Hardware System Verification (HSV)
Vertical Solutions Engineering (VSE)

DFI PHY 5.0
Palladium/Protium Memory Model
User Guide

Document Version: 1.8

Document Date: March 2021

Contents

List of Figures

List of Tables

General Information

The Cadence Memory Model Portfolio provides memory device models for the Cadence Palladium/Protium series systems. Optimizing the acceleration and/or emulation flow on these platforms for MMP memory models may require information outside the scope of the MMP user guides and related MMP documentation.

1.1 Related Publications

For basic information regarding emulation and acceleration, please refer to the following documents:

For Palladium Z1:

VXE User Guide
VXE Library Developer's Guide
VXE Known Problems and Solutions
VXE Command Reference Manual
Palladium Z1 Planning and Installation Guide
Palladium Target System Developer's Guide
What's New in VXE

For Protium:

Protium User Guide
Protium Command Guide
Protium Express Toolkit Command Reference Guide
Protium S1 Hardware Installation and Reference Guide
Protium X1 Hardware Installation and Reference Guide
Protium Release Information
Protium Known Problems and Solutions
What's New in Protium

DFI PHY Model

1. Introduction

The Cadence Palladium/Protium DFI PHY model and wrappers are based on the DFI 5.0 Specification (April 27, 2018).

The Palladium/Protium DFI PHY models are designed to work with Cadence Palladium/Protium memory models. They are not intended to be used for other forms of verification like simulation.

The models support MC-initiated update and PHY-initiated update. The training interface is not supported. The models support several types (DDR/DDR2/DDR3/DDR4/DDR5/Mobile DDR (LPDDR) /LPDDR2/LPDDR3/LPDDR4/LPDDR5/GDDR6) of memory device models.

The model is configurable; it can be configured to meet a range of customer design requirements. Please follow the configuration instructions in the following sections of this User Guide.

2. Model Release Levels

All models in the Memory Model Portfolio are graded with a release level. This release level informs users of the current maturity and status of the model. All families in the library are graded at one of these levels.

The different levels give an overall indication of the amount of testing, level of quality and feature availability in the model. For details on supported features check the User Guide for that particular model family.

There are three release levels for models in the MMP release.

Release Level		Model Status	Available in Release	Listed in Catalog	Requires Beta Agreement
Mainstream Release	MR	Fully released and available in the catalog for all customers to use.	Yes	Yes	No
Emerging Release	ER	Model has successfully completed Beta engagement(s). Most, but not all features have been tested. Documentation is available.	No	Yes	Yes
Initial Release	IR	Model has completed initial development and has been released to Beta customer(s). The model may have missing features, may not be fully tested and may not have documentation. Model may contain defects.	No	Yes	Yes

Access to Initial Release and Emerging Release versions of the models will require a Beta Agreement to be signed before the model can be delivered.

3. DFI PHY Model Overview

Almost all designs today contain high speed memory interfaces. These are now typically serviced by memory controller IP from a third party. The memory controller IP has two parts, the controller itself and the PHY. The DFI PHY protocol is an open PHY interface protocol that allows controllers and PHYs to interoperate.

Typically, the implementation netlist of the PHY contains higher speed clocks, PLLs or DLLs, and other high speed PHY structures that are either problematic to compile for Palladium/Protium or impact the overall performance of the Palladium/Protium environment. The Cadence Palladium/Protium DFI PHY model is a generic, cycle accurate model that is intended to replace the implementation PHY in the customer's Palladium/Protium environment. This DFI PHY model allows customers to perform system level testing of their design including the memory controller interface. It is not intended to be used for memory controller or PHY verification.

The Palladium/Protium DFI PHY model is not based on any implementation PHY from Cadence or from other third parties.

It is strongly recommended that prior to starting to integrate the DFI PHY model the customer first become familiar with the DFI protocol and appropriate DDR memory protocols as well as reviewing in detail the simulation results of their DFI PHY simulation model.

The current structure for the MMP DFI PHY model comprises one protected netlist (*.vp) for the core named dfiphy5_pd.vp and a set of wrappers, one for each of the supported memory models (DDRx, LPDDRx) named dfiphy_<model>.v, and from which the user should select that which is required. Each wrapper uses parameters to set the particular sdram type and dfi phy related timing parameters for the core model -- dfiphy5_pd.vp.

The figure below (Figure 1: Block Diagram of Controller, DFI PHY, and DDR5 Memory) shows an overall block diagram of a controller and DFI PHY along with a two rank DDR5. The DDR5 is used as an example throughout this document to clarify interface and timing information.

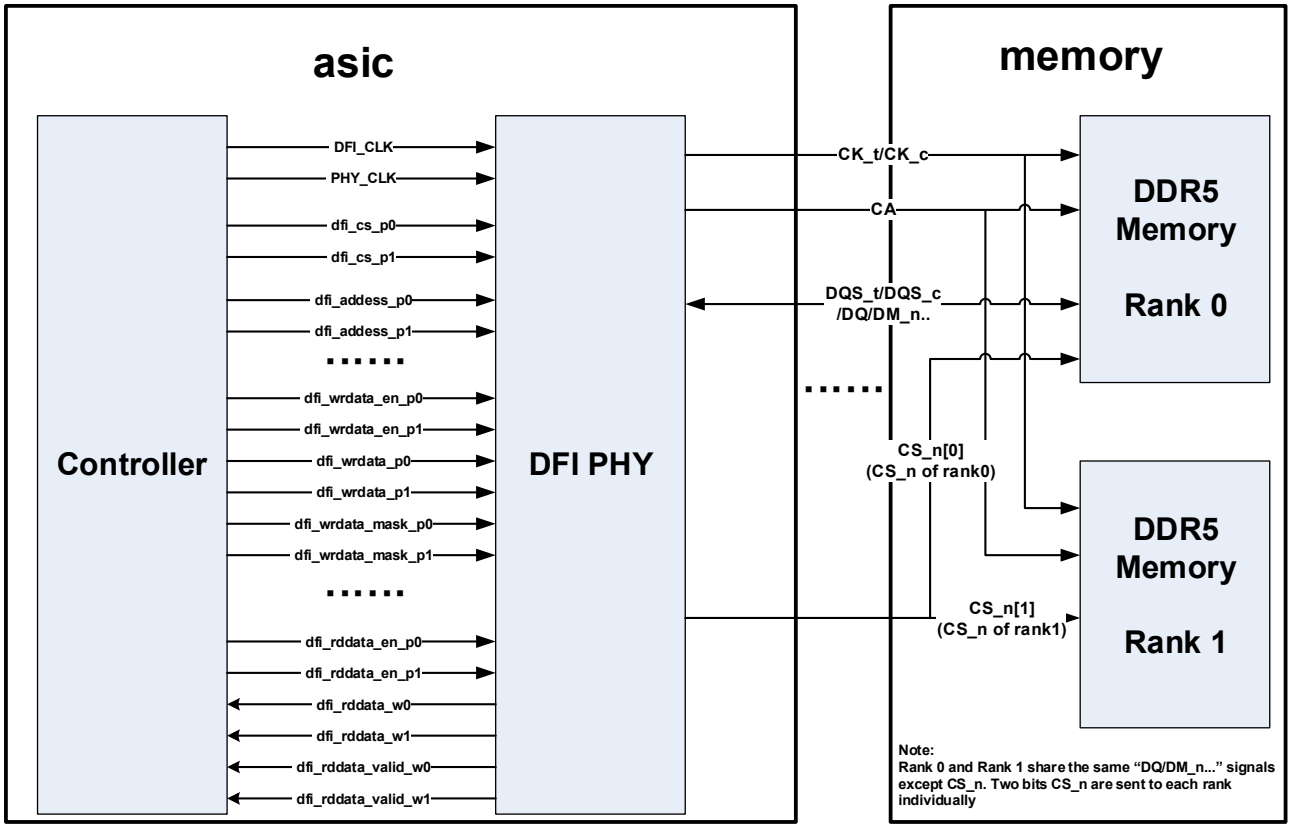


Figure 1: Block Diagram of Controller, DFI PHY, and DDR5 Memory

Below are the steps about how to integrate the MMP generic DFI PHY and an MMP SDRAM model (labelled below as DDR MEM) into the user's environment for emulation.

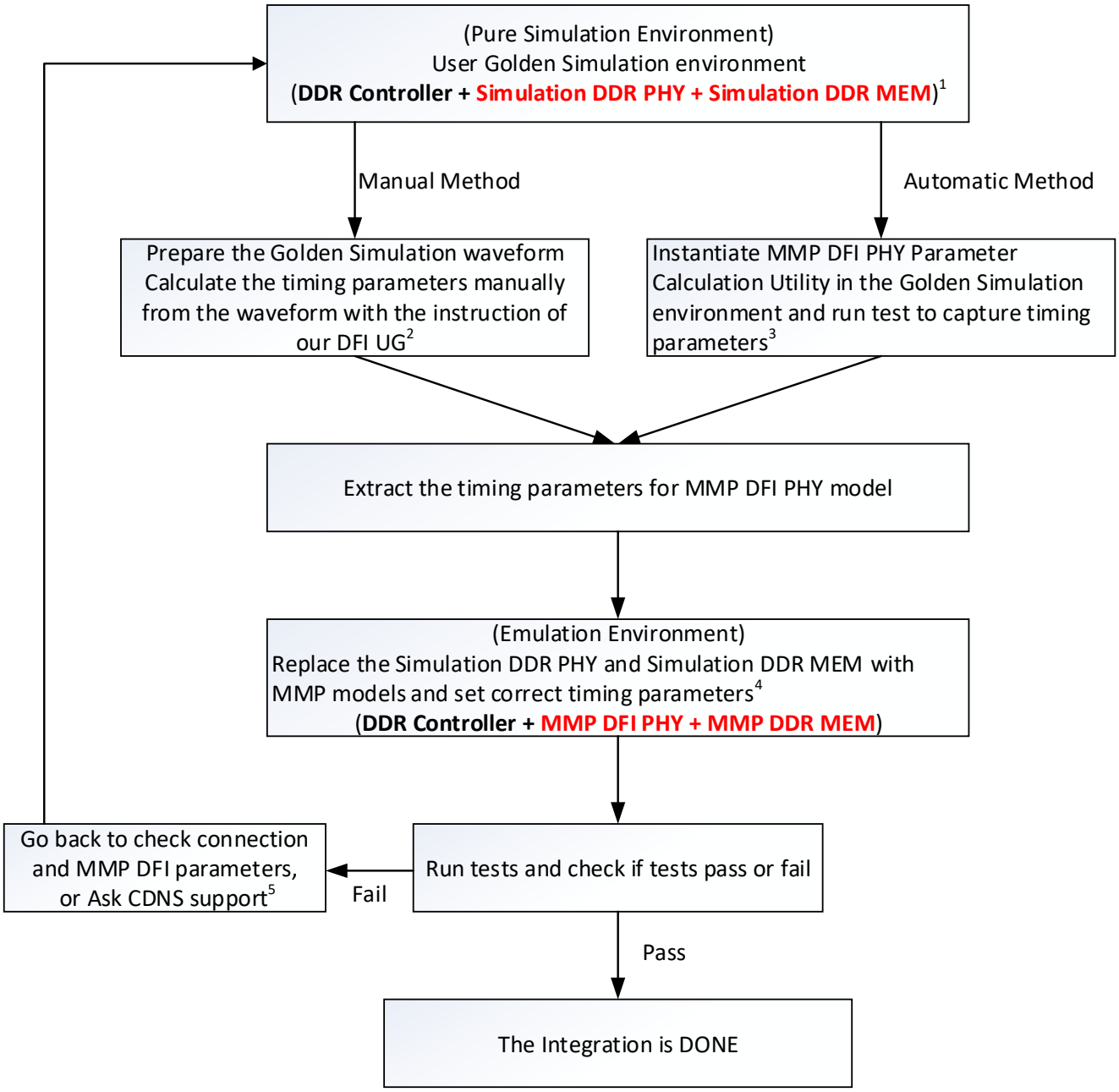


Figure 2: Steps for MMP DFIPHY Integration

Notes:

The bolded red text highlights the simulation models to be replaced by the equivalent MMP models (also highlighted in red).

1. The test should include the initialization stage, and several read and write operation (data should be non-zero for debugging ease) for golden simulation for extracting MMP DFI PHY timing parameters.

2. Refer to MMP DFI PHY 5.0 UG section 4.4-4.7 for 1:1 ratio system and Appendix A for 1:2 ratio system for MMP DFI PHY timing parameter calculation manually.

3. Refer to MMP DFI_PHY_Parameter_Calculation_Utility UG for MMP DFI PHY timing parameter calculation automatically. Please read this UG carefully about its constraints.

4. Refer to MMP DFI PHY 5.0 UG section 5 for the connections. Based on the calculation result, besides the timing parameter setting for MMP DFI PHY, there may be extra need for MMP DDR MEM parameter or Mode Register forcing.

5. When asking CDNS support, it is required to provide the golden simulation waveform and the emulation waveform with MMP DFI PHY and MMP DDR MEM **ports and all internal signals** probed.

4. DFI PHY Model Configuration

All parameters and configurations of the DFI PHY model are controlled in the model using Verilog parameters or defines. The following sections provide an overview of the user adjustable parameters, the exposed localparams, and the Verilog macro defines and then treat each adjustable parameter and define in more detail. Some additional configuration information and signals in the model are exposed for debugging purposes. It is not an intent, however, of this section to describe all exposed information.

4.1 Overview of User adjustable Parameters and Non Adjustable Localparam

The following tables provide details on the **user adjustable** parameters for the Palladium/Protium DFI PHY Memory Model. These parameters may be modified when instantiating a DFI PHY wrapper. Additional information about derived but non-adjustable localparams that are exposed for debugging purposes is also considered below.

The wrappers for these memory models (DDRx, LPDDRx, GDDR6) vary as to which user adjustable parameters are required. Table 1: DFI PHY Model User Adjustable Parameters Per Memory Model shows which memory models require which user adjustable parameters.

Table 2: Description of DFI PHY Model User Adjustable **Parameters** lists and describes all of the parameters used by the DFI PHY model.

Additional description and critical details for these parameters are provided in user guide sections below these tables.

