ZeBu DFI Interface User Manual

Version V-2024.03-SP1, February 2025

SYNOPSYS®

Copyright and Proprietary Information Notice

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 https://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 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

Feedback

Contents

1

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.

2

Introduction

DDR PHY Interface is an industry-standard interface protocol for integrating a 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 5.0. It supports DDR3, DDR4, LPDDR1, LPDDR2, LPDDR3, LPDDR4, HBM, and GDDR6.

This section explains the following topics:

Overview

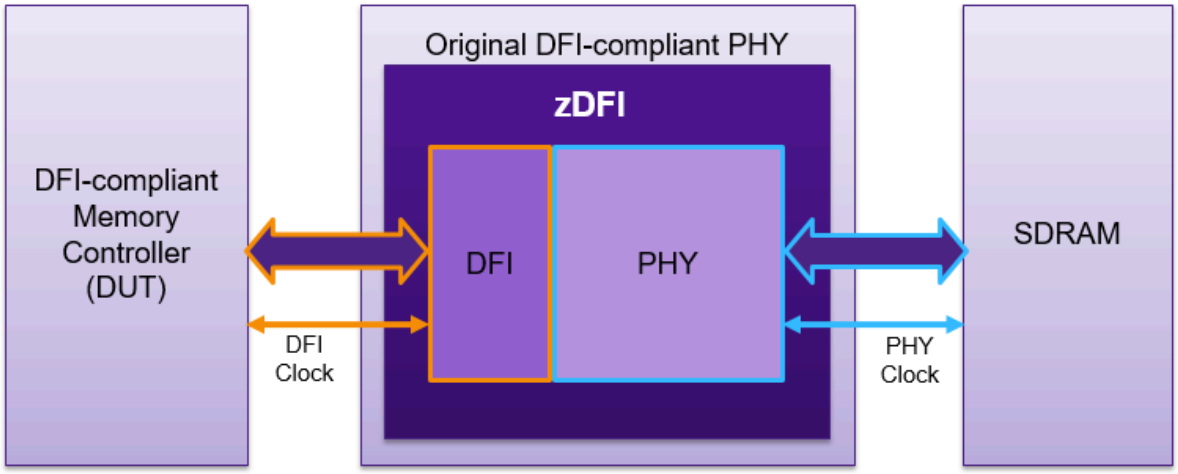
zDFI is a PHY model, compliant with the DFI standard, for ZeBu DDR Memory Models. It is delivered as an encrypted SystemVerilog file, including a SystemVerilog module called zdfi. One package is delivered for each DDR family.

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.

Feedback

The following figure illustrates the DFI Interface:

Figure 1 DFI Interface



Features

The following are the features supported by DFI:

Table 1 Supported Features

SDRAM Family	1:2 (HDR)	Comment
Read transactions	Supported	Transmitted to the DDR model.
Write transactions	Supported	Transmitted to the DDR model.
Configuration operations (MRS/MRW)	Supported	Transmitted to the DDR model.
Configuration operations detection (MRS/MRW)	Supported (Except on HBM)	The zDFI detects these accesses and extracts useful information (Burst length and Write latency).
On-the-fly Burst Length	Supported	DDR3, DDR4, LPDDR1, LPDDR2, LPDDR3, LPDDR4
DFI Update Interface	Not Supported	
DFI optional protocols: • System frequency change • DFI Low Power interface • Error interface	Not Supported	

Feedback

Table 1 Supported Features (Continued)

SDRAM Family	1:2 (HDR)	Comment
Cyclic redundancy check (CRC)	Supported	
Status Interface	Partially supported	Only dfi_init_start/complete
Data Bus Inversion (DBI/DMI)	Supported	HBM, DDR4, LPDDR4 and GDDR6
Parity	Supported	HBM only
Check Bit (CB)	Supported	HBM

Requirements

You need hw_xtormm_dfi license feature.

Limitations

The following are the limitations of the DFI interface:

- Does not support clock ratios 1:1 and 1:4.
- For the HBM memory, initialize the following parameters at compilation time:
  - MEM_BURST_LENGTH
  - MEM_WRITE_LATENCEY
  - MEM_READ_LATENCY

Set the value of the parameters to the value that will be programmed in the Mode registers of the memory at run time. As a consequence, it is not possible to modify the burst length, write latency, or read latency of the memory at run time after the first write or read access completes.

