Preliminary

DFI

DDR PHY Interface

Preliminary DFI 4.0 Specification
Addendum to DFI 3.1

Version 2

JANUARY 30, 2015

Release Information

Rev #	Date	Change
--	March 21, 2014	Initial release.
--	January 30, 2015	Updated figures for scaling and consistency with DFI 3.x. Added more chapters, updated per reviewers' feedback. Changed participants: Uniquify (new), LSI to Avago.

4. NOTWITHSTANDING ANYTHING ELSE WILL CADENCE'S TOTAL AGGREGATE LIABILITY FOR ANY CLAIM, SUIT, PROCEEDING OR OTHERWISE, RELATING IN ANYWAY TO THE DFI SPECIFICATION EXCEED $1.00USD.
5. NOTWITHSTANDING ANYTHING ELSE WILL ANY PARTICIPANT'S TOTAL AGGREGATE LIABILITY FOR ANY CLAIM, SUIT, PROCEEDING OR OTHERWISE, RELATING IN ANYWAY TO THE DFI SPECIFICATION EXCEED $1.00USD.
6. Licensee agrees that Cadence and the Participants may use, copy, modify, reproduce and distribute any written comments or suggestions ("Communications") provided regarding the DFI Specification by Licensee and that Licensee will not claim any proprietary rights in the DFI Specification, or implementations thereof by any Participant or third party, as a result of the use of the Communications in developing or changing the DFI Specification. Cadence and the participants will have no confidentiality obligations with respect to the Communications and Licensee will not include any confidential information of Licensee or any third party in any Communications.

Participants

ARM Avago Cadence Intel Samsung ST Synopsis Uniquify

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 Introduction

This preliminary specification is a compilation of approved changes to the DFI Specification. This document is intended to be used in conjunction with the *DDR PHY Interface (DFI) Specification Version 3.1*, March 21, 2014. The final version of the "DFI 4.0 Specification" will incorporate these changes into the current version of the specification.

DFI 4.0 extends the current specification to include LPDDR4 memory support, DDR4 RDIMM and LRDIMM support, and adds other DFI specific feature updates. The Preliminary DFI 4.0 Addendum is a separate document detailing the updates necessary to add these features to the DFI 3.1 Specification. The addendum will be integrated into the DFI 3.1 Specification at a later date. Each change description includes background, detailed description, compatibility with older versions, and compliance sections.

2.0 LP4 General Changes

2.1 Background

This chapter illustrates the general changes needed for IO and control for LPDDR4 support. The information does not include support discussed in other LPDDR4 related topics, including command/address (CA) training, Write DQ training, and channel support.

2.2 Detailed Description

The DFI 4.0 is to be modified to incorporate LPDDR4 signal changes. A preliminary set of general changes required for LPDDR4 is as follows:

1. Similar to LPDDR2/3, CA bus is used for transferring address and control. Therefore, similar to LPDDR3/2, the LPDDR4 interface will not use **dfi_cas_n**, **dfi_ras_n**, **dfi_we_n**, **dfi_act_n**, **dfi_bank**, and **dfi_bank_group**. Unlike LPDDR3/2, this is an SDR (single data rate) CA bus and is only 6 bits wide.
2. While termination is used in the devices, commands control the termination, not an explicit ODT pin. LPDDR4 does not require the **dfi_odt** signal. The specification does not need changes because DFI 3.1 already lists supported DRAM classes.
3. Burst lengths of 16 and 32 will require updates to RD/WR diagrams for including examples at higher burst lengths.
4. DBI support will be extended to include LPDDR4 as well as DDR4.

   Polarity of DBI reversed (active high instead of active low)
   A single DBI signal will be transmitted across DFI. The signal will be renamed **dfi_rddata_dbi** for all memories. A note is to be added indicating that the polarity of the signal is defined by the polarity of the corresponding memory signal. The text and diagrams need updating to reference **dfi_rddata_dbi** instead of **dfi_rddata_dbi_n**. Timing diagrams are to illustrate the proper application of the signal polarity for various memory types. On DFI, the MC must drive this signal with the same polarity as the corresponding memory signal. The PHY will not change the polarity of the signal.
5. Polarity of all chip select signals reversed (active high instead of active low)
   The following signals will be inverted from active low to active high for LPDDR4 because of the polarity inversion of the memory CS signal: **dfi_cs**, **dfi_rddata_cs**, **dfi_wrdata_cs**, **dfi_phy_rdlvl_cs**, **dfi_rdlvl_gate_cs**, **dfi_phy_wrlvl_cs**, **dfi_phy_wdqlvl_cs**, and **dfi_phy_calvl_cs**. For the control interface, a single CS bus will be transmitted across DFI.

   The signal will be renamed **dfi_cs** for all memories. A note is to be added indicating that the polarity of the signal is defined by the polarity of the corresponding memory signal. The text and diagrams need updating to reference **dfi_cs** instead of **dfi_cs_n**. Timing diagrams are to illustrate the proper application of the signal polarity for various memory types. On DFI, the MC must drive this signal with the same polarity as the corresponding memory signal. The PHY will not change the polarity of the signal.

   Likewise, all of the chip select signals on other interfaces will be renamed with the "_n" post-fix removed from the signal name and a note will be added indicating that the signal polarity is the same as dfi_cs. When dfi_cs is active high, all chip select signals are active high; when dfi_cs is active low, all chip select signals are active low. The text and diagrams need updating to the new signal names. The PHY needs to account for the polarity change and might require additional control logic if multiple memory classes are supported with different CS polarities. For the control interface, a single CS bus transmits across DFI.
6. Polarity of write data mask is already defined with both polarities (active high and active low). A single write data mask signal will be transmitted across DFI. The signal is already named **dfi_wrdata_mask** for all memories and,

for consistency, the same note is to be added indicating that the polarity of the signal is defined by the polarity of the corresponding memory signal. Timing diagrams are to illustrate the proper application of the signal polarity for various memory types. On DFI, the MC must drive this signal with the same polarity as the corresponding memory signal. The PHY will not change the polarity of the signal.

7. The DFI **dfi_reset_n** signal will be extended to include LPDDR4.
8. DFI timing parameters for read and write commands will be referenced from the second tick of the second command in the same manner that JEDEC LPDDR4 references RL and WL. Figure 1 and Figure 2 show examples of read timing, and Figure 3 and Figure 4 show examples of write timing.

FIGURE 1. *LPDDR4 Read Command*

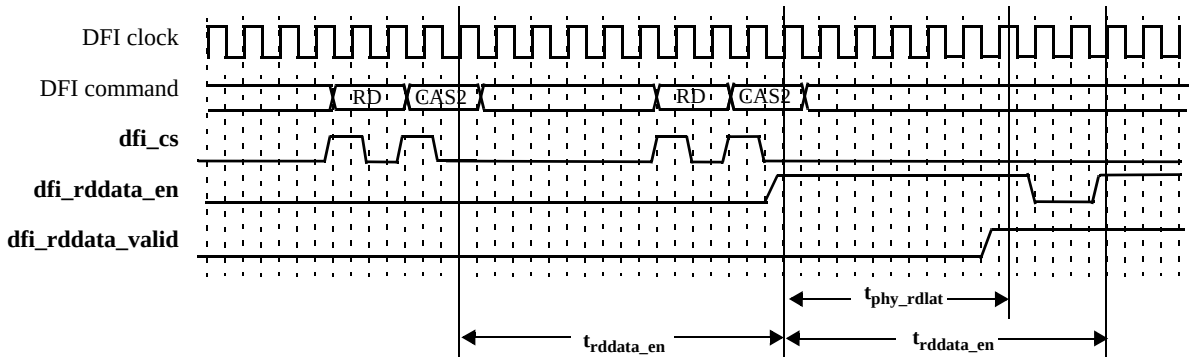


FIGURE 2. *LPDDR4 Read Command, 1:2 Frequency Ratio*

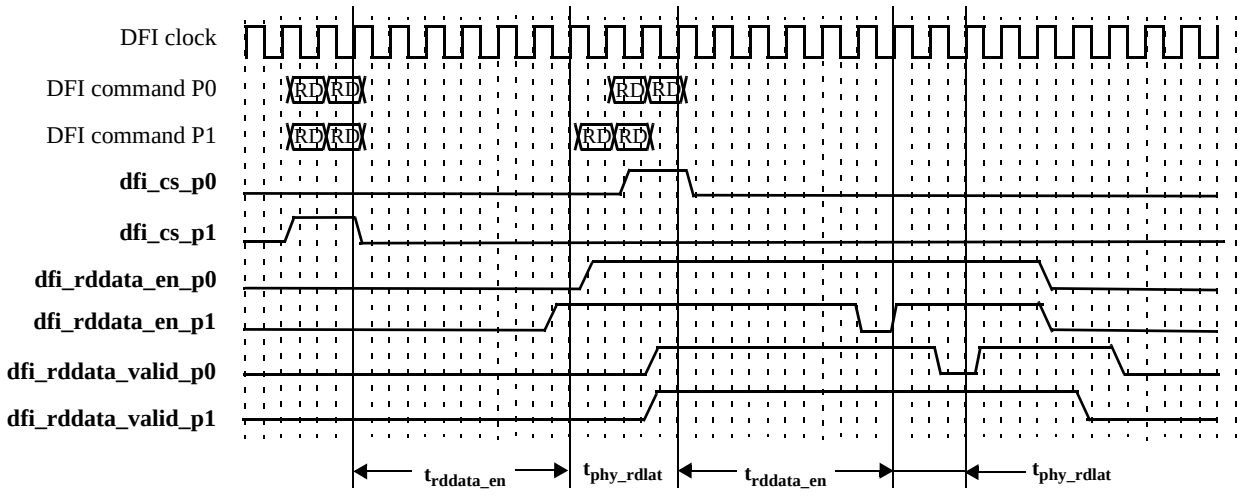


FIGURE 3. *LPDDR4 Write Command*

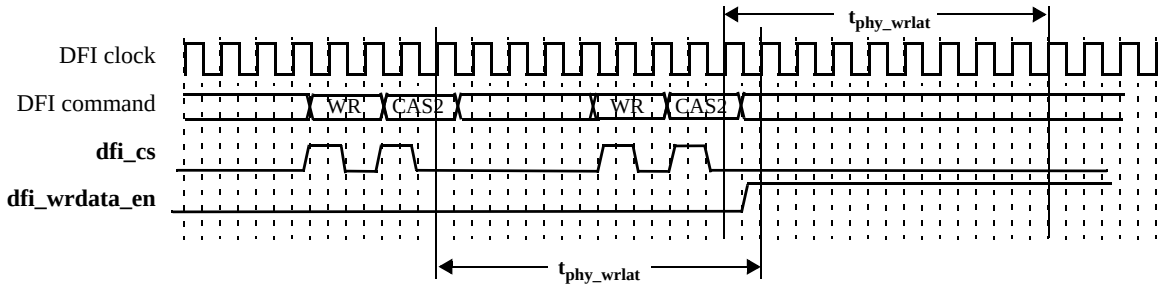
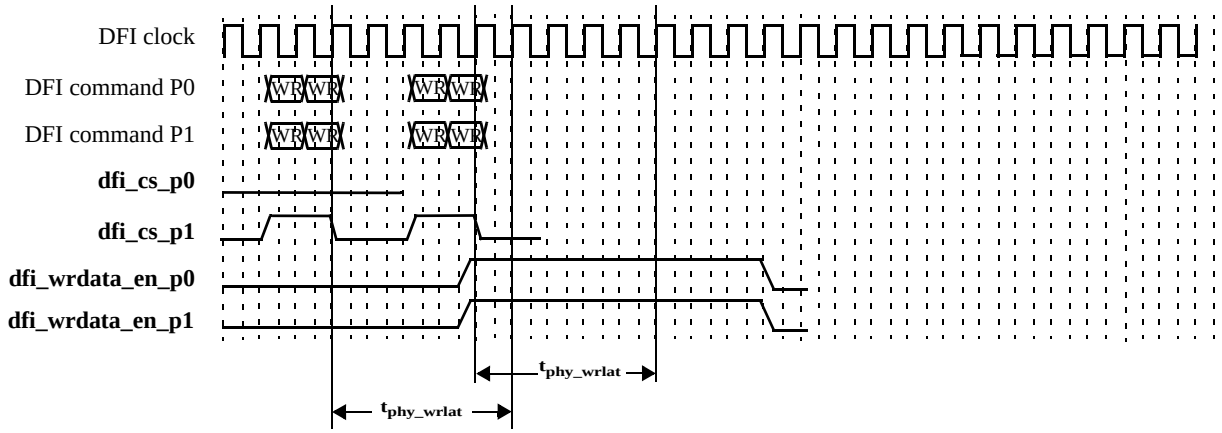


FIGURE 4. *LPDDR4 Write Command, 1:2 Frequency Ratio*



2.3 Compatibility with Older Version of DFI

LPDDR4 is not supported by older versions of DFI. All changes in this chapter are specific to LPDDR4 and should not create any compatibility issues.

2.4 Compliance

The changes proposed are relevant to DFI 4.0 only.

3.0 LPDDR4 Channel

3.1 Background

LPDDR4 defines the DRAM signaling in terms of channels; a single LPDDR4 device will have two channels. The channels operate independently, with the exception of a shared reset signal. A system can organize the memory channels as two “independent”16-bit memory interfaces or “combined” as a single 32-bit memory. It might be desirable to support both options in the MC and PHY.

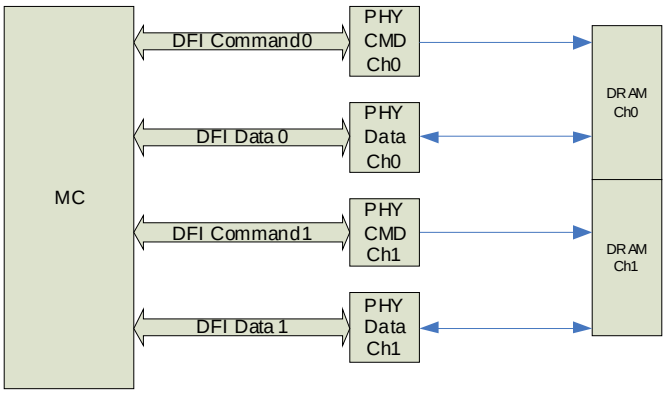
3.2 Detailed Description

To provide maximum flexibility and interoperability, a DFI bus will be defined as a single channel. When operating LPDDR4 as independent interfaces, two DFI busses must interface to a single LPDDR4 memory. These DFI busses will operate independently except for the reset. When operating LPDDR4 as a combined interface, two DFI busses will operate in lock-step. The read and write data interfaces will be used together for transferring the two channels’ worth of data. For all other interfaces, the PHY can optionally connect to a single interface or to both interfaces.

3.2.1 Independent Operation

Figure 5 illustrates connecting an LPDDR4 device for independent channels. The MC can represent one or two memory controllers. The memory reset is shared, Therefore, the MC must ensure that all DFI channels change their **dfi_reset_n** signals to the PHY simultaneously. The system can use any reset signal.

FIGURE 5. *LPDDR4 Independent Channels*



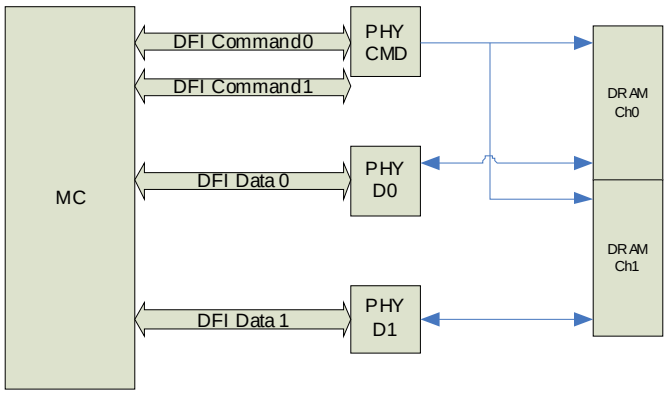
For combined operation, the PHY will be connected to two channels of read and write data interfaces. All signals for both channels must be present and operational as defined for the interface. For all other interfaces (control, update, status, training, low power) the PHY can connect to one channel's interfaces or to both channels’ interfaces. The connectivity is defined by a programmable parameter, $\mathbf{phy_{channel\_en}}$. If the PHY is connected to a single interface, the two channel lock-step operation is easily maintained. Furthermore, if the PHY connects to both channels but always generates the same responses to both channels (might be the same signal fanned out to both channels), the lock-step operation is maintained. However, if the PHY connects to both channels and the PHY driven signals on the two channels operate independently

(and might be different), special handling is required for the various interfaces. The following paragraphs discuss the implications of having independent responses on these interfaces.

For combined operation with the PHY connected to both command channels, Figure 5 remains unchanged. The command signals from the MC to the PHY are identical for both channels. The signals from the PHY to the MC might or might not be identical.

Figure 6 illustrates connecting an LPDDR4 device for combined channels, with a single control interface connecting to both channels. The MC can represent one or two memory controllers.

FIGURE 6. *LPDDR4 Combined Channels*



For combined operations, the control interface, read data interface control signals, and write data interface control signals that the MC drives operate in lock-step. In general, other MC-driven signals also operate in lock-step, unless defined otherwise. Some signals, such as **dfi_freq_ratio**, must drive the same value. Other interfaces, discussed later, require some special considerations in a combined configuration when using the interfaces from both channels, if the channels are not required to be driven identically. Using both channels for some interfaces and only a single channel for other interfaces is permitted. Unused interfaces should be tied inactive.

NOTE: A channel can be disabled, but not a single interface within a channel.

The update interface can be driven uniquely by the two channels. The controller update must be driven identically to both channels. If the PHY on one channel asserts the acknowledge and the other channel does not respond, or if the acknowledges from the channels are de-asserted at different times, both DFI channels remain idle until the update is completed on both channels. If the PHY update request is asserted differently on the two channels, when the MC acknowledges, both channels must remain idle. When the MC acknowledges the PHY update request, the acknowledge will be asserted to one or both channels, depending on whether one or both requests are asserted. When the acknowledge is asserted, no additional PHY update requests can be serviced until the update in progress completes.

At the status interface, initialization does not complete until both channels assert **dfi_init_complete**. For frequency change, the MC requests a frequency change in lock-step by asserting **dfi_init_start** to both channels. If one channel acknowledges and the other channel does not acknowledge, the MC terminates the frequency change on the channel that acknowledged by de-asserting **dfi_init_start** without changing the clock frequency. Both channels must wait for the completion of the frequency change handshake on the acknowledging channel before running additional commands. The **dfi_freq_ratio** signal must be identical for both channels.

The training interface will work as currently defined. Training is done on a per-slice basis, and slices should align within channels. In addition, independent slice training is proposed in Section 5.0, “Per-Slice Read Leveling,” on page 23.

The low power interface can receive unique responses from the different channels. If one channel acknowledges a low power request, and the other channel does not; both channels must maintain the same operation and adhere to the wakeup time that is associated with the channel that acknowledged the low power request. The MC can assert additional low power requests to the channel that is not in low power state. However, the MC cannot de-assert the acknowledged request when the other channel request is still pending. If the two channels are not in the same low power state, the larger wakeup time applies to both interfaces.

NOTE: The timing parameters for all interfaces operating in lock-step must be programmed identically.

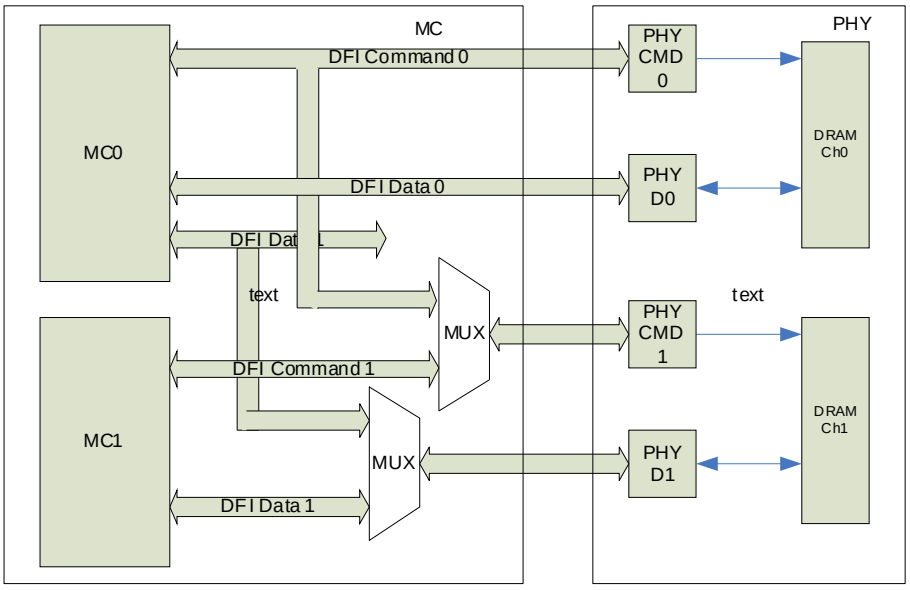
3.2.2 Multi-Configuration Support

It might be desirable to develop MCs and PHYs that can operate in both an independent and combined mode of operation. For interoperability, it would be preferable to define the device, MC or PHY, that is responsible for multiplexing the bus when supporting both independent and combined operations. The recommendation is placing the burden for multiplexing within the MC rather than the PHY.

