Errata for the Compute Express Link^{TM} (CXL^{TM}) Specification Revision 1.1

October 2019

Any use or implementation of this specification is expressly subject to the terms and conditions described in the CXL Promoter, Contributor, and/or Adopter Agreement, as appropriate for your status. The implementation or use of industry specifications made available by any other standards body, such as PCIe, if referenced in a CXL Final Specification (as such term is defined in those agreements) is not governed by CXL policies and procedures, and may be subject to patent, copyright, and other intellectual property right protection (collectively “IPR”). Licensing of such IPR in a referenced industry specification shall be governed by the policies and procedures of the organization promulgating such industry specification.

This document is provided ‘as is’ with no warranties, express or implied, including any warranty of merchantability, fitness for a particular purpose, noninfringement of any third-party intellectual property rights, or any warranty otherwise arising out of any proposal, specification or sample. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

Contents

Revision History

Revision	Description	Date
1.0	First Release: Errata E1-E16.	October 2019

E1 CXL.AL and Flex Bus.AL Typo

In general, Flex Bus.AL should be substituted with Flex Bus.CXL throughout the specification document. Additionally, in section 11.4, make the following changes:

11.4 CXL Viral Handling

. . .

When a CXL~~.AL~~ device goes into Viral, the upstream CXL.io shall perform the following:

- Master Abort Upstream Requests
- Completer Abort Upstream Completions
- Signal Failed Response for Downstream Completions

E2 Control SKP Ordered Set Frequency

In section 6.7.1, make the following changes:

6.7.1 Control SKP Ordered Frequency and L1/Recovery Entry

In Flex Bus.CXL mode, if sync header bypass is enabled, the following rules apply:

- After the SDS, the physical layer must schedule a control SKP Ordered Set or SKP Ordered Set after every 340 data blocks, unless it is exiting the data stream. Note: The control SKP OSs are alternated with regular SKP OSs at 16 GT/s or higher speeds; at 8 GT/s, only SKP OSs are scheduled.

E3 CXL Downstream Port Supported PCIe Capabilities

In section 7.2.1.1, Table 60, make the following changes:

Table 60. CXL Downstream Port Supported PCIe Capabilities and Extended Capabilities

Supported PCIe Capabilities and Extended Capabilities	Exceptions[1]	Notes
PCI Express Capability	Slot Capabilities, Slot Control, Slot Status, Slot Capabilites 2, Slot Control 2, and Slot Status 2 registers are not applicable.	None
PCI Power Management Capability	None	None
MSI Capability	None	None
Advanced Error Reporting Extended Capability	None	Required for CXL despite being optional for PCIe
ACS Extended Capability	None	None
Multicast Extended Capability	None	None

Downstream Port Containment Extended Capability	None	None
Designated Vendor-Specific Extended Capability (DVSEC)	None	Please refer to section ~~Figure~~ 7.2.1.3 for Flex Bus Port DVSEC definition.
Secondary PCI Express Extended Capability	None	None
Data Link Feature Extended Capability	None	None
Physical Layer 16.0 GT/s Extended Capability	None	None
Physical Layer 32.0 GT/s Extended Capability	None	None
Lane Margining at the Receiver Extended Capability	None	None
Alternate Protocol Extended Capability	None	None

E4 CXL Upstream Port Supported PCIe Capabilities

In section 7.2.1.2, Table 61, make the following changes:

Table 61. CXL Upstream Port Supported PCIe Capabilities and Extended Capabilities

Support PCIe Capabilties and Extended Capabilties	Exceptions[1]	Notes
PCI Express Capability	None	~~N/A~~None
Advanced Error Reporting Extended Capability	None	Required for CXL despite being optional for PCIe.
~~Multicast Extended Capability~~	~~None~~	~~N/A~~
Virtual Channel Extended Capability	None	VC0 and VC1

Designated Vendor-Specific Extended Capability (DVSEC)	None	Please refer to section ~~Figure~~ 7.2.1.3 for Flex Bus Port DVSEC definition.
Secondary PCI Express Extended Capability	None	None
Data Link Feature Extended Capability	None	None
Physical Layer 16.0 GT/s Extended Capability	None	None
Physical Layer 32.0 GT/s Extended Capability	None	None
Lane Margining at the Receiver Extended Capability	None	None
Alternate Protocol Extended Capability	None	None

E5 CXL Power Management Messages

In section 3.1.2, Table 3, make the following changes:

Field	Description	Notes
Payload[95:0]	. . . AGENT_INFO: If Param.Index == 0, 7:0 - ~~REVISION_ID~~ CAPABILITY_VECTOR 0 - Always set to indicate support for CXL 1.1 PM messages 7:1 - Reserved . . . 15:8 – PrepType 0x00 => General Prep All others reserved ~~0x01 => Early Prep;~~ ~~0x02 => Reset Entry Start (first checkpoint for CXL device blocks during a Reset event/power state transition);~~	CXL Agent must treat the TARGET_AGENT_ID field as Reserved when returning credits to Host. Only Index 0 is defined for AGENT_INFO, all other Index values are reserved.

