Arm CoreLink NI-700 Non-Coherent Interconnect

Software Developer Errata Notice

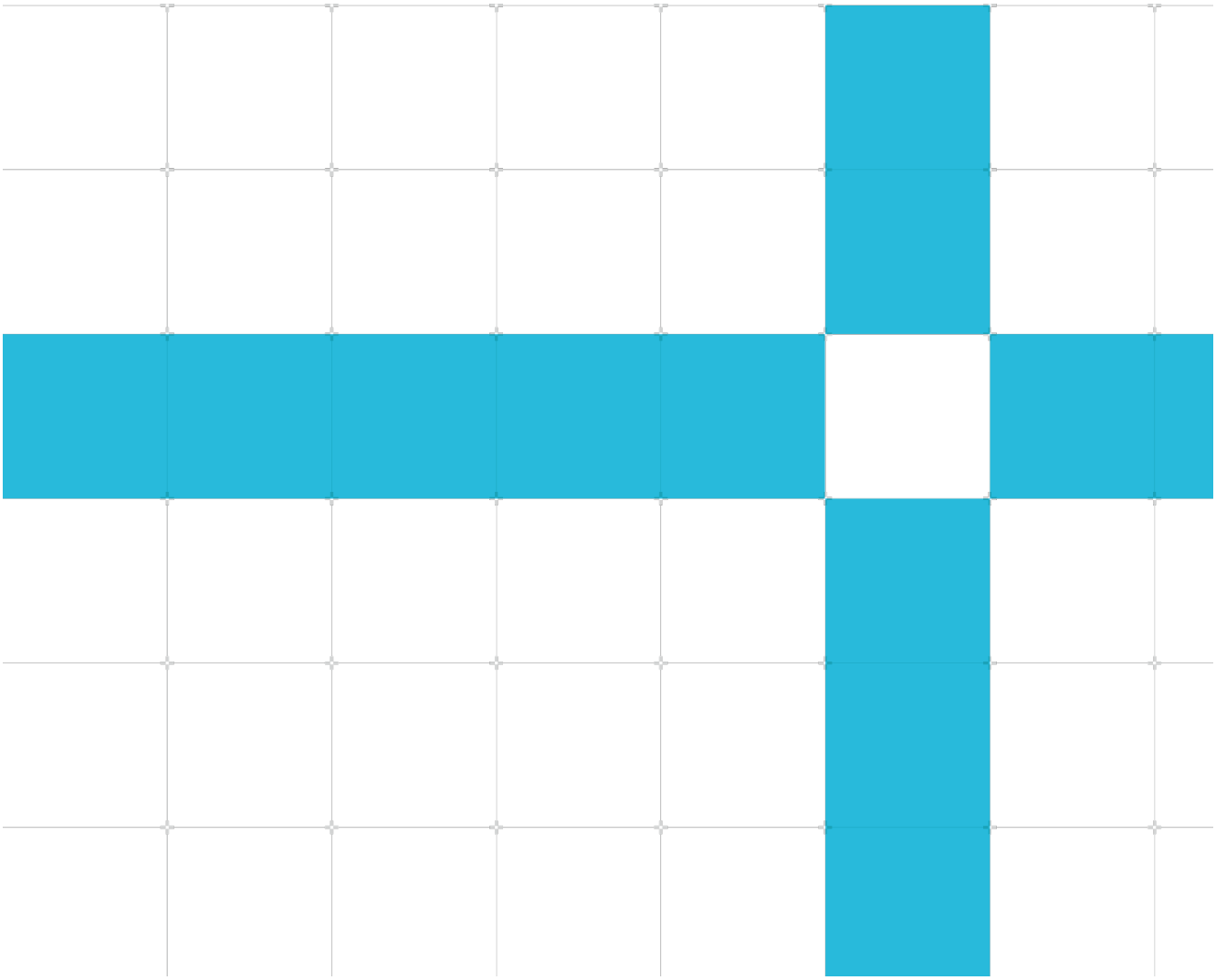
Date of issue: October 10, 2024

Non-Confidential

Document version: 11.0

Document ID: SDEN-1780251

This document contains all known errata since the r1p0 release of the product.



