Ncore 3.6 Supplemental Architecture Specification

Rev:0.80, January the 11th 2024March 18, 2024

Arteris® Ncore 3.6 Supplemental Architecture Specification

Copyright © 2020 Arteris© or its affiliates. All rights reserved.

**Release Information**

|  |  |  |  |
| --- | --- | --- | --- |
| *Version* | *Editor* | *Change* | *Date* |
| 0.3 | MK | Initial Document inputs from Memory tagging Arch document by MF | 09/21/2022 |
| 0.5 | MK | Update based on feedback after initial review | 10/05/2022 |
| 0.6 | MK | Update after detailed review feedback | 10/18/2022 |
| 0.62 | MK | Update on MakeReadUnique non-exclusive mapping | 10/25/2022 |
| 0.64 | CCW | Update on DVM | 01/24/2023 |
| 0.66 | CCW | Add error handling of new added transactions | 02/13/2023 |
| 0.67 | MF | Added parameters for cache and snoop filter replacement policy | 02/15/2023 |
| 0.68 | CCW | Add improve RB ID management | 04/04/2023 |
| 0.69 | CCW | Add remote/local cache agents parameter to DCE, and modify the snooping condition of sharer promotion feature  Add WriteEvictOrEvict support  Add Cbusy support | 05/16/2023 |
| 0.7 | CCW | Add more description to several features | 07/11/2023 |
| 0.71 | MF | CONC-12399, add 4th network control in Trace Capture Control Register | 08/15/2023 |
| 0.72 | SD | CONC-12299, fnCsrAccess changes added in section 5 | 08/17/2023 |
| 0.73 | SD | CONC-12203, update parameter "DvmVersionSupport" in section 1.1 | 08/21/2023 |
| 0.74 | SD | CONC-12436 non coherent exclusive attribute in section 2.4.1 | 08/24/2023 |
| 0.75 | SD | CONC-12061 adding NSIDR register description in section 1.2 | 08/24/2023 |
| 0.76 | SD | DVM Revision Register added in DVE register space see table 9 | 08/25/2023 |
| 0.77 | SD | DVM Revision default value changed to 8.0  Snoop handling modified in section 2.1.2 updated to fix CONC-13156 | 11/6/2023 |
| 0.78 | SD | NS bit check for CSR access supported | 11/16/2023 |
| 0.79 | SD | DII AXI ID mapping new parameter | 1/11/2024 |
| 0.80 | HL | Update and further clarify how stash works in section 2.1.2 with CHI-B and CHI-E CAIU per described in [[CONC-13156](https://arterisip.atlassian.net/browse/CONC-13156)], [[CONC-14058](https://arterisip.atlassian.net/browse/CONC-14058)] and [[CONC-14033](https://arterisip.atlassian.net/browse/CONC-14058)] | 3/15/2024 |
| Legend: | MK | Mohammed |  |
|  | MF  CCW | Michael Frank  Cheng Chung Wang |  |
|  | SD  HL | Said Derradji  Hao Luan |  |
|  | Xx | Whoever else edited this document |  |

**Confidential Proprietary Notice**

This document is CONFIDENTIAL AND PROPRIETARY to Arteris, Inc. or its applicable subsidiary or affiliate (collectively or as applicable, “Arteris” or “Arteris IP”), and any use by you is subject to the terms of the agreement between you and Arteris IP or the terms of the agreement between you and the party authorized by Arteris IP to disclose this document to you.

This document is also protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arteris IP.  **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.  You are prohibited from altering or deleting this notice from any use by you of this document.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information: (i) for the purposes of determining whether implementations infringe any third party patents;(ii) for developing technology or products which avoid any of Arteris IP’s intellectual property; or (iii) as a reference for modifying existing patents or patent applications or creating any continuation, continuation in part, or extension of existing patents or patent applications; or (iv) for generating data for publication or disclosure to third parties, which compares the performance or functionality of the Arteris IP technology described in this document with any other products created by you or a third party, without obtaining Arteris IP’s prior written consent.

THIS DOCUMENT IS PROVIDED “AS IS”.  ARTERIS IP PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arteris IP makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.  This document may include technical inaccuracies or typographical errors.  Arteris IP makes no representations or warranties against the risk or presence of same.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARTERIS IP BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARTERIS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be solely responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arteris IP’s customers is not intended to create or refer to any partnership relationship with any other company. Arteris IP may make changes to this document at any time and without notice. If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arteris IP, then the click-through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the agreement shall prevail.

The Arteris IP name and corporate logo, and words marked with ® or ™ are registered trademarks or trademarks of Arteris (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arteris IP’s trademark usage guidelines, available from Arteris IP upon request by emailing to contracts@arteris.com.

Copyright © 2020 Arteris Inc. or its applicable subsidiary or affiliate. All rights reserved.

**Confidentiality Status**

This document is Confidential and Proprietary. This document may only be used and distributed in accordance with the terms of the agreement entered into by Arteris IP and the party that Arteris IP delivered this document to.

**Product Status**

The information in this document is ***Preliminary***.

**Web Address**

[http://www.arteris.com](http://www.arteris.com/)

Table of Contents

1 Introduction 10

1.1 Parameters 12

1.2 Global Register Block modifications 15

1.2.1 Ncore Subsystem Identification Register(NSIDR) 15

2 CHI-E support 15

2.1 Request bus 15

2.1.1 Command transactions 15

2.1.2 Snoop transactions 16

2.1.3 New CHI E request fields 16

2.2 Data bus 17

2.3 Response bus 17

2.4 Exclusive transaction support 17

2.4.1 Non coherent exclusive support 18

2.5 DVM support 18

2.6 Error Handling 23

2.7 Constraint of Mixed Coherent and Non-coherent Transactions 24

2.8 Improve RB ID Management 24

2.9 Local/Remote Cache Agents (pending) 25

2.10 WriteEvictOrEvict CHI.E command support (pending) 26

2.11 Cbusy Support (pending) 26

2.12 Trace and Debug Support 28

2.12.1 Trace Capture 28

2.12.1.1 Capture Control Register (xCCTRLR) 28

3 SecSubRows Feature 29

4 Slow SRAM Fix 30

5 CSR access security 31

5.1 Parameters 31

5.2 Detailed Description 31

5.3 Ncore register space attribute register (xNRSAR) 31

6 Opens 32

7 Glossary 33

8 Notes 34

Table of Figures

**No table of figures entries found.**

Table of Tables

[Table 1: CHI Evolution 11](#_Toc143874375)

[Table 2: fnNAtiveInterface Parameter 12](#_Toc143874376)

[Table 3: enHalfSpeedDataSRAM Parameter 12](#_Toc143874377)

[Table 4: enSRAMPipe Parameter 12](#_Toc143874378)

[Table 5: cacheReplPolicy Parameter 12](#_Toc143874379)

[Table 6: cacheReplStateWidth Parameter 13](#_Toc143874380)

[Table 7: RemoteCachingAgents Parameter 13](#_Toc143874381)

[Table 8: LocalCachingAgents Parameter 13](#_Toc143874382)

[Table 9: DVMVersionSupport Parameter 13](#_Toc143874383)

[Table 10: NRR Subsystem Identification Register 15](#_Toc143874384)

[Table 11: CHI E request mapping 16](#_Toc143874385)

[Table 12 Command request fields 16](#_Toc143874386)

[Table 13: Snoop request fields 17](#_Toc143874387)

[Table 14: Data fields 17](#_Toc143874388)

[Table 15: Response fields 17](#_Toc143874389)

[Table 16: CHI E Exclusive request mapping 18](#_Toc143874390)

[Table 17: DVM command mapping 19](#_Toc143874391)

[Table 18: DVM data mapping 20](#_Toc143874392)

[Table 19: DVM 1st snoop mapping 21](#_Toc143874393)

[Table 20: DVM 2nd snoop mapping 22](#_Toc143874394)

[Table 21 DVM version capability 23](#_Toc143874395)

[Table 22 DCE Snooping Decision of REmote/Local Caching Agents 1st 25](#_Toc143874396)

[Table 23 DCE Snooping Decision of REmote/Local Caching Agents 2nd 26](#_Toc143874397)

[Table 24 WriteEvictOrEvict decision 26](#_Toc143874398)

Preface

This preface introduces the Arteris® Network-on-Chip Hierarchical Coherency Engine Architecture Specification.