Table 1: DFI PHY Model User Adjustable Parameters Per Memory Model

User Adjustable Parameter	DDR	DDR2	DDR3	DDR4	DDR5	MOBILE DDR	LPDDR2	LPDDR3	LPDDR4	LPDDR5	GDDR6
t_ctrl_delay	√	√	√	√	√	√	√	√	√	√	√
t_phy_wrdata	√	√	√	√	√	√	√	√	√	√	√
t_wrlat_adj	√	√	√	√	√	√	√	√	√	√	√
t_rddata_en_adj	√	√	√	√	√	√	√	√	√	√	√
t_rddata_edc_en_adj											√
mem_data_bits	√	√	√	√	√	√	√	√	√	√	√
dfi_addr_bits	√	√	√	√	√	√	√	√	√	√	√
dfi_bank_addr_width	√	√	√	√		√					
dfi_bg_addr_width				√							
dfi_cid_width				√	√						
dfi_cs_width	√	√	√	√	√	√	√	√	√	√	√
dfi_wck_width										√	
mem_ca_bits					√		√	√	√	√	√
t_shift_p1_adjust_delay	√	√	√	√	√	√	√	√	√	√	√
t_shift_p1_adjust_delay_wr	√	√	√	√	√	√	√	√	√	√	√
dfi_data_bit_enable	√	√	√	√	√	√	√	√	√	√	√

t_lp_resp	√	√	√	√	√	√	√	√	√	√	√
t_lp_ctrl_wakeup	√	√	√	√	√	√	√	√	√	√	√
t_lp_data_wakeup	√	√	√	√	√	√	√	√	√	√	√
dfi_lp_version	√	√	√	√	√	√	√	√	√	√	√
phy_init_mode	√	√	√	√	√	√	√	√	√	√	√

Table 2: Description of DFI PHY Model User Adjustable Parameters

User Adjustable Parameter	Default Value	Description
t_ctrl_delay	2	Delay from DFI interface control to memory interface control, measured in DFI clock cycles
t_phy_wrdata	1	Delay from wrdata_en assert until first wrdata value, measured in PHY clock cycles
t_wrlat_adj	0	Adjustment between PD model and DUT PHY write latency, measured in PHY clock cycles
t_rddata_en_adj	0	Adjustment between PD model and DUT PHY read latency, measured in PHY clock cycles
t_rddata_edc_en_adj	0	Adjustment between PD model and DUT PHY read edc latency, measured in PHY clock cycles
mem_data_bits	72	Memory device interface data bits
dfi_addr_bits	12	Memory controller interface port address bits
dfi_bank_addr_width	3	Memory controller interface port bank address bits
dfi_bg_addr_width	2	Memory controller interface port bank group bits, required for DDR4
dfi_cid_width	3	Memory controller interface port CID bits, required for DDR4
dfi_cs_width	4	Memory controller interface port chip select signal bits
dfi_wck_width	1	The number of individual WCK control signal interfaces driven on the DFI bus
mem_ca_bits	6	Memory device interface port command and address bits(used only for LPDDR2 / LPDDR3 / LPDDR4/LPDDR5/GDDR6/DDR5) See Section “Parameter mem_ca_bits”
mem_data_byte_width	8	Width of a byte. The valid value is 8 or 4, set to 4 when x4 data width
t_shift_p1_adjust_delay	3	Adjustment used when ADJUST_P1 define is set for read data path
t_shift_p1_adjust_delay_wr	3	Adjustment used when ADJUST_P1 define is set for write data path
dfi_data_bit_enable	{(mem_data_bits*2) {1'b1}}	Valid data bits for data bus
t_lp_resp	7	Specifies the maximum number of DFI clock cycles after the assertion of the **dfi_lp_ctrl_req** or **dfi_lp_data_req** signal to the assertion of the **dfi_lp_ack** signal. Max value of this parameter is 7 and it is recommended to fix it to 7.

t_lp_ctrl_wakeup	16	Specifies the **REAL** number of DFI clock cycles that the **dfi_lp_ctrl_ack** signal remain asserted after the de-assertion of the **dfi_lp_ctrl_req** signal.[1*]
t_lp_data_wakeup	16	Specifies the **REAL** number of DFI clock cycles that the **dfi_lp_data_ack** signal remain asserted after the de-assertion of the **dfi_lp_data_req** signal.[1*]
dfi_lp_version	0	0: Connecting with DFI5 (above) low power interface 1: Connecting with older version DFI5 low power interface
phy_init_mode	0	0: MC init mode 1: PHY init mode

Note 1: Since the wakeup time is fixed by parameter t_lp_ctrl_wakeup and t_lp_data_wakeup, the value sent on ports “dfi_lp_ctrl_wakeup/dfi_lp_data_wakeup” only indicates maximum wakeup time. Cadence DFI model ignores the value on ports “dfi_lp_ctrl_wakeup/dfi_lp_data_wakeup”.

The following table lists some of the exposed localparams made visible for debug purposes and which are NOT user adjustable. Many of these localparams are derived from or dependent upon user adjustable parameters listed above.

Table 3: DFI PHY Model Visible Localparams

Localparam	Default Value	Description
t_data_path_delay	2	= t_ctrl_delay
t_ctrl_delay_max	16	Max value of timing parameter t_ctrl_delay
t_phy_wrdata_max	32	Max value of timing parameter t_phy_wrdata
t_wrlat_adj_max	32	Max value of timing parameter t_wrlat_adj
t_rddata_en_adj_max	32	Max value of timing parameter t_rddata_en_adj
t_p1_adjust_max_wr	24	Max value of timing parameter t_shift_p1_adjust_delay
t_p1_adjust_max	24	Max value of timing parameter t_shift_p1_adjust_delay_wr
dfi_data_bits	144	mem_data_bits*2
dfi_num_bytes	2	mem_num_bytes*2
t_rd_en_max	2*4+0+1 = 9	Determine the max size of a delay buffer which is used to delay dfi_rddata_en; sets relevant shift register to the max length for 1:4 mode. t_data_path_delay*4+t_rddata_en_adj+1;
phy_dbi_mode	0	Determines which device generates DBI and inverts the data; PD model only supports phy_dbi_mode=0(MC perform DBI operation)
t_phy_wrdelay	0	Delay for frequency ratio systems. In DFI 3.1 and later this must always be 0. Not adjustable in DFI PHY 4.0.
For delay wr_en and wrdata		
t_dq_max	2*4+0+0 = 8	t_data_path_delay*4+t_wrlat_adj+t_phy_wrdelay
t_dqoe_max	8+1 = 9	t_dq_max+t_phy_wrdata
t_dq	0+0 = 0	t_wrlat_adj+t_phy_wrdelay
t_dqoe	0+1 = 1	t_dq+t_phy_wrdata

Note that there may be additional exposed localparams and information in the model HDL that are not described here nor intended to be described here.

4.2 Optional Verilog Macro Defines

The following table lists the optional Verilog macro defines the user may want to consider.

Table 4: DFI PHY Optional Verilog Defines

Verilog Macro `define Purpose	Optional Verilog `define Values
DFI initialization sequence support; define when controller does not support initialization interface. See section “Initialization of the PHY”	MMP_NO_DFI_INIT_START
Phase alignment support; define to enable DFI PHY to handle phase alignment. See section “Phase 0/1 alignment special case”	MMP_DFI_ADJUST_P1
Some users have a specific requirement to adjust tphy_rdlat. Before setting a value on tphy_rdlat, this user MUST define this macro. See section 4.10	MMP_DFI_ADJUST_TPHY_RDLAT
Enable GDDR6 X8 Mode (This macro is only valid when the wrapper “mmp_cdns_gddr6_phy_wrapper.v” used)	MMP_PHY_GDDR6_MODE_X8

4.3 DFI Clocks

In the following sections that cover the timing parameter settings for the DFI PHY model, the time units used are either DFI clock or DFI PHY clock. DFI clock refers to the clock used for the DFI interface signals. DFI PHY clock refers to the memory interface signals and is used in specifying many timing parameters in DFI model. The names used in MMP DFI model, Denali DFI IP and spec are listed in the table below.

Table 5: Clock Names in Different Environments

DFI Specification	MMP DFI Model	Cadence DFI IP
DFI clock	DFI_CLK	dfi_clk
DFI PHY clock	PHY_CLK	clk
DFI data clock	DAT_CLK	-

In a 1:1 frequency ratio system DFI clock and DFI PHY clock are the same frequency and phase aligned. In a 1:2 frequency ratio system DFI PHY clock is 2x faster than DFI clock and similarly in a 1:4 frequency ratio system DFI PHY clock is 4x faster than DFI clock. The figures below show clocking for a 1:2 frequency ratio system and a 1:4 frequency ratio system.
For LPDDR5, the frequency ratio of DFI clock and DFI PHY clock is fixed 1:1, and the DFI data clock is 2x or 4x faster than DFI PHY clock and phase aligned.

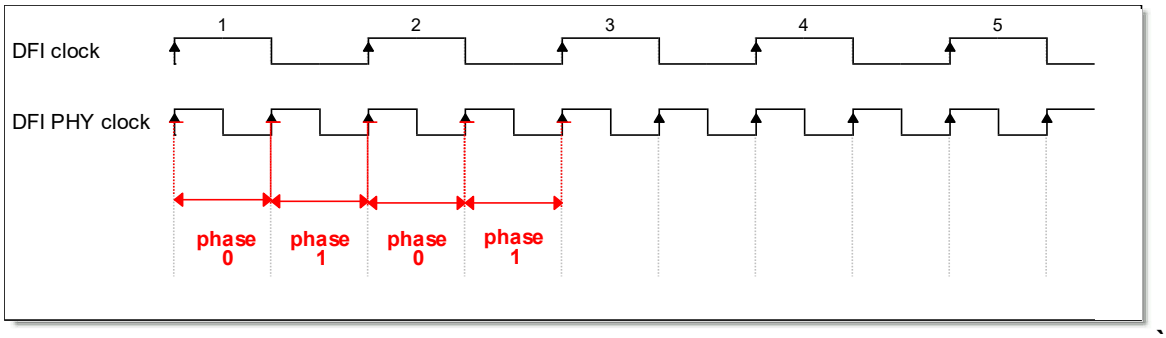


Figure 3: Clocking For a 1:2 Frequency Ratio System (all models except LPDDR5)

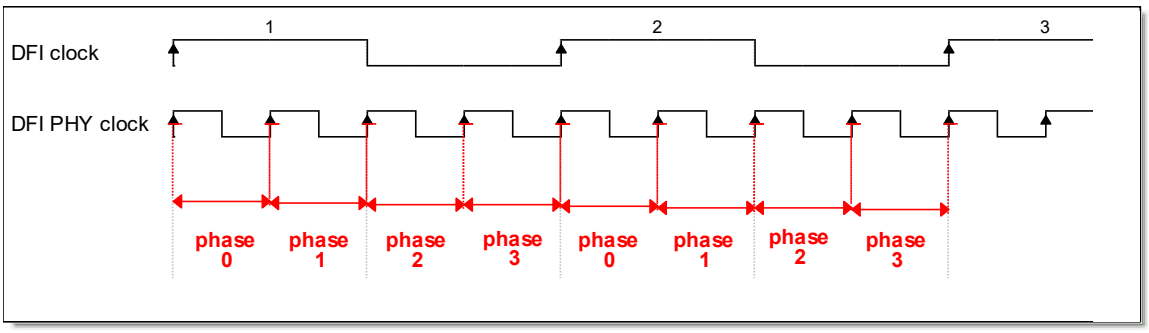


Figure 4: Clocking For a 1:4 Frequency Ratio System (all models except LPDDR5)

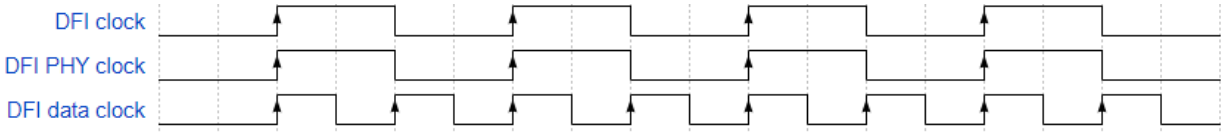


Figure 5: Clocking for LPDDR5 1:2 Ratio between DFI PHY clock and DFI data clock

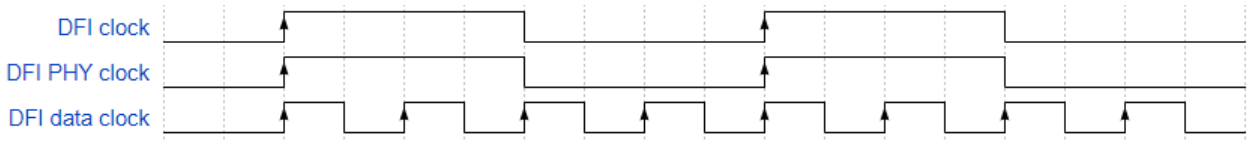


Figure 6: Clocking for LPDDR5 1:4 Ratio between DFI PHY clock and DFI data clock

The parameter calculations in the directly following sections are for a 1:1 frequency ratio system that includes a DDR5 memory.

4.4 Parameter t_ctrl_delay

This parameter defines the number of clock cycles delay through the DFI PHY for the control path signals in units of *DFI clock (DFI_CLK)* cycles. This is described in Section 4.3 of the DFI (DDR PHY Interface) 5.0 specification (April 27, 2018). The following waveform (Figure 7: Parameter t_ctrl_delay Waveform) shows t_ctrl_delay value of 3. As the delay is measured in DFI clocks and not *DFI PHY (PHY_CLK)* clocks it is possible that t_ctrl_delay is a fraction of a *DFI clock (DFI_CLK)*. In this case round the value up.

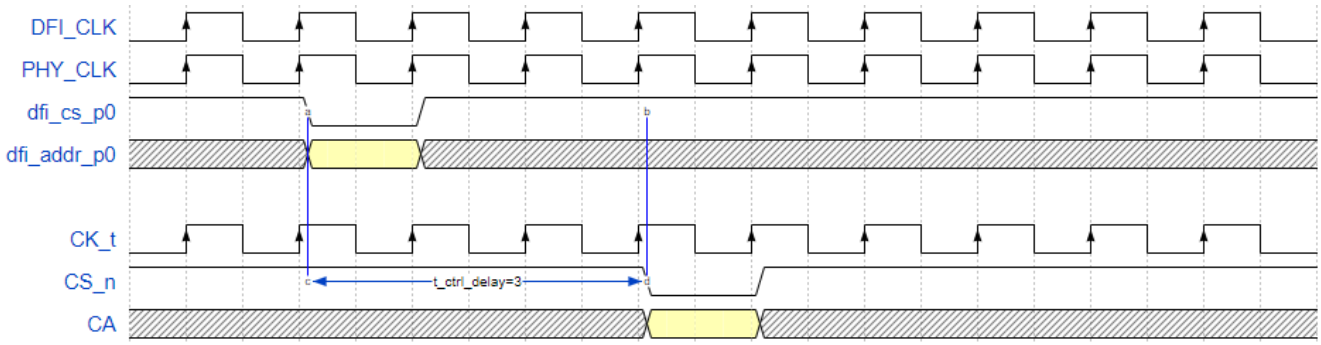


Figure 7: Parameter t_ctrl_delay Waveform

4.5 Parameter t_phy_wrdata

This parameter defines the delay from the dfi_wrdata_en being asserted to the first valid data values on the dfi_wrdata bus in units of *DFI PHY (PHY_CLK)* clocks. This is described in Section 4.7 of the DFI (DDR PHY Interface) 5.0 specification (April 27, 2018). The following waveform (Figure 8: Parameter t_phy_wrdata Waveform) shows a t_phy_wrdata value of 2 for a 1:1 frequency ratio system.

Note that when reviewing the simulation waveform, it is important to correctly determine when the first valid data occurs. It is possible that the controller can send valid data with the value of zero.

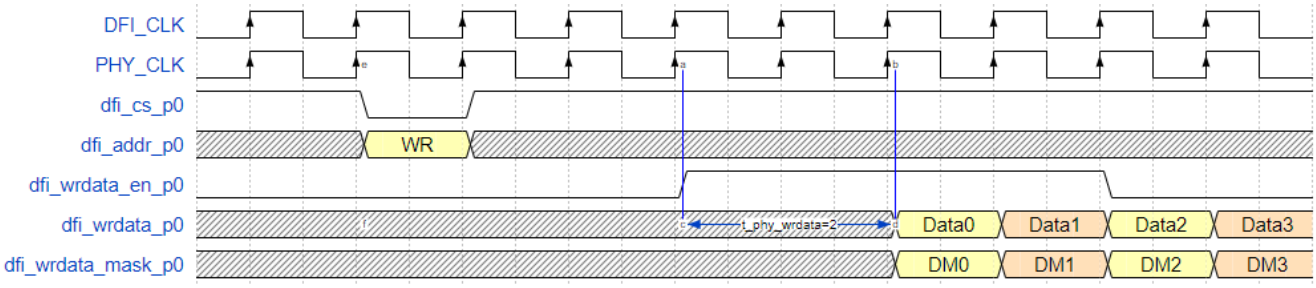


Figure 8: Parameter t_phy_wrdata Waveform

4.6 Parameter t_wrlat_adj

This parameter is used to adjust the DFI PHY parameter t_phy_wrlat. The DFI PHY parameter t_phy_wrlat defines the number of *DFI PHY (PHY_CLK)* clock cycles delay between the write command on the DFI interface and the assertion of the dfi_wrdata_en signal. This is described in Section 4.7 of the DFI (DDR PHY Interface) 5.0 specification (April 27, 2018).

The Cadence Palladium/Protium DFI PHY model has an internal write latency itself, which varies based on protocol. This may need to be adjusted to match the latency of the users DFI PHY. The parameter t_wrlat_adj is used for this purpose. The internal write latency of the DFI PHY model is calculated as follows:

```
DDR/Mobile_DDR:     t_pd_wrlat = WL - t_phy_wrdata
DDR2/3/4:           t_pd_wrlat = WL - t_phy_wrdata
DDR4 RDIMM:         t_pd_wrlat = WL - t_phy_wrdata
DDR5:               t_pd_wrlat = WL + 1 - t_phy_wrdata
DDR5 RDIMM:         t_pd_wrlat = WL + 5 - t_phy_wrdata
LPDDR2/3:           t_pd_wrlat = WL + tDQSS - t_phy_wrdata
LPDDR4:             t_pd_wrlat = WL + tDQSS + tDQSS_half - t_phy_wrdata
LPDDR5:             t_pd_wrlat = (WL+1)*ratio-1 - t_phy_wrdata
                    (ratio=4 for CKR(WCK:CK)=4:1; ratio=2 for CKR(WCK:CK)=2:1)
GDDR6:              t_pd_wrlat = WL - t_phy_wrdata
```

Note: WL above refers to the DDR or LPDDR memory Write Latency.

