10.0 Interface Definitions

This chapter will cover the details of interface operation and signal definitions for the Raw Die-to-Die Interface (RDI), as well as the Flit-Aware Die-to-Die Interface (FDI). Common rules across RDI and FDI are covered as a separate section. The convention used in this chapter is that "assertion" of a signal is for 0b to 1b transition, and "de-assertion" of a signal is for 1b to 0b transition. A "pulse" of "n" cycles for a signal is defined as an event where the signal transitions from 0b to 1b, stays 1b for "n" clock cycles, and subsequently returns to 0b. A receiver sampling this signal on the same clock as the transmitter will see it being asserted for "n" clock cycles. If a value of "n" is not specified, it is interpreted as a value of one. In the context of error signals defined as pulses, the receiving logic for error logging must treat the rising edge as a new event indication and not rely on the length of the pulse.

In this chapter, interface reset/domain reset also applies to all forms of Conventional Reset defined in *PCIe Base Specification*, if the Protocol is PCIe or CXL. In the sections that follow, "UCIe Flit mode" refers to scenarios in which the Link is not operating in Raw Format, and "UCIe Raw Format" or "Raw Format" refers to scenarios in which the Link is operating in Raw Format.

10.1 Raw Die-to-Die Interface (RDI)

This section defines the signal descriptions and functionality associated with a single instance of Raw Die-to-Die Interface (RDI). A single instance could be used for a configuration associated with a single Die-to-Die module (i.e., one Die-to-Die Adapter for one module), or a single instance is also applicable for configurations where multiple modules are grouped together for a single logical Die-to-Die Link (i.e., one Die-to-Die Adapter for multiple modules). Figure 10-1 shows example configurations using RDI.

Figure 10-1. Example configurations using RDI

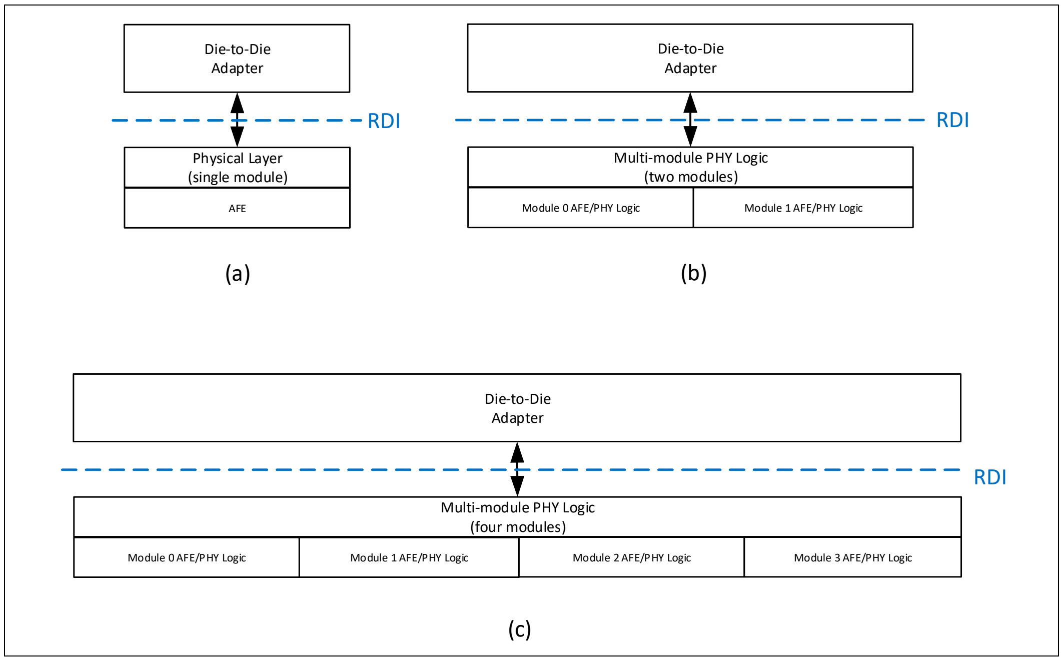


Table 10-1 lists the RDI signals and their descriptions. All signals are synchronous with **`lclk`**.

In Table 10-1:

- **`pl_*`** indicates that the signal is driven away from the Physical Layer to the Die-to-Die Adapter.
- **`lp_*`** indicates that the signal is driven away from the Die-to-Die Adapter to the Physical Layer.

Table 10-1. RDI signal list (Sheet 1 of 5)

Signal Name	Signal Description
`lclk`	The clock at which RDI operates.
`lp_irdy`	Adapter to Physical Layer signal indication that the Adapter has data to send. This must be asserted if **`lp_valid`** is asserted and the Adapter wants the Physical Layer to sample the data. **`lp_irdy`** must not be presented by the Adapter when **`pl_state_sts`** is Reset except when the status transitions from LinkError to Reset. On a LinkError to Reset transition, it is permitted for **`lp_irdy`** to be asserted for a few clocks but it must be de-asserted eventually. Physical Layer must ignore **`lp_irdy`** when status is Reset.
`lp_valid`	Adapter to Physical Layer indication that data is valid on the corresponding **`lp_data`** bytes.
`lp_data[NBYTES-1:0][7:0]`	Adapter to Physical Layer data, where 'NBYTES' equals number of bytes determined by the data width for the RDI instance.
`lp_retimer_crd`	When asserted at a rising clock edge, it indicates a single credit return from the Adapter to the Physical Layer for the Retimer Receiver buffers. Each credit corresponds to 256B of mainband data. This signal must NOT assert for dies that are not UCIe Retimers.
`pl_trdy`	The Physical Layer is ready to accept data. Data is accepted by the Physical Layer when **`pl_trdy`**, **`lp_valid`**, and **`lp_irdy`** are asserted at the rising edge of **`lclk`**. This signal must only be asserted if **`pl_state_sts`** is Active or when performing the **`pl_stallreq/lp_stallack`** handshake when the **`pl_state_sts`** is LinkError (see Section 10.3.3.7).

Table 10-1. RDI signal list (Sheet 2 of 5)

Signal Name	Signal Description
`pl_valid`	Physical Layer to Adapter indication that data is valid on **`pl_data`**.
`pl_data[NBYTES-1:0][7:0]`	Physical Layer to Adapter data, where NBYTES equals the number of bytes determined by the data width for the RDI instance.
`pl_retimer_crd`	When asserted at a rising clock edge, it indicates a single credit return from the Retimer to the Adapter. Each credit corresponds to 256B of mainband data. This signal must NOT assert if the remote Link partner is not a Retimer.
`lp_state_req[3:0]`	Adapter request to Physical Layer to request state change. Encodings as follows: 0000b: NOP 0001b: Active 0100b: L1 1000b: L2 1001b: LinkReset 1011b: Retrain 1100b: Disabled All other encodings are reserved.
`lp_linkerror`	Adapter to Physical Layer indication that an error has occurred which requires the Link to go down. Physical Layer must move to LinkError state and stay there as long as **`lp_linkerror`**=1. The reason for having this be an indication decoupled from regular state transitions is to allow immediate action on part of the Adapter and Physical Layer in order to provide the quickest path for error containment when applicable (for example, a viral error escalation must map to the LinkError state). The Adapter must OR internal error conditions with **`lp_linkerror`** received from Protocol Layer on FDI.
`pl_state_sts[3:0]`	Physical Layer to Adapter Status indication of the Interface. Encodings as follows: 0000b: Reset 0001b: Active 0011b: Active.PMNAK 0100b: L1 1000b: L2 1001b: LinkReset 1010b: LinkError 1011b: Retrain 1100b: Disabled All other encodings are reserved. The status signal is permitted to transition from Physical Layer autonomously when applicable. For example the Physical Layer asserts the Retrain status when it decides to enter retraining either autonomously or when requested by remote agent.
`pl_inband_pres`	Physical Layer to the Adapter indication that the Die-to-Die Link has finished training and is ready for RDI transition to Active and Stage 3 of bring up. Once it transitions to 1b, this must stay 1b until Physical Layer determines the Link is down (i.e., the Link Training State Machine transitions to TrainError or Reset).

Table 10-1. RDI signal list (Sheet 3 of 5)

Signal Name	Signal Description
`pl_error`	Physical Layer to the Adapter indication that it has detected a framing related error which is recoverable through Link Retrain. An example is where the Physical Layer received an invalid encoding on the Valid Lane. It is a pulse of one or more cycles that must occur only when RDI is in Active state. It is permitted to de-assert at the same clock edge where the state transitions away from Active state. It is pipelined with the receive data path such that the error indication reaches the Adapter before or at the same time as the corrupted data. Physical Layer is expected to go through Retrain flow after this signal has been asserted and it must not send valid data to Adapter until the Link has retrained. It is permitted for the Physical Layer to squash the **`pl_valid`** internally for the corrupted data. Once **`pl_error`** is asserted, **`pl_valid`** should not be asserted (without **`pl_error`** assertion in the same cycle) until the state status has transitioned to Active after completing a successful Retrain entry and exit. If **`pl_error`**=1 and **`pl_valid`**=1 in the same clock cycle, the Adapter must discard the corresponding Flit (even if it is only partially received when **`pl_error`** asserted). In UCIe Flit mode, when retry is enabled, it is the responsibility of the Adapter to ensure data integrity for Flits forwarded to FDI, and that they are canceled following the rules of **`pl_flit_cancel`** if they are suspected of corruption (see Section 10.2). A couple of examples are given below: • For 68B Flit Format, the Adapter could discard partially received Flits, but in 256B Latency optimized modes, it could have processed one half correctly, and the error may have happened on the other half, and so it has to track that and process future flits accordingly. • Another example is if it is not doing store/forward and only received 64B of a 128B half, and **`pl_error`** happened before receiving the remaining 64B of the 128B half, it needs to send dummy data for the second 64B and do a **`pl_flit_cancel`** for that half of the Flit. In UCIe Flit mode with Retry enabled for the Adapter, Retrain exit would naturally result in a Replay of any partially received Flits eventually (see Section 3.8). In UCIe Flit mode with Retry disabled, the Adapter must map **`pl_error`** assertion to an Uncorrectable Internal Error and escalate it accordingly. If the Link is operating in Raw Format, the Adapter forwards **`pl_error`** to the Protocol Layer such that it is pipeline matched to the data bus, and Protocol Layer handles it in an implementation-specific manner.
`pl_cerror`	Physical Layer to the Adapter indication that a correctable error was detected that does not affect the data path and will not cause Retrain on the Link. In UCIe Flit mode with Retry enabled, the Adapter must OR the **`pl_error`** and **`pl_cerror`** signals for Correctable Internal Error Logging. In UCIe Flit mode with Retry disabled or when the Link is operating in Raw Format, the Adapter must only use **`pl_cerror`** for Correctable Internal Error Logging. It is a pulse of one or more cycles which can occur in any RDI state. If it is a state in which clock gating is permitted, it is the responsibility of the Physical Layer to perform the clock gating exit handshake with the Adapter before asserting this signal. Clock gating can resume once **`pl_cerror`** de-asserts and all other conditions permitting clock gating are satisfied.
`pl_nferror`	Physical Layer to the Adapter indication that a non-fatal error was detected. There is no architecturally defined error condition for the Physical Layer currently asserting this signal; however, the signal is provided on the interface for any implementation-specific non-fatal errors. The Adapter treats this in the same manner as when it received a Sideband Non-Fatal Error Message from the remote Link partner. It is a pulse of one or more cycles that can occur in any RDI state. If it is a state where clock gating is permitted, it is the responsibility of the Physical Layer to perform the clock gating exit handshake with the Adapter before asserting this signal. Clock gating can resume after **`pl_nferror`** is de-asserted and all other conditions permitting clock gating have been met.
`pl_trainerror`	Indicates a fatal error from the Physical Layer. Physical Layer must transition **`pl_state_sts`** to LinkError if not already in LinkError state. This must be escalated to upper Protocol Layers based on the mask and severity programming of Uncorrectable Internal Error in the Adapter. Implementations are permitted to map any fatal error to this signal that require upper layer escalation (or interrupt generation) depending on system-level requirements. It is a level signal that can assert in any RDI state but remains asserted until RDI exits the LinkError state to Reset state.

Table 10-1. RDI signal list (Sheet 4 of 5)

Signal Name	Signal Description
pl_phyinrecenter	Physical Layer indication to Adapter that the Physical Layer is training or retraining. If this is asserted during a state where clock gating is permitted, the **pl_clk_req/ lp_clk_ack** handshake must be performed with the upper layer. The upper layers are permitted to use this to update the "Link Training/Retraining" bit in the UCIe Link Status register.
pl_stallreq	Physical Layer request to Adapter to align Transmitter at Flit boundary and not send any new Flits to prepare for state transition. See Section 10.3.2.
lp_stallack	Adapter to Physical Layer indication that the Flits are aligned and stalled (if **pl_stallreq** was asserted). It is strongly recommended that this response logic be on a global free running clock, so the Adapter can respond to **pl_stallreq** with **lp_stallack** even if other significant portions of the Adapter are clock gated. See Section 10.3.2.
pl_speedmode[2:0]	Current Link speed. The following encodings are used: 000b: 4GT/s 001b: 8GT/s 010b: 12GT/s 011b: 16GT/s 100b: 24GT/s 101b: 32GT/s other encodings are reserved. The Adapter must only consider this signal to be relevant when the RDI state is Active or Retrain. For multi-module configurations, all modules must operate at the same speed.
pl_lnk_cfg[2:0]	Current Link Configuration. Indicates the current operating width of a module. 000b: x4 001b: x8 010b: x16 011b: x32 100b: x64 101b: x128 110b: x256 other encodings are reserved. This is the width of the UCIe physical die-to-die Link which may be composed of one to four modules. For UCIe-S the maximum encoding would be x64, for UCIe-A the maximum encoding would be x128 for UCIe-A x32 and x256 for UCIe-A x64. The Adapter must only consider this signal to be relevant when the RDI state is Active or Retrain. This signal indicates the total width across all Active Modules corresponding to the RDI instance.
pl_clk_req	Request from the Physical Layer to remove clock gating from the internal logic of the Adapter. This is an asynchronous signal relative to **lclk** from the Adapter's perspective since it is not tied to **lclk** being available in the Adapter. Together with **lp_clk_ack**, it forms a four-way handshake to enable dynamic clock gating in the Adapter. When dynamic clock gating is supported, the Adapter must use this signal to exit clock gating before responding with **lp_clk_ack**. If dynamic clock gating is not supported, it is permitted for the Physical Layer to tie this signal to 1b.
lp_clk_ack	Response from the Adapter to the Physical Layer acknowledging that its clocks have been ungated in response to **pl_clk_req**. This signal is only asserted when **pl_clk_req** is asserted, and de-asserted after **pl_clk_req** has de-asserted. When dynamic clock gating is not supported by the Adapter, it must stage **pl_clk_req** internally for one or more clock cycles and turn it around as **lp_clk_ack**. This way it will still participate in the handshake even though it does not support dynamic clock gating.

Table 10-1. RDI signal list (Sheet 5 of 5)

Signal Name	Signal Description
lp_wake_req	Request from the Adapter to remove clock gating from the internal logic of the Physical Layer. This is an asynchronous signal from the Physical Layer's perspective since it is not tied to **lclk** being available in the Physical Layer. Together with **pl_wake_ack**, it forms a four-way handshake to enable dynamic clock gating in the Physical Layer. When dynamic clock gating is supported, the Physical Layer must use this signal to exit clock gating before responding with **pl_wake_ack**. If dynamic clock gating is not supported, it is permitted for the Adapter to tie this signal to 1b.
pl_wake_ack	Response from the Physical Layer to the Adapter acknowledging that its clocks have been ungated in response to **lp_wake_req**. This signal is only asserted after **lp_wake_req** has asserted, and is de-asserted after **lp_wake_req** has de-asserted. When dynamic clock gating is not supported by the Physical Layer, it must stage **lp_wake_req** internally for one or more clock cycles and turn it around as **pl_wake_ack**. This way it will still participate in the handshake even though it does not support dynamic clock gating.
pl_cfg[NC-1:0]	This is the sideband interface from the Physical Layer to the Adapter. See Chapter 7.0 for packet format details. NC is the width of the interface. Supported values are 8, 16, and 32. Register accesses must be implemented by hardware to be atomic regardless of the width of the interface (i.e., all 32 bits of a register must be updated in the same cycle for a 32-bit register write, and similarly all 64 bits of a register must be updated in the same cycle for a 64-bit register write).
pl_cfg_vld	When asserted, indicates that **pl_cfg** has valid information that should be consumed by the Adapter.
pl_cfg_crd	Credit return for sideband packets from the Physical Layer to the Adapter for sideband packets. Each credit corresponds to 64 bits of header and 64 bits of data. Even transactions that do not carry data or carry 32 bits of data consume the same credit and the Physical Layer returns the credit once the corresponding transaction has been processed or deallocated from its internal buffers. See Section 7.1.3.1 for additional flow control rules. A value of 1 sampled at a rising clock edge indicates a single credit return. Because the advertised credits are design parameters, the Adapter transmitter updates the credit counters with initial credits on domain reset exit, and no initialization credits are returned over the interface. Credit returns must follow the same rules of clock gating exit handshakes as the sideband packets to ensure that no credit returns are dropped by the receiver of the credit returns.
lp_cfg[NC-1:0]	This is the sideband interface from Adapter to the Physical Layer. See Chapter 7.0 for details. NC is the width of the interface. Supported values are 8, 16, and 32. Register accesses must be implemented by hardware to be atomic regardless of the width of the interface (i.e., all 32 bits of a register must be updated in the same cycle for a 32-bit register write, and similarly all 64 bits of a register must be updated in the same cycle for a 64-bit register write).
lp_cfg_vld	When asserted, indicates that **lp_cfg** has valid information that should be consumed by the Physical Layer.
lp_cfg_crd	Credit return for sideband packets from the Adapter to the Physical Layer for sideband packets. Each credit corresponds to 64 bits of header and 64 bits of data. Even transactions that do not carry data or carry 32 bits of data consume the same credit and the Adapter returns the credit once the corresponding transaction has been processed or deallocated from its internal buffers. See Section 7.1.3.1 for additional flow control rules. A value of 1 sampled at a rising clock edge indicates a single credit return. Because the advertised credits are design parameters, the Physical Layer transmitter updates the credit counters with initial credits on domain reset exit, and no initialization credits are returned over the interface. Credit returns must follow the same rules of clock gating exit handshakes as the sideband packets to ensure that no credit returns are dropped by the receiver of the credit returns.

Signals in Table 10-2 apply only when supporting MPM over sideband. The choice for whether these signals run on the **lclk** or the **Mgmt_Clk** is implementation-specific.

Table 10-2. RDI Config interface extensions for Management Transport (Sheet 1 of 3)

Signal Name	Signal Description
pm_param_done	Management transport negotiation phase completed. Signal de-asserts after being asserted for two clocks. This signal asserts when MBINIT.PARAM management transport negotiation phase completes. Note that this signal is asserted even if MBINIT.PARAM Configuration or SBFE exchanges indicate no support for management transport in the partner chiplet.
pm_param_local_count[N-1:0]	Number of modules that successfully negotiated Management transport on transmit side. This field is sampled only when **pm_param_done** signal is asserted. 000b: 0 modules 001b: 1 module 010b: 2 modules 011b: 3 modules 100b: 4 modules Others: Reserved N=2 for 1, 2, or 3 modules scenarios, and N=3 for 4 modules scenario.
pm_param_remote_count[N-1:0]	Number of modules that successfully negotiated Management transport on receive side. This field is sampled only when **pm_param_done** signal is asserted. 000b: 0 modules 001b: 1 module 010b: 2 modules 011b: 3 modules 100b: 4 modules Others: Reserved N=2 for 1, 2, or 3 modules scenarios, and N=3 for 4 modules scenario.
mp_mgmt_init_done	Indication from Management Port Gateway that initialization phase completed (successfully or unsuccessfully). This signal is used by the PHY to advance the state machine state beyond MBINIT.PARAM, if other conditions allow. Signal de-asserts after being asserted for two clocks. The PHY should not depend on this signal for advancing the state machine when the management path is already up or when the partner chiplet indicated no support for management transport.
mp_mgmt_init_start	A two-clock trigger pulse from Management Port Gateway to PHY to start negotiation on the sideband links. Management Port Gateway must ensure that the **mp_mgmt_up** signal is de-asserted when this signal is pulsed. This signal forces the link state machine to RESET state (if it is not already there) and hence can bring the mainband link down from link up state. The standard TRAINERROR flow applies here as well for transitioning the state machine to RESET if the state machine is not already in that state when this signal is pulsed.
mp_mgmt_up	Indication from Management Port Gateway that is signaled along with **mp_mgmt_init_done**, that Management Transport Initialization Phase completed successfully (**mp_mgmt_up**=1) or unsuccessfully (**mp_mgmt_up**=0). This is used by PHY to set the SB_MGMT_UP flag.
mp_mgmt_port_gateway_ready	Indication to PHY that Management Port Gateway is ready for management transport path initialization. The PHY uses this as one of the conditions to trigger or respond to a trigger for Management Transport path initialization. This asserts after the Management Port Gateway is ready for management path setup after Management Reset. Once asserted, this signal de-asserts on a Management Port Gateway reset (either because of management domain reset or after a heartbeat timeout or an 'Init Done' Timeout or any fatal error on sideband) condition.
mp_stall_after_mbinit_param	Management Port Gateway asserts this signal concurrent with asserting **mp_mgmt_port_gateway_ready** to indicate to the PHY that it must stall the training on the receive side using an {MBINIT.PARAM SBFE resp} sideband message stall encoding as described in Section 4.5.3.3.1.2. This signal remains asserted until the MPG determines that stalling is no longer necessary (i.e., that it is okay for link initialization to proceed). When de-asserted, the receive side training does not cause a "stall". When a sideband-only link is negotiated, this signal is not used by the PHY to determine the Link training state machine progress.

Table 10-2. RDI Config interface extensions for Management Transport (Sheet 2 of 3)

Signal Name	Signal Description
pm_cfg_credit[N-1:0]	This is credit return for the Flow control buffers over RDI (see Section 8.2.5.1.1) used by the Management Port Gateway to transmit management packets to the remote Management Port Gateway. Each credit corresponds to 64 bits of buffer space. Physical Layer returns the credit once the corresponding transaction has been deallocated from its internal buffers. See Section 8.2.5.1.1 for additional flow control rules. Because the advertised credits are design parameters, the Management Port Gateway transmitter updates the credit counters with initial credits on Management reset exit or on 'Heartbeat timeout', and no initialization credits are returned over the interface for these conditions. Credit returns must follow the same rules of clock gating exit handshakes as the sideband packets to ensure that no credit returns are dropped by the receiver of the credit returns. There is a signal per RxQ-ID in the design and hence N can be 1, 2, 3, or 4.
mp_rxqid[N-1:0]	RxQ-ID associated with the message. Has meaning when **mp_mgmt_pkt** signal is asserted on a RDI transfer. Used by PHY to steer the packet to the correct SB link. On encapsulated MTPs and PM Req messages, this carries the far-end Rx queue's RxQ-ID. On Credit return, Init Done and PM Ack messages this carries the RxQ-ID of the local Rx queue associated with the message. N is either 2 (for 4 modules links scenarios) or 1 (1 or 2 modules links scenarios). There is a fixed mapping in the PHY between this value and a physical SB link and the mapping is determined post successful completion of management transport negotiation on the transmit side. The chosen SB link for a given RxQ-ID must be one of the SB links that successfully trained for management transport on the transmit side.
pm_rxqid[N-1:0]	RxQ-ID associated with the message. Has meaning when **pm_mgmt_pkt** signal is asserted on a RDI transfer. Used by Management Port Gateway to internally steer the packet to the correct RxQ. N is either 2 (for 4 modules/sideband-only links scenarios) or 1 (1 or 2 modules/sideband-only links scenarios). Valid for all MPM config bus transmissions. PHY uses the RxQ-ID from the first credit return message received from a given sideband link to drive these signals on config interface. These signals are undefined for SoC Capabilities message. The captured RxQ-ID value is reset only when the management path is reinitialized.
mp_wake_req	Request from the Management Port Gateway to remove clock gating from the internal logic of the Physical Layer that handles management transport traffic. This is an asynchronous signal from the Physical Layer's perspective since it is not tied to lclk being available in the Physical Layer. Together with **pm_wake_ack**, it forms a four-way handshake to enable dynamic clock gating in the Physical Layer for logic that handles management transport traffic. When dynamic clock gating is supported, the Physical Layer must use this signal to exit clock gating before responding with **pm_wake_ack**. If dynamic clock gating is not supported, Management Port Gateway must tie this signal to 1.
pm_wake_ack	Response from the Physical Layer to the Management Port Gateway acknowledging that its clocks have been ungated in response to **mp_wake_req**. This signal is only asserted after **mp_wake_req** has asserted, and is de-asserted after **mp_wake_req** has de-asserted. When dynamic clock gating is not supported by the Physical Layer, it must stage **mp_wake_req** internally for one or more clock cycles and turn it around as **pm_wake_ack**. This way it will still participate in the handshake even though it does not support dynamic clock gating.
pm_clk_req	Request from the Physical Layer to remove clock gating from the internal logic of the Management Port Gateway. This is an asynchronous signal relative to **lclk/Mgmt_clk** from the Management Port Gateway perspective because it is not tied to **lclk/Mgmt_clk** being available in the Management Port Gateway. Together with **mp_clk_ack**, it forms a four-way handshake to enable dynamic clock gating in the Management Port Gateway. When dynamic clock gating is supported, the Management Port Gateway must use this signal to exit clock gating before responding with **mp_clk_ack**. If dynamic clock gating is not supported, Physical Layer must tie this signal to 1.