	~~0x03 => Link Turnoff (typically the last checkpoint during a Reset event/power state transition);~~ 17:16 - Reserved ~~Phase~~ ~~0x00 => Phase 0~~ ~~0x01 => Phase 1~~ ~~0x02 => Phase 2~~ ~~0x03 => Phase 3~~ ~~All others reserved~~ PMREQ: 31:0 - PCIe LTR format All others reserved	

E6 CXL Configuration Space Registers

In chapter 7, Table 57, make the following changes:

Attribute	Description
RO	Read Only
RO-V	Read-Only-Variant
RW	Read-Write
RWS	Read-Write-Sticky
RWO	Read-Write-On~~c~~e-To Lock Field becomes RO after writing one to it.
RWL	Read-Write-Lockable
RW1CS	Read-Write-One-To-Clear-Sticky

In section 7.1.1.3, make the following changes:

Bit	Attributes	Description
13:0	N/A	Reserved (RSVD).
14	RW1CS	Viral_Status: When set, indicates that the CXL device has entered Viral self-isolation mode. See Section 11.4, "CXL Viral Handling" on page 198 for more details. Note: Viral condition is cleared by warm reset or cold reset, but Viral_Status bit survives warm reset.
15	N/A	Reserved (RSVD).

E7 CXLCM Flit Packing Rules

In Section 4.2.5, add following Flit Packing Rules that clarify Bit Ordering requirements.

4.2.5 Flit Packing Rules

. . .

- For a given slot, lower bit positions are defined as bit positions that appear starting from lower order Byte #. That is, bits are ordered starting from (Byte# 0, Bit# 0) through (Byte #15, Bit# 7).
- For multi-bit message fields like Address[MSB:LSB], lesser significant bits will appear in lower order bit positions.
- Message ordering within a flit is based on flit bit numbering, i.e. the earliest messages are placed at the lowest flit bit positions and progressively later messages are placed at progressively higher bit positions. Examples: An M2S Req 0 packed in Slot 0 precedes an M2S Req 1 packed in Slot 1. Similarly, a Snoop packed in Slot 1 follows a GO packed in Slot 0, and this ordering must be maintained. Finally, for Header Slot Format H1, an H2D Response packed starting from Byte# 7 precedes an H2D Response packed starting from Byte# 11.

E8 CXLCM Slot Format Diagram Updates

In Section 4.2.3, following diagrams for Header Slot Formats need to be updated to show Slot 3 bit encodings in correct bit positions. Also fixed Figure 50, which showed Addr[51:6] in CXL1.1, but should have show Addr[51:5] to match address bits required for M2S Req.