NOTE: It is also possible to achieve the same system by using external multiplexing on the DFI interface.

Figure 7 illustrates the MC multiplexing the DFI bus for operating in both modes. Combined operation means a single MC (MC0), and independent operation means two MCs (MC0 and MC1). The MC block internally multiplexes the command and data for MC0 and MC1, and no multiplexing of the command and data is required within the PHY.

FIGURE 7. *Example of LPDDR4 Multi-Configuration*



When operating in combined mode, the DFI channels operate in lock-step. In this mode, the PHY can either connect to the interfaces for both channels (as Figure 5 and Figure 6 show), or connect to a single interface. When connecting to a single interface, the other channel can be disabled. When using only a single channel, channel 0 should be used and channel 1 disabled.

3.2.3 New Parameter Description

The **phy$_{channel_en}$** programmable parameter indicates whether the PHY connects to a single channel or both channels when it operates in combined mode. For independent mode, both channels must always be enabled. It is assumed that support for combined and independent modes and current mode of operation are defined outside of DFI. Table 1 defines the new programming parameter.

TABLE 1. *LPDDR4 Channel Parameter*

Parameter	Defined By	Min	Max	Description
phy$_{channel_en}$	System/PHY	Hh	Hh	Defines which DFI channels are enabled. 01: Channel 0 enabled, channel 1 disabled. 10: Channel 1 enabled, channel 0 disabled. 11: Channel 0 and channel 1 enabled. 00 is not a legal value.

3.3 Compatibility with Older Version of DFI

The channel concept is specific to LPDDR4. Older versions of DFI will not support LPDDR4. All other memory types do not have channels, so they will operate in a similar manner as the combined channel, as Figure 6 shows. However, when operating with a non-LPDDR4 memory, only a single command interface will be active. If this is a shared memory configuration supporting LPDDR4, only channel 0 must be driven by the MC when using non-LPDDR4 memory. If LPDDR4 memory is not a supported memory type, the channel concept does not apply, and only a single command channel should be present.

3.4 Compliance

DFI 4.0 compliant MCs and PHYs that support LPDDR4 memory must support the channel architecture as described in this chapter. For MC's and PHY's not supporting LPDDR4, this chapter is not applicable.

4.0 DB Data Buffer (DB) Training

4.1 Background

A new training interface is added for DB-to-DRAM training. This interface defines a mode of operation within the PHY and transfer of training data between the PHY and MC.

4.2 Detailed Description

DDR4 LRDIMMs include a data buffer (DB) between the PHY and DRAM components. The DB will capture and re-drive the DQ/DQS bus to and from the DRAM and PHY. The DB-to-DRAM interface must be trained to accurately capture read data and drive write data and strobes. In PHY independent mode, this training is to occur without MC involvement. In PHY evaluation mode, the MC is to set up the training in the DB and DRAM and receive the training response. Table 2 shows the DFI signals to be added for this purpose.

TABLE 2. *DFI Signals to Be Added*

DFI Signal	Driven By	Width	Description
dfi_db_train_en	MC	NUM_SLICES	Enable training mode in PHY. In this mode, the PHY performs the following tasks: • Transfers commands from the MC (commands will not use DQ/DQS bus) • Captures DQ inputs and transfer data to MC on response bus
dfi_db_train_resp or **dfi_db_train_resp_wN**	PHY	NUM_SLICES * bits per slice	DQ data that is returned to MC for evaluation. All DQ bits are returned. DQ data is to be defined per phase in frequency ratio systems. For example, in a 1:2 frequency ratio system, the DFI bus will include the following signals: • **dfi_db_train_resp_w0** • **dfi_db_train_resp_w1**

All DB-to-DRAM training, which includes up to four training modes, uses these signals.

4.2.1 Parameters

Table 3 defines the new parameters to be added for LRDIMM DB training.

TABLE 3. *Parameters to Be Added*

Timing Parameter	Defined By	Description
$t_{phy\_db\_train\_en}$	PHY	Number of PHY DFI clocks after **dfi_db_train_en** asserts until the 1st training command can be issued.

TABLE 3. *Parameters to Be Added (Continued)*

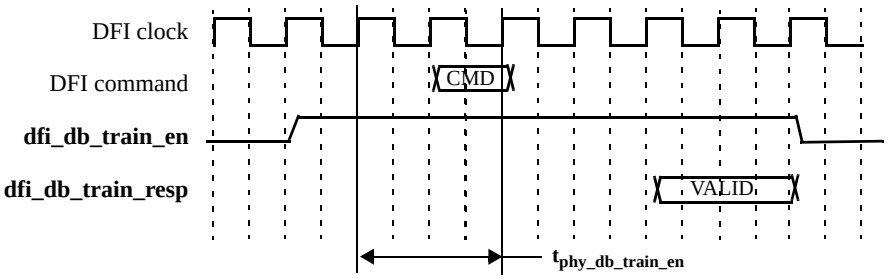
Timing Parameter	Defined By	Description
$t_{phy\_db\_train\_resp}$	PHY	Maximum number of DFI clocks required for transferring DQ data through the PHY. Specifically, this is the maximum number of DFI clocks from PHY DQ input to the PHY **db_train_resp** signal output.

When the **dfi_db_train_en** signal is de-asserted, the **dfi_db_train_resp** signal is no longer valid.

NOTE: No timing parameter defined for de-assertion of **dfi_db_train_en**.

Figure 8 shows a DB training example interface, used for multiple DB-to-DRAM training modes. Not all modes require commands from the MC, and round trip times are not consistent.

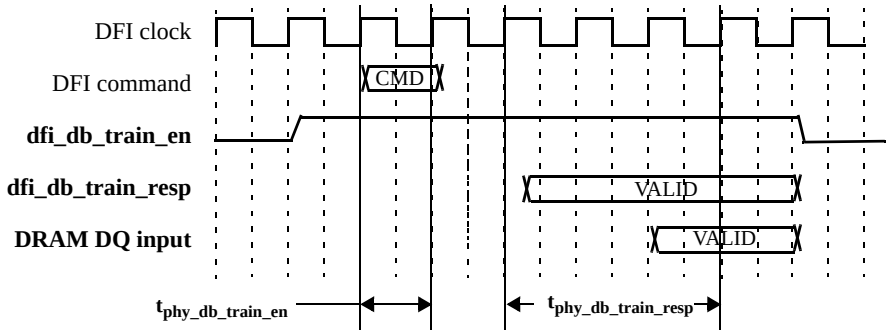
FIGURE 8. *DB Training*



No explicit timing relationship has been defined from the **dfi_db_train_en** signal or DFI CMD to **dfi_db_train_resp** signal. Instead, the MC is to use the $t_{db\_train\_resp}$ parameter for determining the delay that is required for transferring

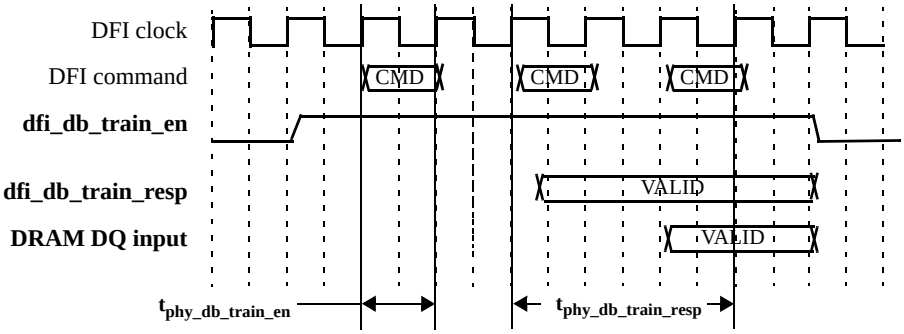
response through the PHY. This information is to be combined with round-trip delay information that are based on JEDEC and board delays. Figure 9 illustrates the $t_{phy\_db\_train\_resp}$ timing.

FIGURE 9. *DB Training with $t_{phy\_db\_train\_resp}$*



DB training can involve one or multiple training sequences, which occur while **db_train_en** is asserted. Figure 10 illustrates the case where multiple MPR read commands are issued to the DRAM/DB for read receiver enable training.

FIGURE 10. *DB Training with Multiple MPR Read Commands*



NOTE: No DFI timing parameters are required for CMD-to-CMD delays. These requirements are determined by JEDEC/ Vendor specifications of the memory and DIMM devices.

The DRAM DQ input is to continuously return to the MC on the **dfi_db_train_resp** bus. Depending on the training algorithm, this DQ input might contain periods of invalid data. The MC is to use the $\mathbf{t_{phy\_db\_train\_resp}}$ timing parameter along with information about the command stream for determining when to capture valid information from DFI.

4.3 Compatibility with Older Version of DFI

A DFI 4.0 compliant PHY that implements DB training interface will need to tie the **dfi_db_train_en** input to 0 when the PHY interfaces with an older DFI version of the MC. A DFI 4.0 MC interfacing to an older DFI version PHY will need to disable the DB training interface.

4.4 Compliance

The DB training interface signals are relevant to MCs and PHYs that support DDR4 LRDIMMs and PHY evaluation-mode training. MCs and PHYs that support only PHY independent mode do not need to include this interface.

5.0 Per-Slice Read Leveling

5.1 Background

A new functionality is added to DFI training signals, driven by MC to the PHY, and DFI training requests, driven by PHY to the MC for indicating the slice that is being trained during a leveling operation. These signals are to be driven independently per slice to allow per-slice training.

5.2 Detailed Description

In DFI 3.1, certain training signals, driven by MC to the PHY, and DFI training requests, driven by PHY to the MC may not be defined per slice. Taking read leveling as an example, the interface provides a signal for all slices in **dfi_rdlvl_en** and **dfi_rdlvl_gate_en** signals. However, the MC replicates the underlying **rdlvl_en** signal across all slices, allowing only an all-or-nothing solution. The proposed modification allows the MC to independently drive each slice when it asserts **dfi_rdlvl_en** and **dfi_rdlvl_gate_en** signals.

In DFI 3.1, certain training signals, driven by MC to the PHY, and DFI training requests, driven by PHY to the MC might not be defined per slice. Taking read training as an example, the interface provides a signal for all slices in **dfi_rdlvl_en** and **dfi_rdlvl_gate_en** signals. However, the MC replicates the underlying **rdlvl_en** signal across all slices, allowing only an all-or-nothing solution. The proposed modification allows the MC to independently drive each slice when it asserts **dfi_rdlvl_en** and **dfi_rdlvl_gate_en** signals.

Furthermore, the PHY must drive a signal **dfi_rdlvl_req** and **dfi_rdlvl_gate_req** from each memory data slice across the interface to the MC. The PHY must not combine the signals from each slice into a single signal.

NOTE: This feature also requires that any subsequently defined training sequence signals that the MC drives to the PHY (and also DFI training requests that the PHY drives to the MC) must be defined independently per slice.

Table 4 lists the affected signals.

TABLE 4. *Affected Signals*

DFI Signal	Driven By	Width
dfi_rdlvl_gate_en **dfi_rdlvl_en** **dfi_wrlvl_en** **dfi_calvl_en**	MC	NUM_SLICES
dfi_rdlvl_gate_req **dfi_rdlvl_req** **dfi_wrlvl_req** **dfi_calvl_req**	PHY	

5.2.1 MC-Driven Training Request

NOTE: Although this information covers all types of training interfaces, this example discusses only read leveling.

To initiate read training, the MC is to drive **dfi_rdlvl_en** signal, which is used for enabling the read training logic in the PHY independently for each slice, as Figure 11 illustrates. During read training, **dfi_rddata_valid** and **dfi_rddata** signals are ignored. All evaluations and delay changes are handled within the PHY. When the PHY finds the necessary edges and completes read training for a slice under training, the PHY drives the corresponding **dfi_rdlvl_resp** signal high, which informs the MC that the procedure is complete for the slice under training. The MC then de-asserts the **dfi_rdlvl_en** signal for the respective slice. When the **dfi_rdlvl_en** signal de-asserts, the PHY stops driving the **dfi_rdlvl_resp** signal.

5.2.2 PHY-Driven Training Request

The PHY can request gate training by driving the **dfi_rdlvl_gate_req** signal or read data eye training by driving the **dfi_rdlvl_req** signal. To support an independent request to level per slice, **dfi_rdlvl_req** and **dfi_rdlvl_gate_req** signals can be driven individually by each slice as Figure 11 and Figure 12 show. If necessary, the MC can use one or more $\mathbf{MC_{rdvl\_slice\_group[n]}}$ parameters for specifying pre-defined sets of dfi_rdlvl_en signals that might be needed to assert as a group.

NOTE: The $\mathbf{MC_{rdvl\_slice\_group[n]}}$ parameter is defined with a bit per slice and has a width [NUM_SLICES -1 :0]. Groups defined by this parameter are mutually exclusive, and can be used to accomplish multiple pass training for RDIMM and LRDIMM.

FIGURE 11. *Per-Slice Read Training $MC_{rdvl\_slice\_group}$ = 8'h0F*

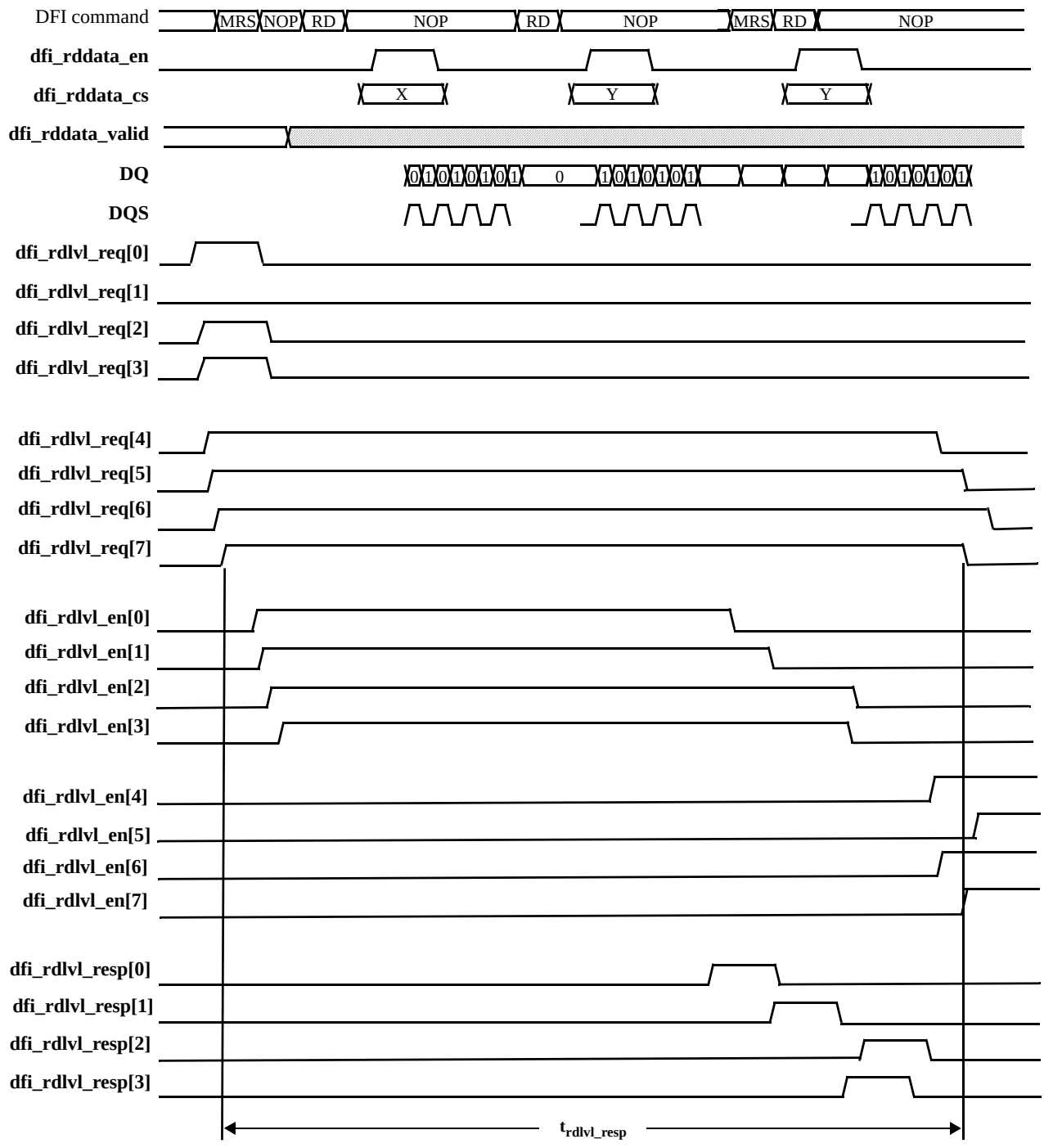
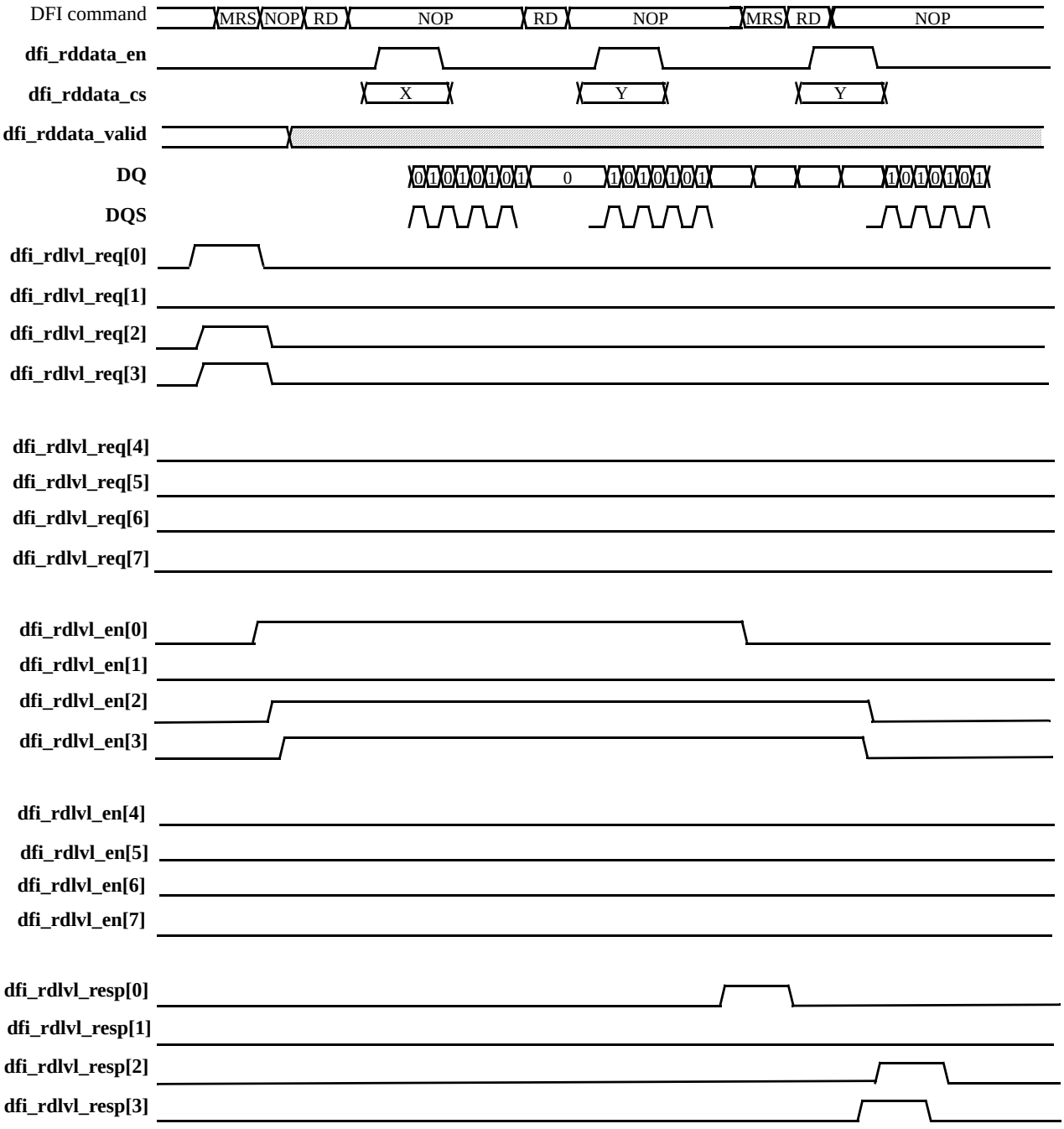


FIGURE 12. *Per-Slice Read Training $MC_{rdvl\_slice\_group}$ = 8'h00*



If the PHY asserts **dfi_rdlvl_req** or **dfi_rdlvl_gate_req** signal for any of the slices in $\mathbf{MC_{rdvl\_slice\_group[n]}}$, the MC drives the **dfi_rdlvl_en** or **dfi_rdlvl_gate_en** signal, respectively, for all the slices that are set in $\mathbf{MC_{rdvl\_slice\_group[n]}}$. Using the $\mathbf{MC_{rdvl\_slice\_group[n]}}$ parameter in this way allows the MC to assert the **dfi_rdlvl_en** signal and complete read training for all slices in multiple sets.

