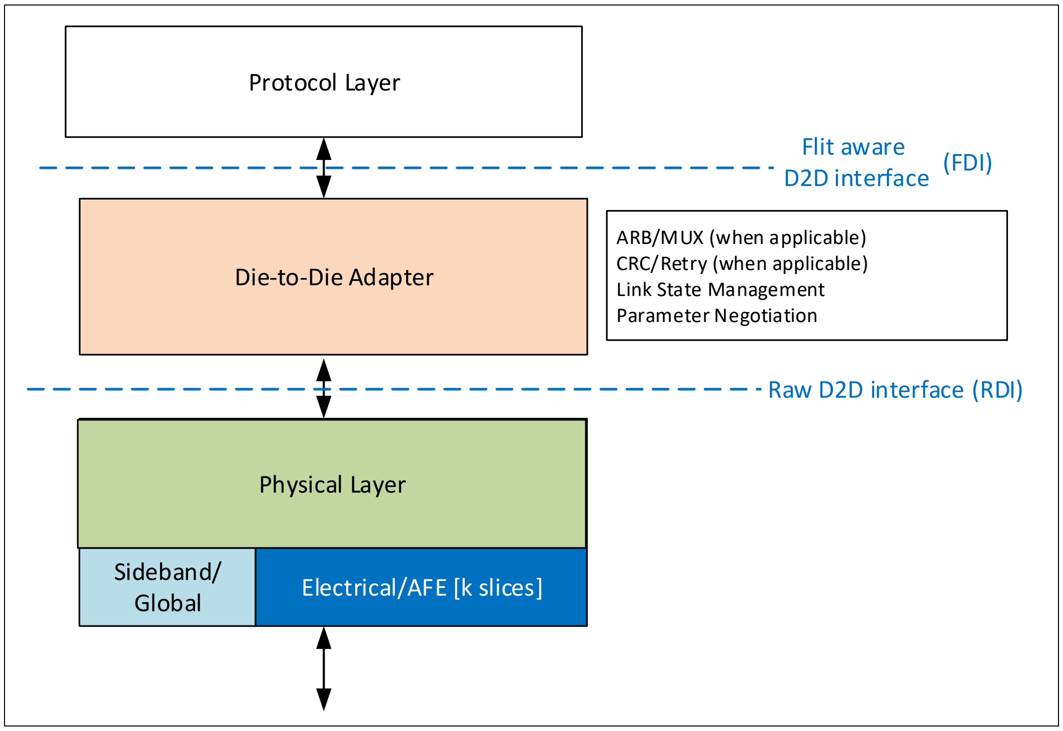
3.0 Die-to-Die Adapter

The Die-to-Die Adapter is responsible for:

- Reliable data transfer (performing CRC computation and Retry, or parity computation when applicable)
- Arbitration and Muxing (in case of multiple Protocol Layers)
- Link State Management
- Protocol and Parameter negotiation with the remote Link partner.

Figure 3-1 shows a high level description of the functionality of the Adapter.

Figure 3-1. Functionalities in the Die-to-Die Adapter



The Adapter interfaces to the Protocol Layer using one or more instances of the Flit-aware Die-to-Die interface (FDI), and it interfaces to the Physical Layer using the raw Die-to-Die interface (RDI). See Chapter 10.0 for interface details and operation.

The D2D Adapter must follow the same rules as the Protocol Layer for protocol interoperability requirements. Figure 3-2 shows example configurations for the Protocol Layer and the Adapter, where the Protocol identifiers (e.g., PCIe) only signify the protocol, and not the Flit Formats. To provide cost

Property of Universal Chiplet Interconnect Express (UCIe) © 2022–2024

and efficiency trade-offs, UCIe allows configurations in which two protocol stacks are multiplexed onto the same physical Link.

3.1 Stack Multiplexing

If the Multi_Protocol_Enable parameter is negotiated, two stacks multiplexed on the same physical Link is supported when each protocol stack needs half the bandwidth that the Physical Layer provides. Both stacks must be of the same protocol with the same protocol capabilities. When Multi_Protocol_Enable and Management Transport protocol are negotiated for mainband and the Protocol Layer implements the Management Port Gateway multiplexer (MPG mux), the MPG mux must be present on both stacks and the same protocols must be present in both stacks. For example, in Figure 8-27 the Multi_Protocol_Enable parameter can be negotiated for config b if both stacks in this configuration have PCIe or both stacks have Streaming. Similarly, the Multi_Protocol_Enable parameter can be negotiated for config d in Figure 8-27 if both CXL stacks are identical.

When Multi_Protocol_Enable is supported and negotiated, the Adapter must guarantee that it will not send consecutive flits from the same protocol stack on the Link. This applies in all cases including when Flits are sourced from FDI, from Retry Buffer, and when the data stream is paused and restarted. Adapter is permitted to insert NOP Flits to guarantee this (these Flits bypass the Tx Retry buffer, and are not forwarded to the Protocol Layer on the receiver). When Flits are transmitted from the Retry Buffer, it is required to insert NOP Flits as needed to avoid sending consecutive Flits from the same Protocol stack. When Management Transport protocol is negotiated for mainband with Multi_Protocol_Enable, the Management Flit carries the same stack identifier as the Protocol Layer it is multiplexed with. From the Adapter perspective, for the purposes of throttling and interleaving, it is treated the same as flits received from the corresponding Protocol Layer. Note that there is no fixed pattern of Flits alternating from different Protocol Layers. For example, a Flit from Protocol Stack 0 followed by a NOP Flit, followed by a Flit from Protocol Stack 0 is a valid transmit pattern. A NOP Flit is defined as a Flit where the protocol identifier in the Flit Header corresponds to the D2D Adapter, and the body of the Flit is filled with all 0 data (the NOP Flit is defined for all Flit Formats supported by the Adapter, for all cases when it is operating in Raw Format). It is permitted for NOP flits to bypass the Retry buffer, as long as the Adapter guarantees that it is not sending consecutive Flits for any of the Protocol Layers. On the receiving side, the Adapter must not forward these NOP flits to the Protocol Layer. The receiving Protocol Layer must be capable of receiving consecutive chunks of the same Flit at the maximum Link speed, but it will not receive consecutive Flits. In addition to the transfer rate, both protocol stacks must operate with the same protocol and Flit Formats. Multi_Protocol_Enable and Raw Format are mutually exclusive. Each stack is given a single bit stack identifier that is carried along with the Flit header for de-multiplexing of Flits on the Receiver. The Stack Mux shown maintains independent Link state machines for each protocol stack. Link State transition-related sideband messages have unique message codes to identify which stack's Link State Management is affected by that message.

IMPLEMENTATION NOTE

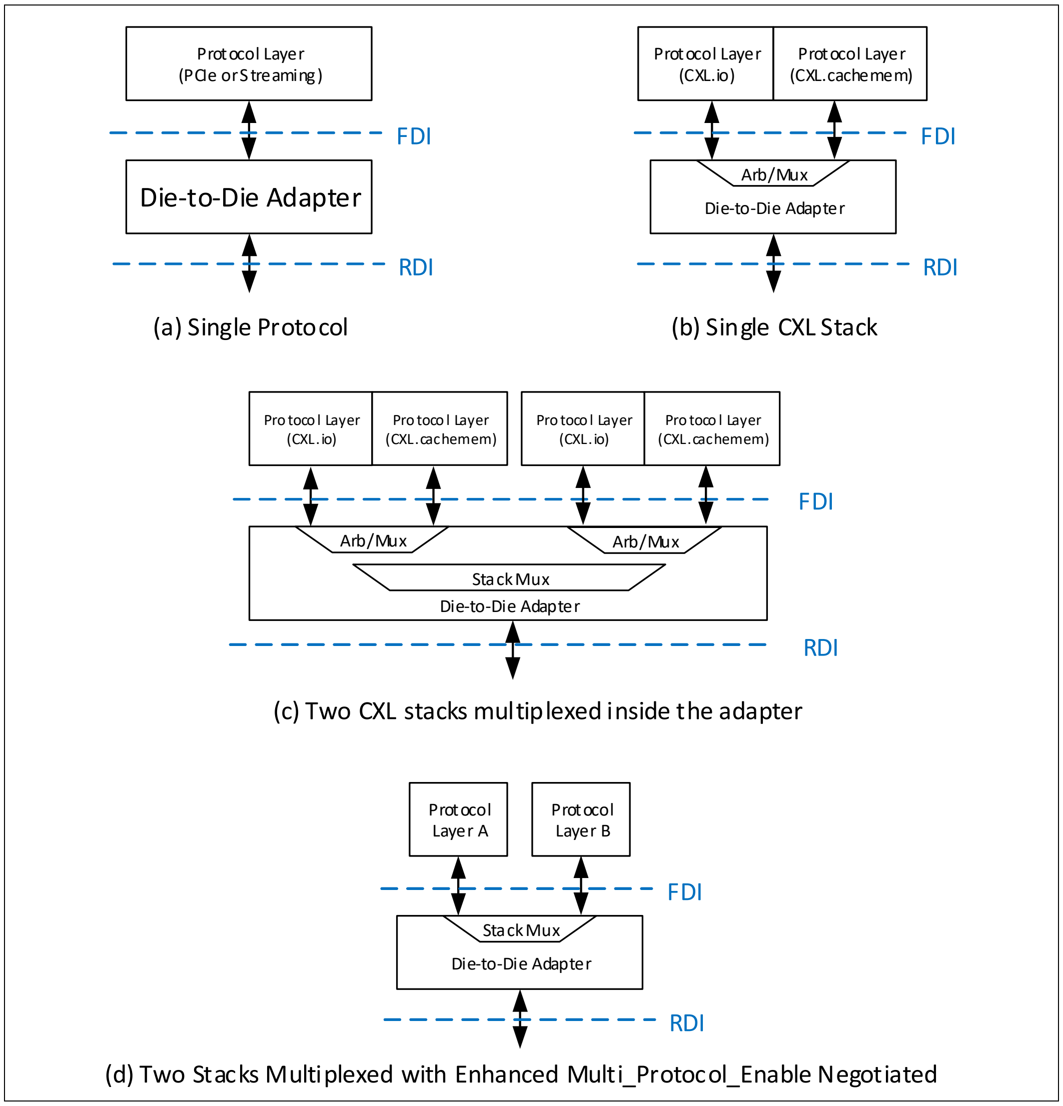
The primary motivation for enabling the Multi_Protocol_Enable parameter is to allow implementations to take advantage of the higher bandwidth provided by the UCIe Link for lower-bandwidth individual Protocol Layers, without the need to make a lot of changes to the UCIe Link. For example, two Protocol Layers that support the maximum bandwidth for CXL 68B Flit Mode (i.e., the equivalent of 32 GT/s CXL SERDES bandwidth) can be multiplexed over a UCIe Link that supports their aggregate bandwidth.

If the Enhanced Multi_Protocol_Enable parameter is negotiated, dynamic multiplexing between two stacks of the same or different protocols on the same physical Link is supported. When Enhanced

Property of Universal Chiplet Interconnect Express (UCIe) © 2022–2024

Multi_Protocol_Enable and Management Transport protocol are negotiated, each stack can have different protocols with or without MPG mux. For example, in Figure 8-27, the Enhanced Multi_Protocol_Enable parameter must be negotiated for configs e, f, and h. The parameter is also negotiated for configs b and d if the two stacks have different protocol pairs. Both protocol stacks and the Adapter must support a common Flit Format for this feature to be enabled. "Enhanced Multi_Protocol_Enable" and Raw Format are mutually exclusive. The Adapter must advertise the maximum percentage of bandwidth that the receiver for each Protocol Layer can accept. The Adapter transmitter must support 100% (no throttling) and throttling one or both Protocol Layer(s) to 50% of maximum bandwidth. When 50% of the maximum bandwidth is advertised for a stack by an Adapter, the remote Link partner must guarantee that it will not send consecutive Flits for the same stack on the Link. This applies in all cases including when Flits are sourced from FDI, from Retry Buffer, and when the data stream is paused and restarted. Adapter is permitted to insert NOP Flits to guarantee this (these Flits bypass the Tx Retry buffer, and are not forwarded to the Protocol Layer on the receiver). When Flits are transmitted from the Retry Buffer, it is required to insert NOP Flits as needed to avoid exceeding the negotiated maximum bandwidth. The receiving Protocol Layer must be capable of sinking Flits at the advertised maximum bandwidth percentage; in addition, Protocol Layer must be able to receive consecutive chunks of the same Flit at the maximum advertised Link speed. When this capability is supported, the Adapter must be capable of allowing each Protocol Layer to independently utilize 100% of the Link bandwidth. Furthermore, the arbitration is per Flit, and the Adapter must support round robin arbitration between the Protocol Layers if both of them are permitted to use 100% of the Link bandwidth. Additional implementation specific arbitration schemes are permitted as long as they are per Flit and do not violate the maximum bandwidth percentage advertised by the remote Adapter for a given stack. The Flit header has a single bit stack identifier to identify the destination stack for the flit. The Stack Mux maintains independent Link state machines for each protocol stack. Link State transition-related sideband messages have unique message codes to identify which stack's Link State Management is affected by that message.

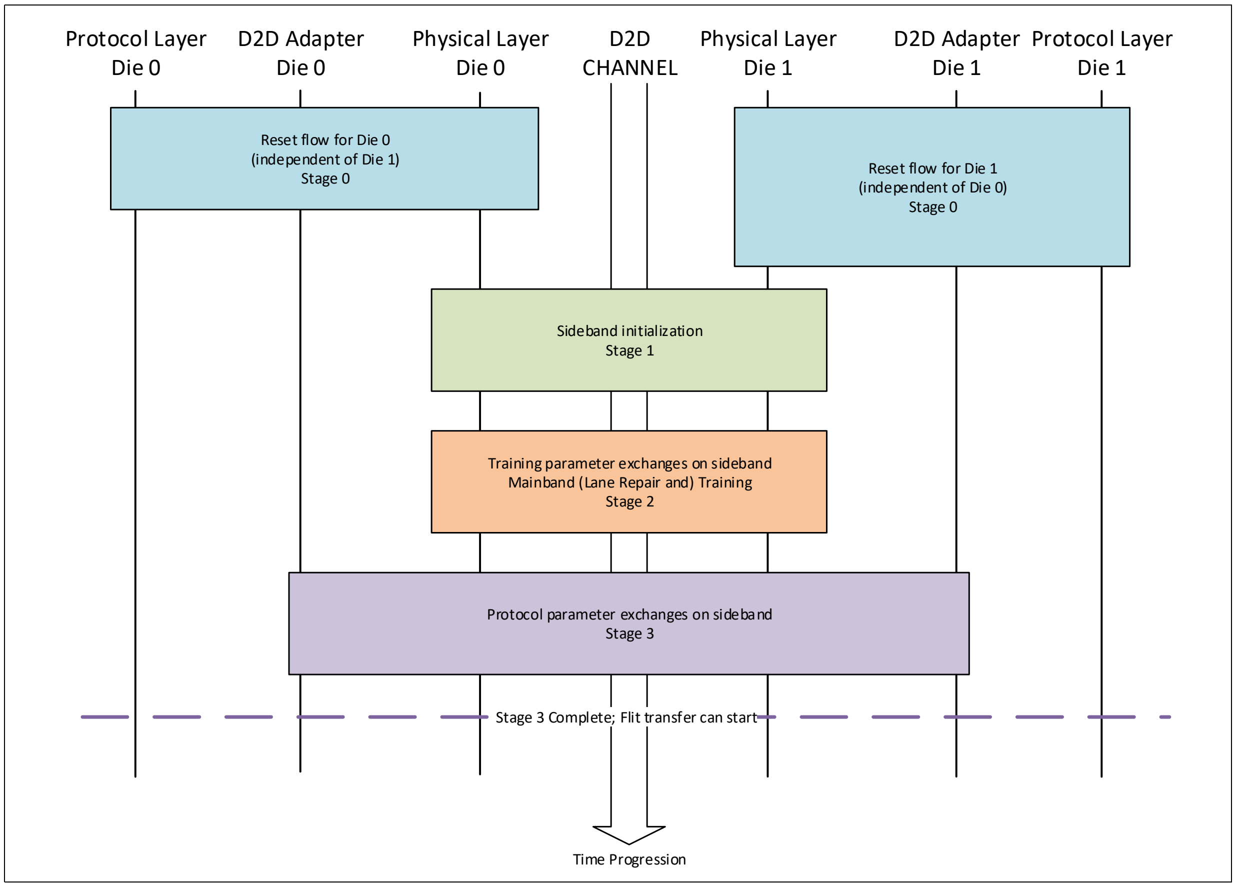
Figure 3-2. Example Configurations



3.2 Link Initialization

Link Initialization consists of four stages before protocol Flit transfer can begin on mainband. Figure 3-3 shows the high-level steps involved in the Link initialization flow for UCIe. Stage 0 is die-specific and happens independently for each die; the corresponding boxes in Figure 3-3 are of different sizes to denote that different die can take different amount of time to finish Stage 0. Stage 1 involves sideband initialization. Stage 2 involves mainband training and repair. Details of Stage 1 and Stage 2 are provided in Chapter 4.0. Stage 3 involves parameter exchanges between Adapters to negotiate the protocol and Flit Formats and is covered in Section 3.2.1.

Figure 3-3. Stages of UCIe Link Initialization


Protocol Layer Die 0
D2D Adapter Die 0
Physical Layer Die 0
D2D CHANNEL
Physical Layer Die 1
D2D Adapter Die 1
Protocol Layer Die 1
Reset flow for Die 0 (independent of Die 1) Stage 0
Reset flow for Die 1 (independent of Die 0) Stage 0
Sideband initialization Stage 1
Training parameter exchanges on sideband Mainband (Lane Repair and) Training Stage 2
Protocol parameter exchanges on sideband Stage 3
Stage 3 Complete; Flit transfer can start
Time Progression

3.2.1 Stage 3 of Link Initialization: Adapter Initialization

Stage 2 is complete when the RDI state machine moves to Active State. The initialization flow on RDI to transition the state from Reset to Active is described in Section 10.1.6. Once Stage 2 is complete, the Adapter must follow a sequence of steps in order to determine Local Capabilities, complete Parameter Exchanges, and bring FDI state machine to Active.

3.2.1.1 Part 1: Determine Local Capabilities

The Adapter must determine the results of Physical Layer training and if Retry is needed for the given Link speed and configuration. See Section 3.8 for the rules on when Retry must be enabled for Link operation. If the Adapter is capable of supporting Retry, it must advertise this capability to the remote

Link partner during Parameter Exchanges. For UCIe Retimers, the Adapter must also determine the credits to be advertised for the Retimer Receiver Buffer. Each credit corresponds to 256B of Mainband data storage.

3.2.1.2 Part 2: Parameter Exchange with Remote Link Partner

The following list of capabilities must be negotiated between Link partners. The capabilities (if enabled) are transmitted to the remote Link partner using a sideband message. In the section below, "advertised" means that the corresponding bit is 1b in the {AdvCap.Adapter} sideband message.