- Write Latency for DDR and Mobile_DDR (LPDDR) is just the CL value (CAS Latency).
- Write Latency for DDR2 is defined as RL – 1 (Read Latency -1) where read latency is the sum of the CL (CAS Latency) and AL (Additive Latency) i.e. RL=AL+CL.
- Write Latency for DDR3 is the sum of CWL (CAS Write Latency) and AL (Additive Latency) i.e. WL=AL + CWL.
- Write Latency for DDR4 is the sum of CL (CAS Latency) and AL (Additive Latency) and Parity Latency i.e. WL=AL + CWL + PL.
- For LPDDR2/3/4/5 and DDR5 and GDDR6 WL is directly defined by mode register content. The default value of tDQSS is 1 for LPDDR2/3; by default, the tDQSS=1 and tDQSS_half=0 for LPDDR4.
- For DDR4 RDIMM, if the WL value from mode registers setting is not equal to the real WL from write operation, use the WL value from write operation for this formula calculation.

These latency settings are controlled by Mode Registers in the memory device and are set by the memory controller during initialization.

Please refer to JEDEC specification for more information.

The parameter t_wrlat_adj is the difference between the actual PHY t_wrlat parameter and the DFI PHY model t_wrlat parameter.

```
t_wrlat_adj = t_pd_wrlat - t_phy_wrlat
```

In the following waveform (Figure 9: Parameter t_wrlat_adj Waveform), showing DDR5, the WL is 24, the t_phy_wrdata is 2. From the waveform the t_phy_wrlat is 3. Therefore, the t_wrlat_adj is 20.

This example:
WL = 24
t_phy_wrdata = 2
(For DDR5) t_pd_wrlat = WL+1 – t_phy_wrdata = 24+1-2 = 23
t_wrlat_adj = t_pd_wrlat – t_phy_wrlat = 23 - 3 = 20

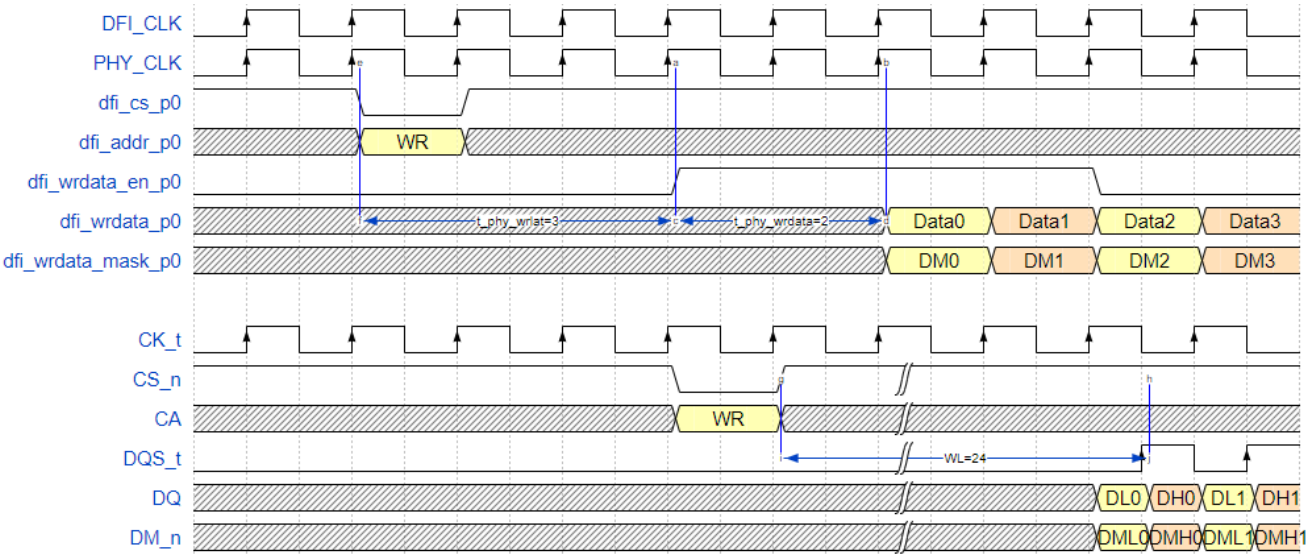


Figure 9: Parameter t_wrlat_adj Waveform

4.7 Parameter t_rddata_en_adj

This parameter is used to adjust the DFI PHY parameter t_rddata_en. The DFI PHY parameter t_rddata_en defines the number of *DFI PHY (PHY_CLK)* clock cycles delay between the read command on the DFI interface and the assertion of the dfi_rddata_en signal. in Section 4.8 of the DFI (DDR PHY Interface) 5.0 specification (April 27, 2018).

The Cadence Palladium/Protium DFI PHY model has an internal read latency itself, which varies based on protocol. This may need to be adjusted to match the latency of the users DFI PHY. The parameter t_rddata_en_adj is used for this purpose. The internal read latency of the DFI PHY model is calculated as follows:

```
DDR/Mobile_DDR:     t_pd_rddata_en = CL
DDR2/3/4:           t_pd_rddata_en = RL
DDR4 RDIMM:         t_pd_rddata_en = RL
DDR5:               t_pd_rddata_en = RL + 1
DDR5 RDIMM:         t_pd_rddata_en = RL + 5
LPDDR2/3:           t_pd_rddata_en = RL + 1 + t_DQSCK
LPDDR4:             t_pd_rddata_en = RL + tDQSCK + tDQSCK_half
LPDDR5:             t_pd_rddata_en = (RL+1)*ratio-1
                    (ratio=4 for CKR(WCK:CK)=4:1; ratio=2 for CKR(WCK:CK)=2:1)
GDDR6:              t_pd_rddata_en = RL + 1
                    t_pd_rddata_edc_en = RL + CRCRL + 1
```

Note: The default value of t_DQSCK in the LPDDR2/3 model is 1.
By default, tDQSCK=1 and tDQSCK_half=1 in LPDDR4 model.

Note: CL above refers to the DDR memory CAS Latency. Latency settings are controlled by Mode Registers in the memory device and are set by the memory controller during initialization. Please refer to JEDEC specification for more information.

Note: RL above refers to the DDR or LPDDR memory Read Latency.

- Read Latency for DDR and Mobile_DDR (LPDDR) is just the CL value (CAS Latency).
- Read Latency for DDR2/3 is the sum of CL (CAS Latency) and AL (Additive Latency) i.e. RL=AL+CL.
- Read Latency for DDR4 is the sum of CL (CAS Latency) and AL (Additive Latency) and Parity Latency i.e. RL=AL+CL+PL.
- For LPDDR2/3/4/5 and DDR5 and GDDR6 RL is directly defined by mode register content.
- For DDR4 RDIMM, if the RL value from mode registers setting is not equal to the real RL from read operation, use the RL value from read operation for this formula calculation.
- For GDDR6, t_rddata_edc_en_adj = t_pd_rddata_edc_en - t_rddata_edc_en. The t_rddata_edc_en is similar to t_rddata_en, which defines the number of PHY_CLK clock cycles delay between the read command on the DFI interface and the assertion of the dfi_rddata_edc_en signal.

The parameter t_rddata_en_adj is the difference between the actual PHY t_rddata_en parameter (t_rddata_en) and the DFI PHY model t_pd_rddata_en parameter.

```
t_rddata_en_adj = t_pd_rddata_en – t_rddata_en
```

In the following waveform (Figure 10: Parameter t_rddata_en_adj Waveform), showing DDR5, the RL is 26. From the waveform the t_rddata_en is 3. Therefore, the t_rddata_en_adj is 24.

This example:
RL = 26
t_rddata_en = 3
(For DDR5) t_pd_rddata_en = RL+1 = 27
t_rddata_en_adj = t_pd_rddata_en – t_rddata_en = 27 – 3 = 24

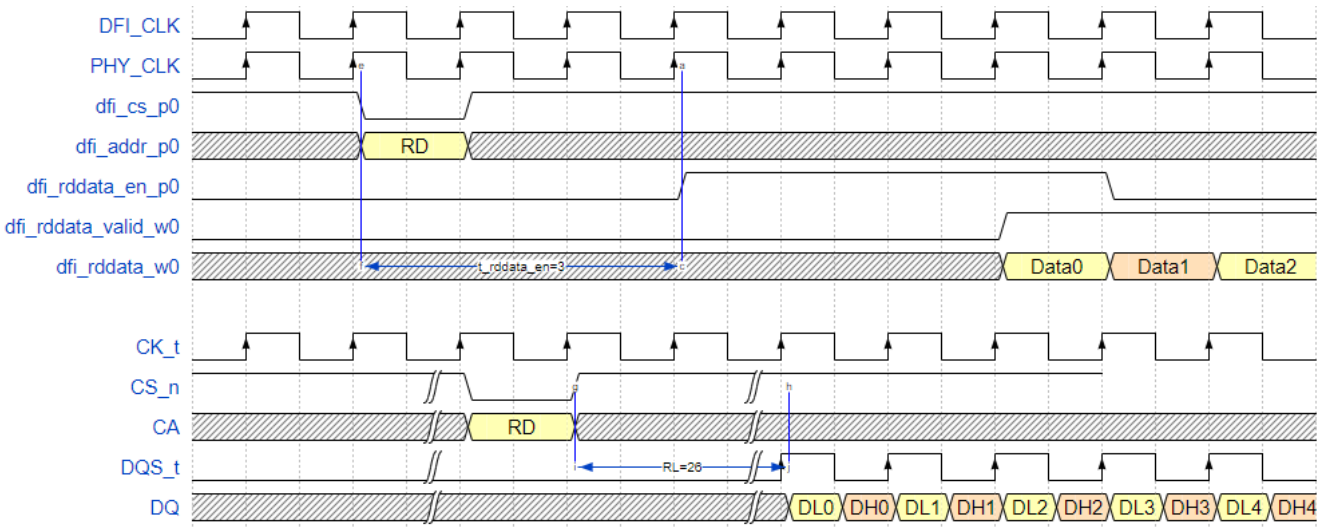


Figure 10: Parameter t_rddata_en_adj Waveform

4.8 Adjusting the Relationship Between dfi_rddata_en and dfi_rddata_valid (t_phy_rdlat)

The DFI PHY model does not provide a direct user controllable parameter for t_phy_rdlat by default. The DFI PHY model determines this latency based on other system parameters. The equation to calculate the relationship is shown below. The value of t_phy_rdlat here is based on DFI clock, not DFI PHY clock.

In 1:1 system
pd_t_phy_rdlat = t_ctrl_delay + t_rddata_en_adj + 3
In 1:2 system
pd_t_phy_rdlat = Floor ((1+t_ctrl_delay*2 + t_rddata_en_adj + 4)/2)
In 1:4 system
pd_t_phy_rdlat = Floor ((1+t_ctrl_delay*4 + t_rddata_en_adj + 8)/4)

Some users have the specific but rare requirement to adjust t_phy_rdlat and these users need the following steps to set a definite desired value for t_phy_rdlat:

- Step1 -- Define MMP_DFI_ADJUST_TPHY_RDLAT
- Step2 – Force xxx.dfiphy_lpddr4.MMP_DFIPHY.t_phy_rdlat_reg

There is a limitation that the forced value CANNOT be less than pd_t_phy_rdlat which is introduced above.

4.9 Parameter mem_data_bits

This parameter is used to define the width of the memory interface data bus.

4.10 Parameter dfi_addr_bits

This parameter is used to define the width of the DFI PHY interface address bus. If LPDDR2/3 is being used, then this should be set to 20bits wide as the CA bus is DDR (double data rate) of 10bits. LPDDR4 uses a 6-bit SDR (single data rate) CA bus and consequently there is a direct mapping from CA[5:0] to dfi_address[5:0]. LPDDR5 CA bus is DDR (double data rate) of 7bits and the dfi_addr_bits should be set to 14bits wide. GDDR6 CA bus is DDR (double data rate) of 10bits and the dfi_addr_bits should be set to 20bits wide. DDR5 CA bus is SDR (single data rate) of 14bits and the dfi_addr_bits should be set to 14bits wide.

4.11 Parameter dfi_bank_addr_width

This parameter is used to define the width of the DFI PHY interface bank bus and is only required for the DDR/DDR2/DDR3/DDR4 models.

4.12 Parameter dfi_bg_addr_width

This parameter is used to define the width of the DFI PHY interface bank group bus. It is only required for model DDR4.

4.13 Parameter dfi_cid_width

This parameter is used to define the width of the DFI PHY interface chip ID. The parameter is only required for DDR4 and DDR5.

4.14 Parameter dfi_cs_width

This parameter is used to define the width of the DFI PHY chip select bus.

4.15 Parameter mem_ca_bits

This parameter is used to define the width of the memory interface bus for LPDDR2/3/4 implementations. It is not used for DDR/DDR2/DDR3/DDR4 models. This parameter should be set to 10bits for LPDDR2/3 and GDDR6, and be set to 6bits for LPDDR4, and be set to 7bits for LPDDR5, and be set to 14bits for DDR5.

4.16 Initialization of the PHY

The DFI PHY model does not support any dynamic PHY initialization or PHY register programming interfaces that are specific to the implementation and not part of the DFI specification.

Although the DFI model does not require any initialization it does support the DFI initialization sequence.

If the memory controller uses this interface then two interface pins, dfi_init_start and dfi_init_complete, can be connected between the DFI PHY model and the controller.

If the memory controller does not support this interface, then tie off the dfi_init_start input to the DFI PHY model to zero and also set the define below in the model RTL.

```
`define MMP_NO_DFI_INIT_START
```

4.17 Modifying Timing Parameters at Runtime

The following mechanism allows users to change the DFI PHY timing parameters at runtime without compiling the model again.

The below timing parameter registers already have 'keep_net' in the model RTL:

t_ctrl_delay_reg
t_phy_wrdata_reg
t_wrlat_adj_reg
t_rddata_en_adj_reg
t_rddata_edc_en_adj_reg
t_shift_p1_adjust_delay_reg
t_shift_p1_adjust_delay_wr_reg
t_phy_rdlat_reg

After download and during runtime use the "force" command to change the values of timing parameters. For example:

```
force TB.phy_mem0.phy_lpddr4.MMP_DFIPHY.t_ctrl_delay_reg    2
force TB.phy_mem0.phy_lpddr4.MMP_DFIPHY.t_phy_wrdata_reg    1
force TB.phy_mem0.phy_lpddr4.MMP_DFIPHY.t_wrlat_adj_reg     2
force TB.phy_mem0.phy_lpddr4.MMP_DFIPHY.t_rddata_en_adj_reg 2
```

Note: There is another timing parameter t_phy_rdlat which can also be changed during runtime. The ability to adjust this parameter is not needed by most users See section 4.8 for the details on this adjustment. For example:

```
force TB.phy_mem0.phy_lpddr4.MMP_DFIPHY.t_phy_rdlat_reg    6
```

Note: For DDR5 and DDR5 RDIMM, there is difference for the parameter value and the forcing register value in 1N mode and 2N mode. Refer to Table 6 for details.