4.2.3.2 H2D and M2S Formats

Figure 45. H0 - H2D Req + H2D Resp

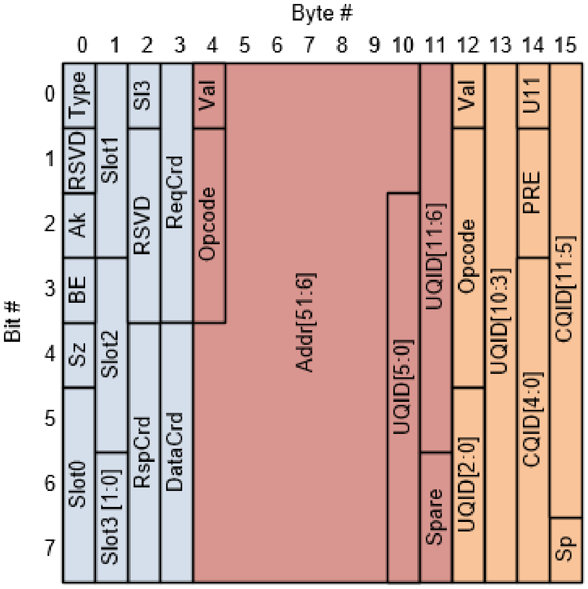


Figure 46. H1 - H2D Data Header + H2D Resp + H2D Resp

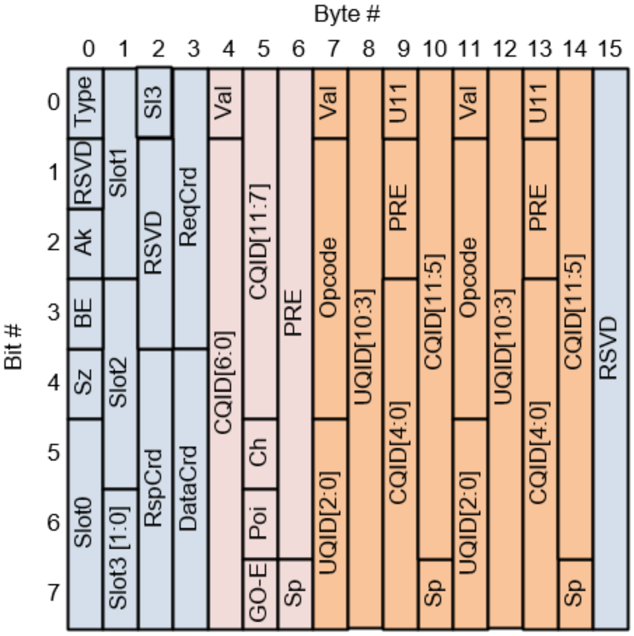


Figure 47. H2 - H2D Req + H2D Data Header