Table 3-1. Capabilities that Must Be Negotiated between Link Partners (Sheet 1 of 2)

Capability	Description and Requirements
"Raw Format"	This parameter is advertised if the corresponding bit in the UCIe Link Control register is 1b. Software/Firmware enables this based on system usage scenario. If the PCIe or CXL protocols are not supported, and Streaming protocol is to be negotiated without any vendor-specific extensions and without Streaming Flit Format capability support, "Raw Format" must be 1b and advertised. If Streaming Flit Format capability or Enhanced Multi-Protocol capability is supported, then this must be advertised as 1b only if Raw Format is the intended format of operation. Software/firmware-based control on setting the corresponding UCIe Link Control is permitted to enable this.
"68B Flit Mode"	This is a protocol parameter. This must be advertised if the Adapter and Protocol Layer support CXL 68B Flit mode (mandatory for CXL) or PCIe Non-Flit mode (mandatory for PCIe). If PCIe Non-Flit mode is the final negotiated protocol, it will use the CXL.io 68B Flit mode formats as defined in *CXL Specification*. This is an advertised Protocol for Stack 0 if "Enhanced Multi_Protocol_Enable" is supported and enabled.
"CXL 256B Flit Mode"	This is a protocol parameter. This must be advertised if the Adapter and Protocol Layer support CXL 256B Flit mode. This is an advertised Protocol for Stack 0 if Enhanced Multi-Protocol capability is supported and enabled.
"PCIe Flit Mode"	This is a protocol parameter. This must be advertised if the Adapter and Protocol Layer support PCIe Flit mode. This is an advertised Protocol for Stack 0 if Enhanced Multi-Protocol capability is supported and enabled.
"Streaming"	This is a protocol parameter. This must be advertised if the Adapter and Protocol Layer support Streaming protocol in Raw Format or Streaming Flit Format capability is supported and the corresponding capabilities are enabled. This is an advertised Protocol for Stack 0 if Enhanced Multi-Protocol capability is supported and enabled. PCIe or CXL protocol must not be advertised if Streaming is advertised for a given Protocol Layer.
"Retry"	This must be advertised if the Adapter supports Retry. With the exception of the Link operating in Raw Format, the Link cannot be operational if the Adapter has determined Retry is needed, but "Retry" is not advertised or negotiated. See also Section 3.8.
"Multi_Protocol_Enable"	This must only be advertised if the Adapter is connected to multiple FDI instances corresponding to two sets of Protocol Layers. It must only be advertised if the Adapter (or SoC firmware in Stage 0 of Link Initialization) has determined that the UCIe Link must be operated in this mode. Both "Stack0_Enable" and "Stack1_Enable" must be 1b if this bit is advertised.
"Stack0_Enable"	This must be advertised if the Protocol Layer corresponding to Stack 0 exists and is enabled for operation with support for the advertised protocols.
"Stack1_Enable"	This must be advertised if the Protocol Layer corresponding to Stack 1 exists and is enabled for operation with support for the advertised protocols.
"CXL_LatOpt_Fmt5"	This must be advertised if the Adapter and Protocol Layer support *Format 5* defined in Section 3.3.4. The Protocol Layer does not take advantage of the spare bytes in this Flit Format. This must not be advertised if CXL protocol and CXL 256B Flit mode are not supported or enabled.
"CXL_LatOpt_Fmt6"	This must be advertised if the Adapter and Protocol Layer support *Format 6* defined in Section 3.3.4. The Protocol Layer takes advantage of the spare bytes in this Flit Format. This must not be advertised if CXL protocol and CXL 256B Flit mode are not supported or enabled.
"Retimer"	This must be advertised if the Adapter of a UCIe Retimer is performing Parameter Exchanges with a UCIe Die within its package.
"Retimer_Credits"	This is a 9-bit value advertising the total credits available for Retimer's Receiver Buffer. Each credit corresponds to 256B data. This parameter is applicable only when "Retimer" is 1b.

Table 3-1. Capabilities that Must Be Negotiated between Link Partners (Sheet 2 of 2)

Capability	Description and Requirements
“DP”	This is set by Downstream Ports to inform the remote Link partner that it is a Downstream Port. It is useful for Retimers to identify whether they are connected to a Downstream Port UCIe die. It is currently only applicable for PCIe and CXL protocols; however, Streaming protocols are not precluded from utilizing this bit. If Enhanced Multi-Protocol capability is supported, this bit is applicable if either of the Protocol Layers is PCIe or CXL. This bit must be set to 0b if “Retimer” is set to 1b.
“UP”	This is set by Upstream Ports to inform the remote Link partner that it is an Upstream Port. It is useful for Retimers to identify whether they are connected to an Upstream Port UCIe die. It is currently only applicable for PCIe and CXL protocols; however, Streaming protocols are not precluded from utilizing this bit. If Enhanced Multi-Protocol capability is supported, this bit is applicable if either of the Protocol Layers is PCIe or CXL. This bit must be set to 0b if “Retimer” is set to 1b.
“68B Flit Format”	This must be advertised if any of the following are true: • Enhanced Multi-Protocol capability is supported and enabled, AND the 68B Flit Format is supported and enabled • The 68B Flit Format for Streaming Protocol capability is supported and enabled Otherwise, it must be set to 0b.
“Standard 256B End Header Flit Format”	This must be advertised if any of the following are true: • Enhanced Multi-Protocol capability is supported and enabled, AND the Standard 256B End Header Flit Format is supported and enabled • The Standard 256B End Header Flit Format for Streaming Protocol capability is supported and enabled Otherwise, it must be set to 0b.
“Standard 256B Start Header Flit Format”	This must be advertised if any of the following are true: • PCIe Flit mode is advertised and Standard Start Header for PCIe protocol capability is supported and enabled • Enhanced Multi-Protocol capability is supported and enabled, AND the Standard 256B Start Header Flit Format is supported and enabled • The Standard 256B Start Header Flit Format for Streaming Protocol capability is supported and enabled Otherwise, it must be set to 0b.
“Latency-Optimized 256B without Optional Bytes Flit Format”	This must be advertised if any of the following are true: • Enhanced Multi-Protocol capability is supported and enabled, AND the Latency-Optimized 256B without Optional Bytes Flit Format is supported and enabled • The Latency-Optimized 256B without Optional Bytes Flit Format for Streaming Protocol capability is supported and enabled Otherwise, it must be set to 0b.
“Latency-Optimized 256B with Optional Bytes Flit Format”	This must be advertised if any of the following are true: • PCIe Flit mode is advertised and Latency-Optimized Flit with Optional Bytes for PCIe protocol capability is supported and enabled • Enhanced Multi-Protocol capability is supported and enabled, AND the Latency-Optimized 256B with Optional Bytes Flit Format is supported and enabled • The Latency-Optimized 256B with Optional Bytes Flit Format for Streaming Protocol capability is supported and enabled Otherwise, it must be set to 0b.
“Enhanced Multi_Protocol_Enable”	This must only be advertised if the Adapter is connected to multiple FDI instances corresponding to two sets of Protocol Layers. The two sets of Protocol Layers are permitted to be different protocols, but must support at least one common Flit Format. This must only be advertised if the Enhanced Multi-Protocol capability is supported and enabled; otherwise, it must be set to 0b. Both “Stack0_Enable” and “Stack1_Enable” must be 1b if this bit is advertised.
“Stack 0 Maximum Bandwidth_Limit”	This must be advertised if Enhanced Multi_Protocol_Enable is advertised and the Stack 0 protocol Receiver is limited to 50% of the maximum bandwidth; otherwise, it must be set to 0b.
“Stack 1 Maximum Bandwidth_Limit”	This must be advertised if Enhanced Multi_Protocol_Enable is advertised and the Stack 1 protocol Receiver is limited to 50% of the maximum bandwidth; otherwise, it must be set to 0b.
“Management Transport Protocol”	This bit must be set to 1 if the Protocol Layer and Adapter both support Management Transport protocol (either as the only protocol or multiplexed with one of CXL.io, PCIe, or Streaming). The mechanism by which this bit is set to 1 is implementation-specific.

Once local capabilities are established, the Adapter sends the {AdvCap.Adapter} sideband message advertising its capabilities to the remote Link partner.

If PCIe or CXL protocol support is going to be advertised, the Upstream Port (UP) Adapter must wait for the first {AdvCap.Adapter} message from the Downstream Port (DP) Adapter, review the capabilities advertised by DP and then send its own sideband message of advertised capabilities. UP is permitted to change its advertised capabilities based on DP capabilities. Once the DP receives the capability advertisement message from the UP, the DP responds with the Finalized Configuration using {FinCap.Adapter} sideband message to the UP as shown in Figure 3-4. See Section 7.1.2.3 to see the message format for the relevant sideband messages.

Final determination for Protocol parameters:

- If "68B Flit Mode" is advertised by both Link partners, it is set to 1 in the {FinCap.Adapter} message
- If "CXL 256B Flit Mode" is advertised by both Link partners, it is set to 1 in the {FinCap.Adapter} message
- If "PCIe Flit Mode" is advertised by both Link partners, "PCIe Flit Mode" bit is set to 1 in the {FinCap.Adapter} message
- If Streaming protocol is negotiated, no {FinCap.Adapter} messages are exchanged for that stack.

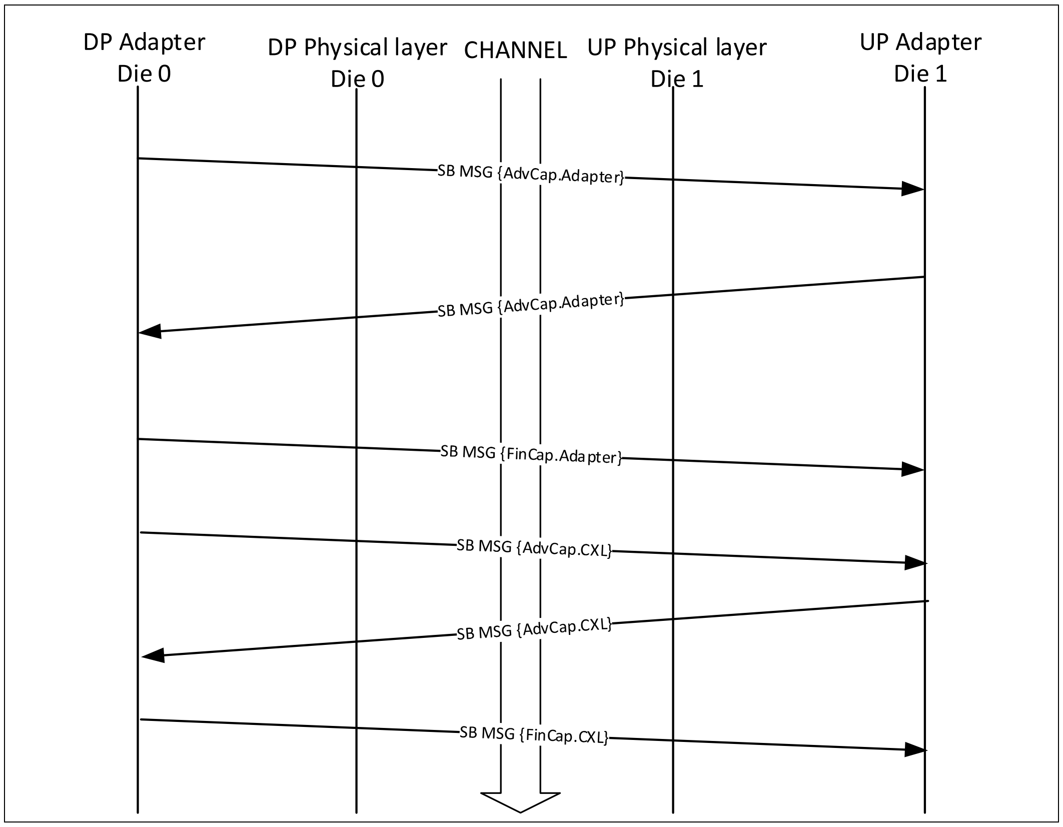
If "68B Flit Mode" or "CXL 256B Flit Mode" is set in the {FinCap.Adapter} message, there must be another handshake of Parameter Exchanges using the {AdvCap.CXL} and the {FinCap.CXL} messages to determine the details associated with this mode. Note that because CXL 68B Flit Mode protocol is mandatory for CXL, and because PCIe Non-Flit Mode protocol is mandatory for PCIe, the "68B Flit Mode" parameter is always set to 1 for CXL or PCIe protocols. This additional handshake is shown in Figure 3-4. The combination of {FinCap.CXL} and {FinCap.Adapter} determine the Protocol and Flit Format. See Section 7.1.2.3 for the message format of the relevant sideband messages. See Section 3.4 for how Protocol and Flit Formats are determined.

Final determination for other parameters if CXL or PCIe protocol is negotiated:

- If "Raw Format" is advertised by both Link partners, "Raw Format" is set to 1 in the {FinCap.Adapter} message.
- If both Link partners advertised "Retry" and "Raw Format" is not negotiated, Adapter Retry is enabled and "Retry" is set to 1 in the {FinCap.Adapter} message.
- If both Link partners advertised "Enhanced Multi_Protocol_Enable", both Stack 0 and Stack 1 are enabled by the adapter, and all three parameters ("Enhanced Multi_Protocol_Enable", "Stack0_Enable" and "Stack1_Enable") are each set to 1 in the {FinCap.Adapter} message (if a {FinCap.Adapter} message is required to be sent).
- If both Link partners advertised "Multi_Protocol_Enable" and "Enhanced Multi_Protocol_Enable" is not negotiated, both Stack 0 and Stack 1 are enabled by the Adapter, and all three parameters ("Multi_Protocol_Enable", "Stack0_Enable", and "Stack1_Enable") are each set to 1 in the {FinCap.Adapter} message.
- If neither "Enhanced Multi_Protocol_Enable" nor "Multi_Protocol_Enable" is negotiated, then the lowest common denominator is used to determine whether Stack 0 or Stack 1 is enabled, and the corresponding bit is set to 1 in the {FinCap.Adapter} message. If both Stack enables are advertised, then Stack 0 is selected for operational mode and only Stack0_Enable is set to 1 in the {FinCap.Adapter} message.
- If CXL_LatOpt_Fmt5 is advertised by both Link partners, then it is set to 1 in the {FinCap.Adapter} message.

- If CXL_LatOpt_Fmt6 is advertised by both Link partners, then it is set to 1 in the {FinCap.Adapter} message.

Figure 3-4. Parameter Exchange for CXL or PCIe (i.e., "68B Flit Mode" or "CXL 256B Flit Mode" is 1 in {FinCap.Adapter})



If Streaming protocol is negotiated, there is no notion of DP and UP for parameter exchanges and each side independently advertises its capabilities. Additional Vendor Defined sideband messages are permitted to be exchanged to negotiate vendor-specific extensions. See Table 7-8 and Table 7-10 for additional descriptions of Vendor Defined sideband messages. Similarly, if Management Transport protocol is negotiated on a stack without "Streaming protocol," "CXL 256B Flit mode," or "PCIe Flit mode," there is no notion of DP and UP for parameter exchanges and each side independently advertises its capabilities.

The Finalized Configuration is implicitly determined based on the intersection of capabilities advertised by each side:

- Flit Formats are chosen based on the Truth Table resolution provided in Table 3-10
- If both Link partners advertised Retry, and "Raw Format" is not negotiated, then Adapter Retry is enabled
- If "Multi_Protocol_Enable" is negotiated, both Stack 0 and Stack 1 are enabled by the adapter
- If neither "Multi_Protocol_Enable" nor "Enhanced Multi_Protocol_Enable" is advertised by at least one of the Link partners, then the lowest common denominator is used to determine whether Stack 0 or Stack 1 is enabled (i.e., if both Stack enables are advertised, then Stack 0 is selected for operational mode)

{FinCap.*} messages are not sent for Streaming protocol. Adapter must determine vendor specific requirements in an implementation specific manner.

If "Enhanced Multi_Protocol_Enable" is negotiated, the {AdvCap.Adapter} and if applicable, the {FinCap.Adapter} messages determine the negotiated Flit Format of operation as well as the protocol for Stack 0. The Adapter uses {MultiProtAdvCap.Adapter} and if applicable, the {MultiProtFinCap.Adapter} sideband messages to negotiate the Stack 1 protocol. For Stack 1, if PCIe or CXL protocol support is going to be advertised, the UP Adapter must wait for the first message from the DP Adapter, review the capabilities advertised by DP and then send its own sideband message of advertised capabilities. UP is permitted to change its advertised capabilities based on DP capabilities. In the section below, "advertised" means that the corresponding bit is 1b in the {MultiProtAdvCap.Adapter} sideband message.

- "68B Flit Mode": This must be advertised if the Adapter and Protocol Layer support CXL 68B Flit Mode or PCIe Non-Flit Mode on Stack 1.
- "CXL 256B Flit Mode": This must be advertised if the Adapter and Protocol Layer support CXL 256B Flit Mode on Stack 1.
- "PCIe Flit Mode": This must be advertised if the Adapter and Protocol Layer support PCIe Flit Mode on Stack 1.
- "Streaming": This must be advertised if the Adapter and Protocol Layer support Streaming Flit Mode on Stack 1.
- "Management Transport Protocol": This must be advertised if the Protocol Layer supports Management Transport protocol on Stack 1.

If "68B Flit Mode" or "CXL 256B Flit Mode" is set in the {MultiProtFinCap.Adapter} message, there must be another handshake of Parameter Exchanges using the {AdvCap.CXL} and the {FinCap.CXL} messages to determine the details associated with this mode. The non-Stall {*.CXL} messages are sent with a MsgInfo encoding of 0001h indicating that these messages are for Stack 1 negotiation.

Figure 3-5 to Figure 3-9 represent examples of different scenarios where Stack 0 and Stack 1 are of different protocols.

