SYNOPSYS®

DesignWare® Cores LPDDR54 PHY Diagnostic Firmware

Application Note

DWC LPDDR54 PHY
Firmware Version:C-2021.10

Firmware Version:C-2021.10
Document Version: 1.70a
October 13, 2021

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 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 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.

Synopsys, Inc.

www.synopsys.com

Contents

Revision History

The following tables lists the revision history of the PHY from release to release. Refer to the PHY release notes for a detailed description of PHY component updates.

Note

- Links and references to section, table, figure, and page numbers in this table are only assured to be valid for the version in which the change is made
- In some instances, documentation-only updates occur. The DesignWare IP product information (http://www.designware.com) has the latest documentation

Document Revision	Date	Description
1.70a	October 13, 2021	Updated: ■ Structure reformatting ■ "PHY SRAM Address Map" on page 18 ■ "Diagnostics Message Block" on page 20 ■ "TEST 0x5 : Tx Eye" on page 31 Added: ■ "Preface" on page 7
1.60a	June 15, 2021	Updated: ■ "TEST 0x6 : Rx Eye" on page 34
1.50a	April 19, 2021	Updated: ■ PHY Version and Date
1.40a	December 7, 2020	Updated: 5.10 Test 10: Mode Register Read Updating to A/B-2020-09
1.30a	September 4, 2020	Updated: ■ PHY Version and Date

Document Revision	Date	Description
1.20a	June 10, 2020	Updated: ■ PHY Version and Date
1.11a	November 2020	Added Figure-1
1.10a	April 30, 2020	Initial release

Preface

This Application Note describes the LPDDR54 PHY Diagnostic Firmware.

Recommended Reading

The following documentation provides essential information about the IP to create a complete solution:

- DesignWare Cores LPDDR54 SDRAM PHY Databook, Synopsys, Inc.
- DesignWare Cores LPDDR54 Utility Block (PUB) Release Notes, Synopsys, Inc.

Web Resources

- DesignWare IP product information: http://www.designware.com
- Your custom DesignWare IP page: http://www.mydesignware.com
- Documentation through SolvNetPlus: http://solvnetplus.synopsys.com (Synopsys password required)
- Synopsys Common Licensing (SCL): http://www.synopsys.com/keys

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.

Customer Support

To obtain support for your product:

- First, prepare the following debug information, if applicable:
- For environment setup problems or failures with configuration, simulation, or synthesis that occur within coreConsultant or coreAssembler, use the following menu entry:

File > Build Debug Tar-file

Check all the boxes in the dialog box that apply to your issue. This menu entry gathers all the Synopsys product data needed to begin debugging an issue and writes it to the file <core tool startup directory>/debug.tar.gz.

- For simulation issues outside of coreConsultant or coreAssembler:
  - Create a waveforms file (such as VPD or VCD)
  - Identify the hierarchy path to the DesignWare instance
  - Identify the timestamp of any signals or locations in the waveforms that are not understood
- Then, contact Support Center, with a description of your question and supplying the above information, using one of the following methods:
  - For fastest response, use the SolvNet website. If you fill in your information as explained below, your issue is automatically routed to a support engineer who is experienced with your product. The Sub Product entry is critical for correct routing.

    Go to http://solvnet.synopsys.com/EnterACall and click on the link to enter a call. Provide the requested information, including:

- Product: Designware Cores
- Sub Product: LPDDR54_PHY
- Tool Version:
- Problem Type:
- Priority:
- Title: Provide a brief summary of the issue or list the error message you have encountered
- Description: For simulation issues, include the timestamp of any signals or locations in waveforms that are not understood

After creating the case, attach any debug files you created in the previous step.

- Or, send an e-mail message to support_center@synopsys.com (your email will be queued and then, on a first-come, first-served basis, manually routed to the correct support engineer):
- Include the Product name, Sub Product name, and Tool Version number in your e-mail (as identified above) so it can be routed correctly.
- For simulation issues, include the timestamp of any signals or locations in waveforms that are not understood
- Attach any debug files you created in the previous step.
- Or, telephone your local support center:

- North America:
  Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday.
- All other countries:
  http://www.synopsys.com/Support/GlobalSupportCenters

1 Introduction

The following sections are included in this chapter:

Note This application note pertains to C-2020.11 version of the DWC LPDDR54 Diagnostic firmware. There may be variations in operation, interface, and usage procedure, depending on the DWC LPDDR54 Diagnostic firmware version

1.1 Purpose

The DWC LPDDR54 Diagnostic Firmware is a software tool to obtain diagnostic information from a trained DDR system.

This is delivered as firmware image which contains a number of tests that may be used to troubleshoot or measure the PHY-DRAM link.

In addition to the descriptions of the individual tests, the process for loading and running the firmware, including the prerequisite steps are described.

Since the nature of many of the diagnostic tests involve placing the memory system beyond its typical operating limits, the system is not supported to run mission mode traffic after the diagnostics complete. After performing the diagnostics, the system is required to be reset/reinitialize or powered off.

The DWC LPDDR54 Diagnostic Firmware is a utility intended for diagnostic debug purposes. The DWC LPDDR54 Diagnostic Firmware is not intended for production use

1.2 Overview and Pre-Requisites

The PHY Diagnostic firmware may be executed immediately following PHY training.

In the case of a system with multiple trained pstates – it is up to the user to appropriately choose when to run the diagnostics.

