	512K
110b	Reserved
111b	Maximal cache

Typically, programs set the L2 mode shortly after reset and leave it static. Some programs, however, change the L2 cache mode on the fly, particularly around OS task switches in a complex system. Programmers must ensure that memory system coherence and correct cache operation are maintained by following this procedure.

Table 59 outlines the required steps that the programmer must perform.

Table 59. Switching L2 Modes

To switch from…	To…	The program must perform the following steps…
A mode with no or some L2 cache	A mode with more L2 cache	• DMA, IDMA or copy any needed data out of the affected range of L2 RAM. (If none requires saving, no DMA is necessary.) • Wait for completion of any DMAs/IDMAs issued in the previous step. • Write the desired cache mode to the L2MODE field in L2CFG. • Issue MFence OR Read back L2CFG. This stalls the CPU until the mode change completes.
A mode with some L2 cache	A mode with less or no L2 cache	• Write the desired cache mode to the L2MODE field in L2CFG. • Issue MFence, OR Read back L2CFG. This stalls the CPU until the mode change completes.

When a program writes a new cache mode to the L2CFG register, UMC performs the following steps, at a minimum.

- The L2 cache, if enabled, must be written back and invalidated.
- The L2 cache must be initialized for the requested mode.
- The SRAM region impacted due to this cache size change is cleared, by writing zeroes.

Changing L2's mode does not affect the contents of either L1 cache.

4.7.11 Global Coherence Operations

The memory system makes a variety of global coherence operations available to programs through memory-mapped control registers. These registers initiate writebacks and invalidates for data held in the various caches, and operate on the entire contents of one or more caches. Table 60 lists the memory-mapped registers that provide these operations. Table 61 lists all of the global cache commands and the operations they perform on each of the three caches.

Table 60. Global Coherence Control Register Summary

Register Name	Address	Function
L2WB	TBD	L2 Global Writeback
L2WBINV	TBD	L2 Global Writeback with Invalidate
L2INV	TBD	L2 Global Invalidate without writeback

Copyright © 2020, Texas Instruments Incorporated

Table 61. Global Coherence Cache Operations

Cache Operation	Register Usage	Effect on L2 Cache[1]
L2 Global Writeback	L2WB	Writes back all updated lines in L2 to external. Leaves all of L2 clean.
L2 Global Writeback with Invalidate	L2WBINV	Writes back all updated lines in l2 to external. Leaves all of L2 invalid.
L2 Invalidate	L2INV	Invalidates the entire contents of L2. Discards all updated lines.

[1] Lines written back from L1D for external addresses through L1DWB and L1DWBINV may be written back externally if L2 cache does not hold the corresponding address. This can occur because CorePac does not keep L1D inclusive within L2 cache.

Programs initiate global cache operations by writing a 1 to the appropriate register bit. For each of L2WB, L2WBINV, and L2INV, this bit is bit 0 of the corresponding register.

Programs can blindly write a 1 to the control register to initiate the coherence operation. UMC stalls the write if necessary. Programs may poll this bit to determine when the command completes.

The hardware does not require programs to poll for completion of these commands. The caches stall programs while the global commands proceed.

Global cache operations are serviced by a simple state machine that queries (and possibly modifies or invalidates) the L2 tags and L1D shadow tags. The net result of this state machine is the movement and invalidation of data held in the cache, as described in Table 61.

If the CPU attempts to initiate a global cache operation while either a global or block cache operation is in progress, the UMC stalls the write if needed, until the prior operation completes.

4.7.12 Single Line and Block Cache Coherence Operations

Unlike previous generations of CorePac, where block coherence operations, similar to the global operations, were initiated by memory-mapped registers, CorePac uses a different approach. Program instructions initiate these block operations in the streaming engine, and the streaming engine then goes through the block, and sends single line coherence operations to L2. This is described in detail in the streaming engine specification document. These are described in Table 62.

Table 62. L2 Single Cache Line Coherence Operations

Cache Operation	L2 Effect
Writeback and Invalidate L2 line	Write back line if line has been updated to external. Line gets invalidate in L2 cache.
Invalidate L2 line	Line is invalidated from L2 cache.

NOTE: The writeback coherence operation, which would have left the line valid in the cache, is not supported in this version of the C7x.

4.7.13 Privilege and Cache Control Registers

UMC performs privilege, security, and memory protection on accesses to its cache control registers, so that none are subverted by program-initiated cache controls. **Table 2-25 ‘‘Permissions for Cache Control Registers’’** summarizes who may access which cache control registers, and what protection checks CorePac performs.

4.7.14 Cache Warm

The UMC cache controller supports pre-allocation of lines into the cache by MSMC or SE. In the SE case, this is implemented by SE sending a read, and UMC is ignorant to the fact that this is cache warm.

UMC supports two flavors of cache warms from MSMC:

- Cache warm shared: allocate the line in shared MESI state.
  - If the line is already present in either L2 or L1D cache in any MESI state, the line is not re-allocated, and the cache warm operation is considered a success.

Copyright © 2020, Texas Instruments Incorporated

- If the line is not present in either L1D or L2 cache, then this line is allocated into the L2 cache.

NOTE: Cache warm shared is not supported for this version of the C7x, and behaves similar to cache warm unique.

- Cache warm unique: allocate the line in unique MESI state.
  - If the line is already present in either L2 or L1D cache in the unique or modified MESI states, the line is not re-allocated, and the cache warm operation is considered a success.
  - If the line is not present in either L1D or L2 cache, then this line is allocated into the L2 cache in the unique state.
  - If the line is present in the L2 cache in shared state, then it is invalidated, and re-allocated into the L2 cache in the unique state.
  - If the line is present in the L1D cache in shared state, then the line does not get invalidated from the L1D cache, but is re-allocated into the L2 cache in the unique state.

There are some more cases where the cache warm is not honored, as listed here. In the first three cases, an allocated is not mastered by UMC.

- L2 cache is off.
- The ccinner attribute indicates that the line is not to be allocated in L2.
- SRAM line is attempted to be allocated.
- UMC attempts to allocate, but the line comes back with a read response error, and does not get allocated.

4.8 Coherence

Previous generations of CorePac supported limited coherency. It maintained coherency only among accesses originating from the CorePac (CPU load/store to L2SRAM), or passing through CorePac with their destination being L2 SRAM (DMA to L2SRAM).

In contract, the C71x CorePac maintains full memory coherency between L1D cache, L2 cache, MSMC SRAM, L3 cache, DDR, and system memory. This is done by a combination of full MESI support and snoop protocols. This feature has extensive dependencies in multiple memory controllers, and thus, the micro-architectural details have been described in a separate document.

The L2 SRAM region is not kept coherent across different cores. MSMC does not track the status of L2 SRAM lines in its snoop filter, and does not snoop cached L2 SRAM lines from a core. Snoops to L2SRAM address region are responded to with no data (ddatavalid = 0) and response status of success (dstatus).

In the case of a CleanSharedPOU or CleanSharedPOC CMO (block coherence operations - WriteBack), UMC may snoop the line out of L1D if present. In that case, DMC drops that line to a shared state, and if the line was dirty, passes down the dirty line to L2.

4.9 Error Detection and Correction (EDC/ECC)

The CorePac UMC provides support for error detection and correction for all memories contained in UMC and some registers files and flip-flops in the datapath. The primary purpose of this is to protect code and largely static data held in L2 memory. Equally important is protecting the state of the cache, which is done by performing ECC on the cache tag states in register files and some flip-flops. Register files have been protected where possible, while flip-flops have been protected by a novel way of passing ECC information between controllers.

The UMC can correct 1-bit and detect 2-bits (SECDED - single error correction, double error detection) on all read from its SRAM/cache, and all cache state RAMs, including TAG and MESI bits. The data read from these memories is corrected before being used for the next computation. In the event of a 1-bit error, correction is done. In the case of a 2 or more bit error that can be detected, care is taken to avoid any data corruption or device lock-up by taking an interrupt to the CPU.

Copyright © 2020, Texas Instruments Incorporated

EDC is performed on all reads from DMC, PMC, MSMC (snoops and DMA), and L2 victims. This is different from previous generations of CorePac where detection and correction was performed on specific accesses. In some cases, based on micro-architectural decisions, the actual correction may be distributed across L1 and L2 cache controllers.

Unlike previous generations, parity is always valid in all the memories that are protected. This removes the need for a parity valid bit.

4.9.1 EDC Implementation

The EDC implementation differs, depending on the memory being protected. These memories have been classified into three types:

- L2 SRAM / cache data memory: The ECC syndrome is computed and maintained at a 256-bit granularity for these memories. The banks are 512 bits wide, and thus each bank contains two 256-bit granules and two syndromes. The syndrome is read with the data. On writes, the syndrome is either pass on the interface transferring that write data, or computed by UMC. One bit errors are corrected. An interrupt is taken on two bit errors and the pertinent information recorded.
- L2 TAG, L1D shadow TAG: The error detection and correction of these memories is part of the cache pipeline. The corrected tag information is used in the rest of the pipeline, while a 2-bit error indicates incorrect cache state, and cannot be relied on.
- Register files, temporary storage: These memories include data re-order buffers, snoop response buffers, and so forth. These buffers hold temporary data, but can still be protected. The presence of absence of detection and correction for these memories depends on their size and micro-architectural decisions.

UMC has different interactions for error detection and correction based on the requestor and type of access.

4.9.2 PMC – UMC ECC Interaction

On reads from PMC, UMC does SECDED correction / detection on this data, and takes the appropriate action on errors. It also generates a single parity bit on 32-bit data chunks, and sends these 16 parity bits along with data to PMC.

4.9.3 SE – UMC ECC Interaction

On reads from the streaming engine (SE), UMC does SECDED correction / detection on this data, and takes the appropriate action on errors. It also generates a single parity bit on 32-bit data chunks, and sends these 16 parity bits along with data to the SE.

4.9.4 MMU – UMC ECC Interaction

On reads from the sCMMU, UMC does SECDED correction / detection on this data, and takes the appropriate action on errors. It also generates a single parity bit on 32-bit data chunks, and sends these 2 parity bits along with data to the MMU.

4.9.5 DMC – UMC ECC Interaction

DMC has a number of different access types as part of its interface with UMC. The ECC interactions for each of these types are:

- L1D read – UMC does not do any error detection or correction on read misses from DMC. Instead, it forwards all the ECC syndrome bits to DMC, and lets DMC do the detection and correction. This allows for ECC protection of not just L2, but also the pipeline registers on the datapath from L2 to L1D cache / CPU.
- L1D victims – DMC sends all ECC syndrome bits to UMC with the data. Because DMC supplies all the data and syndrome, UMC writes this information to its memory. It does not do any detection or correction. If this victim misses L2 and had to be forwarded to MSMC, UMC forwards the ECC syndrome bits that it received from DMC to MSMC, and lets MSMC handle the detection and correction of that data. This, again, allows for ECC protection of pipeline registers on the datapath from L1D to MSMC.

Copyright © 2020, Texas Instruments Incorporated

- L1D writes for less than 64 bytes – DMC sends writes down to UMC for less than 64 bytes in two situations: non-cacheable writes and merged writes. In both cases, DMC does not have enough bits to generate the ECC syndrome. UMC generates the correct ECC syndrome, by performing a read-modify write operation, and commits the data and syndrome to L2. In the case that this write misses L2, and had to be forwarded to MSMC, UMC does not generate ECC syndrome for that data. MSMC has the responsibility of generating that information. These types of L1D writes could be cacheable, and UMC writes allocate that line. The EDC interaction in described in Section 4.9.6.
- L1D snoop response data – This is treated similar to the L1D victim case.

In all the cases where DMC is responsible for detection and correction, it is DMC's responsibility to take an interrupt and record all pertinent information.

4.9.6 MSMC- UMC EDC Interaction

MSMC has a number of different access types as part of its interface with UMC. The ECC interactions for each of these types is listed below.

- MSMC reads – UMC does not do any error detection or correction on reads from MSMC. Instead, it forwards all the ECC syndrome bits to MSMC, and lets MSMC do the detection and correction. This allows for ECC protection of not just L2, but also the pipeline registers on the datapath from L2 to MSMC / L3 cache.
- MSMC 32-byte writes – MSMC sends all ECC syndrome bits to UMC with the data. UMC just writes this information to its memory.
- MSMC writes for less than 32 bytes – MSMC does not have enough bits to generate the ECC syndrome. UMC generates the correct ECC syndrome by performing a read-modify write operation, and commits the data and syndrome to L2.
- MSMC snoop – This is treated similar to MSMC reads.

In all the cases where MSMC is responsible for detection and correction, it is MSMC's responsibility to take an interrupt and record all pertinent information.

4.9.7 Functional Test for EDC Logic

UMC aligns to the Keystone 3 safety architecture provided testing mechanisms for the EDC detection and correction logics.

4.9.8 Error Capture and Reporting

UMC aligns to the Keystone 3 safety architecture for reporting of detected parity and EDC faults both from scrubbing and functional access.

4.9.9 EDC Interaction with Emulation

Parity data is calculated for emulation writes to the L2 memory in the same manner as for CPU writes. Emulation reads are detected/corrected, but may not generate an exception in case of an ECC error.

4.9.10 Memory Scrubbing Techniques

Even though single-bit errors are corrected and passed on to the receiving block, the corrected data is not written back to the memory. The result is that single-bit errors can accumulate over time. If multiple such single-bit errors accumulate in the same line of data, they manifest as two or more erroneous bits. These errors are not recoverable. To prevent correctable single-bit errors escalating into un-correctable and un-recoverable multi-bit errors, UMC implement memory and ECC syndrome scrubbing.

This is accomplished by periodically reading data out of the memories, correcting any single-bit errors, and writing them back to the memory. In previous generations of CorePac, this was accomplished by IDMAs. CorePac has dedicated hardware that implements this.

Copyright © 2020, Texas Instruments Incorporated

Each read-modify-write of a location by the scrubbing engine must be atomic; that is, when the scrubbing engine wins arbitration for the bank, it needs uninterrupted access for the duration of the read and write back of a location. Thus, the accesses by the scrubbing engine are accorded the highest priority by the bank arbiter. A fully pipelined scrub burst sequence contains 4 reads followed by 4 writes. This locks out each bank for 8 to 10 cycles, but results in better use of the bandwidth available at the banks.

The scrub behavior, including the scrub rate, is controlled by the ECR registers, as described in Table 66.

4.9.10.1 Scrubbing Error Logging and Statistics Collection

Scrubbing error logging and detection conforms to the Keystone 3 safety architecture.

4.9.11 L2 Error Scrubbing Command Register (L2CFG)

The L2 Error Detection Command register (L2EDCFG) lets the user suspend the error detection and parity generation logic, and enable or disable auto-scrub. Only a secure supervisor may write this register on secure devices. Table 66 describes the control bits.

4.9.12 Parity RAM Initialization at Reset

The UMC EDC hardware invalidates all the parity and data RAM entries at reset. During this initialization cycle, all transactions are stalled.

4.10 L2 Protection

L2 and external memory protection is different in the C7x as compared to previous generations. Protection is offered as part of virtual memory support. The page table entry in the MMU indicates the protection for any given page, and the CPU and PMC do the permission checks. The UMC does have responsibility for other checks:

- Address error
- Config registers (ECR) permissions
- L2 SRAM firewall
- L2 Config registers (ECR) firewall

4.10.1 Address Error

All CPU or MSMC perform access checks for L2 SRAM accesses at a large granularity (1MB or larger). This means that UMC would receive accesses on any of these interfaces for accesses that are outside the range of implemented L2 SRAM. In this case, UMC detects this error, and sends back a status with the read data.

4.10.2 Config Registers (ECR)

A finer granular check is performed on the ECR registers with a similar error handling mechanism. This is based on the access permissions, which are unique for each register, and have been listed in Table 66.

4.10.3 L2 Firewall

The UMC instantiates the Keystone 3 Firewall IP. The behavior of the firewall is documented in the KS3_FW_IP specification document. The UMC instantiates two firewall IPs:

- MDMA firewall: MDMA transactions mastered by UMC are checked by this firewall. Read responses, snoop responses, L1D victims, and L2 victims bypass this firewall.
- L2 SRAM firewall: There is one instance for each L2 SRAM bank. This is a list of all transactions checked by this firewall:
  – All L1P, MMU, SE accesses to L2 SRAM
  – All L1D read and writes to L2 SRAM, but not L1D victims
  – All MSMC DMA reads and writes to L2 SRAM

For cacheable lines, the firewall has limited checks on priv, debug. The firewall IP spec has details on this.

Copyright © 2020, Texas Instruments Incorporated

4.11 Events – Exceptions and Interrupts

All exceptions and interrupts are referred to as events in this document. There are two types of events that UMC memory system takes: synchronous and asynchronous.

4.11.1 Synchronous Events

These accompany the read or write response to the requestor, and are usually signalled through the read response status (dstatus) or the write response status (wstatus). There is no separate event pin for these events. The requestor is responsible for responding to these events. The UMC does not log any information regarding these events after it has responded with the status.

Table 63 lists all such events. On interfaces where no read or write status bus exists, this event is signalled through the asynchronous event mechanism.

NOTE: DMC / CPU do not record write status, and thus, UMC may not drive the write status up to DMC. This is a micro-architectural decision. For writes that terminate in L2, UMC takes the exception and log the error. For writes terminating in MSMC, MSMC does the error logging.