Figure 3-5. Both Stacks are CXL or PCIe

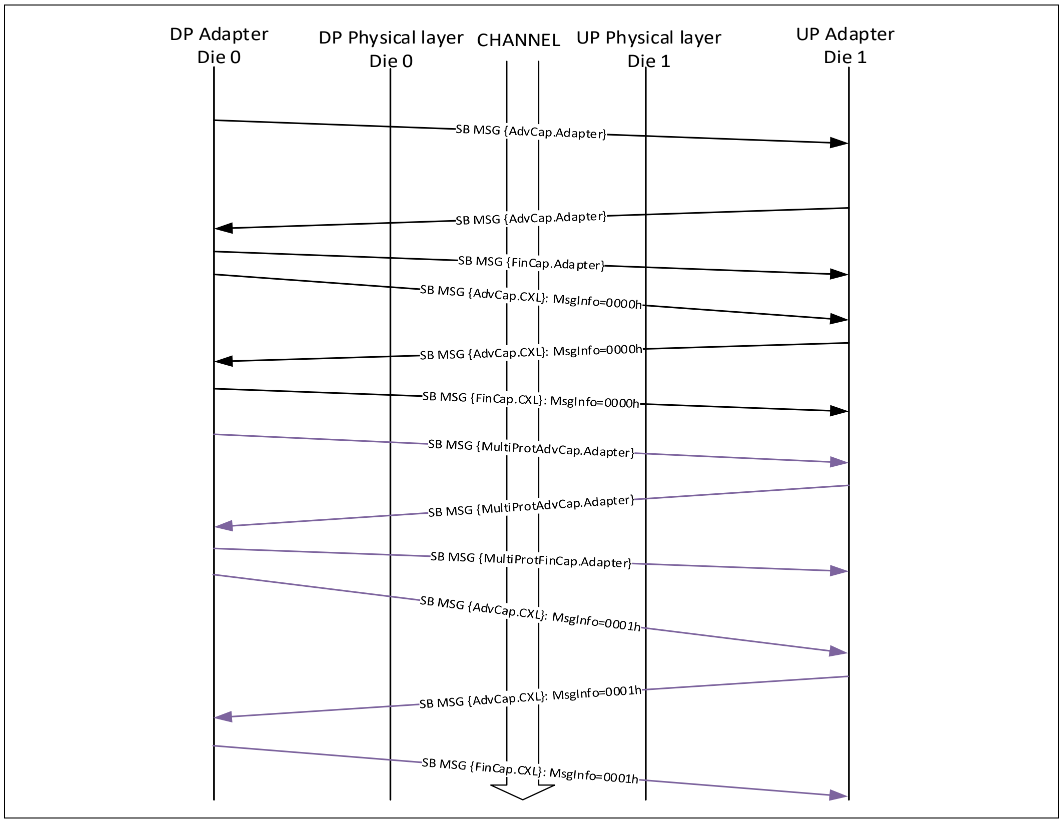

DP Adapter
Die 0
DP Physical layer
Die 0
CHANNEL
UP Physical layer
Die 1
UP Adapter
Die 1
SB MSG {AdvCap.Adapter}
SB MSG {AdvCap.Adapter}
SB MSG {FinCap.Adapter}
SB MSG {AdvCap.CXL}: MsgInfo=0000h
SB MSG {AdvCap.CXL}: MsgInfo=0000h
SB MSG {FinCap.CXL}: MsgInfo=0000h
SB MSG {MultiProtAdvCap.Adapter}
SB MSG {MultiProtAdvCap.Adapter}
SB MSG {MultiProtFinCap.Adapter}
SB MSG {AdvCap.CXL}: MsgInfo=0001h
SB MSG {AdvCap.CXL}: MsgInfo=0001h
SB MSG {FinCap.CXL}: MsgInfo=0001h

Figure 3-6. Stack 0 is PCIe, Stack 1 is Streaming

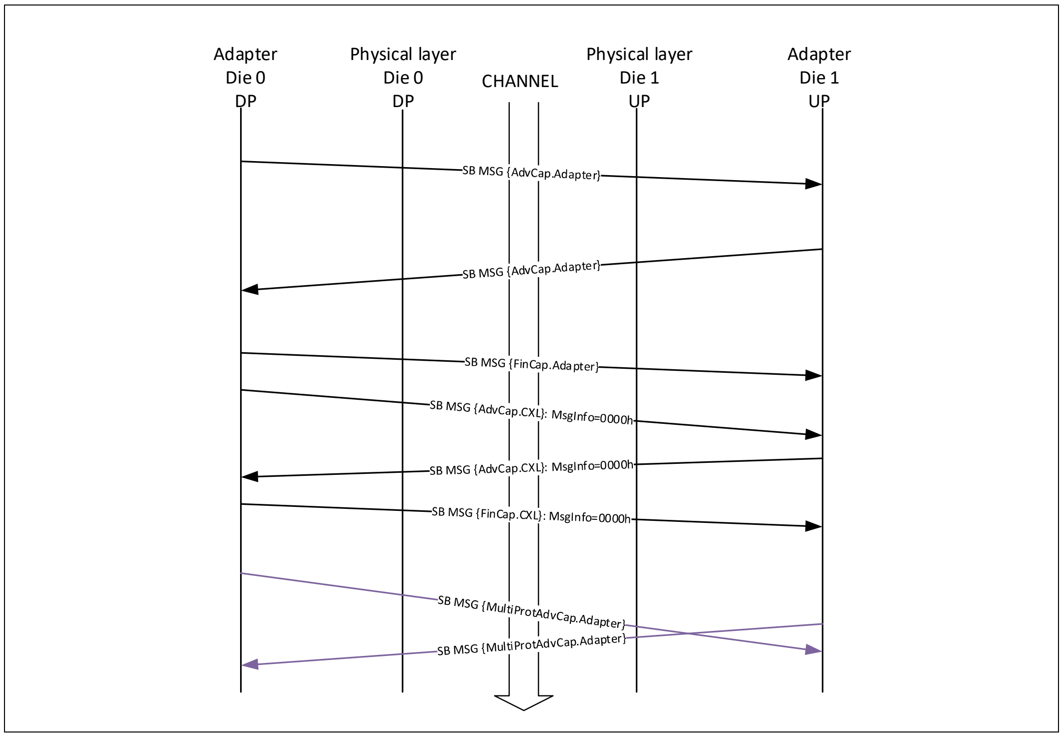

Adapter Die 0 DP
Physical layer Die 0 DP
CHANNEL
Physical layer Die 1 UP
Adapter Die 1 UP
SB MSG {AdvCap.Adapter}
SB MSG {AdvCap.Adapter}
SB MSG {FinCap.Adapter}
SB MSG {AdvCap.CXL}: MsgInfo=0000h
SB MSG {AdvCap.CXL}: MsgInfo=0000h
SB MSG {FinCap.CXL}: MsgInfo=0000h
SB MSG {MultiProtAdvCap.Adapter}
SB MSG {MultiProtAdvCap.Adapter}

Figure 3-7. Stack 0 is Streaming, Stack 1 is PCIe

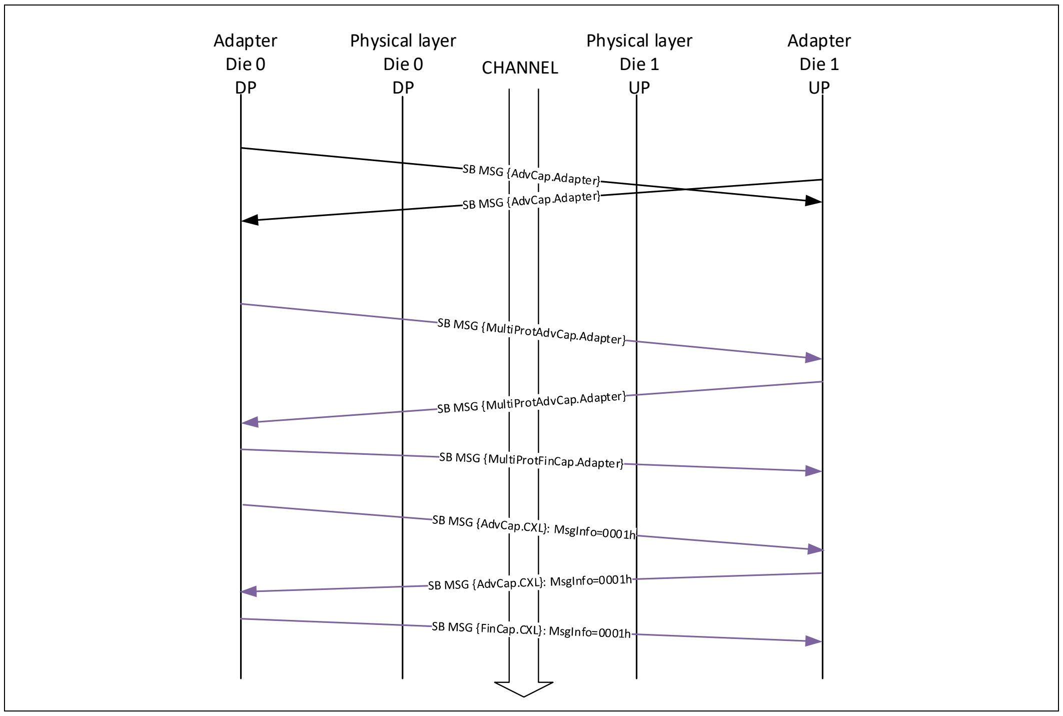

Adapter Die 0 DP
Physical layer Die 0 DP
CHANNEL
Physical layer Die 1 UP
Adapter Die 1 UP
SB MSG {AdvCap.Adapter}
SB MSG {AdvCap.Adapter}
SB MSG {MultiProtAdvCap.Adapter}
SB MSG {MultiProtAdvCap.Adapter}
SB MSG {MultiProtFinCap.Adapter}
SB MSG {AdvCap.CXL}: MsgInfo=0001h
SB MSG {AdvCap.CXL}: MsgInfo=0001h
SB MSG {FinCap.CXL}: MsgInfo=0001h

Figure 3-8. Both Stacks are Streaming

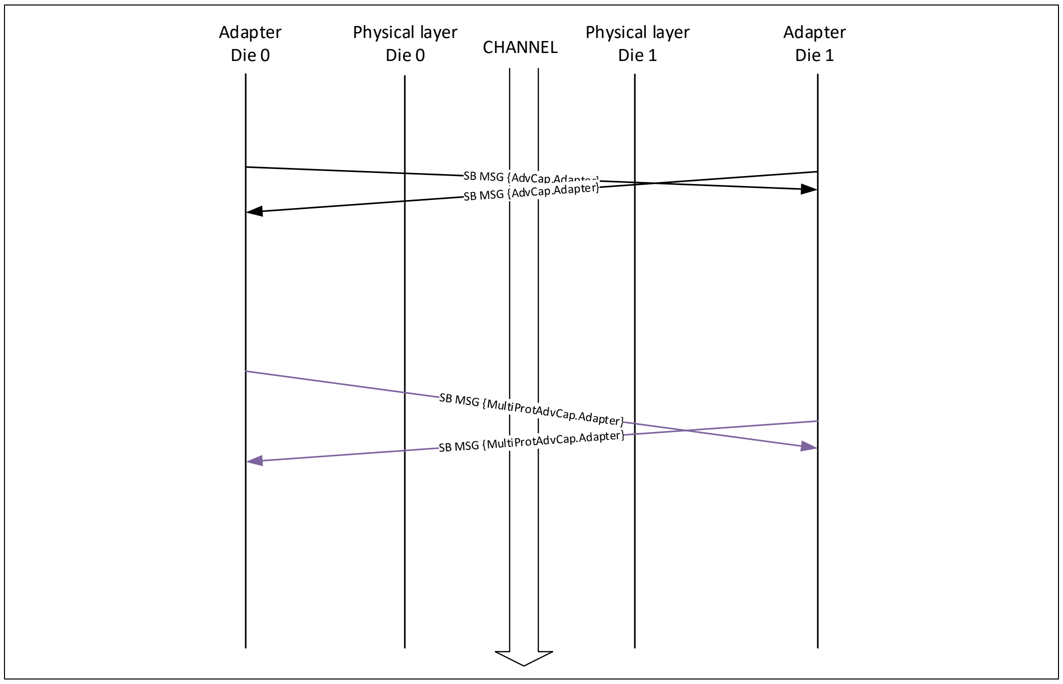
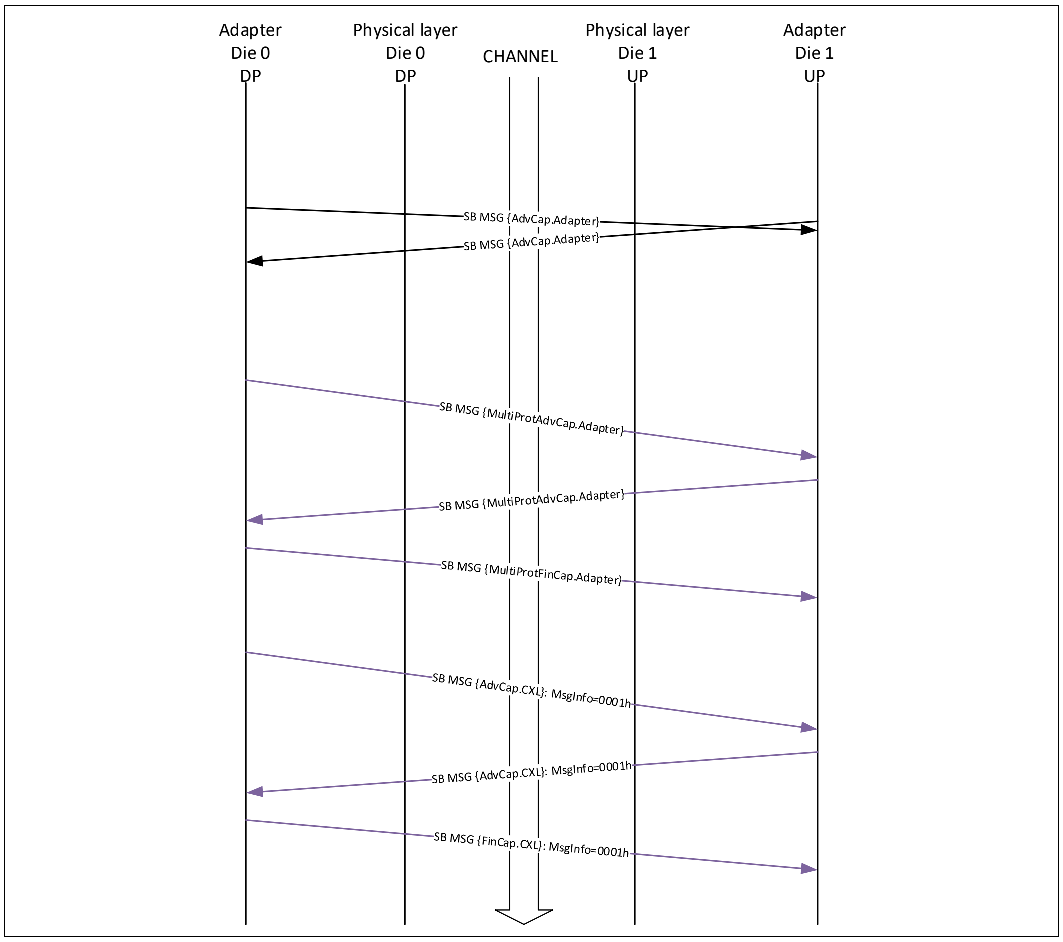


Figure 3-9. Stack 0 is Streaming, Stack 1 is CXL


Adapter Die 0 DP
Physical layer Die 0 DP
CHANNEL
Physical layer Die 1 UP
Adapter Die 1 UP
SB MSG {AdvCap.Adapter}
SB MSG {AdvCap.Adapter}
SB MSG {MultiProtAdvCap.Adapter}
SB MSG {MultiProtAdvCap.Adapter}
SB MSG {MultiProtFinCap.Adapter}
SB MSG {AdvCap.CXL}: MsgInfo=0001h
SB MSG {AdvCap.CXL}: MsgInfo=0001h
SB MSG {FinCap.CXL}: MsgInfo=0001h

The Adapter must implement a timeout of 8 ms (-0%/+50%) for successful Parameter Exchange completion. For the purposes of measuring a timeout for Parameter Exchange completion, all steps in Part 1 and Part 2 of Stage 3 of Link Initialization are included. The timer only increments while RDI is in Active state. The timer must reset if the Adapter receives an {AdvCap.*.Stall}, {FinCap.*.Stall}, {MultiProtAdvCap.*.Stall}, or {MultiProtFinCap.*.Stall} message from the remote Link partner. The 8-ms timeouts for Parameter Exchanges or Link State Machine transitions are treated as UIE and the Adapter must take the RDI to LinkError state. UCIe Retimers must ensure that they resolve the capability advertisement with remote Retimer partner (and merge with their own capabilities) before responding/initiating parameter exchanges with the UCIe die within its package. While resolution is in progress, they must send the corresponding stall message once every 4 ms to ensure that a timeout does not occur on the UCIe die within its package.

3.2.1.3 Part 3: FDI bring up

Once Parameter Exchanges have successfully completed, the Adapter reflects the result to the Protocol Layers on FDI, and moves on to carry out the FDI bring up flow as defined in Section 10.2.8. Once FDI is in Active state, it concludes Stage 3 of Link Initialization and protocol Flit transfer can begin. When multiple stacks are enabled on the same Adapter, each stack may finish the FDI bring up flow (see Section 10.2.8) at different times.

The data width on FDI is a function of the frequency of operation of the UCIe stack as well as the total bandwidth being transferred across the UCIe physical Link (which in turn depends on the number of Lanes and the speed at which the Lanes are operating). The data width on RDI is fixed to at least one byte per physical Lane per module that is controlled by the Adapter. The illustrations of the formats in this chapter are showing an example configuration of RDI mapped to a 64 Lane module of Advanced Package configuration on the Physical Layer of UCIe.

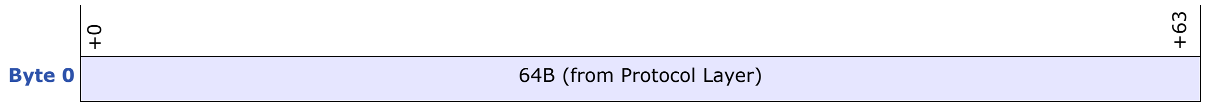
3.3 Operation Formats

In subsequent sections, when referring to CRC computation, a byte mapping of the Flit to CRC message (CRC message is the 128B input to CRC computation logic) is provided. See Section 3.7 for more details.

3.3.1 Raw Format for All Protocols

Raw Format can only be used for scenarios in which Retry support from the Adapter is not required. If Raw Format is negotiated for CXL or PCIe protocols, the Adapter transfers data from Protocol Layer to Physical Layer without any modification. Figure 3-10 shows an example of this for a 64B data path on FDI and RDI. This is identified as *Format 1* during parameter negotiation.

Figure 3-10. Format 1: Raw Format[a]



a. See Figure 2-1 for color mapping.

3.3.2 68B Flit Format

This Flit Format is identified as *Format 2* on UCIe. Support for this is mandatory when CXL 68B Flit Mode protocol or PCIe Non-Flit Mode protocol is supported. 68B Flit Format support is optional for Streaming protocols.

The Protocol Layer sends 64B of protocol information. The Adapter adds a two byte prefix of Flit Header and a two byte suffix of CRC. Table 3-3 gives the Flit Header format for *Format 2* when Retry from the Adapter is required. If Retry from the Adapter is not required, then the Flit Header format is as provided in Table 3-2.

Even if Retry is not required, the Adapter still computes and drives CRC bytes — the Receiver is strongly recommended to treat a CRC error as an Uncorrectable Internal Error in this situation. For CRC computation, Flit Byte 0 (i.e., Flit Header Byte 0) is assigned to CRC message Byte 0, Flit Byte 1 (i.e., Flit Header Byte 1) is assigned to CRC message Byte 1 and so on until Flit Byte 65 is assigned to CRC message Byte 65.

Retry is performed over this 68B Flit.

Table 3-2. Flit Header for Format 2 without Retry

Byte	Bit	Description	
		PCIe or CXL	Streaming Protocol
Byte 0	[7:6]	Protocol Identifier: 2'b00 : D2D Adapter NOP Flit or PDS Flit Header 2'b01 : CXL.io Flit 2'b10 : CXL.cachemem Flit 2'b11 : ARB/MUX Flit (Reserved encoding for PCIe)	Protocol Identifier: 2'b00 : D2D Adapter NOP Flit 2'b01 : Protocol Layer Flit Remaining encodings are Reserved.
	[5]	Stack Identifier: 1'b0 : Stack 0 1'b1 : Stack 1	
	[4]	1'b0 : Regular Flit Header 1'b1 : Pause of Data Stream (PDS) Flit Header	
	[3:0]	Reserved	
Byte 1[a]	[7]	1'b0 : Regular Flit Header 1'b1 : Pause of Data Stream (PDS) Flit Header	
	[6:0]	Reserved	

a. For a Test Flit, bits [7:6] of Byte 1 are 01b. See Section 11.2 for more details.