This document (SDEN_1780251_11.0_en) was issued on October 10, 2024.

There might be a later issue at **http://developer.arm.com/documentation/SDEN-1780251**

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

If you find offensive language in this document, please email **terms@arm.com**.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on Arm CoreLink NI-700 Non-Coherent Interconnect, create a ticket on **https://support.developer.arm.com**.

To provide feedback on the document, fill the following survey:
https://developer.arm.com/documentation-feedback-survey.

Contents

Introduction

Scope

This document describes errata categorized by level of severity. Each description includes:

- The current status of the erratum.
- Where the implementation deviates from the specification and the conditions required for erroneous behavior to occur.
- The implications of the erratum with respect to typical applications.
- The application and limitations of a workaround where possible.

Categorization of errata

Errata are split into three levels of severity and further qualified as common or rare:

Category A	A critical error. No workaround is available or workarounds are impactful. The error is likely to be common for many systems and applications.
Category A (Rare)	A critical error. No workaround is available or workarounds are impactful. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category B	A significant error or a critical error with an acceptable workaround. The error is likely to be common for many systems and applications.
Category B (Rare)	A significant error or a critical error with an acceptable workaround. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category C	A minor error.

Change Control

Errata are listed in this section if they are new to the document, or marked as "updated" if there has been any change to the erratum text. Fixed errata are not shown as updated unless the erratum text has changed. The **errata summary table** identifies errata that have been fixed in each product revision.

October 10, 2024: Changes in document version v11.0

ID	Status	Area	Category	Summary
3740492	New	Programmer	Category B	PCIe peer to peer writes when atomics is enabled deadlock under error scenarios

June 28, 2024: Changes in document version v10.0

No new or updated errata in this document version.

March 11, 2024: Changes in document version v9.0

No new or updated errata in this document version.

December 06, 2023: Changes in document version v8.0

No new or updated errata in this document version.

July 24, 2023: Changes in document version v7.0

ID	Status	Area	Category	Summary
2864508	Updated	Programmer	Category B	PCIe peer to peer writes blocked by atomic leads to deadlock

April 04, 2023: Changes in document version v6.0

ID	Status	Area	Category	Summary
2864508	New	Programmer	Category B	PCIe peer to peer writes blocked by atomic leads to deadlock

September 17, 2021: Changes in document version v5.0

ID	Status	Area	Category	Summary
2247267	New	Programmer	Category B	AHB Cacheable No-allocate transactions are converted to Non-cacheable
2231124	New	Programmer	Category C	ASNI/AMNI PMU counter miscounts prefetch and writeCMO transactions towards stash operations

April 01, 2021: Changes in document version v4.0

No new or updated errata in this document version.

March 15, 2021: Changes in document version v3.0

ID	Status	Area	Category	Summary
2072323	New	Programmer	Category C	HMNI SILDBG outstanding transactions field goes to 0x0 if there is a BUSY state in the middle of an AHB burst

October 30, 2020: Changes in document version v2.0

No new or updated errata in this document version.

March 30, 2020: Changes in document version v1.0

No errata in this document version.

Errata summary table

The errata associated with this product affect the product versions described in the following table.

ID	Area	Category	Summary	Found in versions	Fixed in version
2247267	Programmer	Category B	AHB Cacheable No-allocate transactions are converted to Non-cacheable	r1p0, r2p0	r2p1
2864508	Programmer	Category B	PCIe peer to peer writes blocked by atomic leads to deadlock	r1p0, r2p0, r2p1	r2p3
3740492	Programmer	Category B	PCIe peer to peer writes when atomics is enabled deadlock under error scenarios	r1p0, r2p0, r2p1, r2p3	Open
2072323	Programmer	Category C	HMNI SILDBG outstanding transactions field goes to 0x0 if there is a BUSY state in the middle of an AHB burst	r1p0	r2p0
2231124	Programmer	Category C	ASNI/AMNI PMU counter miscounts prefetch and writeCMO transactions towards stash operations	r1p0, r2p0	r2p1

Errata descriptions

Category A

There are no errata in this category.

Category A (rare)

There are no errata in this category.