Table 63. Synchronous Events

Category	Name	Details	Read / Write Status
Address Error	CPU Store Address	CPU store address was outside the valid L2 address space. CPU store was mapped to a region allocated for L2 cache (SRAM under cache)	UMC drives the appropriate error code on the sstatus field of the write response to back to the requestor.
Address Error	CPU Read Address Error	CPU read address was outside the valid L2 address space CPU read was mapped to a regioErrorn allocated for L2 cache (SRAM under cache)	UMC drives the appropriate error code on the rstatus field of the read response back to the requestor.
Protection Error	ECR Store Access Error	ECR Write Access with insufficient privileges ECR Write to a Write Only register ot register field	UMC drives the appropriate error code on the sstatus field of the write response back to the requestor.
Protection Error	ECR Load Access Error	ECR Read Access with insufficient privileges ECR Read to a Read Only register or register field	UMC drives the appropriate error code on the rstatus field of the read response back to the requestor.
Firewall Error	CPU Store Firewall Error	CPU store failed L2SRAM firewall check	UMC drives the appropriate error code on the sstatus field of the write response to back to the requestor.
Address Error	DMA Store Address Error	DMA Store address was outside the valid L2 address space DMA Store was mapped to a region allocated for L2 cache (SRAM under cache)	UMC drives the appropriate error code on the sstatus field of the write response to back to the requestor.
Address Error	DMA Read Address Error	DMA Read address was outside the valid L2 address space DMA Read was mapped to a region allocated for L2 cache (SRAM under cache)	UMC drives the appropriate error code on the rstatus field of the read response back to the requestor.
Firewall Error	DMA Write to L2SRAM Firewall Error	CPU Store failed L2SRAM Firewall check	UMC will send a bad Write Status (sstatus) to MSMC / Requesting Core
ECC	Non-Correctable ECC Error for CPU Fetch / SE Load	Non-Correctable Data Error encountered on ECC syndrome conversion for data to PMC, SE.	UMC will indicate Data Error on the Read Status (dstatus) to the requestor

Copyright © 2020, Texas Instruments Incorporated

4.11.2 Asynchronous Events

These refer to the events that do not have an accompanying read or write response. These are signalled through event pins. Table 64 lists all such events. The UMC signals five types of asynchronous events, as shown in Table 64. Events from the three controllers are combined into a single set of events. Each event is independently controlled.

Table 64. Asynchronous Events

Category	Name	Details	Severity / User Response
UMC_ADDERR_ EV	Addressing Error	DMA / CPU Store address was outside the valid L2 space DMA / CPU Store was mapped to a region allocated for cache	Transaction is dropped, and may point to software errors
UMC_CACHEERR_EV	Cache Allocation Error	Write Allocate was dropped due to a bad read status on the read response Allocate response came with an error for transactions that do not send a response back to the requestor (example: pre-warm)	This could point to either a software / configuration error or an error at the endpoint memory / IP.
UMC_FWFAIL_EV (1:0)	Firewall Error	CPU / DMC Store failed L2SRAM firewall check	This could point to a software configuration error or an attempt at circumventing security.

NOTE: The firewall error logging is part of the firewall IP and described in the Firewall IP Spec.

After signalling an event, each of the memory controllers logs the event in the ECR registers. The core is then required to read through all the ECR registers related to that particular event to acquire details of the event. After processing the event, the core is also required to clear the fault by writing to the ECR Clear register.

A number of ECR registers are provided to record details of the error, and for the core to clear the fault. Table 65 lists the various ECR registers provided.

Table 65. Event Logging and Processing Registers

Category	Name	Details
UMC_ADDRERR_EV	UMC_ADDRERR_EAR	Addressing Error Event Address Register
	UMC_ADDRERR_ESR	Addressing Error Event Status Register
	UMC_ADDRERR_ECR	Addressing Error Event Clear Register
UMC_ALLOCERR_EV	UMC_ALLOCERR_EAR	Cache Allocation Error Event Address Register
	UMC_ALLOCERR_ESR	Cache Allocation Error Event Status Register
	UMC_ALLOCERR_ECR	Cache Allocation Error Event Clear Register
UMC_FWFAIL_EV	Described in the Firewall IP Spec	

4.12 Bandwidth Management (L2 BW)

Basic bandwidth management controls are provided to the user. Table 66 has details on this register. These include the following:

- SE_BW_CNT: This provides control over the relative priority between the two SE ports. This field defines the number of cycles that each SE port wins in arbitration, before the priority of the other SE port is escalated to make it more probable to win arbitration. The SE ports map to the ports from SE to UMC, and not to the actual stream.
- DMA_BW_CNT: This provides control over how many arbitration cycles a DMA command can lose before its priority is escalated to make it more probable to win arbitration.
- CPU_BW_CNT: This provides control over how many arbitration cycles a CPU command can lose before its priority is escalated to make it more probable to win arbitration.
- CPRI_HI: Elevated priority for mastered transactions: this provides control over the priority driven on

Copyright © 2020, Texas Instruments Incorporated

the epriority (elevated priority) VBUSM field out of the MDMA port.

- CPRI_LO: Default priority for mastered transactions: this provides control over the priority driven on VBUSM cpriority field out of the MDMA port.

4.13 Memory-Mapped ECR Registers

Table 66 includes details of all implemented memory-mapped ECR registers.

Table 66. UMC ECR Register Details

REGID	REGISTER	Register Fields	MSB	LSB	Field Access	Field Reset	Register Description
0x280	L2CFG	RSVD	63	5	R	0	
		M3_CACHE	4	4	RW	0	Enable L2 caching of M3 SRAM via ccinner
		RSVD	3	3	R	0	
		L2MODE	2	0	RW	0	L2 Cache size
0x281	L2CC	RSVD	63	54	R	0	
		SE_BW_CNT	53	51	RW	010	SE0,SE1 relative BW management Counter
		RSVD	50	49	R	0	
		DMA_BW_CNT	48	46	RW	011	DMA Bandwidth Management Counter
		RSVD	45	44	R	0	
		CPU_BW_CNT	43	41	RW	011	CPU (Load/Store) Bandwidth Management Counter
		RSVD	40	38	R	0	
		CPRI_HI	37	35	RW	001	Elevated priority for mastered commands
		CPRI_LO	34	32	RW	010	Default priority for mastered commands
		RSVD	31	4	R	0	
		SETSKEW	3	3	RW	0	Set / Bank skewing enable
		RSVD	2	0	R	0	
0x282	L2WB	RSVD	63	1	R	0	
		WB	0	0	RW	0	Set to start the writeback operation. Will get cleared when the writeback is complete
0x283	L2WBINV	RSVD	63	1	R	0	
		WBINV	0	0	RW	0	Set to start the writeback-inv operation. Will get cleared when the writeback-inv is complete
0x284	L2INV	RSVD	63	1	R	0	
		INV	0	0	RW	0	Set to start the invalidate operation. Will get cleared when the invalidate is complete
0x286	L2CTAG	RSVD	63	52	R	0	
		MESI	51	50	R	0	MESI Bits includes valid and dirty bits
		SECURE	49	49	R	0	Secure bit
		TAG	48	12	R	0	L2 Tag
		RSVD	11	0	R	0	
0x287	L2EDCFG	SCDELAY	63	32	RW	h8000_0000	Delay between full scrub cycles
		BTDELAY	31	16	RW	h400	Delay between scrub bursts
		RSVD	15	1	R	0	
		SCEN	0	0	RW	1	Enable L2 Scrubbing
0x291	L2ADDREEA	RSVD	63	44	R	0	
		EADDR	43	0	R	0	Transaction address for which error was detected

Copyright © 2020, Texas Instruments Incorporated

Table 66. UMC ECR Register Details (continued)

REGID	REGISTER	Register Fields	MSB	LSB	Field Access	Field Reset	Register Description
0x292	L2ADDREES	RSVD	63	8	R	0	
		RQSTR	7	5	R	0	Requestor of the transaction for which error was detected
		ERROR_TYPE	4	3	R	0	Type of Error
		BANK	2	1	R	0	Bank in which error was detected [0,1,2,3]
		ERROR_FLAG	0	0	R	0	Indicates that an error has occurred and has not been cleared yet. Clear occurs on a write to L2ADDREER
0x293	L2ADDREER	RSVD	63	1	R	0	
		ERROR_RESET	0	0	W		
0x294	L2ALLOCEEA	RSVD	63	44	R	0	
		EADDR	43	0	R	0	Transaction address for which error was detected
0x295	L2ALLOCEES	RSVD	63	8	R	0	
		RQSTR	7	5	R	0	Requestor of the transaction for which error was detected
		ERROR_TYPE	4	3	R	0	Type of Error
		BANK	2	1	R	0	Bank in which error was detected [0,1,2,3]
		ERROR_FLAG	0	0	R	0	Indicates that an error has occurred and has not been cleared yet. Clear occurs on a write to L2ALLOCEER
0x296	L2ALLOCEER	RSVD	63	1	R	0	
		ERROR_RESET	0	0	W		
0x297	L2MEMMAP	RSVD	63	5	R	0	
		LCL_SDMA_ALIAS	4	4	RW	0	Local SDMA View - Full Memory Map or Aliased
		RSVD	3	1	R	0	
		CPU_ALIAS	0	0	RW	0	CPU View - Full Memory Map or Aliased
0x298	L2MSWCTL	RSVD	63	17	R	0	
		WBUF	16	16	RW	0	WBUF Buffer Ownership
		RSVD	15	13	R	0	
		IBUFHB	12	12	RW	0	IBUFHB Buffer Ownership
		RSVD	11	9	R	0	
		IBUFLB	8	8	RW	0	IBUFLB Buffer Ownership
		RSVD	7	5	R	0	
		IBUFHA	4	4	RW	0	IBUFHA Buffer Ownership
		RSVD	3	1	R	0	
		IBUFLA	0	0	RW	0	IBUFLA Buffer Ownership
0x299	L2MEMMAPIBUFA	IBUFHA_BASE	63	47	RW	0	IBUFHA Base Address
		RSVD	46	32	R	0	
		IBUFLA_BASE	31	15	RW	0	IBUFLA Base Address
		RSVD	14	0	R	0	
0x29A	L2MEMMAPIBUFB	IBUFHB_BASE	63	47	RW	0	IBUFHB Base Address
		RSVD	46	32	R	0	
		IBUFLB_BASE	31	15	RW	0	IBUFLB Base Address
		RSVD	14	0	R	0	
0x29B	L2MEMMAPWBUF	RSVD	63	32	R	0	
		WBUF_BASE	31	15	RW	0	WBUF Base Address
		RSVD	14	0	R	0	

Copyright © 2020, Texas Instruments Incorporated

Table 66. UMC ECR Register Details (continued)

REGID	REGISTER	Register Fields	MSB	LSB	Field Access	Field Reset	Register Description
0x29C	L2MSWERRSTAT	RSVD	63	8	R	0	
		ST_WBUF_ERR	7	7	R	0	Store to WBUF is outside address range
		LD_WBUF_ERR	6	6	R	0	Load to WBUF is outside address range
		ST_IBUF_ERR	5	5	R	0	Store to IBUF is outside address range
		LD_IBUF_ERR	4	4	R	0	Load to IBUF is outside address range
		RSVD	3	3	R	0	
		DMA_ERR	2	2	R	0	DMA initiated buffer ownership error
		CPU_ERR	1	1	R	0	CPU initiated buffer ownership error
		RSVD	0	0	R	0	
0x29D	L2MSWERRCLR	RSVD	63	8	R	0	
		ST_WBUF_ERR	7	7	W		Clear ST_WBUF_ERR
		LD_WBUF_ERR	6	6	W		Clear LD_WBUF_ERR
		ST_IBUF_ERR	5	5	W		Clear ST_IBUF_ERR
		LD_IBUF_ERR	4	4	W		Clear LOD_IBUF_ERR
		RSVD	3	3	R	0	
		DMA_ERR	2	2	W		Clear DMA_ERR
		CPU_ERR	1	1	W		Clear CPU_ERR
		RSVD	0	0	R	0	
0x29E	L2MSWERRADDR	RSVD	63	44	R	0	
		ADDR	43	0	R	0	Error address
0x29F	L2EDTST	SCADDR	63	48	RW	0	Starting RAM physical address for test mode
		SCCOUNT	47	32	RW	0	Number of scrubs to perform in test mode
		RSVD	31	1	R	0	
		SCTST	0	0	RW	0	Enable L2 scrub test mode

4.14 L2 Address Aliasing

This feature is described in detail in the Caches Introduction Architectural Specification Document, 1.1.10 ‘‘Aliased Address Mode’’.

4.15 Trace and Events

Details of the events are in the Caches Introduction Architectural Specification Document, 2.4.16 ‘‘Trace and Events’’.

4.16 Cache Tag View

Details of the cache tag view feature are in the Caches Introduction Architectural Specification Document, 2.4.17 ‘‘Cache Tag View’’.

4.17 Powerdown

UMC supports powerdown through two sets of features. Details of the powerdown functionality are included in the Cluster Powerdown Spec.

- pwr_req input to UMC prevents it from mastering any new MDMA transactions for MSMC initiated cache prewarm transactions.
- umc_idle, which indicates that all pending MDMA transactions have completed.

Copyright © 2020, Texas Instruments Incorporated

5 Corepac Memory Management Unit (CMMU)

5.1 Corepac MMU Introduction

The CorePac Memory Management Unit (CMMU) extends the C7x architecture with support for address translation, access permission and protections, and memory attributes determination and checking. It is implemented per C7x cluster as a two-level TLB structure. The CMMU works with the C7x L1 caches, streaming engines in each processor, and the CorePac memory system of the CorePac cluster to translate virtual addresses to physical addresses through management of the table walk hardware that accesses translation tables in main memory. The CMMU enables fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes held in the L1 level micro translation look-aside buffer, (uTLB) and CorePac cluster level translation look-aside buffers (TLBs).

5.1.1 CorePac MMU Usage Model Enhancement

5.1.1.1 Substituting of MPAX

Memory protection and extension unit (MPAX) is implemented in the C6x DSP XMC module. It can specify memory protection and address translation for large regions of memory in a small number of entries. The full register-based implementation adds little translation overhead (1 CPU clock cycle). However:

- The MPAX model is difficult to configure for complex memory maps. It requires excessive effort to map many CPUs into a common physical address space.
- In C6x, XMC Virtual-to-physical translation happens after L1 and L2 cache accesses. This makes changing MPAX registers expensive, because they require cache flushes and they don't support cache coherence properly.
- MPAX is a non-standard use model. Linux does not natively support MPAX. The MPAX is primarily set it up to statically partition the address space between ARM and DSP.

5.1.1.2 Substituting of MAR Bits

In C6x, a set of Memory Attribute Registers (MAR) are used to assign and control transaction types for memory accesses. They provide fixed lookup times due to the register-based implementation, but have the following disadvantages compared with traditional MMU:

- Very large: Over 700 discrete register bits for basic set of MARs
- Very coarse grain: 16-MB increments. This leads to using aliasing tricks with MPAX to get fine grain control over cacheability, prefetchability, and writethrough versus writeback in some memories, such as MSMC.
- Expensive lookup in L1D critical path: Must mux 256 MAR values down to a single set of cacheable, prefetchable, writethrough bits quickly.
- MAR model is also non-standard usage.

5.1.1.3 CorePac MMU Scheme

In the C7x processor, the CorePac MMU (CMMU) provides a traditional MMU solution. However, compared with register-based lookups provided by MPAX and MAR, a traditional MMU has variable translation time due to TLB misses and translation table walks.

To address real-time applications and use-models, CorePac MMU has a software-managed mode to ensure fixed or minimum address lookup time. This mode enables the software to directly populate entries into the uTLBs.