Table 6: The forcing value difference between parameters and registers

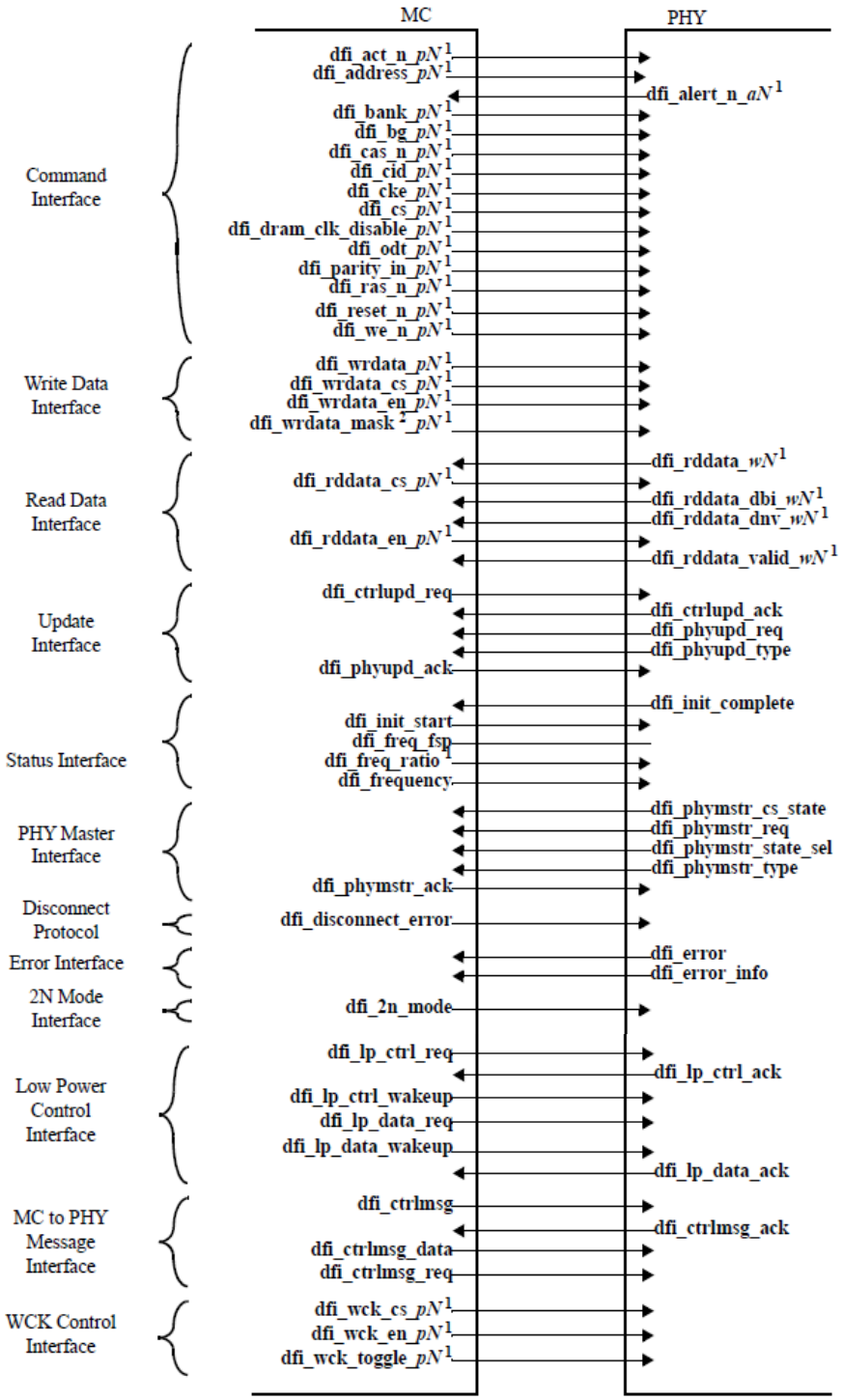
Model Type	**1N/2N Mode**	**t_wrlat_adj_reg**	**t_rddata_en_adj_reg**	**Other timing parameter registers** t_ctrl_delay_reg t_phy_wrdata_reg t_rddata_edc_en_adj_reg t_shift_p1_adjust_delay_reg t_shift_p1_adjust_delay_wr_reg t_phy_rdlat_reg
	1N Mode	t_wrlat_adj	t_rddata_en_adj	Same as parameter value

DFIPHY5 for DDR5	2N Mode	t_wrlat_adj + 1	t_rddata_en_adj + 1	Same as parameter value
DFIPHY5 for DDR5 RDIMM	1N Mode	t_wrlat_adj - 3	t_rddata_en_adj - 3	Same as parameter value
	2N Mode	t_wrlat_adj	t_rddata_en_adj	Same as parameter value
DFIPHY5 for other SDRAMs		t_wrlat_adj	t_rddata_en_adj	Same as parameter value

5. Model Block Diagram

5.1 Interface of Memory Controller Port

The interface of the Memory Controller Port to the PHY are shown below.



1. Optional suffix for frequency ratio systems.
2. Dual-function signal. In DDR4/LPDDR4 systems with write DBI enabled, the signal transforms from a mask to a write DBI signal.

Italicized text indicates that the phase/word/cycle is optional.

Figure 11: Memory Controller Port to PHY Interface

5.2 Interface of Memory Device Port

The interface between the DFI PHY and the MMP memory model – DDRx or LPDDRx — is shown below.

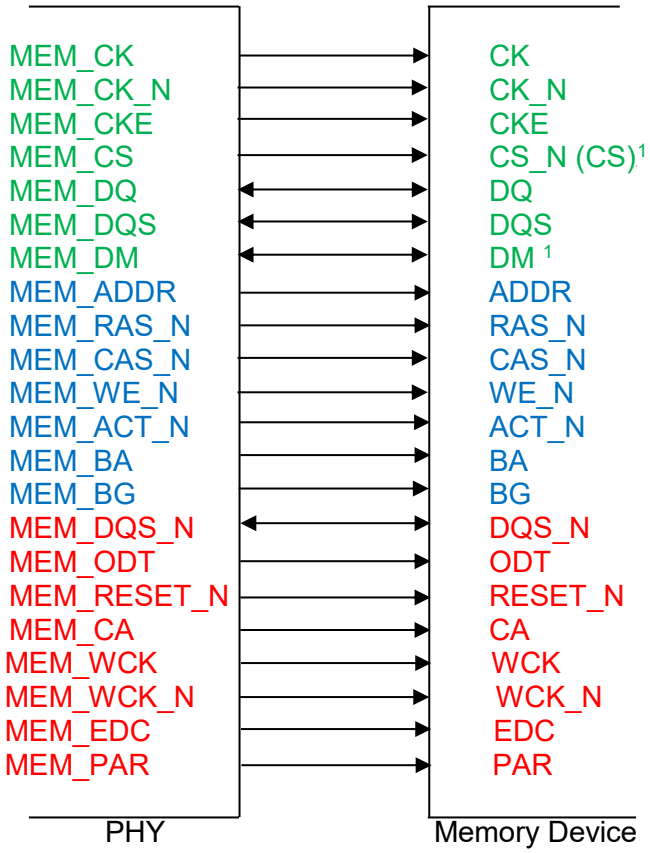


Figure 12: DFI PHY to MMP DDRx/LPDDRx Interface

The above diagram and the following table show the necessary groups of signals to connect between the DFI PHY model and the Cadence Palladium/Protium memory model. In the figure above there are common signals for all protocols, marked in GREEN, and other protocol-specific signals that are marked in BLUE and RED.

1. The polarity of the MEM_CS and MEM_DM is defined by the polarity of the corresponding memory signal. The MEM_CS and MEM_DM ports should connect to memory port directly without caring about the polarity.

Table 7: DFI PHY to MMP DDRx / LPDDRx Signal Connections Required

	DDR	DDR2	DDR3	DDR4	DDR5	MOBILE DDR	LPDDR2	LPDDR3	LPDDR4	LPDDR5	GDDR6
		Ports Between Controller and PHY									
DFI_CLK	√	√	√	√	√	√	√	√	√	√	√
PHY_CLK	√	√	√	√	√	√	√	√	√	√	√
DAT_CLK										√	
RST_N	√	√	√	√	√	√	√	√	√	√	√
dfi_address_px	√	√	√	√	√	√	√	√	√	√	√
dfi_bank_px	√	√	√	√		√					
dfi_ras_n_px	√	√	√	√		√					
dfi_cas_n_px	√	√	√	√		√					
dfi_we_n_px	√	√	√	√		√					
dfi_cs_px	√	√	√	√	√	√	√	√	√	√	√
dfi_act_n_px				√							
dfi_bg_px				√							
dfi_cid_px				√	√						
dfi_cke_px	√	√	√	√		√	√	√	√		√
dfi_odt_px		√	√	√							
dfi_reset_n_px			√	√	√				√	√	√
dfi_wrdata_en_px[1]	√	√	√	√	√	√	√	√	√	√	√
dfi_wrdata_px	√	√	√	√	√	√	√	√	√	√	√
dfi_wrdata_cs_n_px[2]	√	√	√	√	√	√	√	√	√	√	√
dfi_wrdata_mask_px	√	√	√	√	√	√	√	√	√	√	√
dfi_wrdata_ecc_px										√	
dfi_rddata_en_px[1]	√	√	√	√	√	√	√	√	√	√	√
dfi_rddata_wx	√	√	√	√	√	√	√	√	√	√	√
dfi_rddata_cs_n_px[2]	√	√	√	√	√	√	√	√	√	√	√
dfi_rddata_valid_wx	√	√	√	√	√	√	√	√	√	√	√
dfi_rddata_dbi_wx				√	√				√	√	√
dfi_rddata_edc_en_px											√
dfi_rddata_edc_wx											√
dfi_rddata_edc_valid_wx											√
dfi_ctrlupd_req	√	√	√	√	√	√	√	√	√	√	√
dfi_ctrlupd_ack	√	√	√	√	√	√	√	√	√	√	√
dfi_dram_clk_disable	√	√	√	√	√	√	√	√	√	√	√
dfi_freq_ratio	√	√	√	√	√	√	√	√	√	√	√
dfi_init_complete	√	√	√	√	√	√	√	√	√	√	√
dfi_init_start	√	√	√	√	√	√	√	√	√	√	√
dfi_parity_in_px	√	√	√	√	√	√	√	√	√		
dfi_alert_n_ax				√	√						
dfi_lp_ctrl_req	√	√	√	√	√	√	√	√	√	√	√
dfi_lp_data_req	√	√	√	√	√	√	√	√	√	√	√
dfi_lp_ctrl_wakeup	√	√	√	√	√	√	√	√	√	√	√
dfi_lp_data_wakeup	√	√	√	√	√	√	√	√	√	√	√
dfi_lp_ctrl_ack	√	√	√	√	√	√	√	√	√	√	√

dfi_lp_data_wakeup	√	√	√	√	√	√	√	√	√	√	√
		Ports Between PHY and Memory									
MEM_DQ	√	√	√	√	√	√	√	√	√	√	√
MEM_DQS	√	√	√	√	√	√	√	√	√	√	
MEM_CK	√	√	√	√	√	√	√	√	√	√	√
MEM_CK_N	√	√	√	√	√	√	√	√	√	√	√
MEM_CKE	√	√	√	√		√	√	√	√		√
MEM_CS	√	√	√	√	√	√	√	√	√	√	
MEM_DM	√	√	√	√	√	√	√	√	√	√	√
MEM_PAR	√	√	√	√	√	√	√	√	√		
MEM_RESET_N			√	√	√				√	√	√
MEM_ODT		√	√	√							
MEM_DQS_N		√	√	√	√		√	√	√	√	
MEM_RAS_N	√	√	√	√		√					
MEM_BA	√	√	√	√		√					
MEM_CAS_N	√	√	√	√		√					
MEM_WE_N	√	√	√	√		√					
MEM_ADDR	√	√	√	√		√					
MEM_CA					√		√	√	√	√	√
MEM_BG				√							
MEM_CID				√							
MEM_ACT_N				√							
MEM_ALERT_N				√	√						
MEM_WCK										√	√
MEM_WCK_N										√	√
MEM_EDC											√
MEM_CABI											√
rd_oe_dq										√	√
rd_oe_edc											√

NOTE:

1. As described in DFIPHY5.0 specification, all bits of the dfi_wrdata_en signal are identical. For MMP DFIPHY5.0 model, the dfi_wrdata_en_px is only 1 bit. If the width is different from controller output, it is ok to just connect the bit0 to MMP DFIPHY5.0 model. This is same for dfi_rddata_en_px.

2. The dfi_wrdata_cs_n_px and dfi_rddata_cs_n_px are not used for MMP DFIPHY model, user can ignore them for connection.

5.3 Frequency Ratio and Port Suffixes

The DFI model supports all frequency ratios: 1:1, 1:2 and 1:4. The dfi_freq_ratio input port on the DFI PHY model must be statically set to the correct frequency ratio for the design. From the DFI specification this is encoded as shown in the table below (Table 7: dfi_freq_ratio Input Port Settings).

Table 8: dfi_freq_ratio Input Port Settings

dfi_freq_ratio	MC:PHY frequency ratio
2'b00	1:1
2'b01	1:2
2'b10	1:4
2'b11	Reserved

The DFI model ports are named with the frequency ratio naming style, so all appropriate ports that are affected by freq_ratio mode will have the _p0, _p1, _p2 and _p3 and _w0, _w1, _w2 and _w3 suffixes.

If a 1:1 frequency ratio is required, then _p0 and _w0 ports should be connected to the memory controller.

For 1:2 systems _p0, _p1 and _w0, _w1 ports should be connected.

Finally, for 1:4 systems all four ports should be connected.

For LPDDR5, the dfi_freq_ratio is the ratio between PHY_CLK and DAT_CLK. The ratio is fixed 1:1 between DFI_CLK and PHY_CLK. The DFI command interface will only use P0. The DFI data interface will use P0/P1/P2/P3 for 1:4 system and will use P0/P1 for 1:2 system.

If the user's memory controller does not use the DFI specification naming convention and the memory controller uses freq ratio of 1:2 or 1:4, then some of the ports, like dfi_wrdata and dfi_rddata from the memory controller may be twice or four times as wide. In this case, the user should split the control and address signals into groups to connect to the appropriate _p0 and _p1 ports. For example, to connect dfi_address to the DFI PHY model connect the lower half to dfi_address_p0 and the upper half to dfi_address_p1, and so on. See below.

