Verification Continuum™

ZeBu® DFI Interface for LPDDR4 User Guide

Version Q-2021.06, June 2021

SYNOPSYS®

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at http://www.synopsys.com/company/legal/trademarks-brands.html.
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Contents

List of Tables

List of Figures

About This Manual

This manual describes how to use the ZeBu DFI interface for easier integration between Memory Controller (MC) and PHY.

Related Documentation

For more relevant information about each memory protocol associated with the zDFI, please refer to the related JEDEC specifications.

For more relevant information about the DFI protocol, please refer to the latest MIPI DFI specification.

Synopsys Statement on Inclusivity and Diversity

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

1 Introduction

DDR PHY Interface is an industry-standard interface protocol for easier integration between Memory Controller (MC) and PHY. DFI defines the signals and timing parameters required to transfer control information and data. These parameters are constrained by the MC and/or the PHY. Current version is DFI 4.0. It supports DDR3, DDR4, DDR5, GDDR6, HBM2/2E, LPDDR1, LPDDR2, LPDDR3, LPDDR4, LPDDR5.

This section explains the following topics:

- *Overview*
- *Features*
- *Limitations*

1.1 Overview

zDFI is a PHY model, compliant with the DFI standard, for ZeBu DDR Memory Models. It is delivered as an encrypted SystemVerilog file, including a SystemVerilog module called zdfi. One package is delivered for all DDR families.

zDFI supports integration of a custom DFI-compliant Memory Controller (DUT). You need to replace your current PHY with the zdfi module assigning proper values to the signal width and timing parameters.

The following figure illustrates the DFI Interface:

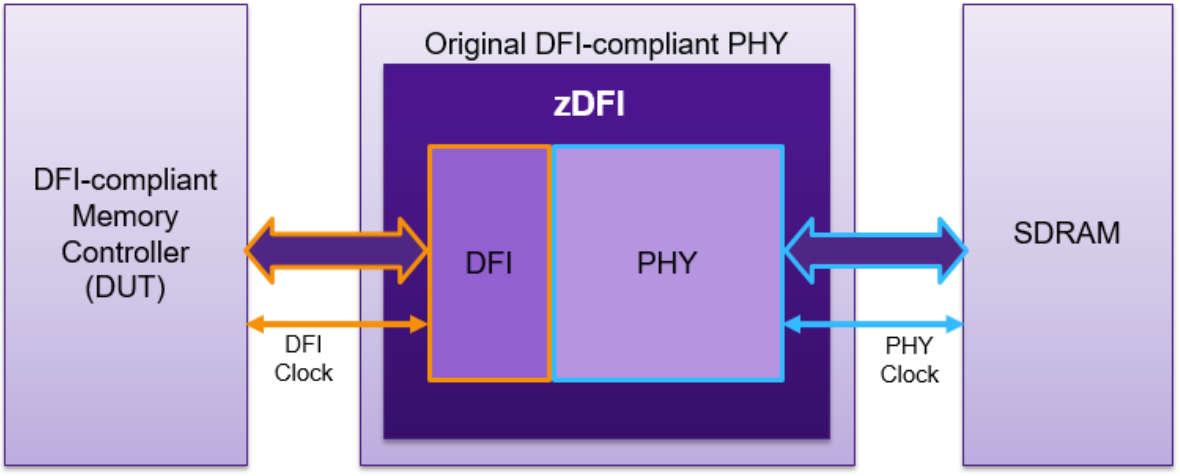

FIGURE 1. DFI Interface

1.2 Features

The following are the features supported by DFI:

TABLE 1 Supported Features

SDRAM Family	Comment
Read transactions	Transmitted to the DDR model.
Write transactions	Transmitted to the DDR model.
Configuration operations (MRW)	Transmitted to the DDR model.
Configuration operations detection (MRW)	The zDFI detects these accesses and extracts useful information (Burst length, Write latency, read latency).
On-the-fly Burst Length	
DFI Update Interface	
DFI optional protocols ● Cyclic redundancy check (CRC) ● System frequency change ● DFI Low Power interface ● Error interface	
Status Interface	Only dfi_init_start/complete
Data Bus Inversion (DBI/DMI)	
1:1 clock ratio	
1:2 clock ratio	Suffix _HDR, 2 phases
1:4 clock ratio	Suffix _QDR, 4 phases
Single channel support	
Multiple channel support	

1.3 Limitations

The following are the limitations of the DFI interface:

- Does not support clock ratios 1:1.
- Initialize the following parameters at compilation time:
  - MEM_BURST_LENGTH
  - MEM_WRITE_LATENCY
  - MEM_READ_LATENCY

  Set the value of these parameters to the required default values, compliant with the default or zebuMR-programmed values in the associated memory. At run time, each time that a Mode Register programming will update the values in the memory, the zDFI will be updated accordingly.

2 Integration Methodology

This section explains the following topics:

- *Integration Process*
- *Connecting Clocks*
- *Design Topologies*

2.1 Integration Process

To integrate a Memory Controller (MC) and PHY, perform the following steps:

1. Gather information for parameterization of the zDFI model (bus sizes + timing parameters).
   a. If the timing parameters are not available, simulate your design using the zDFI model and the memory simulation model.
   b. Use the default timing parameters in the first stage of the integration process.
   c. Extract the timing parameters from the waveforms. See section 3.1 for a methodology to compute DFI_T_PHY_WRDATA and DFI_T_PHY_WRLAT.
   d. Use the memory probe vector to extract MEM_WRITE_LATENCY and MEM_READ_LATENCY. See ZeBu LPDDR4 Memory Models User Manual for more information.
   e. Once the value of DFI_T_PHY_WRDATA, DFI_T_PHY_WRLAT and MEM_WRITE_LATENCY is computed, compute DFI_T_CTRL_DELAY as detailed in section 3.1.
   f. Modify your simulation with the new values of the timing parameters, relaunch simulation and check that write and read access are correct.
2. Emulate your design using the zDFI model and the VS DDR model with the same parameter values as above.

2.2 Connecting Clocks

zDFI provides two clock inputs, dfi_clk and mem_clk.

- dfi_clk is generated by the MC (DFI controller).
- The mem_clk is user-generated.

The clock domain for mem_clk should be the same as that for dfi_clk (derived from the same primary clock). The two clocks must have their rising edges aligned.