Category B

2247267
AHB Cacheable No-allocate transactions are converted to Non-cacheable

Status:

Fault Type: Cat-B Programmer
Fault Status: Present in r1p0, r2p0, Fixed in r2p1

Description:

AHB transactions presented to an HSNI as Normal Cacheable No-allocate are converted to a Normal Non-cacheable type. This can result in transactions bypassing a cache downstream of the NI-700.

Configurations Affected:

All the following apply:

- An HSNI has extended memory types enabled (AHB5).
- Based on the address map, the HSNI can send transactions to one or more of either:

- An AMNI (AXI egress endpoint).
- An HMNI that has extended memory types enabled.

- There is a cache downstream of the AMNI or HMNI.

Conditions:

All the following apply:

- A transaction is sent to the HSNI with HPROT[5:3] == 0b011, indicating a Normal Cacheable No-allocate transaction.
- Another observer in the system accesses the same memory locations using Normal Write-back Cacheable transactions which allocate into the cache.

In Arm Cortex-M processors with an AHB interface, Normal Cacheable No-allocate transactions are only issued if the processor includes an MPU. This includes Cortex-M23.

Implications:

The transaction might not look up the downstream cache. If another observer has written data into the cache that has not been written back to memory:

- An AHB read transaction might return stale data from memory.
- An AHB write transaction might leave stale data in the cache.

Work arounds:

Software should avoid selecting a Cacheable No-allocate memory type.

- For systems with Cortex M processors, this can be done in the MPU.
- For other systems, software specific to an impacted peripheral needs to be modified to not generate Normal Cacheable No-allocate transactions.

Where a peripheral consistently emits Normal Cacheable No-allocate transactions, software may be able to treat it as a non-coherent peripheral emitting Normal Non-Cacheable transactions, provided that the Cacheable attribute upstream of the NI-700 is not visible to any other observers.

A hardware work around to the issue is to connect the HPROT[5] input to the HSNI to what is driving the HPROT[4] input. This converts all Normal Cacheable No-allocate transactions to Normal Cacheable Allocate transactions.

2864508
PCIe peer to peer writes blocked by atomic leads to deadlock

Status

Affects: NI-700
Fault Type: Programmer CatB
Fault Status: Present in r1p0, r2p0, r2p1. Fixed in: r2p3.

Description

This issue can happen when NI-700 is used for transport of PCIe transactions in the Root Complex, PCIe peer-to-peer traffic is present, and PCIe atomics are present.

In AMBA terms, the atomic request in question must be load/swap/compare that needs both a read and write response. Atomic requests that have both a read and a write response require an entry in both read and write trackers at the AXI subordinate (ASNI) that the request enters the NI-700 and at the AXI manager (AMNI) that the transaction exits the NI-700. Under the conditions described below, an atomic request can be blocked from making forward progress in an ASNI because it gets stuck behind a read request. Any younger write requests are blocked behind the atomic request in the ASNI's processing pipeline. Read responses are withheld in the peer PCIe controller until prior write requests have completed. Since the read responses cannot be released, there is deadlock. Note that the same behavior also applies an an AMNI.

In PCIe terms, this means a Non-Posted Read with Data (Atomic) Transaction can incorrectly block a Posted Write Transaction from making progress when Non-Posted Transaction resources are exhausted (back-pressured). If the Non-Posted Transaction resources depend on Completions ordered behind the Posted Transaction, as is the case for sustained multi-device P2P read traffic, forward progress cannot be made (deadlock).

Configurations Affected

This issue happens in a configuration where all the following conditions are true:

- NI-700 is used as part of the PCIe Root Complex for transport of PCIe transactions (converted to appropriate AMBA transactions)
- The PCIe Root Complex supports PCIe peer to peer transactions
- The PCIe Root Complex supports PCIe Atomic Transactions (even if only as a completer and not peer-to-peer routing)

Conditions

The precise conditions that cause this issue within an NI-700:

- The read channel is backpressured (stalled).

- The read channel depends on the write channel to make progress (pass the stalled reads) in order for the read backpressure to be released.
- An atomic transaction, because of this bug, causes the write channel to depend on the read channel (deadlock).