5.4 Example Instantiation of a DFI PHY model

The following (Figure 13: Example Instantiation of a DFI PHY Model Configured for DDR4) is an example of the instantiation of a DFI PHY model configured for DDR4 mode using freq ratio of 1:2.

```
// DDR PHY interface
dfiphy_ddr4 #(
    .dfi_cs_width                   (1              ),
    .dfi_addr_bits                  (24             ),
    .dfi_bank_addr_width            (3              ),
    .mem_data_bits                  (16             ),
    .t_ctrl_delay                   (2              ),
    .t_phy_wrdata                   (1              ),
    .t_wrlat_adj                    (3              ),
    .t_rddata_en_adj                (2              ),
    .t_shift_p1_adjust_delay        (0              )
)       DFI_0(
    .DFI_CLK                (MC_CLK                 ),
    .PHY_CLK                (PHY_CLK                ),
    .RST_N                  (SYS_XRST               ),

    .dfi_bank_p0            ( dfi_bank[2:0]         ), // Split Bank and other
    .dfi_cas_n_p0           ( dfi_cas_n[0]          ), // bused signals across
    .dfi_ras_n_p0           ( dfi_ras_n[0]          ), // p0 and p1 ports.
    .dfi_we_n_p0            ( dfi_we_n[0]           ),
    .dfi_act_n_p0           ( dfi_act_n[0]          ),
    .dfi_bank_p1            ( dfi_bank[5:3]         ),
    .dfi_cas_n_p1           ( dfi_cas_n[1]          ),
    .dfi_ras_n_p1           ( dfi_ras_n[1]          ),
    .dfi_we_n_p1            ( dfi_we_n[1]           ),
    .dfi_act_n_p1           ( dfi_act_n[1]          ),
    .dfi_bank_p2            ( 3'b0                  ), // Tie off unused PHY
    .dfi_cas_n_p2           ( 1'b1                  ), // inputs to inactive
    .dfi_ras_n_p2           ( 1'b1                  ), // state
    .dfi_we_n_p2            ( 1'b1                  ),
    .dfi_act_n_p2           ( 1'b1                  ),
    .dfi_bank_p3            ( 3'b0                  ),
    .dfi_cas_n_p3           ( 1'b1                  ),
    .dfi_ras_n_p3           ( 1'b1                  ),
    .dfi_we_n_p3            ( 1'b1                  ),
    .dfi_act_n_p3           ( 1'b1                  ),

    .dfi_address_p0         ( dfi_address[23:0]     ),
    .dfi_bg_p0              ( dfi_bg[1:0]           ),
    .dfi_cke_p0             ( dfi_cke[0]            ),
    .dfi_cs_p0              ( dfi_cs_n[0]           ),
    .dfi_address_p1         ( dfi_address[47:24]    ),
    .dfi_bg_p1              ( dfi_bg[3:2]           ),
    .dfi_cke_p1             ( dfi_cke[1]            ),
    .dfi_cs_p1              ( dfi_cs_n[1]           ),
    .dfi_address_p2         ( 24'b0                 ),
    .dfi_cke_p2             ( 1'b0                  ),
    .dfi_cs_p2              ( 1'b1                  ),
    .dfi_address_p3         ( 24'b0                 ),
    .dfi_cke_p3             ( 1'b0                  ),
    .dfi_cs_p3              ( 1'b1                  ),

    .dfi_odt_p0             ( 1'b0                  ),
    .dfi_odt_p1             ( 1'b0                  ),
    .dfi_odt_p2             ( 1'b0                  ),
```

```
.dfi_odt_p3             ( 1'b0                  ),
.dfi_reset_n_p0         ( s_dfi_reset_n[0]      ),
.dfi_reset_n_p1         ( s_dfi_reset_n[1]      ),
.dfi_reset_n_p2         ( 1'b1                  ),
.dfi_reset_n_p3         ( 1'b1                  ),
.dfi_cid_p0             ( 3'b0                  ),
.dfi_cid_p1             ( 3'b0                  ),
.dfi_cid_p2             ( 3'b0                  ),
.dfi_cid_p3             ( 3'b0                  ),
.dfi_alert_n_a0         ( dfi_alert_n[0]        ),
.dfi_alert_n_a1         ( dfi_alert_n[1]        ),
.dfi_alert_n_a2         ( dfi_alert_n[2]        ),
.dfi_alert_n_a3         ( dfi_alert_n[3]        ),

.dfi_wrdata_p0          ( dfi_wrdata[31:0]      ),
.dfi_wrdata_en_p0       ( dfi_wrdata_en[0]      ),
.dfi_wrdata_mask_p0     ( dfi_wrdata_mask[1:0]  ),
.dfi_wrdata_p1          ( dfi_wrdata[63:32]     ),
.dfi_wrdata_en_p1       ( dfi_wrdata_en[1]      ),
.dfi_wrdata_mask_p1     ( dfi_wrdata_mask[3:2]  ),
.dfi_wrdata_p2          ( 32'b0                 ),
.dfi_wrdata_en_p2       ( 1'b0                  ),
.dfi_wrdata_mask_p2     ( 2'b0                  ),
.dfi_wrdata_p3          ( 32'b0                 ),
.dfi_wrdata_en_p3       ( 1'b0                  ),
.dfi_wrdata_mask_p3     ( 2'b0                  ),
.dfi_rddata_en_p0       ( s_dfi_rddata_en[0]    ),
.dfi_rddata_w0          ( s_dfi_rddata[31:0]    ),
.dfi_rddata_valid_w0    ( s_dfi_rddata_valid[0] ),
.dfi_rddata_dbi_w0      ( s_dfi_rddata_dbi_n[3:0]),
.dfi_rddata_en_p1       ( s_dfi_rddata_en[1]    ),
.dfi_rddata_w1          ( s_dfi_rddata[63:32]   ),
.dfi_rddata_valid_w1    ( s_dfi_rddata_valid[1] ),
.dfi_rddata_dbi_w1      ( s_dfi_rddata_dbi_n[7:4]),
.dfi_rddata_en_p2       ( 1'b0                  ),
.dfi_rddata_w2          (                       ), // Leave unused PHY
.dfi_rddata_valid_w2    (                       ), // outputs floating
.dfi_rddata_dbi_w2      (                       ),
.dfi_rddata_en_p3       ( 1'b0                  ),
.dfi_rddata_w3          (                       ),
.dfi_rddata_valid_w3    (                       ),
.dfi_rddata_dbi_w3      (                       ),

.dfi_ctrlupd_req        ( dfi_ctrlupd_req       ),
.dfi_ctrlupd_ack        ( dfi_ctrlupd_ack       ),

.dfi_dram_clk_disable   ( dfi_dram_clk_disable  ),
.dfi_init_complete      ( dfi_init_complete     ),
.dfi_init_start         ( dfi_init_start        ),
.dfi_freq_ratio         ( 2'b01                 ),

.dfi_lp_ctrl_req        ( dfi_lp_ctrl_req       ),
.dfi_lp_data_req        ( dfi_lp_data_req       ),
.dfi_lp_ctrl_wakeup     ( dfi_lp_ctrl_wakeup    ),
.dfi_lp_data_wakeup     ( dfi_lp_data_wakeup    ),
.dfi_lp_ctrl_ack        ( dfi_lp_ctrl_ack       ),
.dfi_lp_data_ack        ( dfi_lp_data_ack       ),

.dfi_parity_in          ( dfi_parity_in         ),

.MEM_DQ                 ( dq[15:0]              ),
.MEM_DQS                ( dqs[1:0]              ),
.MEM_CK                 ( clk                   ),
```

```
    .MEM_CK_N               ( clk_n                 ),
    .MEM_CKE                ( cke                   ),
    .MEM_CS                 ( cs_n                  ),
    .PAR_IN                 ( par_in                ),
    .MEM_RESET_N            ( reset_n               ),
    .MEM_ODT                ( odt                   ),
    .MEM_DQS_N              ( dqs_n[1:0             ],
    .MEM_RAS_N              ( ras_n                 ),
    .MEM_ACT_N              ( act_n                 ),
    .MEM_BA                 ( ba[2:0]               ),
    .MEM_BG                 ( bg[1:0]               ),
    .MEM_CAS_N              ( cas_n                 ),
    .MEM_WE_N               ( we_n                  ),
    .MEM_ADDR               ( addr[13:0             ],
    .MEM_DM                 ( dm[1:0]               ),
    .MEM_ALERT_N            ( alert_n               )
);
```

Figure 13: Example Instantiation of a DFI PHY Model Configured for DDR4

For the LPDDR2 and LPDDR3 DFI PHY models and GDDR6 PHY model, the dfi_address ports are 20 bits, where the lower 10 bits contain the rising edge CA value and the upper 10 bits contain the falling edge CA value. In a freq ratio 1:2 system where the controller has a single bus output, split this bus on the 20 bit boundary. For LPDDR4 DFI PHY model, the dfi_address port is 6 bits. In a freq ratio 1:2 system the controller needs to send commands both on phase0 and phase1 to align with the SDR CA bus. For LPDDR5 DFI PHY model, the dfi_address port is 14bits, where the lower 7bits contain the rising edge CA value and the upper 7bits contain the falling edge CA value. For DDR5 DFI PHY model, the dfi_address port is 14 bits. In a freq ratio 1:2 system the controller needs to send commands both on phase0 and phase1 to align with the SDR CA bus.

Note that the control signals dfi_cas_n_pN, dfi_ras_pN and dfi_we_n_pN signal widths are fixed to one. Similarly, dfi_rddata_en_pN and dfi_wrdata_en_pN signal widths are also fixed to one. For the case where the controller supports multiple data slices the user should connect slice bit 0 from the controller dfi_rddata_en_pN and dfi_wrdata_en_pN ports to the equivalent ports on the PHY.

6. Phase 0/1 alignment Special Case

In some PHY implementations a special case occurs for 1:2 frequency ratio systems where the PHY changes the t_phy_wrlat and t_rddata_en latencies to be different based on the phase of the command is issued. For example, the latency from a command on Phase 0 is ‘6’ but the same latency is only ‘5’ if the command is issued on Phase 1. This forces the dfi_rddata_valid for both phases and/or the dfi_wrdata_en for both phases to always be asserted together.

The user should first determine if this special case applies to their environment.

If the controller at hand is a Cadence controller, then this situation does not apply and no phase adjustment is needed. Let us look at an example for read data to understand why. If a user sets t_rddata_en=6 with a Cadence controller, the latency between command (control signals) and t_rddata_en on both phases is the same value. So, whenever the user sends a read command on phase0 or on phase1, the model uses that same value to derive the related parameters. Therefore, the user should input the latency value in parameter calculations *as is*.

If controller is unknown or not a Cadence controller, then first measure the read and write latencies to see if each is the same value on phase0 and phase1. If the read latency is the same for phase0 and phase1 and the write latency is the same for phase0 and phase1, then use the read and write latency values in parameter calculations with no modification. If, however, either or both the read and write latencies on phase0 and phase1 are not the same, then configure the DFI PHY model to handle the case as follows:

- Use the phase0 read & write latency values in parameter calculations detailed above
- Set Verilog macro MMP_DFI_ADJUST_P1
- Set the corresponding special t_shift_p1_adjust_delay and/or t_shift_p1_adjust_delay_wr parameter(s).

As outlined above, the following Verilog macro should be defined to enable the phase adjust functionality. Note that this Verilog macro was named ADUST_P1 in older DFI PHY models.

```
`define MMP_DFI_ADJUST_P1
```

In order to support the phase adjust capability in the DFI PHY model, up to two additional parameters are required. These parameters are not part of the DFI specification. The parameters—*t_shift_p1_adjust_delay* and *t_shift_p1_adjust_delay_wr*—need to be set to align the enable and data signals for the reads and writes, respectively, when the MMP_DFI_ADJUST_P1 mode is required. In some cases the enable and data signals for only one operation—read or write—requires adjustment. The equations for these parameters are shown below.

```
t_shift_p1_adjust_delay    = t_rddata_en – 2;
t_shift_p1_adjust_delay_wr = t_phy_wrlat – 1;
```

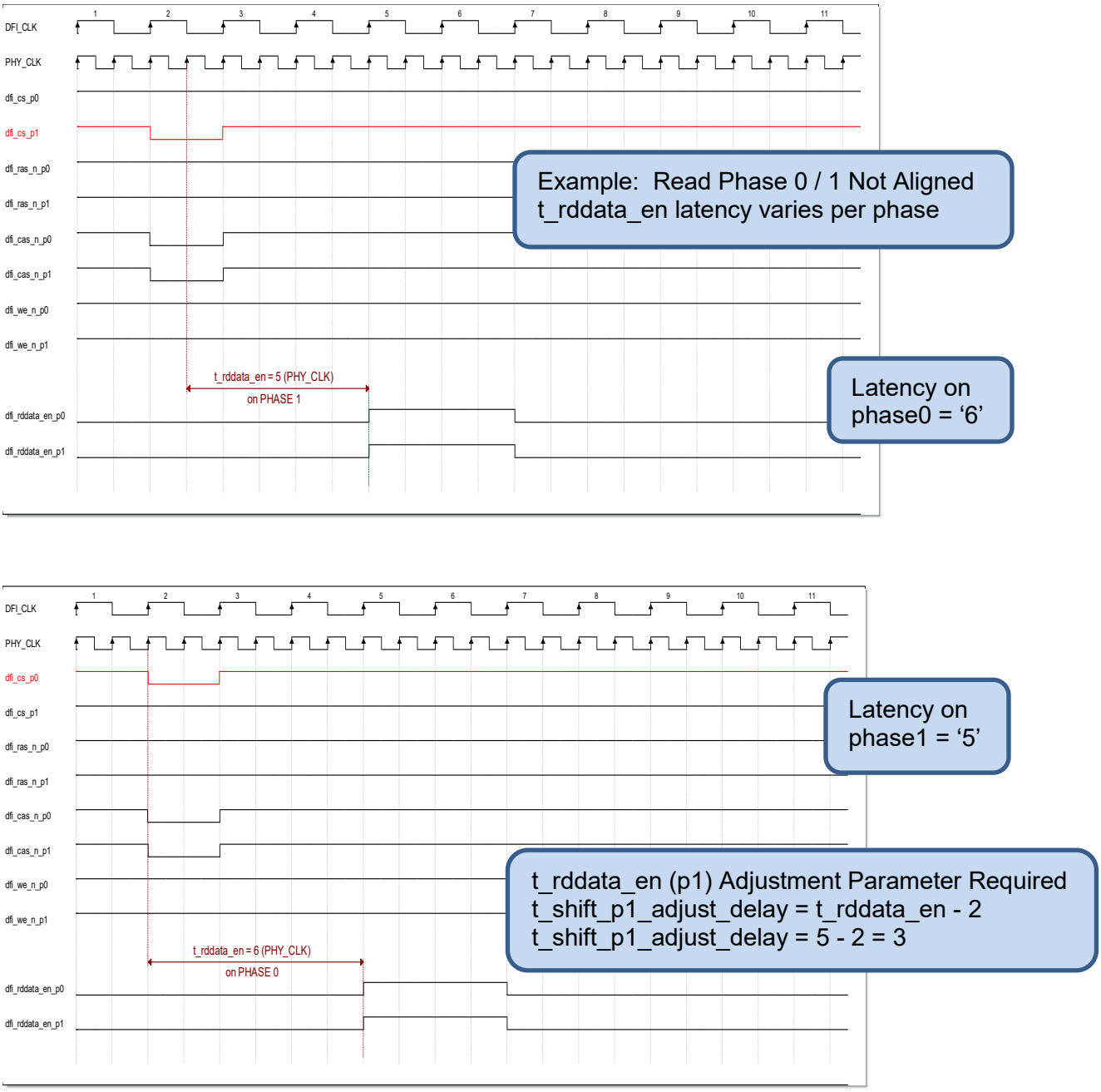


Figure 14: Example Phase Adjustment for Read Operation

7. PHY Init Mode

The MMP DFI PHY 5.0 model can enable PHY init mode by setting the parameter 'phy_init_mode=1' when doing instantiation.

The PHY init mode is only used when the controller didn't send Mode Register Write for SDRAM memory configuration through DFI interface. When the PHY init mode is enabled, MMP DFI PHY 5.0 will read the Mode Register configuration from a preload data file and issue Mode Register Write command sequence on the SDRAM memory interface.

Currently this feature is supported in the DFI PHY 5.0 model for LPDDR5, DDR5, LPDDR4 and DDR4, and not supported in the DFI PHY 5.0 model for other SDRAMs, including DDR5 RDIMM.

Here is the example for preload the data file of Mode Register configuration. The name of small array is "*array_mr_cfg_info*".

memory -load %readmemh TB.phy_mem0.DFI_0.MMP_DFIPHY.array_mr_cfg_info dfiphy5_lpddr5_phy_init_mr.dat

Here is the example of the data file content in *dfiphy5_lpddr5_phy_init_mr.dat*.
1_03_0000_0020
1_01_0000_00B0
1_02_0000_00BB
0_00_0000_0000

Each line is the configuration for one mode register and totally 41bits. The data structure is *{entry_vld[0], mr_index[7:0], mr_content[31:0]}.*
For example, the first line '*1_03_0000_0020*' means 'Write MR3 with value 32'h0000_0020'. The last line '*0_00_0000_0000*' means the end of the mode register configuration sequence. The mr_index and mr_content are in **HEX** number.

For LPDDR5, DDR5 and LPDDR4, the mode register width is 8bits: {OP[7:0]}.
For DDR4, the mode register width is 19bits: {BG[1:0], BA[1:0], A17, A[13:0]}.

8. File list for each SDRAM DFI PHY 5.0 model

The below table lists all the required files for each SDRAM DFI PHY 5.0 model.