**About this document**

This technical document is for the Arteris Network-on-Chip Hierarchical Coherency Engine Architecture. It describes the subsystems and their function along with the system’s interactions with the external subsystems. It also provides reference documentation and contains programming details for registers.

**Product revision status**

*TBD*

**Intended audience**

This manual is for system designers, system integrators, and programmers who are designing or programming a System-on-Chip (SoC) that uses or intend to use the Arteris Network-on-Chip Hierarchical Coherency System (AnoC-HCS).

**Using this document**

*TBD*

**Glossary**

The Arteris© Glossary is a list of terms used in Arteris© documentation, together with definitions for those terms. The Arteris© Glossary does not contain terms that are industry standard unless the Arteris© meaning differs from the generally accepted meaning.

**Typographic conventions**

*italic*

Introduces special terminology, denotes cross-references, and citations.

**Bold**

Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

Monospace

Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

*Monospace italic*

Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name. monospace italic Denotes arguments to monospace text where the argument is to be replaced by a specific value. Monospace bold Denotes language keywords when used outside example code.

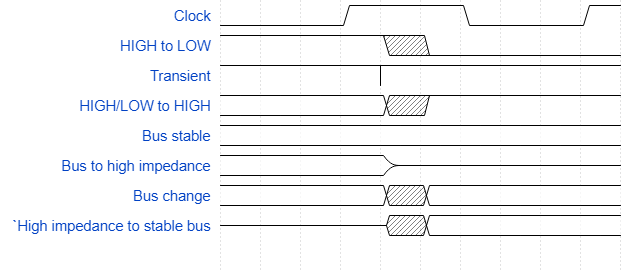
SMALL CAPITALS

Used in body text for a few terms that have specific technical meanings, that are defined in the Arteris® Glossary. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

**Timing diagrams**

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



**Signals**

The signal conventions are:

**Signal level**

The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW.   
Asserted means:

* HIGH for active-HIGH signals.
* LOW for active-LOW signals.

**Lowercase n**

At the start or end of a signal name denotes an active-LOW signal.

**Additional reading**

This book contains information that is specific to this product. See the following documents for other relevant information.

History of the World II, Mel Brooks.

# Introduction

This is a supplemental specification to specify additional features for Ncore 3.6. This is a stopgap product to support new ARM v9a processors, changes and features are limited to make Ncore functionally compatible with optional features disabled. New features added are

* CHI-E transaction only support
  + Emulation of combined commands and new CHI E commands
  + Emulation of state change exclusives
  + No data source information
  + No stash group support
  + No persistence group support
  + No DBID order support
  + No split transaction support
  + No MTE
  + No MPAM
* Slow SRAM fix
  + Only DMI data SRAMS at half clock frequency
  + Ott data SRAM and tag SRAM with added pipe stages
* Deprecated CHI-A support
* A mixed system with CHI-B and CHI-E will be supported.

Architectural changes recommended in this spec are focused on limiting design and verification changes to:

* CHI-AIU for CHI-E transactions
* DVE, CHI AIU and IOAIU for DVM mapping
* ProxyCache, DCE, DMI for improved replacement algorithms
* DMI for slow SRAM support.

These changes are not focused on improving power performance or area compared to Ncore 3.4 design and architecture; they are more focused on adding minimum needed features.

Evolution of CHI is shown in Table 1

|  | A 🡪 B | B 🡪 C | C 🡪 D | D 🡪 E |
| --- | --- | --- | --- | --- |
| Performance/ Protocol | * + Retry Transaction   + Forwarding Snoops   + CleanSharedPersist   + Data Stashing   + Prefetch Req   + Far atomics | * + Response after first Data packet   + Separate Data-Only & Non-Data Response   + Combined CompAck with WriteData   + Persistent CMO | * + Persistent CMO with two-part response   + Deep Persistent Cache Maintenance   + OWO for WriteNoSnp | * + MakeReadUnique   + Write with optional data   + Write Zero with no data   + Two-part Stash transactions   + SnpQuery   + DBIDRespReqOrd   + Direct Write-data transfer   + Combined Write and CMO   + Combined Write and PCMO   + MakeReadUnique with Exclusive access   + SnpPreferUnique & SnpPreferUniqueFwd   + Permitted Fwd-snoops to be handled as non-fwd snoop   + Enhance Exclusive Read Transactions   + SLC replacement hint   **Responses**   * + StashDoneResp   + CompStashDoneResp   + CompCMOResp   + TagMatchResp |
| Interface Changes |  |  | * + Increased TXID width to 10 bits   + Data Source Indication | * + Increased TXID width to 12 bits   + Request Channel 7 bits (New transactions)   + RSP Channel OpCode 5 bits (New Responses)   + DBID increased to 12 bits (More buffers)   + Multiple CHI interfaces per block   + Replicating channels on single interfaces |
| ARM Architecture | * + Armv8.1-A Large System Extensions   + Improved virtualization   + 52-bit physical address space | * + DVM enhancements (Arm v8.2)   Change in Prefetch semantics (TgtID must be 0) | * + ICache Invalidation broadcast signal (DVM)   + MPAM support added   + CBusy (feedback for QoS)   + Persistance GroupID | * + MTE Support   + DVM updates (v8.4 and v8.5) – invalidate TLB by address range   + StashGroupID   + TagGroupID |
| RAS | * + Data Poisoning   + Data Check   + Enables common error signaling, logging, and reporting |  | * + Interface Parity |  |

Table 1: CHI Evolution

## Parameters

Following new parameters have been introduced and updated.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name: fnNativeInterface | | Type: Enum | Visibility: user Settable | |
|  | **Architecture** | **Release** | | **Default** |
|  | *Valid Values* | *Valid Values* | |  |
| Value | ACE, ACE-LITE, ACE-LITE-E, AXI4, CHI-A, CHI-B, CHI-E | ACE, ACE-LITE, ACE-LITE-E, AXI4, CHI-B, CHI-E | | CHI B |
| Constraint |  | | | |
| Customer Description | Selects native interface type for a CAIU | | | |
| Engineering Description | Selects native interface type  CHIA is deprecated in 3.6  AXI4 results in NCAIU with base modules of IOAIU  ACE results in CAIU with base module of IOAIU  CHI\* result in CAIU with base module of CHI AIU | | | |

Table 2: fnNAtiveInterface Parameter

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name: enHalfSpeedDataSRAM | | | Type: Int | | Visibility: user Settable | |
|  | **Architecture** | | **Release** | | | **Default** |
|  | *Min* | *Max* | *Min* | *Max* | |  |
| Value | 0 | 1 | 0 | 1 | | 0 |
| Constraint | Only available in DMI with SMC enabled | | | | | |
| Customer Description | Enable SMC data SRAM to run at half clock frequency. Enabling this will add a couple of cycle latency and may affect BW in the case of partial cache line accesses. | | | | | |
| Engineering Description | This applies to only DMI with SMC enabled | | | | | |

Table 3: enHalfSpeedDataSRAM Parameter

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name: enSRAMPipe | | | Type: Int | | Visibility: user Settable | |
|  | **Architecture** | | **Release** | | | **Default** |
|  | *Min* | *Max* | *Min* | *Max* | |  |
| Value | 0 | 1 | 0 | 1 | | 0 |
| Constraint | Available in DMI with SMC enabled  Available in IOAIU for OTT data SRAM  In the case of DMI this must be enabled if enHalfSpeedDataSRAM is set. | | | | | |
| Customer Description | Enable SRAM pipe. Enabling this will add a cycle latency. | | | | | |
| Engineering Description | Enable SRAM pipe. Enabling this will add a cycle latency. In the case of DMI this must be enabled if enHalfSpeedDataSRAM is set. | | | | | |

