



DesignWare® Cores LPDDR5/4/4X Memory Controller

Databook

DWC LPDDR5/4/4X Controller - Product Code: E092-0
DWC AP LPDDR5/4/4X Controller - Product Code: E094-0
DWC LPDDR5/4/4X Controller AFP - Product Code: E093-0

Copyright Notice and Proprietary Information

© 2021 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

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	13
Preface	17
Databook Organization	17
Reference Documentation	18
Web Resources	18
STAR on the Web (SotW)	18
Synopsys Statement on Inclusivity and Diversity	18
Customer Support	18
Chapter 1	
Product Overview	21
1.1 General Product Description	22
1.2 System-Level Block Diagram	23
1.3 Features	26
1.3.1 General Features	26
1.3.2 Power Saving and Low-Power Features	27
1.3.3 Performance Features	27
1.3.4 Programmable SDRAM Parameters	28
1.3.5 Refresh Control Features	28
1.3.6 Supported Data Rates	28
1.3.7 LPDDR5-Specific Features	28
1.3.8 System Bus Interface	29
1.3.9 Unsupported Features and Limitations	29
1.4 Clock And Reset Requirements	33
1.5 Supported Standards	35
Chapter 2	
Overview of Architecture	37
2.1 Block Diagram	38
2.2 Block Descriptions	39
2.3 Functions of DDRCTL	40
2.4 Signal Naming Conventions	41
Chapter 3	

System Interfaces	43
3.1 AXI Port Interface	44
3.1.1 Overview of AXI Port Interface	44
3.1.2 Read Address Channel	45
3.1.3 Write Address Channel	46
3.1.4 Wrap Burst	47
3.1.5 Read Data and Response Channel	48
3.1.6 Write Data Channel	53
3.1.7 Data Width Conversion	53
3.1.8 Write Response Channel	54
3.1.9 Exclusive Access	54
3.1.10 Transaction Poisoning	56
3.1.11 Signals Related To AXI Port Interface	56
3.1.12 Registers Related To AXI Port Interface	56
3.2 APB Interface	58
3.2.1 Overview of APB Interface	58
3.2.2 Register Classes	58
3.2.3 Signals Related to APB Interface	58
3.3 Host Interface (HIF)	59
3.3.1 Overview of Host Interface (HIF)	59
3.3.2 Dual HIF Functionality	60
3.3.3 Credit Mechanism	61
3.3.4 Valid and Stall	63
3.3.5 Reads	63
3.3.6 Writes	66
3.3.7 Read-Modify-Write Requests	74
3.3.8 WR Versus RMW for HIF Configurations With DBICTL.dm_en=0	75
3.3.9 Half/Quarter Bus Width Mode	76
3.3.10 HIF Data Alignment when MEMC_DRAM_DATA_WIDTH is Non-Power of 2	76
3.3.11 hif_cmd_latency	77
3.3.12 hif_go2critical	78
3.3.13 hif_cmd_autopre	78
3.3.14 hif_mrr_data_valid and hif_mrr_data	78
3.3.15 Additional Error Signal Outputs	78
3.4 Hardware Low-Power Interfaces	79

Chapter 4

DFI	81
4.1 DFI Interface	82
4.1.1 Overview of DFI Interface	82
4.1.2 1:2 Frequency Ratio Mode Considerations	83
4.1.3 DFI Write/Read Data to SDRAM Conversion	83
4.1.4 Command Interface	84
4.1.5 Write Data Interface	84
4.1.6 Read Data Interface	85
4.1.7 Update Interface	85
4.1.8 Status Interface	86
4.1.9 DFI Training	87

4.1.10 Low-Power Control Interface	87
4.1.11 PHY Master Interface	88
4.1.12 Signals Related to DFI Interface	91
4.1.13 Registers Related to DFI Interface	91
4.2 DFI Updates	92
4.2.1 Overview of DFI Updates	92
4.2.2 DFI MC-Initiated Update Requests	92
4.2.3 DFI PHY-initiated Update Requests	93
4.2.4 Registers Related to DFI Updates	94
4.3 DFI Constraints	95
4.3.1 Overview of DFI Constraints	95
4.3.2 WCK Related Constraints (LPDDR5)	95
4.3.3 WCK Constraints (LPDDR5)	96
4.4 DDRCTL to PHY Connections	98

Chapter 5

Channel Modes	101
5.1 Overview of LPDDR4 / LPDDR5 Channel Modes	102
5.2 LPDDR5/4/4X Configurations	103
5.2.1 Single DDRC Single DFI Configuration	104
5.2.2 Single DDRC Dual DFI Configuration	107

Chapter 6

Address Mapping	119
6.1 Overview of Address Mapping	120
6.1.1 System Address Regions	120
6.2 Application to HIF Address Mapping	123
6.3 HIF Address to SDRAM Address Mapping	124
6.4 Recommendations for Optimum SDRAM Utilization	126
6.5 Non-binary Device Densities	128
6.6 Registers Related to Address Mapper	129

Chapter 7

Command Scheduling	131
7.1 Port Arbitration (PA)	132
7.1.1 Overview of Port Arbitration (PA)	132
7.1.2 Read/Write Arbitration	134
7.1.3 Port Timeout	136
7.1.4 DDRC Read Priorities (HPR/LPR/VPR) and Write Priorities (NPW/VPW) for Ports	137
7.1.5 Port Command Priority	137
7.1.6 Round-Robin Arbitration	138
7.1.7 Page Match	138
7.1.8 Write Exclusive Access Lock	139
7.1.9 Signals Related to Port Arbiter	139
7.1.10 Registers Related to Port Arbiter	139
7.2 Queue Status Interface and Port Throttling	140
7.2.1 Overview of Queue Status Interface and Port Throttling	140

7.2.2	CAM Credit Information	140
7.2.3	XPI Queues Information	141
7.2.4	Port Arbiter (PA) Port Throttling	142
7.3	Transaction Service Control	143
7.3.1	Overview of Transaction Service Control	143
7.3.2	Page Policy	143
7.3.3	Transaction Stores	145
7.3.4	Address Collision Handling	159
7.3.5	Write Combine	160
7.3.6	Registers Related to Transaction Service Control	162
7.4	Quality of Service	163
7.4.1	Overview of Quality of Service	163
7.4.2	Traffic Classes	163
7.4.3	Dual Read Address Queue	165
7.4.4	VPR/VPW Timeout	166
7.4.5	Urgent Signaling	168
7.4.6	Enabling QOS	168

Chapter 8

Memory Scheduling	169
8.1 Burst Mode Operation	170
8.1.1 Overview of Burst Mode Operation	170
8.1.2 Bus Width Selection	170
8.1.3 Sequential Operations	170
8.2 Dynamic SDRAM Constraints	172
8.2.1 Overview of Dynamic SDRAM Constraints	172
8.2.2 Timing Constraints	172
8.2.3 Preamble and Postamble (LPDDR4)	179

Chapter 9

Periodic Memory and PHY Maintenance	181
9.1 Refresh Controls	182
9.1.1 Overview of Refresh Controls	182
9.1.2 Refresh Using Direct Software Request of Refresh Command	182
9.1.3 Refresh Using Auto-Refresh Feature	183
9.1.4 Automatic Temperature Derating	186
9.1.5 Signals Related to Refresh Controls	188
9.1.6 Registers Related to Refresh Controls	188
9.1.7 Dynamic SDRAM Rank Constraints	188
9.2 Refresh Management (RFM)	189
9.2.1 Feature Restrictions	189
9.2.2 Rolling Accumulated ACT (RAA) Counter	189
9.2.3 RFM Command Control	189
9.3 ZQ Calibration	191
9.3.1 Overview of ZQ Calibration	191
9.3.2 LPDDR4 Devices	191
9.3.3 LPDDR5 Devices	193

9.3.4	Automatic and Software Initiated ZQ Calibration Commands	195
9.3.5	LPDDR4/LPDDR5 ZQ Reset Command	196
9.3.6	Registers Related to ZQ Calibration	197
9.4	Controller Assisted Drift Tracking (LPDDR5)	198
9.4.1	Overview of Controller Assisted Drift Tracking	198
9.4.2	PHY Snoop Control	198
9.4.3	Registers Related to Controller Assisted Drift Tracking	199
9.5	Enhanced Incremental Periodic Phase Training (PPT2)	200
9.5.1	Overview of PPT2	200
9.5.2	PPT2 Limitations	200
9.5.3	Retraining Interval	202
9.5.4	Normal PPT2	203
9.5.5	Burst PPT2	207
9.5.6	Registers Related to PPT2	209

Chapter 10

Chapter 10		
Memory Control		211
10.1	Mode Register Reads and Writes	212
10.1.1	Overview of Mode Register Reads and Writes	212
10.1.2	Mode Register Writes	212
10.1.3	Mode Register Reads	212
10.1.4	Registers Related to Mode Register Reads and Writes	213
10.2	Data Bus Inversion (DBI)	214
10.2.1	Overview of Data Bus Inversion (DBI)	214
10.2.2	Registers Related to DBI	215

Chapter 11

Low-Power and Power-Saving Features	217
11.1 Overview of Power Saving Features	218
11.2 SDRAM Power Saving Features	219
11.2.1 Overview of SDRAM Power Saving Features	219
11.2.2 Precharge Power-Down	220
11.2.3 Self-Refresh	221
11.2.4 Deep Sleep Mode (LPDDR5)	225
11.2.5 Assertion of dfi_dram_clk_disable	226
11.3 Power Saving in PHY Through DFI Low-Power Control Interface	228
11.4 Hardware Low-Power Interfaces	229
11.4.1 Overview of Hardware Low-Power Interfaces	229
11.4.2 DDRC Hardware Low-Power Interface	229
11.4.3 AXI Low-Power Interface	234
11.5 Fast Frequency Change	239
11.6 Hardware Fast Frequency Change (HWFFC) Support	240
11.6.1 Hardware Fast Frequency Change (HWFFC) Overview	240
11.6.2 Limitation of HWFFC	243
11.6.3 LPDDR4 HWFFC Procedure	243
11.7 Low-Power Optimized Write Data for LPDDR5/4 Masked Write	250
11.8 BSM Clock Removal	251

11.9 Signals Related to Power Saving.....	253
---	-----

Chapter 12

LPDDR5 Specific Features	255
12.1 Overview of LPDDR5 Specific Features	256
12.2 Bank Organization	257
12.2.1 Address Mapping	257
12.2.2 Burst Length	258
12.3 WCK Clocking.....	259
12.3.1 WCK Behavior	259
12.3.2 Enhanced WCK Always On Mode	260
12.4 Link ECC.....	261
12.4.1 Overview of Link ECC Support	261
12.4.2 Enabling Link ECC	261
12.4.3 Link ECC Restrictions	261
12.4.4 Controller Behavior During Read Link ECC Errors	261
12.4.5 Inline ECC and Link ECC Features Combination	262
12.4.6 Link ECC Error Reporting	264
12.4.7 Link ECC Data Poisoning	265
12.4.8 Performance Impact	266
12.4.9 Registers Related to Link ECC Support	266

Chapter 13

RAS Features	267
13.1 Memory ECC.....	268
13.1.1 Inline ECC Support	268
13.2 Scrubber	301
13.2.1 Scrub Period	302
13.2.2 Restrictions on Scrub Interval With Auto Power-down/ Auto Self-refresh Idle Time	303
13.2.3 Normal Operation	303
13.2.4 Address Configuration	303
13.2.5 Address Generation	304
13.2.6 Additional Note	305
13.2.7 Status	305
13.2.8 Hardware Controlled Low-Power Operation	306
13.2.9 Software Controlled Low-Power Operation	307
13.2.10 Initialization Writes	307
13.3 On-Chip Parity (OCPAR).....	309
13.3.1 Overview of On-Chip Parity	309
13.3.2 Enabling On-Chip Parity	310
13.3.3 Write Data Parity	310
13.3.4 Read Data Parity	311
13.3.5 Read Modify Write Operation	313
13.3.6 Parity Poisoning	313
13.3.7 Signals Related to On-Chip Parity	314
13.3.8 Registers Related to On-Chip Parity	315
13.4 On-Chip ECC (OCECC).....	316

13.4.1 Overview of On-chip ECC	316
13.4.2 Enabling On-chip ECC Support	317
13.4.3 ECC Encoding	317
13.4.4 Write Data Protection	318
13.4.5 Read Data Protection	318
13.4.6 Error Detection	318
13.4.7 Address Parity	319
13.4.8 Embedded SRAM Protection	319
13.4.9 ECC/Parity Poisoning	320
13.4.10 Signals Related to On-chip ECC	321
13.4.11 Registers Related to On-chip ECC	321
13.5 Registers Parity Protection (REGPAR).....	322
13.5.1 Overview of Registers Parity Protection (REGPAR)	322
13.5.2 Enabling Registers Parity Protection	322
13.5.3 Register Parity Generation	322
13.5.4 Register Parity Checking	324
13.5.5 Register Parity Poisoning	325
13.5.6 Registers Related to REGPAR	326
13.6 On-Chip Command and Address Path Protection (OCCAP).....	327
13.6.1 Overview of On-Chip Command and Address Path Protection (OCCAP)	327
13.6.2 Enabling On-Chip Command and Address Path Protection	327
13.6.3 Protection Mechanisms	327
13.6.4 Arbiter Protection	330
13.6.5 DDRC Protection	330
13.6.6 Poisoning	332
13.6.7 Signals Related to OCCAP	334
13.6.8 Registers Related to OCCAP	334
13.7 On-Chip External SRAM Address Protection (OCSAP)	335
13.7.1 Overview of On-Chip External SRAM Address Protection	335
13.7.2 Enabling On-Chip External SRAM Address Protection (OCSAP)	335
13.7.3 On-Chip External SRAM Address Parity	335
13.7.4 On-Chip External SRAM Address Parity Poisoning	335
13.7.5 Signals Related to On-Chip External SRAM Address Protection	336
13.7.6 Registers Related to On-Chip External SRAM Address Protection	336

Chapter 14

Parameter Descriptions	337
14.1 HW Configuration / Product Parameters.....	338
14.2 HW Configuration / DDRC Parameters	340
14.3 HW Configuration / Multiport Parameters	347
14.4 HW Configuration / AXI Parameters	356
14.5 HW Configuration / Multi-channel Parameters	360
14.6 HW Configuration / CHI Bridge settings Parameters	361
14.7 HW Configuration / Reliability Features Parameters.....	366
14.8 HW Configuration / RTL Assertions Parameters	372

Chapter 15

Signal Descriptions	373
15.1 Clocks and Resets Signals	376
15.2 AXI Port n Global Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)	378
15.3 AXI Port n Write Address Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals	379
15.4 AXI Port n Write Address On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)	384
15.5 AXI Port n Write Data Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals	385
15.6 AXI Port n Write Data On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)	387
15.7 AXI Port n Write Response Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals	388
15.8 AXI Port n Read Address Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals	389
15.9 AXI Port n Read Address On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)	395
15.10 AXI Port n Read Data Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals	396
15.11 AXI Port n Read Data On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)	398
15.12 Read Reorder Buffer Data RAM Interface (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals	399
15.13 Write Data Parity Signals	402
15.14 Read Data Parity Signals	403
15.15 Write Address Parity Signals	404
15.16 Read Address Parity Signals	405
15.17 On-Chip Command/Address Path Protection Signals	406
15.18 On-Chip ECC Signals	409
15.19 HIF Read Command Interface Signals	410
15.20 HIF Write Command Interface Signals	416
15.21 HIF Command Interface Signals	422
15.22 HIF Write Data Interface Signals	436
15.23 HIF Read Data Interface Signals	439
15.24 Write Data RAM Interface Signals	443
15.25 Mode Register Read/Write Signals	448
15.26 DDRC Hardware Low Power Signals	449
15.27 DDRC Self Refresh Signals	453
15.28 Inline ECC Debug Signals	454
15.29 Performance Logging Signals	455
15.30 Credit Counters Signals	476
15.31 Port Arbiter Signals	478
15.32 ECC Scrubber Signals	479
15.33 DFI Command Interface Signals	480
15.34 DFI Write Data Interface (for n = 0; n <= 1)(for p = 0; p <= 3) Signals	482
15.35 DFI Read Data Interface (for n = 0; n <= 1)(for p = 0; p <= 3) Signals	484
15.36 DFI Update Interface (for n = 0; n <= 1) Signals	486
15.37 DFI Status Interface (for n = 0; n <= 1) Signals	488
15.38 DFI PHY Master Interface (for n = 0; n <= 1) Signals	490
15.39 DFI Low Power Interface (for n = 0; n <= 1) Signals	492
15.40 DFI MC to PHY Message Interface (for n = 0; n <= 1) Signals	495
15.41 DFI WCK Control Interface (for n = 0; n <= 1)(for p = 0; p <= 3) Signals	496
15.42 Non-DFI DDRCTL PHY Sideband Interface (for n = 0; n <= 1)(for p = 0; p <= 3) Signals	497
15.43 LPDDR4 Initialization Handshake Interface Signals	498
15.44 Register Visibility Control Signals	499
15.45 Interrupts Signals	500
15.46 APB Device Interface Signals	507
15.47 APB4 Device Interface Signals	510
15.48 Per-bank refresh Bank number Signals	511

15.49 Register Parity Protection Signals.....	512
Appendix A	
Data Lane Mapping Examples	513
Appendix B	
Controller Performance Details	515
B.1 Timing.....	516
B.2 Area and Power.....	517
B.3 Latency Analysis.....	518
B.3.1 Write Latency With Controller Idle	518
B.3.2 Read Latency With Controller Idle	519
Appendix C	
DWC_ddrctl Automotive Safety.....	521
C.1 Overview of Automotive Safety Feature	522
C.2 Automotive Safety Package Documents.....	523
C.3 Automotive Specific Features.....	524
Appendix D	
Internal Parameter Descriptions.....	525
Appendix E	
Password Protected Features	547

Revision History

The following table provides the history of changes to this databook:

Version	Date	Description
1.10a-lca00	September 2021	<p>Added:</p> <ul style="list-style-type: none"> ■ “STAR on the Web (SotW)” on page 18 ■ “Synopsys Statement on Inclusivity and Diversity” on page 18 ■ “Read Modify Write (RMW) Bypass Path” on page 47 ■ “Byte Mode (x8) Consideration” on page 178 ■ “Refresh Management (RFM)” on page 189 ■ “Enhanced Incremental Periodic Phase Training (PPT2)” on page 200 ■ “Enhanced WCK Always On Mode” on page 260 ■ “Inline ECC and Link ECC Features Combination” on page 262 <p>Updated:</p> <ul style="list-style-type: none"> ■ “General Product Description” on page 22 ■ “Unsupported Features and Limitations” on page 29 ■ “Supported Standards” on page 35 ■ “Block Descriptions” on page 39 ■ “Functions of DDRCTL” on page 40 ■ “Host Interface (HIF)” on page 59 ■ “DFI Write/Read Data to SDRAM Conversion” on page 83 ■ “Frequency Change” on page 86 ■ “DFI MC-Initiated Update Requests” on page 92 ■ “DDRCTL to PHY Connections” on page 98 ■ Figure 5-1 on page 104 ■ Figure 5-4 on page 107 ■ Figure 5-7 on page 112 ■ “Address Mapping” on page 119 ■ “Internal Port Priorities (UMCTL2_EXT_PORTPRIO = 0)” on page 137 ■ “CAM Credit Information” on page 140 ■ “Address Collision Handling” on page 159

Version	Date	Description
1.10a-lca00	September 2021	<p>Continued</p> <ul style="list-style-type: none"> ■ “Registers Related to Transaction Service Control” on page 162 ■ “Burst Mode Operation” on page 170 ■ “Refresh Using Auto-Refresh Feature” on page 183 ■ “Automatic Temperature Derating” on page 186 ■ “Signals Related to Refresh Controls” on page 188 ■ “Registers Related to Refresh Controls” on page 188 ■ “Overview of Mode Register Reads and Writes” on page 212 ■ “Exiting Deep Sleep Mode” on page 226 ■ “Fast Frequency Change” on page 239 ■ “Hardware Fast Frequency Change (HWFFC) Support” on page 240 ■ Table 12-2 on page 258 ■ “WCK Behavior” on page 259 ■ “Controller Behavior During Read Link ECC Errors” on page 261 ■ “Scrubber” on page 301 ■ “Parameter Descriptions” on page 337 ■ “Signal Descriptions” on page 373 ■ “Timing” on page 516 ■ Figure B-2 on page 519 ■ Figure B-4 on page 520 ■ “Internal Parameter Descriptions” on page 525 ■ “Password Protected Features” on page 547 <p>Deleted:</p> <ul style="list-style-type: none"> ■ DFI MC to PHY Message Interface ■ Read/Write Turnaround (MEMC_ENH_RDWR_SWITCH == 0) ■ Refresh Using Direct Access from IOs

Version	Date	Description
1.01a-lca01	January 2021	<p>Added:</p> <ul style="list-style-type: none"> ■ “Hardware Fast Frequency Change (HWFFC) Support” ■ “BSM Clock Removal” <p>Updated:</p> <ul style="list-style-type: none"> ■ “Preface” ■ “General Product Description” ■ “System-Level Block Diagram” ■ “Unsupported Features and Limitations” ■ “Supported Standards” ■ “Write Response Channel” ■ “Software Coherency for AXI Ports” ■ “Half/Quarter Bus Width Mode” ■ “LPDDR5/4 Considerations” ■ Figure 5-3 “DFI Clocks/Data of Single DDRC Single DFI Configuration (1:2/1:1:2 Frequency Ratio Mode)” ■ Figure 5-6 “DFI Clocks/Data of Single DDRC Dual DFI Configuration (1:2/1:1:2 Frequency Ratio Mode)” ■ Figure 5-9 “DFI Clocks/Data of Single DDRC Dual DFI Configuration with 16-bit PHY (1:2/1:1:2 Frequency Ratio Mode)” ■ “Application to HIF Address Mapping” ■ Table 6-5 “Non-binary Device Densities” ■ “Transaction Stores” ■ “Sequential Operations” ■ Table 8-2 “LPDDR4 Command Timing Control Registers” ■ Table 9-1 “Buffer Size Depending on All-Bank and Per-Bank Refresh Modes” ■ “LPDDR4/LPDDR5 ZQ Reset Command” ■ “WCK Behavior” ■ “Link ECC Restrictions” ■ “Scrubber” ■ “Initialization Writes” ■ “Protection Mechanisms” ■ “Parameter Descriptions” ■ “Signal Descriptions” ■ “Automotive Safety Package Documents” ■ “Internal Parameter Descriptions”
1.00a-lca01	June 2020	Initial release

**Note**

In some instances, documentation-only updates occur. The DesignWare IP product <https://www.synopsys.com/designware-ip.html> has the latest information.

Preface

This databook describes the implementation and use of the DesignWare® Cores DDR Memory Controller (DDRCTL), which is a part of a complete LPDDR5/4/4X interface solution.

Databook Organization

The chapters of this databook are organized as follows:

- [“Product Overview”](#) on page 21 provides an introduction to the DDRCTL, including features list.
- [“Overview of Architecture”](#) on page 37 describes the DDRCTL architecture, functional blocks, and signal naming conventions.
- [“System Interfaces”](#) on page 43 describes the DDRCTL interfaces.
- [“DFI”](#) on page 81 describes the DFI interface.
- [“Channel Modes”](#) on page 101 describes LPDDR4 Channel Modes.
- [“Address Mapping”](#) on page 119 discusses DDRCTL address mapping.
- [“Command Scheduling”](#) on page 131 discusses DDRCTL Port Arbitration, Queue Status Interface, Transaction Control, and Quality of Service.
- [“Memory Scheduling”](#) on page 169 describes Burst Mode operation and Dynamic SDRAM Constraints.
- [“Periodic Memory and PHY Maintenance”](#) on page 181 describes Refresh Controls, ZQ Calibration, and Controller Assisted Drift Tracking.
- [“Memory Control”](#) on page 211 describes Mode Register reads and writes, ODT control, and Data Bus Inversion.
- [“Low-Power and Power-Saving Features”](#) on page 217 describes DDRCTL low-power, power saving and Hardware Fast Frequency Change features.
- [“LPDDR5 Specific Features”](#) on page 255 describes LPDDR5 specific features of the DDRCTL.
- [“RAS Features”](#) on page 267 describes Memory ECC, On-Chip Parity, Registers Parity Protection, and On-Chip Command and Address Path protection features of the DDRCTL.
- [“Parameter Descriptions”](#) on page 337 provides details about DDRCTL configuration parameters.
- [“Signal Descriptions”](#) on page 373 provides details about I/O signals of the DDRCTL.

Reference Documentation

- JEDEC LPDDR4 SDRAM Specification, JESD209-4B
- JEDEC LPDDR4 SDRAM Specification, JESD209-4C
- JEDEC LPDDR4X SDRAM Specification, JESD209-4-1
- JEDEC LPDDR5 SDRAM Specification, JESD209-5A
- DDR PHY Interface (DFI) Specification, Version 5.0, April 27, 2018
- Synopsys DWC LPDDR5/4/4X PHY Utility Block (PUB) Databook

JEDEC Specifications are available at <http://www.jedec.org/>

The DDRCTL may not support all the features outlined in the standards listed above. It only supports the features described in this databook.

Web Resources

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

STAR on the Web (SotW)

You must review all STARS on the Web (SotWs) associated with your product. SotWs are considered a part of the Synopsys documentation suite, and show critical information related to your product. To review product SotWs, refer to the DesignWare IP product information:

<https://www.synopsys.com/designware-ip.html>

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 <coreTool 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.
- *For the fastest response, enter a case through SolvNetPlus:*
 - a. Go to the SolvNetPlus website: <https://solvnetplus.synopsys.com>



Note

SolvNetPlus does not support Internet Explorer. Use a supported browser such as Microsoft Edge, Google Chrome, Mozilla Firefox, or Apple Safari.

- b. Click the **Cases** menu and then click the link **Create a New Case** (which is below the list of cases).
- c. Complete the mandatory fields that are marked with an asterisk and click **Save**.
Make sure to include the following:
 - Product L1:** DesignWare Cores
 - Product L2:** Memory-Controller
- d. After creating the case, attach any debug files you created. For more general usage information, refer to the following article in SolvNetPlus:
<https://solvnetplus.synopsys.com/s/article/SolvNetPlus-Usage-Help-Resources>
- Or, send an e-mail message to mailto:support_center@synopsys.com (your email will be queued and manually routed to the correct support engineer on a first-come, first-served basis):
 - ❑ Include the Product L1 and Product L2 names, and Version number in your e-mail 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.
- 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:** <https://www.synopsys.com/support/global-support-centers.html>

1

Product Overview

This chapter includes the following sections:

- [“General Product Description”](#) on page 22
- [“System-Level Block Diagram”](#) on page 23
- [“Features”](#) on page 26
- [“Clock And Reset Requirements”](#) on page 33
- [“Supported Standards”](#) on page 35

1.1 General Product Description

The DesignWare Cores (DWC) LPDDR5/4/4X Controller (DDRCTL) combined with a DWC DDR PHY is a complete memory interface solution for DDR memory subsystems. The DDRCTL is delivered as configurable SystemVerilog source and is compatible with all the popular EDA environments. The DDRCTL is a flexible and advanced solution for ASIC and System-on-Chip (SoC) designers who need very low-power memory interface while achieving industry-leading high efficiency, low latency, and high performance.

The DDRCTL supports the following SDRAM types:

- LPDDR4/4X¹
- LPDDR5

On the host side, there is a choice of an AMBA 4 AXI interface or a custom Host Interface (HIF). The AMBA AXI interface supports single or multi-port configurations, while the HIF supports a single port only per memory channel.

The DDRCTL can accept memory access requests from up to 16 application-side host ports. External AMBA AXI manager ports can be connected to subordinate ports of DDRCTL through the standard AMBA 4 AXI bus interfaces. The configuration registers are programmed through the AMBA 3.0/4.0 APB software interface.



Note

We only support limited configurations with this release.

Contact Synopsys to check whether your configuration is supported.

1. The term LPDDR4 when used in this databook, refers to both SDRAM types LPDDR4 and LPDDR4X.

1.2 System-Level Block Diagram

The DDRCTL and the DWC LPDDR54 PHY combine to create a complete solution for connecting an SoC application bus to DDR memory devices. The combined DDRCTL and DWC LPDDR54 PHY solution enables the highest DDR performance and bandwidth, with the DDRCTL initiating DDR commands to transfer data with maximum efficiency.

The DDRCTL SoC application bus interface supports a choice of an AMBA 4 AXI4 interface or a low-latency HIF. The DDRCTL has a flexible address mapper logic to allow application-specific mapping of row, column, bank, bank group, and rank bits. The DDRCTL includes a DFI interface that supports the following standards:

- LPDDR4/LPDDR4X (JEDEC JESD209-4B/JESD209-4-1)
- LPDDR5 (JEDEC JESD209-5A)

The DDRCTL is highly configurable, making it suitable for a wide range of system architectures. You can use the coreConsultant or coreAssembler (GUI or batch mode) tool to configure and generate the RTL source code.

While coreConsultant is the basic tool used to create a “workspace” for a single component, coreAssembler enables you to work with a component within the context of a subsystem. A workspace is your working version of a DesignWare IP component. The coreConsultant and coreAssembler GUI interfaces enables you to run basic sanity testing using the provided System Verilog Verification environment. You can also perform Synthesis runs using these interfaces. A full System Verilog test environment and tests are also supplied. For more information about using coreConsultant to configure, simulate, or synthesize the DDRCTL, refer to the DesignWare Cores LPDDR54 Memory Controller User Guide.

[Figure 1-1](#) on page 24 is a system-level block diagram of the DDRCTL in a multi-port configuration. The DDRCTL can also be configured to use a single custom host port interface with no port arbitration. In this configuration, the DDRCTL receives read requests, write requests, and data through the single-port lowest latency HIF. Write requests are sent to the DDRCTL followed by the associated write data from the SoC. Read requests are made with a tag field. The DDRCTL returns the tag field with the data when it returns the read data.

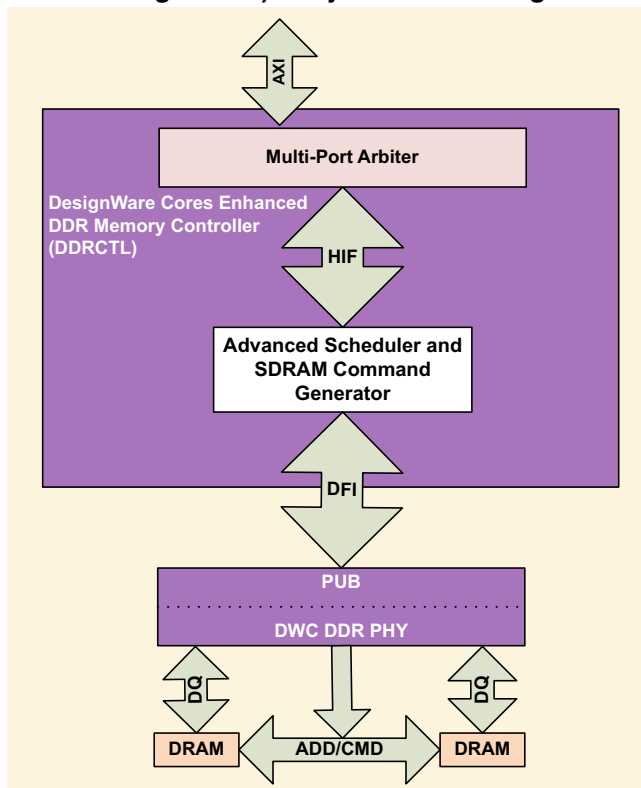
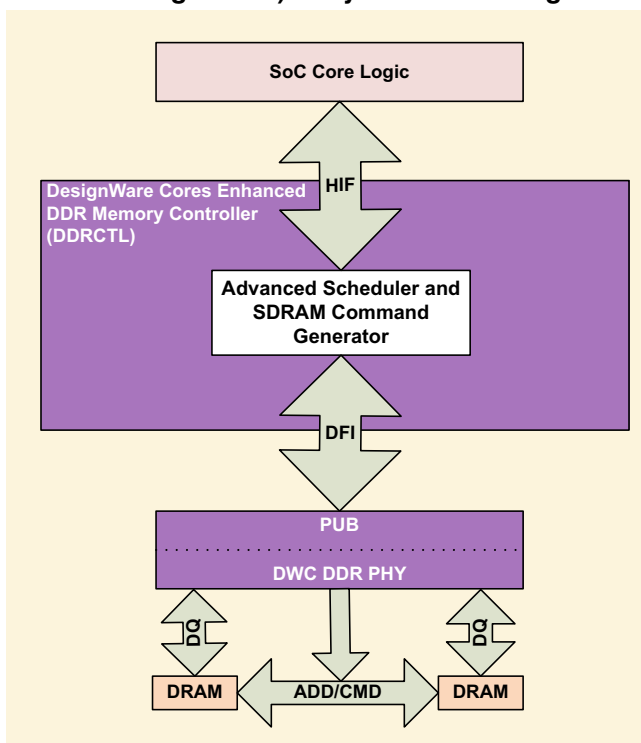
Figure 1-1 DDRCTL (Multi-Port AXI Configuration) in System-Level Diagram

Figure 1-2 is a system-level block diagram of the DDRCTL in a single-port HIF configuration.

Figure 1-2 DDRCTL (Single-Port HIF Configuration) in System-Level Diagram

The DDRCTL receives transactions from the SoC. These transactions are queued internally and scheduled for access while satisfying the SDRAM protocol timing requirements, transaction priorities, and dependencies between the transactions. The DDRCTL in turn issues commands on the DFI interface to the PHY module, which launches and captures data to and from the SDRAM.

1.3 Features

This section describes the features of the DDRCTL controller. It consists of the following subsections:

- “General Features” on page 26
- “Power Saving and Low-Power Features” on page 27
- “Performance Features” on page 27
- “Programmable SDRAM Parameters” on page 28
- “Refresh Control Features” on page 28
- “Supported Data Rates” on page 28
- “LPDDR5-Specific Features” on page 28
- “System Bus Interface” on page 29
- “Unsupported Features and Limitations” on page 29
- “Simulators” on page 30

1.3.1 General Features

The DDRCTL controller supports the following features:

- Complete, integrated, single-vendor , LPDDR4, LPDDR5 solution when combined with the Synopsys DWC LPDDR5/4/4X PHY
- DDR PHY Interface (DFI) for easy integration with industry-standard PHYs:
 - All control, write data, and read data interface signals
 - Update interface:
 - MC-initiated requests
 - PHY-initiated requests
 - Training: PHY-independent training mode
 - Low-power interface
 - DFI PHY master interface
- Scalable and software-controlled 1:2/1:4 (LPDDR4) or 1:1:2/1:1:4 (LPDDR5) frequency ratio architecture
- For LPDDR4 protocol:
 - Direct software request control or programmable internal control for ZQ calibration
 - MPC (ZQCal Start/Latch) commands can be issued automatically after SR-Powerdown exit
 - ZQ Reset MRW command can be issued through software
- Dynamic scheduling to optimize bandwidth and latency
- Read and write buffers in fully associative CAMs, configurable in powers of two, from 16 up to 64 reads and 64 writes
- Delayed writes for optimum performance on SDRAM data bus
- 1 or 2 memory ranks
- Control options to avoid starvation of lower priorities

- Guaranteed coherency for write-after-read (WAR) and read-after-write (RAW) hazards (always on the HIF interface and on the AXI interface only if appropriate hardware configuration parameter and software register are set)
- Flexible address mapper logic to allow application specific mapping of row, column, bank, and rank bits
- Low-area, low-power architecture

1.3.2 Power Saving and Low-Power Features

- Automatic SDRAM power-down entry and exit caused by lack of transaction arrival for a programmable time
- Automatic Clock Stop entry and exit caused by lack of transaction arrival
- Automatic DDRCTL low-power mode operation caused by lack of transaction arrival for a programmable time through the hardware low-power interface
- Advanced power-saving design including no unnecessary toggling of command, address, and data pins (RAS/CAS/WE/BA/A hold last state after each command; DQ does not transition on writes when bytes are disabled)
- Self-refresh entry and exit:
 - Automatic self-refresh entry and exit caused by lack of transaction arrival for a programmable time
 - Self-refresh entry and exit under software control
 - Self-refresh entry and exit using dedicated DDRC hardware low-power interface control (similar to the AMBA 4 AXI protocol low-power control interface)

1.3.3 Performance Features

- For maximum SDRAM efficiency, commands executed out-of-order:
 - Read requests accompanied by a unique token (tag) from HIF
 - Read data returned with token (tag) for SoC to associate read data with correct read request
- Hardware configurable and software programmable Quality-of-Service (QoS) support:
 - Three traffic classes on read commands – high priority reads, variable priority reads, and low priority reads
 - Two traffic classes on write commands – normal priority writes and variable priority writes
 - Port urgent and port throttling control
- If QoS support is not configured in the hardware:
 - Two traffic classes on read commands – high priority reads and low priority reads
 - One traffic class on write commands – normal priority writes
- Write combine to allow multiple writes to the same address to be combined into a single write to SDRAM; supported for same starting address
- 5-clock cycle typical command latency through the DDRCTL (HIF interface):
 - Can be reduced to 4 cycles by choosing not to register DFI outputs (Configuration parameter)
 - 3-clock cycles for high priority read

- Leverages out-of-order requests with CAM to maximize throughput

1.3.4 Programmable SDRAM Parameters

- Configurable maximum SDRAM data-bus width (denoted as “full data-bus width”)
- Programmable support for all of the following SDRAM data-bus widths:
 - Full data-bus width, or
 - Half of the full data-bus width (see “LPDDR5/4/4X Configurations” on page 103)
- Paging policy selectable by configuration registers as any of the following:
 - Leave pages open after accesses, or
 - Close page when there are no further accesses available in the controller for that page, or
 - Auto-precharge with each access, with an optimization for page-close mode which leaves the page open after a flush for read-write and write-read collision cases
- Explicit SDRAM mode register updates under software control

1.3.5 Refresh Control Features

- Controller-generated auto-refreshes at programmable average intervals.
- In multi-rank designs, an offset can be applied to the refresh timer for each rank to allow rank refreshes to expire at different times (this can increase efficiency by allowing traffic to continue to other ranks while a given rank is being refreshed).
- Ability to group up to eight controller-generated refreshes together to be issued consecutively (this reduces the frequency of page closings, increases overall efficiency).

Per-bank refreshes are scheduled for banks with no traffic in the CAM to minimize impact of refreshes on throughput.
- When controller-generated refreshes are grouped, some refreshes can be issued speculatively when the controller is idle for a programmable period of time.
- Ability to disable controller-generated auto-refreshes.
- Ability to issue a refresh through direct software request.
- Selectable ability to perform per-bank refreshes rather than all-banks refreshes.

1.3.6 Supported Data Rates

DDRCTL supports LPDDR4 protocol up to LPDDR4-4267 speed grade and LPDDR5 protocol up to LPDDR5-6400.

1.3.7 LPDDR5-Specific Features

DDRCTL supports the following LPDDR5 features:

- Support for two bank architectures selected by mode register. The maximum data rate depends on the bank architecture.
- WCK clocking.

1.3.8 System Bus Interface

- APB interface for the DDRCTL software accessible registers
- Up to 16 host ports using AMBA AXI
- For host ports with the AXI interface:
 - Compatibility with the AMBA 4 AXI4 AXI protocol
 - AXI burst types: Incremental and wrap
 - AXI clock asynchronous/synchronous to the controller clock
 - Exclusive access support
 - Read reorder buffer with reduced latency options (for example, bypass)

1.3.9 Unsupported Features and Limitations



Note

For the most current information about unsupported features and limitations, refer to the Release Notes.

Table 1-1 lists unsupported features and limitations in this version of the LPDDR5/4/4X Controller.

Table 1-1 Unsupported and Unverified Features in DWC LPDDR5/4/4X Memory Controller

Category	Known Issues and Limitations	Notes
LPDDR5	8-bank mode is not supported.	
LPDDR5	BL32 is not supported.	
LPDDR5	Write X is not supported.	
LPDDR5	Data Copy is not supported.	
LPDDR5	CAS (B3) command (that is, Non-Zero Burst Start Address for RD) is not supported.	This limitation is only for HIF configuration (hif_address[3:0] must always be 0).
LPDDR5/4	Data rate less than 1066Mbps is not fully verified.	Contact Synopsys for more information.
LPDDR5/4	DFI Low-Power Control Handshaking for data is not fully verified.	
LPDDR5/4	Changing DFI frequency ratio dynamically is not supported.	
LPDDR5	DVFSC mode change is verified only during SR/SRPD.	Contact Synopsys for more information.
LPDDR4	HWFFC is supported only for LPDDR4/4X SDRAM with limited configuration.	Data rate has to be within the range between 1066Mbps and 4266Mbps. Contact Synopsys for more information.
LPDDR5	LPDDR5 HWFFC is not supported.	

Category	Known Issues and Limitations	Notes
DFI or PHY	Controller Assisted Drift Tracking for LPDDR4 (DQSOSC) is not recommended to be used.	Contact Synopsys for more information.
DFI or PHY	Changing DBI mode dynamically is not supported after initialization.	LPDDR54 PHY does not support it.
DFI or PHY	DBI mode in the PHY is not verified (only DFIMISC.phy_dbi_mode=0 is supported).	
Config/topology	Controller HW cannot be configured as dual channel configuration (Only single DDRC can exist in the controller).	Refer to the databook for more details.
Config/topology	Only 1:4 HW configuration is supported (DFI 1:2 mode or DFI 1:4 mode can be chosen by TMGCFG register during reset).	
Config/topology	Half Bus Width (HBW) is supported only with MEMC_DATA_WIDTH=32 and 16-bit PHY.	Refer to the databook for more details. For HBW mode, contact Synopsys for more information.
Config/topology	Quarter Bus Width (QBW) is not supported.	
Application Interface	CHI is not supported in LPDDR5/4/4X Controller.	
Automotive	ASIL- B Ready Certification packaged in this release is for Config1, Config2 and Config4. Config3 in FMEDA/Safety Manual is for reference only.	Contact Synopsys directly to agree usage of this Config3 in your application following future certification process.
TB	Warning-[FCPSBU] Invalid values in bin Warning-[FCIBR] Invalid bin range.	To avoid these warnings, if UMCTL2_A_IDW is set to a larger value than 23, the following command needs to be executed in the command line of coreConsultant GUI or included it in your configuration batch file: set_activity_parameter DWC_ddrctl_Simulate suppress_warning_noFCPSBU_plus_noFCIBR 1
TB	UVM_ERROR uvm_test_top [write_export_dfi] test_dwc_ddrctl_perf_vseq cast failed	If DDRCTL_HW_RFM_CTRL is set to 1, this UVM_ERROR is expected to be seen in the test test_dwc_ddrctl_default_settings_lpddr5.

1.3.9.1 Simulators

Only Synopsys VCS simulator is supported.

1.3.9.2 DFI

- DFI 5.0 Interface to PHY (April 27, 2018)
 - “DFI disconnect protocol” feature
 - LPDDR54 Dual Channel
 - DFI Error Interface

1.3.9.3 LPDDR4

- Multi-purpose command (MPC) is not supported except ZQ Calibration
- Precharge-All (PREA) command
- WR32/RD32/WFF/RFF/RDC commands
- Modified refresh (for derating)
- BL32 functionality
- Self-refresh abort mode
- Density 1 Gb
- Partial array self-refresh (PASR)
- Mixed package configuration (Packages configured with both Standard and Byte-Mode die)

1.3.9.4 LPDDR5

- 8B mode
- Write X
- Data Copy
- Command-based calibration mode
- Only ZQCal Latch, Start WCK2DQI Osc, Start WCK2DQO Osc are the supported Multi-Purpose Command (MPC).
- Precharge-All (PREA) command
- WR32/RD32/WFF/RFF/RDC commands
- RFMab command is not supported (only RFMpb is supported regardless of RFSH-MOD0.per_bank_refresh)
- First CS toggle (PDX) after power on
- BL32 functionality
- Self-refresh abort mode
- Partial array self-refresh (PASR)
- Mixed package configuration (Packages configured with both Standard and Byte-Mode die)
- Refresh Management (RFM) single-bank counter implemented (RFMSBC)
- CAS-WS_FS command is not issued when WCK Always On Mode is disabled (MSTR4.wck_on=0)

1.3.9.5 Channel Modes and Supported Memory Bit Width

- LPDDR Dual-Channel Mode
- Memory bit width (`MEMC_DRAM_DATA_WIDTH`) must be set to '16' or '32' in the LPDDR5/4/4X Controller.

1.3.9.6 ECC

- Sideband ECC

1.3.9.7 AXI

- `axregion` and `axcache` signals
- Locked accesses
- FIXED transaction type (that is, only INCR and WRAP are supported)
- AxPROT signaling
- ACE (as well as ACE-lite)

1.3.9.8 Limitations

- When the register `DRAMSET1TMG4.t_rcd` is set to '1', the activate to column timing must be one cycle. However, the actual activate to column timing is two cycles.
- Limitation with `RFSHMOD0.refresh_burst` update:
 - When the `RFSHMOD0.refresh_burst` register is changed and the `RFSHCTL0.refresh_update_level` register is toggled, the `DDRCTL` sends the first burst of refreshes using the previous value of the register `RFSHMOD0.refresh_burst`. Subsequent bursts of refreshes use the new value of `RFSHMOD0.refresh_burst`.
 - The workaround is to toggle the `RFSHCTL0.refresh_update_level` register twice to avoid this issue. The value of `refresh_burst` is latched internally on the first toggle, and the `DDR` controller uses it on the second toggle.

1.4 Clock And Reset Requirements

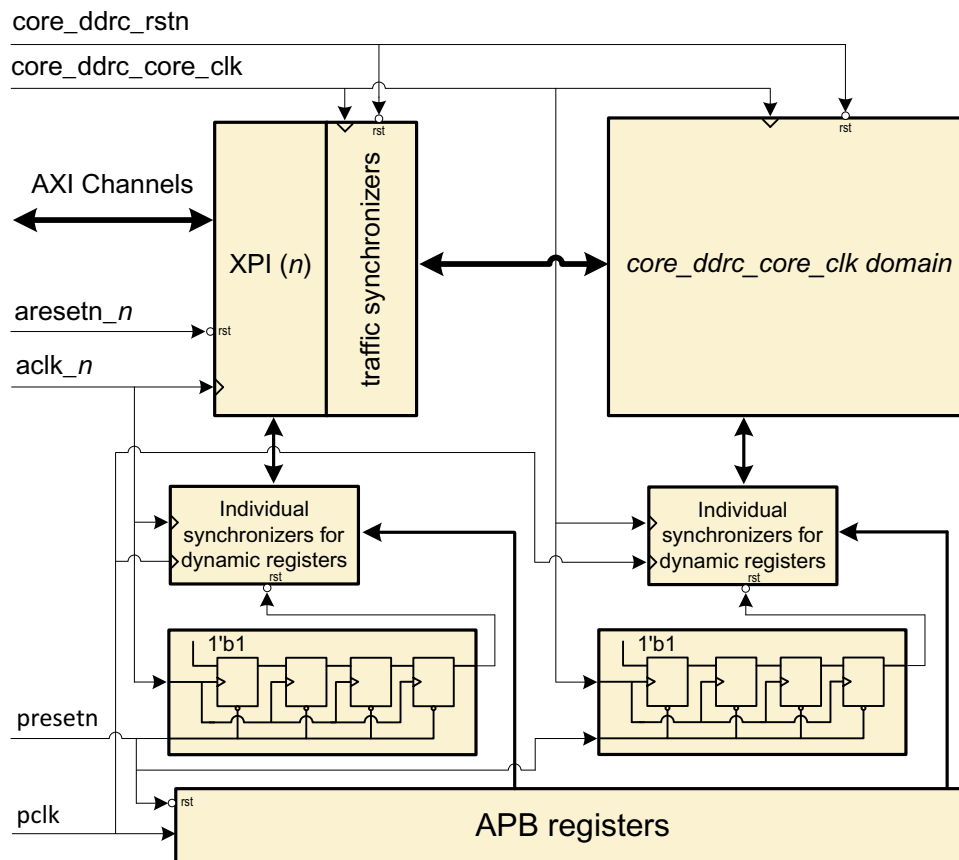
The main clock for the DDRCTL is `core_ddrc_core_clk`. The `core_ddrc_core_clk` clock is synchronous to the PHY clock. It can be driven from an external clock tree driving the DDRCTL and PHY clocks together.

All application port clocks (`aclk_n`) are independently configurable as asynchronous or synchronous with respect to the main DDRCTL clock (`core_ddrc_core_clk`). The APB interface clock (`pclk`) is normally asynchronous to the `core_ddrc_core_clk`. The interfaces that need to be asynchronously configured, incur latency and area increase due to the additional clock domain crossing circuitry.

The number of clock domains within the DDRCTL depends on the setting of the hardware parameters `UMCTL2_P_ASYNC_EN` and `UMCTL2_A_SYNC`.

Figure 1-3 shows a typical example with three clock domains—a `core_ddrc_core_clk` domain, a `pclk` domain, and `aclk` domain. The hardware parameter `UMCTL2_P_ASYNC_EN=1` and the hardware parameter `UMCTL2_A_SYNC_n =0`.

Figure 1-3 Example with Three Clock Domains



An application port's clock (`aclk_n`) is considered synchronous when it is phase-aligned and has equal frequency to the DDRCTL `core_ddrc_core_clk`.



Note

- The clock to the SBR block (`sbr_clk`) is separated from the controller clock (`core_ddrc_core_clk`), but they must be synchronous with each other.

For more information on reset and programming of the registers, see the “Clocks, Resets, and Clocking Scheme” section of the DesignWare Cores LPDDR54 Memory Controller User Guide.

**Note**

`pclk` frequency must be equal to or less than `core_ddrc_core_clk` frequency.

The following range of ratios of clocks is verified when asynchronous support is enabled:

- `pclk`: `core_ddrc_core_clk` = 1:20 <-> 1:1 (frequency)
- `ac1k_n`: `core_ddrc_core_clk` = 1: 10 <-> 9:1 (frequency)

Every input signal is assumed to be synchronous to a specific clock unless otherwise specified.

Port related signals are always synchronous to the correspondent `ac1k_n`, while DDR inputs are synchronous to `core_ddrc_core_clk`.

Exception list:

- `cactive_in_ddrc`, which is present only if no arbiter is instantiated, is asynchronous. This means that you can register the signal in the source clock domain before feeding it to DDRCTL.

Reset requirements:

- The `core_ddrc_rstn` reset must always be asserted before the `presetn` reset.
- If `pclk` is synchronous to `core_ddrc_core_clk` (`UMCTL2_P_ASYNC_EN=0`), then `core_ddrc_rstn` reset assertion must be always followed by `presetn` reset assertion. Only resetting `core_ddrc_core_clk` domain is not allowed.
- The Scrubber reset signal (`sbr_rstn`) must have the same source as `core_ddrc_rstn` (must be asserted at the same moment).
- In dual channel configurations, the `core_ddrc_rstn_dch1` reset signal must have the same source as `core_ddrc_rstn` (must be asserted at the same moment).
- In arbiter configurations (`UMCTL2_INCL_ARB == 1`) the `aresetn` signal must always be asserted before `presetn`. The `aresetn` and `core_ddrc_rstn` signals must have the same source (must be asserted at the same moment).

**Note**

All resets are active low, so "assert" means "logic low".

1.5 Supported Standards

The DDRCTL implements the following standards:

- JEDEC LPDDR4 SDRAM Specification, JESD209-4B (unless otherwise specified)
- JEDEC LPDDR4X SDRAM Specification, JESD209-4-1
- JEDEC LPDDR5 Specification, JESD209-5A
- DDR PHY Interface DFI 5.0 Specification, April 27, 2018
- AMBA 4 AXI4 Specification from Arm

**Note**

The DDRCTL may not support all features outlined in the standards listed above. It only supports the features described in this databook.

2

Overview of Architecture

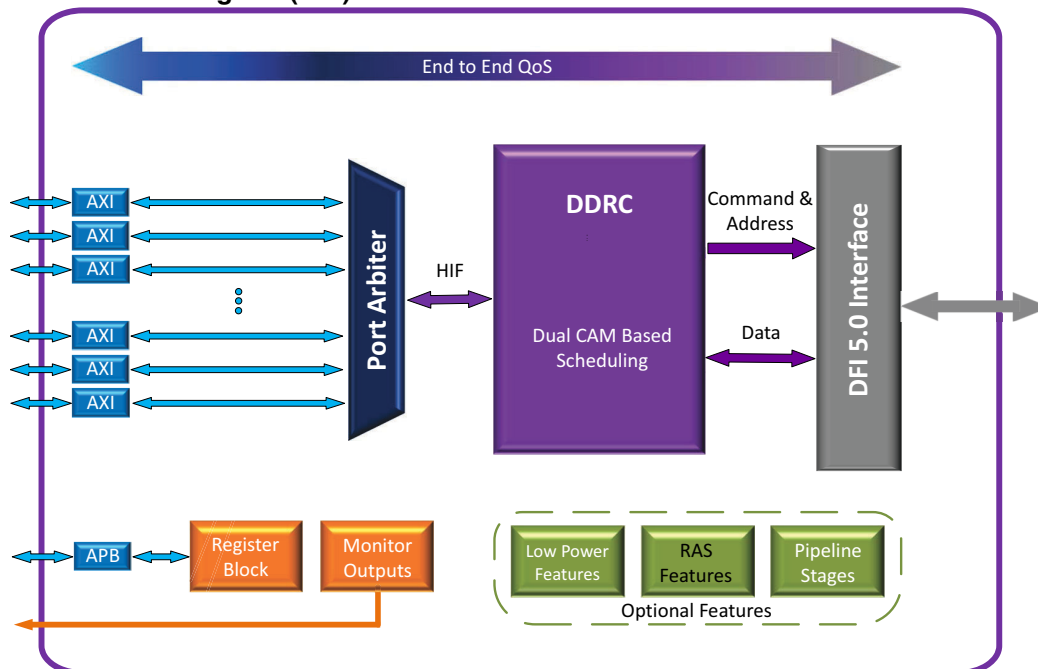
This chapter explains the functionality of the DDRCTL controller. The DDR controller consists of the following main architectural blocks:

- [“Block Diagram”](#) on page 38
- [“Block Descriptions”](#) on page 39
- [“Functions of DDRCTL”](#) on page 40
- [“Signal Naming Conventions”](#) on page 41

2.1 Block Diagram

The DDRCTL controller converts system bus transactions into memory commands on the DFI interface that are compliant with the DDR protocols. [Figure 2-1](#) is the block diagram of the DDRCTL.

Figure 2-1 DDRCTL Block Diagram (AXI)



2.2 Block Descriptions

The DDR controller contains the following main architectural components:

- The AXI Port Interface (XPI) block: This block provides the interface to the application ports. It provides bus protocol handling, data buffering and reordering for read data, data bus size conversion (up-sizing or downsizing), and memory burst address alignment. Read data is stored in a SRAM, read re-order buffer and returned in order, to the AXI ports. The SRAM may be instantiated as embedded memory external to the DDRCTL or implemented as flip-flops within the DDRCTL.
- The Port Arbiter (PA) block: This block provides latency sensitive, priority based arbitration between the addresses issued by the XPIs (by the ports).
- The DDR Controller (DDRC) block: This block contains a logical CAM (Content addressable memory), which can be synthesized using standard cells. This holds information on the commands, which is used by the scheduling algorithms to optimally schedule commands to be sent to the PHY, based on priority, bank/rank status and DDR timing constraints.
- The APB Register Block: This block contains the software accessible registers.

Details on DDRC are as follows:

- Write data is stored in a SRAM (internal and external to the DDRCTL) until its associated command is issued to the PHY. The SRAM can be instantiated as embedded memory external to the DDRCTL or internal.
- Read data is handled by the response engine in the DDRC and is returned in the order of scheduled read commands on the HIF.
- ECC handling is an optional function, which is handled by logic modules within the DDRC in the write data path and in the response engine.

The frequency ratio between the DDRC clock (`core_ddrc_core_clk`) and the memory clock is controlled by register `TMGCFG.frequency_ratio`. For further information on the valid values, refer to `TMGCFG` section in the DWC DDRCTL LPDDR5/4 Programming Guide.

The interface between the DDRCTL controller and PHY follows the DFI 5.0 Specification.

The terms “DFI clock” and “DFI PHY clock” are used in this document in accordance with the DFI specification:

- The DFI clock is the clock on which the DFI interface runs. It is the `core_ddrc_core_clk`.
- The DFI PHY clock frequency is the same frequency as the SDRAM clock (corresponds to `MEMCLK` for certain DWC DDR PHYs).

In 1:2 frequency ratio (LPDDR4), this has twice the frequency of the DFI clock. In 1:4 frequency ratio (LPDDR4), this has four times the frequency of the DFI clock.

In 1:1:2/1:1:4 frequency ratio (LPDDR5), this has same frequency of the DFI clock.

- The DFI PHY data frequency is the same frequency as DFI PHY clock frequency for memory system other than LPDDR5.

For LPDDR5, in 1:1:2 frequency ratio, this has twice the frequency of the DFI PHY clock, in 1:1:4 frequency ratio, this has four times the frequency of the DFI PHY clock.

2.3 Functions of DDRCTL

The DDRCTL controller performs the following functions:

- Accepts requests from the SoC with system addresses and associated data for writes.
- Performs address mapping from system addresses to SDRAM addresses (rank, bank, bank group, row).
- Prioritizes requests to minimize the latency of reads (especially high priority reads) and maximize page hits.
- SDRAM initialization can be performed for LPDDR4, but is not recommended. SDRAM initialization engine does not exist for LPDDR5.
- Ensures that all requests made to the SDRAM are legal (accounting for associated SDRAM constraints).
- Ensures that refreshes and other SDRAM and PHY maintenance requests are inserted as required.
- Controls that the SDRAM enters and exits the various power saving modes appropriately.
- Generates different interrupt signals. All the interrupt signals are level-sensitive and active-high unless otherwise stated.

For example, *_intr_fault signals are exceptions. Those interrupt signals are defined as 2-bit width, and 2'b10 means fault detected.

2.4 Signal Naming Conventions

The input and output signals of the DDRCTL controller follow the following conventions:

- AXI signals are given the names from the AXI Specification.
- DFI signals are given the names from the DFI Specification.
- All HIF signals have the prefix “hif_”:
 - HIF commands are prefixed “hif_cmd_”.
 - HIF write data signals are prefixed “hif_wdata_”.
 - HIF read data signals are prefixed “hif_rdata_”.
 - HIF MRR data signals are prefixed “hif_mrr_data_”.
 - HIF credit signals are named “hif*_credit”.
 - HIF signals instructing the DDRCTL to go to critical state are prefixed “hif_go2critical_”.
 - The remaining part is a description of the signal.
- The interfaces to the on-chip write data SRAM and read reorder buffer are exceptions to the HIF signal naming conventions. These interfaces use prefixes “wdataram_” and “rdataram_” for both inputs and outputs.

3

System Interfaces

This chapter contains the following sections:

- [“AXI Port Interface”](#) on page 44
- [“APB Interface”](#) on page 58
- [“Host Interface \(HIF\)”](#) on page 59
- [“Hardware Low-Power Interfaces”](#) on page 79

3.1 AXI Port Interface

This section covers the following topics:

- [“Overview of AXI Port Interface”](#) on page 44
- [“Read Address Channel”](#) on page 45
- [“Write Address Channel”](#) on page 46
- [“Wrap Burst”](#) on page 47
- [“Read Data and Response Channel”](#) on page 48
- [“Write Data Channel”](#) on page 53
- [“Data Width Conversion”](#) on page 53
- [“Write Response Channel”](#) on page 54
- [“Exclusive Access”](#) on page 54
- [“Transaction Poisoning”](#) on page 56
- [“Signals Related To AXI Port Interface”](#) on page 56
- [“Registers Related To AXI Port Interface”](#) on page 56

**Note**

Unless explicitly mentioned, features described under this section apply only for AXI configurations.

3.1.1 Overview of AXI Port Interface

The AXI protocol is burst-based with read and write request channels that specify the host ID for the request, start byte address, burst length, burst size, and burst type. This information is processed by the interface and is used subsequently by the DDRCTL.

The AXI port interface (XPI) connects the AXI application port to the DDRCTL and performs the following main functions:

- Read address generation
- Write address generation
- Write data generation
- Read data and response generation
- Write response generation

The XPI converts AXI bursts into read and write requests, which are forwarded to the Port Arbiter (PA). In the opposite direction, the XPI converts the responses from the DDRC into appropriate AXI responses.

The interface between XPI and PA is the same as HIF (but with independent channels for read and write commands). Each AXI port can be independently configured to operate with a clock that is synchronous or asynchronous to the memory controller clock (refer to the hardware parameter [“HW Configuration / Multiport Parameters”](#) on page 347).

The AXI Specification requires that transactions must not cross a 4K address boundary. AXI compliant managers must meet this requirement. However, if a manager does not comply with this AXI protocol requirement, you can relax this restriction in the DDRCTL and set the boundary between 4K and 4G using the hardware parameter `UMCTL2_AXI_ADDR_BOUNDARY`.

3.1.2 Read Address Channel

The AXI read address channel has the following features:

- The read transaction can be of any length up to 256 for incremental bursts, and 16 for wrapping bursts.
- The burst types supported are incremental and wrapping.
- The burst start address can be unaligned to the AXI data width boundaries.
- The size of the burst can be less than the full width of the AXI data bus (also known as sub-sized transfers).

The signals on the read address channel are registered into the XPI in accordance with the AXI valid/ready handshaking protocol and are synchronous to the AXI clock (`ac1k`).

Read requests are accepted if the request can be written into the Read Address Queue (RAQ). This is used to store all the read address requests. If `UMCTL2_XPI_USE2RAQ_n` is set to '0', there is single address queue on a given port. If `UMCTL2_XPI_USE2RAQ_n` is set to '1', there are two address queues on a given port, named Blue and Red address queues. For more information on the usage of dual address queues, see [“Dual Read Address Queue”](#) on page 165. Clock domain crossing from `ac1k` to `DDRCTL` clock (`core_ddrc_core_clk`) is performed in the RAQ block. The depth of the RAQ can be configured to suit your system requirements (refer to the hardware configuration parameter `UMCTL2_AXI_RAQD_n` in [“HW Configuration / AXI Parameters”](#) on page 356).

After registering the read address channel by the RAQ, the following operations are performed:

- Generation of new read requests is based on alignment (derived from AXI address and size), burst lengths (derived from AXI length and memory burst length), and burst type (derived from AXI incremental or wrapping). Each AXI burst is divided into packets of length equal to the memory burst length (BL8, BL16).
- Generation of new HIF address in the case of unaligned burst and burst expansion.

A burst is aligned to the memory burst boundary when:

- $A(R|W)ADDR[2+X:X] = 0$ if BL8 or
- $A(R|W)ADDR[3+X:X] = 0$ if BL16.

Where X is log2 of the number of data bytes in the SDRAM interface.

Therefore:

- If `MEMC_DRAM_DATA_WIDTH = 8`, $X = 0$.
- If `MEMC_DRAM_DATA_WIDTH = 16`, $X = 1$.
- If `MEMC_DRAM_DATA_WIDTH = 32`, $X = 2$, and so on.

In case of an unaligned burst, the first read request is unaligned and the remaining read requests are aligned.

In general, realignment to a memory burst boundary potentially causes some data beats to be discarded (affecting bandwidth) and potentially introduces additional latency on the read data and response channel.