5.2 CorePac MMU Feature List

The C7x CorePac MMU (CMMU) features include the following:

- Supports AArch32 LPAE and AArch64 virtual address architecture and page table format
  - Up to 49 bits of virtual address space for each table base
  - Programmable levels of page table translation

Copyright © 2020, Texas Instruments Incorporated

  - 4KB, 16KB, and 64KB translation granules
- Supports multiple page sizes: 4KB, 16KB, 64KB, 2MB, 32MB, 512MB, 1GB, and 16GB. Large page sizes enable flatter translation table walk and reduced translation overhead, particularly for 2-stage translations.
- Both hardware- and software-managed page table walks
  - Software-managed mode after hard reset and during boot. Flexible device initialization and memory regions available for boot.
  - 1-stage hardware table walk in hardware-managed mode
    - Secure privilege level translation
    - Non-secure root privilege level translation
  - 2-stage hardware table walk in hardware-managed mode
    - Virtual to intermediate physical address and intermediate physical to physical address translations
    - Non-secure guest privilege level translation
- Hierarchical table look-aside buffer (TLB) caching Stage 1 and Stage 2 page entries, intermediate page walk pointers, for optimal address translation performance
- uTLBs, the first level TLB
  - 16-entry fully associative dual interface L1 Data uTLB per C7x core, 0 latency overhead translation, one interface per C7x core functional D-unit, with page size support for 4KB, 64KB, 2MB, 32MB, 512MB, 1GB, 16GB
  - 8-entry fully associative L1 Instruction uTLB per C7x core, 0 latency overhead translation
  - 2 separate 8-entry fully associative uTLBs per dual-threaded stream engine, one per thread
  - Supports both pseudo-random and FIFO replacement policies
  - Supports entry invalidation by secure, root, ASID, VMID, and/or virtual address
  - All uTLB entries are accessible and manageable by software in software mode
  - Trigger CPU exception when permission fault detected during address translation
  - TLB conflict detection support
- CorePac TLB, level 2 TLB cache at C7x CorePac cluster level
  - 512-entry 4-way associative TLB cache per C7x core, caching both instruction and data page translations with virtual address to physical address mapping
  - Intermediate table walk caches to reduce memory access during multi-level page walk
  - Allow caching and sharing of intermediate table walks and page tables in system cache hierarchy such as CorePac L2 caches and MSMC L3 cache
  - Separated large page table entry buffers, accessed in parallel with main small page cache
  - Up to 4 outstanding page table walk (PTW) requests
  - TLB entries are tagged with a 8-bit address space identifier (ASID) or global bit and guest entries are further tagged with a 8-bit virtual machine identifier (VMID), allowing caching of multiple address spaces without TLB flush during context switches between applications and/or OSes.
  - Supports software-driven TLB invalidation by secure, root, ASID, VMID, and/or virtual address or intermediate physical address
  - Parity protection and detection per entry, detection results in invalidation and reallocation of the error-ed page table entry or pointer.
  - Page hazarding to improvement cache efficiency
- Contextual banking of MMU configuration registers
- Restartable-interrupt support during page table entry allocation
- Trigger page fault exception for page faults encountered during the page table walk, logging fault status
- Page table entry field includes memory attributes, access permission, security attributes, ASID, VMID, page sizes, shareability attribute, global bit, VA, PA

Copyright © 2020, Texas Instruments Incorporated

- Security extensions that facilitate the development of secure applications
- Soft-error protection on all RAM and data storage structures
- Debug support for all uTLB and TLB data entries through configuration register accesses
- C7x core triggered on-demand address translation service
- Performance event monitor for various MMU events

The following features are not supported:

- Support receiving distributed virtual memory transactions (DVMs) from ARM CPU and complete requested actions:
  - TLB invalidation based on requested attributes
  - Synchronization and completion messages
- UDMA and external CPU triggered page table descriptor pre-warming

5.3 Cache Architecture

The CMMU TLB structure is composed of multiple levels of hierarchy ranging from multiple Level 1 - uTLBs, to several page caches that make up the Level 2 TLBs for both Stage 1 and Stage 2 translations.

5.3.1 CMMU uTLB Caches

- 16 entry Load/Store data uTLB
- 8 entry Instruction uTLB
- 2x 8 entry streaming engine uTLBs
- All uTLBs are fully associative caches
- Page sizes supported: 4K, 16K, 64K, 2MB, 32MB, 512MB, 1GB, 16GB
- Translation types cached:
  - VA->PA translations (when Stage 1 translation is enabled)
  - IPA->PA translations (when Guest Stage 1 translation is disabled and Stage 2 translation is enabled)

5.3.2 CMMU TLB Small Page Cache

- 512 entries
- 128 set associative 4-way cache
- Caches level 3 page VA-to-PA translations

5.3.3 CMMU TLB Large Page/Walking Caches

- Shared Stage 1 page and walk caches:
  - 16 entry MB cache: 2MB, 32MB, and 512MB. Blocks and pointers (page granule dependent)
  - 8 entry GB cache: 1GB, 16GB, and 64GB. Blocks and pointers (page granule dependent)
  - 4 entry PTR cache: 512GB, 4TB, 128TB. Pointers only.
  - All large page/walk caches are fully associative.
  - Cache VA-to-PA translations
- Shared Stage 2 page and walk caches:
  - 8 entry MB caches: 2MB, 32MB, and 512MB. Blocks and pointers (page granule dependent)
  - 4 entry GB caches: 1GB, 16GB, and 64GB. Blocks and pointers (page granule dependent)
  - All large page/walk caches are fully associative. –
  - Cache IPA-to-PA translations

5.4 References

This document contains references to the following documents:

Copyright © 2020, Texas Instruments Incorporated

- ARMv8 Architectural Reference Manual - The AArch64 Virtual Memory System Architecture

The CMMU chapter expects that the reader is familiar with the ARMv8 AArch64 Virtual Memory System Architecture.

5.5 Functional Description

5.5.1 C7x Address Translation System

This section describes the C7x virtual memory architecture that applies to C7x processor core executions. This is compliant to ARM VMSAv8-64, as defined in the ARMv8 Architecture Reference Manual.

Each C7x core provides one central memory management unit (CMMU), which controls address translation, access permissions, and memory attribute determination and checking, for memory accesses made by the CPU.

The process of address translation maps the virtual addresses (VAs) used by C7x onto the physical addresses (PAs) of the physical memory system. These translations are defined independently for different exception levels and security states, as shown in Table 67.

Table 67. ARM Exception Levels and C7x Privilege Levels

ARM Exception Level	C7x with Hypervisor	C7x without Hypervisor
EL3	-	-
EL2	Non-Secure Supervisor	-
EL1 Secure	Secure Supervisor	Secure Supervisor
EL0 Secure	Secure User	Secure User
EL1 Non-Secure	Guest Supervisor	Non-Secure Supervisor
EL0 Non-Secure	Guest User	Non-Secure User

5.5.2 Supported Memory Management Modes

CMMU supports two distinct memory management modes: software managed and hardware managed. The two modes are targeted for different use models. The software-managed mode enables RTOS to directly manage large pages in the uTLB for fast deterministic response times, without the need to program an MMU. The hardware-managed mode enables conventional general purpose OSes to setup a typical hardware-based MMU and program page tables for fine-grain control of memory/pages and their attributes.

5.5.2.1 Software-Managed Mode

CMMU begins in software-managed mode when coming out of reset, as denoted by the reset value of the register field SCR.HW = 0.

This mode allows the uTLB entries to be directly accessed by software through reading and writing to an External Configuration register (ECR). Each uTLB in the system has an index-based ECR address for accessing the Match and Attribute registers associated with each uTLB entry.

5.5.2.1.1 Writing and Reading the uTLB Entries

Each uTLB entry is made up of two External Config registers (ECR). The Match register contains fields used for matching when the uTLB receives a translation request including Input address, page size, context identifiers, and so forth. The Attribute register contains attributes and details for the translated page once a match has been found, including output address, memory attributes, and so forth.

The uTLB entry's Match and Attribute registers can be accessed by the software by reading and writing to CMMU index-based ECRs. Each uTLB has associated ECR address to represent its specific Match and Attribute register groups. A particular uTLB entry is targeted with the ECR index.

Software accesses should be managed by the supervisor and can only be done in software-managed mode. However, software emulation debug reads are still allowed during hardware-managed mode.

Copyright © 2020, Texas Instruments Incorporated

An entry is considered valid only when both the Match and Attribute register's valid bits are set to 1. For example, a translation request would not match on an entry that only set up the Match register but not the Attribute Register. This would miss and result in a translation fault.

The software reads to the uTLB Match and Attribute registers and returns all 0s when their associated valid bits are 0. Entries can be "reserved" by the software by setting the Match register valid bit and the proper secure/priv value.

The software can also set up and view entries for lower level contexts (Secure->Non-Secure->Non-Secure Guest) but lower level contexts cannot view or create higher privilege entries. Non-secure read access to secure entries return all 0s. This also applies to non-secure emulation debug accesses of secure entries.

5.5.2.1.2 uTLB Entries

5.5.2.1.2.1 uTLB Match Register

The Match register contains fields used for matching when the uTLB receives a translation request including input address, page size, context identifiers, and so forth. See Section 5.5.11.35 for the register format. The tables below provide the encoding and definition of the register fields.

Table 68. UTLB_MATCH.VALID – Valid Entry

Encoding	Definition
0	Match Entry Invalid
1	Match Entry Valid

Table 69. UTLB_MATCH.SECURE – Secure Match

Encoding	Definition
0	Non-Secure Entry
1	Secure Entry

Table 70. UTLB_MATCH.ROOT - Root Match

Encoding	Definition
0	Non-Root Entry
1	Root Entry

Table 71. UTLB_MATCH.GBL - Global Page

Encoding	Definition
0	Non-Global Entry (Check ASID)
1	Global Entry (Ignore ASID)

Table 72. UTLB_MATCH.VMID - VMID Match

Encoding	Definition
8-bit VMID	Virtual Machine ID, used for Guest Matching (Non-Secure, Non-Root)

Table 73. UTLB_MATCH.ASID - ASID Match

Encoding	Definition
8-bit ASID	Address Space ID, used for non-global entry matching

Copyright © 2020, Texas Instruments Incorporated

Table 74. UTLB_MATCH.PG_SIZE - Page Size

Encoding	Definition
000	4 KB Page
001	16 KB Page
010	64 KB Page
011	2 MB Page
100	32 MB Page
101	512 MB Page
110	1 GB Page
111	16 GB Page

Table 75. UTLB_MATCH.IADDR - Input Address

Encoding	Definition
37-bit Input Address	Input Address[48:12] used for Match

NOTE: While in software-managed mode, TI recommends that only software write 0s to the ASID/VMID match fields, and use only global pages. Writing non-zero values to the ASID/VMID field and setting uTLB entries/pages to non-global will have undefined results.

5.5.2.1.2.2 uTLB Attribute Register

The Attribute register contains attributes and details for the translated page when a match has been found, including output address, memory attributes, and so forth.

See Section 5.5.11.36 for the register format. The tables below provide the encoding and definition of the register fields.

Table 76. UTLB_ATTR.VALID – Valid Entry

Encoding	Definition
0	Attribute Entry Invalid
1	Attribute Entry Valid

Table 77. UTLB_ATTR.STATUS - Entry Status

Encoding	Definition
12-bit Status Encoding	Error Status of the Entry

NOTE: The software writes to the Attribute register of a uTLB l always result in *_UTLB_ATTR.STATUS = 0 (No Fault). Software cannot write a faulted entry to the uTLB. See Section 5.5.6.

Table 78. UTLB_ATTR.ACC_PERM - Access Permissions

Bit Position	Definition
acc_perm[0]	Stage 1, enable user access
acc_perm[1]	Stage 1, disable write access
acc_perm[2]	Stage 1, user never execute
acc_perm[3]	Stage 1, supervisor never execute
acc_perm[4]	Stage 1, permissions valid
acc_perm[5]	Stage 2, read-only, writes not permitted regardless of Stage 1 permissions

Copyright © 2020, Texas Instruments Incorporated

Table 78. UTLB_ATTR.ACC_PERM - Access Permissions (continued)

Bit Position	Definition
acc_perm[6]	Stage 2, access bit[0]
acc_perm[7]	Stage 2, access bit[1]
acc_perm[8]	Stage 2, permissions valid

Notes:

- Additional Stage 1 execution check – supervisor cannot execute when the user has write permissions to the uTLB entry/page.
- Stage 1 permission checks are only valid when acc_perm[4]=1
- Stage 2 permission checks are only valid when acc_perm[8]=1

Table 79. Stage 1 Read/Write Permissions

acc_perm[1]	acc_perm[0]	User Permission	Supervisor Permission
0	0	None	Read/Write
0	1	Read/Write	Read/Write
1	0	None	Read-Only
1	1	Read-Only	Read-Only

Table 80. Stage 2 Access Permissions

Encoding[1:0]	Definition
00	No access permission
01	Read-only, Writes are not permitted, regardless of Stage 1 permissions
10	Write-only, Reads are not permitted, regardless of Stage 1 permissions
11	Read/Write allowed. Use Stage 1 permissions

NOTE: Stage 2 permissions only apply to non-secure, non-root (guest) transactions.

Table 81. UTLB_ATTR.PG_NSEC - Non-Secure Page

Encoding	Definition
0	Secure Page
1	Non-Secure Page

Notes:

- The non-secure page attribute only applies to pages used by the secure context.
- The secure context is disallowed from executing code on pages marked as non-secure.

Table 82. UTLB_ATTR.MEMTYPE - Memory Type Attributes

Bit Position	Definition
[1:0]	Memory Type [1:0]
[3:2]	Inner Allocation Policy [1:0]
[5:4]	Outer Allocation Policy [1:0]
[7:6]	Shareability [1:0]
[8]	Prefetchable Page

Copyright © 2020, Texas Instruments Incorporated

For the encoding for each field, see Section 5.5.3.16.2.

The 2-bit shareability encoding is based on the encoding used in the page table entry shareability field in the ARMv8 64-bit architecture. See Table 97.

5.5.2.1.3 uTLB Reset State

Each uTLB has a default 16-GB secure mode page stored at the time of reset for use by the secure supervisor. Essentially, this region delivers a translation with input address = output address for the first 16 GB of the memory, which the software can use for initial booting. The software can configure the uTLBs to enable access to memory regions outside this space by writing to the necessary uTLB entries to set up additional translations. Match and attributes of this default entry are defined in Table 83 and Table 84.

The default page uses the device memory type and has full access permissions for the secure supervisor. The default page is entry 0 in each uTLB instance, and can be overwritten by software as needed.

NOTE: In the current architecture, software is not allowed to create a secure-root page (only secure non-root pages).

Table 83. Default Match Entry

Field	Value
Match Entry Valid	1
Secure	1
Root	Ignored for secure context
Global	1
VMID	0
ASID	0
Page Size	16GB
Input Address	0

Table 84. Default Attribute Entry

Field	Value
Attribute Entry Valid	1
Status	0 (Good Status)
Access Permissions	Stage 1 Permissions Enabled Supervisor - Read, Write, Execute User - No Permissions No Stage 2 Permissions
Page Non-Secure	0 (Secure Page)
Page Memory Type	Device Memory No Inner/Outer Allocation Non-Shareable (System Shared) Non-Prefetchable
Output Address	0

5.5.2.1.4 Software-Managed Mode - Programmer Consideration

When modifying the default entry 0 of the instruction uTLB used initially by the software, any changes to the entry must ensure a continuity of code execution before and after changes are made to the entry. The software should always perform a memory fence/barrier operation after making any changes to the uTLBs in software-managed mode to ensure any memory operations using stale translations have completed.

Copyright © 2020, Texas Instruments Incorporated

5.5.2.1.5 Transition from Hardware-Managed Mode to Software-Managed Mode

The only method to transition from hardware-managed mode to software-managed mode is by resetting CMMU. CMMU does not support the transition to software-managed mode through ECR accesses or any means other than reset.

5.5.2.1.6 uTLB External Config Register Error Conditions

Listed in Table 85 are all of the errors that can occur when the software attempts to write or read the External Config registers (ECR) to program the uTLB entries.

Table 85. uTLB ECR Error Conditions

Condition	Error Type
ECR index out of range of uTLB entries	Addressing Error
Writing non-zero values to the reserved fields in the Match and Attribute Entries	Permission Error
User Access	Permission Error
Secure access attempts to create a secure-root page (only secure non-root pages are supported)	Permission Error
Non-Secure access attempts to create or overwrite a secure page	Permission Error
Non-Secure, Guest access attempts to create, overwrite, or read a Non-Secure, Root page	Permission Error
SW writes or Non-Debug SW Reads during HW Managed Mode (SCR.HW=1)	Invalid Operation
SW attempts to create an Unsupported Page size in the Load/Store uTLB (16K page size)	Invalid Operation

A successful software access occurs when none of the above rules have been violated. The CPU typically takes an exception when it encounters an ECR error.

5.5.2.1.7 Errors on uTLB Translation Requests

The encoding for the error statuses can be found on Table 111.

5.5.2.1.7.1 TLB Conflict

A translation request that hits on multiple pages (uTLB entries) due to software creating overlapping pages results in a TLB conflict abort.

5.5.2.1.7.2 Permission Fault

A translation request that does not have valid access permissions as described by the UTLB_MATCH.ACC_PERM field results in a permission fault.

5.5.2.1.7.3 Translation Fault Due to Missing All uTLB Entries

Translation requests that miss the uTLB entirely during software-managed mode result in a Stage 1 translation fault. To ensure that this does not occur, the software must only use the 16-GB default entry which each uTLB has as its reset state, or first write to the uTLB Match and Attribute registers to create valid entries for the memory region.

NOTE: Software cannot create an entry in the uTLB with a bad status. All software writes to the Attribute register result in UTLB_ATTR.STATUS = 0 (good status).

5.5.2.2 Hardware-Managed Mode

CMMU can enter hardware-managed mode, which enables hardware-based memory management, by having the secure supervisor write SCR_SS.HW=1. This should be done when all page table descriptors have been set up in memory, or at a minimum the table descriptors necessary to continue the boot up process.

Copyright © 2020, Texas Instruments Incorporated

NOTE: The CMMU hardware-managed mode is compliant to the table description format as defined by the ARMv8 64-bit architecture, and thus many details for the CMMU can be found in the ARMv8 64-bit Virtual Memory Architecture.

5.5.2.2.1 Translation Lookaside Buffer (TLB)

CMMU contains a separate translation lookaside buffer (TLB) for Stage 1 and Stage 2 translations. Details for the TLBs including structure, allocation/replacement policy, TLB conflict handling, and so forth are given in Section 5.5.8.

5.5.2.2.2 Page Table Walkers

CMMU has 4 separate page table walkers that support the page descriptor format, as defined by the ARMv8 64-bit architecture. The ARM 32-bit LPAE format is also supported because ARMv8 ARCH64 is an extension of the LPAE long descriptor format. CMMU can support up to 4 simultaneous address translation walks.

