AMBA® CHI Issue F Errata

Architecture & Technology Group

Document number	ARM-AES-0077
Document quality	BET
Document version	4.0
Document confidentiality	Non-confidential
Date of issue	11-Jan-2024

AMBA® CHI Issue F Errata

Release information

Date	Version	Changes
2024/Jan/11	4.0	• Second public release
2023/Oct/30	3.0	• Second limited release
2023/Jun/06	2.0	• First public release
2023/Apr/21	1.0	• First limited release

Non-Confidential Proprietary Notice

This document is **NON-CONFIDENTIAL** and any use by you is subject to the terms of this notice and the Arm AMBA Specification Licence set about below.

This document is 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 Arm. **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

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 for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM 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, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM 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 ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be 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 Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

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.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-21451 version 2.2

AMBA SPECIFICATION LICENCE

THIS END USER LICENCE AGREEMENT ("LICENCE") IS A LEGAL AGREEMENT BETWEEN YOU (EITHER A SINGLE INDIVIDUAL, OR SINGLE LEGAL ENTITY) AND ARM LIMITED ("ARM") FOR THE USE OF ARM'S INTELLECTUAL PROPERTY (INCLUDING, WITHOUT LIMITATION, ANY COPYRIGHT) IN THE RELEVANT AMBA SPECIFICATION ACCOMPANYING THIS LICENCE. ARM LICENSES THE RELEVANT AMBA SPECIFICATION TO YOU ON CONDITION THAT YOU ACCEPT ALL OF THE TERMS IN THIS LICENCE. BY CLICKING "I AGREE" OR OTHERWISE USING OR COPYING THE RELEVANT AMBA SPECIFICATION YOU INDICATE THAT YOU AGREE TO BE BOUND BY ALL THE TERMS OF THIS LICENCE.

"LICENSEE" means You and your Subsidiaries. "Subsidiary" means, if You are a single entity, any company the majority of whose voting shares is now or hereafter owned or controlled, directly or indirectly, by You. A company shall be a Subsidiary only for the period during which such control exists.

1. Subject to the provisions of Clauses 2, 3 and 4, Arm hereby grants to LICENSEE a perpetual, non-exclusive, non-transferable, royalty free, worldwide licence to:

   (i) use and copy the relevant AMBA Specification for the purpose of developing and having developed products that comply with the relevant AMBA Specification;

   (ii) manufacture and have manufactured products which either: (a) have been created by or for LICENSEE under the licence granted in Clause 1(i); or (b) incorporate a product(s) which has been created by a third party(s) under a licence granted by Arm in Clause 1(i) of such third party's AMBA Specification Licence; and

   (iii) offer to sell, sell, supply or otherwise distribute products which have either been (a) created by or for LICENSEE under the licence granted in Clause 1(i); or (b) manufactured by or for LICENSEE under the licence granted in Clause 1(ii).

2. LICENSEE hereby agrees that the licence granted in Clause 1 is subject to the following restrictions:

   (i) where a product created under Clause 1(i) is an integrated circuit which includes a CPU then either: (a) such CPU shall only be manufactured under licence from Arm; or (b) such CPU is neither substantially compliant with nor marketed as being compliant with the Arm instruction sets licensed by Arm from time to time;

   (ii) the licences granted in Clause 1(iii) shall not extend to any portion or function of a product that is not itself compliant with part of the relevant AMBA Specification; and

   (iii) no right is granted to LICENSEE to sublicense the rights granted to LICENSEE under this Agreement.

3. Except as specifically licensed in accordance with Clause 1, LICENSEE acquires no right, title or interest in any Arm technology or any intellectual property embodied therein. In no event shall the licences granted in accordance with Clause 1 be construed as granting LICENSEE, expressly or by implication, estoppel or otherwise, a licence to use any Arm technology except the relevant AMBA Specification.

4. THE RELEVANT AMBA SPECIFICATION IS PROVIDED "AS IS" WITH NO REPRESENTATION OR WARRANTIES EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF SATISFACTORY QUALITY, MERCHANTABILITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE, OR THAT ANY USE OR IMPLEMENTATION OF SUCH ARM TECHNOLOGY WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADE SECRETS OR OTHER INTELLECTUAL PROPERTY RIGHTS.

5. NOTWITHSTANDING ANYTHING TO THE CONTRARY CONTAINED IN THIS AGREEMENT, TO THE FULLEST EXTENT PETMITTED BY LAW, THE MAXIMUM LIABILITY OF ARM IN AGGREGATE FOR ALL CLAIMS MADE AGAINST ARM, IN CONTRACT, TORT OR OTHERWISE, IN CONNECTION WITH THE SUBJECT MATTER OF THIS AGREEMENT (INCLUDING WITHOUT LIMITATION (I) LICENSEE'S USE OF THE ARM TECHNOLOGY; AND (II) THE IMPLEMENTATION OF THE ARM TECHNOLOGY IN ANY PRODUCT CREATED BY LICENSEE UNDER THIS AGREEMENT) SHALL NOT EXCEED THE FEES PAID (IF ANY) BY LICENSEE TO ARM UNDER THIS AGREEMENT. THE EXISTENCE OF MORE THAN ONE CLAIM OR SUIT WILL NOT ENLARGE OR EXTEND THE LIMIT. LICENSEE RELEASES ARM FROM ALL OBLIGATIONS, LIABILITY, CLAIMS OR DEMANDS IN EXCESS OF THIS LIMITATION.