Assuming the example Training + mission mode sequence below:

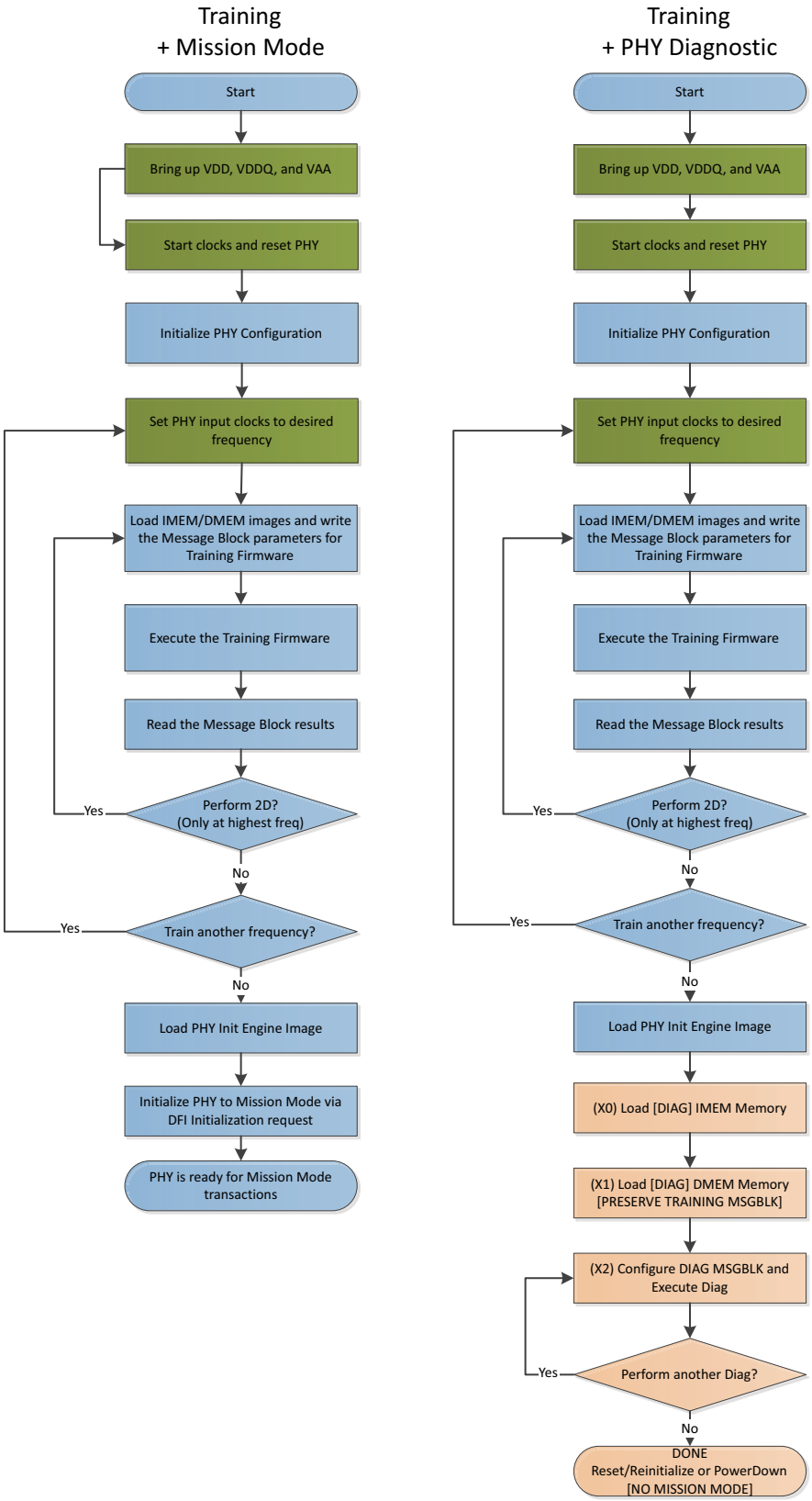
1. Train -P0 @3200
2. DFI Initialization @ P0

The sequence may be interrupted after completion of any of the training steps (a,b,c); and proceed to diagnostics instead of continuing with the complete sequence. The following (figure 1) shows the differences between the standard training + mission mode flow and the PHY Diagnostic flow.

For the Diagnostic Firmware sequence to execute, the PHY must be in the following state:

- Previous Training step executed as expected and returned Message 0x7 [Firmware complete]
- The complete message block from the previous training step is preserved in the PHY DMEM.
- The DFICLK is stable and operating at the same frequency as the previous training step.
- The PHY Register MicroReset = 0x1
- The PHY Register MicroContMuxSel = 0x0
- The PHY Register UcclkHclkEnables= 0x3

Figure 1-1 Initialization flow



The Diagnostic Firmware does not alter the mode register state (except where required by the specific test-type), and relies on the DRAM state to be initialized by the preceding training steps.

In LPDDR4 systems the following dram mode register settings must be constrained if Diagnostics Firmware will be run (mode registers not listed may be set as desired). Individual tests may have additional limitations, refer to the descriptions for each test for further details.

LPDDR4 Mode Register #	Constrained Values	Comments
MR13	OP[3] = {1}	VRCG must be in high current mode for Diags
MR16	OP[7:0] = {0,0,0,0,0,0,0,0}	PASR Mode not supported by Diags
MR17	OP[7:0] = {0,0,0,0,0,0,0,0}	PASR Mode not supported by Diags
MR24	OP[7:4] = {0,0,0,0}	TRR Mode not supported by Diags

1.3 Load Diag IMEM and DMEM (Step X0, X1)

Loading the firmware is accomplished by accessing the PMU SRAMs over the APB bus or JTAG. The SRAMs are mapped into the APB address space starting at address 0x50000 for the instruction memory and 0x58000 for the data memory, out of which the initial 1KB is reserved for training firmware message block. The diagnostic firmware message block starts at 0x58200.