- For the other memory types (all except HBM), initialize the following parameters at compilation time:
  - MEM_BURST_LENGTH
  - MEM_WRITE_LATENCY
  - MEM_READ_LATENCY

Feedback

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.

3

Integration Methodology

This section explains the following topics:

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).
2. Simulate the zDFI example.
3. Simulate the zDFI example using your design’s configuration (bus sizes + timing parameters).
4. Simulate your design using the zDFI model and the VS DDR simulation model.
5. Emulate your design using the zDFI model and the VS DDR model.

Connecting Clocks

DFI standard defines PHY_CLK and DFI_CLK as clock signals, but not explicitly as inputs. zDFI implements them as input clocks, dfi_clk and mem_clk (DFI_PHY_CLK).

- With 1:1 clock ratio (Single Data Rate, SDR): The dfi_clk and mem_clk signals should be the same. However, this is not supported in the DFI Interface.
- With 1:2 clock ratio (Half Data Rate, HDR): The dfi_clk signal should be two times slower than the mem_clk signal, with rising edges aligned.

- Ensure that following conditions are met when connecting a clock:
  - mem_clk is synchronous to the CLK_p or CLK_t of the memory.
  - dfi_clk is synchronous to all the DFI interface signals.

Memory Architecture

In some cases the memory controller is connected to more than one memory. This can happen when there are multiple devices or one device / DIMM with multiple channels or multiple ranks.

If these devices /ranks /channels are accessed independently (i.e. if they have different CS) then each of them should be connected to a different instance of the zDFI.

4

zDFI Parameters

This section explains the following topics:

zDFI Timing Parameters

The following table describes the zDFI timing parameters

Table 2 zDFI Timing Parameters

Parameter	Description
DFI_T_PHY_WRLAT	T_PHY_WRLAT_ is defined as from the last rising edge of the clock that completes a WRITE command to the rising edge of the clock when wrdata_en_ is high.
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.
DFI_T_PHY_RDLAT	Delay between DFI read command and DFI rddata_valid. This parameter is used only with GDDR6. This is not the maximum delay between rddata_en and rddata_valid as described in the DFI specification, but this is the exact delay between the sampling edge of rddata_en and the assertion of rddata_valid.On other devices (all except GDDR6) DFI_T_PHY_RDLAT is not used by the zDFI and the rddata are returned as soon as available

Note:

The above parameters are programmed inside the DFI controller. They should be extracted from the DFI controller parameters and the same values should be used for the zDFI. If lower than 3, DFI_T_CTRL_DELAY should be rounded up to 3.

Feedback

However, if you can not extracted the values of these parameters from the controller parameters, compute them by using the steps described in the below sections.

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.

Specifying Zdfi Timing Parameters Using Zebu_tcfg Register Bank

The zDFI memory timing parameters can also be defined at runtime using the zebu_TCFG register bank.

The following table describes the zDFI timing parameters and corresponding mapping of the zebu_TCFG register bank:

Table 3 zDFI Timing Parameters

Parameter	zebu_TCFG Register Mapping	Description
DFI_T_CTRL_DELAY	Zebu_TCFG[7:0]	Delay between command on the DFI interface and command reaching the memory. Minimum value is 3.
DFI_T_PHY_RDLAT	Zebu_TCFG[15:8]	
DFI_T_PHY_WRLAT	Zebu_TCFG[23:16	Delay between DFI write command and DFI write_en
DFI_T_PHY_WRDATA	Zebu_TCFG[31:24]	Delay between DFI write_en and DFI wrdata

Each timing parameter is accessible with a backdoor access. The path to the zebu_TCFG register is the following.

```
<path_to_zDFI_instance>.zebu_TCFG.zebuReg
```

Any update on any timing parameter at runtime is overwriting the related value defined at compilation time.