The `arready` output is defaulted to logic one and remains in logic one unless the RAQ becomes full or a WRAP burst is requested (`arready` is low on the next cycle after a WRAP burst is accepted). The read transaction is popped from the RAQ when it is not empty.

The XPI handles the generation of the token which is used by the DDRC for identifying the read command and corresponding data. For more information on how the DDRC handles read commands, see “[Port Timeout](#)” on page 136.

3.1.3 Write Address Channel

The AXI write address channel has the following features:

- The write transaction can be of any length up to 256 for incremental bursts, and 16 for wrapping bursts.
- The burst types supported are incremental and wrapping.
- The burst start address can be unaligned to the AXI data width boundaries.
- The size of the burst can be less than the full width of the AXI data bus (also known as sub-sized transfers).

Write Address Queue (WAQ) is used to store all the addresses for write requests from a given port. There is a single queue for all AXI IDs from a given port. The write address channel behavior is similar to the read address channel.

The depth of the WAQ can be configured to suit your system requirements (see hardware configuration parameter `UMCTL2_AXI_WAQD_n` in “[HW Configuration / AXI Parameters](#)” on page 356). Write address and read address channels are independent and the ordering between the write and read requests may not be preserved.

To preserve the sequence, a higher-level protocol needs to wait for read/write response before sending the next transaction.

Transactions across the ports are independent and can be issued in any order.

3.1.3.1 Read-Modify-Write (RMW) Generation

The read-modify-write (RMW) feature is enabled when ECC is enabled in the hardware (hardware parameter `MEMC_ECC_SUPPORT` is set to greater than 0) and the software (software register `ECCECFG0.ecc_mode` is set to 3b'100), and RMW is enabled in the hardware (hardware parameter `MEMC_USE_RMW` is set). RMW generation hardware (`MEMC_USE_RMW`) is always enabled in AXI configurations.

RMWs are generated for Inline ECC mode (`MEMC_INLINE_ECC = 1`) consideration when:

- The AXI burst start or end address is unaligned with 64 bits, or
- When data strobes are used to mask data bytes falling within 64 bits.

RMWs are generated when DBI, data mask disable or X4 devices when:

- There is insufficient valid write data available from the AXI transaction to write every byte of data for the memory burst.
- The AXI burst start or end address is unaligned with the DDR memory burst address boundaries

When RMW is enabled, the write command is not forwarded to the DDRC until the write data is collected and the strobes evaluated.

3.1.3.2 Read Modify Write (RMW) Bypass Path

XPI_RMW block exists in XPI to handle the generation of HIF Read Modify Write (RMW) commands. XPI_RMW chooses either HIF write or RMW after evaluating write data strobes, ECC enabled/disabled conditions, and register field `DBICTL.dm_en`. This block adds one cycle latency to write command and data paths by default. Read Modify Write (RMW) Bypass Path feature enables a bypass path to avoid the one cycle latency in cases where RMW prediction is not needed.

Read Modify Write (RMW) Bypass Path can be enabled by setting configuration parameter `DDRCTL_XPI_USE_RMWR = 0`. Enabling `DDRCTL_XPI_USE_RMWR`, that is, setting to '1' disables the bypass path.

When the RMW Bypass Path is enabled (`DDRCTL_XPI_USE_RMWR = 0`), there is a parallel bypass path to the RMW generation logic in `DWC_ddr_umctl2_xpi_rmwr` module. When there are no RMW generation conditions enabled, this bypass path is activated and write command and data flows through this bypass path. This saves one cycle latency for write command and data paths. The RMW generation conditions are:

- Inline/Sideband ECC is enabled in hardware and software.
- Data mask is disabled (`DBICTL.dm_en = 0`).

If any of the above conditions are true, RMW generation logic cannot be bypassed and hence the bypass path is disabled.



Note

Enabling the RMW Bypass Path in multi-port configurations will introduce longer timing paths.

3.1.4 Wrap Burst

A WRAP burst may be expanded by the XPI into multiple SDRAM transactions. This is also true for cases where wrap burst size is the same as DRAM burst size.

Table 3-1 MEMC_BURST_LENGTH 16 Wrap Expansion

Direction	Port Data Width	SDRAM Burst Length 16 & Full Burst
Read	Native-Sized	Single
	Up-Sized	Single
	Down-Sized	Single
Write	Native-Sized	Single
	Up-Sized	Single (only if <code>awaddr</code> is aligned to the HIF data width)
	Down-Sized	Single

For information about size conversion definitions, see “[Data Width Conversion](#)” on page 53.

In [Table 3-1](#) full burst is when `DDR_bytes = AXI_bytes`.

Where `DDR_bytes` refers to the number of bytes transferred in a DDR burst:

`MSTR0.burst_rdwr*2*MEMC_DRAM_DATA_WIDTH/8`

`AXI_bytes` refers to the number of bytes transferred in an AXI burst:¹ $(ALEN+1) * (UMCTL2_PORT_DW_n/8)$

Effective HIF Bus width depends on `MSTR0.data_bus_width` setting. When the current effective HIF bus width is larger than the AXI port data width its in up-sized mode. When the current effective HIF bus width is smaller than the AXI port data width its in down-sized mode. When the current effective HIF bus width is equal to AXI port data width its in native-sized mode.

3.1.5 Read Data and Response Channel

The XPI handles the common response interface to the DDRC to process the read data from the memory. For details of the common response interface, see “[Host Interface \(HIF\)](#)” on page 59.

AXI read data and response channel has a single data storage queue, which is called read data queue (RDQ) with two clocks – AXI clock for reading and DDRCTL clock for writing.

Data from different IDs are stored in the same queue and are returned in the order of read address acceptance per AXI ID.

You can configure the depth of the queue independently for each port (see hardware parameter `UMCTL2_AXI_RDQD_n` in “[HW Configuration / AXI Parameters](#)” on page 356).

The FSM handling the AXI handshake as well as the FIFO push and pop has the following features:

- Based on the information stored, filter the invalid beats.
- Based on the signal `hif_rdata_last`, generate the `rlast`.

The controller provides an OKAY response for each read, except for exclusive read transactions. The DDRCTL controller provides an EXOKAY response. For more information about this, see “[Exclusive Access](#)” on page 54.

SLVERR response may be returned for read transactions (both normal and exclusive) in the following cases:

- On-chip parity address or data error (see “[On-Chip Parity \(OCPAR\)](#)” on page 309)
- Invalid LPDDR5/4 row address (see “[Non-binary Device Densities](#)” on page 128)
- Transaction has been poisoned (see “[Transaction Poisoning](#)” on page 56)
- Link ECC uncorrected error detected (see “[Link ECC](#)” on page 261).

Note, that SLVERR has the highest priority.



Note

For sub-sized transfer, when a parity error is detected at the AXI read data interface, SLVERR is going to be asserted for all beats where the failing bytes are presented, even though they are not valid for that specific cycle. `rparity` information is provided at the output along with the data so that specific failing bytes can be distinguished by the AXI Manager.

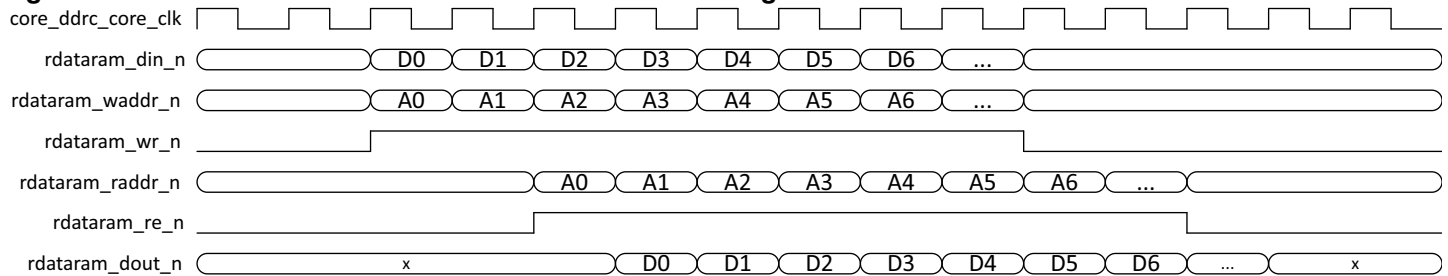
1. Every transfer is considered as full-sized and calculations always take into consideration the maximum size allowed. Sub-sized transfers are always expanded into multiple DRAM transactions.

3.1.5.1 Read Reorder Buffer

The read data can be returned from the DDRC in a different order from which the read commands are forwarded from the XPI. (due to the re-ordering of read commands in the DDRC to maximize SDRAM bandwidth). A read reorder buffer is implemented in each port to reorder the read data for that port to the same order (per AXI ID) as the order of the AXI read commands.

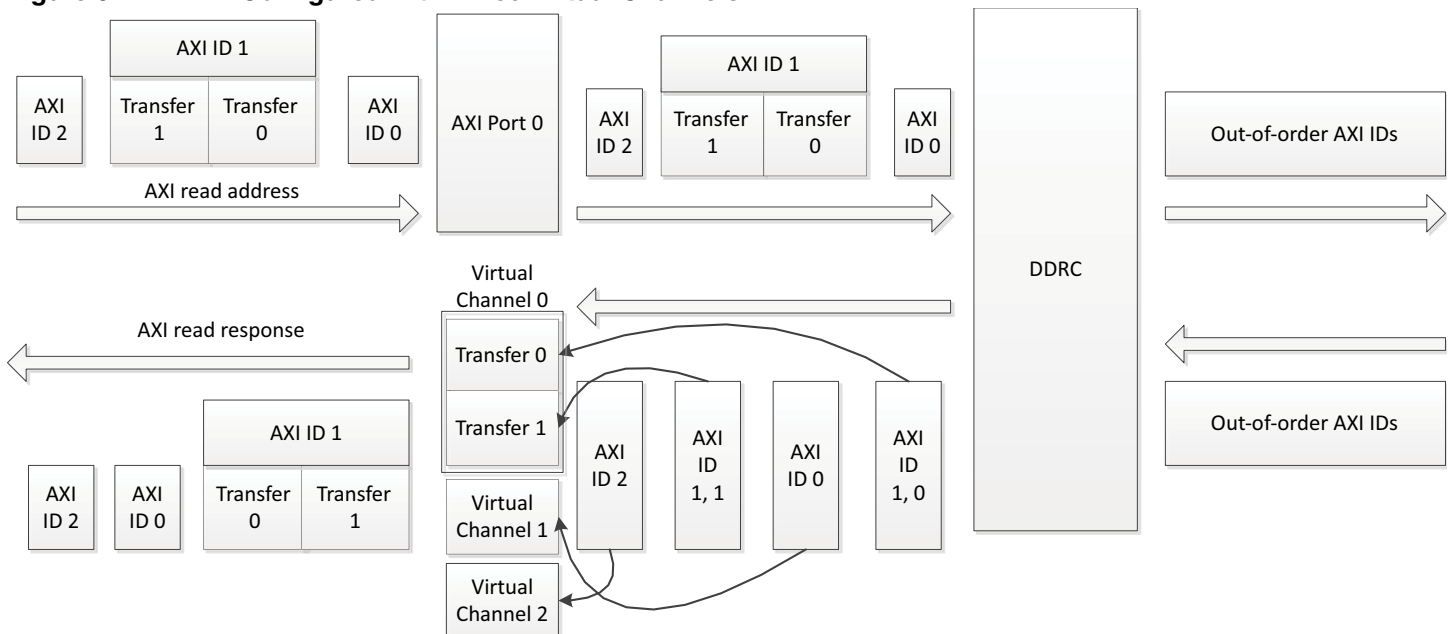
The read reorder buffer SRAM holds the same number of entries as the read CAM and each entry holds the read data corresponding to a DDR command. Storage for the reorder buffer can be implemented internally in the DDRCTL or implemented externally to the DDRCTL as embedded SRAM and accessed through an External RAM Interface. In either case, the control circuits for the read reorder SRAM are clocked by the DDRCTL clock (and not AXI clock). If implemented as embedded SRAM, a two-port SRAM is required.

Figure 3-1 Read Reorder Buffer Data RAM Interface Timing



The AXI protocol allows the read data for transactions of different IDs to be interleaved. To reduce potential delays where read data for one ID is blocked waiting for data associated with another ID, the read reorder buffer is organized as number of virtual channels (see [Figure 3-1](#)). The Read Reorder Virtual Channel is a mechanism to allow independent read data reordering between multiple groups of AXI IDs.

Figure 3-2 RRB Configured with Three Virtual Channels



3.1.5.2 AXI Read Data Interleaving

DDRCTL supports AXI read data interleaving between beats of different IDs. This feature is controlled per port by the parameter `UMCTL2_READ_DATA_INTERLEAVE_EN_$`. The following points have to be considered while enabling this feature:

- To enable read data interleaving for a port:
 - a. `UMCTL2_READ_DATA_INTERLEAVE_EN_$` must be set to '1' for that port.
 - b. `UMCTL2_NUM_VIR_CH_$` must be greater than 1 for that port.
- If enabled, read data interleaving can happen between the beats of `UMCTL2_NUM_VIR_CH_$` number of unique IDs at a time.
- Efficiency of interleaving increases as the number of virtual channels increases.

When Dual Read Address Queue is selected (`UMCTL2_XPI_USE2RAQ_n==1`) and AXI Read Data Interleaving is disabled (`UMCTL2_READ_DATA_INTERLEAVE_EN_n==0`), there is a restriction: the number of RRB virtual channels (`UMCTL2_NUM_VIR_CH_n`) must be equal to the number of CAM entries (`MEMC_NO_OF_ENTRY`). The number of virtual channels is not a configurable parameter in this case, its value defaults to `MEMC_NO_OF_ENTRY`. If AXI Read Data Interleaving is enabled, there is no such restriction.

3.1.5.3 RRB Threshold Based VC Selection Feature for Read Data Interleaving Disabled Configuration

When AXI read data interleaving is disabled, there are two hardware configuration options -- to pop-out read data from RRB:

- `UMCTL2_RRB_THRESHOLD_EN == 0`
In this mode, RRB starts popping the read data when any data is available on a virtual channel (VC) of RRB.
- `UMCTL2_RRB_THRESHOLD_EN == 1`
In this mode, RRB starts popping the read data when any of the following conditions is satisfied:
 - The number of read data bursts available in the virtual channel is more than the value specified by `PCFGR_n.rrb_lock_threshold`.
 - All read data bursts of a given AXI read transaction are available in the virtual channel.

For better performance, it is recommended to turn-on this feature for AXI read data interleaving disabled with multiple VC configurations.



Note

The effective size of HIF burst becomes half and quarter in Half and Quarter Bus Width modes respectively. You need to calculate the value to be programmed into the `PCFGR_n.rrb_lock_threshold` register field based on this effective HIF burst size.

3.1.5.4 Virtual Channels: Dynamic Mapping

The number of virtual channels is determined by the hardware parameter `UMCTL2_NUM_VIR_CH_n` (see "[HW Configuration / AXI Parameters](#)" on page 356), which can be up to 64. By default, the hardware parameter `UMCTL2_STATIC_VIR_CH_n` is set to '0', so that AXI read IDs are assigned dynamically to RRB

virtual channels. Dynamic mapping mode of operation is recommended since it is completely transparent and there are no software registers to program.

In this mode, the distribution of AXI read ID across virtual channels is maximized to reduce the reordering dependencies between different read IDs.

When a new AXI read ID cannot be assigned to a free virtual channel, a round-robin is used to select one of the virtual channels already in use. As a consequence, the read data for the new ID may be blocked waiting for the data associated with the other IDs in the same virtual channel. Increase the number of virtual channels to minimize this effect. To optimize the average latency and minimize unnecessary reordering, MEMC_NO_OF_ENTRY (CAM depth) number of RRB virtual channels are required, in other words, if MEMC_NO_OF_ENTRY is 64 or less, set UMCTL2_NUM_VIR_CH_n = MEMC_NO_OF_ENTRY. If the number of AXI IDs supported for a given port is less than the CAM depth, then UMCTL2_NUM_VIR_CH_n must be set to that smaller number.

The following algorithm can be used to set the number of virtual channels (UMCTL2_NUM_VIR_CH_n) in an optimum way:

```
if (number of AXI IDs <= 64) && (number of AXI IDs <= CAM depth)
    (number of Virtual channels) = (number of AXI IDs); // optimum setting
else if (CAM depth <= 64)
    (number of Virtual channels) = (CAM depth); // optimum setting
else
    (number of Virtual channels) = 64; // non-optimum setting since CAM depth = 64
```

Note that the number of virtual channels can always be set to less than the optimum settings to save area.

3.1.5.5 Virtual Channels: Static Mapping

The dynamic virtual channel mode can be replaced by a static virtual channel mode, when UMCTL2_STATIC_VIR_CH_n = 1. In the static virtual channel mode, AXI read IDs are assigned statically to the virtual channels through registers.

When static virtual channel mode is selected, an extra bypass virtual channel is added and the maximum number of static virtual channels is restricted to 16.

The registers PCFGIDMASKCHn.id_mask and PCFGIDVALUECHn.id_value (see “REGB_ARB_PORTp” in “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide) are programmed to assign AXI transaction IDs to these virtual channels. Within a single virtual channel, data is reordered in the same sequence as the originating read commands for the IDs programmed for that particular channel.

The AXI read ID is assigned to a virtual reorder channel m for a port n if the following two conditions are true:

- (ARID & PCFGIDMASKCHn.id_mask) == (PCFGIDVALUECHn.id_value & PCFGIDMASKCHn.id_mask)
- PCFGIDMASKCHn.id_mask != 0

In the first condition, ‘&’ is a bit wise operator. The first bullet identifies the matching ID through mask and value. The second bullet expresses the condition where a given channel must be enabled. In other words, to enable a channel, its “id_mask” must always be set to a non-zero value.

Table 3-2 shows example virtual channel assignments when $UMCTL2_A_NPORTS = 3$, $UMCTL2_A_IDW = 4$, $UMCTL2_NUM_VIR_CH_0 = 1$, $UMCTL2_NUM_VIR_CH_1 = 2$, $UMCTL2_NUM_VIR_CH_2 = 8$.

Table 3-2 Virtual Channel Registers Programming Example

Port	AXI IDs	Virtual Channel	id_mask Register	id_value Register
0	0	0	PCFGIDMASK0.id_mask = 0xF	PCFGIDVALUECH0_0 = 0x0
1	1,3,5,7	0	PCFGIDMASK0.id_mask = 0x9	PCFGIDVALUECH0_1 = 0x1
	0,2,4,6	1	PCFGIDMASKCH1_1 = 0x9	PCFGIDVALUECH1_1 = 0x0
2	0	0	PCFGIDMASKCH0_2 = 0xF	PCFGIDVALUECH0_2 = 0x0
	1	1	PCFGIDMASKCH1_2 = 0xF	PCFGIDVALUECH1_2 = 0x1
	2	2	PCFGIDMASKCH2_2 = 0xF	PCFGIDVALUECH2_2 = 0x2
	3	3	PCFGIDMASKCH3_2 = 0xF	PCFGIDVALUECH3_2 = 0x3
	4	4	PCFGIDMASKCH4_2 = 0xF	PCFGIDVALUECH4_2 = 0x4
	5	5	PCFGIDMASKCH5_2 = 0xF	PCFGIDVALUECH5_2 = 0x5
	6	6	PCFGIDMASKCH6_2 = 0xF	PCFGIDVALUECH6_2 = 0x6
	7	7	PCFGIDMASKCH7_2 = 0xF	PCFGIDVALUECH7_2 = 0x7

If the AXI read ID verifies the condition for more than one channel, it is always assigned to the channel with the lower channel number. For example, if a given ID matches both VC2 and VC3, it is always assigned to VC2.

If the AXI read ID does not verify the condition for any of the virtual channels, it is deemed to be 'unassigned'.

Two modes of operation are possible for AXI read IDs that are not assigned by the register mapping. These modes are selected by the register `PCFGR.read_reorder_bypass_en` (see "REGB_ARB_PORTp" in "Register Descriptions" chapter of the DWC DDRCTL LPDDR5/4 Programming Guide):

- If `REGB_ARB_PORTp.PCFGR.read_reorder_bypass_en = 0`, the unassigned AXI read IDs are associated to virtual channel 0 (VC0) in addition to all the other IDs that are mapped to VC0.
- If `REGB_ARB_PORTp.PCFGR.read_reorder_bypass_en = 1`, the unassigned AXI read IDs are associated to the bypass path and are not re-ordered.



Note

The AXI read ID is assigned to a channel *m* only when `PCFGIDMASKCHn.id_mask` is not equal to '0'. After controller reset, all traffic is assigned to VC0 as `PCFGIDMASKCHn.id_mask`, `PCFGIDVALUECHn.id_value` and `PCFGR.read_reorder_bypass_en` are initialized to '0'.

The "bypass path" bypasses the read reorder buffer, and the read data for those transactions are not re-ordered back to the order of the originating commands. When using the bypass path, the following restrictions apply to ensure compliance with the AXI ID ordering rules:

- The AXI burst length must correspond to one command on the DDR interface (if an AXI burst is split into more than one DDR command, the data ordering for that AXI burst cannot be guaranteed and leads to a protocol violation).
- Each outstanding AXI read burst must have a unique ID. The same ID is only be reused, when the AXI Manager (`rlast`) accepts the last beat of the read data.
- DDR memory burst (BL) boundary crossing for INCR type bursts is not allowed. In other words, AXI start address (`ARADDR`) and AXI end address (`ARADDR + (ALEN+1) * 2*ARSIZE - 1`) must be contained within the same DDR burst (BL).

For example, if memory data width is 64 bits and BL8, AXI read must be bound within 64 bytes.

An AXI transfer starts at `Start_addr = ARADDR` and ends at `End_addr = ARADDR + (ALEN+1) * 2*ARSIZE - 1`. The burst is bound to DDR burst boundary if `Start_addr[UMCTL2_A_ADDRW-1:6] == End_addr[UMCTL2_A_ADDRW-1:6]`.

- For AXI ports performing data size conversion (`UMCTL2_PORT_DW_n != MEMC_DRAM_DATA_WIDTH * (MEMC_FREQ_RATIO*2)`), WRAP bursts must not cross the DDR BL boundary and must be address aligned with the DDR BL.
- For AXI ports not performing data size conversion (`UMCTL2_PORT_DW_n == MEMC_DRAM_DATA_WIDTH * (MEMC_FREQ_RATIO*2)`), WRAP bursts must be full size (`ARSIZE == LOG2(AXI_DW/8)`) and their length (`ARLEN`) must be less than or equal to `MEMC_BURST_LENGTH / (MEMC_FREQ_RATIO*2) - 1`.

3.1.6 Write Data Channel

The AXI write data channel has a data storage queue, which is called write data queue (WDQ) with the following clocks:

- AXI clock (`acclk_n`) for writing
- DDRCTL clock (`core_ddrc_core_clk`) for reading

This queue is used for clock domain crossing between the AXI clock domain and the controller clock domain and acts as a registering layer in the case of a synchronous interface.

At the output of WDQ, some beats of a write data packet can be masked depending on alignment, burst size and burst length.

Write data from each port is forwarded to the DDRC, which forwards the data to a common data storage. For more information, see [“Write Requests”](#) on page 66.

3.1.7 Data Width Conversion

The XPI performs the data width conversion between the data width at the AXI interface and effective internal HIF data width. The AXI data width can be set by the hardware parameter `UMCTL2_PORT_DW` (see [“HW Configuration / Multiport Parameters”](#) on page 347).

Both data width down-conversion and data width up-conversion are supported.

The value of effective HIF bus width depends on the DRAM data width and `MSTR0.data_bus_width` setting. Based on the current effective HIF bus width, an AXI PORT can be in upsize mode, downsize mode, or native mode. When Dual Channel with data channel interleaving is used (`UMCTL2_DATA_CHANNEL_INTERLEAVE_EN = 1`), multiply effective HIF bus width by 2 to determine Upsize/Native/Downsize mode.

Figure 3-3 PDBW Wdata Conversion

DRAM DW (bits)	FREQ_RATIO	Bus_width	Effective HIF BUS WIDTHS (bits)	AXI Data Widths (bits)				
				32	64	128	256	512
8	1:4	FBW	64	Upsize mode	Native	Downsize mode	Downsize mode	Downsize mode
16	1:4	HBW						
32	1:4	QBW						
16	1:4	FBW	128	Upsize mode	Upsize mode	Native	Downsize mode	Downsize mode
32	1:4	HBW						
64	1:4	QBW						
32	1:4	FBW	256	Upsize mode	Upsize mode	Upsize mode	Native	Downsize mode
64	1:4	HBW						
64	1:4	FBW	512	Upsize mode	Upsize mode	Upsize mode	Upsize mode	Native

The width of the queues, write data queue (WDQ) and read data queue (RDQ), are always set to the wider data width. In case of data width down-conversion, these queues are set to the width of the AXI data bus. In case of data width up-conversion, these queues are set to the width of the internal HIF data-bus width.

For data lane mapping examples, see “[Data Lane Mapping Examples](#)” on page 513.

3.1.8 Write Response Channel

The write response is generated once the last beat of write data, for a given AXI burst, is accepted by the DDRC. The write response queue can be instantiated with configurable depth which is set by the hardware parameter UMCTL2_AXI_WRQD_n (see “[HW Configuration / AXI Parameters](#)” on page 356).

The write response generation makes use of the result of the exclusive access monitor. For a write transaction, the response is always returned as OKAY. For an exclusive write transaction, the response can be returned as OKAY or EXOKAY. For more information about this, see “[Exclusive Access](#)” on page 54.

SLVERR response may be returned in the following cases:

- Invalid LPDDR5/4 row address (see “[Non-binary Device Densities](#)” on page 128)
- On-chip parity address or data error (see “[On-Chip Parity \(OCPAR\)](#)” on page 309)
- Transaction is poisoned (see “[Transaction Poisoning](#)” on page 56)

Note that SLVERR has the highest priority.



Note

A false SLVERR may be reported if parity error is detected for invalid bytes when OCPARCFG0.oc_parity_en = 1 and OCPARCFG0.par_rdata_slvrr_en = 1. This is due to the fact that parity check is performed on the complete data bus without the knowledge of valid bytes in that specific cycle.

3.1.9 Exclusive Access

The DDRCTL controller supports the AXI Exclusive Access feature. This feature is enabled using the hardware parameter UMCTL2_EXCL_ACCESS. This parameter defines the number of addresses the DDRCTL can monitor.

All exclusive read transactions have an EXOKAY response (except in the case of an ECC uncorrectable error). Successful exclusive write accesses have an EXOKAY response, unsuccessful exclusive write accesses return an OKAY response. If an exclusive write fails, the data mask for the exclusive write is forced low so the data is not written. Masked writes or RMW must be supported when using exclusive access. If system does not support RMW (MEMC_USE_RMW = 0), data mask must be enabled (DDBICTL.dm_en = 1), so that the controller can properly mask failing exclusive writes.

One address range per AXI ID per port is monitored for exclusivity. Therefore, if a manager does not complete the write portion of an exclusive operation, a subsequent exclusive read to the same ID changes the address that is being monitored for exclusivity.

Once an exclusive access monitor for a given address is enabled, all write transactions are monitored for violation, regardless of the originating port. In other words, the violation check operates across the ports.

The exclusive access monitor compares the exclusive write transaction address, size, length, ID and port number against the exclusive read transaction address, size, length, ID and port number, and only accepts an exclusive write when these parameters match. Otherwise, the exclusive write is considered as fail.

If the `EXCL_ACCESS` parameter is configured to support no (0) exclusive accesses, the DDRCTL behaves like an AXI subordinate that does not support exclusive accesses. Therefore, all exclusive accesses return OKAY responses.

In some cases, SLVERR response may be returned for exclusive access transactions. For more information about this, see [“Read Data and Response Channel”](#) on page 48 and [“Write Response Channel”](#) on page 54.



Note

- The read response SLVERR is generated for an exclusive read transaction for ECC configurations, when ECC is enabled and an uncorrectable error is detected. The AXI Manager, upon receiving a SLVERR to an exclusive read command, must not complete the exclusive operation by sending an exclusive write command. If it does, the DDRCTL monitors return EXOKAY if there is no violation.
- The maximum number of bytes that are monitored as a region per address cannot exceed 128 bytes. The AXI exclusive transaction length must always be a power of two.
- When all the monitors are active, further exclusive read must not be issued. If this happens, one of the active monitors is evicted using a round robin control and new exclusive read is accepted. The selection of the monitor to be evicted is based on a pointer which rotates across all the locations when a new exclusive read is received.

3.1.9.1 Software Coherency for AXI Ports

[“Address Collision Handling”](#) on page 159 describes how the DDRC handles in-order execution of commands to the same address.

For the commands issued at the AXI port, the logic in the DDRC protects against all types of software coherency hazards when the AXI Manager waits for write (read) response before sending the next same address read (same address write) or vice versa.

Some non-native AXI Managers expect an ordering relationship between the reads and writes when the opposite direction transactions are sent without waiting for the previous response. For these special cases, additional logic must be instantiated in the XPI to enforce the order of acceptance within the reads and writes between the AXI and HIF interfaces at the DDRC. This logic is included by setting of the hardware parameter `UMCTL2_RDWR_ORDERED_n` (see [“HW Configuration / Multiport Parameters”](#) on page 347) and enabled by the register `REGB_ARB_PORTp.PCFGR.rdwr_ordered_en` (see [“REGB_ARB_PORTp”](#) in [“Register Descriptions”](#) chapter of the DWC DDRCTL LPDDR5/4 Programming Guide). When this feature is enabled, AXI port ensures that opposite direction transactions (a read and a write) to the same address are executed on the DFI interface in the same order as they were received on the AXI interface. Transactions to different addresses will still be executed in out of order as per the scheduling. This is ensured by passing the Read and Write commands from System interface to HIF in the same order as they are received irrespective of their addresses. DDRC ensures the order of same address transactions once it is received on the HIF interface. When this feature is enabled and read and write transactions are presented at the same cycle, the pre-arbiter gives precedence to read transactions, incrementing the write transaction order token by 1 with

respect to the read transaction order token. For example, if the last order token issued is equal to '0', then the read order token is equal to '1' and the write order token is equal to '2'.

3.1.10 Transaction Poisoning

Sideband signals `arpoison/awpoison`, render an AXI transaction (read or write) invalid and the DDRCTL signals that AXI transaction poisoning has occurred. The input signal must be asserted coinciding with the associated AXI transaction. If a write is poisoned, all of its strobes are de-asserted, making the write effectively transparent to the memory. If a read is poisoned, the command is issued to the DDR memory and all the read data beats are overwritten and returned as zeros.

The DDRCTL controller may be programmed to signal that an AXI transaction poisoning has occurred by setting an interrupt or by driving the AXI SLVERR response for that transaction response. The AXI Poison Configuration Register `POISONCFG` is used to define whether an interrupt and/or AXI SLVERR is generated when an AXI transaction poisoning has occurred. There is separate control for read and write transactions. The port specific AXI Poison status register is asserted whenever an AXI transaction is poisoned on that port. There is a separate status for read and write transactions. The interrupt and status register is cleared by the register

`POISONCFG.rd_poison_intr_clr/POISONCFG.wr_poison_intr_clr`. Interrupt status is stored in the AXI Poison Status register `POISONSTAT`. This register is a mirror in the APB clock domain of the actual interrupts, so status is updated with 2 or 3 `pclk` cycles of delay depending on the CDC logic.

An external block is expected to drive these new sideband signals depending on the access security requirements on transaction by transaction basis. Therefore, this feature replaces the need to implement ARM Trustzone.

3.1.11 Signals Related To AXI Port Interface

The following are the signals related to the AXI Port Interface:

- AXI Port n Global Signals (for $n = 0; n \leq \text{UMCTL2_A_NPORTS}-1$)
- AXI Port n Write Address Channel (for $n = 0; n \leq \text{UMCTL2_A_NPORTS}-1$) Signals
- AXI Port n Write Address On-Chip Parity Signals (for $n = 0; n \leq \text{UMCTL2_A_NPORTS}-1$)
- AXI Port n Write Data Channel (for $n = 0; n \leq \text{UMCTL2_A_NPORTS}-1$) Signals
- AXI Port n Write Data On-Chip Parity Signals (for $n = 0; n \leq \text{UMCTL2_A_NPORTS}-1$)
- AXI Port n Write Response Channel (for $n = 0; n \leq \text{UMCTL2_A_NPORTS}-1$) Signals
- AXI Port n Read Address Channel (for $n = 0; n \leq \text{UMCTL2_A_NPORTS}-1$) Signals
- AXI Port n Read Address On-Chip Parity Signals (for $n = 0; n \leq \text{UMCTL2_A_NPORTS}-1$)
- AXI Port n Read Data Channel (for $n = 0; n \leq \text{UMCTL2_A_NPORTS}-1$) Signals
- AXI Port n Read Data On-Chip Parity Signals (for $n = 0; n \leq \text{UMCTL2_A_NPORTS}-1$)

For more information about these signals, see [“Signal Descriptions”](#) on page 373.

3.1.12 Registers Related To AXI Port Interface

The following are the registers related to the AXI Port Interface:

- `POISONCFG`
- `POISONSTAT`

For more information about these registers, see “Register Descriptions” chapter in DWC DDRCTL LPDDR5/4 Programming Guide.

3.2 APB Interface

This section contains the following sections:

- [“Overview of APB Interface”](#) on page 58
- [“Register Classes”](#) on page 58
- [“Signals Related to APB Interface”](#) on page 58

3.2.1 Overview of APB Interface

The DDRCTL provides a dedicated APB 3.0 bus interface that is used to access the DDRCTL software programmable registers. The DDRCTL converts APB reads and writes into accesses to the internal register file. The APB data width is fixed to 32 bits for compatibility reasons with the Synopsys DWC DDR PHYs. The APB address width is 12 bits.

All APB interface signals (p*) are synchronous to pclk. pclk can be asynchronous to the core_ddrc_core_clk if the configuration parameter UMCTL2_P_ASYNC_EN is set. However, pclk frequency must be the same or less than the core_ddrc_core_clk frequency.

3.2.2 Register Classes

[Table 3-3](#) describes three classes of registers in the DDRCTL:

Table 3-3 Register Classes

Register Type	Description
Dynamic	Can be written at any time during operation.
Quasi Dynamic	Can be written when the controller is in reset and some specific conditions outside reset. To write them, SWCTL.sw_done has to be set to '0' at the beginning of the programming sequence and set back to '1' at the end. After that, SWSTAT.sw_done_ack has to be polled for acknowledge. For more information about SWCTL.sw_done and SWSTAT.sw_done_ack, refer to the DWC DDRCTL LPDDR5/4 Programming Guide.
Static	Can be written only when the controller is in reset.

For more information about dynamic and quasi dynamic registers, see the “Dynamic Registers” section in the DWC DDRCTL LPDDR5/4 Programming Guide.

3.2.3 Signals Related to APB Interface

For more information about signals related to the APB Interface, see “APB Subordinate Interface Signals” in the chapter [“Signal Descriptions”](#) on page 373.

3.3 Host Interface (HIF)

This topic contains the following sections:

- [“Overview of Host Interface \(HIF\)”](#) on page 59
- [“Dual HIF Functionality”](#) on page 60
- [“Credit Mechanism”](#) on page 61
- [“Valid and Stall”](#) on page 63
- [“Reads”](#) on page 63
- [“Writes”](#) on page 66
- [“Read-Modify-Write Requests”](#) on page 74
- [“WR Versus RMW for HIF Configurations With DBICTL.dm_en=0”](#) on page 75
- [“Half/Quarter Bus Width Mode”](#) on page 76
- [“HIF Data Alignment when MEMC_DRAM_DATA_WIDTH is Non-Power of 2”](#) on page 76
- [“hif_cmd_latency”](#) on page 77
- [“hif_go2critical”](#) on page 78
- [“hif_cmd_autopre”](#) on page 78
- [“hif_mrr_data_valid and hif_mrr_data”](#) on page 78
- [“Additional Error Signal Outputs”](#) on page 78

3.3.1 Overview of Host Interface (HIF)

The HIF is an internal interface in AXI configurations but is available as a HIF system interface when the hardware parameter `DDRCTL_SYS_INTF` is set to '0'. In this case there is no AXI system interfaces.

This section must be viewed when the hardware parameter `DDRCTL_SYS_INTF` is set to '0' with customer logic required to interface to this HIF system interface. However, for information, the HIF signals still exist as internal only in designs with AXI system interfaces, and signal descriptions and functionality remain the same.

The signals are defined in [“HIF Read Command Interface Signals”](#) on page 410, [“HIF Write Command Interface Signals”](#) on page 416, [“HIF Command Interface Signals”](#) on page 422, [“HIF Write Data Interface Signals”](#) on page 436, [“HIF Read Data Interface Signals”](#) on page 439.

The HIF system interface can be configured as Dual HIF to provide concurrent write (including RMW) and read interfaces, which is only enabled when `UMCTL2_DUAL_HIF` set to '1'. Those signals are defined in [“HIF Read Command Interface Signals”](#) on page 410, [“HIF Write Command Interface Signals”](#) on page 416 replacing [“HIF Command Interface Signals”](#) on page 422. This feature converts HIF single command channel into separate/dual HIF command channels for read and write commands.

- RMW commands are performed on the Write HIF command channel.
- Read/Write port arbitration does not occur as there are separate Read and Write command channels.
- Enabling this logic improves the SDRAM utilization, depending on your traffic profile. However, it increases the overall area due to additional logic.

Following the receipt of a request and acceptance (`hif_cmd_valid == '1' && hif_cmd_stall == '0'`) of a write or RMW request, the `DDRCTL` requests write data from the interface. The interface

subsequently provides the write data in the requested order (matches the order in which requests are made to the DDRCTL). For more information, see [“Writes”](#) on page 66.

When `UMCTL2_DUAL_HIF = 1`, reads are on one command bus and writes/RMWs are on another command bus. For more information, see [“Dual HIF Functionality”](#) on page 60.

When `UMCTL2_DUAL_HIF = 0`, read, write, and RMW requests are provided through a common command bus. Both read and write requests can be reordered later, before being output on the DFI interface. Requests to the DDRCTL are throttled in two ways:

- A credit mechanism which ensures that buffer space is available for any request the SoC makes to the DDRCTL prior to the request being made. For more information, see [“Credit Mechanism”](#) on page 61.
- An independent stall mechanism that throttles requests when address collisions take place or if entering self-refresh through the software self-refresh or hardware low-power self-refresh is provided by the output signal `hif_cmd_stall`. HIF commands are not accepted starting from the first clock edge after the stall goes high until the clock edge on which stall is detected low. Note that `hif_cmd_stall` cannot be used instead of credit mechanism, that is, `hif_cmd_stall` will not be asserted even if CAM is full.

Data is handled separately. The DDRCTL throttles write data by potentially waiting after it receives a write or RMW request before requesting the associated data. If there is collision, then the throttling lasts until the collision is cleared in the DDRCTL. Once write data is requested, it is always accepted by the DDRCTL. However, if RMW requests are present, the write data can be throttled using the `hif_wdata_stall` signal. For more information, see [“Write Data”](#) on page 69.

Read data is sent to the host through a read response interface. For more information, see [“Read Response Interface”](#) on page 65.

When `UMCTL2_DUAL_CHANNEL` is true for dual channel configurations, there are two sets of signals with second channel signals suffixed with `'_dch1'`.

3.3.2 Dual HIF Functionality

Dual HIF functionality (selected when `UMCTL2_DUAL_HIF = 1`) is as follows:

- Replaces the single (shared) HIF command channel with two separate HIF command channels, one for Read Commands and one for Write commands. RMW commands are performed on the Write HIF command channel.
- Dual HIF functionality continues to guarantee coherency for write-after-read (WAR) and read-after-write (RAW) hazards. Also, if a Read and Write/RMW to the same address are taken at the same time (both `hif_rcmd_stall = 0` and `hif_wcmd_stall = 0`), the Write/RMW is performed first.

[Table 3-4](#) provides the details of the signals modified by the `UMCTL2_DUAL_HIF` parameter.

Table 3-4 Comparison of Signals Modified by `UMCTL2_DUAL_HIF` Parameter

UMCTL2_DUAL_HIF = 0 Single HIF Command Channel	UMCTL2_DUAL_HIF = 1 Separate HIF Command Channels One for Reads (*_rd) One for Writes/RMW (*_wr)
<code>hif_cmd_valid</code>	<code>hif_rcmd_valid</code> <code>hif_wcmd_valid</code>

UMCTL2_DUAL_HIF = 0 Single HIF Command Channel	UMCTL2_DUAL_HIF = 1 Separate HIF Command Channels One for Reads (*_rd) One for Writes/RMW (*_wr)
hif_cmd_type	hif_rcmd_type hif_wcmd_type
hif_cmd_addr	hif_rcmd_addr hif_wcmd_addr
hif_cmd_pri	hif_rcmd_pri hif_wcmd_pri
hif_cmd_token	hif_rcmd_token hif_wcmd_token
hif_cmd_length	hif_rcmd_length hif_wcmd_length
hif_cmd_wdata_ptr	hif_rcmd_wdata_ptr hif_wcmd_wdata_ptr
hif_cmd_autopre	hif_rcmd_autopre hif_wcmd_autopre
hif_cmd_stall	hif_rcmd_stall hif_wcmd_stall
hif_cmd_latency	hif_rcmd_latency hif_wcmd_latency

**Note**

The databook uses UMCTL2_DUAL_HIF = 0 notation.

If you are using UMCTL2_DUAL_HIF = 1, the internal signals must be interpreted in the following way:

- hif_cmd_valid refers to hif_rcmd_valid and/or hif_wcmd_valid.
- hif_cmd_stall refers to hif_rcmd_stall and/or hif_wcmd_stall, and so on.

3.3.3 Credit Mechanism

The DDRCTL employs a credit mechanism to control the request and data flow thus ensuring that buffers do not overflow. The interface making the request to the DDRCTL, can only request commands for which it has been granted “credits”.

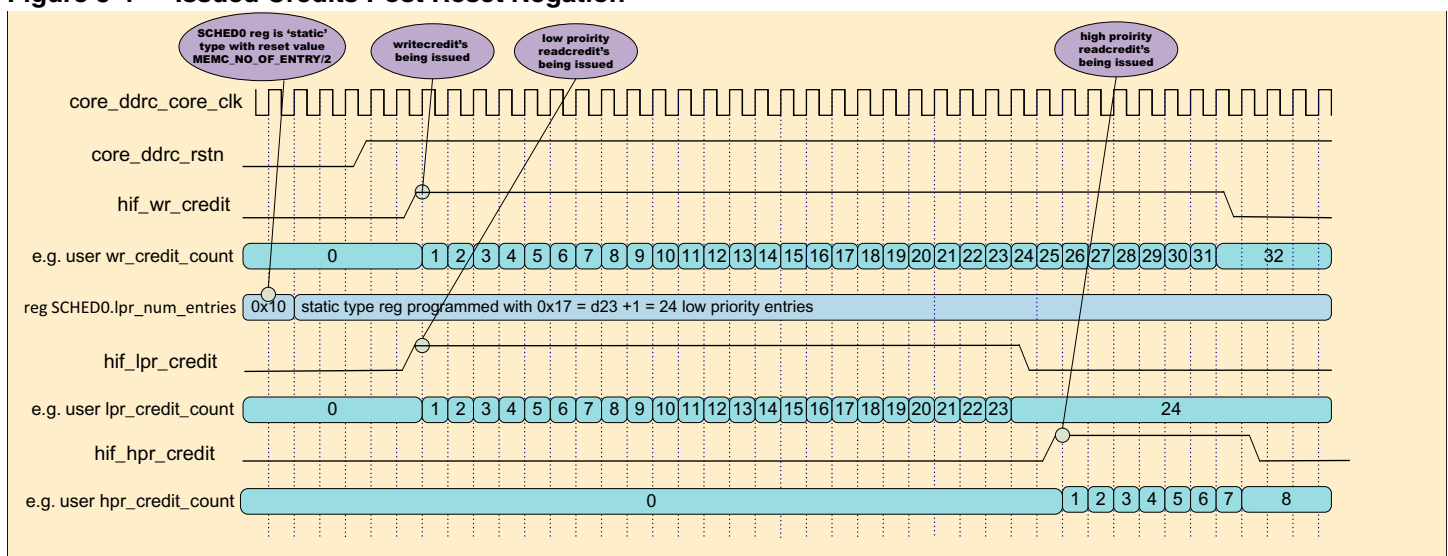
Credits are tracked separately for the following three command types:

- High Priority Read (HPR)
- Low Priority Read/Variable Priority Read (LPR/VPR)
- Write

Credits are counted for each command type independently according to the following rules:

- Initially the interface has zero credits.
- The interface logic must include counters to track the number of credits granted by the DDRCTL and decremented due to commands issued by the interface logic to the DDRCTL. The interface logic must have separate credit counters for each individual command type. The maximum number of LPR credits is configured by `SCHED0.lpr_num_entries` register out of the total number of entries (depth of CAM) defined by the parameter `MEMC_NO_OF_ENTRY`. The number of CAM slots available for LPR/VPW is indicated by `SCHED0.lpr_num_entries + 1`. While the number of slots available for HPR is indicated by:
 - $\text{MEMC_NO_OF_ENTRY} - (\text{SCHED0.lpr_num_entries} + 1)$.
 - This means that the maximum value for the signal `lpr_credit_cnt` is always `SCHED0.lpr_num_entries + 1`.
- Following the de-assertion of reset to the DDRCTL, credits are issued to the interface for each command type. A given credit count increments every time the DDRCTL issues a credit, indicated by the assertion of the appropriate `*_credit` signal on the rising edge of `core_ddrc_core_clk`. The credit counting logic implemented externally to the DDRCTL must run on the same clock.
- When the credit count is greater than zero, the interface can issue requests of that type to the DDRCTL. Each time a request is issued to the DDRCTL, the associated credit count is decremented. An RMW command decrements both the LPR and write credit counters by one.
- When `UMCTL2_VPRW_EN` is set to '1', the arbiter or the host can send Variable Priority Read (VPR) and Variable Priority Write (VPW) commands.
- VPR commands do not have a pre-allocated storage resource in the Read CAM (unlike HPR commands). VPR commands share the same resource with the LPR commands. As far as the credit mechanism is concerned, the VPR commands are counted in the LPR credit bucket.
- Similarly, VPW commands do not have a pre-allocated storage resource in the Write CAM. VPW commands share the same resource with the Normal Write commands. As far as credit mechanism is concerned, the VPW commands are counted in the WR credit bucket.
- For reads, the `*pr_credit` is issued when that request has been sent on the DFI interface.
- For writes, the `*pw_credit` is issued when that request and data has been sent on the DFI interface.

Figure 3-4 Issued Credits Post Reset Negation



**Note**

Ensure that if the DDRCTL is reset via `core_ddrc_rstn`, the customer logic credit counters are also zero. If this is not ensured, then the customer logic credit counts may be misaligned with the available credits causing lost data/fatal errors.

3.3.4 Valid and Stall

A request is indicated by the assertion of `hif_cmd_valid` but is only accepted when `hif_cmd_stall == '0'`. The DDRCTL can throttle the requests by asserting `hif_cmd_stall`. For any cycle in which `hif_cmd_stall` is asserted on the rising edge of the clock, the `hif_cmd_valid` and all other associated request signals are ignored by the DDRCTL. When this happens, the interface must not yet decrement the associated credit counter. When the DDRCTL de-asserts `hif_cmd_stall` on the rising edge of the clock, the request is accepted and the interface can then de-assert `hif_cmd_valid`, decrement the requester credit counter and issue the next read, write or RMW request (credits allowing).

The DDRCTL asserts `hif_cmd_stall` in any of the following conditions:

1. Software driven self-refresh entry requests. This is to ensure that DDRCTL is empty when memories are in self-refresh. For more information, see “[Low-Power and Power-Saving Features](#)” on page 217.
2. Successful DDRCTL hardware low-power entry requests to self-refresh. This is to ensure that the DDRCTL is empty when memories are in self-refresh. For more information, see “[Low-Power and Power-Saving Features](#)” on page 217.
3. Address collisions in the DDRCTL, where flow control is applied to prevent further commands from entering the DDRCTL. For more information, see “[Address Collision Handling](#)” on page 159.
4. In case when Write CAM is full and no corresponding data for any of the write commands has arrived.
5. When setting `OPCTRL1.dis_hif = 1`, which causes `hif_cmd_stall` to be asserted.

**Note**

If Dual HIF functionality is enabled (`UMCTL2_DUAL_HIF = 1`), `hif_cmd_stall/hif_wcmd_stall` is asserted as follows:

- Conditions (1), (2), (3), and (5) cause both `hif_rcmd_stall` and `hif_wcmd_stall` to assert at the same time.
- Condition (4) only affects `hif_wcmd_stall`.

3.3.5 Reads

3.3.5.1 Read Requests

When the LPR or HPR credit count is greater than zero, the customer logic can issue read requests to the DDRCTL accordingly via this HIF system interface.

A command is issued to the DDRCTL by asserting `hif_cmd_valid` on the rising edge of the clock. All read request fields must be driven to the appropriate values at the same time. These fields are:

- `hif_cmd_type`: specifies the request type as follows:
 - '00' indicates a write request.
 - '01' indicates a read request.

- '10' indicates an RMW request.
- '11' is not supported.
- `hif_cmd_pri`:
 - In HIF-only configurations () where `UMCTL2_VPRW_EN` is set to '0', this signal is 1-bit wide. A '1' indicates the read request is high priority. '0' indicates the read request is low priority. This field is meaningless for write commands.
 - In configurations where `UMCTL2_VPRW_EN` is set to '1', this signal is 2-bit wide.
The encoding for Read is:
 - 2'b00 - LPR (Low Priority Read)
 - 2'b01 - VPR (Variable Priority Read)
 - 2'b10 - HPR (High Priority Read)
 - 2'b11 - Reserved
 VPR commands are sent only in configurations with `UMCTL2_VPRW_EN = 1`.
- `hif_cmd_addr`: indicates the address for the read. For more information, see [“HIF Address to SDRAM Address Mapping”](#) on page 124.
- `hif_cmd_length` is 2 bits.
Note the following:
 - For `MEMC_BURST_LENGTH = 16`, a normal read is 16 SDRAM words.
If `MEMC_BURST_LENGTH = 16`:
 - 2'b10 - Partial (Half) Read
 - 2'b00 - Full Read
 In each case, a partial (half) read is half the length of a normal full read (where a word is the amount of data transferred on one edge of the DQS to the SDRAM in a fully configured system). This field is meaningless for write or RMW commands.
- `hif_cmd_token`: a bit field that is presented with the read command and is returned with the read response. As the read responses may be presented out-of-order, the token is the identifier that indicates the read for which data is being returned on the response side. Therefore, multiple reads with the same token must never be pending in the DDRCTL simultaneously, as the responses would be indecipherable. This field is meaningless for write or RMW commands. It is sized by the parameter `MEMC_TAGBITS`. The `hif_cmd_token` signal is carried internally as a sideband and returned with the corresponding read response `hif_rdata_*` signal group, but otherwise has no internal use.

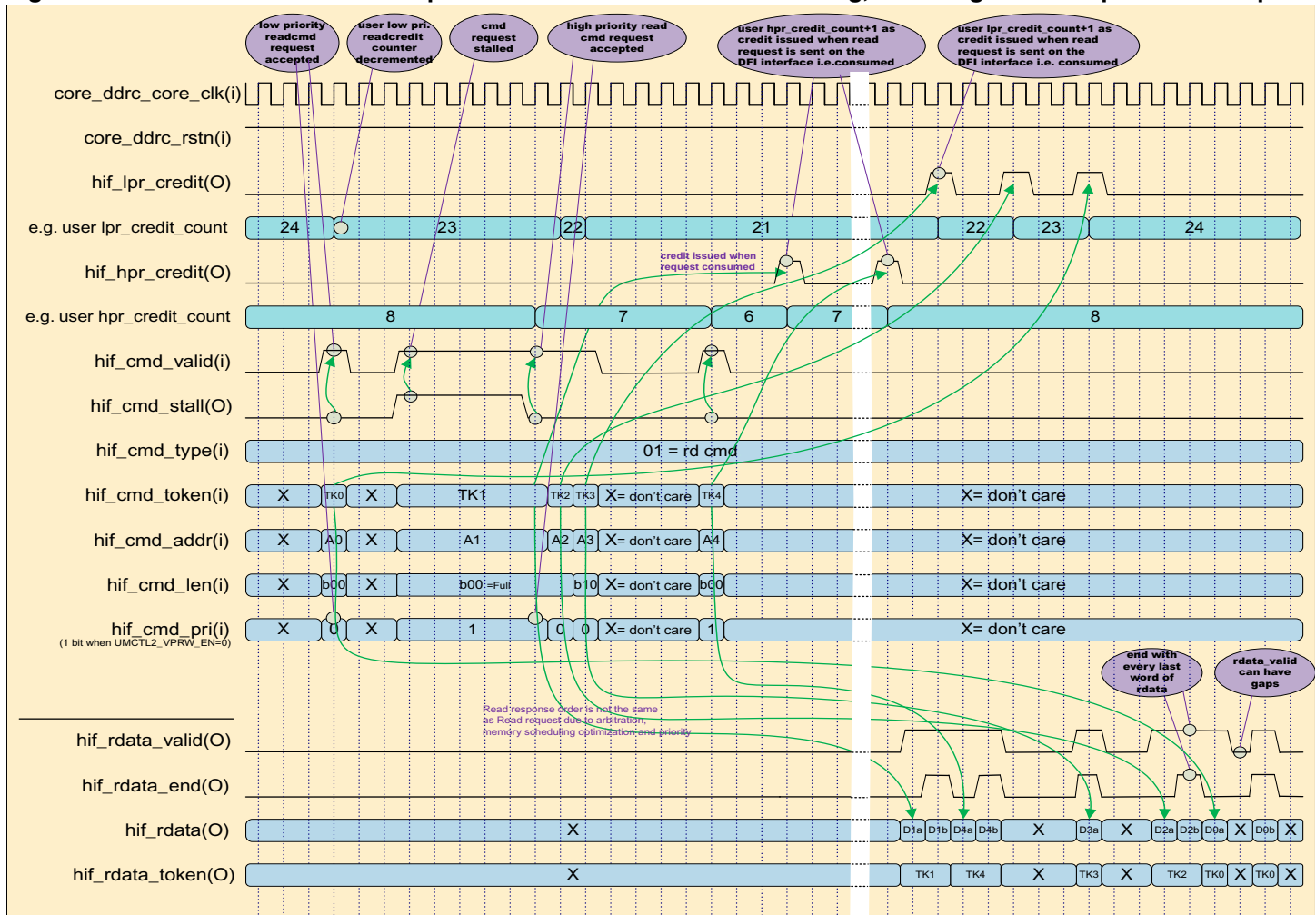
**Note**

`hif_cmd_token` must be a customer logic supplied unique number specific for their application with a minimum number of values of `MEMC_NO_OF_ENTRY`. As reads may be returned out of order, a read reorder buffer is required in the customer logic to provide ordered read responses with the correct requester ID.

Figure 3-5 on page 65 shows a series of low and high priority reads and demonstrates the reordering of the read response data due to memory scheduling optimization including priority.

The matching credit will be issued as soon as the read request command is selected and issued to the DFI interface (that is, “consumed” from a CAM perspective). The delay in the data is the read latency through the PHY to the actual DDR back through the PHY to `core_ddrc_core_clk` domain to the HIF interface.

Figure 3-5 LPR and HPR Read Requests with Stalls and Credit Counting, Including Read Responses Example



3.3.5.2 hif_rdata_addr_err

`hif_rdata_addr_err` is asserted when the requested address for a read is not valid. Invalid address can be caused by using `ADDRMAP12.nonbinary_device_density`. In these cases, DDRCTL performs an aliased read by changing the physical address to a valid one (row [MSB:MSB-1] changes from 2'b11 to 2'b10). The returned data is masked to '0' and `hif_rdata_addr_err` is asserted along with the relevant `hif_rdata_valid/hif_rdata_token`. For more information, see “[Non-binary Device Densities](#)” on page 128.

3.3.5.3 Read Response Interface

Figure 3-5 on page 65 shows multiple read requests and read responses.

The response token indicates which read data is being provided. Once started, read data for a given token always completes for that token before the next token's read data is returned. The `hif_rdata_valid` signal indicates valid data is on the bus and the `hif_rdata_end` signal indicates that the last data packet is being transmitted. The `hif_rdata_valid` can de-assert in the middle of returning read data for one or more cycles, so `hif_rdata_valid` must be monitored to validate each cycle of returning read data.

There is no stall supported on the read response data phase so the user logic must use every `hif_rdata`, `hif_rdata_end`, and `hif_rdata_token` when the `hif_rdata_valid` is asserted.

3.3.5.4 Response Data Ordering

Read data is returned on the HIF in the same order in which it is received from the PHY on the DFI interface. It is independent of whether the read address is aligned or unaligned. For information on how unaligned addresses are handled, see [“Sequential Operations”](#) on page 170. For data lane mapping examples where the addresses are aligned, see [“Data Lane Mapping Examples”](#) on page 513.

3.3.6 Writes

3.3.6.1 Write Requests

When the user Write credit count is greater than zero, the interface can issue write requests to the DDRCTL accordingly. A command is issued to the DDRCTL by asserting `hif_cmd_valid` on the rising edge of the clock.

All write request fields must be driven to the appropriate values at the same time. These fields are:

- `hif_cmd_type`: specifies the request type as follows:
 - '00' Write request.
 - '01' Read request.
 - '10' RMW request.
 - '11' Reserved.
- `hif_cmd_pri`: specifies the priority of the write request:
 - When `UMCTL2_VPRW_EN` is set to '0', this signal is 1-bit wide. This field is meaningless for write commands.
 - When `UMCTL2_VPRW_EN` is set to '1', this signal is 2-bit wide.
 The encoding for Write is:
 - 2'b00 - NPW (Normal Priority Writes)
 - 2'b01 - VPW (Variable Priority Writes)
 - 2'b10 - Reserved
 - 2'b11 - Reserved
 VPW commands are sent only in configurations where `UMCTL2_VPRW_EN` = 1.
- `hif_cmd_addr`: indicates the address for the write. For more information, see [“HIF Address to SDRAM Address Mapping”](#) on page 124.

- `hif_cmd_wdata_ptr`: a pointer into the requester own buffers that can be used to subsequently retrieve the write data. Once a write or RMW request is accepted, this pointer is returned to the interface to retrieve the associated write data. (This field is not required, as requests are always accepted in order; the interface can choose to FIFO these pointers internally and ignore this field).

The stall mechanism for write or RMW requests is identical to that for read requests as explained in the section “[Valid and Stall](#)” on page 63.

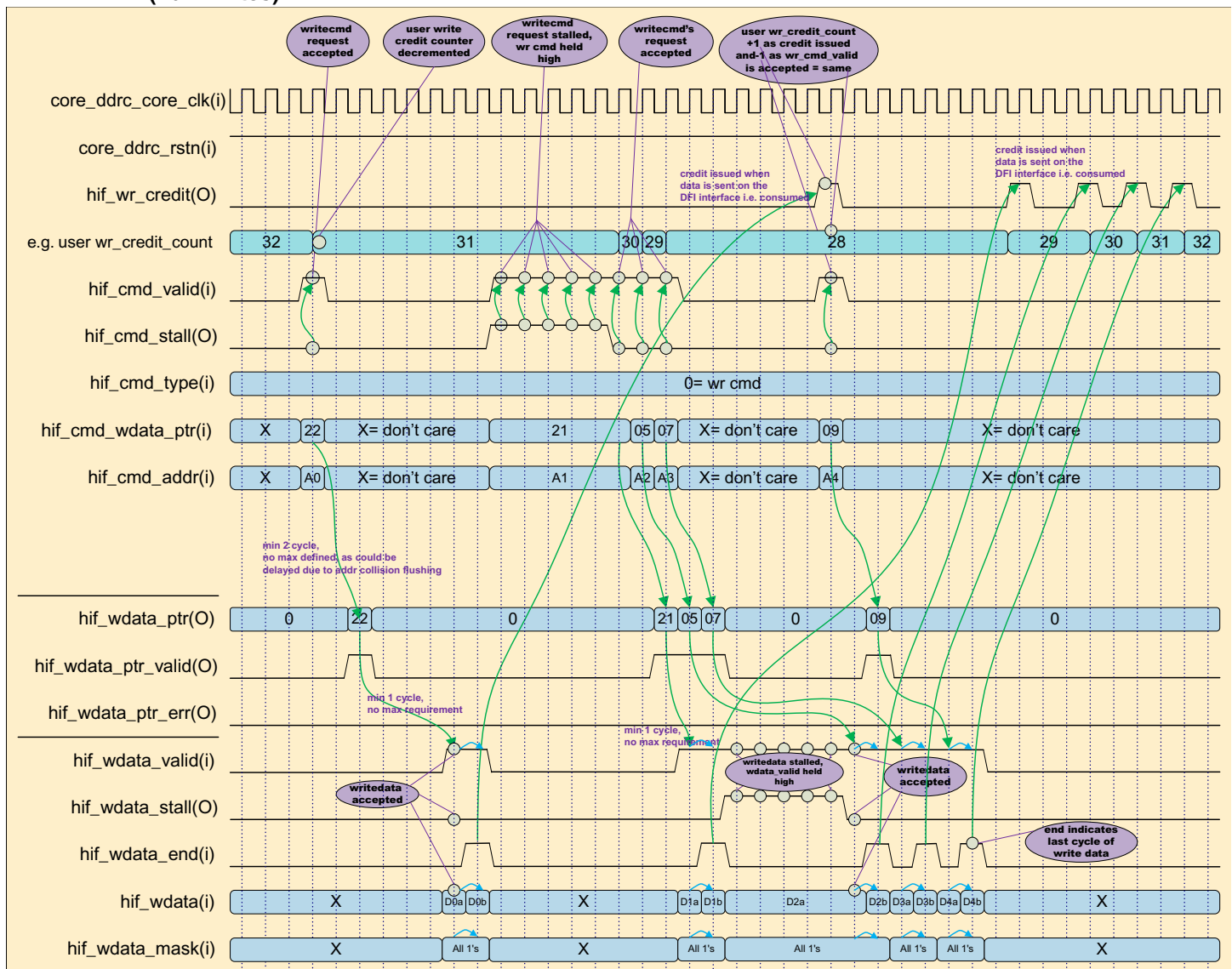
Writes have three phases of operation: a write command, a write data pointer return, and finally write data provided to the DDRCTL. The write command phase indicates to the DDRCTL that a write transfer is required. The write data pointer return phase indicates to the SoC that the DDRCTL is ready to receive the write data. The write data phase provides the write data to the DDRCTL.

In all these three phases, the transactions must follow the same order. The data pointer is returned in the same order in which the command is sent. The write data must also be sent in the same order. The DDRCTL can perform re-ordering to the commands later, before sending them on the DFI interface, to make efficient use of the SDRAM bandwidth.

[Figure 3-6](#) on page 68 is an example of 5 writes, with 2 stalls, of which one write cmd request and one wdata phase is stalled, each with the behavior for `MEMC_BURST_LENGTH = 16`, `FREQ_RATIO 1:4` configurations, where there are two cycles of data flagged by `hif_wdata_valid` with the `hif_wdata_end` asserted in the second cycle.

The credit is issued when that request has been sent on the DFI interface.