6. No licence, express, implied or otherwise, is granted to LICENSEE, under the provisions of Clause 1, to use the Arm tradename, or AMBA trademark in connection with the relevant AMBA Specification or any products based thereon. Nothing in Clause 1 shall be construed as authority for LICENSEE to make any representations on behalf of Arm in respect of the relevant AMBA Specification.
7. This Licence shall remain in force until terminated by you or by Arm. Without prejudice to any of its other rights if LICENSEE is in breach of any of the terms and conditions of this Licence then Arm may terminate this Licence immediately upon giving written notice to You. You may terminate this Licence at any time. Upon expiry or termination of this Licence by You or by Arm LICENSEE shall stop using the relevant AMBA Specification and destroy all copies of the relevant AMBA Specification in your possession together with all documentation and related materials. Upon expiry or termination of this Licence, the provisions of clauses 6 and 7 shall survive.
8. The validity, construction and performance of this Agreement shall be governed by English Law.

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

http://www.arm.com

Contents

AMBA® CHI Issue F Errata

Preface

This document lists errata on the AMBA CHI Issue F specification [1].

Each errata description is organized as a brief reason for the change, along with the precise change.

Errata classification

Each listed errata has a classification ID, of the form XYYY, where:

X is the errata classification type as follows, C, R, E or D:

C	**Clarification**	Informative change only
R	**Relaxation**	Backward-compatible normative change, modifying existing functionality
E	**Enhancement**	Backward-compatible normative change, adding new functionality
D	**Defect**	Non-backward compatible normative change

YYY is an Arm internal tracking number.

Additional reading

This section lists publications by Arm and by third parties.

See Arm Developer (http://developer.arm.com) for access to Arm documentation.

[1] *AMBA 5 CHI Architecture Specification*. (ARM IHI 0050 F) Arm Ltd.

Chapter 1
New/Updated Errata

1.1 C768: DoNotGoToSD value encoding description incorrect for SnpQuery and SnpStash*

Affects:

CHI-E.a, CHI-E.b, CHI-E.c, CHI-F

Description:

In CHI-D, the DoNotDataPull field shared a common field with DoNotGoToSD, and the DoNotDataPull field took precedence in the various Stash Snoops. In the move from CHI-D to CHI-E.a, the DoNotDataPull feature was removed to simplify the Stash transaction flows and reduce hardware complexity. This meant that DoNotGoToSD was promoted to a dedicated field in Stashing snoops, and the following text was added to the specification to align with this:

> Page 13-420 - Applicable, and must be set to 1 in: SnpStashShared, SnpStashUnique. SnpUniqueStash. SnpMakeInvalidStash.

A SnpStash* operation must not not change the cache line state at the Snoopee. This contradicts with table 13-29 “DoNotGoToSD value encoding” which states that “If already in SD state, must exit SD state in response to the snoop, except for SnpQuery”. This statement incorrect and will be updated.

At the same time, the text in the row for DoNotGoToSD=1 will have the mention of SnpQuery removed, as per the text above the table in CHI-F, DoNotGoToSD is inapplicable, and must be set to zero to SnpQuery.

The precise change(s)

Table 13-29 “DoNotGoToSD value encoding” will be updated from:

DoNotGoToSD	Description
0	Permitted to transition to SD state.
1	Transitioning to SD state is not permitted. If already in SD state, must exit SD state in response to the snoop, except for SnpQuery. The bit value can be ignored by the Snoopee if the Snoop request is SnpOnce or SnpOnceFwd.

To:

DoNotGoToSD	Description
0	Permitted to transition to SD state.
1	Transitioning to SD state is not permitted. If already in SD state, must exit SD state in response to the snoop, except for SnpQuery or SnpStash*. The bit value can be ignored by the Snoopee if the Snoop request is SnpOnce or SnpOnceFwd.

1.2 C774: Comp can be used for WriteBack, WriteClean and WriteEvictFull flows

Affects:

CHI-F

Description:

The CopyAtHome feature was introduced in CHI-F, allowing Home to optimize away redundant data transfers in CopyBack transactions. Upon receipt of a CopyBack transaction, if Home provides:

- CompDBIDResp, the transaction proceeds with a data transfer
- Comp, the transaction proceeds without a data transfer

Table 9-6 “Write transaction Response packets legal RespErr field values” on page 9-354 of CHI-F details the associated packets that can be sent for the various Write transactions. Currently, for WriteBack, WriteClean, and WriteEvictFull transactions, the Comp response is shown as not being available for use, which is incorrect. Comp can be used in these transaction flows when Home wants to proceed without a data transfer, and the valid RespErr encodings are either OK or NDERR. The column values for each of these Write transactions should match WriteEvictOrEvict.

Table 9-6 will be updated to reflect this.

The precise change(s)

Table 9-6 “Write transaction Response packets legal RespErr field values” on page 9-354 of CHI-F will be updated from:

Write transaction	**Associated Response packets**									**CompAck**
	DBIDResp*	**Comp**				**CompDBIDResp**				
		OK	**EXOK**	**DERR**	**NDERR**	**OK**	**EXOK**	**DERR**	**NDERR**	
WriteNoSnp	OK	Y	Y	Y	Y	Y	Y	Y	Y	OK
WriteNoSnpDef	OK	Y[a]	N	Y	Y	Y[a]	N	Y	Y	-
WriteUnique	OK	Y	N	Y	Y	Y	N	Y	Y	OK
WriteNoSnpZero WriteUniqueZero	OK	Y	N	Y	Y	Y	N	Y	Y	-
WriteBack WriteClean WriteEvictFull	-	-	-	-	-	Y	N	Y	Y	-
WriteEvictOrEvict	-	Y	N	N	Y	Y	N	Y	Y	OK

[a] In WriteNoSnpDef transactions, used in combination with Resp[2:0] to communicate Successful, Unsupported, and Defer responses.