The following are the clock frequency requirements:

- mem_clk 2x faster than dfi_clk in 1:2 / HDR / 2-phase zDFI
- mem_clk 4x faster than dfi_clk in 1:4 / QDR / 4-phase zDFI

The **mem_clk_p** output is the positive side of the differential clock. It must be connected to the positive clock input of the memory. On the memory model, the input is named CK_t_A/B.

The **mem_clk_n** output is the negative side of the differential clock. It must be connected to the negative clock input of the memory. On the memory model, the input is named CK_c_A/B.

The following is the clock structure inside the zDFI:

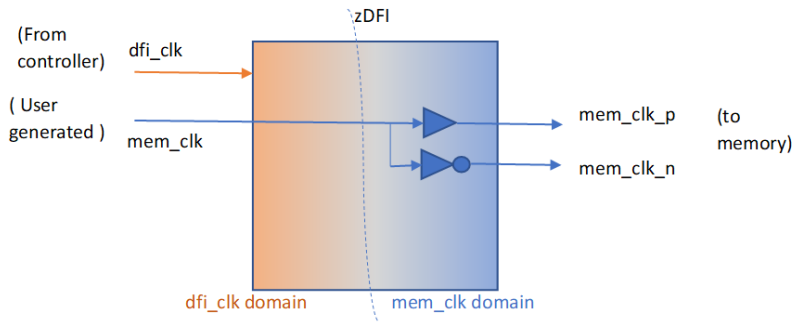

FIGURE 2. Clock Structure

2.3 Design Topologies

The following design topologies are available in the zDFI Interface for LPDDR4:

- *Independent Channels / Independent Memories*
- *Parallel Channels*

2.3.1 Independent Channels / Independent Memories

In this topology, two channels or two memories do not share the same address buses. In this case, instantiate one DFI per channel.

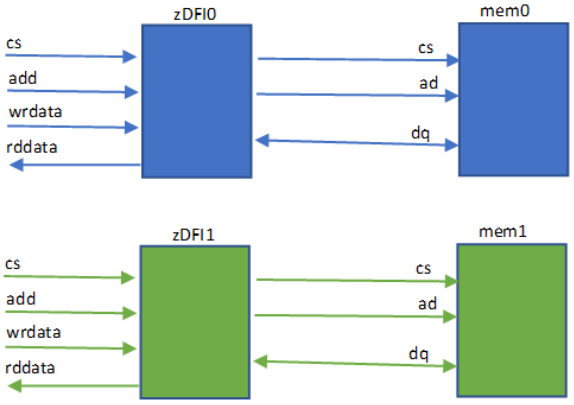

FIGURE 3. 2 Independent Channels / 2 Independent Memories

2.3.2 Parallel Channels

In this topology, two channels or two memories share the same address buses. They also share the same clock enable. The mem_dq bus of the zDFI is split in 2 to/from the 2 memories (mem_dq = {mem1_dq,mem0_dq}). In this case, instantiate one DFI with double data size.

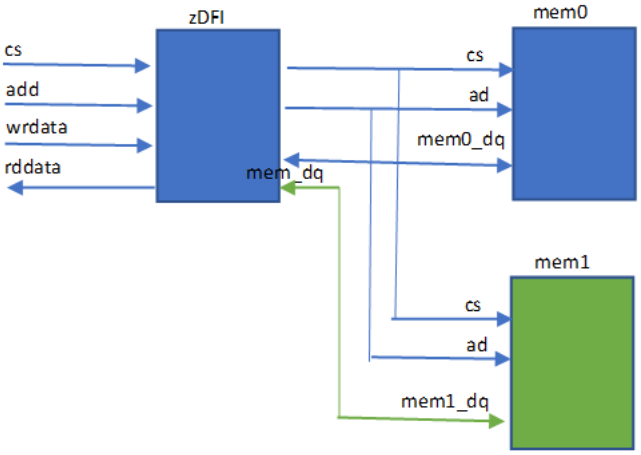

FIGURE 4. Instantiate One DFI With Double Data Size

However, in this topology, the two channels or two memories share the same chip select (mem_cs = {mem1_cs,mem0_cs}) but they do not share the same clock enable (mem_dq = mem0_dq = mem1_dq). In this case, instantiate one DFI with double chip select size.

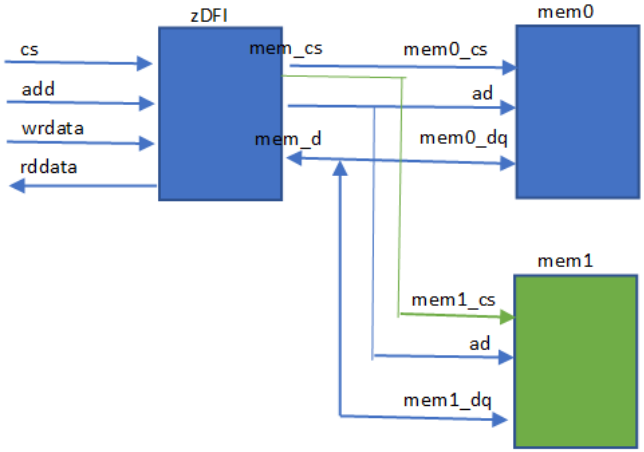

FIGURE 5. Instantiate one DFI with Double Chip Select Size

3 zDFI Parameters

This section explains the following topics:

- *zDFI Timing Parameters*
- *zDFI Vector Width Parameters*

3.1 zDFI Timing Parameters

The following table describes the zDFI timing parameters

TABLE 2 zDFI Timing Parameters

Parameter	Description
DFI_T_PHY_WRLAT	Delay between DFI write command and DFI write_en
DFI_T_PHY_WRDATA	Delay between DFI write_en and DFI wrdata
DFI_T_CTRL_DELAY	Delay between command on the DFI interface and command reaching the memory. Minimum value is 3.

All delays are specified as a number of PHY_CLK cycles. Also, the frequency of PHY_CLK is the same frequency as the frequency of mem_clk.