This issue is possible with the following PCIe scenario (though more complex scenarios will also expose the issue):

- Two PCIe peers are participating in traffic through the NI-700 (example sequence of transactions described below)
- Transactions issued from both PCIe controllers are:
  - PCIe peer-to-peer Non-Posted reads
  - PCIe Atomics (to any target)
  - PCIe Posted Writes (to any target)

Sequence of Transactions

Multiple (at least two) PCIe peers must be issuing traffic that fits the following profile in order to hit the deadlock. The sequence below describes an ASNI as the block where the deadlock occurs. A similar sequence can be shown where the AMNI that supports atomic transactions is the block where the deadlock occurs.

(1) Peer to peer reads (non-posted by definition). These transactions get sent from the ASNI associated with the originating PCIe controller to the AMNI connected to its PCIe peer

(2) Peer to memory atomic

(3) Posted writes

Given transactions 1-3, the following leads to the deadlock:

(4) The PCIe controllers accept read requests from their peer (transactions (1)) until they hit the maximum number of read requests they can accept and then back pressure their AXI read request channel. This channel is being driven by the NI700 AMNI associated with that PCIe controller.

(5) The back pressure on the read request channel ripples back into the NI700 AMNI, through the NI700 interconnect, and to the ASNI connected to the peer PCIe controller that is the source of the reads.

(6) The ASNI in (5) has read requests it has accepted from its PCIe controller that can't be issued because of the backpressure

(7) The non-posted atomic (transaction (2)) is accepted by the ASNI. It incorrectly blocks the posted write transactions while for waiting for read backpressure to be removed (6).

(8) As read responses are received at the PCIe controller in (4), they are blocked because they can't bypass the write requests from (3) due to PCIe ordering rules

(9) Due to (8), the backpressure on processing read requests in the ASNI is not released and transaction (2) makes no progress, and the write transactions (3) are stuck in the ASNI behind transaction (2).

(10) Deadlock occurs because no write responses to (2) and (3) are received by the PCIe controllers, which keep them from issuing read responses.

Implications

This leads to a deadlock

WorkAround(s)

There is no workaround for the issue other than to avoid the conditions. For all PCIe root ports connected through the same NI-700 instance, either one of the two following options will avoid the issue:

1. Disable PCIe P2P Support through the Root Complex so that forward progress does not depend on Posted Write Transactions (or Completions) passing Non-Posted Transactions.
2. Disable PCIe Atomic Support such that the NI-700 is never presented with an Atomic Transaction on the ASNIs connected to a PCIe controller.

3740492
PCIe peer to peer writes when atomics is enabled deadlock under error scenarios

Status

Affects: PL619
Fault Type: Programmer CatB
Fault Status: Present in r1p0, r2p0, r2p1, r2p3. Open

Description

When PCIe peer to peer traffic is enabled and AXI atomic transactions are allowed either from the PCIe RC into the interconnect, or from a different requester into the PCIe RC, then under certain error conditions there can be a deadlock.

Configurations Affected

This issue happens in a configuration where all the following conditions are true:

- NCI is used as part of the PCIe Root Complex for transport of PCIe transactions (converted to appropriate AMBA transactions)
- The PCIe Root Complex supports PCIe peer to peer transactions
- The PCIe Root Complex supports PCIe Atomic Transactions (as a completer or as a requester)

Additional deadlock cases are hit if:

- IDM is enabled on ASNI or AMNI

Conditions

There are two scenarios that can cause this issue, they are:

- Conditions for Scenario 1

1. Multiple reads issued from PCIe ASNI to PCIe AMNI outstanding to PCIe RC
2. Youngest read issued encounters an Error Scenario. Error Scenario handling uses 'block and drain' which needs read tracker to fully drain before new requests are issued. Error scenarios include:
    - Address decode error
    - IDM isolation request (note that this request can be disabled by SW at the expense of losing IDM functionality)
    - Combined QoS OT is lowered by SW to a value where reads can occupy all of the transactions allowed by the QoS combined output threshold
3. PCIe Atomic Write Transaction issued to PCIe ASNI going to the DMC AMNI. The atomic transaction needs both write tracker and read tracker entries. The atomic transaction cannot be

processed/issued until the read tracker is drained.
4. Posted writes issued from PCIe RC into PCIe ASNI are stuck behind the Atomic Write Transaction
5. Non-posted queue in the PCIe root complex can't drain the reads because read completions cannot bypass Posted writes