This section explains the following topics:

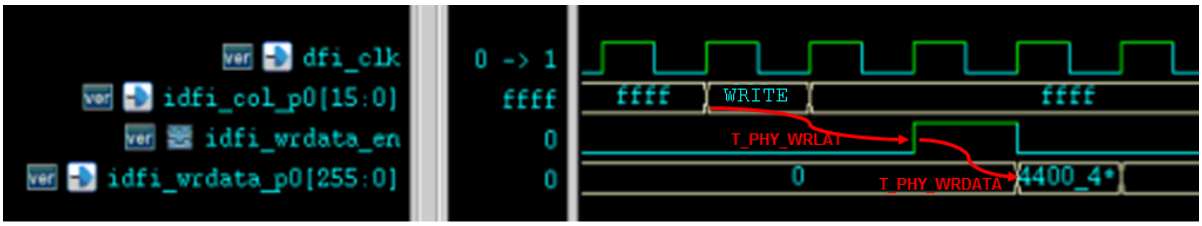
- DFI Timing Parameters for the 1:1 Clock Ratio
- DFI Timing Parameters for 1:2 Clock Ratio
- zDFI Memory Timing Parameters
- Timing Constraints with 1:2 Clock Ratio

Feedback

DFI Timing Parameters for the 1:1 Clock Ratio

In the following waveform, the write command is decoded from idfi_col_p0. Depending on the memory type connected to the zDFI, the write command is extracted from a combination of some or all signals among idfi_cke, idfi_cs, idfi_ras, idfi_cas, idfi_we, and idfi_address.

Figure 2 DFI Timing Parameters for 1:1 Clock Ratio



This clock ratio is not supported in the zDFI but it gives easier understanding of how the timings are computed.

In the above example:

```
T_PHY_WRLAT = 2
T_PHY_WRDATA = 1
```

Note:

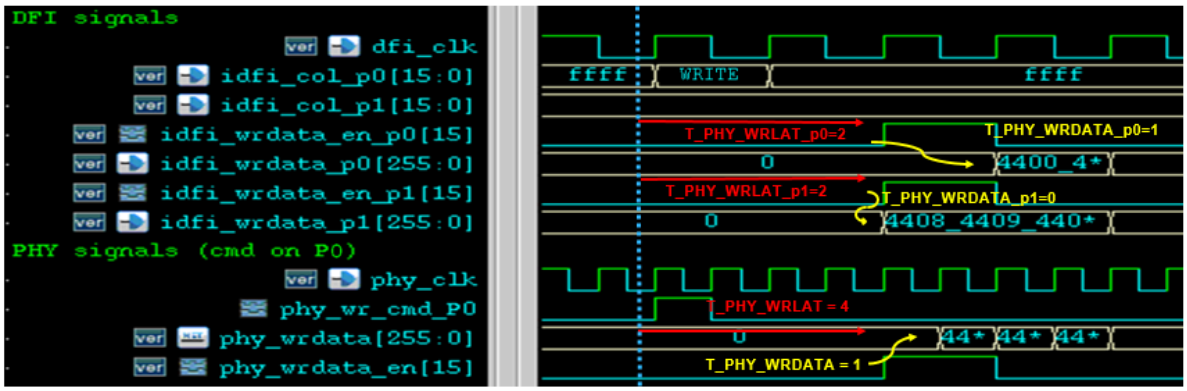
The 1:1 clk ratio is not supported by the DFI Interface. This information is only for reference.

DFI Timing Parameters for 1:2 Clock Ratio

In the following waveform, the write command is decoded from idfi_col_p0/p1. Depending on the memory type connected to the zDFI, the write command is extracted from a combination of some or all signals among idfi_cke, idfi_cs, idfi_ras, idfi_cas, idfi_we, and idfi_address.