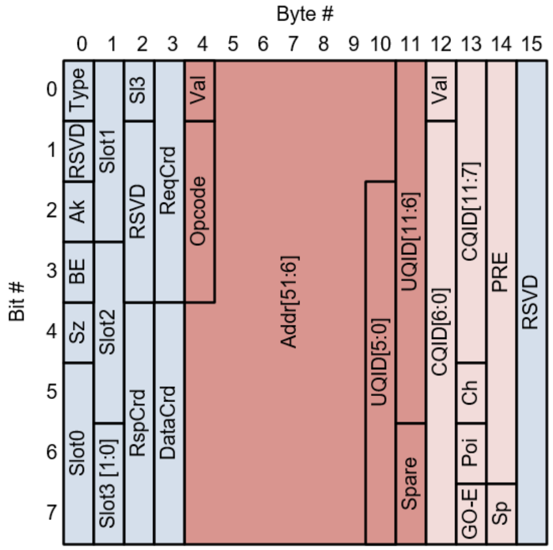


Figure 48. H3 - 4 H2D Data Header

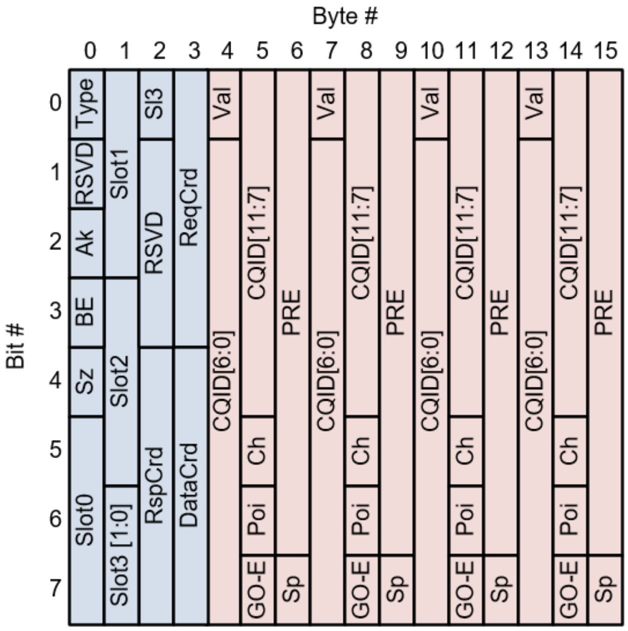


Figure 49. H4 - M2S RwD Header

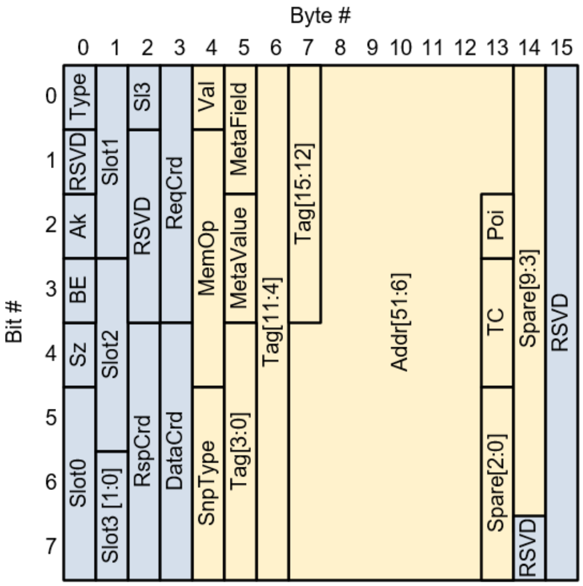
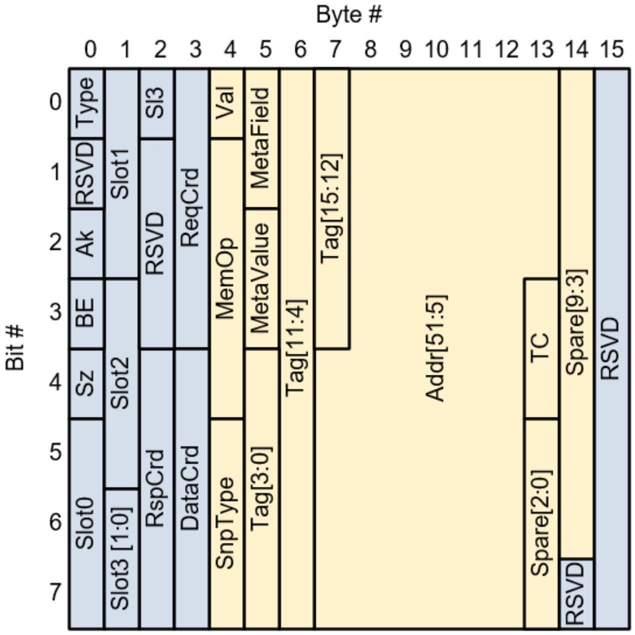