Table 3-3. Flit Header for Format 2 with Retry

Byte	Bit	Description	
		PCIe or CXL	Streaming Protocol
Byte 0	[7:6]	Protocol Identifier: 2'b00 : D2D Adapter NOP Flit or PDS Flit Header 2'b01 : CXL.io Flit 2'b10 : CXL.cachemem Flit 2'b11 : ARB/MUX Flit (Reserved encoding for PCIe)	Protocol Identifier: 2'b00 : D2D Adapter NOP Flit 2'b01 : Protocol Layer Flit Remaining encodings are Reserved.
	[5]	Stack Identifier: 1'b0 : Stack 0 1'b1 : Stack 1	
	[4]	1'b0 : Regular Flit Header 1'b1 : Pause of Data Stream (PDS) Flit Header	
	[3:0]	The upper four bits of Sequence number "S" (i.e., S[7:4])	
Byte 1[a]	[7:6]	2'b00 : Regular Flit Header 2'b11 : Pause of Data Stream (PDS) Flit Header Other encodings are reserved	
	[5:4]	Ack or Nak information 2'b00 : Explicit Sequence number "S" of Flit if not PDS, otherwise the bitwise inverted value of "NEXT_TX_FLIT_SEQ_NUM - 1". (See *PCIe Base Specification* for the definition of NEXT_TX_FLIT_SEQ_NUM and the subtraction operation for sequence numbers) 2'b01 : Ack. The Sequence number "S" carries the Ack'ed sequence number. 2'b10 : Nak. The Sequence number "S" carries 255 if N=1, otherwise it carries N-1, where N is the Nak'ed sequence number. 2'b11 : Reserved	
	[3:0]	The lower four bits of Sequence number "S" (i.e., S[3:0]) Sequence number 0 is reserved and if present, it implies no Ack or Nak is sent.	

a. For a Test Flit, bits [7:6] of Byte 1 are 01b. See Section 11.2 for more details.

3.3.2.1 68B Flit Format Alignment and Padding Rules

Because of the four bytes added by D2D Adapter, the alignment of the Flit does not always match the number of Lanes of the physical Link. The bytes added by D2D Adapter require the Adapter to shift the data arriving over FDI by four bytes for consecutive Flits transmitted over RDI. Data is always transferred in multiples of 256B (note that Retimer credits have a 256B data granularity). A mechanism to Pause the Data Stream is provided as a way to save power when the Link is idle. Before pausing the data stream, the data stream is terminated with a Pause of Data Stream (PDS) Flit Header followed by 0b padding to the next 64B count multiple boundary and at least two subsequent 64B chunks of all 0 value data. If the transfer is not at a 256B count multiple boundary, additional 64B chunks of all 0 value data are required to bring the transferred bytes to a 256B count multiple. The subsequent transfers of all 0 data mentioned above give the Receiver at least two 64B chunks to reset the receiving byte shifter. The PDS Flit Header and the 0 padding bytes following it must not be forwarded to the Protocol Layer. The PDS token is a variable-size Flit that carries a 2B special Flit Header (referred to as the PDS Flit Header), and 0 bytes padded as described above. The Transmitter of PDS drives the following on the Flit header:

1. Bit [4] of Byte 0 as 1
2. Bit [7] of Byte 1 as 1
3. Bit [6] of Byte 1 as 1
4. Bits [5:4] of Byte 1 as 00b and the sequence number[7:0] is matching the expected value for a PDS Flit Header in this position as defined in Table 3-3.

The Adapter may optionally insert continuous NOPs instead of terminating the data stream with a PDS when no other flits are available to transmit. There is a trade-off between the longer idle latency for a new flit to be transmitted after a PDS vs. the power consumption of continuously transmitting NOP flits. It is the responsibility of the transmitting Adapter to make the determination between transmitting NOP flits vs. inserting a PDS in an implementation-specific manner.

If Retry is enabled, the Receiver must interpret this Flit header as PDS if any two of the above four conditions are true. If Retry is disabled, the Receiver must interpret this Flit header as a PDS if conditions (1) and (2) are true.

A PDS must be inserted when Retry is triggered or RDI state goes through Retrain. The transmitter must insert PDS Flit Header and corresponding padding of 0s as it would for an actual PDS and start the replayed Flit from fresh alignment (i.e., flit begins from a 256B-aligned boundary). Note that for Retry, this should occur before the Transmitter begins replaying the Flits from the Retry buffer; and for Retrain entry, this should occur before asserting **`lp_stallack`** to the Physical Layer.

For Retry and Retrain scenarios, the Receiver must also look for the expected sequence number in Byte 0 and Byte 1 of the received data bus with a corresponding valid Flit (i.e., CRC passes). Note that for a Retrain scenario, a PDS might not be received at the receiver before the RDI state changes to Retrain, and the Adapter must discard any partially received 68B Flits after state change.

When resuming the data stream after a PDS token (i.e., a PDS Flit Header and the corresponding padding of 0s), the first Flit is always 256B aligned; any valid Flit transfer after a PDS token will resume the data stream. After a PDS Flit Header has been transmitted, the corresponding padding of 0b to satisfy the PDS token padding requirements must be finished before resuming the data stream with new Flits.

Property of Universal Chiplet Interconnect Express (UCIe) © 2022–2024

IMPLEMENTATION NOTE — BIT ERRORS AND ALIASING

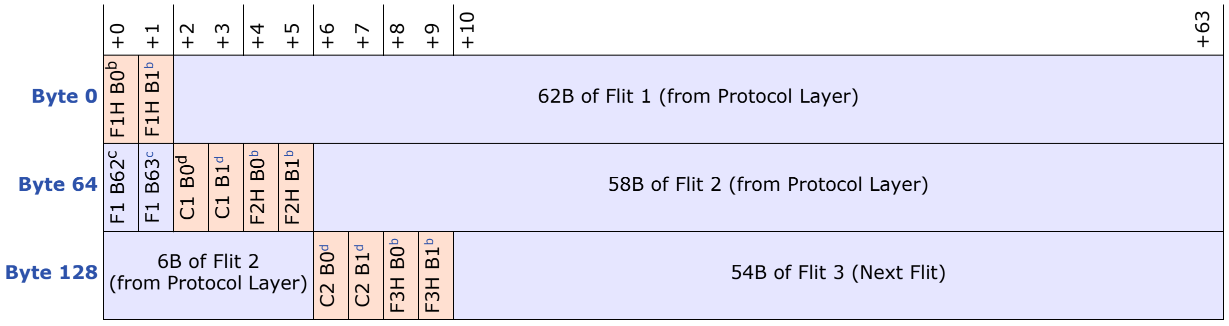
When Retry is disabled, the BER of the Link is 1E-27 or lower. In these cases, any bit error is an uncorrectable error for the Link. As a best practice, it is strongly recommended for receiver implementations to have an uncorrectable internal error condition for scenarios in which neither a valid Flit Header nor a valid PDS Flit Header is detected.

When Retry is enabled, the BER is 1E-15 or lower, which results in the probability of two or more bit errors within the Flit Header is very low. However, implementations must consider the following two scenarios:

- **PDS Flit Header aliasing to a regular Flit Header**: Checking for two out of the four conditions guarantees that at least three bit errors must occur within the two bytes of the PDS Flit Header for it to alias to a regular Flit Header. Even for three bit errors, there will be a CRC which will result in a retry and will be handled seamlessly through the retry rules.
- **Regular Flit Header aliasing to a PDS Flit Header**: It is possible for two bit errors to cause a Regular Flit Header to alias to a PDS Flit Header. This will likely result in a CRC error for future Flits. However, to reduce the probability of a data corruption that escapes CRC even further, it is strongly recommended that if a PDS Flit Header was detected without all four conditions being satisfied (i.e., two out of four or three out of four were satisfied), the receiver checks for an explicit sequence number Flit with the expected sequence number in Byte 0 and Byte 1 of the first received data transfer and that it is a valid Flit (i.e., CRC passes) after the PDS (including the PDS token and the corresponding padding) have completed; and triggers a Retry if it does not pass the check. Note that this is the same check a Receiver performs after a Retry or Retrain.

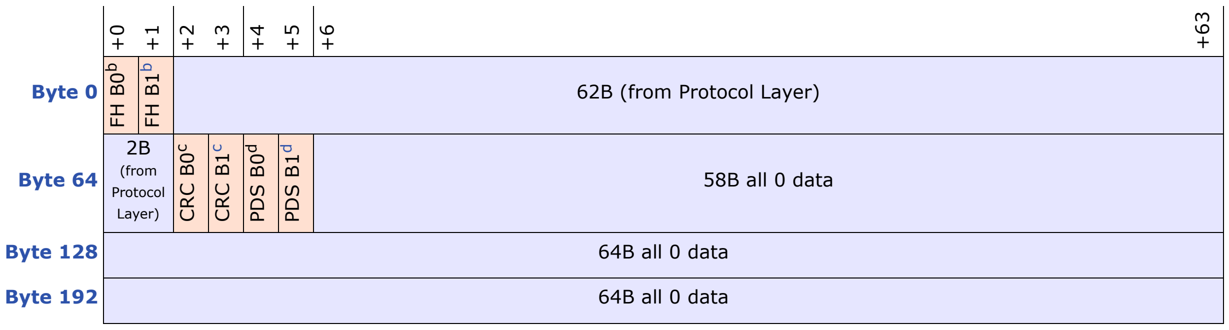
Figure 3-11 shows the 68B Flit Format. Figure 3-12 and Figure 3-13 provide examples of PDS insertion.

Figure 3-11. Format 2: 68B Flit Format[a]



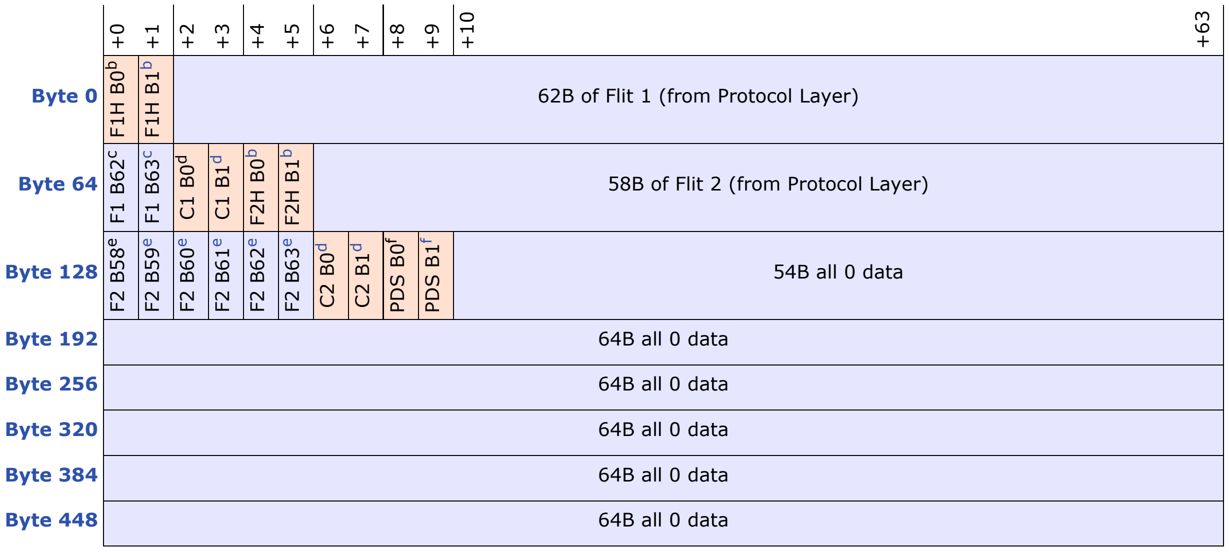
a. See Figure 2-1 for color mapping.
b. Flit 1 Header Byte 0, Flit 1 Header Byte 1, Flit 2 Header Byte 0, Flit 2 Header Byte 1, Flit 3 Header Byte 0, and Flit 3 Header Byte 1 respectively.
c. Flit 1 Byte 62 and Byte 63, respectively (from Protocol Layer).
d. Flit 1 CRC Byte 0, Flit 1 CRC Byte 1, Flit 2 CRC Byte 0, and Flit 2 CRC Byte 1, respectively.

Figure 3-12. Format 2: 68B Flit Format PDS Example 1[a]



a. See Figure 2-1 for color mapping.
b. Flit Header Byte 0 and Byte 1, respectively.
c. CRC Byte 0 and Byte 1, respectively.
d. PDS Flit Header Byte 0 and Byte 1, respectively.

Figure 3-13. Format 2: 68B Flit Format PDS Example 2 — Extra 0s Padded to Make the Data Transfer a Multiple of 256B[a]



a. See Figure 2-1 for color mapping.
b. Flit 1 Header Byte 0, Flit 1 Header Byte 1, Flit 2 Header Byte 0, and Flit 2 Header Byte 1, respectively.
c. Flit 1 Byte 62 and Byte 63, respectively (from Protocol Layer).
d. Flit 1 CRC Byte 0, Flit 1 CRC Byte 1, Flit 2 CRC Byte 0, and Flit 2 CRC Byte 1, respectively.
e. Flit 2 Bytes 58 through Byte 63, respectively (from Protocol Layer).
f. PDS Flit Header Byte 0 and Byte, respectively.

3.3.3 Standard 256B Flit Formats

These are the Standard Flit Formats defined in *PCIe Base Specification* for PCIe Flit Mode and *CXL Specification* for CXL 256B Flit Mode. These are identified as "Standard 256B End Header Flit Format" (or *Format 3*) and "Standard 256B Start Header Flit Format" (or *Format 4*), respectively. Support for this is mandatory when PCIe Flit Mode or CXL 256B Flit Mode protocols are negotiated. Standard 256B Flit Formats (Start Header or End Header) support is optional with Streaming protocols.

The Protocol Layer sends data in 256B Flits, but it drives 0 on the bytes reserved for the Adapter (shown in light orange in Figure 3-14 through Figure 3-19). The 6B of DLP defined in *PCIe Base Specification* exist in *Format 3* and *Format 4* as well for PCIe and CXL.io protocols. However, since

DLLPs are required to bypass the Tx Retry buffer in PCIe and CXL.io protocols, the DLP bytes end up being unique since they are partially filled by the Protocol Layer and partially by the Adapter. DLP0 and DLP1 are replaced with the Flit Header for UCIe and are driven by UCIe Adapter. However, if the Flit carries a Flit Marker, the Protocol Layer must populate bit 4 of Flit Header Byte 0 to 1b, as well as the relevant information in the Flit_Marker bits (these are driven as defined in *PCIe Base Specification*). Protocol Layer must also populate the Protocol Identifier bits in the Flit Header for the Flits it generates.

For Streaming protocols, Figure 3-17 shows the applicable Flit Format. Protocol Layer only populates bits [7:6] of Byte 0 of the Flit Header, and it must never set 00b for bits [7:6].

Standard 256B Start Header Flit Format is optional for PCIe Flit Mode protocol. Figure 3-18 shows the Flit Format example.

FDI provides a separate interface for DLLP transfer from the Protocol Layer to the Adapter and vice-versa. The Adapter is responsible for inserting DLLP into DLP Bytes 2:5 if a Flit Marker is not present. The credit update information is transferred as regular Update_FC DLLPs over FDI from the Protocol Layer to the Adapter. The Adapter is also responsible for formatting these updates as Optimized_Update_FC format when possible and driving them on the relevant DLP bytes. The Adapter is also responsible for adhering to all the DLLP rules defined for Flit Mode in *PCIe Base Specification*. On the receive path, the Adapter is responsible for extracting the DLLPs or Optimized_Update_FC from the Flit and driving it on the dedicated DLLP interface provided on FDI.

Two sets of CRC are computed (CRC0 and CRC1). The same 2B over 128B CRC computation as previous formats is used.

For PCIe, CXL, and Streaming:

- For *Format 3*, CRC0 is computed using Flit Bytes 0 to 127 assigned to the corresponding bytes of the CRC message input. CRC1 is computed using Flit Bytes 128 to 241 as the message input with Flit Byte 128 assigned to CRC message Byte 0, Flit Byte 129 assigned to CRC message Byte 1 and so on until Flit Byte 241 is assigned to CRC message Byte 113 (including the Flit Header bits inserted by the Adapter, which for PCIe and CXL.io, includes the DLP bytes inserted by the Adapter).
- For *Format 4*, CRC0 is computed using Flit Bytes 0 to 127 assigned to the corresponding bytes of the CRC message input (including the Flit Header bits inserted by the Adapter). CRC1 is computed using Flit Bytes 128 to 241 as the message input with Flit Byte 128 assigned to CRC message Byte 0, Flit Byte 129 assigned to CRC message Byte 1 and so on until Flit Byte 241 is assigned to CRC message Byte 113 (for PCIe and CXL.io, this includes the DLP bytes inserted by the Adapter).

If Retry is not required, the Adapter still computes and drives CRC bytes — the Receiver is strongly recommended to treat a CRC error as an Uncorrectable Internal Error (UIE) in this situation.

The Flit Header byte formats are shown in Table 3-5 when Retry is required; otherwise, it is as shown in Table 3-4.

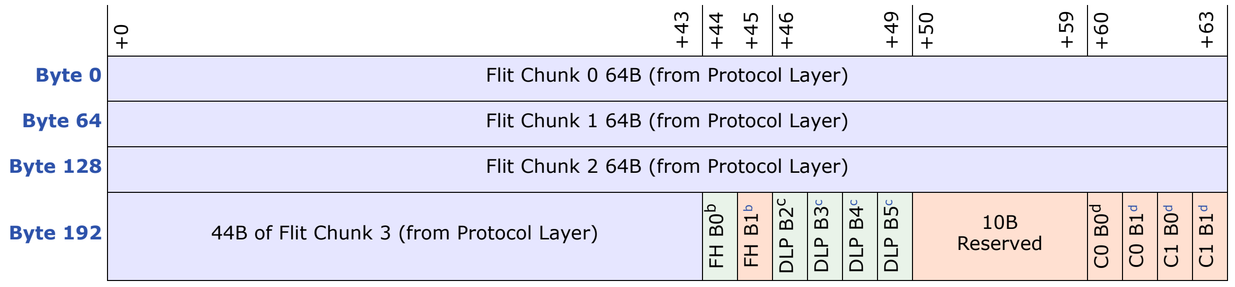
The Protocol Layer must drive bits [7:6] in Byte 1 of Flit Header to 00b for CXL/PCIe/Streaming protocol Flits and to 10b for Management Flits (when successfully negotiated).

For Management Flits, Bytes 238 to 241 are driven from the Protocol Layer with Management Transport Credit Return DWORD (CRD) Bytes 0 to 3 (see Section 8.2.5.2.2 for CRD format). Bytes 232 to 235 in *Format 3* and Bytes 234 to 237 in *Format 4* are driven from the Protocol Layer with 0s for Management Flits. See Figure 3-16 and Figure 3-19 for details of *Format 3* and *Format 4* for Management Flits, respectively.