Table 4: enSRAMPipe Parameter

For the caches and snoop filters, the following parameters need to be updated or added

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name: cacheReplPolicy | | Type: Enum | Visibility: user Settable | |
|  | **Architecture** | **Release** | | **Default** |
|  | *Enum* | *Enum* | |  |
| Value | RANDOM, NRU, SRRIP, pLRU | RANDOM, NRU, pLRU | | RANDOM |
| Constraint | Available in DMI with SMC enabled  Available in IOAIU with ProxyCache enabled  Available in DCE with Snoop Filters (only Random and pLRU are available) | | | |
| Customer Description | Cache Replacement Policy | | | |
| Engineering Description | Depending on the selected policy, a dependent parameter cacheReplStateWidth needs to be calculated . That parameter defines the number of bits required to represent the current position in the replacement algorithm for each cacheline in the set | | | |

Table 5: cacheReplPolicy Parameter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name: *cacheReplStateWidth* | | Type: Int | Visibility: derived | |
|  | **Architecture** | **Release** | | **Default** |
|  | *Int* | *Int* | |  |
| Value | 0, 1, 2 | 0, 1 | | 0 |
| Constraint | Available in DMI with SMC enabled  Available in IOAIU with ProxyCache enabled  Available in DCE with Snoop Filters | | | |
| Customer Description | Cache Replacement Policy | | | |
| Engineering Description | This parameter value is derived based on the cacheReplPolicy parameter:  RANDOM: 0, NRU: 1, SRRIP: 2, pLRU: 1  **Note:** For the SSRIP implementation we may want to consider an optimization - reserving state 00 as indication of an invalid cache line can save one of the standard state bits that indicate valid, dirty. | | | |

Table 6: cacheReplStateWidth Parameter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name: *RemoteCachingAgents* | | Type: array of strings | Visibility: User | |
|  | **Architecture** | **Release** | | **Default** |
|  | array of strings | array of strings | |  |
| Value |  |  | |  |
| Constraint | Available in DCE  And only valid if associated caching agents is connected to the DCE | | | |
| Customer Description | Specify if the caching agents is considered remote to the DCE  Eg. [AIU0, AIU2] | | | |
| Engineering Description | Derive the value into array of integers for DCE, to indicate if the corresponding caching agent is remote (sync up with DCE’s Jason file parameter) | | | |

Table 7: RemoteCachingAgents Parameter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name: *LocalCachingAgents* | | Type: array of strings | Visibility: User | |
|  | **Architecture** | **Release** | | **Default** |
|  | array of strings | array of strings | |  |
| Value |  |  | |  |
| Constraint | Available in DCE  And only valid if associated caching agents is connected to the DCE | | | |
| Customer Description | Specify if the caching agents is considered local to the DCE  Eg. [AIU1] | | | |
| Engineering Description | Derive the value into array of integers for DCE, to indicate if the corresponding caching agent is local (sync up with DCE’s Jason file parameter) | | | |

Table 8: LocalCachingAgents Parameter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name: *DVMVersionSupport* | | Type: Int | Visibility: User | |
|  | **Architecture** | **Release** | | **Default** |
|  | *Int* | *Int* | |  |
| Value | {8,0}, {8,1}, {8,4} | {8,0}, {8,1}, {8,4} | | {8,4} |
| Format | The version number is encoded as the concatenation of two 4 bit integers {4'd,4'd}. The first integer represents the main DVM version and the second the subversion number. For example: DVM\_v8.1 the version number is {4'd8,4'd1} or {8,1} | | | |
| Constraint | Refer to table 20, and based on the interface find the maximum common capability of all AIU interface. | | | |
| Customer Description | DVM version capability of the system. The value is suggested for the User to configure the system. | | | |
| Engineering Description | Pass the parameter to register DVEUDVMRR “DVM Revision Register” in DVE register space | | | |

Table 9: DVMVersionSupport Parameter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name: addressBits | | Type: Array | Visibility: user Settable | |
|  | **Architecture** | **Release** | | **Default** |
|  | *Integer Array* | *Integer Array* | |  |
| Value |  |  | |  |
| Constraint | Applied only to DII. | | | |
| Customer Description | This feature specifies an array of integers that consists of the bit indexes in an AXI transaction that can be used to encode the corresponding AXI ID. | | | |
| Engineering Description | Selected address bits will be used to generate corresponding axi ID.  Minimum Array Size: 0  Maximum Array Size: min(wArId, wAwid)  Default Array Size: 0 (no entries)  Array entry integer range: 0 to (system.concertocparams.wAddr-1)  addressBits = addressIdMap.addressBits  For Reads:  1. Fill the bottom bits with the corresponding address bit for example:  a. Arid[0] = addressBits[0]  b. Arid[1] = addressBits[1] …  2. If addressBits is an empty array this step is skipped. If there are more address bits defined than the ID width the bits above the ID width are ignored.  3. Starting from where above left off fill the rest of the bits with the bits above the cacheline. For example:  a. Arid[x] = addressBits[wCachelineOffset]  b. Arid[x+1] = addressBits[wCachelineOffset+1] …  For Writes:  1. Fill the bottom bits with the corresponding address bit for example:  a. Awid[0] = addressBits[0]  b. Awid[1] = addressBits[1] …  2. If addressBits is an empty array this step is skipped. If there are more address bits defined than the ID width the bits above the ID width are ignored.  3. Starting from where above left off fill the rest of the bits with the bits above the cacheline plus two. For example:  a. Awid[x] = addressBits[wCachelineOffset+2]  b. Awid[x+1] = addressBits[wCachelineOffset+1+2] … | | | |

Table 10: addressBits Parameter

## Global Register Block (GRB)modifications

### Ncore Subsystem Identification Register(NSIDR)

[Offset 0xFFc]

The parameter implVerId is expanded from 12 bits to 16 bits to include a 4 digits large Ncore version number.

Release version is provided as 16 bits in NSIDR register. The new description of NSIDR is hereunder:

|  |  |  |  |
| --- | --- | --- | --- |
| Bits | Field Name | Access | Description |
| 15:0 | RelVer | RO | Release Version |
| 19:16 | CachelineOffset | RO | Directory Cacheline Offset. This field indicates the width of the directory cache line offset. The number of bits in the directory cache line offset equals the value of this field + 5, e.g. for a 6-bit offset the value of this field equals 1. The size in bytes of the directory cache line is equal to two to the power of the offset. |
| 27:20 | nSnoopFilters | RO | Number of Snoop Filters. This field indicates the number of snoop filters in the coherent derived. The value of this field equals the number of snoop filters minus one. |
| 31:28 | Rsvd | RO | Reserved |

Table 11: NRR Subsystem Identification Register

# CHI-E support

This chapter goes over how different CHI-E features are mapped on to concerto protocol.

## Request bus

This section describes how the CHI request bus is integrated into current Ncore protocol, changes in this section are primarily limited to AIUs. Note that DVMs are covered in a section 2.5.