Figure 3-6 Write Request and Data - One cmd Stall, One wdata Stall, Each wdata Phase with Two Cycles of Data (Full Writes)



3.3.6.2 Write Data Pointer Return

Following the acceptance of a write or RMW request, the DDRCTL then returns the write data pointer to the interface logic to retrieve the associated write data. This is done by asserting `hif_wdata_ptr_valid` on the rising edge of clock. In the same cycle, `hif_wdata_ptr` indicates the value of the write data pointer. This is same as a pointer that is presented to the DDRCTL as `hif_cmd_wdata_ptr` during the write command phase.

There is no mechanism for stalling the acceptance of the write data pointer from the DDRCTL; the interface logic must present write data for every cycle in which `hif_wdata_ptr_valid` is asserted by the DDRCTL. The mechanism to throttle `hif_wdata_ptr_valid` assertions is for the interface to throttle write requests presented to the DDRCTL.

There is no required timing between a write or RMW command presented to the DDRCTL, and the same pointer is returned as `hif_wdata_ptr`. Multiple pointers can also be presented to the DDRCTL with multiple requests before the first is returned to the interface. The order of the pointers returned matches the order of the requests presented to the DDRCTL.

The use of the output signal `hif_wdata_ptr` returned write pointer is optional for the customer logic as the order of the write data must be in the same order as the write commands.

The next `hif_wdata` must not be sent until the next `hif_wdata_ptr_valid` has been received, indicating that the DDRCTL is ready to receive the next `hif_wdata`, and accept that `wdata`, subject to `hif_wdata_stall`.

3.3.6.3 `hif_wdata_ptr_addr_err`

There is an extra output for DDRCTL, named `hif_wdata_ptr_addr_err`. This is asserted only when the requested address for a write or RMW is not valid. Invalid address can be caused by using `ADDRMAP12.nonbinary_device_density` or LUT based heterogeneous table. When this happens, `hif_wdata_ptr_addr_err` is asserted along with the relevant pair of `hif_wdata_ptr_valid/hif_wdata_ptr`. In these cases, DDRCTL expects to receive write data, but discards them along with the relevant write or RMW request. For more information, see “[Non-binary Device Densities](#)” on page 128.

3.3.6.4 Write Data

After a write data pointer is returned to the interface logic, the interface logic must retrieve the associated data and present it to the DDRCTL. The write data must not be presented to the DDRCTL until the corresponding write data pointer is returned to the interface logic. There is no other requirement for timing between write data and write data pointer. However, the write data must be presented in the same order that the pointers returned to the interface (same order in which the original requests are made).

The write data can be throttled using `hif_wdata_stall` signal. The write data is accepted by the DDRCTL when `hif_wdata_valid` is detected high and `hif_wdata_stall` is detected low at the same positive edge of the clock. The signal `hif_wdata_stall` is asserted by the DDRCTL during an RMW operation. When the DDRCTL writes the read data for the RMW operation into the write data SRAM, it blocks the SoC interface from sending any write data during those cycles by asserting this signal.

3.3.6.5 Write Data SRAM

The write data SRAM can be implemented internally as synthesized flip-flops or implemented externally to the DDRCTL as embedded SRAM using `UMCTL2_WDATA_EXTRAM` parameter. In either case, the control circuits for the Write Data RAM are clocked by the core DDRCTL clock. For more information about `wdata_ram` signals, see “[Signal Descriptions](#)” on page 373.

For data lane mapping examples, see “[Data Lane Mapping Examples](#)” on page 513.

Figures [Figure 3-7](#) on page 70 to [Figure 3-9](#) on page 72 show the timings of write and read data RAM interfaces.

Figure 3-7 Write Data RAM Interface Timing (DDRCTL_WDATARAM_WR_LATENCY=0, DDRCTL_WDATARAM_RD_LATENCY=1)

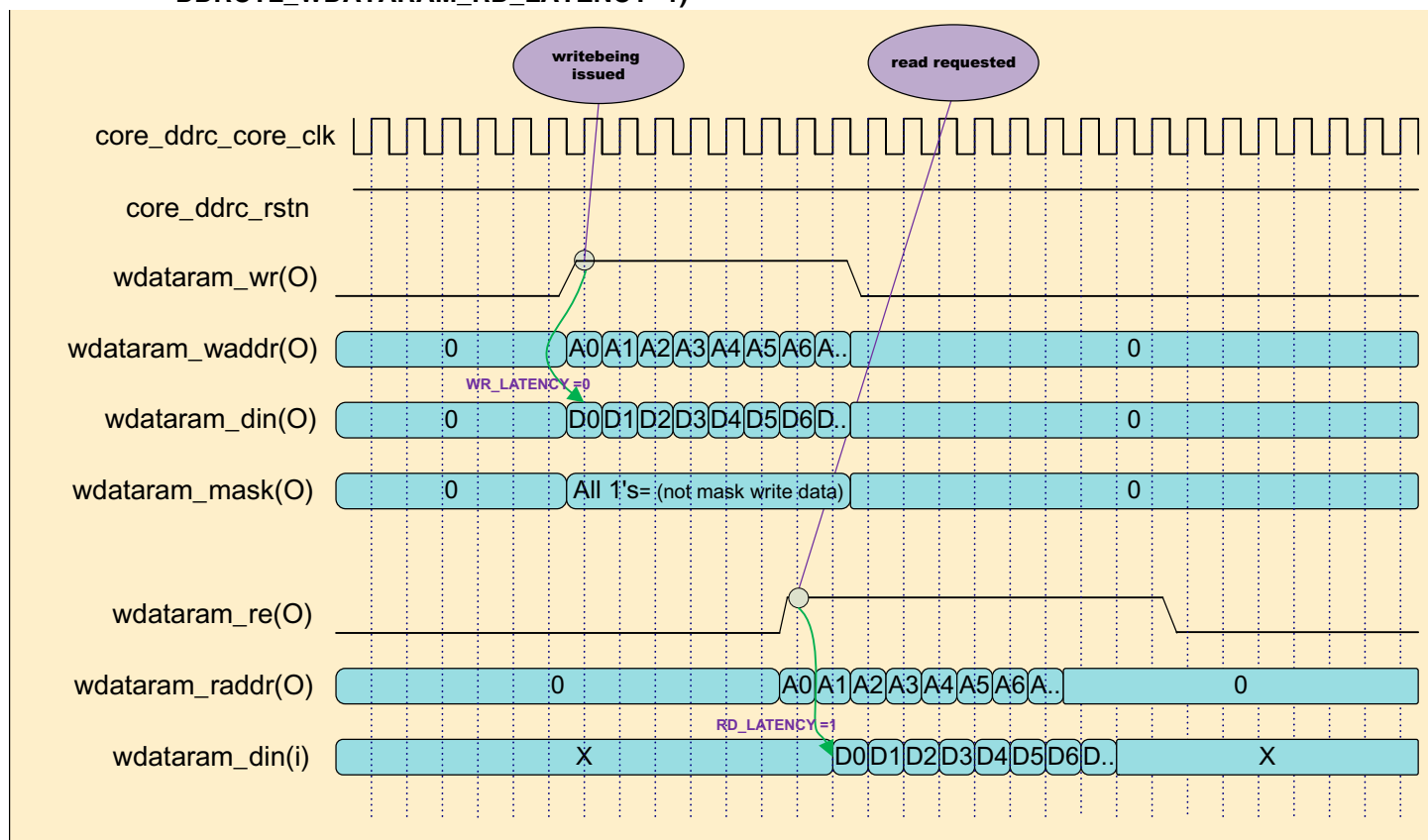
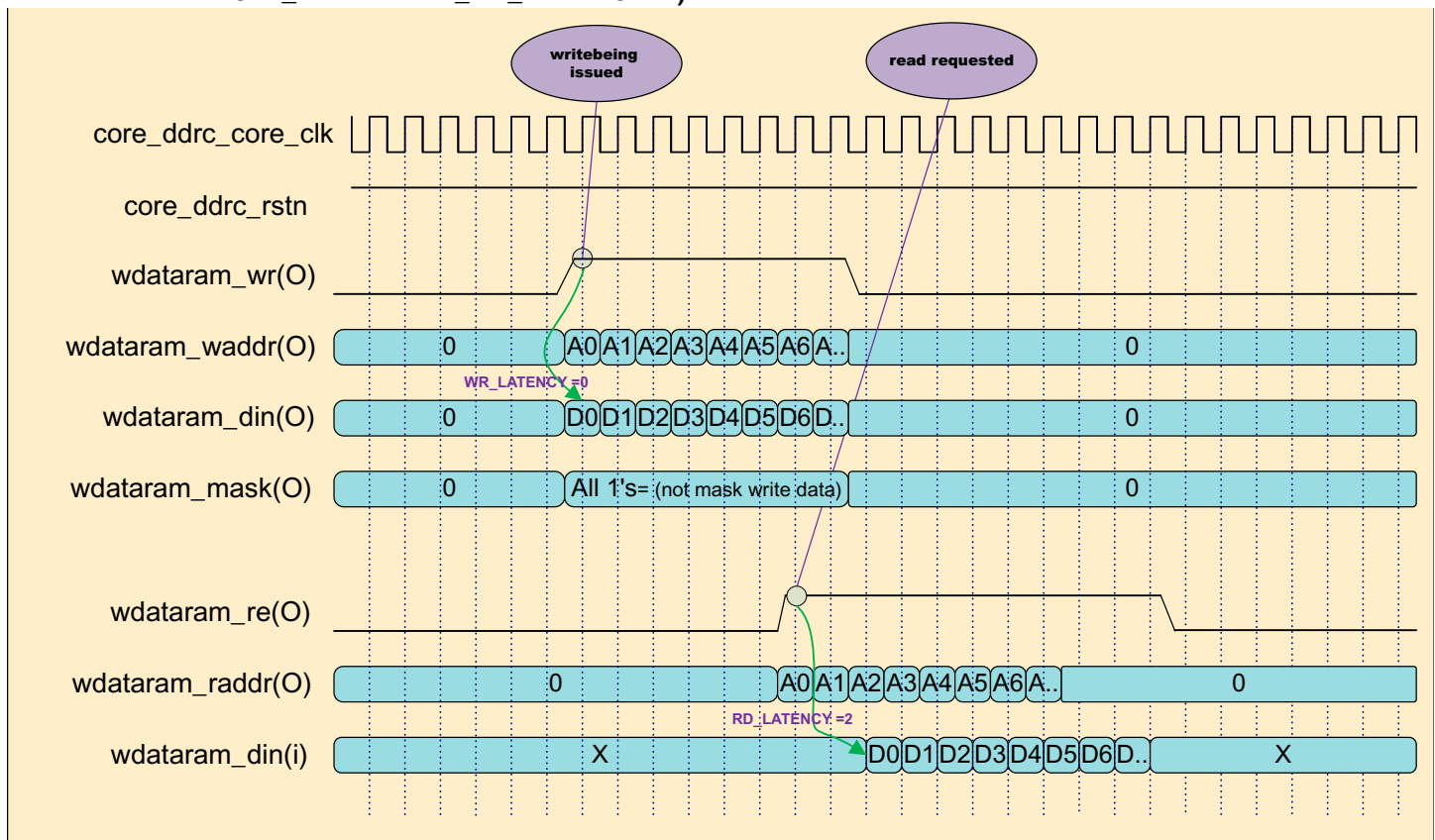
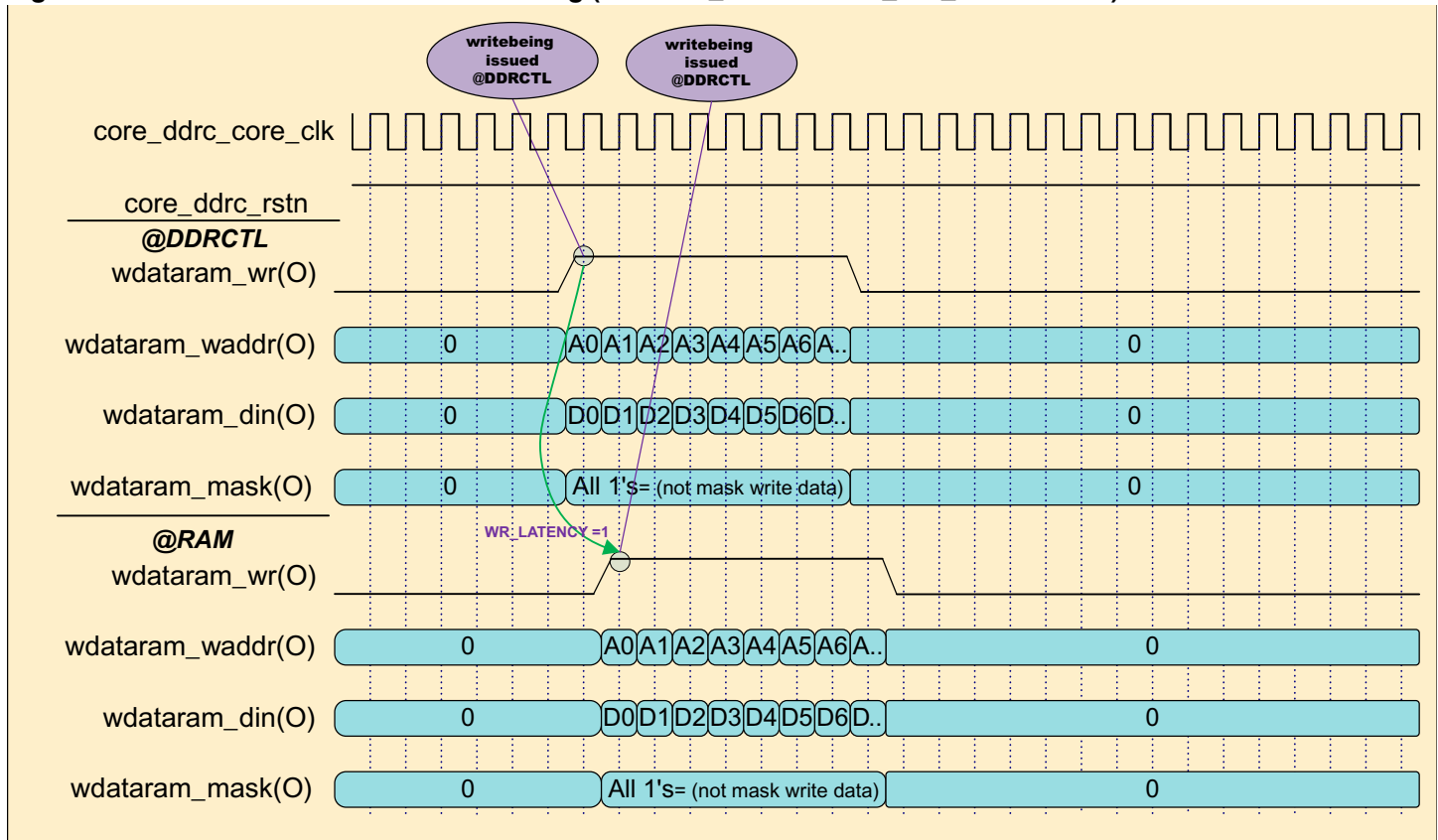


Figure 3-8 Write Data RAM Interface Timing (DDRCTL_WDATARAM_WR_LATENCY=0, DDRCTL_WDATARAM_RD_LATENCY=2)



The parameter `DDRCTL_WDATARAM_WR_LATENCY` specifies the number of clock cycles for external Write Data SRAM write that can be delayed before the RAM is written to. It is enabled when `(UMCTL2_WDATA_EXTRAM == 1) && (DDRCTL_DDR == 1)`.

Figure 3-9 on page 72 shows signals as seen at the DDRCTL and at the RAM.

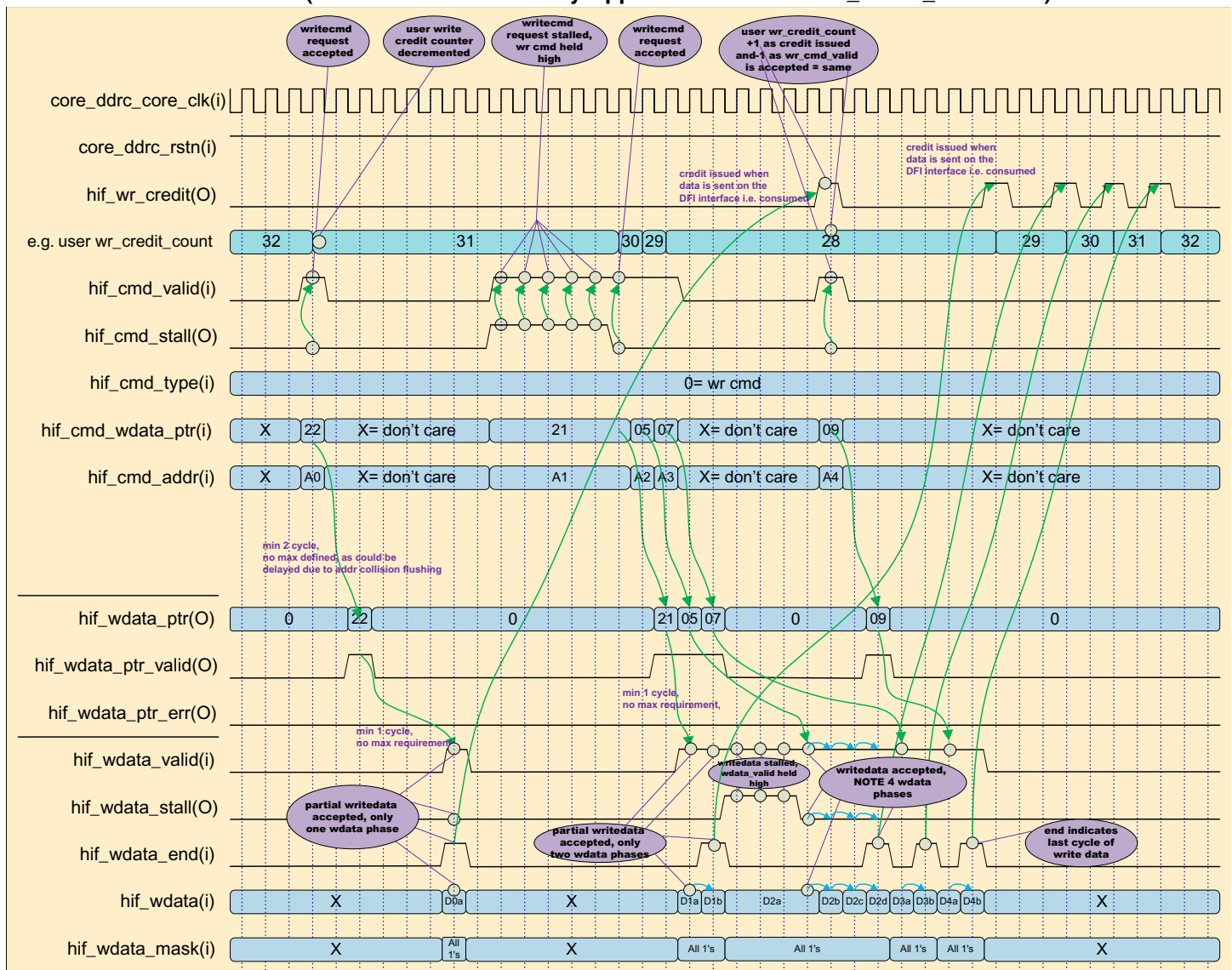
Figure 3-9 Write Data RAM Interface Timing (DDRCTL_WDATARAM_WR_LATENCY=1)**3.3.6.6 Write Data for the DDRCTL in BL16 Mode (MEMC_BURST_LENGTH=16)**

Write data is presented to the DDRCTL with the assertion of `hif_wdata_valid`. Each request can have one to two (1:4 frequency ratio) or four (1:2 frequency ratio) data beats. `hif_wdata_valid` must be asserted for each data phase and `hif_wdata_end` must be asserted on the last data beat. `hif_wdata` is the data. Both `hif_wdata_end` and `hif_wdata` are valid only when `hif_wdata_valid` is '1'.

In 1:4 frequency ratio mode (`MEMC_FREQ_RATIO = 4`), a normal write or RMW command has two clocks of data associated with it. In this case, `hif_wdata_end` must be asserted on the second data beat. If the SoC has less than two cycles of write data for a write command, it can choose to send one cycle of data, and `hif_wdata_end` must be asserted in the appropriate last data beat - these are referred to as Partial Writes. In this case, the DDRCTL has all the data that it requires one clock cycle earlier than in case of a full write.

Figure 3-10 shows sample timing diagrams.

Figure 3-10 Write Request and Data - One cmd Stall, One Write Data Stall With Mix of Partial Writes With 1, 2, 4 wdata Phases (4 wdata Phases Are Only Applicable When MEMC_FREQ_RATIO = 2)



3.3.6.7 Partial Writes

A Partial Write is where the number of HIF write data beats is less than the number required for a normal (full) write for the MEMC_BURST_LENGTH.

The DDRCTL issues the minimum number of SDRAM beats on the DDR interface required, depending on the number of HIF write data beats and the HIF address alignment with respect to the SDRAM Column address - as SDRAM Writes must be sent BL-aligned. This additional logic impacts the achievable synthesis timing and increases the area.

For LPDDR5/4/4X a partial write will result in a masked write if `DBICTL.dm_en = 1`, else an RMW if `MEMC_USE_RMW = 1`, otherwise partial write are not permitted. For more information, see “[Burst Mode Operation](#)” on page 170.

3.3.7 Read-Modify-Write Requests

Read-modify-write (RMW) functionality in the DDRCTL is intended to support sub-sized write accesses in the following cases, where it is not possible to perform masked writes:

- ECC configurations: To maintain the correct ECC value in SDRAM, a sub-sized write access of size less than the full memory data width (`MEMC_ECC_SUPPORT = 1`) or twice the memory data width (`MEMC_ECC_SUPPORT = 2`) must be implemented as an RMW so that the ECC can be recalculated over the entire memory data width.
- When Data Mask (DM) is not available, it is necessary to use an RMW to perform a sub-sized write access.
- LPDDR5/4 Data Mask (DM) is disabled: If DM function is disabled through MR13 OP[5], it is necessary to use an RMW to perform a sub-sized write access.

`MEMC_USE_RMW` can be always set to '1'.

For ECC configurations, if RMW is not used and single beat SECDED ECC mode is enabled, sub-size accesses of size less than the full memory data width result in incorrect ECC values being stored in the SDRAM. Therefore, RMW must be always set if sub-size accesses of size less than the full memory data width are required.

For LPDDR4/LPDDR5 HIF configurations, if x4 devices are used and RMW is not used, only full BL16 bursts are allowed.

More specifically, to perform an RMW, the HIF input signal, `hif_cmd_type` must be driven to “10”.

RMW requests result in a low priority read (LPR) and a write in the DDRCTL. The SoC must decrement one credit each for LPR and write credit counters for every RMW request sent to the DDRCTL. There is no RMW that results in a high priority read (HPR) part.

The DDRCTL allocates one CAM entry in the write CAM and the LPR CAM to handle this request. There is no read data going back to the SoC for RMW commands. The DDRCTL first schedules the read part of the RMW request. The read data from the SDRAM is merged with the write data of the RMW command in the on-chip write data SRAM (external to the DDRCTL). The merged data is then written to the SDRAM when the write is scheduled. The read and write requests of the RMW commands are treated as normal commands in the DDRCTL. The only difference is that the write command is not enabled in the CAM until the read data is merged with the write data.

The following signals are ignored for RMW commands:

- `hif_cmd_length`: the read associated with RMW command is always full read.
- `hif_cmd_token`: there is no read data going back to the SoC for RMW.
- `hif_cmd_pri`: when `UMCTL2_VPRW_EN` is set to '0', this signal is 1-bit wide. This field must be '0' for RMW commands.

When `UMCTL2_VPRW_EN` is set to '1', `hif_cmd_pri` signal is 2-bit wide.

The encoding for RMW is:

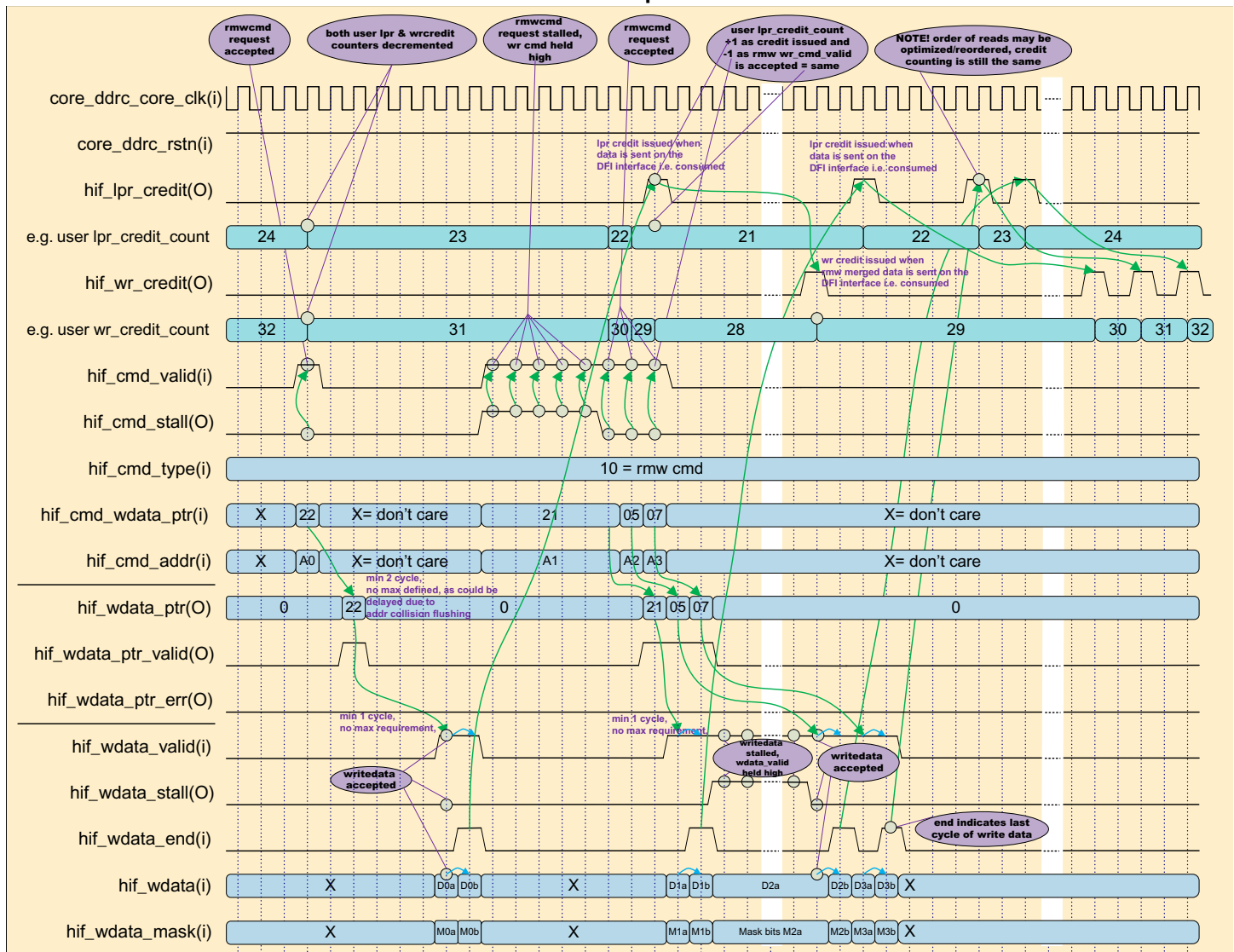
- 2'b00 - NPW (Normal Priority RMW)
- 2'b01 - VPW (Variable Priority RMW, both read part and write part of RMW are Variable Priority)

- 2'b10 - Reserved
- 2'b11 - Reserved

VPW commands are sent only in configurations where `UMCTL2_VPRW_EN = 1`.

The SoC must decrement a low priority read credit and a write credit for every RMW command that it sends to the DDRCTL.

Figure 3-11 RMW Request and WData with wmask - One cmd Stall, One Write Data Stall with Four RMW Requests with User LPR and Write Credit Counter Examples



3.3.8 WR Versus RMW for HIF Configurations With `DBICTL.dm_en=0`

If RMW is not used, only full BL16 aligned bursts are allowed. For DDR5 HIF configurations, if `DBICTL.dm_en = 0` (x4 devices are used) and RMW is not used, only full BL16 aligned bursts are allowed. Otherwise, an RMW must be used. More specifically:

Writes are allowed only in specific HIF address and number of HIF data beats combinations. Otherwise, RMW is required.

3.3.9 Half/Quarter Bus Width Mode

When half bus width mode is used (`MSTR0.data_bus_width = 2'b01`), the least significant half data for each DRAM beat of HIF read/write data is valid.

Invalid data of `hif_wdata` have to be set to '0'. For invalid data of `hif_wdata`, corresponding bits of `hif_wdata_mask` must be set to '1'.

For more information, see “[Bus Width Selection](#)” on page 170.



Note

- Quarter bus width must only be programmed in configurations where `MEMC_DRAM_DATA_WIDTH` is 32. In other cases, programming `MSTR0.data_bus_width = 2'b10` is deemed illegal and will lead to unpredictable behavior.
- The HBW/QBW feature is currently unsupported in configurations with OCECC (`UMCTL2_OCECC_EN == 1`).

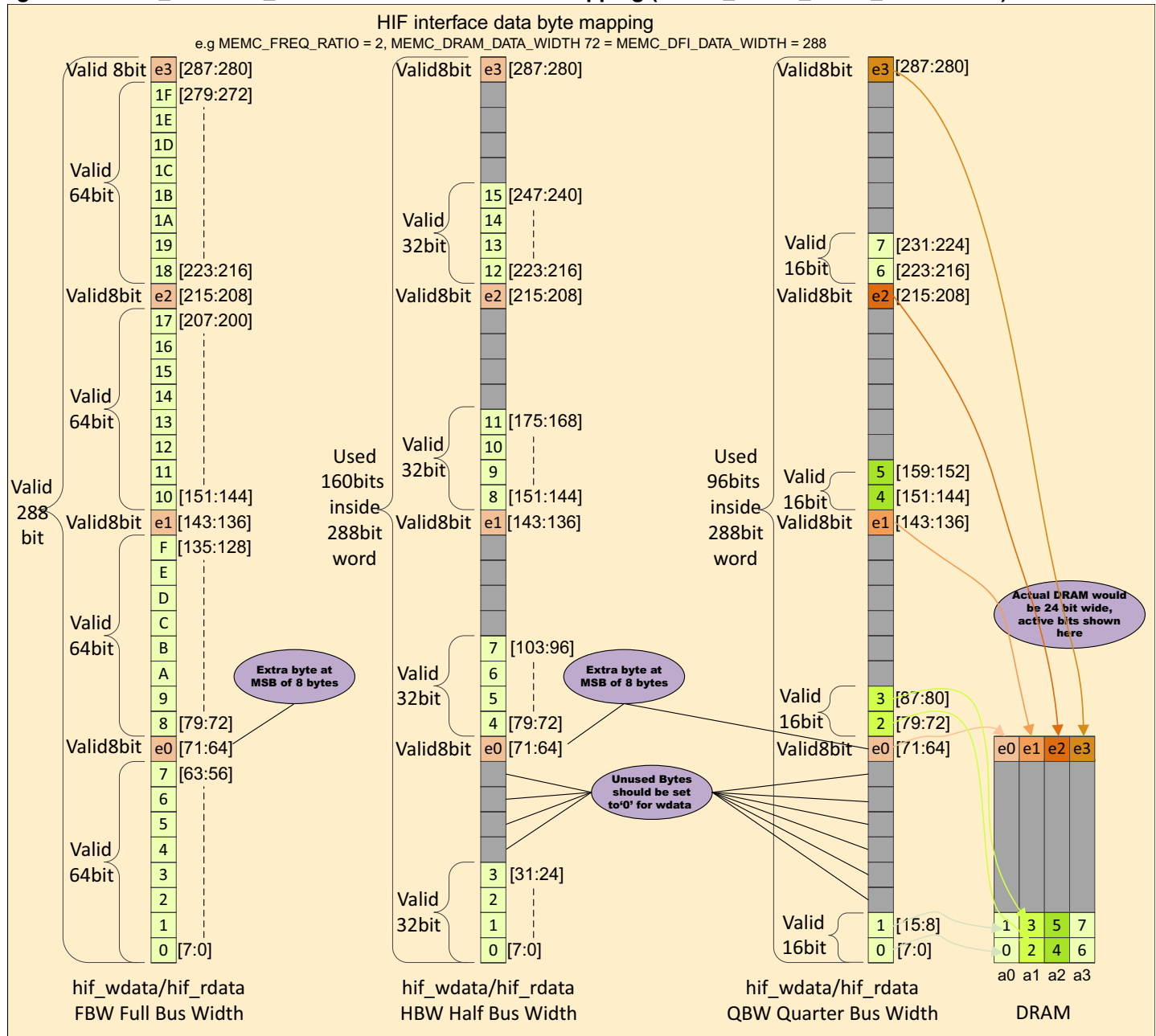
3.3.10 HIF Data Alignment when `MEMC_DRAM_DATA_WIDTH` is Non-Power of 2

When the `MEMC_DRAM_DATA_WIDTH` is set to '24', '40', '48', '56', or '72', this enables you to use your own ECC, encryption, and so on. As the `hif_wdata/hif_rdata` widths are defined as $(\text{MEMC_DRAM_DATA_WIDTH} * \text{MEMC_FREQ_RATIO}) * 2$, for example, when `MEMC_DRAM_DATA_WIDTH` is 72 and `MEMC_FREQ_RATIO` is 2, the `hif_wdata/hif_rdata` will be 288 bits wide. [Figure 3-12](#) on page 77 shows the data byte mapping for full, half and quarter bus width modes where the extra bits locations are always at the MSB of `MEMC_DRAM_DATA_WIDTH` word of data regardless whether in full, half or quarter bus width modes.



Note

- ECC by the controller is not supported when `MEMC_DRAM_DATA_WIDTH` is 24, 40, 48, 56 or 72, that is, a non-power of 2.
- For HIF configurations, `MEMC_DRAM_DATA_WIDTH` can be any multiple of 8, with a maximum of 72.

Figure 3-12 hif_wdata/hif_rdata FBW/HBW/QBW Data Mapping (MEMC_DRAM_DATA_WIDTH = 72)

3.3.11 hif_cmd_latency

Valid when `hif_cmd_valid` is high and `hif_cmd_pri` indicates variable priority read/write (VPR/VPW). Don't care for other priority types.

Specifies the timeout value of VPR/VPW request with a vector size of $[(HIF_RQOS_TW-1):0]$. The controller starts counting down VPR/VPW timer when the request is accepted in the controller. If VPR/VPW timeout value is set to '0', the VPR/VPW request expires immediately as it enters the controller, thereby making it the highest priority transaction class within the device. While in the controller the VPR/VPW timer for that transaction is > 0 , it is treated a low_priority transaction, when the VPR/VPW timer for that transaction is $= 0$ = expired-VP*, it is treated a high_priority (priority0) transaction.

3.3.12 hif_go2critical

When the DDRCTL is configured as HIF-only (`DDRCTL_SYS_INTF == 0`), it provides a feature where the SoC can force this state transition using external signals. It is useful in cases where the SoC may have additional information that is helpful in determining state transitions. For example, if the data stream has real-time requirements and the SoC has knowledge about FIFO depths or time-till-failure, it can use this information to explicitly request the DDRCTL to prioritize a particular data stream when it is critical to do so. The signals associated with this feature are `hif_go2critical_lpr`, `hif_go2critical_hpr` and `hif_go2critical_wr`.

For example, if the user logic implemented timing monitoring (duration of outstanding transaction waiting for credits) in the queue of transactions which are requested via the HIF interface, if any critical transaction is delayed (held for longer than is critical) due to 0 credits available, it can set the corresponding `hif_go2critical_(lpr/hpr/wr)` signal. These must be asserted only when necessary as it will cause a switch to service that queue (to make credits available) which can lead to less than optimal scheduling decisions.

3.3.13 hif_cmd_autopre

The explicit auto-precharge feature can enable auto-precharge on a per-command basis. If the HIF signal `hif_cmd_autopre` is set during a valid command, then the auto-precharge bit for that command is set when it is sent to the SDRAM.

For more information, see “[Page Policy](#)” on page 143 and for options, see [Table 7-1](#) on page 145.

3.3.14 hif_mrr_data_valid and hif_mrr_data

There are additional output signals `hif_mrr_data[(MEMC_MRR_DATA_TOTAL_DATA_WIDTH-1):0]` and `hif_mrr_data_valid` which make the mode register contents available only when `hif_mrr_data_valid` is asserted.

For more information, see “[Mode Register Reads](#)” on page 212.

3.3.15 Additional Error Signal Outputs

There are additional error output signals (subject to configuration) for indicating crc (`hif_rdata_crc_err`), ecc (`hif_rdata_uncorr_ecc_err`), parity (`hif_rdata_eapar_err`), and kbd (`hif_rdata_kbd`). For more information, see “[HIF Read Data Interface Signals](#)” on page 439.

3.4 Hardware Low-Power Interfaces

For information about Hardware low-power Interfaces, refer to “[Hardware Low-Power Interfaces](#)” on page [229](#).

4

DFI

This chapter contains the following sections:

- [“DFI Interface” on page 82](#)
- [“DFI Updates” on page 92](#)
- [“DFI Constraints” on page 95](#)
- [“DDRCTL to PHY Connections” on page 98](#)

4.1 DFI Interface

This topic contains the following sections:

- [“Overview of DFI Interface”](#) on page 82
- [“1:2 Frequency Ratio Mode Considerations”](#) on page 83
- [“DFI Write/Read Data to SDRAM Conversion”](#) on page 83
- [“Command Interface”](#) on page 84
- [“Write Data Interface”](#) on page 84
- [“Read Data Interface”](#) on page 85
- [“Update Interface”](#) on page 85
- [“Status Interface”](#) on page 86
- [“DFI Training”](#) on page 87
- [“Low-Power Control Interface”](#) on page 87
- [“PHY Master Interface”](#) on page 88
- [“Signals Related to DFI Interface”](#) on page 91
- [“Registers Related to DFI Interface”](#) on page 91

4.1.1 Overview of DFI Interface

The DDRCTL controller contains a DFI MC (Memory Controller) interface, which is used to connect to a DFI-compliant PHY, and to transfer address, control, and data to the PHY. In case of a Synopsys DWC DDR PHY, the DDRCTL interfaces to a PUB block, which acts as the DFI PHY interface. The DDRCTL controller including its DFI Interface, runs at:

- Half data rate (HDR) clock (referred to as 1:2 frequency ratio mode in the DFI Specification)
- Quarter data rate (QDR) clock (referred to as 1:4 frequency ratio mode in the DFI Specification)

You can get more information about the DFI interface from the following specification:

- DDR PHY Interface (DFI) Specification, version 5.0, April 27, 2018

The DFI interface is sub-divided into the following interface groups:

- **Command Interface:** The command interface is a reflection of the SDRAM control interface including address, bank, chip select, row strobe, column strobe, write enable, clock enable, and ODT control, as applicable for the memory technology.
- **Write Data Interface:** The write data interface handles the transmission of write data across the DFI interface. The DFI Specification defines signals and timing relationships. The timing of this interface is configurable, to support all DFI-compliant PHYs through the DFITMG0.dfi_tphy_wrlat and DFITMG0.dfi_tphy_wrdata registers.
- **Read Data Interface:** The read data interface handles the return of read data across the DFI interface. The DFI Specification defines signals and timing relationships. The timing of this interface is configurable, to support all DFI-compliant PHYs through the DFITMG0.t_rddata_en register.
- **Update Interface:** During system operation, the system may require updates to the internal settings to compensate for environmental conditions. To ensure that updates do not interfere with signals on the SDRAM interface, the DFI interface supports update modes where the DFI read, write, and command interface is suspended from the normal activity.

- Status Interface: The DFI interface requires status information for initialization and clock control to the SDRAM devices. These signals are used to convey information between the MC and PHY.
- Training (PHY-Independent mode supported by the DDRCTL): Write and read leveling function performed in a DFI-compliant PHY with no DDRCTL interaction.
- Low-power control interface: this interface consists of signals that are used to inform the PHY of a low-power mode opportunity, as well as how quickly the DDRCTL requires the PHY to resume normal operation.

4.1.2 1:2 Frequency Ratio Mode Considerations

The DDRCTL controller, including its DFI interface runs at the half data rate (HDR) clock (referred to as 1:2 frequency ratio mode in the DFI Specification), so all DFI command and address signals are twice the width of the equivalent DDR SDRAM signals. DFI data signals are four times the width of the equivalent DDR SDRAM signals. The DDRCTL controller operates with an HDR clock. The PHY handles HDR-to-SDR conversion on the address bus to memory and HDR/DDR conversion on the data buses. In 1:2 frequency ratio mode, `core_ddrc_core_clk` runs at half the frequency of CK/CK#.

4.1.3 DFI Write/Read Data to SDRAM Conversion

This conversion is performed in the PHY, and therefore PHY-Specific. The following details are applicable only to Synopsys DWC DDR PHYs; other PHYs may differ from this.

The DFI data (both read and write) must contain data for multiple byte lanes and for multiple data beats (4 in 1:2 mode, 8 in 1:4 mode). The DFI Specification does not explicitly state how this data must be packed within `dfi[0|1]_rddata_W[0|1|2|3]` and `dfi[0|1]_wrdata_P[0|1|2|3]`. The Synopsys DWC DDR PHYs and the DDRCTL DFI interfaces are designed with the assumption that the data is ordered by beat and then (within each beat) by byte.

For a 4-byte configuration with 32 bits of SDRAM in 1:2 mode, it is more complicated than 16 bits of SDRAM because both DFI0 and DFI1 need to be used. The DFI data order is as follows (from MSB to LSB):

- DFI1 (Phase 1)=[byte3-beat3, byte2-beat3, byte3-beat2, byte2-beat2]
- DFI0 (Phase 1)=[byte1-beat3, byte0-beat3, byte1-beat2, byte0-beat2]
- DFI1 (Phase 0)=[byte3-beat1, byte2-beat1, byte3-beat0, byte2-beat0]
- DFI0 (Phase 0)=[byte1-beat1, byte0-beat1, byte1-beat0, byte0-beat0]

Therefore, the PHY sends the data it receives out to the memory in the following way:

- In beat 0, the data from `dfi0_wrdata_P0[15:0]` are sent on DQ[15:0]
- In beat 1, the data from `dfi0_wrdata_P0[31:16]` are sent on DQ[15:0]
- In beat 2, the data from `dfi0_wrdata_P1[15:0]` are sent on DQ[15:0]
- In beat 3, the data from `dfi0_wrdata_P1[31:16]` are sent on DQ[15:0]
- In beat 0, the data from `dfi1_wrdata_P0[15:0]` are sent on DQ[31:16]
- In beat 1, the data from `dfi1_wrdata_P0[31:16]` are sent on DQ[31:16]
- In beat 2, the data from `dfi1_wrdata_P1[15:0]` are sent on DQ[31:16]
- In beat 3, the data from `dfi1_wrdata_P1[31:16]` are sent on DQ[31:16]

For more information on addressing, see “[Channel Modes](#)” on page 101.

Similar ordering is true for `dfi_wrdata_mask` and `dfi_rddata/dfi_rddata_dbi`.

The `dfi_wrdata_en`, `dfi_rddata_en`, and `dfi_rddata_valid` signals have one bit corresponding to each byte of DQ (not each byte of the DFI data signals).

This means, for a 4-byte configuration (32 bits of SDRAM) in 1:2 mode, the DFI ordering for `dfi_wrdata_en` is as follows (from MSB to LSB):

[byte3-phase1, byte2-phase1, byte1-phase1, byte0-phase1, byte3-phase0, byte2-phase0, byte1-phase0, byte0-phase0]

The DFI ordering for `dfi_rddata_en` signal is same as `dfi_wrdata_en`. The DFI ordering of `dfi_rddata_valid` is same except that phase0/1 is actually word0/1.

There may be confusion due to these different mappings:

- `dfi_wrdata/dfi_wrdata_mask/dfi_rddata/dfi_rddata_dbi`
- `dfi_wrdata_en/dfi_rddata_en/dfi_rddata_valid`

For clarification, refer to the provided example.

For a 4-byte configuration (32 bits of SDRAM) in 1:2 mode, `dfi_wrdata_en` and `dfi_wrdata` mapping are as follows:

```
dfi0_wrdata_en_P0[0] -> {dfi0_wrdata_P0[23:16], dfi0_wrdata_P0[ 7: 0]} (byte0-phase0)
dfi0_wrdata_en_P0[1] -> {dfi0_wrdata_P0[31:24], dfi0_wrdata_P0[15: 8]} (byte1-phase0)
dfi1_wrdata_en_P0[0] -> {dfi1_wrdata_P0[23:16], dfi1_wrdata_P0[ 7: 0]} (byte2-phase0)
dfi1_wrdata_en_P0[1] -> {dfi1_wrdata_P0[31:24], dfi1_wrdata_P0[15: 8]} (byte3-phase0)

dfi0_wrdata_en_P1[0] -> {dfi0_wrdata_P1[23:16], dfi0_wrdata_P1[ 7: 0]} (byte0-phase1)
dfi0_wrdata_en_P1[1] -> {dfi0_wrdata_P1[31:24], dfi0_wrdata_P1[15: 8]} (byte1-phase1)
dfi1_wrdata_en_P1[0] -> {dfi1_wrdata_P1[23:16], dfi1_wrdata_P1[ 7: 0]} (byte2-phase1)
dfi1_wrdata_en_P1[1] -> {dfi1_wrdata_P1[31:24], dfi1_wrdata_P1[15: 8]} (byte3-phase1)
```



Note

If non-Synopsys DWC DDR PHY is used, these signals may be reordered depending on the bytes/beats ordering of the non-Synopsys DWC DDR PHY.

4.1.4 Command Interface

The command interface is a reflection of the SDRAM control interface including address, bank, chip select, row strobe, column strobe, write enable, and clock enable, as applicable for the memory technology.

4.1.5 Write Data Interface

The write data interface handles the transmission of write data across the DFI interface. The DFI Specification defines signals and timing relationships.

The timing of this interface is configurable, to support all DFI-compliant PHYs through the `DFITMG0.dfi_tphy_wrlat` and `DFITMG0.dfi_tphy_wrdata` register.

For DFI 1:2, if HDR, timing for `dfi_wrdata*` generation for write commands on phase 0/phase 1 are the same and `dfi_wrdata*` signals are aligned to HDR clock. If SDR, timing for `dfi_wrdata*` generation for write commands on phase 0/phase 1 are treated separately and `dfi_wrdata*` signals are not guaranteed to be aligned to HDR clock. Check your PHY requirements for correct programming.

The signal `dfi_wrdata_mask` can act as Write DBI signal depending on the programming of the SDRAM mode register. For more information on DBI, see [“Data Bus Inversion \(DBI\)”](#) on page 214.

4.1.5.1 Support For dfiN_wrddata_cs_Px/dfiN_rddata_cs_Px Signals

The polarity of these signals is defined by `DFIMISC.dfi_data_cs_polarity`.

The signal `dfi_wrddata_cs` is driven as follows:

- The `dfi_wrddata_cs` is driven according to the relevant chip select `tphy_wrclat` DFI PHY clock cycles after any command that generates `dfi_wrddata`. This includes a write. It is guaranteed to remain at this value for a minimum of `tphy_wrcsgap` plus the time for the write data (usually `MSTR0.burst_rdw`).

Similarly, for `dfi_rddata_c`:

- The `dfi_rddata_cs` is driven according to the relevant chip select `tphy_rdcslat` DFI PHY clock cycles after any command that generates `dfi_rddata`. This includes a read, in LPDDR4, and a MRR. It is guaranteed to remain at this value for a minimum of `tphy_rdcsgap` plus the time for the read data (usually `MSTR0.burst_rdw`).

Table 4-1 outlines the DFI timing value and the equivalent register in the DDRCTL.

Table 4-1 DFI Timing Values for dfi_wrddata_cs/dfi_rddata_cs

DFI Timing Value	DDRCTL Register
<code>tphy_wrclat</code>	<code>DFITMG2.dfi_tphy_wrclat</code>
<code>tphy_wrcsgap</code>	<code>RANKTMG0.diff_rank_wr_gap</code>
<code>tphy_rdcslat</code>	<code>DFITMG2.dfi_tphy_rdcslat</code>
<code>tphy_rdcsgap</code>	<code>RANKTMG0.diff_rank_rd_gap</code>

4.1.6 Read Data Interface

The read data interface handles the return of read data across the DFI interface. The DFI Specification defines signals and timing relationships.

The timing of this interface is configurable, to support all DFI-compliant PHYs through the `DFITMG0.t_rddata_en` register.

For DFI 1:2, in case of HDR, timing for `dfi_rddata_en` generation for read commands on phase 0/phase 1 is the same and `dfi_rddata*` signals are aligned to HDR clock. In case of SDR, timing for `dfi_rddata*` generation for read commands on phase 0/phase 1 are treated separately and `dfi_rddata*` signals are not guaranteed to be aligned to HDR clock. Check your PHY requirements for correct programming.

The DDRCTL controller has been verified with the Synopsys PHYs. It assumes that the DFI parameter `tphy_rdlat` does not exceed 48 cycles.

4.1.7 Update Interface

During system operation, the system may require updates to the internal settings to compensate for environmental conditions. To ensure that updates do not interfere with signals on the SDRAM interface, the DFI interface supports update modes where the DFI read, write, and command interface is suspended from the normal activity. The DFI Specification defines both MC-initiated and PHY-initiated updates.

4.1.7.1 MC-initiated Updates

MC-initiated updates can be acknowledged or ignored by the PHY. DFI MC-initiated updates are performed periodically by the DDRCTL. For more details, see “[DFI MC-Initiated Update Requests](#)” on page 92.

4.1.7.2 PHY-initiated Updates

For more details, see “[DFI PHY-initiated Update Requests](#)” on page 93.

4.1.8 Status Interface

The DFI interface requires status information for initialization and clock control to the SDRAM devices. These signals are used to convey information between the MC and PHY.

4.1.8.1 Initialization

The `dfi_init_start` signal is used to trigger the PHY initialization by setting it to ‘1’. It is driven by the DDRCTL through the APB interface with the `DFIMISC.dfi_init_start` read-write register field. The signal `dfi_init_complete` indicates that the PHY has completed its initialization. This information can be read/pollled by the DDRCTL through the APB interface with the `DFISTAT.dfi_init_complete` read-only register field.

4.1.8.2 Clock Disabling

The `dfi_dram_clk_disable` signal depends on setting of `PWRCTL.en_dfi_dram_clk_disable`. For more information about clock disabling, see “[Assertion of dfi_dram_clk_disable](#)” on page 226.

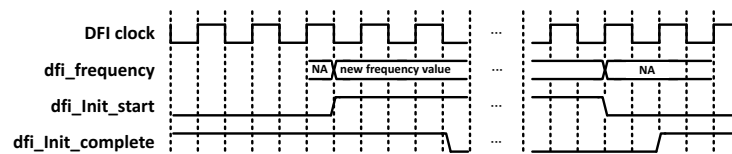
4.1.8.3 Frequency Change

The interface consists of three signals

- `dfi_init_start`: Output from the controller to the PHY
- `dfi_init_complete`: Output from the PHY to the controller
- `dfi_frequency`: Output from the controller to the PHY

The procedure is as follows:

Figure 4-1 Frequency Change



`dfi_frequency` indicates the operating frequency of the system. This signal must change only at initialization, during a DFI frequency change operation, or other times that the system defines. This signal

must be constant during normal operation. The number of supported frequencies and the mapping of signal values to clock frequencies are defined by the PHY, system, or both. For timing, the `dfi_frequency` signal must be set to a legal value and remain unchanged when `dfi_init_start` is asserted. The `dfi_frequency` signal can change any time when `dfi_init_start` is low and must be ignored at this time.

The behavior of the `dfi_init_start` signal is dependent on the `dfi_init_complete` signal.

If the PHY accepts the frequency change request, it must de-assert the `dfi_init_complete` signal within `tinit_start` cycles of the `dfi_init_start` assertion. The MC continues to hold the `dfi_init_start` signal asserted until the clock frequency change has been completed. The de-assertion must be used by the PHY to re-initialize on the new clock frequency.

If the frequency change is not acknowledged (the `dfi_init_complete` signal remains asserted), the `dfi_init_start` signal must de-assert after `tinit_start` cycles. This requirement needs to be handled by the system as `dfi_init_start` is programmed through the software.

**Note**

Once DDRCTL asserts HWFFC driven `dfi_init_start`, it will not be de-asserted even if `tinit_start` is elapsed until the PHY de-asserts `dfi_init_complete` followed by assertion of `csysreq_ddrc`.

4.1.9 DFI Training

The additional functions of SDRAM memories allow accurate alignment of critical timing signals. These are trained through write and read leveling functions in a DFI-compliant PHY.

The supported modes in DDRCTL are described in “[PHY Independent Mode](#)” on page 87.

If you are using the Synopsys DWC DDR PHY, use PHY independent mode. While interfacing to Synopsys DWC DDR PHYs, the training is initiated by the PHY Utility Block (PUB) present in the PHY, and PHY independent training mode is used.

4.1.9.1 PHY Independent Mode

PHY independent mode are enabled by default and no dedicated signals are required.

Training is assumed to be performed automatically by the PHY, without any intervention from the DDRCTL.

If you are using the Synopsys DWC DDR PHY, the PUB block (part of the PHY) performs all required MRS commands to move the SDRAM to the correct state for each training stage.

If you are using a non-Synopsys PHY that supports PHY independent training mode, DDRCTL may have to perform these MRS commands.

However, DDRCTL has no knowledge on when the PHY is ready to execute each training stage, so this must be controlled by the user.

4.1.10 Low-Power Control Interface

In a DDR memory subsystem, it is advantageous to place the PHY in a low-power state when the DDRCTL has knowledge that the memory subsystem remains idle for a period of time. Depending on the state of the system, the DDRCTL communicates state information to the PHY allowing the PHY to enter the appropriate

power saving state. This interface consists of signals that are used to inform the PHY of a low-power mode opportunity, as well as how quickly the DDRCTL requires the PHY to resume normal operation.

Software control is provided in DFILPCFG0/DFILPCFG1 registers for the following:

- This is an optional interface for a DFI-compliant PHY. The interface signals always exist in the DDRCTL. But the software can enable both `dfi_lp_ctrl_req` and `dfi_lp_data_req` interfaces in self-refresh power down and/or power down and/or deep sleep mode - `DFILPCFG0.dfi_lp_en_pd`, `DFILPCFG0.dfi_lp_en_sr`, and `DFILPCFG0.dfi_lp_en_dsm` respectively.
The software can also enable `dfi_lp_data_req` interface in data bus idle `DFILPCFG0.dfi_lp_en_data`.
- Software can control the gap from de-asserting `dfi_lp_data_ack` to asserting `dfi_wrdata_en` by `DFILPCFG0.dfi_lp_extra_gap_wr`.
- Software can control `dfi_lp_data_req` interface by `DFILPCFG0.dfi_lp_data_req_en`. When `DFILPCFG0.dfi_lp_data_req_en` is set to '0', `dfi_lp_data_req` is never asserted regardless of `DFILPCFG0.dfi_lp_en_pd`, `DFILPCFG0.dfi_lp_en_sr`, `DFILPCFG0.dfi_lp_en_dsm`, and `DFILPCFG0.dfi_lp_en_data`.
- Software control over what is driven on `dfi_lp_ctrl_wakeup` and `dfi_lp_data_wakeup` signals (and its associated timing) in self-refresh power down and/or power down and/or deep sleep mode - `DFILPTMG0.dfi_lp_wakeup_pd`, `DFILPTMG0.dfi_lp_wakeup_sr`, and `DFILPCFG0.dfi_lp_wakeup_dsm` respectively.
The software can also control what is driven on `dfi_lp_data_wakeup` signal (and its associated timing) in data bus idle `DFILPTMG1.dfi_lp_wakeup_data`.
- The timing of this DFI low-power interface - `DFILPTMG0.dfi_tlp_resp`.

For more information, see [“Low-Power and Power-Saving Features”](#) on page 217.

4.1.11 PHY Master Interface

This topic contains the following sections:

- [“Overview of PHY Master Interface”](#) on page 88
- [“LPDDR5/4 Considerations”](#) on page 90
- [“Registers Related to PHY Master Interface”](#) on page 90

4.1.11.1 Overview of PHY Master Interface

The PHY uses the PHY Master Interface for requesting DDRCTL to relinquish the DFI bus and take control of the DRAM bus. When the PHY has control of the DFI bus, it performs all operations independent of the DDRCTL. This interface is only supported in LPDDR4 or LPDDR5. This interface requires that the "Refresh Control Interface" feature be disabled in software.

The PHY can request the memory to be placed in a specific state (such as, IDLE or self-refresh through `dfi_phymstr_cs_state/dfi_phymstr_state_sel`). However, the DDRCTL puts the SDRAM into Self-Refresh, without consideration of `dfi_phymstr_cs_state/dfi_phymstr_state_sel`.

DDRCTL is not expected to acknowledge `dfi_phymstr_req` in the following situations:

- The PHY does not issue a PHY Master request during SDRAM Initialization, or Deep Sleep Mode (LPDDR5 only).

- The PHY does not issue PHY Master request and PHY Update request simultaneously.

Support for this interface can be enabled/disabled using `DFIPHYMSTR.dfi_phymstr_en`. If enabled through the software, when a PHY Master request occurs, DDRCTL performs the following steps:

1. Ensures that there are no DFI low-power requests pending or in-service and de-asserts both `dfi_lp_ctrl_req` and `dfi_lp_data_req`.
2. Precharges all banks.
3. If `dfi_phymstr_req` is asserted after a SRX and there has been no REF after SRX, DDRCTL issues (`RFSHMOD0.refresh_burst+1`) REF/REFpb commands to speed up the SR entry. There is a JEDEC requirement for at least a REF (or 8 REFpb) to occur between SRX and SRE.
4. Enters self-refresh mode without draining the CAMs (to speed up the response time).
5. Acknowledges the PHY Master request.

After `dfi_phymstr_req` is dropped, DDRCTL de-asserts `dfi_phymstr_ack` signal and initiates Self-Refresh Exit.

The DDRCTL is expected to respond a `dfi_phymstr_req` in `tphymstr_resp` = 8k clock cycles. This is not guaranteed to be satisfied in the following situations:

- Automatic refreshes are disabled (that is, `RFSHCTL0.dis_auto_refresh` = 1); if at least one REF command is needed before entering Self-Refresh, response time depends on software sending refreshes.
- While sending burst of automatic REF commands (needed before entering Self-Refresh); these REF commands are postponed due to JEDEC requirement of 16 refreshes in 2xtREFI.
- While sending burst of automatic REF commands (needed before entering Self-Refresh); these REF commands are postponed due to JEDEC requirement of 8 refreshes in a rolling tREFBW window.
- LPDDR4 case:
 - `RFSHMOD0.per_bank_refresh` = 1
 - `MEMC_NUM_RANKS` = 2
 - `MSTR0.active_ranks` = 0x3
 - `ZQCTL2.zq_resistor_shared` = 1
 - `ZQSET1TMG.t_zq_long_nop` > 900
- While DFI LP is active. In this situation, upon DFI LP exit, the following needs to hold:
 - `dfi_lp_ctrl_wakeup`
 - `dfi_lp_data_wakeup`

Note that, if clock removal feature is used (`core_ddrc_core_clk` is removed from the DDRCTL):

- It is not guaranteed to satisfy DFI's `tphymstr_resp` requirement of PHY, as `dfi_phymstr_req` = 1 is only observed once the clock has been re-enabled.
- If `dfi_phymstr_req` transitions from '1' to '0' and back to '1' while the clock has been removed, logic does not ensure that `dfi_phymstr_req` assertion is acknowledged through `dfi_phymstr_ack`.

**Note**

- Enable DFI low-power interface for self-refresh (`DFILPCFG0.dfi_lp_en_sr`) setting is ignored during self-refresh entry caused by PHY Master request. DFI low-power interface needs to remain inactive during PHY Master request.
- If the PHY does not support PHY Master, it is recommended to set the following inputs:
 - `DFIPHYMSTR.dfi_phymstr_en = 1'b0`
 - `dfi_phymstr_req = 1'b0`
 - `dfi_phymstr_cs_state = {MEMC_NUM_RANKS{1'b0}}`
 - `dfi_phymstr_state_sel = 1'b0`
 - `dfi_phymstr_type = 2'b00`

4.1.11.2 LPDDR5/4 Considerations

In LPDDR5/4, there are two flavors of Self-Refresh state:

- Self-Refresh (SR): CKE is high, clock frequency cannot be changed, clock cannot be removed, some commands such as MRW/MRR can be performed.
- Self-Refresh Power Down (SRPD): CKE is low, clock frequency can be changed, clock can be removed, no commands can be performed.

LPDDR5/4 SRPD is the equivalent of Self-Refresh in other protocols. LPDDR5/4 SR is unique to LPDDR5/4 protocol. According to DFI 4.0, `dfi_phymstr_req` must put SDRAM into SR, not SRPD. To support this, DDRCTL is able to move:

- Into and out of SR only (IDLE → SR → IDLE)
- Out of and into SRPD (SRPD → SR → SRPD)

If hardware low-power interface is active, SRPD → SR → SRPD is not performed upon `dfi_phymstr_req` assertion, because in this situation, for most cases, the clock is removed, so transition is not possible. When this happens, DDRCTL informs external clock controller to re-enable the clock, by setting `cactive_ddrc` to '1'.

If software Self-Refresh is in progress (`PWRCTL.selfref_sw = 1`), SRPD → SR → SRPD is performed if enabled by the `PWRCTL.lpddr4_sr_allowed` register (see the “Register Descriptions” chapter of the [DWC DDRCTL LPDDR5/4 Programming Guide](#)).

**Note**

`DFIPHYMSTR.dfi_phymstr_blk_ref_x32` must be used with the default value (0x80) unless Synopsys suggests to update it.

4.1.11.3 Registers Related to PHY Master Interface

The following are the registers related to the PHY Master Interface:

- `DFIPHYMSTR.dfi_phymstr_en`
- `PWRCTL.lpddr4_sr_allowed`
- `DFIPHYMSTR.dfi_phymstr_blk_ref_x32`

For more information about these registers, see the “Register Descriptions” chapter of the [DWC DDRCTL LPDDR5/4 Programming Guide](#).

4.1.12 Signals Related to DFI Interface

The following are the signals related to the DFI Interface:

- DFI Command Interface Signals
- DFI Write Data Interface Signals
- DFI Read Data Interface Signals
- Non-DFI DDRCTL PHY Sideband Interface Signals
- DFI Update Interface Signals
- DFI Status Interface Signals
- DFI Low-Power Control Interface Signals
- DFI PHY Master Interface Signals
- DFI MC to PHY Message Interface Signals

For more information about these signals, see [“Signal Descriptions”](#) on page 373.

4.1.13 Registers Related to DFI Interface

The following are the registers related to the DFI Interface:

- DFITMGx
- DFILPCFGx
- DFIUPDx
- DFIMISC
- DFISTAT
- DFIPHYMSTR
- DFIMSGTMG0
- DFI0MSGCTL0
- DFI0MSGSTAT0

For more information about these registers, see the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

4.2 DFI Updates

This topic contains the following sections:

- [“Overview of DFI Updates”](#) on page 92
- [“DFI MC-Initiated Update Requests”](#) on page 92
- [“DFI PHY-initiated Update Requests”](#) on page 93
- [“Registers Related to DFI Updates”](#) on page 94

4.2.1 Overview of DFI Updates

DFI update interface handles the transmission of updates to internal settings to compensate for environmental conditions. The interface includes signals and timing parameters. To ensure that updates do not interfere with signals on the DRAM interface, the DFI supports update modes where the DFI bus is placed must be in an idle state.