To support an independent request to level per slice, $\mathbf{MC_{rdvl\_slice\_groupn[n]}}$ can be set to 0. When $\mathbf{MC_{rdvl\_slice\_group[n]}}$ is set to 0, the **dfi_rdlvl_req** and **dfi_rdlvl_gate_req** signals can be driven individually by each slice.

To revert back to DFI 3.1 behavior, each slice in $\mathbf{MC_{rdvl\_slice\_group[n]}}$ can be set to 1. When $\mathbf{MC_{rdvl\_slice\_group[n]}}$ is set to 1, **dfi_rdlvl_req** and **dfi_rdlvl_gate_req** signals can be driven individually by each slice. The MC must respond to any of these requests by asserting all enable signals (**dfi_rdlvl_en** or **dfi_rdlvl_gate_en**).

If necessary, the PHY can use one or more parameters $\mathbf{PHY_{rdvl\_slice\_group[n]}}$ for specifying pre-defined sets of dfi_rdlvl_en signals that are needed to assert as a group by the MC when the PHY asserts **dfi_rdlvl_req** or **dfi_rdlvl_gate_req** signal. $\mathbf{MC_{rdvl\_slice\_group[n]}}$ and $\mathbf{PHY_{rdvl\_slice\_groupn[n]}}$ values might or might not be compatible.

To support an independent request to level per slice, $\mathbf{PHY_{rdvl\_slice\_groupn[n]}}$ can be set to 0. To revert back to DFI3.1 behavior, each slice in $\mathbf{PHY_{rdvl\_slice\_groupn[n]}}$ can be set to 1. The PHY and MC have independent parameters for the purpose of identifying compatibility and training requirements for both the PHY and MC.

The MC is not required to respond to each individual per-slice **dfi_rdlvl_req** signal concurrently. In addition, the PHY must initiate training early enough for the following situations:

- For allowing for larger read training intervals in LRDIMM/RDIMM systems
- In cases where the MC does not respond to each individual per-slice **dfi_rdlvl_req** signal concurrently in LRDIMM/RDIMM systems
- In cases where the MC does not respond to each individual per-slice **dfi_rdlvl_req** signal concurrently

NOTE: The MC must not assert a **dfi_rddata_en** signal for slices that do not assert **dfi_rdlvl_en** signals.

5.3 Compatibility with Older Version of DFI

A DFI 4.0 compliant PHY that implements the new behavior for signaling will be incompatible with an MC that does not drive or require per-slice read training. A DFI 4.0 MC might need to restrict itself to all slice-read leveling, if a PHY does not support independent per-slice leveling responses, regardless of DFI version. A DFI 4.0 MC can be compatible with a PHY that does not support independent per-slice leveling.

NOTE: $\mathbf{MC_{rdvl\_slice\_groupn[n]}}$ and $\mathbf{PHY_{rdvl\_slice\_group[n]}}$ settings might or might not be compatible.

5.4 Compliance

Independent assertion of DFI training signals is required for the MC if it is DFI 4.0 compliant. To be DFI 4.0 compliant, the PHY must drive a DFI training request signal from each memory data slice independently.

6.0 CA Training

6.1 Background

This chapter describes the additional signals and timing parameters required for LPDDR4 CA training. (LPDDR4 DRAMs support CA training.) CA training for LPDDR4 has several new features, including support for adjustment to train the VREF value, frequency changes to train at various frequency set points, and additional training pattern options. The DFI CA training interface is being enhanced to support these new features.

6.2 Detailed Description

Similar to LPDDR3, LPDDR4 supports CA training. With LPDDR4, CA training is used for training the CA data eye as well as the internal CA VREF value. The DRAM devices define a sequence that incorporates frequency change events during CA training, which are used for training higher, terminated speeds.

The defined CA training interface is to be re-used for LPDDR4 with changes made to the interface for simplification and to add specific LPDDR4 functionality.

6.2.1 Simplifying the DFI CA Training Bus

CA training was defined in DFI 3.1. In the DFI 3.1 CA training description, the MC sends the CA training pattern across DFI on the **dfi_address** bus. However, the pattern is not used in the MC and applies to only the PHY. To simplify the interface, the following changes are to be made.

1. MC no longer sends training patterns (background and foreground) during CA training. Instead, the PHY sends the required pattern to DRAM.
2. MC sends a new signal, **dfi_calvl_ca_sel**, to indicate when the foreground data should be transferred. This signal is to be defined per phase when the PHY operates at a multiple of the MC frequency.

   For example, in a system that uses 1:2 frequency ratio, the DFI bus is to include the following signals:

   dfi_calvl_ca_sel_p0

   dfi_calvl_ca_sel_p1

   The timing from **dfi_calvl_ca_sel** to **dfi_cs**, and the width of this pulse is to be defined by the following timing parameters:

   $t_{calvl\_cs\_ca}$
   Number of PHY DFI clocks from **dfi_calvl_ca_sel** assertion to **dfi_cs**.

   $t_{calvl\_ca\_sel}$
   Width of **dfi_calvl_ca_sel** in PHY DFI clock cycles

   NOTE: When the PHY operates at a multiple of the MC frequency, this timing parameter defines the width that is generated by combining the per-phase PHY inputs.
3. The encoding of the **dfi_calvl_resp** from the PHY requires modification. The 'b01 encoding that indicates that the PHY is done training with the current pattern no longer applies. This encoding will be RESERVED.

Table 5 and Table 6 summarize the changes to DFI CA training for simplification. These changes include signal encoding changes, new DFI signals, and new DFI timing parameters. Additionally, CA training no longer uses **dfi_address** to send the pattern across DFI. With DFI 4.0, the controller is NOT responsible for sending training data to the PHY.

TABLE 5. *New DFI CA Training Signal for CA Control*

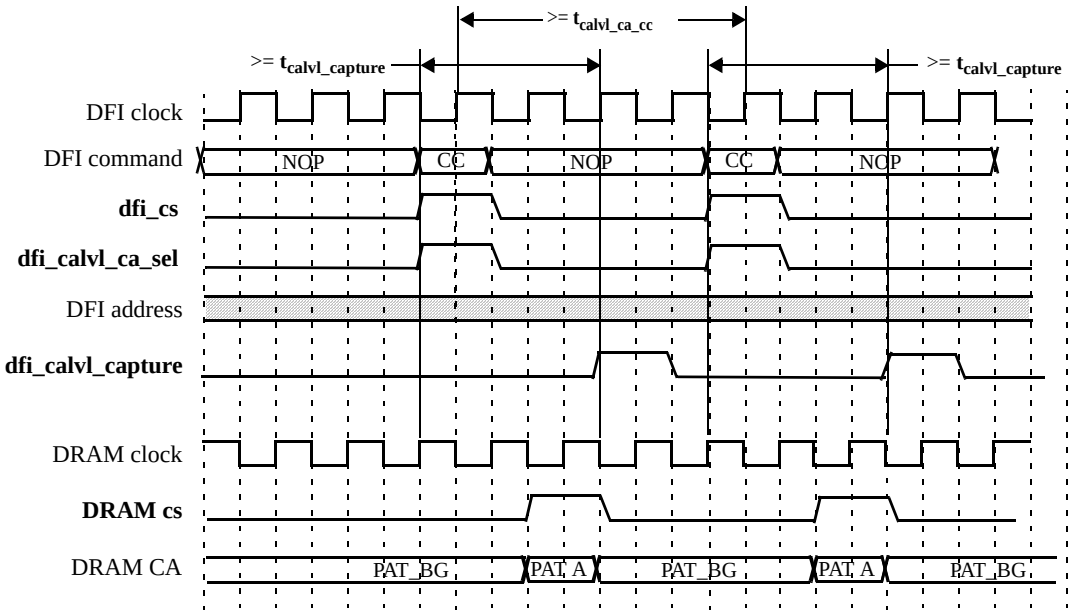
Signal	Driven By	Description
dfi_calvl_ca_sel or **dfi_calvl_ca_sel_pN**	MC	Strobe indicating when foreground pattern should be transmitted to the DRAM. The timing from **dfi_calvl_ca_sel** to **dfi_cs** and the width of **dfi_calvl_ca_sel** is defined by DFI timing parameters. The width of **dfi_calvl_ca_sel** is defined by DFI timing parameters. In a system that uses frequency ratio, this signal interfaces to the CA slices, which is defined per-phase. For example, in a system that uses 1:2 frequency ratio, the DFI bus includes the following signals: • **dfi_calvl_ca_sel_p0** • **dfi_calvl_ca_sel_p1**

TABLE 6. *New DFI CA Training Timing Parameters for CA Control*

Timing Parameter	Defined By	Description
$\mathbf{t_{calvl\_cs\_ca}}$	PHY	Number of PHY DFI clocks from **dfi_calvl_ca_sel** assertion to **dfi_cs**.
$\mathbf{t_{calvl\_ca\_sel}}$	PHY	Width of **dfi_calvl_ca_sel** in PHY DFI clock cycles. In a system that uses frequency ratio, this timing parameter defines the width that is generated by combining the per-phase PHY inputs.

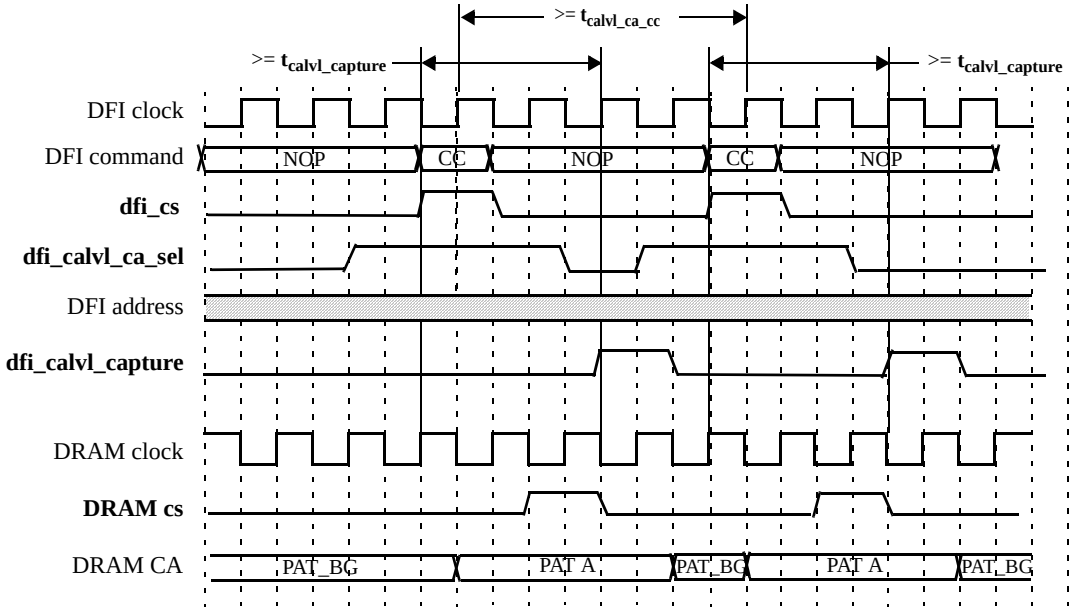
Figure 13 and Figure 14 illustrate the functionality of the **dfi_calvl_ca_sel** signal. In both diagrams, each system uses matched frequency ratio. The diagrams show two different sets of timing values.

FIGURE 13. *dfi_calvl_ca_sel Timing (Matched Frequency), $t_{calvl\_cs\_ca}$ = 0; $t_{calvl\_ca\_sel}$ = 1*



NOTE: The polarity of the chip select signal is defined by the DRAM. Applies to **dfi_cs** and **DRAM_cs**.

FIGURE 14. *dfi_calvl_ca_sel Timing (Matched Frequency), $t_{calvl\_cs\_ca}$ = 1; $t_{calvl\_ca\_sel}$ = 3*



NOTE: The polarity of the chip select signal is defined by the DRAM. Applies to **dfi_cs** and **DRAM_cs**.

6.2.2 Setting CA VREF Values During CA Training

To perform CA VREF training during CA training, the MC must drive the VREF data on a new bus, **dfi_calvl_data**. This bus is to be 7-bits wide to match the width of the DQ bus that is used for transmitting VREF data. DQ data bit reordering between the ASIC and DRAM must be taken into account for correct operation.

In addition to the VREF data, the MC must drive a new DFI signal, **dfi_calvl_strobe,** for CA VREF programming during CA training. This signal contains one bit per data slice. When a bit within this bus is asserted, the PHY drives a single pulse on the corresponding DQS output.

To ensure that the data that is driven on the DQ bus meets setup and hold time around the DQS strobe, two new timing parameters are defined. The **t_{calvl_strobe}** timing parameter defines the required delay from driving the **dfi_calvl_data** busses to driving the **dfi_calvl_strobe** busses. The **t_{calvl_data}** timing parameter defines the required number of clocks for holding the data pattern. These timing parameters are to be defined by the system (DRAM setup/hold) requirements as well as PHY delay skews.

The response from the PHY is to determine when a new VREF value is needed. The 'b01 encoding is to define the condition where training is done for the current VREF value. The next value should be transferred across DFI with the **dfi_calvl_strobe** and **dfi_calvl_data** busses.

Because CA training might require iterating through multiple variables, including VREF values, CA data patterns, delay settings, frequency, etc., sequence control might be required from both the MC and the PHY. To facilitate this requirement, a new signal, **dfi_calvl_done**, defines when the MC sends the final pattern associated with the training

sequence. For example, in CA training, this signal might indicate the final VREF value. If training is complete at the end of the training sequence in which the MC asserts the **dfi_calvl_done** signal, the PHY asserts **dfi_calvl_resp** = 'b11. Alternatively, if the PHY requires iterating across additional variables, the PHY must assert **dfi_calvl_resp** = 'b01, and the training routine continues. In this case, the MC should loop back to the first value that was driven at the beginning of training and iterate through the values as before. The MC must continue looping through the values until the PHY responses with **dfi_calvl_resp** = 'b11 when **dfi_calvl_done** is asserted, indicating that training is complete.

The results for a specific VREF value would be useful to the MC when selecting the next VREF. To facilitate communication of this information on the interface; a new signal, **dfi_calvl_result**, is defined. The signal indicates how the results of the current value compares with the best previous value, either better or worse. The evaluation process continues until the PHY asserts **dfi_calvl_resp** = 'b11, at which point, the evaluation is complete and subsequent training sequences will begin a new evaluation process.

Table 7 summarizes the new DFI signals and the new encoding for LPDDR4 CA training.

TABLE 7. *New DFI CA Training Signals for VREF and Response*

Signal	Driven By	Description
dfi_calvl_strobe or **dfi_calvl_strobe_pN**	MC	Defined per slice. Each assertion of 1 DFI clock generates a pulse of DQS to the DRAM. Asserting for multiple clocks generate the same number of consecutive DQS pulses. This signal interfaces to the data slices.
dfi_calvl_data or **dfi_calvl_data_pN**	MC	Defined per slice. Each slice receives a 7-bit DFI bus. The signal sets the value to transfer to DRAM for VREF training. NOTE: This MC generates a single value that replicates on each slice. Therefore, *the value that is transferred on* **dfi_calvl_data** *will be the same for each slice.*
dfi_calvl_done	MC	Completion of the MC iterating across different VREF training values. If the PHY completes the training sequence with **dfi_calvl_resp** = 'b11, training is complete. If the PHY completes the training sequence with **dfi_calvl_resp** = 'b01, training continues, and the MC should restart with the beginning VREF value and drive **dfi_calvl_done** accordingly. The **dfi_calvl_done** signal is valid during **dfi_calvl_en** and should be unchanged during **dfi_calvl_en**. The signal is invalid when **dfi_calvl_en** = 'b0. If the MC does not have anything to iterate on, **dfi_calvl_done** can be driven to a “1” all the time.

TABLE 7. *New DFI CA Training Signals for VREF and Response (Continued)*

<table>
<tr><th>Signal</th><th>Driven By</th><th>Description</th></tr>
<tr><td>dfi_calvl_resp</td><td>PHY</td><td>Training response from PHY. The following table shows the encoding for each DRAM class.
<table>
<tr><th>Encoding</th><th>LPDDR3</th><th>LPDDR4</th></tr>
<tr><td>'b00</td><td colspan="2">Not done</td></tr>
<tr><td>'b01</td><td>RESERVED for LP3 (previously used to indicate a new pattern which is now handled internally in the PHY).</td><td>Done with current VREF sequence. The MC should update the VREF value. If the dfi_calvl_done signal is asserted, the MC should start from the initial VREF value.</td></tr>
<tr><td>'b10</td><td>Done with current training. Change segment.</td><td>RESERVED</td></tr>
<tr><td>'b11</td><td colspan="2">Training complete</td></tr>
</table></td></tr>
<tr><td>dfi_calvl_result</td><td>PHY</td><td>Training result from PHY. Indicates how the current VREF value being driven by the MC compares with all previous values.
0: Current value is worse than the previous best value.
1: Current value is better than the previous best value.
The dfi_calvl_result signal is valid when the dfi_calvl_resp signal is asserted (non-ZERO) and should be unchanged when the dfi_calvl_resp signal is asserted—not valid when dfi_calvl_resp = 'b00.</td></tr>
</table>

The feature includes a separate data bus for transferring VREF values that are used during CA training. This bus is defined as 7-bits—the width of the VREF value. The bit order and slice location is to be determined by the PHY.

NOTE: While the VREF value is transferred across DFI on **dfi_calvl_data**, the PHY can disregard this input and drive a different value on the appropriate DQ outputs.

Table 8 summarizes new timing parameters.

TABLE 8. *New DFI VREF Data and Strobe Timing Parameters*

Timing Parameter	Defined By	Description
t_{calvl_strobe}	System	Minimum number of DFI PHY clocks from **dfi_calvl_data** to **dfi_calvl_strobe** that are required for meeting setup time at the DRAM.
t_{calvl_data}	System	Minimum number of DFI PHY clocks from **dfi_calvl_data** assertion to **dfi_calvl_data** de-assertion that are required for meeting hold time at the DRAM.

When training completes, training values that the MC (VREF, for example) and the PHY (delay settings) control must be in sync so that the DRAM VREF and PHY delays are set to corresponding values.

NOTE: If the CA VREF values are programmed by using MRS DRAM commands, the explicit VREF CA training signals **dfi_calvl_data_pN** and **dfi_calvl_strobe_pN** are not used.

6.2.3 Frequency Change Events

The defined LPDDR4 CA training procedure could include multiple frequency changes. The frequency change interface is to be re-used for this purpose. With DFI 4.0, the frequency change handshake interface, using **dfi_init_start** and **dfi_init_complete** is applicable during CA training. Additionally, timing parameters in DFI4.0 define the timing requirements before and after the frequency change event. Table 9 summarizes new timing parameters.

TABLE 9. *New DFI CA Training Timing Parameters for Frequency Change*

Timing Parameter	Defined By	Description
$t_{init\_start\_min}$	PHY	Minimum number of DFI clocks before **dfi_init_start** can be driven after a previous command, training event, or both. With CA training, such events are as follows: **1.** De-assertion of **dfi_cke** **2.** Assertion of **dfi_calvl_capture** **3.** Assertion of **dfi_calvl_strobe**
$t_{init\_complete\_min}$	PHY	Minimum number of DFI clocks before **dfi_init_complete** can be driven after a previous command, training event, or both. With CA training, such events are as follows: **1.** Assertion of **dfi_cke** **2.** Assertion of **dfi_cs** **3.** Assertion of **dfi_calvl_strobe**

NOTE: The defined timing parameters are generally applicable and are not specific to CA training.

Figure 15 and Figure 16 illustrate the new CA training DFI bus. Figure 16 demonstrates the timing and control when a value of 'b10 is returned on the **dfi_calvl_resp** signal. In this case, the **dfi_cke** signal is not required to re-assert.