The Diagnostic firmware requires up to 32KB of instruction memory and up to 15KB of data memory. The firmware is provided in two formats:

- Binary image (.bin files)
- Text files with APB address/data pairs (.incv files)

The two formats contain identical firmware images, and the user can use whichever format is more convenient. To use the APB address / data pairs file, write each address with the associated data. To use the binary image file, write the binary contents of the file 16 bits at a time to the memory.

There is a separate file for:

- the instruction memory image (XXX_diags_imem.bin or XXX_diags_imem.incv)
- the data memory image (XXX_diags_dmem.bin or XXX_diag_dmem.incv)

Where XXX is correct protocol for the trained dram.

Both the instruction memory and the data memory image must be loaded. The instruction memory image contains the executable training code. The data memory image contains data structures that the instruction memory image needs to run.

After running the previous training step, the training messageblock is still resident in the PHY SRAM data memory. If the messageblock is not overwritten by the user, it does not need to be reloaded. The full address regions for the Preserved Training Message block, Diag FW IMEM, and Diag FW DMEM is shown in Table 1-1 on page 18.

Table 1-1 PHY SRAM Address Map

Image	Start Address	End Address	Data Width per address increment
DIAG FW IMEM	0x50000	0x53FFF (32kB)	16
Preserved Training MsgBlk	0x58000	0x581FF (1kB)	16
DIAG FW DMEM	0x58200	0x5A200 (16kB)	16

After loading with the Binary image (.bin files), it is necessary to reload the original content of trainning's message block content, that is from 0x58000 ~ 0x581ff. There is no need to reload training's message block if using Text files with APB address/data pairs (.incv files)

While loading firmware IMEM/DMEM images, the csrMicroReset must be set so that {Reset = 0; stall =1 }.

1.4 Configure Diag Message Block and Execute (Step X2)

To run the desired test, the Diagnostics message block needs to be configured. The exact settings needed will vary by test. Refer to Section 4.3 for information on the size and location of the diagnostics message block and chapter for details regarding the individual diagnostic tests.

The diagnostics execution process is as follows:

1. Configure the Diagnostic Message block with the desired message block
2. Reset 1st Microcontroller Message Protocol Bit [write PHY Register UctWriteProt = 0x1]
3. Reset 2nd Microcontroller Message Protocol Bit [write PHY Register DctWriteProt = 0x1]
4. Reset Microcontroller Message [write PHY Register UctWriteOnly = 0x0]
5. Set control of the internal CSR bus to the PMU [write PHY Register MicroContMuxSel = 0x1]
6. Assert PMU Reset and Stall [write PHY Register MicroReset = 0x9]
7. De-assert PMU Reset only [write PHY Register MicroReset = 0x1]
8. De-assert PMU Stall [write PHY Register MicroReset = 0x0]
9. Wait for the Diagnostics test to finish
10. [Poll PHY Register UctWriteOnlyShadow until a completion message is received]
11. See Table 1-2 on page 19 Mailbox Message Values
12. Assert PMU Stall [write PHY Register MicroReset = 0x1]
13. Set control of the internal CSR bus to the APB [write PHY Register MicroContMuxSel = 0x0]
14. Read back diagnostics return data per test.

Refer to each test for details of the return data format

If additional tests are desired, repeat the above execution process.

Table 1-2 Mailbox Message Values

UctWriteOnlyShadow Value	Message meaning
0x00	Reset Value
0x07	Diagnostics finished successfully (firmware complete)
0xFF	Abnormal exit (firmware complete)

1.5 Diagnostics Message Block

The diagnostics firmware has a message block area to allow for configuring of the test and returning of the data. There are three major sections of the message block, the Training Message block, the Diagnostics Message block and the return data.

The beginning of the message block area is the training message block which needs to be preserved from the training. The location is the same as it is used in training. The training message block area is preserved so the Diagnostics Firmware knows the current state of the PHY and the DRAM so that it can correctly execute the test.

The diagnostics message block follows the training message block. It is located 0x400 bytes past the beginning of the training message block at CSR Address 0x58200. It contains the information needed to choose and configure the Diagnostic tests. The names of the arguments are described in the header file format below.

Diagnostics Message Block Definition:

```
typedef struct _PMU_SMB_DIAG_t {
uint8_t DiagTestNum; // Byte offset 0x400, CSR Addr 0x58200, Direction=N/A

uint8_t DiagSubTest; // Byte offset 0x401, CSR Addr 0x58200, Direction=N/A

uint8_t DiagPrbs; // Byte offset 0x402, CSR Addr 0x58201, Direction=N/A

uint8_t DiagRank; // Byte offset 0x403, CSR Addr 0x58201, Direction=N/A

uint8_t DiagChannel; // Byte offset 0x404, CSR Addr 0x58202, Direction=N/A

uint8_t DiagRepeatCount; // Byte offset 0x405, CSR Addr 0x58202, Direction=N/A

uint8_t DiagLoopCount; // Byte offset 0x406, CSR Addr 0x58203, Direction=N/A

uint8_t DiagByte; // Byte offset 0x407, CSR Addr 0x58203, Direction=N/A

uint8_t DiagLane; // Byte offset 0x408, CSR Addr 0x58204, Direction=N/A

uint8_t DiagVrefInc; // Byte offset 0x409, CSR Addr 0x58204, Direction=N/A

uint8_t DiagReserved0A; // Byte offset 0x40a, CSR Addr 0x58205, Direction=N/A

uint8_t DiagXCount; // Byte offset 0x40b, CSR Addr 0x58205, Direction=N/A

uint16_t DiagAddrLow; // Byte offset 0x40c, CSR Addr 0x58206, Direction=N/A

uint16_t DiagAddrHigh; // Byte offset 0x40e, CSR Addr 0x58207, Direction=N/A

uint16_t DiagPatternLow; // Byte offset 0x410, CSR Addr 0x58208, Direction=N/A

uint16_t DiagPatternHigh; // Byte offset 0x412, CSR Addr 0x58209, Direction=N/A

uint8_t DiagMisc0; // Byte offset 0x414, CSR Addr 0x5820a, Direction=N/A

uint8_t DiagMisc1; // Byte offset 0x415, CSR Addr 0x5820a, Direction=N/A
```

```
uint8_t DiagMisc2; // Byte offset 0x416, CSR Addr 0x5820b, Direction=N/A

uint8_t DiagReserved17; // Byte offset 0x417, CSR Addr 0x5820b, Direction=N/A



[…]
uint8_t DiagReserved3F; // Byte offset 0x43F, CSR Addr 0x5821F, Direction=N/A


uint16_t DiagReturnData; // Byte offset 0x440, CSR Addr 0x58220, Direction=N/A
// This byte offset is the pointer to the first address
// of the DiagReturnData ; please see individual tests for
// details of the data structure


} __attribute__ ((packed)) PMU_SMB_DIAG_t;
```