If PCIe/CXL.io is negotiated along with Management Transport protocol on the same stack:

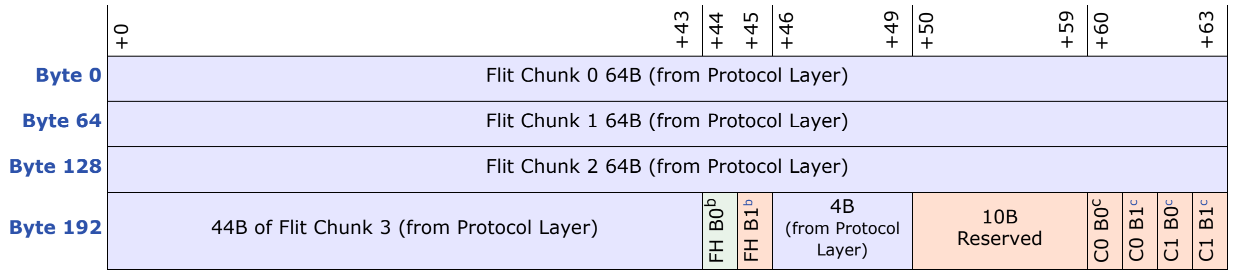
- If bits [7:6] of Byte 1 are 10b, the Adapter passes through Bytes 238 to 241 from the Protocol Layer to the Link
- If bits [7:6] of Byte 1 are 00b, Bytes 238 to 241 are treated per PCIe/CXL.io DLP rules for this flit format

Figure 3-14. Format 3: Standard 256B End Header Flit Format for PCIe[a]


+0
+43
+44
+45
+46
+49
+50
+59
+60
+63
Byte 0
Flit Chunk 0 64B (from Protocol Layer)
Byte 64
Flit Chunk 1 64B (from Protocol Layer)
Byte 128
Flit Chunk 2 64B (from Protocol Layer)
Byte 192
44B of Flit Chunk 3 (from Protocol Layer)
FH B0[b]
FH B1[b]
DLP B2[c]
DLP B3[c]
DLP B4[c]
DLP B5[c]
10B Reserved
C0 B0[d]
C0 B1[d]
C1 B0[d]
C1 B1[d]

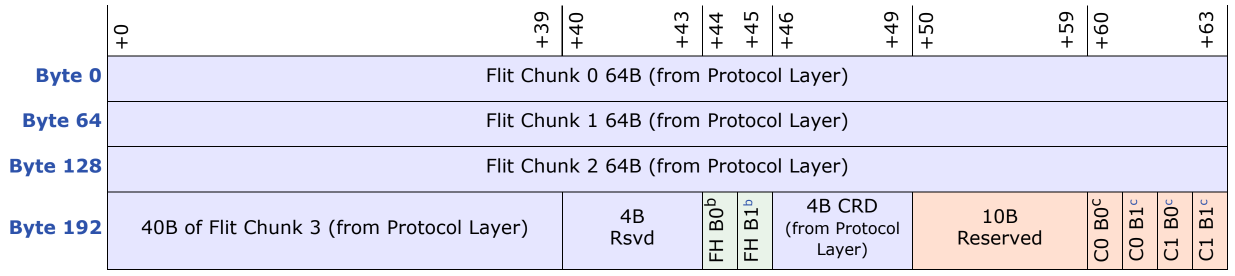
a. See Figure 2-1 for color mapping.
b. Flit Header Byte 0 and Byte 1, respectively.
c. DLP Byte 2, Byte 3, Byte 4, and Byte 5, respectively.
d. CRC0 Byte 0, CRC0 Byte 1, CRC1 Byte 0, and CRC1 Byte 1, respectively.

Figure 3-15. Format 3: Standard 256B End Header Flit Format for Streaming Protocol[a]


+0
+43
+44
+45
+46
+49
+50
+59
+60
+63
Byte 0
Flit Chunk 0 64B (from Protocol Layer)
Byte 64
Flit Chunk 1 64B (from Protocol Layer)
Byte 128
Flit Chunk 2 64B (from Protocol Layer)
Byte 192
44B of Flit Chunk 3 (from Protocol Layer)
FH B0[b]
FH B1[b]
4B (from Protocol Layer)
10B Reserved
C0 B0[c]
C0 B1[c]
C1 B0[c]
C1 B1[c]

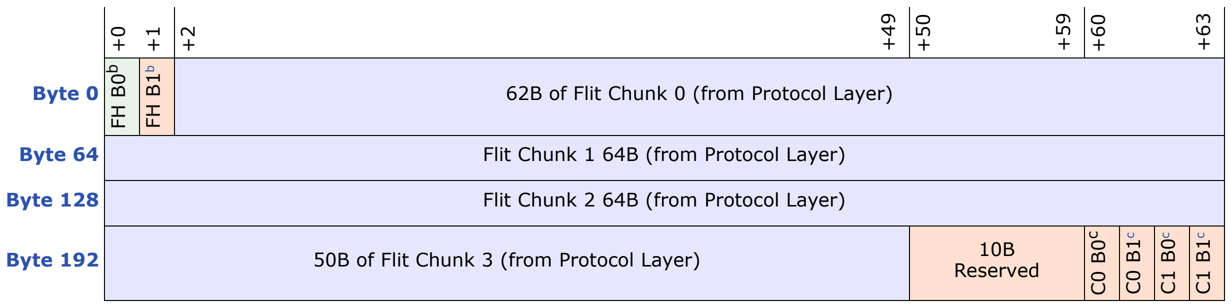
a. See Figure 2-1 for color mapping.
b. Flit Header Byte 0 and Byte 1, respectively.
c. CRC0 Byte 0, CRC0 Byte 1, CRC1 Byte 0, and CRC1 Byte 1, respectively.

Figure 3-16. Format 3: Standard 256B End Header Flit Format for Management Transport Protocol[a]


+0
+39
+40
+43
+44
+45
+46
+49
+50
+59
+60
+63
Byte 0
Flit Chunk 0 64B (from Protocol Layer)
Byte 64
Flit Chunk 1 64B (from Protocol Layer)
Byte 128
Flit Chunk 2 64B (from Protocol Layer)
Byte 192
40B of Flit Chunk 3 (from Protocol Layer)
4B Rsvd
FH B0[b]
FH B1[b]
4B CRD (from Protocol Layer)
10B Reserved
C0 B0[c]
C0 B1[c]
C1 B0[c]
C1 B1[c]

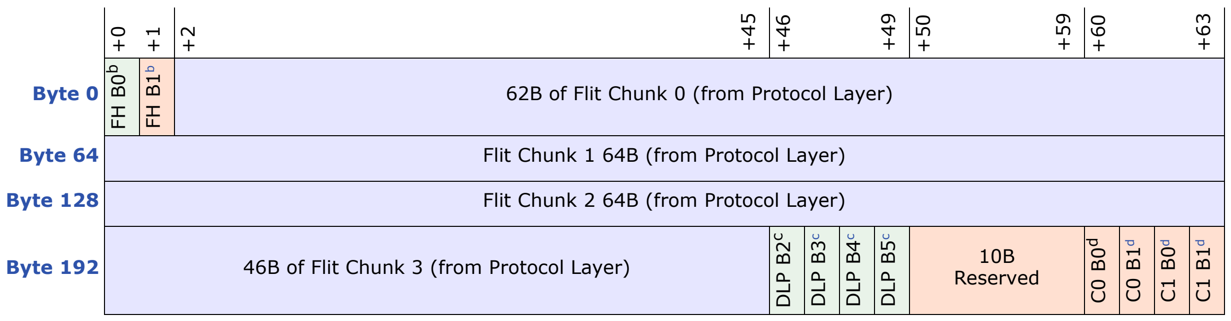
a. See Figure 2-1 for color mapping.
b. Flit Header Byte 0 and Byte 1, respectively.
c. CRC0 Byte 0, CRC0 Byte 1, CRC1 Byte 0, and CRC1 Byte 1, respectively.

Figure 3-17. Format 4: Standard 256B Start Header Flit Format for CXL.cachemem or Streaming Protocol[a]


+0
+1
+2
+49
+50
+59
+60
+63
Byte 0
FH B0[b]
FH B1[b]
62B of Flit Chunk 0 (from Protocol Layer)
Byte 64
Flit Chunk 1 64B (from Protocol Layer)
Byte 128
Flit Chunk 2 64B (from Protocol Layer)
Byte 192
50B of Flit Chunk 3 (from Protocol Layer)
10B Reserved
C0 B0[c]
C0 B1[c]
C1 B0[c]
C1 B1[c]

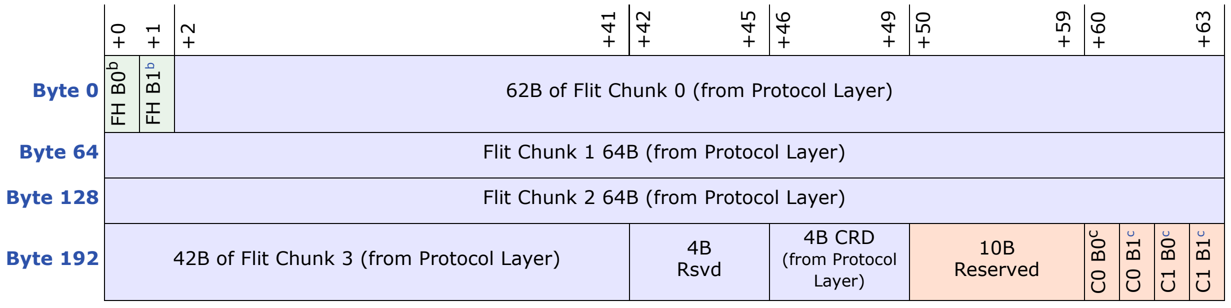
a. See Figure 2-1 for color mapping.
b. Flit Header Byte 0 and Byte 1, respectively.
c. CRC0 Byte 0, CRC0 Byte 1, CRC1 Byte 0, and CRC1 Byte 1, respectively.

Figure 3-18. Format 4: Standard 256B Start Header Flit Format for CXL.io or PCIe[a]


+0
+1
+2
+45
+46
+49
+50
+59
+60
+63
Byte 0
FH B0[b]
FH B1[b]
62B of Flit Chunk 0 (from Protocol Layer)
Byte 64
Flit Chunk 1 64B (from Protocol Layer)
Byte 128
Flit Chunk 2 64B (from Protocol Layer)
Byte 192
46B of Flit Chunk 3 (from Protocol Layer)
DLP B2[c]
DLP B3[c]
DLP B4[c]
DLP B5[c]
10B Reserved
C0 B0[d]
C0 B1[d]
C1 B0[d]
C1 B1[d]

a. See Figure 2-1 for color mapping.
b. Flit Header Byte 0 and Byte 1, respectively.
c. DLP Byte 2, Byte 3, Byte 4, and Byte 5, respectively.
d. CRC0 Byte 0, CRC0 Byte 1, CRC1 Byte 0, and CRC1 Byte 1, respectively.

Figure 3-19. Format 4: Standard 256B Start Header Flit Format for Management Transport Protocol[a]


+0
+1
+2
+41
+42
+45
+46
+49
+50
+59
+60
+63
Byte 0
FH B0[b]
FH B1[b]
62B of Flit Chunk 0 (from Protocol Layer)
Byte 64
Flit Chunk 1 64B (from Protocol Layer)
Byte 128
Flit Chunk 2 64B (from Protocol Layer)
Byte 192
42B of Flit Chunk 3 (from Protocol Layer)
4B Rsvd
4B CRD (from Protocol Layer)
10B Reserved
C0 B0[c]
C0 B1[c]
C1 B0[c]
C1 B1[c]

a. See Figure 2-1 for color mapping.
b. Flit Header Byte 0 and Byte 1, respectively.
c. CRC0 Byte 0, CRC0 Byte 1, CRC1 Byte 0, and CRC1 Byte 1, respectively.

Table 3-4. Flit Header for Format 3, Format 4, Format 5, and Format 6 without Retry

Byte	Bit	Description			
		CXL 256B Flit Mode	PCIe Flit Mode	Streaming Protocol	Management Transport Protocol
0	[7:6]	Protocol Identifier: 00b: D2D Adapter/CXL.io NOP Flit 01b: CXL.io Flit 10b: CXL.cachemem Flit 11b: ARB/MUX Flit	Protocol Identifier: 00b: D2D Adapter/PCIe NOP Flit 01b: PCIe Flit All other encodings are reserved.	Protocol Identifier: 00b: D2D Adapter NOP Flit Remaining encodings are permitted to be used by Protocol Layer in a vendor defined manner. Protocol Layer must never set this to 00b for Flits sent across FDI.	Protocol Identifier: 00b: D2D Adapter NOP Flit 01b: Management Flit All other encodings are reserved.
	[5]	Stack Identifier: 0: Stack 0 1: Stack 1			
	[4]	Reserved for CXL.cachemem For CXL.io or PCIe Flit Mode: 0: DLLP Payload in DLP 2..5 1: Optimized_Update_FC or Flit_Marker in DLP2..5		Reserved	
	[3:0]	Reserved			
1	[7:6]	Flit Type: 00b: CXL/PCIe/Streaming Flit/D2D Adapter NOP Flit 01b: Test Flit (see Section 11.2 for details) 10b: Management Flit 11b: Reserved			
	[5:0]	Reserved			

Table 3-5. Flit Header for Format 3, Format 4, Format 5, and Format 6 with Retry

Byte	Bit	Description			
		CXL 256B Flit Mode	**PCIe Flit Mode**	**Streaming Protocol**	**Management Transport Protocol**
0	[7:6]	Protocol Identifier: 00b: D2D Adapter/CXL.io NOP Flit 01b: CXL.io Flit 10b: CXL.cachemem Flit 11b: ARB/MUX Flit	Protocol Identifier: 00b: D2D Adapter/PCIe NOP Flit 01b: PCIe Flit All other encodings are reserved	Protocol Identifier: 00b: D2D Adapter NOP Flit Remaining encodings are permitted to be used by Protocol Layer in a vendor defined manner. Protocol Layer must never set this to 00b for Flits sent across FDI.	Protocol Identifier: 00b: D2D Adapter NOP Flit 01b: Management Flit All other encodings are reserved.
	[5]	Stack Identifier: 0: Stack 0 1: Stack 1			
	[4]	Reserved for CXL.cachemem For CXL.io or PCIe Flit Mode: 0: DLLP Payload in DLP 2..5 1: Optimized_Update_FC or Flit_Marker in DLP2..5		Reserved	
	[3:0]	The upper four bits of Sequence number “S” (i.e., S[7:4])			
1	[7:6]	Flit Type: 00b: CXL/PCIe/Streaming Flit/D2D Adapter NOP Flit 01b: Test Flit (see Section 11.2 for details) 10b: Management Flit 11b: Reserved			
	[5:4]	Ack or Nak Information: 00b: Explicit Sequence number “S” of the current Flit is present. 01b: Ack. The sequence number “S” carries the Ack’ed sequence number. 10b: Nak. The sequence number “S” carries 255 if N=1; otherwise, it carries N-1; where N is the Nak’ed sequence number. 11b: Reserved			
	[3:0]	The lower four bits of Sequence number “S” (i.e., S[3:0]). Sequence number 0 is reserved and if present, it implies no Ack or Nak is sent.			

3.3.4 Latency-Optimized 256B Flit Formats

Two Latency-Optimized 256B Flit Formats are defined: *Format 5* and *Format 6*. It is strongly recommended that UCIe implementations support *Format 6* for CXL 256B Flit Mode protocol to get the best latency benefits.

Both formats look the same from the Adapter perspective, the only difference is whether the Protocol Layer is filling in the optional bytes of protocol information . The Latency-Optimized 256B without Optional bytes Flit Format (or *Format 5*) is when the Protocol Layer is not filling in the optional bytes, whereas the Latency-Optimized 256B with Optional bytes Flit Format (or *Format 6*) is when the Protocol Layer is filling in the optional bytes.

Latency-Optimized 256B Flit Formats (with Optional bytes or without Optional bytes) support is optional with Streaming protocols. Protocol Layer only populates bits [7:6] of the Flit Header, and it must never set 00b for bits [7:6].

Latency-Optimized Flit with Optional Bytes Flit Format is optional for PCIe Flit Mode protocol. Figure 3-23 shows the Flit Format example.

Two sets of CRC are computed. CRC0 is computed using Flit Bytes 0 to 125 assigned to the corresponding bytes of the CRC message input (including the Flit Header bits and if applicable, the DLP bits inserted by the Adapter). CRC1 is computed using Flit Bytes 128 to 253 as the message input with Flit Byte 128 assigned to CRC message Byte 0, Flit Byte 129 assigned to CRC message Byte 1 and so on until Flit Byte 253 assigned to CRC message Byte 125. If Retry is not required, the Adapter still computes and drives CRC bytes — the Receiver is strongly recommended to treat a CRC error as UIE in this situation.

For Management Flits (when successfully negotiated), the Protocol Layer must drive bits [7:6] in Byte 1 of Flit Header to 00b for Protocol Flit and to 10b.

For Management Flits using *Format 5*, Bytes 240 to 243 are driven from the Protocol Layer with Management Transport Credit Return DWORD (CRD) Bytes 0 to 3 (see Section 8.2.5.2.2 for CRD format). See Figure 3-22 for details.

If CXL.io is negotiated along with Management Transport protocol on the same stack for *Format 5*:

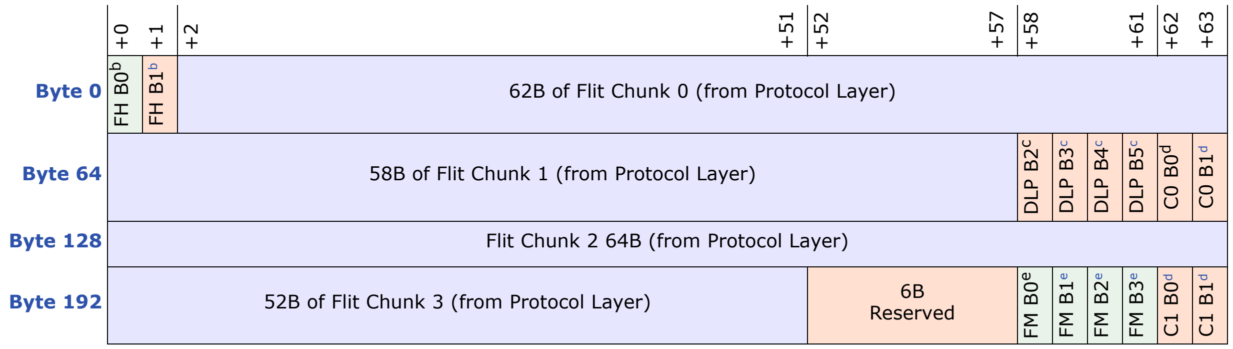
- If bits [7:6] of Byte 1 are 10b, the Adapter drives 0 on Bytes 122 to 125 and 244 to 253
- If bits [7:6] of Byte 1 are 00b, then Bytes 122 to 125 are treated per the CXL.io DLP rules of this flit format and Bytes 250 to 253 are treated per the CXL.io FM rules of this flit format

For Management Flits using *Format 6*, Bytes 250 to 253 are driven from the Protocol Layer with Management Transport Credit Return DWORD (CRD) Bytes 0 to 3 (see Section 8.2.5.2.2 for CRD format). Similarly, Bytes 244 to 249 are driven from the Protocol Layer as 0. See Figure 3-26 for details.