Figure 50. H5 - M2S Req



4.2.3.3 D2H and S2M Formats

Figure 58. H0 - D2H Data Header + 2 D2H Resp + S2M NDR

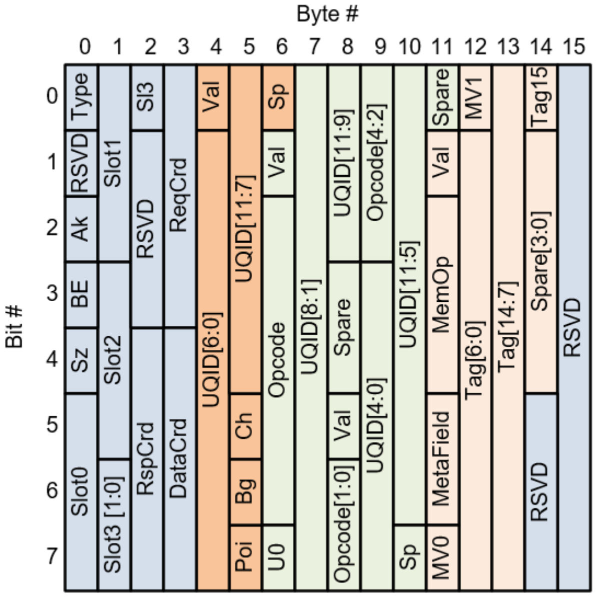


Figure 59. H1 - D2H Req + D2H Data Header

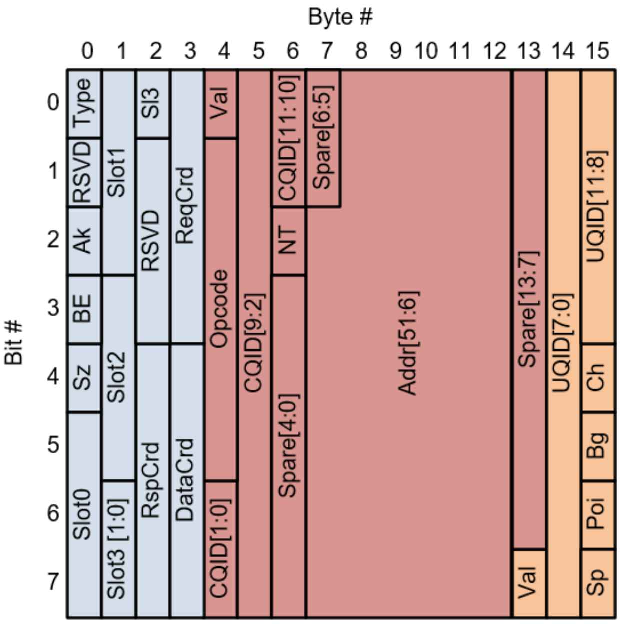


Figure 60. H2 - 4 D2H Data Header + D2H Resp

Bit # \ Byte #	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Type	Slot1	Sl3	ReqCrd	Val	UQID[11:7]	Sp	UQID[11:6]	Poi	UQID[11:5]	Bg	UQID[11:4]	Ch	Opcode[4:3]	UQID[11:6]	RSVD
1	RSVD		RSVD		UQID[6:0]		Val		Sp		Poi		Bg			
2	Ak						UQID[5:0]		Val		Sp		Poi	UQID[5:0]		
3	BE	Slot2							UQID[4:0]		Val		Sp			
4	Sz		RspCrd	DataCrd							UQID[3:0]		Val			
5	Slot0					Ch							Opcode[2:0]			
6		Slot3 [1:0]				Bg		Ch							Spare	
7						Poi		Bg		Ch						

Figure 61. H3 - S2M DRS Header + S2M NDR

Bit # \ Byte #	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Type	Slot1	Sl3	ReqCrd	Val	Tag[7:0]	Tag[15:8]	Poi	Spare[14:7]	Val	Tag[7:0]	Tag[15:8]	Spare[3:0]	RSVD	RSVD	RSVD
1	RSVD		RSVD		MemOp			Spare[6:0]		MemOp						
2	Ak															
3	BE	Slot2														
4	Sz		RspCrd	DataCrd	MetaField					MetaField			RSVD			
5	Slot0															
6		Slot3 [1:0]			MetaValue					MetaValue						
7																

Figure 62. H4 - 2 S2M NDR

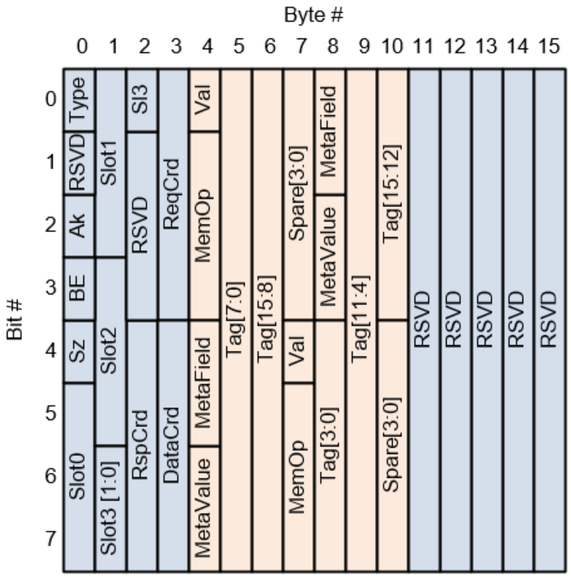
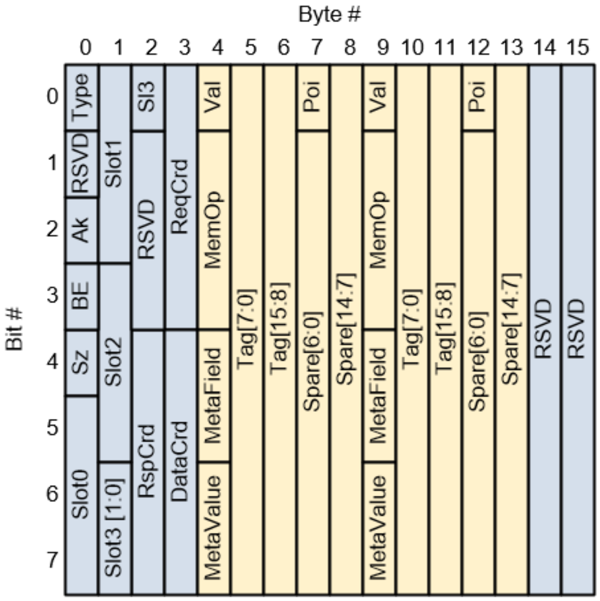


Figure 63. H5 - 2 S2M DRS



E9 CLFlush Definition and Clarification for Biasing

In section 3.2.4.1.13, make the following updates:

3.2.4.1.13 ClFlush

This is a request to the Host to invalidate the cache-line specified in the address field. The typical response is GO-I that will be sent from the Host upon completion in memory.

However, the Host may keep tracking the cache line in Shared state if the Core has issued a Monitor to an address belonging in the cache line. Thus, the Device must not rely on CLFlush/GO-I as an only and sufficient condition to flip a cache line from Host to Device bias mode. Instead, the Device must initiate RdOwnNoData and receive an H2D Response of GO-E before it updates its Bias Table and may subsequently access the cache-line without notifying the Host.