Table 10-2. RDI Config interface extensions for Management Transport (Sheet 3 of 3)

Signal Name	Signal Description
mp_clk_ack	Response from the Management Port Gateway to the PHY acknowledging that its clocks have been ungated in response to **pm_clk_req**. This signal is asserted only when **pm_clk_req** is asserted, and de-asserted after **pm_clk_req** has de-asserted. When dynamic clock gating is not supported by the Management Port Gateway, it must stage **pm_clk_req** internally for one or more clock cycles and turn it around as **mp_clk_ack**. This way it will still participate in the handshake even though it does not support dynamic clock gating. When supporting dynamic clock gating of the Management Port Gateway, PHY must ensure that pulsed signals (e.g., **pm_param_done**), are delivered only after the **mp_clk_ack** is set to ensure that the Management Port Gateway saw those pulses.
mp_mgmt_pkt	During a valid RDI data transfer to PHY, this signal indicates whether the transfer is for an MPM. 0: Link management packet. 1: MPM. Used by PHY to steer the packet to the correct RDI credit buffer.
pm_mgmt_pkt	During a valid RDI data transfer from PHY, this signal indicates whether the transfer is for an MPM. 0: Link management packet. 1: MPM. Used by the Management Port Gateway to steer the packet to RxQ buffers or to D2D Adapter.
pm_so	When asserted, indicates to Management Port Gateway that SO mode was negotiated. On ports that have sideband-only link physically present, this can be tied off to 1.
Mgmt_clk	Optional clock used for the Configuration interface on the RDI for implementations in which the main RDI clock is not available for Management Transport path initialization.
pm_fatal_error	Set by any sideband link fatal error indication, such as parity error on a sideband packet. Cleared by a Management Reset.
mp_fatal_error	Used by Management Port Gateway to instruct the PHY to transition to TRAINERROR state. This is a two-clock pulse.

10.1.1 Interface reset requirements

RDI does not define a separate interface signal for reset; however, it is required that the logic entities on both sides of RDI are in the same reset domain and the reset for each side is derived from the same source. Because reset may be staggered due to SoC routing, all signals coming out of reset must be driven to 0, unless otherwise specified.

10.1.2 Interface clocking requirements

RDI requires both sides of the interface to be on the same clock domain. The clock domain for the sideband interface (***cfg***) is the same as the mainband signals when Management Transport is not supported. When Management Transport is supported, the sideband interface is permitted to be on a separate **Mgmt_clk** domain.

Each side is permitted to internally instantiate clock-crossing FIFOs if needed, as long as it does not violate the requirements at the interface itself.

It is important to note that back pressure is not possible from the Adapter to the Physical Layer on the main data path. So any clock-crossing-related logic internal to the Adapter must take this into consideration.

For example, for a 64-Lane module with a maximum speed of 16 GT/s, the RDI could be 64B wide running at 2 GHz to be exactly bandwidth matched.

10.1.3 Dynamic clock gating

Dynamic coarse clock gating is permitted in the Adapter and Physical Layer when **`pl_state_sts`** is Reset, LinkReset, Disabled, or PM. This section defines the rules around entry and exit of clock gating. Note that clock gating is not permitted in LinkError state; it is expected that for UCIe usages, error handlers will be enabled to make sure the Link is not stuck in LinkError state if the intent is save power for Links in error state.

10.1.3.1 Rules and description for lp_wake_req/pl_wake_ack handshake

Adapter can request removal of clock gating of the Physical Layer by asserting **`lp_wake_req`** (asynchronous to **`lclk`** availability in the Physical Layer). All Physical Layer implementations must respond with a **`pl_wake_ack`** (synchronous to **`lclk`**). The extent of internal clock ungating when **`pl_wake_ack`** is asserted is implementation-specific, but **`lclk`** must be available by this time to enable RDI signal transitions from the Adapters. The Wake Req/Ack is a full handshake and it must be used for state transition requests (on **`lp_state_req`** or **`lp_linkerror`**) when moving away from a state in which clock gating is permitted. It must also be used for sending packets on the sideband interface.

Rules for this handshake:

1. Adapter asserts **`lp_wake_req`** to request ungating of clocks by the Physical Layer.
2. The Physical Layer asserts **`pl_wake_ack`** to indicate that clock gating has been removed. There must be at least one clock cycle bubble between **`lp_wake_req`** assertion and **`pl_wake_ack`** assertion.
3. **`lp_wake_req`** must de-assert before **`pl_wake_ack`** de-asserts. It is the responsibility of the Adapter to control the specific scenario of de-assertion. As an example, when performing the handshake for a state request, it is permitted to keep **`lp_wake_req`** asserted until it observes the desired state status. Adapter is also permitted to keep **`lp_wake_req`** asserted through states where clock gating is not permitted in the Physical Layer (i.e., Active, LinkError, or Retrain).
4. **`lp_wake_req`** should not be the only consideration for Physical Layer to perform clock gating, it must take into account **`pl_state_sts`** and other internal or Link requirements before performing global and/or local clock gating.
5. When performing **`lp_wake_req/pl_wake_ack`** handshake for **`lp_state_req`** transitions or **`lp_linkerror`** transition, the Adapter is permitted to not wait for **`pl_wake_ack`** before changing **`lp_state_req`** or **`lp_linkerror`**.
6. When performing **`lp_wake_req/pl_wake_ack`** handshake for **`lp_cfg`** transitions, Adapter must wait for **`pl_wake_ack`** before changing **`lp_cfg`** or **`lp_cfg_vld`**. Because **`lp_cfg`** can have multiple transitions for a single packet transfer, it is necessary to make sure that the Physical Layer clocks are up before transfer begins.

10.1.3.2 Rules and description for pl_clk_req/lp_clk_ack handshake

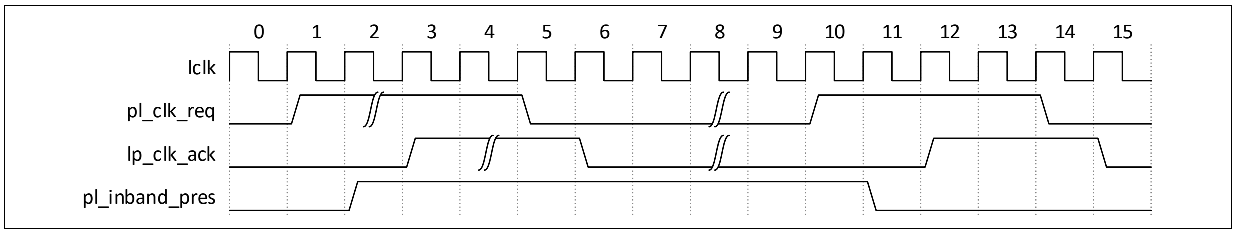
Physical Layer is permitted to initiate **`pl_clk_req/lp_clk_ack`** handshake at any time and the Adapter must respond.

Rules for this handshake:

1. Physical Layer asserts **`pl_clk_req`** to request removal of clock gating by the Adapter. This can be done anytime, and independent of current RDI state.
2. The Adapter asserts **`lp_clk_ack`** to indicate that clock gating has been removed. There must be at least one clock cycle bubble between **`pl_clk_req`** assertion and **`lp_clk_ack`** assertion.

3. **pl_clk_req** must de-assert before **lp_clk_ack**. It is the responsibility of the Physical Layer to control the specific scenario of de-assertion, after the required actions for this handshake are completed.
4. **pl_clk_req** should not be the only consideration for the Adapter to perform clock gating, it must take into account **pl_state_sts** and other protocol-specific requirements before performing trunk and/or local clock gating.
5. The Physical Layer must use this handshake to ensure transitions of **pl_inband_pres** have been observed by the Adapter. Since **pl_inband_pres** is a level oriented signal (once asserted it stays asserted during the lifetime of Link operation), the Physical Layer is permitted to let the signal transition without waiting for **lp_clk_ack**. When this is done during initial Link bring up, it is strongly recommended for the Physical Layer to keep **pl_clk_req** asserted until the state status transitions away from Reset to a state where clock gating is not permitted.

Figure 10-2. Example Waveform Showing Handling of Level Transition

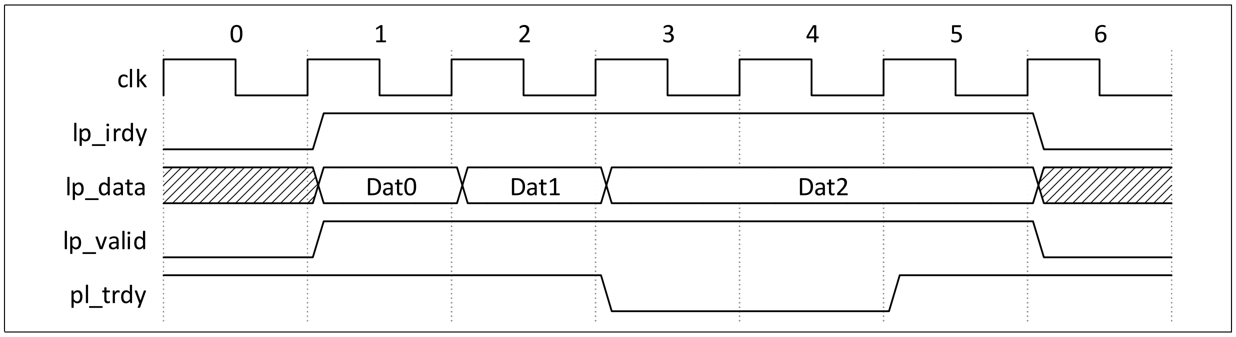


6. The Physical Layer must also perform this handshake before transition to LinkError state from Reset or PM state (when the LinkError transition occurs by the Physical Layer without being directed by the Adapter). It is permitted to assert **pl_clk_req** before the state change, in which case it must stay asserted until the state status transitions. It is also permitted to assert **pl_clk_req** after the state status transition, but in this case Physical Layer must wait for **lp_clk_ack** before performing another state transition.
7. The Physical Layer must also perform this handshake when the status is PM and remote Link partner is requesting PM exit. For exit from Reset or PM states to a state that is not LinkError, it is required to assert pl_clk_req before the status change, and in this case it must stay asserted until the state status transitions away from Reset or PM.
8. When clock-gated in RESET states, Adapters that rely on dynamic clock gating to save power must wait in clock gated state for **pl_inband_pres**=1. The Physical Layer will request clock gating exit when it transitions **pl_inband_pres**, and the Adapter must wait for **pl_inband_pres** assertion before requesting **lp_state_req** = ACTIVE. If **pl_inband_pres** de-asserts while **pl_state_sts** = RESET, then the Adapter is permitted to return to clock-gated state after moving **lp_state_req** to NOP.
9. Physical Layer must also perform this handshake for sideband traffic to Adapter. When performing the handshake for **pl_cfg** transitions, Physical Layer must wait for **lp_clk_ack** before changing **pl_cfg** or **pl_cfg_vld**. Because **pl_cfg** can have multiple transitions for a single packet transfer, it is necessary to make sure that the Adapter clocks are up before transfer begins.

10.1.4 Data Transfer

As indicated in the signal list descriptions, when Adapter is sending data to the Physical Layer, data is transferred when **`lp_irdy`**, **`pl_trdy`**, and **`lp_valid`** are asserted. Figure 10-3 shows an example waveform for data transfer from the Adapter to the Physical Layer. Data is transmitted on clock cycles 1, 2, and 5. No assumption should be made by Adapter about when **`pl_trdy`** can de-assert or for how many cycles it remains de-asserted before it is asserted again, unless explicitly guaranteed by the Physical Layer. If a Flit transfer takes multiple clock cycles, the Adapter is not permitted to insert bubbles in the middle of a Flit transfer. This means that **`lp_valid`** and **`lp_irdy`** must be asserted continuously until the Flit transfer is complete. Of course, data transfer can stall because of **`pl_trdy`** de-assertion.

Figure 10-3. Data Transfer from Adapter to Physical Layer



As indicated in the signal list descriptions, when the Physical Layer is sending data to the Adapter, there is no backpressure mechanism, and data is transferred whenever **`pl_valid`** is asserted. The Physical Layer is permitted to insert bubbles in the middle of a Flit transfer and the Adapter must be able to handle that.

IMPLEMENTATION NOTE

For the transmit side of the Physical Layer for data sent over the UCIe Link, it must ensure that if the Adapter has a continuous stream of packets to transmit (**`lp_irdy`** and **`lp_valid`** do not de-assert), it does not insert bubbles in valid frames on the Physical Link.

For the Runtime Link Testing feature with parity insertion, the Adapter as a receiver of parity bytes is permitted to issue a {ParityFeature.Nak} if software sets up a number of parity byte insertions ("Number of 64 Byte Inserts" field in the "Error and Link Testing Control" register) that does not amount to 256B or a multiple of the RDI width (to save the implementation cost of barrel shifting the parity bytes). For example, if the RDI width is 64B then either 64B, 128B, or 256B of inserted parity bytes are okay, but if the RDI width is 256B or larger, then it is better to always have 256B of inserted parity bytes so that it matches the data transfer granularity of Flits.

IMPLEMENTATION NOTE

It is permitted to use **`lp_irdy`** as an early indication that the valid data will be resuming imminently, and the Physical Layer needs to ungate clocks and assert **`pl_trdy`** when it is ready to receive data. A couple of examples are shown in Figure 10-4 and Figure 10-5. Note that **`pl_trdy`** could have asserted as early as Clock Cycle 1 in Figure 10-4.

Figure 10-4. lp_irdy asserting two cycles before lp_valid

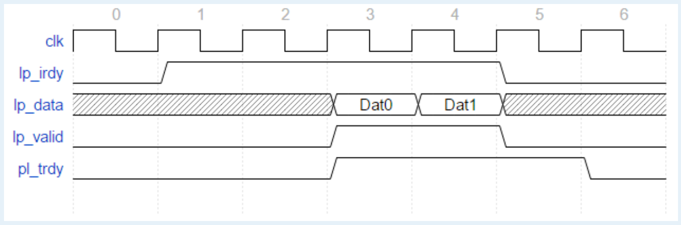
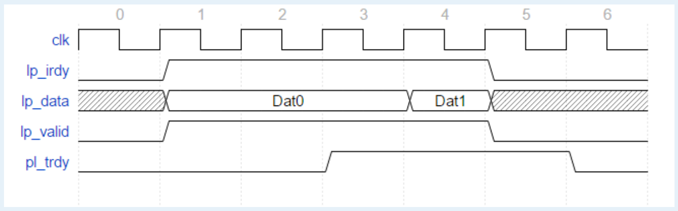


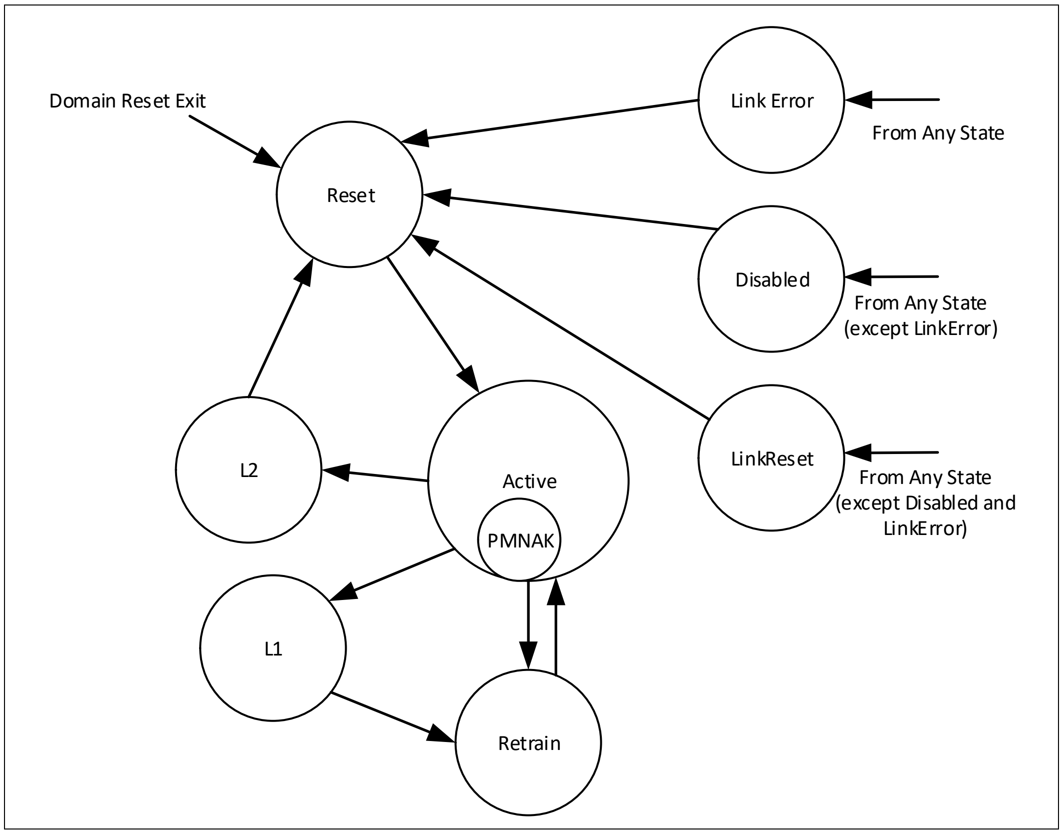
Figure 10-5. lp_irdy asserting at the same cycle as lp_valid



10.1.5 RDI State Machine

Figure 10-6 shows the RDI state machine.

Figure 10-6. RDI State Machine



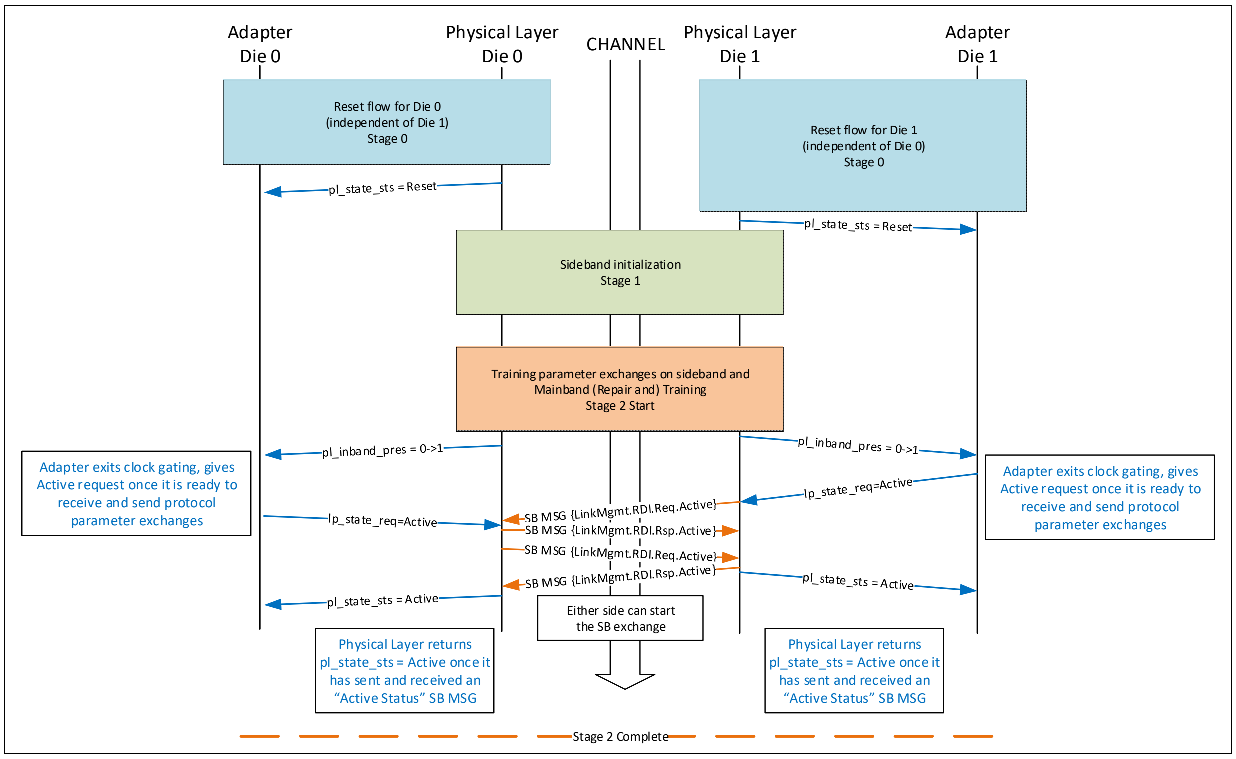
10.1.6 RDI bring up flow

Figure 10-7 shows an example flow for Stage 2 of the Link bring up highlighting the transitions on RDI. This stage requires sequencing on RDI that coordinates the state transition from Reset to Active.

1. Once Physical Layer has completed Link training, it must do the **`pl_clk_req`** handshake with the Adapter and reflect **`pl_inband_pres`**=1 on RDI. Note that the **`pl_clk_req`** handshake is not shown in the example flow in Figure 10-7
2. This is the trigger for Adapter to request Active state. It must perform the **`lp_wake_req`** handshake as described in Section 10.1.3. Note that the **`lp_wake_req`** handshake is not shown in the example flow in Figure 10-7.
3. Only after sampling **`lp_state_req`** = Active, the Physical Layer must send the {LinkMgmt.RDI.Req.Active} sideband message to remote Link partner's Physical Layer.
4. The Physical Layer must respond to the {LinkMgmt.RDI.Req.Active} sideband message with a {LinkMgmt.RDI.Rsp.Active} sideband message. The {LinkMgmt.RDI.Rsp.Active} sideband message must only be sent after the Physical Layer has sampled **`lp_state_req`** = Active from its local RDI.
5. Once the Physical Layer has sent and received the {LinkMgmt.RDI.Rsp.Active} sideband message, it must transition **`pl_state_sts`** to Active.
6. This opens up the Adapter to transition to Stage 3 of the bring up flow.

Steps 3 to 5 are referred to as the "Active Entry handshake" and must be performed for every entry to Active state. Active.PMNAK to Active transition is not considered here because Active.PMNAK is only a sub-state of Active.

Figure 10-7. Example flow of Link bring up on RDI


Adapter Die 0
Physical Layer Die 0
CHANNEL
Physical Layer Die 1
Adapter Die 1
Reset flow for Die 0 (independent of Die 1) Stage 0
Reset flow for Die 1 (independent of Die 0) Stage 0
pl_state_sts = Reset
pl_state_sts = Reset
Sideband initialization Stage 1
Training parameter exchanges on sideband and Mainband (Repair and) Training Stage 2 Start
pl_inband_pres = 0->1
pl_inband_pres = 0->1
Adapter exits clock gating, gives Active request once it is ready to receive and send protocol parameter exchanges
Adapter exits clock gating, gives Active request once it is ready to receive and send protocol parameter exchanges
lp_state_req=Active
lp_state_req=Active
SB MSG {LinkMgmt.RDI.Req.Active}
SB MSG {LinkMgmt.RDI.Rsp.Active}
SB MSG {LinkMgmt.RDI.Req.Active}
SB MSG {LinkMgmt.RDI.Rsp.Active}
pl_state_sts = Active
pl_state_sts = Active
Either side can start the SB exchange
Physical Layer returns pl_state_sts = Active once it has sent and received an "Active Status" SB MSG
Physical Layer returns pl_state_sts = Active once it has sent and received an "Active Status" SB MSG
Stage 2 Complete

10.1.7 RDI PM flow

This section defines the rules for PM entry, exit and abort flows as they apply to handshakes on the RDI. The rules for L1 and L2 are the same, except that exit from L2 is to Reset state, whereas exit from L1 is to Retrain state. This section uses PM to denote L1 or L2. A "PM Request" sideband message is {LinkMgmt.RDI.Req.L1} or {LinkMgmt.RDI.Req.L2}. A "PM Response" sideband message is {LinkMgmt.RDI.Rsp.L1} or {LinkMgmt.RDI.Rsp.L2}.