This section explains the following topics:

- *DFI Timing Parameters for 1:2 Clock Ratio*
- *DFI Timing Parameters for 1:4 Clock Ratio*
- *zDFI Memory Timing Parameters*
- *Timing Constraints*

3.1.1 DFI Timing Parameters for 1:2 Clock Ratio

Per the LPDDR4 specification, a write command is a 4-cycle sequence (2 cycles for WRITE-1, 2 cycles for CAS-2). The write latency is computed from the CAS-2 command. As per the LPDDR4 command truth table, a CAS-2 is recognized by values 0x32 or 0x12 on the idfi_address bus. As detailed below, this methodology is slightly different depending on which lane the command is sent on.

The following figures illustrates how to compute DFI_T_PHY_WRLAT and DFI_T_PHY_WRDATA from a reference simulation waveform.

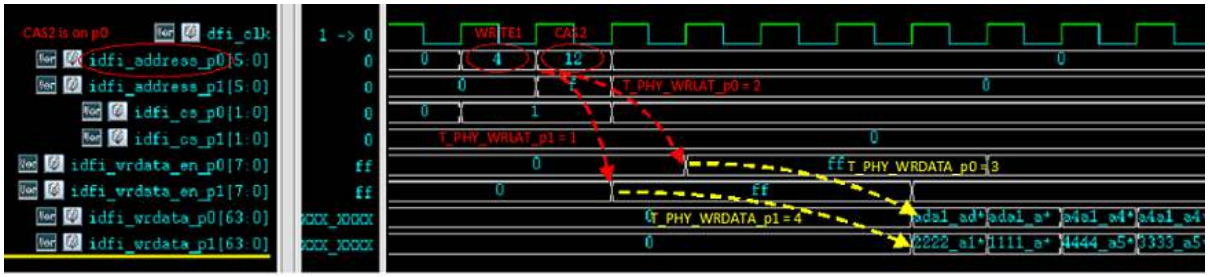

FIGURE 6. Command is on P0

In the above example:

```
DFI_T_PHY_WRLAT = T_PHY_WRLAT_p0+T_PHY_WRLAT_p1 = 2+1 = 3
DFI_T_PHY_WRDATA = T_PHY_WRDATA_p0+T_PHY_WRDATA_p1 = 3+4 = 7
```

For the above example, calculate the following values:

- T_PHY_WRLAT_p0: number of dfi_clk cycles between CAS-2 command and idfi_wrdata_en_p0
- T_PHY_WRLAT_p1: number of dfi_clk cycles between CAS-2 command and idfi_wrdata_en_p1
- T_PHY_WRDATA_p0: number of dfi_clk cycles between idfi_wrdata_en_p0 and idfi_wrdata_p0
- T_PHY_WRDATA_p1: number of dfi_clk cycles between idfi_wrdata_en_p1 and idfi_wrdata_p1

Below is an example of timing computing with the CAS-2 command sent on phase 1. The same method applies, except that the final result on DFI_T_PHY_WRLAT is decreased by 1.

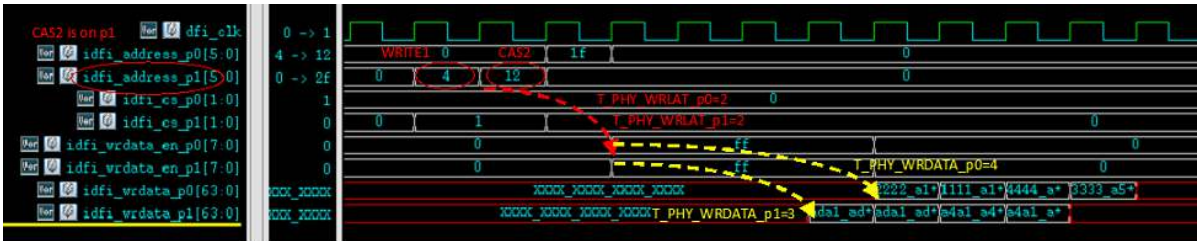

FIGURE 7. Command is on P1

In the above figure, since the command is on p1, substract 1 to the T_PHY_WRLAT parameter computation, as shown below:

```
DFI_T_PHY_WRLAT = T_PHY_WRLAT_p0+T_PHY_WRLAT_p1-1 = 2+2-1 = 3
DFI_T_PHY_WRDATA = T_PHY_WRDATA_p0+T_PHY_WRDATA_p1 = 4+3 = 7
```

3.1.2 DFI Timing Parameters for 1:4 Clock Ratio

Per the LPDDR4 specification, a write command is a 4-cycle sequence (2 cycles for WRITE-1, 2 cycles for CAS-2). The write latency is computed from the CAS-2 command. Per the LPDDR4 command truth table, a CAS-2 is recognized by values 0x32 or 0x12 on the idfi_address bus.

As detailed below, the methodology is slightly different, depending on which lane the command is sent.

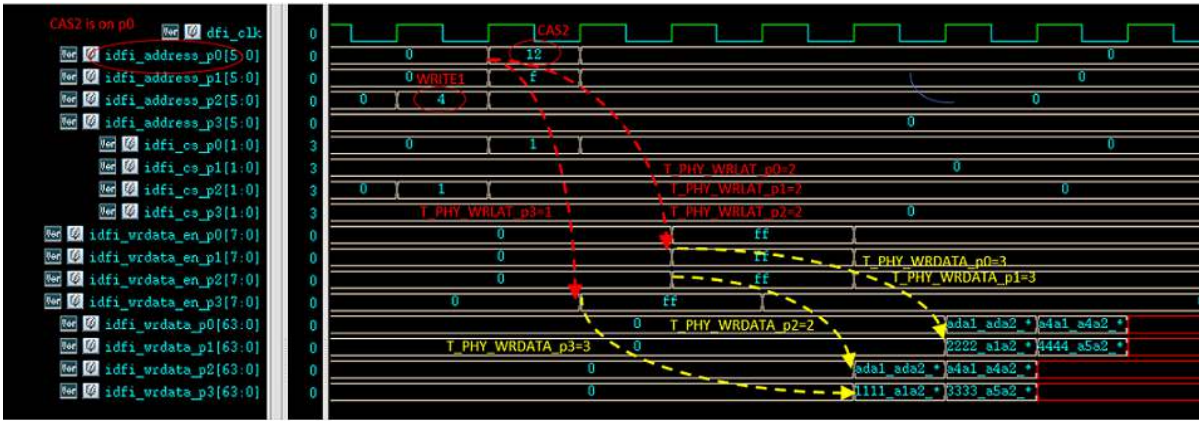

FIGURE 8. Command is on P0

In the above example:

```
DFI_T_PHY_WRLAT =
T_PHY_WRLAT_p0+T_PHY_WRLAT_p1+T_PHY_WRLAT_p2+T_PHY_WRLAT_p3 =
2+2+2+1 = 7

DFI_T_PHY_WRDATA =
T_PHY_WRDATA_p0+T_PHY_WRDATA_p1+T_PHY_WRDATA_p2+T_PHY_WRDATA_p3 =
3+3+2+3 = 11
```