FIGURE 15. *CA Training with CA VREF Set*

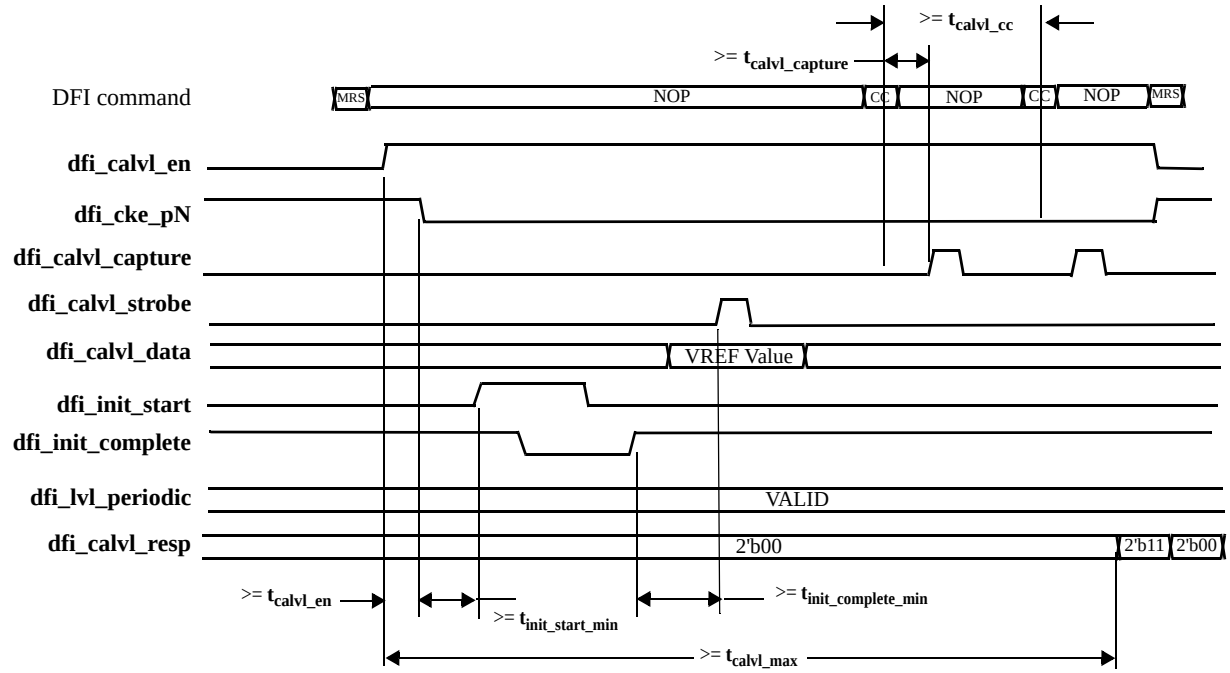
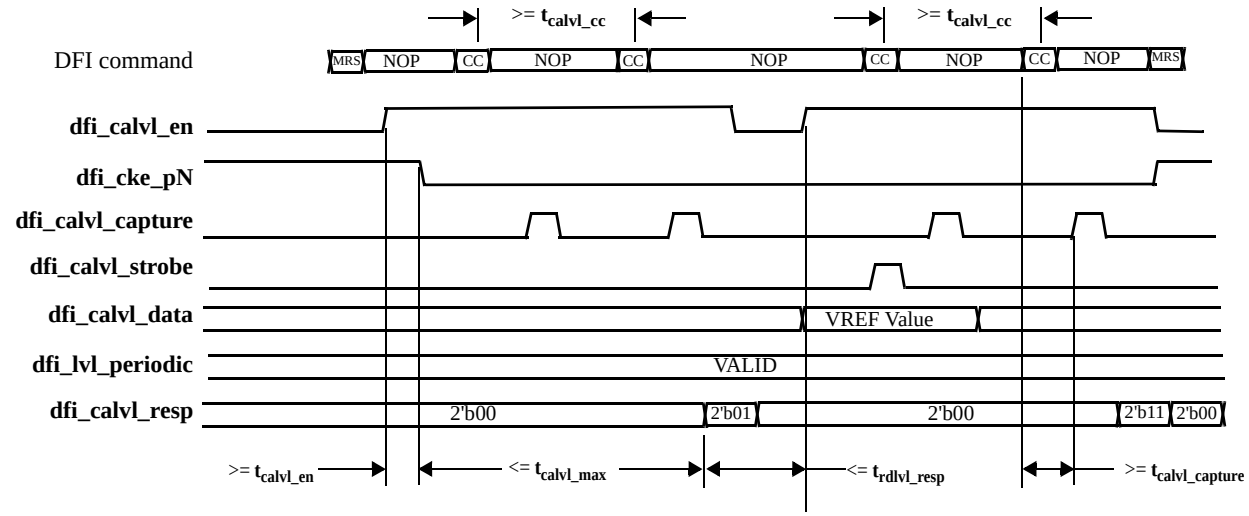


FIGURE 16. *VREF Change During CA Training*



6.3 Compatibility with Older Version of DFI

DFI 3.1 is the only previous version of DFI that supports CA training. DFI 3.1 does not support LPDDR4 memories. The only backwards compatibility scenario is using DFI 4.0 CA training with LPDDR3 memory.

6.3.1 Using a DFI 4.0 PHY that is Operating with a DFI 3.1 MC

The VREF interface to the PHY should be tied inactive (**dfi_calvl_strobe**, **dfi_calvl_data**). The CA bus during training should be controlled by the MC; the MC will not have support for the **dfi_calvl_ca_sel** signal. The DFI 4.0 PHY should have support for the LPDDR3 responses as defined in Table 7. The new response handshake signals (**dfi_calvl_done**, **dfi_calvl_result**) will not be supported by the MC and should be tied inactive. The frequency change enhancements are not applicable. A DFI 4.0 PHY should provide necessary support for DFI 3.1 MC's if backwards compatibility is required.

6.3.2 Using a DFI 4.0 MC that is Operating with a DFI 3.1 PHY

The VREF interface will not be connected (**dfi_calvl_strobe**, **dfi_calvl_data**). The CA bus during training should be controlled by the MC; the PHY will not have support for the **dfi_calvl_ca_sel** signal and will not generate the CA foreground and background patterns. The DFI 4.0 MC should have support for the LPDDR3 responses as defined in Table 7. The new response handshake signals (**dfi_calvl_done**, **dfi_calvl_result**) will not be supported by the PHY and should be tied inactive. The frequency change enhancements are not applicable. A DFI 4.0 MC should provide necessary support for DFI 3.1 PHY's if backwards compatibility is required.

6.4 Compliance

The proposed changes are relevant to DFI 4.0 only. This chapter defines changes that are required with “DFI Training Mode” support. No changes are required for PHY Independent training or software-based training, “Non-DFI Mode”.

7.0 DFI Read Data Eye Training Sequence Enhancement

7.1 Background

DFI Read Data Eye Training in PHY evaluation mode defines various read data patterns that are supported by the DRAM and used by the system for training the PHY read data capture logic. The system defines some patterns and order to be used for finding the optimal delay setting in the PHY. The data patterns are defined in DRAM, and the MC can modify the data by using mode register write commands. This information must be communicated to the PHY during training via the **dfi_lvl_pattern** signal.

As currently defined, no method is defined for communicating to the PHY the number of read data patterns to expect, and no method defined for the PHY to communicate to the MC to send, whether next pattern or indication that training is complete. Because the PHY is responsible for evaluating the responses from memory and adjusting the timing delays, the PHY requires additional information and control in execution of the training sequences.

7.2 Detailed Description

For LPDDR4, the Calibration register definition has been expanded to support many read data patterns. Because the data patterns are programmed by the MC, the MC communicates the read data patterns to the PHY by using the **dfi_lvl_pattern** signals. The signal width is to be expanded for LPDDR4 for supporting a larger number of patterns. The **dfi_lvl_pattern** signal does not pass the pattern, but it passes an index to the pattern. The actual pattern that is associated with the index must be programmed or otherwise defined in both devices. The additional patterns are applicable to read data eye training only, not read DQS gate training.

For all memory types that support read data eye training, to communicate training sequence information to the PHY and between the PHY and the MC, new signals are being defined. The **dfi_rdlvl_resp** signal is to be changed from a single bit to a 2-bit signal. The new definition is to be similar to the 2-bit **dfi_calvl_resp** signal where the PHY communicates either pattern complete or sequence complete. LPDDR4 has enhanced the read DQ training pattern options, thus, enhancing the DQ Calibration Registers A and B. These registers are loaded with default patterns that can be overwritten or modified by using the Invert registers. These registers can be used for customizing the data patterns that are read from the DQ pattern mode registers. The LPDDR4 **dfi_lvl_pattern** signal is four bits, the equivalent size to the DDR4 definition. LPDDR4 is to support the two defined patterns and support up to eight custom patterns. Custom pattern information is not sent across DFI and must be programmed into the MC and PHY so that both devices can detect the data that is associated with the selected pattern.

Table 10 shows **dfi_lvl_pattern** encoding.

TABLE 10. *dfi_lvl_pattern encoding*

<table>
<tr><th colspan="4">DFI_RDLVL_MPR_<MC|PHY>_OUT</th><th colspan="4" rowspan="2">Description</th></tr>
<tr><th>Non-Default</th><th>Format</th><th colspan="2">MPR</th></tr>
<tr><th>[3]</th><th>[2]</th><th>[1]</th><th>[0]</th><th colspan="2">LPDDR3</th><th colspan="2">LPDDR4</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td rowspan="2">Default</td><td>Access MPR32</td><td rowspan="2">Default</td><td>Access MPR32</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>Access MPR40</td><td>Access MPR40</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td colspan="2" rowspan="14">Reserved</td><td colspan="2" rowspan="6">Reserved</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td rowspan="8">Non-Default MPR Value. Format and value determined programmatically</td><td>Sequence 0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>Sequence 1</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>Sequence 2</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>Sequence 3</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>Sequence 4</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>Sequence 5</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>Sequence 6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>Sequence 7</td></tr>
</table>

Because read data eye training might require iterating through multiple variables, including VREF values, read data patterns, delay settings, etc., sequence control might be required from both the MC and the PHY. To facilitate this requirement, the **dfi_rdlvl_done** signal is to be defined for indicating when the MC sends the final pattern that is associated with the training sequence.

For example, in read DQ training, this might indicate the final read data pattern. If training is complete upon completing the training sequence in which the MC asserts **dfi_rdlvl_done**, at the end of the sequence, the PHY is to assert **dfi_rdlvl_resp** = 'b11 as the response. However, if the PHY continues to iterate through additional variables, the PHY is to assert **dfi_rdlvl_resp** = 'b01, which continues the training routine. In this case, the MC should loop back to the first value that was driven at the beginning of training, iterating through the values as before. The MC is to continue looping through the values until the PHY responds with **dfi_rdlvl_resp** = 'b11 and training completes. The handshake is defined in the same manner as CA training and Write DQ training.

Because read gate training does not use multiple sequences, a similar handshake is not considered necessary. Currently, the **dfi_rdlvl_resp** signal is used for both read data eye training and read gate training. For read gate training, only the 2'b00 and 2'b11 responses are to be defined. The **dfi_rdlvl_done** signal applies to only read data eye training, not to read gate training.

Table 11 defines the new signals.

TABLE 11. *New Signal Description*

<table>
<tr><th>Signal</th><th>From</th><th>Width</th><th>Default</th><th>Description</th></tr>
<tr><td>dfi_rdlvl_resp</td><td>PHY</td><td>2 bits per slice</td><td>0</td><td>Training response from PHY. The following table shows the encoding
<table>
<tr><th rowspan="2">Encoding</th><th colspan="2">Description</th></tr>
<tr><th>Read data eye training</th><th>Read gate training</th></tr>
<tr><td>'b00</td><td>Not done</td><td>Not done</td></tr>
<tr><td>'b01</td><td>Done with current sequence. MC should update value if applicable (ex: read data pattern). If dfi_rdlvl_done is asserted, MC should restart from initial value.</td><td>RESERVED</td></tr>
<tr><td>'b10</td><td>RESERVED</td><td>RESERVED</td></tr>
<tr><td>‘b11</td><td>Training complete</td><td>Training complete</td></tr>
</table>
When training is not complete and the PHY responds with 'b01, the MC performs the following actions:
1. De-asserts dfi_rdlvl_en
2. Sets up the next sequence.
3. Re-asserts dfi_rdlvl_en.</td></tr>
<tr><td>dfi_rdlvl_done</td><td>MC</td><td>1 bit per slice</td><td>0</td><td>MC completes iterating across different training values (final training pattern, etc.). If the PHY completes the training sequence with dfi_rdlvl_resp = 'b11, training is complete. If the PHY completes the training sequence with dfi_rdlvl_resp = 'b01, training continues, and the MC restarts with the beginning value and drive dfi_rdlvl_done accordingly.
The dfi_rdlvl_done signal is valid during dfi_rdlvl_en and should be unchanged during dfi_rdlvl_en. The signal is invalid when dfi_rdlvl_en = 'b0.</td></tr>
</table>

Figure 17 and Figure 18 illustrate the new disconnect signaling protocol.

FIGURE 17. *Beginning of LPDDR4 Read Data Eye Training Sequence*

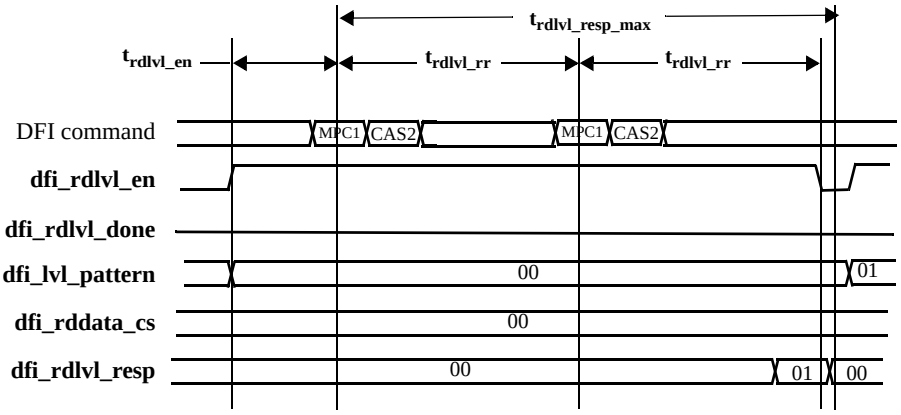
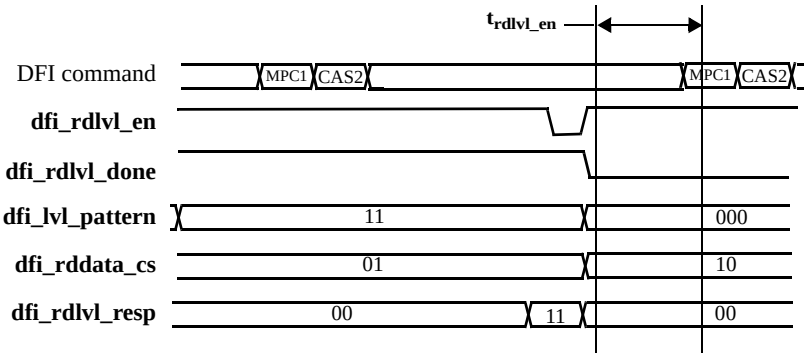


FIGURE 18. *End of LPDDR4 Read Data Eye Training Sequence for CS=1; Start of Training of CS=2*



7.3 Compatibility with Older Version of DFI

If an MC uses the new **dfi_rdlvl_resp** signal with an earlier DFI version PHY, connect the single bit PHY **dfi_rdlvl_resp** signal to the MC **dfi_rdlvl_resp** bit 0 and tie off the other bit. Because the PHY does not provide sequence information through the response, the MC sequence will need to be controlled by some other means. The **dfi_rdlvl_done** signal is to be unconnected. The **dfi_lvl_pattern** change is applicable to only LPDDR4 memory, which is supported for only DFI 4.0.

If a PHY is working with an MC that is earlier than DFI 4.0, connect **dfi_rdlvl_resp**, bit 0, to the single bit **dfi_rdlvl_resp** of the MC. The **dfi_rdlvl_done** signal will not be provided by the MC, so the sequence will need to be controlled by other means. The **dfi_lvl_pattern** change is applicable to only LPDDR4 memory, which is supported for only DFI 4.0.

7.4 Compliance

DFI 4.0 PHYs must drive **dfi_rdlvl_resp** according to the new definition. Support for **dfi_rdlvl_done** and **dfi_lvl_pattern** is optional.

DFI 4.0 MCs must support the new **dfi_rdlvl_resp** definition and **dfi_rdlvl_done**. MC support for non-default **dfi_lvl_patterns** is optional.

8.0 DFI Read/Write Chip Select

8.1 Background

In the DFI 3.1 specification, the chip selects were added to DFI Read as well as Write interface so that PHY can independently compensate for timing differences on the data interface accessing different chip selects. However, in the Tables 6 & 9 describing these signals, these are declared as OPTIONAL during memory access and thus negating the benefit.

Modify the “description” field of **dfi_wrdata_cs** (Table 6) and **dfi_rddata_cs** (Table 9).

8.2 Detailed Description

1. Existing and proposed descriptions of **dfi_wrdata_cs** (Table 6) are as follows:

   DFI 3.1 (existing):

   > DFI Write Data Chip Select. This signal has two functions and is defined similar to the **dfi_cs_n** signal.
   >
   > During write leveling, **dfi_wrdata_cs_n** indicates which chip select is currently active.
   >
   > During non-leveling operation, **dfi_wrdata_cs_n** is an optional signal for the MC to indicate which chip select is accessed or targeted for associated write data.

   .DFI 4.0 (proposed):

   > DFI Write Data Chip Select, This signal has two functions and is defined similar to the **dfi_cs** signal.
   >
   > During write leveling, **dfi_wrdata_cs** indicates the chip select that is currently active.
   >
   > During non-leveling operation, **dfi_wrdata_cs** indicates the chip select that is accessed or targeted for associated write data.

2. Existing and proposed descriptions of **dfi_rddata_cs** (Table 9) are as follows:

   DFI 3.1 (existing):

   > DFI Read Data Chip Select. This signal has two functions.
   >
   > During read training, **dfi_rddata_cs_n** indicates which chip select is currently active.
   >
   > During non-leveling operation, **dfi_rddata_cs_n** is an optional signal for the MC to indicate which chip select is accessed or targeted for associated read data.

   .DFI 4.0 (proposed):

   > DFI Read Data Chip Select. This signal has two functions.
   >
   > During read training, **dfi_rddata_cs** indicates the chip select that is currently active.
   >
   > During non-leveling operation, **dfi_rddata_cs** indicates the chip select that is accessed or targeted for associated read data.

Additional information for the tables as follows:

- Signal: No new signals.
- Width: No new signals.
- Initial value: NA.
- Parameter: NA.
- Timing:

8.3 Compatibility with Older Version of DFI

- DFI 4.0 PHY can be used with DFI 4.0 controller or with DFI 3.x controller if these optional signals are present on that controller.
- DFI 4.0 MC can be connected to DFI 4.0 as well as to DFI 3.x PHY.

8.4 Compliance

The interface is relevant to MCs and PHYs that support the DFI 4.0. The new signals are required for the MC and optional for the PHY. If the PHY does not have these signals, MC signals are unused.

9.0 Write Leveling Strobe Update

9.1 Background

This chapter provides additional details for the DFI write leveling interface for allowing the write leveling strobes to be asserted for multiple cycles during write levelling. This is used in the PHY to generate multiple, back to back DQS strobe outputs.

9.2 Detailed Description

The goal of write leveling is locating the delay at which the write DQS rising edge aligns with the rising edge of the memory clock. By identifying this delay, the system can accurately align the write DQS with the memory clock.

To allow two or more DQS pulses to be driven to the DRAM, during DFI write leveling, **dfi_wrlvl_strobe** can be driven for multiple clocks. For DFI4.0, each cycle that the **dfi_wrlvl_strobe** signal is driven generates a single DQS pulse. To support frequency ratio, the **dfi_wrlvl_strobe** signal is a phased signal. The DFI 4.0 signals, timing parameters, and system parameter changes are as follows in Table 12, Table 13 and Table 14.

TABLE 12. *DFI Signal*

DFI Signal	Driven By	Width	Description
dfi_wrlvl_strobe_pN	MC	NUM_SLICES	Write leveling strobe. Asserts for one or more DFI clocks; initiates the capture of the write level response from the DQ bus within the PHY. The number of DFI clocks the write leveling strobe is asserted per write leveling request [a] is specified by the $\mathbf{SYS_{wrlvl\_strobe\_num}}$ parameter. specifies.

a. The minimum supportable value is 1; the DFI does not specify a maximum value. The range of values supported is implementation-specific.

TABLE 13. *Modified Timing Parameter*

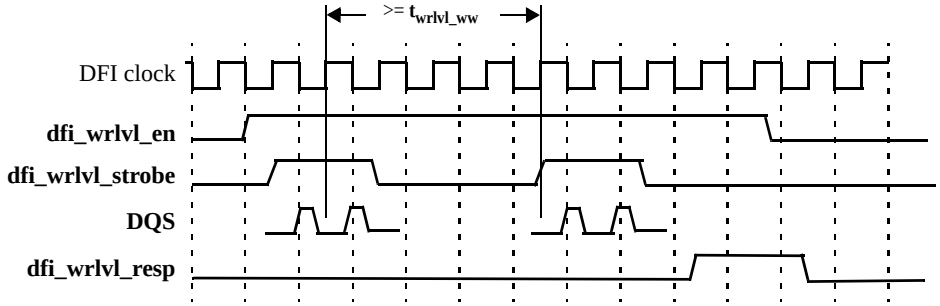
Timing Parameter	Defined By	Description
$\mathbf{t_{wrlvl\_ww}}$	PHY	Write leveling write-to-write delay. During a write leveling training request, **twrlvl_ww** specifies the number of DFI clock cycles after the **dfi_wrlvl_strobe** signal is asserted, to the next assertion of the **dfi_wrlvl_strobe** signal that a subsequent training request generates. The delay begins at the last rising clock edge for which **dfi_wrlvl_strobe** signal is asserted

TABLE 14. *New Parameter for LPDDR4 write leveling*

Parameter	Defined By	Description
$\mathbf{SYS_{wrlvl\_strobe\_num}}$	MC	The minimum number clocks that the **dfi_wrlvl_strobe** signal is asserted during a write leveling training request. [a]

a. The minimum supportable value is 1; the DFI does not specify a maximum value. The range of values supported is implementation-specific.