- Regardless of protocol, the PM entry or exit flow is symmetric on RDI. Both Physical Layer must issue PM entry request through a sideband message once the conditions of PM entry have been satisfied. PM entry is considered successful and complete once both sides have received a valid "PM Response" sideband message. Figure 10-8 shows an example flow for L1. Once the RDI status is PM, the Physical Layer can transition itself to a power savings state (turning off the PLL for example). Note that the sideband logic and corresponding PLL needs to stay on even during L1 state.
- All the Adapter state machines (Adapter LSMs) in the Adapter must have moved to the corresponding PM state before the Adapter requests PM entry from remote Link partner. Adapter LSM in PM implies the retry buffer of the Adapter must be empty, and it must not have any new Flits (or Ack/Nak) pending to be scheduled. Essentially there should be no traffic on mainband when PM entry is requested by the Adapter to the Physical Layer. The Adapter is permitted to clock gate its sideband logic once RDI status is PM and there are no outstanding transactions or responses on sideband. Physical Layer must do **`pl_clk_req`** handshake (if **`pl_clk_req`** is not already asserted or status is not Active) before forwarding sideband requests from the Link to the Adapter.
- Adapter requests PM entry by transitioning **`lp_state_req`** to the corresponding PM encoding. Once requested, the Adapter cannot change this request until it observes PM, Active.PMNAK, Retrain, or LinkError state on **`pl_state_sts`**. While requesting PM state, if the Adapter receives Active request from the Protocol Layer, or a PM exit request for the Adapter LSM on sideband, it must sink the message but delay processing it until **`pl_state_sts`** has resolved. Once the RDI state is resolved, the Adapter must first bring it back to Active before processing the other requests.
  — If the resolution is PM (upon successful PM entry) and the Protocol Layer needs to exit PM (or there is a pending Protocol Layer Active request from remote Link partner), then the Adapter must initiate PM exit flow on RDI by requesting **`lp_state_req`** = Active. All PM entry-related handshakes must have finished prior to this (this is when the Physical Layer on both sides of the Link have received a valid "PM Response" sideband message).

  — If the resolution is Active.PMNAK, the Adapter must initiate a request of Active on RDI. Once the status moves to Active, the Adapter is permitted to re-request PM entry (if all conditions of PM entry are still met). Figure 10-9 shows an example of PM abort flow. The PM request could have been from either side.

  — If the resolution is LinkError, then the Adapter must propagate this to Protocol Layers. This also resets any outstanding PM handshakes.
- Physical Layer initiates a "PM Request" sideband message once it samples the corresponding PM encoding on **`lp_state_req`** and has completed the StallReq/Ack handshake with its Adapter.
- Once a Physical Layer receives a "PM request" sideband message, it must respond to it within 2 us:
  — If its local Adapter is requesting the corresponding PM state, it must respond with the corresponding "PM Response" sideband message. If the current status is not PM, it must transition **`pl_state_sts`** to PM after responding to the sideband message.

  — If the current **`pl_state_sts`** = PM, it must respond with "PM Response" sideband message.

  — If **pl_state_sts** = Active and **lp_state_req** = Active and it remains this way for 1us after receiving the "PM Request" sideband message, it must respond with {LinkMgmt.RDI.Rsp.PMNAK} sideband message.

- If a Physical Layer receives a "PM Response" sideband message in response to a "PM Request" sideband message, it must transition **pl_state_sts** on its local RDI to PM (if it is currently in Active state). If the current state is not Active, no action needs to be taken.
- If a Physical Layer receives a {LinkMgmt.RDI.Rsp.PMNAK} sideband message in response to a "PM Request" sideband message, it must transition **pl_state_sts** on its local RDI to Active.PMNAK state if it is currently in Active state. If it is not in Active state, no action needs to be taken. The Physical Layer is permitted to retry PM entry handshake (if all conditions of PM entry are satisfied) at least 2 us after receiving the {LinkMgmt.RDI.Rsp.PMNAK} sideband message OR if it received a corresponding "PM Request" sideband message from the remote Link partner.
- PM exit is initiated by the Adapter requesting Active on RDI. This triggers the Physical Layer to initiate PM exit by sending a {LinkMgmt.RDI.Req.Active} sideband message. Physical Layer must make sure it has finished any Link retraining steps before it responds with the {LinkMgmt.RDI.Rsp.Active} sideband message. Figure 10-10 shows an example flow of PM exit on RDI.

  — PM exit handshake completion requires both Physical Layers to send as well as receive a {LinkMgmt.RDI.Rsp.Active} sideband message. Once this has completed, the Physical Layer is permitted to transition **pl_state_sts** to Active on RDI.

  — If **pl_state_sts** = PM and a {LinkMgmt.RDI.Req.Active} sideband message is received, the Physical Layer must initiate **pl_clk_req** handshake with the Adapter, and transition **pl_state_sts** to Retrain. This must trigger the Adapter to request Active on **lp_state_req** (if not already doing so), and this in turn triggers the Physical Layer to send {LinkMgmt.RDI.Req.Active} sideband message to the remote Link partner. Figure 10-11 shows an example of the L1 exit flow on RDI and its interaction with the LTSM in the Physical Layer. It is permitted for the LTSM to begin the Link PM exit and retraining flow when a {LinkMgmt.RDI.Req.Active} sideband message is received or when the Adapter requests Active on RDI. The timeout counters for the Active Request sideband message handshake must begin only after LTSM is in the LINKINIT state. L2 exit follows a similar flow for cases in which graceful exit is required without domain reset; however, the L2 exit is via Reset state on RDI, and not Retrain. Exit conditions from Reset state apply for L2 exit (i.e., a NOP -> Active transition is required on **lp_state_req** for the Physical Layer to exit Reset state on RDI).

Note that the following figures are examples for L1, and do not show the **lp_wake_req**, **pl_clk_req** handshakes. Implementations must follow the rules outlined for these handshakes in previous sections.

Figure 10-8. Successful PM entry flow

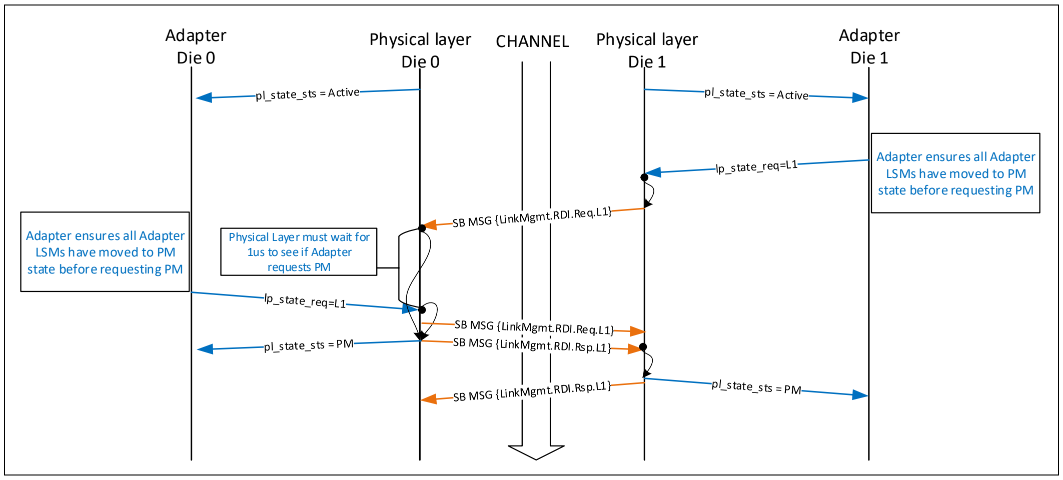

Adapter Die 0
Physical layer Die 0
CHANNEL
Physical layer Die 1
Adapter Die 1
pl_state_sts = Active
pl_state_sts = Active
Adapter ensures all Adapter LSMs have moved to PM state before requesting PM
lp_state_req=L1
SB MSG {LinkMgmt.RDI.Req.L1}
Adapter ensures all Adapter LSMs have moved to PM state before requesting PM
Physical Layer must wait for 1us to see if Adapter requests PM
lp_state_req=L1
SB MSG {LinkMgmt.RDI.Req.L1}
SB MSG {LinkMgmt.RDI.Rsp.L1}
pl_state_sts = PM
SB MSG {LinkMgmt.RDI.Rsp.L1}
pl_state_sts = PM

Figure 10-9. PM Abort flow

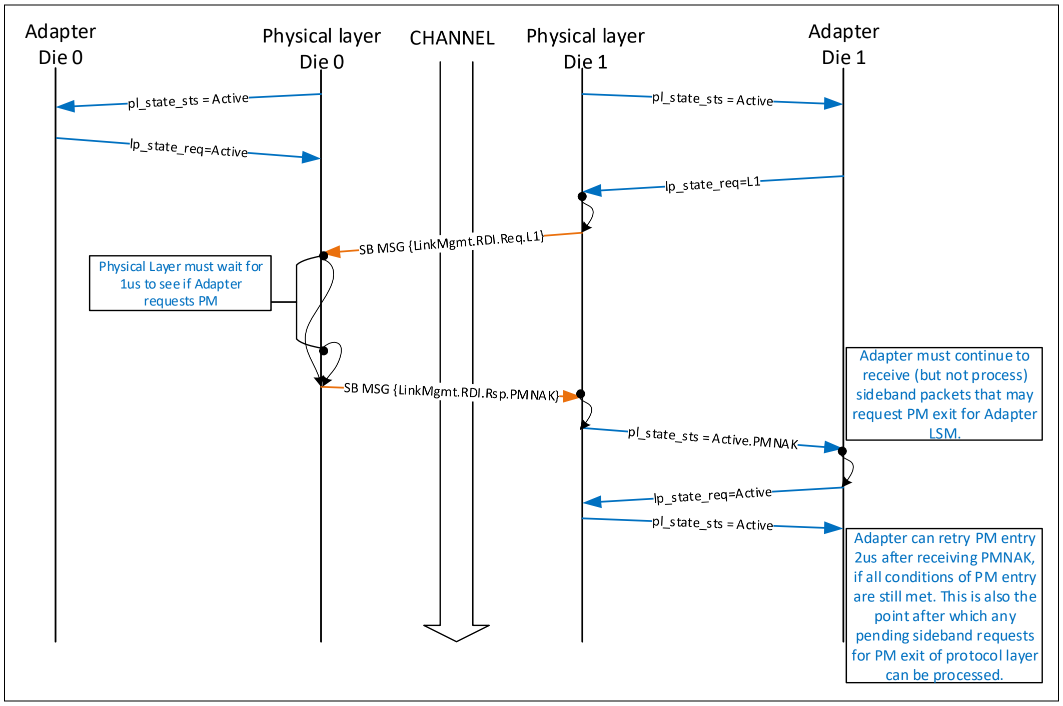

Adapter Die 0
Physical layer Die 0
CHANNEL
Physical layer Die 1
Adapter Die 1
pl_state_sts = Active
pl_state_sts = Active
lp_state_req=Active
lp_state_req=L1
SB MSG {LinkMgmt.RDI.Req.L1}
Physical Layer must wait for 1us to see if Adapter requests PM
Adapter must continue to receive (but not process) sideband packets that may request PM exit for Adapter LSM.
SB MSG {LinkMgmt.RDI.Rsp.PMNAK}
pl_state_sts = Active.PMNAK
lp_state_req=Active
pl_state_sts = Active
Adapter can retry PM entry 2us after receiving PMNAK, if all conditions of PM entry are still met. This is also the point after which any pending sideband requests for PM exit of protocol layer can be processed.

Figure 10-10. PM Exit flow

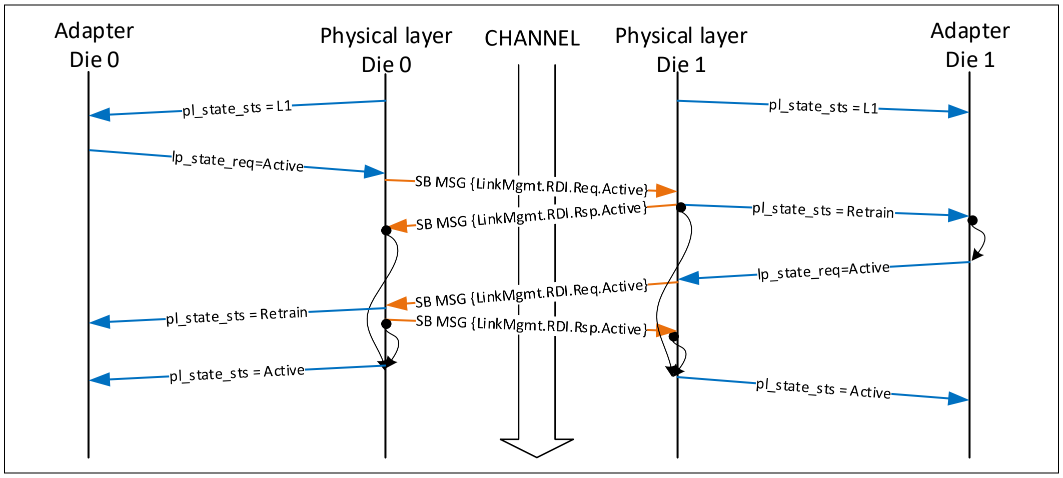
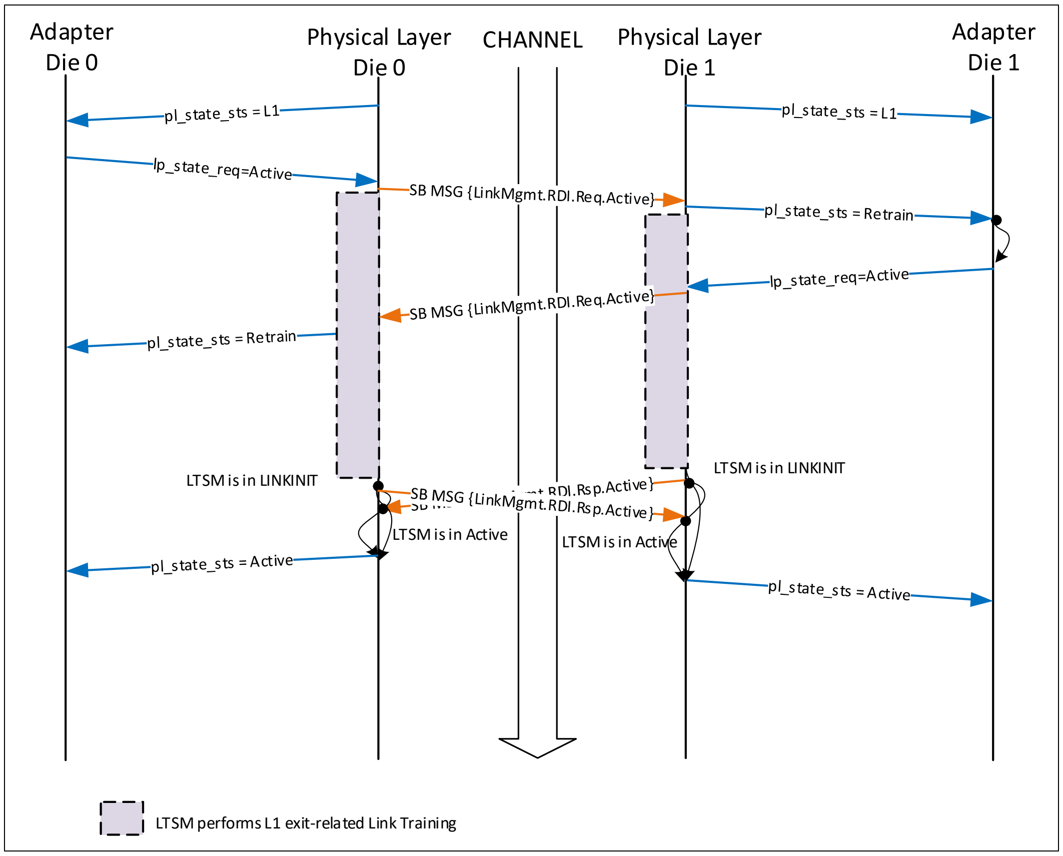


Figure 10-11. RDI PM Exit Example Showing Interactions with LTSM



10.2 Flit-Aware Die-to-Die Interface (FDI)

This section defines the signal descriptions and functionality associated with a single instance of Flit-Aware Die-to-Die Interface (FDI). A single instance is used for a Protocol Layer to Adapter connection. However, a single Adapter can host multiple protocol stacks using multiple instances of FDI. Figure 10-12 shows example configurations using multiple instances of FDI.

Figure 10-12. Example configurations using FDI

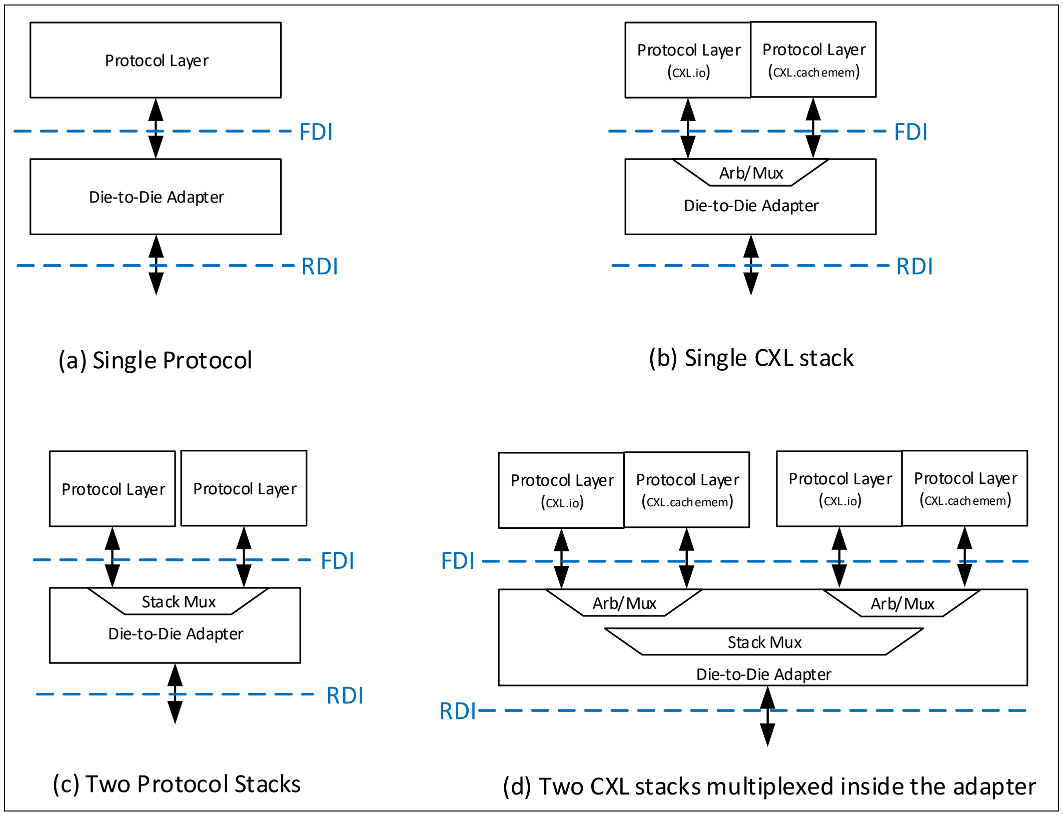


Table 10-3 lists the FDI signals and their descriptions. All signals are synchronous with **`lclk`**.

In Table 10-3:

- **`pl_*`** indicates that the signal is driven away from the Die-to-Die Adapter to the Protocol Layer.
- **`lp_*`** indicates that the signal is driven away from the Protocol Layer to the Die-to-Die Adapter.

Note: The same signal-naming convention as RDI is used to highlight that RDI signal list is a proper subset of FDI signal list.

Signal encodings pertaining to 'Management Transport protocol' are applicable only when Management Transport protocol was successfully negotiated on the mainband. Otherwise, those encodings are reserved. Also, **`dm_*`** signals in Table 10-3 are applicable only when supporting Management Transport path over the mainband ("dm" is an abbreviation for "d2d_adapter-to-management_port_gateway").

Table 10-3. FDI signal list (Sheet 1 of 8)

Signal Name	Signal Description
lclk	The clock at which FDI operates.
lp_irdy	Signal indicating that the Protocol Layer potentially has data to send. This must be asserted if lp_valid is asserted and the Protocol Layer wants the Adapter to sample the data. **lp_irdy** must not be presented by the Protocol Layer when **pl_state_sts** is Reset except when the status transitions from LinkError to Reset. On a LinkError to Reset transition, it is permitted for **lp_irdy** to be asserted for a few clocks but it must be de-asserted eventually. Physical Layer must ignore **lp_irdy** when status is Reset.
lp_valid	Protocol Layer to Adapter indication that data is valid on the corresponding **lp_data** bytes.
lp_data[NBYTES-1:0][7:0]	Protocol Layer to Adapter data, where 'NBYTES' equals number of bytes determined by the data width for the FDI instance.
lp_retimer_crd	When asserted at a rising clock edge, it indicates a single credit return for the Retimer Receiver buffer. Each credit corresponds to 256B of mainband data (including Flit header and CRC, etc.). This signal must NOT assert if a Retimer is not present. On FDI, this is an optional signal. It is permitted to have the Receiver buffers in the Protocol Layer for Raw Format only. If this is not exposed to Protocol Layer, Adapter must track credit at 256B granularity even for Raw Format and return credits to Physical Layer on RDI. When this is exposed on FDI, the Adapter must have the initial credits knowledge through other implementation specific means in order to advertise this to the remote Link partner during parameter exchanges.
lp_corrupt_crc	This signal is only applicable for CXL.cachemem in UCIe Flit Mode (i.e., the Adapter doing Retry) for CXL 256B Flit Mode. It is meant as a latency optimization that enables detection and containment for viral or poison using the Adapter to corrupt CRC of outgoing Flit. It is recommended to corrupt CRC by performing a bitwise XOR of the computed CRC with the syndrome 138Eh. The syndrome was computed such that no 1-bit or 2-bit errors alias to this syndrome, and it has the least probability of aliasing with 3-bit errors. For Standard 256B Flits, Protocol Layer asserts this along with **lp_valid** for the last chunk of the Flit that needs containment. Adapter corrupts CRC for both of the 128B halves of the Flit which had this set. It also must make sure to overwrite this flit (with the next flit sent by the Protocol Layer) in the Tx Retry buffer. For Latency-Optimized 256B Flits, Protocol Layer asserts this along with **lp_valid** for the last chunk of the 128B Flit half that needs containment. If **lp_corrupt_crc** is asserted on the first 128B half of the Flit, Protocol Layer must assert it on the second 128B half of the Flit as well. The very next Flit from the Protocol Layer after this signal has been asserted must carry the information relevant for viral, as defined in the CXL specification. If this was asserted on the second 128B half of the Flit only, it is the responsibility of the Protocol Layer to send the first 128B half exactly as before, and insert the viral information in the second half of the Flit. Adapter corrupts CRC for the 128B half of the Flit which had this set. It also must make sure to overwrite this flit (with the next flit sent by the Protocol Layer) in the Tx Retry buffer.
lp_dllp[NDLLP-1:0]	Protocol Layer to Adapter transfer of DLLP bytes. This is not used for 68B Flit Mode, CXL.cachemem or Streaming protocols. For a 64B data path on lp_data, it is recommended to assign NDLLP >= 8, so that 1 DLLP per Flit can be transferred from the Protocol Layer to the Adapter on average. The Adapter is responsible for inserting DLLP into DLP bytes 2:5 if the Flit packing rules permit it. See Section 10.2.4.1 for additional rules.
lp_dllp_valid	Indicates valid DLLP transfer on **lp_dllp**. DLLP transfers are not subject to backpressure by **pl_trdy** (the Adapter must have storage for different types of DLLP and this can be overwritten so that the latest DLLPs are sent to remote Link partner). DLLP transfers are subject to backpressure by **pl_stallreq** - Protocol Layer must stop DLLP transfers at DLLP Flit aligned boundary before giving **lp_stallack** or requesting PM.
lp_dllp_ofc	Indicates that the corresponding DLLP bytes on **lp_dllp** follow the Optimized_Update_FC format. It must stay asserted for the entire duration of the DLLP transfer on **lp_dllp**.