Table 9: File list for DFIPHY 5.0 models

SDRAM TYPE	File list
DDR1	dfiphy5_ddr.v dfiphy5_pd.vp
DDR2	dfiphy5_ddr2.v dfiphy5_pd.vp
DDR3	dfiphy5_ddr3.v dfiphy5_pd.vp
DDR4	dfiphy5_ddr4.v dfiphy5_pd.vp
DDR5	dfiphy5_ddr5.v dfiphy5_pd.vp
DDR5 RDIMM	dfiphy5_ddr5_rdimm.v dfiphy5_pd.vp
LPDDR1	dfiphy5_lpddr.v dfiphy5_pd.vp
LPDDR2	dfiphy5_lpddr2.v dfiphy5_pd.vp
LPDDR3	dfiphy5_lpddr3.v dfiphy5_pd.vp
LPDDR4	dfiphy5_lpddr4.v dfiphy5_pd.vp
LPDDR5	dfiphy5_lpddr5.v dfiphy5_pd.vp
GDDR6	dfiphy5_gddr6.v dfiphy5_pd.vp dfiphy5_gddr6_intf_trans.vp
GDDR6 with CDNS PHY interface	mmp_cdns_gddr6_phy_wrapper.v dfiphy5_gddr6.v dfiphy5_pd.vp dfiphy5_gddr6_intf_trans.vp

9. Limitations

1. DFI_CLK and PHY_CLK must be phase aligned on rising edge. The DFI_CLK and PHY_CLK and DAT_CLK must be phase aligned on rising edge for LPDDR5 DFI PHY 5.0 model.
2. All DFI port signals from MC are required to be driven by registers referenced to a rising edge of the DFI clock and no delay.
3. The PHY master interface and Message interface are not supported.
4. Signal “dfi_data_dnv_wN” for LPDDR2/3/4 is not supported.
5. Any PHY implementation specific interfaces are not supported.
6. For GDDR6, if the wrapper ‘mmp_cdns_gddr6_phy_wrapper.v’ is NOT used, the user needs to add ‘pullup/pulldown’ for EDC pin to configure the GDDR6 x16/x8 mode.
   x16 Mode:
   pullup(EDC_A[0]); pullup(EDC_A[1]); pullup(EDC_B[0]); pullup(EDC_B[1]);
   x8 Mode:
   pullup(EDC_A[0]); pulldown(EDC_A[1]); pulldown(EDC_B[0]); pullup(EDC_B[1]);

10. Synthesis & Compilation

The Cadence Palladium/Protium DFI PHY models should only be used with Cadence Palladium/Protium memory device models.

10.1 Palladium

The model is provided as protected RTL file(s). The files need to be synthesized prior to compiling for Palladium. An example of the command for compilation (including synthesis) and run of this model in IXCOM flow is shown below:

```
ixcom    -64bit +sv -ua \
         +dut+dfiphy5_lpddr4 \
         ./dfiphy5_lpddr4.v \
         ./dfiphy5_pd.vp \
         ./lpddr4_pd.vp \
         ./tb_top.v \
         -incdir <mmp_release_install_path>/utils/cdn_mmp_utils/sv \
        <mmp_release_install_path>/utils/cdn_mmp_utils/sv/cdn_mmp_utils.sv \
         ……

xeDebug -64 --ncsim \
    -sv_lib \
<mmp_release_install_path>/utils/cdn_mmp_utils/lib/64bit/libMMP_utils.so -- \
    -input auto_xedebug.tcl
```

The scripts below show two examples for Palladium classic ICE synthesis:

1)

```
 hdlInputFile -add dfiphy5_lpddr4.v
 hdlInputFile -add dfiphy5_pd.vp
 hdlInputFile -add lpddr4_pd.vp
 hdlInputFile -add <user_top_level_model>.v
 hdlImport -full -2001 -l qtref
 hdlOutputFile -add -f Verilog <user_top_level_model>.vg
 hdlSynthesize -memory -keepRtlSymbol -keepAllFlipFlop <user_top_level_model>
 ……
```

2)

```
 vavlog      dfiphy5_lpddr4.v \
             dfiphy5_pd.vp \
             lpddr4_pd.vp \
             <user_top_level_model>.v \
             ……
 vaelab      -keepRtlSymbol -keepAllFlipFlop -outputVlog
 <user_top_level_model>.vg <user_top_level_model>
```

- **Manual Configuring of this MMP Model Family**

  This MMP model supports manual configuration by accompanying the model mode register or configuration register declarations with synthesis directives, such as keep_net directives, that instruct the compiler to ensure that the relevant nets remain available for runtime forcing. For a general description of this support please see the user guide in the MMP release with path and filename *docs/MMP_FAQ_for_All_Models.pdf*.

  While MMP strongly recommends following protocol-based commands to configure MMP models, MMP recognizes that the design test environment may desire to trade off the risks inherent in streamlining or circumventing the initialization sequence part of the protocol in order to better support some testing environments.

  The following table lists the internal register path and naming along with the specification or datasheet naming for model mode registers or configuration registers that are accompanied by keep_net synthesis directives in support of such manual configuration. ONLY writeable configuration registers or fields are supported thusly. Please read the relevant datasheet for details about individual register behavior and mapping to fields.

Table 10: Writeable Mode Register / Configuration Register Info

Hierarchical RTL Naming for Writeable Configuration Related Registers & Signals	Specification or Vendor Datasheet Naming for Configuration Related Registers	Access
<model_name>.t_ctrl_delay_reg	the register of parameter t_ctrl_delay	
<model_name>.t_phy_wrdata_reg	the register of parameter t_phy_wrdata	
<model_name>.t_wrlat_adj_reg	the register of parameter t_wrlat_adj	
<model_name>.t_rddata_en_adj_reg	the register of parameter t_rddata_en_adj	
<model_name>.t_rddata_edc_en_adj_reg	the register of parameter t_rddata_edc_en_adj	
<model_name>.t_shift_p1_adjust_delay_reg	the register of parameter t_shift_p1_adjust_delay	
<model_name>.t_shift_p1_adjust_delay_wr_reg	the register of parameter t_shift_p1_adjust_delay_wr	
<model_name>.t_phy_rdlat_reg	the register of parameter t_phy_rdlat	

Note: please refer to Table 6 for the forcing value difference between parameters and registers.

10.2 Protium

Starting with Protium 19.11 release, IXCOM is supported for the Protium X series platform. IXCOM is a transaction-based acceleration (TBA) tool for the Protium X series. This tool enables evaluation of non-synthesizable part of the design on Xcelium simulator and the RTL part of the design on Protium prototyping platform. IXCOM involves xeCompile to compile the HDL for FPGA prototyping and xrun for non-synthesizable testbench.

10.2.1 In-Circuit Emulation (ICE) Flow

Protium ICE compilation is identical to Palladium ICE compilation, except ATB mode. ATB constructs are not supported in Protium. Refer to *Protium User Guide* for more information.

There are two options supported in Protium ICE compilation. See the examples below for details. The first example shows the hdl* command defined in the compile script file (for example it's called *compile.qel* below) and use xeCompile to do the RTL compile and synthesis. The second example shows using vavlog/vaelab commands to do the same thing as first example.

1) compile.qel:

```
hdlInputFile –add dfiphy5_lpddr4.v
hdlInputFile –add dfiphy5_pd.vp
hdlInputFile –add lpddr4_pd.vp
hdlInputFile –add <user_top_level_model>.v
hdlImport –full -2001 –l qtref
hdlOutputFile –add –f Verilog <user_top_level_model>.vg
hdlSynthesize –memory –keepRtlSymbol –keepAllFlipFlop <user_top_level_model>
  ……
```

2)

```
vavlog      dfiphy5_lpddr4.v \
            dfiphy5_pd.vp \
            lpddr4_pd.vp \
            <user_top_level_model>.v \
            ……
vaelab      –keepRtlSymbol –keepAllFlipFlop –outputVlog
<user_top_level_model>.vg <user_top_level_model>
```

10.2.2 IXCOM Stub and Check (SC) Flow

The IXCOM Stub and Check flow is available on both Protium X series and Protium S1 platforms. IXCOM Stub and Check flow enables use of constructs which are synthesizable in IXCOM flow but not in ICE flow. The IXCOM SC flow can help to bring up a design on Protium hardware.

Here is the step-by-step guideline to bring up design on IXCOM SC flow:

1) Stub out all non-RTL code in the design; use *+moduleStub* to stub out verification modules
2) Use *+no_sva* and *+no_psl* to disable assertions.
3) The compile option script file should be present in the compilation flow. The first line of the file should be:

```
"precompileOption -add stubIXCOM"
```

4) **MMP_IXCOM_SC** Verilog macro is required for Protium IXCOM SC flow. User must define this macro for MMP models during compile stage. Here are the examples:

```
hdlInputFile +define+MMP_IXCOM_SC

or

vlan +define+MMP_IXCOM_SC
```

For more detailed information, please refer to "IXCOM Stub and Check Flow" section in the *Protium User Guide*.

10.2.3 IXCOM Flow

A Protium IXCOM flow is only supported on Protium X series platforms. During the Protium IXCOM X series flow, the files need to be synthesized prior to the back-end Protium IXCOM compile. An example of the command for compilation (including synthesis) and run of this model in the IXCOM flow is shown below.

```
ixcom –ptm +sv \
          +dut+dfiphy5_lpddr4 \
          ./dfiphy5_lpddr4.v \
          ./dfiphy5_pd.vp \
          ./lpddr4_pd.vp \
          <your testbench>.v \
          -top <your testbench top> \
          -incdir <mmp_release_install_path>/utils/cdn_mmp_utils/sv \
      <mmp_release_install_path>/utils/cdn_mmp_utils/sv/cdn_mmp_utils.sv \
          ……

xrun -R -64 \
    -sv_lib
<mmp_release_install_path>/utils/cdn_mmp_utils/lib/64bit/libMMP_utils.so \
    -input ptmRun.tcl \
    +ptm_lock_script=./ptm_lock.qel
```

The runtime script *ptmRun.tcl* collects the run-time commands. *xrun* executes the commands one-by-one as soon as the hardware resource is successfully locked. The Protium hardware resource (blade serial number) is defined in the script file, which must be matched to the *.et3config file used during the compile stage.

Here is an example for the lock script (*ptm_lock.qel*)

```
system -set 184190096
```

where **184190096** is the serial number of the target blade allocated.

11. Revision History

The following table shows the revision history for this document

Date	Version	Revision
May 2019	1.0	Initial version
July 2019	1.1	Update for DDR5 support
September 2019	1.2	Update valid value range for mem_data_byte_width Update for DDR5 RDIMM support
October 2019	1.3	Update LPDDR5 related information
November 2019	1.4	Update information for GDDR6 x16/x8 configuration
December 2019	1.5	Update information for LPDDR5 ECC support
February 2020	1.6	Update for forcing value difference between parameters and registers for DDR5 and DDR5 RDIMM
March 2020	1.7	Update for PHY init support for DDR4 and LPDDR4 Move DFIPHY 5.0 for DDR5 and DDR5 RDIMM model from Beta to non-Beta
March 2021	1.8	Update timing examples for DDR5 Update for MMP21.03 release

APPENDIX A: Parameters For A 1:2 Frequency Ratio System

For general information about DFI PHY clocking see section “DFI Clocks”. In a 1:2 frequency ratio system DFI PHY clock is 2x faster than DFI clock. The figure below shows the primary clocks in a 1:2 frequency ratio system.

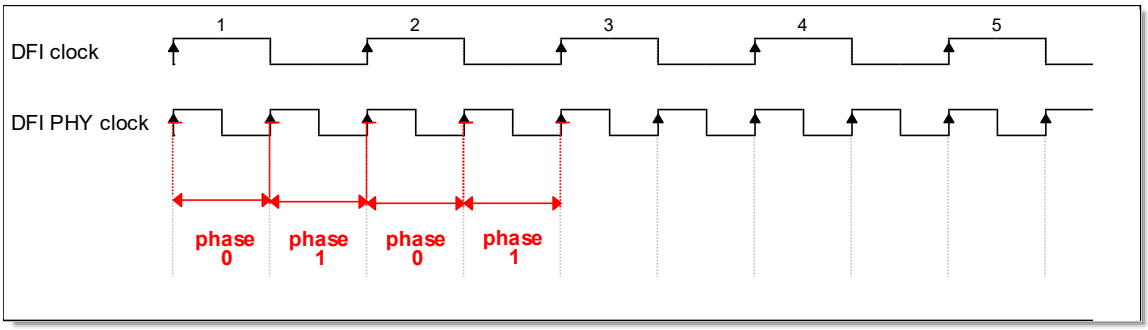


Figure 15: Clocking for 1:2 Frequency Ratio System

The following sections show example timing diagrams for calculating parameters for a 1:2 frequency ratio system with an MMP DFI PHY + DDR5.

In a 1:2 MC to PHY frequency ratio system, the valid dfi control information is clock phase dependent.

DFI 5.0 Specification (April 27, 2018) defines commands for a frequency ratio system in a vectored format where the control signal interface signals (write data interface and read data enable signals) are suffixed with “_pN” where N is the phase number. So, one sees, for a 1:2 frequency ratio system, instead of a single **dfi_cs** signal, 2 signals: **dfi_cs_p0** and **dfi_cs_p1.** The **dfi_cs_p0** information is the only valid chip select information during phase0 (dfi clock signal is high) of the **dfi clock**; likewise, the **dfi_cs_p1** information is the only valid chip select information during phase1 (dfi clock signal is low) of the **dfi clock**. In order to develop a complete picture of the signal behavior one must merge, or account for, the behavior in both phases. The following diagrams attempt to show when and how the phase information is accounted for.

1. Parameter t_ctrl_delay

This parameter defines the number of clock cycles delay through the DFI PHY for the control path signals in units of *DFI clock (DFI_CLK)* cycles. This is in Section 4.3 of the DFI (DDR PHY Interface) 5.0 specification (April 27, 2018).

The following waveform shows t_ctrl_delay value of 3 for a 1:2 frequency ratio system. As the delay is measured in DFI clock (DFI_CLK) cycles and not *DFI PHY (PHY_CLK)* clocks it is possible that t_ctrl_delay is a fraction of a *DFI clock (DFI_CLK)*. In this case, round the value up.

Note that because t_ctrl_delay is measured in terms of DFI_CLK clock signal it is therefore unaffected by the phases of a 1:2 or 1:4 system.

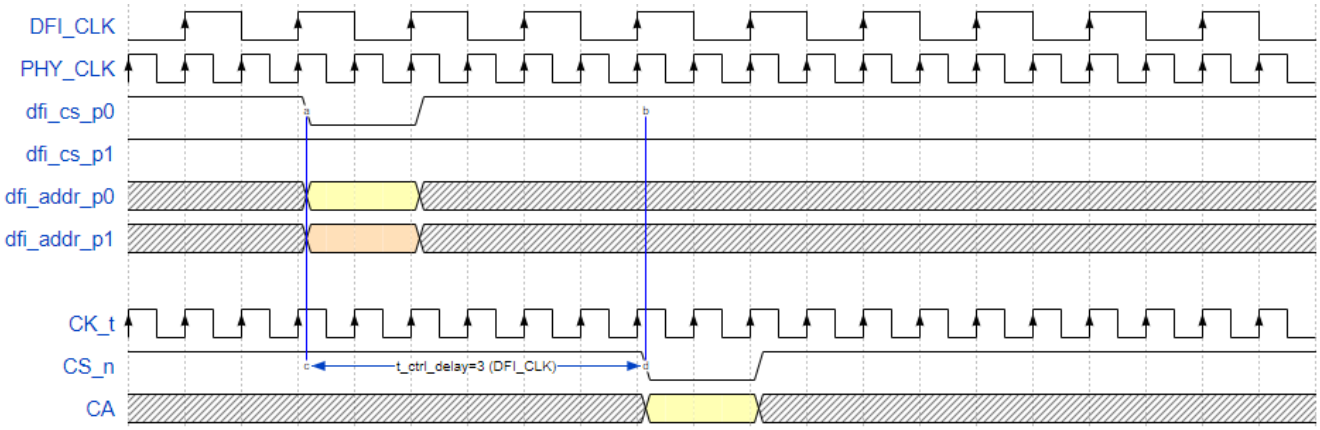


Figure 16: Parameter t_ctrl_delay Waveform in 1:2 System

2. Parameter t_phy_wrdata

This parameter defines the delay from the dfi_wrdata_en being asserted to the first valid data values on the dfi_wrdata bus in units of *DFI PHY (PHY_CLK)* clocks. This is described in Section 4.7 of the DFI (DDR PHY Interface) 5.0 specification (April 27, 2018). The following waveform (Figure 17: Parameter t_phy_wrdata Waveform in 1:2 System (dfi_wrdata_en_p# aligned)) shows a t_phy_wrdata value of 4 for a 1:2 frequency ratio system.

Note that for LPDDR5, the dfi_freq_ratio is the clock ratio between DFI_CLK and DAT_CLK (more details in 4.3 DFI clocks), so the parameter t_phy_wrdata for LPDDR5 is in units of DAT_CLK clocks.

Note that when reviewing the simulation waveforms it is important to correctly determine when the first valid data occurs. It is possible that the controller can send valid data with the value of zero.