Under error conditions, a ClFlush request may receive the line in the Error (GO-Err) state. The device is responsible for handling the error appropriately.

E10 Buried Cache State Rules and Multiple Access Over CXL.cache

Add a Section 3.2.5.14 for Buried Cache State Rules. Update text in sections 3.2.5.6, 3.2.5.7, and 3.2.5.8

3.2.5.14 Buried Cache State Rules

Whenever the Device initiates a new request on CXL.Cache protocol, Buried Cache state refers to the state of the cache line registered in the Device's Coherency engine (DCOH) for which a particular request is being sent.

Buried Cache State Rules:

- The Device must not issue a Read for a cache line if it is buried in Modified, Exclusive, or Shared state.
- The Device must not issue RdOwnNoData if the cache line is buried in Modified or Exclusive state. The Device may request for ownership in Exclusive state as an upgrade request from Shared state.
- The Device must not issue a Read0-Write if the cache line is buried in Modified, Exclusive, or Shared state. The Host typically responds with GO-I (downgraded state) for such requests.
- All *Evict opcodes must adhere to apropos use case. For example, the Device is allowed to issue DirtyEvict for a cache line only when it is buried in Modified state. For performance benefits, it is recommended that the Device should not silently drop a cache line in Exclusive or Shared state and instead use CleanEvict* opcodes towards the Host.
- The CacheFlushed Opcode is not specific to a cache line, it is an indication to the Host that all of the Device's caches are flushed. Thus, the Device must not issue CacheFlushed if there is any cache line buried in Modified, Exclusive, or Shared state.

Table 20E describes which Opcodes in D2H requests are allowed for a given Buried Cache State:

Table 20E: Allowed Opcodes Per Buried Cache State

D2H Requests		Buried Cache State			
Opcodes	Semantic	Modified	Exclusive	Shared	Invalid
RdCurr	Read				✓
RdOwn	Read				✓

RdShared	Read				√
RdAny	Read				√
RdOwnNoData	Read0			√	√
ItoMWr	Read0-Write				√
MemWr	Read0-Write				√
CLFlush	Read0				√
CleanEvict	Write		√		
DirtyEvict	Write	√			
CleanEvictNoData	Write		√	√	
WOWrInv	Write				√
WOWrInvF	Write				√
WrInv	Write				√
CacheFlushed	Read0				√

3.2.5.6 Multiple Reads to the same cache line

Multiple read requests (cacheable or uncacheable) to the same cache line are allowed only in the following specific cases where host tracking state is consistent regardless of the order requests are processed. The Host can freely reorder requests, so the device is responsible for ordering requests when required. For host memory, multiple RdCurr and/or CLFlush are allowed. For these commands the device ends in I-state, so there is no inconsistent state possible for host tracking of a device cache. With Type 2 devices, in addition to RdCurr and/or CLFlush, multiple RdOwnNoData (bias flip request) is allowed for device attached memory. This case is allowed because with device attached memory the host does not track the device's cache so re-ordering in the host will not create ambiguous state between device and host.

3.2.5.7 Multiple Evicts to the same cache line

~~Multiple Evicts to the same cache line are not allowed. The second Evict may only be issued after the first receives both the CXL.cache GO-I response and the WritePull.~~

~~Since Evict guarantees that the evicted cache line is otherwise in the initiating device, it is impossible to send another Evict without an intervening cacheable Read/Read0 request to that address.~~

Multiple Evicts to the same cache line are not allowed. All Evict messages from the Device provide a guarantee to the Host that the evicted cache line will no longer be present in the Device’s caches. Thus, it is impossible to send another Evict for the same cache line without an intervening cacheable Read/Read0 request to that address.

3.2.5.8 Multiple Write Requests to the same cache line

~~Multiple WrInv/WOWrInv/ItoMWr/MemWr to the same cache line are allowed to be outstanding on CXL.cache. The Host can freely reorder requests, so the device is responsible for ordering requests when required.~~

Multiple WrInv/WOWrInv/ItoMWr/MemWr to the same cache line are allowed to be outstanding on CXL.cache. The Host can freely reorder requests, and the Device may receive corresponding H2D Responses in reordered fashion. However, it is generally recommended that the Device should issue no more than one outstanding Write request for a given cache line, and order multiple write requests to the same cache line one after another whenever stringent ordering is warranted.