5.5.2.2.3 Supported Modes of Operation

CMMU supports several modes of operation, depending on the value written to fields in its External Config registers. The following sections describe these modes. Each mode matches a corresponding mode of translation, as defined in the ARMv8 64-bit architecture.

For details on the transition between these modes, see Section 5.5.2.3.

5.5.2.2.3.1 Flat Translation / No Translation Mode

When SCR_SS.HW=1 and SCR.S1_EN=0 (Stage 1 Disabled) and VCR.S2_EN=0 (Stage 2 Disabled, for Guest Context-only), CMMU performs flat translation/no translation. This mode returns the input address back to the output address. Memory attributes for instruction accesses are controlled through CMMU ECRs. See Section 5.5.3.16.1.

5.5.2.2.3.2 Stage 1 Enabled

When SCR.S1_EN=1 (Stage 1 enabled for the current context) and VCR.S2_EN=0 (Stage 2 disabled, for guest context-only), CMMU performs Stage 1 translations for the current context (VA->PA). For details on CMMU ECRs that affect memory attributes in this mode see Section 5.5.3.16.2.

Additional steps and considerations are needed when transitioning from the Stage 1 disabled state to the Stage 1 enabled state. See Section 5.5.2.2.3.6 for details.

5.5.2.2.3.3 Stage 1 and Stage 2 Enabled (Guest Context-Only)

For the guest context, when SCR_GS.S1_EN=1 (Stage 1 enabled for guest) and VCR.S2_EN=1 (Stage 2 enabled), CMMU performs two stage translations for that context – Stage 1 as defined by the guest context (VA->IPA), and Stage 2 as defined by the non-secure root supervisor (Hypervisor) (IPA->PA). For details on CMMU ECRs for this mode or additional details on CMMU ECRs that affect memory attributes in this mode, see Section 5.5.3.16.3.

5.5.2.2.3.4 Stage 2 Enabled (Guest Context-Only)

For guest context, when SCR_GS.S1_EN=0 (Stage 1 disabled for the current context) and VCR.S2_EN=1, CMMU uses the guest default memory attribute values for instruction and data accesses where input address equals output address, and further applies Stage 2 translation to the addresses and memory attributes.

5.5.2.2.3.5 Stage 2 Default Cacheable Support

CMMU supports the default cacheability feature as defined by the ARMv8 64-bit architecture, which includes overriding behaviors of the Stage 1 and Stage 2 translation functionality. See Section 5.5.3.16.4.

Copyright © 2020, Texas Instruments Incorporated

5.5.2.2.3.6 Transition Between Stage 1 Disabled and Stage 1 Enabled

The location and permissions of the program code and data involved with the mode transitions between Stage 1 disabled and Stage 1 enabled must match before and after the mode transition to ensure continuous code execution.

The software is also required to perform a manual context-based TLB invalidation after the transition of a context from Stage 1 disabled to Stage 1 enabled. The TLB invalidation is necessary to remove stale TLB entries used by the hardware to enable the Stage 1 disabled mode functionality. The same TLB invalidation is required when making the transition from Stage 1 enabled to Stage 1 disabled, to remove stale translation entries in the TLBs.

5.5.2.2.4 Hint Bit Support

CMMU supports use of the contiguous Hint bit in the descriptor for all table page granules and page sizes, as defined by the ARMv8 64-bit architecture.

5.5.2.2.5 Access Flag Support

CMMU supports a software-managed access flag. If the page table walker encounters a descriptor with the access flag set, CMMU returns an access flag fault for the particular stage of the table walk.

CMMU does not support hardware-managed access flag updating. Software is expected to manage the updating of the table descriptor access flag bit when an access flag fault is reported.

By default, translations with access flag faults are not cached in the TLBs. See Section 5.5.3.19 for more details.

5.5.2.2.6 Sharing Table Entries Between ARM and C7x

When sharing page table entries for data between ARM and C7x, the software must ensure that ARM's MAIR and CMMU MAR registers have equivalent attributes for the particular memory attribute indices used by the shared page table entries.

5.5.2.2.7 Additional External Config Register Error Conditions

Listed in Table 86 are the additional errors that can occur for External Config register accesses to the CMMU ECRs. The errors relating to uTLB ECR accesses can be found in Section 5.5.2.1.6.

Table 86. Additional ECR Errors for CMMU ECRs

Condition	Error Type
Invalid ECR address space access	Address Error
User access	Permission Error
Non-Secure Guest access to context specific “GS/S” ECRs and Stage 2 Virtualization ECRs	Permission Error
Non-Secure Supervisor access to context specific “S” ECRs	Permission Error
Non-Secure Guest accessing or creating TDRR ECR results with Stage 2 Translation details	Permission Error
Writing non-zero values to reserved fields on TBR/VTBR ECRs	Permission Error
Non-Debug access to Debug ECRs: reading the Debug Access ECR (TLB_DBG) writing to the Debug Data ECRs (TLB_DBG_DATA)	Permission Error
Enabling HW Mode (SCR.HW =1) from a non-secure context	Invalid Operation
Enabling Stage 2 (VCR.S2_EN=1) while in SW mode	Invalid Operation
TLB Invalidation (TLB_INV) accesses: ECR read ECR write during SW mode Unprivileged access to certain Invalidation Type IPA Invalidation beyond 40-bits	Invalid Operation
Translation Debug Address (TDAR) ECRs accesses: ECR read ECR write during SW mode	Invalid Operation

Copyright © 2020, Texas Instruments Incorporated

5.5.2.3 Usage and Transition Between Modes

5.5.2.3.1 Reset State - Software-Managed Mode

CMMU begins in software mode upon coming out of reset. See Section 5.5.2.1 for more details.

5.5.2.3.2 Transition to Hardware-Managed Mode

CMMU can enter hardware-managed mode, which enables hardware-based virtualization, by having the secure supervisor write SCR_SS.HW=1. This should be done when all page table descriptors have been set up in memory, or at a minimum the table descriptors necessary to continue the boot up process. See Section 5.5.2.2 for more details.

The transition to hardware-managed mode triggers an internal hardware TLB invalidation of all translation entries created during software-managed mode.

NOTE: The location and permissions of the program code and data involved with the mode transition from software-managed mode to hardware-managed mode must match before and after the mode transition to ensure continuous code execution. Software can directly transition from software-managed mode to hardware-managed mode with either Stage 1 translation enabled or disabled.

5.5.2.3.3 Setup Registers

Additional CMMU ECRs require setup to properly control CMMU functionality during hardware-managed mode. See Section 5.5.2.2.3 and Section 5.5.3.

5.5.2.3.4 Transition from Hardware-Managed Mode to Software-Managed Mode

The only way to transition to software-managed mode from hardware-managed mode is to reset CMMU. CMMU does not support any other means of re-entering software-managed mode.

5.5.2.3.5 Privilege Considerations

CMMU does not support a mixture of software-managed mode and hardware-managed mode between the various privilege levels. The transition into hardware-managed mode can only be done by the secure supervisor. Attempts by other privilege levels to initiate the mode transition through writing to the SCR.HW bit result in an exception. All privilege levels must make the transition into hardware-managed mode concurrently.

5.5.2.3.6 Pinned Pages

CMMU does not support pinned (locked) pages in its TLBs.

Pinned (locked) pages are intended to remain in a TLB until it is flushed out by a TLB maintenance operation. These pages are not affected by normal TLB cache entry replacement, and therefore provide a performance benefit by allowing frequently-used translations to not be thrashed out of the TLB.

5.5.3 Page Translation

5.5.3.1 Supported Features

The CMMU supports reading translations tables of the ARMv8 Arch64 descriptor format.

The supported page table granules sizes are 4 KB, 16 KB, and 64 KB.

The following addressing modes are supported in the hardware-managed address translation mode:

- Flat addressing (Stage 1)
- Flat addressing (Stage 1) + Stage 2 (guest context)
- Stage 1

Copyright © 2020, Texas Instruments Incorporated

- Stage 1 + Stage 2 (guest context)
- Stage 2 (guest context)

Each Stage 1 context consists of two 48-bit address space regions. The 49th bit of the virtual address is used to select between the two regions, Region 0 and Region 1, each with its own translation tables. Typically, one region is dedicated to the supervisor privilege level and the other region to the user privilege level.

The CMMU supports all Stage 1 table pointer hierarchical attributes: NSTable, APTable, UXNTable, and PXNTable.

All fault types in the ARMv8 Arch64 virtual memory system architecture are supported, with the exception of the alignment fault type. See Section 5.5.6 for details on the fault reporting.

The CMMU supports the merging of shareability, cacheability, and memory type page attributes between Stage 1 and Stage 2.

5.5.3.2 Control Registers

The CMMU has several registers that provide control over all aspects of address translation, including:

- Translation management mode – software versus hardware mode
- Translation stage control, Stage 1 and Stage 2
- Translation table control
- Table memory attributes control

The following contextually banked control registers manage Stage 1 translations:

- System Control Register (SCR): Section 5.5.11.10
- Table Control Register (TCR): Section 5.5.11.11 and Section 5.5.11.12
- Table Base Register (TBR): Section 5.5.11.13 and Section 5.5.11.14
- Memory Attribute Register (MAR): Section 5.5.11.15

Aliases are provided to allow higher level contexts to access lower level banked registers, and for the secure context to access the non-secure context banked registers. See the *_GS and *_S variation of the Stage 1 translation registers in the register list Table 128. Examples are: Section 5.5.11.17 and Section 5.5.11.24.

The translation table registers TCR/TBR each have 2 set of registers, one for each 48-bit address space region. The 49th bit of the virtual address is used to select which translation table registers are used for the Stage 1 address translation.

- VA 49th bit = 0, use Region 0 - TCR0/TBR0
- VA 49th bit = 1, use Region 1 - TCR1/TBR1

The following control registers manage Stage 2 translations:

- Virtualization Control register (VCR): Section 5.5.11.31
- Virtualization Table Control register (VTCR): Section 5.5.11.32
- Virtualization Table Base register (VTBR): Section 5.5.11.33

5.5.3.3 Address Translation Mode

The SCR.HW bit controls the CMMU translation mode.

Only the secure context SCR is allowed to transition the CMMU from software-managed mode into hardware-managed mode. Attempts to perform this transition using the non-secure context are faulted with a status code of 0x3 (invalid operation).

Copyright © 2020, Texas Instruments Incorporated

Table 87. SCR.HW - Hardware Translation Mode Definition

Encoding	Description
0	SW-Managed Translation Mode.
1	HW-Managed Translation Mode

The reset value of SCR.HW is 0.

5.5.3.4 Stage 1 Address Translation Mode

When the CMMU is in the hardware-managed translation mode, the SCR.S1_EN bit is used to enable or disable Stage 1 address translation. Each context can independently control whether its first stage of address goes through address translation or uses flat addressing.

Table 88. SCR.S1_EN - Stage 1 Address Translation Enabled Definition

Encoding	Definition
0	Stage 1 Address Translation Disabled
1	Stage 1 Address Translation Enabled

The reset value of SCR.S1_EN is 0.

See Section 5.5.2.2.3.6 for details on the transition between enabling and disabling Stage 1 address translation.

NOTE: Stage 2 has a default cacheable mode that can override Stage 1 control. See Section 5.5.3.16.4.

5.5.3.5 Stage 1 ASID

Stage 1 translations cached in the TLB are tagged with an 8-bit ASID (address space identifier). Each Stage 1 context can select one active ASID for TLB tagging and TLB entry matching. The SCR.ASEL bit is used select the active ASID value from either the TBR0.ASID or TBR1.ASID.

Table 89. SCR.ASEL - ASID Selection Definition

Encoding	Definition
0	TBR0.ASID is active
1	TBR1.ASID is active

For entry matching, the ASID is ignored when a page translation is marked as Global.

5.5.3.6 Stage 2 Address Translation Mode

When the CMMU is in the hardware-managed translation mode, the VCR.S2_EN bit is used to enable or disable Stage 2 address translation. This stage of translation is meant to be controlled by the non-secure root context to virtualize the non-secure guest context.

Table 90. VCR.S2_EN - Stage 2 Address Translation Enabled Definition

Encoding	Definition
0	Stage 2 Address Translation Disabled
1	Stage 2 Address Translation Enabled

NOTE: Stage 2 has a default cacheable mode that can override the functionality of the VCR.S2_EN control bit. See Section 5.5.3.16.4.

Copyright © 2020, Texas Instruments Incorporated

5.5.3.7 Stage 2 VMID

When Stage 2 address translation is enabled for the guest context, cached translations for the guest context are further tagged with and matched on an 8-bit VMID (virtual machine identifier). Each guest context is assigned a unique VMID. The currently active guest context VMID value is saved in VTBR.VMID.

5.5.3.8 Table Walk Enable

The TCR.WALK_EN bit can be used to disable translation table walks, if there is a translation request that misses the TLB and must be walked.

Table 91. TCR.WALK_EN - Table Walk Enable Definition

Encoding	Definition
0	Translation requests that miss the TLB generate Translation Faults.
1	Translation requests that miss the TLB perform translation table walks.

5.5.3.9 Page Granule

The CMMU supports the following page granule sizes: 4 KB, 16 KB, and 64 KB.

The page granule size used by each translation table is configured by the GRANULE field in the table control registers.

- Stage 1 TCRs - TCR0/TCR1
- Stage 2 VTCR

Table 92. TCR/VTCR.GRANULE - Page Granule Size Definition

Encoding	Definition
00	4 KB page granule
01	64 KB page granule
10	16 KB page granule
11	Reserved

5.5.3.10 Table Size

The CMMU supports the same table size encoding and table size limits as the ARMv8 Arch64 Virtual Memory Specification.

The TCR/VTCR.TBL_SZ field is used to configure the associated Stage 1, Stage 2 translation table size. The exact translation table size = $2^{(64\text{-TBL\_SZ})}$ Bytes.

Example: TBL_SZ = 16, sets the table size as 2^{64-16}=2^{48} Bytes, the large possible configuration.

Stage 1 translation tables can be as small as 32 MB (TBL_SZ = 39) and as large 2^{48} Bytes (TBL_SZ =16). Stage 2 translation tables can be as small as 32 MB (TBL_SZ = 39) and as large as 2^{40} Bytes (TBL_SZ = 24). Functionally, the CMMU saturates at these table size limits. Writing values beyond these limits to the TBL_SZ field have no affect on the CMMU. The actual values can be read back and are not modified by the CMMU.

5.5.3.11 Stage 2 Translation Starting Level

The CMMU VTCR.SLEVEL field is functionally equivalent to the VTCR_EL2.SL0 defined in the ARMv8 64-bit Virtual Memory System Architecture. The encoded values have identical meaning for the given Stage 2 page granule and page table size. CMMU supports up to 16 aligned, concatenated pages as defined in the ARMv8 architecture. The legal page granule, table size, and start levels for the 40-bit physical address space in the Keystone 3 architecture are listed in Table 93.

Copyright © 2020, Texas Instruments Incorporated

Table 93. Legal Stage 2 Page Granule, Table Size, and Start Level Combinations

Page Granule	Start Level (SL0)	Initial Lookup Level	Table Size Encoding
4k page	00	Two	30 <= size <= 39
4k page	01	One	24 <= size <= 33
16k page	00	Three	35 <= size <= 39
16k page	01	Two	24 <= size <= 38
64k page	00	Three	31 <= size <= 39
64k page	01	Two	24 <= size <= 34

5.5.3.12 Table Base Address

Translation tables are required to be placed in memory at an alignment equal to the size of the table. The CMMU assumes that all page tables (including the initial table placed in the table base registers – TBR0/TBR1/VTBR) are adhered to this requirement. The CMMU does not perform an alignment check of the TBR/VTBR.BADDR value with respect to the initial table size as determined by the attributes in the TCR/VTCR registers.

For future address expansion, the CMMU expects the 8-bit Reserved section above the TBR/VTBR.BADDR field (bits [47:40]) to be written with 0s. Writing non-zeros to these bits is treated as a permission fault (status = 0x2).

5.5.3.13 Table Memory Endian

The CMMU supports reading translation tables in LITTLE or BIG Endian Memory (BE-8 format). All page table translation levels of a stage of address translation are required to reside in the same memory endian. The memory endian is only allowed to vary between stages of address translation.

For Stage 1, the SCR.ENDIAN0/ENDIAN1 fields control the endian for region 0 and region 1 translation tables, respectively, and are not required to be the same value.

For Stage 2, the VCR.ENDIAN field controls the endian of the Stage 2 translation table.

Table 94. SCR/VCR.ENDIAN - Table Memory Endian Definition

Encoding	Definition
0	Little Endian Memory
1	Big Endian Memory

5.5.3.14 Table Memory Attributes

The TCR/VTCR registers provide bit fields to specify the attributes of the memory where the translation page tables reside. Software can specify the memory type, allocation policy, and shareabililty.

Table 95. TCR/VTCR.MEMTYPE - Table Memory Type Definition

Encoding	Definition
00	Reserved
01	Normal Write-Back Cacheable Memory
10	Reserved
11	Normal Non-Cacheable Memory

Table 96. TCR/VTCR.CINNER/COUTER - Table Memory Inner/Outer Allocation Definition

Encoding	Definition
00	Non-Allocatable
01	Write Allocate

Copyright © 2020, Texas Instruments Incorporated

Table 96. TCR/VTCR.CINNER/COUTER - Table Memory Inner/Outer Allocation Definition (continued)

Encoding	Definition
10	Read Allocate
11	Read-Write Allocate

See Section 5.5.3.17.