The location of the message block areas are shown in Diagnostics Message Block Definition before.

Table 1-3 Message Block Address Map

Section	CSR Address	DMEM offset (bytes)
1D Message block	0x58000	0x00
Diagnostics Message block	0x58200	0x400
Data Return Data	0x58220	0x440

The last item in the Diagnostics message block is the DiagReturnData field. It points to the beginning of the return data, the format of which varies between tests. The field is defined as two bytes long in the header, but the return data size is defined by the test that is being run.

1.6 Simulating Diagnostics

When running diagnostics in RTL simulation, memory models may flag certain protocol violations. Refresh requirements, such as tRASmax are not always met and some tests (such as the READ test) may use uninitialized data. In these cases those errors can be ignored or waived.

2

Diagnostic Test Details

This chapter contains the following sections:

2.1 Diagnostic Functions Overview

The following table describes the tests that are available for each of the supported DRAM protocols. The Diagnostic Firmware supports multiple protocols, not all Diagnostic routines work with all protocols. Only the diagnostic routines supported by the customer PHY should be used

Test Number	Test	LPDDR4	LPDDR5
1	Reserved		
2	Send Burst Writes	SUPPORTED	SUPPORTED
3	Send Burst Reads	SUPPORTED	SUPPORTED
4	Simple Write Read	SUPPORTED	SUPPORTED
5	Write Eye	SUPPORTED	SUPPORTED
6	Read Eye	SUPPORTED	SUPPORTED
7	Reserved		
8	Reserved		
9	Mode register write	SUPPORTED	SUPPORTED
A	Mode register read	SUPPORTED	SUPPORTED
B	Reserved		
C	Reserved		
D	Reserved		
E	Reserved		
F	Reserved		

2.2 Adjusting Test Run Time

The diagnostic tests allow for the number of bursts per run to be adjusted using the messageblock parameters DiagXCnt, DiagLoopCount, and DiagRepeatCount. Tests that support these looping variables have an internal loop following the below pseudocode.

```
For (x=0; x < diagXCnt; x++){
Activate bank
For (loop=0; loop <= (diagLoopCount); loop++){
For (repeat=0; repeat <= diagRepeatCount; repeat++){
Write 1 burst
}//for
For (repeat=0; repeat <= diagRepeatCount; repeat++){
Read 1 burst
}//for
}//for
Precharge bank
}//for
```

This can be summarized in an equation for the number of bursts sent/received per test run.

BurstCount = DiagXCnt * (1+DiagLoopCount)*(1+DiagRepeatCount).

Tests can offset loop variables, such as making DiagLoopCount specify multiples of 255. When calculating the number of bursts a test will run, it's important to take these offsets into account as part of the loop variables they modify.

2.3 PPGC configuration

All the diagnostics tests have either/both of these two DRAM operations:

- WR command to DRAM to send the pattern specified by DiagPrbs
- RD command to receive back pattern specified by DiagPrbs

To make sure that the PPGC sends correct data on the DiagLane during write operation and uses correct data to compare the received data, the hardware mode to generate DBI lane pattern is used. Each DQ lane and DBI lane carries depending upon diagnostics' message block configuration (DiagLane, DiagPrbs) and the feature status of ReadDbi and WriteDbi. In the case of PRBS mode, PRBS is sent and received on every lane. In the case of pattern mode, data is only sent on the lane under test, and the other lanes are static.

2.4 TEST 0x2 : Send Burst Writes

"Send burst writes" test generates large amounts of write traffic on the bus for customers to observe the trained-behavior of their system during writes. During the burst write test, the phy will send continuous write bursts to the row address provided in two different banks (or bank groups for DDR4), nominally generating write data on all lanes using the input data-pattern. There is a break in DQ traffic continuity for every DiagLoopCount iteration.

Field Used	Range	Description
DiagPrbs	[0,2]	0: Reserved 1: Prbs23 mode 2: pattern mode
DiagRepeatCount	[0,255]	Number of bursts per loop
DiagLoopCount	[0,255]	How many loops to run (multiples of 255)
DiagXCnt	[0,255]	How many times to iterate (0 and 1 will iterate once)
DiagRank		Which rank to test
DiagAddrHigh, DiagAddrLow		Row address to activate
DiagPatternHigh, DiagPatternLow		32 bit-time repeating Pattern to send when [DiagPrbs==2 pattern mode]
DiagMisc4[1]	[0,1]	0: finite burst write 1: infinite burst write (DiagLoopCount DiagXCount are ignored)

Example test pseudocode:

```
For DiagXCnt {
For (DiagLoopCount*255) {
Write diagAddr diagRepeatCount times
}//for
}//for
```

No data is returned by the write memory test and there is no guarantee that the written data will persist in memory once the test has ended.

When in pattern mode, the data pattern restarts at every iteration of the DiagXcnt Loop.