Feedback

Figure 3 Command is on P0

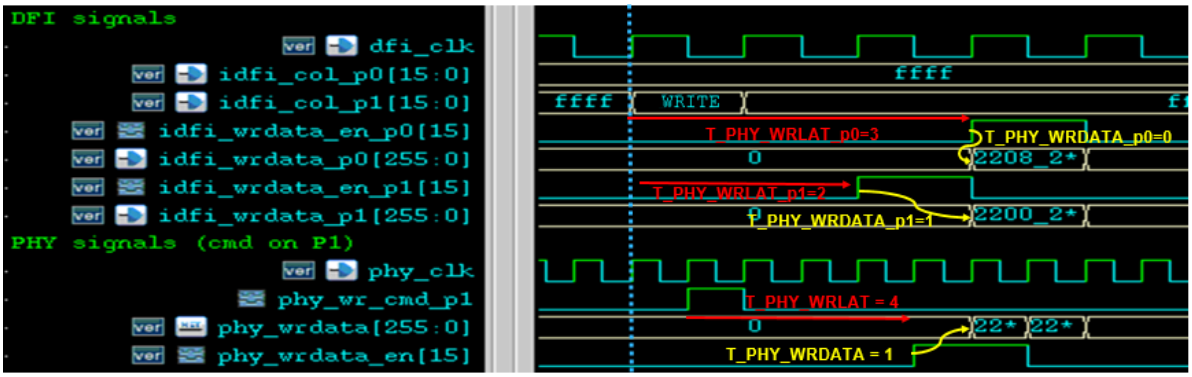


In the above example:

```
T_PHY_WRLAT=T_PHY_WRLAT_p0 + T_PHY_WRLAT_p1 = 2 + 2 = 4
T_PHY_WRDATA=T_PHY_WRDATA_p0 + T_PHY_WRDATA_p1 = 1 + 0 = 1
```

The following figure illustrates how to chose T_PHY_WRLAT or T_PHY_WRDATA when command is on P1:

Figure 4 Command is on P1



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

```
T_PHY_WRLAT=T_PHY_WRLAT_p0 + T_PHY_WRLAT_p1 - 1 = 3 + 2 - 1 = 4
T_PHY_WRDATA=T_PHY_WRDATA_p0 + T_PHY_WRDATA_p1 = 0 +1 = 1
```

Note:

Note the following points:

In case of doubts, calculate the delay between signal1 and signal2 from the first sampling edge of dfi_clk for signal1 to first sampling edge of dfi_clk for signal2.

Feedback

Data can be stable several clock cycles before it is actually sampled and/or can remain valid after the last data is sampled It is important to carefully select the data burst your are using to compute the DFI_T_PHY_WRDATA parameter. As far as possible, find a data burst with data changing at each cycle and not equal to all 0s or all 1s. Masked writes can also be helpful as dfi_wrdata_mask (when active) will toggle when the data is valid.

zDFI Memory Timing Parameters

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

Note the following points:

- For HBM memories, you can not change parameter values at runtime. The corresponding values in the memory mode register must match and cannot change after the first write or read access.
- For all other memories, parameter values are default values. They are overridden by MRS/MRW commands sent to the memory through the DFI.

The following table describes the parameters for zDFI memory timing:

Table 4 Memory Timing Parameters

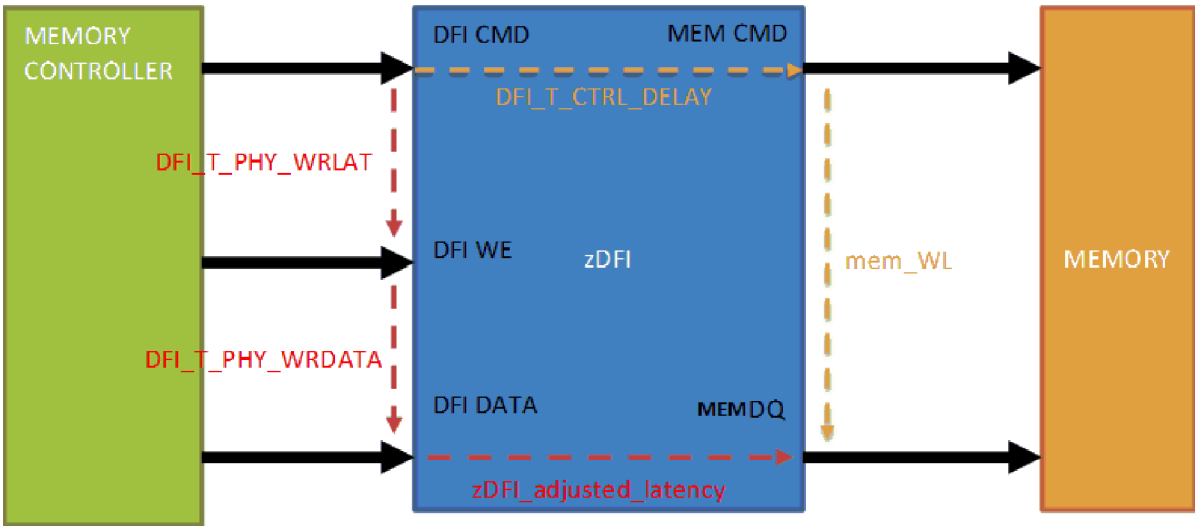
Parameter	Description
MEM_WRITE_LATENCY	Must match the SDRAM settings.
MEM_READ_LATENCY	Must match the SDRAM settings. Needed for HBM only.
MEM_BURST_LENGTH	Must match the SDRAM settings.0= on-the-fly BL