Table 97. TCR/VTCR.SHARE - Table Memory Shareability Definition

Encoding	Definition
00	Non-shareable
01	Reserved
10	Outer Shareable
11	Inner Shareable

5.5.3.15 Protected Table Walks

Only normal non-cacheable memory and normal write-back memory types are allowed to be specified for the table walk memory type through the TCR/VTCR.MEMTYPE field. It is possible for Stage 2 translation to designate a different memory type as a part of the translation of the Stage 1 IPA, including the device memory type. For all other memory types, the CMMU follows the same Stage 1 and Stage 2 attribute merging as defined in the ARMv8 Arch64 Virtual Memory Specification.

Software can disallow Stage 1 table walks to memory designed as device memory by Stage 2 translation through the VCR.PROT bit.

Table 98. VCR.PROT - Protected Table Walk Definition

Encoding	Definition
0	Allow Stage 1 table walks to Device memory to be treated as Normal non-cacheable memory
1	Stage 1 table walks to Device memory are treated as Stage 2 Permission Faults.

See Section 5.5.6 for the information on the CMMU fault reporting.

5.5.3.16 Page Memory Attributes

5.5.3.16.1 Stage 1 Address Translation Disabled

When Stage 1 translation is disabled, default memory attributes are applied to data and instruction memory accesses.

- Data accesses – Device memory
- Instruction accesses – Normal memory with cacheability controlled through SCR.INSTC

Table 99. SCR.INSTC - Instruction Cacheability Defintion

Encoding	Definition
0	Normal Non-cacheable, Outer Shareable Memory
1	Normal Write-Through, Outer Shareable, Read Allocate Memory

NOTE: Stage 2 has a default cacheable mode that can override Stage 1 control and memory attributes. See Section 5.5.3.16.4.

Copyright © 2020, Texas Instruments Incorporated

5.5.3.16.2 Stage 1 Address Translation Enabled

When Stage 1 translation is enabled, the SCR.DATAC and SCR.INSTC fields can be used to disable cacheability of data accesses (including translation table walks) and instruction accesses.

Table 100. SCR.DATAC - Data Cacheability Definition

Encoding	Definition
0	Data accesses and translation table walks to Normal Memory are Non-cacheable.
1	No effect on the Cacheability of data accesses and translation table walks to Normal Memory.

Table 101. SCR.INSTC - Instruction Cacheability Definition

Encoding	Definition
0	Instruction accesses to Normal Memory are Non-cacheable.
1	No effect on the Cacheability of instruction accesses to Normal Memory.

The page memory type is determined through the 3-bit memory index field [4:2] of the Stage 1 page table entry descriptor. As specified in the ARMv8 Arch64 Virtual Memory System Architecture, this 3-bit field provides an index into a Memory Attribute register.

The MAR control register, which is banked by context, supports up to 8 memory attribute definitions.

NOTE: The SCR.DATAC and SCR.INSTC fields can override the memory attributes specified by the translation page table entry and MAR control register when the MAR memory type is normal memory.

Table 102. MAR - Memory Attribute Format

Bit Position	Definition
[1:0]	Memory Type
[3:2]	Inner Allocation Policy
[5:4]	Outer Allocation Policy
[6]	Prefetchability
[7]	Reserved, Read as Zero

Table 103. MAR - Memory Type Definition

Encoding	Definition
00	Device Memory
01	Normal Write-Back Cacheable Memory
10	Normal Write-Through Cacheable Memory
11	Normal Non-Cacheable Memory

Table 104. MAR - Inner/Outer Allocation Policy Definition

Encoding	Definition
00	Non-Allocatable
01	Write Allocate
10	Read Allocate
11	Read-Write Allocate

See Section 5.5.3.17.

Copyright © 2020, Texas Instruments Incorporated

Table 105. MAR - Prefetchability Definition

Encoding	Definition
0	Non-Prefetchable Memory
1	Prefetchable Memory

NOTE: Device memory and normal non-cacheable memory are inner/outer non-allocatable and non-prefetchable, regardless of the values written to the MAR control register.

NOTE: Stage 2 has a default cacheable mode that can override Stage 1 control and memory attributes. See Section 5.5.3.16.4.

5.5.3.16.3 Stage 2 Address Translation Enabled

When Stage 2 address translation is enabled, the Stage 2 translation provides its own memory type, shareability, and permissions of the translated physical address (from the translated Stage 1 intermediate physical address) for Stage 1 table walks as well as the final Stage 1 translation result.

The merging of the memory type and shareability between Stage 1 and Stage 2 follows the behavior defined in the ARMv8 Arch64 Virtual Memory Architecture:

- Device > Normal Non-Cacheable > Normal Write-Through Cacheable > Normal Write-Back Cacheable
- Outer Shareable > Inner Shareable > Non-Shareable

NOTE: The same behavior is used to merge the Stage 2 inner and outer memory type attributes into a single Stage 2 memory type.

The prioritization of the Stage 1 and Stage 2 permissions also follows the behavior defined in the ARMv8 Arch64 Virtual Memory Architecture, where a Stage 1 permission fault takes priority over a Stage 2 permission fault.

In addition, when Stage 2 is enabled, the VCR provides control to convert all Stage 2 translations to normal memory to be non-cacheable.

Table 106. VCR.ID - Stage 2 Instruction Cache Disable Definition

Encoding	Definition
0	No effect on Stage 2 translations of instruction accesses
1	All Stage 2 translations of instruction accesses to Normal memory are converted to Non-cacheable

Table 107. VCR.CD - Stage 2 Data Cache Disable Definition

Encoding	Definition
0	No effect on Stage 2 translations of data accesses and translation table walks
1	All Stage 2 translations of data accesses and translation table walks to Normal memory are converted to Non-cacheable

5.5.3.16.4 Stage 2 Default Cacheable Mode

As defined in the ARMv8 Arch64 Virtual Memory Architecture, the Stage 2 default cacheable mode forces the guest context's Stage 1 address translation to be disabled and the Stage 2 address translation to be enabled, regardless of the values of SCR.S1_EN and VCR.S2_EN.

Copyright © 2020, Texas Instruments Incorporated

In the Stage 2 default cacheable mode, Stage 1 address translation returns the memory type of normal non-shareable, write-back, inner/outer read-write allocate memory for all accesses. Additionally, the SCR.INSTC and SCR.DATAC control bits have no effect on the Stage 1 memory type returned for instruction or data and table walk accesses.

Table 108. VCR.DC - Stage 2 Default Cacheable Mode Definition

Encoding	Definition
0	No effect on Stage 1 or Stage 2 address translation
1	Disables the Guest context's Stage 1 of address translation. Enables Stage 2 of address translation. Stage 1 address translation return the memory type of Normal Non-Shareable, Write-Back, Inner/Outer read-write allocate. SCR.INSTC has no effect on the cacheability of instruction accesses.

5.5.3.16.5 Instruction Access to Device Memory

Instruction accesses to memory designated as device memory by Stage 1 and/or Stage 2 address translation are instead converted to normal non-cacheable outer shareable memory.

5.5.3.17 Inner/Outer Allocation Policy

In general, inner refers to the L1 and unified L2 caches and outer refers to the shared L3 cache. Regarding the shared L3 SRAM, the unified L2 cache controller uses a separate 'M3' configuration bit to enable caching.

5.5.3.18 Permissions

The SCR.WXN provides Stage 1 of the software context direct control over disallowing any code execution of memory that is writable. See Section 5.5.4 for the full list of access controls available to software.

Table 109. SCR.WXN - Stage 1 Writable Memory Execute Never

Encoding	**Definition**
0	No effect, execution is determined by XN related Page Table Entry fields
1	All writeable memory regions are execute never

5.5.3.19 Faults

The SCR.FAULT field provides the Stage 1 software context with the ability to have the uTLBs cache faulted translations. This feature is intended to enable faster accesses where a precise CPU exception is not usually taken, namely streaming prefetches. By using this feature, software is expected to perform TLB cache maintenance on faulted translations which normally are not cached in the TLBs - (that is, translation faults, access flag faults, external errors, and external aborts).

Table 110. SCR.FAULT - Caching Faults in uTLBs

Encoding	**Definition**
0	No effect
1	Faulted translations are cached in uTLBs

5.5.3.20 Secure and Non-Secure Accessing the Same Physical Memory

For physical memory accessed by both the secure and non-secure contexts, the secure context is recommended to mark memory as non-secure (NS) in its Stage 1 page table descriptors. Not doing so can result in memory accesses with undefined behavior.

Copyright © 2020, Texas Instruments Incorporated

5.5.3.21 Hint Support

CMMU supports the contiguous page hint bit as defined in the ARMv8 64-bit architecture. The contiguous page hint bit is ignored for the illegal combinations where the hint bit for a given page granule and translation level exceed the defined table size.

5.5.4 Memory Access Controls

The CMMU supports the same Stage 1 access permission checks as defined in the ARMv8 Arch64 Specification:

1. The supervisor can always read memory.
2. Writeable memory is always readable.
3. Supervisor cannot execute user writeable memory.
4. Writeability control
5. User access control
6. Execution control
7. Writable memory not executable control (SCR.WXN)
8. Secure execution of non-secure memory (permanent check, not controllable)

See Section 5.5.5 for the differences in how the translation table descriptors are interpreted by the CMMU.

The CMMU supports the same Stage 2 access permission checks as defined in the ARMv8 Arch64 Specification:

- Disallow reads
- Disallow writes
- Disallow execution

The CMMU checks Stage 1 and Stage 2 permissions separately, and reports the permission faults with the same priority as specified in the ARMv8 Spec (Stage 1 permission faults take priority over Stage 2 permission faults).

Stage 2 read permission control also apply to Stage 1 page table entry memory fetches.

5.5.5 C7x, ARMv8 Aarch64 Translation Table Format Differences

The C7x translation table format is compliant with the ARMv8 Arch64 descriptor format. The only exception is the interpretation of never-execute fields of the Stage 1 descriptors.

Specifically, table descriptors:

- bit[60] - UXNTable, XNTable
- bit[59] - PXNTable

And block/page descriptors:

- bit[54] - UXN, XN
- bit[53] - PXN

ARMv8 uses UXN and PXN bitfields for EL0/EL1 tables and only the XN bitfield for EL2/EL3 tables.

C7x uses the UXN and PXN bitfield to control code execution for the supervisor(PXN) and user(UXN) privilege levels across all context levels: secure, non-secure, and non-secure guest.

5.5.6 CMMU Fault and C7x Page Fault Handling

By default, all fault types, with the exception of permission faults (excluding protected table walk - PTW faults), are not cached in the uTLB/TLB. The software therefore does not have to perform any TLB maintenance operation when servicing a translation-related exception.

The SCR.FAULT bit provides the ability to cache faults for the purpose of enabling faster return of successive translation requests (due to prefetching), which result in translation-related faults. By enabling this behavior, the software must perform TLB maintenance to remove the cached faulting entry.

Copyright © 2020, Texas Instruments Incorporated

See Table 111 for a description on the translation failure reporting definition.

The fault types are aligned with the ARMv8 AArch64 Virtual Memory System Architecture.

Table 111. Translation Failure Reporting

Bit	Failure Type	Definition
rstatus[11]	Fault Valid	0 = No Fault 1 = Fault
rstatus[10]	Security Mismatch Failure	Secure-mode accessing page marked non-secure
rstatus[9]	Protected Table Walk Fault	0 = Not Protected Fault 1 = Protected Fault
rstatus[8]	Failure Translation Stage	0 = Stage 1 1 = Stage 2
rstatus[7]	Cache Maintenance Operation Failure	Fault transaction was a CMO
rstatus[6]	Read not Write Failure	0 = write 1 = read
rstatus[5:0] = 0000LL	Address Size Fault	Encountered PA beyond 40-bits or IPA beyond Stage 2 table size boundary
rstatus[5:0] = 0001LL	Translation Fault	LL = level of fault
rstatus[5:0] = 0010LL	Access Flag Fault	LL = level of fault
rstatus[5:0] = 0011LL	Permission Fault	LL = level of fault
rstatus[5:0] = 0100LL	External Parity Error on Table Memory Access	LL = level of fault
rstatus[5:0] = 011000	TLB Conflict Abort	Multiple TLB Entry Match

rstatus[5:0] = 1SSSLL	External Abort on Table Memory Access	LL = level of fault
		SSS = Memory Status Error Encoding

5.5.7 Address Translation Debug

The CMMU supports on-demand address translation through the Translation Debug Address registers (TDAR). The CMMU performs address translation on the address written in the register and logs the results in the Translation Debug Response register (TDRR) and Translation Debug Fault Address register (TDFAR).

5.5.7.1 Translation Debug Address Register (TDAR/VTDAR)

The TDAR is used to perform an on-demand translation request for the current context or for lower privileged context (through the aliased registers). The translation results are written into the TDRR/TDFAR registers.

For the non-secure guest context, translation results are for the Stage 1 view of memory (IPA result), while using the VTDAR produces the final Stage 1 + Stage 2 view of memory (PA result).

See the following registers:

- Section 5.5.11.16
- Section 5.5.11.34

The TDAR.ADDR field expects bits [63:4] of the input address with the necessary address sign extension.

The TDAR.INTEREST field to used to enable the debug probing of the translation request.

Table 112. TDAR.INTEREST - Interest Flag

Encoding	Definition
0	No Effect
1	Use Interest Flag

The translation access type and access privilege level of the on-demand translation request is determined by the TDAR.ACC_TYPE and TDAR.PRIV fields, respectively.

Copyright © 2020, Texas Instruments Incorporated

Table 113. TDAR.ACC_TYPE - Access Type

Encoding	Definition
00	Data Write Access
01	Data Read Access
10	Code Execution Access
11	Reserved

Table 114. TDAR.PRIV - Privileged Access

Encoding	Definition
0	User Access
1	Supervisor Access

5.5.7.2 *Translation Debug Response Register (TDRR)*

The TDRR holds the following on-demand translation results:

- Output address[39:12]
- Memory attributes – prefetchability, cacheability, memory type, shareability
- Status response

See Section 5.5.11.3.

The encoding and definition of the TDRR fields match those found in these sections:

- Table 97
- Table 102
- Table 111

The TDRR includes a completion bit, where a ‘1’ signifies that the translation result is ready.

In the event of a bad status response (TDRR.STATUS[11] = 1), the memory attributes and output address are undefined, except in the event of a Stage 2 fault, when the output address represents the faulting IPA – intermediate physical address used by the non-secure guest context.

5.5.7.3 *Translation Debug Fault Address Register (TDFAR)*

In the event of a bad status response for the on-demand translation access, the TDFAR register holds the entire 64-bit value that was originally written to the TDAR/VTDAR register, which includes the input address, access type, and access privilege level. See Section 5.5.11.4.

5.5.7.4 *TDRR/TDFAR Restrictions*

In the event an on-demand translation request for the non-secure guest context results in a Stage 2 fault, access of the TDRR/TDFAR results by the non-secure guest context is disallowed, with the CMMU returning a permission fault. Software is expected to treat this as a normal Stage 2 fault to be handled by the non-secure supervisor/hypervisor context.

5.5.7.5 *Saving and Restoring TDRR/TDFAR*

The TDRR and TDFAR registers are writeable in the event multiple contexts are using the TDAR feature and the accesses occur during interruptible code. Software can perform a save and restore, before and after a context switch and return.

Due to the restrictions described in Section 5.5.7.4, the guest context is disallowed from writing a TDRR.STATUS value which represents a valid Stage 2 fault.

Copyright © 2020, Texas Instruments Incorporated

5.5.7.6 Pre-warm CMMU TLBs

The on-demand translation request updates the CMMU main TLBs in the same manner as a normal translation request. Software can technically use the TDAR/VTDAR feature as a method of pre-warming or priming the CMMU TLBs for a subsequent functional translation request, in the event that more deterministic translation response latencies are required while in hardware-managed mode.

5.5.8 Translation Lookaside Buffer (TLBs)

5.5.8.1 TLB Hierarchical Structure

The CMMU TLB structure is composed of multiple levels of hierarchy, ranging from multiple Level 1 uTLBs, to several page caches that make up the Level 2 TLBs for both Stage 1 and Stage 2 translations.

5.5.8.1.1 CMMU uTLB Caches

- 16 entry load/store data uTLB
- 8 entry instruction uTLB
- 2 × 8 entry streaming engine uTLBs
- All uTLBs are fully associative caches
- Page sizes supported: 4K, 16K, 64K, 2MB, 32MB, 512MB, 1GB, 16GB
- Translation types cached:
  - VA->PA translations (when Stage 1 translation is enabled)
  - IPA->PA translations (when Guest Stage 1 translation is disabled and Stage 2 translation is enabled)

5.5.8.1.2 CMMU TLB Small Page Cache

- 512 entries
- 128 set associative 4-way cache
- Caches level 3 page VA-to-PA translations

5.5.8.1.3 CMMU TLB Large Page/Walking Caches

- Shared Stage 1 page and walk caches:
  - 16 entry MB cache: 2MB, 32MB, and 512MB. Blocks and pointers (page granule dependent)
  - 8 entry GB cache: 1GB, 16GB, and 64GB. Blocks and pointers (page granule dependent)
  - 4 entry PTR cache: 512GB, 4TB, 128TB. Pointers only.
  - All large page/walk caches are fully associative.
  - Cache VA-to-PA translations
- Shared Stage 2 page and walk caches:
  - 8 entry MB caches: 2MB, 32MB, and 512MB. Blocks and pointers (page granule dependent)
  - 4 entry GB caches: 1GB, 16GB, and 64GB. Blocks and pointers (page granule dependent)
  - All large page/walk caches are fully associative.
  - Cache IPA-to-PA translations

5.5.8.2 Allocation and Replacement Policy

The uTLB/TLB caches support two allocation/replacement modes:

- Pseudo-random mode
- FIFO mode

The replacement mode for the uTLBs are controlled through independent registers for each uTLB.