E11 ARB/MUX Virtual LSM Resolution Table Clarifications

In Section 5.1, modifications are applied to Table 47 and the Note below it, as follows:

Table 47. ARB/MUX Multiple Virtual LSM Resolution Table

Resolved Request from ARB/MUX to Flex Bus Physical Layer (Row = current vLSM[0] state; Column = current vLSM[1] state)	**Reset**	**Active**	**L1.1**	**L1.2**	**L1.3**	**L1.4**	**SLEEP_L2**
Reset	RESET	Active	L1.1	L1.2 or lower	L1.3 or lower	L1.4 or lower	SLEEP_L2
Active	Active	Active	Active	Active	Active	Active	Active
L1.1	L1.1	Active	L1.1	L1.1	L1.1	L1.1	L1.1
L1.2	L1.2 or lower	Active	L1.1	L1.2 or lower	L1.2 or lower	L1.2 or lower	L1.2 or lower
L1.3	L1.3 or lower	Active	L1.1	L1.2 or lower	L1.3 or lower	L1.3 or lower	L1.3 or lower
L1.4	L1.4 or lower	Active	L1.1	L1.2 or lower	L1.3 or lower	L1.4 or lower	L1.4 or lower
SLEEP_L2	SLEEP_L2	Active	L1.1	L1.2 or lower	L1.3 or lower	L1.4 or lower	SLEEP_L2

~~Note: Table 47 is presented as a suggestion, not a requirement.~~

E12 Text Updates in CXLCM Link Layer Chapter

In sections 4.2.8.1 and 4.2.7, make the following updates:

4.2.8.1 LLR Variables

- WrPtr: This indexes the entry of the LLRB that will record the next new flit. When an entity sends a flit, it copies that flit into the LLRB entry indicated by the WrPtr and then increments the WrPtr by one (modulo the size of the LLRB). This is implemented using a wrap-around counter that wraps around to 0 after reaching the depth of the LLRB. Certain LLCTRL flits do not affect the WrPtr. WrPtr stops incrementing after receiving an error indication at the remote entity (RETRY.Req message), until normal operation resumes again (all flits from the LLRB have been retransmitted). WrPtr is initialized to 0 and is incremented only when a flit is put into the LLRB.

  Implementation Note: WrPtr may continue to increment after receiving Retry.Req message if there are pre-scheduled All Data Flits that are not yet sent over the link. This implementation willl ensure that All Data Flits not interleaved with other flits are correctly logged into the Link Layer Retry Buffer.

- NumFreeBuf: This indicates the number of free LLRB entries at the entity. NumFreeBuf is decremented by 1 whenever an LLRB entry is used to store a transmitted flit. NumFreeBuf is incremented by the value encoded in the Ack/ Full_Ack field of a received flit. NumFreeBuf is initialized at reset time to the size of the LLRB. The maximum number of retry queues at any entity is limited to 255 (8 bit counter). Also, note that the retry buffer at any entity is never filled to its capacity, therefore NumFreeBuf is never ‘0. If there is only 1 retry buffer entry available, then the sender cannot send an ACK bearing flit which may either be a protocol flit or an LLCRD Control flit. This restriction is required to avoid ambiguity between a full or an empty retry buffer during a retry sequence that may result into incorrect operation. This implies if there are only 2 retry buffer entries left (NumFreeBuf = 2), then the sender can send an Ack bearing flit only if the outgoing flit encodes a value of at least 1 (which may be a Protocol flit with Ak bit set), else a LLCRD control flit is sent with Full_Ack value of at least 1. This is required to avoid deadlock at the link layer due to retry buffer becoming full at both entities on a link and their inability to send ACK through header flits.

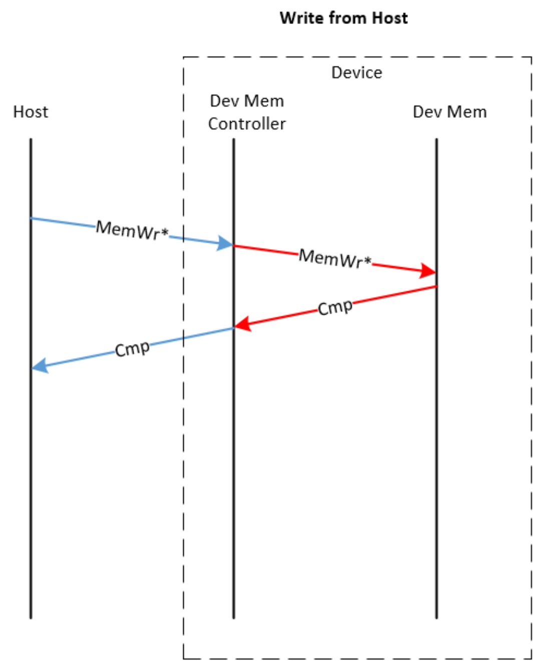
4.2.7 Link Layer Initialization

- The Tx portion of the Link Layer must wait until the Rx portion of the Link Layer has received at least one valid flit that is CRC clean before sending the LLCTRLINIT.Param flit. Before this condition is met, the Link Layer must transmit only LLCTRL-Retry flits, i.e., Retry.Frame/Req/Ack/Idle flits.