4.2.2 DFI MC-Initiated Update Requests



Note

The DDRCTL supports two types of DFI MC-initiated update.

DFI MC-initiated update means the type 0 of `dfi_ctrlupd_req` unless otherwise stated.

The type 1 of `dfi_ctrlupd_req` is described in [“Enhanced Incremental Periodic Phase Training \(PPT2\)”](#) on page 200.

DFI MC-initiated update request (`dfi_ctrlupd_req`) is issued by the DDRCTL on the DFI interface so that the PHY can update delay line values from the master delay line. This signal must be asserted periodically to ensure that PVT (Process/Voltage/Temperature) variations are accounted for in the delay lines in PHY over time. An acknowledgment signal comes from PHY (`dfi_ctrlupd_ack`) to indicate that the delay line update process in PHY is complete. The DDRCTL follows the DFI Specification minimum and maximum requirements for duration to keep the DFI update request asserted.

It is important to control this carefully to ensure that the delay line is never updated during a read or a write as this could degrade the data eye.

The DDRCTL delays the assertion of `dfi_ctrlupd_req` if a PHY-initiated update is in progress (`dfi_phyupd_ack` = 1).

There are two methods for performing the DFI updates:

- [“Automatic MC-Initiated Update Request”](#) on page 92
- [“Direct Software Request of MC-initiated Update Request”](#) on page 93

4.2.2.1 Automatic MC-Initiated Update Request

This method is used when `DFIUPD0.dis_auto_ctrlupd` is set to '0'. The DDRCTL must carefully control the assertion of `dfi_ctrlupd_req`. The DDRCTL asserts `dfi_ctrlupd_req` any time the DDRCTL is idle, where idle is defined as having no read or write cycles in the CAMs, no read responses buffered in the RT (Response Tracker) block, and no write data pending in the MR (memory read) block.

To force the DFI update requests, when the DDRCTL is not idle for a long period of time (this depends on the PHY - how frequently it requires an update request), the DDRCTL uses a timeout mechanism. When a

DFI MC-initiated update timeout occurs, the DDRCTL ceases to schedule reads and writes until the MR and RT blocks are idle and then `dfi_ctrlupd_req` is asserted.

For single-rank configurations with read/write traffic only, with `DFITMG1.dfi_t_ctrlupd_interval_max_x1024 = 255`, `DFIUPD0.dis_auto_ctrlupd_srx = 1`, and with per-bank refreshes disabled (`RFSHMOD0.per_bank_refresh = 0`), DDRCTL sends `dfi_ctrlupd_req` only after refresh commands. However, `dfi_phymstr_req` can be asserted between refresh command and `dfi_ctrlupd_req`. The `DFITMG1.dfi_t_ctrlupd_interval_min_x1024` field sets how often the `dfi_ctrlupd_req` is triggered. Setting a smaller value to this field leads to having a `dfi_ctrlupd_req` after each refresh command, while setting a bigger value leads to having a `dfi_ctrlupd_req` after each *nth* refresh. If `DFIUPD0.dis_auto_ctrlupd_srx = 0`, the `dfi_ctrlupd_req` gets triggered either before or after each Self-Refresh Exit command, depending on the value of `DFIUPD0.ctrlupd_pre_srx`.



Note

If per-bank refresh feature is enabled or `DFIUPD1.dfi_t_ctrlupd_interval_min_x1024` is set to a high value and refresh bursts are used, it is possible to have the `dfi_ctrlupd_req` signal asserted after a command other than refresh.

4.2.2.2 Direct Software Request of MC-initiated Update Request

This method is used when `DFIUPD0.dis_auto_ctrlupd` is set to '1' through the software. In this case, the SoC decides when to do the DFI MC-initiated update requests. This is useful in cases where the bandwidth utilization of the SDRAM bus is extremely crucial and the DDRCTL does not want any break in transactions.

To perform the request, SoC must set `OPCTRLCMD.ctrlupd` to '1'. When the request is stored in the DDRCTL, the register bit is automatically cleared. The SoC can perform this request only if `OPCTRLSTAT.ctrlupd_busy` is low. The `OPCTRLSTAT.ctrlupd_busy` signal goes high in the clock after the DDRCTL accepts the request. It goes low when the DFI update operation is initiated in the DDRCTL.

The DDRCTL may assert `dfi_ctrlupd_req` at the same time as refresh assertion. For both the refreshes and the DFI MC-initiated update request, the DDRCTL ensures that these happen in a functionally-correct manner. The SoC logic ensures that they are scheduled frequently, and done at a time when the controller can tolerate the associated break in read/write scheduling.

4.2.3 DFI PHY-initiated Update Requests

DFI PHY-initiated update requests must be acknowledged by the DDRCTL. DFI specifies 4 different update PHY-initiate request modes. Each mode differs only in the number of cycles that the DFI interface must be suspended when the update occurs.

Support for this interface can be enabled/disabled using `DFIUPD0.dfi_phyupd_en`. If enabled through the software, when a PHY-initiated update request occurs, the DFI command, read data and write data channels are stalled as soon as possible (within a maximum of 64 clock cycles in 1:2 frequency ratio mode), and the DDRCTL drives `dfi_phyupd_ack` high in response.

The only exception to this is when both update request signals (`dfi_ctrlupd_req` and `dfi_phyupd_req`) are asserted at the same time. When both request signals are driven, the DDRCTL and

the PHY could violate the DFI protocol by simultaneously acknowledging the other's request. To prevent this situation, the DDRCTL does not assert `dfi_phyupd_ack` while `dfi_ctrlupd_req` is asserted.

Note that if burst refreshes are being used with PHY-initiated updates, care must be taken to avoid tREFI violations, which could occur if a PHY update request is received shortly before a refresh burst is due to be transmitted. In this situation, the refresh burst is delayed until after the PHY update is complete. This can be avoided by reducing the value of `RFSHMOD0.refresh_burst`.

Note that if clock removal feature is exercised (`core_ddrc_core_clk` is removed from the DDRCTL):

- It is not guaranteed to satisfy DFI's `tphyupd_resp` requirement of PHY, as `dfi_phyupd_req` = 1 is only observed once clock has been re-enabled.
- If `dfi_phyupd_req` transitions from '1' to '0' and back to '1' while clock has been removed, logic does not guarantee that `dfi_phyupd_req` assertion is acknowledged through `dfi_phyupd_ack`.



Note

- If PHY does not support the PHY-initiated updates, it is recommended to set the following inputs:
 - `DFIUPD2.dfi_phyupd_en` = 1'b0
 - `phy_ddrc_dfi_phyupd_req` = 1'b0
 - `phy_ddrc_dfi_phyupd_type` = 2'b00
- The DDRCTL controller supports the following `tPHYUPD_RESP` value in terms of DFI clocks (controller clocks):

MSTR.lpddr4	0		1	
Frequency Ratio Mode	1:2	1:4	1:2	1:4
tPHYUPD_RESP	64	32	128	64

4.2.4 Registers Related to DFI Updates

The following are the registers related to the DFI Updates:

- `DFIUPD0.dfi_phyupd_en`
- `DFITMG0.dfi_t_ctrl_delay`

For more information about these registers, see the "Register Descriptions" chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

4.3 DFI Constraints

This topic contains the following sections:

- “[Overview of DFI Constraints](#)” on page 95
- “[WCK Related Constraints \(LPDDR5\)](#)” on page 95
- “[WCK Constraints \(LPDDR5\)](#)” on page 96

4.3.1 Overview of DFI Constraints

Some constraints apply to every command regardless of the rank or bank targeted by the command. These constraints are primarily related to gaining access to the data bus while avoiding bus contention. A global constraints block enforces each of these constraints. Using the global constraints block, the scheduler dynamically obeys all of the constraints in [Table 4-2](#) when scheduling transactions.

Table 4-2 DFI Constraints

Control Register	Constraint Name	Description
DFITMG0.dfi_tphy_wrlat	Write to write-enable	This constraint is the time after a write command that dfi_wrdata_en must be driven to SDRAM. This corresponds to the DFI parameter tphy_wrlat.
DFITMG0.dfi_tphy_wrdata	Write-enable to write data time	This constraint is the time after dfi_wrdata_en that the data is driven to SDRAM. This corresponds to the DFI parameter tphy_wrdata.
DFITMG0.dfi_t_rddata_en	Read command to read enable time	This constraint is the time from the assertion of a read command on the DFI interface to the assertion of the dfi_rddata_en signal. This constraint corresponds to the DFI parameter trddata_en.
DFITMG0.dfi_t_ctrl_delay	Time for command to pass through PHY	This constraint corresponds to the DFI parameter tctrl_delay.

4.3.2 WCK Related Constraints (LPDDR5)

WCK related constraints are listed in [Table 4-3](#):

Table 4-3 WCK Related Constraints (LPDDR5)

Control Register	Constraint Name	Description
DFITMG4.dfi_twck_en_rd	$t_{wck_en_rd}$	WCK Enable Read Timing. Defines the timing from the CAS_WS_RD command to driving of the dfi_wck_en=ENABLED.
DFITMG4.dfi_twck_en_wr	$t_{wck_en_wr}$	WCK Enable Write Timing. Defines the timing from the CAS_WS_WR command to driving of the dfi_wck_en=ENABLED.

Control Register	Constraint Name	Description
DFITMG4.dfi_twck_dis	t_{wck_dis}	WCK Off Timing. Defines the timing from the last command opportunity to the de-assertion of dfi_wck_en and dfi_wck_toggle_en assuming that no command is being sent.
DFITMG5.dfi_twck_fast_toggle	$t_{wck_fast_toggle}$	This constraint defines the number of clock cycles between the dfi_wck_signal being driven to TOGGLE to when the dfi_wck_signal is driven to FAST_TOGGLE. This timing is only applicable when the WCK transitions from the slow to fast toggle. Otherwise, this timing parameter must be set to 0x0.
DFITMG5.dfi_twck_toggle	t_{wck_toggle}	This constraint defines the number of clock cycles between the dfi_wck_en signal being enabled to when the dfi_wck_toggle signal is driven to TOGGLE.
DFITMG5.dfi_twck_toggle_cs	$t_{wck_toggle_cs}$	This constraint defines the number of clock cycles between a read or write command to when the dfi_wck_cs signal must be stable. This timing is applicable when the WCK is synchronized for multiple CS's and commands are to different CS's. During WCK synchronization, the CS must be static from the CAS command to the completion of the synchronization sequence.
DFITMG5.dfi_twck_toggle_post t	$t_{wck_toggle_post}$	This constraint defines the number of clock cycles after a read or write command data burst completion during which the WCK must remain in the current toggle state. During this time, the dfi_wck_cs signal must also remain stable.
DFITMG6.dfi_twck_toggle_rd	$t_{wck_toggle_rd}$	This constraint defines the number of clock cycles between a read command to when the WCK is transitioned from static to toggle. This timing is applicable when the WCK is synchronized and being transitioned between static and toggle modes.
DFITMG6.dfi_twck_toggle_wr	$t_{wck_toggle_wr}$	This constraint defines the number of clock cycles between a write command to when the WCK is transitioned from static to toggle. This timing is applicable when the WCK is synchronized and being transitioned between static and toggle modes.

4.3.3 WCK Constraints (LPDDR5)

WCK constraints are listed in [Table 4-4](#):

Table 4-4 WCK Constraints (LPDDR5)

Control Register	Description
DRAMSET1TMG24.max_rd_sync	Maximum time between read commands without a new WCK2CK sync start $RL + BL/n_{max} + RD(tWCKPST/tCK)$
DRAMSET1TMG24.max_wr_sync	Maximum time between write commands without a new WCK2CK sync start $WL + BL/n_{max} + RD(tWCKPST/tCK)$

4.4 DDRCTL to PHY Connections

This section shows the diagrams which indicate how DFI sideband signals are connected between DDRCTL and PHY in case of Single DDRC Dual DFI configuration. Those DFI signals which come straight through are not displayed.

Figure 4-2 shows DDRCTL to PHY connections for Command Interface.

Figure 4-2 Command Interface

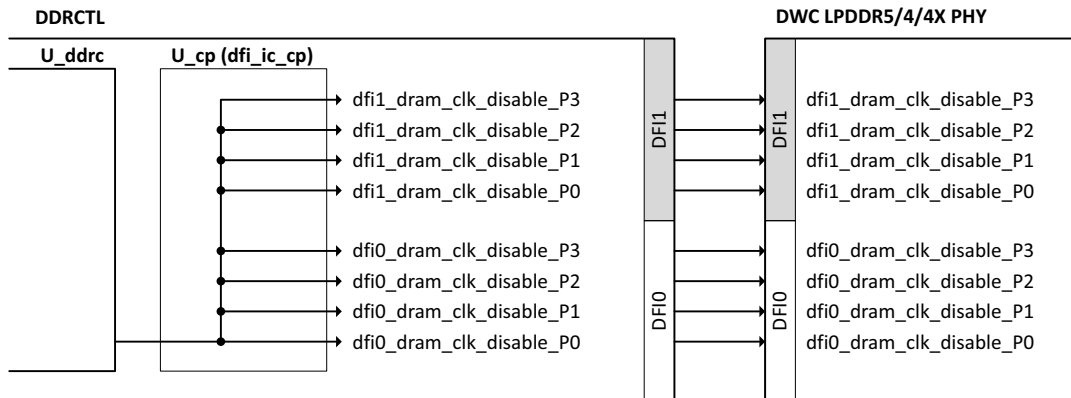
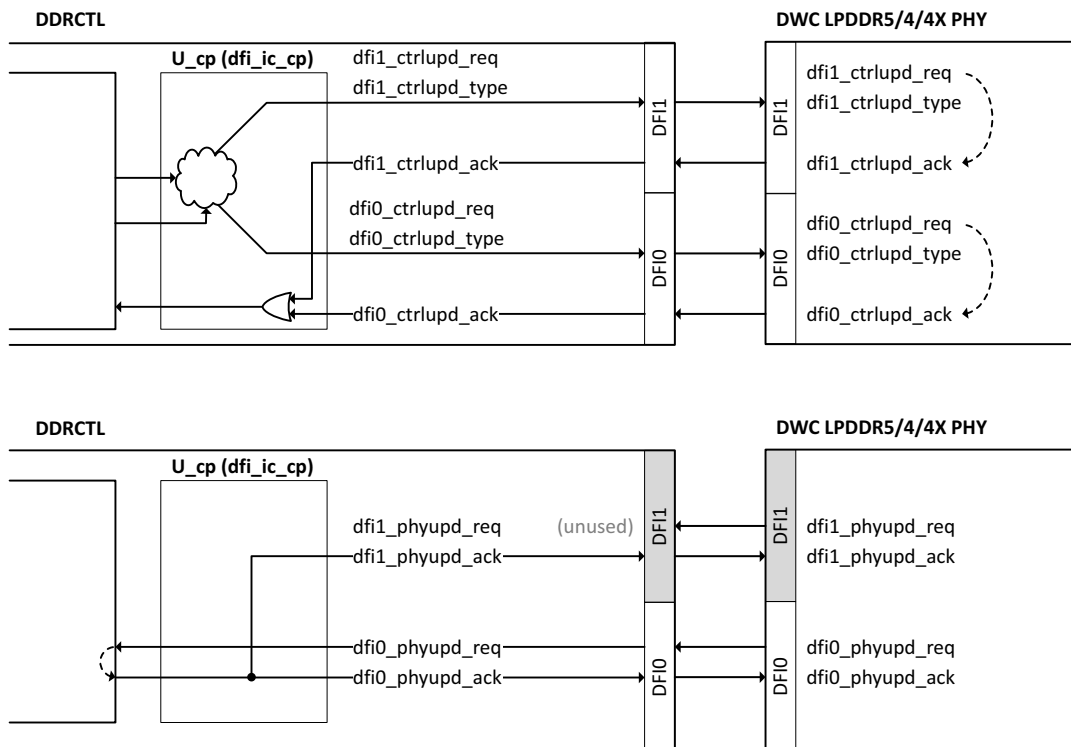


Figure 4-3 shows DDRCTL to PHY connections for Update Interface.

Figure 4-3 Update Interface



Note that when `DDRCTL_PPT2` is not defined or `PPT2CTRL0.ppt2_en` is 0, `dfi1_ctrlupd_ack` is unused in DDRCTL, and `dfi1_ctrlupd_req` is the same as `dfi0_ctrlupd_req`.

Figure 4-4 shows DDRCTL to PHY connections for Status Interface.

Figure 4-4 Status Interface

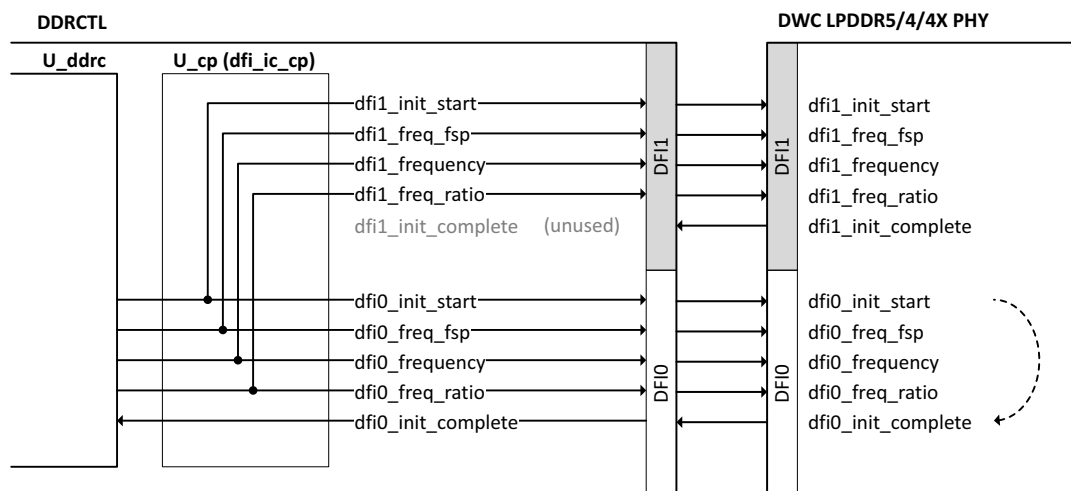


Figure 4-5 shows DDRCTL to PHY connections for low-power control interface.

Figure 4-5 Low-Power Control Interface

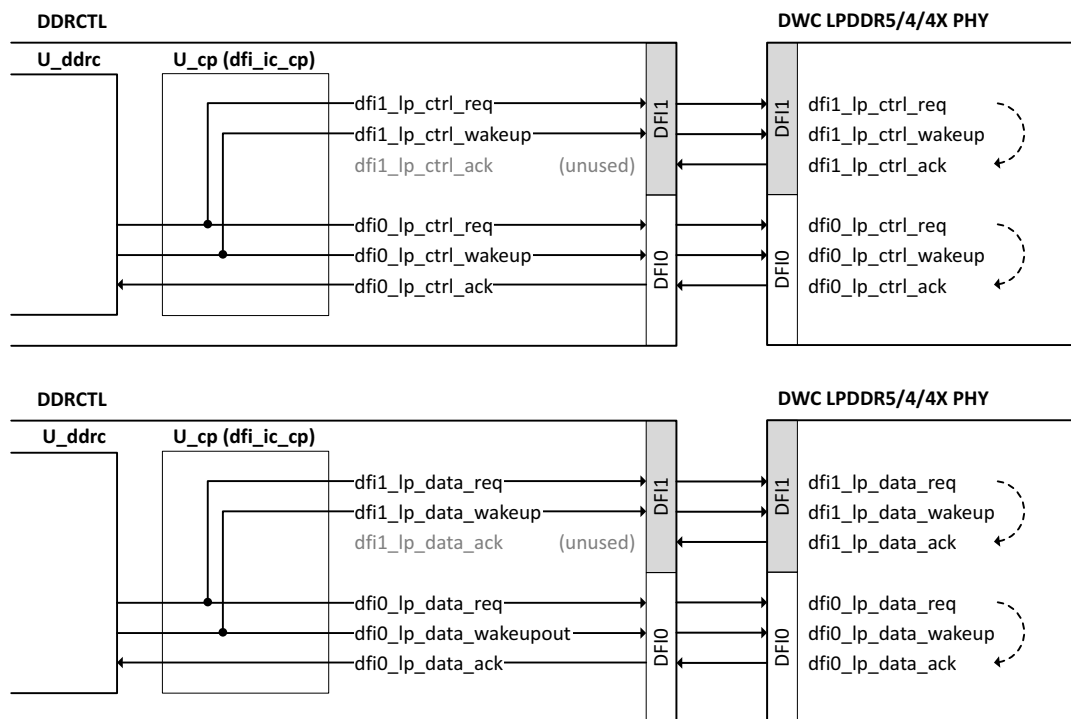


Figure 4-6 shows DDRCTL to PHY connections for PHY Master Interface.

Figure 4-6 PHY Master Interface

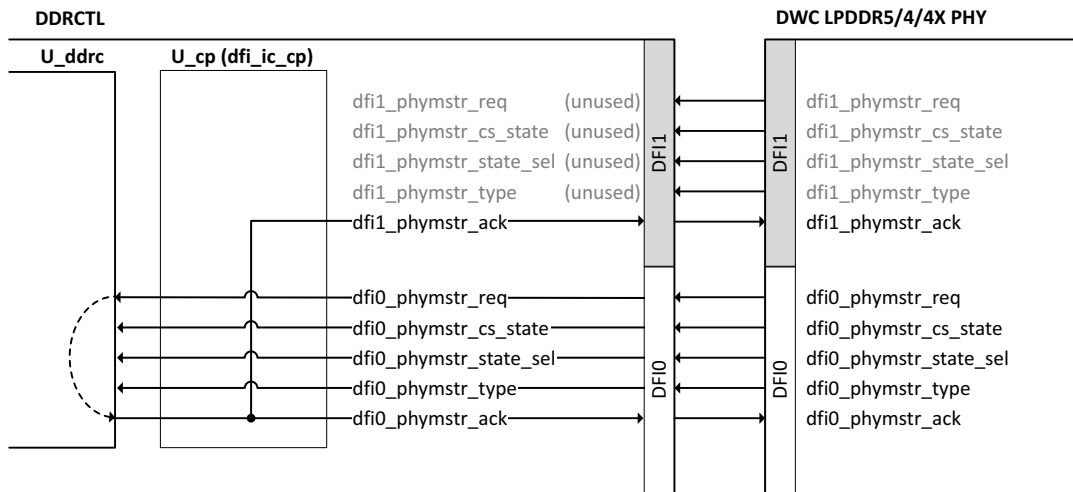
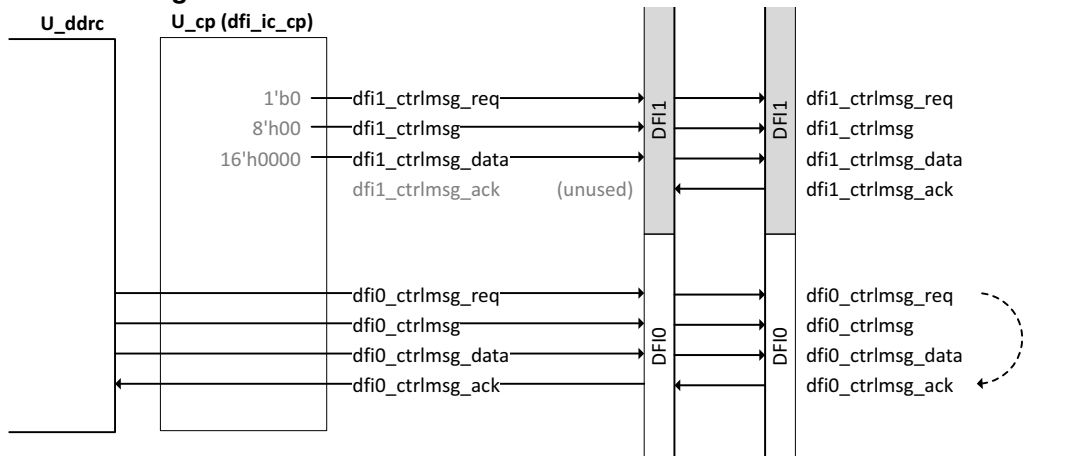


Figure 4-7 shows DDRCTL to PHY connections for MC PHY Message Interface.

Figure 4-7 MC PHY Message Interface



5

Channel Modes

This chapter contains the following sections:

- [“Overview of LPDDR4 / LPDDR5 Channel Modes” on page 102](#)
- [“LPDDR5/4/4X Configurations” on page 103](#)

5.1 Overview of LPDDR4 / LPDDR5 Channel Modes

LPDDR4 SDRAM consists of two independent channels, and Synopsys DWC DDR PHY has two sets of DFI interfaces. LPDDR 5/4 controller also has two sets of DFI interfaces, called "DFI channels" in this document. Each DFI channel is to connect to its respective SDRAM channel (that is, DFI channel 0 connects to SDRAM channel A, DFI channel 1 connects to SDRAM channel B).

In terms of memory/DFI topology, only Single DDRC Single DFI configuration and Single DDRC Dual DFI configuration are supported, as shown in [Figure 5-1](#) on page [104](#) and [Figure 5-4](#) on page [107](#).

5.2 LPDDR5/4/4X Configurations

The LPDDR5/4/4X controller currently supports the following hardware configurations:

1. Single DDRC Single DFI configuration
2. Single DDRC Dual DFI configuration

[Table 5-1](#) shows the relationship between `MEMC_DRAM_DATA_WIDTH` and `UMCTL2_NUM_DFI`.

Table 5-1 Relationship Between `MEMC_DRAM_DATA_WIDTH` and `UMCTL2_NUM_DFI`

<code>MEMC_DRAM_DATA_WIDTH</code>	<code>UMCTL2_NUM_DFI</code>	Notes
16	1	Single DDRC Single DFI configuration
32	2	Single DDRC Dual DFI configuration

**Note**

The hardware parameter `UMCTL2_NUM_DFI` is automatically selected.

This section contains the following subsections:

- [“Single DDRC Single DFI Configuration”](#) on page 104
- [“Single DDRC Dual DFI Configuration”](#) on page 107

5.2.1 Single DDRC Single DFI Configuration

Figure 5-1 shows the single DDRC, single DFI Configuration for 1:4/1:1:4 frequency ratio mode.

Figure 5-1 Single DDRC Single DFI Configuration for 1:4/1:1:4 Frequency Ratio Mode

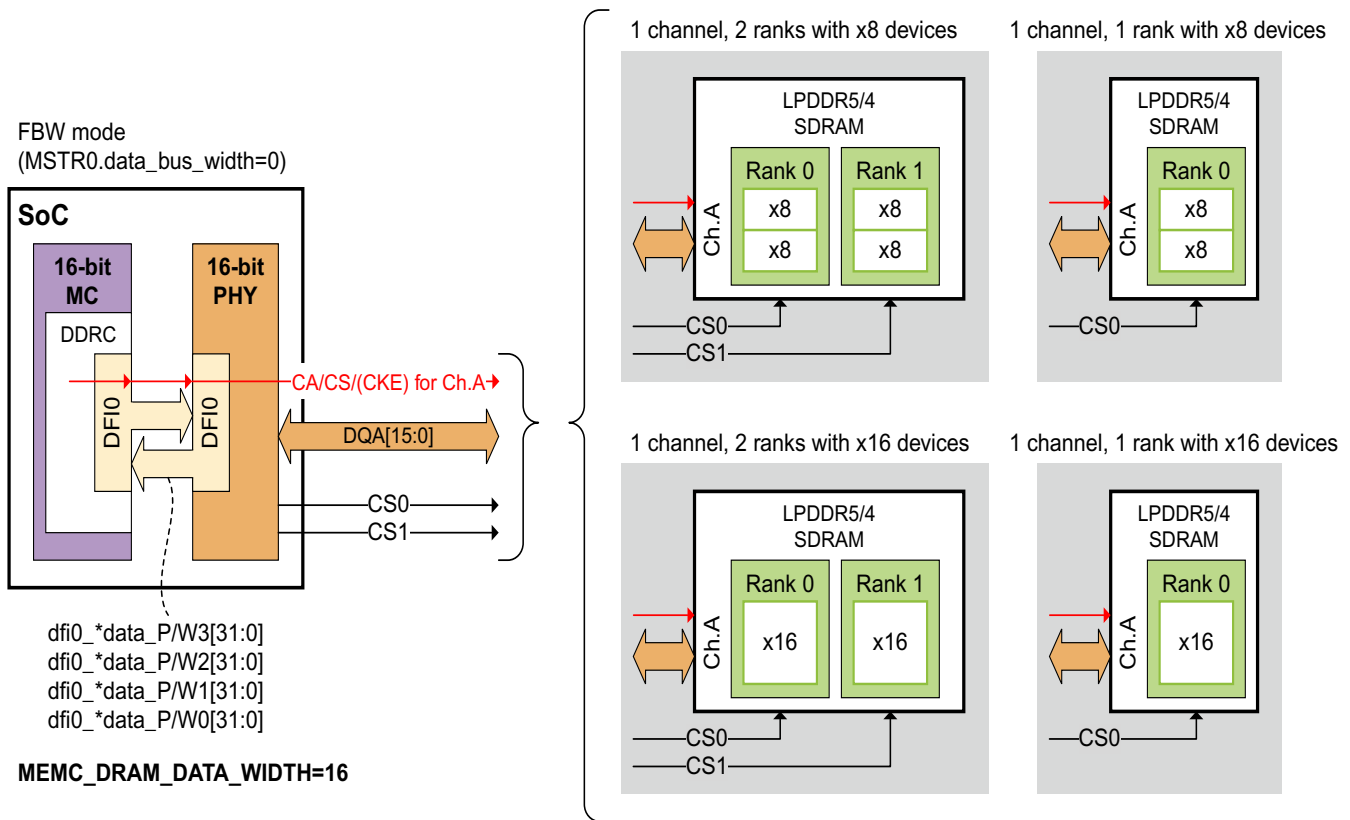
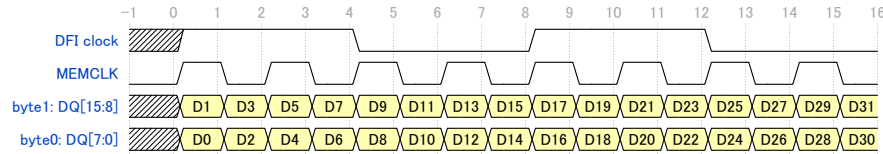


Figure 5-2 shows the DFI Clocks/Data of single DDRC, single DFI configuration for 1:4/1:1:4 frequency ratio mode.

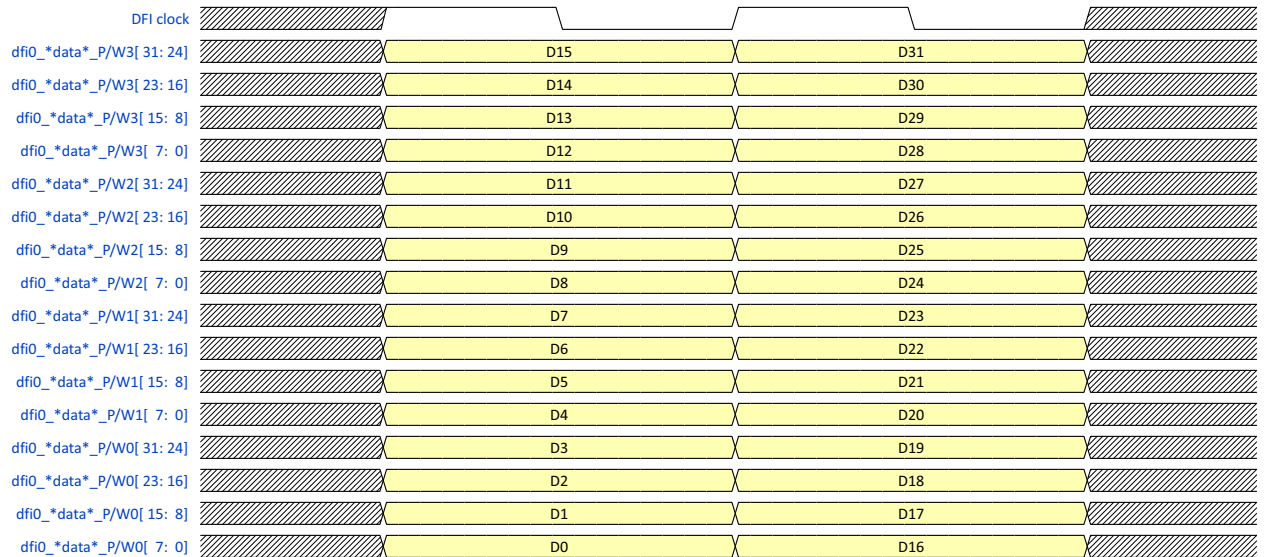
Figure 5-2 DFI Clocks/Data of Single DDRC Single DFI Configuration for 1:4/1:1:4 Frequency Ratio Mode

TMGCFG.frequency_ratio=1 (DFI 1:4/1:1:4 frequency ratio mode)
MSTR0.data_bus_width=0 (Full Bus Width mode)

SDRAM



DFI



HIF

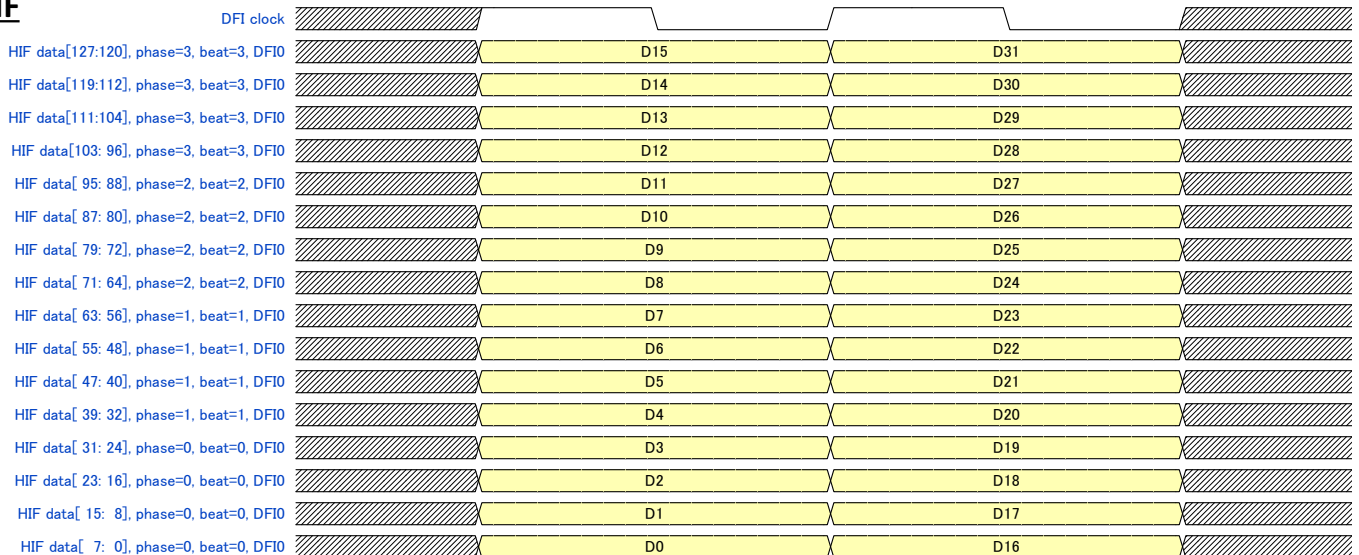
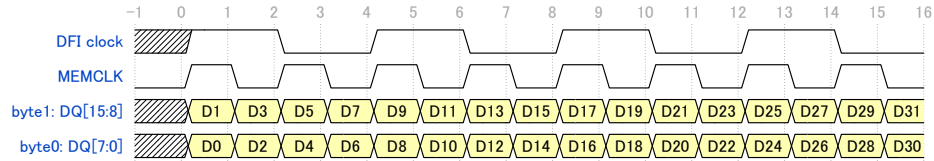


Figure 5-3 shows the DFI clocks of single DDRC, single DFI configuration for 1:2/1:1:2 frequency ratio mode.

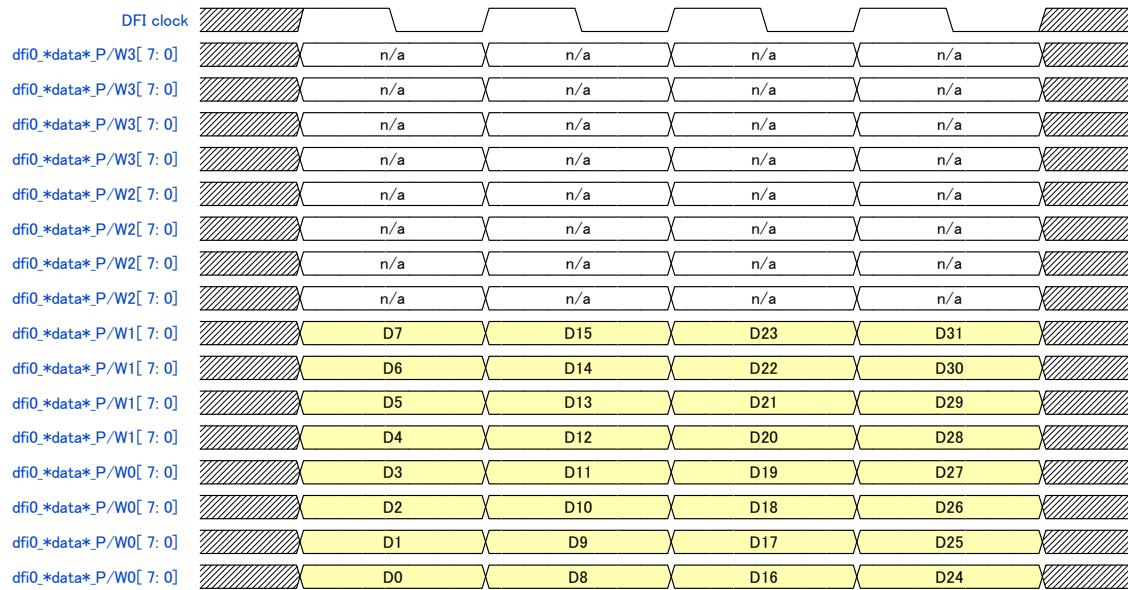
Figure 5-3 DFI Clocks/Data of Single DDRC Single DFI Configuration (1:2/1:1:2 Frequency Ratio Mode)

TMGCFG.frequency_ratio=0 (DFI 1:2/1:1:2 frequency ratio mode)
MSTR0.data_bus_width=0 (Full Bus Width mode)

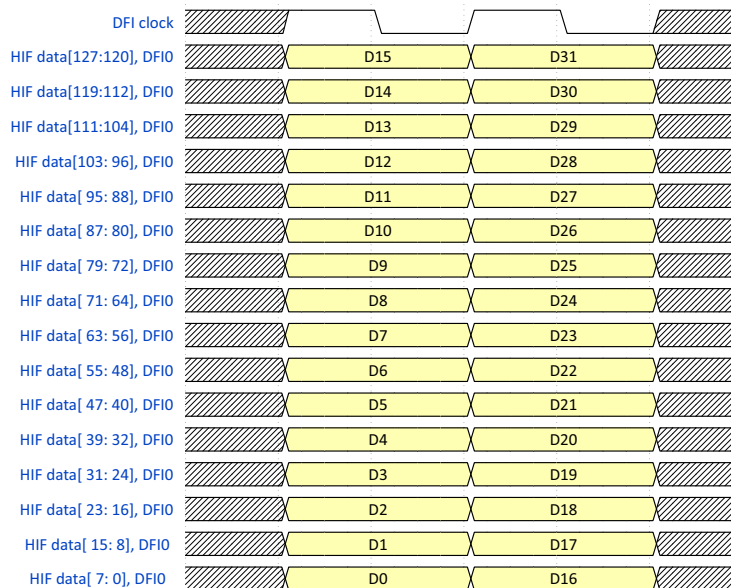
SDRAM



DFI



HIF



**Note**

For HIF write data (hif_wdata), "n/a" shown in the diagrams needs to be set to '0'. That is, invalid data of hif_wdata have to be set to '0'.

For HIF write/read data, diagrams show continuous beat, but invalid beat can be inserted between beats (non-back-to-back).

5.2.2 Single DDRC Dual DFI Configuration

5.2.2.1 FBW (Full Bus Width) Mode

Figure 5-4 shows the Single DDRC, Dual DFI configuration for 1:4/1:1:4 frequency ratio mode.

Figure 5-4 Single DDRC Dual DFI Configuration for 1:4/1:1:4 Frequency Ratio Mode

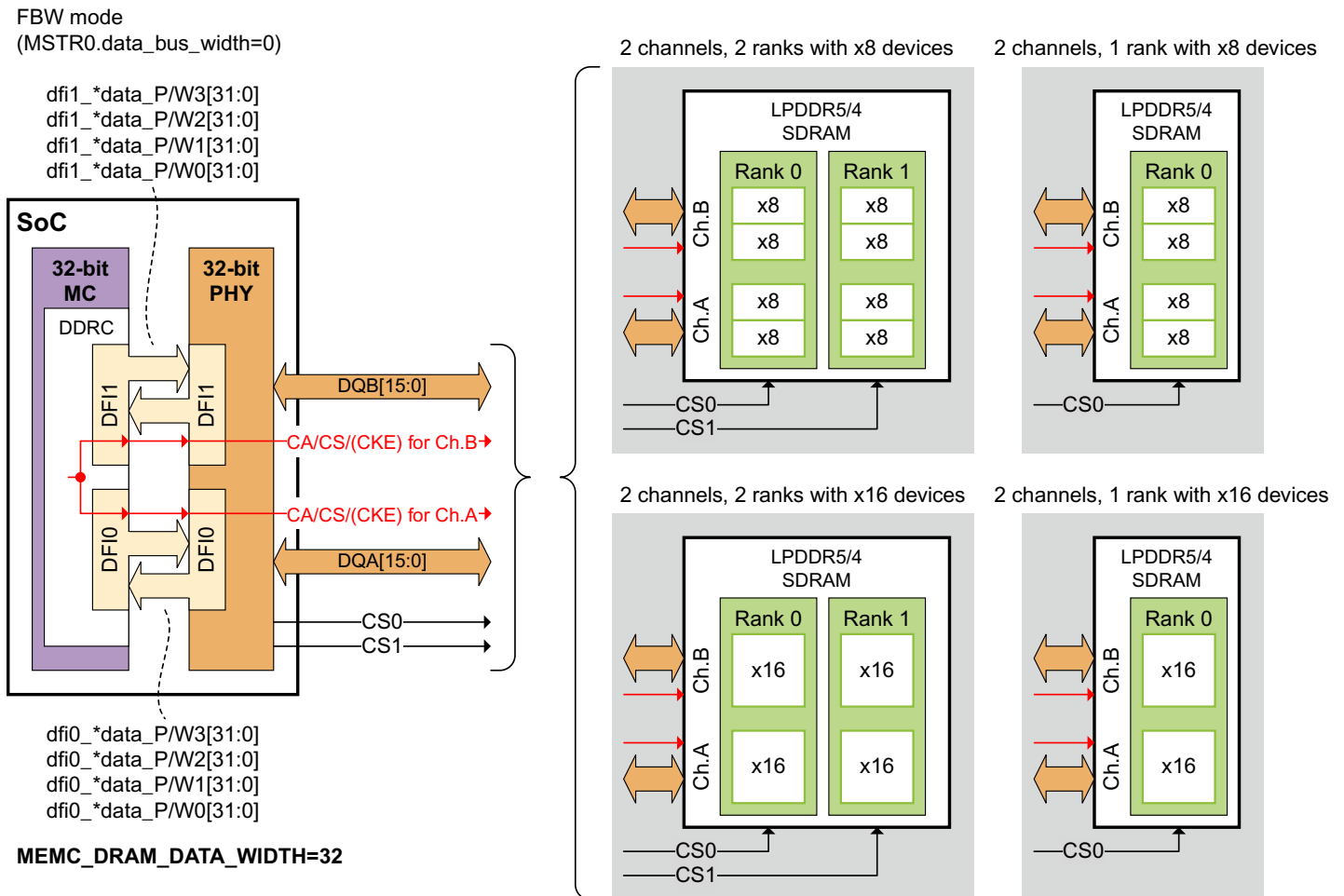


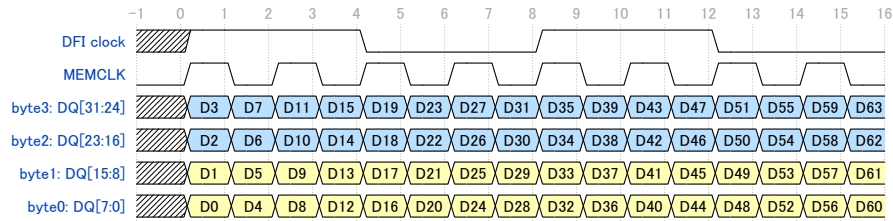
Figure 5-5 shows the DFI clocks/data of Single DDRC, Dual DFI configuration for 1:4/1:1:4 frequency ratio mode.

Figure 5-5 DFI Clocks/Data of Single DDRC Dual DFI Configuration for 1:4/1:1:4 Frequency Ratio Mode

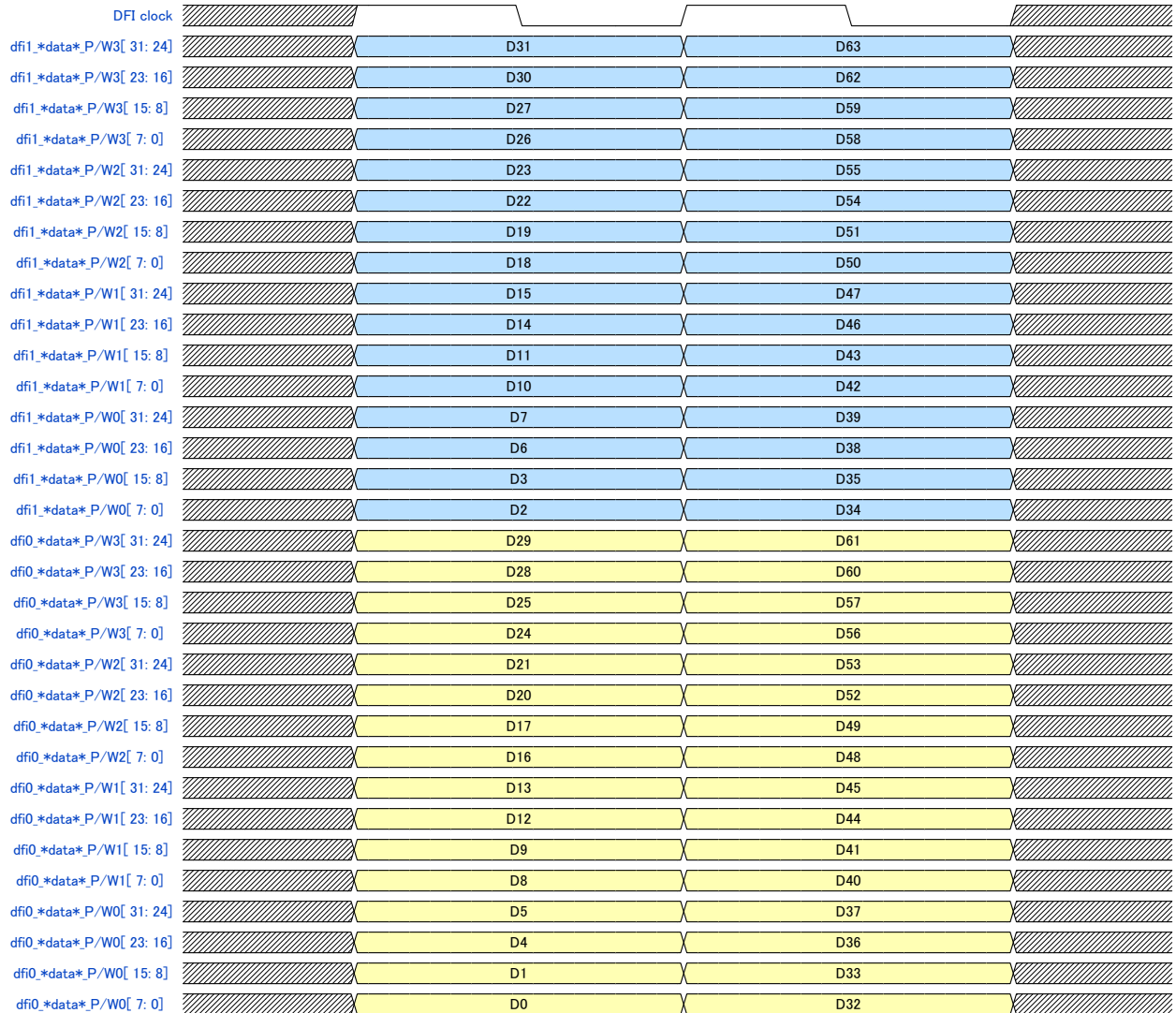
TMGCFG.frequency_ratio=1 (DFI 1:4/1:1:4 frequency ratio mode)

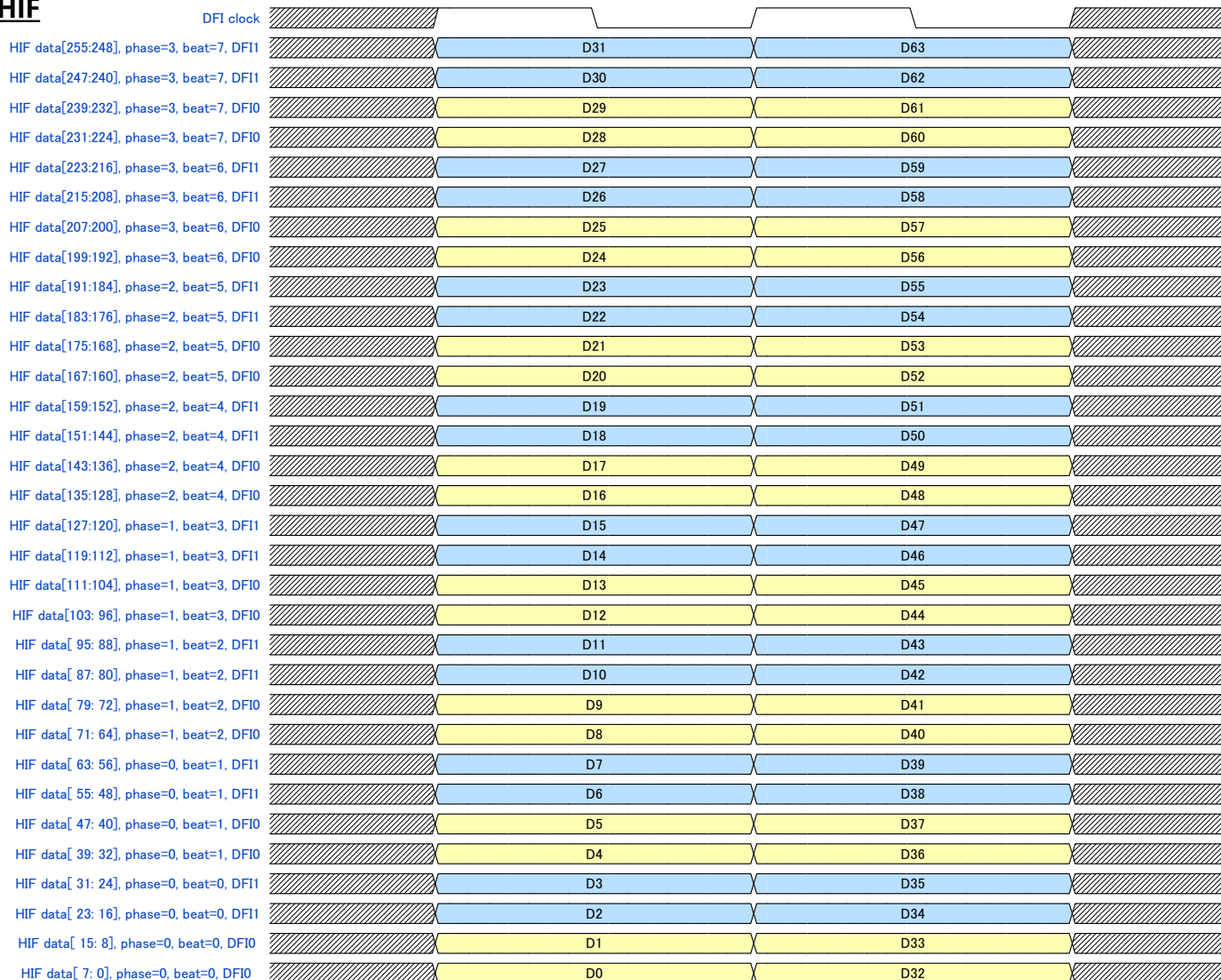
MSTR0.data_bus_width=0 (Full Bus Width mode)

SDRAM



DFI



HIF**Note**

For HIF write data (hif_wdata), "n/a" shown in the diagrams needs to be set to '0'. That is, invalid data of hif_wdata have to be set to '0'.

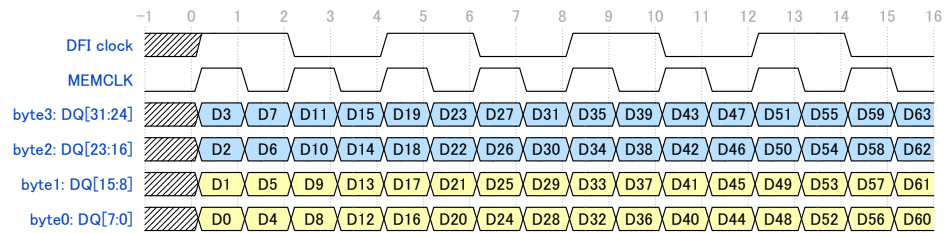
For HIF write/read data, diagrams show continuous beat, but invalid beat can be inserted between beats (non-back-to-back).

Figure 5-6 shows the DFI clocks/data of Single DDRC, Dual DFI configuration for 1:2/1:1:2 frequency ratio mode.

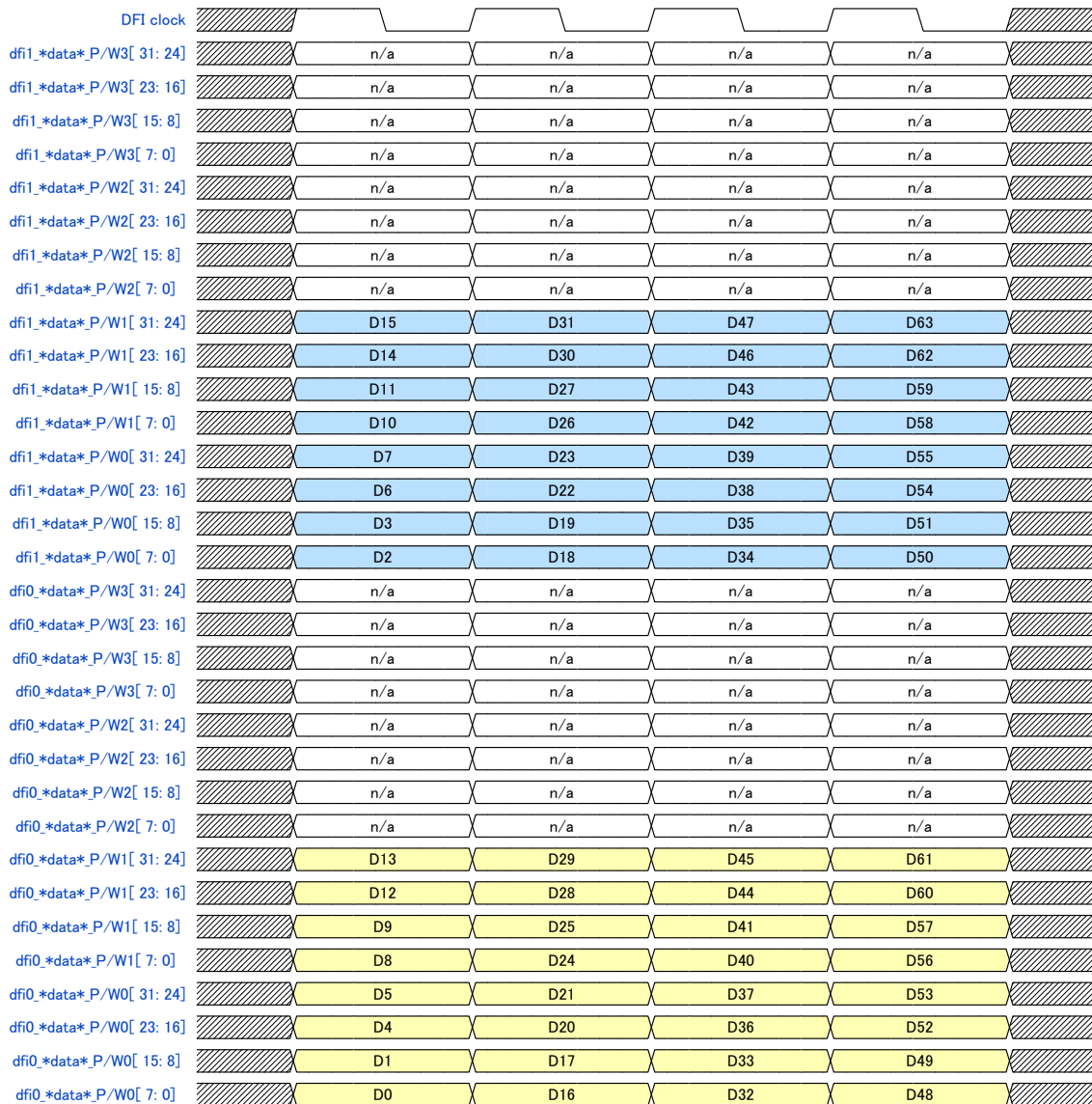
Figure 5-6 DFI Clocks/Data of Single DDRC Dual DFI Configuration (1:2/1:1:2 Frequency Ratio Mode)

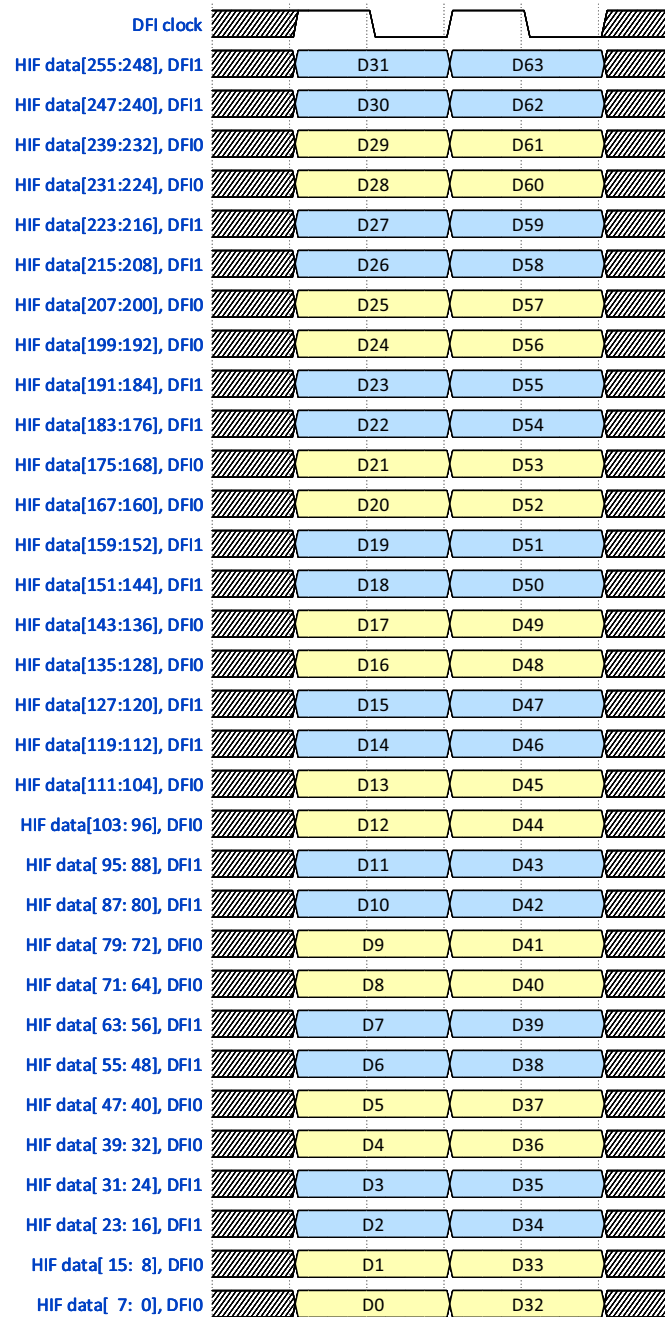
TMGCFG.frequency_ratio=0 (DFI 1:2/1:1:2 frequency ratio mode)
MSTR0.data_bus_width=0 (Full Bus Width mode)

SDRAM



DFI



HIF**Note**

For HIF write data (hif_wdata), "n/a" shown in the diagrams needs to be set to '0'. That is, invalid data of hif_wdata have to be set to '0'.

For HIF write/read data, diagrams show continuous beat, but invalid beat can be inserted between beats (non-back-to-back).