- Load/Store uTLB: Section 5.5.11.37

Copyright © 2020, Texas Instruments Incorporated

- Instruction uTLB: Section 5.5.11.40
- Stream Engine 0 uTLB: Section 5.5.11.43
- Stream Engine 1 uTLB: Section 5.5.11.46

The replacement mode for the TLB caches are controlled through MMU_REPL See Section 5.5.11.2.

Table 115. Replacement Policy Definition

Encoding	Description
0	Pseudo random Policy
1	FIFO Policy

The reset value for the POLICY field is 0.

The pseudo-random policy is implemented using a 16-tap maximal length LFSR.

In the case of an inadvertent transition to the all 0's state, the LFSR transitions to the all 1’s state to prevent becoming stuck.

The LFSR field is software-writeable to introduce greater variability in the allocation/replacement process between the individual uTLBs and the TLB caches.

5.5.8.2.1 CMMU uTLB Replacement Policy

The uTLBs allocate address translation entries into the first available entry. The replacement policy is used only when a uTLB is completely full. The replacement policy can be controlled on a per uTLB instance basis.

5.5.8.2.2 CMMU TLB Cache Allocation/Replacement Policy

The pseudo-random mode is shared across the small and large page/walk caches.

The FIFO mode is individualized per cache. The small page cache has a counter shared across all of the cache sets. Likewise, each large page/walk cache has its own counter.

For allocation in the large page/walk caches, the replacement policy applies only when the respective cache is full, otherwise the first available slot is chosen for allocation.

The replacement policy is used for all allocations into the small page cache.

5.5.8.3 TLB Maintenance

Software is responsible for maintaining proper coherency between translations potentially cached in the CMMU’s TLBs and the page table entries stored in main memory. When page table entries are updated, the CMMU TLBs are required to be invalidated such that modifications to address translation or page attributes can be re-fetched from main memory. Failure to do so can potentially result in undefined behavior.

5.5.8.3.1 Recommended Software Use Model

To avoid issues with possible in-flight translation requests, TLB maintenance, stale translations, and page table descriptor updates, Ti strongly recommends that the software use a break-before-make approach when updating any page table descriptor attributes. The break-before-make approach is the following:

1. Change the page table descriptor to an invalid entry.
2. Perform a memory barrier/fence operation to make the descriptor update visible.
3. Perform a TLB maintenance/invalidation operation remove the translation from the CMMU TLBs.
4. Perform a memory barrier/fence operation to ensure the completion of the invalidation operation.
5. Change the page table descriptor to be a valid entry with the new attributes.
6. Perform a memory barrier/fence operation to make the descriptor update visible.

Copyright © 2020, Texas Instruments Incorporated

TI recommends that any necessary cache maintenance operations (CMOs) be completed prior to any memory attributes changes which may conflict with prior memory attributes in terms of memory type or cacheability attributes.

5.5.8.3.2 TLB Invalidation

The External Config register (ECR) - TLB_INV (TLB Invalidation register) is used to execute TLB invalidation operations on the CMMU TLBs/uTLBs. The TLB_INV register handles Stage 1 and Stage 2 TLB invalidations for all CPU contexts, both secure and non-secure. See Section 5.5.11.5.

NOTE: The CMMU architecture supports TLB invalidation requests through distributed virtual memory transactions (DVMs), but this feature is de-scoped from the C7x Corepac/compute cluster level.

5.5.8.3.2.1 Invalidation Type

The CMMU supports several different types of TLB invalidation:

- Stage 1 of the current context
- Stage 1 of the lower-level context (secure can invalidate non-secure, and non-secure supervisor can invalidate non-secure guest)
- Stage 2 IPA match (intermediate physical address)
- Stage 1 and Stage 2 of guest contexts

Table 116. TLB_INV.INV_TYPE - Invalidation Type Definition

Encoding	Definition
000	Invalidation of the Stage 1 entries of the current context
001	Invalidation of the Stage 1 entries of the Guest context
010	Invalidation of the Stage 1 entries of the Non-secure Supervisor context
011	Reserved
100	Invalidation of the Stage 2 entries with IPA Match (both pointers and last level entries)
101	Invalidation of the Stage 2 entries (last level entries only)
110	Invalidation of all Stage 1 and Stage 2 entries for the current Guest context
111	Invalidation of all Stage 1 and Stage 2 entries for all Guests contexts

5.5.8.3.2.2 Stage 1 Match Attributes

The Stage 1 TLB invalidation type is determined by the ASID, VA, and LL fields of the TLB_INV register.

- ASID – match on ASID
- VA – match on virtual address
- LL – last level only invalidation (pages, not table pointers)

Table 117 lists all of the possible Stage 1 TLB invalidation types.

Table 117. Stage 1 TLB Invalidation Types

ASID	VA	LL	Invalidation Type
0	0	0	All entries for the context
0	0	1	Only last level entries for the context
0	1	0	All entries that match the target Virtual Address
0	1	1	All last level entries that match the target Virtual Address
1	0	0	All entries that match the target ASID
1	0	1	All last level entries that match the target ASID
1	1	0	All entries that match the target ASID and Virtual Address

Copyright © 2020, Texas Instruments Incorporated

Table 117. Stage 1 TLB Invalidation Types (continued)

ASID	VA	LL	Invalidation Type
1	1	1	All last level entries that match the target ASID and Virtual Address

5.5.8.3.2.3 ASID/Address Match

For the ASID-based matching, the 8-bit ID field represents the ASID used for entry matching.

For the Stage 1 VA-based matching, the 37-bit ADDR field represents bits[48:12] of the virtual address.

For Stage 2 IPA-based matching, the 37-bit ADDR field represents bits [39:12] of the IPA. Due to the 40-bit physical address memory space of the Keystone 3 Architecture, the 9 MSBs of the ADDR field must be written with all 0's for all Stage 2 IPA TLB invalidation operations.

5.5.8.3.2.4 Invalidation Completion

TI recommends that the software poll the TLB_INVC (TLB Invalidation Completion register) COMP field to determine when a TLB invalidation has completed. See Section 5.5.11.6.

Table 118. TLB_INV.COMP - TLB Invalidation Completion

Encoding	Definition
0	TLB Invalidation Request still in progress (reset value)
1	TLB Invalidation Operation Completed

5.5.8.4 TLB Conflict Detection

When the TLBs are not properly maintained using TLB maintenance operations, the TLBs can enter invalid states such as having overlapping matches for a given translation request. In cases where the uTLBs and TLB caches can detect a conflict, a TLB conflict fault is returned. See Section 5.5.6.

5.5.8.5 Parity Error Detection Support

The CMMU's main TLB support parity error detection and invalidation. Entries with detected parity errors are treated as misses and are immediately invalidated.

- The CMMU large page/walk caches support single-bit parity detection per entry.
- The TLB small page RAM cache supports even-odd bit parity support per RAM instance.
- Certain pipe stages are also parity protected.
- Table walk memory access returns are checked for parity errors on a per 32-bit basis.

5.5.8.6 TLB Debug Access

In hardware-managed mode, the CMMU supports debug read access of the contents of the TLB entries.

5.5.8.6.1 uTLB Debug Access

The CMMU registers used in software-managed mode of the uTLBs are also available for debug read access when the CMMU is in hardware-managed mode. See Section 5.5.2.1.

5.5.8.6.2 TLB Debug Access

To read the contents of the individual TLB entries, the software must write to the TLB Debug Access register (TLB_DBG) followed by reading back the TLB entry contents populated into the TLB Debug Data registers (TLB_DBG_DATA0, TLB_DBG_DATA1).

Copyright © 2020, Texas Instruments Incorporated

5.5.8.6.2.1 TLB_DBG Register

The write-only TLB _DBG register provides debug read visibility to all of the caching structures in the main TLB. See Table 119 for the encoding of the TLB_DBG.TLB field. See Section 5.5.11.7.

Table 119. TLB_DBG.TLB - TLB Type Definition

Encoding	Definition
00	Stage 1 Small page RAM cache
01	Stage 1 Large page RF cache
10	Stage 2 Large page RF cache
11	Reserved

The TLB_DBG.WAY field selects the way of the 4-way set-associative Stage 1 small page RAM cache, and has no affect on the Stage 1-2 large page RF cache accesses.

The TLB_DBG.INDEX field selects the cache-set of the Stage 1 small page RAM cache, and the entry index of the Stage 1-2 large page RF caches.

See the following tables for the index range assignments for the Stage 1-2 large page RF caches.

Table 120. Stage 1 Large page RF Cache Index Assignment

Index	Cache Structure
0-15	16 entry Stage 1 MB Cache
16-23	8 entry Stage 1 GB Cache
24-27	4 entry Stage 1 PTR cache
28-127	Reserved

Table 121. Stage 2 Large page RF Cache Index Assignment

Index	Cache Structure
0-7	8 entry Stage 2MB Cache
8-11	4 entry Stage 2GB Cache
12-127	Reserved

5.5.8.6.2.2 TLB_DBG_DATA Registers

The read-only TLB_DBG_DATA registers are populated with the contents of the TLB entry referenced by the software write to the TLB_DBG register. See the following register definitions:

- Section 5.5.11.8
- Section 5.5.11.9

The TLB_DBG_DATA registers are populated with all zeros for invalid TLB entries and non-secure debug context accesses to secure TLB entries.

Table 122. TLB_DBG_DATA0 - Stage 1 Small Page Cache

Bits	Name	Description
[63]	NS	Non-Secure Page (only applies to the Secure context) 0 - Secure memory 1 - Non-secure memory Includes table descriptor contribution.
[62:59]	DS _SIZE	Descriptor Size: Read as 0000, level 3 (non-contiguous) page. Page size determined by the page granule of the context.
[58]	DS_TYPE	Descriptor Type: Read as 1, page entry

Copyright © 2020, Texas Instruments Incorporated

Table 122. TLB_DBG_DATA0 - Stage 1 Small Page Cache (continued)

Bits	Name	Description
[57:28]	IADDR	Virtual Address
[27:20]	VMID	Virtual Machine Identifier
[19:12]	Reserved	Read as zeros
[11:4]	ASID	Address Space Identifier
[3]	GBL	Global bit 0 - Non-global entry 1 - Global Entry
[2]	ROOT	Root Bit (applies to non-secure only) 0 - Entry for the Guest context 1 - Entry for the Root context
[1]	SEC	Secure Bit 0 - Entry for the Non-Secure context 1 - Entry for the Secure context
[0]	VALID	Valid Bit 0 - Entry is invalid and all data has been zeroed out 1 - Entry is valid

Table 123. TLB_DBG_DATA1 - Stage 1 Small Page Cache

Bits	Name	Description
[63:48]	Reserved	Read as zeros
[47:46]	SHARE	Shareability (ARMv8 Based) 00 - Non-Shareable 01 - Reserved 10 - Outer Shareable 11 - Inner Shareable
[45:44]	S2_LVL	The Stage 2 level that produced the translation: 00 - No Stage 2 translation performed 01 - Level 1 10 - Level 2 11 - Level 3
[43:40]	S2_MEM_TYPE	Stage 2 Memory Type For Stage 1 only entries, the value is 0000.
[39:36]	S2_PERM	Stage 2 Permissions: [39] - Stage 2 Permissions Valid Bit [38] - XN [37:36] - S2AP[1:0]
[35:33]	S1_MEM_INDEX	Stage 1 Memory Type Index
[32:28]	S1_PERM	Stage 1 Permissions: [32] - Stage 1 Permissions Valid Bit [31] - PXN* [30] - UXN* [29:28] - AP[2:1]* * Includes table descriptor contributions.
[27:0]	OADDR	Physical Address

Table 124. TLB_DBG_DATA0 - Stage 1 Large Page RF

Bits	Name	Description
[63]	NS	Non-Secure Page (only applies to the Secure context) 0 - Secure memory 1 - Non-secure memory Includes table descriptor contribution.

Copyright © 2020, Texas Instruments Incorporated

Table 124. TLB_DBG_DATA0 - Stage 1 Large Page RF (continued)

Bits	Name	Description
[62:59]	DS _SIZE	Descriptor Size: 0011 - 2 MB 0100 - 32 MB 0101 - 512 MB 0110 - 1 GB 0111 - 16 GB 1000 - 64 GB* 1001 - 512 GB* 1010 - 4 TB* 1011 - 128 TB* All other values are Reserved * Only pointer entries
[58]	DS_TYPE	Descriptor Type: 0 - Pointer 1 - Page
[57:28]	IADDR	Virtual Address
[27:20]	VMID	Virtual Machine Identifier
[19:12]	Reserved	Read as zeros
[11:4]	ASID	Address Space Identifier
[3]	GBL	Global bit 0 - Non-global entry 1 - Global Entry For pointer entries, the value is 0.
[2]	ROOT	Root Bit (applies to non-secure only) 0 - Entry for the Guest context 1 - Entry for the Root context
[1]	SEC	Secure Bit 0 - Entry for the Non-Secure context 1 - Entry for the Secure context
[0]	VALID	Valid Bit 0 - Entry is invalid and all data has been zeroed out 1 - Entry is valid

Table 125. TLB_DBG_DATA1 - Stage 1 Large Page RF

Bits	Name	Description
[63:48]	Reserved	Read as zeros
[47:46]	SHARE	Shareability (ARMv8 Based) 00 - Non-Shareable 01 - Reserved 10 - Outer Shareable 11 - Inner Shareable For Stage 1 only pointer entries, the value is 00. For Stage 2 translated Stage 1 pointer entries, the value represents the Stage 2 Shareability. For Stage 2 translated Stage 1 page entries, the value represents the merged Stage 1/Stage 2 Shareability.
[45:44]	S2_LVL	The Stage 2 level that produced the translation: 00 - No Stage 2 translation performed 01 - Level 1 10 - Level 2 11 - Level 3
[43:40]	S2_MEM_TYPE	Stage 2 Memory Type For Stage 1 only entries, the value is 0000.
[39:36]	S2_PERM	Stage 2 Permissions: [39] - Stage 2 Permissions Valid Bit [38] - XN [37:36] - S2AP[1:0]
[35:33]	S1_MEM_INDEX	Stage 1 Memory Type Index For pointer entries, the value is 000.

Copyright © 2020, Texas Instruments Incorporated

Table 125. TLB_DBG_DATA1 - Stage 1 Large Page RF (continued)

Bits	Name	Description
[32:28]	S1_PERM	Stage 1 Permissions: [32] - Stage 1 Permissions Valid Bit [31] - PXN* [30] - UXN* [29:28] - AP[2:1]* * Includes table descriptor contributions.
[27:0]	OADDR	Physical Address

Table 126. TLB_DBG_DATA0 - Stage 2 Large Page RF

Bits	Name	Description
[63]	Reserved	Read as zero.
[62:59]	DS _SIZE	Descriptor Size: 0011 - 2 MB 0100 - 32 MB 0101 - 512 MB 0110 - 1 GB 0111 - 16 GB All other values are Reserved
[58]	DS_TYPE	Descriptor Type: 0 - Pointer 1 - Page
[57:28]	IADDR	Virtual Address
[27:20]	VMID	Virtual Machine Identifier
[19:12]	Reserved	Read as zeros.
[11:4]	Reserved	Read as zeros.
[3]	Reserved	Read as zero.
[2]	Reserved	Read as zero.
[1]	Reserved	Read as zero.
[0]	VALID	Valid Bit 0 - Entry is invalid and all data has been zeroed out 1 - Entry is valid

Table 127. TLB_DBG_DATA1 - Stage 2 Large Page RF

Bits	Name	Description
[63:48]	Reserved	Read as zeros
[47:46]	SHARE	Shareability (ARMv8 Based) 00 - Non-Shareable 01 - Reserved 10 - Outer Shareable 11 - Inner Shareable For Pointer entries, the value is 00.
[45:44]	Reserved	Read as zeros.
[43:40]	S2_MEM_TYPE	Stage 2 Memory Type For Pointer entries, the value is 0000
[39:36]	S2_PERM	Stage 2 Permissions: [39] - Stage 2 Permissions Valid Bit [38] - XN [37:36] - S2AP[1:0] For Pointers entries, the value is 0000
[35:33]	Reserved	Read as zeros.
[32:28]	Reserved	Read as zeros.
[27:0]	OADDR	Physical Address

Copyright © 2020, Texas Instruments Incorporated

5.5.9 Safety Support

The CMMU has parity detection and invalidation support as mentioned in Section 5.5.8.5; however, the parity checkers are not software testable through any safety diagnostics.

Software can perform its own software-based scrubbing of the CMMU TLBs by executing a virtual address match-based TLB Invalidation. This operation iteratively scans through all TLB entries and performs a parity detection and invalidation on any TLB entries with a parity error.

5.5.10 Distributed Virtual Memory Transactions (DVMs)

The ARMv7 and ARMv8 architecture supports sending messages between processors to facilitate multi-processing. The CMMU supports receiving and responding to DVMs broadcasted by ARM cores in the compute cluster.