2.5 TEST 0x3 : Send Burst Reads

The send burst read test generates large amounts of read traffic on the bus for customers to observe the trained-behavior of their system during reads.

The burst read test issues DiagRepeatCount times READ commands for alternating banks, to read data from the specified row. There is a break in DQ traffic continuity for every DiagLoopCount iteration.

Field Used	Range	Description
DiagRepeatCount	[0,255]	Number of repeated bursts per loop
DiagLoopCount	[0,255]	How many repeated loops to run (multiples of 255)
DiagXCnt	[0,255]	How many times to iterate (0 and 1 will iterate once)
DiagRank		Which rank to test
DiagAddrHigh, DiagAddrLow		Row address to activate
DiagMisc4[1]	[0,1]	0: finite burst read 1: infinite burst read (DiagLoopCount DiagXCount are ignored)

Example test pseudocode:

```
For DiagXCnt {
Write diagAddr diagRepeatCount times
For (DiagLoopCount*255) {
Read diagAddr DiagRepeatCount times
}//for
}//for
```

Limitations:

The data received on DQ lanes is completely random as there's no WR operation involved in this test.

2.6 TEST 0x4 : Simple Write Read

Field Used	Range	Description
DiagPrbs	[0,2]	0: Reserved 1: Prbs23 mode 2: pattern mode
DiagRepeatCount	[0, 127] [Don't care]	Number of extra bursts per loop(DiagMisc1=2) If DiagMisc1=0
DiagLoopCount	[0, 255]	Number of extra bursts per loop
DiagRank		Which rank to test
DiagAddrHigh, DiagAddrLow		Row address to activate
DiagPatternHigh, DiagPatternLow		32 bit-time repeating Pattern to send when [DiagPrbs==2 pattern mode]

The simple write read test quickly proves that writes and reads to the dram devices are working.

Normally, the write read test sends 63 continuous writes to the target row address using the data pattern provided. The write burst is followed by a read burst checking the contents of the memory addresses just written. If DiagLoopCount is non-zero, the target row will then be overwritten and read back another DiagLoopCount times.

The simple read write test returns data with the following encodings.

Table 2-1 Read Write Diag pass/fail data

Value	Description
0x0	Lane passed
0x1	Lane failed
0xff	Lane was not tested

Results are reported beginning at the address tied to the DiagReturnData messageblock field. Each piece of data is one byte in width. The first byte of the return data is a global pass/fail; it will be set to 1 if any lane in the phy saw errors. The rest of the results report on a per-lane basis which lanes saw errors (1) or ran without issue (0).

Table 2-2 Read Write Test Return Data

Byte Offset	Result	Size in Bytes
0	Pass/Fail	1
1	Reserved	1
2	Dbyte 0 Lane 0 pass/fail	1

Byte Offset	Result	Size in Bytes
3	Dbyte 0 Lane 1 pass/fail	1
4	Dbyte 0 Lane 2 pass/fail	1
6	Dbyte 0 Lane 3 pass/fail	1
7	Dbyte 0 Lane 4 pass/fail	1
8	Dbyte 0 Lane 5 pass/fail	1
9	Dbyte 0 Lane 6 pass/fail	1
10	Dbyte 0 Lane 7 pass/fail	1
11	Dbyte 0 Lane 8 pass/fail	1
12	Dbyte 1 Lane 0 pass/fail	1
2+(9*N)+M	Dbyte N Lane M pass fail	1

The sequence for this test is as follows:

1. All DRAMS on all channels and ranks are sent the exit power down command
2. All channels and ranks are sent the exit self-refresh command
3. The row address specified in DiagAddrLow and DiagAddrHigh is activated
4. The test is run on both channels and the selected rank for DiagLoopCount+1 iterations
5. The complete row is written
6. The complete row is read
7. The hardware training registers is checked for errors
8. All lanes are set to either pass fail or not tested
9. All channels and ranks are sent the enter self-refresh command
10. All channels and ranks are sent the power down command
11. The tests exits

2.7 TEST 0x5 : Tx Eye

Field Used	Range	Description
DiagPrbs	[0,2]	0: Reserved 1: Prbs23 mode 2: pattern mode
DiagRepeatCount	[0, 127] [Don't care]	Number of extra bursts per loop(DiagMisc1=2) If DiagMisc1=0
DiagLoopCount	[0,255]	How many extra loops to run
DiagXCnt	[0,255]	How many times to iterate (0 and 1 will iterate once)
DiagByte		Which byte to measure
DiagLane	[0,8] [9]	Which dq to measure in DiagByte [LPDDR5 only] measure Write Data Link ECC, MR22[6:4]=1 is required.
DiagRank		Which rank to test
DiagAddrHigh, DiagAddrLow		Row address to activate
DiagPatternHigh, DiagPatternLow		Pattern to send when DiagPrbs is 2 (pattern mode)
DiagVrefInc	[1, 127]	Controls granularity of voltage domain. Larger number means less resolution.
DiagMisc0	DiagMisc0 = 0 DiagMisc0 = 1 DiagMisc0 = 2	No filtering Ignore odd bits Ignore even bits All other values reserved
DiagMisc1	DiagMisc1 = 0	Use Write Read FIFO All other values reserved

The TxEye test collects the transmit eye associated with a specifc byte and bit. The eye is measured by running traffic while varying the DRAM VREF and Phy's TxDq delay settings. The test records how many errors occur at each delay and voltage setting and returns a matrix of encoded error counts through the DiagReturnData messageblock field.

TxEye sends a continuous group of DiagRepeatCount write bursts to the target row address using the data pattern provided. The write burst is followed by a continuous group of DiagRepeatCount read burst checking the contents of the memory addresses just written. If DiagLoopCount or diagXCnt is non-zero, the target row will then be overwritten and read back another DiagLoopCount*DiagXCnt times. Once finished the test reads the errors on the target lane and stores the results. TxEye repeats this set of reads and writes for the powerset of all device voltages and txDqDelay settings, storing values all the while.