For the above example, calculate the following values:

- T_PHY_WRLAT_p0: number of dfi_clk cycles between CAS-2 command and idfi_wrdata_en_p0
- T_PHY_WRLAT_p1: number of dfi_clk cycles between CAS-2 command and idfi_wrdata_en_p1)
- T_PHY_WRLAT_p2: number of dfi_clk cycles between CAS-2 command and idfi_wrdata_en_p2
- T_PHY_WRLAT_p3: number of dfi_clk cycles between CAS-2 command and idfi_wrdata_en_p3
- T_PHY_WRDATA_p0: number of dfi_clk cycles between idfi_wrdata_en_p0 and idfi_wrdata_p0

- T_PHY_WRDATA_p1: number of dfi_clk cycles between idfi_wrdata_en_p1 and idfi_wrdata_p1
- T_PHY_WRDATA_p2: number of dfi_clk cycles between idfi_wrdata_en_p2 and idfi_wrdata_p2)
- T_PHY_WRDATA_p3: number of dfi_clk cycles between idfi_wrdata_en_p0 and idfi_wrdata_p0)

Below is an example of timing computing with the CAS-2 command sent on phase 1. The same method applies, except that the final result on DFI_T_PHY_WRLAT is decreased by 1.

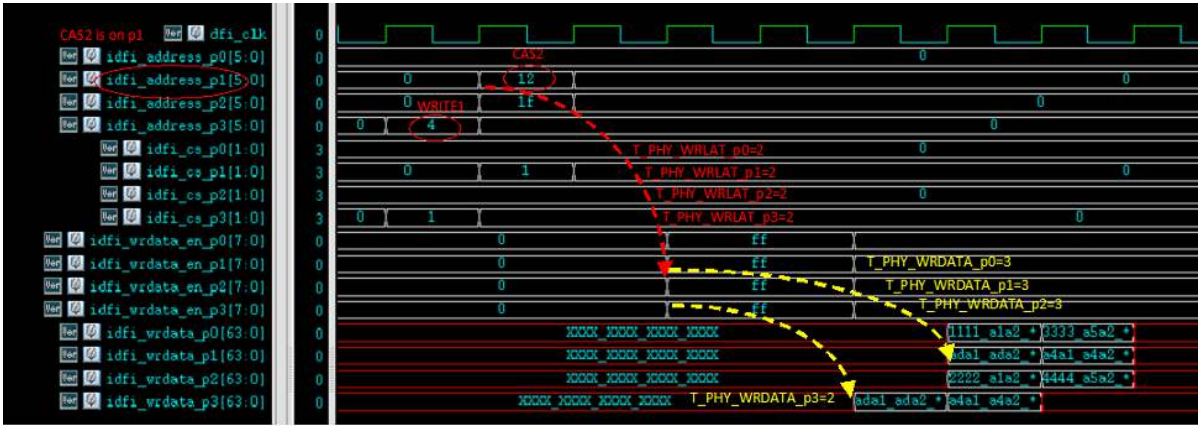

FIGURE 9. Command is on P1

In the above figure, since the command is on p1, substract 2 to the T_PHY_WRLAT parameter computation, as shown below:

```
DFI_T_PHY_WRLAT =
T_PHY_WRLAT_p0+T_PHY_WRLAT_p1+T_PHY_WRLAT_p2+T_PHY_WRLAT_p3-1 =
2+2+2+2-1= 7
DFI_T_PHY_WRDATA =
T_PHY_WRDATA_p0+T_PHY_WRDATA_p1+T_PHY_WRDATA_p2+T_PHY_WRDATA_p3 =
3+3+3+2 = 11
```

Below is an example of timing computing with the CAS-2 command sent on phase 2. The same method applies, except that the final result on DFI_T_PHY_WRLAT is decreased by 2.

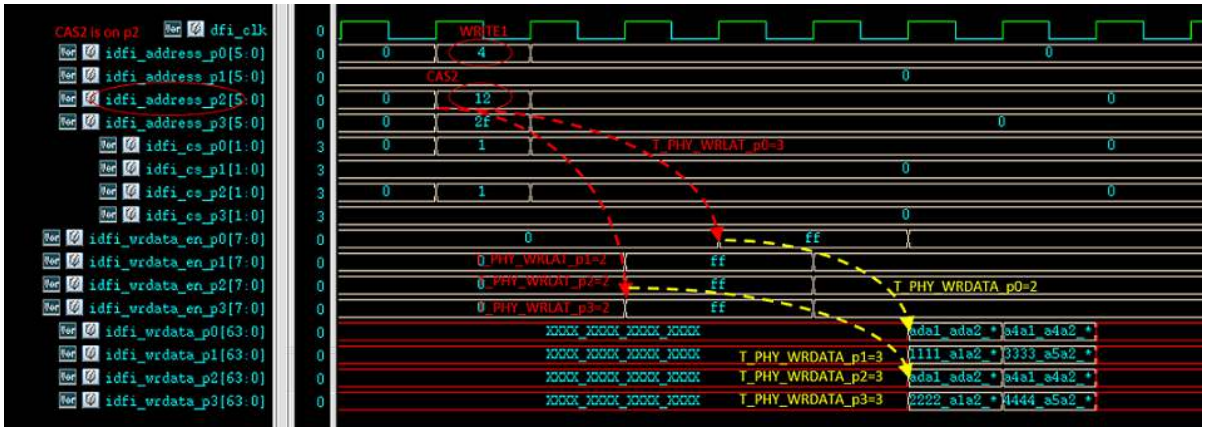

FIGURE 10. Command is on P2

In the above figure, since the command is on p2, substract 2 to the T_PHY_WRLAT parameter computation, as shown below:

```
DFI_T_PHY_WRLAT =
T_PHY_WRLAT_p0+T_PHY_WRLAT_p1+T_PHY_WRLAT_p2+T_PHY_WRLAT_p3-2 =
3+2+2+2-2= 7
DFI_T_PHY_WRDATA =
T_PHY_WRDATA_p0+T_PHY_WRDATA_p1+T_PHY_WRDATA_p2+T_PHY_WRDATA_p3 =
2+3+3+3 = 11
```

Below is an example of timing computing with the CAS-2 command sent on phase 3. The same method applies, except that the final result on DFI_T_PHY_WRLAT is decreased by 3.