NOTE: The CMMU architecture supports distributed virtual memory transactions (DVMs), but this feature is de-scoped from the C7x Corepac/compute cluster level.

The supported DVMs are:

- TLB invalidation
- Synchronization

The CMMU does not support generating its own DVM master transactions. The only exception is a synchronization complete DVM in response to a sync DVM. The C7x/CMMU responds to all DVM transactions regardless of support.

Clarification of supported DVMs:

- EL3 TLB invalidation is not supported (that is, it does not trigger any TLB invalidation activity in the CMMU)
- Hypervisor TLB invalidation DVMs map to the non-secure root privilege
- Non-secure guest OS TLB invalidation DVMs map to the non-secure root privilege when there is no 2-Stage virtualization (no guest OS virtualization)

The DVM-based TLB invalidation is functionally identical to the CPU ECR-based TLB invalidation. Both methods of TLB invalidation affect the CMMU’s TLB and the uTLBs and nano-TLBs in the C7x.

Due to the non-blocking nature of DVMs, the TLB invalidation DVM progresses to completion irrespective of any in-flight translation requests. During a TLB invalidation, all in-flight translation requests are marked as “do-not-cache” to ensure that post-synchronization DVM there are no stale translations in use or cached in the TLBs.

The generated synchronization complete DVM signals that all DVM operations received prior to the corresponding synchronization DVM have been completed, or at a minimum are now observable by all other masters in the system.

The CMMU services a synchronization DVM only when all prior DVM operations have been completed, and likewise generates the synchronization complete DVM when there are no outstanding translation requests and no outstanding memory requests.

Micro arch details:

- The TLB and uTLBs are invalidated at the time of the TLB invalidation DVM operation.
- The nano-TLBs are invalidated at the time of the synchronization DVM.

The CMMU supports receiving the maximum number of DVM syncs allowed in the system. The maximum number is dependent on the number of ARM clusters supported in the compute cluster architecture.

Copyright © 2020, Texas Instruments Incorporated

The CMMU supports newer TLB invalidation DVMs preempting a prior synchronization DVM operation.

5.5.11 Memory Mapped ECR Registers

Table 128. Memory Mapped ECR Registers

Offset	Acronym	Register Description	Section
400h	MMU_PID	Physical ID Register	Section 5.5.11.1
402h	MMU_REPL	Replacement Policy Register	Section 5.5.11.2
404h	TDRR	Translation Debug Response Register	Section 5.5.11.3
405h	TDFAR	Translation Debug Fault Address Register	Section 5.5.11.4
408h	TLB_INV	TLB Invalidation Register	Section 5.5.11.5
40Ah	TLB_INVC	TLB Invalidation Complete Register	Section 5.5.11.6
40Bh	TLB_DBG	TLB Debug Access Register	Section 5.5.11.7
40Ch	TLB_DBG_DATA0	TLB Debug Data 0 Register	Section 5.5.11.8
40Dh	TLB_DBG_DATA1	TLB Debug Data 1 Register	Section 5.5.11.9
410h	SCR	System Control Register	Section 5.5.11.10
411h	TCR0	Table Control Register 0	Section 5.5.11.11
412h	TCR1	Table Control Register 1	Section 5.5.11.12
413h	TBR0	Table Base Register 0	Section 5.5.11.13
414h	TBR1	Table Base Register 1	Section 5.5.11.14
415h	MAR	Memory Attribute Register	Section 5.5.11.15
416h	TDAR	Translation Debug Address Register	Section 5.5.11.16
420h	SCR_GS	System Control Register - Non-Secure Guest Supervisor	Section 5.5.11.17
421h	TCR0_GS	Table Control Register 0 - Non-Secure Guest Supervisor	Section 5.5.11.18
422h	TCR1_GS	Table Control Register 1 - Non-Secure Guest Supervisor	Section 5.5.11.19
423h	TBR0_GS	Table Base Register 0 - Non-Secure Guest Supervisor	Section 5.5.11.20
424h	TBR1_GS	Table Base Register 1 - Non-Secure Guest Supervisor	Section 5.5.11.21
425h	MAR_GS	Memory Attribute Register - Non-Secure Guest Supervisor	Section 5.5.11.22
426h	TDAR_GS	Translation Debug Address Register - Non-Secure Guest Supervisor	Section 5.5.11.23
430h	SCR_S	System Control Register - Non-Secure Supervisor	Section 5.5.11.24
431h	TCR0_S	Table Control Register 0 - Non-Secure Supervisor	Section 5.5.11.25
432h	TCR1_S	Table Control Register 1 - Non-Secure Supervisor	Section 5.5.11.26
433h	TBR0_S	Table Base Register 0 - Non-Secure Supervisor	Section 5.5.11.27
434h	TBR1_S	Table Base Register 1 - Non-Secure Supervisor	Section 5.5.11.28
435h	MAR_S	Memory Attribute Register - Non-Secure Supervisor	Section 5.5.11.29
436h	TDAR_S	Translation Debug Address Register - Non-Secure Supervisor	Section 5.5.11.30

Copyright © 2020, Texas Instruments Incorporated

Table 128. Memory Mapped ECR Registers (continued)

Offset	Acronym	Register Description	Section
460h	VCR	Virtualization Control Register	Section 5.5.11.31
461h	VTCR	Virtualization Table Control Register	Section 5.5.11.32
462h	VTBR	Virtualization Table Base Register	Section 5.5.11.33
463h	VTDAR	Virtualization Translation Debug Address Register	Section 5.5.11.34
480h	L1D_UTLB_MATCH	Indexed uTLB Page Match Entry	Section 5.5.11.35
482h	L1D_UTLB_ATTR	Indexed uTLB Page Attribute Entry	Section 5.5.11.36
484h	L1D_UTLB_REPL	uTLB Replacement Policy Register	Section 5.5.11.37
488h	L1P_UTLB_MATCH	Indexed uTLB Page Match Entry	Section 5.5.11.38
48Ah	L1P_UTLB_ATTR	Indexed uTLB Page Attribute Entry	Section 5.5.11.39
48Ch	L1P_UTLB_REPL	uTLB Replacement Policy Register	Section 5.5.11.40
490h	SE0_UTLB_MATCH	Indexed uTLB Page Match Entry	Section 5.5.11.41
492h	SE0_UTLB_ATTR	Indexed uTLB Page Attribute Entry	Section 5.5.11.42
494h	SE0_UTLB_REPL	uTLB Replacement Policy Register	Section 5.5.11.43
498h	SE1_UTLB_MATCH	Indexed uTLB Page Match Entry	Section 5.5.11.44
49Ah	SE1_UTLB_ATTR	Indexed uTLB Page Attribute Entry	Section 5.5.11.45
49Ch	SE1_UTLB_REPL	uTLB Replacement Policy Register	Section 5.5.11.46

5.5.11.1 MMU_PID (offset = 400h)

Physical ID Register

Table 129. MMU_PID Field Descriptions

Bit	Field	Type	Reset	Description
63-32	Reserved	R	0h	Reserved
31-30	SCHEME	R	1h	PID naming scheme
29-28	BU	R	2h	Business Unit
27-16	FUNC	R	F0h	MMU Function code
15-11	R	R	0h	Minor Revision
10-8	X	R	0h	Architecture Revision
7-6	CUSTOM	R	0h	Reuseable/Custom
5-0	Y	R	0h	Configuration Revision

5.5.11.2 MMU_REPL (offset = 402h)

Replacement Policy Register

Table 130. MMU_REPL Field Descriptions

Bit	Field	Type	Reset	Description
63-32	Reserved	R	0h	Reserved
31-16	LFSR	RW	0h	LFSR Value

Copyright © 2020, Texas Instruments Incorporated

Table 130. MMU_REPL Field Descriptions (continued)

Bit	Field	Type	Reset	Description
15-1	Reserved	R	0h	Reserved
0	POLICY	RW	0h	Replacement Policy

5.5.11.3 TDRR (offset = 404h)

Translation Debug Response Register

Table 131. TDRR Field Descriptions

Bit	Field	Type	Reset	Description
63	COMP	RW	0h	Translation Completion Bit
62-57	Reserved	R	0h	Reserved
56	PREF	RW	0h	Prefetchable
55-54	OUTER	RW	0h	Outer Cacheability
53-52	INNER	RW	0h	Inner Cacheability
51-50	MEMTYPE	RW	0h	Memory Type
49-48	SHARE	RW	0h	Shareability
47-40	Reserved	R	0h	Reserved
39-12	ADDR	RW	0h	Output Address, Faulting Stage-2 IPA
11-0	STATUS	RW	0h	Translation Response Status

5.5.11.4 TDFAR (offset = 405h)

Translation Debug Fault Address Register

Table 132. TDFAR Field Descriptions

Bit	Field	Type	Reset	Description
63-0	ADDR	RW	0h	Faulted Input Address

5.5.11.5 TLB_INV (offset = 408h)

TLB Invalidation Register

Table 133. TLB_INV Field Descriptions

Bit	Field	Type	Reset	Description
63-56	Reserved	R	0h	Reserved
55-48	ID	W	0h	ASID Value
47-43	Reserved	R	0h	Reserved
42-40	INV_TYPE	R	0h	Invalidation Type
39	ASID	W	0h	ASID Match
38	VA	W	0h	VA Match
37	LL	W	0h	Last Level Only
36-0	ADDR	W	0h	VA/IPA

Copyright © 2020, Texas Instruments Incorporated

5.5.11.6 TLB_INVC (offset = 40Ah)

TLB Invalidation Complete Register

Table 134. TLB_INVC Field Descriptions

Bit	Field	Type	Reset	Description
63-1	Reserved	R	0h	Reserved
0	COMP	RW	0h	Invalidation Completed

5.5.11.7 TLB_DBG (offset = 40Bh)

TLB Debug Access Register

Table 135. TLB_DBG Field Descriptions

Bit	Field	Type	Reset	Description
63-18	Reserved	R	0h	Reserved
17-16	TLB	W	0h	TLB Type
15-14	Reserved	R	0h	Reserved
13-12	WAY	W	0h	Way
11-7	Reserved	R	0h	Reserved
6-0	INDEX	W	0h	Index

5.5.11.8 TLB_DBG_DATA0 (offset = 40Ch)

TLB Debug Data 0 Register

Table 136. TLB_DBG_DATA0 Field Descriptions

Bit	Field	Type	Reset	Description
63	NS	R	0h	Non-Secure Page, NSTable accumulation
62-59	DS_SIZE	R	0h	Descriptor Size
58	DS_TYPE	R	0h	Descriptor Type
57-28	IADDR	R	0h	Input Address
27-20	VMID	R	0h	VMID
19-12	Reserved	R	0h	Reserved
11-4	ASID	R	0h	ASID
3	GBL	R	0h	Global Page
2	ROOT	R	0h	Root Context
1	SEC	R	0h	Security Context
0	VALID	R	0h	Valid Entry

5.5.11.9 TLB_DBG_DATA1 (offset = 40Dh)

TLB Debug Data 1 Register

Table 137. TLB_DBG_DATA1 Field Descriptions

Bit	Field	Type	Reset	Description
63-48	Reserved	R	0h	Reserved
47-46	SHARE	R	0h	Shareability
45-44	S2_LVL	R	0h	Stage 2 Level
43-40	S2_MEM_TYPE	R	0h	Stage 2 Memory Type
39-36	S2_PERM	R	0h	S2 Access Permissions

Copyright © 2020, Texas Instruments Incorporated

Table 137. TLB_DBG_DATA1 Field Descriptions (continued)

Bit	Field	Type	Reset	Description
35-33	S1_MEM_INDEX	R	0h	Stage 1 Memory Attribute Index
32-28	S1_PERM	R	0h	Stage 1 Access Permissions
27-0	OADDR	R	0h	Output Address

5.5.11.10 SCR (offset = 410h)

System Control Register

Table 138. SCR Field Descriptions

Bit	Field	Type	Reset	Description
63	HW	RW	0h	HW Mode
62-8	Reserved	R	0h	Reserved
7	INSTC	RW	0h	Stage 1 Instruction Caching
6	DATAC	RW	0h	Stage 1 Data Caching
5	FAULT	RW	0h	Caching faults in uTLBs
4	ASEL	RW	0h	ASID Selection
3	WXN	RW	0h	Writeable memory Execute Never
2	ENDIAN1	RW	0h	Stage 1 Region 1 Endian
1	ENDIAN0	RW	0h	Stage 1 Region 0 Endian
0	S1_EN	RW	0h	Enable Stage 1 Address Translation

5.5.11.11 TCR0 (offset = 411h)

Table Control Register 0

Table 139. TCR0 Field Descriptions

Bit	Field	Type	Reset	Description
63-17	Reserved	R	0h	Reserved
16-15	MEMTYPE	RW	0h	Memory Type of Translation Table Walks
14-13	COUTER	RW	0h	Outer Cacheability attribute of memory for Translation Table Walks
12-11	CINNER	RW	0h	Inner Cacheability attribute of memory for Translation Table Walks
10-9	SHARE	RW	0h	Shareability attribute of memory for Translation Table Walks
8-7	GRANULE	RW	0h	Translation Table Page Granule Size
6-1	TBL_SZ	RW	0h	Translation Table Region Size

Copyright © 2020, Texas Instruments Incorporated

Table 139. TCR0 Field Descriptions (continued)

0	WALK_EN	RW	0h	Enable Translation Table Walks

5.5.11.12 TCR1 (offset = 412h)

Table Control Register 1

Table 140. TCR1 Field Descriptions

Bit	Field	Type	Reset	Description
63-17	Reserved	R	0h	Reserved
16-15	MEMTYPE	RW	0h	Memory Type of Translation Table Walks
14-13	COUTER	RW	0h	Outer Cacheability attribute of memory for Translation Table Walks
12-11	CINNER	RW	0h	Inner Cacheability attribute of memory for Translation Table Walks
10-9	SHARE	RW	0h	Shareability attribute of memory for Translation Table Walks
8-7	GRANULE	RW	0h	Translation Table Page Granule Size
6-1	TBL_SZ	RW	0h	Translation Table Region Size
0	WALK_EN	RW	0h	Enable Translation Table Walks

5.5.11.13 TBR0 (offset = 413h)

Table Base Register 0

Table 141. TBR0 Field Descriptions

Bit	Field	Type	Reset	Description
63-56	Reserved	R	0h	Reserved
55-48	ASID	RW	0h	Application Space ID
47-40	Reserved	R	0h	Reserved
39-0	BADDR	RW	0h	Table Base Address

5.5.11.14 TBR1 (offset = 414h)

Table Base Register 1

Table 142. TBR1 Field Descriptions

Bit	Field	Type	Reset	Description
63-56	Reserved	R	0h	Reserved
55-48	ASID	RW	0h	Application Space ID
47-40	Reserved	R	0h	Reserved
39-0	BADDR	RW	0h	Table Base Address

Copyright © 2020, Texas Instruments Incorporated

5.5.11.15 MAR (offset = 415h)

Memory Attribute Register

Table 143. MAR Field Descriptions

Bit	Field	Type	Reset	Description
63-56	ATTR7	RW	0h	Memory Attribute Index 7
55-48	ATTR6	RW	0h	Memory Attribute Index 6
47-40	ATTR5	RW	0h	Memory Attribute Index 5
39-32	ATTR4	RW	0h	Memory Attribute Index 4
31-24	ATTR3	RW	0h	Memory Attribute Index 3
23-16	ATTR2	RW	0h	Memory Attribute Index 2
15-8	ATTR1	RW	0h	Memory Attribute Index 1
7-0	ATTR0	RW	0h	Memory Attribute Index 0

5.5.11.16 TDAR (offset = 416h)

Translation Debug Address Register

Table 144. TDAR Field Descriptions

Bit	Field	Type	Reset	Description
63-4	ADDR	W	0h	Input Address
3	INTEREST	W	0h	Interest Flag
2-1	ACC_TYPE	W	0h	Access Type
0	PRIV	W	0h	Supervisor Access

5.5.11.17 SCR_GS (offset = 420h)

System Control Register - Non-Secure Guest Supervisor

Table 145. SCR_GS Field Descriptions

Bit	Field	Type	Reset	Description
63	HW	RW	0h	HW Mode
62-8	Reserved	R	0h	Reserved
7	INSTC	RW	0h	Stage 1 Instruction Caching
6	DATAC	RW	0h	Stage 1 Data Caching
5	FAULT	RW	0h	Caching faults in uTLBs
4	ASEL	RW	0h	ASID Selection
3	WXN	RW	0h	Writeable memory Execute Never
2	ENDIAN1	RW	0h	Stage 1 Region 1 Endian
1	ENDIAN0	RW	0h	Stage 1 Region 0 Endian
0	S1_EN	RW	0h	Enable Stage 1 Address Translation

Copyright © 2020, Texas Instruments Incorporated

5.5.11.18 TCR0_GS (offset = 421h)

Table Control Register 0 - Non-Secure Guest Supervisor

Table 146. TCR0_GS Field Descriptions

Bit	Field	Type	Reset	Description
63-17	Reserved	R	0h	Reserved
16-15	MEMTYPE	RW	0h	Memory Type of Translation Table Walks
14-13	COUTER	RW	0h	Outer Cacheability attribute of memory for Translation Table Walks
12-11	CINNER	RW	0h	Inner Cacheability attribute of memory for Translation Table Walks
10-9	SHARE	RW	0h	Shareability attribute of memory for Translation Table Walks
8-7	GRANULE	RW	0h	Translation Table Page Granule Size
6-1	TBL_SZ	RW	0h	Translation Table Region Size
0	WALK_EN	RW	0h	Enable Translation Table Walks