FIGURE 19. *DFI Write Leveling Interface with LPDDR4 Support*



9.3 Compatibility with Older Version of DFI

A DFI 4.0 compliant PHY that implements the write leveling enhancement interface will not support DFI 4.0 write leveling with an older DFI version of the MC. The assumption is that $\mathbf{SYS_{wrlvl\_strobe\_num}}$ is set to 1. A DFI 4.0 MC interfacing to an older DFI version PHY will set $\mathbf{SYS_{wrlvl\_strobe\_num}}$ to 1. Consequently, DFI 4.0 write leveling will not be supported.

9.4 Compliance

The write leveling interface signals are relevant to MCs and PHYs that support write leveling in PHY evaluation mode training. MCs and PHYs that support only PHY independent mode do not need to include this interface.

10.0 WR DQ Training

10.1 Background

This chapter describes the additional signals and timing parameters required for write DQ data-eye training.

10.2 Detailed Description

DFI 4.0 adds a write DQ training interface to DFI. Using this interface, the write DQ training involves the following steps:

1. The MC or PHY initiates training.
2. The MC issues write bursts, followed by read bursts. Commands either access a scratch pad FIFO within the DRAM (LPDDR4) or the DRAM array. While the MC controls data timing with **dfi_wrdata_en** and **dfi_rddata_en** signals, the data is generated and verified internally in the PHY.

   If necessary, the write to write command timing can be modified to add delay between write commands by using the $\mathbf{t_{wdqlvl\_ww}}$ timing parameter. This parameter defines a minimum number of DFI clocks to be inserted between write commands.
3. The PHY transfers the commands to the DRAM, generating the required data on the DQ outputs for the write that is based on **dfi_wrdata_en** timing.
4. The PHY receives the read data from the DRAM and compares it to the expected data. The PHY delays are to be adjusted as needed.
5. The sequence, starting from step 2 repeats until the PHY indicates that the training is complete.

Because write DQ training might require iterating through multiple variables, including VREF values, write data patterns, delay settings, etc.; sequence control might be required from both the MC and the PHY. To facilitate this requirement, **dfi_wdqlvl_done** is defined, which is to indicate when the MC has set the final VREF value associated with the training sequence. For example, in write DQ training, the indication might be the final VREF value. If training is complete at the end of the write DQ training sequence in which the MC asserts **dfi_wdqlvl_done**, the PHY must assert **dfi_wdqlvl_resp** = 'b11 as the response at the end of the sequence. Alternatively, if the PHY requires additional variables, the PHY must assert **dfi_wdqlvl_resp** = 'b01 and the training routine continues. In this case, the MC should loop back to the first VREF value that was driven at the beginning of training, and iterate through the VREF values again. The MC is to continue looping through the values until the PHY responds with **dfi_wdqlvl_resp** = 'b11 and training is complete.

The results for a specific VREF value would be useful to the MC when selecting the next VREF. To facilitate communication of this information on the interface, **dfi_wdqlvl_result** is defined, which indicates how the results of the current value compare with the best previous value, either better or worse. The evaluation process continues until the PHY asserts **dfi_wdqlvl_resp** = 'b11, at which point, the evaluation is complete and subsequent training sequences start a new evaluation process.

Table 15 defines the new signals being added to DFI for write DQ training. This table includes signals already defined in DFI that are to be re-used for write DQ training. The full command interface is included and used for data commands (WR, RD, MPC), bank operations, and potentially maintenance operations, such as auto-refresh.

TABLE 15. *New Signals Added to DFI or Requiring Information Revision for Write DQ Training*

Signal	Driven By	Description
dfi_phy_wdqlvl_cs	PHY	Target chip select associated with the **dfi_wdqlvl_req** signal
dfi_wdqlvl_en	MC	Asserted during write DQ training; used for enabling training in PHY and acknowledging a PHY training request on the **dfi_wdqlvl_req** signal
dfi_wdqlvl_req	PHY	PHY-initiated write DQ training request
dfi_wdqlvl_resp	PHY	Training response from PHY. The following table shows the encoding. • 'b00 Not Done • 'b01 Done with current VREF sequence. MC should update VREF value if applicable. If dfi_wdqlvl_done asserted, MC should restart from initial value. • 'b10 RESERVED • 'b11 Training complete When training is incomplete and the PHY responds with 'b01, the MC performs the following actions: **1.** De-asserts **dfi_wdqlvl_en**. **2.** Sets up the next VREF value. **3.** Re-asserts **dfi_wdqlvl_en**.
dfi_wdqlvl_result	PHY	Training result from PHY. Indicates how the current VREF value being driven by the MC compares with all previous values. 0: Current value is worse than the previous best value. 1: Current value is better than the previous best value. The **dfi_wdqlvl_result** signal is valid when **dfi_wdqvl_resp** is asserted (non-ZERO) and should be unchanged when **dfi_wdqvl_resp** is asserted; not valid when **dfi_wdqvl_resp** = 'b00.
dfi_wdqlvl_done	MC	The MC completes iterating across different VREF training values. If the PHY completes the training sequence with **dfi_wdqlvl_resp** = 'b11, training is complete. If the PHY completes the training sequence with **dfi_wdqlvl_resp** = 'b01, training continues and the MC should restart with the beginning VREF value and drive **dfi_wdqlvl_done** accordingly. The **dfi_wdqlvl_done** signal is valid during **dfi_wdqlvl_en** and should be unchanged during **dfi_wdqlvl_en**. The signal is invalid when **dfi_wdqlvl_en** = 'b0.
dfi_wrdata_cs	MC	Chip select that corresponds to the write DQ training commands. This bus is re-used from DFI 3.1
dfi_wrdata_en	MC	Write data enable, defining duration and timing of write data transfer relevant to the command. The **dfi_wrdata_en** signal is to be asserted $\mathbf{t_{phy\_wrlat}}$ cycles after the write command, similar to non-training write commands.

TABLE 15. *New Signals Added to DFI or Requiring Information Revision for Write DQ Training (Continued)*

Signal	Driven By	Description
dfi_rddata_en	MC	Read data enable, defining duration and timing of read data transfer and capture relevant to the command. The **dfi_rddata_en** signal is to be asserted $\mathbf{t_{rddata\_en}}$ cycles after the read command, similar to non-training write commands.
dfi_cs, dfi_address, dfi_bank, **dfi_bank_group**, **dfi_ras_n, dfi_cas_n**, **dfi_we_n, dfi_act_n**	MC	Full command interface that is be used for transferring all or a subset of the following commands: • Data commands - RD, WR - MPC (LPDDR4) • Required Bank Operations • Maintenance operations like auto-refresh.

NOTE: The dfi_wrdata bus is not used during write DQ training. The DRAM write data is generated in the PHY.

When training completes, training values controlled by the MC (VREF for example) and the PHY (delay settings) must be in sync so that the DRAM VREF and PHY delays are set to final, trained values.

Table 16 summarizes parameters that are needed for supporting write DQ training.

TABLE 16. *New Parameters Needed to Support Write DQ Training*

Timing Parameter	Defined By	Description
$\mathbf{t_{wdqlvl\_en}}$	PHY	Minimum number of DFI clocks from **dfi_wdqlvl_en** to the first write command (with **dfi_cs** asserted)
$\mathbf{t_{wdqlvl\_max}}$	MC	Maximum number of DFI clocks that the MC is to wait for a response (**dfi_wdqlvl_resp**) after **dfi_wdqlvl_en** is asserted
$\mathbf{t_{wdqlvl\_resp}}$	MC	Maximum number of cycles after **dfi_wdqlvl_req** is asserted until the MC is to respond with **dfi_wdqlvl_en**
$\mathbf{t_{wdqlvl\_rw}}$	PHY	Minimum numbers of DFI clocks from the last read in a calibration sequence to the first write in the next set of calibration commands. NOTE: The controller is to issue read commands that correspond to the write commands for the PHY to check the success of the writes. The timing between the write and read commands are to be based on DRAM requirements. No DFI parameters are defined.
$\mathbf{t_{wdqlvl\_ww}}$	PHY	Minimum number of DFI clocks to be inserted between write commands during write DQ training. This timing parameter is applicable only when $\mathbf{phy_{wdqlvl\_bst}} > 1$.

TABLE 16. *New Parameters Needed to Support Write DQ Training (Continued)*

Timing Parameter	Defined By	Description
$\mathbf{phy_{wdqlvl\_bst}}$	PHY	Number of consecutive write bursts that are issued during a training sequence. These are to be followed by the same number of consecutive read bursts for completing the training sequence. NOTE: The timing of the read and write bursts within a sequence defaults to back-to-back commands. No DFI parameter is defined.

Figure 20, Figure 21, and Figure 22 illustrate the new Write DQ training DFI bus.

FIGURE 20. *WR DQ Training with phy_{wdqlvl_bst} = 1*

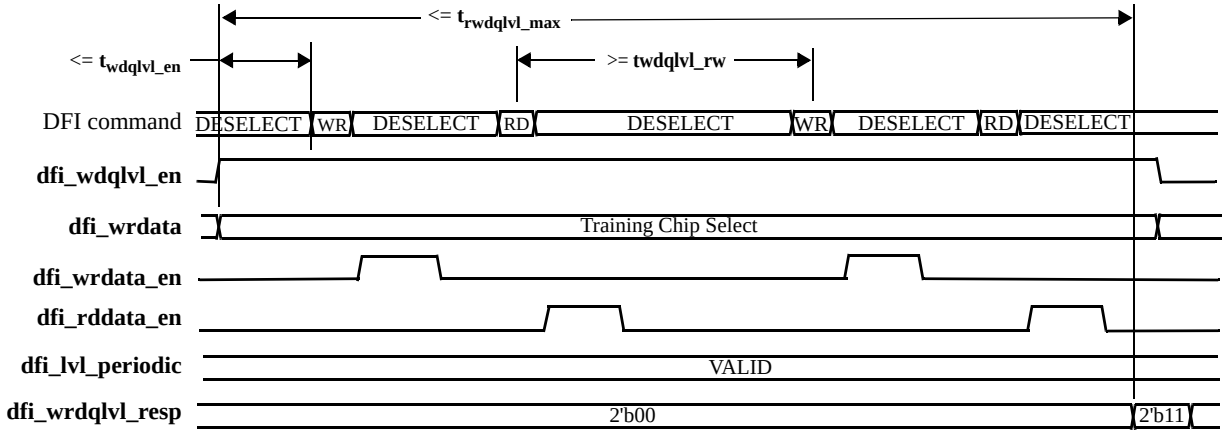


FIGURE 21. *WR DQ Training with phy_{wdqlvl_bst} = 2*

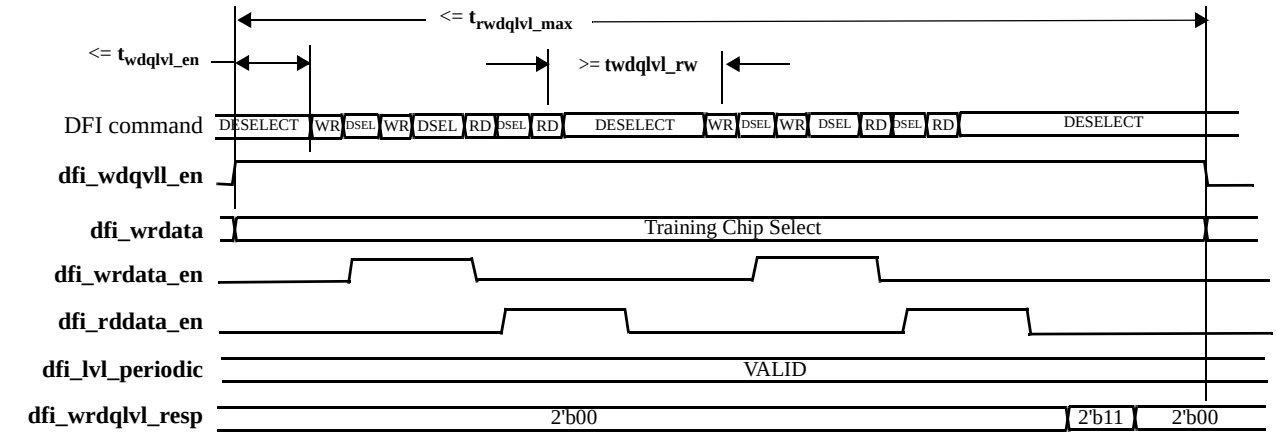
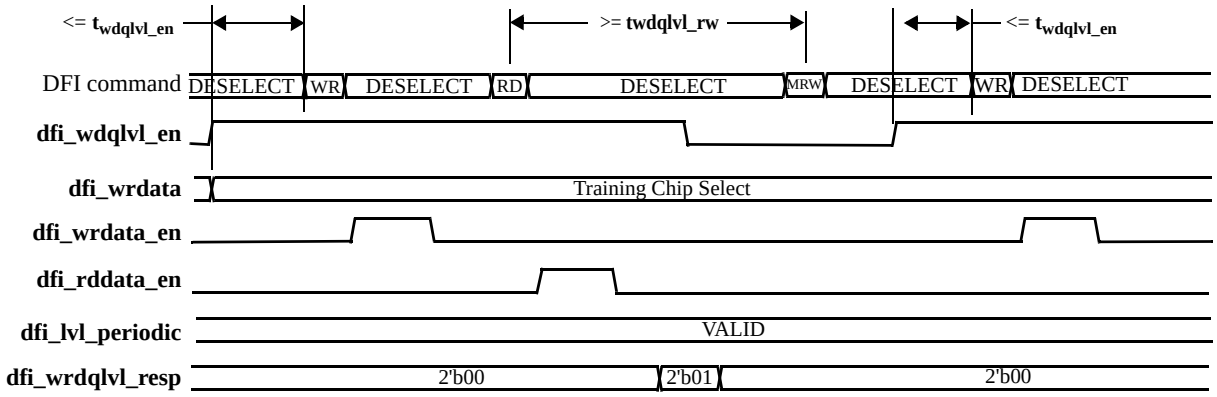


Figure 22, “WR DQ Training with phy_{wdqlvl_bst} = 1, dfi_wdqlvl_resp = 2'b10” illustrates write DQ training with **phy_{wdqlvl_bst}** = 1, **dfi_wdqlvl_resp** = 2'b01, and an MRW issued to set new VREF value by MC for the next sequence.

FIGURE 22. *WR DQ Training with phy_{wdqlvl_bst} = 1, dfi_wdqlvl_resp = 2'b10*



NOTE: While the MC encodes the MRW command on the **dfi_address** bus for the next VREF value, the PHY can disregard the controller VREF setting in the DFI command. It can set the appropriate address bits to an alternate value.

10.3 Compatibility with Older Version of DFI

Write DQ training is a new interface for DFI 4.0; there are no compatibility issues with older versions. The interface is unused if both the MC and PHY are not DFI 4.0 devices.

10.4 Compliance

The proposed changes are relevant to DFI 4.0 only. This chapter defines changes that are required with “DFI Training Mode”. No changes are required for PHY Independent training or software-based training, “Non-DFI Mode”.

11.0 PHY Master Interface

11.1 Background

The DFI 3.1 specification supports “DFI Requested training in non-DFI training mode”. A new interface is defined so that the PHY can use the new interface for any function instead of limiting the PHY's use to only training operation. In addition, the interface allows DRAM to be placed in various low power states and use new capabilities of LPDDR4 memory.

Existing DFI subsection (3.6.6) is removed, replaced by a new DFI section (3.9) named “PHY Master Interface”.

11.2 Detailed Description

The PHY uses the PHY Master Interface for requesting that the controller relinquish the DFI bus and take the control of the DRAM bus, with the DRAM in a defined state. When the PHY has control of the DFI bus, it performs all operations independent of the MC with or without accessing DRAM. The PHY requests the bus, and the MC puts the DRAM into the state that the PHY indicates, and grants the PHY control of the bus.

The PHY can request that memory be placed in a specific state (such as IDLE or self refresh), or the PHY can indicate that the MC is permitted to transfer memory in any of the supported states. The DFI bus must remain idle while the PHY has the control; with one exception that during this mode. The MC may periodically send memory refreshes on the DFI bus as required by the memory, based on the memory state when the MC acknowledged the PHY Master request. The MC maintains the memory refresh timing for the system. The PHY forwards all refreshes that it receives while the interface is active to memory before it releases the DFI bus back to the MC.

Table 17 defines the new interface signals.

TABLE 17. *New Signal Description*

Signal	From	Width	Default	Description
dfi_phymstr_req	PHY	1	'b0	DFI PHY Master Request: When asserted, the PHY requests control of the DFI bus.
dfi_phymstr_ack	MC	1	'b0	DFI PHY Master Acknowledge: When asserted, the MC places the DRAM in a known state: IDLE, self refresh. or self refresh power-down. When the **dfi_phymstr_ack** signal is asserted, the PHY is the master of DRAM bus. If required by the DRAM, the controller continues sending refresh commands on the DFI bus.

TABLE 17. *New Signal Description (Continued)*

Signal	From	Width	Default	Description
dfi_phymstr_cs_state	PHY	DFI Chip Select Width	'b0	DFI PHY Master CS State: This signal indicates the state of the DRAM when the PHY becomes the master. Each memory rank uses one bit. 'b0: IDLE or self refresh. The PHY specifies the required state, using the **dfi_phymstr_state_sel** signal. For LPDDR4, the self refresh state is without power-down. 'b0: The PHY specifies the required state, using the **dfi_phymstr_state_sel** signal. 'b1: IDLE or self refresh or self refresh with power-down. The PHY does not specify the state; the MC can optionally choose any supported state. 'b1: The PHY does not specify the state; the MC can optionally choose. The MC closes all the pages. This signal is valid only when the **dfi_phymstr_req** signal is asserted by the PHY and should remain constant while the **dfi_phymstr_req** signal is asserted. The self refresh with power-down state is specific to LPDDR4. The **dfi_phymstr_cs_state** bit values are not relevant for chip selects with $\mathbf{sys_{cs\_state}}$ set to 'b0 (inactive chip selects). The MC can leave the chip selects with $\mathbf{sys_{cs\_state}}$ set to 'b0 in their current, inactive state, regardless of the corresponding **dfi_phymstr_cs_state** bit value. The PHY must not require these chip selects to be in IDLE or self refresh states. The system must maintain a consistent, stable view of $\mathbf{sys_{cs\_state}}$ after **dfi_phymstr_req** is asserted to ensure synchronization between the MC and PHY.

TABLE 17. *New Signal Description (Continued)*

Signal	From	Width	Default	Description
dfi_phymstr_state_sel	PHY	1	'b0	DFI PHY Master State Select: Indication from the PHY to the MC whether the requested memory state is IDLE or self refresh. When **dfi_phymstr_cs_state**=0, setting **dfi_phymstr_state_sel**=0 indicates that the corresponding CS must be put into the IDLE state. When **dfi_phymstr_cs_state**l=0, setting **dfi_phymstr_state_sel**=1 indicates that the corresponding CS must be put into the self refresh state. For LPDDR4 devices, this is self refresh without power-down. When **dfi_phymstr_cs_state**=1, this signal does not apply. 0: If dfi_phymstr_cs_state = 0 (per CS from the PHY), the MC must place the memory on the associated CS in the IDLE state. 1: If **dfi_phymstr_cs_state** = 0 (per CS from the PHY), the MC must place the memory on the associated CS in the self refresh state. If using LPDDR4 devices, the self refresh state is without power-down. For chip selects where **dfi_phymstr_cs_state** = 1, the PHY does not place any requirement on the low power state of the memory, the state may be IDLE, self refresh, or self refresh with power-down. This signal is valid only when the **dfi_phymstr_req** signal is asserted by the PHY and should remain constant while the **dfi_phymstr_req** signal is asserted.
dfi_phymstr_type	PHY	2-bits	'b00	DFI PHY Master Type: Indicates which one of the 4 types of PHY master interface times that the **dfi_phymstr_req** signal is requesting. The value of the **dfi_phymstr_type** signal determines which one of the timing parameters (**tphymstr_type0**, **tphymstr_type1**, **tphymstr_type2**, **tphymstr_type3**) is relevant. The **dfi_phymstr_type** signal must remain constant during the entire time that the **dfi_phymstr_req** signal is asserted.

The PHY Master interface enables the MC to grant control of the DFI bus to the PHY by using the following sequence:

1. The PHY asserts the **dfi_phymstr_req** signal. The PHY drives the **dfi_phymstr_cs_state** and **dfi_phymstr_state_sel** to indicate to the MC the state of the memory required before handing off the bus. The PHY drives **dfi_phymstr_type** to indicate to the MC the length of the command.
2. The MC places the DRAM in a state that is based on the **dfi_phymstr_cs_state** and **dfi_phymstr_state_sel** signals.
3. The MC asserts **dfi_phymstr_ack** within $\mathbf{t_{phymstr\_max}}$ cycles.
4. The MC de-asserts the **dfi_phymstr_ack** signal upon sampling the **dfi_phymstr_req** signal low.