Feedback

Timing Constraints with 1:2 Clock Ratio

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

Figure 5 Computing Timing Constraints



Method 1 (T1; Orange Pass)

By definition in DFI spec, the write command reaches the memory interface

DFI_T_CTRL_DELAY after the command reaches the DFI interface.

In addition, by definition in memory spec, the data reaches the memory interface,

mem_WL (as set in mode register), after the command.

Therefore, the timing constraint is calculated as (Figure 4)

```
T1 = DFI_T_CTRL_DELAY + mem_WL
```

Method 2 (T2; Red Pass)

By definition in 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 (Figure 4)

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

T1=T2 ó 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 WL value (when a Mode register write is performed), and automatically adapts the zDFI_adjusted_latency accordingly, in order to comply with T1=T2. The following are the exceptions to this rule:

- In some cases, the mem_WL can be programmed via zebuMR, meaning that there will be no Mode Register write. In that case, the zDFI will keep the default value programmed by the T_MEM_WRITE_LATENCY parameter.
- The zDFI for HBM does not detect the Mode register writes so it keeps the default value of T_MEM_WRITE_LATENCY during the whole run. The HBM memory must be programmed accordingly.

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, it is possible that zDFI_adjusted_latency reaches the minimum possible value, called zDFI_LATENCY.