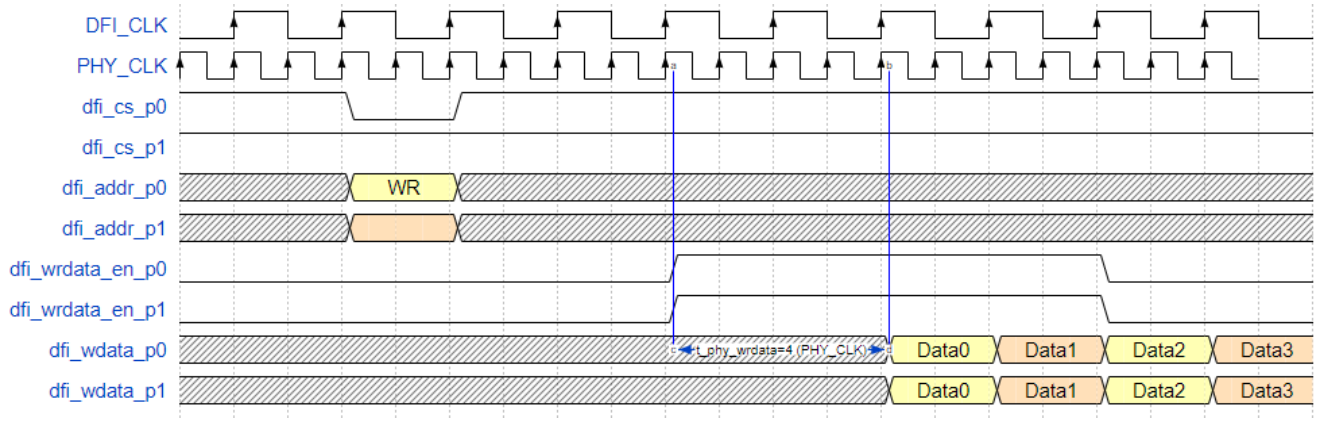


Figure 17: Parameter t_phy_wrdata Waveform in 1:2 System (dfi_wrdata_en_p# aligned)

This below example shows a write data enable starting on Phase 1 and therefore the measurement is made from the phase 1 point within the cycle. Similarly, the write data is first active on Phase 1 and so the measurement is made from the phase 1 point within the cycle.

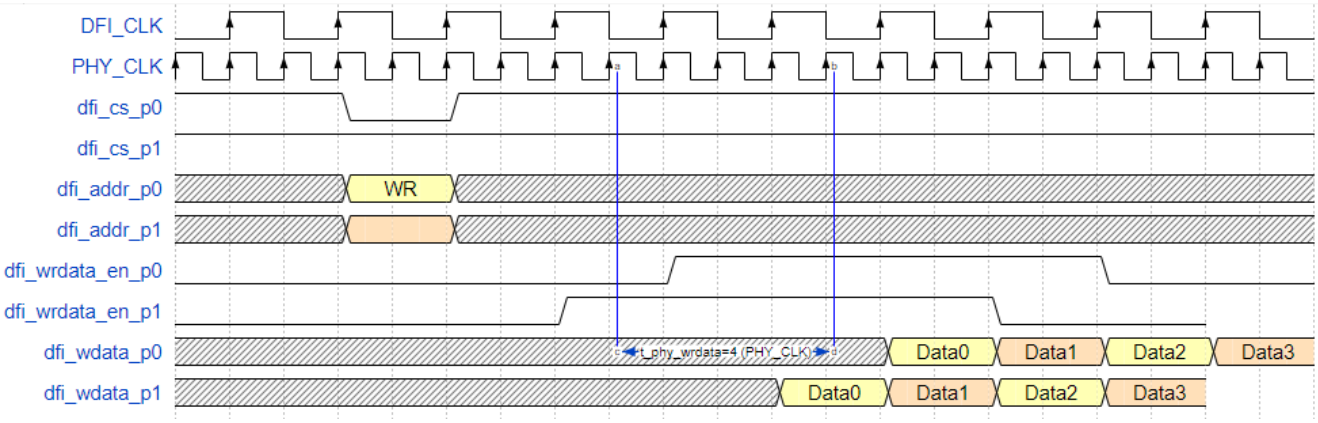


Figure 18: Parameter t_phy_wrdata Waveform in 1:2 System (dfi_wrdata_en_p# not aligned)

3. Parameter t_wrlat_adj

This parameter is used to adjust the DFI PHY parameter t_phy_wrlat. The DFI PHY parameter t_phy_wrlat defines the number of *DFI PHY (PHY_CLK)* clock cycles delay between the write command on the DFI interface and the assertion of the dfi_wrdata_en signal. This is described in Section 4.7 of the DFI (DDR PHY Interface) 5.0 specification (April 27, 2018).

Note that for LPDDR5, the dfi_freq_ratio is the clock ratio between DFI_CLK and DAT_CLK (more details in 4.3 DFI clocks), so the parameter t_phy_wrlat for LPDDR5 is in units of DAT_CLK clocks.

The Cadence Palladium/Protium DFI PHY model has an internal write latency itself, which varies based on protocol. This may need to be adjusted to match the latency of the users DFI PHY. The parameter t_wrlat_adj is used for this purpose. The internal write latency of the DFI PHY model is calculated as follows:

```
DDR/Mobile_DDR:     t_pd_wrlat = 1 - t_phy_wrdata
DDR2/3/4:           t_pd_wrlat = WL - t_phy_wrdata
DDR4 RDIMM:         t_pd_wrlat = WL - t_phy_wrdata
DDR5:               t_pd_wrlat = WL + 1 - t_phy_wrdata
DDR5 RDIMM:         t_pd_wrlat = WL + 5 - t_phy_wrdata
LPDDR2/3:           t_pd_wrlat = WL + tDQSS - t_phy_wrdata
LPDDR4:             t_pd_wrlat = WL + tDQSS + tDQSS_half - t_phy_wrdata
LPDDR5:             t_pd_wrlat = (WL+1)*ratio-1 - t_phy_wrdata
                    (ratio=4 for CKR(WCK:CK)=4:1; ratio=2 for CKR(WCK:CK)=2:1)
GDDR6:              t_pd_wrlat = WL - t_phy_wrdata
```

Note: WL above refers to the DDR or LPDDR memory Write Latency.

- Write Latency for DDR and Mobile_DDR (LPDDR) is just the CL value (CAS Latency).
- Write Latency for DDR2 is defined as RL – 1 (Read Latency -1) where read latency is the sum of the CL (CAS Latency) and AL (Additive Latency) i.e. RL=AL+CL.
- Write Latency for DDR3 is the sum of CWL (CAS Write Latency) and AL (Additive Latency) i.e. WL=AL + CWL.
- Write Latency for DDR4 is the sum of CL (CAS Latency) and AL (Additive Latency) and Parity Latency i.e. WL=AL + CWL + PL.
- For LPDDR2/3/4/5 and GDDR6 and DDR5 WL is directly defined by mode register content. The default value of tDQSS is 1 for LPDDR2/3; by default, the tDQSS=1 and tDQSS_half=0 for LPDDR4.
- For DDR4 RDIMM, if the WL value from mode registers setting is not equal to the real WL from write operation, use the WL value from write operation for this formula calculation.

These latency settings are controlled by Mode Registers in the memory device and are set by the memory controller during initialization.

Please refer to JEDEC specification for more information.

The parameter t_wrlat_adj is the difference between the actual PHY t_wrlat parameter and the DFI PHY model t_wrlat parameter.

```
t_wrlat_adj = t_pd_wrlat - t_phy_wrlat
```

In the following waveform (Figure 19: Parameter t_wrlat_adj Waveform in 1:2 System (dfi_wrdata_en_p# aligned)), showing DDR5, the WL is 24, the t_phy_wrdata is 2. From the waveform the t_phy_wrlat is 4. Therefore the t_wrlat_adj is 19.

This example:

WL = 24

t_phy_wrdata = 2

(For DDR5) t_pd_wrlat = WL + 1 – t_phy_wrdata = 25 – 2 = 23

t_wrlat_adj = t_pd_wrlat – t_phy_wrlat = 23 – 4 = 19

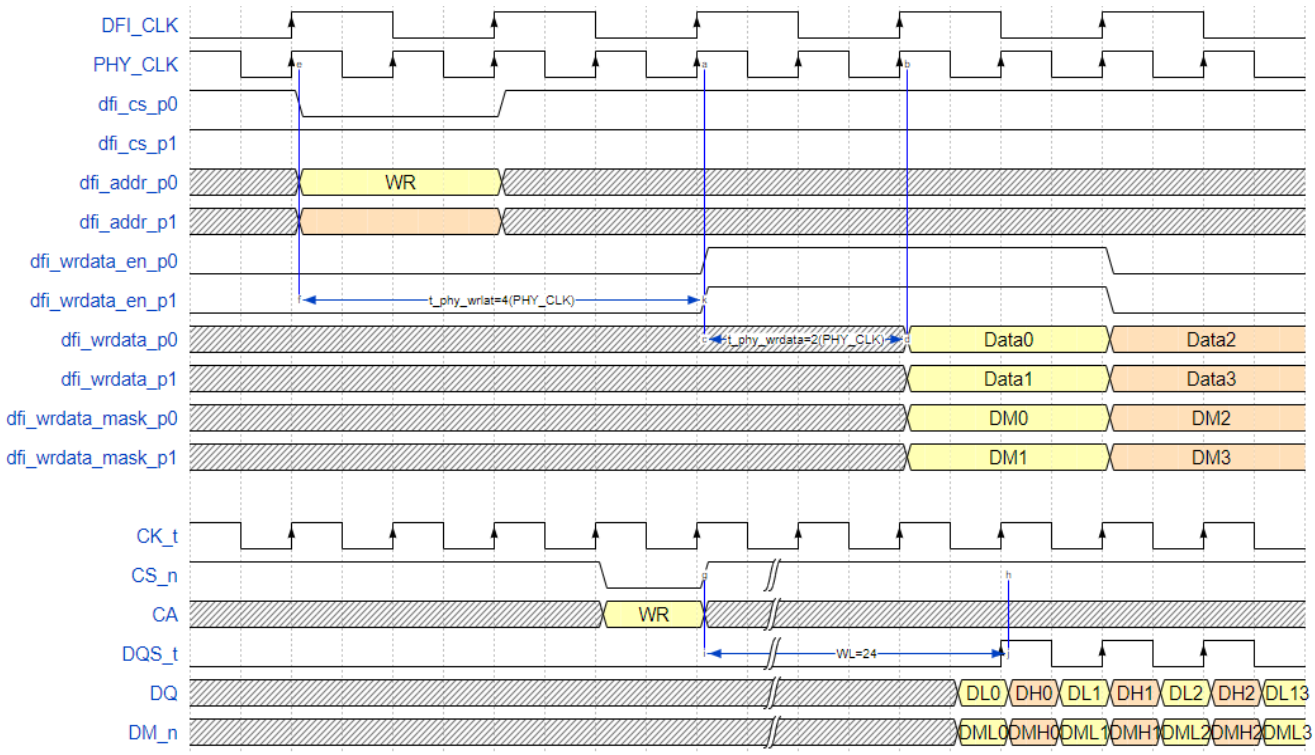


Figure 19: Parameter t_wrlat_adj Waveform in 1:2 System (dfi_wrdata_en_p# aligned)

This example (write command send on phase1):
WL = 24
t_phy_wrdata = 2
(For DDR5) t_pd_wrlat = WL + 1 – t_phy_wrdata = 25 – 2 = 23
t_wrlat_adj = t_pd_wrlat – t_phy_wrlat = 23 – 2 = 21

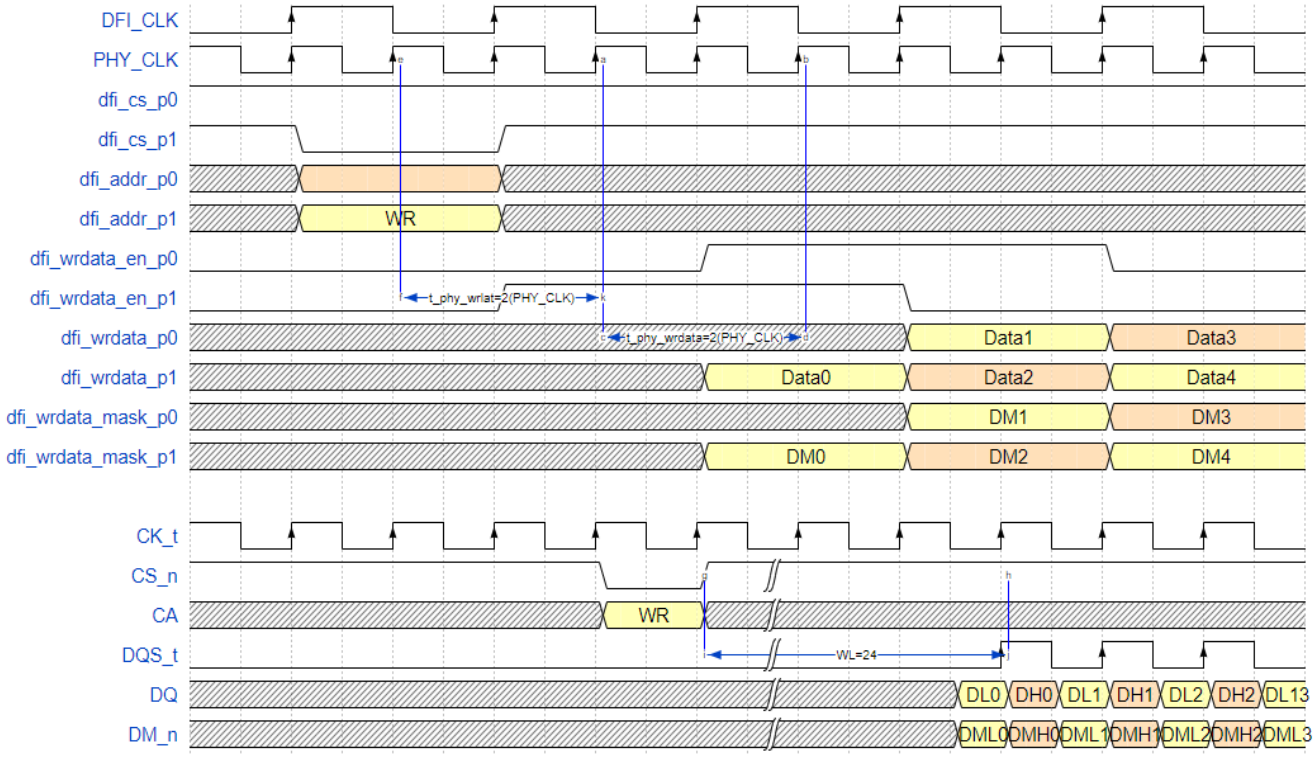


Figure 20: Parameter t_wrlat_adj Waveform in 1:2 System (dfi_wrdata_en_p# not aligned)

Below is the example for the parameter t_phy_wrlat and t_phy_wrdata calculation with LPDDR4 DFI PHY for a 1:2 frequency ratio system. It needs 2 DFI_CLK cycle for LPDDR4 WRITE command, so the measure start point is different from other SDRAMs.