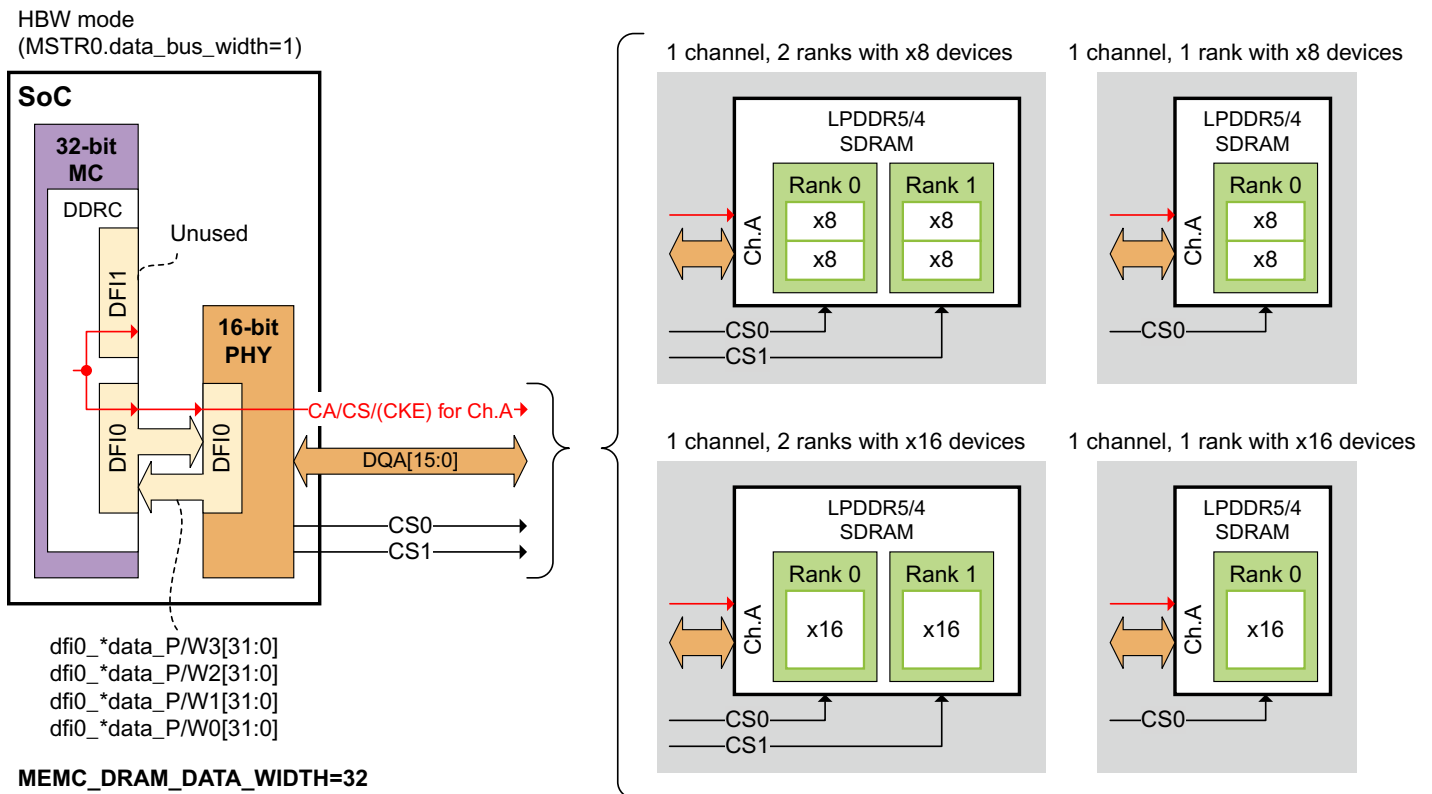
In Single DDRC Dual DFI configuration in FBW mode, `MEMC_DRAM_DATA_WIDTH` must be set to the total data width. Otherwise, `MEMC_DRAM_DATA_WIDTH` must be set to the data width per channel.

5.2.2.2 HBW (Half Bus Width) Mode

LPDDR5/4/4X Controller basically operates only in FBW mode (`MSTR0.data_bus_width=0`). It can also work in HBW mode (`MSTR0.data_bus_width=1`) if a 16-bit PHY (which has DFI0 only) is connected. In this case, the relevant signals for DFI1 in the controller are not connected to the PHY.

Figure 5-7 shows a Single DDRC Dual DFI configuration (32-bit) with 16-bit PHY.

Figure 5-7 Single DDRC Dual DFI Configuration (32-bit) With 16-bit PHY



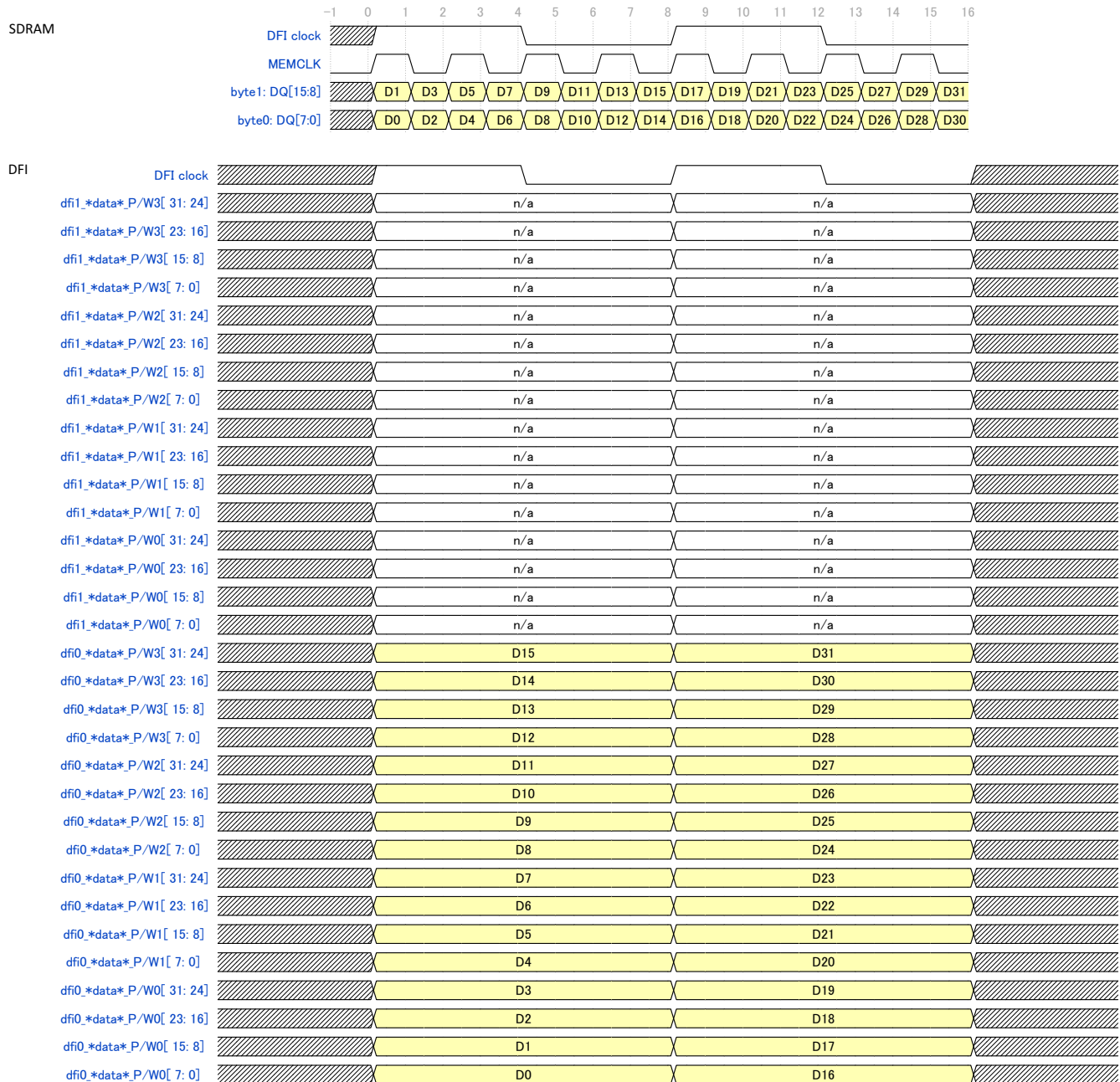
Note

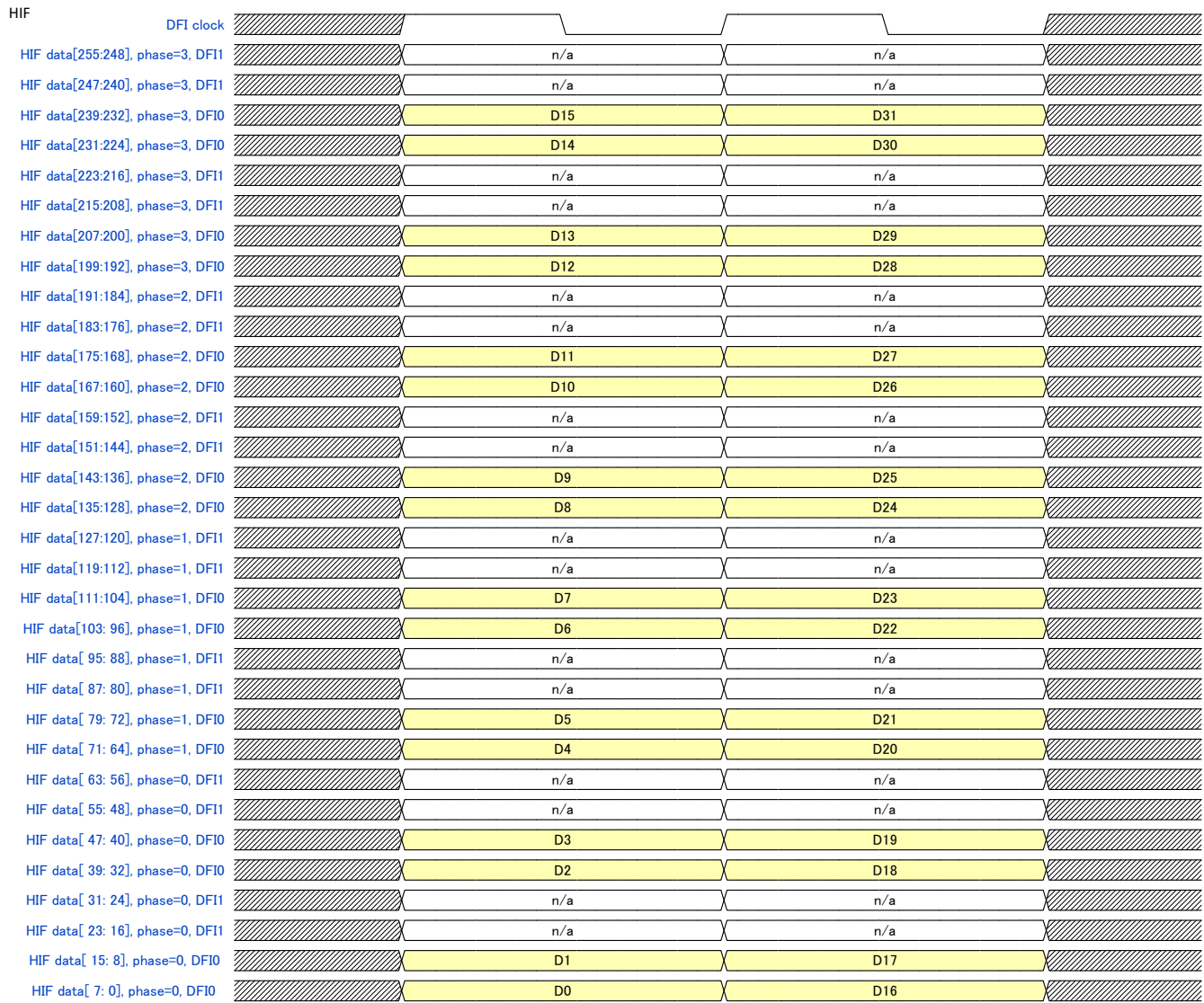
HBW mode is not supported with 32-bit PHY and Single DDRC Dual DFI configuration (32-bit). If you want to use HBW mode, contact Synopsys Customer Support.

Figure 5-8 shows the DFI clocks/data of Single DDRC Dual DFI configuration with 16-bit PHY for 1:4/1:1:4 frequency ratio mode.

Figure 5-8 DFI Clocks/Data of Single DDRC Dual DFI Configuration with 16-bit PHY (1:4/1:1:4 Frequency Ratio Mode)

TMGCFG.frequency_ratio=1 (DFI 1:4/1:1:4 frequency ratio mode)
MSTR0.data_bus_width=1 (Half Bus Width mode)



**Note**

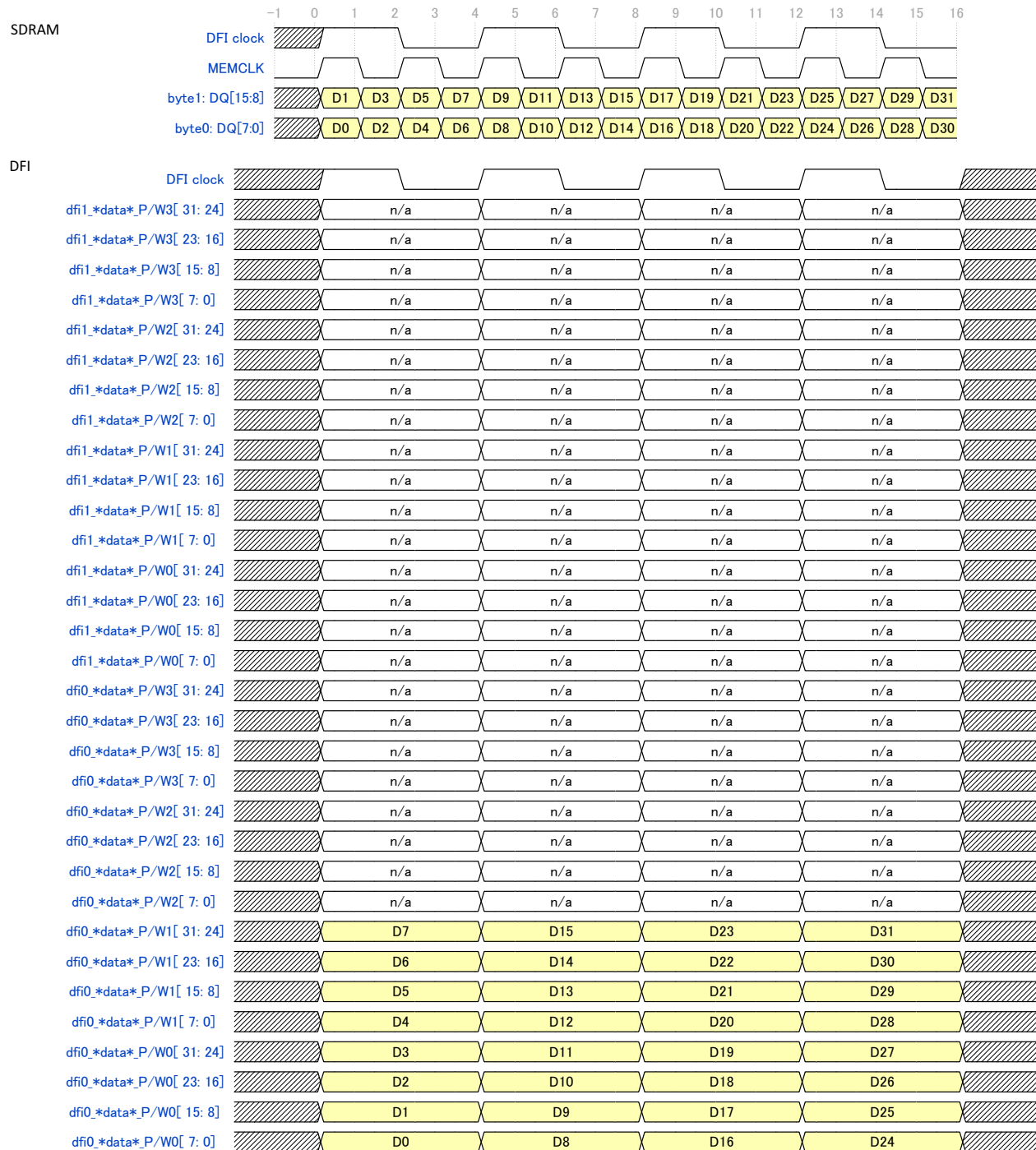
For HIF write data (hif_wdata), "n/a" shown in the diagrams needs to be set to '0'. That is, invalid data of hif_wdata have to be set to '0'.

For HIF write/read data, diagrams show continuous beat, but invalid beat can be inserted between beats (non-back-to-back).

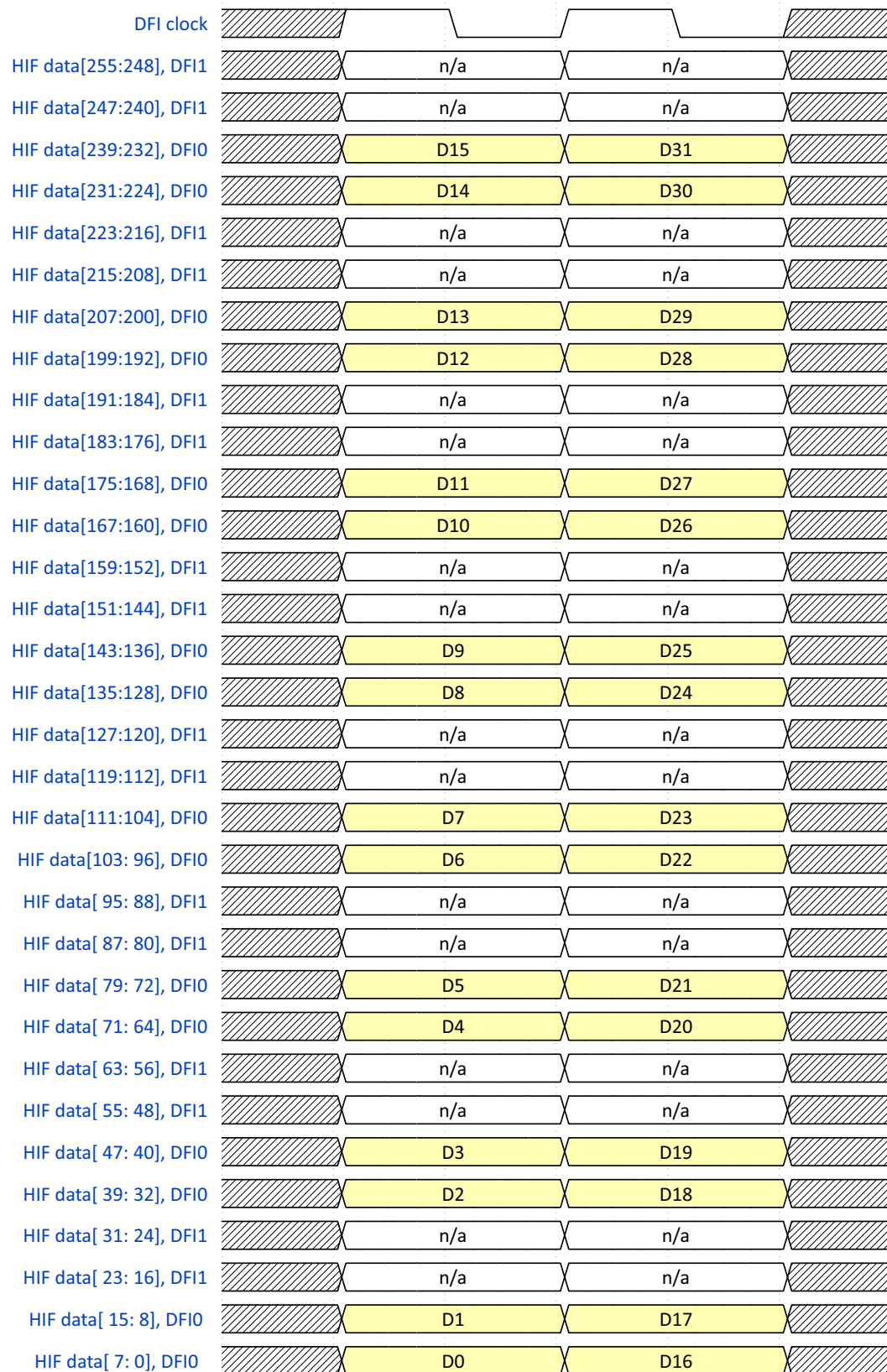
Figure 5-9 shows the DFI clocks/data of Single DDRC Dual DFI configuration with 16-bit PHY for 1:2/1:1:2 frequency ratio mode.

Figure 5-9 DFI Clocks/Data of Single DDRC Dual DFI Configuration with 16-bit PHY (1:2/1:1:2 Frequency Ratio Mode)

TMGCFG.frequency_ratio=0 (DFI 1:2/1:1:2 frequency ratio mode)
MSTR0.data_bus_width=1 (Half Bus Width mode)



HIF





For HIF write data (hif_wdata), "n/a" shown in the diagrams needs to be set to '0'. That is, invalid data of hif_wdata have to be set to '0'.

For HIF write/read data, diagrams show continuous beat, but invalid beat can be inserted between beats (non-back-to-back).

6

Address Mapping

This chapter contains the following sections:

- [“Overview of Address Mapping” on page 120](#)
- [“Application to HIF Address Mapping” on page 123](#)
- [“HIF Address to SDRAM Address Mapping” on page 124](#)
- [“Recommendations for Optimum SDRAM Utilization” on page 126](#)
- [“Non-binary Device Densities” on page 128](#)
- [“Registers Related to Address Mapper” on page 129](#)

6.1 Overview of Address Mapping

Read and write requests are provided to the DDRCTL with a system address. The system address is the command address of a transaction as presented on one of the data ports. The DDRCTL converts this system address to a physical address. It maps the system address to the SDRAM rank, bank group, bank, row, and column addresses.

If the system address regions are enabled ($UMCTL2_A_NSAR > 0$), the first part of the mapping is conversion of system address to AXI¹ byte address. The controller maps disjoint address regions to internal consecutive addresses. Otherwise, the controller assumes that the DRAM is always mapped as a monolithic block. For more information about this, see “[System Address Regions](#)” on page 120.

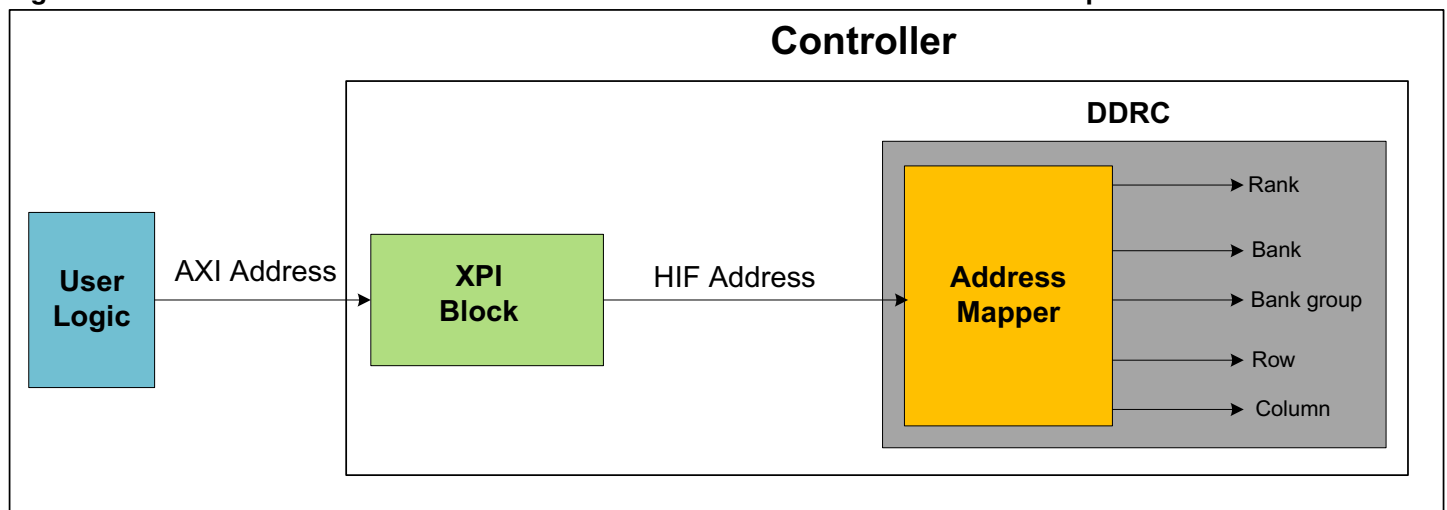
The second part of this mapping is conversion of AXI byte address to HIF word address. This is performed in the XPI block, and is applicable to AXI configurations only. For more information about this, see “[Application to HIF Address Mapping](#)” on page 123.

The last part is the conversion of HIF word address to SDRAM address. A flexible address mapper maps the HIF word address to the SDRAM rank/bank/bank group/row/ column address. This address mapper is located within the DDRC.

The address mapping to be used depends on the use-case. The DDRCTL provides a set of registers that allows flexible re-programming of logical to physical address mapping. For more information about this, see “[HIF Address to SDRAM Address Mapping](#)” on page 124.

[Figure 6-1](#) explains the conversion of AXI to HIF to the SDRAM rank/bank/row/column/bank group addresses.

Figure 6-1 Conversion of AXI to HIF to SDRAM Rank/Bank/Row/Column/Bank Group Addresses



6.1.1 System Address Regions

The system address regions add the capability to define up to four disjoint memory regions mapping to the DRAM as consecutive addresses. The number of regions and minimum block size are determined by the hardware parameters $UMCTL2_A_NSAR$ and $UMCTL2_SARMINSIZE$.

Each region is defined by:

- Base Address: Starting address of the region aligned to the minimum block size.

1. System Address Regions are only supported with AXI configurations.

- Number of Blocks: Size of the region in multiples of minimum block size.

The base address of each region is specified by the register `SARBASEn.base_addr` and the total number of blocks is specified by the register `SARSIZEn.nblocks` (see “REGB_ARB_PORTp” in “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide).

System address region specification is the same for all ports. Error response is not generated by the controller for addresses falling outside the specified address regions and the same address translation is applied from one base address to the next base address.

The base addresses must be specified such that they are in the ascending order ($SARBASE0 < SARBASE1 < SARBASE2 < SARBASE3$) and such that regions do not overlap. It is assumed that an outside agent can generate address decode errors, if they happen to occur.

6.1.1.1 System Address Regions Example

Consider the following example: the controller is set to have three system address regions (as specified on [Table 6-1](#)), with a minimum block size of 2 Gb. Therefore, the `UMCTL2_SARMINSIZE` parameter must be set to 4.

The system address bits `[UMCTL2_A_ADDRW-1:31]` is used to represent the base addresses in the multiples of the minimum block size.

The registers must be programmed as follows:

- `SARBASE0.base_addr = 1` (see “REGB_ARB_PORTp” in “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide)
- `SARBASE1.base_addr = 17`
- `SARBASE2.base_addr = 272`
- `SARSIZE0.nblocks = 0`
- `SARSIZE1.nblocks = 14`
- `SARSIZE2.nblocks = 15`

The specifications in [Table 6-1](#) translates to the register settings mentioned in [Table 6-2](#).

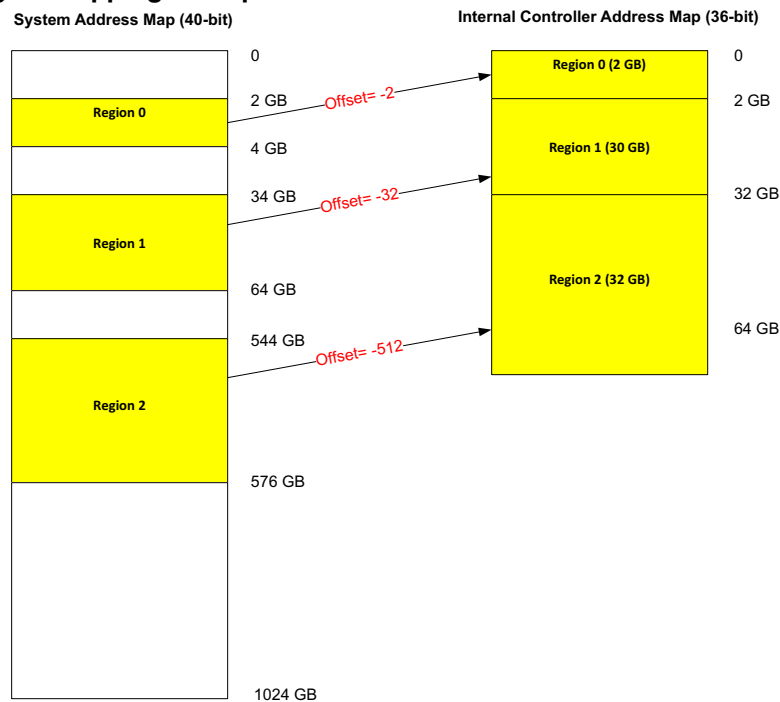
Table 6-1 Address Region Mapping Example - Specification

Region	System Address (40-bit)	Offset	Internal Controller Address (36-bit)
Region 0 (2 GB)	0x00 8000 0000–0x00 FFFF FFFF	0x00 8000 0000	0x00 0000 0000–0x00 7FFF FFFF
Region 1 (30 GB)	0x08 8000 0000–0x0F FFFF FFFF	0x08 0000 0000	0x00 8000 0000–0x07 FFFF FFFF
Region 2 (32 GB)	0x88 0000 0000–0x8F FFFF FFFF	0x80 0000 0000	0x08 0000 0000–0x0F FFFF FFFF

Table 6-2 Example Base Addresses, Number of Blocks and Offset

Region	Base Address (Binary)	Register Value (base_addr)	Register Value (nblocks)	Calculated Base Address	Calculated Offset (Decimal/GB)
Region 0	9'b0_0000_0001	1	0	2 GB	1 / 2 GB
Region 1	9'b0_0001_0001	17	14	34 GB	16 / 32 GB
Region 2	9'b1_0001_0000	272	15	544 GB	256 / 512 GB

Figure 6-2 shows the graphical illustration of the system address regions as specified in Table 6-1.

Figure 6-2 Address Region Mapping Example**Note**

When `UMCTL2_AXI_BOUNDARY > UMCTL2_SARMINSIZE + 27`, portion of the address below the boundary can be changed by the conversion from system to AXI address. In this case, you must ensure that the converted AXI address does not cross the boundary.

6.2 Application to HIF Address Mapping

The {ARADDR | AWADDR | HADDR} is a byte address. Based on the MEMC_BURST_LENGTH, MEMC_FREQ_RATIO MEMC_DRAM_DATA_WIDTH, and MSTR0.data_bus_width, XPI maps the MSB bits of the application address to the HIF address (hif_cmd_addr) in the following ways:

- $\text{hif_cmd_addr}[\text{40}:\text{log2}(\text{MEMC_BURST_LENGTH})] = \{\text{ARADDR} \mid \text{AWADDR} \mid \text{HADDR}\} [\text{UMCTL2_A_ADDRW}-1:\text{log2}(\text{MEMC_BURST_LENGTH})+\text{log2}(\text{MEMC_DRAM_DATA_WIDTH}/(8*2^{\text{MSTR0.data_bus_width}}))]$
- $\text{hif_cmd_addr}[\text{log2}(\text{MEMC_BURST_LENGTH})-1:(\text{MEMC_FREQ_RATIO}-1)]$ is internally generated by XPI
- $\text{hif_cmd_addr} [\text{MEMC_FREQ_RATIO}-2:0] = 0$

6.3 HIF Address to SDRAM Address Mapping

The address mapper maps HIF word addresses to SDRAM addresses by selecting the HIF address bit that maps to each and every applicable SDRAM address bit. While it is possible to map HIF address bits to a SDRAM address in any desired manner, the full available address space is accessible only when no two SDRAM address bits are determined by the same HIF address bit. Each SDRAM address bit has an associated register vector to determine its source.

Registers ADDRMAP_x (x = 0 to 11) are used to program the address mapper. For more information on ADDRMAP registers, see REGB_ADDR_MAP0 in the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

The HIF address bit number is determined by adding the internal base of the ADDRMAP_x (x = 0 to 11) register to the programmed value for that register, as described in the following equation:

$$\text{HIF address bit number} = [\text{internal base}] + [\text{register value}]$$

For example, for ADDRMAP5.addrmap_col_b7, the internal base is 7. When full data bus is in use, column bit 7 is determined by the following equation:

$$7 + [\text{register value}]$$

If this register is programmed to 2, the HIF address bit can be calculated by using following equation:

$$[\text{HIF address bit number}] = 7 + 2 = 9$$

In other words, the column address bit 7 sent to SDRAM would always be equal to hif_cmd_addr[9] of the corresponding HIF source address.

The system address to physical address mapping can be done by choosing any one of the possible combinations from [Figure 6-3](#) on page 125. It explains the different possible ways to map the HIF address bits to the SDRAM rank/bank/bank group/row/column address.



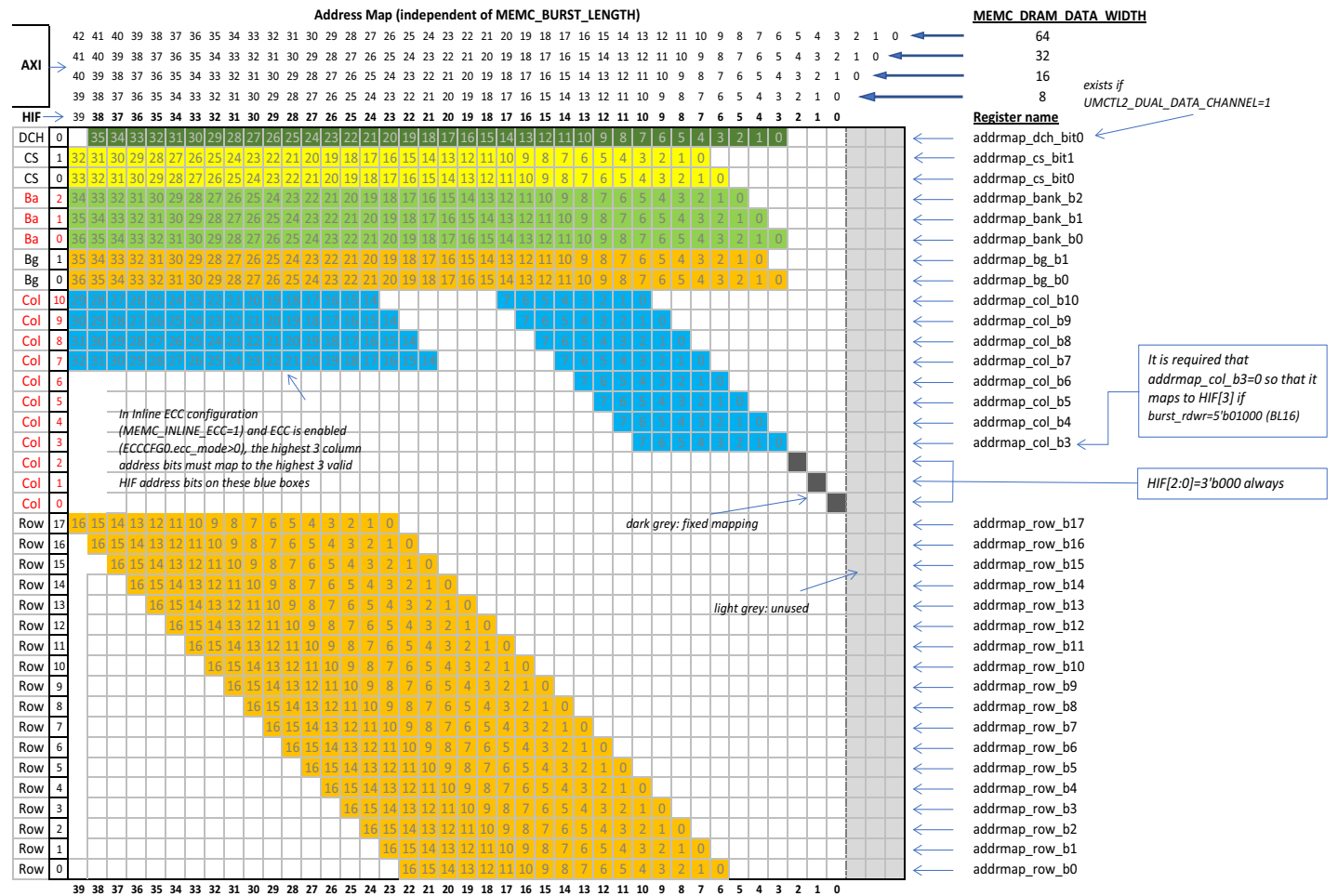
Note

- COL[3:0] == HIF[3:0] must be true for both LPDDR4 and LPDDR5 protocols.
- burst_rdwrr == 5'b01000 must be true for both LPDDR4 and LPDDR5 protocols.
- Register constraints are independent of ARB or HIF configuration and bus width mode.
- SDRAM column address [2:0] does not exist and hence is not addressable. If UMCTL2_INCL_ARB == 1, this is automatically the case. The HIF master must generate hif_cmd_addr[2:0] = 000.
- For any address bits which cannot be in use in all cases, all bits of the associated address map register field must be set to '1' when the associated SDRAM address bit is not in use.



Note

Ensure that no two SDRAM address bits are determined by the same HIF address bit.

Figure 6-3 System Address to Physical Address Mapping (Independent of MEMC_BURST_LENGTH)

6.4 Recommendations for Optimum SDRAM Utilization

Write or Masked Write to Masked Write have a t_{CCDMW} penalty instead of the minimum t_{CCD} . There is no performance hit for reads as t_{CCDMW} is not related to them. These performance hits can be minimized/avoided by using address mapping recommendations in arbiter configurations and traffic recommendations in HIF configurations, as outlined in [Table 6-3](#) and [Table 6-4](#).

It is recommended to perform bank or bank group rotation between back to back SDRAM commands. For LPDDR4, t_{CCDMW} is equal to $4 * t_{CCD}$, so two bank bits need to be rotated. For LPDDR5, t_{CCDMW} in the case of BG mode is equal to $8 * t_{CCD}$, so two bank group bits and one bank bit need to be rotated.

Description of terms used in the columns of the tables are as follows:

- In the “HIF behavior” and “HIF burst size” columns:
 - “Full writes, reads, RMWs” corresponds to writes/RMWs with HIF burst size equal to $MEMC_BURST_LENGTH / (MEMC_FREQ_RATIO * 2)$ in write data channel, and to reads with $hif_cmd_length = 2'b00$.
 - “Half writes, reads, RMWs” corresponds to writes/RMWs with HIF burst size equal to $MEMC_BURST_LENGTH / (MEMC_FREQ_RATIO * 4)$ in write data channel, and to reads with $hif_cmd_length = 2'b10$.
- In the “Performance” column:
 - “Data of each CAM entry used” refers to the data size stored in each CAM entry.
 - Full means that each entry stores data corresponding to $MEMC_BURST_LENGTH$ size.
 - Half means that each entry stores data corresponding to $MEMC_BURST_LENGTH / 2$ size.

For setting the Address Map register, `ADDRMAP6.addrmap_col_b3` must be set to '0'.

Table 6-3 Address Map Recommendations for Arbiter Configurations

Address Map	HIF Behavior	Performance
LPDDR5		
<code>addrmap_col_b3 = 0 (HIF[3])</code> <code>addrmap_bg_b0 = 1 (HIF[4])</code> <code>addrmap_bg_b1 = 1 (HIF[5])</code> <code>addrmap_bank_b0 = 3 (HIF[6])</code>	Full writes, reads, RMWs	SDRAM utilization: full Data of each CAM entry used: full
LPDDR4		
<code>addrmap_col_b3 = 0 (HIF[3])</code> <code>addrmap_bank_b0 = 1 (HIF[4])</code> <code>addrmap_bank_b1 = 1 (HIF[5])</code>	Full writes, reads, RMWs	SDRAM utilization: full Data of each CAM entry used: full

Table 6-4 Address Map Recommendations for HIF Configurations

Traffic	Maximum HIF Burst Size	Performance
LPDDR5		

Traffic	Maximum HIF Burst Size	Performance
Rotate 2 bank group bits and 1 bank bit between HIF commands	Full writes, reads, RMWs	SDRAM utilization: full Data of each CAM entry used: full
LPDDR4		
Rotate 2 bank bits between HIF commands	Full writes, reads, RMWs	SDRAM utilization: full Data of each CAM entry used: full

**Note**

If Mask write is not predominant in the traffic, different address map might be fine to the SDRAM utilization.

6.5 Non-binary Device Densities

LPDDR4 3Gb/6Gb/12Gb per channel devices, LPDDR5 3Gb/6Gb/12Gb/24Gb per channel devices do not support addresses which have both MSB and MSB-1 row bits high. Any attempt to read or write to this address space results in memory misbehavior or system halts. When such devices are used, you must set the register `ADDRMAP12.nonbinary_device_density` accordingly. This prevents the DDRCTL from accessing this address space when there is a read or write request from you/SoC.

Table 6-5 Non-binary Device Densities

		LPDDR4	LPDDR5
3Gb	X16	[R14:R13] != 2'b11	[R13:R12] != 2'b11
3Gb	X8	[R15:R14] != 2'b11	[R14:R13] != 2'b11
6Gb	X16	[R15:R14] != 2'b11	[R14:R13] != 2'b11
6Gb	X8	[R16:R15] != 2'b11	[R15:R14] != 2'b11
12Gb	X16	[R16:R15] != 2'b11	[R15:R14] != 2'b11
12Gb	X8	[R17:R16] != 2'b11	[R16:R15] != 2'b11
24Gb	X16	N/A	[R16:R15] != 2'b11
24Gb	X8	N/A	[R17:R16] != 2'b11
24Gb	X4	N/A	N/A

When `ADDRMAP12.nonbinary_device_density` is set to non-zero, any write or RMW request with `row[MSB:MSB-1] == 2'b11` is discarded, while the HIF output `hif_wdata_ptr_addr_err` is asserted. Also, any read request with `row[MSB:MSB-1] == 2'b11` is executed (spare read) by changing `row[MSB:MSB-1]` from 2'b11 to 2'b10. The return data for such reads are masked to zeros, while the HIF output `hif_rdata_addr_err` is asserted.

For more information, see “[hif_wdata_ptr_addr_err](#)” on page 69 and “[hif_rdata_addr_err](#)” on page 65.

6.6 Registers Related to Address Mapper

The following are the registers related to the Address Mapper:

- ADDRMAP0
- ADDRMAP1
- ADDRMAP3
- ADDRMAP4
- ADDRMAP5
- ADDRMAP6
- ADDRMAP7
- ADDRMAP8
- ADDRMAP9
- ADDRMAP10
- ADDRMAP11
- ADDRMAP12

For more information about these registers, see the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

7

Command Scheduling

This chapter contains the following sections:

- [“Port Arbitration \(PA\)”](#) on page 132
- [“Queue Status Interface and Port Throttling”](#) on page 140
- [“Transaction Service Control”](#) on page 143
- [“Quality of Service”](#) on page 163

7.1 Port Arbitration (PA)

This section covers the following topics:

- [“Overview of Port Arbitration \(PA\)” on page 132](#)
- [“Read/Write Arbitration” on page 134](#)
- [“Port Timeout” on page 136](#)
- [“DDRC Read Priorities \(HPR/LPR/VPR\) and Write Priorities \(NPW/VPW\) for Ports” on page 137](#)
- [“Port Command Priority” on page 137](#)
- [“Round-Robin Arbitration” on page 138](#)
- [“Page Match” on page 138](#)
- [“Write Exclusive Access Lock” on page 139](#)
- [“Signals Related to Port Arbiter” on page 139](#)
- [“Registers Related to Port Arbiter” on page 139](#)

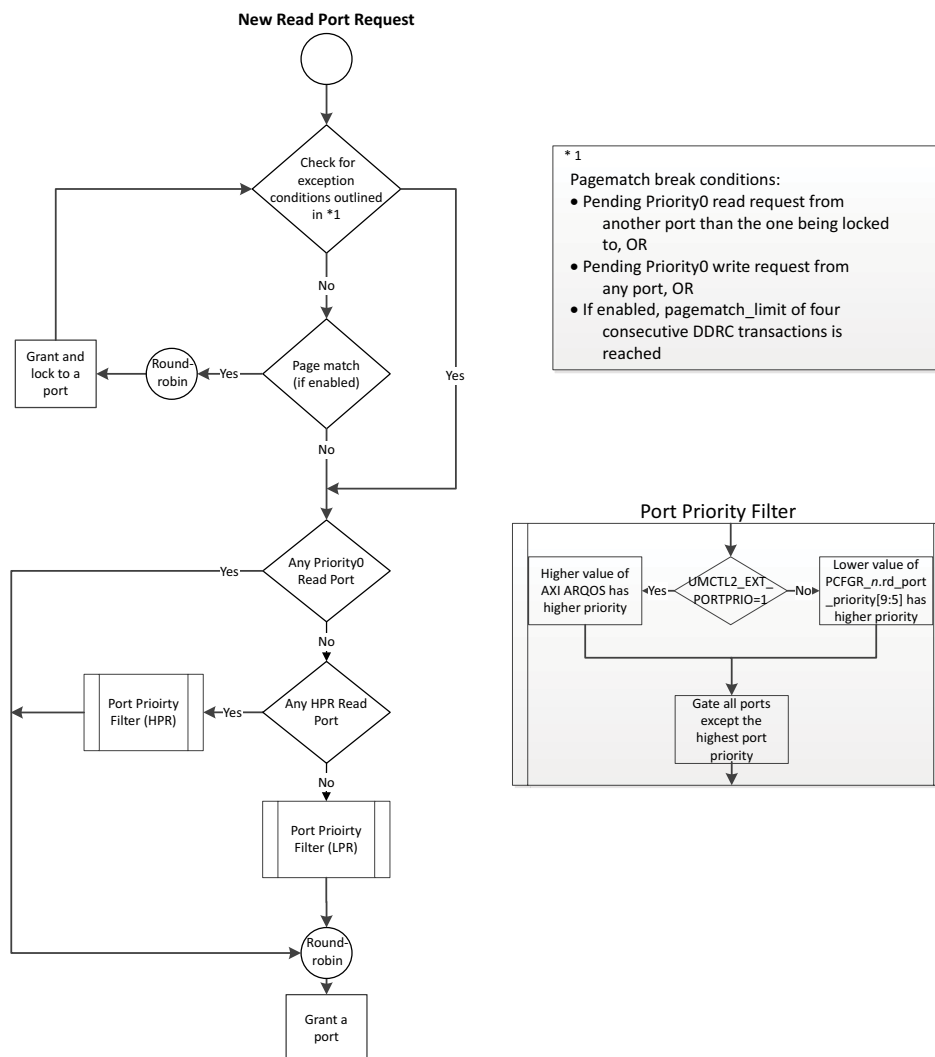
7.1.1 Overview of Port Arbitration (PA)

The Port Arbiter (PA) block arbitrates command requests from up to 16 AXI ports to the HIF of the DDR Controller (DDRC). PA is comprised of multiple tiers of arbitration stages which include:

- Read/write arbitration (when `UMCTL2_DUAL_HIF` is set to '0')
- 2-priority level arbitration based on port aging and expired-VPR/expired-VPW commands (timeout - priority0)
- 2-priority level arbitration for read requests based on DDRC read priorities (HPR/LPR-VPR)
- 32-priority level arbitration based on internal port aging or 16-priority level arbitration based external AXI QoS inputs (selectable by hardware parameter)
- Round-robin arbitration to resolve ports having the same priority after passing all stages of arbitration

Figure 7-1 provides a high-level overview of how a Read port is selected.

Figure 7-1 Selecting a Read Port to Grant to DDRC

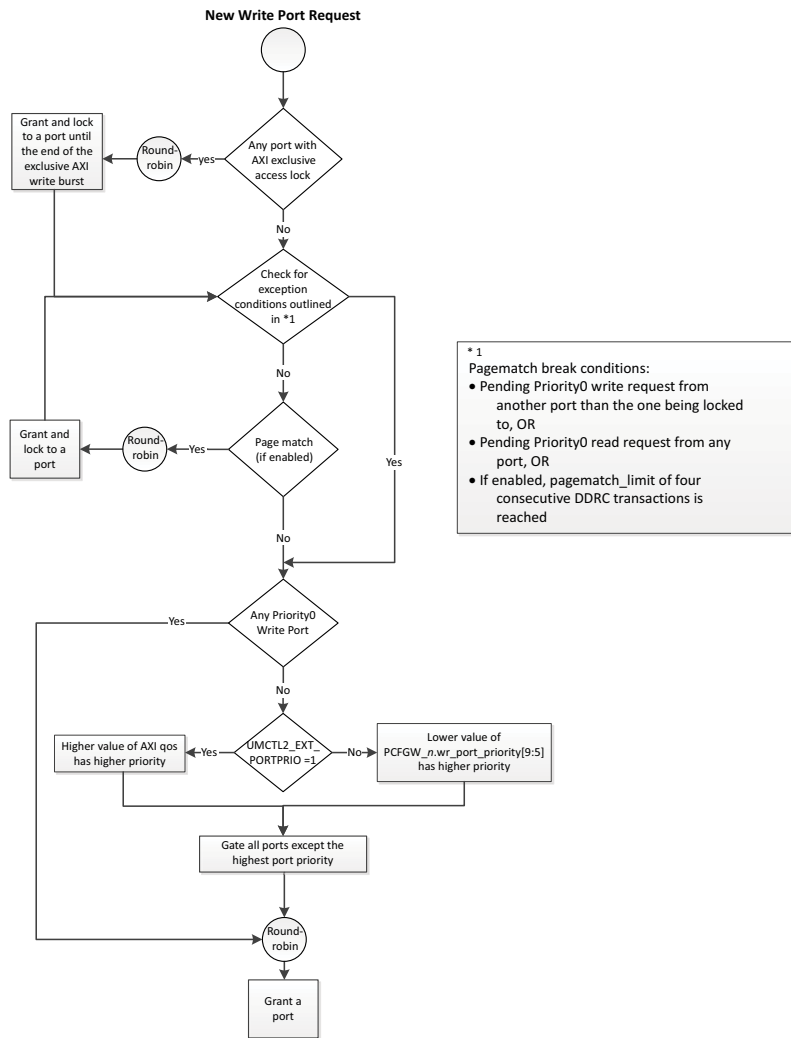


Note

- Read port priority0 is either write port is timed out by port aging or VPW is expired.
- The `awurgent_n` signal leads to port time out immediately and hence leads to port priority0.

Figure 7-2 provides a high-level overview of how a Write port is selected.

Figure 7-2 Selecting a Write Port to Grant to DDRC



Note

- Write port priority0 is either write port is timed out by port aging or VPW is expired.
- The `awurgent_n` signal leads to port time out immediately and hence leads to port priority0.

Port aging refers to counting down the time when a port has a request, but is not granted access to the DDRC. Port timeout refers to when the port age becomes 0, which is also referred as the highest priority - 'priority0'. Expired-VPW refers to Variable Priority Read commands whose counter has expired. Read/write credits mentioned in "Read/Write Arbitration" on page 134 refers to the DDRC 'Credit Mechanism' employed at the HIF interface.

7.1.2 Read/Write Arbitration

This is the first level of arbitration where all requests to the ports are observed and a decision is made based on:

- Read (LPR-VPR, HPR) and write credits
- Timeout/Expired-VPR/Expired-VPW (Priority0)

The main goal of the read/write arbiter is to combine reads and writes together as long as the selected direction has available credits and the timeout has not occurred for any port of the opposite direction. If all conditions are equal, reads are prioritized over writes. Minimize direction switches to improve the memory bus efficiency.

The algorithm can be summarized as:

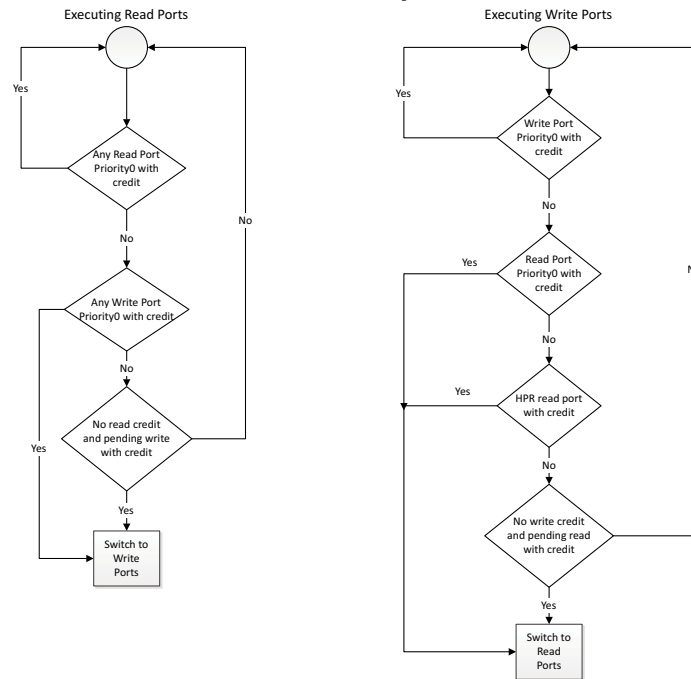
- While executing reads:
 - Stay on the reads as long as there is a timed-out read port or expired-VPR (which are both Priority0 conditions) with available credit, else
 - Switch to writes if there is a timed-out write port or expired-VPW (Priority0) with available credit, else
 - Switch to writes when there is no read credit left and there is a pending write with available credit.
- While executing writes:
 - Stay on the writes as long as there is a timed-out write port or expired-VPW (Priority0) with available credit, else
 - Switch to reads if there is a timed-out read port or expired-VPR (which are both Priority0 conditions) with available credit, else
 - Switch to reads if there is an HPR read port with available credit, else
 - Switch to reads when there is no write credit left and there is a pending read with available credit.

AXI off-band signals, `arurgent` and `awurgent`, when asserted, cause the read/write direction to switch immediately in the Port Arbiter if enabled by `PCFGR.rd_port_urgent_en` and `PCFGW.wr_port_urgent_en` registers (see “REGB_ARB_PORTp” in “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide). Urgent signals also cause the `hif_go2critical_wr` / `hif_go2critical_lpr` signals to be asserted at the HIF interface, which in turn force the read/write direction switching in the DDRC, if enabled by `PCFG.go2critical_en` register.

If `UMCTL2_DUAL_HIF` = 1, there is no Read-Write arbitration performed in the PA. This means that read and write commands can be issued at the same time to the HIF interface. The rest of the PA functionality remains the same. For more information about this, see “Dual HIF Functionality” on page 60.

Figure 7-3 illustrates how the switching occurs between the Read and Write ports when `UMCTL2_DUAL_HIF` = 0.

The PA makes sure that a grant is not given to a requester type (Write, HPR, LPR-VPR) that has no credits. The PA does not give any grants when there is a ‘stall’ indication from the DDRC.

Figure 7-3 Switching Between Read and Write Ports – Only Valid if UMCTL2_DUAL_HIF = 0**Note**

- Read port priority0 is either read port is timed out by port aging or VPR is expired.
- Write port priority0 is either write port is timed out by port aging or VPR is expired.
- `arurgent_n/awurgent_n` will lead to port time out immediately and hence lead to port priority0.

7.1.3 Port Timeout

The per port, per direction (read/write) aging counters count down the time when a port is requesting but is not granted access. When a port aging counter becomes 0, timeout condition occurs and the port becomes the highest priority requester (Priority0) to the Port Arbiter. The initial value of the counters is determined by registers `PCFGR.rd_port_priority` and `PCFGW.wr_port_priority`. The aging feature and the timeout are enabled by `PCFGR.rd_port_aging_en` and `PCFGW.wr_port_aging_en` registers. For a port with two RAQs (that is, `UMCTL2_XPI_USE2RAQ_n` is set for that port), an aging counter is present per RAQ, although only one port priority register is present to set the initial value of the aging counter for both queues.

Each manager can indicate to the Port Arbiter that it is going to starve soon by asserting the AXI off-band signals, `arurgent` and `awurgent`. These sideband signals can be asserted anytime and are not associated with any particular command. As long as the `a[r/w]urgent` signal is set, the priority of that port is the highest (Priority0). This feature is enabled by `PCFGR.rd_port_urgent_en` and `PCFGW.wr_port_urgent_en` registers.

Even if the aging feature is disabled, the `a[r/w]urgent` signal is functional if enabled. The system must ensure that no manager abuses the urgent signal mechanism as this could result in a significant drop of available bandwidth and/or starve other aging ports.

7.1.4 DDRC Read Priorities (HPR/LPR/VPR) and Write Priorities (NPW/VPW) for Ports

The read channel of a port can be set to operate as high priority reads (HPR), low priority reads (LPR), or variable priority reads (VPR). The write channel of a port can be set as normal priority writes (NPW) or variable priority writes (VPW). Priority mapping is done through the `arqos` and `awqos` input based on the `PCFGQOS0` and `PCFGWQOS0` configuration registers (see “`REGB_ARB_PORTp`” in the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide). The PA gives higher priority to a HPR read port than to a LPR/VPR read port. The initial priorities for NPW and VPW are the same.

If any port is set to issue an HPR, the `SCHED0.lpr_num_entries` register must be programmed to a suitable value to reserve locations in the high priority queue of the read CAM.

VPR that are not expired are treated as LPR from the arbiter point of view. If a VPR transaction expires in the XPI, it has higher priority than HPR or writes.

VPW that are not expired are treated as NPW from the arbiter point of view. If a VPW transaction expires in the XPI, it has higher priority than normal writes. When multiple VPR/VPW commands expire simultaneously, the PA executes them in round-robin order to the DDRC.

Maximum timeout for VPR and VPW commands per port and per read queue are set by the `PCFGQOS1_n` and `PCFGWQOS1_n` registers (see “`REGB_ARB_PORTp`” in “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide). VPR and VPW time-out values are loaded in the XPI and decremented at each cycle. If the counter becomes 0 before reaching the DDRC, the command is considered as expired and is referred as the highest priority - ‘priority0’. Otherwise, VPR command is treated as a LPR and VPW command, as NPW and the remaining latency are forwarded to the DDRC. For more information about this, see “[Quality of Service](#)” on page 163.

7.1.5 Port Command Priority

The next tier of arbitration policy is the multiple-level port command priorities. The priorities are per-command and can dynamically change for a given port based on AXI AxQoS signals (`arqos/awqos`). Alternatively, the internal port aging counters can drive priorities if the hardware parameter `UMCTL2_EXT_PORTPRIO` is disabled.

This tier of arbitration has lower-level priority than the Timeout tier. In addition, for reads, Port Priority tier has lower priority than HPR/LPR-VPR tier.

7.1.5.1 External Port Priorities (`UMCTL2_EXT_PORTPRIO = 1`)

If the QoS is managed end-to-end in a system, the port priorities can be adaptively determined by the average latency requirements of a particular port using closed-loop feedback. Therefore, providing external controllability to priorities allows the DDRC to be easily employed in these applications.

Even if port priorities are driven externally, the port timeout feature can still be used through the port aging counters to prevent starvation when required (see “[Port Timeout](#)” on page 136). There are 16-level priorities set by 4-bit wide `arqos/awqos` signals where the higher binary value represents a higher priority.

7.1.5.2 Internal Port Priorities (`UMCTL2_EXT_PORTPRIO = 0`)

Another way of setting port priorities is through port aging counters. When a request is pending and not serviced, a decrementing aging counter kicks off. The starting value of this counter is derived from a 10-bit value in the port priority registers `PCFGR.rd_port_priority` and `PCFGW.wr_port_priority`. The

register resets to the start value when the request is serviced. The upper-most 5 bits of the counter determine the port priority out of 32 levels which approximates an age-based priority for each port. The lower is the value of this counter, the higher the priority.

7.1.6 Round-Robin Arbitration

After passing all tiers of arbitration, a tie is resolved by the final round-robin arbitration stage. The round-robin pointer starts from a port which has the lowest port index and after a grant, the pointer is moved to the first active requester after the one which has just received the grant. When page match feature is enabled, the PA locks on a port and grants to that port. The pointer of the round-robin arbiter continues to subsequent ports, but, they are not granted in that round. They are granted in the subsequent rounds.

7.1.7 Page Match

The Page Match feature is the ability of the Port Arbiter locking on a port when there are consecutive transactions to the same rank, bank and row (same page). The grouping increases the DDR efficiency of the controller by adaptively choosing the back-to-back DDRC transactions to be granted at a time from a given port. Along with the request, each port also sends a 'pagematch' signal to the Port Arbiter. The pagematch attribute of a command indicates that it is a continuation of the previous page. The Page Match feature does not introduce an additional tier of arbitration. It does not affect which port is granted first. After the grant of a port, if there is an immediate transaction with a pagematch attribute set to '1', port lock condition occurs.

A lock on a write port is released if any of the following conditions occur, with the assumption that there is continuous write request with the pagematch attribute set to '1':

1. Pending Priority0 (timed-out) write request from another port than the one being locked to,
2. Pending Priority0 (timed-out) read request from any port,
3. Pending HPR read request from any port,
4. If enabled, `pagematch_limit` of four consecutive DDRC transactions is reached,
5. There is no write credit available and there is read credit available.

A lock on a read port is released if any of the following conditions occurs, with the assumption that there is continuous read request with pagematch attribute set:

1. Pending Priority0 (timed-out) read request from another port than the one being locked to,
2. Pending Priority0 (timed-out) write request from any port,
3. If enabled, `pagematch_limit` of four consecutive DDRC transactions is reached,
4. There is no read credit available and there is write credit available.

The Page Match function can be enabled by the `PCFGR.rd_port_pagematch_en` and `PCFGW.wr_port_pagematch_en` registers. The same page command limit is set by the `PCCFG.pagematch_limit` register (see "REGB_ARB_PORTp" in "Register Descriptions" chapter of the DWC DDRCTL LPDDR5/4 Programming Guide).

7.1.7.1 Address Mapping

To calculate the pagematch attribute of a port, the XPI needs to know the logical to physical address mapping. Logical address is the command address of a transaction as presented on one of the ports. A physical address is the set of bank group, bank, row and column address as presented to the SDRAM memory. The actual conversion from logical to physical address is performed by the DDRC (see `ADDRMAPx`

registers with $x = 0$ to 12). The bank and bank group address locations are indicated in the ADDRMAP3 and ADDRMAP4 registers. The XPI block uses these registers to determine the pagematch attribute of each DDRC command forwarded to the PA.

7.1.8 Write Exclusive Access Lock

An AXI write exclusive access can be comprised of more than one DDRC command. The PA locks on to a port while there are DDRC commands that belong to the same AXI write exclusive access transaction. The write exclusive access lock cannot be released by any other condition and is required for functional correctness. This type of lock occurs transparently if exclusive access monitors are enabled and if there are long write exclusive bursts.

7.1.9 Signals Related to Port Arbiter

For more information on signals related to the Port Arbiter, see “[Signal Descriptions](#)” on page 373 chapter.

7.1.10 Registers Related to Port Arbiter

The following are the registers related to the PA:

- SCHED0
- SCHED1
- SCHED2

For more information about these registers, see the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

7.2 Queue Status Interface and Port Throttling

This topic contains the following subsections:

- [“Overview of Queue Status Interface and Port Throttling”](#) on page 140
- [“CAM Credit Information”](#) on page 140
- [“XPI Queues Information”](#) on page 141
- [“Port Arbiter \(PA\) Port Throttling”](#) on page 142

7.2.1 Overview of Queue Status Interface and Port Throttling

The Information about the internal usage of resources, such as address queues and CAM slots, is provided at the output. This information can be used by an external bridge between the interconnect and the controller to manage virtual channels. For this purpose, port mask inputs are provided as well. These inputs enable you to force the Port Arbiter to grant a specific ports masking all the others.

7.2.2 CAM Credit Information

The controller interface provides the information about the available credits in the CAM.

The information is provided per data channel. In case a dual-channel configuration is selected, *_dch1 version of the signals are available at the output and they refer to data channel 1.

The following are the output signals:

- `lpr_credit_cnt`: indicates the number of available low priority read CAM slots (LPR and VPR reads share the same CAM resources and every burst consumes 1 LPR slot).
- `hpr_credit_cnt`: indicates the number of available high priority read CAM slots (HPR does not share resources with other traffic classes, only HPR reads consume HPR slots).
- `wr_credit_cnt`: indicates the number of available write CAM slots (both NPW and VPW writes share the same CAM resources and every write burst consumes 1 WR slot).

LPR/VPR and HPR resources in the CAM are separate. Total number of read CAM slots can be specified through the parameter `MEMC_NO_OF_ENTRY` (same as for writes). Resources can then be split between LPR/VPR and HPR by programming the register `SCHED0.lpr_num_entries`.

The number of CAM slots available for LPR/VPR is indicated by:

$$\text{SCHED0.lpr_num_entries} + 1$$

While, the number of slots available for HPR is indicated by:

$$\text{MEMC_NO_OF_ENTRY} - (\text{SCHED0.lpr_num_entries} + 1)$$

This means that the maximum value for the signal `lpr_credit_cnt` is always `SCHED0.lpr_num_entries + 1`.