### Command transactions

The mapping of native CHI-E commands that have been added in addition to CHI-B is shown in Table 12. CHI AIU is expected to meet the CHI E interface requirements as per the CHI E spec based on the native command issued by the CHI agent, the mapping here is for internal Ncore handling of the commands.

|  |  |  |
| --- | --- | --- |
| CHI E Native Command | Concerto Commands | Notes |
| ReadNoSnpSep | N/A | This command can only be issued by a Home Node. Not Issued by DSU |
| CleanSharedPersistSep | CmdClnShPsist |  |
| MakeReadUnique (Not Excl) | CmdRdUnq |  |
| MakeReadUnique (Excl) | CleanUnique + CmdRdNshDty | See details at chapter 2.4 |
| WriteEvictOrEvict |  | See details at chapter 2.10 |
| WriteUniqueZero | CmdWrUnqFull | Emulate a DtwReq with 64 bytes of data as Zeros (DtwDataFullDty) |
| WriteNoSnpZero | CmdWrNCFull | Emulate a DtwReq with 64 bytes of data as Zeros (DtwDataFullDty) |
| StashOnceSepShared | N/A | Disable in DSU by setting appropriate bits in **CLUSTERECTLR\_EL1** |
| StashOnceSepUnique | N/A | Disable in DSU by setting appropriate bits in **CLUSTERECTLR\_EL1** |
| ReadPreferUnique | CmdRdNshDty | Treat it as Read not shared dirty |
| WriteNoSnpFullCleanSh | CmdWrNCFull + CmdClnVld | Combine cmd: Issue two commands, write followed by CMO |
| WriteNoSnpFullCelanInv | CmdWrNCFull + CmdClnInv | Combine cmd: Issue two commands, write followed by CMO |
| WriteNoSnpFullCleanShPerSep | CmdWrNCFull + CmdClnShPsist | Combine cmd: Issue two commands, write followed by CMO |
| WriteUniqueFullCleanSh | N/A | Not issued by DSU |
| WriteUniqueFullCleanShPerSep | N/A | Not issued by DSU |
| WriteBackFullCleanSh | CmdWrBkFull + CmdClnVld | Combine cmd: Issue two commands, write followed by CMO |
| WriteBackFullCelanInv | CmdWrBkFull + CmdClnInv | Combine cmd: Issue two commands, write followed by CMO |
| WriteBackFullCleanShPerSep | CmdWrBkFull + CmdClnShPsist | Combine cmd: Issue two commands, write followed by CMO |
| WriteCleanFullCleanSh | CmdWrClnFull + CmdClnVld | Combine cmd: Issue two commands, write followed by CMO |
| WriteCleanFullCleanShPreSep | CmdWrClnFull + CmdClnShPsist | Combine cmd: Issue two commands, write followed by CMO |
| WriteNoSnpPtlCleanSh | N/A | Not issued by DSU |
| WriteNoSnpPtlCleanInv | N/A | Not issued by DSU |
| WriteNoSnpPtlCleanShPerSep | N/A | Not issued by DSU |
| WriteUniquePtlCleanSh | N/A | Not issued by DSU |
| WriteUniquePtlCleanShPerSep | N/A | Not issued by DSU |

Table 12: CHI E request mapping

Combined commands which include write followed by a CMO must be executed one after another. The write must be completed at-lest to a point where it has been ordered by DCE/DMI/DII i.e., StrReq has been received by the issuing AIU before processing the CMO. Implementation may decide to execute the write completely before processing the CMO. If the write returns an error, then the CMO must not be executed, and an error response must be returned on the native interface. Implementation of combined commands using a single transaction table entry is desirable.

### Snoop transactions

Ncore will only issue currently supported CHI-B snoops. Support for new CHI E snoops is not required and they will not be generated by Ncore.

Because CHI-E specification has removed the DoNotDataPull bit in the snoop request. For a CHI-E AIU, when it hasn’t any stashing entry available to accommodate the incoming stash, this CHI AIU, which is the stash target, would downgrade the stash to the non-stashing version for the write stash snoop and it would drop the stash for the read stash.

Since the DoNotDataPull bit is supported in CHI-B specification, for a CHI-B AIU, when it hasn’t any stashing entry available to accommodate the incoming stash, the stash target CHI AIU, it asserts DoNotDataPull on its reply for both a read or write stash.

If there is room to accommodate a stashing request, the stash target CHI AIU would accept the stash request, which it can be either a read or write stash. This CHI AIU would further pass the stash to the upstream coherent PE for further processing.

### New CHI E request fields

Command request fields are shown in Table 13 and the snoop request fields are shown in Table 14. Note that the fields shown are the new fields that were added after CHI B. Increase in field widths like different IDs must be stored in transaction table to generate proper response or padded with zeros as needed.

|  |  |
| --- | --- |
| CHI E Request Field | Notes |
| PGroupID | The field is not used in Ncore 3.6 and must be ignored. Must be stored in OTT to generate proper Persist and CompPersist responses |
| Deep | The field is not used in Ncore 3.6 and must ignored |
| DoDWT | The field is not used in Ncore 3.6 and must be ignored. In general, Ncore already does direct write transfers as the data does not flow though DCE |
| SLCRepHint | The field is not used in Ncore 3.6 and must ignored. |
| TagOp | The field is not used in Ncore 3.6 and must ignored. |
| TagGroupID | The field is not used in Ncore 3.6 and must ignored. |
| MPAM | The field is not used in Ncore 3.6 and must ignored. The field must be configured to a width of zero. |

Table 13 Command request fields

|  |  |
| --- | --- |
| CHI E Request Field | Notes |
| MPAM | The field is not used in Ncore 3.6 and must be ignored. The field must be configured to a width of zero. |

Table 14: Snoop request fields

## Data bus

This section describes how the CHI data bus is integrated into current Ncore protocol, changes in this section are primarily limited to AIUs. Note that DVMs are covered in a separate section. Data fields are shown in Table 15. Note that the fields shown are the new fields that were added after CHI B.

|  |  |
| --- | --- |
| CHI E Data Field | Notes |
| CBusy | See details at chapter 2.11 |
| OpCode | DataSepResp is not issued by Ncore  Ncore must handle NCBWrDataCompAck issued by RN |
| TagOp | The field is not used in Ncore 3.6 and must ignored or tied to zero |
| Tag | The field is not used in Ncore 3.6 and must ignored or tied to zero |
| TU | The field is not used in Ncore 3.6 and must ignored or tied to zero |

Table 15: Data fields

## Response bus

This section describes how the CHI response bus is integrated into current Ncore protocol, changes in this section are primarily limited to AIUs. Note that DVMs are covered in a separate section. Response fields are shown in Table 16. Note that the fields shown are the new fields that were added after CHI B.

|  |  |
| --- | --- |
| CHI E Response Field | Notes |
| CBusy | See details at chapter 2.11 |
| PGroupID | The field is not used in Ncore 3.6 and must be copied over form the original request |
| StashGroupID | The field is not used in Ncore 3.6 and must be copied over form the original request |
| Opcode | ***TagMatch*** is not used  ***RespSepData*** is not used  ***Persist*** is not used, always issue combined response  ***CompPersist*** issued for Persist CMOs, always use combined response  ***DBIDRespOrd*** is not used  ***StashDone*** is not used as Sep stash are not supported  ***CompStashDone*** is not used as Sep stash are not supported  ***CompCMO*** Used for non-persist CMO, in the case of Persist CMO combined response is not used <double check> |
| TagOp | The field is not used in Ncore 3.6 and must ignored or tied to zero |
| TagGroupID | The field is not used in Ncore 3.6 and must ignored or tied to zero |

Table 16: Response fields

## Exclusive transaction support

CHI E supports a mechanism for exclusive transactions that does not require point of coherency PoC monitor. This mechanism relies on a precise snoop filter and snooping the requester. To limit the changes in Ncore 3.6 these exclusive transactions are emulated within the CHI AIU. The emulation mapping and process is shown in

|  |  |  |
| --- | --- | --- |
| CHI E Native Command | Concerto Commands | Notes |
| ReadPreferUnique | CmdRdNshDty | Treated as read not shared dirty |
| MakeReadUnique | CleanUnique + CmdRdNshDty | Issue CleanUnique exclusive if it passes issue Comp\_UC response, if the CleanUnique exclusive fails then issue CmdRdNshDty to get the data. always send the data back to agent as SC i.e., CompData\_SC even if the DtrReq received is unique. Downgrading here is okay from coherency perspective and is required by CHI E spec |

Table 17: CHI E Exclusive request mapping

Exclusive MakeReadUnique can end up being processed as two transactions, in this case it is desirable to use a single transaction table entry.