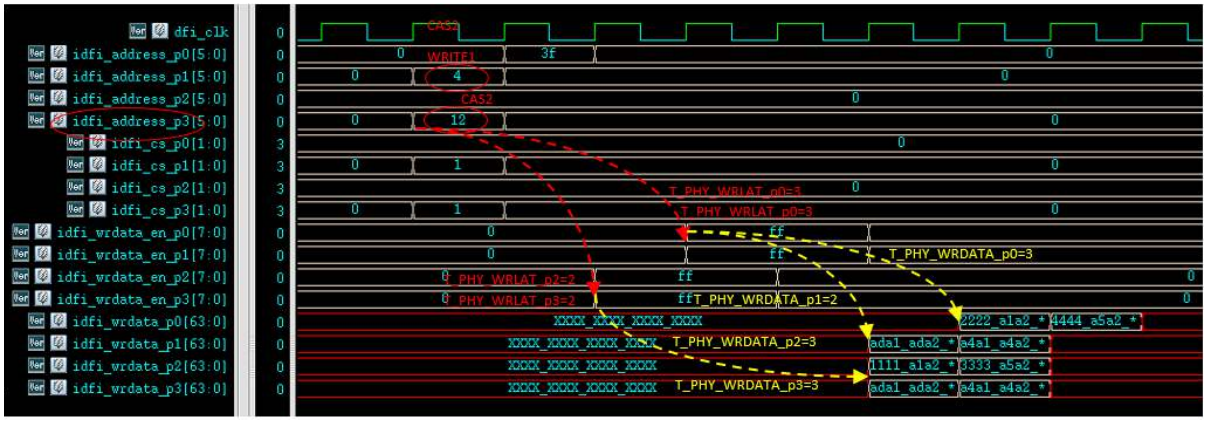

FIGURE 11. Command is on P3

In the above figure, since the command is on p3, substract 3 to the T_PHY_WRLAT parameter computation, as shown below:

```
DFI_T_PHY_WRLAT =
T_PHY_WRLAT_p0+T_PHY_WRLAT_p1+T_PHY_WRLAT_p2+T_PHY_WRLAT_p3-3 = 3+3+2+2-
3= 7
DFI_T_PHY_WRDATA =
T_PHY_WRDATA_p0+T_PHY_WRDATA_p1+T_PHY_WRDATA_p2+T_PHY_WRDATA_p3 = 3+2+3+3
= 11
```

3.1.3 zDFI Memory Timing Parameters

You can define the zDFI memory timing parameters at compile time.

The memory timing parameter values are fixed values. They are NOT overridden by MRS/MRW commands sent to the memory through the DFI. Therefore, any mismatch between the actual latencies programmed in the LPDDR4 memory and the related parameters of the zDFI lead to an incorrect behavior.

The following table describes the parameters for zDFI memory timing:

TABLE 3 Memory Timing Parameters

Parameter	Description
MEM_WRITE_LATENCY	Must match the LPPDDR4 settings (lowest value).
MEM_READ_LATENCY	Must match the LPDDR4 settings (lowest value).
MEM_BURST_LENGTH	Must match the LPDDR4 settings. 0= on-the-fly BL.

3.1.4 Timing Constraints

You can compute the time between a DFI write command and the data reaching the memory interface as described in the following figure:

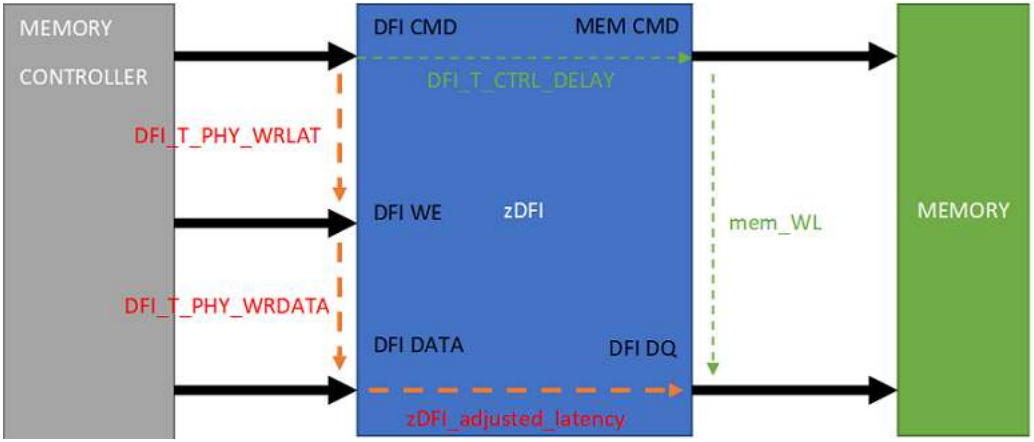

FIGURE 12. Timing Constraints in zDFI

Method 1 (T1; Green Path)

Per the MIPI DFI spec, the write command reaches the memory interface, DFI_T_CTRL_DELAY, after the command reaches the DFI interface.

In addition, per the LPDDR4 JEDEC specification, the data reaches the memory interface, mem_WL (as set in mode register), after the command.

Therefore, the timing constraint is calculated as:

```
T1 = DFI_T_CTRL_DELAY + mem_WL
```

Note that mem_WL can change during the run or from one run to another.

Method 2 (T2; Orange Path)

By definition in MIPI DFI Spec, the data reaches the DFI interface (DFI_T_PHY_WRLAT + DFI_T_PHY_WRDATA) after the command reached the DFI interface.

In addition, by construction (zDFI), the data uses zDFI_adjusted_latency to go through the zDFI (that is, from the DFI interface to the memory interface).