From the CAM, any command can be chosen out-of-order based on scheduling criteria, and a credit is returned whenever a command within a CAM section is scheduled out. Therefore, whenever CAM has empty spaces, there are credits available.



Note

By the definition, the maximum out-of-order depth is not limited within a CAM. It is therefore recommended to enable anti-starvation mechanisms within a CAM (page-hit limiter, `max_rank_rd/max_rank_wr`, and visible window limiter) to minimize worst case latency.

7.2.3 XPI Queues Information

The controller interface provides the information about the address queues in the XPI.

The information involves address queues for both read and write, and for reads separate information for blue/red queues in case dual queue is selected for that specific port (UMCTL2_XPI_USE2RAQ_\$ = 1).

Information is provided per AXI port, the suffix at the end of each signal indicates which port the signal refers to (*_\$ with \$ the port index).

Signals naming convention is as follows:

- Read address queues:
 - raq_* (if single queue)
 - raqb_*/raqr_* (blue/red queue in dual queue configuration)
- Write address queue:
 - waq_*

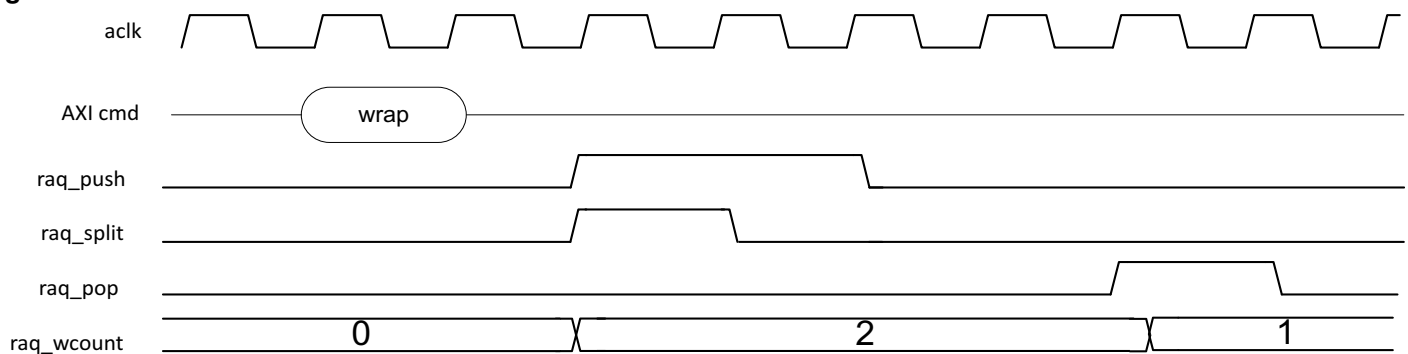
For each of them, the main information provided are:

- *_push_* (command is pushed into the queue, sync to aclk)
- *_split_* (first portion of a wrap pushed into the queue, sync to aclk)
- *_pop_* (command popped from the queue, sync to core_ddrc_core_clk)
- *_wcount_* (number of outstanding commands in the queue, sync to core_ddrc_core_clk)

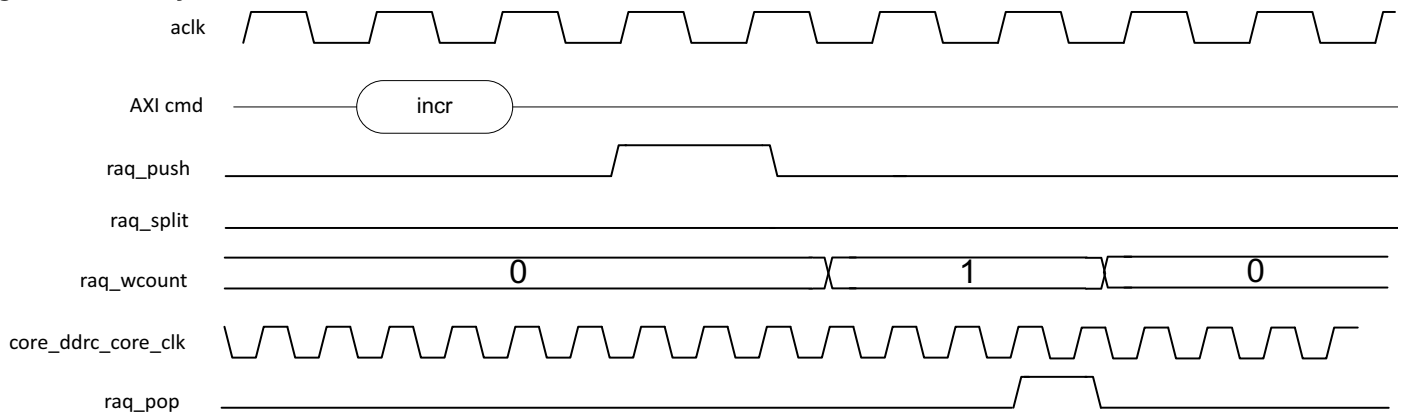
In some cases, an AXI WRAP command at the input may be split into two internal INCR commands and are pushed separately into the queue. This can be recognized when split is asserted. When split and push are high, it means that the first generated INCR command has been pushed into the queue, and the following push means that the second generated INCR command has been pushed into the queue.

Figure 7-4 shows an example of a WRAP command being pushed to the read address queue.

Figure 7-4 WRAP Pushed into the RAQ



The word count of the queue is synchronous to the destination clock domain (that is, core clock), same as the pop signal. Correct value may take some cycles before it gets propagated from the push side. Figure 7-5 shows an example of an INCR command being pushed to an asynchronous queue.

Figure 7-5 Asynchronous Queue

7.2.4 Port Arbiter (PA) Port Throttling

Inputs are provided to control the PA throttle feature:

- `pa_rmask` (read port mask)
- `pa_wmask` (write port mask)

Width of `pa_rmask` is twice the number of ports, 2 bit per ports. First bit is for the blue queue, and the second bit is for the red queue:

- `pa_rmask[0]` -> mask blue queue port 0
 - `pa_rmask[1]` -> mask red queue port 0
 - `pa_rmask[2]` -> mask blue queue port 1
- and so on.

If port is a single queue, the corresponding `pa_rmask` bit is unused.

Width of `pa_wmask` is equal to the number of ports, 1 bit per port.

- `pa_wmask[0]` -> mask port 0
 - `pa_wmask[1]` -> mask port 1
- and so on.

Asserting the corresponding bit prevents that port/queue to be granted at the PA level.

The only exception to the rule is for exclusive access. When an AXI exclusive access write is split into multiple DDR commands, once the first is granted, the exclusive access gets the PA lock until the burst is finished. No masking is possible when an exclusive access has the lock, the mask is applied once the exclusive access is finished.

7.3 Transaction Service Control

This topic contains the following sections:

- [“Overview of Transaction Service Control”](#) on page 143
- [“Page Policy”](#) on page 143
- [“Transaction Stores”](#) on page 145
- [“Address Collision Handling”](#) on page 159
- [“Write Combine”](#) on page 160
- [“Registers Related to Transaction Service Control”](#) on page 162

7.3.1 Overview of Transaction Service Control

Transaction service control allows you to carefully manage the following:

- Costly read/write bus turnaround
- Priorities of read requests to generally favor high priority traffic while also preventing starvation of low priority traffic

This functionality is implemented in a simple 2-state machine for each traffic type with completely configurable controls. The states of the 2-state machine determine when reads/writes are serviced and the relative priority (high priority versus low priority reads) at any given moment.

In this controller, the enhanced RD/WR switching features are always enabled to use.

The key features of the transaction service control are:

- During read mode, issue ACT commands pro-actively for a write request as page preparation in certain condition so that write command can be issued soon after switching to write mode (and vice versa).
- Prefer page-hit on the other direction rather than executing page-miss command on the current direction to reduce number of PRE-ACT cycles in certain conditions.
- Autonomous switching to write mode when Write CAM reaches a certain fill level to avoid Write CAM full as well as keeping read mode to optimize read latency if the WR CAM has enough space.
- In Inline ECC configuration, Write ECC CAM reaches a certain fill level that causes switching to write mode and avoids Write ECC CAM full.



Note

If `MEMC_RDWR_SWITCH_POL_SEL == 1` and `SCHED0.rdwr_switch_policy_sel == 0`, the DDRCTL behavior is compatible with the original one.

7.3.2 Page Policy

This section contains the following subsections:

- [“Explicit Auto-Precharge \(Per Command\)”](#) on page 144
- [“Intelligent Precharges”](#) on page 144

7.3.2.1 Explicit Auto-Precharge (Per Command)

The explicit auto-precharge feature can enable auto-precharge on a per-command basis. If the HIF signal `hif_cmd_autopre` is set during a valid command, then the auto-precharge bit for that command is set when it is sent to the SDRAM.

If you do not want to use the per-command auto-precharge feature, then this bit can be tied to 1'b1 or 1'b0. Tying it to 1'b1 causes all the commands to be executed with auto-precharge, and tying it to 1'b0 causes no commands to be executed with auto-precharge. For AXI configurations, the HIF signal `hif_cmd_autopre` is tied to '0' internally, and is not accessible.

In cases where a HIF transaction is translated into multiple DFI transactions, only the last DFI transaction is executed with auto-precharge. The earlier ones keep the page open, to allow subsequent transactions to benefit from the page hit.

7.3.2.2 Intelligent Precharges

The `SCHED0.pageclose` register enables precharge commands to be issued smartly through auto-precharges or explicit precharges. The exact functionality depends on the value programmed in

`SCHEDTMG0.pageclose_timer`:

- If `SCHED0.pageclose` is set to '1' and `SCHEDTMG0.pageclose_timer` = 0, a bank is kept open while there are page hit transactions available in the CAM to that bank. The last read or write command in the CAM with a bank and page hit is executed with auto-precharge. The intelligent precharge logic considers only the CAM or CAM region that is currently active, that is, while executing reads it looks at the LPR or HPR region of the read CAM (whichever is active at the time) and does not look at transactions in the other region of the read CAM or in the write CAM; while executing writes the logic does not look at transactions in the LPR or HPR regions of the read CAM. Whenever a mode switch occurs between LPR, HPR and writes, the page context is lost and therefore, a redundant auto-precharge may be issued followed by an explicit activate to the same bank or only an explicit precharge may be issued instead of an auto-precharge. The read and write commands that are executed as part of the ECC scrub requests are also executed with explicit precharge if the page needs to be closed.
- `SCHED0.pageclose` is set to '1' and `SCHEDTMG0.pageclose_timer` > 0, a bank is kept open while there are page hit transactions available in the CAM to that bank. The last read or write command in the CAM with a bank and page hit is not executed with auto-precharge. Instead, a timer is started with `pageclose_timer` as the initial value. There is a timer on a per bank basis. The timer decrements unless the next read or write in the CAM to a bank is a page hit. It resets to `pageclose_timer` value if the next read or write in the CAM to a bank is a page hit. Once the timer reaches zero, an explicit precharge is scheduled. Prior to a bank's timer expiring, an explicit precharge may also be scheduled if the transaction scheduler chooses a CAM entry that is a row miss to the same bank.
- If `SCHED0.pageclose` is set to '0', the bank remains open only until there is a need to close it (to open a different page or for page timeout or refresh timeout). This is also known as open page policy. The open page policy can be overridden by setting the per command auto pre bit on the HIF interface (`hif_cmd_autopre`). When the multi-port arbiter is configured, the HIF signal `hif_cmd_autopre` is tied to '0' and you do not have control over it.

The intelligent precharge feature provides a midway between open and close page policies.

`SCHEDTMG0.pageclose_timer` gives you control over the time waited when there are no page hits to a bank in the CAM before auto-precharge or explicit precharge is scheduled. It is useful when the multi-port arbiter is configured. This timer allows the page to be kept open for a configurable number of clock cycles after there are no commands pending in the CAM to a bank. If there are pending commands higher up in

the stream (for example, XPI/PA of multi-port arbiter) to the same bank/page, it gives a chance to be scheduled by the DDRCTL as an open page command.

Table 7-1 provides a summary of all paging policy options available in the DDRCTL.

Table 7-1 **Paging Policy Options**

Paging Policy	Availability	Description
Open page policy	All configurations	Setting <code>SCHED0.pageclose</code> to '0' enables the open page policy. In HIF configurations, the open page policy can be overridden by sending commands with <code>hif_cmd_autopre = 1</code> . In AXI configurations, the open page policy can be overridden in certain cases by sending commands with <code>arautopre_n=1</code> .
Intelligent precharge (Midway between Closed page and Open page policy)	All configurations	Setting <code>SCHED0.pageclose</code> to '1' enables this smart paging policy as described in "Intelligent Precharges" on page 144. The exact functionality is dependent on <code>SCHEDTMG0.pageclose_timer</code> value. In HIF configurations, this paging policy can be overridden by sending commands with <code>hif_cmd_autopre = 1</code> . In AXI configurations, the open page policy can be overridden in certain cases by sending commands with <code>arautopre_n=1</code> .
Closed page policy	Only in HIF configurations. Not in multi-port configuration.	Control the paging policy per-command by asserting or de-asserting the auto precharge signal on the HIF interface – <code>hif_cmd_autopre</code> . If all commands are to be executed with auto-precharge, <code>hif_cmd_autopre</code> can be tied to '1'. In AXI configurations, the open page policy can be overridden in certain cases by sending commands with <code>arautopre_n=1</code> .



Note

A combination of `SCHED0.pageclose = 1` and `SCHEDTMG0.pageclose_timer = 0` is not supported but issuing of auto-precharge commands may not be accurate for all cases.

7.3.3 Transaction Stores

Transactions in the DDRCTL are separated into five categories:

- Low Priority Reads (LPR)
- Variable Priority Reads (VPR)
- High Priority Reads (HPR)
- Normal Priority Writes (NPW)
- Variable Priority Writes (VPW)



Note

VPR and VPW categories are supported in HIF-only configurations or in arbiter configurations with AXI port. Hardware parameter `UMCTL2_VPRW_EN` must be defined to enable the VPR/VPW feature.

7.3.3.1 Read Transaction Store

This section describes how LPR, VPR and HPR traffic classes work inside the DDRC. For information on how to map the AXI `arqos` value to LPR, VPR, and HPR traffic queues, see [“Quality of Service”](#) on page 163.

`SCHED0.lpr_num_entries` register splits the Read CAM in the controller into LPR and HPR sections. The LPR and VPR commands are sent to the LPR section of the CAM, and the HPR commands are sent to the HPR section. The VPR commands come in with an associated ‘timeout’ value. The LPR and HPR commands do not have ‘timeout’ value associated with them. The ‘timeout’ value of the VPR commands are counted down every cycle in which the commands are pending in the LPR store. If any VPR command has not been serviced and its ‘timeout’ value reaches 0, the command is considered as expired-VPR command. When there are any expired-VPR commands in the DDRC, those commands are given higher priority than HPR and LPR commands.

The Read CAM handles three traffic classes - LPR, VPR and HPR. But the Scheduling Engine that gets its input from the Read CAM, handles only two traffic classes - LPR and HPR. When there are no expired-VPR commands, all the VPR commands are treated as LPR.

7.3.3.2 Write Transaction Store

This section describes how NPW and VPW traffic classes work inside the DDRC. See [“Quality of Service”](#) on page 163 for information on how to map the AXI `awqos` value to NPW and VPW traffic queues.

The NPW and VPW commands are sent to the Write CAM. VPW commands do not have reserved space in the Write CAM. The VPW commands come in with an associated ‘timeout’ value. The NPW commands do not have ‘timeout’ value associated with them. The ‘timeout’ value of the VPW commands are counted down every cycle in which the commands are pending in the WR store. If any VPW command has not been serviced and its ‘timeout’ value reaches 0, the command is considered as expired-VPW command. When there are any expired-VPW commands in the DDRC, those commands are given higher priority than NPW commands.

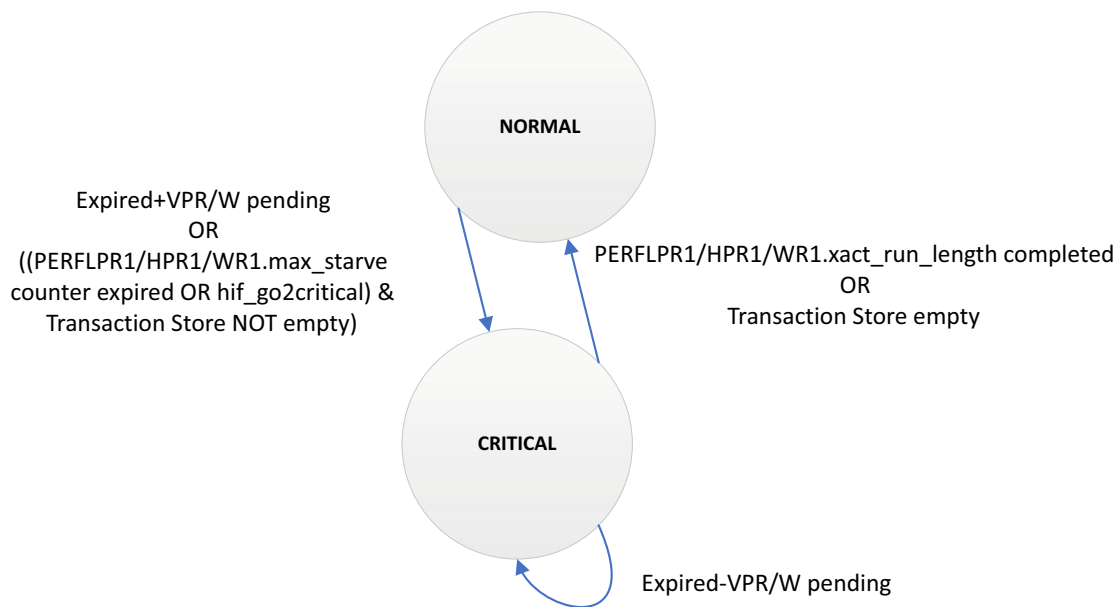
The Write CAM handles two traffic classes - NPW and VPW. But the Scheduling Engine that gets its input from the Write CAM, handles only one traffic class - NPW. When there are no expired-VPW commands, all the VPW commands are treated as NPW. When there are any expired-VPW commands in DDRC, all expired-VPW commands are given higher preference over NPW commands.

7.3.3.3 Transaction Store State Transitions

Each class of transaction store (LPR, HPR, or WR) can be in any one of the following two states:

- Normal: the state where the transaction store starts.
- Critical: indicates that the transaction store must be prioritized for service.

The transaction stores move between these two states under the control of `*_max_starve` and `*_xact_run_length` registers.

Figure 7-6 Transaction Store FSM (One FSM per store - WR, LPR, HPR)

Note: When there is more than one transition from a given state, the priority is shown in each state, with smaller number representing higher priority

Table 7-2 describes the transaction store state transitions.

Table 7-2 Transaction Store State Transitions

Current State	Next State	State Transition
Normal	Critical	This transaction store is not serviced for a count of *_max_starve clock cycles, or if an address collision happens.
Critical	Normal	*_xact_run_length number of transactions is serviced from this transaction store.

Taking the low priority read transaction store as an example, it is expected that the transaction store generally functions independently based on the following registers (see “REGB_FREQf_CHc Registers” in the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide):

- PERFLPR1.lpr_max_starve
- PERFLPR1.lpr_xact_run_length

In the normal mode of operation, the FSM moves from normal to critical state by the starvation timer timeout. The store remains in Critical state until the number of serviced transactions becomes equal to PERFLPR1.lpr_xact_run_length, or when no more commands are available for that store, whichever happens first.

When the DDRCTL is configured as HIF-only (UMCTL2_INCL_ARB == 0), it provides a feature where the SoC can force this state transition using external signals. It is useful in cases where the SoC may have additional information that is helpful in determining state transitions. For example, if the data stream has real-time requirements and the SoC has knowledge about FIFO depths or time-till-failure, it can use this information to explicitly request the DDRCTL to prioritize a particular data stream when it is critical to do so. The signals associated with this feature are hif_go2critical_lpr and hif_go2critical_wr.

In case of multi-port arbiter configuration (`UMCTL2_INCL_ARB == 1`), the `hif_go2critical_*` signals are driven by the Port Arbiter. The generation of three signals involved in this case - `hif_go2critical_wr`, `hif_go2critical_lpr` and `hif_go2critical_hpr` is described in “[Quality of Service](#)” on page 163 and “[Urgent Signaling](#)” on page 168.

The assertion of `hif_go2critical_*` signals cause their respective queue FSMs to go to Critical state. The change in Read/Write mode switching and the Read priority level selection based on the presence of the `hif_go2critical_*` signals is mentioned in the following section.

**Note**

If `SCHED.opt_wrcam_fill_level = 1`, `SCHED3.wrcam_highthresh/SCHED5.wrecc_cam_highthresh` and `SCHED3.wrcam_lowthresh/SCHED5.wrecc_cam_lowthresh` are used for the transition in addition to `PERFWR1.w_max_starve` and `PERFWR1.w_xact_run_length`.

7.3.3.3.1 Store Critical

Depending factors, levels of critical are introduced on top of the original FSM.

Critical_L1 is the highest priority and Critical_L3 is the lowest priority within Critical_L1/2/3.

Table 7-3 Priorities for Configurations with `UMCTL2_INCL_ARB == 1`

Priority	External Signal	Source
Critical_L1	<code>awurgent_n</code> <code>arurgent_n</code> (<code>arurgentb_n</code> , <code>arurgentn_n</code> when <code>UMCTL2_XPI_USE2RAQ_n</code> is defined)	AXI sideband signals from SoC. These make corresponding store Critical_L1. For more information, see “ Urgent Signaling ” on page 168.
Between L1 and L2		If <code>opt_wrcam_fill_level == 1</code> , write CAM fill-level makes write store Critical/Normal, the priority is higher than Critical_L2, and lower than Critical_L1.
Critical_L2	N/A	Asserted by XPI/PA for VPR/VPW timeout with corresponding CAM credit. For more information, see “ VPR/VPW Timeout ” on page 166.
Critical_L3	N/A	Critical caused by <code>PERF*.*_max_starve</code> . AXI port timeout only force read write switching inside PA, does not impact on transaction store FSM directly.

Table 7-4 Priorities for Configurations with `UMCTL2_INCL_ARB == 0`

Priority	External Signal	Source
Critical_L1	<code>hif_go2critical_wr</code> <code>hif_go2critical_hpr</code> <code>hif_go2critical_lpr</code>	HIF sideband signals from SoC. Drive these make corresponding store Critical_L1.
Between L1 and L2		If <code>opt_wrcam_fill_level == 1</code> , write CAM fill-level makes write store Critical/Normal, the priority is higher than Critical_L2, lower than Critical_L1.

Priority	External Signal	Source
Critical_L2	N/A	Not used source for the configurations.
Critical_L3	N/A	Critical caused by PERF*.*_max_starve.

7.3.3.4 Read Priority Management

In a read mode, read priority is controlled by `SCHED1.opt_hit_gt_hpr` and critical state.

If `SCHED1.opt_hit_gt_hpr == 0`, or the register does not exist in your configuration, the preferred priority is:

1. Page-hit high priority read
2. Page-miss high priority read
3. Page-hit low priority read
4. Page-miss low priority read

If `SCHED1.opt_hit_gt_hpr == 1`, the preferred priority is:

1. Page-hit high priority read
2. Page-hit low priority read
3. Page-miss high priority read
4. Page-miss low priority read

If the low priority read transaction store is in Critical state due to a transaction that is pending for a long time (see [Table 7-2](#) on page 147) and the high priority read transaction store is not, the low priority read requests are preferred over high priority read requests. This prevents starvation of low priority reads.

The exceptions to the previous rule are as follows:

- When there are any expired-VPR commands in the DDRC, these are given priority ahead of any LPR commands, even if the LPR queue is in a critical state.
- If `hif_go2critical_hpr` is high, the HPR queue is given priority over LPR queue, even if LPR queue is in Critical state.
- If `hif_go2critical_lpr` is high and `hif_go2critical_hpr` is low, the LPR queue is given priority over HPR queue, even if HPR queue is in Critical state.

7.3.3.5 Read/Write Turnaround

The following terminology is used to describe the read/write turnaround algorithm:

- Write/Read pending: A write or a read command is pending in the CAM.
- Write/LPR/HPR critical: Indicates that the write, LPR or HPR queue is in critical state due to starvation.
- Read/Write idle timeout: A read/write idle timer based on the register `SCHEDTMG0.rdwr_idle_gap` has expired.
- Page-miss: A request targeting to a closed bank or an opened bank with different page.
- Page-hit: A request targeting to an opened page on a bank.

- Page-hit timing based on ACT (page-hit timing 0): A page-miss request in Next Transaction Table (NTT), which stores the next candidate per bank/per direction, becomes page-hit once an ACT command is issued to the page. A read/write command cannot be issued until tRCD is satisfied. For more information, see [Figure 7-7](#) on page 151.
- Page-hit timing based on ACT + tRCD (page-hit timing 1): A page-miss request in Next Transaction Table becomes page-hit once an ACT command is issued to the page and tRCD is elapsed. Now, read/write command can be issued.
- Read Collision: An incoming command (write or RMW) on HIF has the same address as the existing read commands in the CAM. These existing reads must be scheduled-out timely (flush) to accept the incoming command.
- Write Collision: An incoming command (read/write/RMW) on HIF has the same address as an existing Write command in the CAM. This existing Write must be scheduled-out timely (flush) to accept the incoming command.
- Colliding Request: Existing entries in WR CAM or RD CAM, which has same address as incoming command.
- Collision Page-hit: Indicates the page-hit for colliding bank. In other words, during flush for address collision, the bank of the colliding request to be flushed is a page-hit. When a collision happens, the colliding request may not or may be loaded in NTT.
 - If colliding request is loaded in NTT, when the colliding bank is a page-hit, the colliding request is a page-hit request and is scheduled out.
 - If colliding request is not loaded in NTT, even when the colliding bank is a page-hit, it does not mean colliding command is a page-hit because the page is opened for non-colliding request. Once the request loaded in 'Next Transaction Table' and it is scheduled out, then the colliding request is loaded to 'Next Transaction Table' and another page can be opened if this is a page-miss against the previous one. In this case, collision page-hit may have two periods, one is previous request in 'Next Transaction Table' is a page-hit, and another is colliding request in 'Next Transaction Table' is a page-hit. Both the periods are collision page-hit.
- Critical: Critical is for each store (LPR, HPR or write). A read critical includes LPR store critical or HPR store critical. Write critical indicates write store critical.
 - Critical could be caused by starvation timer or sideband signals (`hif_go2critical_*` in HIF configuration or `arurgenttr/arurgentb` and `awurgent` in AXI configuration)

Page-hit status is used as hint of global read/write switching. Depending on conditions, the DDRCTL selectively uses page-hit timing 0 or page-hit timing 1 as shown in [Table 7-5](#).

Table 7-5 Page-hit Timing

Transaction	Expired-VPR/W	Collision	Critical	Normal
Read	Page-hit timing 0	Page-hit timing 0	Page-hit timing 0	Page-hit timing 1
Write	Page-hit timing 0	Page-hit timing 0	Page-hit timing 1	Page-hit timing 1

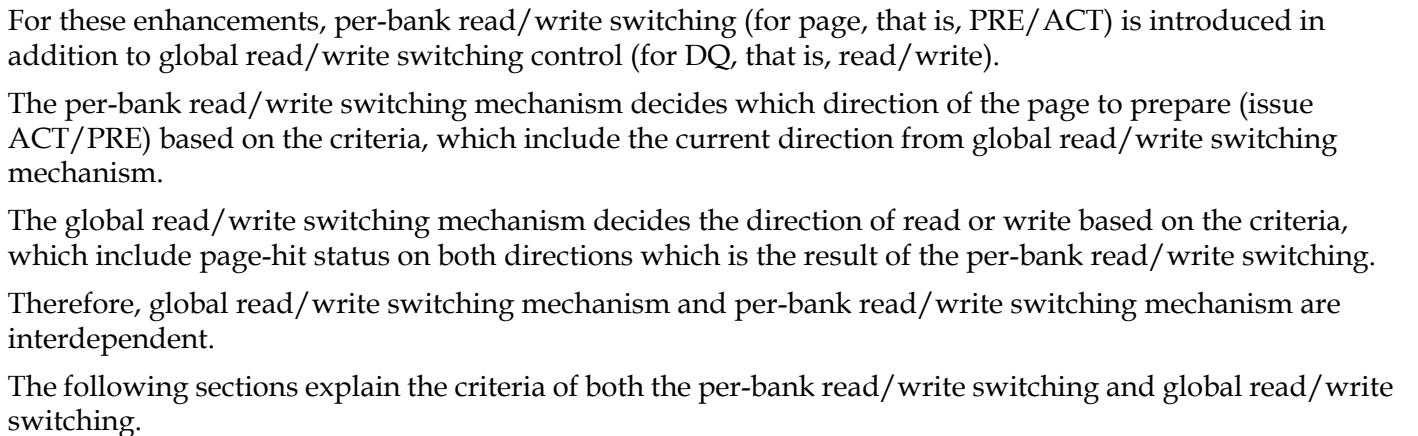
Some examples:

- There is Expired-VPR, it is treated as a page-hit once ACT for the Expired-VPR is issued.
- Write store is critical, it is treated as page-hit once ACT for a write command is issued and tRCD is elapsed.

As described in “[Overview of Transaction Service Control](#)” on page 143, the following new features are introduced:

- During read mode, issue an ACT command pro-actively for a write request as page preparation in certain conditions, so that the write command can be issued soon after switching to write mode (and vice versa).
- Prefer page-hit on the other direction rather than executing page-miss command on the current direction to reduce the number of PRE-ACT cycles in certain conditions.
- Autonomous switching to write mode when Write CAM reaches a certain fill level to avoid Write CAM full as well as keeping read mode to optimize read latency if WR CAM has enough space.

Figure 7-7 Comparison Between Enhanced/Original Read/Write Switching Mechanism



7.3.3.5.2 Per-bank Read/Write Switching for Page (ACT/PRE)

For a given bank, issue activate for the current direction request or the other direction request will be decided by the following information:

- Global read write mode for DQ (that is, read/write commands)
- Critical status of Transaction Store FSMs, that is, HPR/LPR/WR store
- Collision status for each bank
- Page-hit status in read and write Next Transaction Table (NTT)

Figure 7-8 shows the decision for page management per bank.

Figure 7-8 Per-bank Page Control



- Expired-VPR/W is the highest priority.
 - If there are Expired-VPR requests, prepare a page of the bank for read direction regardless of the current global direction.
 - If there are Expired-VPW requests and no Expired-VPR requests, prepare a page of the bank for write direction regardless the current global direction.

- Collision is the next priority.
 - If only one direction has colliding request to be flushed, prepare page of the bank for collision direction.
 - If both directions have colliding request to be flushed, prefer to prepare a page of the bank for current direction if any page of the bank is not open for the other direction.
- Transaction Store FSM Critical is the next priority after collision. The behavior is similar to collision.
 - If both directions are critical, compare the critical level, prefer to prepare a page of the bank for higher level critical direction ($L1 > L2 > L3$).

**Note**

Write CAM fill level or Write ECC CAM fill level exceeding high-threshold, causes write store Critical. The critical level is lower than Critical_L1 but higher than Critical_L2.

- If both directions are at the same critical level, prefer to prepare a page of the bank for current direction unless the bank is already open for the other direction.

**Note**

Transaction Store FSM critical is per-store, and not per-bank. Therefore, this applies to all the banks.

- Except previous cases, prefer to prepare a page of the bank for the current direction if page of the bank is not open for the other direction.

**Note**

While global read/write switching mechanism is in an intermediate state from read mode to write mode, in other words, waiting for `SCHED1.delay_switch_write` is expired, nothing is done for the bank. That is, do not prepare any page of the bank because it is uncertain which direction is finally chosen by global read/write switching.

- When the register `rd_act_idle_gap` is set to non-zero value, and in the read mode, do not prepare page of the bank for the write direction until the number of cycles determined by the `rd_act_idle_gap` is elapsed with no read request to the bank. This is intended to reduce the frequent read to write switching.
- Similarly, when the register `wr_act_idle_gap` is set to non-zero value, and in the write mode, do not prepare page of the bank for the read direction until the number of cycles determined by the `wr_act_idle_gap` is elapsed with no write request to the bank. This is intended to reduce the frequent write to read switching.
- The intention of the two separated activate idle gap registers, namely `rd_act_idle_gap` and `wr_act_idle_gap`, are to make hysteresis for page preparation for the other direction.
- `SCHED4.rd_page_exp_cycles` indicate number of cycles to keep the bank opened for read direction in the write mode when both directions have requests to the bank and to different pages. Similarly, `SCHED4.wr_page_exp_cycles` indicate number of cycles to keep the bank opened for write direction in read mode when both directions have requests to the bank and to different pages.
 - The purpose is to avoid a bank being opened for a long time in the opposite direction, while there is a page-miss requesting in the current direction, so that more banks can be used for the current

direction to improve utilization. This is critical especially for bank-group rotation case, if only one bank group can be used for current direction, and the other bank-groups are opened for the opposite direction, the tCCD_L penalty cannot be avoided.

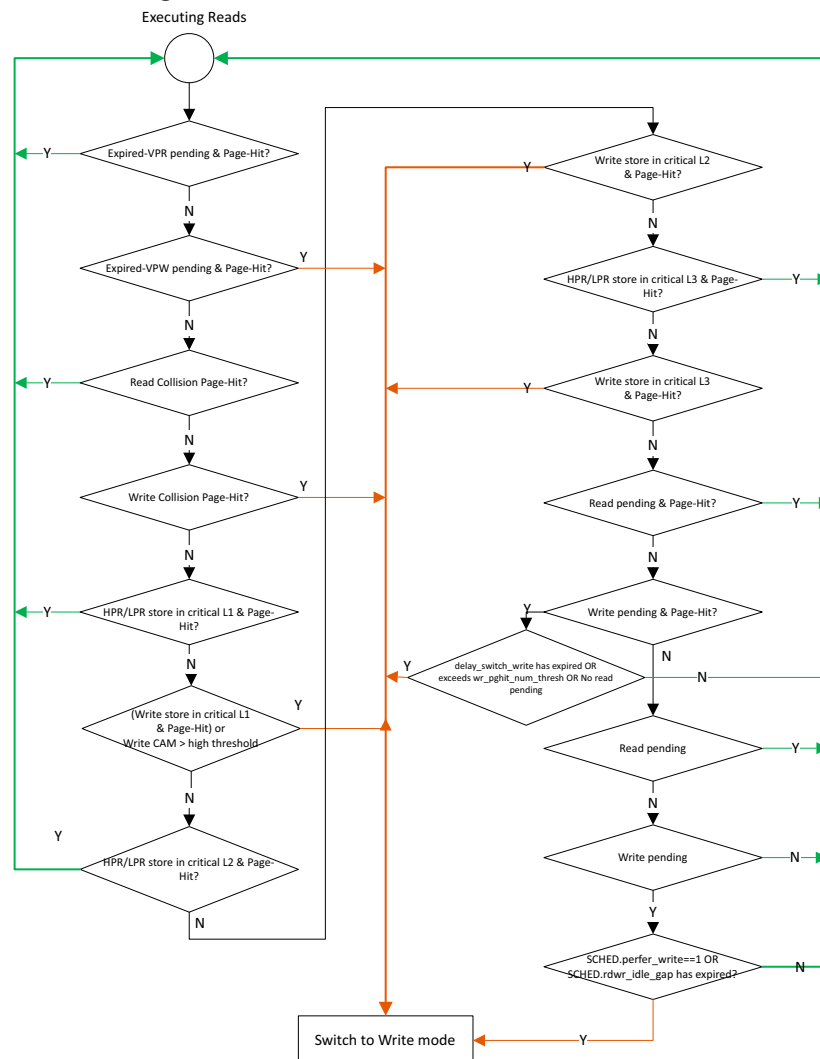
- While global read/write switching mechanism is in an intermediate state from read mode to write mode, in other words, waiting for `SCHED1.delay_switch_write` to expire, pause timer for `SCHED4.wr_page_exp_cycles` for all the banks because it is uncertain which direction is finally chosen by global read/write switching.

7.3.3.5.3 Global Read/Write Switching for DQ (Read/Write commands)

Read to Write Switching

Figure 7-9 shows the decision for global Read to Write switching.

Figure 7-9 Read to Write Switching



- Switch to write once a write request becomes a page-hit even if there are pending read requests but all are page-miss. This is intended to reduce the number of PRE-ACT cycles.
- Keep reading direction if both read and write requests are same priority, such as both read and write are collision and page-hit, both are in same critical level and page-hit, or both are page-miss.

- Expired-VPR/W is the highest priority, that is, if there are any Expired-VPR or Expired-VPW, only Expired-VPR and Expired-VPW could be candidate to cause read write switching, as the other requests are masked.

**Note**

If both Expired-VPR and Expired-VPW are present and one of them is a page-hit, the page-hit one is higher priority than the other (If Expired-VPR and Expired VPW are on the same bank, Expired-VPR has priority for the page preparation, this is per-bank control).

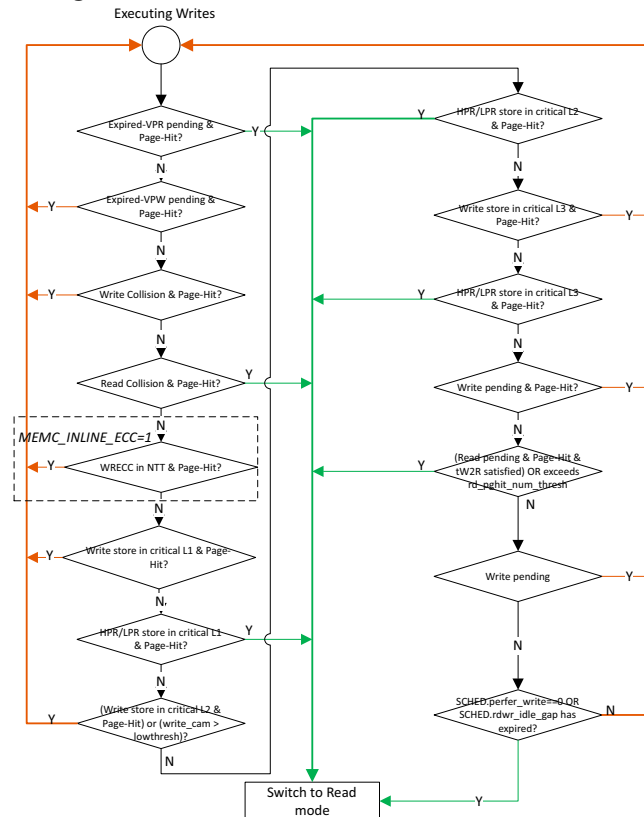
- Collision page-hit is higher priority than store critical. The write collision page hit causes read to write switching if no read collision page hit exists even there is a read critical page-hit.
- Critical page-hit is higher priority than normal read/write page-hit. Write store critical page-hit causes read to write switching if LPR or HPR store are not critical.
 - Critical is separated into 3 levels, the priority is Critical_L1 > Critical_L2 > Critical_L3. Prefer to switch to the direction with higher level critical. For more information, see “Store Critical” on page 148.
 - If the register `SCHED.opt_wrcam_fill_level = 1`, write store starvation is managed by `SCHED3.wrcam_highthresh/SCHED3.wrcam_lowthresh/SCHED5.wrecc_cam_high-thresh/SCHED5.wrecc_cam_lowthresh` in addition to `PERFWR1.w_max_starve` and `PERFWR1.w_xact_run_length`. The switch from read to write is higher priority than Critical_L2, a lower priority than Critical_L1.
- A normal page-hit is of higher priority than page-miss. The write page-hit causes read to write switching if read request is not page-hit.
- Read page-miss is higher priority than `SCHED0.prefer_write` register. Even if `SCHED0.prefer_write = 1`, keep read mode until all the read request are served regardless of a page-hit or page-miss as long as there is no write page-hit.
- To delay switching to write mode `SCHED1.delay_switch_write` is provided. It is possible that the incoming read request could be a page-hit or the pending read request will be a page-hit in the next several cycles even if there are no page-hit right now. With this function, frequent ‘read -> write -> read’ turnaround can be avoidable (keep read mode), and therefore read latency can be reduced.
- While waiting for `SCHED1.delay_switch_write` timeout, switch to write mode immediately once the number of write page-hit requests (total in all banks) exceeds the threshold set in `SCHED1.wr_pghit_num_thresh`. As this gives more confidence to switch to write as there are several page-hits command on write.

**Note**

`SCHED1.wr_pghit_num_thresh` cannot be used when `SCHED1.delay_switch_write = 0`.

7.3.3.5.4 Write to Read Switching

Figure 7-10 shows the decision for global Write to Read switching.

Figure 7-10 Write to Read Switching

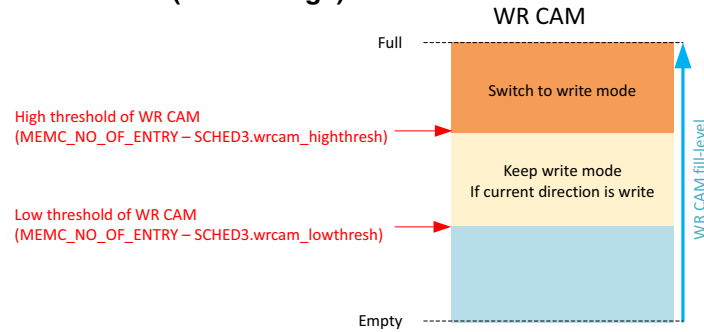
Write to read switching is almost symmetrical with read to write switching. The differences are:

- Expired-VPB is higher priority than Expired-VPW even if it is a write mode.
- If there are page-hit in read direction and no page-hit in write direction, switch to read after tW2R time is satisfied. During waiting for write to read turnaround time specified in `DRAMSET1TMG2.t_wr2rd` (tW2R) is satisfied:
 - If a write becomes a page-hit, keep write mode.
 - If number of read page-hit exceeds threshold (`SCHED3.rd_pghit_num_thresh`), switch to read mode without waiting for tW2R is satisfied. As this gives more confidence to switch to read as there are several page-hits command on read.

When write CAM fill-level exceeds low threshold (`SCHED3.wrcam_lowthresh`), do not switch to read until write CAM fill level is under low threshold. For more information, see [Figure 7-11](#).

7.3.3.5.5 Optimize WR CAM Fill-Level

[Figure 7-11](#) shows the basic image of optimize WR CAM fill-level.

Figure 7-11 Optimize WR CAM Fill-Level (Basic Image)

When write CAM is almost full, switch to write mode to prevent write CAM full.

The two thresholds register provided for this feature are:

1. `SCHED3.wrcam_highthresh`
2. `SCHED3.wrcam_lowthresh`

In Inline ECC configurations (`MEMC_INLINE_ECC==1`), in addition to write CAM fill-level, write ECC CAM fill-level is also optimized by observing the number of valid write ECC entries and number of loaded write ECC entries:

- Valid write ECC entry indicates that the write ECC entry can be served as write CAM has no write entries belonging to the same Block Token (BT).
- Loaded write ECC entry indicates that the write ECC entry is loaded in the write ECC CAM but cannot be served as write CAM has the write entries belonging to the same Block Token.

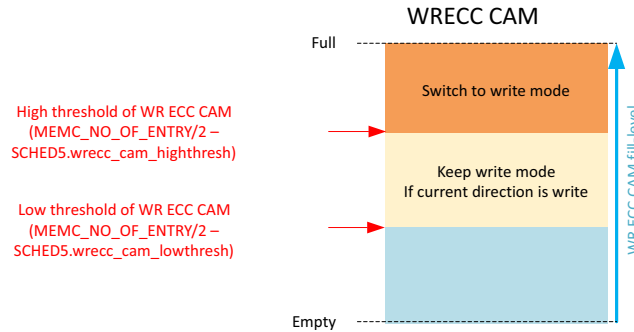
This feature can also be controlled by the following two registers:

- `PERFWR1.w_max_starve`
- `PERFWR1.w_xact_run_length`

If `PERFWR1.w_max_starve > 0`, after `PERFWR1.w_xact_run_length` number of write commands are served while WR CAM threshold exceeds the low threshold, the controller can switch to read even if the number of WR CAM entries is higher than the low threshold. After that, `PERFWR1.w_max_starve` defines the cycle: how long the feature allows to read mode. Once the `PERFWR1.w_max_starve` is expired and WR CAM fill level exceeds the high threshold, the controller switches to write mode again. This is to avoid extreme read latency in case write demand is very high.

If `PERFWR1.w_max_starve = 0`, keep write mode as long as the WR CAM fill level exceeds the low threshold (with the exception of exVPR and collisions).

Figure 7-12 shows the basic image of optimize valid write ECC CAM fill-level.

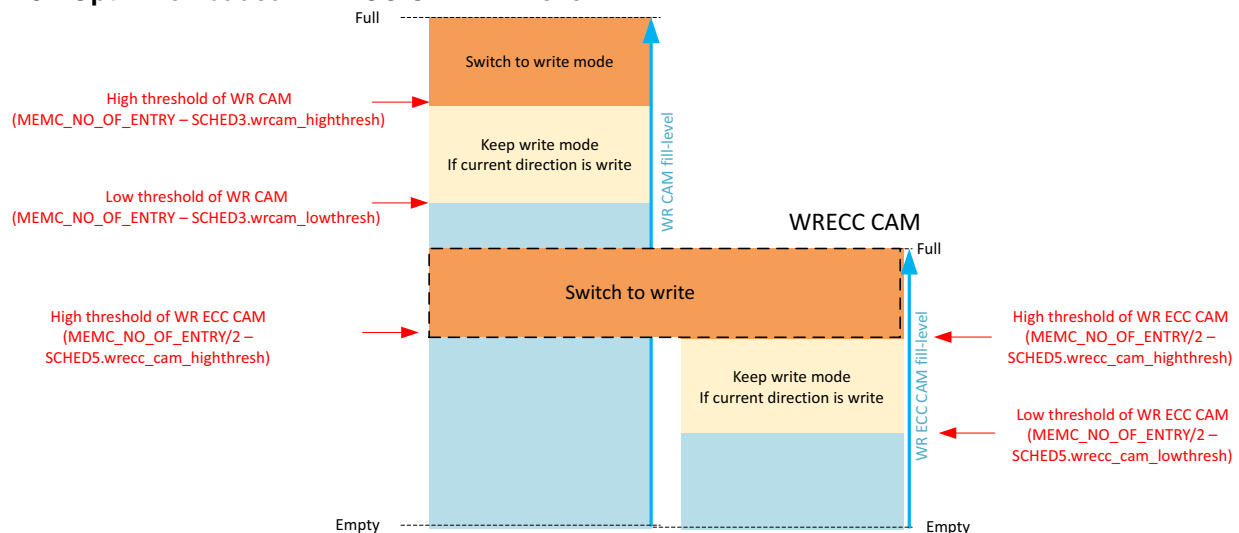
Figure 7-12 Optimize WR ECC CAM Fill-Level

When a valid write ECC CAM exceeds high threshold, switch to write mode to prevent write ECC CAM full.

The two threshold registers provided for this feature are:

- SCHED5.wrecc_cam_highthresh
- SCHED5.wrecc_cam_lowthresh

Figure 7-13 shows the basic image of optimize loaded write ECC CAM fill-level.

Figure 7-13 Optimize Loaded WR ECC CAM Fill-Level

In some extreme cases, for example, one write entry corresponds to one write ECC entry, while write ECC CAM depth is half of write CAM depth. As a result, write ECC CAM is full while write CAM is not full (such as half of entries are used, not exceeds high threshold), and the write ECC entries are just loaded. In this case SoC cannot send write requests to the protected region because no write ECC credits are available.

To optimize write ECC CAM fill level, the DDRCTL switches to write mode to serve write entries when loaded write ECC entries exceed the high threshold of WR ECC CAM ($\text{MEMC_NO_OF_ENTRY}/2 - \text{SCHED5.wrecc_cam_highthresh}$) and write entries exceed the higher threshold of WR ECC CAM ($\text{MEMC_NO_OF_ENTRY}/2 - \text{SCHED5.wrecc_cam_highthresh}$). This function is disabled by setting `SCHED5.dis_opt_load_wrecc_cam_fill_level` as default.

This optimized write CAM and write ECC CAM fill level feature is enabled/disabled by `SCHED.opt_wrcam_fill_level`.

- In read direction, when the following condition is satisfied, switch to write direction.

- The number of write entries exceeds high threshold of WR CAM, and if `PERFWR1.w_max_starve > 0`, `PERFWR1.w_max_starve` timer is expired if previous read to write transition was done while WR CAM fill level exceeded the low threshold.
 - In Inline ECC configuration, the number of valid write ECC entries exceeds high threshold of WR ECC CAM.
 - In Inline ECC configuration, the number of loaded write ECC entries exceeds high threshold of WR ECC CAM and the number of write entries exceeds high threshold of WR ECC CAM.
 - In write direction, do not switch to read mode and do not prepare page of banks for read direction while number of write entries exceeds the low threshold of WR CAM unless `PERFWR1.w_xact_run_length` number of writes is served if `PERFWR1.w_max_starve > 0`, or the number of valid write ECC entries exceeds the low threshold of WR ECC CAM.
 - If the read to write switching is caused by the third condition, in write direction, do not switch to read mode and do not prepare page of banks for read direction until at least one write entry or write ECC entry is issued.
 - Expired-VPR and read collision are of higher priority than write CAM fill-level, and therefore it is subject to switch to read.
- For example, when the number of write entries exceed high threshold of WR CAM, switch to write direction. But, if a read collision occurs after that, switch back to read direction to flush the colliding command. Same for Expired-VPR as well.
- The priority is lower priority than write `Critical_L1`, but higher than write `Critical_L2` and write `Critical_L3`.

**Note**

- A write command cannot be scheduled-out if the corresponding write data is not provided. The write data is provided by the system (AXI or HIF) or read data path if this is RMW.
- The thresholds are for number of WR CAM entries with write data ready. Therefore, if provided the write data is delayed, it is possible situation that WR CAM is (almost) full in terms of credit, but WR CAM is (almost) empty in terms of number of WR CAM entries with write data ready.

7.3.4 Address Collision Handling

The DDRC can execute transactions out-of-order while ensuring that all transactions appear as if they are executed in the order in which they are received. Every transaction that requires a response from the DDRC arrives with a token number which is provided back to the SoC as part of the response. Since the DDRC queues transactions prior to execution, it is possible that multiple transactions to the same SDRAM address can arrive before the first transaction to that address is issued.

For address collision, two HIF addresses are considered the same address if all of the HIF address bits (except for the LSB column values) are the same. That is, the “collision” ignores the LSB column values when comparing addresses.

So the following HIF address bits are ignored during comparison.

- `HIF[3+addrmap_col_b3]`
- `HIF[2:0]`

To enforce ordering of accesses to the same address, the DDRC uses the following algorithm:

1. **New read colliding with queued read:** This collision causes no problems. The two reads can end up being executed out-of-order.
2. **New write colliding with queued write:** If write combine is enabled, the DDRC overwrites the data for the old write with that from the new write and only performs one write transaction (write combine). For more information, see [“Write Combine”](#) on page 160.
3. **New read (or write) colliding with queued write (or read) respectively:** In this case, the DDRC performs the following sequence:
 - a. Holds the new transactions in two deep temporary buffer.
 - b. Once the two deep temporary buffer is filled by subsequent command, applies flow control back to the SoC to prevent more transactions from arriving (hif_cmd_stall).
 - c. Flushes the internal queue holding the colliding transaction until that transaction is serviced.
 - d. Accepts the new transaction from the two deep temporary buffer and removes the flow control (hif_cmd_stall).
4. **New read colliding with both read and write:** This can happen when a read collides with an RMW command. In this case, the reads are flushed until the read collision is cleared, then the writes are flushed in the same manner as described in Step 3.
5. **New write colliding with both read and write:** This can happen when a write collides with an RMW command.
 - ❑ If write combine is enabled, the new write is combined with write part of queued RMW.
 - ❑ If write combine is disabled, the new write is held in two deep temporary buffer until the RMW has completed.
6. **New RMW colliding with queued write:** In this case, the new RMW is stored in two deep temporary buffer until the queued write is completed.

7.3.5 Write Combine

The write combine feature can combine multiple writes to the same address into a single write to SDRAM. When a new write collides with a queued write in the CAM:

- If write combine is enabled, the DDRC overwrites the data for the old write with that from the new write and only performs one write transaction (write combine).
- If write combine is disabled, the DDRC performs the following sequence:
 - a. Holds the new write transaction in a temporary buffer.
 - b. Applies flow control back to the SoC to prevent more transactions from arriving.
 - c. Flushes the internal queue holding the colliding transaction until that transaction is serviced.
 - d. Accepts the new transaction and removes flow control.

If for example, there are 5 transactions, namely A,B,B,B,C (A,B, and C are addresses):

- If write combine is enabled, 3 commands go to DRAM as follows: (A-B-C).
- If write combine is disabled, 5 commands go to DRAM as follows: (A-B-B-B-C).

**Note**

When a write combine occurs, QOS information (Write Classes, that is, NPW, VPW latency timer) of a queued write inside the DDRC is not updated by a new write from the HIF Interface.

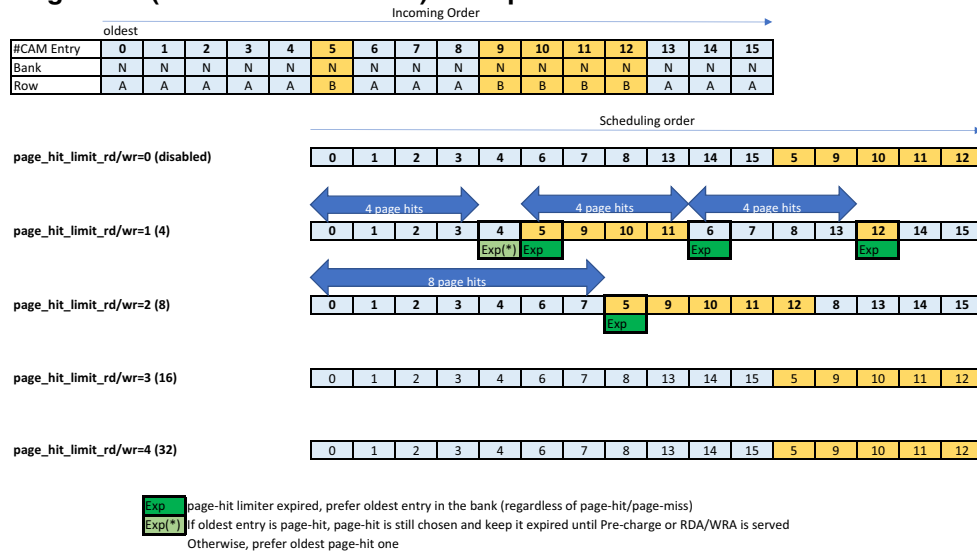
7.3.5.1 Page-hit Limiter

The Page-hit Limiter is an anti-starvation mechanism.

When this feature is enabled, after a certain number of page-hits are executed on a bank without PRE-ACT cycle, the next command loaded into the NTT is the oldest entry regardless of the page-hit or page-miss. This eliminates starvation caused by preferring page-hit.

Figure 7-14 shows an example incoming order (RD CAM or WR CAM).

Figure 7-14 Incoming Order (RD CAM or WR CAM) Example

**Note**

- This feature works only within same priority:
 - LPR and non-expired VPR
 - HPR
 - Expired VPR
 - NPW and non-expired VPW
 - Expired VPW
- Bypass read can happen if this is targeting to active bank even if the page-close timer is expired

7.3.5.2 Visible Window Limiter (UMCTL2_VPRW_EN==1 only)

The Visible Window Limiter feature is introduced to eliminate following extreme starvation:

- NPW against expired-VPW
- LPR against expired-VPR

■ HPR against expired-VPR

With the enhanced CAM pointer mechanism, the visible window (age differences between the newest entry and the oldest entry) can be more than `MEMC_NO_OF_ENTRY` (unlimited by definition). This feature works as follows:

- Each CAM entry has a counter for this feature.
- When the entry is pushed, the counter is initialized to the value set by register `SCHED1.visible_window_limit_rd/wr`.
- Whenever younger entry is scheduled-out from its CAM, the counter is decremented by one.
- Once the counter reaches to '0', the entry becomes expired VPR/expired VPW.
- Within exVPR or exVPWs, the priorities are the same, therefore, this can eliminate the extreme starvation.



Note

- This feature works also for VPR/VPW. Depending on the latency timer value, this feature makes it expired before the latency timer expires.
- This feature does not guarantee the actual visible window.

7.3.6 Registers Related to Transaction Service Control

The following are the registers related to the Transaction Service Control:

- Low priority read transaction store:
 - `PERFLPR1.lpr_max_starve`
 - `PERFLPR1.lpr_xact_run_length`
 - `SCHED1.page_hit_limit_rd`
 - `SCHED1.visible_window_limit_rd` (`UMCTL2_VPRW_EN == 1`)
- High priority read transaction store:
 - `PERFHPR1.hpr_max_starve`
 - `PERFHPR1.hpr_xact_run_length`
 - `SCHED1.page_hit_limit_rd`
 - `SCHED1.visible_window_limit_rd` (`UMCTL2_VPRW_EN == 1`)
- Write transaction store:
 - `PERFWR1.w_max_starve`
 - `PERFWR1.w_xact_run_length`
 - `SCHED1.page_hit_limit_wr`
 - `SCHED1.visible_window_limit_wr` (`UMCTL2_VPRW_EN == 1`)
- Bus turn-around control:
 - `SCHED0.prefer_write`
 - `SCHEDTMG0.rdwr_idle_gap`

For more information about these registers, see in the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

7.4 Quality of Service

This topic contains the following sections:

- [“Overview of Quality of Service”](#) on page 163
- [“Traffic Classes”](#) on page 163
- [“Dual Read Address Queue”](#) on page 165
- [“VPR/VPW Timeout”](#) on page 166
- [“Urgent Signaling”](#) on page 168
- [“Enabling QOS”](#) on page 168

7.4.1 Overview of Quality of Service

In a SoC based sub-systems, Quality of Service (QoS) are extremely important in distributing the available resources (bandwidth or latency) amongst the different devices, in a fair mechanism to meet the overall performance requirements.

In any SoC based on-chip communication system, there are multiple Managers trying to communicate with different Subordinates through Interconnect. Interconnect is used to route the traffic between the Managers and Subordinates. There are potential chances of bandwidth sensitive Manager starving for data and latency sensitive Manager requirements are not met. This impacts the overall performance of the sub-system. QoS addresses these kinds of performance issues in sub-system.

7.4.2 Traffic Classes

The DDRCTL supports different traffic classes that are distinguished from each other by the `argos` signal.

7.4.2.1 Read Classes

- **Low Priority Read (LPR):** It is also called as the best effort traffic. There is no resource allocation and it shares the resources with the other traffic types. LPR is always treated as low priority in the PA and in the DDRC. There are timeout mechanisms that can be used to prevent starvation for both the PA and the DDRC. When there is a timeout in the PA, that port becomes the highest priority (priority0). When there is a timeout in the LPR store in the DDRC (in other words, LPR store becomes critical), the LPR entries are served before HPR entries.
- **Variable Priority Read (VPR):** It is also called as the maximum latency bound traffic for meeting real time deadlines in video or audio applications. VPR traffic shares the same resources with LPR in the DDRC. But, based on the configuration, it may have a dedicated queue (red or blue) in the XPI. It has the same priority as LPR, but lower priority than HPR for the PA. For the DDRC, VPR has the same initial priority as LPR. Each command tagged as VPR has an associated latency timer. When expired, VPR transactions have the highest priority in the controller, both in the PA and in the DDRC. The purpose of this traffic class is to limit the maximum latency, where this latency bound is expected to be a few hundred clock cycles.
- **High Priority Read (HPR):** It has allocated resources. If the XPIs are configured to have dual read address queues, then the red queue can be allocated to HPR. For DDRC, HPR traffic goes to its own dedicated store. HPR traffic has higher priority than LPR. HPR is meant for latency critical but not real time applications such as CPU.

7.4.2.2 Write Classes

- Normal Priority Write (NPW): It is also called as the best effort traffic. There is no resource allocation and it shares the resources with the other traffic types. It is always treated as normal priority in the PA and DDRC. There are timeout mechanisms that can be used to prevent starvation for the PA. When there is a timeout in the PA, that port becomes the highest priority (priority0).
- Variable Priority Write (VPW): It is also called as the maximum latency bound traffic to meet real time deadlines in video or audio applications. VPW traffic shares the same resources with NPW in the DDRC. For the DDRC, VPW has the same initial priority as NPW. Each command tagged as VPW has an associated latency timer. When expired, VPW transactions have the highest priority in the controller, both in the PA and in the DDRC. The purpose of this traffic class is to limit the maximum latency, where this latency bound is expected to be a few hundred clock cycles.

7.4.2.3 QoS Mapping

The priority of an initiator (manager) is susceptible to change by the intermediate agents (for example, interconnect) before reaching the destination subordinate depending on the service levels provided with respect to the required targets. Therefore, the relationship between the IDs and their class representations must be dynamically determined.

arqos/awqos signal determines both port priorities and DDRC priorities dynamically. 16 QoS levels are divided to three regions for reads, two regions for writes. Each region can be assigned to any of the following traffic class—LPR, VPR, and HPR for reads, NPW and VPW for writes.

Region 0 and 1 are assigned to the blue address queue (see “Dual Read Address Queue” on page 165). Typically, LPR/NPW and VPR/VPW may share this resource.

Region 2 is assigned to the red queue for reads (when dual queue is selected), while for writes it is assigned to the same queue as Region 0 and Region 1 (being writes always single queue). Region 2 can be mapped to HPR or VPR traffic for reads, VPW or NPW traffic for writes. In the DDRC, LPR/NPW and VPR/VPW share the same CAM store called LPR/NPW store. HPR has its own store in the DDRC.

The regions are mapped per port using `PCFGQOS0_n` and `PCFGWQOS0_n` registers (see “REGB_ARB_PORTp” in “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide).

Fields for `PCFGQOS0_n` register are as follows:

- `rqos_map_level<x>` (where x is 1 to 2) indicates two separation levels for three regions. Possible values 0 to 14 correspond to an arqos value. These registers indicate the upper end of the region. For example, if `rqos_map_level2` is set to 14, then Region 2 is identified as all transactions with arqos value of 15.
- `rqos_map_region<y>` (where y is 0 to 2) for region identifier. This register indicates the traffic class of each of the three regions. `rqos_map_region2` is present only in dual read queue configurations.

Valid values are:

- 0-LPR
- 1-VPR
- 2-HPR

For dual read address queue configurations, Region 0 and Region 1 map to the blue queue, and Region 2 maps to the red queue. The blue queue can only be set to LPR, VPR or both. The red queue can only be set to VPR or HPR.