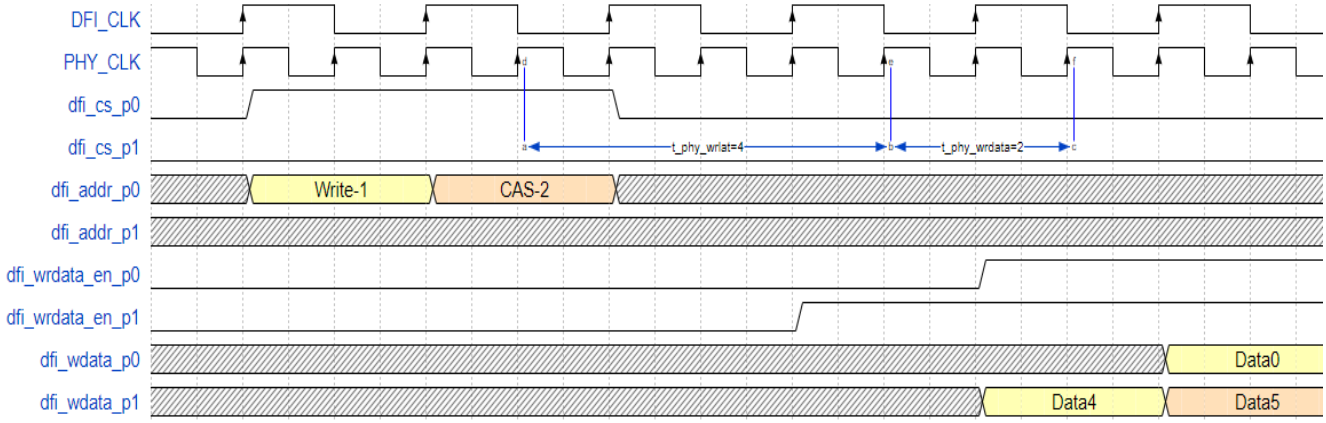


Figure 21: Parameter t_phy_wrlat and t_phy_wrdata Waveform in 1:2 System for LPDDR4

4. Parameter t_rddata_en_adj

This parameter is used to adjust the DFI PHY parameter t_rddata_en. The DFI PHY parameter t_rddata_en defines the number of *DFI PHY (PHY_CLK)* clock cycles delay between the read command on the DFI interface and the assertion of the dfi_rddata_en signal. This is described in Section 4.8 of the DFI (DDR PHY Interface) 5.0 specification (April 27, 2018).

Note that for LPDDR5, the dfi_freq_ratio is the clock ratio between DFI_CLK and DAT_CLK (more details in 4.3 DFI clocks), so the parameter t_rddata_en for LPDDR5 is in units of DAT_CLK clocks.

The Cadence Palladium/Protium DFI PHY model has an internal read latency itself, which varies based on protocol. This may need to be adjusted to match the latency of the users DFI PHY. The parameter t_rddata_en_adj is used for this purpose. The internal read latency of the DFI PHY model is calculated as follows:

```
DDR/Mobile_DDR:     t_pd_rddata_en = CL
DDR2/3/4:           t_pd_rddata_en = RL
DDR4 RDIMM:         t_pd_rddata_en = RL
DDR5:               t_pd_rddata_en = RL + 1
DDR5 RDIMM:         t_pd_rddata_en = RL + 5
LPDDR2/3:           t_pd_rddata_en = RL + 1 + t_DQSCK
LPDDR4:             t_pd_rddata_en = RL + tDQSCK + tDQSCK_half
LPDDR5:             t_pd_rddata_en = (RL+1)*ratio-1
                    (ratio=4 for CKR(WCK:CK)=4:1; ratio=2 for CKR(WCK:CK)=2:1)
GDDR6:              t_pd_rddata_en = RL + 1
                    t_pd_rddata_edc_en = RL + CRCRL + 1
```

Note: The default value of t_DQSCK in the LPDDR2/3 model is 1.
By default, tDQSCK=1 and tDQSCK_half=1 in LPDDR4 model.

Note: CL above refers to the DDR memory CAS Latency. Latency settings are controlled by Mode Registers in the memory device and are set by the memory controller during initialization. Please refer to JEDEC specification for more information.

Note: RL above refers to the DDR or LPDDR memory Read Latency.

- Read Latency for DDR and Mobile_DDR (LPDDR) is just the CL value (CAS Latency).
- Read Latency for DDR2/3 is the sum of CL (CAS Latency) and AL (Additive Latency) i.e. RL=AL+CL.
- Read Latency for DDR4 is the sum of CL (CAS Latency) and AL (Additive Latency) and Parity Latency i.e. RL=AL+CL+PL.
- For LPDDR2/3/4/5 and GDDR6 and DDR5 RL is directly defined.
- For DDR4 RDIMM, if the RL value from mode registers setting is not equal to the real RL from read operation, use the RL value from read operation for this formula calculation.

The parameter t_rddata_en_adj is the difference between the actual PHY t_rddata_en parameter (t_rddata_en) and the DFI PHY model t_pd_rddata_en parameter.

```
t_rddata_en_adj = t_pd_rddata_en - t_rddata_en
```

In the following waveform (Figure 22: Parameter t_rddata_en_adj Waveform in 1:2 System), showing DDR5, the RL is 26. From the waveform the t_rddata_en is 4. Therefore the t_rddata_en_adj is 23.

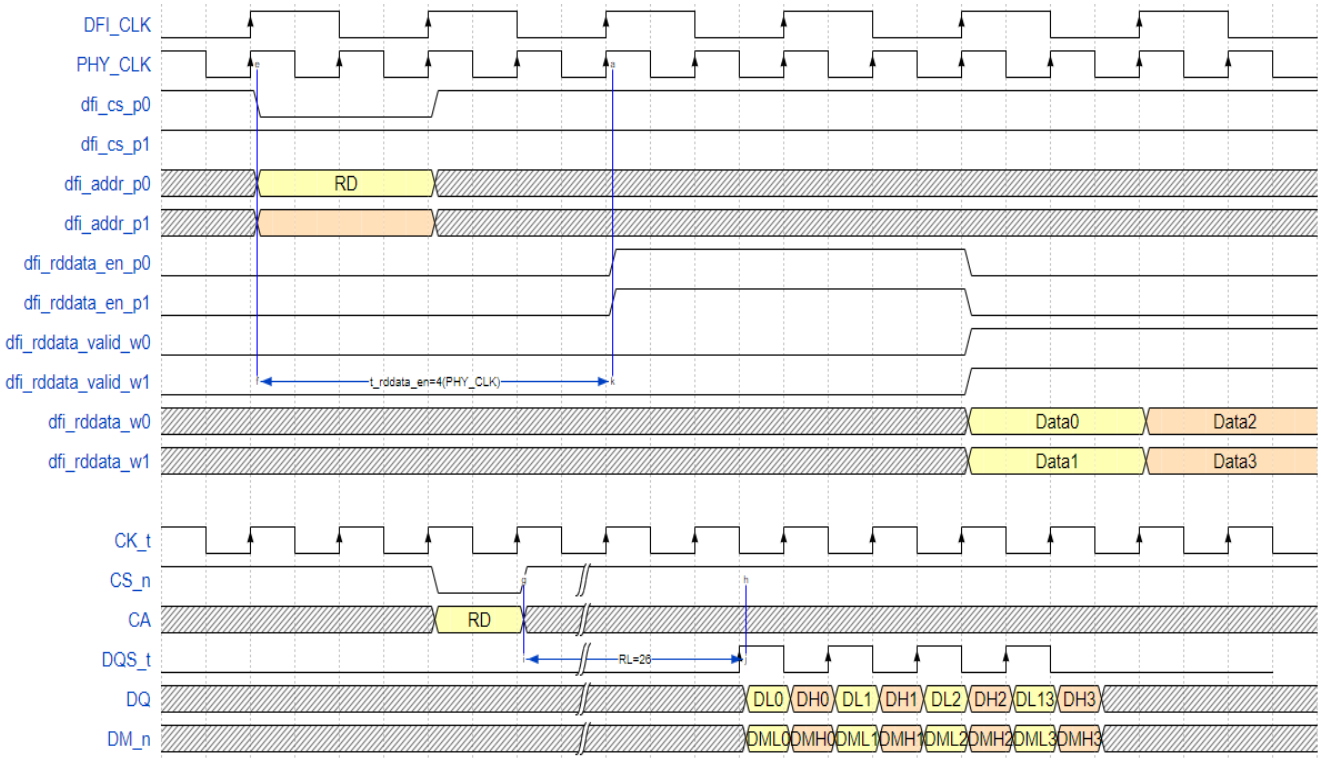


Figure 22: Parameter t_rddata_en_adj Waveform in 1:2 System

This example:
RL = 26
t_rddata_en = 4
(For DDR5) t_pd_rddata_en = RL+1 = 27
t_rddata_en_adj = t_pd_rddata_en – t_rddata_en = 27 – 4 = 23

Below is the example for the parameter t_rddata_en calculation with LPDDR4 DFI PHY for a 1:2 frequency ratio system. It needs 2 DFI_CLK cycle for LPDDR4 READ command, so the measure start point is different from other SDRAMs.

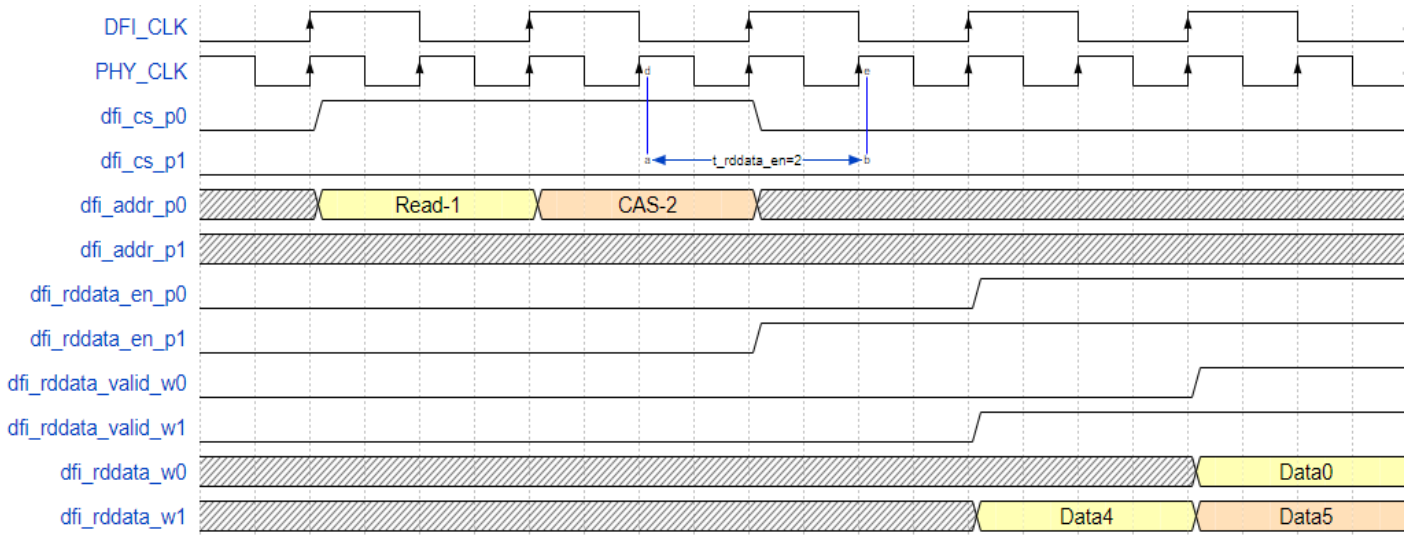


Figure 23: Parameter t_rddata_en Waveform in 1:2 System for LPDDR4

5. Other Interface Parameters

The remaining parameters (mem_data_bits, dfi_addr_bits, dfi_bank_addr_width, dfi_bg_addr_width, dfi_cid_width, dfi_cs_width, mem_ca_bit) are independent of the frequency ratio system.

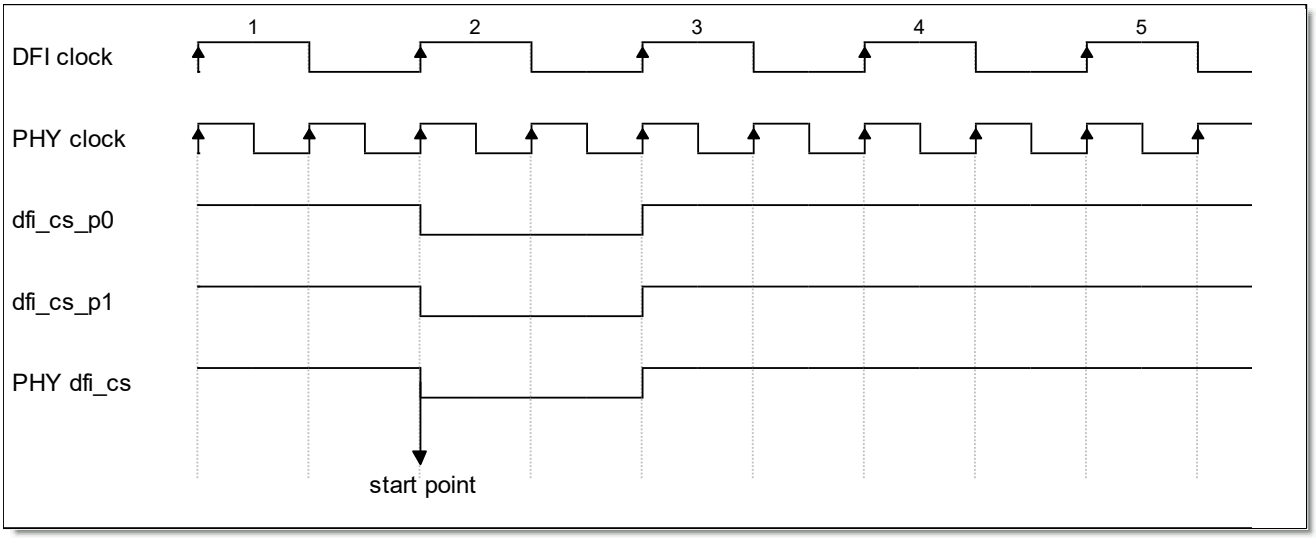
6. Example of finding start/end point of measuring

In 1:2 frequency ratio system, it may be confusing to determine the start/end point when users measure timing parameters especially when signals on p0 and p1 are NOT aligned.

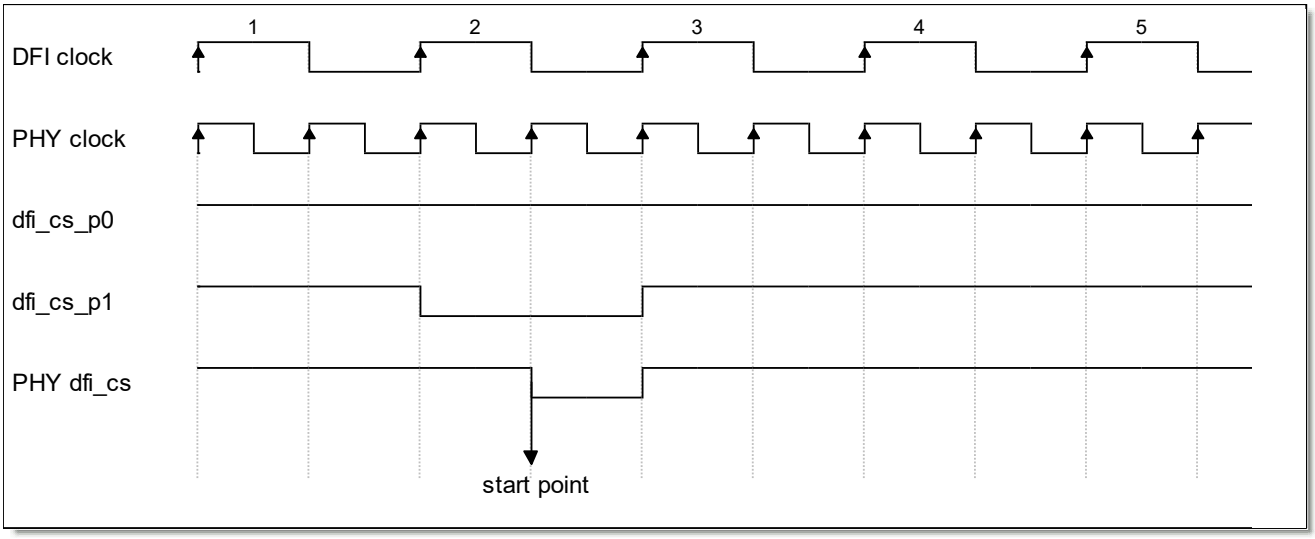
The following examples show how to determine the start/end point. With these examples, users shall know: the start of measuring does not depend on individual p0 or p1 signal, it shall depend on the “merged” signal of p0 and p1 according to phases.

In 1:2 frequency ratio system, all the signals used to measure timing parameters need to be such “merged” signals.

Example 1.
dfi_cs_p0 and **dfi_cs_p1** are aligned to each other.



Example 2.
dfi_cs_p0 and **dfi_cs_p1** are NOT aligned to each other.



Example 3.
dfi_wrdata_en_p0 and **dfi_wrdata_en_p1** are NOT aligned to each other.