Therefore, the timing constraint is calculated as:

```
T2 = DFI_T_PHY_WRLAT + DFI_T_PHT_WRDATA + zDFI_adjusted_latency
```

T1=T2 ˇ DFI_T_CTRL_DELAY + mem_WL = DFI_T_PHY_WRLAT + DFI_T_PHT_WRDATA + zDFI_adjusted_latency

Note *Mem_WL can change, from run to run, or even during the same run.*

The zDFI detects the MEM_WRITE_LATENCY as mem_WL value (when a Mode register write is performed), and automatically adapts the zDFI_adjusted_latency accordingly, in order to comply with T1=T2.

Points to Consider

- If the mem_WL decreases, then the zDFI_adjusted_latency decreases accordingly so that T1 and T2 remain equal. If mem_WL is small enough, the zDFI_adjusted_latency reaches the minimum possible value, called zDFI_LATENCY. For LPDDR4, this value is 4. Ensure that the following is true:

  ```
  DFI_T_CTRL_DELAY + MEM_WRITE_LATENCY = DFI_T_PHY_WRLAT +
  DFI_T_PHT_WRDATA + zDFI_LATENCY
  ```

- If MEM_WRITE_LATENCY is too small, modify the value of the DFI_T_CTRL_DELAY parameter to ensure that T1=T2.

Increasing the DFI_T_CTRL_DELAY reduces the performance (the zDFI is holding the command instead of transmitting it to the memory ASAP).

This results in the following equation:

```
DFI_T_CTRL_DELAY + MEM_WRITE_LATENCY ≥ DFI_T_PHY_WRLAT +
DFI_T_PHY_WRDATA + zDFI_LATENCY, with zDFI_LATENCY=4
```

¢ The minimum time between command detection on the zDFI interface and command transmission to the memory interface is used by the zDFI to decode the DFI command. This minimum time is 3 CK. That is, DFI_T_CTRL_DELAY(min)=3

Summary

The following formula summarizes the computation of timing constraints:

```
DFI_T_CTRL_DELAY+MEM_WRITE_LATENCY ≥
DFI_T_PHY_WRLAT+DFI_T_PHY_WRDATA+zDFI_LATENCY
zDFI_LATENCY= 3
DFI_T_CTRL_DELAY ≥ 3
```

3.2 zDFI Vector Width Parameters

The following table describes the zDFI width parameters and their default value:

TABLE 4 Width Parameters

Parameter	Default Value (1:2 Clock Ratio)
DFI_ADDRESS_WIDTH	6
DFI_CHIP_SELECT_WIDTH	2
DFI_DATA_ENABLE_WIDTH	8
DFI_DATA_WIDTH	64
DFI_DBI_WIDTH	8
DFI_READ_DATA_VALID_WIDTH	1

The default values are suitable for a zDFI connected to a dual channel LPDDR4 device with 16 data bits per channel. For more information, see *Parallel Channels*.

The following are the constraints regarding bus sizes:

- The following signals have the same size parameterized by DFI_CHIP_SELECT_WIDTH:
  - idfi_cs_pN
  - idfi_cke_pN
  - idfi_odt_pN
  - idfi_reset_n_PN
  - mem_cs, mem_cke
  - mem_odt
  - mem_reset_n
- The following signals have the same size parameterized by DFI_ADDRESS_WIDTH:
  - iidfi_address_pN
  - mem_ad

- The mem_dq signal is a DDR signal. Therefore, it has half the size of idfi_wrdata_pN/ odfi_rddata_wN, parameterized by DFI_DATA_WIDTH.
- The mem_dqs_p, mem_dqs_n and mem_dm signals are 8 times smaller than mem_dq (1 bit per mem_dq byte). Similarly, idfi_wrdata_mask_pN and odfi_rddata_dbi_wN are 8 times smaller than idfi_wrdata_pN/odfi_rddata_wN(1 bit per idfi_wrdata_pN/odfi_rddata_wN). All of them are parameterized by DFI_DATA_WIDTH.

4 DFI Ports

When a port is not used with a given memory type, input ports should be tied to 0 and output ports should be left unconnected.

This section describes the following DFI interfaces:

- *DFI Control Interface*
- *DFI Write Data Interface*
- *DFI Read Data Interface*
- *DFI Status Interface*
- *zDFI Memory Interface*
- *DFI Probe Vector*

4.1 DFI Control Interface

The following table lists the DFI control interface:

TABLE 5 DFI Control Interface

Interface Type	Value	Signal Name	Description
input	1	rst_n	global hardware reset, active low.
input	1	dfi_clk	dfi clock input.
input	1	mem_clk	memory clock input / also used as internal phy clock.
input	DFI_ROW_WIDTH	idfi_address_pN	address input per phase.
input	DFI_CHIP_SELECT_WIDTH	idfi_cs_pN	chip select input per phase .
input	DFI_CHIP_SELECT_WIDTH	idfi_cke_pN	clock enable input per phase
input	DFI_CHIP_SELECT_WIDTH	idfi_reset_n_pN	reset input per phase
input	DFI_CHIP_SELECT_WIDTH	idfi_odt_pN	on-die termination input per phase

4.2 DFI Write Data Interface

The following table lists the DFI write data interface:

TABLE 6 DFI Write Data Interface

Interface Type	Value	Signal Name	Description
input	DFI_DATA_ENABLE_WIDTH	idfi_wrdata_en_pN	write data enable input per phase.
input	DFI_DATA_WIDTH	idfi_wrdata_pN	write data input per phase
input	DFI_DATA_WIDTH/8	idfi_wrdata_mask_pN	write data mask input per phase.
input	DFI_DATA_WIDTH /8	idfi_wrdata_dbi_pN	write data bus inversion input per phase
input	DFI_CHIP_SELECT_WIDTH * DFI_DATA_ENABLE_WIDTH	idfi_wrdata_cs_pN	Unused internally

4.3 DFI Read Data Interface

The following table lists the DFI read data interface:

TABLE 7 DFI Read Data Interface

Interface Type	Value	Signal Name	Description
input	DFI_DATA_ENABLE_WIDTH	idfi_rddata_en_p0N	read data enable input per phase. Not used in zDFI, expect optimization at synthesis.
output	DFI_DATA_WIDTH	odfi_rddata_wN	read data output per phase
output	DFI_READ_DATA_VALID_WIDTH	iodfi_rddata_valid_w0	read data valid output per phase. All bits are identical