Assume that MEM_WRITE_LATENCY is the minimal mem_WL used in a given test environment and 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, you should 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 in DFI for HBM, and 6 in all other
 DFIs
```

- 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

Feedback

Summary:

		DFI_T_PHY_WRLAT
DFI_T_CTRL_DELAY		+
+	≥	DFI_T_PHY_WRDATA
MEM_WRITE_LATENCY		+
		zDFI_LATENCY

where,

zDFI_LATENCY=4 (HBM), or 6 (other memory types)

DFI_T_CTRL_DELAY ≥ 3

zDFI Width Parameters

The zDFI parameters are classified under the following two categories:

- Width Parameters
- Memory Width Parameters

DFI Width Parameters

The zDFI interface requires the following parameters:

- DFI_ADDRESS_WIDTH
- DFI_BANK_WIDTH
- DFI_BANK_GROUP_WIDTH
- DFI_CHIP_ID_WIDTH
- DFI_CHIP_SELECT_WIDTH
- DFI_CONTROL_WIDTH
- DFI_DATA_ENABLE_WIDTH
- DFI_DATA_WIDTH
- DFI_READ_DATA_VALID_WIDTH

- DFI_DBI_WIDTH

  Note:

  In case the above specified parameters are not applicable for the current memory, it is optional to specify a value for these parameters. That is, DFI uses default value for the above parameters.

Memory Width Parameters

The following table describes the zDFI DDR Memory Width parameters and their default value:

Table 5 Memory Width Parameters

Parameter	Default Value (1:2 Clock Ratio)
MEM_ADDRESS_WIDTH (except DDR3)	DFI_ADDRESS_WIDTH
MEM_ADDRESS_WIDTH (for DDR3)	DFI_ADDRESS_WIDTH / 2
MEM_DATA_WIDTH	DFI_DATA_WIDTH / 2
MEM_BANK_WIDTH	DFI_BANK_WIDTH / 2
MEM_DQS_WIDTH	2
MEM_CHIP_SELECT_WIDTH	DFI_CHIP_SELECT_WIDTH
MEM_CKE_WIDTH	1
MEM_CK_WIDTH	1
MEM_ODT_WIDTH	DFI_CHIP_SELECT_WIDTH

5

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

The following table lists the DFI control interface:

Table 6 DFI Control Interface

Memory Type	Interface Type	Signal Name
DDR3 DDR4 LPDDR1 LPDDR3 LPDDR4	input	idfi_address_p0
HBM	input	idfi_row_p0
HBM	input	idfi_col_p0
DDR3 DDR4 LPDDR1	input	idfi_bank_p0
DDR3 DDR4 LPDDR1	input	idfi_ras_n_p0

Feedback

Table 6 DFI Control Interface (Continued)

Memory Type	Interface Type	Signal Name
DDR3 DDR4 LPDDR1	input	idfi_cas_n_p0
DDR3 DDR4 LPDDR1	input	idfi_we_n_p0
DDR3 DDR4 LPDDR1 LPDDR3 LPDDR4	input	idfi_cs_p0
DDR4	input	idfi_act_n_p0
DDR4	input	idfi_bg_p0
DDR4	input	idfi_cid_p0
ALL	input	idfi_cke_p0
DDR3 DDR4 LPDDR3 LPDDR4	input	idfi_odt_p0
DDR3 DDR4 LPDDR4 HBM	input	idfi_reset_n_p0 connect to 1 on LPDDR1 and LPDDR3
DDR3 DDR4 LPDDR1 LPDDR3 LPDDR4	input	idfi_address_p1
HBM	input	idfi_row_p1
HBM	input	idfi_col_p1
DDR3 DDR4 LPDDR1	input	idfi_bank_p1
DDR3 DDR4 LPDDR1	input	idfi_ras_n_p1
DDR3 DDR4 LPDDR1	input	idfi_cas_n_p1,
DDR3 DDR4 LPDDR1	input	idfi_we_n_p1
DDR3 DDR4 LPDDR1 LPDDR3 LPDDR4	input	idfi_cs_p1
DDR4	input	idfi_act_n_p1
DDR4	input	idfi_bg_p1
DDR4	input	idfi_cid_p1
ALL	input	idfi_cke_p1
DDR3 DDR4 LPDDR3 LPDDR4	input	idfi_odt_p1
DDR3 DDR4 LPDDR4 HBM	input	idfi_reset_n_p1

Feedback

Table 6 DFI Control Interface (Continued)

Memory Type	Interface Type	Signal Name
HBM	input	analyzer_en
GDDR6	input	add idfi_abi_p0 / idfi_abi_p1

DFI Write Data Interface

The following table lists the DFI write data interface:

Table 7 DFI Write Data Interface

Memory Type	Interface Type	Signal Name
ALL	input	idfi_wrdata_en_p0
ALL	input	idfi_wrdata_p0
DDR3 DDR4 LPDDR1 LPDDR3 LPDDR4	input	idfi_wrdata_cs_p0
ALL	input	idfi_wrdata_mask_p0
HBM	input	idfi_wr_dbi_p0
HBM	input	idfi_wr_par_p0
ALL	input	idfi_wrdata_en_p1
ALL	input	idfi_wrdata_p1
DDR3 DDR4 LPDDR1 LPDDR3 LPDDR4	input	idfi_wrdata_cs_p1