Note: CmdRdNshDty generated from ReadPreferUnique(Excl) or second part of MakeReadUnique(Excl) needs to set ES (Exclusive Access) bit in CmdReq. Same thing applies to CleanUnique(Excl).

### Non coherent exclusive support

Ncore 3.6 implements an optional non-coherent exclusive monitor. The monitor shall be instantiated at the PoS (point of serialization) namely DMI or DII. Please note that if a cache is present in DMI the exclusive monitor is then required.

In the absence of exclusive monitor for non-coherent transactions, the target device connected to DMI or DII is required to implement the exclusive monitor.

In contrast with Ncore 3.4, all non-coherent exclusive transactions should be cacheable requests with visibility attribute set as system visible.

## DVM support

DVM\_v8.4 flow stays the same in CHI -E.c apart from few extra fields which require mapping changes within Ncore for both DVM commands and DVM snoops. In addition, Ncore supports ACE5-H.c. This requires changes in CHI AIU, DVE and IOAIU. DVM command mapping is shown in Table 18 and the DVM data mapping shown in Table 19.

Note:

1. for ACE request only Physical Instruction Cache Invalidate (PICI) will contain PA information in 2nd command and VA in 1st command.
2. for ACE agent, if bit[0] Addr(single/two part command) is 0, it means it is a single part DVM message and does not carry VA information, bit[23:16]/bit[39:32] and bit[31:24]/bit[43:40] are used for ASID and VMID if applicable.
3. for both ACE and CHI, the field VMID[7:0]/VA(VI)[27:20] and ASID[7:0]/VA(VI)[19:12] are all in same format, it is sent on same CCMP command and directly pass to corresponding ACE/CHI snpDVM.
4. for ACE 1st command and 1st snoop bit[43:40] = VA[48:45] or VMID[15:12], if it is one part command/snoop, it carries VMID[15:12], otherwise it carries VA[48:45], this is for ACE AIU to determine which information to be passed in the field.
5. If range bit is not set, Num[] must be set to 0 in snpDVM to CHI interface.
6. for CHI.E SnpDVMop the NS bit must be set to 0.
7. For ACE bit[15] of the 1st snoop is derived from DVM opcode
   1. 1’b0: TLB Invalidation, Branch Predictor Invalidation, Instruction Cache Invalidation, Hint
   2. 1’b1: Synchronization

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CCMP CmdAddress field | | CHI E CmdAddress field | | ACE CmdAddress Field (1st Command Field) | |
| Addr | **Contents** | **Addr** | **Contents** | **Addr** | **Contents** |
| 0 | Reserved 0 | 0 | Reserved 0 |  |  |
| 1 | Reserved 0 | 1 | Reserved 0 | 1 | Reserved 0 |
| 2 | Reserved 0 | 2 | Reserved 0 | - | - |
| 3 | ~~Part Number~~ | ~~3~~ | ~~Part Number~~ | - | - |
| 4 | Va Valid | 4 | Va Valid | 0 | Single/ two part (require address or not) |
| 5 | VMID Valid | 5 | VMID Valid | 6 | VMID Valid |
| 6 | ASID Valid | 6 | ASID Valid | 5 | ASID Valid |
| 8:7 | Security | 8:7 | Security | 9:8 | Security |
| 10:9 | Exception Level | 10:9 | Exception Level | 11:10 | Exception Level |
| 13:11 | DVMOp Type | 13:11 | DVMOp Type | 14:12 | DVMOp Type |
| 21:14 | VMID[7:0]/(ACE: VA[27:20]) | 21:14 | VMID[7:0]/VI[27:20] | 31:24 | VMID[7:0]/VA[27:20] |
| 29:22 | ASID[7:0]/(ACE:VA[19:12]) | 29:22 | ASID[7:0]/VI[19:12] | 23:16 | ASID[7:0]/VA[19:12] |
| 37:30 | ASID[15:8] | 37:30 | ASID[15:8] | 39:32 | ASID[15:8] |
| 39:38 | Staged Invalidation (S2,S1) | 39:38 | Staged Invalidation (S2,S1) | 3:2 | Staged Invalidation (S2,S1) |
| 40 | Leaf Entry Invalidation | 40 | Leaf Entry Invalidation | 4 | Leaf Entry Invalidation |
| 41 | Range | 41 | Range | 7 | Range |
| 42 | Num[4] (for CHI agent) | 42 | Num[4] | - | - |
| Max:43 | Reserved 0 | Max:43 | Reserved 0 | Max:48 | Reserved 0 |

Table 18: DVM command mapping

| CCMP DtwReq field | | CHI E Data field | | ACE Data Field (1st and 2nd Command Field) | |
| --- | --- | --- | --- | --- | --- |
| Addr | Contents | Addr | Contents | Addr | Contents |
| 0 | NUM[0] | 0 | NUM[0] | 0 | NUM[0] (2nd cmd) |
| 1 | NUM[1] | 1 | NUM[1] | 1 | NUM[1] (2nd cmd) |
| 2 | NUM[2] | 2 | NUM[2] | 2 | NUM[2] (2nd cmd) |
| 3 | NUM[3](/PA[4]/VA[4] for ACE agent) | 3 | NUM[3] | 4 | NUM[3]/PA[4]/VA[4] (2nd cmd) |
| 5:4 | Scale[1:0]/VA[7:6]/PA[7:6] | 5:4 | Scale[1:0]/VA[7:6]/PA[7:6] | 7:6 | Scale[1:0]/VA[7:6]/PA[7:6] (2nd cmd) |
| 7:6 | TTL[1:0]/VA[9:8]/PA[9:8] | 7:6 | TTL[1:0]/VA[9:8]/PA[9:8] | 9:8 | TTL[1:0]/VA[9:8]/PA[9:8] (2nd cmd) |
| 9:8 | TG[1:0]/VA[11:10]/PA[11:10] | 9:8 | TG[1:0]/VA[11:10]/PA[11:10] | 11:10 | TG[1:0]/VA[11:10]/PA[11:10] (2nd cmd) |
| 37:10 | VA[39:12]/PA[39:12] | 37:10 | VA[39:12]/PA[39:12] | 39:12 | VA[39:12]/PA[39:12] (2nd cmd) |
| 38 | VA[40]/PA[40] | 38 | VA[40]/PA[40] |  | If PICI : PA[40]  AxADDR[40] (2nd cmd)  If not PICI: VA[40]  AxADDR[3] (2nd cmd) |
| 42:39 | VA[44:41]/PA[44:41] | 42:39 | VA[44:41]/PA[44:41] |  | If PICI : PA[44:41]  AxADDR[44:41] (2nd cmd)  If not PICI: VA[44:41] AxADDR[43:40] (2nd cmd) |
| 46:43 | VA[48:45]/PA[48:45] | 46:43 | VA[48:45]/PA[48:45] |  | If PICI : PA[48:45]  {1’b0, AxADDR[47:45] (2nd cmd)}  If not PICI: VA[48:45] AxADDR[43:40] (1st cmd) |
| 50:47 | VA[52:49]/PA[52:49] | 50:47 | VA[52:49]/PA[52:49] | 47:44 | VA[52:49] (2nd cmd) |
| 54:51 | VA[56:53] for ACE agent | 54:51 | Reserved 0 | 47:44 | VA[56:53] (1st cmd) |
| 55 | NUM[4]/PA[5]/VA[5] (for ACE agent) | 55 | Reserved 0 | 5 | NUM[4]/PA[5]/VA[5] (2nd cmd) |
| 59:56 | VMID[11:8] | 59:56 | VMID[11:8] |  | VMID[11:8]: AxVMIDEXT[3:0] |
| 63:60 | VMID[15:12] | 63:60 | VMID[15:12] |  | VMID[15:12]:  If one part command: AxADDR[43:40] (1st cmd)  If two part command:  AxVMIDEXT[3:0] (2nd cmd) |