Table 10-3. FDI signal list (Sheet 2 of 8)

Signal Name	Signal Description
lp_stream[7:0]	Protocol Layer to Adapter signal that indicates the stream ID to use with data. Each stream ID maps to a unique protocol and stack. It is relevant only when **lp_valid** is 1. 00h: Reserved 01h: Stack 0: PCIe 02h: Stack 0: CXL.io 03h: Stack 0: CXL.cachemem 04h: Stack 0: Streaming protocol 05h: Stack 0: Management Transport protocol 11h: Stack 1: PCIe 12h: Stack 1: CXL.io 13h: Stack 1: CXL.cachemem 14h: Stack 1: Streaming protocol 15h: Stack 1: Management Transport protocol Other encodings are Reserved.
pl_trdy	The Adapter is ready to accept data. Data is accepted by the Adapter when **pl_trdy**, **lp_valid**, and **lp_irdy** are asserted at the rising edge of **lclk**. This signal must be asserted only if **pl_state_sts** is Active or when performing the **pl_stallreq/lp_stallack** handshake when the **pl_state_sts** is LinkError (see Section 10.3.3.7).
pl_valid	Adapter to Protocol Layer indication that data is valid on **pl_data**.
pl_data[NBYTES-1:0][7:0]	Adapter to Protocol Layer data, where NBYTES equals the number of bytes determined by the data width for the FDI instance.
pl_retimer_crd	When asserted at a rising clock edge, it indicates a single credit return from the Retimer. Each credit corresponds to 256B of mainband data (including Flit header and CRC, etc.). This signal must NOT assert if a Retimer is not present. On FDI, this is an optional signal. It is permitted to expose these credits to Protocol Layer for Raw Format only. If this is not exposed to Protocol Layer, Adapter must track credit at 256B granularity even for Raw Format and back-pressure the Protocol Layer using **pl_trdy**. When this is exposed on FDI, the Adapter converts the initial credits received from the Retimer over sideband to credit returns to the Protocol Layer on this bit after Adapter LSM has moved to Active state.
pl_dllp[NDLLP-1:0]	Adapter to Protocol Layer transfer of DLLP bytes. This is not used for 68B Flit mode, CXL.cachemem or Streaming protocols. For a 64B data path on **pl_data**, it is recommended to assign NDLLP >= 8, so that 1 DLLP per Flit can be transferred from the Adapter to the Protocol Layer, on average. The Adapter is responsible for extracting DLLP from DLP Bytes 2:5 if a Flit Marker is not present. The Adapter is also responsible for indicating Optimized_Update_FC format by setting **pl_dllp_ofc** = 1 for the corresponding transfer on FDI.
pl_dllp_valid	Indicates valid DLLP transfer on **pl_dllp**. DLLPs can be transferred to the Protocol Layer whenever valid Flits can be transferred on **pl_data**. There is no backpressure and the Protocol Layer must always sink DLLPs.
pl_dllp_ofc	Indicates that the corresponding DLLP bytes on **pl_dllp** follow the Optimized_Update_FC format. It must stay asserted for the entire duration of the DLLP transfer on **pl_dllp**.

Table 10-3. FDI signal list (Sheet 3 of 8)

Signal Name	Signal Description
`pl_stream[7:0]`	Adapter to Protocol Layer signal that indicates the stream ID to use with data. Each stream ID maps to a unique protocol. It is relevant only when **`pl_valid`** is 1. 00h: Reserved 01h: Stack 0: PCIe 02h: Stack 0: CXL.io 03h: Stack 0: CXL.cachemem 04h: Stack 0: Streaming protocol 05h: Stack 0: Management Transport protocol 11h: Stack 1: PCIe 12h: Stack 1: CXL.io 13h: Stack 1: CXL.cachemem 14h: Stack 1: Streaming protocol 15h: Stack 1: Management Transport protocol Other encodings are Reserved.
`pl_flit_cancel`	Adapter to Protocol Layer indication to dump a Flit. This enables latency optimizations on the Receiver data path when CRC checking is enabled in the Adapter. It is not applicable for Raw Format or 68B Flit Format. For Standard 256B Flit, it is required to have a fixed number of clock cycle delay between the last chunk of a Flit transfer and the assertion of **`pl_flit_cancel`**. This delay is fixed to be 1 cycle (i.e., the cycle after the last chunk transfer of a Flit). When this signal is asserted, Protocol Layer must not consume the associated Flit. For Latency-Optimized 256B Flits, it is required to have a fixed number of clock cycle delay between the last chunk of a 128B half Flit transfer and the assertion of **`pl_flit_cancel`**. This delay is fixed to be 1 cycle (i.e., the cycle after the last transfer of the corresponding 128B chunk). When this signal is asserted, Protocol Layer must not consume the associated Flit half. When this mode is supported, Protocol Layer must support it for all applicable Flit Formats associated with the corresponding protocol. Adapter must guarantee this to be a single cycle pulse when dumping a Flit or Flit half. It is the responsibility of the Adapter to ensure that the canceled Flits or Flit halves are eventually replayed on the interface without cancellation in the correct order once they pass CRC after Retry etc. See Section 10.2.5 for examples. When operating in UCIe Flit mode, it is permitted to use this signal to also cancel valid NOP Flits for the Protocol Layer to prevent forwarding these to the Protocol Layer. However for interoperability, if a Protocol Layer receives a NOP Flit without a corresponding **`pl_flit_cancel`**, it must discard these Flits.
`lp_state_req[3:0]`	Protocol Layer request to Adapter to request state change. Encodings as follows: 0000b: NOP 0001b: Active 0100b: L1 1000b: L2 1001b: LinkReset 1011b: Retrain 1100b: Disabled All other encodings are reserved.
`lp_linkerror`	Protocol Layer to Adapter indication that an error has occurred which requires the Link to go down. Adapter must propagate this request to RDI, and move the Adapter LSMs (and CXL vLSMs if applicable) to LinkError state once RDI is in LinkError state. It must stay there as long as **`lp_linkerror`**=1. The reason for having this be an indication decoupled from regular state transitions is to allow immediate action on part of the Protocol Layer and Adapter in order to provide the quickest path for error containment when applicable (for example, a viral error escalation could map to the LinkError state)

Table 10-3. FDI signal list (Sheet 4 of 8)

Signal Name	Signal Description
pl_state_sts[3:0]	Adapter to Protocol Layer Status indication of the Interface. Encodings as follows: 0000b: Reset 0001b: Active 0011b: Active.PMNAK 0100b: L1 1000b: L2 1001b: LinkReset 1010b: LinkError 1011b: Retrain 1100b: Disabled All other encodings are reserved. The status signal is permitted to transition from Adapter autonomously when applicable. For example the Adapter asserts the Retrain status when it decides to enter retraining either autonomously or when requested by remote agent. For PCIe/Streaming protocols, the Adapter LSM is exposed as **pl_state_sts** to the Protocol Layer. For CXL protocol, the ARB/MUX vLSM is exposed as **pl_state_sts** to the Protocol Layer. The Link Status is considered to be Up from Protocol Layer perspective when FDI status is Active, Active.PMNAK, Retrain, L1, or L2. The Link Status is considered Down for other states of FDI.
pl_inband_pres	Adapter to the Protocol Layer indication that the Die-to-Die Link has finished negotiation of parameters with remote Link partner and is ready for transitioning the FDI Link State Machine (LSM) to Active. Once it transitions to 1b, this must stay 1b until FDI moves to Active or LinkError. It stays asserted while FDI is in Retrain, Active, Active.PMNAK, L1, or L2. It must de-assert during LinkReset, Disabled or LinkError states.
pl_error	Adapter to the Protocol Layer indication that it has detected a framing related error. It is pipeline matched with the receive data path. It must also assert if **pl_error** was asserted on RDI by the Physical Layer for a Flit which the Adapter is forwarding to the Protocol Layer. In UCIe Flit Mode, it is permitted for Protocol Layer to use **pl_error** indication to log correctable errors when Retry is enabled from the Adapter. The Adapter must finish any partial Flits sent to the Protocol Layer and assert **pl_flit_cancel** in order to prevent consumption of that Flit by the Protocol Layer. Adapter must initiate Link Retrain on RDI following this, if it was a framing error detected by the Adapter. In UCIe Flit Mode, if Retry is disabled, the Adapter is responsible for mapping internally detected framing errors or Physical Layer received **pl_error** to an Uncorrectable Internal Error and escalate it as **pl_trainerror** if the mask and severity registers permit the escalation. If the Link is operating in Raw Format, the Adapter has no internal detection of framing errors, it just forwards any **pl_error** indication received from the Physical Layer on FDI such that it is pipeline matched to the data path. It is a pulse indication that can occur only when FDI receiver is Active (i.e. **pl_rx_active_req** = **lp_rx_active_sts** = 1).
pl_cerror	Adapter to the Protocol Layer indication that a correctable error was detected that does not affect the data path. The Protocol Layer must OR the **pl_error** and **pl_cerror** signals for Correctable Error Logging. Errors logged in the Correctable Error Status register are mapped to this signal if the corresponding mask bit in the Correctable Error Mask register is cleared to 0. It is a pulse of one or more cycles that can occur in any FDI state. If it is a state in which clock gating is permitted, it is the responsibility of the Adapter to perform the clock gating exit handshake with the Protocol Layer before asserting this signal. Clock gating can resume after **pl_cerror** is de-asserted and all other conditions permitting clock gating have been met.

Table 10-3. FDI signal list (Sheet 5 of 8)

Signal Name	Signal Description
`pl_nferror`	Adapter to the Protocol Layer indication that a non-fatal error was detected. This is used by Protocol Layer for error logging and corresponding escalation to software. The Adapter must OR any internally detected errors with **`pl_nferror`** on RDI and forward the result on FDI. Errors logged in Uncorrectable Error Status Register are mapped to this signal if the corresponding Severity and Mask bits are cleared to 0. It is a pulse of one or more cycles that can occur in any FDI state. If it is a state in which clock gating is permitted, it is the responsibility of the Adapter to perform the clock gating exit handshake with the Protocol Layer before asserting this signal. Clock gating can resume after **`pl_nferror`** is de-asserted and all other conditions permitting clock gating have been met.
`pl_trainerror`	Indicates a fatal error from the Adapter. Adapter must transition **`pl_state_sts`** to LinkError if not already in LinkError state. (Note that the Adapter first takes RDI to LinkError, and that LinkError is eventually propagated to all the FDI states). Implementations are permitted to map any fatal error to this signal that require upper layer escalation (or interrupt generation) depending on system level requirements. Errors logged in Uncorrectable Error Status Register are mapped to this signal if the corresponding Severity is set to 1 and the corresponding Mask bit is cleared to 0. It is a level signal that can assert in any FDI state but stays asserted until FDI exits the LinkError state to Reset state.
`pl_rx_active_req`	Adapter asserts this signal to request the Protocol Layer to open its Receiver's data path and get ready for receiving protocol data or Flits. The rising edge of this signal must be when **`pl_state_sts`** is Reset, Retrain or Active. Together with **`lp_rx_active_sts`**, it forms a four way handshake. See Section 10.2.7 for rules related to this handshake.
`lp_rx_active_sts`	Protocol Layer responds to **`pl_rx_active_req`** after it is ready to receive and parse protocol data or Flits. Together with **`pl_rx_active_req`**, it forms a four way handshake. See Section 10.2.7 for rules related to this handshake.
`pl_protocol[3:0]`	Adapter indication to Protocol Layer of the protocol that was negotiated during training. 0000b: PCIe without Management Transport 0011b: CXL.1 [Single protocol, i.e., CXL.io] without Management Transport 0100b: CXL.2 [Multi-protocol, Type 1 device] without Management Transport 0101b: CXL.3 [Multi-protocol, Type 2 device] without Management Transport 0110b: CXL.4 [Multi-protocol, Type 3 device] without Management Transport 0111b: Streaming protocol without Management Transport 1000b: PCIe with Management Transport 1001b: Management Transport 1011b: CXL.1 [Single protocol, i.e., CXL.io] with Management Transport 1100b: CXL.2 [Multi-protocol, Type 1 device] with Management Transport 1101b: CXL.3 [Multi-protocol, Type 2 device] with Management Transport 1110b: CXL.4 [Multi-protocol, Type 3 device] with Management Transport 1111b: Streaming protocol with Management Transport Other encodings are Reserved
`pl_protocol_flitfmt[3:0]`	This indicates the negotiated Format. See Chapter 3.0 for the definitions of these formats. 0001b: *Format 1*: Raw Format 0010b: *Format 2*: 68B Flit Format 0011b: *Format 3*: Standard 256B End Header Flit Format 0100b: *Format 4*: Standard 256B Start Header Flit Format 0101b: *Format 5*: Latency-Optimized 256B without Optional Bytes Flit Format 0110b: *Format 6*: Latency-Optimized 256B with Optional Bytes Flit Format Other encodings are Reserved

Table 10-3. FDI signal list (Sheet 6 of 8)

Signal Name	Signal Description
pl_protocol_vld	Indication that **pl_protocol**, and **pl_protocol_flitfmt** have valid information. This is a level signal, asserted when the Adapter has determined the appropriate protocol, but must only de-assert again after subsequent transitions to LinkError or Reset state depending on the Link state machine transitions. Protocol Layer must sample and store **pl_protocol** and **pl_protocol_flitfmt** when **pl_protocol_vld** = 1 and **pl_state_sts** = Reset and **pl_inband_pres** = 1. It must treat this saved value as the negotiated protocol until **pl_state_sts** = Reset and **pl_inband_pres** = 0. The Adapter must ensure that if **pl_inband_pres** = 1, **pl_protocol_vld** = 1 and **pl_state_sts** = Reset, then **pl_protocol** and **pl_protocol_flitfmt** are the correct values that can be sampled by the Protocol Layer.
pl_stallreq	Adapter request to Protocol Layer to flush all Flits for state transition and not prepare any new Flits. See Section 10.2.6 for details.
lp_stallack	Protocol Layer to Adapter indication that the Flits are aligned and stalled (if **pl_stallreq** was asserted). It is strongly recommended that this response logic be on a global free running clock, so the Protocol Layer can respond to **pl_stallreq** with **lp_stallack** even if other significant portions of the Protocol Layer are clock gated.
pl_phyinrecenter	Adapter indication to Protocol Layer that the Link is doing training or retraining (i.e., RDI has **pl_phyinrecenter** asserted or the Adapter LSM has not moved to Active yet). If this is asserted during a state where clock gating is permitted, the **pl_clk_req/ lp_clk_ack** handshake must be performed with the upper layer. The upper layers are permitted to use this to update the "Link Training/Retraining" bit in the UCIe Link Status register.
pl_phyinl1	Adapter indication to Protocol Layer that the Physical Layer is in L1 power management state (i.e., RDI is in L1 state).
pl_phyinl2	Adapter indication to Protocol Layer that the Physical Layer is in L2 power management state (i.e., RDI is in L2 state).
pl_speedmode[2:0]	Current Link speed. The following encodings are used: 000b: 4GT/s 001b: 8GT/s 010b: 12GT/s 011b: 16GT/s 100b: 24GT/s 101b: 32GT/s other encodings are reserved. The Protocol Layer must only consider this signal to be relevant when the FDI state is Active or Retrain. For multi-module configurations, all modules must operate at the same speed.
pl_lnk_cfg[2:0]	Current Link Configuration. Indicates the current operating width of a module. 000b: x4 001b: x8 010b: x16 011b: x32 100b: x64 101b: x128 110b: x256 other encodings are reserved. The Protocol Layer must only consider this signal to be relevant when the FDI state is Active or Retrain. This is the total width across all Active modules for the corresponding FDI instance.

Table 10-3. FDI signal list (Sheet 7 of 8)

Signal Name	Signal Description
pl_clk_req	Request from the Adapter to remove clock gating from the internal logic of the Protocol Layer. This is an asynchronous signal from the Protocol Layer's perspective since it is not tied to **lclk** being available in the Protocol Layer. Together with **lp_clk_ack**, it forms a four-way handshake to enable dynamic clock gating in the Protocol Layer. When dynamic clock gating is supported, the Protocol Layer must use this signal to exit clock gating before responding with **lp_clk_ack**. If dynamic clock gating is not supported, it is permitted for the Adapter to tie this signal to 1b.
lp_clk_ack	Response from the Protocol Layer to the Adapter acknowledging that its clocks have been ungated in response to **pl_clk_req**. This signal is only asserted when **pl_clk_req** is asserted, and de-asserted after **pl_clk_req** has de-asserted. When dynamic clock gating is not supported by the Protocol Layer, it must stage **pl_clk_req** internally for one or more clock cycles and turn it around as **lp_clk_ack**. This way it will still participate in the handshake even though it does not support dynamic clock gating.
lp_wake_req	Request from the Protocol Layer to remove clock gating from the internal logic of the Adapter. This is an asynchronous signal relative to **lclk** from the Adapter's perspective since it is not tied to **lclk** being available in the Adapter. Together with **pl_wake_ack**, it forms a four-way handshake to enable dynamic clock gating in the Adapter. When dynamic clock gating is supported, the Adapter must use this signal to exit clock gating before responding with **pl_wake_ack**. If dynamic clock gating is not supported, it is permitted for the Protocol Layer to tie this signal to 1b.
pl_wake_ack	Response from the Adapter to the Protocol Layer acknowledging that its clocks have been ungated in response to **lp_wake_req**. This signal is only asserted after **lp_wake_req** has asserted, and is de-asserted after **lp_wake_req** has de-asserted. When dynamic clock gating is not supported by the Adapter, it must stage **lp_wake_req** internally for one or more clock cycles and turn it around as **pl_wake_ack**. This way it will still participate in the handshake even though it does not support dynamic clock gating.
pl_cfg[NC-1:0]	This is the sideband interface from the Adapter to the Protocol Layer. See Chapter 7.0 for details. NC is the width of the interface. Supported values are 8, 16, and 32. Register accesses must be implemented by hardware to be atomic regardless of the width of the interface (i.e., all 32 bits of a register must be updated in the same cycle for a 32-bit register write, and similarly all 64 bits of a register must be updated in the same cycle for a 64-bit register write).
pl_cfg_vld	When asserted, indicates that **pl_cfg** has valid information that should be consumed by the Protocol Layer.
pl_cfg_crd	Credit return for sideband packets from the Adapter to the Protocol Layer for sideband packets. Each credit corresponds to 64 bits of header and 64 bits of data. Even transactions that do not carry data or carry 32 bits of data consume the same credit and the Receiver returns the credit once the corresponding transaction has been processed or de-allocated from its internal buffers. See Section 7.1.3.1 for additional flow control rules. A value of 1 sampled at a rising clock edge indicates a single credit return. Because the advertised credits are design parameters, the Protocol Layer transmitter updates the credit counters with initial credits on domain reset exit, and no initialization credits are returned over the interface. Credit returns must follow the same rules of clock gating exit handshakes as the sideband packets to ensure that no credit returns are dropped by the receiver of the credit returns.
lp_cfg[NC-1:0]	This is the sideband interface from Protocol Layer to the Adapter. See Chapter 7.0 for details. NC is the width of the interface. Supported values are 8, 16, and 32. Register accesses must be implemented by hardware to be atomic regardless of the width of the interface (i.e., all 32 bits of a register must be updated in the same cycle for a 32-bit register write, and similarly all 64 bits of a register must be updated in the same cycle for a 64-bit register write).
lp_cfg_vld	When asserted, indicates that **lp_cfg** has valid information that should be consumed by the Adapter.

Table 10-3. FDI signal list (Sheet 8 of 8)

Signal Name	Signal Description
`lp_cfg_crd`	Credit return for sideband packets from the Protocol Layer to the Adapter for sideband packets. Each credit corresponds to 64 bits of header and 64 bits of data. Even transactions that do not carry data or carry 32 bits of data consume the same credit and the Receiver returns the credit once the corresponding transaction has been processed or de-allocated from its internal buffers. See Section 7.1.3.1 for additional flow control rules. A value of 1 sampled at a rising clock edge indicates a single credit return. Because the advertised credits are design parameters, the Adapter transmitter updates the credit counters with initial credits on domain reset exit, and no initialization credits are returned over the interface. Credit returns must follow the same rules of clock gating exit handshakes as the sideband packets to ensure that no credit returns are dropped by the receiver of the credit returns.
`dm_param_exchange_done`	Signal resets to 0 on a Domain Reset. In single stack management transport implementations, this signal is asserted when adapter parameter exchange has completed between both sides and flit format/protocol have been finalized. It is reset whenever the link status=down. In multi-stack management transport implementations, this signal is asserted only when both stacks have completed their individual adapter parameter exchanges and protocol has been finalized (successfully or unsuccessfully) across both stacks. If at run time one of the active stacks enters link status=down condition, this signal de-asserts and asserts again only when the above condition is again met.
`dm_param_stack_count[N-1:0]`	Number of stacks that successfully negotiated Management Transport protocol. This field is sampled only when **`dm_param_exchange_done`** signal is asserted. If 68B Flit format was finalized, this field must be cleared to 00b. 00b: 0 stack 01b: 1 stack 10b: 2 stacks Others: reserved N=1 for single stack and 2 for 2 stacks.

10.2.1 Interface reset requirements

FDI does not define a separate interface signal for reset; however, it is required that the logic entities on both sides of FDI are in the same reset domain and the reset for each side is derived from the same source. Because reset may be staggered due to SoC routing, all signals coming out of reset must be driven to 0, unless otherwise specified. **`lp_stream`** and **`pl_stream`** are exceptions to this rule if they are tied off to their expected values at the time of integration. If **`lp_stream`** and **`pl_stream`** are not tied off, they must be driven to 0 when coming out of reset.

10.2.2 Interface clocking requirements

FDI requires both sides of the interface to be on the same clock domain. Moreover, the clock domain for sideband interface (**`*cfg*`**) is the same as the mainband signals.

Each side is permitted to instantiate clock crossing FIFOs internally if needed, as long as it does not violate the requirements at the interface itself.

It is important to note that there is no back pressure possible from the Protocol Layer to the Adapter on the main data path. So any clock crossing related logic internal to the Protocol Layer must take this into consideration.

10.2.3 Dynamic clock gating

Dynamic coarse clock gating is permitted in the Adapter and Protocol Layer when **`pl_state_sts`** is Reset, LinkReset, Disabled or PM states. This section defines the rules around entry and exit of clock gating. Note that clock gating is not permitted in LinkError states - it is expected that the UCIe usages will enable error handlers to make sure the Link is not stuck in a LinkError state, if the intent is to save power when a Link is in an error state.

10.2.3.1 Rules and description for lp_wake_req/pl_wake_ack handshake

Protocol Layer can request removal of clock gating of the Adapter by asserting **`lp_wake_req`** (asynchronous to **`lclk`** availability in the Adapter). All Adapter implementations must respond with a **`pl_wake_ack`** (synchronous to **`lclk`**). The extent of internal clock ungating when **`pl_wake_ack`** is asserted is implementation-specific, but lclk must be available by this time to enable FDI transitions from the Protocol Layers. The Wake Req/Ack is a full handshake and it must be used for state transition requests (on **`lp_state_req`** or **`lp_linkerror`**) when moving away from a state in which clock gating is permitted. It must also be used for sending packets on the sideband interface.

Rules for this handshake:

1. Protocol Layer asserts **`lp_wake_req`** to request ungating of clocks by the Adapter.
2. The Adapter asserts **`pl_wake_ack`** to indicate that clock gating has been removed. There must be at least one clock cycle bubble between **`lp_wake_req`** assertion and **`pl_wake_ack`** assertion.
3. **`lp_wake_req`** must de-assert before **`pl_wake_ack`** de-asserts. It is the responsibility of the Protocol Layer to control the specific scenario of de-assertion. As an example, when performing the handshake for a state request, it is permitted to keep **`lp_wake_req`** asserted until it observes the desired state status. Protocol Layer is also permitted to keep **`lp_wake_req`** asserted through states where clock gating is not permitted in the Adapter (i.e., Active, LinkError or Retrain).
4. **`lp_wake_req`** should not be the only consideration for Adapter to perform clock gating, it must take into account **`pl_state_sts`** and other internal or Link requirements before performing global and/or local clock gating.
5. When performing **`lp_wake_req/pl_wake_ack`** handshake for **`lp_state_req`** transitions or **`lp_linkerror`** transition, the Protocol Layer is permitted to not wait for **`pl_wake_ack`** before changing **`lp_state_req`** or **`lp_linkerror`**.
6. When performing **`lp_wake_req/pl_wake_ack`** handshake for **`lp_cfg`** transitions, Protocol Layer must wait for **`pl_wake_ack`** before changing **`lp_cfg`** or **`lp_cfg_vld`**. Because **`lp_cfg`** can have multiple transitions for a single packet transfer, it is necessary to make sure that the Adapter clocks are up before transfer begins.