   If the PHY does not de-assert the **dfi_phymstr_req** signal within $\mathbf{t_{phymstr\_resp}}$ cycles following the assertion of the **dfi_phymstr_ack** signal, a DFI protocol violation occurs.
5. The PHY must return control of the DFI bus to the MC with DRAM in the identical state that it received it, including the state of memory.Also, all memory pages must be closed, and the refresh count must correspond with the MC refresh count.

To meet DRAM refresh timing, the MC and the PHY perform the following steps:

1. The MC performs following actions:

a. The MC sends refresh commands as required on the DFI bus. When the PHY is the master, refresh commands are the only commands the MC can send.

b. The MC must maintain all refresh timing requirements relative to a refresh command that is issued during PHY master mode.

c. The MC takes into account the PHY delay (**$t_{phymstr\_rfsh}$** parameter) for meeting all DRAM refresh timing.

NOTE: If the MC passes the memory to the PHY in the self refresh state, no refreshes are issued during the PHY master mode.

2. The PHY performs following actions:

a. The PHY generates a refresh command to the DRAM within **$t_{phymstr\_rfsh}$** cycles.

b. The PHY ensures that all DRAM refresh timings are met.

c. The PHY ensures that all DRAM refresh timings have expired before it returns control to the MC

Table 18 defines the new parameters.

TABLE 18. *New Parameter Description*

Parameter	Defined By	Min	Max	Description
$t_{phymstr\_resp}$	MC	1	_[a]	Specifies the maximum number of DFI clock cycles after the dfi **phymstr_req** signal asserts to the assertion of the **dfi_phymstr_ack** signal.
$t_{phymstr\_rfsh}$	PHY	1	_[a]	Specifies the maximum number of DFI clock cycles that the PHY takes for generating a refresh command to the DRAM after the PHY receives the refresh command from the MC.
$t_{phymstr\_type0}$	PHY	1	_[a]	Specifies the maximum number of DFI clock cycles that the **dfi_phymstr_req** signal may remain asserted after the assertion of the **dfi_phymstr_ack** signal for **dfi_phymstr_type** = 0x0. The **dfi_phymstr_req** signal may de-assert at any cycle after the assertion of the **dfi_phymstr_ack** signal.
$t_{phymstr\_type1}$	PHY	1	_[a]	Specifies the maximum number of DFI clock cycles that the **dfi_phymstr_req** signal may remain asserted after the assertion of the **dfi_phymstr_ack** signal for **dfi_phymstr_type** = 0x1. The **dfi_phymstr_req** signal may de-assert at any cycle after the assertion of the **dfi_phymstr_ack** signal.
$t_{phymstr\_type2}$	PHY	1	_[a]	Specifies the maximum number of DFI clock cycles that the **dfi_phymstr_req** signal may remain asserted after the assertion of the **dfi_phymstr_ack** signal for **dfi_phymstr_type** = 0x2. The **dfi_phymstr_req** signal may de-assert at any cycle after the assertion of the **dfi_phymstr_ack** signal.
$t_{phymstr\_type3}$	PHY	1	_[a]	Specifies the maximum number of DFI clock cycles that the **dfi_phymstr_req** signal may remain asserted after the assertion of the **dfi_phymstr_ack** signal for **dfi_phymstr_type** = 0x3. The **dfi_phymstr_req** signal may de-assert at any cycle after the assertion of the **dfi_phymstr_ack** signal.

a. The actual value depends on the system.

Figure 23 illustrates the new master interface timing.

FIGURE 23. *Master Interface Timing*

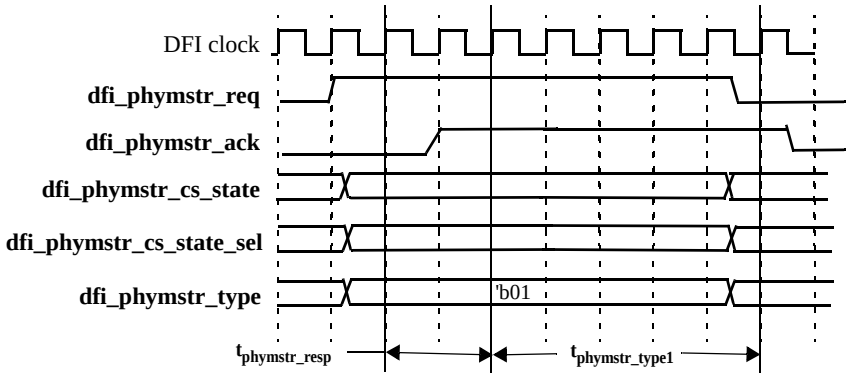
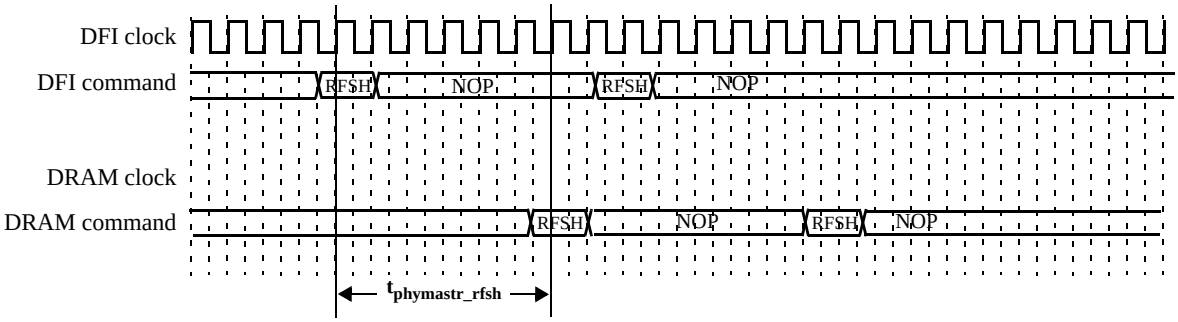


Figure 24 illustrates the new master refresh timing.

FIGURE 24. *Master Refresh Timing*



11.3 Compatibility with Older Version of DFI

When connecting a DFI 4.0 PHY with a DFI 3.1 MC, the **dfi_phymstr_req** from the PHY can be connected to the **dfi_phylvl_req_cs_n** input of the MC if the system designer is careful about polarity and width.

When a DFI 4.0 MC is connected to DFI 3.1 PHY, the **dfi_phy_req_cs** from PHY can be connected to **dfi_phymstr_req** input of the MC if the system designer is careful about polarity and width.

This interface cannot be supported when one of the devices (PHY or MC) supports DFI 3.0 or lower version. In this case, **dfi_phymstr_req** input of DFI 4.0 MC must be tied low, or **dfi_phymstr_ack** input of DFI 4.0 PHY must be tied high.

11.4 Compliance

The interface is relevant to MCs and PHYs that support the DFI 4.0. The new signals are required for the MC and optional for the PHY.

12.0 Frequency Indicator

12.1 Background

A new frequency select signal is added to the interface driven from the MC to the PHY for indicating the new frequency that is selected during a frequency change operation. Adding this signal allows the PHY to have necessary information about the target frequency. Currently, the interface does provide a signaling handshake for changing frequencies, but the PHY has no information about the target frequency that the PHY might need for executing the operation. The current frequency change protocol does not change; the additional signal is to work with the existing protocol.

12.2 Detailed Description

The **dfi_frequency** signal is an encoded frequency. The encoding is defined by the PHY, the system, or both, DFI does not impose any fixed frequency mapping. Frequencies should be mapped so that frequencies increase with increasing values of the **dfi_frequency** signal. Doing so allows the devices to set thresholds as necessary. The MC width is 5 bits, encoding 32 unique frequencies.

No initial value is defined for **dfi_frequency**. The system is to determine the initial frequency. The **dfi_frequency** signal is to be set to the initial value when the MC asserts the **dfi_init_start** signal during initialization. Similar to existing signals, the PHY might wait for the **dfi_init_start** signal to assert. Or if the PHY has no dependencies on initial values of these signals, it might initialize without a **dfi_init_start** signal, including assertion of a **dfi_init_complete** signal. The MC must adhere to defined functionality of **dfi_init_start** as previously defined.

Table 19 shows the signal definition to be added to the Status Interface Signals table. Additional information about this signal is defined and included in either the description or in text in the related sections.

TABLE 19. *DFI Frequency Indicator Signal*

Signal	From	Width	Default	Description
dfi_frequency	MC	5	–[a]	DFI frequency. This signal indicates the operating frequency for the system. This signal should change only at initialization, during a DFI frequency change operation, or other times that the system defines. tem; This signal should be constant during normal operation. The number of supported frequencies and the mapping of signal values to clock frequencies are defined by the PHY, system, or both.

a. a.The frequency at initialization is defined by the system.

The **Phy$_{freq_range}$** parameter defines the number of frequencies that the PHY supports. The frequency range must be a number between 1 and 32, inclusive, and must start from the value of ZERO to the defined range. The PHY may only support a subset of these frequencies; if so, the PHY should clearly define supported encodings. Requiring the encodings to increase with increasing frequencies might be useful in setting simple decoding of threshold values. The MC must be able to support the full range of values as defined by DFI. The MC must never drive a value outside the selected range nor any value within the range that is defined as unsupported by the PHY.

The parameter definition that Table 20 shows is to be added to the Status Programmable Parameters table. Additional information about this signal is defined and included in either the description or in text in the related sections.

TABLE 20. *DFI Frequency Indicator Programmable Parameter*

Parameter	Defined By	Description
$\mathbf{phy_{freq\_range}}$	PHY	PHY Frequency Range. Defines the range of frequency values supported by the PHY. The frequency range must be a number between 1 and 32 inclusive and must start from the value of ZERO to the defined range. The PHY may only support a subset of these frequencies. The PHY must clearly define supported encodings.

For timing, the **dfi_frequency** signal must be set to a legal value and remain unchanged when **dfi_init_start** is asserted. The **dfi_frequency** signal can change any time when **dfi_init_start** is low and should be ignored at this time. The definitions of **dfi_init_start**, **dfi_init_complete**, $\mathbf{t_{init\_complete}}$, and $\mathbf{t_{init\_start}}$ are unchanged.

Figure 25 illustrates the **dfi_frequency** initialization.

FIGURE 25. *dfi_frequency Initialization*

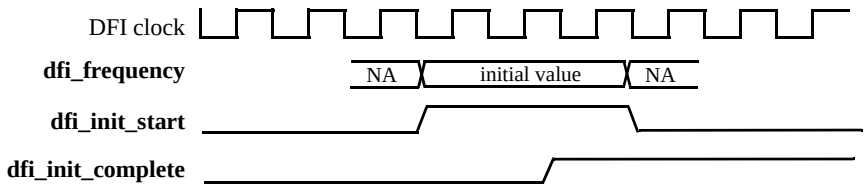
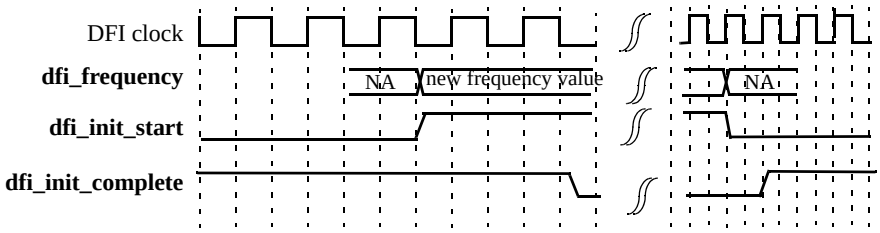


Figure 26 illustrates the **dfi_frequency** during a frequency change.

FIGURE 26. *dfi_frequency During a Frequency Change*



12.3 Compatibility with Older Version of DFI

For compatibility, a DFI 4.0-compliant PHY that implements the new frequency select signaling must operate with an MC that does not drive the signal. A DFI 4.0 MC is to operate with a PHY that does not support this signal regardless of DFI version. The signal can be unconnected if the PHY does not use it.

12.4 Compliance

The signal is relevant to MCs and PHYs that support the DFI Frequency Change feature. The new signal is optional for the PHY. The new signal is required for the MC if it is DFI 4.0 compliant and also if the MC supports the DFI Frequency Change feature.

13.0 DFI Disconnect Protocol

13.1 Background

DFI defines several interface handshakes between the MC and the PHY. In some cases, the interface handshake can be terminated by only one of the devices, and the disconnect timing might be long or indeterminate. In some circumstances, it may be desirable for a device to disconnect the handshake. The disconnect might be required for a high-priority operation that the system must continue to be fully operational. A disconnect might be required for responding to an error condition even if the system stops being fully operational.

As an example of a high priority operation, the memory system can have a quality of service (QOS) requirement that might be violated if the interface is not disconnected in a reasonable time. As example of an error, the system might encounter a condition that requires interruption of the normal memory sub-system operation and placing memory into self refresh. This action might be required even if the interruption has detrimental effects on the current operating state of the memory sub-system. The desired outcome of disconnecting is a relatively short and predictable disconnect time and also a defined system state for the error condition.

13.2 Detailed Description

The current interface handshakes are as follows:

- Update
  - Controller update
  - PHY update
- PHY Master Interface
- Training in PHY evaluation mode
  - CA training
  - Read data eye training
  - Read gate training
  - Write leveling
- Frequency change
- Low power

The disconnect protocol requires modification of some existing signal definitions. A new signal must distinguish between the two following conditions:

- The system remains fully operational (QOS).
- The system is no longer guaranteed to be operational (Error).

A new timing parameter is to be associated with each disconnect and defined by the disconnected device. The disconnect state is defined by the disconnected device.

13.2.1 DFI Update Interface

Two update handshakes are the controller update interface (**dfi_ctrlupd_req**, **dfi_ctrlupd_ack**) and the PHY update interface (**dfi_phyupd_req**, **dfi_phyupd_ack**). For both of these interfaces, when the request and acknowledge signals are asserted, only the PHY can disconnect the handshake. Although a timing parameter defines the maximum handshake time, the delay might be unacceptably long in some cases.

To indicate a disconnect, the MC de-asserts its handshake signal. For a controller update, the MC de-asserts the **dfi_ctrlupd_req** signal. For a PHY update, the MC de-asserts the **dfi_phyupd_ack** signal. The de-assertion of this signal indicates to the PHY that the MC intends to disconnect the handshake. For a controller update disconnect, the MC must assume that the PHY did not complete the requested operation, and that a subsequent request should be scheduled. For controller update, this requires meeting $\mathbf{t_{ctrlupd\_interval}}$ timing.

To indicate a QOS event, as opposed to an error condition event, the **dfi_disconnect_error** signal is to be asserted for the error condition and de-asserted for QOS. The **dfi_disconnect_error** signal must be valid on the same clock as the disconnect and remain unchanged until the disconnecting device completes the disconnection. The signal is not meaningful at any other time.

Table 21 shows the DFI disconnect error signal.

TABLE 21. *DFI Disconnect Error Signal*

Signal	From	Width	Default	Description
dfi_disconnect_error	MC	5	0	DFI Disconnect Error. Indicates if the current disconnect is an error or a QOS request. If de-asserted, the disconnect request requires that the PHY remain fully operational after the disconnect. If asserted, the PHY might not be fully operational after the disconnect.

For the QOS condition, the PHY must de-assert the corresponding signal—**dfi_ctrlupd_ack** or **dfi_phyupd_req**—within $\mathbf{t_{ctrlupd\_disconnect}}$ or $\mathbf{t_{phylupd\_disconnect}}$ clocks, respectively. After the QOS disconnects, the system must be fully operational.

For the error condition, the PHY must de-assert the corresponding signal—**dfi_ctrlupd_ack** or **dfi_phyupd_req**—within $\mathbf{t_{ctrlupd\_disconnect\_error}}$ or $\mathbf{t_{phylupd\_disconnect\_error}}$ clocks, respectively. The state of the PHY that follows the disconnect error condition is defined by the PHY and should be defined in the PHY specification. Ideally, the PHY disconnects on a boundary that preserves the most memory service possible.

NOTE: In previous versions of the DFI interface, the signaling that is defined for disconnect is an illegal operation.

Table 22 defines the new timing parameters.

TABLE 22. *Update Interface Disconnect Timing Parameters*

Parameter	Defined By	Min	Max	Description
$\mathbf{t_{ctrlupd\_disconnect}}$	PHY	0	–[a]	Defines the maximum number of clocks that are required to disconnect the PHY during a control update sequence, from the de-assertion of **dfi_ctrlupd_req** to the de-assertion of **dfi_ctrlupd_ack** when **dfi_disconnect_error** = 0.

TABLE 22. *Update Interface Disconnect Timing Parameters (Continued)*

Parameter	Defined By	Min	Max	Description
$t_{phylupd\_disconnect}$	PHY	0	–[a]	Defines the maximum number of clocks required to disconnect the PHY during a PHY update sequence, from the de-assertion of **dfi_phyupd_ack** to the de-assertion of **dfi_phyupd_req** when **dfi_disconnect_error** = 0.
$t_{ctrlupd\_disconnect\_error}$	PHY	0	–[a]	Defines the maximum number of clocks required to disconnect the PHY during a control update sequence, from the de-assertion of **dfi_ctrlupd_req** to the de-assertion of **dfi_ctrlupd_ack** when **dfi_disconnect_error** = 1.
$t_{phylupd\_disconnect\_error}$	PHY	0	–[a]	Defines the maximum number of clocks required to disconnect the PHY during a PHY update sequence, from the de-assertion of **dfi_phyupd_ack** to the de-assertion of **dfi_phyupd_req** when **dfi_disconnect_error** = 1.

a. The actual value depends on the system.

Figure 27 and Figure 28 illustrate the new disconnect signaling protocol.

FIGURE 27. *PHY Update QOS Disconnect Protocol, PHY Update*

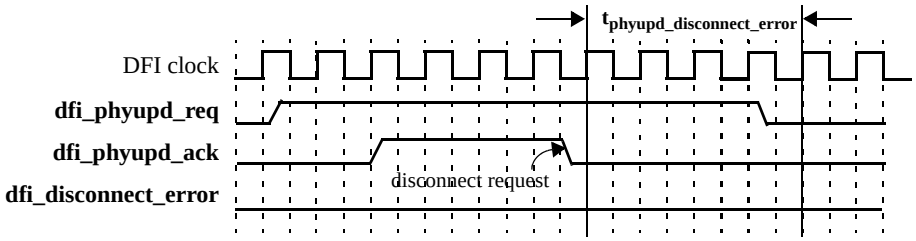
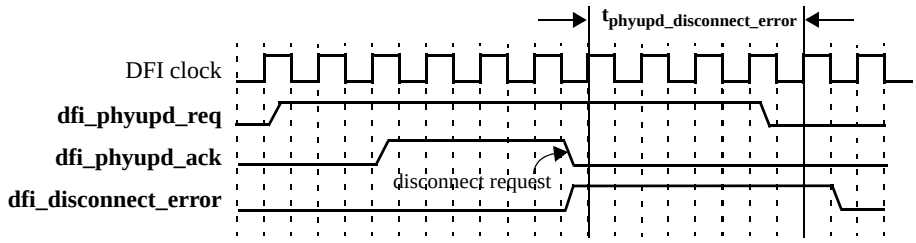


FIGURE 28. *PHY Update Error Disconnect Protocol, PHY Update*



13.2.2 PHY Master Interface

The PHY Master Interface defines a PHY request and MC acknowledge. When the MC asserts the acknowledge, only the PHY can disconnect the handshake, by de-asserting the request.

To indicate disconnect, the MC de-asserts the acknowledge signal. The de-assertion of this signal indicates to the PHY that the MC intends to disconnect the handshake. To indicate a QOS condition, as opposed to an error condition, the **dfi_disconnect_error** signal asserts for the error condition and de-asserts for the QOS condition.

For the QOS condition, the PHY must assert the corresponding request signal within the disconnect time ($t_{phymstr\_disconnect}$). The PHY after disconnect must be fully operational.

For the error condition, the PHY must assert the corresponding request signal within the disconnect time ($t_{phymstr\_disconnect\_error}$). The state of the PHY that follows disconnect is defined by the PHY and should be defined in the PHY specification. Ideally, the PHY disconnects on a boundary that preserves the most memory service possible.

NOTE: In previous versions of the DFI interface, the signaling that is defined for disconnect is an illegal operation.

Table 23 defines the new timing parameters.

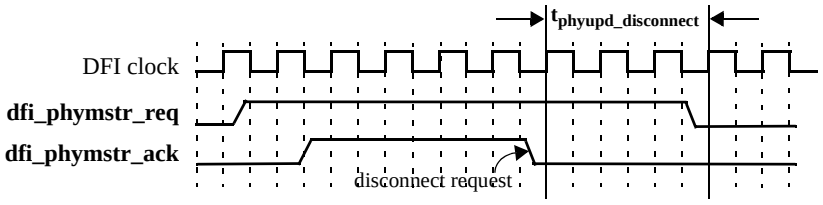
TABLE 23. *DFI Training Disconnect Error Timing Parameters*

Parameter	Defined By	Min	Max	Description
$t_{phymstr\_disconnect}$	PHY	0	–[a]	Defines the maximum number of clocks required to disconnect a PHY during a PHY master request from the de-assertion of **dfi_phymstr_req** to the de-assertion of **dfi_phymstr_ack** when **dfi_disconnect_error** = 0.
$t_{phymstr\_disconnect\_error}$	PHY	0	–[a]	Defines the maximum number of clocks required to disconnect a PHY during a PHY master request from the de-assertion of **dfi_phymstr_req** to the de-assertion of **dfi_phymstr_ack** when **dfi_disconnect_error** = 1.

a. The actual value depends on the system.