If PCIe/CXL.io is negotiated along with Management Transport protocol on the same stack for *Format 6*:

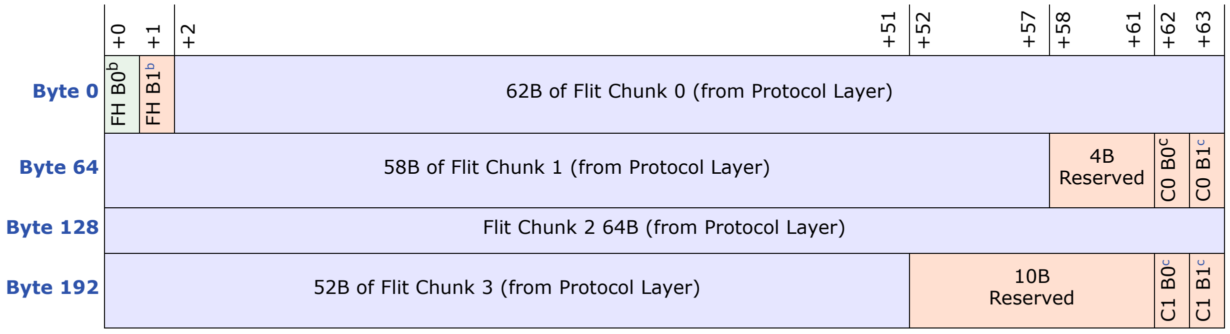
- If bits [7:6] of Byte 1 are 10b, the Adapter passes through Bytes 122 to 125 and 248 to 253
- If bits [7:6] of Byte 1 are 00b, then Bytes 122 to 125 are treated per the PCIe/CXL.io DLP rules of this flit format, Bytes 250 to 253 are treated per the PCIe/CXL.io FM rules of this flit format, and the Adapter drives 0 on Bytes 248 and 249

Figure 3-20. Format 5: Latency-Optimized 256B without Optional Bytes Flit Format for CXL.io[a]



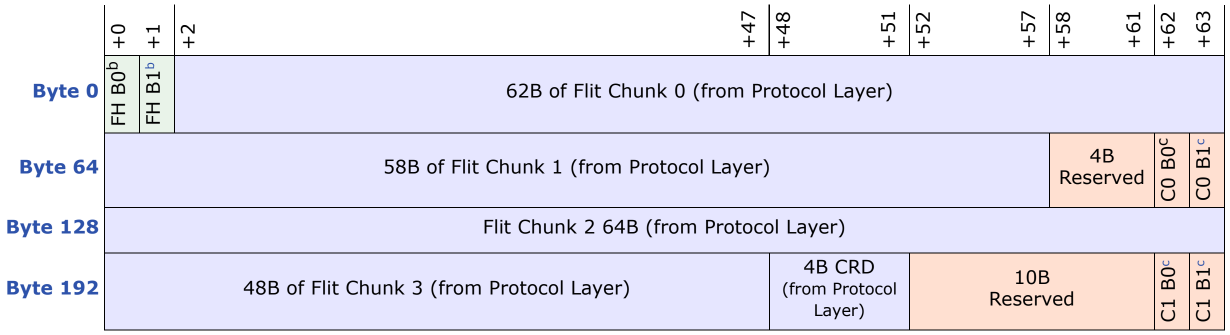
a. See Figure 2-1 for color mapping.
b. Flit Header Byte 0 and Byte 1, respectively.
c. DLP Byte 2, Byte 3, Byte 4, and Byte 5, respectively.
d. CRC0 Byte 0, CRC0 Byte 1, CRC1 Byte 0, and CRC1 Byte 1, respectively.
e. Flit_Marker or Optimized_Update_FC Byte 0, Byte 1, Byte 2, and Byte 3, respectively.

Figure 3-21. Format 5: Latency-Optimized 256B without Optional Bytes Flit Format for CXL.cachemem and Streaming Protocol[a]


+0
+1
+2
+51
+52
+57
+58
+61
+62
+63
Byte 0
FH B0b
FH B1b
62B of Flit Chunk 0 (from Protocol Layer)
Byte 64
58B of Flit Chunk 1 (from Protocol Layer)
4B Reserved
C0 B0c
C0 B1c
Byte 128
Flit Chunk 2 64B (from Protocol Layer)
Byte 192
52B of Flit Chunk 3 (from Protocol Layer)
10B Reserved
C1 B0c
C1 B1c

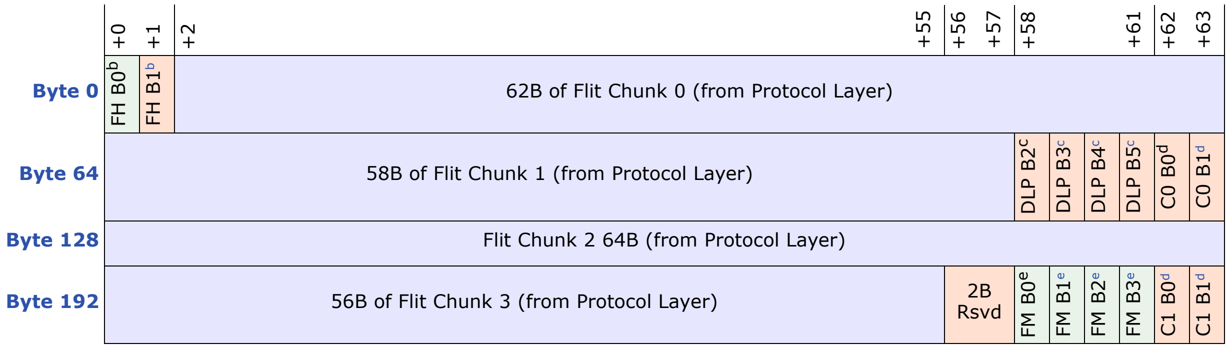
a. See Figure 2-1 for color mapping.
b. Flit Header Byte 0 and Byte 1, respectively.
c. CRC0 Byte 0, CRC0 Byte 1, CRC1 Byte 0, and CRC1 Byte 1, respectively.

Figure 3-22. Format 5: Latency-Optimized 256B without Optional Bytes Flit Format for Management Transport Protocol[a]


+0
+1
+2
+47
+48
+51
+52
+57
+58
+61
+62
+63
Byte 0
FH B0b
FH B1b
62B of Flit Chunk 0 (from Protocol Layer)
Byte 64
58B of Flit Chunk 1 (from Protocol Layer)
4B Reserved
C0 B0c
C0 B1c
Byte 128
Flit Chunk 2 64B (from Protocol Layer)
Byte 192
48B of Flit Chunk 3 (from Protocol Layer)
4B CRD (from Protocol Layer)
10B Reserved
C1 B0c
C1 B1c

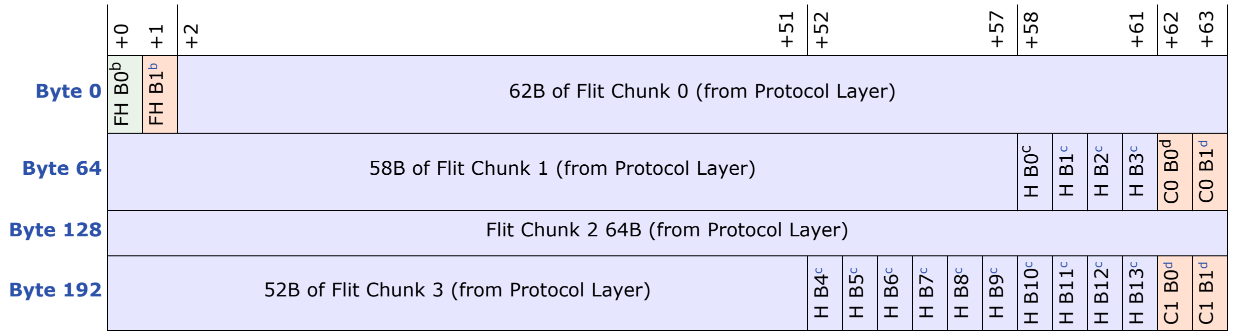
a. See Figure 2-1 for color mapping.
b. Flit Header Byte 0 and Byte 1, respectively.
c. CRC0 Byte 0, CRC0 Byte 1, CRC1 Byte 0, and CRC1 Byte 1, respectively.

Figure 3-23. Format 6: Latency-Optimized 256B with Optional Bytes Flit Format for CXL.io or PCIe[a]


+0
+1
+2
+55
+56
+57
+58
+61
+62
+63
Byte 0
FH B0b
FH B1b
62B of Flit Chunk 0 (from Protocol Layer)
Byte 64
58B of Flit Chunk 1 (from Protocol Layer)
DLP B2c
DLP B3c
DLP B4c
DLP B5c
C0 B0d
C0 B1d
Byte 128
Flit Chunk 2 64B (from Protocol Layer)
Byte 192
56B of Flit Chunk 3 (from Protocol Layer)
2B Rsvd
FM B0e
FM B1e
FM B2e
FM B3e
C1 B0d
C1 B1d

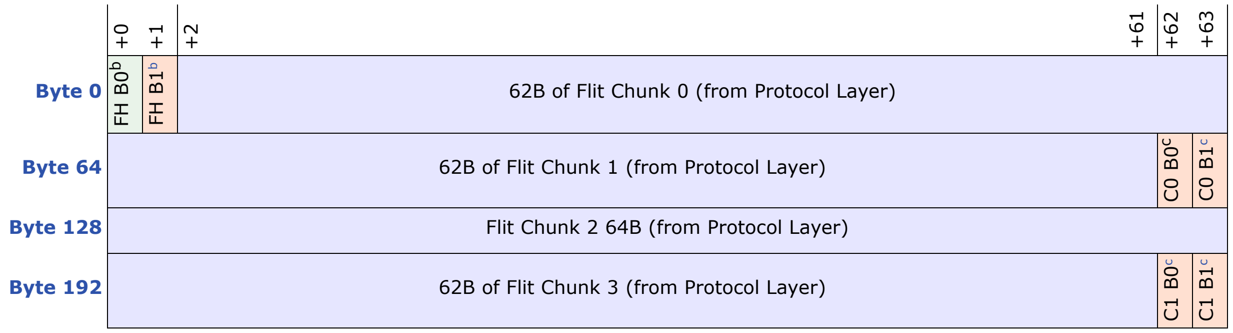
a. See Figure 2-1 for color mapping.
b. Flit Header Byte 0 and Byte 1, respectively.
c. DLP Byte 2, Byte 3, Byte 4, and Byte 5, respectively.
d. CRC0 Byte 0, CRC0 Byte 1, CRC1 Byte 0, and CRC1 Byte 1, respectively.
e. Flit_Marker Byte 0, Byte 1, Byte 2, and Byte 3, respectively.

Figure 3-24. Format 6: Latency-Optimized 256B with Optional Bytes Flit Format for CXL.cachemem[a]



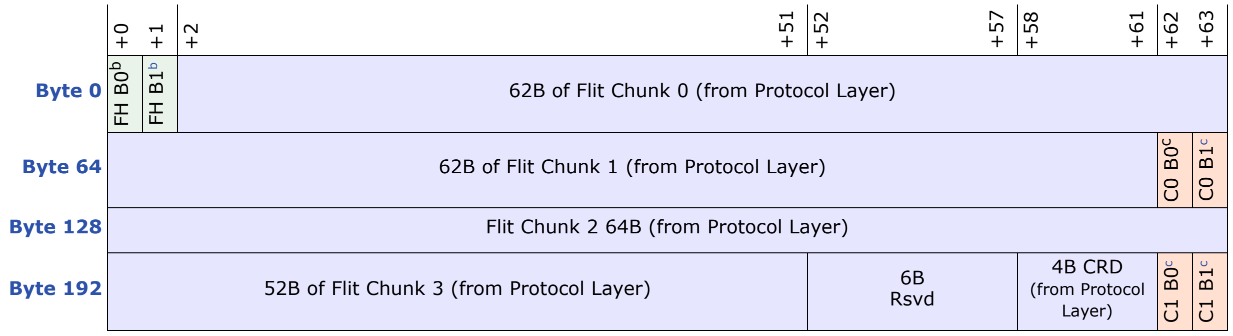
a. See Figure 2-1 for color mapping.
b. Flit Header Byte 0 and Byte 1, respectively.
c. H-slot Byte 0 through Byte 13, respectively (from Protocol Layer).
d. CRC0 Byte 0, CRC0 Byte 1, CRC1 Byte 0, and CRC1 Byte 1, respectively.

Figure 3-25. Format 6: Latency-Optimized 256B with Optional Bytes Flit Format for Streaming Protocol[a]


+0
+1
+2
+61
+62
+63
Byte 0
FH B0[b]
FH B1[b]
62B of Flit Chunk 0 (from Protocol Layer)
Byte 64
62B of Flit Chunk 1 (from Protocol Layer)
C0 B0[c]
C0 B1[c]
Byte 128
Flit Chunk 2 64B (from Protocol Layer)
Byte 192
62B of Flit Chunk 3 (from Protocol Layer)
C1 B0[c]
C1 B1[c]

a. See Figure 2-1 for color mapping.
b. Flit Header Byte 0 and Byte 1, respectively.
c. CRC0 Byte 0, CRC0 Byte 1, CRC1 Byte 0, and CRC1 Byte 1, respectively.

Figure 3-26. Format 6: Latency-Optimized 256B with Optional Bytes Flit Format for Management Transport Protocol[a]


+0
+1
+2
+51
+52
+57
+58
+61
+62
+63
Byte 0
FH B0[b]
FH B1[b]
62B of Flit Chunk 0 (from Protocol Layer)
Byte 64
62B of Flit Chunk 1 (from Protocol Layer)
C0 B0[c]
C0 B1[c]
Byte 128
Flit Chunk 2 64B (from Protocol Layer)
Byte 192
52B of Flit Chunk 3 (from Protocol Layer)
6B Rsvd
4B CRD (from Protocol Layer)
C1 B0[c]
C1 B1[c]

a. See Figure 2-1 for color mapping.
b. Flit Header Byte 0 and Byte 1, respectively.
c. CRC0 Byte 0, CRC0 Byte 1, CRC1 Byte 0, and CRC1 Byte 1, respectively.

The Flit Header byte formats are the same as Table 3-5 when Retry is required; otherwise, they are the same as Table 3-4. The DLP rules are also the same as defined in Section 3.3.3 for CXL protocol, except that Flit_Marker/Optimized_Update_FC has dedicated space in the Flit (i.e., bit [4] of Byte 0 corresponds to the Flit_Marker bytes, and not the DLP bytes). If Optimized_Update_FC is sent, the DLP Bytes 2:5 shown in Figure 3-20 must be reserved. If bit [4] of Byte 0 in the Flit Header is 0b, then the Flit_Marker bytes are reserved.

3.3.5 Flit Format-related Implementation Requirements for Protocol Layer and Adapter

Table 3-6 lists the different Flit Formats supported in UCIe.

Table 3-6. Summary of Flit Formats

Format Number	Name	Notes	For Details, See Also
Format 1	Raw	Protocol Layer populates all the bytes on FDI. Adapter passes to RDI without modifications or additions.	• Section 3.3.1 • Figure 3-10
Format 2	68B Flit	Protocol Layer transmits 64B per Flit on FDI. Adapter inserts two bytes of Flit header and two bytes of CRC and performs the required barrel shifting of bytes before transmitting on RDI. On the RX, Adapter strips out the Flit header and CRC only sending the 64B per Flit to the Protocol Layer on FDI.	• Section 3.3.2 • Figure 3-11 • Figure 3-12
Format 3	Standard 256B End Header Flit	Protocol Layer transmits 256B of Flit on FDI, while driving 0b on the bits reserved for the Adapter. Adapter fills in the relevant Flit header and CRC information before transmitting on RDI. On the Rx, Adapter forwards the Flit received from the Link to the Protocol Layer without modifying any bits applicable to the Protocol Layer, and the Protocol Layer must ignore any bits not applicable for it. Flit Header is located on Byte 236 and Byte 237 of the Flit.	• Section 3.3.3 • Figure 3-14 • Figure 3-15
Format 4	Standard 256B Start Header Flit	Protocol Layer transmits 256B of Flit on FDI, while driving 0b on the bits reserved for the Adapter. Adapter fills in the relevant Flit header and CRC information before transmitting on RDI. On the Rx, Adapter forwards the Flit received from the Link to the Protocol Layer without modifying any bits applicable to the Protocol Layer, and the Protocol Layer must ignore any bits not applicable for it. Flit Header is located on Byte 0 and Byte 1 of the Flit.	• Section 3.3.3 • Figure 3-17 • Figure 3-18
Format 5	Latency-Optimized 256B without Optional Bytes Flit	Protocol Layer transmits 256B of Flit on FDI, while driving 0b on the bits reserved for the Adapter. Adapter fills in the relevant Flit header and CRC information before transmitting on RDI. On the Rx, Adapter forwards the Flit received from the Link to the Protocol Layer without modifying any bits applicable to the Protocol Layer, and the Protocol Layer must ignore any bits not applicable for it. CRC bytes sent with each 128B of the Flit. The optional Protocol Layer bytes are reserved in this format and not used by the Protocol Layer.	• Section 3.3.4 • Figure 3-20 • Figure 3-21
Format 6	Latency-Optimized 256B with Optional Bytes Flit	Protocol Layer transmits 256B of Flit on FDI, while driving 0b on the bits reserved for the Adapter. Adapter fills in the relevant Flit header and CRC information before transmitting on RDI. On the Rx, Adapter forwards the Flit received from the Link to the Protocol Layer without modifying any bits applicable to the Protocol Layer, and the Protocol Layer must ignore any bits not applicable for it. CRC bytes sent with each 128B of the Flit, and optional bytes are used by the Protocol Layer.	• Section 3.3.4 • Figure 3-23 • Figure 3-24 • Figure 3-25

Table 3-7 gives the implementation requirements and Protocol Mapping for the different Flit Formats. For PCIe and CXL protocols, the implementation requirements must be followed by the Protocol Layer as well as the Adapter implementations. For Streaming protocols, the implementation requirements are for the Adapter only; Protocol Layer interoperability and implementation requirements are vendor specific.

Table 3-7. Protocol Mapping and Implementation Requirements

Format Number	Flit Format Name	PCIe Non-Flit Mode	PCIe Flit Mode	CXL 68B Flit Mode	CXL 256B Flit Mode	Streaming Protocol	Management Transport Protocol
1	Raw	Optional	Optional	Optional	Optional	Mandatory	Optional
2	68B	Mandatory	N/A	Mandatory	N/A	Optional[a]	N/A
3	Standard 256B End Header	N/A	Mandatory	N/A	N/A	Optional[a]	Optional
4	Standard 256B Start Header	N/A	Optional[b]	N/A	Mandatory	Optional[a]	Optional
5	Latency-Optimized 256B without Optional Bytes	N/A	N/A	N/A	Optional	Optional[a]	Optional
6	Latency-Optimized 256B with Optional Bytes	N/A	Strongly Recommended[c]	N/A	Strongly Recommended	Strongly Recommended[a]	Optional

a. If Streaming Flit Format capability is supported, else it is N/A.
b. If Standard Start Header for PCIe protocol capability is supported, else it is N/A.
c. If Latency-Optimized Flit with Optional Bytes for PCIe protocol capability is supported, else it is N/A.
If Enhanced Multi-Protocol capability is supported where at least one of the stacks supports PCIe, this format and the corresponding capability are strongly recommended.

3.4 Decision Table for Flit Format and Protocol

Table 3-8 shows the Truth Table for determining Protocol. Once the protocol and Flit Format have been negotiated during initial Link bring up, they cannot be changed until the UCIe Physical Layer transitions to Reset state.

If a valid Protocol and Flit Format are not negotiated, then the Adapter takes the Link down and reports the error if applicable.

Table 3-8. Truth Table for Determining Protocol[a]

{FinCap.Adapter} bits or {MultiProtFinCap.Adapter} bits[b]					{FinCap.CXL} bits		Protocol
68B Flit Mode	CXL 256B Flit Mode	PCIe Flit Mode	Streaming Protocol	Management Transport Protocol	PCIe	CXL.io	
1	0	0	x	x	0	1	CXL[c] without Management Transport protocol
1	1	1	x	0	0	1[d]	CXL[c] without Management Transport protocol
1	1	1	x	1	0	1[d]	CXL[c] with Management Transport protocol
1	0	0	x	x	1	0	PCIe[e] without Management Transport protocol
1	0	1	x	0	1[f]	0	PCIe without Management Transport protocol
1	0	1	x	1	1[f]	0	PCIe with Management Transport protocol
N/A	N/A	N/A	N/A	N/A	N/A	N/A	Streaming[g] with or without Management Transport protocol
N/A	N/A	N/A	N/A	N/A	N/A	N/A	Management Transport protocol[h]

a. x indicates don't care in this Table.
b. If Enhanced_Multi-Protocol capability is negotiated then {MultiProt*.Adapter} messages are used to determine the protocol for Stack 1. Stack 0 protocol is determined using the {FinCap.*} messages.
c. For CXL protocol, the specific combination of Single Protocol vs Type 1 vs. Type 2 vs. Type 3 is determined using the CXL.cache and CXL.mem capable/enable bits in addition to the CXL.io capable/enable bit in {FinCap.CXL}. The rules for that follow *CXL Specification*. When CXL is the protocol, if CXL 256B Flit mode is 1, then the protocol follows CXL 256B Flit mode rules; otherwise, the protocol follows CXL 68B Flit mode rules.
d. CXL.io capable/enable must be 1 if CXL 256B Flit mode is negotiated.
e. For PCIe protocol, if PCIe Flit mode is 1, then the protocol follows PCIe Flit mode rules; otherwise, the protocol follows PCIe Non-Flit mode rules.
f. PCIe capable/enable must be 1 if PCIe Flit mode is 1 but CXL 256B Flit mode is 0.
g. No {FinCap.*} message is sent for Streaming protocol negotiation, Streaming is the negotiated protocol if PCIe or CXL are not advertised, but Streaming protocol is advertised. If Management Transport protocol was also advertised along with Streaming protocol, then Management Transport protocol is enabled along with Streaming protocol.
h. No {FinCap.*} message is sent for Management Transport protocol negotiation, Management Transport is the negotiated protocol if PCIe or CXL or Streaming are not advertised, but Management Transport protocol is advertised.