10.2.3.2 Rules and description for pl_clk_req/lp_clk_ack handshake

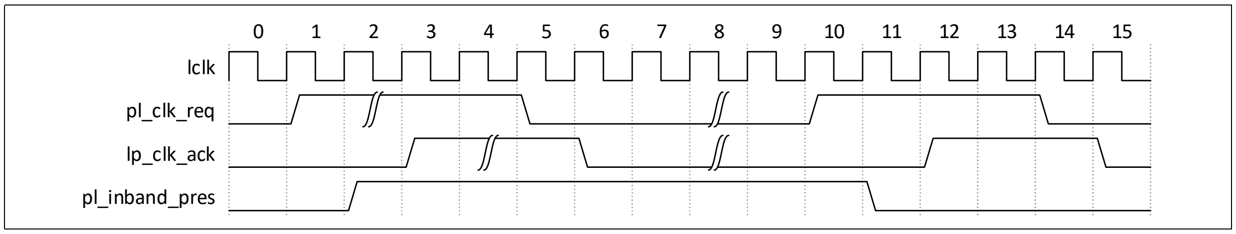
Adapter is allowed to initiate **`pl_clk_req/lp_clk_ack`** handshake at any time and the Protocol Layer must respond.

Rules for this handshake:

1. Adapter asserts **`pl_clk_req`** to request removal of clock gating by the Protocol Layer. This can be done anytime, and independent of current FDI state.
2. The Protocol Layer asserts **`lp_clk_ack`** to indicate that clock gating has been removed. There must be at least one clock cycle bubble between **`pl_clk_req`** assertion and **`lp_clk_ack`** assertion.
3. **`pl_clk_req`** must de-assert before **`lp_clk_ack`**. It is the responsibility of the Adapter to control the specific scenario of de-assertion, after the required actions for this handshake are completed.

4. **pl_clk_req** should not be the only consideration for the Protocol Layer to perform clock gating, it must take into account **pl_state_sts** and other protocol-specific requirements before performing trunk and/or local clock gating.
5. The Adapter must use this handshake to ensure transitions of **pl_inband_pres**, **pl_phyinl1**, **pl_phyinl2**, **pl_phyinrecenter**, and **pl_rx_active_req** have been observed by the Protocol Layer. Since these are level oriented signals, the Adapter is permitted to let the signal transition without waiting for **lp_clk_ack**. When this is done during initial Link bring up, it is strongly recommended for the Adapter to keep **pl_clk_req** asserted until the state status transitions away from Reset to a state where clock gating is not permitted or until the state status is Reset and **pl_inband_pres** de-asserts.

Figure 10-13. Example Waveform Showing Handling of Level Transition



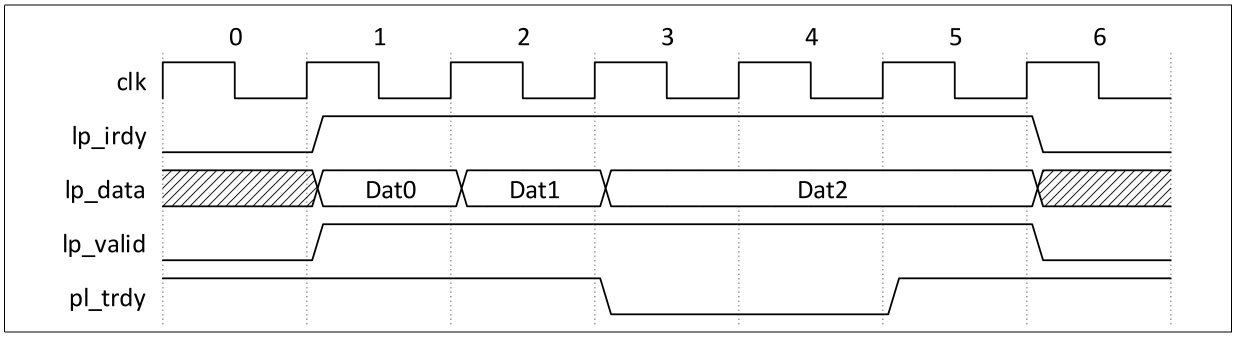
6. The Adapter must also perform this handshake before transition to LinkError state from Reset, LinkReset, Disabled or PM state (especially when the LinkError transition occurs by the Adapter without being directed by the Protocol Layer). It is permitted to assert **pl_clk_req** before the state change, in which case it must stay asserted until the state status transitions. It is also permitted to assert **pl_clk_req** after the state status transition, but in this case Adapter must wait for **lp_clk_ack** before performing another state transition.
7. The Adapter must also perform this handshake when the status is PM and remote Link partner is requesting PM exit. For exit from Reset, LinkReset, Disabled or PM states to a state that is not LinkError, it is required to assert **pl_clk_req** before the status change, and in this case it must stay asserted until the state status transitions away from Reset or PM.
8. The Adapter must also perform this handshake for sideband transfers from the Adapter to the Protocol Layer. When performing the handshake for **pl_cfg** transitions, Adapter must wait for **lp_clk_ack** before changing **pl_cfg** or **pl_cfg_vld**. Because **pl_cfg** can have multiple transitions for a single packet transfer, it is necessary to make sure that the Protocol Layer clocks are up before transfer begins.

When clock-gated in Reset states, Protocol Layers that rely on dynamic clock gating to save power must wait in clock gated state for **pl_inband_pres**=1. The Adapter will request clock gating exit when it transitions **pl_inband_pres**, and the Protocol Layer must wait for **pl_inband_pres** assertion before requesting **lp_state_req** = ACTIVE. If **pl_inband_pres** de-asserts while **pl_state_sts** = Reset, then the Protocol Layer is permitted to return to clock-gated state after moving **lp_state_req** to NOP.

10.2.4 Data Transfer

As indicated in the signal list descriptions, when Protocol Layer is sending data to the Adapter, data is transferred when **lp_irdy**, **pl_trdy** and **lp_valid** are asserted. Figure 10-14 shows an example waveform for data transfer from the Protocol Layer to the Adapter. Data is transmitted on clock cycles 1, 2, and 5. No assumption should be made by Protocol Layer about when **pl_trdy** can de-assert or for how many cycles it remains de-asserted before it is asserted again, unless explicitly guaranteed by the Adapter. If a Flit transfer takes multiple clock cycles, the Protocol Layer is not permitted to insert bubbles in the middle of a Flit transfer (i.e., **`lp_valid`** and **`lp_irdy`** must be asserted continuously until the Flit transfer is complete. Of course, data transfer can stall because of **`pl_trdy`** de-assertion).

Figure 10-14. Data Transfer from Protocol Layer to Adapter



As indicated in the signal list descriptions, when Adapter is sending data to the Protocol layer, there is no back-pressure mechanism, and data is transferred whenever **`pl_valid`** is asserted. The Adapter is permitted to insert bubbles in the middle of a Flit transfer and the Protocol Layer must be able to handle that.

10.2.4.1 DLLP transfer rules for 256B Flit Mode

For PCIe and CXL.io 256B Flits (both Standard and Latency-Optimized), FDI provides a separate signal for DLLP transfers from the Protocol Layer to the Adapter and vice-versa. Since the DLLPs have to bypass the Retry buffer, the separate signal enables the Adapter to insert DLLPs into the Flits from the Protocol Layer or the Retry buffer, if it is permitted to do so per the Flit packing rules of the corresponding Flit Format. Rules relevant for FDI operation (per FDI instance) are outlined below:

For the Transmitting side:

- Protocol Layer is responsible for sending the relevant DLLPs at the rate defined by the underlying Protocol to prevent timeouts of DLLP exchanges. If the Protocol Layer has no TLPs to send, it must insert NOP Flits to ensure that the Adapter gets an opportunity to insert the DLLP bytes.
- When transferring DLLP or Optimized_Update_FC, the least significant byte is sent over Byte 0 of the FDI bus, the next byte over Byte 1 and so on. When the transfer is over multiple chunks across FDI, Byte 0 is transferred on the first chunk LSB, Byte 1 following it and so on.
- The Adapter must have storage for at least 1 DLLP of every unique DLLP encoding (including Optimized_Update_FC) per supported VC that is possible for transfer to remote Link partner. The Adapter tracks pending DLLPs and schedules them on the next available opportunity for the relevant Flits. Credit update DLLPs must not be reordered for a VC by the Adapter. It is however permitted to discard a pending credit DLLP if the Protocol Layer presented a new credit DLLP of the same FC and VC. This extends to Optimized_Update_FC packets; i.e., it is permitted to discard any pending NP or P Update FC DLLPs, if the Protocol Layer transferred an Optimized_Update_FC for the corresponding VC.

On the Receiving side:

- The Adapter must extract DLLPs from received Flits of the corresponding protocol and forward them to the Protocol Layer. The FDI signal width of **`pl_dllp`** must be wide enough to keep up with the maximum rate of DLLPs that could be received from the Link.
- When transferring DLLP over multiple chunks across FDI, Byte 0 is transferred on the first chunk LSB, Byte 1 following it and so on.

- The Protocol Identifier corresponding to D2D Adapter in the Flit Header overlaps with the Flit usage of NOP Flits defined in PCIe and CXL specifications. The Adapter must check for available DLLPs in these Flits as well. All 0 bits in the DLLP byte positions indicate a NOP DLLP, and must not be forwarded to the Protocol Layer.

10.2.5 Examples of pl_flit_cancel Timing Relationships

In all the examples shown in this section, a 64B datapath on FDI is shown, and "F0Bytes" in the figures correspond to "Flit 0 Bytes".

Figure 10-15 shows an example timing relationship for **`pl_flit_cancel`** and **`pl_data`** for Latency-Optimized Flits when the first Flit half fails CRC check. Both Flit halves are canceled by the Adapter in this example by asserting **`pl_flit_cancel`** one clock after the last chunk transfer of the corresponding Flit half. It is permitted for the Adapter to de-assert **`pl_valid`** on clock cycles 5 and 6 instead of canceling that Flit half; however, this might have implications to meeting physical design timing margins in the Adapter. The use of **`pl_flit_cancel`** allows the Adapter to perform the CRC check on the side without putting the CRC logic in the critical timing path of the data flow and thus permitting higher frequency operation for implementations. In the example shown, after replay flow the entire Flit is transferred to the Protocol Layer without canceling as CRC checks pass.

Figure 10-15. Example for pl_flit_cancel for Latency-Optimized Flits and CRC Error on First Flit Half

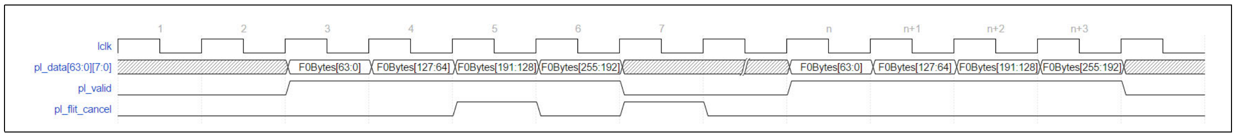


Figure 10-16 and Figure 10-17 show examples of two possible implementations of timing relationship for **`pl_flit_cancel`** and **`pl_data`** for Latency-Optimized Flits when the second Flit half fails CRC check. In both cases, the first half of the Flit is consumed by the Protocol Layer because it is not canceled by the Adapter (the data transferred on clock cycles 3 and 4).

In the first case (shown in Figure 10-16), after the replay flow, CRC passes, and the Adapter ensures that the Protocol Layer does not re-consume the first half again by asserting **`pl_flit_cancel`** for it. In this case, **`pl_valid`** asserts for the entire Flit, but only the second half is consumed because the first half was canceled on clock cycle (n+2).

In the second case (shown in Figure 10-17), after the replay flow, CRC passes, and the Adapter ensures that the Protocol Layer does not re-consume the first half again by not asserting **`pl_valid`** for it.

Figure 10-16. Example for pl_flit_cancel for Latency-Optimized Flits and CRC Error on Second Flit Half

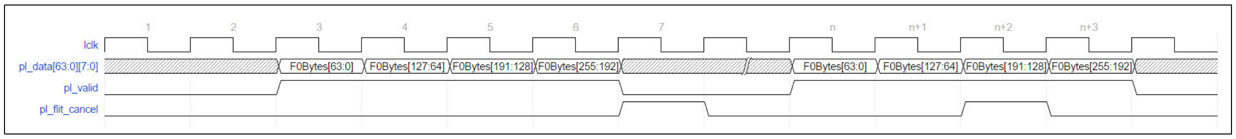


Figure 10-17. Example for pl_flit_cancel for Latency-Optimized Flits and CRC Error on Second Flit Half, Alternate Implementation Example

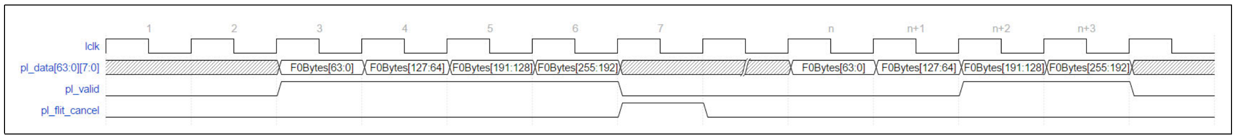
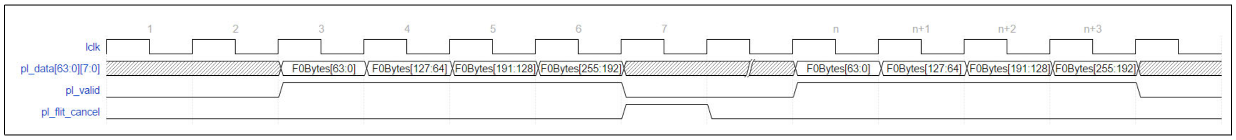


Figure 10-18 shows an example for a Standard 256B Flit. In this case, the CRC bytes are packed toward the end of the Flit and thus a CRC error on either of the two halves cancels the entire Flit. After replay flow, CRC passes, and the entire Flit is sent to the Protocol Layer without canceling it.

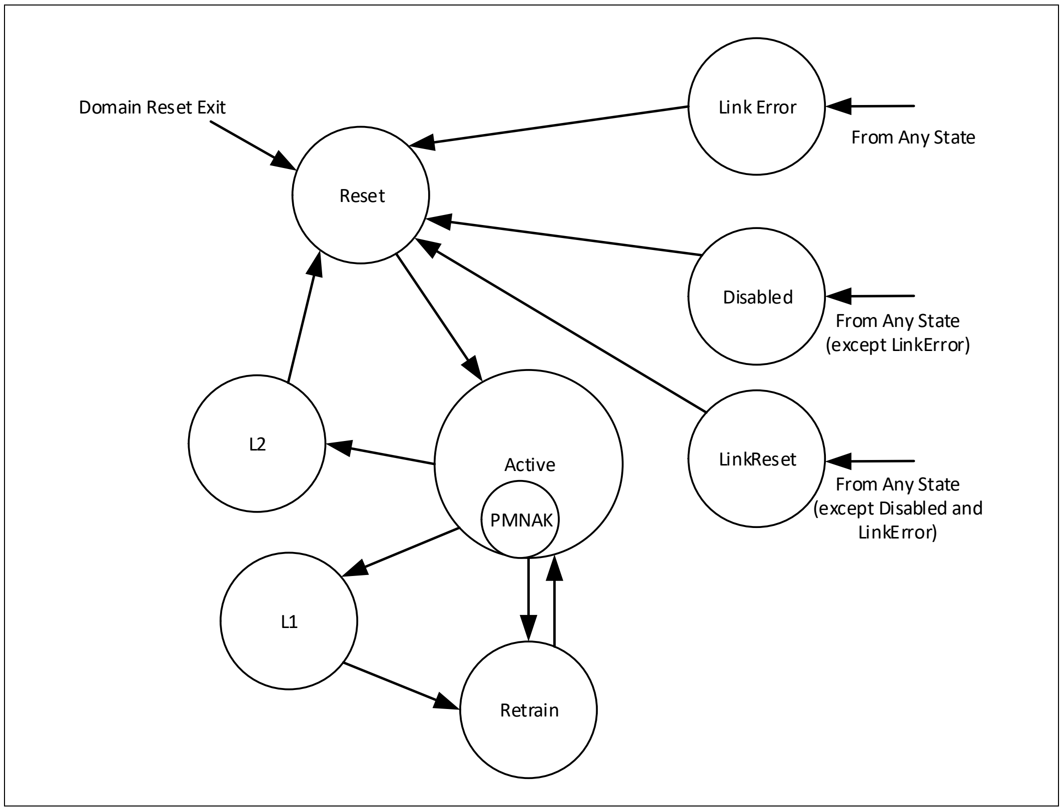
Figure 10-18. Example for pl_flit_cancel for Standard 256B Flits



10.2.6 FDI State Machine

Figure 10-19 shows the FDI state machine.

Figure 10-19. FDI State Machine



10.2.7 Rx_active_req/Sts Handshake

The Adapter negotiates Active state transitions on FDI using sideband messages when the Adapter LSM is exposed to the Protocol Layer. Since the sideband Link is running slower than the mainband Link, the Adapter needs to make sure that the Protocol Layer's Receiver is already in Active state (even though **`pl_state_sts`** might not have moved to Active yet) before responding to the Active request sideband message from remote Link partner. Rx_active_req/Sts handshake facilitates this.

When CXL is sent over UCIe, ARB/MUX functionality is performed by the Adapter and CXL vLSMs are exposed on FDI. Although ALMPs are transmitted over mainband, the interface to the Protocol Layer is FDI and it follows the rules of Rx_active_req/Sts Handshake as well.

Rules for this handshake:

1. The Adapter (or ARB/MUX) asserts **pl_rx_active_req** to trigger the Protocol Layer to open its Receiver's data path for receiving protocol data or Flits. This signal does not affect the Transmitter data path (it must wait for **pl_state_sts** to move to Active and follow the rules of **pl_trdy**). **pl_rx_active_req** should have a rising edge only when **lp_rx_active_sts** = 0 and **pl_state_sts** is Reset, Retrain or Active.
2. The Protocol Layer asserts **lp_rx_active_sts** after **pl_rx_active_req** has asserted and when it is ready to receive protocol data or Flits. There must be at least one clock cycle delay between **pl_rx_active_req** assertion and **lp_rx_active_sts** assertion to prevent a combinatorial loop.
3. When **pl_rx_active_req** = 1 and **lp_rx_active_sts** = 1, the Receiver is in Active state if **pl_state_sts** is Reset, Retrain, or Active.
4. **pl_rx_active_req** should have a falling edge only when **lp_rx_active_sts** = 1. This must trigger Protocol Layer to de-assert **lp_rx_active_sts**, and this completes the transition of the Receiver away from Active state.
5. For graceful exit from Active state (i.e., a transition to PM, Retrain,LinkReset or Disabled states), both **pl_rx_active_req** and **lp_rx_active_sts** must de-assert before **pl_state_sts** transitions away from Active.
6. If **pl_rx_active_req** = 0 while **pl_state_sts** = Active, the Adapter must guarantee no Flits would be sent to the Protocol Layer (for example, this can happen if the Adapter LSM or RDI is in Retrain, but the vLSM exposed to Protocol Layer is still in Active). Thus, it is permitted to perform this handshake even when the state status on FDI remains Active throughout.
7. For Active to LinkError transition, it is permitted for **pl_state_sts** to transition to LinkError before **pl_rx_active_req** de-asserts, but both **pl_rx_active_req** and **lp_rx_active_sts** must de-assert before **pl_state_sts** transitions away from LinkError.

10.2.8 FDI Bring up flow

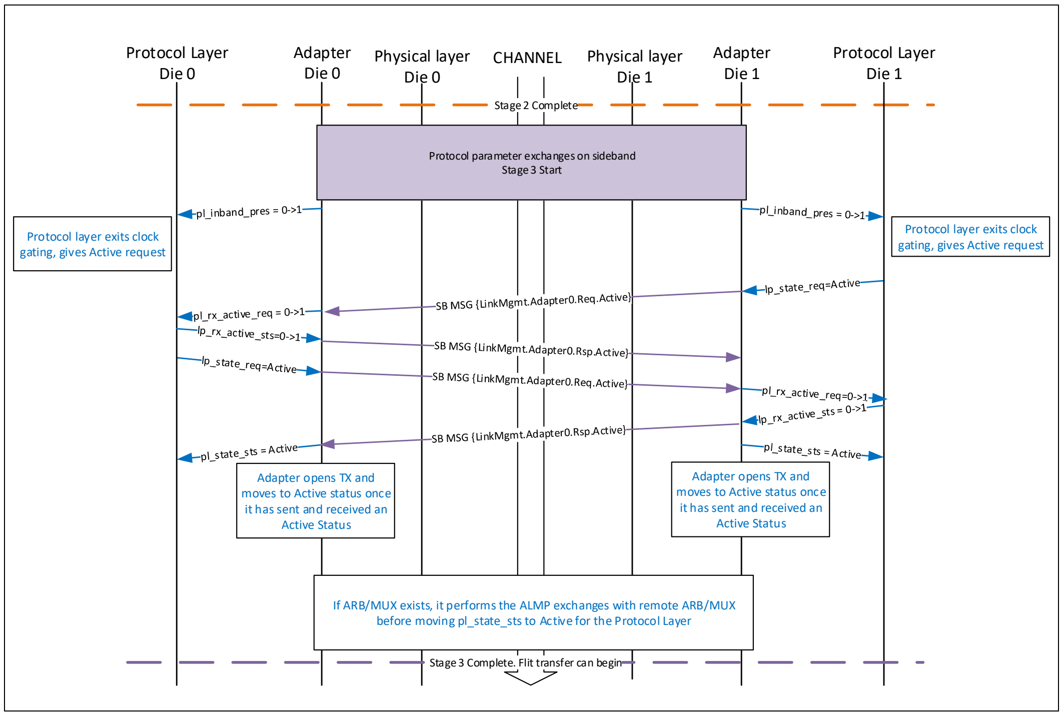
Figure 10-20 shows an example flow for Stage 3 of the Link bring up highlighting the transitions on FDI. This stage requires sequencing on FDI that coordinates the state transition from Reset to Active. If multiple stacks of protocol or ARB/MUX is present, the same sequence happens independently for each Protocol Layer stack. The flows on FDI are illustrated for Adapter 0 LSM in the sideband message encodings, however Adapter 1 LSM must send the sideband message encodings corresponding to Adapter 1 to its remote Link partner.

1. Once Adapter has completed transition to Active on RDI and successful parameter negotiation with the remote Link partner, it must do the **pl_clk_req** handshake with the Protocol Layer and reflect **pl_inband_pres**=1 on FDI. Note that the **pl_clk_req** handshake is not shown in the example flow in Figure 10-20
2. This is the trigger for Protocol Layer to request Active state. It is permitted for the Protocol Layer to wait unlit **pl_protocol_vld** = 1 before requesting Active. It must perform the **lp_wake_req** handshake as described in Section 10.2.3.1. Note that the **lp_wake_req** handshake is not shown in the example flow in Figure 10-20.
3. On sampling **lp_state_req** = Active, the Adapter must send the {LinkMgmt.Adapter0.Req.Active} sideband message to remote Link partner.

4. The Adapter must respond to the {LinkMgmt.Adapter.Req.Active} sideband message with a {LinkMgmt.Adapter.Rsp.Active} sideband message after making sure that the Protocol Layer's Receiver is ready. The {LinkMgmt.Adapter0.Rsp.Active} must only be sent after the Adapter has sampled **`pl_rx_active_req`** = **`lp_rx_active_sts`** = 1. As mentioned previously, the **`pl_clk_req`** handshake applies to **`pl_rx_active_req`** as well; it is permitted for the Adapter to keep **`pl_clk_req`** asserted continuously (once it has been asserted for **`pl_inband_pres`**) while doing the bring up flow. Note once the Adapter has sent the {LinkMgmt.Adapter0.Rsp.Active} sideband message, if it receives Flits from the remote Link partner, it must process them as applicable (i.e. for UCIe Flit mode, the Adapter must respond to the Sequence Number Handshake initiated by the remote Link or respond with Ack/Nak for Payload Flits. The Adapter will have to insert NOPs in case the **`pl_state_sts`** signal has not yet transitioned to Active).
5. If no ARB/MUX is present, once the Adapter has sent and received the {LinkMgmt.Adapter0.Rsp.Active} sideband message, it must transition **`pl_state_sts`** to Active for the Protocol Layer, and Flit transfer can begin (i.e., new Flits can be accepted from the Protocol Layer, and in UCIe Flit mode, the Adapter is permitted to initiate the Sequence Number Handshake Phase if it has not already done so as a result of Step 4).
6. If ARB/MUX is present, the sending and receipt of {LinkMgmt.Adapter0.Rsp.Active} sideband message opens up the ARB/MUX to perform ALMP exchanges over mainband and eventually transition the vLSMs to Active state.