Figure 29 illustrates the new disconnect signaling protocol.

FIGURE 29. *PHY Master Interface Disconnect Protocol*



13.2.3 DFI Training Interface

PHY evaluation mode training handshakes include CA training, read data eye training, read gate training, write DQ training, and write leveling. The handshake, as it pertains to disconnect, is the same for each of the training interfaces. When the MC drives the training enable (**dfi_calvl_en**, **dfi_rdlvl_en**, **dfi_rdlvl_gate_en**, **dfi_wdqlvl_en**, **dfi_wrlvl_en**) signal, only the PHY can disconnect the handshake. It disconnects the handshake by driving the associated response (**dfi_calvl_resp**, **dfi_rdlvl_resp**, **dfi_wdqlvl_resp**, or **dfi_wrlvl_resp**).

To indicate disconnect, the MC de-asserts the training enable signal. The de-assertion of this signal indicates to the PHY that the MC intends to disconnect the handshake. The PHY responds to the disconnect by asserting a done response on the corresponding training done signal. For training responses with 2-bit response, only a training complete response (2'b11) indicates that training has terminated. Any other response requires training to continue. For training disconnect, the MC must assume that the PHY did not complete the requested training operation, and that a subsequent training request should be scheduled.

To indicate a QOS event, as opposed to an error condition event, the **dfi_disconnect_error** signal asserts for the error condition and de-asserts for QOS. The **dfi_disconnect_error** signal must be valid on the same clock as the disconnect and remain unchanged until the disconnecting device completes the disconnection. The signal is not meaningful at any other time.

For the QOS condition, the PHY must assert the corresponding response signal within the disconnect time (**$t_{calvl\_disconnect}$**, **$t_{rdlvl\_disconnect}$**, **$t_{rdlvl\_gate\_disconnect}$**, **$t_{wdqlvl\_disconnect}$**, **$t_{wrlvl\_disconnect}$**, respectively). After the QOS disconnects, the system must be fully operational. The state of system after the disconnect should be the same as if the state would be if not disconnected.

For the error condition, the PHY must de-assert the corresponding signal within the disconnect time (**$t_{calvl\_disconnect\_error}$**, **$t_{rdlvl\_disconnect\_error}$**, **$t_{rdlvl\_gate\_disconnect\_error}$**, **$t_{wdqlvl\_disconnect\_error}$**, **$t_{wrlvl\_disconnect\_error}$**, respectively). The state of the PHY that follows disconnect is to be defined by the PHY and should be defined in the PHY specification. Ideally, the PHY disconnects on a boundary that preserves the most memory service possible.

NOTE: In previous versions of the DFI interface, the signaling that is defined for disconnect is an illegal operation.

Table 24 defines the new timing parameters.

TABLE 24. *DFI Training Disconnect Error Timing Parameters*

Parameter	Defined By	Min	Max	Description
$t_{calvl\_disconnect}$	PHY	0	_[a]	Defines the maximum number of clocks required to disconnect the PHY during CA training, from the de-assertion of **dfi_calvl_en** to the assertion of **dfi_calvl_resp** when **dfi_disconnect_error** = 0.
$t_{rdlvl\_disconnect}$	PHY	0	_[a]	Defines the maximum number of clocks required to disconnect the PHY during read data eye training, from the de-assertion of **dfi_rdlvl_en** to the assertion of **dfi_rdlvl_resp** when **dfi_disconnect_error** = 0.
$t_{rdlvl\_gate\_disconnect}$	PHY	0	_[a]	Defines the maximum number of clocks required to disconnect the PHY during read gate training, from the de-assertion of **dfi_rdlvl_gate_en** to the assertion of **dfi_rdlvl_resp** when **dfi_disconnect_error** = 0.

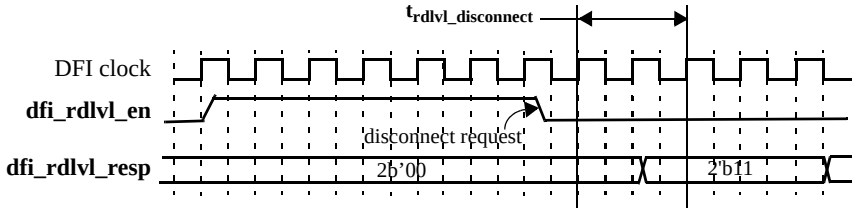
TABLE 24. *DFI Training Disconnect Error Timing Parameters (Continued)*

Parameter	Defined By	Min	Max	Description
$t_{wrlvl\_disconnect}$	PHY	0	_[a]	Defines the maximum number of clocks required to disconnect the PHY during write leveling, from the de-assertion of **dfi_wrlvl_en** to the assertion of **dfi_wrlvl_resp** when **dfi_disconnect_error** = 0.
$t_{calvl\_disconnect\_error}$	PHY	0	_[a]	Defines the maximum number of clocks required to disconnect the PHY during CA training, from the de-assertion of **dfi_calvl_en** to the assertion of **dfi_calvl_resp** when **dfi_disconnect_error** = 1.
$t_{rdlvl\_disconnect\_error}$	PHY	0	_[a]	Defines the maximum number of clocks required to disconnect the PHY during read data eye training, from the de-assertion of **dfi_rdlvl_en** to the assertion of **dfi_rdlvl_resp** when **dfi_disconnect_error** = 1.
$t_{rdlvl\_gate\_disconnect\_error}$	PHY	0	_[a]	Defines the maximum number of clocks required to disconnect the PHY during read gate training, from the de-assertion of **dfi_rdlvl_gate_en** to the assertion of **dfi_rdlvl_resp** when **dfi_disconnect_error** = 1.
$t_{wrlvl\_disconnect\_error}$	PHY	0	_[a]	Defines the maximum number of clocks required to disconnect the PHY during write leveling, from the de-assertion of **dfi_wrlvl_en** to the assertion of **dfi_wrlvl_resp** when **dfi_disconnect_error** = 1.

a. The actual value depends on the system.

Figure 30 illustrates the new disconnect signaling protocol.

FIGURE 30. *Training Disconnect Protocol, Read Data Eye Training*



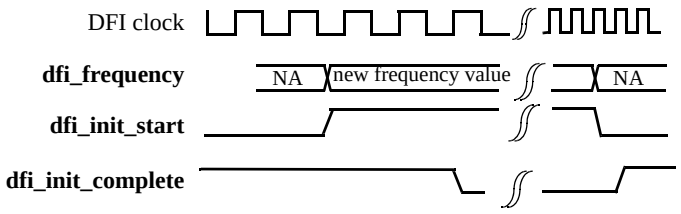
13.2.4 Frequency Change

NOTE: The disconnect protocol is not applicable to the frequency change interface.

The frequency change handshake involves the **dfi_init_start** and **dfi_init_complete** signals. When the PHY de-asserts **dfi_init_complete** and waits for the de-assertion of **dfi_init_start**, the PHY cannot disconnect the interface. And after the MC de-asserts **dfi_init_start** and waits for the assertion of **dfi_init_complete**, the MC cannot disconnect the interface. When the PHY is waiting, the clock frequency is transitioning, which is outside the scope of the DFI bus. When the MC is waiting, the PHY is locking its DLLs to the new clock frequency. Because both states (as described) should not be disconnected, the disconnect protocol does not apply to the frequency change handshake.

Figure 31 illustrates the frequency change.

FIGURE 31. *Frequency Change*



13.2.5 Low Power

NOTE: The disconnect protocol is not applicable to the low power interface.

The low power handshake involves the **dfi_lp_req** and **dfi_lp_ack** signals. When the PHY asserts the acknowledge, the PHY cannot disconnect the interface. The PHY is not required to stay in a low power state during a low- power operation and, because memory is in a low power state, there is no reason for the PHY to disconnect the handshake.

13.3 Compatibility with Older Version of DFI

An MC that uses the disconnect protocol with an earlier DFI version PHY violates the earlier protocol and produces unpredictable results. For a DFI 4.0 MC to work with earlier versions of the PHY, the MC should implement a method for disabling the disconnect protocol.

A PHY that works with an MC that is earlier than DFI 4.0 does not detect a disconnect, thus, no compatibility issues.

13.4 Compliance

DFI 4.0 PHYs must be able to tolerate a disconnect requests on all interfaces. Before DFI 4.0, the disconnect commands were illegal. However, the PHY is not required to disconnect, and can define the disconnect timing accordingly. A PHY can support disconnect on only a subset of the interfaces and not on others. The PHY can support either the error or QOS disconnect on an interface. However, if the PHY goes into an error state, it should not disconnect if **dfi_disconnect_error** = 0. The PHY should clearly define support for disconnect on all interfaces.

DFI 4.0 MCs are not required to issue disconnect commands.

14.0 DFI Data Bit Disable

14.1 Background

DFI 3.1 and earlier versions include the **dfi_data_byte_disable** signal. This signal can define, on a byte granularity, portions of the DFI data bus that are unused. However, the byte granularity is not fine enough to support all systems as shown in the following cases.

- The PHY has x4 slices for supporting x4 DRAM devices. In this case, a byte can be composed of two slices. It might be necessary to disable one of the x4 slices while leaving the other slices enabled.
- Architectures could have slices that are larger than the required DRAM data bus. For example, a system could support ECC with 7-bits of check data. With x8 slices, only 7 of the 8 bits are valid data to be transferred to DRAM. With x4 slices, only 3 of the 4 bits in the second x4 slice are valid data. In this case, DFI does not define the bits that are used for transferring valid data and ensuring that the PHY and MC are connected correctly.

In addition, the **dfi_data_byte_disable** signal is on the interface. However, for most applications, the signal is useful only during initialization for determining if the current configuration uses all or a sub-set of the slices. Upon determination, the signal is considered to be static. A system can support changing this signal. However, this event would need to occur when the interface is idle and would likely be a very infrequent event. Therefore, including this signal on the interface is unnecessary, and the signal is replaced by a DFI programmable parameter in DFI 4.0.

14.2 Detailed Description

In DFI 4.0, the signal **dfi_data_byte_disable** is removed from DFI, and the $\mathbf{dfi_{data\_bit\_enable}}$ parameter defines the valid data bits for both the MC and PHY on both the **dfi_wrdata** and **dfi_rddata** interfaces. The parameter must be the same for both read and write data. This parameter is defined by both the MC and the PHY.

These devices can have multiple operating modes where the value of the parameter might be different in the different modes. In this case, the device must define the parameter for each operating mode. For example, a device might operate with ECC on and ECC off. If so, the value of the parameter differs for both modes. If the parameter is defined the same for both the MC and PHY, they are compatible.

In some cases, the devices can define the parameter differently so that the devices can still operate together, and the differences are resolved at the bus interconnection of the devices. For example, in an ECC system, the two devices might place the ECC data in different locations within the data bus, which can be resolved by interconnecting the devices to the bus for aligning these signals as necessary.

Table 25 shows the parameter definition to be added to the Status Programmable Parameter table. The **dfi_data_byte_disable** signal is removed from the Status Interface Signals table in DFI 4.0.

TABLE 25. *New Programmable Parameter Description*

Parameter	Defined By	Width	Description
dfi$_{data_bit_enable}$	MC/PHY	DFI Data Width	Data Bit Enable. Defines the bits that are used for transferring valid data on both the **dfi_wrdata** and **dfi_rddata** busses. This parameter is defined by both the MC and the PHY. The parameter can have different values for different operating modes for each device.

14.3 Compatibility with Older Version of DFI

DFI 4.0-compliant PHY that operates with an older MC that drives the **dfi_data_byte_disable** signal should have no backward compatibility issue. The MC signal is not connected, and the PHY programmable register would be programmed according to the definition of the signal.

A DFI 4.0-compliant MC that implements the new DFI data bit disable parameter (and does not have the **dfi_data_byte_disable** signal), working with an older PHY that does use the **dfi_data_byte_disable** signal requires a programmable register that emulates the behavior of the **dfi_data_byte_disable** signal. The **dfi_data_byte_disable** input to the PHY is driven by external logic or hardwired, based on the PHY requirements and current mode.

14.4 Compliance

The current **dfi_data_byte_disable** signal is optional; thus, the new parameter would be optional. A device that does not define the parameter does not provide any applicable functionality regarding unused data signals on the interface.

15.0 Slice Parameter

15.1 Background

DFI 3.1 and earlier versions use the term “slice” to refer to a sub-set of the data interface. Terms include slice, data slice, PHY data slice, and memory data slice. No definition is given for the term slice in the specification. The term slice is generally used to define signal widths—one signal per data slice—to fan out a signal from the MC to PHY or define the number of signals that the PHY sends to the MC.

In the PHY, data slices are generally individual components of the PHY's data logic. These components generally operate independently; the interface is defined to minimize slice to slice communication. Furthermore, the interface replicates data control signaling to allow point-to-point connectivity between the MC and PHY.

From an interface perspective, all slices are expected to be identical. Some systems do not have a data width that aligns to a slice boundary. However, the alignment of data to the interface is expected to be defined by the MC and the PHY. The interface does not require that all data bits be connected. Furthermore, the DFI data bit disable parameter defines how the MC and PHY place data on the interface so that any alignment issues can be resolved, based on connectivity to the interface if required.

To more exactly quantify the interface requirements and to facilitate interoperability of devices, a new DFI slice width parameter is added to DFI 4.0. This parameter is defined by the PHY and defines the width of a common slice component. It is acceptable for the PHY to have slices that are less than the defined slice width. In this scenario, the DFI data bit disables would be utilized to define which bits are not available.

15.2 Detailed Description

The PHY is to define the slice width as Table 26 shows.

TABLE 26. *DFI Slice Programmable Parameter Definition*

Parameter	Defined By	Description
phy$_{\text{data_slice_width}}$	PHY	PHY Data Slice Width. Defines the width of a common PHY data slice component. All slices are defined having the same width. Unused bits can be disabled by using the **dfi$_{\text{data_bit_enable}}$** parameter.

The number of PHY slices is determined by dividing the DFI Data Width by the **phy$_{\text{data_slice_width}}$** parameter. The MC and PHY must define the DFI Data Width of the device in terms of slices even if the device does not implement all of the signals. The unused signals should be clearly defined in terms of bit location by using the **dfi$_{\text{data_bit_enable}}$** parameter. Otherwise, the number of slices might not be deterministic.

15.3 Compatibility with Older Version of DFI

There are no compatibility issues with previous versions of DFI.

15.4 Compliance

All DFI 4.0 PHY's should define the DFI Slice Width. All DFI 4.0 MCs should define Slice Widths that are supported and adhere to the data width requirements for all DFI signals as defined by the DFI Slice Width. It is acceptable for the MC to drive more copies of fanned out signals than the PHY requires. These signals would be expected to be “no connects”.

The MC might have more input signals than defined for PHY-driven signals on the interface. The MC must be able to operate with these signals tied inactive at the MC. The DFI Data Width for the MC and PHY can be less than the width defined for the interface as previously discussed. In this case, the MC and/or PHY must clearly define how the data signals are to be connected to the slice-based interface. Any “no connects” to the interface should be clearly defined.

16.0 Geardown Mode

16.1 Background

This chapter describes adding DDR4 geardown mode as part of DFI 4.0 DDR4 support. This change would include addition of a new signal, **dfi_geardown_en**, and also a new timing parameter for defining when commands can transmit after geardown mode is entered.

16.2 Detailed Description

DDR4 DRAMs add geardown mode for enabling command bus operation at high frequencies. Previously, systems would use 2T operation for increasing setup and hold times on the command bus, but 2T does not work for the **CS** signal. With geardown mode, all command/address (CA) bus signals can benefit from larger setup and hold times.

Larger setup and hold times are accomplished by operating the CA bus at ½ the frequency of memory clock. The DRAM internally samples the bus every other clock. To establish the sampling clock, synchronization between MC and DRAM is established during the geardown mode entrance procedure via a sync pulse. When this relationship is established, the MC can change the CA bus signals only on the boundaries that the sync pulse established.

In geardown mode, the PHY might need to change the alignment of the memory command relative to the memory clock. In normal operation, the memory command is generally aligned to the falling edge of the memory clock to center on the following rising edge. In geardown mode, the command should be aligned to the command's first clock rising edge to center on the command's second clock rising edge. To accomplish this, the PHY receives notice when the memory changes between normal mode and geardown mode via the **dfi_geardown_en** signal.

To enter geardown mode on DFI, the MC performs the DRAM Geardown entrance procedure. When the $\mathbf{t_{ctrl\_delay}}$ time elapses from the defined sync pulse, the MC asserts the **dfi_geardown_en** signal. The CA bus signals must be maintained in an idle or de-select state or both for the $\mathbf{t_{geardown\_delay}}$ parameter. The PHY time makes adjustments in CA timing.

Table 27 shows the geardown mode signal.

TABLE 27. *Geardown Mode Signal*

Signal	From	Width	Default	Description
dfi_geardown_en	MC	1	0	DFI Geardown Enable. When de-asserted, the MC and PHY operate normally. When asserted, the MC and PHY are in geardown mode. The MC can change CA signals only every other DFI PHY clock as defined by the synchronization pulse from the PHY.

Table 28 defines the geardown mode timing parameter.

TABLE 28. *Geardown Mode Timing Parameter*

Parameter	Defined By	Min	Max	Description
$t_{geardown\_delay}$	PHY	0	–[a]	The delay from **dfi_geardown_en** assertion to the time that the PHY is ready to receive commands.

a. The minimum supportable value is 1. The DFI does not specify a maximum value. The range of values supported is implementation-specific.

When **dfi_geardown_en** = 1, the MC must drive commands for two DFI PHY clocks—for two DFI clocks in a matched frequency configuration, or for two phases in a Frequency Ratio configuration. These commands must be aligned to the proper clock edge as defined by the sync pulse.

The MC must de-assert the **dfi_geardown_en** signal at least $t_{geardown\_delay}$ time before sending an exit self refresh command, as Figure 32 shows. Additionally, the MC must de-assert the **dfi_geardown_en** signal at least $t_{geardown\_delay}$ time before sending a Maximum Power Savings Mode (MPSM) entry MRS command, as Figure 33 shows.

FIGURE 32. *Self Refresh Exit*

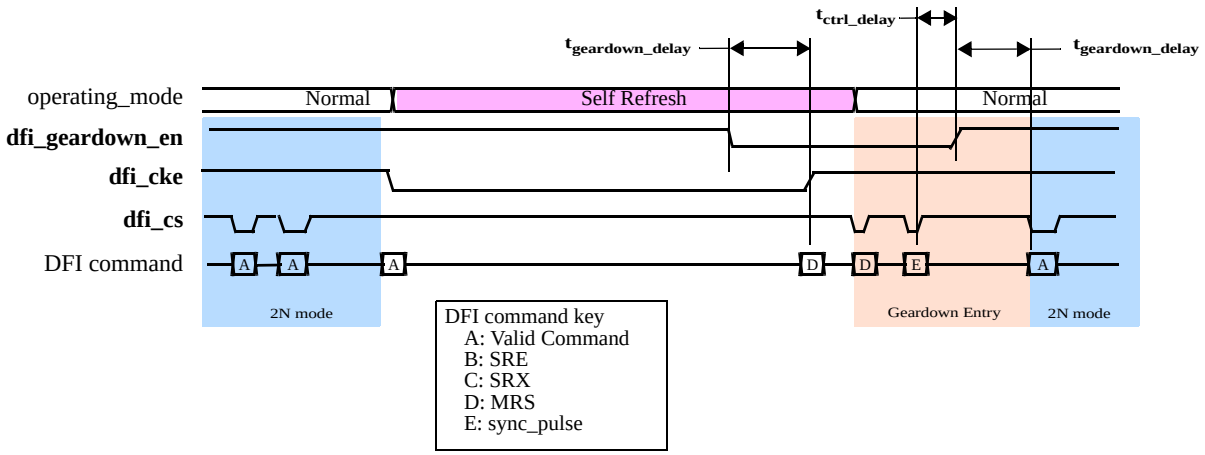
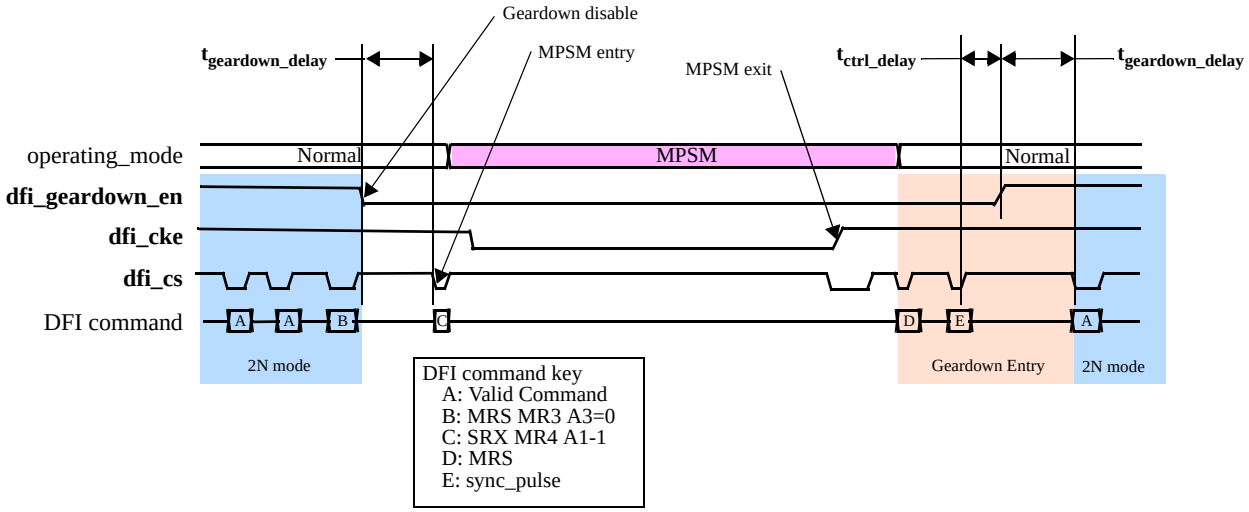


FIGURE 33. *MPSM Entry and Exit*



16.3 Compatibility with Older Version of DFI

When connecting a DFI 4.0 PHY with a DFI 3.1 MC, the **dfi_geardown_en** input signal should be tied low. When connecting a DFI 4.0 MC with a DFI 3.1 PHY, the **dfi_geardown_en** output signal should be left unconnected. If the PHY requires configuration changes for operating in geardown mode, these must be done by using software at the appropriate time.