Table 19: DVM data mapping

DVM Snoop mapping for address bits and MPF fields are shown in Table 20 and Table 21. Note MPF1 and MPF 2 fields stays the same but MPF3 field now contains range flag and Num dvm fields in addition to identifying the snoop number.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CCMP 1st Snp field | | CHI E 1st Snp field | | ACE Snp Field (1st & 2nd) | |
| Addr | **Contents** | **Addr** | **Contents** | **Addr** | **Contents** |
| 2:0 | Reserved 0 | - | - | - | - |
| 3 | 0 | 0 | 0 |  | Snoop Sequence ID |
| 4 | Va Valid | 1 | Va Valid | 0 | Single/ two part *(1st )* |
| 5 | VMID Valid | 2 | VMID Valid | 6 | VMID Valid *(1st )* |
| 6 | ASID Valid | 3 | ASID Valid | 5 | ASID Valid *(1st )* |
| 8:7 | Security | 5:4 | Security | 9:8 | Security *(1st )* |
| 10:9 | Exception Level | 7:6 | Exception Level | 11:10 | Exception Level *(1st )* |
| 13:11 | DVMOp Type | 10:8 | DVMOp Type | 14:12 | DVMOp Type *(1st )* |
| 21:14 | VMID[7:0]/(ACE: VA[27:20]) | 18:11 | VMID[7:0] | 31:24 | VMID[7:0]/VA[27:20] *(1st )* |
| 29:22 | ASID[7:0]/(ACE:VA[19:12]) | 26:19 | ASID[7:0] | 23:16 | ASID[7:0]/VA[19:12] *(1st )* |
| 37:30 | ASID[15:8] | 34:27 | ASID[15:8] | 39:32 | ASID[15:8] *(1st )* |
| 39:38 | Staged Invalidation (S2,S1) | 36:35 | Staged Invalidation (S2,S1) | 3:2 | Staged Invalidation (S2,S1) *(1st )* |
| 40 | Leaf Entry Invalidation | 37 | Leaf Entry Invalidation | 4 | Leaf Entry Invalidation *(1st )* |
| 43:41 | VA[48:46] | 40:38 | VA[48:46] | 43:41 | VA[48:46] *(1st )* if two part snoop |
| 44 | VA[50] | 41 | VA[50] | 45 | If not PICI : VA[50] *(2nd)* |
| 45 | VA[52] | 42 | VA[52] | 47 | If not PICI : VA[52] *(2nd)* |
| 51:46 | Reserved 0 | 48:43 | Reserved 0 | 1 | Reserved 0 |
| Max:52 | Reserved 0 | Max:49 | Reserved 0 | 15 | Completion (depending on message type) |
| MPF3 | **Contents** | **FwdNID** | **Contents** |  |  |
| 0 | 0 (snoop number) | - | - | - | - |
| 1 | Range | 0 | Range | 7 | Range |
| Max:2 | Reserved 0 | 6:1 | Reserved 0 | - | - |
| MPF2 | **Contents** | **N/A** | **N/A** | **N/A** | **N/A** |
|  | DVE transaction ID | - | - | - | - |
| MPF1 | **Contents** | **VMIDExt** | **Contents** | **N/A** | **N/A** |
| 7:0 | CHI/ACE: VMID[15:8] | 7:0 | VMID[15:8] | VMID[11:8] | AxVMIDEXT[3:0] |
| VMID[15:12] | If one part snoop: AxADDR[43:40] (1st)  If two part snoop:  AxVMIDEXT[3:0] (2nd) |
| Max:8 | Reserved 0 | - | - | - | - |

Table 20: DVM 1st snoop mapping

| CCMP 2nd Snp field | | CHI E 2nd Snp field | | ACE Snp Field (1st & 2nd) | |
| --- | --- | --- | --- | --- | --- |
| Addr | Contents | Addr | Contents | Addr | Contents |
| 2:0 | Reserved 0 | - | - | - | - |
| 3 | 1 | 0 | 1 |  | Snoop Sequence ID |
| 5:4 | Scale[1:0]/VA[7:6]/PA[7:6] | 2:1 | Scale[1:0]/VA[7:6]/PA[7:6] | 7:6 | Scale[1:0]/VA[7:6]/PA[7:6] (2nd) |
| 7:6 | TTL[1:0]/VA[9:8]/PA[9:8] | 4:3 | TTL[1:0]/VA[9:8]/PA[9:8] | 9:8 | TTL[1:0]/VA[9:8]/PA[9:8] (2nd) |
| 9:8 | TG[1:0]/VA[11:10]/PA[11:10] | 6:5 | TG[1:0]/VA[11:10]/PA[11:10] | 11:10 | TG[1:0]/VA[11:10]/PA[11:10] (2nd) |
| 37:10 | VA[39:12]/PA[39:12] | 34:7 | VA[39:12]/PA[39:12] | 39:12 | VA[39:12]/PA[39:12] (2nd) |
| 38 | VA[40]/PA[40] | 35 | VA[40]/PA[40] |  | If PICI : PA[40] AxADDR[40] (2nd)  If not PICI: VA[40] AxADDR[3] (2nd) |
| 42:39 | VA[44:41]/PA[44:41] | 39:36 | VA[44:41]/PA[44:41] |  | If PICI : PA[44:41] AxADDR[44:41] (2nd)  If not PICI: VA[44:41] AxADDR[43:40] (2nd) |
| 43 | VA[45]/PA[45] | 40 | VA[45]/PA[45] |  | If PICI : PA[45] AxADDR[45] (2nd)  If not PICI : VA[45]   1. If two part snoop: AxADDR[40] (1st) |
| 44 | VA[49]/PA[46] | 41 | VA[49]/PA[46] |  | If PICI : PA[46] AxADDR[46] (2nd)  If not PICI: VA[49] AxADDR[44] (2nd) |
| 45 | VA[51]/PA[47] | 42 | VA[51]/PA[47] |  | If PICI : PA[47] AxADDR[47] (2nd)  If not PICI : VA[51] AxADDR[46] (2nd) |
| 49:46 | PA[51:48] | 46:43 | PA[51:48] |  |  |
| 51:50 | Reserved 0 | 48:47 | Reserved 0 |  |  |
| Max:52 | Reserved 0 | Max:49 | Reserved 0 |  |  |
|  |  |  |  |  |  |
| MPF3 | **Contents** | **FwdNID** | **Contents** | **N/A** | **N/A** |
| 0 | 1 (snoop number) | - | - | - | - |
| 1 | Num[0] | 0 | Num[0] | 0 | Num[0] (2nd ) |
| 2 | Num[1] | 1 | Num[1] | 1 | Num[1] (2nd ) |
| 3 | Num[2] | 2 | Num[2] | 2 | Num[2] (2nd ) |
| 4 | NUM[3](/PA[4]/VA[4] for ACE agent) | 3 | Num[3] | 4 | Num[3] (2nd )/VA[4]/PA[4] |
| 5 | NUM[4](/PA[5]/VA[5] for ACE agent) | 4 | Num[4] | 5 | Num[4] (2nd )/VA[5]/PA[5] |
| Max:6 | Reserved 0 | 6:5 | Reserved 0 | - | - |
| MPF2 | **Contents** | **N/A** | **N/A** | **N/A** | **N/A** |
|  | DVE transaction ID | - | - | - | - |
| MPF1 | **Contents** | **VMIDExt** | **Contents** | **N/A** | **N/A** |
| 7:0 | CHI: 0  ACE: {4’b0, VA[56:53]} | 7:0 | Reserved 0 | 47:44 | VA[56:53] (1st) |
| Max:8 | Reserved 0 | - | - |  |  |