Step 3 through Step 6 constitute the "Active Entry Handshake" on FDI and must be performed for every entry to Active state. Active.PMNAK to Active transition is not considered here because Active.PMNAK is only a sub-state of Active.

Figure 10-20. FDI Bring up flow


Protocol Layer Die 0
Adapter Die 0
Physical layer Die 0
CHANNEL
Physical layer Die 1
Adapter Die 1
Protocol Layer Die 1
Stage 2 Complete
Protocol parameter exchanges on sideband
Stage 3 Start
pl_inband_pres = 0->1
pl_inband_pres = 0->1
Protocol layer exits clock gating, gives Active request
Protocol layer exits clock gating, gives Active request
lp_state_req=Active
SB MSG {LinkMgmt.Adapter0.Req.Active}
pl_rx_active_req = 0->1
lp_rx_active_sts=0->1
SB MSG {LinkMgmt.Adapter0.Rsp.Active}
lp_state_req=Active
SB MSG {LinkMgmt.Adapter0.Req.Active}
pl_rx_active_req=0->1
lp_rx_active_sts = 0->1
SB MSG {LinkMgmt.Adapter0.Rsp.Active}
pl_state_sts = Active
pl_state_sts = Active
Adapter opens TX and moves to Active status once it has sent and received an Active Status
Adapter opens TX and moves to Active status once it has sent and received an Active Status
If ARB/MUX exists, it performs the ALMP exchanges with remote ARB/MUX before moving pl_state_sts to Active for the Protocol Layer
Stage 3 Complete. Flit transfer can begin

10.2.9 FDI PM Flow

This section describes the sequencing and rules for PM entry and exit on FDI. The rules are the same for L1 or L2 entry. L1 exit transitions the state machine through Retrain to Active, whereas L2 exit transitions the state machine through Reset to Active. The flow illustrations in the section use L1 as an example. A "PM request" sideband message is {LinkMgmt.Adapter*.Req.L1} or {LinkMgmt.Adapter*.Req.L2}. A "PM Response" sideband message is {LinkMgmt.Adapter*.Rsp.L1} or {LinkMgmt.Adapter*.Rsp.L2}. The flows on FDI are illustrated for Adapter 0 LSM in the sideband message encodings; however, Adapter 1 LSM must send the sideband message encodings corresponding to Adapter 1 to its remote Link partner.

- The Protocol Layer requests PM entry on FDI after idle time criteria has been met. The criteria for idle time is implementation specific and could be dependent on the protocol. For PCIe and CXL.io protocols, PM DLLPs are **not** used to negotiate PM entry/exit when using D2D Adapter's Retry buffer (i.e., UCIe Flit mode).
- If operating in UCIe Flit mode, ARB/MUX is present within the D2D Adapter, and it follows the rules of *CXL Specification* (for 256B Flit mode) to take the vLSMs to the corresponding PM state. Note that even for CXL 68B Flit mode, the same ALMP rules as 256B Flit mode are used. This is a simplification on UCIe, because ALMPs always go through the retry buffer. Once vLSMs are in the PM state, ARB/MUX requests the Adapter Link State Machine to enter the corresponding PM state. The Adapter Link State Machine transition to PM follows the same rules as outlined for Protocol Layer and Adapter below.
- If CXL or PCIe protocol has been negotiated, only the upstream port (UP) can initiate PM entry. This is done using a sideband message from the UP Adapter to the downstream port (DP) Adapter. DP Adapter must not initiate entry into PM. PM support is required for CXL and PCIe protocols. PM entry is considered successful and complete once UP receives a valid "PM Response" sideband message. Figure 10-21 shows an example flow for CXL or PCIe protocol PM Entry on FDI and Adapter. Once the FDI status is PM for all Protocol Layers, the Adapter can request PM transition on RDI.
- If Streaming protocol has been negotiated, OR UCIe is in Raw Format, OR Management Transport protocol was negotiated over the mainband without CXL or PCIe, then both side Adapters must issue a PM entry request through a sideband message once the conditions of PM entry have been satisfied. PM entry is considered successful and complete once both sides have received a valid "PM Response" sideband message. Figure 10-22 shows an example flow for symmetric protocols. Once the FDI status is PM for all Protocol Layers, the Adapter can request PM transition on the RDI.
- Protocol Layer requests PM entry once it has blocked transmission of any new Protocol Layer Flits, by transitioning **`lp_state_req`** to L1 or L2 encoding. Once requested, the Protocol Layer cannot change this request until it observes the corresponding PM state, Retrain, Active.PMNAK or LinkError state on **`pl_state_sts`**; unless it is a DP Protocol Layer for PCIe or CXL. Once the FDI state is resolved, the Adapter must first bring it back to Active before processing any new PM requests from the Protocol Layer.
  - If the resolution is PM (upon successful PM entry) and the Protocol Layer needs to exit PM (or there is a pending Protocol Layer Active request from remote Link partner) then the Protocol Layer must initiate PM exit flow on FDI by requesting **`lp_state_req`** = Active. All PM entry related handshakes must have finished prior to this (for CXL/PCIe protocols, this is when UP has received a valid "PM Response" sideband message. For symmetric protocols, this is when both sides Adapter have received a valid "PM Response" sideband message).
  - If the resolution is Active.PMNAK, the Protocol Layer must initiate a request of Active on FDI. Once the status moves to Active, the Protocol Layer is permitted to re-request PM entry (if all conditions of PM entry are still met). Figure 10-23 shows an example of PM abort flow. The PM

request could have been from either side depending on the configuration. Protocol Layer must continue receiving protocol data or Flits while the status is Active or Active.PMNAK.

- — DP Protocol Layer for PCIe or CXL is permitted to change request from PM to Active without waiting for PM or Active.PMNAK (the DP FDI will never have **`pl_state_sts`**=Active.PMNAK since it does not send "PM Request" sideband messages); however, it is still possible for the Adapter to initiate a stallreq/ack and complete PM entry if it was in the process of committing to PM entry when the Protocol Layer changed its request. In this scenario, the Protocol Layer will see **`pl_state_sts`** transition to PM and it is permitted to continue asking for the new state request.
- — If the resolution is LinkError, then the Link is down and it resets any outstanding PM handshakes.

- Adapter (UP port only if CXL or PCIe protocol), initiates a "PM request" sideband message once it samples a PM request on **`lp_state_req`** and has completed the StallReq/Ack handshake with the corresponding Protocol Layer and its Retry buffer is empty of Flits from the Protocol Layer that is requesting PM (all pending Acks have been received).
- If the Adapter LSM moves to Retrain while waiting for a "PM Response" sideband message, it must wait for the response. Once the response is received, it must transition back to Active before requesting a new PM entry. Note that the transition to Active requires Active Entry handshake with the remote Link partner, and that will cause the remote partner to exit PM. If the Adapter LSM receives a "PM Request" sideband message after it has transitioned to Retrain, it must immediately respond with {LinkMgmt.Adapter0.Rsp.PMNAK}.

Note: The precise timing of the remote Link partner that is observing Link Retrain is unknown; thus, the safer thing to do is to go to Active and redo the PM handshake when necessary for this scenario. There is a small probability that there might be an exit from PM and re-entry back in PM under certain scenarios.

- Once the Adapter receives a "PM request" sideband message, it must respond to it within 2 us (the time is only counted during the Adapter LSM being in Active state):
  - — if its local Protocol Layer is requesting PM, it must respond with the corresponding "PM Response" sideband message after finishing the StallReq/Ack handshake with its Protocol Layer and its Retry buffer being empty. If the current status is not PM, it must transition **`pl_state_sts`** to PM after responding to the sideband message.
  - — If the current **`pl_state_sts`** = PM, it must respond with "PM Response" sideband message.
  - — If **`pl_state_sts`** = Active and **`lp_state_req`** = Active and it remains this way for 1us after receiving the "PM Request" sideband message, it must respond with {LinkMgmt.Adapter0.Rsp.PMNAK} sideband message. The time is only counted during all the relevant state machines being in Active state.
- If the Adapter receives a "PM Response" sideband message in response to a "PM Request" sideband message, it must transition **`pl_state_sts`** on its local FDI to PM (if it is currently in Active state).
- If the Adapter receives a {LinkMgmt.Adapter0.Rsp.PMNAK} sideband message in response to a "PM Request" sideband message, it must transition **`pl_state_sts`** on its local FDI to Active.PMNAK state if it is currently in Active state. If it is not in Active state, no action needs to be taken. It is permitted to retry PM entry handshake (if all conditions of PM entry are satisfied) at least 2us after receiving the {LinkMgmt.Adapter0.Rsp.PMNAK} sideband message OR if it received a corresponding "PM Request" sideband message from the remote Link partner.
- PM exit is initiated by the Protocol Layer requesting Active on FDI. After RDI is in Active, triggers the Adapter to initiate PM exit by performing the Active Entry handshakes on sideband. Figure 10-24 shows an example flow of PM exit on FDI when Adapter LSM is exposed.

— PM exit handshake completion requires both Adapters to send as well as receive a {LinkMgmt.Adapter0.Rsp.Active} sideband message. Once this has completed, the Adapter is permitted to transition Adapter LSM to Active.

— If **`pl_state_sts`** = PM and a {LinkMgmt.Adapter0.Req.Active} sideband message is received, the Adapter must initiate **`pl_clk_req`** handshake with the Protocol Layer, and transition Adapter LSM to Retrain (For L2 exit, the transition is to Reset). This must trigger the Protocol Layer to request Active on **`lp_state_req`** (if not already doing so), and this in turn triggers the Adapter to send {LinkMgmt.Adapter0.Req.Active} sideband message to the remote Link partner.

Note that the following figures are examples and do not show the **`lp_wake_req`**, **`pl_clk_req`**, and/or **`pl_rx_active_req`** handshakes. Implementations must follow the rules outlined for these handshakes in previous sections.

Figure 10-21. PM Entry example for CXL or PCIe protocols

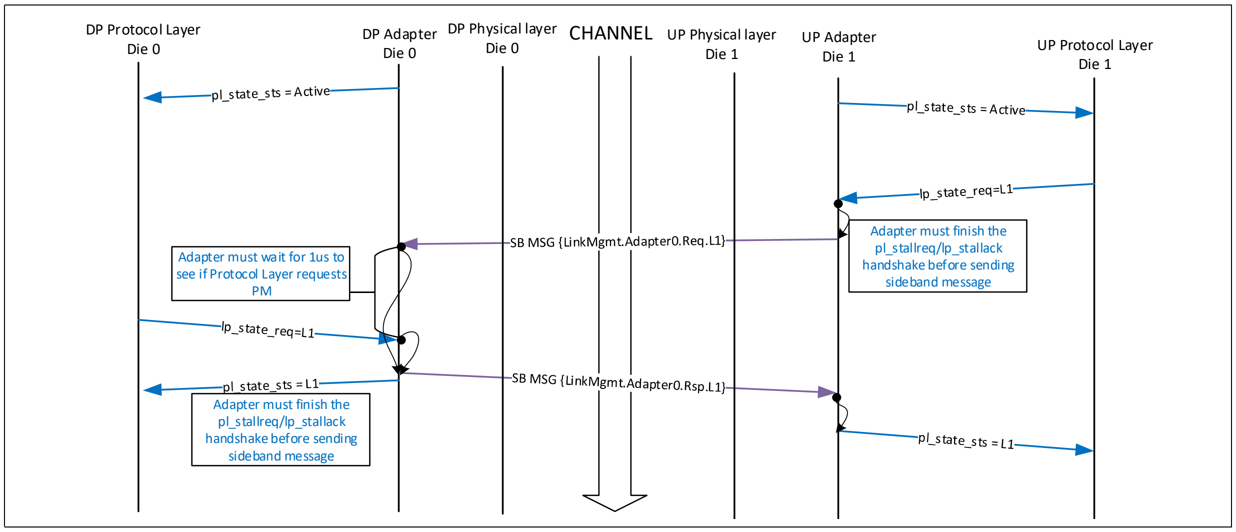

DP Protocol Layer Die 0
DP Adapter Die 0
DP Physical layer Die 0
CHANNEL
UP Physical layer Die 1
UP Adapter Die 1
UP Protocol Layer Die 1
pl_state_sts = Active
pl_state_sts = Active
lp_state_req=L1
SB MSG {LinkMgmt.Adapter0.Req.L1}
Adapter must finish the pl_stallreq/lp_stallack handshake before sending sideband message
Adapter must wait for 1us to see if Protocol Layer requests PM
lp_state_req=L1
pl_state_sts = L1
SB MSG {LinkMgmt.Adapter0.Rsp.L1}
Adapter must finish the pl_stallreq/lp_stallack handshake before sending sideband message
pl_state_sts = L1

Figure 10-22. PM Entry example for symmetric protocol

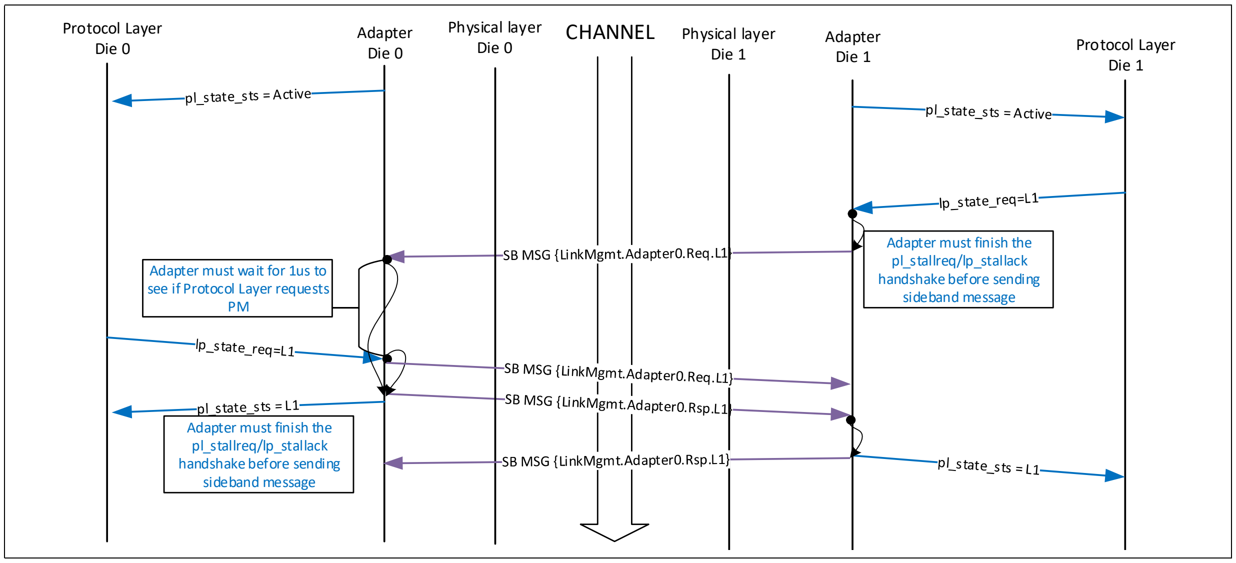

Protocol Layer Die 0
Adapter Die 0
Physical layer Die 0
CHANNEL
Physical layer Die 1
Adapter Die 1
Protocol Layer Die 1
pl_state_sts = Active
pl_state_sts = Active
lp_state_req=L1
SB MSG {LinkMgmt.Adapter0.Req.L1}
Adapter must finish the pl_stallreq/lp_stallack handshake before sending sideband message
Adapter must wait for 1us to see if Protocol Layer requests PM
lp_state_req=L1
SB MSG {LinkMgmt.Adapter0.Req.L1}
SB MSG {LinkMgmt.Adapter0.Rsp.L1}
pl_state_sts = L1
Adapter must finish the pl_stallreq/lp_stallack handshake before sending sideband message
SB MSG {LinkMgmt.Adapter0.Rsp.L1}
pl_state_sts = L1

Figure 10-23. PM Abort Example

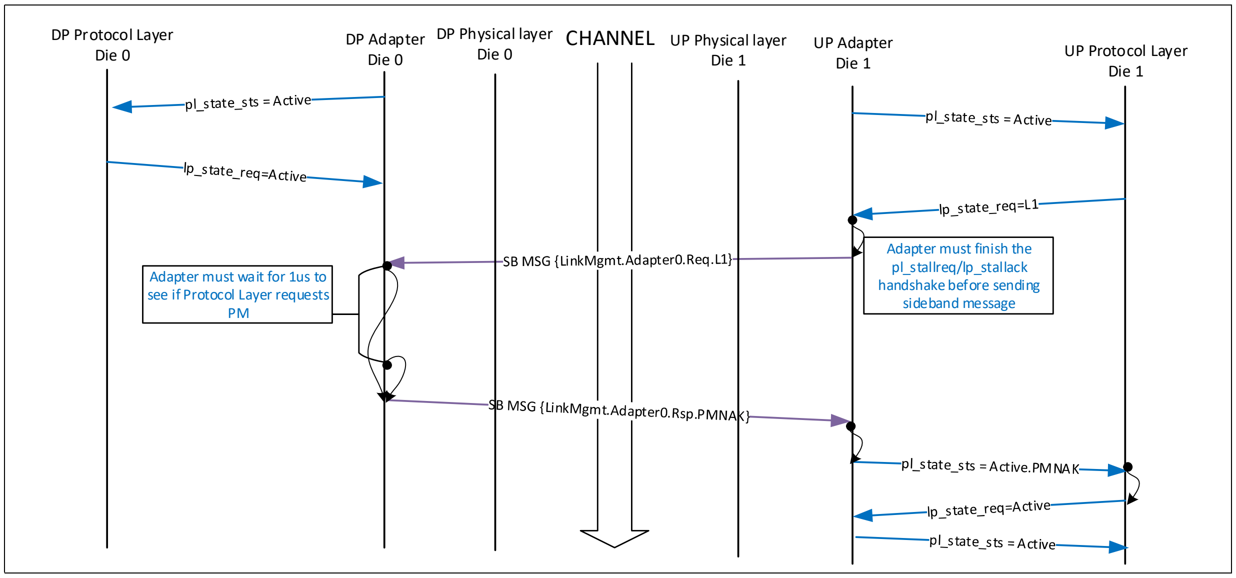
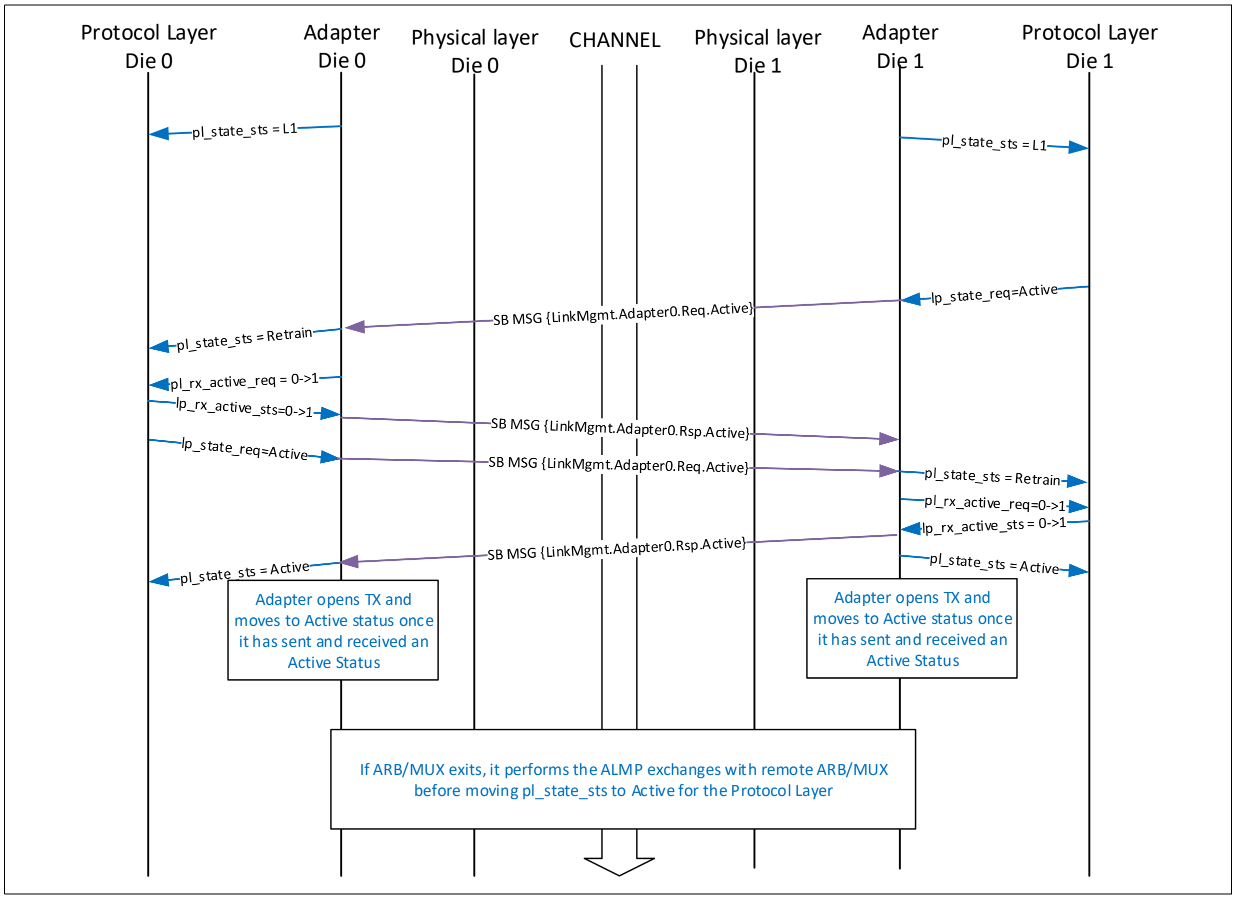

DP Protocol Layer Die 0
DP Adapter Die 0
DP Physical layer Die 0
CHANNEL
UP Physical layer Die 1
UP Adapter Die 1
UP Protocol Layer Die 1
pl_state_sts = Active
lp_state_req=Active
pl_state_sts = Active
lp_state_req=L1
Adapter must finish the pl_stallreq/lp_stallack handshake before sending sideband message
SB MSG {LinkMgmt.Adapter0.Req.L1}
Adapter must wait for 1us to see if Protocol Layer requests PM
SB MSG {LinkMgmt.Adapter0.Rsp.PMNAK}
pl_state_sts = Active.PMNAK
lp_state_req=Active
pl_state_sts = Active

Figure 10-24. PM Exit Example


Protocol Layer Die 0
Adapter Die 0
Physical layer Die 0
CHANNEL
Physical layer Die 1
Adapter Die 1
Protocol Layer Die 1
pl_state_sts = L1
pl_state_sts = L1
lp_state_req=Active
SB MSG {LinkMgmt.Adapter0.Req.Active}
pl_state_sts = Retrain
pl_rx_active_req = 0->1
lp_rx_active_sts=0->1
SB MSG {LinkMgmt.Adapter0.Rsp.Active}
lp_state_req=Active
SB MSG {LinkMgmt.Adapter0.Req.Active}
pl_state_sts = Retrain
pl_rx_active_req=0->1
lp_rx_active_sts = 0->1
SB MSG {LinkMgmt.Adapter0.Rsp.Active}
pl_state_sts = Active
pl_state_sts = Active
Adapter opens TX and moves to Active status once it has sent and received an Active Status
Adapter opens TX and moves to Active status once it has sent and received an Active Status
If ARB/MUX exits, it performs the ALMP exchanges with remote ARB/MUX before moving pl_state_sts to Active for the Protocol Layer

10.3 Common rules for FDI and RDI

This section covers common set of rules applicable to FDI and RDI and cross-interactions between them. Any applicable differences are called out as well. To have common terminology for the common set of rules, Upper Layer is used to refer to Adapter for RDI, and Protocol Layer for FDI. Lower Layer is used to refer to Physical Layer for RDI and Adapter for FDI.

Because Active.PMNAK is a sub-state of Active, all rules that apply for Active are also applicable for Active.PMNAK; however the state status cannot move from Active.PMNAK directly to L1 or L2 due to the rules requiring the Upper Layer to request a transition to Active before requesting PM again.