Error counts for each (delay,voltage) pair are 16 bit values compressed into a single byte to save space. The compression scheme is as follows.

Compressed Result Value	Actual Error count
X=[0x0,0x80]	X
X=[0x81,0xfe]	(X&0x7f) <<7 (approximate)
X=0xff	> 16256 (approximate)

DiagReturnData's results for TxEye are mapped as described below, including the dimensions of the error matrix (nVref x nDly) and the center point found by previous training stages in addition to the error matrix itself. Please note that the data starts with the minimum voltage and increases until the maximum voltage.

Byte Offset	Description	Size in Bytes
Byte Offset	Description	Size in Bytes
0	Number of Delays (nDly)	1
1	Number of Vrefs (nVREF)	1
2	Trained DRAM VREF	1
3	Reserved	1
4	Trained TxDqDelay CSR Value	2
6	Error Count (Delay=0, Vref=0)	1
…	…	…
6+nDly	Error Count (Delay=nDly, Vref=0)	1
7+nDly	Error Count (Delay=0, Vref=1)	1
…	…	..
6+(2*nDly)	Error Count (Delay=nDly, Vref=1)	1
…		…
6+(nVref*nDly)	Error Count (Delay=nDly, Vref=nVref)	1

The rows of the matrix (index nVref) correspond to the DRAM Mode Register value for VrefDQ.

The following equation is used to convert the row index to the VrefDQ:

VrefDQ= (row index) * DiagVrefInc

To convert VrefDq to Volts, please see the Jedec specification for the DRAM protocol of interest.

The columns of the matrix are in units of 1/64*UI.

72columns are correspond to the DQ delay (relative to DQS) -4/64*UI: +68/64*UI from the trained position.

Column Index 0	= Trained position - 36/64*UI
Column Index 36	= Trained position - 36/64*UI
Column Index 71	= Trained position + 35/64*UI

In the case where the Trained position of TxDq is less than 36/64*UI the column indexes correspond to:

Column Index 0 = 0/64* UI

Example test pseudocode:

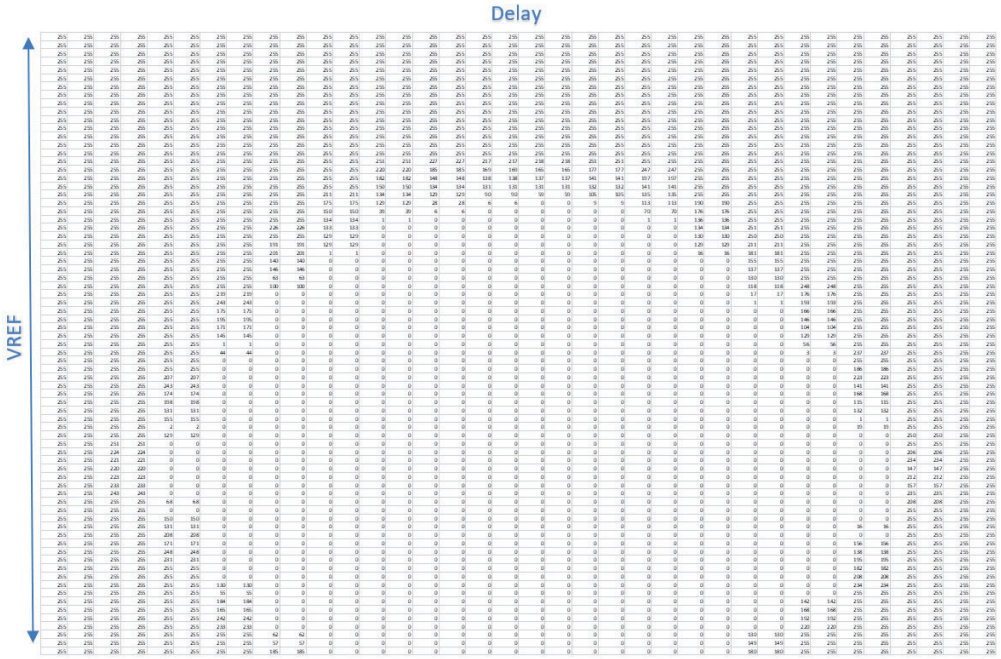
```
CenterDelay = read (csr_txdqdly)
centerVoltage = read messageblock
For voltage (0 to max in steps DiagVrefInc)
For delays (max(0,centerDelay-36) to centerDelay+35){
For DiagXCnt{
For DiagLoopCnt {
Write diagAddr DiagRepeatCount times
Read diagAddr DiagRepeatCount times
}//for
}//for
Record errors
}//for
}//for
```

Example Messageblock ErrorCount Output

TxEye

nVref = 127(LPDDR5) or 82(LPDDR4)