Table 21: DVM 2nd snoop mapping

DVM version system level parameter is introduced, and it is calculated and user settable based on the following table. A read only register is also introduced in GRB block to let the software/firmware know the capability/suggested DVM version of the system.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Configuration | DVM version | DVM master | DVM snoopee | Note |
| ACE | v8 | yes | Yes |  |
| ACE | v8.1 | Yes | yes | AxVMIDEXT presence denotes v8.1 support |
| ACE | v8.4 | no | no | ACE does not support v8.4 |
| ACE-Lite | v8, v8.1, v8.4 | no | no |  |
| ACE-LiteDVM | v8 | yes | Yes |  |
| ACE-LiteDVM | v8.1 | Yes | yes | AxVMIDEXT presence denotes v8.1 support |
| ACE-LiteDVM | v8.4 | no | no | ACE-Lite4 DVM does not support v8.4 |
| ACE-LiteE | v8, v8.1, v8.4 | no | no |  |
| ACE-LiteEDVM | v8 | Yes | yes |  |
| ACE-LiteEDVM | v8.1 | Yes | yes | AxVMIDEXT presence denotes v8.1 support |
| ACE-LiteEDVM | v8.4 | Yes | yes | AxVMIDEXT presence denotes v8.1 support |
| CHI.B | v8, v8.1 | Yes | yes |  |
| CHI.E | v8, v8.1, v8.4 | Yes | yes |  |

Table 22 DVM version capability

## Error Handling

The section will focus on error handling of new added transactions by utilizing the existing mechanism of Ncore Error Architecture.

1. For transactions of 1-to-1 mapping: CleanSharedPersistSep, MakeReadUnique(non-exclusive), WriteUniqueZero, WriteNoSnpZero, and ReadPreferUnique.

The error handling Architecture/uArchitecture remains the same with current supported transactions.

1. For transactions that is not supported in Ncore 3.6: CHI-AIU will resolve the transaction inside the unit and return Non-Data error response, no further messages(CCMP) would be generated to the Ncore system. CHI-AIU also needs to finish the transaction semantics if the transaction flow requires certain messages such as DBIDresp, etc.
2. For Combined Write with CMO transactions: WriteNoSnpFull\_CleanSh/CleanInv/CleanShPerSep, WriteBackFull\_CleanSh/CleanInv/CleanShPerSep, and WriteCleanFull\_CleanSh/CleanShPerSep.

These transactions will be mapped into 2 steps

1. If error occurs during 1st part write transaction, the 2nd part CMO transaction will not be executed, and the final response of Comp(write response) and CompCMO/CompPersist(CMO response) must be synced and returned with same type error response.
2. If 1st part write transaction is successfully conducted but error occurs during 2nd part CMO transaction, it is permitted to return Comp(write response) with success and CompCMO/CompPersist(CMO response) with error response.
3. For exclusive MakeReadUnique transaction: it will be mapped into CleanUnique + optional ReadNotShareDirty
   1. If error occurs during 1st part CleanUnique transaction, the 2nd part ReadNotShareDirty transaction will not be executed, and Comp with error response will be returned.
   2. If 1st part CleanUnique transaction is successful conducted with exclusive fail, but error occurs during 2nd part ReadNotShareDirty transaction, either Comp or CompData response is allowed to be returned. Based on uArchitecture, if error occurs at command/address decoding/control part there won’t be any data movement, the final response will be Comp with error. If error is data corruption or any data related error, the final response will be CompData with error. It is subject to uArchitecture.

## Constraint of Mixed Coherent and Non-coherent Transactions

Currently Ncore rely on CPU to control the ordering behavior of the issued transactions. For cacheable transactions from same AIU, it is based on the assumption there wont be outstanding cacheable transactions to the same cacheline address. For non-coherent(ReadNoSnoop/WriteNoSnoop) and non-cacheable(ReadOnce and WriteUnique) transactions, Ncore does not guarantee order of mixed non-coherent transactions with non-cacheable transactions.

## Improve RB ID Management

The CCMP protocol of RB ID management is revised, the RbrReq\_release, RbrRsp\_release, RbuReq and RbuRsp messages are removed from CCMP. The messages of RB ID management are exchanged by RbrReq and RbrRsp only. A new concept of Generation ID(GID) as a sub field of RB ID is introduced. GID is carried along with RB ID on CCMP messages if applied. DCE is allowed to send multiple RbrReq of an RB ID with different GID to update the information in DMI. Besides the original scenario of RbuReq, now replaced with message RbrRsp, DMI is required to return RbrRsp if it detects the information is update by a new received RbrReq. This improvement ensures that for every initiated transaction there will be only one and exactly one pair of RbrReq and RbrRsp to be transmitted on the network, resulting in less bandwidth requirement of the network and more efficiency of CCMP protocol. For details, please refer to CCMP Specification.

## Local/Remote Cache Agents (pending)

Introduce a vector of parameter for every DCE, indicating the corresponding caching agents are considered local or remote. The parameter will affect the snooping decision specifically to Sharer Promotion features. The new snooping decision architecture specification is described as follows:

For non-invalidating snoops:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Remote Owner | Local Owner | Remote Sharer | Local Sharer | Decision |
| Yes | No | No | No | Remote Owner |
| Yes | No | Yes | No | Remote Owner |
| Yes | No | No | Yes | Remote Owner |
| Yes | No | Yes | Yes | Remote Owner |
| No | Yes | No | No | Local Owner |
| No | Yes | Yes | No | Local Owner |
| No | Yes | No | Yes | Local Owner |
| No | Yes | Yes | Yes | Local Owner |
| No | No | No | No | Miss go to DMI |
| No | No | Yes | No | Go to DMI |
| No | No | No | Yes | Local Sharer |
| No | No | Yes | Yes | Local Sharer |

Table 23 DCE Snooping Decision of REmote/Local Caching Agents 1st

**Note:** the last 3 rows will affect the sharer promotion feature.

For invalidating snoops:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Remote Owner | Local Owner | Remote Sharer | Local Sharer | Snoop data from | Broadcast invalidate snoop |
| Yes | No | No | No | Remote Owner | Non |
| Yes | No | Yes | No | Remote Owner | Remote Sharer |
| Yes | No | No | Yes | Remote Owner | Local Sharer |
| Yes | No | Yes | Yes | Remote Owner | All Sharer |
| No | Yes | No | No | Local Owner | Non |
| No | Yes | Yes | No | Local Owner | Remote Sharer |
| No | Yes | No | Yes | Local Owner | Local Sharer |
| No | Yes | Yes | Yes | Local Owner | All Sharer |
| No | No | No | No | Miss go to DMI | non |
| No | No | Yes | No | Go to DMI | Remote Sharer |
| No | No | No | Yes | Local Sharer | Non promoted Sharer |
| No | No | Yes | Yes | Local Sharer | Non promoted Sharer |

Table 24 DCE Snooping Decision of REmote/Local Caching Agents 2nd

**Note:**

the last 3 rows will affect the sharer promotion feature.

**Implementation Notes:**

add CSR to turn on/off sharer promotion scheme, if the feature is off, DCE will not snoop sharers.

## WriteEvictOrEvict CHI.E command support (pending)

Ncore 3.6 will support new CHI.E command WriteEvictOrEvict. The application is, DCE/SF/HN will have its own decision to perform write back to the memory. General case might be, if the cacheline is the last copy among all caching agents, and SLC is a miss, DCE/SF/HN is likely to keep the copy in SLC, resulting a WriteEvict operation, otherwise, DCE/SF/HN will simply do a Evict and update the SF. There might be more specific usecases, in the future, by add more condition in DCE/SF/HN can easily fulfill the user’s requirement by supporting WriteEvictOrEvict command.

|  |  |  |  |
| --- | --- | --- | --- |
| Owner | More than 1 sharers | The last sharer  (other than Owner) | decision |
| Yes | No | No | Owner in UC state, Perform WriteEvict |
| Yes | No | Yes | Owner in SD, sharer SC, perform Evict |
| Yes | Yes | No | Perform Evict |
| No | Yes | No | Perform Evict |
| No | No | Yes | Perform WriteEvict |

Table 25 WriteEvictOrEvict decision

**Implementation Notes:** add CSR to turn on/off WriteEvictOrEvict scheme, if the feature is off, DCE will always perform Evict.