To:

Write transaction	**Associated Response packets**									**CompAck**
	DBIDResp*	**Comp**				**CompDBIDResp**				
		OK	**EXOK**	**DERR**	**NDERR**	**OK**	**EXOK**	**DERR**	**NDERR**	
WriteNoSnp	OK	Y	Y	Y	Y	Y	Y	Y	Y	OK
WriteNoSnpDef	OK	Y[a]	N	Y	Y	Y[a]	N	Y	Y	-
WriteUnique	OK	Y	N	Y	Y	Y	N	Y	Y	OK
WriteNoSnpZero WriteUniqueZero	OK	Y	N	Y	Y	Y	N	Y	Y	-
WriteBack WriteClean WriteEvictFull WriteEvictOrEvict	-	Y	N	N	Y	Y	N	Y	Y	OK

[a] In WriteNoSnpDef transactions, used in combination with Resp[2:0] to communicate Successful, Unsupported, and Defer responses.

1.3 D796: UCE not a valid start state for ReadOnce*

Affects:

CHI-A, CHI-B, CHI-C, CHI-D, CHI-E.a, CHI-E.b, CHI-E.c, CHI-F

Description:

An RN-F can issue a CleanUnique transaction from the Invalid cache state, which enables a final cache state of Unique Clean Empty (UCE). To this point, the CHI specification has allowed for the UCE state to be a start state for the ReadOnce* collection of transactions. However, this has been constrained with the rule that once an RN-F has issued a ReadOnce* transaction from the UCE state, it cannot update that line. Additionally, there is then the requirement that at the end of a ReadOnce* transaction, the final cache state must be Invalid. This effectively removes any benefit for the RN-F in keeping the line in UCE when issuing the ReadOnce transaction.

ReadOnce* transactions do not mandate the use of the CompAck message as part of the transaction, although it can be specified in the request through the ExpCompAck field. Without a CompAck, Home has no way of knowing when the ReadOnce* transaction has completed at the Requester. If Home has been tracking the occupancy of the associated line in the Requester, and it receives a transaction from another RN-targeting a write operation at the same line, it would be forced to perform an invalidating snoop to ensure that line is no longer allocated in the RN-F. Without this invalidating snoop, there is the very slight risk of a scenario where two different RN-Fs later report a Unique cache state in response to a broadcast snoop. This would not cause any data coherence issues, but complicates the design of a Home for no meaningful benefit.

The start state of the ReadOnce* transactions will retrospectively be constrained to Invalid only.

The precise change(s)

In CHI-F, Table 4-4 “Permitted Requester cache state at the sending of Read request” on page 4-182 will be updated from:

Request	Initial state						
	UD	UC	SD	SC	I	UDP	UCE
ReadOnce	-	-	-	-	Y	-	Y
ReadOnceCleanInvalid	-	-	-	-	Y	-	Y
ReadOnceMakeInvalid	-	-	-	-	Y	-	Y

To:

Request	Initial state						
	UD	UC	SD	SC	I	UDP	UCE
ReadOnce	-	-	-	-	Y	-	-
ReadOnceCleanInvalid	-	-	-	-	Y	-	-
ReadOnceMakeInvalid	-	-	-	-	Y	-	-

Table 4-36 “Cache state transitions at the Requester for Read request transactions” on page 4-228 will be updated from:

Request type	Initial expected state	Initial permitted state	Final state	Comp responses	Separate responses
ReadOnce	I, UCE [a]	-	I	CompData_UC, CompData_I	RespSepData + DataSepResp_UC
ReadOnceCleanInvalid	I, UCE [a]	-	I	CompData_UC, CompData_I	RespSepData + DataSepResp_UC
ReadOnceMakeInvalid	I, UCE [a]	-	I	CompData_UD_PD, CompData_UC, CompData_I	RespSepData + DataSepResp_UC

[a] For ReadOnce*, ReadNotSharedDirty, and ReadShared transactions, the Requester with an initial state of UCE must not upgrade the cache line to UDP or UD while the request is outstanding.

To:

Request type	Initial expected state	Initial permitted state	Final state	Comp responses	Separate responses
ReadOnce	I	-	I	CompData_UC, CompData_I	RespSepData + DataSepResp_UC
ReadOnceCleanInvalid	I	-	I	CompData_UC, CompData_I	RespSepData + DataSepResp_UC
ReadOnceMakeInvalid	I	-	I	CompData_UD_PD, CompData_UC, CompData_I	RespSepData + DataSepResp_UC

[a] For ReadNotSharedDirty and ReadShared transactions, the Requester with an initial state of UCE must not upgrade the cache line to UDP or UD while the request is outstanding.