ALL	input	idfi_wrdata_mask_p1
HBM /	input	idfi_wr_dbi_p0 / idfi_wr_par_p1
GDDR6	input	idfi_wr_par_p1

Feedback

DFI Read Data Interface

The following table lists the DFI read data interface:

Table 8 DFI Read Data Interface

Memory Type	Interface Type	Signal Name
ALL	input	idfi_rddata_en_p0
ALL	output	odfi_rddata_w0
DDR3 DDR4 LPDDR1 LPDDR3 LPDDR4	input	idfi_rddata_cs_p0,
ALL	output	odfi_rddata_valid_w0
HBM	output	odfi_rddata_dnv_w0
DDR4	output	odfi_rddata_dbi_w0/odfi_rddata_dbi_w1
LPDDR4	output ,	odfi_rddata_dbi_w0/odfi_rddata_dbi_w1
HBM		odfi_rddata_dbi_w0/odfi_rddata_dbi_w1
GDDR6	output	odfi_rddata_dbi_w0/odfi_rddata_dbi_w1
HBM	output	odfi_rd_par_w0
ALL	input	idfi_rddata_en_p1
ALL	output	odfi_rddata_w1
DDR3 DDR4 LPDDR1 LPDDR3 LPDDR4	input	idfi_rddata_cs_p1
ALL	output	odfi_rddata_valid_w1
HBM	output	odfi_rddata_dnv_w1
DDR4	output	odfi_rddata_dbi_w0/odfi_rddata_dbi_w1
LPDDR4		odfi_rddata_dbi_w0/odfi_rddata_dbi_w1
HBM	output	odfi_rddata_dbi_w0/odfi_rddata_dbi_w1

Feedback

Table 8 DFI Read Data Interface (Continued)

Memory Type	Interface Type	Signal Name
GDDR6	output	odfi_rddata_dbi_w0/odfi_rddata_dbi_w1

DFI Update Interface

The following table lists the DFI update interface:

Table 9 DFI Update Interface

Memory Type	Interface Type	Signal Name
ALL	input	idfi_ctrlupd_req
ALL	output	odfi_ctrlupd_ack
Not supported (default 0)	output	odfi_phyupd_req
Not supported (default 00)	output	odfi_phyupd_type
Not supported	input	idfi_phyupd_ack

DFI Status Interface

The following table lists the DFI status interface:

Table 10 DFI Status Interface

Memory Type	Interface Type	Signal Name
Not supported (default 0)	output	odfi_aerr_a0
Not supported (default 0)	output	odfi_aerr_a1
Not supported (default 0)	output	odfi_derr_e0
Not supported (default 0)	output	odfi_derr_e1
Not supported	input	idfi_dram_clk_disable
Not supported	input	idfi_data_byte_disable
Not supported	input	idfi_freq_ratio

Feedback

Table 10 DFI Status Interface (Continued)

Memory Type	Interface Type	Signal Name
Not supported (default 1)	output	odfi_init_complete
Not supported	input	idfi_init_start,
Not supported	input	idfi_parity_in_p0,
Not supported (default 1)	output	odfi_alert_n_a0,
Not supported	input	idfi_parity_in_p1,
Not supported (default 1)	output	odfi_alert_n_a1,

DFI Training Interface

The following table lists the DFI training interface:

Table 11 DFI Training Interface

Memory Type	Interface Type	Signal Name
Not supported (Default 0)	output	odfi_rdlvl_req
Not supported (Default Z)	output	odfi_phy_rdlvl_cs
Not supported	input	idfi_rdlvl_en
Not supported (Default 0)	output	odfi_rdlvl_resp
Not supported (Default 0)	output	odfi_rdlvl_gate_req
Not supported (Default Z)	output	odfi_phy_rdlvl_gate_cs
Not supported	input	idfi_rdlvl_gate_en
Not supported	input	idfi_rdlvl_cs
Not supported (Default 0)	output	odfi_wrlvl_req
Not supported (Default Z)	output	idfi_phy_wrlvl_cs,
Not supported	input	idfi_wrlvl_en
Not supported	input	idfi_wrlvl_strobe
Not supported (Default 0)	output	odfi_wrlvl_resp

Feedback

Table 11 DFI Training Interface (Continued)

Memory Type	Interface Type	Signal Name
Not supported	input	idfi_wrlvl_cs,
Not supported (Default Z)	output	odfi_calvl_req
Not supported (Default Z)	output	idfi_phy_calvl_cs
Not supported	input	idfi_calvl_en
Not supported	input	idfi_calvl_capture
Not supported (Default 0)	output	odfi_calvl_resp
Not supported	input	idfi_lvl_pattern
Not supported	input	idfi_lvl_periodic