TABLE 7 DFI Read Data Interface

Interface Type	Value	Signal Name	Description
Output	DFI_DATA_WIDTH /8	odfi_rddata_dbi_wN	read data bus inversion per phase
Output	DFI_CHIP_SELECT_WIDTH * DFI_DATA_ENABLE_WIDTH	idfi_rddata_cs_pN	Unused internally

4.4 DFI Status Interface

The following table lists the DFI status interface:

TABLE 8 DFI Status Interface

Interface Type	Value	Signal Name	Description
Input	MEM_CK_ WIDTH	idfi_dram_clk_disable	clock disable. Not used in zDFI. Expect optimization at synthesis
output	1	odfi_init_complete	init complete output
input	1	idfi_init_start	init start input

4.5 zDFI Memory Interface

The following table lists the interface of zDFI on the memory side:

TABLE 9 Memory Interface

Interface Type	Value	Signal Name	Description
output	MEM_CK_WIDTH	mem_clk_p	memory clock differential input, positive side of the pair
output	MEM_CK_WIDTH	mem_clk_n	memory clock differential input, negative side of the pair
output	DFI_CHIP_SELECT_WIDTH	mem_cke	memory clock enable
output	DFI_CHIP_SELECT_WIDTH	mem_cs	memory chip select
output	DFI_ADDRESS_WIDTH	mem_ad	memory address
inout tri0	DFI_DATA_WIDTH /2	mem_dq	memory data width (pull down).
inout tri0	DFI_DATA_WIDTH/16	mem_dqs_p	memory data strobe differential I/O; positive side of the pair (pull down).
inout tri1	DFI_DATA_WIDTH/16	mem_dqs_n	memory data strobe differential I/O, negative side of the pair (pull up).
inout	DFI_DATA_WIDTH / 16	mem_dm	memory data mask/ bus inversion.
output	DFI_CHIP_SELECT_WIDTH	mem_reset_n	memory reset. Active low.

4.6 DFI Probe Vector

The following table describes the complete mapping of the debug vector.

TABLE 10 DFI Probe Vector

Probe Vector	Signal Name
dfi_probe[0]	dfi_write_command_0
dfi_probe[1]	dfi_write_command_1
dfi_probe[2]	dfi_read_command_0
dfi_probe[3]	dfi_read_command_1
dfi_probe[4]	dfi_mrw1_command_0
dfi_probe[5]	dfi_mrw1_command_1
dfi_probe[6]	dfi_mrw2_command_0
dfi_probe[7]	dfi_mrw2_command_1
dfi_probe[8]	dfi_cas2_command_0
dfi_probe[9]	dfi_cas2_command_1
dfi_probe[10]	mem_write_command
dfi_probe[11]	mem_read_command
dfi_probe[12]	read_in_progress (Reserved)
dfi_probe[14: 13]	burst_length_detected[1:0] (Reserved)
dfi_probe[20: 15]	burst_length_otf[5:0]
dfi_probe[26: 21]	burst_length_otf_delay[5:0] (Reserved)
dfi_probe[32: 27]	burst_length_actuel[5:0]
dfi_probe[40: 33]	cas_latency_detected
dfi_probe[48: 41]	cas_wr_latency_detected
dfi_probe[52: 49]	additional_rd_latency
dfi_probe[60: 53]	read_latency_detected

TABLE 10 DFI Probe Vector

Probe Vector	Signal Name
dfi_probe[68: 61]	write_latency_detected
dfi_probe[69]	mem_wrdata_out_cnt_start (Reserved)
dfi_probe[77: 70]	mem_wrdata_out_cnt (Reserved)
dfi_probe[78]	write_in_progress[-2] (Reserved)
dfi_probe[79]	write_in_progress[-1] (Reserved)
dfi_probe[80]	write_in_progress[0] (Reserved)
dfi_probe[81]	write_in_progress[1] (Reserved)
dfi_probe[82]	mem_dq_enable
dfi_probe[83]	mem_dqs_enable
dfi_probe[84]	mem_ad_enable (Reserved)
dfi_probe[85]	phy_write_command (Reserved)
dfi_probe[86]	phy_read_command (Reserved)
dfi_probe[87]	phy_wrdata_en_del_r (Reserved)
dfi_probe[88]	mem_wrdata_enable (Reserved)
dfi_probe[89]	~empty_resynch_data_fifo (Reserved)
dfi_probe[90]	~empty_resynch_data_en_fifo (Reserved)
dfi_probe[91]	mem_dqs_toggling (Reserved)
dfi_probe[99: 92]	min_wl[7:0]
dfi_probe[102:100]	param_error[1:0]
dfi_probe[108:103]	suggested_ctrl_delay_min[5:0]
dfi_probe[114:109]	mem_addr_fifo[5:0] (Reserved)
dfi_probe[119:115]	0 (Reserved)
dfi_probe[183:120]	mem_wrdata

TABLE 10 DFI Probe Vector

Probe Vector	Signal Name
dfi_probe[311:184]	mem_rddata
dfi_probe[511:384]	0 (Reserved)

5 Analyzer Feature

The Analyzer feature in DFI enables you to log high-level memory transactions. This feature reports the data, such as, data traffic from / to HBM memory model, in a log file.

This section explains the following topics:

- *Setup*
- *Run Time*
- *Log File Content*
- *Relationship Between Log File and Timing Waveforms*

5.1 Setup

The setup process is divided under the following two phases:

- Before Design Compilation
- After Design Compilation

5.1.1 Before Design Compilation

This model provides an additional input called `analyzer_en` that is used to activate the analyzer feature. This input must be tied @ 1 or connected to a user's design register to prevent the DFI analyzer logic from being optimized during compilation.

If connected to a design register, you can activate or mute the analyzer feature during runtime by forcing a specified register output value.

The Analyzer uses the System Verilog DPI to generate a DFI log file. Therefore, before compiling the design it is mandatory to activate the Zebu DPI support.

The LD_LIBRARY_PATH must be updated to add the directory where the DFI analyzer library (libzDFI_analyzer_64.so) is located.