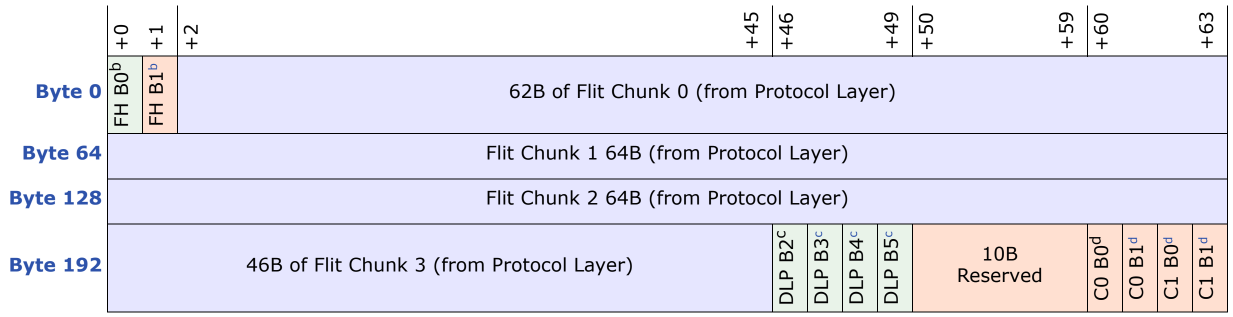
10.3.1 Byte Mapping for FDI and RDI

The Flit Format figures in Chapter 3.0 show examples of how a Flit is laid out on a 64B datapath when sent over FDI or RDI. Figure 10-25 shows an example of a CXL.io Standard 256B Start Header Flit for reference. Each Flit takes four data transfers across FDI or RDI when the data width is 64 Bytes. Each data transfer is referred to a Flit Chunk, numbered in increasing order within an entire Flit transfer.

For every data transfer, the Least Significant Byte from the corresponding Flit Chunk is mapped to Byte 0 on FDI (or RDI), the next Byte from the Flit is mapped to Byte 1 on FDI (or RDI), and so on. Within each Byte, bit 0 of the Byte from the Flit maps to bit 0 of the corresponding Byte on FDI (or RDI), and so on. The same mapping applies for both transmit and receive directions.

For example, in Transfer 0, Byte 0 of the Flit is mapped to Byte 0 of FDI (or RDI), Byte 1 of the Flit is mapped to Byte 1, and so on. In transfer 1, Byte 64 of the Flit is mapped to Byte 0 of FDI (or RDI), Byte 65 of the Flit is mapped to Byte 1 of FDI (or RDI) and so on. This example is illustrated in Figure 10-26. Data transfers follow the rules outlined in Section 10.1.4 for RDI and Section 10.2.4 for FDI and hence do not necessarily correspond to consecutive clock cycles.

Figure 10-25. CXL.io Standard 256B Start Header Flit Format Example[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 10-26. FDI (or RDI) Byte Mapping for 64B Datapath to 256B Flits

Transfer (Rows)	FDI (or RDI) Bytes (Columns)							
	0	1	2	...	60	61	62	63
0	Flit Byte 0	Flit Byte 1	Flit Byte 2	...	Flit Byte 60	Flit Byte 61	Flit Byte 62	Flit Byte 63
1	Flit Byte 64	Flit Byte 65	Flit Byte 66	...	Flit Byte 124	Flit Byte 125	Flit Byte 126	Flit Byte 127
2	Flit Byte 128	Flit Byte 129	Flit Byte 130	...	Flit Byte 188	Flit Byte 189	Flit Byte 190	Flit Byte 191
3	Flit Byte 192	Flit Byte 193	Flit Byte 194	...	Flit Byte 252	Flit Byte 253	Flit Byte 254	Flit Byte 255

If the FDI or RDI datapath width is increased (or decreased), the Byte mapping follows the same convention of increasing order of Flit bytes mapped to increasing order of FDI (or RDI) bytes. Figure 10-27 shows an illustration of a 128B data path.

Figure 10-27. FDI (or RDI) Byte Mapping for 128B Datapath to 256B Flits

Transfer (Rows)	FDI (or RDI) Bytes (Columns)											
	0	1	2	...	62	63	64	65	...	125	126	127
0	Flit Byte 0	Flit Byte 1	Flit Byte 2	...	Flit Byte 62	Flit Byte 63	Flit Byte 64	Flit Byte 65	...	Flit Byte 125	Flit Byte 126	Flit Byte 127
1	Flit Byte 128	Flit Byte 129	Flit Byte 130	...	Flit Byte 190	Flit Byte 191	Flit Byte 192	Flit Byte 193	...	Flit Byte 253	Flit Byte 254	Flit Byte 255

For 68B Flit Formats, the Protocol Layer transfers only 64B of payload information from the Flit over FDI (the Flit Header and CRC are inserted by the Adapter). Thus, if the datapath is 128B wide, two such transfers will happen at a given clock cycle as shown in Figure 10-28. The numbering in the figure still uses the Byte positions relative to the overall Flit, hence Byte 0 corresponds to Flit 0 Byte 2, etc. On the Transmit path, the Protocol Layer inserts empty slots (i.e., bytes with a value of 00h) to populate the entire width of the bus if the interface width is greater than 64B and there is insufficient payload information to transmit. The Adapter does the same on the Receive path.

Figure 10-28. FDI Byte Mapping for 128B Datapath for 68B Flit Format

Transfer (Rows)	FDI Bytes (Columns)											
	0	1	2	...	62	63	64	65	...	125	126	127
0	Flit 0 Byte 2	Flit 0 Byte 3	Flit 0 Byte 4	...	Flit 0 Byte 64	Flit 0 Byte 65	Flit 1 Byte 2	Flit 1 Byte 3	...	Flit 1 Byte 63	Flit 1 Byte 64	Flit 1 Byte 65

For 68B Flit Formats, Adapter inserts the Flit Header and CRC bytes, and performs the necessary shifting before transferring the bytes over RDI. Thus, if the data path is 128B wide, the byte mapping will follow as shown in Figure 10-29. The remainder of Flit 1 continues on the next transfer, etc. Given that the Adapter must insert PDS bytes before pausing the data stream, which makes the transfer a multiple of 256B, the transfer naturally aligns when the width of RDI is 64B, 128B, or 256B on both the Transmit and Receive directions. For wider than 256B interfaces, see the Implementation Note below.

Figure 10-29. RDI Byte Mapping for 128B Datapath for 68B Flit Format

Transfer (Rows)	RDI Bytes (Columns)											
	0	1	2	...	65	66	67	68	...	125	126	127
0	Flit 0 Byte 0	Flit 0 Byte 1	Flit 0 Byte 2	...	Flit 0 Byte 65	Flit 0 Byte 66	Flit 0 Byte 67	Flit 1 Byte 0	...	Flit 1 Byte 57	Flit 1 Byte 58	Flit 1 Byte 59
1	Flit 1 Byte 60	Flit 1 Byte 61	Flit 1 Byte 62	...	...	...	...	...	...	...	...	...

The frequency of operation of the interfaces along with the data width determines the maximum bandwidth that can be sustained across the FDI (or RDI) interface. For example, a 64B datapath at 2 GHz of clock frequency is required to sustain a 16 GT/s Link for an Advanced Package configuration with a single module. Similarly, to scale to 32 GT/s of Link speed operation for Advanced Package configuration with a single module, a 128B datapath running at 2 GHz would be required to support the maximum Link bandwidth.

The FDI (or RDI) byte mapping for the transmit or receive direction does not change for multi-module configurations. The MMPL logic within the Physical Layer is responsible for ensuring that the bytes are transmitted in the correct order to the correct module. Any byte swizzling or rearrangement to resolve module naming conventions, etc., is thus the responsibility of the MMPL logic.

IMPLEMENTATION NOTE — NBYTES

For Raw Format, the value of NBYTES is vendor-defined. This Implementation Note is for UCIe Flit mode.

It is strongly recommended that when operating in UCIe Flit mode, NBYTES is chosen to be one of 64, 128, 256, or 512 and is selected to get the best KPI (e.g., latency, area, etc.) for the desired bandwidth from the UCIe Link. If NBYTES is chosen to be larger than or equal to 512, it is strongly recommended that it is a multiple of 256 and is only done for the case of a four module Advanced Package Link designed for 16 GT/s or higher. Data transfer over the Link for all Flit formats defined in UCIe Flit mode are in a granularity of 256B, so aligning to a multiple of that avoids unnecessary shifting and corresponding tracking.

For situations in which the RDI or FDI data path is wider than 256B, the following considerations apply for interoperability:

- On the Transmit side, it is required to send valid data corresponding to the full width of the interface. For FDI, this would mean the Protocol Layer might need to pack a Protocol Flit with empty slots. For RDI, this would mean the Adapter might need to insert NOP Flits (for 68B Flit Format, PDS bytes are also included as valid data for this purpose).
- On the Receive side, for RDI:

  It is possible that the Physical Layer has to wait to accumulate sufficient bytes before transmitting over RDI. The Physical Layer must accumulate data in multiples of 256B and if the accumulated data is less than the RDI width, it must wait for a sufficient gap in valid data transfer on the Physical Link (at least 16 UI for differential clock and 32 UI for quadrature clock) before transmitting this data on RDI. In this scenario, the accumulated data is sent on the lower significant bytes of the RDI, and any remaining bytes on the interface are assigned to all 0s.

  For 256B Flit Formats, a Flit Header which is 0000h with a CRC of 0000h is silently discarded by the Adapter. It is also not included for the purposes of Runtime Link Testing.

  For 68B Flit Formats, the Adapter is expected to keep track of the PDS bytes (because these are included in Runtime Link Testing). Any extra padding beyond that is silently discarded and not included for the purposes of Runtime Link Testing.
- On the Receive side, for FDI:

  The Adapter must accumulate data in multiples of 256B before forwarding to the Protocol Layer. If the accumulated data is less than the FDI width, it gets sent on the lower significant bytes of the FDI, and any remaining bytes on the interface are assigned to 0b.

  For 256B Flit Formats, a Flit Header of 0000h is a NOP for the Protocol Layer and is discarded. For 68B Flit Formats, 00h are IDLE symbols for PCIe/CXL.io or Empty slots for CXL.cachemem, both of which get discarded by the Protocol Layer. For Streaming protocols that use 68B Flit Formats, it is recommended to use the same approach.
- **`lp_corrupt_crc`**, **`pl_flit_cancel`**, and **`pl_error`** apply to all the Flits that are transferred at the corresponding clock cycle. If applicable, it is recommended to set NDLLP to 32 for these applications and limit the DLLP throughput to be 1 per clock cycle on FDI.

10.3.2 Stallreq/Ack Mechanism

The Stallreq/Ack mechanism is used by the Lower Layer to interrupt the Flit transfers by the Upper Layer at a Flit boundary. On RDI, the Stallreq/Ack mechanism must be used when exiting Active state to Retrain, PM, LinkReset or Disabled states. On FDI, for UCIe Raw Format, the Stallreq/Ack mechanism must be used when exiting Active state to Retrain, PM, LinkReset or Disabled states. On FDI, for UCIe Flit Mode, the Stallreq/Ack mechanism must only be used when exiting Active State to a PM state. For other scenarios that exit Active state for UCIe Flit mode, the Adapter must simply de-assert **`pl_trdy`** at a Flit boundary before state transition.

The Stallreq/Ack mechanism is mandatory for all FDI and RDI implementations. **`lp_stallack`** assertion implies that Upper Layer has stalled its pipeline at a Flit aligned boundary.

The **`pl_stallreq/lp_stallack`** handshake is a four-phase sequence that follows the rules below:

1. The **`pl_stallreq`** and **`lp_stallack`** must be de-asserted before domain reset exit.
2. A rising edge on **`pl_stallreq`** must only occur when **`lp_stallack`** is de-asserted.
3. A falling edge on **`pl_stallreq`** must only occur when **`lp_stallack`** is asserted or when the domain is in reset.
4. A rising edge on **`lp_stallack`** must only occur when **`pl_stallreq`** is asserted.
5. A falling edge on **`lp_stallack`** must only occur when **`pl_stallreq`** is de-asserted or when domain is in reset.
6. When **`lp_stallack`** is asserted **`lp_valid`** and **`lp_irdy`** must both be de-asserted.
7. While **`pl_stallreq`** is asserted, any data presented on the interface must be accepted by the physical layer until the rising edge of **`lp_stallack`**. **`pl_trdy`** is not required to be asserted consecutively.
8. The logic path between **`pl_stallreq`** and **`lp_stallack`** must contain at least one flip-flop to prevent a combinatorial loop.
9. A complete stallreq/stallack handshake is defined as the completion of all four phases: Rising edge on **`pl_stallreq`**, rising edge on **`lp_stallack`**, falling edge on **`pl_stallreq`**, falling edge on **`lp_stallack`**.
10. It is strongly recommended that Upper Layer implements providing **`lp_stallack`** on a global free running clock so that it can finish the handshake even if the rest of its logic is clock gated.

To avoid performance penalties, it is recommended that this handshake be completed as quickly as possible while satisfying the above rules.

IMPLEMENTATION NOTE

In multiple places within this specification, for state transitions, it is referring to completing the Stallreq/Ack handshake before the state transition. In the context of state transitions, there are two acceptable ways to implement this from the lower layer:

- One implementation from the lower layer would follow the sequence:
    - i. Assert **`pl_stallreq`**.
    - ii. After **`lp_stallack`** is asserted, perform the necessary actions for state transition (including deassertion of **`pl_trdy`**).
    - iii. De-assert **`pl_stallreq`**. Once **`lp_stallack`** de-asserts, the state transition is considered complete.
- The alternate implementation from the lower layer would follow the sequence:
    - i. Assert **`pl_stallreq`**.
    - ii. After **`lp_stallack`** is asserted, de-assert **`pl_trdy`**.
    - iii. De-assert **`pl_stallreq`** and perform the necessary actions for state transition.

State transition is considered complete after **`pl_state_sts`** update and **`lp_stallack`** de-assertion.

10.3.3 State Request and Status

Table 10-4 describes the Requests considered by the Lower layer in each of the interface states. The Upper layer must take into account the interface state status and make the necessary request modifications.

The requests are listed on the Row and the state status is listed in the Column.

The entries in Table 10-4 denote the following:

- Yes: Indicates that the request is considered for next state transition by the lower layer.
- N/A: Not Applicable
- Ignore: Indicates that the request is ignored and has no effect on the next state transition.

Table 10-4. Requests Considered in Each State by Lower Layer

Request (Row) Versus Status (Column)	Reset	Active	L1	LinkReset	Retrain	Disable	L2	LinkError
NOP	Yes	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore
Active	Yes[a]	Ignore[b]	Yes	Yes	Yes	Yes	Yes	Yes
L1	Ignore	Yes	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore
LinkReset	Yes[a]	Yes	Yes	Ignore	Yes	Ignore	Yes	Ignore
Retrain	Ignore	Yes	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore
Disable	Yes[a]	Yes	Yes	Yes	Yes	Ignore	Yes	Ignore
L2	Ignore	Yes	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore
LinkError (sideband wire)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

a. Requires request transition from NOP
b. If the Status is Active.PMNAK, then the Lower Layer transitions to Active upon sampling the Active Request.

10.3.3.1 Reset State rules

The Reset State can be entered on de-assertion of interface reset signal or from LinkReset/Disable/LinkError/L2 states. In Reset state, the physical layer is permitted to begin its initialization/training process.

The **`pl_state_sts`** is not permitted to exit Reset state until requested by the upper layer. The exit from Reset state is requested by the upper layer by changing the **`lp_state_req`** signal from NOP encoding value to the permitted next state encoding value.

The rules for Reset state transition are as follows:

1. Reset→Active: The lower layer triggers transitions to the Active state upon observing **`lp_state_req`** == Reset (NOP) for at least one clock while **`pl_state_sts`** is indicating Reset followed by observing **`lp_state_req`** == Active. The transition to Active is only completed once the corresponding Active Entry handshakes have completed on the Link. For RDI, it is when the Physical Layer has sent and received an Active Response sideband message to and from the remote Physical Layer respectively. For the Adapter LSM, it is when the Adapter has sent and received an Active Status sideband message to and from the remote Adapter respectively. For the ARB/MUX vLSM, it is when the ARB/MUX has sent and received an Active Status ALMP to and from the remote ARB/MUX respectively.
2. Reset→LinkReset: The lower layer transitions to the LinkReset state upon observing **`lp_state_req`** == Reset (NOP) for at least one clock while **`pl_state_sts`** is indicating Reset followed by observing **`lp_state_req`** == LinkReset OR when requested by remote Link partner through the relevant sideband message. The lower layer is permitted to transition through Active State, and when it does, Active state exit conditions apply.
3. Reset→Disabled: The lower layer transitions to the Disabled state upon observing **`lp_state_req`** == Reset (NOP) for at least one clock while **`pl_state_sts`** is indicating Reset followed by observing **`lp_state_req`** == Disabled OR when requested by the remote Link partner through the relevant sideband message. The lower layer is permitted to transition through Active State, and when it does, Active state exit conditions apply.
4. Reset→LinkError: The lower layer transitions to LinkError based on observing an internal request to move to the LinkError or **`lp_linkerror`** assertion. For RDI, this transition is permitted if requested by the remote Link partner through the relevant sideband message.

10.3.3.2 Active State rules

The Active state to next state transitions are described below.

The rules for Active State transition are as follows:

1. Active→Retrain: The Lower layer transitions to the Retrain state upon observing **`lp_state_req`** == Retrain or due to an internal request to retrain the Link while **`pl_state_sts`** == Active. This arc is not applicable for CXL vLSMs exposed on FDI (CXL Flit Mode with Retry in the Adapter).
2. Active→L1: The physical layer transitions to L1 based on observing **`lp_state_req`** == L1 while in the Active state, if other conditions of PM entry have also been satisfied.
3. Active→L2: The physical layer transitions to L2 based on observing **`lp_state_req`** == L2 while in the Active state, if other conditions of PM entry have also been satisfied.

It is permitted to have an Active.PMNAK to Retrain/LinkReset/Disable/LinkError transition for cases where Lower Layer is waiting for the Upper Layer to change the request to Active and the corresponding Link event triggers it. There is no scenario where there is a transition from Active.PMNAK to L1 or L2.

Section 10.3.3.8 describes the transition from Active or Active.PMNAK to LinkReset, Disable, or LinkError states.

10.3.3.3 PM Entry/Exit Rules

See the PM entry and exit sequences in the RDI and FDI sections.

10.3.3.4 Retrain State Rules

Adapter requests Retrain on RDI if any of the following events occur:

- Software writes to the Retrain bit and the Link is in Active state.
- Number of CRC or parity errors detected crosses a threshold. The specific algorithm for determining this is implementation specific.
- Protocol Layer requests Retrain (only applicable for UCIe Raw Format).
- any other implementation specific condition (if applicable).

Physical Layer triggers a Retrain transition on RDI if:

- Valid framing errors are observed
- Remote Physical Layer requests Retrain entry
- Adapter requests Retrain

Protocol Layer must not request Retrain on FDI, unless UCIe is operating in UCIe Raw Format.

A Retrain transition on RDI must always be propagated to Adapter LSMs that are in Active. Retrain transitions of the UCIe Link are not propagated to CXL vLSMs. Upon Retrain entry, the credit counter for UCIe Retimer (if present) must be reset to the value advertised during initial Link bring up (the value is given by the "Retimer_Credits" Parameter in the {AdvCap.Adapter} sideband message during initial Link bring up). The Retimer must drain or dump any Flits in flight or its internal transport buffers upon entry to Retrain. Additionally, the Retimer must trigger Retrain of the remote UCIe Link (across the Off-Package Interconnect).

Entry into Retrain state resets power management state for the corresponding state machine, and power management entry if required must be re-initiated after the interface enters Active state. If there was an outstanding PM request that returns PM Response, the corresponding state machine must perform Active Entry handshakes to bring that state machine back to Active.

The rules for Retrain state transition are as follows:

1. Retrain→Active: If Retrain was entered from L1, the lower layer begins Active Entry handshakes upon observing **`lp_state_req`** == Active while the **`pl_state_sts`** == Retrain. If Retrain was entered from Active, the lower layer begins Active Entry handshakes only after observing a NOP-> Active transition on **`lp_state_req`**. Lower layer transitions to Active once the corresponding Active Entry handshakes have completed. Exit from Retrain on RDI requires the Active Entry handshakes to have completed between Physical Layers. Exit from Retrain on FDI must ensure that RDI has moved back to Active, and Active Entry handshakes have successfully completed between Adapters (for the Adapter LSM).
2. Transitional state: The lower layer is permitted to transition to the Active state upon observing **`lp_state_req`** == LinkReset or Disabled while the **`pl_state_sts`** == Retrain. Following the entry into Active the lower layer is permitted to make a transition to the requested state.

Section 10.3.3.8 describes Retrain exit to LinkReset, Disable, or LinkError states.

Note: The requirement to wait for NOP->Active transition ensures that the Upper Layer has a way to delay Active transition in case it is waiting for any relevant sideband handshakes to complete (for example the Parity Feature handshake).

10.3.3.5 LinkReset State Rules

LinkReset is used for reset flows (HotReset equivalent in PCIe, Protocol Layer must use this to propagate SBR to the device as well) to convey device and/or Link Reset across the UCIe Link.

Adapter triggers LinkReset transition upon observing a LinkReset request from the Protocol Layer, OR on receiving a sideband message requesting LinkReset entry from the remote Link partner OR an implementation specific internal condition (if applicable). Implementations must make best efforts to gracefully drain the Retry buffers when transitioning to LinkReset, however, entry to LinkReset must not timeout on waiting for the Retry buffer to drain. The Protocol Layer and Adapter must drain/flush their pipelines and retry buffer of the Flits for the corresponding Protocol Stack once the FDI state machines have entered LinkReset.

If all the FDI state machines and Adapter LSMs are in LinkReset, the Adapter triggers RDI to enter LinkReset as well.

The rules for LinkReset State transitions are as follows:

1. LinkReset→Reset: The lower layer transitions to the Reset state due to an internal request to move to Reset (example reset pin trigger) or **`lp_state_req`** == Active while **`pl_state_sts`** == LinkReset and all necessary actions with respect to LinkReset have been completed.
2. LinkReset→Disabled: The lower layer transitions to Disabled based on observing **`lp_state_req`** == Disabled or due to an internal request to move to Disabled while **`pl_state_sts`** == LinkReset.
3. Transitional State: The PHY is permitted to transition through Reset State, and when it does, Reset state exit conditions apply.
4. LinkReset→LinkError: The lower layer transitions to LinkError due to an internal request to move to LinkError or **`lp_linkerror`** assertion while **`pl_state_sts`** == LinkReset.

10.3.3.6 Disabled State Rules

Adapter triggers Disabled entry when any of the following events occur:

- Protocol Layer requests entry to Disabled state
- Software writes to the Link Disable bit corresponding to the underlying Protocol (e.g., the Link Disable bit in the Link Control register in PCIe)
- Remote Link partner requests entry to Disabled state through the relevant sideband message
- An implementation specific internal condition (if applicable)

Implementations must make best efforts to gracefully drain the Retry buffers when transitioning to Disabled, however, entry to Disabled must not timeout on waiting for the Retry buffer to drain. The Protocol Layer and Adapter must drain/flush their pipelines and retry buffer of the Flits for the corresponding Protocol Stack once the FDI state machines have entered Disabled.

If all the FDI state machines and Adapter LSMs are in Disabled, the Adapter triggers RDI to enter Disabled as well.

The rules for Disabled State are as follows:

- Disabled→Reset: The lower layer transitions to the Reset state due to an internal request to move to Reset (example reset pin trigger) or **`lp_state_req`** == Active while **`pl_state_sts`** == Disabled and all necessary actions with respect to Disabled transition have completed.
- Disabled→LinkError: The lower layer transitions to LinkError due to an internal request to move to LinkError or **`lp_linkerror`** assertion while **`pl_state_sts`** == Disabled.

10.3.3.7 LinkError State Rules

The lower layer enters LinkError state when directed by an **`lp_linkerror`** signal or due to Internal LinkError conditions. For RDI, the entry is also triggered if the remote Link partner requested LinkError entry through the relevant sideband message. It is not required to complete the stallreq/ack handshake before entering this state. However, for implementations where LinkError state is not a terminal state (terminal implies SoC needs to go through reset flow after reaching LinkError state), it is expected that software can come and retrain the Link after clearing error status registers, etc., and the following rules should be followed:

- If the lower layer decides to perform a **`pl_stallreq/lp_stallack`** handshake, it must provide **`pl_trdy`** to the upper layer to drain the packets. In cases where there is an uncorrectable internal error in the lower layer, these packets could be dropped and not transmitted on the Link.
- It is required for the upper layer to internally clean up the data path, even if **`pl_trdy`** is not asserted and it has sampled LinkError on **`pl_state_sts`** for at least one clock cycle.