## Cbusy Support (pending)

Ncore 3.6 will report buffer usage value on Cbusy field in Data and Response channel. The value will be carried through StrReq message from DCE/DMI/DII to CHI AIU. The value is implementation defined, suggested to report the ATT/WTT/RTT usage. Refer to CHI.E spec.

|  |  |
| --- | --- |
| CBusy[2] | When asserted, this indicates multiple cores are actively making requests |
| CBusy[1:0] | Indicates the degree of fullness of the tracker at the Completer as:   * 00 = Less than 50 % full * 01 = Greater than 50 % full * 10 = Greater than 75 % full * 11 = Greater than 90 % full |

**Implementation Note**:

add CSR in CHI AIU to turn on/off the feature, if the feature is off, always return 3’h101 on Cbusy field.

## Trace and Debug Support

### Trace Capture

All AIUs, DMIs and DIIs shall support trace capturing capability. The block snoops SMI interface and captures messages that have the TraceMe field set. It captures non-data request, response and data messages (this includes all NDP header fields that are of non-zero width and NDP payload); actual data within the data message is not captured. Which SMI ports are to be snooped and captured is configurable via CSR settings. Multiple SMI ports can be snooped and captured simultaneously. The capture block has a capture buffer that is sized based on the parameter nUnitTraceBufSize, this parameter specifies the number of 64-byte entries in the buffer. The captured data packed into DTWs, once enough data is captured to build a single DTW, it is sent out on the SMI data network to the DVE in the system. If enough data is not accumulated to build a single DTW then a DTW with at-least a single concerto message and padding must be sent out once a timeout is reached (the timeout value is implementation defined example 12bit counter). If the capture buffer is full then no more messages are captured until the buffer drains to accommodate at least one more DTW.

The capture block will give out two events. These events indicate SMI messages that were dropped and captured in a clock cycle respectively. As there can be upto 8 SMI messages that are dropped or captured, these event maybe 3 bits wide each.

With the version 3.6, Ncore will support an additional Non-Data network Ndn3 - this network was added to improve performance for mixed read/write traffic. To support trace capture on this network, two additional control bits have been added for the CaptureControl Register.

#### Capture Control Register (xCCTRLR)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bits | Name | Access | Scope | Reset | Description |
| 0 | Ndn0 SMI TX | R/W | All | 0x0 | Ndn0 SMI TX snoop and capture enable |
| 1 | Ndn0 SMI RX | R/W | All | 0x0 | Ndn0 SMI RX snoop and capture enable |
| 2 | Ndn1 SMI TX | R/W | All | 0x0 | Ndn1 SMI TX snoop and capture enable |
| 3 | Ndn1 SMI RX | R/W | All | 0x0 | Ndn1 SMI RX snoop and capture enable |
| 4 | Ndn2 SMI TX | R/W | All | 0x0 | Ndn2 SMI Tx snoop and capture enable |
| 5 | Ndn2 SMI RX | R/W | All | 0x0 | Ndn2 SMI Rx snoop and capture enable |
| 6 | Dn0 SMI TX | R/W | All | 0x0 | Dn0 SMI Tx snoop and capture enable |
| 7 | Dn0 SMI RX | R/W | All | 0x0 | Dn0 SMI Rx snoop and capture enable |
| 8 | Ndn3 SMI TX | R/W | All | 0x0 | Ndn3 SMI TX snoop and capture enable |
| 9 | Ndn3 SMI RX | R/W | All | 0x0 | Ndn3 SMI RX snoop and capture enable |
| 15:10 | Reserved | RO | All | 0x0 |  |
| 19:16 | gain value | R/W | All | 0x2 | 4-bit gain value |
| 31:20 | Inc value | R/W | All | 0x100 | Time stamp counter increment value:   * top 4 bits are integer * lower 8 bits are fractional |

# SecSubRows Feature

Ncore 3.6 will support extra configuration of address hashing to AIU/DCE/DMI interleaving and CCP set select function.

New array of objects is introduced, and example is described as follows.

**Notes:** the notation of “0,1,2,3” in SecSubRows are in software order manner, not the hardware bit map location.

"SetSelectInfo": {

          "nAddrBits": 4,

          "nRsrcIdxBits": 4,

          "PriSubDiagAddrBits": [16, 17, 18, 19],

          "SecSubRows": {

            "0": [9, 13, 11, 14], // MSB of Set Select

            "1": [13, 15, 10, 12],

            "2": [11, 10, 13, 14, 19],

            "3": [10, 12, 15]} // LSB of Set Select

        } (this is the format we generate as config JSON shown

to the customer and on GUI)

And it is corresponding to the following address hashing function

select[0] = addr[19] ^ (addr[10] ^ addr[12] ^ addr[15]);

select[1] = addr[18] ^ (addr[11] ^ addr[10] ^ addr[13] ^ addr[14] ^ addr[19]);

select[2] = addr[17] ^ (addr[13] ^ addr[15] ^ addr[10] ^ addr[12]);

select[3] = addr[16] ^ (addr[9] ^ addr[13] ^ addr[11] ^ addr[14]);

And to be Noted, the actual JSON file parameter is expressed in the following format for RTL and DV team internal usage. The conversion is done by software team to translate array of objects into array of integers.

          "SecSubRows": {

            "0": 0x6A00, // MSB of Set Select

            "1": 0xB400,

            "2": 0x6E00,

            "3": 0x9400 // LSB of Set Select

          }

# Slow SRAM Fix

This is a very micro architecture specific fix, and the details are described in the micro arch document. Associated parameters are described in section 0.

# CSR access security

This section describes the CSR access method, improved from a security perspective. Ncore shall enforce security to configuration accesses. Each AIU shall be enabled or disabled to initiate CSR accesses by programming its NRS valid bit. This secure access method was required in NCore 3.4 because the existing security check based on NS (Non Secure) field was don't care for transactions targeting CSRs and boot region. Ncore 3.6 introduces NS bit check also for CSR access. A non-secure access to the CSR region will result in an error.

At least one AIU shall be enabled in the system. As a reminder, Ncore includes a debug APB port which is also master of the configuration network. The NRS valid bit security check does not apply for accesses from the debug APB port.

As a safety requirement, Ncore may limit to only one, the number of masters in the configuration network by disabling all AIUs, the APB debug port remains the unique master.

## Parameters

A unit level parameter named fnCsrAccess that applies to all AIUs is introduced. It enables or disables CSR access via each AIU. At least one AIU is enabled in the system. Please refer to the parameters specification for more detail.

## Detailed Description

A NRSAR register is added. This register has a valid field that is set based on fnCsrAccess parameter. If the parameter is true then the valid field resets to 1 and if the parameter is false then the valid field resets to 0. Hardware should use this valid field to qualify a transaction hit to the CSR address space. When valid is not set this will likely result in no BAR hit error.

NRSAR becomes a read-only register when the corresponding fnCsrAccess parameter is set. This is required to prevent the access from being removed. Nevertheless, an exception is made for write accesses coming from the debug APB interface.

A write access coming from the debug APB port shall always be allowed to modify the NRS valid bit for all agents. This will allow the debug port granting permission to other agents to access CSR.

## Ncore register space attribute register (xNRSAR)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bits | Name | Access | Reset | Description |
| 30:0 | Reserved | RO | 0x0 |  |
| 31 | NRS valid | R/W or RO\* | fnCsrAccess | Set high to enable CSR access via this AIU (This field must be RO\* (\*except for writes from the debug APB port) for AIUs where the fnCsrAccess parameter is set and R/W where fnCsrAccess parameter is not set) |

# Opens

Questions/Feedback/Need to discuss:

# Glossary

Arteris

A NoC Company

NCore3

A coherent NoC provided by Arteris with AMBA interfaces and built-in caches.

# Notes

Notes ……