Not supported (Default 0)	output	odfi_phylvl_req_cs
Not supported	input	idfi_phylvl_ack_cs

DFI Low Power Control Interface

The following table lists the DFI low power control interface:

Table 12 DFI Low Power Control Interface

Memory Type	Interface Type	Signal Name
Not Supported	input	idfi_lp_ctrl_req
Not Supported	input	idfi_lp_data_req
Not Supported	input	idfi_lp_wakeup
Not Supported (default 0)	output	odfi_lp_ack

Feedback

DFI Error Interface

The following table lists the DFI error interface:

Table 13 Error Interface

Signal Name	Memory Type	Interface Type
odfi_error	Not Supported (default Z)	output
odfi_error_info	Not Supported (default Z)	output
idfi_rdlvl_load	Not Supported	input
odfi_rdlvl_mode	Not Supported (default 01)	output
odfi_rdlvl_gate_mode	Not Supported (default 01)	output
idfi_rdlvl_edge	Not Supported	input
idfi_rdlvl_delay_X	Not Supported	input
idfi_rdlvl_gate_delay_X	Not Supported	input
idfi_wrlvl_load	Not Supported	input
odfi_wrlvl_mode	Not Supported (default 01)	output
idfi_wrlvl_delay_X	Not Supported	input

zDFI Interface with Memory

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

Table 14 Memory Side

Signal Name	Memory Type	Interface Type	Comments
mem_clk_p	ALL	output	clk_p has to be connected to ck, ck_p or ck_t
mem_clk_n	ALL	output	|clk_n has to be connected to ck_n or ck_c
mem_cke	ALL	output	

Feedback

Table 14 Memory Side (Continued)

Signal Name	Memory Type	Interface Type	Comments
mem_cs	DDR3 DDR4 LPDDR1 LPDDR2 LPDDR3 LPDDR4	output	|mem_cs is connected to cs on LPDDR4, cs_n on others
mem_act_n	DDR4	output	
mem_cas_n	DDR3 DDR4 LPDDR1	output	
mem_ras_n	DDR3 DDR4 LPDDR1	output	
mem_we_n	DDR3 DDR4 LPDDR1	output	
mem_ba	DDR3 DDR4 LPDDR1	output	
mem_bg	DDR4	output	
mem_odt	DDR3 DDR4 LPDDR3 LPDDR4	output	
mem_ad	DDR3 DDR4 LPDDR1 LPDDR2 LPDDR3 LPDDR4 GDDR6	output	
mem_row	HBM	output	
mem_col	HBM	output	
mem_aerr	HBM	input	
mem_derr	HBM	input	
mem_alert	DDR4	input	
mem_par	HBM	inout	
mem_dbi	HBM GDDR6	inout	inout with pull up for GDDR6, inout HiZ for HBM
mem_dq	ALL	inout	mem_dq has a pull up for GDDR6 and DDR4, pull down for other memories
mem_dqs_p	DDR3 DDR4 LPDDR1 LPDDR2 LPDDR3 LPDDR4	inout	mem_dqs_p has to be connected to dqs, dqs_p or dqs_tmem_dqs_p has a pull up for DDR4, pull down for other memories

Feedback

Table 14 Memory Side (Continued)

Signal Name	Memory Type	Interface Type	Comments
mem_dqs_n	DDR3 DDR4 LPDDR3 LPDDR4	inout	mem_dqs_n has to be connected to dqs_n or dqs_cmem_dqs_n has a pull down for DDR4, pull up for other memories
mem_dm	ALL	inout	
mem_parity	DDR4	output	
mem_wdqs_p	HBM	output	mem_rdqs_p has to be connected to rdqs_t
mem_wdqs_n	HBM	output	mem_wdqs_n has to be connected to wdqs_c,
mem_rdqs_p	HBM	input	mem_rdqs_p has to be connected to rdqs_t
mem_rdqs_n	HBM	input	mem_rdqs_n has to be connected to rdqs_c,
mem_reset_n	DDR3 DDR4 LPDDR4 HBM	output	
mem_wclk_p / mem_wclk_n	GDDR6	output	
mem_cabi_n	GDDR6	output	
mem_edc	GDDR6	inout	with pull up

6

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.

Setup

The setup process is divided under the following two phases:

- Before Design Compilation
- After Design Compilation

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.

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
```

Feedback

```
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);
}
```

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".

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.

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.