For single address queue configurations, Region 0 and Region 1 map to the blue queue. Being single queue, Region 2 is not present. In this case registers `rqos_map_level2` and `rqos_map_region2` do not exist. In this case Region 0 can be set to HPR or VPR, Region 1 to VPR or LPR. Fields for `PCFGWQOS0_n` register are as follows:

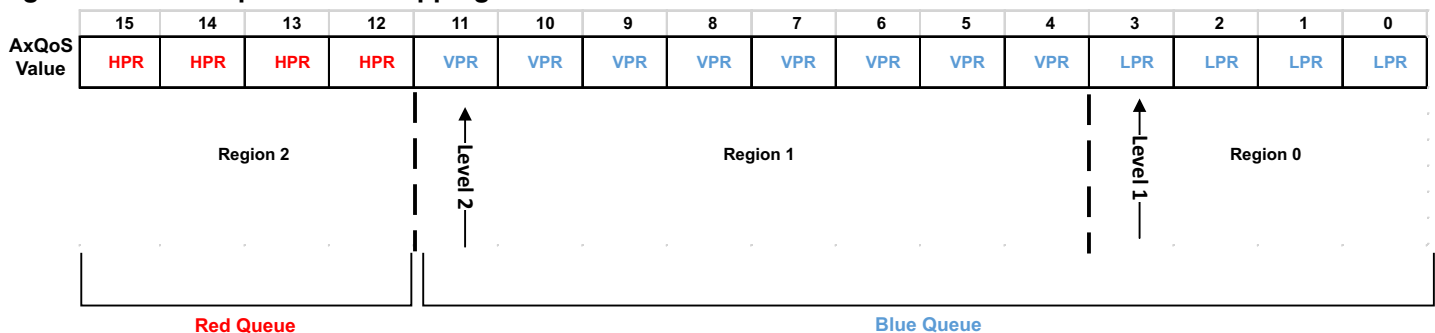
- `wqos_map_level<x>` (where x is 1 to 2) indicates the separation level for three regions. Possible values 0 to 14 correspond to an `awqos` value. These registers indicate the upper end of the region. For example, if `wqos_map_level2` is set to 14, then Region 2 is identified as all transactions with `awqos` value of 15.
- `wqos_map_region<y>` (where y is 0 to 2) for region identifier. This register indicates the traffic class of each of the two regions.

Valid values are:

- 0-NPW
- 1-VPW

Writes are always single queue: Region 0, Region 1 and Region 2 map all to the same queue. All fields always exist. Figure 7-15 shows an example dynamic mapping from AXI QoS input to internal controller traffic classes.

Figure 7-15 Example AxQoS Mapping to Traffic Classes



7.4.3 Dual Read Address Queue

There are two separate read address queues in the XPI block—red and blue queue. They can be enabled by the parameter `UMCTL2_XPI_USE2RAQ_n` for each port.

Both read queues request independently to the PA; therefore if enabled, maximum configurable number of XPI ports which is 16 is reduced. Each dual-queue XPI consumes two consecutive PA ports. The worst case is that there can be 8 ports where each XPI is configured to have dual read address queue.

Red queue can be used for one traffic class only and associated to Region 2 in the QoS mapper (HPR or VPR). Blue queue can be used for two traffic classes and associated to Region 0 and Region 1 in the QoS mapper (VPR and/or LPR).

There are separate time-outs for the blue and red queues. Timers are started to down count when the transaction is accepted in the XPI, which are then forwarded to DDRC together with the command (see “VPR/VPW Timeout” on page 166).

Optional retiming block (register slice for pipelining) at the input of the XPI read address channel is enabled by `UMCTL2_XPI_USE_INPUT_RAR`. Optional retiming block at the output of the XPI read address channel (enabled by `UMCTL2_XPI_USE_RAR`) cannot be used if dual queue is enabled.

When Dual Read Address Queue is selected (`UMCTL2_XPI_USE2RAQ_n==1`) and AXI Read Data Interleaving is disabled (`UMCTL2_READ_DATA_INTERLEAVE_EN_n==0`), there is a restriction: the number of RRB virtual channels (`UMCTL2_NUM_VIR_CH_n`) defaults to the number of CAM entries (`MEMC_NO_OF_ENTRY`). This is necessary to guarantee the `UMCTL2_READ_DATA_INTERLEAVE_EN_n==0` functionality and is managed automatically. If AXI Read Data Interleaving is enabled, there is no such restriction.

7.4.3.1 ID Collisions

Due to the dynamic nature of the QoS with respect to the IDs, two transactions of the same ID can potentially exist in two queues. The collisions can be classified as:

- Blue after Red (BAR): A command with ID *x* is presented to the blue queue where a command with the same ID *x* is outstanding in the red queue.
- Red after Blue (RAB): A command with ID *x* is presented to the red queue where a command with same ID *x* is outstanding in the blue queue.

In the XPI, for both collision types, the ordering consistency within a given ID is preserved by stalling the incoming transaction to be pushed into the address queue it is mapped to until the colliding transaction in the opposite queue no longer exists. This is required for functional correctness.

In case of any ID collision, the DDRCTL does not speed up the drain of the stored transactions in the address queue causing collision even if the incoming transaction is HPR. In other words, address queues request arbitration to the Port Arbiter with their normal mapped priorities.

7.4.4 VPR/VPW Timeout

There are separate timeouts for the blue and red queues for read transactions, and one timeout for write transactions. Timers are started to down count when the transaction is accepted in the XPI, which are then forwarded to DDRC together with the command.

Timeouts are set per port and queue using `PCFGQOS1_n` and `PCFGWQOS1_n` registers (see “REGB_ARB_PORTp” in “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide).

For `PCFGQOS1_n` register:

- `rqos_map_timeoutb`: specifies the timeout value for transactions mapped to the blue queue.
- `rqos_map_timeoutr`: specifies the timeout value for transactions mapped to the red queue.

For `PCFGWQOS1_n` register:

- `wqos_map_timeout1`: specifies the timeout value for write transactions for region 0 and 1.
- `wqos_map_timeout2`: specifies the timeout value for write transactions for region 2.

When expired, VPR/VPW transactions from XPI are tagged as expired-VPR or expired-VPW instead of LPR/NPW during normal priority transaction flow. The Port Arbiter treats these transactions with the highest priority (priority0). In addition, the Port Arbiter asserts `hif_go2critical_lpr` signal to the DDRC if there are no LPR credits available (LPR store of the read CAM is full) when an expired-VPR port is pending, and `hif_go2critical_wr` signal to the DDRC if there are no write credits available when an expired-VPW is pending. If an expired-VPW is issued as RMW at the HIF interface, the Port Arbiter asserts `hif_go2critical_lpr` signal to the DDRC if there are no LPR credits, and/or `hif_go2critical_wr`

signal if there are no write credits available. If both `hif_go2critical_lpr` and `hif_go2critical_wr` are asserted at the same time, priority in the DDRC is given to the read.

If VPR/VPW timeout registers are set to '0', the VPR/VPW transactions expire immediately as they enter the DDRCCTL, thereby making them the highest priority transaction class within the device.

If multiple VPR/VPW transactions expire at the same time in the XPI, they are executed by the PA in round robin order.

To understand how the DDRC handles the VPR/VPW transactions, see [“Overview of Transaction Service Control”](#) on page 143.

7.4.4.1 Head of Line Blocking

Blue queue can be assigned to LPR and/or VPR if single or dual queue is selected, HPR and/or VPR if single queue is selected. Head of line blocking can occur in the XPI if more than one traffic class is mapped to the same address queue.

If a VPR expires inside the read address queue there can be two scenarios:

- Expired-VPR is blocked by one or more LPR transactions (single or dual queue configuration)
- Expired-VPR is blocked by one or more HPR transactions (only single queue configuration)

In case an LPR is blocking, that port becomes the highest priority (priority0) and if there are no LPR credits available, `hif_go2critical_lpr` is asserted.

In case an HPR is blocking, that port becomes the highest priority (priority0) and if there are no HPR credits available, `hif_go2critical_hpr` is asserted.

If a VPW expires inside the write address queue, expired-VPW may be blocked by one or more NPW or a non expired VPW transaction.

In case another transaction is blocking, that port becomes the highest priority (priority0) and if there are no write credits available, `hif_go2critical_wr` is asserted.

This behavior allows the queue where the VPR/VPW is expired to be flushed as quickly as possible.

This mechanism is applied only for the expired-VPR and expired-VPW commands.

For single read address queue configurations, HPR commands may be blocked by LPR or VPR commands (not-expired); in this case, the port priority is not increased.

Even in configurations with Data channel interleaving enabled (UMCTL2_DATA_CHANNEL_INTERLEAVE_EN set to '1'), if a VPR/VPW expires within the XPI, that port becomes the highest priority - priority0. An expired VPR can be blocked by one or more LPR or HPR commands. If a VPW expires within the XPI, it can be blocked by a NPW or unexpired VPW command.

- If a VPR expires within XPI and a HPR is blocking it, that port becomes the highest priority (priority0) and if there are no HPR credits available, `hif_go2critical_hpr` is asserted to one or both the data channels.
- If a VPR expires within XPI and a LPR is blocking it, that port becomes the highest priority (priority0) and if there are no LPR credits available, `hif_go2critical_lpr` is asserted to one or both the data channels.
- If a VPW expires within the XPI and a NPW or an unexpired VPW is blocking it, that port becomes the highest priority (priority0) and if there are no write credits available, then `hif_go2critical_wr` is

asserted to one or both the data channels. This will ensure that the queue with the expired VPR/VPW will be flushed as quickly as possible.

7.4.5 Urgent Signaling

AXI off-band signals per port, `arurgent` (`arurgentb`, `arurgent_r` when `UMCTL2_XPI_USE2RAQ_n` is defined) and `awurgent`, when asserted (and as long as asserted), set a given port to the highest priority (priority0) and when applicable- cause the read/write direction to switch immediately in the Port Arbiter if enabled by `PCFGR.rd_port_urgent_en` and `PCFGW.wr_port_urgent_en` registers (see “REGB_ARB_PORTp” in “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide). Urgent signals also cause the `hif_go2critical_wr` / `hif_go2critical_lpr` / `hif_go2critical_hpr` signals to be asserted at the HIF interface, which in turn force the read/write direction switching in the DDRC, if enabled by the `PCFG.go2critical_en` register. Urgent signals are ignored by the PA if there are no requesters from the asserted address channel and port. Similarly, `hif_go2critical*` signals are ignored by the DDRC if the corresponding store is empty. When urgent feature is enabled for reads by `PCFGR.rd_port_urgent_en`, `arurgent_n` signal causes a `hif_go2critical_hpr` signal to be asserted at the HIF interface if the read command at the head of the RAQ (Read Address Queue) is of HPR traffic class. If the read command at the head of the RAQ is not of HPR traffic class, then this `arurgent_n` signal causes a `hif_go2critical_lpr` signal to be asserted at the HIF interface.

7.4.6 Enabling QOS

The feature-specific to QOS can be configured in the coreConsultant GUI under the DDRC parameters/QOS option.

8

Memory Scheduling

This chapter contains the following sections:

- [“Burst Mode Operation”](#) on page 170
- [“Dynamic SDRAM Constraints”](#) on page 172

8.1 Burst Mode Operation

This topic contains the following sections:

- [“Overview of Burst Mode Operation”](#) on page 170
- [“Bus Width Selection”](#) on page 170
- [“Sequential Operations”](#) on page 170

8.1.1 Overview of Burst Mode Operation

DDRCTL supports BL16 for LPDDR5/4/4X.

8.1.2 Bus Width Selection

The DDRCTL allows you to select whether all or part of the DQ data-bus width is connected to the SDRAM. This allows the DDRCTL to connect to “full width” SDRAMs (where the width of the DQ bus is equal to the width configured in `MEMC_DRAM_DATA_WIDTH`), half of that width, or a quarter of that width. To select the DQ data bus width, set `MSTR0.data_bus_width`.

In case of half bus width mode, the half data width on the HIF is used.



Note

In configurations that contain the AXI Port Interface, the XPI block generates multiple bursts on the HIF interface.

The DDRCTL does not support the optional DFI signal `dfi_data_byte_disable`, so it may be necessary to program the PHY to indicate that certain bytes are not used. If a Synopsys DWC DDR PHY is used, program the `DXnGCR.DXEN` register bit to '0'.

8.1.3 Sequential Operations

The DDRCTL provides support for sequential burst mode operations.

SDRAM writes are always executed as aligned operations on the DFI interface in all modes. If the HIF write has an unaligned address, DDRCTL reorders the data so that the data beats appear in the correct order when the aligned write is done on the DFI interface (and hence the SDRAM interface).

[Table 8-1](#) shows examples of addressing for sequential and interleaved burst modes.

This table applies to HIF-only configurations (`DDRCTL_SYS_INTF = 0`). For Arbiter configurations (`DDRCTL_SYS_INTF = 1`), a fixed addressing sequence is used. Therefore, `MSTR.burst_mode` does not exist for Arbiter configurations.



Note

For a sequential burst, bit 2 to bit 0 of the starting HIF address must be set to 'b000'.

Table 8-1 Addressing for MSTR0.burst_rdw = 4'b01000 (BL16) in MEMC_BURST_LENGTH = 16 (LPDDR4/5³ Configurations)

Starting HIF Address (A3 ^a A2 A1 A0)	Starting SDRAM Address (A3 A2 A1 A0)	Sequential Addressing
0000	0000	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
1000	1000	8,9,A,B,C,D,E,F,0,1,2,3,4,5,6,7

a. HIF[3] must be mapped to SDRAM A3 as ADDRMAP6.addrmap_col_b3 = 0



Note

For LPDDR4/5, only Sequential Addressing is supported.

For LPDDR5 read, only HIF[3:0] = 0000 is supported (CAS-B3 command is not supported).

8.2 Dynamic SDRAM Constraints

This topic contains the following sections:

- [“Overview of Dynamic SDRAM Constraints”](#) on page 172
- [“Timing Constraints”](#) on page 172

8.2.1 Overview of Dynamic SDRAM Constraints

Dynamic SDRAM constraints are the restrictions placed on the transaction scheduler by the rules of the SDRAM specification. The exact value of each constraint is programmable and varies with the specification for the exact SDRAM parts being used. These constraints must be set up before traffic is sent to the DDRCTL.

Dynamic SDRAM constraints can be subdivided into three basic categories:

- Bank constraints: It affects the transactions that can be scheduled to a given bank. See [“Timing Constraints”](#) on page 172.
- Rank constraints: It affects the transactions that can be scheduled to a given rank.
- Global constraints: It affects all the transactions.

Write latency (WL) and read latency (RL) descriptions for different SDRAMs are defined as follows:

- For LPDDR4:
 - RL and WL are directly defined in the mode register.

For 1:2 frequency ratio mode, the following limitations apply on the DFI bus:

- All commands are sent from the lower half of the bus, corresponding to the even cycle on the SDRAM bus (LPDDR4 only).

For 1:4 frequency ratio mode, the following limitations apply on the DFI bus (LPDDR4 only):

- All commands except refresh are sent on the first phase of the bus, corresponding to the $4n$ cycle on the SDRAM bus.
- Refresh command is sent on the third phase of the bus, corresponding to the $4n+2$ cycle on the SDRAM bus.

As a result of these limitations gaps occur between some commands on the SDRAM bus being one cycle longer than programmed by the timing registers as described in this section.

8.2.2 Timing Constraints

Sections [“LPDDR4 Command Timing Constraints”](#) on page 172, and [“LPDDR5 Command Timing Constraints”](#) on page 174 describe command timing constraints for LPDDR4, and LPDDR5 respectively.

8.2.2.1 LPDDR4 Command Timing Constraints

[Table 8-2](#) to [Table 8-4](#) list the command timing control registers and command timing constraints for LPDDR4.

Table 8-2 LPDDR4 Command Timing Control Registers

Control Register	Value
DRAMSET1TMG4.t_ccd	8
DRAMSET1TMG13.t_ccd_mw	tCCDMW
DRAMSET1TMG4.t_rcd	$RU(tRCD/tCK)$
DRAMSET1TMG2.wr2rd	$WL + 1 + BL/2 + RU(tWTR/tCK)$
DRAMSET1TMG0.t_ras_min	$RU(tRAS/tCK)$
DRAMSET1TMG1.rd2pre	$BL/2 + \max\{8, RU(tRTP/tCK)\} - 8$
DRAMSET1TMG0.wr2pre	$WL + 1 + BL/2 + RU(tWR/tCK)$
DRAMSET1TMG4.t_rp	$RU(tRP/tCK)$
DRAMSET1TMG4.t_rrd	$RU(tRRD /tCK)$
DRAMSET1TMG13.t_ppd	tPPD
DRAMSET1TMG25.rda2pre	
DRAMSET1TMG2.rd2wr (Refer to Table 101 in JESD209-4C)	DQ ODT is disabled $RL + RU(tDQSCK(max)/tCK) + BL/2 - WL + tWPRE + RD(tRPST)$
	DQ ODT is enabled $RL + RU(tDQSCK(max)/tCK) + BL/2 + RD(tRPST) - ODTLon - RD(tODTon,min/tCK) + 1$

Table 8-3 Command Timing Constraints for Same Banks

Next CMD Current CMD	Active	Read	Write	Mask Write	Precharge
ACTIVE	Illegal	t_rcd	t_rcd	t_rcd	t_ras_min
READ	Illegal	t_ccd	rd2wr	rd2wr	rd2pre
WRITE	Illegal	wr2rd	t_ccd	t_ccd_mw	wr2pre
MASK WRITE	Illegal	wr2rd	t_ccd	t_ccd_mw	wr2pre
PRECHARGE	t_rp	Illegal	Illegal	Illegal	t_ppd
READ with AP	rda2pre + t_rp	Illegal	Illegal	Illegal	n/a
WRITE with AP	wra2pre + t_rp	Illegal	Illegal	Illegal	n/a
MASK WRITE with AP	wra2pre + t_rp	Illegal	Illegal	Illegal	n/a

Table 8-4 Command Timing Constraints for Different Banks

Next CMD Current CMD	Active	Read	Write	Mask Write	Precharge
ACTIVE	t_rrd	1	1	1	1
READ	1	t_ccd	rd2wr	rd2wr	1
WRITE	1	wr2rd	t_ccd	t_ccd	1
MASK WRITE	1	wr2rd	t_ccd	t_ccd	1
PRECHARGE	1	1	1	1	2
READ with AP	1	t_ccd	rd2wr	rd2wr	1
WRITE with AP	1	wr2rd	t_ccd	t_ccd	1
MASK WRITE with AP	1	wr2rd	t_ccd	t_ccd	1

8.2.2.2 LPDDR5 Command Timing Constraints

Sections “[BG Mode](#)” on page 174 and “[16B Mode](#)” on page 176 describe the LPDDR5 command timing constraints for BG mode and 16B mode, respectively.

8.2.2.2.1 BG Mode

[Table 8-5](#) to [Table 8-11](#) list the command timing control registers and command timing constraints for LPDDR5 in BG mode.

Table 8-5 LPDDR5 Command Timing Control Registers in BG Mode

Control Register	Value
t_ccd	BL/n (same BG)
t_ccd_s	BL/n (different BG)
t_ccd_mw	4*BL/n_max
t_rcd	RU(tRCD/tCK)
wr2rd	WL + BL/n_max + RU(tWTR_L/tCK)
wr2rd_s	WL + BL/n_min + RU(tWTR_S/tCK)
t_ras_min	RU(tRAS/tCK)
rd2pre	BL/n_min + RU(tRBTP/tCK)
wr2pre	WL + BL/n_min + RU(tWR/tCK)
t_rp	RU(tRPpb/tCK)
t_rrd	RU(tRRD_L/tCK)

Control Register	Value
t_rrd_s	$RU(tRRD_S/tCK)$
t_ppd	2
DQ ODT is disabled	
rd2wr	$RL + BL/n_max + RU(tWCKDQO(max)/tCK) - WL$
rd2wr_s	$RL + BL/n_min + RU(tWCKDQO(max)/tCK) - WL$
DQ ODT is enabled	
rd2wr	$RL + BL/n_max + RU(tWCKDQO(max)/tCK) + RD(tRPST/tCK) - ODTLon - RD(tODTon(min)/tCK)$
rd2wr_s	$RL + BL/n_min + RU(tWCKDQO(max)/tCK) + RD(tRPST/tCK) - ODTLon - RD(tODTon(min)/tCK)$

Table 8-6 Command Timing Constraints for Same Banks in Same Bank Group

Next CMD Current CMD	Active	Read	Write	Mask Write	Precharge
ACTIVE	Illegal	t_rcd	t_rcd	t_rcd	t_ras_min
READ	Illegal	t_ccd	rd2wr	rd2wr	rd2pre
WRITE	Illegal	wr2rd	t_ccd	t_ccd_mw	wr2pre
MASK WRITE	Illegal	wr2rd	t_ccd	t_ccd_mw	wr2pre
PRECHARGE	t_rp	Illegal	Illegal	Illegal	t_ppd
READ with AP	rda2pre + t_rp	Illegal	Illegal	Illegal	n/a
WRITE with AP	wra2pre + t_rp	Illegal	Illegal	Illegal	n/a
MASK WRITE with AP	wra2pre + t_rp	Illegal	Illegal	Illegal	n/a

Table 8-7 Command Timing Constraints for Different Banks in Same Bank Group

Next CMD Current CMD	Active	Read	Write	Mask Write	Precharge
ACTIVE	t_rrd	1	1	1	1
READ	1	t_ccd	rd2wr	rd2wr	1
WRITE	1	wr2rd	t_ccd	t_ccd	1
MASK WRITE	1	wr2rd	t_ccd	t_ccd	1
PRECHARGE	1	1	1	1	2
READ with AP	1	t_ccd	rd2wr	rd2wr	1

Next CMD Current CMD	Active	Read	Write	Mask Write	Precharge
WRITE with AP	1	rd2wr	t_ccd	t_ccd	1
MASK WRITE with AP	1	rd2wr	t_ccd	t_ccd	1

Table 8-8 Command Timing Constraints for Different Bank Groups

Next CMD Current CMD	Active	Read	Write	Mask Write	Precharge
ACTIVE	t_rrd	1	1	1	1
READ	1	t_ccd_s	rd2wr_s	rd2wr_s	1
WRITE	1	wr2rd_s	t_ccd_s	t_ccd_s	1
MASK WRITE	1	wr2rd_s	t_ccd_s	t_ccd_s	1
PRECHARGE	1	1	1	1	t_ppd
READ with AP	1	t_ccd_s	rd2wr_s	rd2wr_s	1
WRITE with AP	1	wr2rd_s	t_ccd_s	t_ccd_s	1
MASK WRITE with AP	1	wr2rd_s	t_ccd_s	t_ccd_s	1

8.2.2.2.2 16B Mode

[Table 8-9](#) to [Table 8-11](#) list the command timing control registers and command timing constraints for LPDDR5 in 16B mode.

Table 8-9 LPDDR5 Command Timing Control Registers in 16B Mode

Control Register	Value
t_ccd	BL/n
t_ccd_mw	4* BL/n
t_rcd	RU(tRCD/tCK)
wr2rd	WL + BL/n + RU(tWTR /tCK)
t_ras_min	RU(tRAS/tCK)
rd2pre	BL/n + RU(tRBTP/tCK)
wr2pre	WL + BL/n + RU(tWR/tCK)
t_rp	RU(tRP/tCK)
t_rrd	RU(tRRD /tCK)
t_ppd	2

Control Register	Value
rd2wr	DQ ODT is disabled $RL + BL/n + RU(tWCKDQO(max)/tCK) - WL$ DQ ODT is enabled $RL + BL/n + RU(tWCKDQO(max) /tCK) +$ $RD(tRPST/tCK) - ODTLon - RD(tODTon(min)/tCK)$

Table 8-10 Command Timing Constraints for Same Banks

Next CMD Current CMD	Active	Read	Write	Mask Write	Precharge
ACTIVE	Illegal	t_rcd	t_rcd	t_rcd	t_ras_min
READ	Illegal	t_ccd	rd2wr	rd2wr	rd2pre
WRITE	Illegal	wr2rd	t_ccd	t_ccd_mw	wr2pre
MASK WRITE	Illegal	wr2rd	t_ccd	t_ccd_mw	wr2pre
PRECHARGE	t_rp	Illegal	Illegal	Illegal	t_ppd
READ with AP	rda2pre + t_rp	Illegal	Illegal	Illegal	n/a
WRITE with AP	wra2pre + t_rp	Illegal	Illegal	Illegal	n/a
MASK WRITE with AP	wra2pre + t_rp	Illegal	Illegal	Illegal	n/a

Table 8-11 Command Timing Constraints for Different Banks

Next CMD Current CMD	Active	Read	Write	Mask Write	Precharge
ACTIVE	t_rrd	1	1	1	1
READ	1	t_ccd	rd2wr	rd2wr_	1
WRITE	1	wr2rd	t_ccd	t_ccd	1
MASK WRITE	1	wr2rd	t_ccd	t_ccd	1
PRECHARGE	1	1	1	1	t_ppd
READ with AP	1	t_ccd	rd2wr	rd2wr	1
WRITE with AP	1	wr2rd	t_ccd	t_ccd	1
MASK WRITE with AP	1	wr2rd	t_ccd	t_ccd	1

8.2.2.3 Limitation With DRAMSET1TMG4.t_rcd = 1

The input `DRAMSET1TMG4.t_rcd` indicates the minimum time from Activate to a read or write to the same bank. The DDRCTL has the following limitation when `DRAMSET1TMG4.t_rcd` is programmed to '1'.

In 1:2 frequency ratio mode setting this register is programmed to '1' when the Activate to Read/Write timing requirement at the DRAM is either one or two cycles. The DDRCTL puts a minimum of two `core_ddrc_core_clk` cycles between an Activate and Read/Write request, and this results in four SDRAM clock cycles gap at the SDRAM interface. As with 1:1 frequency ratio mode configurations, this is not a violation of DRAM protocol, but results in sub-optimal performance.

This limitation can be avoided by programming DRAM Additive Latency appropriately as described as follows:

`DRAMSET1TMG4.t_rcd` is programmed as: `tRCD - AL`

where `tRCD` is RAS-to-CAS delay of the DRAM, and `AL` is Additive Latency.

8.2.2.4 Byte Mode (x8) Consideration

DDRCTL supports byte mode (x8) devices, however the mixed package defined in JEDEC specification, which has both byte mode devices and x16 device, is not supported.

For details, refer to [Figure 5-1](#) on page 104, [Figure 5-4](#) on page 107, and [Figure 5-7](#) on page 112 in the section “LPDDR5/4/4X Configurations” on page 103.

To support byte mode devices, the following timing parameters must be calculated properly and used to set appropriate DDRCTL registers:

- LPDDR4: `WL`, `RL`, `nWR`, `nRTP`, `tWR`, `tWTR`.
- LPDDR5: `RL`, `nWR`, `nRBTP`, `tWR`, `tWTR` (`tWTR_S`, `tWTR_L` for BG mode), `tWCKENL_RD`.

Note, that the following Mode Registers in SDRAMs are not directly related to the DDRCTL behavior, but they need to be programmed by software to support byte mode Vref/ODT/DQ Calibration/Training appropriately during initialization:

- LPDDR4
 - MR12: CBT mode for byte mode
 - MR22: ODTD for x8 2ch (Byte) mode
 - MR32: Byte mode Vref Selection
- LPDDR5
 - MR12: VBS (V REF (CA) Byte Select)
 - MR14: VDLC (V REF DQ Lower byte copy)
 - MR17: X8 ODTD Lower (CA/CS/CK ODT termination disable, Lower Byte select), X8 ODTD Upper (CA/CS/CK ODT termination disable, Upper Byte select)
 - MR30: DCA for Lower Byte (DCAL), DCA for Upper Byte (DCAU)
 - MR31: Lower-byte per-bit control Register for DQ Calibration
 - MR32: Upper-byte per-bit control Register for DQ Calibration

8.2.3 Preamble and Postamble (LPDDR4)

8.2.3.1 LPDDR4 Constraints

LPDDR4 supports write preamble of $2 \cdot t_{CK}$ and programmable write postamble of 0.5 tCK or 1.5 tCK:

- Write postamble is selected through `MR3[1] - adjust INT4.emr2`.
- If write postamble of 1.5 tCK is used, the following timing values are affected and you must program the registers accordingly:
 - Adjust `RANKTMG0.diff_rank_wr_gap` for multi-rank systems: program this with N+1.

LPDDR4 supports read preamble of $2 \cdot t_{CK}$ and programmable read postamble of 0.5 tCK or 1.5 tCK:

- Read preamble is selected "Static" or "Toggle" through `MR1[3]`. Both preambles are $2 \cdot t_{CK}$. - `adjust INITTMG3.mr`.
- Read postamble is selected through `MR1[7] - adjust INITTMG3.mr`.
- If read postamble of 1.5 tCK is used, the following timing values are affected and you must program the registers accordingly:
 - Adjust `RANKTMG0.diff_rank_rd_gap` for multi-rank systems: program this with N+1.

9

Periodic Memory and PHY Maintenance

This chapter contains the following sections:

- [“Refresh Controls”](#) on page 182
- [“Refresh Management \(RFM\)”](#) on page 189
- [“ZQ Calibration”](#) on page 191
- [“Controller Assisted Drift Tracking \(LPDDR5\)”](#) on page 198
- [“Enhanced Incremental Periodic Phase Training \(PPT2\)”](#) on page 200

9.1 Refresh Controls

This topic contains the following sections:

- [“Overview of Refresh Controls”](#) on page 182
- [“Refresh Using Direct Software Request of Refresh Command”](#) on page 182
- [“Refresh Using Auto-Refresh Feature”](#) on page 183
- [“Automatic Temperature Derating”](#) on page 186
- [“Signals Related to Refresh Controls”](#) on page 188
- [“Registers Related to Refresh Controls”](#) on page 188
- [“Dynamic SDRAM Rank Constraints”](#) on page 188

9.1.1 Overview of Refresh Controls

Refresh can be issued using the auto-refresh feature in the DDRCTL or using the direct software request of the refresh command. The `RFSHCTL0.dis_auto_refresh` register bit selects the refresh method.



Note

Even if the contents of DRAM are not to be preserved, some devices may require periodic refresh operations while VDDQ, VDD, and VPP are asserted and the DRAM state is IDLE or Powerdown. See the DRAM documentation before configuring it.

The purpose of refresh control is to:

- Reduce the bandwidth impact of refresh cycles.
- Increase the likelihood of refreshes being serviced during idle periods.
- Provide fine-gain control of the trading-off the previous benefits (to gather refreshes) versus the increased worst case latencies associated with gathering refreshes.
- Allow traffic to flow to other ranks while a given rank is being refreshed (for multi-rank configurations of the DDRCTL only).

9.1.2 Refresh Using Direct Software Request of Refresh Command

Follow these steps to put the DDRCTL in direct software request of refresh command mode:

1. Set the `RFSHCTL0.dis_auto_refresh` bit to '1'. When the register bit set, the DDRCTL checks for any pending refreshes. Any pending refreshes are issued right away using the 'critical refresh' feature inside the DDRCTL. After these refreshes are issued, all the refresh timers inside the DDRCTL are reset to '0'. They are re-activated only when the auto-refresh feature is enabled.

Any pending refreshes are issued in same way, in this case after exiting self-refresh.

2. The SoC must keep track of the refresh requirements of the SDRAM.
3. The refresh command can be issued by setting the register bits `OPCTRLCMD.rank*_refresh` to '1'. When the `rank*_refresh` request is stored in the DDRCTL, the corresponding register bit is automatically cleared. The SoC can initiate a `rank*_refresh` operation only if `OPCTRLSTAT.rank*_refresh_busy` is low. The DDRCTL issues refresh to the SDRAM at the earliest. If `DIMMCTL.dimmm_stagger_cs_en` is enabled with `MSTR0.ddr4 == 0`, and the refresh command is destined for both even and odd ranks, the DDRCTL issues two refresh commands, one to even and one to odd ranks.

4. Software-driven refresh commands for each rank are loaded into a buffer, and are issued by the DDRCTL on the DFI as soon as it is legal to do so (the DDRCTL controller must wait $t_{RFC(min)}$ between each refresh request). If the buffer saturates, the `OPCTRLSTAT.rank*_refresh_busy` remains asserted to prevent software from initiating further refreshes.

Depending on all-bank or per-bank refresh mode, the buffer size is different.

Table 9-1 Buffer Size Depending on All-Bank and Per-Bank Refresh Modes

All-Bank and Per-Bank Refresh Mode	Buffer Size
Per-bank refresh	65 entries
All-bank refresh	9 entries



Note

- You can give a burst of back-to-back refresh commands without polling the `OPCTRLSTAT.rank*_refresh_busy` register field to reduce the time between the software refresh commands to a minimum with the following risk:
 - If the refresh buffer is not empty when the burst starts or the burst is longer than 9 refresh commands, the buffer saturates (`OPCTRLSTAT.rank*_refresh_busy` is asserted) and some APB writes given in the burst are not added to the buffer. This means that the number of APB writes do not reflect the number of REF commands sent out to the memory.
- For LPDDR4/LPDDR5 configurations manual refreshes (`RFSHCTL0.dis_auto_refresh = 1`) may not be used if the PHY Master Interface is enabled (`DFIPHYMSTR.dfi_phymstr_en = 1`).

9.1.3 Refresh Using Auto-Refresh Feature

The DDRCTL provides advanced refresh controls. Besides fully-configurable refresh constraints ($t_{RFC(min)}$ and t_{REFI}), the DDRCTL can also be programmed to gather refreshes to each rank of SDRAM to reduce the bandwidth consumed by refreshes and to increase the likelihood that refreshes can be serviced during an idle period.

Fine-grain control of the refreshes ensures these benefits can be balanced against worst case latencies associated with servicing refreshes together. Staggered refresh timers for multi-rank configurations of the DDRCTL allow transactions to continue to other ranks while refreshes are taking place to just one rank.

To minimize the worst case impact of a forced refresh cycle, the DDRCTL can be programmed to issue single refreshes at a time by forcing `RFSHMOD0.refresh_burst = 0`. It can be programmed to burst up to 8 refreshes (`RFSHMOD0.refresh_burst = 7`). With per-bank refresh enabled (`RFSHMOD0.per_bank_refresh = 1`), the maximum refresh burst supported by DDRCTL is 64 (`RFSHMOD0.refresh_burst = 63`).

9.1.3.1 Single Refresh

When using single refresh (`RFSHMOD0.refresh_burst = 0`), the DDRCTL issues refreshes every time the refresh timer (t_{REFI}) expires. This is the optimal mode of operation for systems that minimize the maximum latency associated with refresh cycles.

9.1.3.2 Burst Refresh

When burst refresh is enabled (`RFSHMOD0.refresh_burst > 0`), the DDRCTL issues refreshes in bursts of (`RFSHMOD0.refresh_burst+1`) refreshes at one time. Bursting refreshes reduces the total latency associated with those refreshes by reducing the number of pre-charges and activates required for refresh, as banks must be precharged only once to perform the entire group of refreshes, instead of once for each refresh.

When explicit precharge commands are required, they are individual precharge (PRE) commands to each open bank.

9.1.3.3 Speculative Refresh

When t_{REFI} has expired at least once, the DDRCTL can also perform speculative refreshes. This is done by automatically inserting refreshes when there are no transactions pending in the CAM to a rank/bank address.

The `RFSHTMG0.refresh_to_x1_x32` register determines how long the transactions must not be stored in the CAM before considering inserting these speculative refreshes. Each time a speculative refresh is performed, the count of t_{REFI} expirations is decremented, and thereby increasing the time before a critical refresh is required. This also ensures that speculative refreshes never occur more often than is required to keep the SDRAM properly refreshed.

If a new read or write transaction is accepted by the DDRCTL during speculative refresh, the DDRCTL services it as soon as legally possible. Most often it entails waiting for the required NOP cycles after a refresh before performing an activate and then servicing the read or write. If the DDRCTL has begun closing pages for a speculative refresh but has not yet issued the refresh when the new transaction arrives, the speculative refresh is canceled.

9.1.3.4 Per-Bank Refresh

If `RFSHMOD0.per_bank_refresh` is set to '1', the DDRCTL performs per-bank refreshes instead of all-bank refreshes. In this case, `RFSHSET1TMG0.t_refi_x1_x32` and `RFSHTMG1.t_rfc_min` must be set to the appropriate values for per-bank refresh (t_{REFIpb} and t_{RFCpb} respectively). In this mode, the DDRCTL keeps track of which bank is being refreshed at any time, and is able to schedule commands to other banks immediately before and after the per-bank refresh commands, resulting in potential efficiency gains. To improve the accuracy of per-bank refresh timing, set `RFSHTMG0.t_refi_x1_sel` to '1' and program `RFSHSET1TMG0.t_refi_x1_x32` accordingly.

The per-bank refresh command can be issued in any bank order. The DDRCTL can send the per-bank refresh to banks as a priority, where no transactions pending in the CAM or the bank is precharged. Also, the DDRCTL does not send the per-bank refresh command to multiple ranks at the same time, as the command contains a bank address, and the DDRCTL cannot specify multiple bank addresses with one command.

If the refresh burst (`RFSHMOD0.refresh_burst`) is set to its maximum value of 63, it means that the DDRCTL can postpone up to 64 per-bank refreshes, in accordance with the JEDEC specification. However, in this case the time between two per-bank refreshes to a specific bank can possibly exceed $9 \times t_{REFI}$. It is not clear from the JEDEC specification whether this is a violation or not; to avoid this, `RFSHMOD0.refresh_burst` must be set to the maximum value of 55.

If the refresh burst (`RFSHMOD0.refresh_burst`) is programmed with a large value in per-bank Refresh mode, and bursting per-bank refreshes start, it may take a longer time than bursting all-bank refreshes. The three possible cases where bursting per-bank refreshes may occur are:

1. Random traffic

The DDRCTL may not be able to issue a speculative refresh because the CAM always has transactions to some banks. This can be avoided with appropriate register settings. For example, setting smaller values to `RFSHCTL0.refresh_burst` or `RFSHMOD0.refresh_to_x1_x32`.

2. Toggling `RFSHCTL0.refresh_update_level`

3. Hardware Fast Frequency Change (HWFFC)

After the 2 or 3 operation, the DDRCTL starts bursting refreshes to compensate for refresh timing parameters which may have been updated.

If `DERATEEN.derate_enable` is set to 1'b1 and `RFSHMOD0.auto_refab_en` is set to a value greater than 0, the DDRCTL may switch automatically from per-bank refresh to all-bank refresh if the refresh period is too small due to temperature derating.

In this case, the DDRCTL adjusts the following refresh parameters internally when sending REFab instead of REFpb:

- `RFSHSET1TMG0.t_refi_x1_x32` is multiplied by 8 (tREFIab instead of tREFIpb).
- `RFSHSET1TMG1.t_rfc_min_ab` is used instead of `RFSHSET1TMG1.t_rfc_min` (tRFCab instead of tRFCpb).
- `RFSHSET1TMG3.refresh_to_ab_x32` is used instead of `RFSHSET1TMG1.refresh_to_x1_x32`.
- `RFSHMOD0.refresh_burst` is divided by 8.

Once temperature derating allows it, the DDRCTL switches back to sending REFpb and adjusts above internal timings to their original software programmed values for per-bank refresh.



Note

In LPDDR5 mode, bank address BA3 and bank group address BG1 are "don't care" for 16B and BG mode, and then per-bank refresh is performed to two banks simultaneously.

9.1.3.5 Constraints on refresh_timerX_start_value_x32

The `refresh_timerX_start_value_x32` register fields control the relative timing of refreshes to different ranks. The register fields must obey the following constraint:

$$\text{refresh_timerX_start_value_x32} + \text{RoundUp}(\text{RFSHTMG1.t_rfc_min}/32) < \text{RFSHSET1TMG0.t_refi_x1_x32}$$

If `RFSHTMG0.t_refi_x1_sel` is set, $\text{RoundDown}(\text{RFSHSET1TMG0.t_refi_x1_x32}/32)$ must be used in the previous constraint.

Also, note that the register field `refresh_timerX_start_value_x32` cannot be changed after initialization. So the value of the register `RFSHSET1TMG0.t_refi_x1_x32` used previously must be anticipated as the minimum value which is used, considering frequency change, fine granularity refresh, and so on.

For LPDDR4, if `DERATECTL0.derate_enable` = 1, use minimum $\text{RFSHSET1TMG0.t_refi_x1_x32}/4$ as value in calculation of `refresh_timerX_star_value_x32` to consider possible effect of derating in high temperature case.

9.1.4 Automatic Temperature Derating

The DDRCTL includes functionality to automatically read the MR4 register in SDRAMs, and to derate the refresh rate and certain timing parameters accordingly. This functionality can be enabled by setting the `DERATECTL0.derate_enable` input to '1'.

The read interval with which the DDRCTL performs MRR operations from MR4 is defined (in clock periods) by the `DERATEINT.mr4_read_interval` input. This value depends on the maximum expected temperature gradient. For more information, refer to the LPDDR4/LPDDR5 JEDEC Specification.

For multi-rank systems, the derated timing values (including refresh rate) are adjusted for all ranks in accordance with the temperature derating value of the worst device.

By default, derating logic uses MR4's TUF flag (`MR4[7]`) as a valid flag before evaluating a new MR4 value from `MR4[2:0]`. If `MR4[7]=1'b0`, then `MR4[2:0]` is ignored and existing refresh rate/timing parameters are maintained. The use of `MR4[7]` as a valid flag can be disabled through a software (`DERATECTL0.derate_mr4_tuf_dis`). For example, if the system is hot or cold prior to `DERATECTL0.derate_enable=1`, the first MR4 value read by derating logic may have `MR4[7]=0` as there was a software driven MR4 previously, even though `MR4[2:0] != 3'b011` (1 x `tREFI` case). It is recommended to set `DERATECTL0.derate_mr4_tuf_dis=0`.

Once `DERATECTL0.derate_enable=1` is set, it is required not to perform MRR to MR4 through software (`MRCTRL*/MRSTAT`).

If the value read from MR4 indicates that derating is required, the effective refresh rate is modified as shown in [Table 9-2](#) and [Table 9-3](#) on page 187. In LPDDR4 and the case of `MR4[2:0] = "110"` or in LPDDR5 and the case of `MR4[4:0] = "01101"` or `"01111"`, the effective values of the following timing registers are used:

- `DERATEVAL0.derated_t_rcd`
- `DERATEVAL0.derated_t_ras_min`
- `DERATEVAL0.derated_t_rp`
- `DERATEVAL0.derated_t_rrd`
- `DERATEVAL0.derated_t_rc`

In case of `MR4[2:0] = "000"` or `"111"` (for LPDDR4) and `MR4[4:0] = "00000"` or `"11111"` (for LPDDR5), or invalid value, the DDRCTL can generate maskable interrupt `derate_temp_limit_intr` and non-maskable interrupt `derate_temp_limit_intr_fault`. If `DERATECTL1.derate_temp_limit_intr_en` is set to '0', `derate_temp_limit_intr` is never asserted, although there is no effect on `derate_temp_limit_intr_fault`. The status register `DERATESTAT.derate_temp_limit_intr` stores the pre-masked `derate_temp_limit_intr`. Both interrupts and the status register are cleared by setting `DERATECTL1.derate_temp_limit_intr_clr` to '1'. Furthermore, for testing or debug purposes, you can force the `derate_temp_limit_intr` by setting `DERATECTL1.derate_temp_limit_intr_force` to '1'.

Table 9-2 Effect of Values of MR4 Read from LPDDR4 SDRAM

MR4[2:0]	Refresh Rate	Derate DRAM Timing Values
001	$\text{RFSHSET1TMG0.t_refi_x1_x32} * 4$	No derating performed
010	$\text{RFSHSET1TMG0.t_refi_x1_x32} * 2$	No derating performed
011	$\text{RFSHSET1TMG0.t_refi_x1_x32}$	No derating performed

MR4[2:0]	Refresh Rate	Derate DRAM Timing Values
100	RFSHSET1TMG0.t_refi_x1_x32 / 2	No derating performed
101	RFSHSET1TMG0.t_refi_x1_x32 / 4	No derating performed
110	RFSHSET1TMG0.t_refi_x1_x32 / 4	Derating performed - timing parameters incremented (see above)
111	RFSHSET1TMG0.t_refi_x1_x32 / 4	Derating performed - timing parameters incremented (see above)
Other values	RFSHSET1TMG0.t_refi_x1_x32	No derating performed

Table 9-3 Effect of Values of MR4 Read from LPDDR5 SDRAM

MR4[4:0]	Refresh Rate	Derate DRAM Timing Values
00001	RFSHSET1TMG0.t_refi_x1_x32 * 8	No derating performed
00010	RFSHSET1TMG0.t_refi_x1_x32 * 6	No derating performed
00011	RFSHSET1TMG0.t_refi_x1_x32 * 4	No derating performed
00100	RFSHSET1TMG0.t_refi_x1_x32 * 3.3	No derating performed
00101	RFSHSET1TMG0.t_refi_x1_x32 * 2.5	No derating performed
00110	RFSHSET1TMG0.t_refi_x1_x32 * 2	No derating performed
00111	RFSHSET1TMG0.t_refi_x1_x32 1.7	No derating performed
01000	RFSHSET1TMG0.t_refi_x1_x32 * 1.3	No derating performed
01001	RFSHSET1TMG0.t_refi_x1_x32	No derating performed
01010	RFSHSET1TMG0.t_refi_x1_x32 * 0.7	No derating performed
01011	RFSHSET1TMG0.t_refi_x1_x32 * 0.5	No derating performed
01100	RFSHSET1TMG0.t_refi_x1_x32 * 0.25	No derating performed
01101	RFSHSET1TMG0.t_refi_x1_x32 * 0.25	Derating performed - timing parameters incremented (see above)
01110	RFSHSET1TMG0.t_refi_x1_x32 * 0.125	No derating performed
01111	RFSHSET1TMG0.t_refi_x1_x32 * 0.125	Derating performed - timing parameters incremented (see above)

If derating is enabled, and is later disabled by the `DERATECTL0.derate_enable` register, the refresh rate and other timing parameters revert to their nominal values - the derated values are not retained.

During frequency change process, to retain the derated values, while stopping to send MRR commands for MR4, `DERATEEN.derate_mr4_pause_fc` must be set to '1' instead of setting `DERATEEN.derate_enable` to '0'.

Automatic temperature derating shall be disabled/paused before enabling OCPAR poisoning.

For LPDDR4/5 6Gb/8Gb/12Gb/16Gb and LPDDR5 24Gb/32Gb devices (per channel), with both per-bank refresh and derating enabled, refreshes can consume all the bandwidth on the SDRAM bus. This is because $t_{REFIpb} < 4 * t_{RFCpb}$. Therefore, if the temperature is such that the refresh period is divided by four, if per-bank refresh (`RFSHMOD0.per_bank_refresh`) and derating (`DERATEEN.derate_enable`) are both set, it is required to set `RFSHMOD0.auto_refab_en` to a value greater than 0. This allows the DDRCTL to switch automatically from per-bank refresh to all-bank refresh when the refresh period is too small.

9.1.5 Signals Related to Refresh Controls

The following signals are related to the Refresh Controls:

- Per-bank refresh Bank number Signals

For more information about these signals, see “[Signal Descriptions](#)” on page 373.

9.1.6 Registers Related to Refresh Controls

The following are the registers related to the Refresh Controls:

- `RFSHMOD0`
- `RFSHCTL0`
- `RFSHSET1TMG0`
- `RFSHSET2TMG0`
- `RFSHSET3TMG0`
- `RFSHSET4TMG0`
- `RFSHSET5TMG0`

For more information about these registers, see the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

9.1.7 Dynamic SDRAM Rank Constraints

One rank constraints block enforces all of the following constraints for each rank supported by the system. Using the rank constraints blocks, the scheduler dynamically obeys all of the constraints on a per-rank basis when scheduling transactions. The values for the timing parameters can be obtained from the relevant JEDEC Specification.

9.2 Refresh Management (RFM)

Periods of high SDRAM activity may require additional Refresh commands to protect the integrity of the SDRAM data. The SDRAM devices that require additional activity-based refreshes include support for an Activation-based refresh management (RFM) command.

The RFM feature is configured by setting configuration parameter `DDRCTL_RFM_HW_CTRL`, and enabled by setting `RFMMOD0.rfm_en` to '1' when LPDDR5 requires RFM (`MR27:OP[1] == 1`).

9.2.1 Feature Restrictions

- Adaptive Refresh Management (ARFM) is not supported.
- Single-bank mode is not supported.
- All-bank Refresh Management (RFMab) command is not supported.
- LPDDR4 (JESD209-4D) RFM is not supported.

9.2.2 Rolling Accumulated ACT (RAA) Counter

Rolling Accumulated ACT (RAA) counter counts the number of ACT commands, and RFM command is scheduled based on RAA count. The controller has RAA counters per bank-pair. The RAA counter is implemented in the following way:

- Increments by 1 when issuing ACT command to bank corresponding to the RAA counter.
- Decrements by RAAIMT when issuing REFab/REFpb to bank(s) corresponding to the RAA counter.
- Decrements by $RAAIMT * RAADEC$ when issuing RFM command to bank(s) corresponding to the RAA counter.

Valid range of the RAA count is from 0 to RAAMMT (= $RAAIMT * RAAMULT$). When RAA count is 0, and any decrement event happens, the count keeps 0. When RAA count reaches RAAMMT, the DDRCTL stops scheduling ACT to that bank, and increment event never happens.

Table 9-4 RAA Counter Relevant Registers are Programmed Based on Mode Registers in the SDRAM

	DDRCTL Register	LPDDR5 Mode Register
RAAIMT	RFMMOD0.raaimt	MR27:OP[5:1]
RAADEC	RFMMOD0.raadec	MR57:OP[1:0]
RAAMULT	RFMMOD0.raamult	MR27:OP[7:6]

9.2.3 RFM Command Control

DDRCTL starts issuing per-bank RFM (RFMpb) command when RAA count reaches RAAMMT. This is to minimize an impact on DDR utilization if a refresh interval (t_{REFIe}) satisfies a Refresh Management Threshold (RFMTH) requirement.

JEDEC specification requirement:

- RAA count \leq RAAMMT
- $t_{REFIe} > RFMTH$

If RAA count == RAAMMT, ACT is not allowed, that is, transaction is blocked. It is possible to issue RFM beforehand.

However, the DDRCTL does not schedule RFM command when current Refresh Multiplier (RM) is greater than or equal to RFMMOD0.rfmth_rm_thr.

DDRCTL specification requirement:

- RAA count == RAAMMT
- $RM < RFMMOD0.rfmth_rm_thr$

The DDRCTL currently supports `RFMMOD0.rfmth_rm_thr == 5'b11111` only, indicating RFM command can be issued in any RM.

The controller does not issue RFM when RAA count < RAAMMT.

Both JEDEC and DDRCTL specifications must be satisfied.

9.3 ZQ Calibration

This topic contains the following sections:

- [“Overview of ZQ Calibration”](#) on page 191
- [“LPDDR4 Devices”](#) on page 191
- [“LPDDR5 Devices”](#) on page 193
- [“Automatic and Software Initiated ZQ Calibration Commands”](#) on page 195
- [“LPDDR4/LPDDR5 ZQ Reset Command”](#) on page 196
- [“Registers Related to ZQ Calibration”](#) on page 197

9.3.1 Overview of ZQ Calibration

The DDRCTL controller uses ZQ calibration command to calibrate SDRAM RON (Resistor ON) and ODT (On-die termination) values over PVT (Process, Voltage, Temperature). LPDDR4 SDRAMs need more time to calibrate RON and ODT at initialization and relatively less time to perform periodic calibrations. For more information, refer to the LPDDR4 (JEDEC JESD209-4B) specification.

9.3.2 LPDDR4 Devices

The DDRCTL supports all the ZQ calibration commands that are supported by the JEDEC Specification.

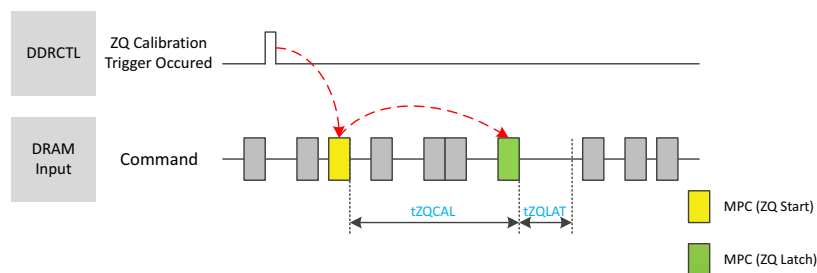
ZQ calibration sequence uses the following MPC commands:

- ZQCal Start
- ZQCal Latch

The ZQCal Start command is used to initiate SDRAM’s calibration procedure.

The ZQCal Latch command is used to capture the result and load it into SDRAM’s drivers. ZQCal Latch command can be issued after t_{ZQCAL} (determined by $ZQSET1TMG0.t_zq_long_nop$) has expired and all DQ Bus operations have completed. If a ZQ calibration is triggered, those commands are sent out consecutively to the DRAM by DDRCTL.

Figure 9-1 ZQ Calibration



ZQCal Start and ZQCal Latch command can be performed automatically at a regular interval or through direct software request. For more information, see [“Automatic and Software Initiated ZQ Calibration Commands”](#) on page 195. Furthermore, both commands are issued automatically after SR-Powerdown exit if $ZQCTL2.dis_srx_zqcl$ is set to '0'. To disable issuing of both commands after SR-Powerdown exit, set $ZQCTL2.dis_srx_zqcl$ to '1'.

Do not change the following mode register fields following a ZQCal Start command and before tZQCAL has expired:

- PU-Cal (Pull-up Calibration VOH Point) - MR3:OP[0]
- PDDS (Pull Down Drive Strength and Rx Termination) - MR3:OP[5:3]
- DQ-ODT (DQ ODT Value) - MR11:OP[2:0]

To change these mode register fields, follow this procedure:

1. Disable the following automatic controls (if using):
 - a. Set PWRCTL.selfref_en to '0'.
 - b. Set ZQCTL0.dis_auto_zq to '1'.
2. Ensure that DDRC is not in self-refresh or SR-Powerdown:
 - a. Wait for STAT.operating_mode not to be 3'b011 (self-refresh /SR-Powerdown).
3. Enter self-refresh 1 state:
 - a. Set PWRCTL.stay_in_selfref to '1'.
 - b. Set PWRCTL.selfref_sw to '1'.
 - c. Wait until STAT.selfref_state is higher than 3'b000 (any self-refresh state).



Note

- When polling the field STAT.selfref_state, do an APB RD for the entire STAT register. That is, selfref_state and selfref_type must be read at the same time (in the same APB read).
- The reading of the field STAT.selfref_type is done along with the reading of the field STAT.selfref_state.

- d. If STAT.selfref_state is 3'b001 and STAT.selfref_type is not 2'b01 ("self-refresh 1" state caused solely by software, not by DFI PHY Master hardware state machine) then jump to step 4, else: program PWRCTL.selfref_sw = 0, program PWRCTL.stay_in_selfref = 0 and return to step 2.

4. Send MRW commands using MRCTRL0 and MRCTRL1.
5. Enter SR-Powerdown mode:
 - a. Set PWRCTL.stay_in_selfref to '0'.
 - b. Wait for STAT.selfref_state to become 3'b010 (SR-Powerdown).
 - c. If DQ-ODT is switched between Disable and Enable, updating DRAMSET1TMG2.rd2wr and RANKTMG0.diff_rank_wr_gap is required accordingly.

If ZQCTL2.dis_srx_zqcl is programmed to '1':

1. Enter self-refresh 2 state:
 - a. Set PWRCTL.stay_in_selfref to '1'.
 - b. Set PWRCTL.selfref_sw to '0'.
 - c. Wait for STAT.selfref_state to become 3'b011 (self-refresh 2).
2. Send the ZQCal command through direct request from hardware (external IOs) or software.

3. Exit self-refresh mode:

- a. Set `PWRCTL.stay_in_selfref` to '0'.
- b. Wait for `STAT.selfref_state` to become 3'b000 (not self-refresh).

4. Re-enable automatic controls disabled in first step.

If `ZQCTL2.dis_srx_zqcl` is not programmed to '1':

1. Exit self-refresh mode:

- a. Set `PWRCTL.selfref_sw` to '0'.
- b. Wait for `STAT.selfref_state` to become 3'b000 (not self-refresh).

2. Re-enable automatic controls disabled in the first step.

ZQCal Reset (ZQ Calibration Reset) command is used to reset the output impedance calibration to a default accuracy of +/- 30% across process, voltage, and temperature. This command is used to ensure output impedance accuracy to +/- 30% when ZQCal Start and ZQCal Latch commands are not used and a time period of `tZQRESET` is allowed, determined by `ZQTMG1.t_zq_reset_nop`. The command is issued by using the registers `ZQTMG1.zq_reset` and `ZQSTAT.zq_reset_busy`. For more information, see [“LPDDR4/LPDDR5 ZQ Reset Command”](#) on page 196.

The DDRCTL performs no other activities for the duration of `tZQLAT` (determined by `ZQTMG0.t_zq_short_nop`) and `tZQRESET` (determined by `ZQTMG1.t_zq_reset_nop`). The quiet time on the SDRAM channel helps in accurate calibration of SDRAM output impedance.

Although not required by the LPDDR4 JEDEC specification, the DDRCTL closes (precharges) all banks before sending ZQCal commands. This implementation is aligned with the ZQ calibration behavior for other protocols.



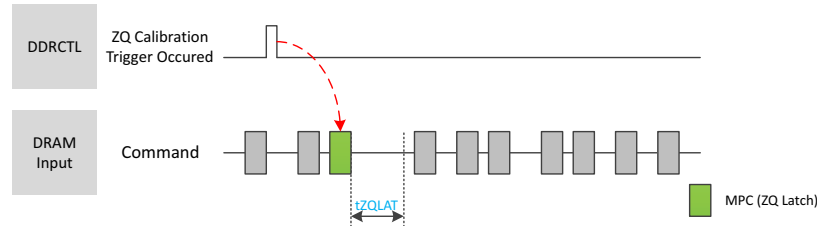
Note For LPDDR4, the access to the MR10 register from software using the mode register interface is prohibited (see [“Mode Register Reads and Writes”](#) on page 212).

9.3.3 LPDDR5 Devices

LPDDR5 JEDEC specification supports two ZQ calibration mode - Background Calibration or Command-Base Calibration. The DDRCTL supports Background Calibration mode only. Command-Base Calibration is not supported. DDRCTL ZQ calibration sequence uses ZQCal Latch command only.

In Background Calibration mode ZQ calibration is performed in the background and kept up-to-date by the DRAM. Re-calibration will be performed by the LPDDR5 SDRAM within the time interval, `tZQINT`, specified in `MR28 OP[3:2]`. The `MR28 OP[3:2]` can be accessed from software using the mode register interface.

At the completion of ZQ calibration, `ZQUF (MR4 OP[5])` bit will be set if the new calibration codes do not match the currently latched codes. It is allowed to issue ZQCal Latch command without monitoring ZQUF bit.

Figure 9-2 LPDDR5 ZQ Calibration

ZQCal Latch command can be performed automatically at a regular interval or through direct software request without monitoring ZQUF bit. For more information, see “[Automatic and Software Initiated ZQ Calibration Commands](#)” on page 195. Furthermore, ZQCal Latch is issued automatically after Self-Refresh power-down exit if ZQCTL2.dis_srx_zqcl is set to ‘0’. To disable issuing of both commands after Self-Refresh exit, set ZQCTL2.dis_srx_zqcl to ‘1’.

The Background calibration must be halted by setting ZQ Stop (MR28 OP[1]) when DVFSQ is active or VDDQ is going to be power off. Automatically ZQCal latch must not be issued while the Background calibration is halted. Therefore, ZQCTL0.dis_auto_zq and ZQCTL2.dis_srx_zqcl need to be set to ‘1’ before ZQ Stop is set to ‘1’. ZQ Stop can be accessed from software using the mode register interface.

To change ZQ Stop (MR28 OP[1]) to ‘1’, follow this procedure:

1. Enter self-refresh state:
 - a. Set PWRCTL.selfref_en to ‘0’.
 - b. Check STAT.operating_mode != 3'b011 (or STAT.selfref_state == 3'b000). If STAT.operating_mode == 3'b011 and STAT.selfref_type == 2'b10, set HWLPCTL.hw_lp_en to ‘0’ in order to exit self-refresh.
 - c. Set PWRCTL.stay_in_selfref to ‘1’.
 - d. Set PWRCTL.selfref_sw to ‘1’.
 - e. Wait until STAT.selfref_state is 3'b001 (self-refresh 1 state).
2. Disable the following automatic controls (if using):
 - a. Set SWCTL.sw_done to ‘0’.
 - b. Set ZQCTL0.dis_auto_zq to ‘1’.
 - c. Set ZQCTL2.dis_srx_zqcl to ‘1’.
 - d. Set SWCTL.sw_done to ‘1’.
 - e. Wait until SWSTAT.sw_done_ack becomes equal to 1'b1.
3. Change ZQ Stop to ‘1’:
 - a. Send MRW commands using MRCTRL0 and MRCTRL1.
4. Exit self-refresh mode:
 - a. Set PWRCTL.selfref_sw to ‘0’.
 - b. Set PWRCTL.stay_in_selfref to ‘0’.
 - c. Wait for STAT.selfref_state to become 3'b000 (not self-refresh).

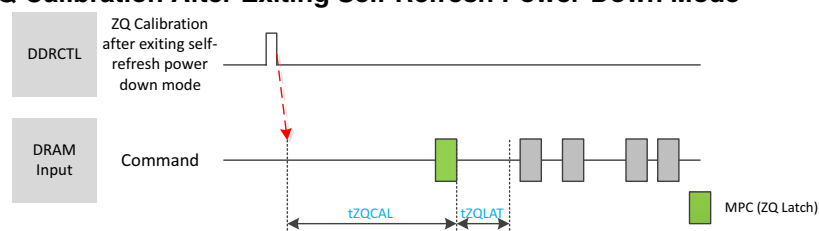
ZQ Stop (MR28 OP[1]) may be reset to ‘0’ when DVFSQ is no longer active or power-down mode is exited. When ZQ Stop is reset to ‘0’, ZQ calibration starts immediately. If ZQCTL0.dis_auto_zq changes from ‘1’

to '0', Read/Write commands are not issued while ZQ calibration is working (the period of $t_{ZQCAL} + t_{ZQLAT}$).

To change ZQ Stop (MR28 OP[1]) to '0', follow this procedure:

1. Enter self-refresh state:
 - a. Set PWRCTL.selfref_en to '0'.
 - b. Check STAT.operating_mode != 3'b011 (or STAT.selfref_state == 3'b000). If STAT.operating_mode == 3'b011 and STAT.selfref_type == 2'b10, set HWLPCTL.hw_lp_en to '0' in order to exit from self-refresh.
 - c. Set PWRCTL.stay_in_selfref to '1'.
 - d. Set PWRCTL.selfref_sw to '1'.
 - e. Wait until STAT.selfref_state is higher than 3'b000 (any self-refresh 1 state).
2. Reset ZQ Stop to '0':
 - a. Send MRW commands using MRCTRL0 and MRCTRL1.
3. Enable the following automatic controls (if using):
 - a. Set SWCTL.sw_done to '0'.
 - b. Set ZQCTL0.dis_auto_zq to '0'.
 - c. Set ZQCTL2.dis_srx_zqcl to '0'.
 - d. Set SWCTL.sw_done to '1'.
 - e. Wait until SWSTAT.sw_done_ack becomes equal to 1'b1.
4. Exit self-refresh mode:
 - a. Set PWRCTL.selfref_sw to '0'.
 - b. Set PWRCTL.stay_in_selfref to '0'.
 - c. Wait for STAT.selfref_state to become 3'b000 (not self-refresh).

Figure 9-3 LPDDR5 ZQ Calibration After Exiting Self-Refresh Power-Down Mode



Note

If an LPDDR5 SDRAM package has multiple dies and two or more ZQ pins, for proper operation an external ZQ resistor cannot be shared by those ZQ pins. That is, each ZQ pin needs dedicated external ZQ resistor.