IMPLEMENTATION NOTE

The “68B Flit Mode” parameter is advertised as set to 1 for both the CXL and PCIe protocols in {AdvCap.Adapter} sideband messages. As seen in Table 3-8, this parameter is set to 1 in {FinCap.Adapter} sideband messages whenever the CXL OR PCIe protocols are negotiated.

The “CXL.io” and “PCIe” bits in the {AdvCap.CXL} sideband message disambiguate between CXL support vs. PCIe support. It is permitted to set both to 1 in {AdvCap.CXL} sideband messages. However, as seen in Table 3-8, only one of these must be set in the {FinCap.CXL} sideband message to reflect the final negotiated protocol for the corresponding stack. For example:

- If the DP and UP both support CXL and PCIe protocols, then both “CXL.io” and “PCIe” will be set to 1 in the {AdvCap.CXL} sideband message
- If the DP decides to operate in CXL, the DP will set “CXL.io” to 1 and clear “PCIe” to 0 in the {FinCap.CXL} sideband message, in which case the remaining CXL-related bits in the {FinCap.CXL} sideband message are also applicable and are assigned as per the negotiation

Table 3-9 (Truth Table 1) shows the truth table for deciding the Flit format in which to operate if PCIe or CXL protocols are negotiated (with or without Management Transport protocol), and none of the following are negotiated:

- Enhanced Multi_Protocol_Enable
- Standard 256B Start Header for PCIe protocol capability
- Latency-Optimized Flit with Optional Bytes for PCIe protocol capability

Table 3-10 (Truth Table 2) provides the Truth Table for determining the Flit Format for Streaming protocols if Streaming Flit Format capability is negotiated or if Management Transport protocol is negotiated without CXL or PCIe or Streaming protocols on the same stack. Note that for Streaming protocol negotiation or for Management Transport protocol negotiation without CXL or PCIe protocol multiplexed on the same stack, there are no {FinCap.*} messages exchanged. Each side of the UCIe Link advertises its own capabilities in the {AdvCap.Adapter} message it sends. The bits in Table 3-10 represent the logical AND of the corresponding bits in the sent and received {AdvCap.Adapter} messages. Truth Table 2 must be followed for determining the Flit Format if both sides of the Link have any of the following capabilities are supported and enabled for both sides of the Link:

- Enhanced Multi-Protocol Capability
- Standard Start Header Flit for PCIe protocol capability
- Latency-Optimized Flit with Optional Bytes for PCIe protocol capability

For situations where {FinCap.Adapter} messages are sent, the bits in the truth table represent the bits set in the {FinCap.Adapter} message.

It is permitted for the Adapter OR the Protocol Layer to take the Link down to LinkError if the desired Flit Format is not negotiated or the negotiated Flit format and protocol combination is illegal (e.g., 68B Flit *Format 2* and Management Transport protocol combination).

Property of Universal Chiplet Interconnect Express (UCIe) © 2022–2024

Table 3-9. Truth Table 1

{FinCap.Adapter} bits[a]						Flit Format
Raw Format	68B Flit Mode	CXL 256B Flit Mode	PCIe Flit Mode	CXL_LatOpt_Fmt5	CXL_LatOpt_Fmt6	
1	x	x	x	x	x	*Format 1*: Raw Format
0	x	1	x	0	0	*Format 4*: Standard 256B Start Header Flit Format for CXL
0	x	1	x	x	1	*Format 6*: Latency-Optimized 256B with Optional Bytes Flit Format for CXL
0	x	1	x	1	0	*Format 5*: Latency-Optimized 256B without Optional Bytes Flit Format for CXL
0	x	0	1	x	x	*Format 3*: Standard 256B End Header Flit Format for PCIe
0	1	0	0	x	x	*Format 2*: 68B Flit Format

a. x indicates don't care.

Table 3-10. Truth Table 2

Logical AND of Corresponding Bits in the Sent and Received {AdvCap.Adapter} Message OR the Bits Sent in the {FinCap.Adapter} Message[c]						Final Negotiated Flit Format[a]
Raw Format[b]	68B Flit Format[c]	Standard 256B End Header Flit Format	Standard 256B Start Header Flit Format	Latency-Optimized 256B without Optional Bytes Flit Format	Latency-Optimized 256B with Optional Bytes Flit Format	
1	x	x	x	x	x	*Format 1*: Raw Format
0	1	0	0	x	0	*Format 2*: 68B Flit Format
0	x	1	0	x	0	*Format 3*: Standard 256B End Header Flit Format
0	x	x	1	x	0	*Format 4*: Standard 256B Start Header Flit Format
0	0	0	0	1	0	*Format 5*: Latency-Optimized 256B without Optional Bytes Flit Format
0	x	x	x	x	1	*Format 6*: Latency-Optimized 256B with Optional Bytes Flit Format

a. *Format 6* is the highest priority format when Raw Format is not advertised because it has the best performance characteristics. Between *Format 4* and *Format 3*, *Format 4* is higher priority because it enables lower latency through the D2D Adapter when multiplexing different protocols. *Format 5* has the highest overhead and therefore has the lowest priority relative to other formats.
b. Raw Format is always explicitly enabled through UCIe Link Control register and advertised only when it is the required format of operation to ensure interoperability, and therefore appears as a higher priority in the decision table.
c. x indicates don't care.

3.5 State Machine Hierarchy

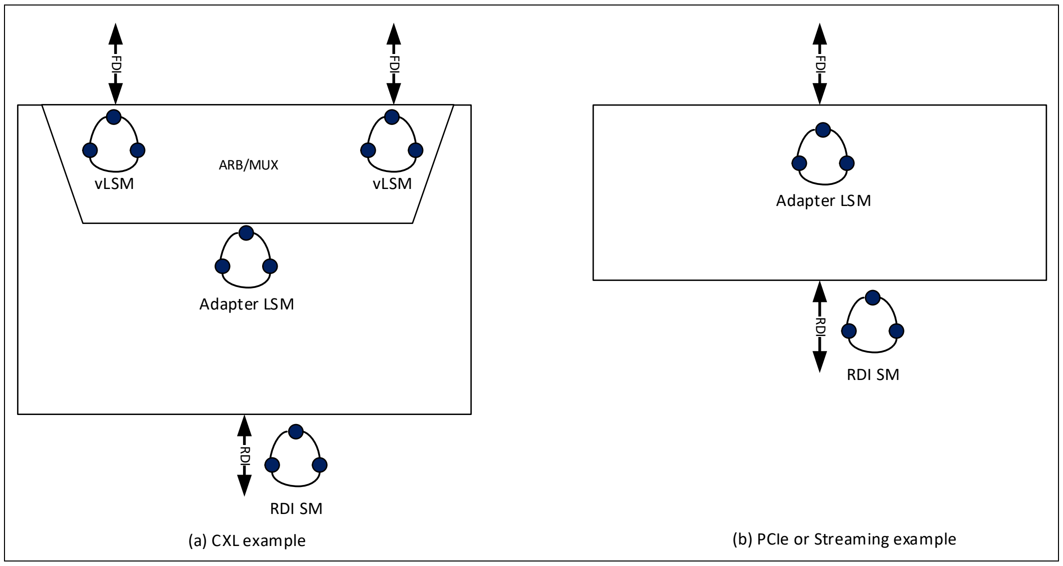
UCIe has a hierarchical approach to Link state management in order to have well-defined functionality partitioning between the different layers and also enabling common state transitions or sequencing at FDI and RDI.

Figure 3-27 shows examples of state machine hierarchy for different configurations. For CXL, the ARB/MUX vLSMs are exposed on FDI **`pl_state_sts`**. The Adapter LSM is used to coordinate Link states with remote Link Partner and is required for all configurations. Each protocol stack has its corresponding Adapter LSM. For PCIe or Streaming protocols, the Adapter LSM is exposed on FDI **`pl_state_sts`**.

The RDI state machine (SM) is used to abstract the Physical Layer states for the upper layers. The Adapter data path and RDI data width can be extended for multi-module configurations; however, there is a single RDI state machine for this configuration. The Multi-module PHY Logic creates the abstraction and coordinates between the RDI state and individual modules. The following rules apply:

- vLSM state transitions are coordinated with remote Link partner using ALMPs on mainband data path. The rules for state transitions follow the CXL 256B Flit Mode rules in the *CXL Specification*.
- Adapter LSM state transitions are coordinated with remote Link partner using {LinkMgmt.Adapter*} sideband messages. These messages are originated and received by the D2D Adapter.
- RDI SM state transitions are coordinated with the remote Link partner using {LinkMgmt.RDI*} sideband messages. These messages are originated and received by the Physical Layer.

Figure 3-27. State Machine Hierarchy Examples



General rules for State transition hierarchy are captured below. For specific sequencing, see the rules outlined in Chapter 10.0.

- Active State transitions: RDI SM must be in Active before Adapter LSM can begin negotiation to transition to Active. Adapter LSM must be in Active before vLSMs can begin negotiations to transition to Active.
- Retrain State transitions: RDI SM must be in Retrain before propagating Retrain to Adapter LSMs. If RDI SM is in Retrain, Retrain must be propagated to all Adapter LSMs that are in Active state.

Adapter must not request Retrain exit on RDI before all the relevant Adapter LSMs have transitioned to Retrain.

- PM State transitions (both L1 and L2): Both CXL.io and CXL.cachemem vLSMs (if CXL), must transition to PM before the corresponding Adapter LSM can transition to PM. All Adapter LSMs (if multiple stacks are enabled on the same Adapter) must be in PM before RDI SM is transitioned to PM.
- LinkError State transitions: RDI SM must be in LinkError before Adapter LSM can transition to LinkError. RDI SMs coordinate LinkError transition with remote Link partner using sideband, and each RDI SM propagates LinkError to all enabled Adapter LSMs. Adapter LSM must be in LinkError before propagating LinkError to both vLSMs if CXL. LinkError transition takes priority over LinkReset or Disabled transitions. Adapter must not request LinkError exit on RDI before all the relevant Adapter LSMs and CXL vLSMs have transitioned to LinkError.
- LinkReset or Disabled State transitions: Adapter LSM negotiates LinkReset or Disabled transition with its remote Link partner using sideband messages. LinkReset or Disabled is propagated to RDI SM only if all the Adapter LSMs associated with it transition to LinkReset or Disabled. Disabled transition takes priority over LinkReset transition. If RDI SM moves to LinkReset or Disabled, it must be propagated to all Adapter LSMs. If Adapter LSM moves to LinkReset or Disabled, it must propagate it to both vLSMs for CXL protocol.

For UCIe Retimers, it is the responsibility of the Retimer die to negotiate state transitions with the remote Retimer partner and make sure the different UCIe Die are in sync and do not time out waiting for a response. As an example, referring to Figure 1-18, if UCIe Die 0 sends an Active Request message for the Adapter LSM to UCIe Retimer 0, UCIe Retimer 0 must resolve with UCIe Retimer 1 that an Active Request message has been forwarded to UCIe Die 1 and that UCIe Die 1 has responded with an Active Status message before responding to UCIe Die 0 with an Active Status message. The Off Package Interconnect cannot be taken to a low power state unless all the relevant states on UCIe Die 0 AND UCIe Die 1 have reached the low power state. UCIe Retimers must respond with "Stall" encoding every 4ms while completing resolution with the remote Retimer partner.

3.6 Power Management Link States

Power management states are mandatory for PCIe and CXL protocols. FDI supports L1 and L2 power states which follow the handshake rules and state transitions of CXL 256B Flit Mode. RDI supports L1 and L2 on the interfaces for Physical Layer to perform power management optimizations; however, the Physical Layer is permitted to internally map both L1 and L2 to a common state. These together allow for global clock gating and enable system level flows like Package-Level Idle (C-states). Other Protocols are permitted to disable PM flows by always sending a PMNAK for a PM request from remote Link partner.

When Management Transport protocol is supported and negotiated with CXL.io/PCIe/Streaming on the same stack, L1 and L2 entry requests to the Adapter from the Management Port Gateway multiplexer (MPG mux) must comprehend L1 and L2 entry readiness of the Management Transport protocol as well as the co-located protocol stack, in an implementation specific manner. Additionally, the MPG mux must also follow the FDI semantics for PM rules of the co-located CXL.io/PCIe/Streaming protocol. Similarly, L1 and L2 exit would wake both the Management Transport protocol and as well as the co-located protocol stack, and exit flow semantics must adhere to the negotiated CXL.io/PCIe/Streaming protocol.

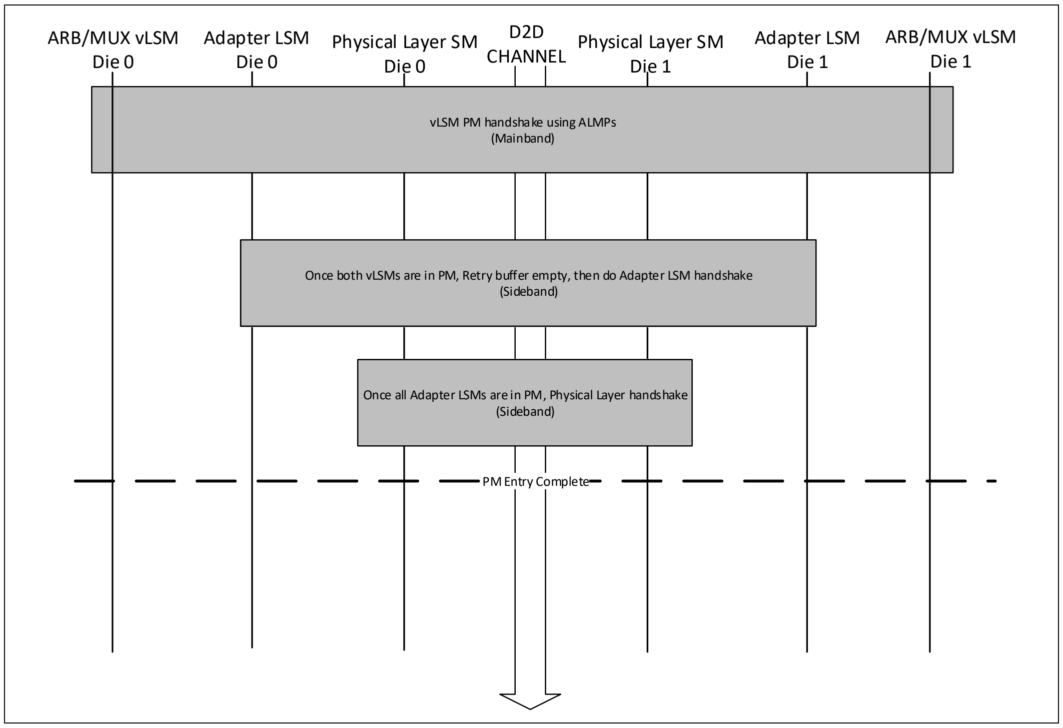
The Power management state entry sequence is as follows:

1. **Protocol Layer PM entry request**: FDI defines a common flow for PM entry request at the interface that is based on Link idle time. All protocols using UCIe must follow that flow when PM needs to be supported. For CXL protocol, D2D Adapter implements the ARB/MUX functionality and follows the handshakes defined in *CXL Specification* (corresponding to the "CXL 256B Flit Mode",

since all ALMPs also go through the Retry buffer in UCIe). Even CXL 68B Flit Mode over UCIe uses the “CXL 256B Flit Mode” ALMP formats and flows (but the Flit is truncated to 64B and two bytes of Flit header and two bytes of CRC are added by the Adapter to make a 68B Flit). For PCIe protocol in UCIe Flit Mode, PM DLLP handshakes are NOT used. Protocol Layer requests PM entry on FDI based on Link idle time. The specific algorithm and hysteresis for determining Link idle time is implementation specific.

2. **Adapter Link State Machine PM entry**: The PM transition for this is coordinated over sideband with remote Link partner. In scenarios where the Adapter is multiplexing between two protocol stacks, each stack’s Link State Machine must transition to PM independently.
3. **PM entry on RDI**: Once all the Adapter’s LSMs are in a PM state, the Adapter initiates PM entry on the RDI as defined in Section 10.2.9.
4. Physical Layer moves to a deeper PM state and takes the necessary actions for power management. Note that the sideband Link must remain active because the sideband Link is used to initiate PM exit.

Figure 3-28. Example of Hierarchical PM Entry for CXL


ARB/MUX vLSM Die 0
Adapter LSM Die 0
Physical Layer SM Die 0
D2D CHANNEL
Physical Layer SM Die 1
Adapter LSM Die 1
ARB/MUX vLSM Die 1
vLSM PM handshake using ALMPs (Mainband)
Once both vLSMs are in PM, Retry buffer empty, then do Adapter LSM handshake (Sideband)
Once all Adapter LSMs are in PM, Physical Layer handshake (Sideband)
PM Entry Complete

PM exit follows the reverse sequence of wake up as mentioned below:

1. Active request from Protocol Layer is transmitted across the FDI and RDI to the local Physical Layer.
2. The Physical Layer uses sideband to coordinate wake up and retraining of the physical Link.
3. Once the physical Link is retrained, the RDI is in Active state on both sides, and the Adapter LSM PM exit is triggered from both sides (coordinated via sideband messages between Adapters as outlined in the FDI PM flow). For PCIe or Streaming protocol scenarios, this also transitions the Protocol Layer to Active state on FDI.
4. For CXL protocol, this step is followed by ALMP exchanges to bring the required protocol to Active state and then protocol Flit transfer can begin.

3.7 CRC Computation

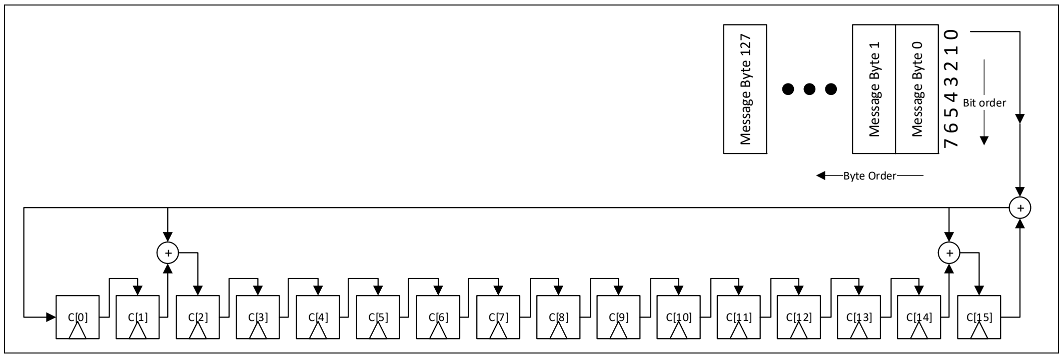
The CRC generator polynomial is (x+1)*(x^{15} + x + 1) = x^{16} + x^{15} + x^2 + 1. This gives a 3-bit detection guarantee for random bit errors: 2 bit detection guarantee is because of the primitive polynomial (x^{15} + x + 1), and 1 additional bit error detection guarantee is provided by making it odd parity because of the (x+1) term in the polynomial.