5.5.11.19 TCR1_GS (offset = 422h)

Table Control Register 1 - Non-Secure Guest Supervisor

Table 147. TCR1_GS Field Descriptions

Bit	Field	Type	Reset	Description
63-17	Reserved	R	0h	Reserved
16-15	MEMTYPE	RW	0h	Memory Type of Translation Table Walks
14-13	COUTER	RW	0h	Outer Cacheability attribute of memory for Translation Table Walks
12-11	CINNER	RW	0h	Inner Cacheability attribute of memory for Translation Table Walks
10-9	SHARE	RW	0h	Shareability attribute of memory for Translation Table Walks
8-7	GRANULE	RW	0h	Translation Table Page Granule Size
6-1	TBL_SZ	RW	0h	Translation Table Region Size
0	WALK_EN	RW	0h	Enable Translation Table Walks

Copyright © 2020, Texas Instruments Incorporated

5.5.11.20 TBR0_GS (offset = 423h)

Table Base Register 0 - Non-Secure Guest Supervisor

Table 148. TBR0_GS Field Descriptions

Bit	Field	Type	Reset	Description
63-56	Reserved	R	0h	Reserved
55-48	ASID	RW	0h	Application Space ID
47-40	Reserved	R	0h	Reserved
39-0	BADDR	RW	0h	Table Base Address

5.5.11.21 TBR1_GS (offset = 424h)

Table Base Register 1 - Non-Secure Guest Supervisor

Table 149. TBR1_GS Field Descriptions

Bit	Field	Type	Reset	Description
63-56	Reserved	R	0h	Reserved
55-48	ASID	RW	0h	Application Space ID
47-40	Reserved	R	0h	Reserved
39-0	BADDR	RW	0h	Table Base Address

5.5.11.22 MAR_GS (offset = 425h)

Memory Attribute Register - Non-Secure Guest Supervisor

Table 150. MAR_GS Field Descriptions

Bit	Field	Type	Reset	Description
63-56	ATTR7	RW	0h	Memory Attribute Index 7
55-48	ATTR6	RW	0h	Memory Attribute Index 6
47-40	ATTR5	RW	0h	Memory Attribute Index 5
39-32	ATTR4	RW	0h	Memory Attribute Index 4
31-24	ATTR3	RW	0h	Memory Attribute Index 3
23-16	ATTR2	RW	0h	Memory Attribute Index 2
15-8	ATTR1	RW	0h	Memory Attribute Index 1
7-0	ATTR0	RW	0h	Memory Attribute Index 0

5.5.11.23 TDAR_GS (offset = 426h)

Translation Debug Address Register - Non-Secure Guest Supervisor

Table 151. TDAR_GS Field Descriptions

Bit	Field	Type	Reset	Description
63-4	ADDR	W	0h	Input Address
3	INTEREST	W	0h	Interest Flag
2-1	ACC_TYPE	W	0h	Access Type

Copyright © 2020, Texas Instruments Incorporated

Table 151. TDAR_GS Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	PRIV	W	0h	Supervisor Access

5.5.11.24 SCR_S (offset = 430h)

System Control Register - Non-Secure Supervisor

Table 152. SCR_S Field Descriptions

Bit	Field	Type	Reset	Description
63	HW	RW	0h	HW Mode
62-8	Reserved	R	0h	Reserved
7	INSTC	RW	0h	Stage 1 Instruction Caching
6	DATAC	RW	0h	Stage 1 Data Caching
5	FAULT	RW	0h	Caching faults in uTLBs
4	ASEL	RW	0h	ASID Selection
3	WXN	RW	0h	Writeable memory Execute Never
2	ENDIAN1	RW	0h	Stage 1 Region 1 Endian
1	ENDIAN0	RW	0h	Stage 1 Region 0 Endian
0	S1_EN	RW	0h	Enable Stage 1 Address Translation

5.5.11.25 TCR0_S (offset = 431h)

Table Control Register 0 - Non-Secure Supervisor

Table 153. TCR0_S Field Descriptions

Bit	Field	Type	Reset	Description
63-17	Reserved	R	0h	Reserved
16-15	MEMTYPE	RW	0h	Memory Type of Translation Table Walks
14-13	COUTER	RW	0h	Outer Cacheability attribute of memory for Translation Table Walks
12-11	CINNER	RW	0h	Inner Cacheability attribute of memory for Translation Table Walks
10-9	SHARE	RW	0h	Shareability attribute of memory for Translation Table Walks
8-7	GRANULE	RW	0h	Translation Table Page Granule Size
6-1	TBL_SZ	RW	0h	Translation Table Region Size
0	WALK_EN	RW	0h	Enable Translation Table Walks

Copyright © 2020, Texas Instruments Incorporated

5.5.11.26 TCR1_S (offset = 432h)

Table Control Register 1 - Non-Secure Supervisor

Table 154. TCR1_S Field Descriptions

Bit	Field	Type	Reset	Description
63-17	Reserved	R	0h	Reserved
16-15	MEMTYPE	RW	0h	Memory Type of Translation Table Walks
14-13	COUTER	RW	0h	Outer Cacheability attribute of memory for Translation Table Walks
12-11	CINNER	RW	0h	Inner Cacheability attribute of memory for Translation Table Walks
10-9	SHARE	RW	0h	Shareability attribute of memory for Translation Table Walks
8-7	GRANULE	RW	0h	Translation Table Page Granule Size
6-1	TBL_SZ	RW	0h	Translation Table Region Size
0	WALK_EN	RW	0h	Enable Translation Table Walks

5.5.11.27 TBR0_S (offset = 433h)

Table Base Register 0 - Non-Secure Supervisor

Table 155. TBR0_S Field Descriptions

Bit	Field	Type	Reset	Description
63-56	Reserved	R	0h	Reserved
55-48	ASID	RW	0h	Application Space ID
47-40	Reserved	R	0h	Reserved
39-0	BADDR	RW	0h	Table Base Address

5.5.11.28 TBR1_S (offset = 434h)

Table Base Register 1 - Non-Secure Supervisor

Table 156. TBR1_S Field Descriptions

Bit	Field	Type	Reset	Description
63-56	Reserved	R	0h	Reserved
55-48	ASID	RW	0h	Application Space ID
47-40	Reserved	R	0h	Reserved
39-0	BADDR	RW	0h	Table Base Address

Copyright © 2020, Texas Instruments Incorporated

5.5.11.29 MAR_S (offset = 435h)

Memory Attribute Register - Non-Secure Supervisor

Table 157. MAR_S Field Descriptions

Bit	Field	Type	Reset	Description
63-56	ATTR7	RW	0h	Memory Attribute Index 7
55-48	ATTR6	RW	0h	Memory Attribute Index 6
47-40	ATTR5	RW	0h	Memory Attribute Index 5
39-32	ATTR4	RW	0h	Memory Attribute Index 4
31-24	ATTR3	RW	0h	Memory Attribute Index 3
23-16	ATTR2	RW	0h	Memory Attribute Index 2
15-8	ATTR1	RW	0h	Memory Attribute Index 1
7-0	ATTR0	RW	0h	Memory Attribute Index 0

5.5.11.30 TDAR_S (offset = 436h)

Translation Debug Address Register - Non-Secure Supervisor

Table 158. TDAR_S Field Descriptions

Bit	Field	Type	Reset	Description
63-4	ADDR	W	0h	Input Address
3	INTEREST	W	0h	Interest Flag
2-1	ACC_TYPE	W	0h	Access Type
0	PRIV	W	0h	Supervisor Access

5.5.11.31 VCR (offset = 460h)

Virtualization Control Register

Table 159. VCR Field Descriptions

Bit	Field	Type	Reset	Description
63-6	Reserved	R	0h	Reserved
5	ID	RW	0h	Stage 2 Instruction Cache Disable
4	CD	RW	0h	Stage 2 Data Cache Disable
3	DC	RW	0h	Default Cacheable Mode
2	PROT	RW	0h	Protected Table Walk
1	ENDIAN	RW	0h	Stage 2 Endian
0	S2_EN	RW	0h	Enable Stage 2 Address Translation

Copyright © 2020, Texas Instruments Incorporated

5.5.11.32 VTCR (offset = 461h)

Virtualization Table Control Register

Table 160. VTCR Field Descriptions

Bit	Field	Type	Reset	Description
63-19	Reserved	R	0h	Reserved
18-17	SLEVEL	RW	0h	Stage 2 Starting Translation Level
16-15	MEMTYPE	RW	0h	Memory Type of Translation Table Walks
14-13	COUTER	RW	0h	Outer Cacheability attribute of memory for Translation Table Walks
12-11	CINNER	RW	0h	Inner Cacheability attribute of memory for Translation Table Walks
10-9	SHARE	RW	0h	Shareability attribute of memory for Translation Table Walks
8-7	GRANULE	RW	0h	Translation Table Page Granule Size
6-1	TBL_SZ	RW	0h	Translation Table Region Size
0	Reserved	R	0h	Reserved

5.5.11.33 VTBR (offset = 462h)

Virtualization Table Base Register

Table 161. VTBR Field Descriptions

Bit	Field	Type	Reset	Description
63-56	Reserved	R	0h	Reserved
55-48	VMID	RW	0h	Virtual Machine ID
47-40	Reserved	R	0h	Reserved
39-0	BADDR	RW	0h	Table Base Address

5.5.11.34 VTDAR (offset = 463h)

Virtualization Translation Debug Address Register

Table 162. VTDAR Field Descriptions

Bit	Field	Type	Reset	Description
63-4	ADDR	W	0h	Input Address
3	INTEREST	W	0h	Interest Flag
2-1	ACC_TYPE	W	0h	Access Type
0	PRIV	W	0h	Supervisor Access

Copyright © 2020, Texas Instruments Incorporated

5.5.11.35 2.1.35 L1D_UTLB_MATCH (offset = 480h)

Indexed uTLB Page Match Entry

Table 163. L1D_UTLB_MATCH Field Descriptions

Bit	Field	Type	Reset	Description
63	VALID	RW	0h	Entry Valid
62	SECURE	RW	0h	Secure Page
61	ROOT	RW	0h	Root Page
60	GBL	RW	0h	Global Page
59-52	VMID	RW	0h	VMID
51-44	ASID	RW	0h	ASID
43	Reserved	R	0h	Reserved
42-40	PG_SIZE	RW	0h	Page Size
39-37	Reserved	R	0h	Reserved
36-0	IADDR	RW	0h	Input Address

5.5.11.36 2.1.35 L1D_UTLB_ATTR (offset = 482h)

Indexed uTLB Page Attribute Entry

Table 164. L1D_UTLB_ATTR Field Descriptions

Bit	Field	Type	Reset	Description
63	VALID	RW	0h	Entry Valid
62-51	STATUS	RW	0h	TLB Status
50-42	ACC_PERM	RW	0h	Access Permissions
41	PG_NSEC	RW	0h	Page Non-Secure
40-32	MEMTYPE	RW	0h	Page Memory Type
31-28	Reserved	R	0h	Reserved
27-0	OADDR	RW	0h	Output Address

5.5.11.37 2.1.35 L1D_UTLB_REPL (offset = 484h)

uTLB Replacement Policy Register

Table 165. L1D_UTLB_REPL Field Descriptions

Bit	Field	Type	Reset	Description
63-32	Reserved	R	0h	Reserved
31-16	LFSR	RW	0h	LFSR Value
15-1	Reserved	R	0h	Reserved
0	POLICY	RW	0h	Replacement Policy

5.5.11.38 2.1.35 L1P_UTLB_MATCH (offset = 488h)

Indexed uTLB Page Match Entry Register

Table 166. L1P_UTLB_MATCH Field Descriptions

Bit	Field	Type	Reset	Description
63	VALID	RW	0h	Entry Valid
62	SECURE	RW	0h	Secure Page
61	ROOT	RW	0h	Root Page
60	GBL	RW	0h	Global Page

Copyright © 2020, Texas Instruments Incorporated

Table 166. L1P_UTLB_MATCH Field Descriptions (continued)

Bit	Field	Type	Reset	Description
59-52	VMID	RW	0h	VMID
51-44	ASID	RW	0h	ASID
43	Reserved	R	0h	Reserved
42-40	PG_SIZE	RW	0h	Page Size
39-37	Reserved	R	0h	Reserved
36-0	IADDR	RW	0h	Input Address

5.5.11.39 2.1.35 L1P_UTLB_ATTR (offset = 48Ah)

Indexed uTLB Page Attribute Entry Register

Table 167. L1P_UTLB_ATTR Field Descriptions

Bit	Field	Type	Reset	Description
63	VALID	RW	0h	Entry Valid
62-51	STATUS	RW	0h	TLB Status
50-42	ACC_PERM	RW	0h	Access Permissions
41	PG_NSEC	RW	0h	Page Non-Secure
40-32	MEMTYPE	RW	0h	Page Memory Type
31-28	Reserved	R	0h	Reserved
27-0	OADDR	RW	0h	Output Address

5.5.11.40 2.1.35 L1P_UTLB_REPL (offset = 48Ch)

uTLB Replacement Policy Register

Table 168. L1P_UTLB_REPL Field Descriptions

Bit	Field	Type	Reset	Description
63-32	Reserved	R	0h	Reserved
31-16	LFSR	RW	0h	LFSR Value
15-1	Reserved	R	0h	Reserved
0	POLICY	RW	0h	Replacement Policy

5.5.11.41 2.1.35 2.1.41 SE0_UTLB_MATCH (offset = 490h)

Indexed uTLB Page Match Entry Register

Table 169. SE0_UTLB_MATCH Field Descriptions

Bit	Field	Type	Reset	Description
63	VALID	RW	0h	Entry Valid
62	SECURE	RW	0h	Secure Page
61	ROOT	RW	0h	Root Page
60	GBL	RW	0h	Global Page
59-52	VMID	RW	0h	VMID
51-44	ASID	RW	0h	ASID
43	Reserved	R	0h	Reserved
42-40	PG_SIZE	RW	0h	Page Size
39-37	Reserved	R	0h	Reserved
36-0	IADDR	RW	0h	Input Address

Copyright © 2020, Texas Instruments Incorporated

5.5.11.42 2.1.35 2.1.41 SE0_UTLB_ATTR (offset = 492h)

Indexed uTLB Page Attribute Entry Register

Table 170. SE0_UTLB_ATTR Field Descriptions

Bit	Field	Type	Reset	Description
63	VALID	RW	0h	Entry Valid
62-51	STATUS	RW	0h	TLB Status
50-42	ACC_PERM	RW	0h	Access Permissions
41	PG_NSEC	RW	0h	Page Non-Secure
40-32	MEMTYPE	RW	0h	Page Memory Type
31-28	Reserved	R	0h	Reserved
27-0	OADDR	RW	0h	Output Address

5.5.11.43 2.1.35 2.1.41 SE0_UTLB_REPL (offset = 494h)

uTLB Replacement Policy Register

Table 171. SE0_UTLB_REPL Field Descriptions

Bit	Field	Type	Reset	Description
63-32	Reserved	R	0h	Reserved
31-16	LFSR	RW	0h	LFSR Value
15-1	Reserved	R	0h	Reserved
0	POLICY	RW	0h	Replacement Policy

5.5.11.44 2.1.35 2.1.41 SE1_UTLB_MATCH (offset = 498h)

Indexed uTLB Page Match Entry Register

Table 172. SE1_UTLB_MATCH Field Descriptions

Bit	Field	Type	Reset	Description
63	VALID	RW	0h	Entry Valid
62	SECURE	RW	0h	Secure Page
61	ROOT	RW	0h	Root Page
60	GBL	RW	0h	Global Page
59-52	VMID	RW	0h	VMID
51-44	ASID	RW	0h	ASID
43	Reserved	R	0h	Reserved
42-40	PG_SIZE	RW	0h	Page Size
39-37	Reserved	R	0h	Reserved
36-0	IADDR	RW	0h	Input Address

5.5.11.45 2.1.35 2.1.41 SE1_UTLB_ATTR (offset = 49Ah)

Indexed uTLB Page Attribute Entry Register

Table 173. SE1_UTLB_ATTR Field Descriptions

Bit	Field	Type	Reset	Description
63	VALID	RW	0h	Entry Valid
62-51	STATUS	RW	0h	TLB Status
50-42	ACC_PERM	RW	0h	Access Permissions
41	PG_NSEC	RW	0h	Page Non-Secure

Copyright © 2020, Texas Instruments Incorporated

Table 173. SE1_UTLB_ATTR Field Descriptions (continued)

Bit	Field	Type	Reset	Description
40-32	MEMTYPE	RW	0h	Page Memory Type
31-28	Reserved	R	0h	Reserved
27-0	OADDR	RW	0h	Output Address

5.5.11.46 2.1.35 2.1.41 SE1_UTLB_REPL (offset = 49Ch)

uTLB Replacement Policy Register

Table 174. SE1_UTLB_REPL Field Descriptions

Bit	Field	Type	Reset	Description
63-32	Reserved	R	0h	Reserved
31-16	LFSR	RW	0h	LFSR Value
15-1	Reserved	R	0h	Reserved
0	POLICY	RW	0h	Replacement Policy

Copyright © 2020, Texas Instruments Incorporated

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2020, Texas Instruments Incorporated