16.4 Compliance

Geardown mode support is relevant to MCs and PHYs that support DFI 4.0 and Geardown mode. If the MC supports geardown, the signal is required. The signal is optional for the PHY.

17.0 DFI Feature and Memory Topology Matrix

17.1 Background

The DFI specification is a single document that defines a MC-PHY interface for numerous memory protocols and topologies. While some features are applicable to all memory interfaces, some are required only for particular memory sub-systems. To improve clarity of MC and PHY requirements, propose adding a section defining what features are required per topology.

17.2 Detailed Description

The following section is to be added to the beginning of the DFI specification, after Section 2.3 Optional Protocols.

2.4 DFI Feature Requirements

The DFI specification defines the MC-PHY interface for numerous memory protocols and topologies. Not all DFI features are applicable or required for any particular memory sub-system. To aid inter-operability and optimal system design, this section describes the features that apply for each supported memory topology.

2.4.1 Global Features

Global features apply to all memory topologies. While these features are valid across all memory sub-systems, they are not always required. For example, any system can utilize “low power control” to reduce system power. However, the requirement of “low power control” is based on system trade-offs and constraints that are outside the scope of DFI. The list of global features are as follows:

- Frequency ratios across DFI
- Frequency change
- Low power control
- Error signaling
- Update interface
- PHY master interface
- Clock disabling
- Data bit enable
- DFI disconnect
- Independent channel support

For specific signal requirements, refer to Section 2.5 DFI Signal Requirements in the full DFI specification.

2.4.2 Memory Topology-Specific Features

A subset of DFI features is not globally applicable to all memory sub-systems. A matrix of memory topology and features is defined in the table below. While the feature matrix defines when a feature applies to a specific memory topology, each feature is not necessarily required. Specific feature requirements are based on system trade-offs and constraints that are outside the scope of DFI. For specific signal requirements per feature, refer to section 2.5 DFI Signal Requirements in the full DFI specification.

TABLE 29. *Memory Topology Specific Features*

Memory Class	Topologies	Applicable Features[a], [b]
DDR1	Discrete, DIMM	Write data-eye training[c]
DDR2	Discrete, Unbuffered DIMM, Registered DIMM	Write data-eye training[c]
DDR3	Discrete, Unbuffered DIMM	• Read data-eye training • Read gate training • Write leveling • Write data-eye training[c] • DFI disconnect during training
	Registered DIMM	• All DDR3 discrete, unbuffered DIMM features • CA parity • CA parity errors
DDR4	Discrete, Unbuffered DIMM	• Read data-eye training • Read gate training • Write leveling • Write date-eye training[c] • DFI disconnect during training • Read data bus inversion • Write data bus inversion • Write data CRC • Write data CRC errors • CA parity • CA parity errors • Geardown mode
	Registered DIMM	All DDR4 discrete and unbuffered DIMM features
	Load Reduced DIMM	• All DDR4 discrete and unbuffered DIMM features • Data buffer training
LPDDR1	Any	No additional features beyond global features

TABLE 29. *Memory Topology Specific Features (Continued)*

Memory Class	Topologies	Applicable Features[a], [b]
LPDDR2	S2, S4	• Read data-eye training • Read gate training • Write data-eye training[c] • DFI disconnect during training
	NVM	• All LPDDR2 S2, S4 features • Data not valid
LPDDR3	Any	• Read data-eye training • Read gate training • Write date-eye training[c] • CA training • DFI disconnect during training
LPDDR4	Any	• Read data-eye training • Read gate training • Write leveling • Write date-eye training[c] • CA training • Read data bus inversion • Write data bus inversion • Combined and multi-configuration channel support

a. DFI defines features for specific memory class and topology. However, features can be extended beyond memory classes that are explicitly supported in DFI. This topic is outside the scope of DFI.

b. Feature support is limited to features that are included in a specific DFI version. For example, DFI support of DFI disconnect is not applicable to DFI 3.1 or previous MC and PHY components

c. JEDEC has defined explicit support of write date-eye training for LPDDR4. DFI has extended this to other DRAM classes by using functional writes and read commands, starting with DFI 4.0. For DFI 3.1 and earlier, write data-eye training is not supported on DFI.

Items as follows describe some limits of the feature topology matrix, and serve to simplify the matrix and DFI.

- Package types (PoP, MCP, DDP, QDP, etc.) are not defined in DFI and therefore are not part of the feature topology matrix.
- Features that are defined completely with a signal, with no other DFI ramifications, are not included in the feature topology matrix. This list includes the following items:
  - Termination: defined by dfi_odt
  - Periodic training: defined by dfi_lvl_periodic
  - Training patterns: defined by dfi_lvl_pattern
  - Per rank delay line support: defined by dfi_rd/wrdata_cs

  For these cases, support is defined in the DFI Signal Requirements section.

DRAM features supported but not explicitly covered by DFI are not included in the matrix. These include:

- DQ VREF training
- Per DRAM addressability

17.3 Compatibility with Older Version of DFI

No compatibility issues; this chapter is a clarification only

17.4 Compliance

This chapter defines compliance per feature for DFI4.0. This is applicable to all previous versions as well.

18.0 Optional DFI Training

18.1 Background

DFI 3.1 and earlier versions define the training interfaces that are required for the MC and optional for the PHY. The purpose of defining the training interface as required in the MC has been to support interoperability between MCs and PHYs of different developers. In DFI 3.1, the PHY Initiated Training Interface was added, and DFI 4.0 replaced this with the PHY Master Interface.

With two methods of training, and because training support by either method requires significant development effort, it is unrealistic to require support for both training methods in the MC. Therefore, it is recommended that training by using the DFI training signals be defined as optional for both the MC and the PHY.

18.2 Detailed Description

Table 2 “DFI Signal Requirements” under the sections “Training Interface Group - Read Training” for signals such as dfi_rdlvl_req under the MC column states “Required for DDR4, DDR3, LPDDR3, and LPDDR2 DRAMs”. This will be replaced with “Optional. Applicable to DDR4, DDR3, LPDDR3, and LPDDR2 DRAMs”. A similar change will be made to each signal within this interface and to all of the other training signal interfaces defined in Table 2.

In the “DFI Training Interface” section, the following statement will be removed:

“To be DFI-compliant, the MC is required to support read training, write leveling, and CA training operations whereas the PHY optionally supports each of the operations.”

Similar statements in the specification may exist and will be removed or modified accordingly.

18.3 Compatibility with Older Version of DFI

All DFI 4.0 MCs and PHYs must clearly define supported methods for training so that compatibility between devices is easily determined.

18.4 Compliance

There are no compatibility issues with previous versions of DFI.

19.0 3D Stack (3DS) Support

19.1 Background

This chapter details the necessary changes to DFI 3.1 specification to describe and clarify 3DS support.

19.2 Detailed Description

3DS Addressing with dfi_cid and dfi_cs for DDR3

While addressing 3DS devices with two and four logical ranks, the MC can use dedicated dfi_cs inputs to select these logical ranks. For 3DS SDRAMs with eight logical ranks, the MC requires the additional dfi_cid (Chip ID) address input to select logical ranks 4 through 7.

TABLE 30. *DDR3 Configuration with 8 Logical and 1 Physical Rank*

Physical Rank	Logical Rank	dfi_cs[3:0]	dfi_cid[0]
0	0	1110	0
0	1	1101	0
0	2	1011	0
0	3	0111	0
0	4	1110	1
0	5	1101	1
0	6	1011	1
0	7	0111	1

3DS Addressing with dfi_cid and dfi_cs for DDR4

For DDR4, the 3DS package is organized into two, four, or eight logical ranks. For DDR4 3DS devices, the MC can select the logical ranks by using the dfi_cid[2:0] signals. The dfi_cs signal selects the physical ranks.

TABLE 31. *DDR4 Configuration with 8 Logical and 2 Physical Ranks*

Physical Rank	Logical Rank	dfi_cs[1:0]	dfi_cid[2:0]
0	0	10	000
0	1	10	001
0	2	10	010
0	3	10	011
0	4	10	100
0	5	10	101

TABLE 31. *DDR4 Configuration with 8 Logical and 2 Physical Ranks (Continued)*

Physical Rank	Logical Rank	dfi_cs[1:0]	dfi_cid[2:0]
0	6	10	110
0	7	10	111
1	0	01	000
1	1	01	001
1	2	01	010
1	3	01	011
1	4	01	100
1	5	01	101
1	6	01	110
1	7	01	111

DFI 4.0 Modifications for 3DS Support

For supporting 3DS in DFI 4.0, content to be added includes a section that describes the 3DS operation and its possible effects on the DFI interface. Any mention of chip_select, rank, or both are to be clarified with a superscripted note reference: “In a 3DS solution control, MRW and training activity are to be limited to physical ranks.” Table 32 shows the row information for the DFI table of signals in the DFI specification.

TABLE 32. *DFI Signal to Be Added*

DFI Signal	Driven By	Width	Description
dfi_cid	MC	MAX_LOGICAL_RANK_WIDTH (3 bits)	This signal defines the chip ID. The dfi_cid signal is required for 3D stacked solutions.

The following changes need to be made to the current DFI specification for supporting 3DS:

- Change requirements of dfi_cid to include DDR3 3DS.
- Change suffix _b: “This signal is not meaningful during initialization; no default value is required” to “This signal initializes to 0 during initialization; a default value of 0 is required.”
- Define DFI Physical Rank Width.
- Add CID to dfi_ca_parity description.
- Change “DFI Rank Width” to “DFI Physical Rank Width”.
- Change the description of DFI Rank Width and DFI Chip Select Width.
- Change training signals that pertain to CS to be DFI Physical Rank Width (Descriptions and usage throughout specification are to be explained by a note referenced by a suffix: “In a 3DS solution control, MRW and training activity are limited to Physical Ranks.”)
- Change width of dfi_wrdata_cs and dfi_rddata_cs to be DFI Physical Rank Width.
- Update the description of rdcsgap and wrcsgap parameters; reference target physical rank instead of target chip select.

- Change CKE, ODT, Reset widths; they should not be DFI Chip Select Width or DFI Physical Rank Width. These signals should have unique widths. However, some additional clarity might be necessary for interoperability. The MC and PHY should support the same number of signals in most cases. For ODT, the MC must detect ODT signals that are valid. It could send more signals than the PHY can use, transmit, or both. However, whatever is dropped should not be asserted by the MC. Similar thought for CKE and power control per CS.
- Change the width of dfi_dram_clk_disable. It is not DFI Chip Select Width.
- Figure 1 dfi_cid superscript 4 should be revised to state DDR4 and DDR3 3DS.
- Figure 15 and Figure 29 might change to show physical and logical rank access.
- Table 4. Definition for dfi_cs might need to include 3DS solutions that are detailed in 3DS operation and its possible effects on the DFI interface.
- Table 4. Definition for dfi_cid needs to reference 3DS solutions that are detailed in 3DS operation and its possible effects on the DFI interface.
- Table 16. References to CS. Need to reference DFI Chip Select Width that is detailed in 3DS operation and its possible effects on the DFI interface. For example: dfi_phylvl_req_cs

19.3 Compatibility with Older Version of DFI

A DFI 4.0-compliant PHY that implements 3DS support does not support 3DS with an older DFI version of the MC. A DFI 4.0 MC that interfaces to an older DFI-version PHY supports dfi_cid and 3DS addressing. An older DFI version PHY might enable 3DS support with a DFI 4.0 MC that has 3DS support.

19.4 Compliance

If an MC or PHY configures and enables 3DS support, it needs to adhere to DFI 4.0 3DS requirements.

20.0 Update Interface Clarification on Self Refresh Exit

20.1 Background

The behavior of the dfi_ctrlupd_req signal just after self refresh exit (as the current DFI PHY specification describes) could lead to incorrect operation in some PHY and bus implementations.

20.2 Detailed Description

Section 3.4 states “Following a self refresh exit, the dfi_ctrlupd_req signal must be asserted prior to allowing read/write traffic to begin”. This statement implies that a ctrlupd request is to occur after the self refresh exit command but before any read/write command. This action could be unsafe with respect to the self refresh exit command itself. Signals that are used in the actual self refresh exit action (such as CKE in DDR4 or CA in LPDDR4) might require an update process before a self refresh exit command is to occur.

Change this requirement to “A control update request and acknowledge via dfi_ctrlupd_req|ack is required immediately before a self refresh exit command runs.”

20.3 Compatibility with Older Version of DFI

No signals change, so there is no connection incompatibility. A DFI 4.0 PHY that connects to a DFI 3.1 controller cannot require the earlier assertion of dfi_ctrlupd_req. A DFI 4.0 controller that connects to a DFI 3.1 PHY can assert dfi_ctrlupd_req earlier without side effects.

20.4 Compliance

All DFI 4.0 MCs must be in compliance.

21.0 Inactive CS

21.1 Background

The PHY Master Interface requires one bit of req/ack and a type that is on a per-chip select basis. PHY requires a chip select to be put in a particular state before the MC acknowledges with **dfi_phymstr_ack**.

A value of **dfi_phymstr_type** = ‘b0 means IDLE, and a value of **dfi_phymstr_type** = 'b1 means IDLE or SREF state. This action provides an encoding scheme for training different memories in particular states and allowing low power states to be maintained for chip selects that are not being trained. For example, DDR4 supports training in only the IDLE state, but LPDDR4 supports training in both IDLE and SREF states. Limitations to this scheme are as follows:

21.1.1 Limitation 1

Chip selects that are not being trained cannot be put into the lowest power states, such as the following examples:

- Deep power-down (LPDDR2 and LPDDR3)
- Maximum power savings mode (DDR4)

When a chip select is in one of the following modes, the MC should optimally retain the memory in the current state.

1. MPSM or deep power-down mode
   The MC should not unnecessarily bring the memory out of the low power mode to an IDLE or SREF state. In these modes, the memory contents might not be maintained by the DRAM.
2. Uninitialized memory
   The memory should be left uninitialized until the system determines otherwise.
3. Chip select that is unpopulated or powered off
   Chip select should remain offline until the system determines otherwise.

21.1.2 Limitation 2

A violation of the $\mathbf{t_{phymstr\_resp}}$ parameter can occur for an uninitialized chip select. As the system has only one req/ack bit, the PHY requires that all chip selects are in a particular state before the MC can issue an ack command.

To bring the memory to an IDLE or SREF state, uninitialized chip selects or chip selects in the MPSM/DPD state might require software intervention. In the case of uninitialized or DPD chip select, the software must run through the full initialization routine. The time can increase greatly, based on the number of uninitialized chip selects. Therefore, defining a maximum number for $\mathbf{t_{phymstr\_resp}}$ parameter is not feasible, as the MC cannot ensure the required time.

NOTE: This chapter does not define behavior of the PHY on violation of $\mathbf{t_{phymstr\_resp}}$ parameter.

21.2 Detailed Description

Previous versions of the DFI have not defined the chip select state. With new interfaces and for clarity, a new global parameter is added, defined by the system that indicates the chip select state as the following table shows.

Term	Definition
sys_{cs_state}	Global parameter that defines the state of each chip select in the system. Width of **sys_{cs_state}** is DFI Chip Select Width, with each bit corresponding to a chip select in ascending order. For example, in a 4-chip select system, **sys_{cs_state}** is encoded as follows: • **sys_{cs_state}**[0]: State of chip select 0 • **sys_{cs_state}**[1]: State of chip select 1 • **sys_{cs_state}**[2]: State of chip select 2 • **sys_{cs_state}**[3]: State of chip select 3 The value of each bit defines the current state of the corresponding chip select with the following encoding: • 'b0 Inactive chip select that is in one of the following states: - Unpopulated, uninitialized, or powered off - Low power state where contents are not guaranteed for the DRAM, such as deep power-down mode or maximum power savings mode • 'b1 Active chip select: An active chip select can be in an active, IDLE, or low power state for which the memory contents are maintained by the DRAM, MC, or both. The system must maintain a consistent, stable view of **sys_{cs_state}** between the PHY and MC for ensuring synchronization between the MC and PHY.

This definition would be added to the glossary of the DFI specification and referenced as needed throughout the specification.

While **sys_{cs_state}** can be applied to any DFI function, this discusses using **sys_{cs_state}** in the PHY Master Interface and PHY initiated training. The following sections describe changes to these interfaces.

21.2.1 PHY Master Interface

To incorporate the new **sys$_{cs_state}$** parameter, a new note is applied to **dfi_phymstr_type** as follows.

Signal	From	Width	Default	Description
dfi_phymstr_req	PHY	1	‘b0	DFI PHY Master Request: When asserted, the PHY makes a request to pass the control to the PHY. The systems must maintain a consistent, stable view of **sys$_{cs_state}$** after **dfi_phymstr_req** is asserted for ensuring synchronization between the MC and PHY.
dfi_phymstr_type	PHY	DFI Chip Select Width	'b0	DFI PHY Master Request Type: This signal indicates the required state of the DRAM when the PHY becomes the master. Each memory chip select uses one bit. • ’b0 IDLE, the MC should close all the pages. • ’b1 IDLE or self refresh. This signal is valid only when the **dfi_phymstr_req** signal is asserted by the PHY and should remain constant while the **dfi_phymstr_req** signal is asserted. The **dfi_phymstr_type** bit values are not relevant for chip selects with **sys$_{cs_state}$** set to 'b0 (inactive chip selects). The MC can continue keeping the chip selects with **sys$_{cs_state}$** set to 'b0 in their current, inactive state, regardless of the corresponding **dfi_phymstr_type** bit value. The PHY must not require these chip selects to be in IDLE or SREF states.

21.2.2 PHY Initiated Training

The concept of “inactive chip selects” should extend to all PHY-initiated training. The table indicates the signals and descriptions:

- The signal names no longer have “_n”.
- The signals' current descriptions are to include the added text.

Relevant Signals (“_n removed”)	Description
dfi_phy_calvl_cs **dfi_phy_wrlvl_cs** **dfi_phy_rdlvl_cs** **dfi_phy_rdlvl_gate_cs**	The PHY must not assert bits with **sys$_{cs_state}$** set to 'b0 (inactive chip selects).

21.3 Compatibility with Older Version of DFI (3.1)

21.3.1 PHY Master Interface

When connecting a DFI 4.0 PHY with a DFI 3.1 MC, the **dfi_phymstr_type** from the PHY can be connected to **dfi_phylvl_req_cs** to the MC by using **dfi_phymstr_req**. The PHY must request a **dfi_phymstr_type** = 'b0 for the active chip selects that are training so that only those chip selects have **dfi_phylvl_req_cs** bits asserted on the MC.

When connecting a DFI 4.0 MC with a DFI 3.1 PHY, **dfi_phylvl_req_cs** from the PHY can be connected to **dfi_phymstr_req**, asserting the request when any bit is asserting in **dfi_phylvl_req_cs**. Care should be taken if PHY asserts a **dfi_phylvl_req_cs** to an inactive chip select. If the DFI 3.1 PHY has no information of the inactive chip selects, external logic on DFI is required to handle this case. The **dfi_phymstr_ack** can be fanned out to all active chip selects on **dfi_phylvl_ack_cs**.

21.3.2 PHY Initiated Training

When connecting a DFI 4.0 PHY with a DFI 3.1 MC, no change is required.

When connecting a DFI 4.0 MC with a DFI 3.1 PHY, care should be taken if the PHY asserts a **dfi_phy_*lvl_cs** to an inactive chip select. If the PHY has no information of the inactive chip selects, external logic on DFI is required to handle this case.