Table 12-2 “Permitted initial Tag states and request TagOp values in Read transactions” on page 12-395 will be updated from:

Request	TagOp	Data state	Tag state
ReadOnce	*Invalid, Transfer*	I, UCE	Invalid
ReadOnceMakeInvalid			
ReadOnceCleanInvalid			
ReadNotSharedDirty			
ReadShared			

To:

Request	**TagOp**	**Data state**	**Tag state**
ReadOnce	*Invalid, Transfer*	I	Invalid
ReadOnceMakeInvalid			
ReadOnceCleanInvalid			
ReadNotSharedDirty	*Invalid, Transfer*	I, UCE	Invalid
ReadShared			

Chapter 2
Previous Errata

2.1 D630: CompAck and NCBWrDataCompAck in OWO Immediate Write flows

Affects:

CHI-E.b, CHI-E.c, CHI-F

Description:

In CHI-E.b, the transaction structure section was rewritten in order to provide clarity on the transaction flows available to each transaction type. Detail was put in around when exactly certain responses can be sent. The text that describes when Immediate Write transactions can see CompAck responses is currently too strict. The update from CHI-E.b has also been present in the same form in CHI-E.c and CHI-F.

For Write transactions, CompAck can only be used for WriteUnique and WriteNoSnp transactions when they require Ordered Write Observation guarantees. For a WriteUnique and WriteNoSnp transaction flow that has ExpCompAck asserted, a write is not made visible until the CompAck is received. At that time, the transaction can be deallocated at the Home and made visible to other observers.

The CompAck response can be sent by the Requester in this transaction flow at any time after DBIDResp, DBIDRespOrd, CompDBIDResp, or Comp is received by the Requester. Similarly, the NCBWrDataCompAck response can be sent by the Requester after it has received CompDBIDResp, DBIDResp or DBIDRespOrd. In the OWO transaction flow, neither CompAck or NCBWrDataCompAck can wait for the return of Comp if DBIDResp or DBIDRespOrd have been received. The text in the Transaction structure section will be updated to reflect this.

The later section that talks about Streaming Ordered Write transactions on page 2-129 also covers the cross transaction dependency by stating that all Comp responses for all earlier related ordered writes must be seen before the current write can see a CompAck. As the transaction structure section is focused on the flow of singular transactions, this inter transaction dependency will not be incorporated into the updates associated with this errata.

The precise change(s)

The text that describes the sequence for **Immediate Write** transactions, step 3b1 on page 2-57, will be updated:

From:

> Sends a completion acknowledge, CompAck, to the Home.
> The Requester must only send this after it has received Comp or CompDBIDResp.
> It is permitted, but not required, to wait for DBIDResp or DBIDRespOrd before sending CompAck.

To:

> Sends a completion acknowledge, CompAck, to the Home.
> The Requester must only send this after it has received DBIDResp, DBIDRespOrd, CompDBIDResp, or Comp.
> It is permitted, but not required, to wait for DBIDResp or DBIDRespOrd before sending CompAck.
> It is not permitted to wait for Comp before sending CompAck.

Additionally, the text for step 3b2 on the same page will be updated:

From:

> The Requester sends a combined write data and completion acknowledge, NCBWrDataCompAck, to the Home.
> The Requester must only send this after it has received CompDBIDResp or both DBIDResp/DBIDRespOrd and Comp.

To:

The Requester sends a combined write data and completion acknowledge, NCBWrDataCompAck, to the Home.
The Requester must only send this after it has received DBIDResp, DBIDRespOrd, or CompDBIDResp.
It is not permitted to wait for Comp, if DBIDResp or DBIDRespOrd have been received, before sending NCBWrDataCompAck.

The text that describes the sequence for **Combined Immediate Write and CMO** transactions, step 3b2a on page 2-63, will be updated:

From:

Sends CompAck to the Home.
The Requester must only send this after it has received Comp or CompDBIDResp.
It is not permitted to wait for CompCMO before sending CompAck.

To:

Sends CompAck to the Home. The Requester must only send this after it has received DBIDResp, DBIDRespOrd, CompDBIDResp, or Comp.
It is permitted, but not required, to wait for DBIDResp or DBIDRespOrd before sending CompAck.
It is not permitted to wait for Comp or CompCMO before sending CompAck.

Additionally, the text for step 3b2b on page 2-63 will be updated:

From:

The Requester sends a combined write data and completion acknowledge, NCBWrDataCompAck, to the Home.
The Requester must only send this after it has received CompDBIDResp or both DBIDResp/DBIDRespOrd and Comp.

To:

The Requester sends a combined write data and completion acknowledge, NCBWrDataCompAck, to the Home.
The Requester must only send this after it has received DBIDResp, DBIDRespOrd, or CompDBIDResp.
It is not permitted to wait for Comp, if DBIDResp or DBIDRespOrd have been received, before sending NCBWrDataCompAck.
It is not permitted to wait for CompCMO before sending NCBWrDataCompAck.

The text that describes the sequence for **Combined Immediate Write and Persist CMO** transactions, step 3b2a on page 2-68, will be updated:

From:

Sends a completion acknowledge, CompAck, to the Home. The Requester must only send this after it has received Comp or CompDBIDResp.
It is not permitted to wait for CompCMO or Persist before sending CompAck.

To:

Sends a completion acknowledge, CompAck, to the Home.
The Requester must only send this after it has received DBIDResp, DBIDRespOrd, CompDBIDResp, or Comp.
It is permitted, but not required, to wait for DBIDResp or DBIDRespOrd before sending CompAck.
It is not permitted to wait for Comp, CompCMO, or Persist before sending CompAck.

Additionally, the text for step 3b2b on the same page will be updated:

From:

The Requester sends a combined write data and completion acknowledge, NCBWrDataCompAck, to the Home.
The Requester must only send this after it has received CompDBIDResp, or both DBIDResp/DBIDRespOrd and Comp.
It is not permitted to wait for CompCMO or Persist before sending NCBWrDataCompAck.