nDly = 72



2.8 TEST 0x6 : Rx Eye

Field Used	Range	Description
DiagPrbs	[0,2]	0: Reserved 1: Prbs23 mode 2: pattern mode
DiagRepeatCount	[0, 127] [Don't care]	Number of extra bursts per loop(DiagMisc1=2) If DiagMisc1=0
DiagLoopCount	[0,255]	How many extra loops to run
DiagXCnt	[0,255]	How many times to iterate (0 and 1 will iterate once)
DiagByte		Which byte to measure
DiagLane	[0,8]	Which dq to measure
DiagRank		Which rank to test
DiagAddrHigh, DiagAddrLow		Row address to activate
DiagPatternHigh, DiagPatternLow		32 bit-time repeating Pattern to send when [DiagPrbs==2 pattern mode]
DiagVrefInc	[1, 127]	Controls granularity of voltage domain. Larger number means less resolution.
DiagMisc0	DiagMisc0[3:0] = 0 DiagMisc0[3:0] = 1 DiagMisc0[3:0] = 2	No filtering Ignore Dqs_C bits Ignore Dqs_T bits All other values reserved
DiagMisc1	DiagMisc1 = 0 DiagMisc1 = 1	Use Write Read FIFO Use Read DQ Calibration All other values reserved
DiagMisc2	DiagMisc2 = 0 DiagMisc2 = 1 DiagMisc2 = 2 DiagMisc2 = 4 DiagMisc2 = 8	Scan all VREFS Scan Only VREF0 Scan Only VREF1 Scan Only VREF2 Scan Only VREF3 All other values reserved
DiagMisc3	DiagMisc3 = 0 DiagMisc3 = 1 DiagMisc3 = 2	No DFE Filtering Filter PrevVal=1 Filter PrevVal=0 All other values reserved

Field Used	Range	Description
DiagMisc4[0]	DiagMisc4[0]=0 DiagMisc4[0]=1	Preserve the trained coarse bit in the RxClk as even/odd valueUse only 1 UI fine delay in RxClk dly

Note If DiagMisc1=1, Rx test runs with Read DQ Calibration. This will alter some mode register value at the SDRAM. The affected MR register are MR15, MR20, MR32 and MR40 in LPDDR4. The affected MR register are MR31, MR32, MR33 and MR34 in LPDDR5.

The RxEye test collects the receive eye associated with a specifc byte and bit. The eye is measured by running traffic while varying the Phy's VREF and RxClkDly delay settings. The test records how many errors occur at each delay and voltage setting and returns a matrix of encoded error counts through the DiagReturnData messageblock field.

RxEye sends a continuous group of DiagRepeatCount write bursts to the target row address using the data pattern provided. The write burst is followed by a continuous group of DiagRepeatCount read burst checking the contents of the memory addresses just written. If DiagLoopCount or diagXCnt is non-zero, the target row will then be overwritten and read back another DiagLoopCount*DiagXCnt times. Once finished the test reads the errors on the target lane and stores the results. RxExe repeats this set of reads and writes for the powerset of all phy voltages and rxClkDelay settings, storing values all the while.

Error counts for each (delay,voltage) pair are 16 bit values compressed into a single byte to save space. The compression scheme is as follows.

Compressed Result Value	Actual Error count
X=[0x0,0x80]	X
X=[0x81,0xfe]	(X&0x7f) <<7 (approximate)
X=0xff	> 16256 (approximate)

DiagReturnData's results for RxEye are mapped as described below, including the dimensions of the error matrix (nVref x nDly) and the center point found by previous training stages in addition to the error matrix itself. Please note that the results are reported from minimum voltage to maximum voltage.

Byte Offset	Description		Size in Bytes
0	Number of Delays (nDly)		1
1	Number of Vrefs (nVref)		1
2	Trained phy VrefDAC0		1
3	Trained phy VrefDAC1		1
4	Trained phy VrefDAC2		1

Byte Offset	Description		Size in Bytes
5	Trained phy VrefDAC3		1
6	Trained Rxclkdly CSR Value		2
8	Error Count (Delay=0, Vref=0)		1
…	…		…
8+Dly	Error Count (Delay=Dly, Vref=0)		1
8+(nDly*Vref)+1	Error Count (Delay=1, Vref=1)		1
…	…		..
8+(Vref*nDly)+Dly	Error Count (Delay=Dly, Vref=Vref)		1

The rows of the matrix (index nVref) correspond to the PHY Register value for VrefDAC0.

The following equation is used to convert the row index to the VrefDAC0 setting:

VrefDAC0 = (row index) * DiagVrefInc

To convert VrefDAC0 to Volts, please see the PUB databook description for PHY Register VrefDAC0.

The columns of the matrix are in units of 1/64*UI.

72 columns correspond to the DQS delay (relative to DQ).

Since the insertion delay of DQS may be larger than DQ, the eye is mapped starting at -4/64*UI.

Column Index 0	= Trained position - 36/64*UI
Column Index 36	= Trained Position
Column Index 71	= Trained position + 35/64*UI

In the case where the Trained position of RxClk is less than 36/64*UI the column indexes correspond to:

Column Index 0 = 0/64* UI

Example test pseudocode:

```
CenterDelay = read (csr_rxclkdly)
centerVoltage = read messageblock
For voltage (0 to 127 in steps DiagVrefInc)
For delays (Trained position - 36/64*UI> to Trained position + 35/64*UI>){
For DiagXCnt{
For DiagLoopCnt {
Write diagAddr DiagRepeatCount times
Read diagAddr DiagRepeatCount times
}//for
}//for
```