The CRC is always computed over 128 bytes of the message. For smaller messages, the message is zero extended in the MSB. Any bytes which are part of the 128B CRC message but are not transmitted over the Link are assigned to 0b. Whenever non-CRC bytes of the Flit populated by the Adapter are included for CRC computation (e.g., the Flit Header or DLP bytes), CRC is computed after the Adapter has assigned those bytes the values that will be sent over the UCIe Link. Any reserved bits which are part of the Flit are assigned 0b for the purpose of CRC computation.

The initial value of CRC bits for CRC LFSR computation is 0000h. The CRC calculation starts with bit 0 of byte 0 of the message, and proceeds from bit 0 to bit 7 of each byte as shown in Figure 3-29. In the figure, C[15] is bit 7 of CRC Byte 1, C[14] is bit 6 of CRC Byte 1 and so on; C[7] is bit 7 of CRC Byte 0, C[6] is bit 6 of CRC Byte 0 and so on.

The Verilog code for CRC code generation is provided in crc_gen.vs (attached to the PDF copy of this Specification). This Verilog code must be used as the golden reference for implementing the CRC during encode or decode. The code is provided for the Transmit side. It takes 1024 bits (bit 1023 is bit 7 of message Byte 127, 1022 is bit 6 of message Byte 127 and so on; bit 1015 is bit 7 of message Byte 126 and so on until bit 0 is bit 0 of message Byte 0) as an input message and outputs 16 bits of CRC. On the Receiver, the CRC is computed using the received Flit bytes with appropriate zero padding in the MSB to form a 128B message. If the received CRC does not match the computed CRC, the flit is declared Invalid and a replay must be requested.

Figure 3-29. Diagram of CRC Calculation



3.8 Retry Rules

For configurations where the raw BER is higher than 1e-27, Retry must be supported in the Adapter, unless the only format of operation is Raw Format. If Retry is not supported by the Adapter, Link speeds where the raw BER is higher than 1e-27 must NOT be advertised by the Physical Layer during Link Training, unless the format of operation is Raw Format. See Table 5-26 for the raw BER characteristics of different configurations. Once Retry has been negotiated during Part 2 of Stage 3 of Link Initialization described in Section 3.2.1.2, it cannot be disabled even if Link speed degrades during runtime. Retry can only be re-negotiated at the next Link Initialization (i.e., RDI moves to Reset). For multiple stacks with a common Adapter, the Tx Retry buffer is shared between the stacks.

The Retry scheme on UCIe is a simplified version of the Retry mechanism for Flit Mode defined in *PCIe Base Specification*. The rules that differ from PCIe are as follows:

- Selective Nak and associated rules are not applicable and must not be implemented. Rx Retry Buffer-related rules are also not applicable and must not be implemented.
- Throughout the duration of Link operation, when not conflicting with PCIe rules of replay, Explicit Sequence number Flits and Ack/Nak Flits alternate. This allows for faster Ack turnaround and thus smaller Retry buffer sizes. It is permitted to send consecutive Explicit Sequence number Flits if there are no pending Ack/Nak Flits to send (see also the Implementation Note below). To meet this requirement, all Explicit Sequence Number Flit transmissions described by the PCIe rules of replay that require the condition "CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLIT < 3" to be met require "CONSECUTIVE_TX_EXPLICIT_SEQ_NUM_FLIT < 1" to be met instead, and it is not required to send three consecutive Flits with Explicit Sequence Number.
- All 10-bit retry related counters are replaced with 8-bit counters, and the maximum-permitted sequence number is 255 (hence 1023 in all calculations is replaced with 255 and any variables defined in the "Flit Sequence Number and Retry Mechanism" section of *PCIe Base Specification* which had an initial value of 1023 instead have an initial value of 255).
- REPLAY_TIMEOUT_FLIT_COUNT is a 9-bit counter that saturates at 1FFh.
  — In addition to incrementing REPLAY_TIMEOUT_FLIT_COUNT as described in *PCIe Base Specification*, the count must also be incremented when in Active state and a Flit Time (Number of Adapter clock cycles (**`lclk`**) that are required to transfer 256B of data at the current Link speed and width) has elapsed since the last flit was sent and neither a Payload Flit nor a NOP flit was transmitted. The counter must be incremented for every Flit Time in which a flit was not sent (this could lead to it being incremented several times in-between flits or prior to the limit being met). The added requirement compensates for the noncontinuous transfer of NOP flits. For 68B Flit Format, data transfers are also in 256B granularity (including the PDS bytes), and thus this counter increments every time 256B of data are transmitted, OR during idle conditions in Active state, it must be incremented according to the time that is required to transfer 256B of data at the current Link speed and width.
  — Replay Schedule Rule 0 of *PCIe Base Specification* must check for REPLAY_TIMEOUT_FLIT_COUNT ≥ 375. Replay Timer Timeout error is logged in the Correctable Internal Error in the Adapter for UCIe.
- For the FLIT_REPLAY_NUM counter, it is strongly recommended to follow the rules provided in *PCIe Base Specification* for speeds ≤ 32.0 GT/s. This counter tracks the number of times that a Replay has occurred without making forward progress. Given the significantly lower probability of Replay for UCIe Links, the rules associated with ≤ 32.0 GT/s PCIe speeds are sufficient for UCIe.
- NAK_WITHDRAWAL_ALLOWED is always cleared to 0. Note that this requires implementations to set the flag NAK_SCHEDULED=1 in the "Nak Schedule 0" set of rules.
- IDLE Flit Handshake Phase is not applicable. This is because the transition to Link Active (equivalent to LTSSM being in L0 for PCIe) is managed via handshakes on sideband, and there is no requirement for IDLE Flits to be exchanged. As per PCIe rules, any Flits received with all 0s in

the Flit Header bytes are discarded by the Adapter. Any variables that are initialized during the IDLE Flit Handshake Phase in *PCIe Base Specification* are initialized to the corresponding value whenever the RDI is in Reset state or Retrain state. Similarly, PCIe rules that indicate relation to “last entry to IDLE Flit Handshake Phase” would instead apply for UCIe to “last exit from Reset or Retrain state on RDI”.

- Variables applicable to Flit Sequence number and Retry mechanism that are initialized during DL_Inactive, as with PCIe, would be initialized to their corresponding values when RDI is in Reset state for UCIe.
- Sequence Number Handshake Phase must be performed on every entry of the RDI to Active state from Reset state or Retrain state (after Flit transfers are permitted). Sequence Number Handshake Phase timeout and exit to Link Retrain is 128 Flits transmitted without exiting Sequence Number Handshake Phase. As with PCIe, both NOP flits or Payload flits are permitted to be used to complete the Sequence Number Handshake Phase. If there are no Payload flits to send, the Adapter must generate NOP flits to complete the Sequence Number Handshake Phase.
- The variable “Prior Flit was Payload” is always set to 1. This bit does not exist in the Flit Header, and thus from the Retry perspective, implementations must assume that it is always set to 1.
- MAX_UNACKNOWLEDGED_FLITS is set to the lesser of:
  - — Number of Flits that can be stored in the Tx Retry Buffer, or
  - — 127
- Flit Discard 2 rule from PCIe does not result in a Data Link Protocol Error condition in UCIe. Receiving an invalid Flit Sequence number in a received Ack or Nak flit (see the corresponding conditions in *PCIe Base Specification* with the adjusted variable widths and values) OR a Payload Flit with an Explicit Sequence number of 0 results in an Uncorrectable Internal Error in UCIe (instead of a Data Link Protocol Error).
- Conditions from the “Flit Sequence Number and Retry Mechanism” section in *PCIe Base Specification* that led to Recovery for the Port must result in the Adapter initiating Retrain on the RDI for UCIe.

IMPLEMENTATION NOTE

In UCIe, to encourage power savings through dynamic clock gating, it is not required to continuously transmit NOP flits during periods in which there are no Payload flits or any Ack/Nak pending. Consider an example in which an Adapter’s Tx Retry Buffer is empty and it transmitted a NOP flit with an Ack as the last flit before it stopped sending additional flits to the Physical Layer. Let’s say this flit had a CRC error and hence the remote Link partner never receives this Ack. Moreover, because the remote Link partner received a flit with a CRC error, it would transmit a Nak to original sender. If the Ack is never re-sent and the remote Link partner has a corresponding Payload flit in its Tx Retry Buffer, eventually a Replay Timeout will trigger from the remote Link partner and resolve this scenario. However, rather than always relying on Replay Timeout for these kind of scenarios, it is recommended for implementations to ensure they have transmitted at least two flits with an Ack (these need not be consecutive Ack flits) before stopping flit transfer whenever a Nak is received and the transmitter has completed all the requirements of received Nak processing, including any Replay related transfers. If no new Payload Flits were received from the remote Link partner, as per PCIe rules, it is permitted to re-send the last transmitted Ack on a NOP flit as well to meet this condition.

3.9 Runtime Link Testing using Parity

UCIe defines a mechanism to detect Link health during runtime by periodically injecting parity bytes in the middle of the data stream when this mechanism is enabled. The receiver checks and logs parity errors for the inserted parity bytes.

When this mechanism is enabled, the Adapter inserts 64*N Bytes every 256*256*N Bytes of data, where N is obtained from the Error and Link Testing Control register (Field name: Number of 64 Byte Inserts). Software must set N=4 when this feature is enabled during regular Link operation for UCIe Flit mode because that makes the parity bytes also a multiple of 256B and is more consistent with the granularity of data transfer. Only bit 0 of the inserted byte has the parity information which is computed as follows:

ParityByte X, bit 0 = ^((DataByte [X]) ^ (DataByte [X + 64*N]) ^(DataByte [X + 128*N])^....^(DataByte [X + (256*256*N - 64*N)]))

The remaining 7 bits of the inserted byte are Reserved.

The Transmitter and Receiver in the Adapter must independently keep track of the number of data bytes elapsed to compute or check the parity information. If the RDI state moves away from Active state, the data count and parity is reset, and both sides must renegotiate the enabling of the Parity insertion before next entry to Active from Retrain (if the mechanism is still enabled in the Error and Link Testing Control register). When entering Active state with Parity insertion enabled, the number of data bytes elapsed begins counting from 0. On the transmitter, following the insertion of the parity information, the counter for the number of bytes elapsed to compute the parity information is reset. On the Receiver, following the receipt and check of parity bytes, the counter for the number of bytes elapsed to check the parity information is reset.

This mechanism is enabled by Software writing 1b to the enable bit in the register located in both Adapters across a UCIe Link (see Section 9.5.3.9 for register details). Software must trigger UCIe Link Retrain after writing to the enable bit on both the Adapters. Support for this feature in Raw Format is beyond the scope of this specification and is implementation-dependent. The Adapters exchange sideband messages while the Adapter LSMs are in Retrain to ensure the remote Link partner’s receiver is prepared to receive the extra parity bytes in the data stream once the states transition to Active. The Adapter must not request Retrain exit to local RDI until the Parity Feature exchanges are completed. It is permitted to enable it during Initial Link bring up, by using sideband to access the remote Link partner’s registers or other implementation specific means; however software must trigger Link Retrain for the feature to take effect.

Adapter sends a {ParityFeature.Req} sideband message to remote Link Partner if its Transmitter is enabled to send parity bytes (“Runtime Link Testing Tx Enable” bit in Section 9.5.3.9). Remote Adapter responds with a {ParityFeature.Ack} sideband message if its receiver is enabled and ready to accept parity bytes (“Runtime Link Testing Rx Enable” bit in Section 9.5.3.9). Figure 3-30 shows an example of a successful negotiation. If Die 0 Adapter Transmitter is enabled to insert parity bytes, it must send a {ParityFeature.Req} from Die 0 to Die 1.

Adapter responds with a {ParityFeature.Nak} if it is not ready to accept parity bytes, or if the feature has not been enabled for it yet. The requesting Adapter must log the Nak in a status register so that Software can determine that a Nak had occurred. Figure 3-31 shows an example of an unsuccessful negotiation.

Note: The Adapters are permitted to transition to a higher latency data path if the Parity Feature is enabled. The explicit Ack/Nak handshake is provided to ensure both sides have sufficient time to transition to alternate data path for this mechanism.

The Parity bytes do not consume Retimer receiver buffer credits. The Retimer receiver must not write the Parity bytes into its receiver buffer or forward these to remote Retimer partner over the Off Package Interconnect. This mechanism is to help characterize local UCIe Links only.

Figure 3-30. Successful Parity Feature negotiation between Die 1 Tx and Die 0 Rx

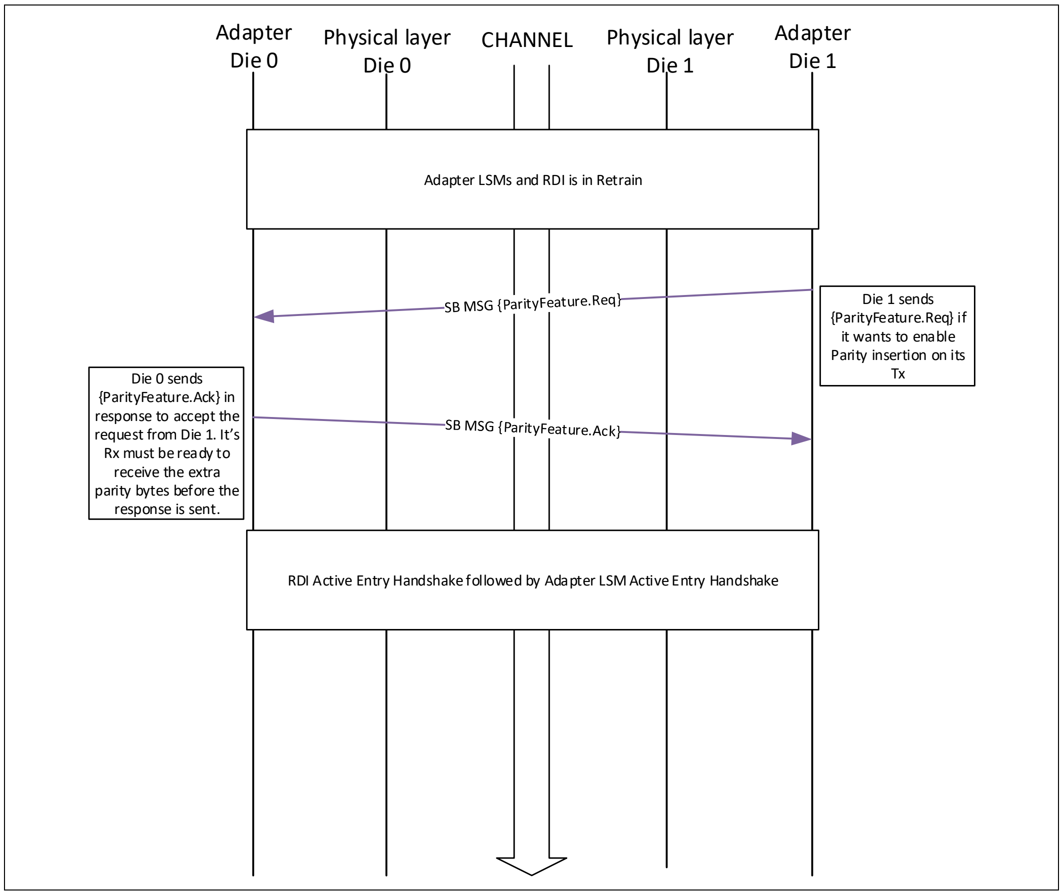
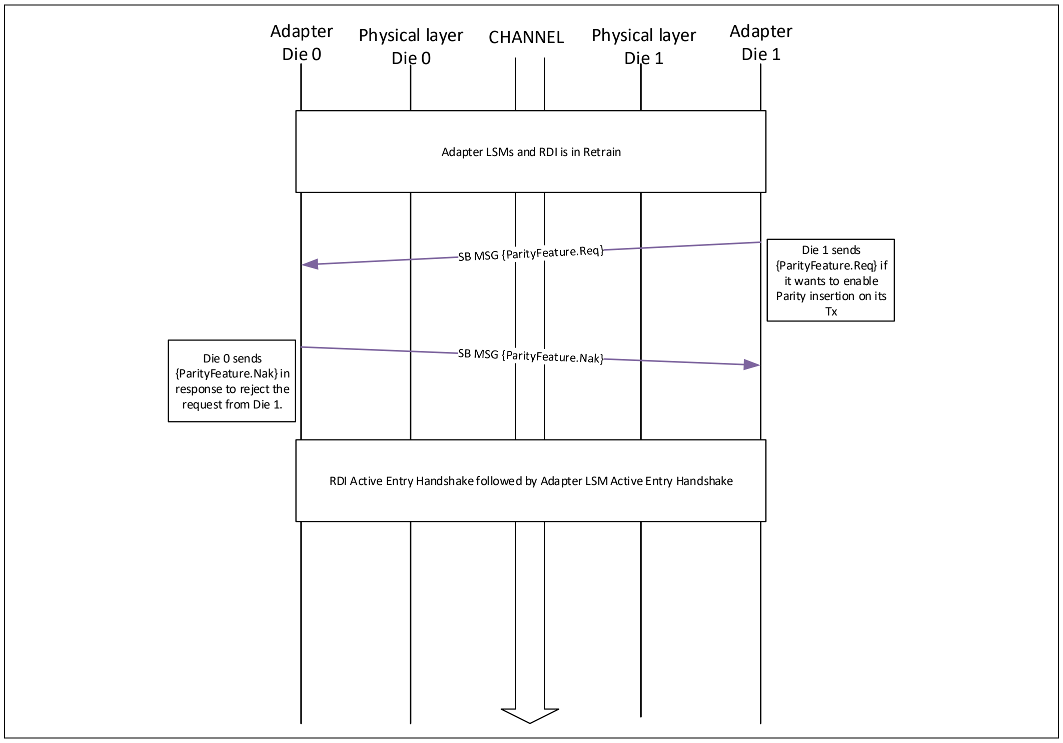

Adapter Die 0
Physical layer Die 0
CHANNEL
Physical layer Die 1
Adapter Die 1
Adapter LSMs and RDI is in Retrain
SB MSG {ParityFeature.Req}
Die 1 sends {ParityFeature.Req} if it wants to enable Parity insertion on its Tx
Die 0 sends {ParityFeature.Ack} in response to accept the request from Die 1. It's Rx must be ready to receive the extra parity bytes before the response is sent.
SB MSG {ParityFeature.Ack}
RDI Active Entry Handshake followed by Adapter LSM Active Entry Handshake

Figure 3-31. Unsuccessful Parity Feature negotiation between Die 1 Tx and Die 0 Rx


Adapter Die 0
Physical layer Die 0
CHANNEL
Physical layer Die 1
Adapter Die 1
Adapter LSMs and RDI is in Retrain
SB MSG {ParityFeature.Req}
Die 1 sends {ParityFeature.Req} if it wants to enable Parity insertion on its Tx
SB MSG {ParityFeature.Nak}
Die 0 sends {ParityFeature.Nak} in response to reject the request from Die 1.
RDI Active Entry Handshake followed by Adapter LSM Active Entry Handshake

If a parity error is detected by a chiplet, the error is treated as a Correctable error and reported via the correctable error reporting mechanism. By enabling interrupt on correctable errors, SW can implement a BER counter in SW, if so desired.

When a Pause Data Stream occurs the Pause Data Stream and corresponding padding bytes are included in the number of bytes elapsed before parity injection as well as parity computation.

§ §