To:

The Requester sends a combined write data and completion acknowledge, NCBWrDataCompAck, to the Home.
The Requester must only send this after it has received DBIDResp, DBIDRespOrd, or CompDBIDResp.
It is not permitted to wait for Comp, if DBIDResp or DBIDRespOrd have been received, before sending NCBWrDataCompAck.
It is not permitted to wait for CompCMO or Persist before sending NCBWrDataCompAck.

2.2 C670: SnpRespData_I cannot be sent from SC state for SnpUniqueStash

Affects:

CHI-B, CHI-C, CHI-D, CHI-E.a, CHI-E.b, CHI-E.c, CHI-F

Description:

The SnpUniqueStash operation can be considered a SnpUnique with a hint that the Snoopee should request a DataPull to reload that cache line. Since the inclusion of the SnpUniqueStash operation in the CHI specification, there has been a requirement that RetToSrc is 0 in the snoop request. When RetToSrc is 0 for a snoop request, the Snoopee must not return a copy if the cache line is Shared Clean.

Table 4-49 (Snoop response to SnpUniqueStash and SnpMakeInvalidStash) on page 4-246 of the CHI-F specification incorrectly indicates that SnpRespData_I can be given by a Snoopee in response to receiving a SnpUniqueStash. Only a SnpResp_I response should be listed on this row.

In a future major revision of the specification, Arm is considering to align the RetToSrc behavior of SnpUniqueStash with SnpUnique. That is, it will be legal to set RetToSrc to 1 for a SnpUniqueStash, with the expectation that a Snoopee receiving that operation when the line is in a SharedClean state will respond with SnpRespData_I.

The precise change(s)

Table 4-49 (Snoop response to SnpUniqueStash and SnpMakeInvalidStash) on page 4-246 will be updated from:

Snoop request type	Cache state			RetToSrc	Snoop response
	Initial	Final			
		Expected	Others permitted		
SnpUniqueStash	SC	I	-	0	SnpResp_I
					SnpRespData_I
	SD	I	-	0	SnpRespData_I_PD

To:

Snoop request type	Cache state			RetToSrc	Snoop response
	Initial	Final			
		Expected	Others permitted		
SnpUniqueStash	SC	I	-	0	SnpResp_I
	SD	I	-	0	SnpRespData_I_PD

Additionally, to align with section 4.9 (Returning Data with Snoop response) on page 4-258, section 13.10.36 (RetToSrc) on page will be updated from:

> Applicable in all snoops except SnpDVMOp.

To:

RetToSrc is inapplicable and must be set to zero in:

- SnpCleanShared, SnpCleanInvalid, and SnpMakeInvalid
- SnpOnceFwd and SnpUniqueFwd
- SnpUniqueStash, SnpMakeInvalidStash, SnpStashUnique, and SnpStashShared
- SnpQuery

RetToSrc is applicable and can take any value in all other snoops except SnpDVMOp.

RetToSrc is inapplicable and must be set to zero in SnpDVMOp.

2.3 C673: Permitted Requester end cache states for a ReadClean transaction

Affects:

CHI-E.a, CHI-E.b, CHI-E.c, CHI-F

Description:

CHI-E introduced the MTE feature, which resulted in additional functionality for the ReadClean transaction. If a Requester has a cached copy of a line with valid data but the Allocation Tags are not valid, it can use a ReadClean transaction with a TagOp value of *Transfer* to get a copy of the Tags.

Prior to CHI-E, it was only possible to issue a ReadClean from the I or UCE initial cache sates. CHI-E allows the ReadClean request to be sent from any initial cache state, and this is reflected in Table 4-4 “Permitted Requester cache state at the sending of Read request” on page 4-182 correctly. Importantly, the additional start states are only permitted when TagOp has a value of *Transfer*.

Table 4-5 “Permitted final Requester cache state” on page 4-182 states that only the UC or SC states are permitted at the Requester on completion of the the transaction. This is incorrect, and should also include the UD and SD states, to align with what is shown in Table 4-36 “Cache state transitions at the Requester for Read request transactions” on page 4-229.

At the Requester, a final cache state of:

- UD is only possible from a start state of UD/UDP/SD, when ReadClean is issued with TagOP = *Transfer*.
- SD is only possible from a start state of SD, when ReadClean is issued with TagOP = *Transfer*.

The precise change(s)

In CHI-F, Table 4-4 “Permitted Requester cache state at the sending of Read request” on page 4-182 will be updated from:

Request	UD	UC	SD	SC	I	UDP	UCE
ReadClean	Y	Y	Y	Y	Y	Y	Y

To:

Request	UD	UC	SD	SC	I	UDP	UCE
ReadClean TagOp = *Transfer*	Y	Y	Y	Y	Y	Y	Y
ReadClean TagOp != *Transfer*	-	-	-	-	Y	-	Y

Additionally, table 4-5 “Permitted final Requester cache state” on page 4-182 will be updated from:

Request	UD	UC	SD	SC	I	UDP	UCE
ReadClean	-	Y	-	Y	-	-	-

To:

Request	**UD**	**UC**	**SD**	**SC**	**I**	**UDP**	**UCE**
ReadClean TagOp = *Transfer*	Y	Y	Y	Y	-	-	-
ReadClean TagOp != *Transfer*	-	Y	-	Y	-	-	-

2.4 C675: SnpMakeInvalidStash a permitted snoop for WriteUniquePtlStash

Affects:

CHI-B, CHI-C, CHI-D, CHI-E.a, CHI-E.b, CHI-E.c, CHI-F

Description:

In Table 4-25 (Expected snoop requests per Request from an RN) on page 4-212 of the CHI-F specification, it is stated that for WriteUniquePtlStash, the expected Snoop is SnpUniqueStash. Additionally, on page 4-213 it's stated that "The interconnect is permitted to: For WriteUniqueFullStash and WriteUniquePtlStash transactions, send SnpStashUnique or SnpMakeInvalidStash to the target Request Node if the target Request Node does not have the cache line."