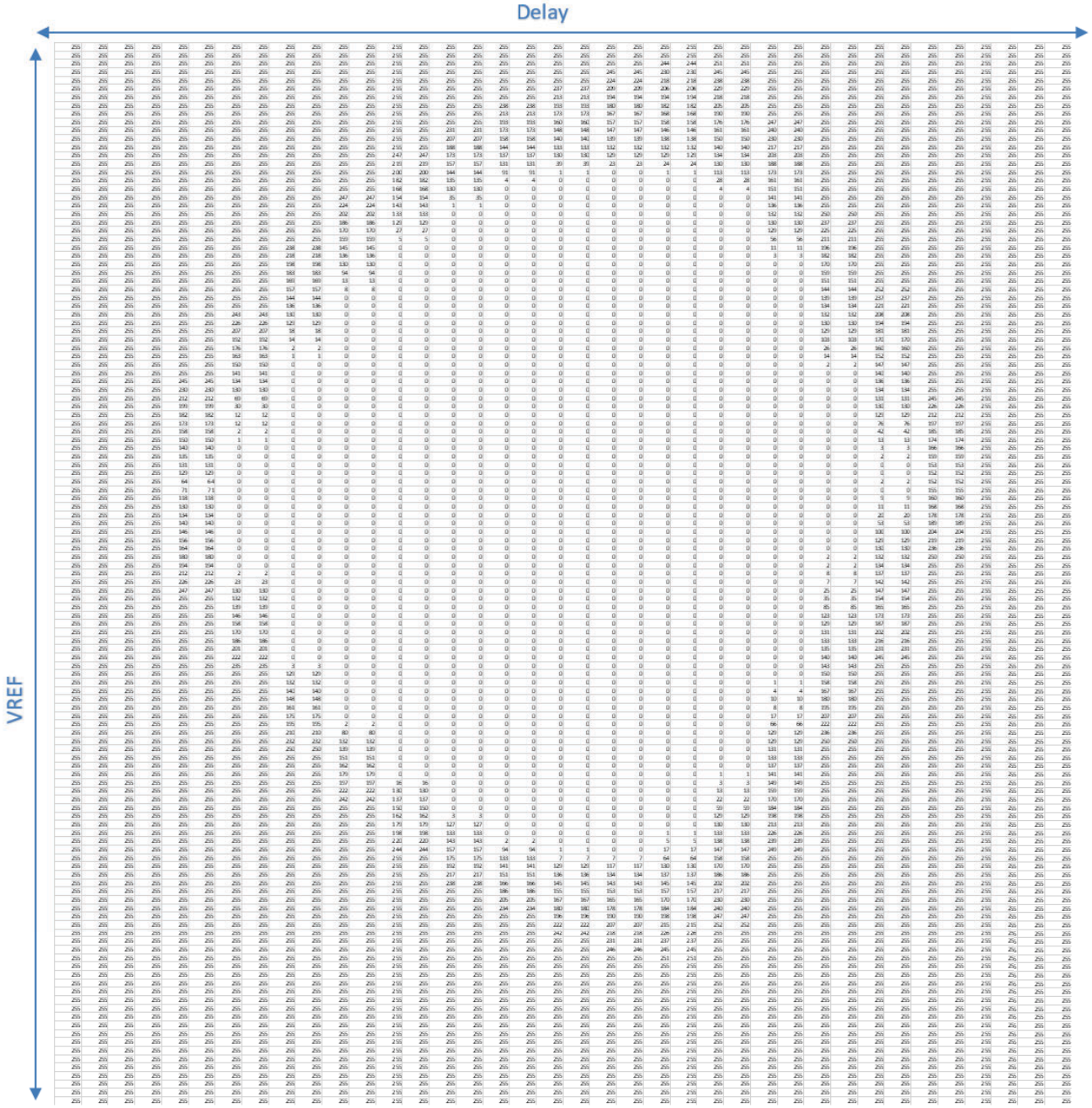
```
Record errors
} //for
}//for
```

When sweeping vref, the phy receiver range is limited to maintain trained difference between vrefDAC0, vrefDAC1, vrefDAC2 and vrefDAC3. Returned data is always relative to VREFDAC0

Example Messageblock ErrorCount Output

nVref = 127

nDly = 72



2.9 Test 9 : Mode Register Write

The “Mode Register Write” test issues MRW command to the specified rank and channel in order to program a given mode register (MR).

Table 2-3 Mode Register Write test parameters

Field used	Range	Description
DiagRank	[0, 1]	Which rank to test
DiagAddrLow	LPDDR4: [0, 40] LPDDR5: [0, 63]	Which MR to write
DiagAddrHigh	[0, 255]	Value to be written
DiagChannel	[0, 1]	ChannelA or ChannelB

After the Mode Register Write test, a MRR is issued by the firmware. The returned data of MRR is reported in DiagReturnData for user to compare with the intended written value specified at DiagAddrHigh.

The sequence for this test is as follows:

LPDDR4/5:

1. All DRAMs on all channels and ranks are sent the power down exit and self-refresh exit commands
2. A MRS command is sent to the specified rank and channel
3. A MRR command is send to the specified rank and channel. Returned data is written to DiagReturn-Data.
4. All DRAMs on all channels and ranks are sent the self-refresh enter and power down commands
5. The test exits

2.10 Test 10 : Mode Register Read

The “Mode Register Read” test issues a MRR command to the specified rank in order to read a given mode register (MR). This test is only supported for LPDDR5 memories. For LPDDR4, a subset of the mode registers can be read using the MPR Read test described in Section 5.7.

Table 2-4 Mode Register Read test parameters

Field used	Range	Description
DiagRank	[0, 1]	Which rank to test
DiagAddrLow	LPDDR4: [0, 40] LPDDR5: [0, 63]	Which MR to read

Results are reported beginning at the DiagReturnData message block field. MR value of all byte will be reported in DiagReturnData area.

The following table illustrates how the MR values are stored in the DiagReturnData area.

Table 2-5 Mode Register Read test return data

Byte offset	Description	Size in bytes
0	Nnmber of bytes return N	2
2	Byte index (eg:byte0)	1
3	MRR value of this byte (eg:byte0)	1
4	Byte index (eg:byte1)	1
5	MRR value of this byte (eg:byte1)	1
…		
(N*2)	Byte index (eg:byteN)	1
(N*2)+1	MRR value of this byte (eg:byteN)	1

The sequence for this test is as follows:

1. All DRAMs on all channels and ranks are sent the power down exit and self-refresh exit commands
2. A MRR command is sent to the specified rank
3. The returned MR values are stored in the return section of the diagnostics message block
4. All DRAMs on all channels and ranks are sent the self-refresh enter and power down commands
5. The test exits