E13 Fixed Typos in Figure depicting Write from Host

Update Figure 39, Red arrows from Device Memory Controller to Device Memory should be MemWr and Cmp*

Figure 39. Write from Host



E14 S2M/D2H G5 Format Update

Updates to S2M/D2H G5 format from 3 NDR to 2 NDR messages. The global flit rules in section 4.2.5 restrict total S2M NDR to 2 total per flit which take precedence, so it is impossible to make use of 3rd NDR message in the S2M/D2H G5 format. Changing the G5 format to only include 2 NDR is done to ensure consistency and avoid confusion.

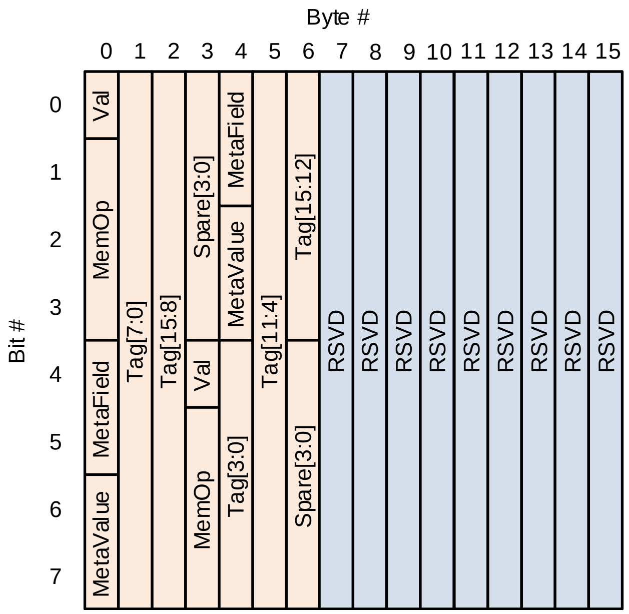
First change is to Table 40. Second change is to change G5 format in Figure 70 to include only the first 2 NDR messages converting the 3rd to RSVD bits.

Table 40. D2H/S2M Slot Formats

Format to Req Type Mapping	**D2H/S2M**	
	Type	**Size**
H0	CXL.cache Data Header + 2 CXL.cache Resp + CXL.mem NDR	85
H1	CXL.cache Req + CXL.cache Data Header	96
H2	4 CXL.cache Data Header + CXL.cache Resp	88
H3	CXL.mem DRS Header + CXL.mem NDR	68
H4	2 CXL.mem NDR	56

H5	2 CXL.mem DRS Header	80
G0	CXL.cache/ CXL.mem Data Chunk	128
G1	CXL.cache Req + 2 CXL.cache Resp	119
G2	CXL.cache Req + CXL.cache Data Header + CXL.cache Resp	116
G3	4 CXL.cache Data Header	68
G4	CXL.mem DRS Header + 2 CXL.mem NDR	96
G5	~~3~~ 2 CXL.mem NDR	~~84~~ 56
G6	3 CXL.mem DRS Header	120

Figure 70. G5 - ~~3~~ 2 S2M NDR



E15 PM timeout value.

CXL1.1 had a "TBD" value in the PM handling in section 9.4.2. We are updating this value to be 1ms with pause in timer for recovery states.

9.4.2 Compute Express Link PM Entry Phase 2

. . .

6. The Downstream Component ARB&MUX port must wait for ~~<TBD> amount of time~~ at least 1mS (not including time spent in recovery states) for a response from the Upstream Component. If no response is received from the Upstream component then the Downstream Component is permitted to abort the PM entry or retry entry into PM again.

E16 Figure 65 Title

Figure 65, which is the format for byte enable slot, incorrectly includes "S2M" in the title. CXL.mem S2M messages cannot carry byte enables as defined in table 36. Update to the title removes S2M. Note that M2S message can carry Byte Enables and is captured in Figure 52.

Figure 65. G0 - D2H~~/S2M~~ Byte Enable