It could be implied from this that SnpMakeInvalidStash cannot be sent for a WriteUniquePtlStash request, when the target Request Node does have the cache line, which is not true.

If Home has the latest copy of of the data (in the SLC for example), and is able to merge the partial WriteUnique data with that local copy, a SnpMakeInvalidStash can be issued to the stash target.

The precise change(s)

Table 4-25 (Expected snoop requests per Request from an RN) on page 4-212 will be updated from:

Request category	Request	Expected Snoop
Write-stash	WriteUniqueFullStash	SnpMakeInvalidStash
	WriteUniquePtlStash	SnpUniqueStash

To:

Request category	Request	Expected Snoop
Write-stash	WriteUniqueFullStash	SnpMakeInvalidStash
	WriteUniquePtlStash	SnpUniqueStash or SnpMakeInvalidStash[a]

[a] Possible if Home already has the latest copy of the line

Additionally, Table 7-1 (Stash request and the corresponding Snoop request) will be updated from:

Stash request	Snoop request
WriteUniquePtlStash	SnpUniqueStash
WriteUniqueFullStash	SnpMakeInvalidStash

To:

Stash request	Snoop request
WriteUniquePtlStash	SnpUniqueStash or SnpMakeInvalidStash[a]
WriteUniqueFullStash	SnpMakeInvalidStash

[a] Possible if Home already has the latest copy of the line

2.5 D678: Width of MPAM field incorrect

Affects:

CHI-F

Description:

The addition of Realm Management Extension support into CHI-F resulted in the addition of the NSE field which, along with the NS field, encodes the *Physical Address Space* (PAS). The MPAM field was also modified in a similar way, with the singular bit sub-field MPAMNS being replaced by a two bit MPAMSP field. This resulted in the width increase of the overall MPAM field from 11 bits to 12 bits. However, this update has not been made correctly through the CHI-F document.

The precise change(s)

The following text in section 11.4 (MPAM) on page 11-382 will be updated from:

Field width is either 0 bits or 11 bits:

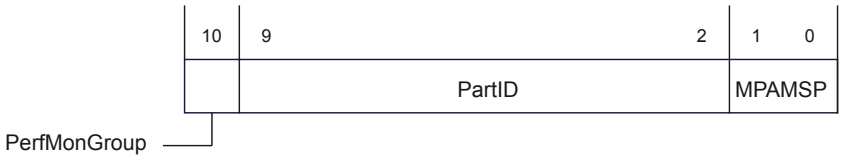
- The width is zero on interfaces that do not support MPAM.
- The width is 11 bits on interfaces that support MPAM. The field is further divided into the subfields:
  - PartID = 9 bits
  - PerfMonGroup = 1 bit
  - MPAMSP = 2 bits

To:

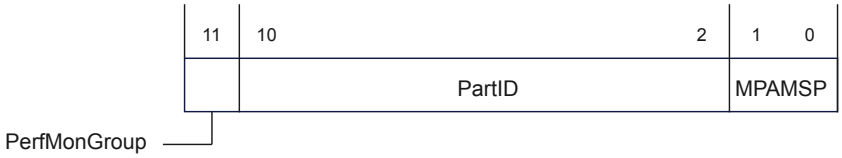
Field width is either 0 bits or 12 bits:

- The width is zero on interfaces that do not support MPAM.
- The width is 12 bits on interfaces that support MPAM. The field is further divided into the subfields:
  - PartID = 9 bits
  - PerfMonGroup = 1 bit
  - MPAMSP = 2 bits

Figure 11-3 on page 11-382 will be updated from:



To:



Finally, the MPAM row in Table 13-8 (Snoop flit format) on page 13-431 will be updated from:

SNPFLIT[$S-1$:0] format		
Field	**Field width**	**Comments**
MPAM	M = 0	No MPAM bus
	M = 11	-

To:

SNPFLIT[$S-1$:0] format		
Field	**Field width**	**Comments**
MPAM	M = 0	No MPAM bus
	M = 12	-

2.6 R679: CBusy allowed to vary in packets that make up a single DAT message

Affects:

CHI-E.c, CHI-F

Description:

Depending on the width of the DAT channel, a singular DAT message can be broken into multiple packets. CHI-E.c saw the introduction of some clarifications that stated in these packets which fields:

- must be consistent
- are expected to be consistent, but can vary
- are expected to vary

CBusy was listed in the “must be consistent” bullet, but this causes some implementation difficultly when the DAT packets are sent far enough apart that the associated component setting the field value sees a change in loading. Additional storage would be required to ensure that CBusy remains constant, which isn’t a preferable solution.