5.1.2 After Design Compilation

After design compilation a check of the availability of the analyzer feature can be done by searching for the following text string in the `grp0_ccall.cc in` or `*_ccall.cc` file in the zebu.work directory, as shown below:

```
// Dummy function so that back-end is able to resolve the name
// for system tasks
extern "C" void grp0_dummy_import_for_pli () {};
namespace ZDPI_MOD_grp0_zdfi_HDR_analyzer_wrapper {
} // of namespace ZDPI_MOD_grp0_zdfi_HDR_analyzer_wrapper
void zDFI_analyzer_operation_ZDPI_MOD_grp0_zdfi_HDR_analyzer_wrapper
(const unsigned int *din)
{
        svBitVecVal _arg_data[SV_PACKED_DATA_NELEMS(416)];
```

```
        memcpy ((void*)_arg_data, (const void*)(&din[0]),
sizeof(_arg_data));
        zDFI_analyzer_operation (_arg_data);
}
```

5.2 Run Time

The Analyzer feature creates log files at run time. These log files are located in the directory where the run is executed, that is, $PWD. DFI analyzer log file name is the DFI instance path in the design, for example, `tb.dut1.zdfi_1.analyzer_wrapper.analyzer.txt`.

You can also change the file location to a specific path. To do so, specify the path where you want to store all the log files to the DFI_ANALYZER_FILE_PATH environment variable.

All analyzer messages are, by default, logged into files, but they can instead be displayed on the Terminal. To do so, create the DFI_ANALYZER_OUTPUT environment variable and set it to "TERM". To revert to files unset this variable or set it to "FILES".

5.3 Log File Content

The following is an example of the content from an Analyzer log file:

```
Scope : tb.dut1.zdfi_1.analyzer_wrapper.analyzer
GLOBAL CLK | DFI CLK | CMD| PSEUDO CHANNEL| DATA| DBI| DM| PAR|
STAMP        STAMP
-----------------------------------------------------------------
 343198    | 42873 | WRITE| PSEUDO CHANNEL 1| 8386786861c81b72| 54| a7|   3|
 343198    | 42873 | WRITE| PSEUDO CHANNEL 1| 958469c87ba1d257| 14| 4a|   1|
 0         | 42919 | READ | PSEUDO CHANNEL 1 | 8386786861c81b72| 54| a7|   3|
 0         | 42919 | READ |PSEUDO CHANNEL 1  | 958469c87ba1d257| 14| 4a|   1|
```

Where,

- Scope refers to the DFI instance the log refers to.
- Information is organized by column as shown below:
  - **GLOBAL CLK STAMP**: reports a time stamp reported by the svGetTimeFromScope() function.
  - **DFI CLK STAMP**: reports the number of DFI clock posedge.
  - **CMD**: reports whether it is a read or write issued command.
  - **PSEUDO CHANNEL**: reports which pseudo channel is being used.
  - **DATA**: reports the read or write payload.
  - **DBI, DM and PAR:** reports the respective payload values.

5.4 Relationship Between Log File and Timing Waveforms

The following explains how the information in the log file is reported based on the timing waveforms:

- ¢ READ access: Data is available on the DFI bus 2 ½ DFI clock cycles before the reported global time stamp.
- ¢ WRITE access: Data is available on the DFI bus 2 ½ DFI clock cycles after the reported global time stamp.

6 Debug

This section provides information on debugging and troubleshooting the following:

- *Timing Debug*
- *Memory interface*
- *DFI interface*

6.1 Timing Debug

The block that checks the timing parameters in the zDFI interface has the following outputs:

- min_wl
- param_error
- suggested_ctrl_delay_min

The min_wl Value

The min_wl output value is always equal to MEM_WRITE_LATENCY.

As defined in section 3.1, the MEM_WRITE_LATENCY parameter defines the write latency used in the memory. This parameter is important to compute DFI_T_CTRL_DELAY.

The param_error Value

The param_error is a 3-bit vector. Possible error codes are:

- 3'b000: No error
- 3'b001: The zDFI timing condition is not met. Parameters must be recomputed.
- 3'b010 / 3'b011: The zDFI timing condition is not met. DFI_T_CTRL_DELAY is too small.
- 3'b101: DFI_T_CTRL_DELAY is lower than 3.

The suggested_ctrl_delay_min Value

The suggested_ctrl_delay_min is a suggested value for T_CTRL_DELAY in case of error (param_error not equal to 3'b000).

If no parameter error is found, suggested_ctrl_delay_min is 0 and the T_CTRL_DELAY is correct.

If no error is detected and the data is not written/read correctly, then DFI_T_PHY_WRLAT and/or DFI_T_PHY_WRDATA might be erroneous.

6.2 Memory interface

To check the correct behavior of the memory, LPDDR4 memory interface provides a debug vector, as well as an alias file to apply in Verdi.

Also, a part of the probe displays the zrm interface. It is used to understand if the write or the read is failing. None of the mem_* signals on the zDFI should be left unconnected.

For more information, see ZeBu LPDDR4 Memory Models Manual.

6.3 DFI interface

Compared to older revisions, all signals that do not apply to LPDDR4 and the non-supported interfaces have been removed from the zDFI interface. As a result, all signals (idfi_* / odfi_*) of the DFI interface are connected to the controller.

Note *To identify the commands received by the zDFI, see the Command truth table of LPDDR4 memory, either from the JEDEC specification or from the datasheet provided by memory vendors.*

Per this truth table, a valid command is identified by CS high.

TABLE 11 ROW Command Truth Table

Command	Value on idfi_address_pN
MRW-1	0x06 or 0x26
MRW-2	0x16 or 0x36
MRR-1	0x0E or 0x2E
REFRESH / REFRESH ALL	0x08 or 0x28
SELF REFRESH ENTRY	0x18 or 0x38
SELF REFRESH EXIT	0x14 or 0x34
ACTIVATE-1	0x*1 or 0x*5 or 0x*9 or 0x*D
ACTIVATE-2	0x*3 or 0x*7 or 0x*B or 0x*F
PRECHARGE / PRECHARGE ALL	0x10 or 0x30
WRITE-1	0x04 (BL16) or 0x24 (BL32)
MASK WRITE-1	0x0C (BL16) or 0x2C (BL32)
READ-1	0x02 (BL16) or 0x22 (BL32)
CAS-2	0x12 or 0x32