- Conditions for Scenario 2

1. Multiple reads issued from PCIe ASNI to PCIe AMNI outstanding to PCIe RC
2. Youngest read issued encounters an error scenario in the PCIe AMNI. This error handling uses 'block and drain' which needs the AMNI's read tracker to fully drain before new requests are issued. Error scenarios include:
   - Opcode that the AMNI doesn't support
   - IDM isolation request (note that this request can be disabled by SW at the expense of losing IDM functionality)
3. Atomic issued to PCIe AMNI. Atomic needs both write tracker and read tracker entry. So Atomic cannot be issued until read tracker is drained.
4. Posted writes issued from PCIe RC into PCIe ASNI are stuck behind the Atomic write in the PCIe AMNI
5. Non-posted queue in the PCIe root complex can't drain the reads because read completions cannot bypass Posted writes
6. Non-Posted queue cannot drain the reads because read completions cannot bypass Posted writes

Implications

Incoming PCIe transactions lead to system deadlock and usually indicate one of the following:

- Hypervisor level error since transactions have attributes that cause errors in the PCIe ASNI or PCIe AMNI
- The combined QoS OT register in the PCIe ASNI is set to a value such that a scenario is possible where read requests take all of the provisioned number of transactions and writes are blocked in the PCIe ASNI as a result of it.

Workaround

If there is an SMMU in the system, ensure that SMMU translations prevent accesses to occur that would cause any of the following:

1. Address decode errors in the ASNI
2. Unsupported ASNI/AMNI transactions

Category B (rare)

There are no errata in this category.

Category C

2072323
HMNI SILDBG outstanding transactions field goes to 0x0 if there is a BUSY state in the middle of an AHB burst

Status:

Fault Type: CAT-C programmer
Fault Status: Present in r1p0, Fixed in r2p0

Description of Issue:

During a Busy State (HTRANS = 0x1) in the middle of an AHB burst the outstanding writes, reads field currently indicates 0x0. Whereas it should continue to indicate 0x1 for the entire burst if there is an outstanding transaction.

Configurations Affected:

NI-700 configurations with HSNI or HMNI endpoints.

Conditions:

If there is a Busy State (HTRANS = 0x1) in the middle of an outstanding AHB burst.

Implications:

HMNI, HSNI SILDBG register does not indicate the outstanding transactions correctly.

WorkAround(s):

For AHB the number of outstanding transactions is not as useful when compared to AXI where there can be multiple outstanding. While there isn't a workaround for this, it is not as consequential.

2231124
ASNI/AMNI PMU counter miscounts prefetch and write_plus_CMO transactions towards stash operations

Status:
Fault Type: CAT C Programmer
Fault Status: Present in r1p0, r2p0. Fixed for r2p1

Description of Issue:

AXI slave and master network interface performance monitor event codes for cache stash operations also count AXI.H prefetch and write CMO transactions.

Configurations Affected:

NI-700 configurations with Prefetch_Transaction, CMO_On_Write, or Write_Plus_CMO enabled on the ACE-Lite slave network interface.

Conditions:

1. AXI slave or master network interface has the PMU event select code programmed to count cache stash transactions (PMU event code = 0x13)
2. Prefetch (AWSNOOP = 0b1111), WritePtlCMO (AWSNOOP = 0b1010), WriteFullCMO (AWSNOOP = 0b1011) are received at that AXI master or slave network interface

Implications:

ASNI/AMNI PMU counter miscounts prefetch and write CMO transactions towards stash operations

WorkAround(s):

The issue only impacts the accuracy of the PMU counter itself. It has no impact on mainline functionality or performance.
Fewer AXI.H Prefetch and write CMO operations will limit the inaccuracy of the cache stash PMU count value.

Proprietary notice

(PRE-1121-V1.0)

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in the Arm documents.

Product status

All products and Services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

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

Product revision status

The rxpy identifier indicates the revision status of the product described in this manual, where:

rx

Identifies the major revision of the product.

py

Identifies the minor revision or modification status of the product.