This relaxation will allow the CBusy value to vary between DAT packets of the same message. This leaves implementation freedom for the consumer of the CBusy information. A non-exhaustive set of implementation options for the consumer is:

- Use all CBusy values
- Use the CBusy value from the critical chunk
- Pick any singular CBusy value

The precise change(s)

In section 2.8.4 (Data packetization) on page 2-149 of the CHI-F specification, the CBusy field will be moved from the top level bullet:

- The fields that must be consistent are:

To the following top level bullet :

- The fields that are expected to be consistent, but can vary, are:

2.7 C687: Range field value for GPT TLBI DVMOp transactions

Affects:

CHI-F

Description:

To support the *Realm Management Extensions* (RME), CHI-F had a number of new DVM operations added. Three of these relate to the *Granule Protection Table* (GPT):

- GPT TLBI by PA
- GPT TLBI by PA, Leaf entry only
- GPT TLBI all

The two TLBI by PA variants are range based operations, utilizing the Invalidation Size (IS) field to define how many TLB entries need to be invalidated. For the GPT TLBI by PA operations, the Range field is applicable and must be set to 1. For the GPT TLBI all operation, the Range field is applicable and must be set to 0. This information is currently omitted from the specification and will be included in a future update for clarity.

The precise change(s)

The following text will be added to page 8-334 in the “Invalidation Size in GPT TLBI by PA operations” sub section:

- The Range field is applicable and must be set to 1 for GPT TLBI by PA operations
- The Range field is applicable and must be set to 0 for GPT TLBI all operations

2.8 R688: Line update rules for CopyBack request with CAH = 1

Affects:

CHI-F

Description:

CHI introduced the CopyAtHome feature, removing the need for redundant data transfers when Home keeps a copy of the data provided to the Requester, the Requester doesn't update it, and then the Requester initiates a CopyBack transaction to the same location. A rule was included in the published specification stating that a Requester which has started a CopyBack with the CAH attribute set to 1, must not make any further updates to the line. This rule remains in place for all CopyBack transactions apart from WriteCleanFull.

WriteCleanFull transactions are used when the Requester is holding onto the cache line, but wants to pass the Dirty responsibility downstream along with a copy of the latest data. In a hierarchical Requester, it may be the lower level caching agent that has initiated the WriteCleanFull. Any cache that might exist above in the hierarchy doesn't need to be informed of the operation and if that cache keeps a copy of the data in a Unique state, updates to the line are still possible. It must then be true that these updates can occur during the time window that the WriteCleanFull transaction remains in progress.

The dataless completion of the WriteCleanFull will effectively be passing the earlier dirty copy and responsibility down to Home. However, a newer dirty copy of the cache line can exist in the Requester at the end of the WriteCleanFull operation. This newer copy would then be snooped as a result of any coherent operation that comes into the Home from another Requester in the system.

The specification will be updated to relax the original statement that was too strict.

The precise change(s)

In section 2.7.8 (CopyAtHome attribute) on page 2-144 of the CHI-F specification, the following rule will be updated from:

- Once a CopyBack request has been sent with the CAH attribute set to 1, the Requester must not further modify the cache line until the transaction completes.

To

- Once a CopyBack request, except for a WriteCleanFull, has been sent with the CAH attribute set to 1, the Requester must not modify the cache line.
- Once a WriteCleanFull request has been sent from a UD state with the CAH attribute set to 1, the Requester must ensure that any later updates to the line are not lost if the Home completes the transaction without requesting a data transfer.

2.9 C689: Allowable MakeReadUnique(Excl) responses for Requester in SC state, when the snoop filter is not precise

Affects:

CHI-E.b, CHI-E.c, CHI-F

Description:

When a Requester is performing a MakeReadUnique(Excl) request, and the exclusive request fails, if Home is not precisely tracking the state of the cache line in the Requester then CompData_SC, or RespSepData+DataSepResp_SC can be provided to complete the transaction.

This was originally captured in Table 4-18 in CHI-E.a, but in some editorial changes made to the structure of the table in CHI-E.b, the above information was made less clear.

The specification will be updated to address this.

The precise change(s)

Table 4-40 “Additional Cache state transitions at the Requester for the MakeReadUnique(Excl) request” will be updated from:

Cache state				**Comp Resp**	**Snoop Filter**	
Initial	**At time of Response**	**Home sent invalidating snoop**	**Final**		**Precise**	**Imprecise or Absent**
SC, SD	SC	No	SC	Comp_SC	Y	-
	I	Yes	SC	CompData_SC	Y	Y
				RespSepData DataSepResp_SC	Y	Y
SD	SD	No	SD	Comp_SC	Y	-
				CompData_SC	-	Y
				RespSepData DataSepResp_SC	-	Y

To:

Cache state				**Comp Resp**	**Snoop Filter**	
Initial	**At time of Response**	**Home sent invalidating snoop**	**Final**		**Precise**	**Imprecise or Absent**
SC, SD	SC	No	SC	Comp_SC	Y	-
				CompData_SC	-	Y
				RespSepData	-	Y
				DataSepResp_SC		
	I	Yes	SC	CompData_SC	Y	Y
				RespSepData	Y	Y
				DataSepResp_SC		
SD	SD	No	SD	Comp_SC	Y	-
				CompData_SC	-	Y
				RespSepData	-	Y
				DataSepResp_SC		

2.10 C690: CMO initiated memory update should use PBHA value from SnpRespData

Affects:

CHI-F

Description:

CHI-F added support for *Page Based Hardware Attributes* (PBHA), including rules on helping ensure value consistency as cache lines move around a system. For certain *Cache Maintenance Operations* (CMOs), the original requester might not provide an accurate PBHA value as the request might not go through local translation. An example of this would be a Clean and Invalidate by PA to PoPA instruction in the Arm RME architecture. This means that if an accurate PBHA value is provided with any SnpRespData, it is likely that this will not match with the value that came in the request.