The lower layer may enter LinkError state due to Internal LinkError requests such as when:

- Encountering uncorrectable errors due to hardware failure or directed by Upper Layer
- Remote Link partner requests entry into LinkError (RDI only)

The rules for LinkError state are as follows:

- LinkError→Reset: The lower layer transitions to Reset due to an internal request to move to Reset (e.g., reset pin assertion, or software clearing the error status bits that triggered the error) OR (**`lp_state_req`** == Active and **`lp_linkerror`** = 0, while **`pl_state_sts`** == LinkError AND minimum residency requirements are met AND no internal condition such as an error state requires the lower layer to remain in LinkError). Lower Layer must implement a minimum residency time in LinkError of 16 ms to ensure that the remote Link partner will be forced to enter

LinkError due to timeouts (to cover for cases where the LinkError transition happened and sideband was not functional).

10.3.3.8 Common State Rules

This section covers some of the common conditions for exit from Active, Retrain, L1, and L2 to LinkReset, Disable and LinkError states. For RDI, PM encoding and rules correspond to L1 in the text below.

The rules are as follows:

- [Active, Retrain, L1, L2]→LinkReset: The lower layer transitions to LinkReset based on observing **lp_state_req** ==LinkReset or due to an internal request to move to LinkReset or the remote Link partner requesting LinkReset over sideband.
- [Active, Retrain, L1, L2]→Disabled: The lower layer transitions to Disabled based on observing **lp_state_req** ==Disabled or due to an internal request to move to Disabled while **pl_state_sts** == Active, or the remote Link partner requesting Disabled over sideband.
- [Active, Retrain, L1, L2]→LinkError: The lower layer transitions to LinkError based on observing an internal request to move to the LinkError or **lp_linkerror** assertion, or the remote Link partner requesting LinkError over sideband. RDI must move to LinkError before propagating LinkError to all Adapter LSMs.

From a state machine hierarchy perspective, it is required for Adapter LSM to move to LinkReset, Disabled or LinkError before propagating this to CXL vLSMs. This ensures CXL rules are followed where these states are "non-virtual" from the perspective of CXL vLSMs.

Adapter LSM can transition to LinkReset or Disabled without RDI transitioning to these states. In the case of multi-protocol stacks over the same Physical Link/Adapter, each Protocol can independently enter these states without affecting the other protocol stack on the RDI.

If all the Adapter LSMs have moved to a common state of LinkReset/Disabled or LinkError, then RDI is taken to the corresponding state. If however, the Adapter LSMs are in different state combinations of LinkError, Disabled or LinkReset, the RDI is moved to the highest priority state. The priority order from highest to lowest is LinkError, Disabled, LinkReset. For a LinkError/LinkReset/Disabled transition on RDI, Physical Layer must initiate the corresponding sideband handshake to transition remote Link partner to the required state. If no response is received from remote Link partner for this message after 8ms, RDI transitions to LinkError.

If RDI moves to a state that is of a higher priority order than the current Adapter LSM, it is required for the Adapter to propagate that to the Adapter LSM using sideband handshakes to ensure the transition with the remote Link partner.

After transition from LinkError/LinkReset/Disable to Reset on RDI, the Physical Layer must not begin training unless the Physical Layer observes a NOP->Active transition on **lp_state_req** from the Adapter or observes one of the Link Training triggers defined in Chapter 4.0. The Adapter should not trigger NOP->Active unless it receives this transition from the Protocol Layer or has internally decided to bring the Link Up. The Adapter must trigger this on RDI if the Protocol Layer has triggered this even if **pl_inband_pres** = 0. Thus, if the Protocol Layer is waiting for software intervention and wants to hold back the Link from training, it can delay the NOP->Active trigger on FDI. Upper Layers are permitted to transition **lp_state_req** back to NOP after giving the NOP->Active trigger in order to clock gate while waiting for **pl_inband_pres** to assert.

If RDI transitions to L2, the exit is through Reset, and complete Link Initialization and Training flow will occur (including a fresh Parameter Exchange for the Adapter). After transition from L2 to Reset on RDI, the LTSM will begin the Link PM exit and retraining flow when a {LinkMgmt.RDI.Req.Active} sideband message is received or when the Adapter requests Active on RDI or it observes one of the Link Training triggers defined in Chapter 4.0.

If the Adapter LSM transitions to L2, but RDI does not go to a Link down state (i.e. Reset, LinkReset, Disabled, LinkError), then this is a “virtual” L2 state. The exit from L2 for the Adapter LSM in this case will go through Reset for the Adapter LSM, but it does not result in a fresh Parameter Exchange for the Adapter, and the protocol parameters and the Flit Formats remain the same as prior to L2 entry. An example of this is if there are multiple stacks on the same Adapter, and only one of the FDIs transitions to L2.

IMPLEMENTATION NOTE — LINKRESET/DISABLED

LinkReset and Disabled flows are primarily provided as a means to notify the remote Link partner that the corresponding Protocol Layer intends to trigger the set of actions defined for these by the underlying protocol (e.g., in the case of PCIe and CXL, both of these result in a Conventional Reset on the Upstream Port as defined in the *PCIe Base Specification*). These are typically controlled and co-ordinated through software/firmware. Note that regardless of protocol, there is no hardware mechanism from the UCIe Adapter or Physical Layer to guarantee quiescence or graceful draining of transactions for the LinkReset or Disabled transitions. If this is required by the underlying protocol, it must be handled through software/firmware or other implementation-specific mechanisms outside the UCIe Adapter and Physical Layer.

If the RDI state is already in a Link down state (i.e., Reset, LinkReset, Disabled, LinkError) and the Link is not currently training (Adapter can infer this from **`pl_phyinrecenter`**), then there is no need to notify the remote Link partner. Adapter or Physical Layer can complete the state transitions locally for this case. If RDI is in RESET and the Link is training, it is recommended to wait for training to complete before triggering a state transition with the remote Link partner to LinkReset or Disabled.

The following is written for Disabled state, but applies to both Disabled and LinkReset states.

- For PCIe or CXL protocols, the Downstream Port initiates the transition to Disabled. Because the Upstream Port goes through a Conventional Reset after transitioning to Disabled, the Upstream Port waits for Downstream Port to re-initiate Link Training once the corresponding SoC reset flow has finished.
- For Streaming protocols,
  - The initiating Protocol Layer transitions **`lp_state_req`** to Disabled. If the necessary conditions are met from the Adapter perspective (for example, attempting to drain the Retry buffer etc.), it forwards the request using the corresponding sideband message to the remote Link partner's Adapter.
  - On the remote Link partner, the Adapter transitions **`pl_state_sts`** to the requested state once the necessary conditions are met from the Adapter perspective (for example, attempting to drain the Retry buffer etc.). It also sends the corresponding sideband message response.

    If the Adapter needs to take the RDI to Disabled state, it is recommended to keep FDI **`pl_state_sts`** in Disabled state until that flow has completed. Otherwise, if the exit conditions for Disabled are met, it is permitted to transition to Reset state on FDI.

    Following this, the Protocol Layer on the remote Link partner in turn is permitted bring the FDI state back to Disabled if required by the underlying protocol. The Adapter must not trigger another sideband handshake for this scenario.
  - The initiating Adapter transitions **`pl_state_sts`** to Disabled upon receiving the sideband message response.
  - The Protocol Layers on either side of the Link can initiate an exit flow by requesting Active when **`pl_state_sts`** is Disabled, followed by a NOP->Active transition after the **`pl_state_sts`** is Reset.
- For configurations in which the Adapter is servicing multiple Protocol Layers, the Disabled or LinkReset handshakes are independent per Protocol Layer. In case the Adapter LSM has transitioned to Reset from Disabled or LinkReset for a given Protocol Layer, the Adapter must keep track of the most-recent previous state to determine the correct resolution for RDI state request.

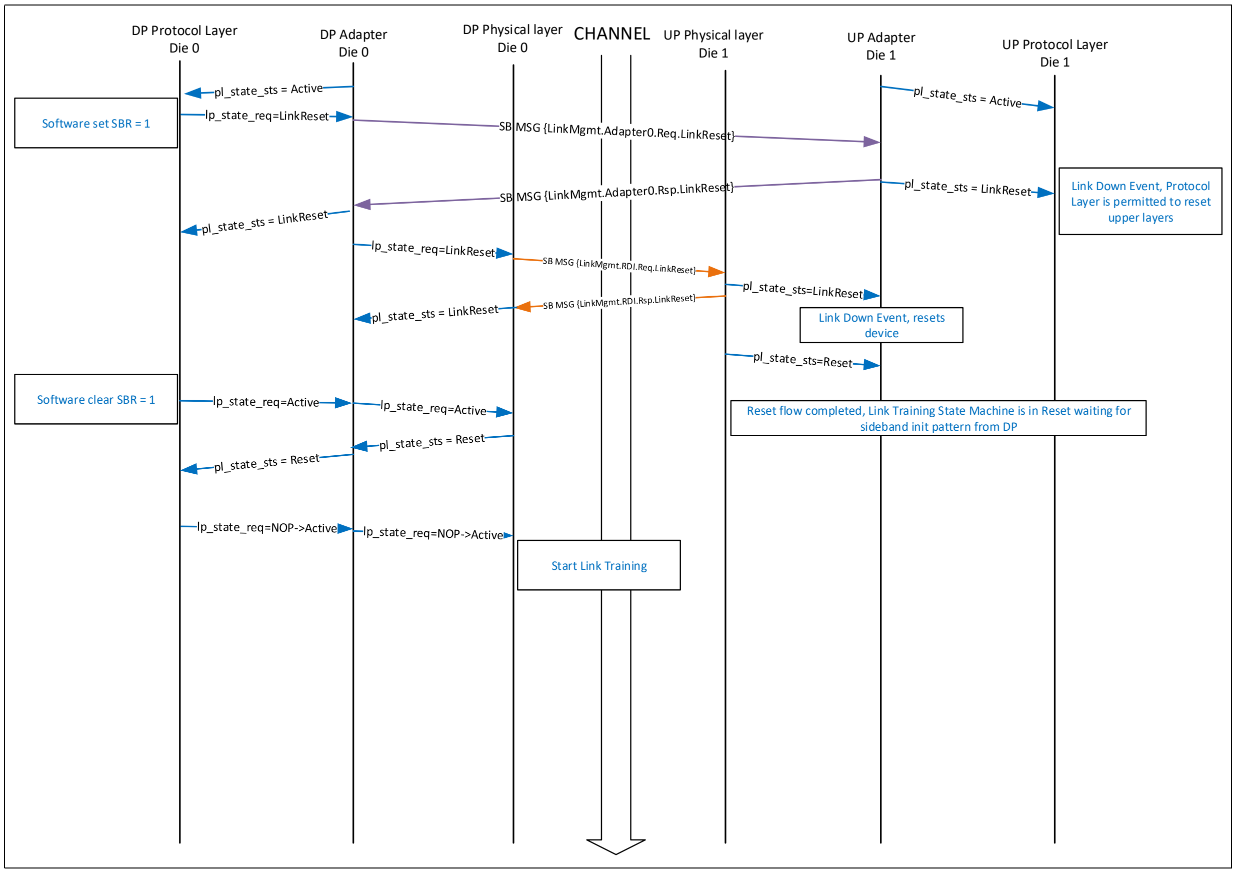
10.3.4 Example Flow Diagrams

10.3.4.1 LinkReset Entry and Exit

Figure 10-30 shows an example flow for LinkReset entry and exit. In the multi-protocol stack scenario, it is permitted for each protocol stack to independently transition to LinkReset. RDI is only transitioned to LinkReset if all the corresponding Adapter LSMs are in LinkReset. For the Link Down Event box, it is expected that SoC does not trigger the overall reset flow until the Physical Layer has completed all the relevant sideband handshakes with the remote Link partner that ensure the LTSM is also in Reset state.

Figure 10-30 also shows the link reset flow for a PCIe/CXL.io protocol. If Management Transport protocol is supported and negotiated on the same stack as PCIe/CXL.io protocol, the Management Port Gateway must still follow the LinkReset flow and reset requirements that correspond to PCIe/CXL.io.

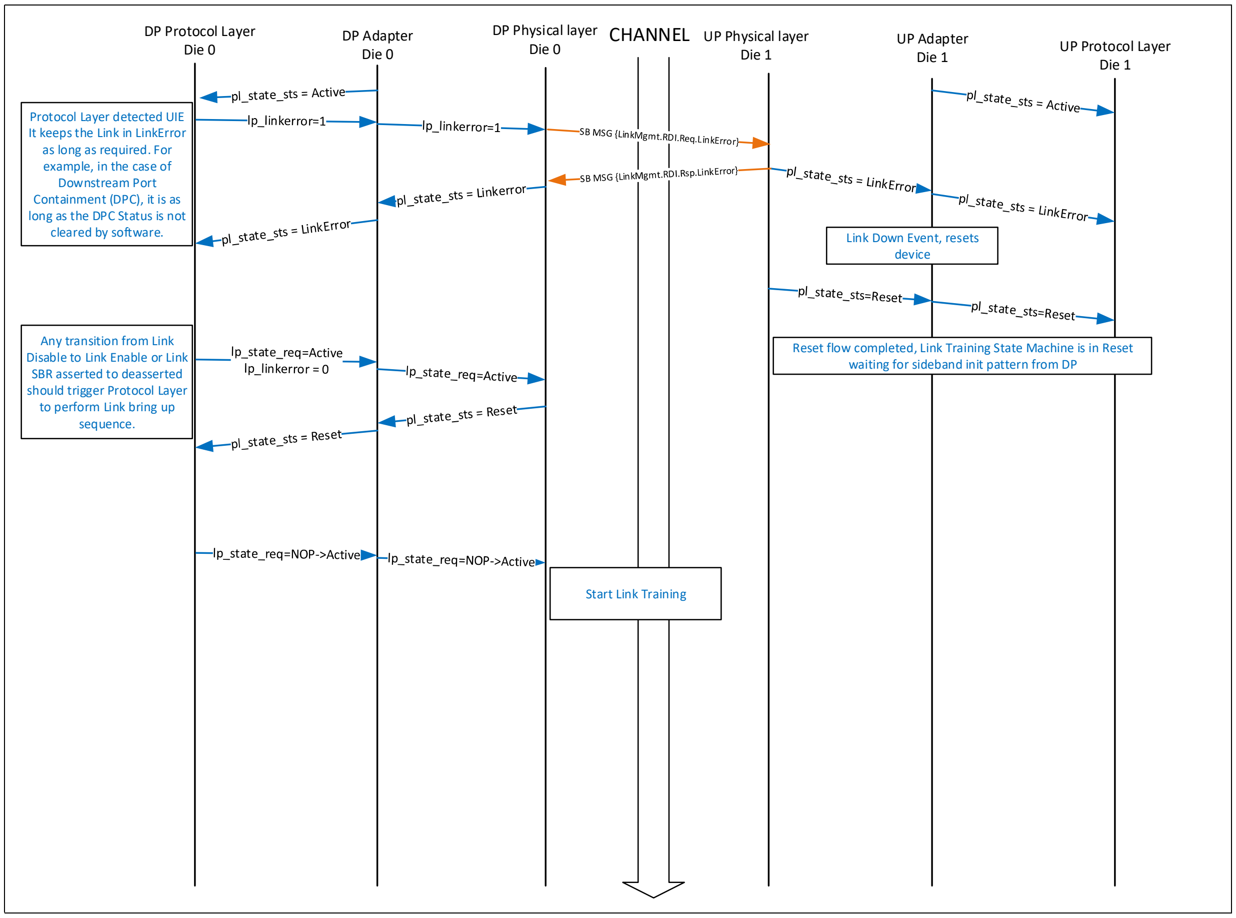
Figure 10-30. LinkReset Example



10.3.4.2 LinkError

Figure 10-31 shows an example of LinkError entry and exit when the Protocol Layer detected an uncorrectable internal error.

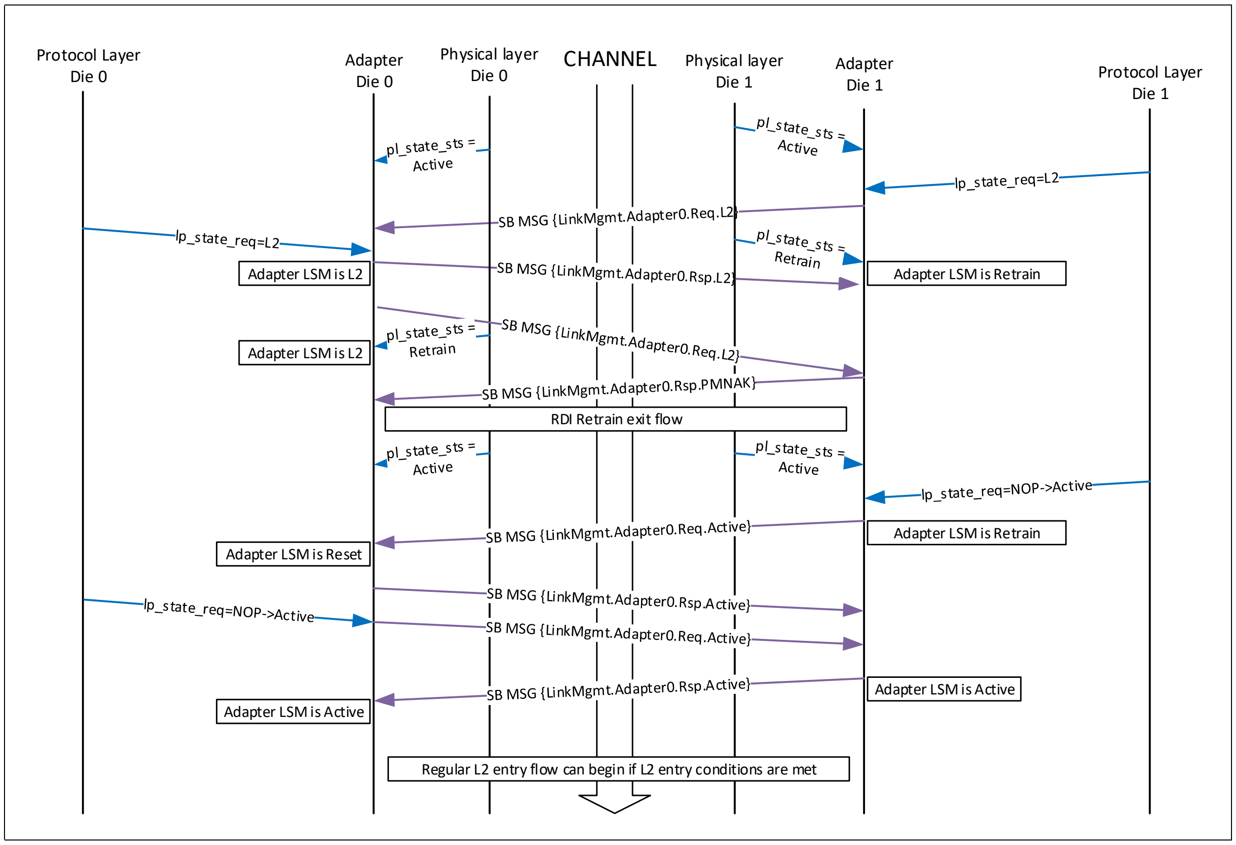
Figure 10-31. LinkError example


DP Protocol Layer Die 0
DP Adapter Die 0
DP Physical layer Die 0
CHANNEL
UP Physical layer Die 1
UP Adapter Die 1
UP Protocol Layer Die 1
pl_state_sts = Active
pl_state_sts = Active
Protocol Layer detected UIE It keeps the Link in LinkError as long as required. For example, in the case of Downstream Port Containment (DPC), it is as long as the DPC Status is not cleared by software.
lp_linkerror=1
lp_linkerror=1
SB MSG {LinkMgmt.RDI.Req.LinkError}
SB MSG {LinkMgmt.RDI.Rsp.LinkError}
pl_state_sts = LinkError
pl_state_sts = LinkError
pl_state_sts = Linkerror
pl_state_sts = LinkError
Link Down Event, resets device
pl_state_sts=Reset
pl_state_sts=Reset
Reset flow completed, Link Training State Machine is in Reset waiting for sideband init pattern from DP
Any transition from Link Disable to Link Enable or Link SBR asserted to deasserted should trigger Protocol Layer to perform Link bring up sequence.
lp_state_req=Active
lp_linkerror = 0
lp_state_req=Active
pl_state_sts = Reset
pl_state_sts = Reset
lp_state_req=NOP->Active
lp_state_req=NOP->Active
Start Link Training

10.3.4.3 Example of L2 Cross Product with Retrain on RDI

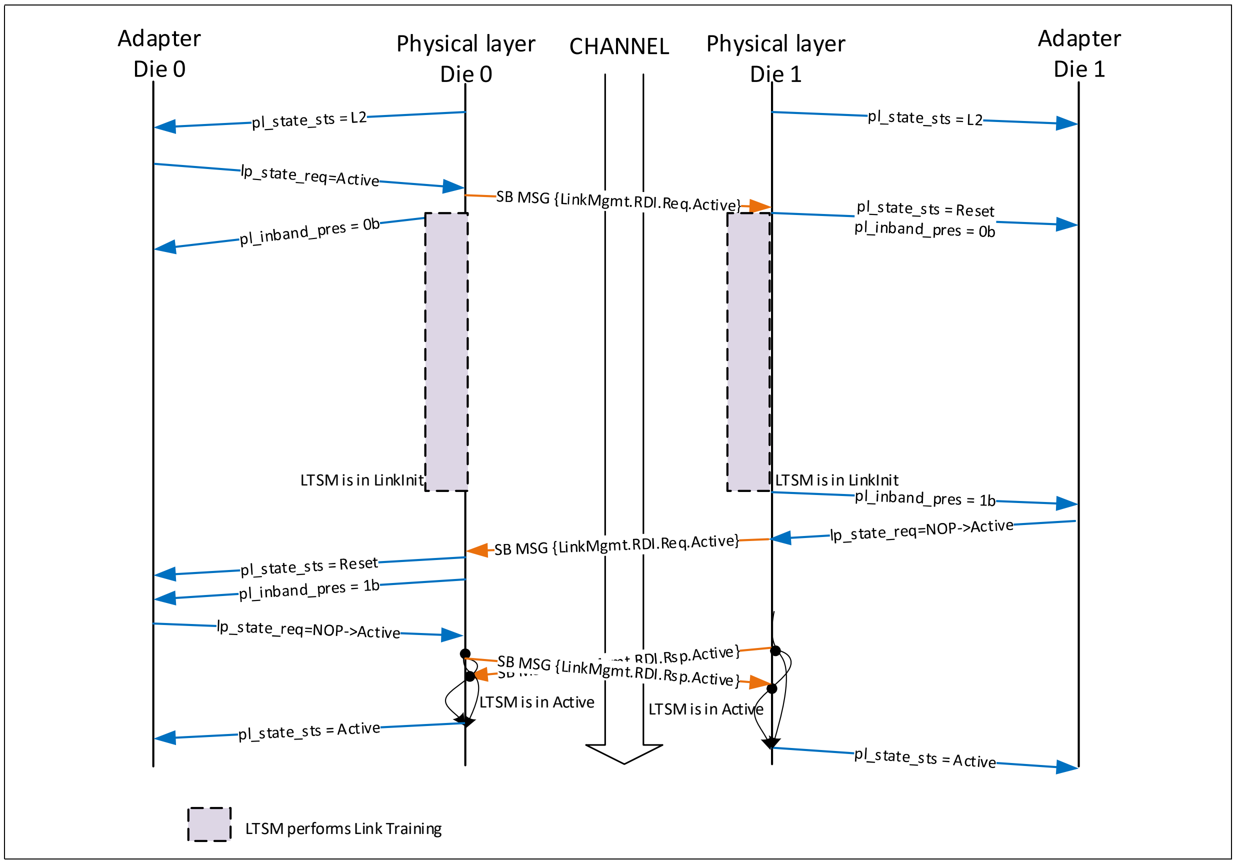
Figure 10-32 shows an example of L2 entry cross product with Retrain transition on RDI (i.e., both flows happen to overlap in a meaningful way) and the corresponding events to resolve the state on either side of the Link.

Figure 10-32. Example of L2 Cross Product with Retrain on RDI



10.3.4.4 L2 Exit Example for RDI

Figure 10-33. L2 Exit Example for RDI



§ §