Additionally, if the BROADCASTCMOPOPA pin is driven low on an interface, a CleanInvalidPoPA can be downgraded to a CleanInvalid. The Home will not be aware of this CleanInvalidPoPA downgrading. This places a more general requirement that for any CMO or snoop filter back invalidation, Home must use the PBHA value with any associated SnpRespData responses, and not the PBHA value that came in the request. The specification will be updated to make this clearer.

The precise change(s)

The following text from section 11.5.4 “PBHA value consistency” on page 11-384 of CHI-F will be updated from:

- Cache evictions must obtain the PBHA value from the value stored in cache instead of using the PBHA value from the request evicting the line, except when the request is a CMO. When a write back is the result of a CMO operation, the write back request is permitted to use the PBHA value from the cache or CMO request, as the two PBHA values are expected to be the same.
  - A CMO that operates on a cache line is also expected to operate on any PBHA value stored along with the line.

To:

- Memory accesses that are the result of cache evictions, dedicated CMOs, or snoop filter back invalidations must use the PBHA value that the entry was cached with, not the PBHA value in any associated request.
  - A CMO that operates on a cache line is also expected to operate on any PBHA value stored along with the line.

2.11 D699: Permitted SnpDVMOp SnpResp.RespErr values

Affects:

CHI-E.a, CHI-E.b, CHI-E.c, CHI-F

Description:

In the transition from CHI-D to CHI-E.a, the structure and formatting of the DVMOp transaction field value restrictions tables changed. In this update, the RespErr field was incorrectly updated for a SnpResp response to a SnpDVMOp operation to state that only a value of OK is permitted. This should also include NDERR as an option.

The precise change(s)

In Table 8-4 “Restrictions for response message field values during DVMOp” in CHI-F, on page 8-328, the text in the cell for the RespErr row and the SnpResp column will be updated from:

> Must be all zeros

To:

> Must be 0b00 or 0b11

2.12 C722: SLCRepHint not applicable for ReadNoSnpSep

Affects:

CHI-E.a, CHI-E.b ,CHI-E.c, CHI-F

Description:

The CHI-C specification introduced the ReadNoSnpSep transaction, which can be sent only from an HN to an SN. The SLCRepHint field on the REQ channel, which was introduced in CHI-E.a, is only applicable in requests from an RN to an HN-F. Therefore, the SLCRepHint field is inapplicable for ReadNoSnpSep transactions. This is incorrectly stated as being applicable in CHI-F, Table A-3 “Read, Dataless and Miscellaneous Request message field mappings Part 2”. This will be corrected.

The precise change(s)

In CHI-F, Table A-3 “Read, Dataless and Miscellaneous Request message field mappings Part 2”, on page A-502 will have the SLCRepHint column in the ReadNoSnpSep row changed from a “Y” to a “-”.

2.13 C723: SnpDVMOp does not need to be sent to original requester, but it can be

Affects:

CHI-B, CHI-C, CHI-D, CHI-E.a, CHI-E.b, CHI-E.c, CHI-F

Description:

The CHI specification provides implementation freedom with regard to the DVM snoop network. An implementation may decide to perform broadcast snoops for DVMOps, where all RN-F and RN-D nodes in the system are targeted, regardless of the source of the request. However, a snoop back to the Requester is not strictly required, but is permitted. This will be clarified in the DVM section as there is discrepancy between what is shown in Figure 8-1 and the descriptive text related to the diagram on the following page.

The precise change(s)

The following text from Section 8.1.1 “Non-sync type DVM transaction flow” on page 8-321 of CHI-F will be updated from:

- The MN broadcasts the SnpDVMOp snoop request to all the RN-F and RN-D nodes in the system. The SnpDVMOp is sent on the Snoop channel, and requires two Snoop requests. The two parts of the SnpDVMOp are labeled by the suffix _P1 and _P2.

To:

- The MN broadcasts the SnpDVMOp snoop request to the remaining RN-F and RN-D nodes in the system. The MN is permitted, but not required, to send the SnpDVMOp to the RN that issued the original DVMOp. The SnpDVMOp is sent on the Snoop channel, and requires two Snoop requests. The two parts of the SnpDVMOp are labeled by the suffix _P1 and _P2.

2.14 C725: WriteNoSnpZero can be to Device memory

Affects:

CHI-E.a, CHI-E.b ,CHI-E.c, CHI-F

Description:

CHI-E.a introduced the new transaction type, WriteNoSnpZero. The Permitted memory attributes for WriteNoSnpZero are the same as those for WriteNoSnp. Specifically, WriteNoSnpZero is permitted to a Device memory location. However, this is not fully reflected in the specification. Table 4-14 (RN to HN Write request attribute values) on page 4-194 of the CHI-F specification details this correctly, as does Table A-5 (Write and Combined Write Request message field mappings Part 2), but they are in contradiction to an earlier statement on page 2-135. The text on page 2-135 is incorrect and will be updated.

The precise change(s)

The following text on page 2-135 of the CHI-F specification will be updated from:

> Accesses to Device memory must use the following types, exclusive variants are permitted:
>
> - Read accesses to a Device memory location must use ReadNoSnp.
> - Write accesses to a Device memory location must use WriteNoSnpPtl, WriteNoSnpFull, or WriteNoSnpDef.

To:

> Accesses to Device memory must use the following types, exclusive variants are permitted:
>
> - Read accesses to a Device memory location must use ReadNoSnp.
> - Write accesses to a Device memory location must use WriteNoSnpPtl, WriteNoSnpFull, WriteNoSnpZero, or WriteNoSnpDef.