9.3.4 Automatic and Software Initiated ZQ Calibration Commands

The DDRCTL issues ZQ Calibration commands in the following ways:

- **Automatic ZQ Calibration Commands by the DDRCTL:** in this case, DDRCTL sends ZQ Calibration commands to SDRAM periodically. The interval is determined by `ZQSETTMG1.t_zq_short_interval_x1024`. This method is used if `ZQCTL0.dis_auto_zq` is set to '0'.
- **ZQ Calibration Commands using direct software request:** in this case, the SoC sends ZQ Calibration commands through software by setting `OPCTRLCMD.zq_calib_short` to '1'. When the ZQ Calibration command request is stored in the DDRCTL, the register bit is automatically cleared. It is recommended not to set `OPCTRLCMD.zq_calib_short` signal in init, self-refresh operating modes or deep sleep mode (DSM). The SoC can initiate a ZQ Calibration commands operation only if `OPCTRLSTAT.zq_calib_short_busy` is low. The `OPCTRLSTAT.zq_calib_short_busy` signal goes high in the clock after the DDRCTL accepts a ZQ Calibration commands request. It goes low when the ZQ Calibration commands operation is initiated in the DDRCTL. For proper SDRAM operation, you or SoC must schedule this command frequently. This method is used if `ZQCTL0.dis_auto_zq` is set to '1'.

DDRCTL sends MPC-based ZQ calibration commands (ZQCal Start or ZQCal Latch) and it is recommended not to set `OPCTRLCMD.zq_calib_short` signal in init or SR-Powerdown or Deep Sleep Mode (LPDDR5) operating mode (this can be checked by observing `STAT.operating_mode` and `STAT.selfref_state`).

- **External ZQCal:** In this case ZQCal commands are sent through top-level IOs. This functionality is enabled by setting the configuration parameter `UMCTL2_REF_ZQ_IO`. This functionality can be switched on by setting the static register `OPCTRLCMD.hw_ref_zq_en` to '1' (see "Register Descriptions" chapter of the DWC DDRCTL LPDDR5/4 Programming Guide) and has effect only if `ZQCTL0.dis_auto_zq` is set to '1'. If these conditions are not satisfied the IO pins related to external zq commands are ignored. External ZQCal can be issued by setting `ext_zq_calib_short` signal to '1' for one clock cycle. It is recommended not to set this signal in init, self-refresh, deep power-down operating modes.

`ext_zq_calib_short_busy` signal goes high in the clock after the DDRCTL accepts a ZQCal request through IOs and it goes low when the ZQCal operation is initiated in the DDRCTL.



Note

External ZQCal feature is currently not supported.

For the most current information about unsupported features and limitations, refer to the Release Notes.

For SR-Powerdown, command is scheduled after SR-Powerdown is exited.

9.3.5 LPDDR4/LPDDR5 ZQ Reset Command

The ZQ Reset command is issued by setting `ZQCTL1.zq_reset` to '1'. In LPDDR4 mode, it is recommended not to set this register field to '1' in init or SR-Powerdown or Deep Sleep Mode (LPDDR5) operating modes (this can be checked by observing `STAT.operating_mode` and `STAT.selfref_state`). The SoC can initiate a ZQ Reset operation only if `ZQSTAT.zq_reset_busy` is low. This register field goes high in the clock after the DDRCTL accepts a ZQ Reset request. It goes low when the ZQ reset command is issued to the SDRAM and the associated NOP period is completed.

For SR-Powerdown, command is scheduled after SR-Powerdown is exited.

In LPDDR5 mode, DDRCTL changes `MR28` to default value during ZQ Reset operation. If ZQ interval (`MR28 OP[3:2]`) is needed to use non-default value, ZQ interval (`MR28 OP[3:2]`) needs to be updated from software using the mode register interface after `ZQSTAT.zq_reset_busy` is low.



When DDRCTL is going to put SDRAM into SR-Powerdown and `ZQCTL1.zq_reset` is about to set to '1', it is possible that the request can be accepted. However, DDRCTL cannot issue ZQ Reset command in case where SDRAM has already been SR-Powerdown.

In this case, ZQCal Start/Latch commands are issued instead of ZQ Reset command just after SR-Powerdown is exited.

9.3.6 Registers Related to ZQ Calibration

The following are the registers related to the ZQ Calibration:

- ZQCTL0
- ZQCTL2
- ZQTMG1
- ZQSTAT

For more information about these registers, see the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

9.4 Controller Assisted Drift Tracking (LPDDR5)

This topic contains the following sections:

- [“Overview of Controller Assisted Drift Tracking”](#) on page 198
- [“PHY Snoop Control”](#) on page 198
- [“Registers Related to Controller Assisted Drift Tracking”](#) on page 199

9.4.1 Overview of Controller Assisted Drift Tracking

This feature is applicable to LPDDR5 SDRAM. As voltage and temperature change on the SDRAM die, the DQS/WCK clock tree will shift and may require re-training. The LPDDR5 includes an internal oscillator to measure the amount of delay over a given time interval. These oscillators started by MPC command with proper opcode. The results are updated in the MR register which can be read using MRR command. This feature enables Synopsys DWC LPDDR5/4/4X PHY to smartly adjust LCDL ("local calibrated delay line") to compensate the drift when DFI update event occurs. It is accomplished by allowing Synopsys DWC LPDDR5/4/4X PHY to periodically snoop the result of these oscillators of SDRAM.

This feature is also referred as "DQS Oscillator" in the controller and it can be used interchangeably with Controller Assisted Drift Tracking.



Note

DDRCTL does not support MPC command to stop the Oscillator.

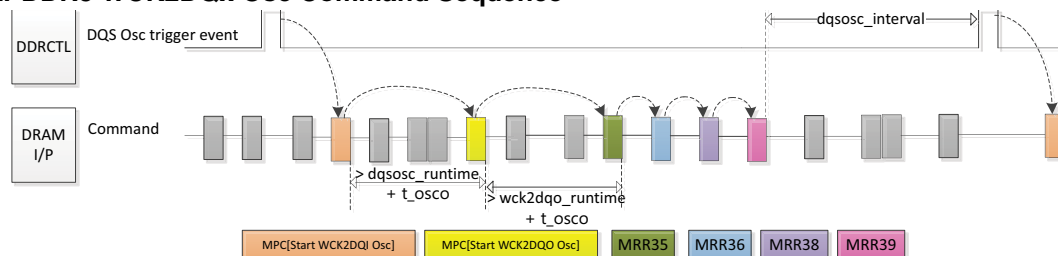
9.4.2 PHY Snoop Control

LPDDR5 SDRAM includes internal oscillators to measure the tWCK2DQI and tWCK2DQO timing parameters.

These oscillators are started by issuing MPC command. After runtime expires, the results are updated in MR register by SDRAM. The DDRCTL takes the control of scheduling MPC command to start these oscillators and MRR commands to read the result of these oscillators periodically as well as during SRE/PDE. The DDRCTL also asserted `dwc_ddrphy*_snoop_en_P*` flag, along with `dfi_rddata_en/dfi_rddata_cs` for full burst-length UIs, to enable the Synopsys DWC LPDDR5/4/4X PHY to capture the MRR data. Subsequently, Synopsys DWC LPDDR5/4/4X PHY smartly adjusts LCDL ("local calibrated delay line") to compensate the drift when DFI update event occurs. The DDRCTL also blocks any activate command to the same rank while reading out the result (MRR command) of these oscillators.

For LPDDR5, the DDRCTL first sends out two MPC commands to start the WCK2DQI and WCK2DQO interval oscillators. Subsequently, followed by four MRR commands to get the result of these oscillators.

Figure 9-4 LPDDR5 WCK2DQx Osc Command Sequence



For a multi rank system, the MPC commands to start the oscillators are scheduled to all the active ranks simultaneously. However, the MRR commands to read out the result are scheduled one after the other in the order of higher rank to lower rank.

To enable the DQS Oscillator, follow this procedure below:

1. Program the following register(s) to "interval timer run time setting" defined in the JEDEC specification:
 - `DQSOSCRUNTIME.dqsosc_runtime` for MR37
 - `DQSOSCRUNTIME.wck2dgo_runtime` for MR40
2. Program the `DQSOSCRUNTIME` register. This is non-zero binary value that corresponds to interval timer run time setting of LPDDR5 SDRAM.
3. Program the `DQSOSCCTL0` register:
 - a. Program DQS Oscillator interval time. This defines the frequency of DQS Oscillator trigger event. It must be appropriately programmed based on the system requirement.
 - b. Enable the DQS oscillator by setting `DQSOSCCTL0.dqsosc_enable = 1`.

To disable the DQS Oscillator, use the following procedure:

1. Disable DQS oscillator by setting `DQSOSCCTL0.dqsosc_enable=0`.
2. If the DQS oscillator has already started (MPC[Start DQS Osc] is already issued), it will continue until the associated MRR commands are issued. Completely disabled state of the DQS Oscillator is indicated by `DQSOSCSTAT0.dqsosc_per_rank_stat=0` and `DQSOSCSTAT0.dqsosc_state=0`.

After disabling the DQS oscillator and before enabling any other periodic phase training method the software must poll these registers (`DQSOSCSTAT0.dqsosc_per_rank_stat=0` and `DQSOSCSTAT0.dqsosc_state=0`). DQS oscillator must be disabled before enabling OCPAR poisoning.



Note

Controller assisted drift tracking and PPT (Periodic Phase Training) using DFI MC to PHY Message Interface are mutually exclusive and must not be enabled at the same time. Do not change `DQSOSCCTL0.dqsosc_enable` (that is, keep the default value of '0') if DFI MC to PHY Message Interface and/or DFI PHY Master interface is used.

9.4.3 Registers Related to Controller Assisted Drift Tracking

The following are the registers related to Controller Assisted Drift Tracking

- `DQSOSCRUNTIME`
- `DQSOSCCTL0`
- `DQSOSCSTAT0`

For more information about these registers, see the "Register Descriptions" chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

9.5 Enhanced Incremental Periodic Phase Training (PPT2)

This section covers the following topics:

- “Overview of PPT2” on page 200
- “PPT2 Limitations” on page 200
- “Retraining Interval” on page 202
- “Normal PPT2” on page 203
- “Burst PPT2” on page 207
- “Registers Related to PPT2” on page 209

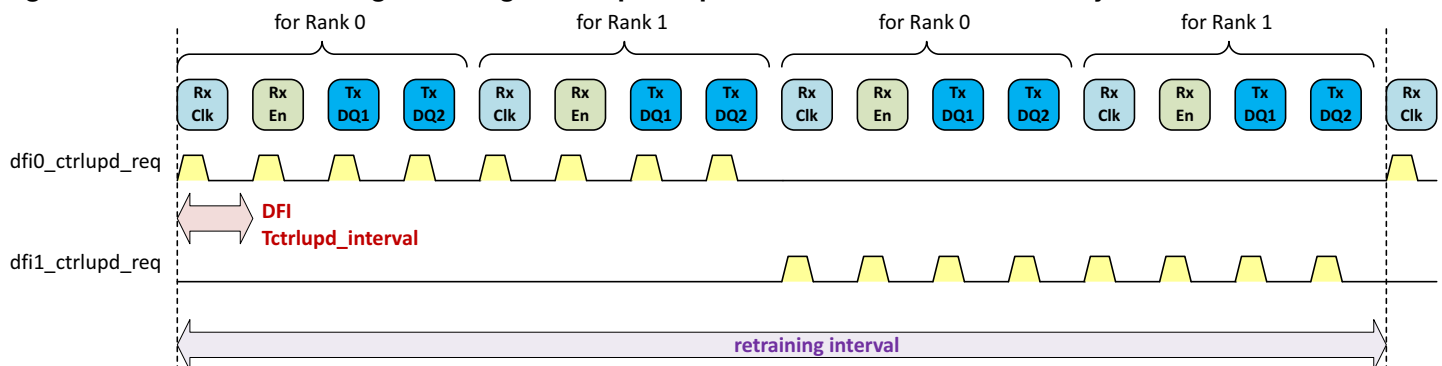
9.5.1 Overview of PPT2

PPT2 is a MC initiated, low latency and accurate DQS/DQ retraining scheme done along with Synopsys DWC LPDDR5/4/4X PHY. DDRCTL issues a number of `dfin_ctrlupd_req/ack` handshake with `dfin_ctrlupd_type = 1`¹ to trigger the PHY PPT2 within a given interval. For each handshake, PHY cyclically retrains its four interfaces of RxClk, RxEn, Tx DQ1, and Tx DQ2 (if Link ECC is enabled). Since they need retraining per a rank per a DFI interface, a hardware configuration which is Link ECC enabled & 2-rank & dual DFI needs $4 \times 2 \times 2$ `ctrlupd_req` to complete retraining all devices. Figure 9-5 shows an example of PPT2 scheduling.

Because each `ctrlupd_req` costs less than 500ns (or 1000ns depending on the configuration) and some part of it is during tRFC, PPT2 has lower latency comparing to PHY master of 10us latency. Also, PPT2 is considered to be accurate because PHY does not rely on any built-in oscillators in DRAM to retrain.

Unless otherwise specified, `ctrlupd_req` in this section refers to the `dfin_ctrlupd_req/ack` handshake with `dfi_ctrlupd_type=1`.

Figure 9-5 PPT2 Scheduling - Sending 16 ctrlupd_req to Retrain 2-rank & Dual DFI System



9.5.2 PPT2 Limitations

The following are current PHY side limitations. For the latest status of these limitations, refer to the PHY databook.

- Supported for data rates ≥ 1600 Mbps.
- Both Write Link ECC and Read Link ECC are enabled or both Write Link ECC and Read Link ECC are disabled.

1. `dfin_ctrlupd_type` is a DFI interface signal, which is Synopsys only and a non-standard one for now.

- Retraining latency is less than:
 - 500ns if data rates \geq 3200Mbps and WL=setA
 - 1000ns if data rates \geq 1600Mbps and WL=setA
- PHY configuration `userInputBasic.RetrainMode[Pstate] = 4` must be set.
- PPT2 is only used at the following DRAM states: Idle, Self-refresh (without Power-down), and Refresh.

The following list shows DDRCTL side limitations.

- When PPT2 is used, `phymstr`, `phyupd`, `ctrlmsg`, and `dqsosc` features are not necessary and must be disabled.
 - PHY pub register `PPTTrainSetup_pX::PhyMstrTrainInterval` must be '0'.
 - Controller register `DFIPHYMSTR.dfi_phymstr_en` can also be 0.
 - PHY pub register `DFIPHYUPD::DFIPHYUPDCNT` must be '0'.
 - Controller register `DFIUPD0.dfi_phyupd_en` can also be '0'.
 - `DFI0MSGCTL0.dfi0_ctrlmsg_req` must be '0'.
 - `DQSOSCCTL0.dqsosc_enable` must be '0'.
- HWFFC must be disabled.
- Existing DFI MC-Initiated Update Requests via `ctrlupd_type = 0` does not conflict with PPT2 according to [Table 9-5](#).

Table 9-5 `dis_auto_ctrlupd` and `ppt2_en` Combination

<code>dis_auto_ctrlupd</code>	<code>ppt2_en</code>	DDRCTL Functionality
0	0	Only <code>ctrlupd</code> type 0 is sent automatically.
0	1	Both <code>ctrlupd</code> type 0 and type 1 are sent automatically.
1	0	Neither <code>ctrlupd</code> type 0 nor type 1 are sent automatically.
1	1	Only <code>ctrlupd</code> type 1 is sent automatically. It is recommended not to set this combination.

- Connect new DDRCTL DFI ports to the PHY:
 - `dfi0_ctrlupd_type`
 - `dfi1_ctrlupd_type` if needed
 - They are not DFI 5.0 standard ports.
- Only Single DDRC Dual DFI configuration is used. Contact Synopsys for Single DFI configuration.
- Refresh multiplier 0.125x (LPDDR5 only) is not supported.
- Automatic All-bank / Per-bank refresh switching is recommended to be enabled.
 - `REFSHMOD0.auto_refab_en = 2` (switched to REFab if refresh rate \leq 0.25x) is recommended.
- Dynamic Frequency Switch (Software Fast Frequency Change) is supported with PPT2, but neither DVFSQ nor DVFSQ is supported.

- `PWRCTL.lpddr4_sr_allowed` must be '1'.
- DDRCTL will close all opened banks before issuing PPT2 to prepare for DRAM clock stop which PHY may do while retraining.
- When postponed refresh count exceeds `RFSHMOD0.refresh_burst`, 'critical refresh' happens. It blocks both Normal PPT2 and Burst PPT2 issues, not to postpone too many refresh commands while `ctrlupd_req` is sent.

9.5.3 Retraining Interval

$t_{\text{retraining_interval}}$ is an interval to complete a single, periodical all device retraining. It must be calculated to make the retraining pace catch up with the VT drifts. Otherwise, retraining cannot sufficiently compensate the drifts as time passes, which leads DQS/DQ bus error as a result. According to the PHY databook, the following is the $t_{\text{retraining_interval}}$ definition and an example configuration while omitting the voltage drift.

$$t_{\text{retraining_interval}}[\text{sec}] = \text{PHY IO delay tap size}[\text{ps}] / (\text{Thermal drift rate}[\text{ps/degC}] * \text{Thermal drift}[\text{degC/sec}] + \text{Voltage drift rate}[\text{ps/mV}] * \text{Voltage drift}[\text{mV/sec}])$$

If the device is LPDDR4:

- PHY IO delay tap size = 3 [ps]
- Thermal drift rate = $t_{\text{DQSCK_temp}} = 4.0$ [ps/degC]
- Thermal drift = 4 [degC/sec]

Then $t_{\text{retraining_interval}} = 3 / (4.0 * 4) = 187.5$ [ms].

In this case, all LPDDR4 devices connected to DDRCTL must get retrained for each 187.5 ms period.

As illustrated in [Figure 9-5](#) on page 200, PHY requires several `ctrlupd_req` sent from DDRCTL for each retraining interval. There is also $t_{\text{ctrlupd_interval}}$, which is an interval between two consecutive `ctrlupd_req`.

The number of `ctrlupd_req` required within $t_{\text{retraining_interval}}$ is four, if Link ECC is enabled, otherwise, it is three per a rank per a DFI interface. For Link ECC enabled & 2-rank & dual DFI system, $t_{\text{ctrlupd_interval}} = t_{\text{retraining_interval}} / (4 * 2 * 2)^1$.

[Table 9-6](#) shows how many `ctrlupd_req` is required for each retraining interval.

Table 9-6 The Number of `ctrlupd_req` Required for Each $t_{\text{retraining_interval}}$

DFI Interface	MSTR0. active_ranks	LNKECCCTL0. wr_link_ecc_enable	Total number of dfi_ctrlupd_req for each $t_{\text{retraining_interval}}$	PPT2CTRL0. ppt2_ctrlupd_num _dfi0	PPT2CTRL0. ppt2_ctrlupd_num _dfi1
Single DFI	1	0	3	3	0
Single DFI	1	1	4	4	0
Single DFI	3	0	6	6	0
Single DFI	3	1	8	8	0
Dual DFI	1	0	6	3	3
Dual DFI	1	1	8	4	4

1. This configuration is illustrated in [Figure 9-5](#) on page 200.

DFI Interface	MSTR0. active_ranks	LNKECCCTL0. wr_link_ecc_enable	Total number of dfi_ctrlupd_req for each t _{retraining_interval}	PPT2CTRL0. ppt2_ctrlupd_num _dfi0	PPT2CTRL0. ppt2_ctrlupd_num _dfi1
Dual DFI	3	0	12	6	6
Dual DFI	3	1	16	8	8

Set the register DFIUPDTMG2.dfi_t_ctrlupd_interval_type1 and DFIUPDTMG2.dfi_t_ctrlupd_interval_type1_unit to t_{ctrlupd_interval}.

Set the register PPT2CTRL0.ppt2_ctrlupd_num_dfi0 and PPT2CTRL0.ppt2_ctrlupd_num_dfi1 to the number of required ctrlupd_req for dfi0 and dfi1.

9.5.4 Normal PPT2

Normal PPT2 sends ctrlupd_req periodically and cyclically to dfi0 and dfi1. It is intended for the retraining to persistently compensate a little amount of short-term drift. When large amount of drift is expected, for example, after initialization or SRX from long-term SRPD/DSM, Burst PPT2 is required. For more details, see “Burst PPT2” on page 207.

To enable Normal PPT2, follow this procedure:

1. Program DFI Update Timing Parameters¹.
 - Set DFIUPDTMG0.dfi_t_ctrlupd_min.
 - Set DFIUPDTMG0.dfi_t_ctrlupd_max.
2. Program Refresh register.
 - Table 9-7 shows the maximum RFSHMOD0.refresh_burst if RFSHMOD0.per_bank_refresh is '0'.

Table 9-7 Upper Limit of All-bank Refresh Burst

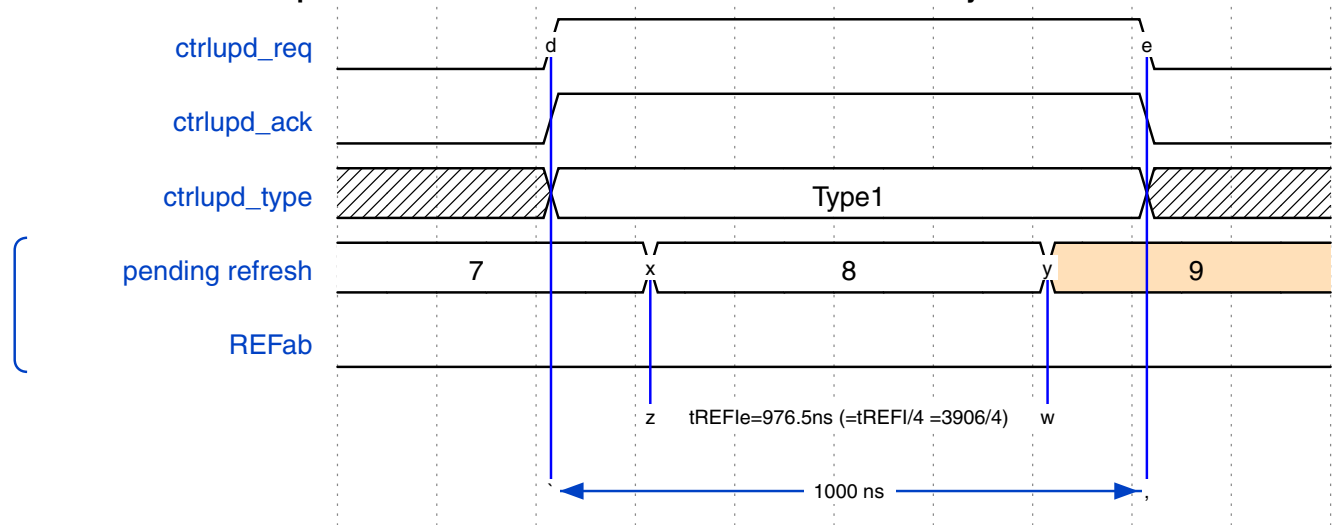
Maximum PPT2 Retraining Latency (ns)	Minimum Refresh Rate	tREFI * (Refresh Rate)	# of REFab Which PPT2 Can Postpone	Maximum RFSHMOD0.refresh_burst
500	0.25	977	1	7 (Up to burst-of-8 refresh)
500	0.125	488	2	6 (Up to burst-of-7 refresh)
1000	0.5	1953	1	7 (Up to burst-of-8 refresh)
1000	0.25	977	2	6 (Up to burst-of-7 refresh)
1000	0.125	488	3	5 (Up to burst-of-6 refresh)

- This limitation prevents any bank from violating tREFI.
- Figure 9-6 shows an example of fatal case where burst-of-8 refreshes is allowed in PPT2 retraining latency = 1000ns with 0.25x refresh rate. 1000ns of PPT2 gap can postpone 2 refreshes

1. Must be programmed while PPT2CTRL0.ppt2_en is '0'.

while 7 refreshes are already postponed so that the number of pending refreshes can be greater than 8. `RFSHMOD0.refresh_burst` must have proper upper bound to avoid this case.

Figure 9-6 PPT2 Can Postpone 2 All-bank Refresh Commands When Its Latency is 1000ns



- Table 9-8 shows the maximum `RFSHMOD0.refresh_burst` if `RFSHMOD0.per_bank_refresh` is 1.

Table 9-8 Upper Limit of Per-bank Refresh Burst

Maximum PPT2 Retraining Latency (ns)	Minimum Refresh Rate in REFpb	tREFIpb* (Refresh Rate)	# of REFpb Which PPT2 Can Postpone	Maximum <code>RFSHMOD0.refresh_burst</code>
500	1	488	2	62
500	0.7	342	2	62
500	0.5	244	3	61
500	0.25	122	5	59
500	0.125	61	9	55
1000	1	488	3	61
1000	0.7	342	3	61
1000	0.5	244	5	59
1000	0.25	122	9	55
1000	0.125	61	17	47

- This limitation prevents any bank from violating `tREFI`.
- If `RFSHMOD0.auto_refab_en==2`, minimum refresh rate in REFpb is 0.5x.
If `RFSHMOD0.auto_refab_en==3`, minimum refresh rate in REFpb is 0.25x.

3. Program `t_ctrlupd_interval`.

- Set `DFIUPDTMG2.dfi_t_ctrlupd_interval_type1` and `DFIUPDTMG2.dfi_t_ctrlupd_interval_type1_unit` to `t_ctrlupd_interval`.
 - PPT2 interval can get longer as much as `REFab/REFpb` is postponed which may require consideration. If `ppt2_wait_ref = 1`, Normal PPT2 will wait for any scheduled `REFab/REFpb` command before sending `ctrlupd_req` type1 after the interval timer expiration, which is 35.1 us ($= 9 * t_{REFI}$) at most if refresh rate is 1x.
- Set `PPT2CTRL0.ppt2_ctrlupd_num_dfi0`.
- Set `PPT2CTRL0.ppt2_ctrlupd_num_dfi1`.

4. Start Normal PPT2.

- Set `DFIUPDTMG2.ppt2_en` to '1'¹.
- PPT2 status can be observed via the register `PPT2STAT0.ppt2_state`.
- Note: DDRCTL sends single `ctrlupd` type1 as soon as possible after the `ppt2_en` is toggled from 0 to 1, except `ppt2_en` is set when DDRCTL is in initialization state.

To disable the Normal PPT2, follow this procedure:

- Poll `PPT2STAT0.ppt2_state` until it is '1'.
- Set `DFIUPDTMG2.ppt2_en` to '0'.
- Poll `PPT2STAT0.ppt2_state` until it is '0'.

Once Normal PPT2 has started, DDRCTL never automatically terminates it until `DFIUPDTMG2.ppt2_en` on current frequency becomes '0'. When DDRCTL has sent `ctrlupd_req` as many as specified at `PPT2CTRL0.ppt2_ctrlupd_num_dfi0/1` to retrain all rank0 and rank1 devices at `dfi0` and `dfi1`, DDRCTL continues sending `ctrlupd_req` from the beginning of rank0 device at `dfi0`.

Once PPT2 has stopped and resumed, DDRCTL sends single `ctrlupd_req` as soon as possible. Also, DDRCTL does not reset how many `ctrlupd_req` has been sent while it stopped. That is, if the previous `ctrlupd_req` was sent to retrain DFI0 Rank0 TxDQ1, it continues sending `ctrlupd_req` from DFI0 Rank0 TxDQ2.

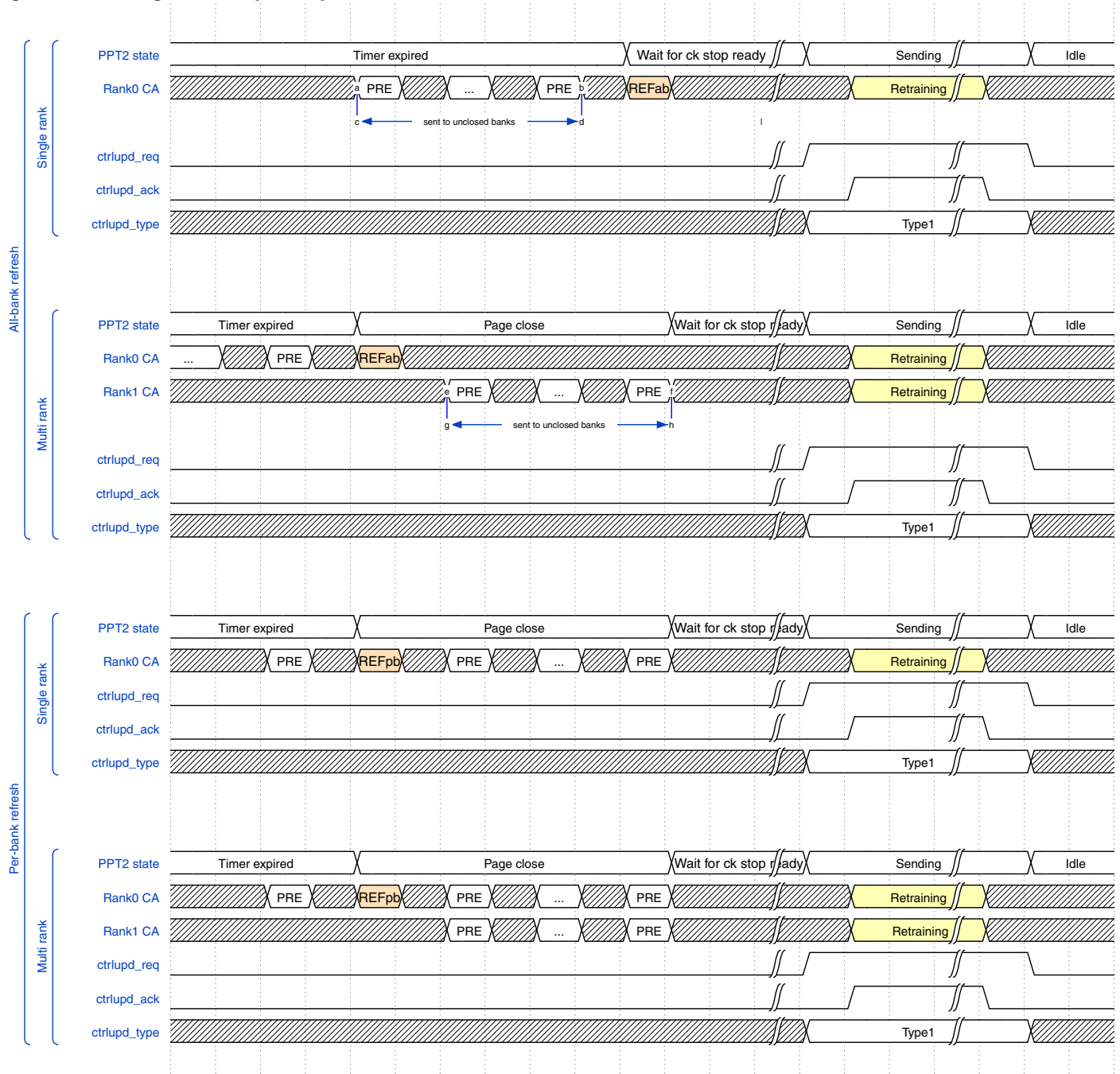
When `dis_auto_ctrlupd` is '0' and `ppt2_en` is '1', DDRCTL issues `ctrlupd_req` with both `dfin_ctrlupd_type = 0` and `dfin_ctrlupd_type = 1` periodically. Their interval is independent so that they can be issued consecutively.

Figure 9-7 on page 206 shows a typical flow of how DDRCTL sends a `ctrlupd_req` as Normal PPT2 during Normal state. In the figure, PRE always represents PREpb regardless of the REF type (REFab/REFpb) because DDRCTL does not support PREab in general.

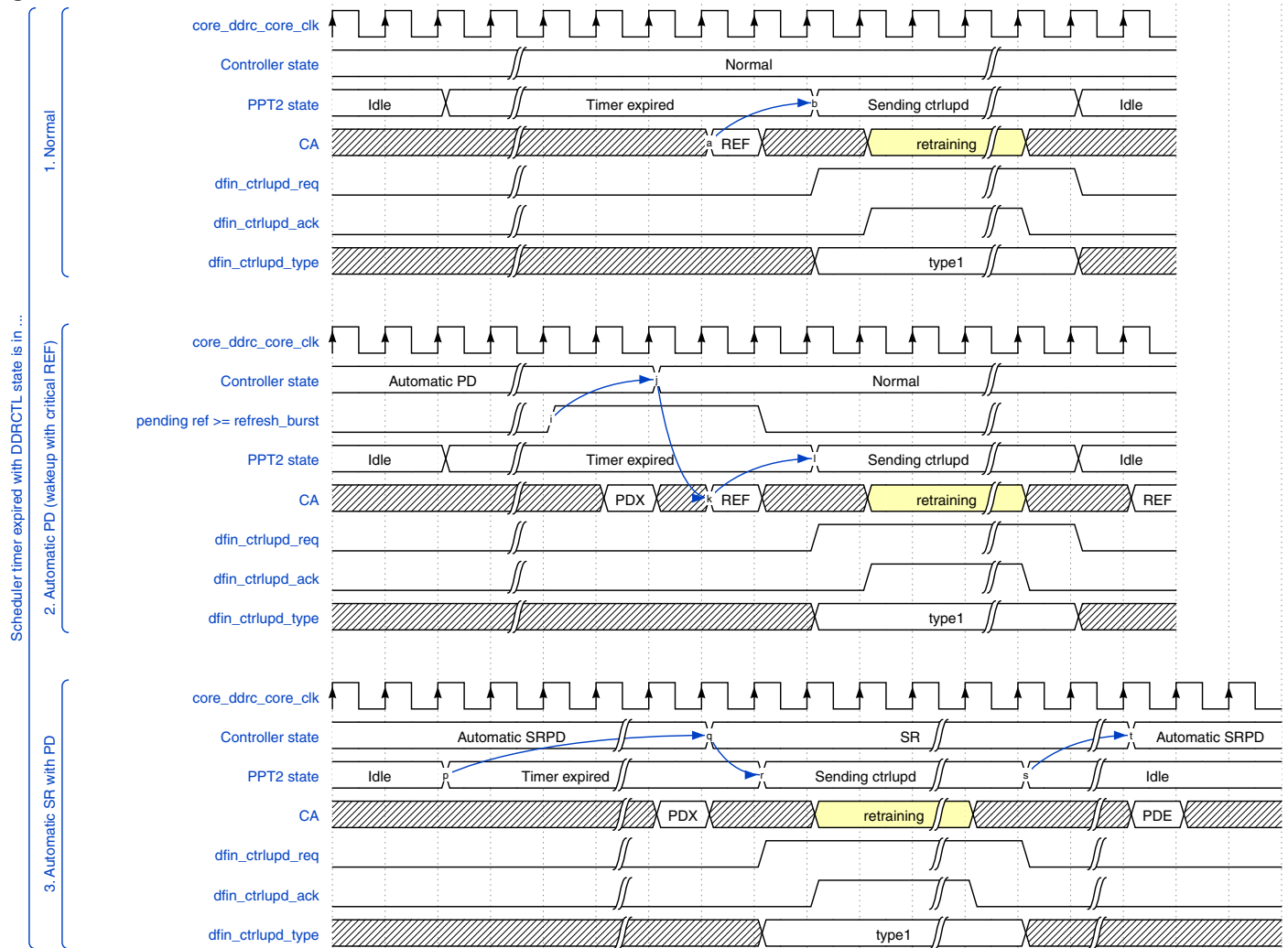
Once DDRCTL PPT2 internal timer that counts `t_ctrlupd_interval` expires,

1. DDRCTL waits for the next (periodic) REFab or REFpb command.
2. Close all opened banks to prepare for clock stop which PHY may do while retraining.
3. Issue `dfin_ctrlupd_req/ack` handshake with `dfin_ctrlupd_type = 1`.
4. Reset PPT2 timer to be `t_ctrlupd_interval` and start counting again.

1. `DFIUPDTMG2.ppt2_en` can be set per target frequency.

Figure 9-7 Regular ctrlupd_req Issue

During software or hardware low-power initiated SRPD, DDRCTL does not issue Normal PPT2 because PPT2 is not allowed during SRPD and exiting SRPD is software's responsibility. In contrast, when Normal PPT2 is required ($t_{ctrlupd_interval}$ timer expired) during automatic SRPD, DDRCTL exits SRPD to go to SR without PD state where PPT2 is allowed, sends `ctrlupd_req`, and then re-enters to SRPD. This case is one of the exceptions where `ctrlupd_req` does not follow any REFab/REFpb command due to Normal PPT2. When PPT2 is required during automatic PD, DDRCTL waits for PD exit which occurs eventually and then issues `ctrlupd_req`. [Figure 9-8](#) compares how DDRCTL issues PPT2 when it is triggered under (1) Normal state, (2) Automatic PD state, and (3) Automatic SRPD state.

Figure 9-8 How PPT2 Follows Automatic PD Exit or Automatic SRPD Exit**Note**

While Normal PPT2 is pending in PD state, it blocks DDRCTL from entering automatic SR even if `PWRTMG.selfref_to_x32` timer expires. DDRCTL enters automatic SR once pending PPT2 is issued.

9.5.5 Burst PPT2

Burst PPT2 can compensate a large amount of VT drift by issuing 32¹ set of the entire DFI/rank retraining. DDRCTL sends `ctrlupd_req` up to $32 \times (4 \times 2 \times 2)$ times as quickly as possible. Burst PPT2 is useful right after the initialization (after the first `dfi_init_complete = 1`) or when returning from a long-term SRPD or DSM, including LP3 I/O retention where the Normal PPT2 is not allowed so it has been a long time since the last PPT2. Figure 9-9 and Figure 9-10 show how the entire `ctrlupd_req` is sent during Burst PPT2. In this period, DDRCTL temporarily ignores `dfi_t_ctrlupd_interval_type1`. Consecutive `ctrlupd_req` will be sent adding 24 cycles gap at minimum between the last `ctrlupd_req` going low and high².

1. You may reduce the number of times. '32 set' expects that the VT-drift is at the maximum.
2. This is a PHY requirement: PHY does not issue an ACK (and does not VT compensate the delay elements) when `dfi_ctrlupd_req` is asserted within 24 cycles of its de-assertion.

Assuming each `ctrlupd_req` consumes 500ns, Burst PPT2 will consume 256us ($=500\text{ns} \times 32 \times (4 \times 2 \times 2)$).

Figure 9-9 Burst PPT2 Schedule

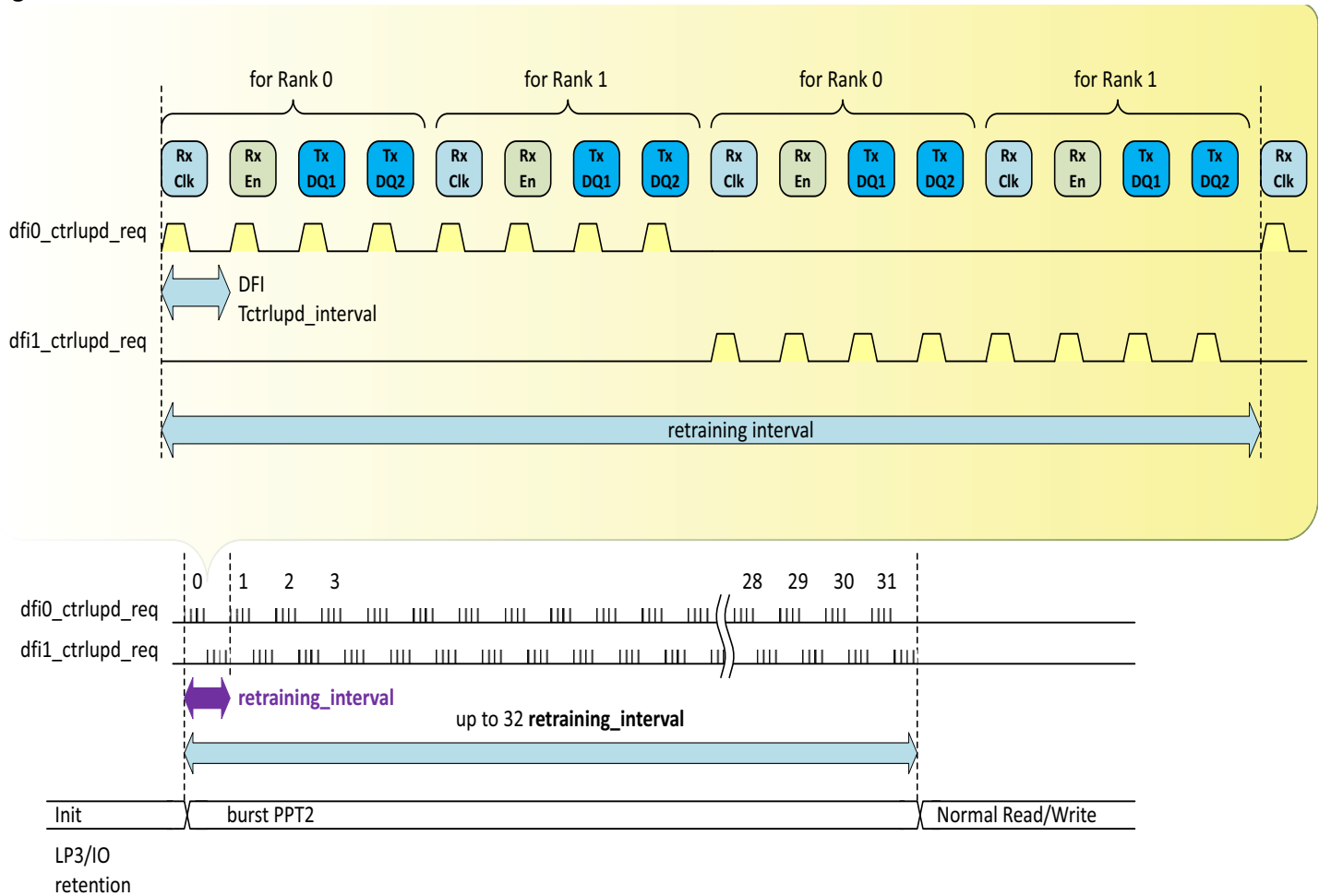
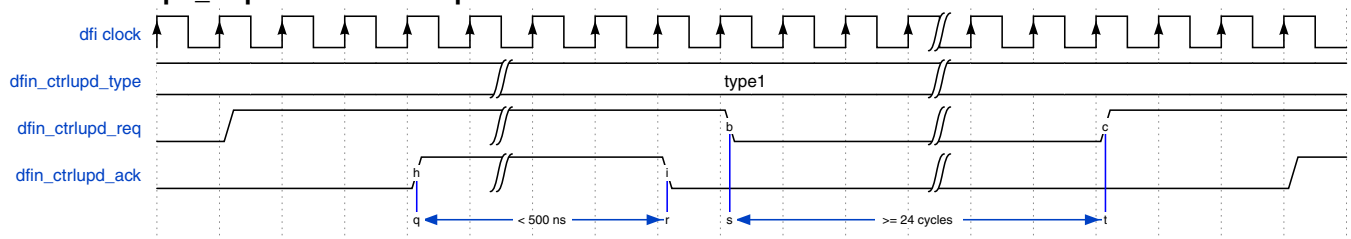


Figure 9-10 ctrlupd_req/ack Relationship in Burst PPT2



Any transaction that uses DQ bus, for example, READ, WRITE and MRR is not allowed once burst PPT2 starts and until it completes. However, other commands such as refresh and ZQCal may get in while burst PPT2 is ongoing and `ctrlupd_req` is low.

DDRCTL drives `cactive_ddrc` of the hardware low-power interface high during Burst PPT2 in order to deny any external low-power request.

To trigger the Burst PPT2, follow this procedure. Retraining functionalities other than PPT2 are assumed to be disabled (see “PPT2 Limitations” on page 200).

1. Program DFI Update Timing Parameters, Refresh registers and $t_{ctrlupd_interval}$ according to the procedure on how to enable Normal PPT2 in “Normal PPT2” on page 203 if needed.
2. Program the sum of the number of times to send `ctrlupd_req` for `dfi0` and `dfi1`.
 - Set `PPT2CTRL0.ppt2_burst_num`.
 - It will be '512' = $32 \times (4 \times 2 \times 2)$ times for Link ECC enabled & 2-rank & dual DFI system.
3. Disable transactions which use DQ. Disable other features that can get in the burst PPT2 if needed.
 - Set `OPCTRL1.dis_hif` to '1', so that no new commands are accepted by the controller and poll `OPCTRLCAM.dbg_wr_q_empty = 1`, `OPCTRLCAM.wr_data_pipeline_empty = 1`, `OPCTRLCAM.dbg_rd_q_empty = 1`, `OPCTRLCAM.rd_data_pipeline_empty = 1`.
 - Set `dis_dq` to '1'.
 - Set `derate_mr4_pause_fc` to '1' if `DERATECTL0.derate_eanble` is '1'.
 - Set `dis_auto_zq` to '1' if needed.
4. Disable low-power activities.
 - Set `PWRCTL.selfref_en` and `PWRCTL.powerdown_en` to '0'.
5. Start the Burst PPT2.
 - In a separate APB transaction, set `PPT2CTRL0.ppt2_burst` to '1'.
 - Note: `DFIUPDTMG2.ppt2_en` and `DFIUPD0.dis_auto_ctrlupd` can either be '0' or '1'.
6. Wait for Burst PPT2 to complete.
 - Burst PPT2 automatically stops when it completes.
 - Poll `PPT2STAT0.ppt2_burst_busy` until it is '0'.
7. Revert registers changed at step 3 and step 4.



Note

While Burst PPT2 is running and once `dfi_ctrlupd_req = 1` with `dfi_ctrlupd_type = 1` is issued, `DDRCTL` does not issue `dfi_ctrlupd_req = 1` with `dfi_ctrlupd_type = 0` even if `dis_auto_ctrlupd` is '0'.

9.5.6 Registers Related to PPT2

The following are the registers related to PPT2:

- `PPT2CTRL0`
- `PPT2STAT0`
- `DFIUPDTMG2`

For more information about these registers, see the “Register Descriptions” chapter of the DWC `DDRCTL` LPDDR5/4 Programming Guide.

10

Memory Control

This chapter contains the following sections:

- [“Mode Register Reads and Writes”](#) on page 212
- [“Data Bus Inversion \(DBI\)”](#) on page 214

10.1 Mode Register Reads and Writes

This topic contains the following sections:

- [“Overview of Mode Register Reads and Writes”](#) on page 212
- [“Mode Register Writes”](#) on page 212
- [“Mode Register Reads”](#) on page 212
- [“Registers Related to Mode Register Reads and Writes”](#) on page 213

10.1.1 Overview of Mode Register Reads and Writes

This section explains how to perform mode register reads and writes through software. Mode Register Reads (MRR) are used to read configuration and status data from mode registers in the SDRAM. Mode Register Writes (MRW) are applicable to all supported DDR protocols, and are used to write configuration data to mode registers in the SDRAM.

Access to the mode register is initiated by programming the MRCTRL0 and MRCTRL1 registers. This must be done in the following steps:

1. If performing MRR and using MRRDATA0/MRRDATA1 registers, write the MRCTRL0.mrr_done_clr to '1'.
This bit is self-clearing, and clears the MRSTAT.mrr_done register.
2. Poll MRSTAT.mr_wr_busy until it is '0'. This checks that there is no outstanding MR transaction. No writes must be performed to MRCTRL0 and MRCTRL1 if MRSTAT.mr_wr_busy = 1.
3. Write the MRCTRL0.mr_type, MRCTRL0.mr_addr, MRCTRL0.mr_rank and (for MRWs) MRCTRL1.mr_data to define the MR transaction.
4. In a separate APB transaction, write the MRCTRL0.mr_wr to '1'. This bit is self-clearing, and triggers the MR transaction. The DDRCTL then asserts the MRSTAT.mr_wr_busy while it performs the MR transaction to SDRAM, and no further accesses can be initiated until it is de-asserted.
5. If performing MRR, the MRR data is made available on the hif_mrr_data signals and the MRRDATA0/MRRDATA1 registers.

After performing step 3, do the following:

1. Set OPCTRL1.dis_auto_zq to '0'.
2. Set OPCTRL1.dis_dq to '0' to resume read and write transactions.

10.1.2 Mode Register Writes

MRW transactions can be specified for either any single rank, or any combination of several ranks, by programming MRCTRL0.mr_rank.

If any part of the SDRAM's MR register is updated by software, it is also the responsibility of the software to update corresponding INIT*. *mr* so that it is aligned to the SDRAM's MR register in LPDDR5.

10.1.3 Mode Register Reads

MRR transactions must only be performed to one rank at a time, to avoid bus contention.

When a MRR is performed, the mode register contents are available on hif_mrr_data, qualified by hif_mrr_data_valid and the MRRDATA0/MRRDATA1 register, after DDRCTL issues the mode register read command to the SDRAM. Note that the entire width of the SDRAM data is mapped to hif_mrr_data

and MRRDATA0/MRRDATA1 register. You must select the appropriate byte, depending on whether the bytes are swapped on the board or not.

**Note**

For Mode Register Read, if you use `hif_mrr_data` rather than MRRDATA0/MRRDATA1, you must add SoC logic to capture read data.

10.1.4 Registers Related to Mode Register Reads and Writes

The following are the registers related to the Mode Register Reads and Writes in LPDDR5:

- MRCTRL0
- MRCTRL1
- MRSTAT
- MRRDATA0
- MRRDATA1
- INITTMG0
- INITTMG1
- INITTMG2
- INITTMG3
- INITMR0
- INITMR1
- INITMR2
- INITMR3

For more information about these registers, see the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

10.2 Data Bus Inversion (DBI)

This topic contains the following sections:

- “Overview of Data Bus Inversion (DBI)” on page 214
- “Registers Related to DBI” on page 215

10.2.1 Overview of Data Bus Inversion (DBI)

Data Bus inversion is a power saving LPDDR5/4 specific feature which aims at minimizing:

- Logic 1s (consume more power than logic 0s)
- Simultaneous switching outputs

This is done by inverting each byte of read or write data, if the number of ‘1’s in that byte is greater than 4.

A DBI signal (1 bit per byte of data) is used to indicate whether this has been done or not. For writes, the DBI signal is shared with `dfi_wrddata_mask` on the DFI interface. For reads, there is a dedicated `dfi_rddata_dbi` signal.

Both write DBI and Data Mask are supported at the same time.

In this mode, a masked byte is indicated by `DDRCTL` setting the `dfi_wrddata` byte to 0xFF or 0xF8 which are selected by `DFIMISC.lp_optimized_write`, and setting the `dfi_wrddata_mask` bit to ‘0’.

Both read DBI and link ECC cannot be enabled simultaneously.

The DFI Specification indicates that DBI can be implemented either in the controller or in PHY. In the `DDRCTL`, DBI is controlled by `DFIMISC.phy_dbi_mode`. If this is set to ‘1’, DBI generation and data inversion are performed in the PHY. The read and write data on the DFI interface are same as that of when DBI is disabled (though masked writes are not allowed), and the signal `dfi_rddata_dbi` is ignored. When `DFIMISC.phy_dbi_mode` is set to ‘0’, DBI generation and data inversion is done by the `DDRCTL`. During write operation, the DBI value is put on the signal `dfi_wrddata_mask`. During read operation, `dfi_rddata_dbi` is evaluated and used to determine if the polarity of the read data needs to be inverted. The register `DBICTL` contains three fields namely:

- `rd_dbi_en` - enables read DBI
- `wr_dbi_en` - enables write DBI
- `dm_en` - enables data masking

The software must ensure that these values correspond to the values written to the mode register.



Note

Data Mask Disable (DMD) which is defined by `MR13`, has the opposite polarity of `DBICTL.dm_en`.

If read DBI is enabled, the JEDEC Specification indicates that the read latency (RL) must be increased. So, the following `DDRCTL` registers must be adjusted accordingly as they depend on CL or RL:

- `DFITMG0.dfi_t_rddata_en`
- `DRAMSET1TMG2.read_latency`
- `DRAMSET1TMG2.rd2wr`
- `DFITMG2.dfi_tphy_rdcslat`

- `DRAMSET1TMG24.rd2wr_s` (LPDDR5 only)
- `INIT4.emr2` (LPDDR4 only)

The required sequence for enabling or disabling DBI is as follows:

1. Enter the appropriate mode for writing quasi-dynamic registers.
2. Set `DBICTL.rd_dbi_en` and `DBICTL.wr_dbi_en` as required.
3. Set the DRAM mode register to match the previous settings.
4. Adjust the registers depending on CL (or RL) and AL (listed previously) as appropriate.
5. Exit quasi-dynamic mode by exiting self-refresh, or by re-enabling read/write traffic (For more information, see “Group 1: Registers that can be Written when No Read/Write Traffic is Present at the DFI” or “Group 2: Registers that can be Written in Self-refresh” in the “Programming” chapter of DWC DDRCTL LPDDR5/4 Programming Guide.

10.2.2 Registers Related to DBI

For more information on registers related to the DBI, see the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

11

Low-Power and Power-Saving Features

This chapter contains the following sections:

- [“Overview of Power Saving Features” on page 218](#)
- [“SDRAM Power Saving Features” on page 219](#)
- [“Power Saving in PHY Through DFI Low-Power Control Interface” on page 228](#)
- [“Hardware Low-Power Interfaces” on page 229](#)
- [“Fast Frequency Change” on page 239](#)
- [“Hardware Fast Frequency Change \(HWFFC\) Support” on page 240](#)
- [“Low-Power Optimized Write Data for LPDDR5/4 Masked Write” on page 250](#)
- [“BSM Clock Removal” on page 251](#)
- [“Signals Related to Power Saving” on page 253](#)

11.1 Overview of Power Saving Features

The DDRCTL supports various methods to save power within the system:

- Power saving opportunities within the SDRAM. The DDRCTL supports various SDRAM power saving modes such as precharge power-down, self-refresh, deep sleep mode, and support for disabling clock to the DRAM through `dfi_dram_clk_disable`.
For more information about this, see [“SDRAM Power Saving Features”](#) on page 219.
- Power saving opportunities within the PHY. For more information about this, see [“Power Saving in PHY Through DFI Low-Power Control Interface”](#) on page 228.
- Power saving opportunities from an external SoC low-power controller driven through an external hardware low-power interface (based on AMBA 4 AXI protocol low-power control interface). For more information about this, see [“Hardware Low-Power Interfaces”](#) on page 229.
- Power saving opportunities within the internal module BSM (Bank State Machine), whose clock can be gated while SDRAM is idle. For more information, see [“BSM Clock Removal”](#) on page 251.

11.2 SDRAM Power Saving Features

This topic contains the following sections:

- [“Overview of SDRAM Power Saving Features”](#) on page 219
- [“Precharge Power-Down”](#) on page 220
- [“Self-Refresh”](#) on page 221
- [“Deep Sleep Mode \(LPDDR5\)”](#) on page 225
- [“Assertion of dfi_dram_clk_disable”](#) on page 226

11.2.1 Overview of SDRAM Power Saving Features

In multi-rank systems, these power saving modes cannot be applied on a per-rank basis. If applied, they are always applied globally.

When enabled, the DDRCTL automatically enters and exits precharge power-down mode based on a programmable idle timeout period.

Self-refresh can be entered/exited through three ways:

- Based on a programmable idle timeout period (similar to precharge power-down idle timeout)
- Explicitly controlled through software
- Through the hardware low-power interfaces

In addition, `dfi_dram_clk_disable` can be asserted to disable the clocks to the DRAM. This can be done in the following modes:

- Self-refresh power-down
- Power-down
- Deep Sleep Mode (LPDDR5 only)

It also includes clock stop.

In relation to the hardware low-power interface, optional support is provided to drive `cactive_ddrc` low under idle conditions in normal or power-down or automatic self-refresh modes. Optional support for power-down, self-refresh, and clock stop to be forcefully exited through `cactive_in_ddrc = 1` is also provided.



Note

It is valid to enable any combination of Power-down and Self-refresh modes simultaneously:

- Power-down: `PWRCTL.powerdown_en = 1`
- Automatic self-refresh: `PWRCTL.selfref_en = 1`
- Software self-refresh: `PWRCTL.selfref_sw = 1`

Enabling assertion of `ddrc_dfi_dram_clk_disable` is valid in combination with any of the power saving modes.

**Note**

The idle timer fields in `PWRTMG/HWLPCTL` register continuously count down the programmed values once the controller enters an idle state irrespective of the enable bits in `PWRCTL` or `HWLPCTL`:

- `PWRTMG.powerdown_to_x32`
- `PWRTMG.selfref_to_x32`
- `HWLPTMG0.hw_lp_idle_x32`

11.2.2 Precharge Power-Down

This section discusses entering and exiting Precharge Power-down.

11.2.2.1 Entering Precharge Power-Down

When `PWRCTL.powerdown_en = 1`, `DDRCTL` automatically enters precharge power-down when the period specified by `PWRTMG.powerdown_to_x32` has passed while the `DDRCTL` is idle (except for issuing refreshes).

Entering precharge power-down mode involves the following steps:

1. If there is a self-refresh exit previously, wait for at least one refresh command (or 8 per-bank refresh commands if per-bank refresh is enabled) to all active ranks. Auto-refresh logic must be enabled, or refresh must be issued using direct software requests of refresh command through `OPCTRLCMD.rank*_refresh`.
2. Precharging (closing) all open pages. Pages are closed one-at-a-time in no specified order.
3. Waiting for `tRP` (row precharge) idle period.
4. Issuing the command to enter precharge power-down. For multi-rank systems, all chip-selects are asserted so that all ranks enter precharge power-down simultaneously. Power-down entry commands are issued separately for even and odd ranks.
5. This step occurs only if DFI low-power interface for power-down is enabled (`DFILPCFG0.dfi_lp_en_pd`). Attempts an entry to low-power mode through DFI low-power interface and with both `dfi_lp_ctrl_wakeup` and `dfi_lp_data_wakeup` set by `DFILPTMG0.dfi_lp_wakeup_pd`. The low-power entry attempt is delayed with `DFITMG0.dfi_t_ctrl_delay + DRAMSET1TMG7.t_cksre` clock cycles, this is needed to satisfy SDRAM timings related to disabling clocks when the PHY is programmed to gate the clock to save maximum power.

If the `DDRCTL` receives a read or write request from the SoC during step 2 or step 3, the power-down entry is immediately canceled. The same is true if `PWRCTL.powerdown_en` is driven to '0' during step 2 or step 3. Once the power-down entry command is issued, then proper power-down exit is required.

11.2.2.2 Exiting Precharge Power-Down

Once the `DDRCTL` has put the DDR SDRAM devices in precharge power-down mode, the `DDRCTL` automatically performs the precharge power-down exit sequence for any of the following reasons:

- A refresh cycle is required to any rank in the system.
- The `DDRCTL` receives a new request from the SoC.

- A self-refresh entry is requested.
- PWRCTL.powerdown_en is set to '0'.

The DDRCTL follows these steps when exiting precharge power-down mode:

1. Inserting any NOP/deselect commands required to satisfy the tCKE requirement after entering precharge power-down.
2. This step occurs only if DFI low-power mode entry during power-down entry is successful. Performs an exit from DFI low-power mode. DFI low-power mode is exited after the wakeup time specified by DFILPTMG0.dfi_lp_wakeup_pd, but not earlier than DFITMG1.dfi_t_dram_clk_enable + DRAMSET1TMG6.t_cksrx clock cycles.
3. Issuing the power-down exit command. For multi-rank systems, all chip-selects are asserted so that all ranks exit precharge power-down simultaneously.
4. Issuing NOP/deselect for the period defined by tXP.



Note

Other supported DDR protocols do not specify the difference between fast and slow power-down exit.

11.2.3 Self-Refresh

This section contains the following subsections:

- [“LPDDR4/LPDDR5 Considerations”](#) on page 221
- [“Entering Self-Refresh Mode”](#) on page 223
- [“Exiting Self-Refresh”](#) on page 224

11.2.3.1 LPDDR4/LPDDR5 Considerations

The following are the self-refresh states for LPDDR4/LPDDR5:

- Self-Refresh
- Self-Refresh Power-Down

During Self-Refresh mode, the external clock input is needed and all input pins of SDRAM are active. SDRAM can accept the following commands, except the PASR Bank/Segment setting:

LPDDR4

- MRR-1
- CAS-2
- SRX
- MPC
- MRW-1
- MRW-2

LPDDR5

- PDE

- DSM
- MRR
- CAS
- SRX
- MPC
- MWR

The DDRCTL has `STAT.selfref_state` register which indicates the DDRCTL's state during Self-Refresh and Self-Refresh Power-down modes. Once the DDRCTL enters `STAT.selfref_state = 3'b001`, the next state must be `3'b010`. Also, the next state after `STAT.selfref_state = 3'b011` must be `3'b000`.

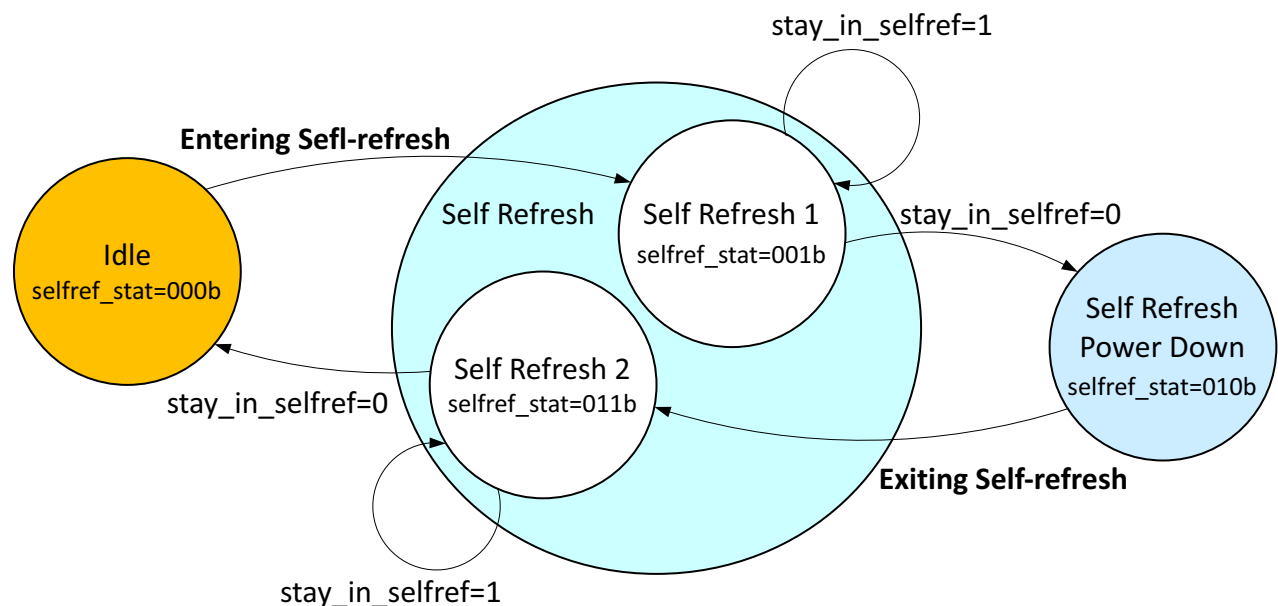
Table 11-1 lists the DDRCTL states and DRAM states for self-refresh.

Table 11-1 List of DDRCTL State and DRAM State for Self-Refresh

STAT.selfref_state	DDRCTL State	DRAM State
000	Neither Self-Refresh nor Self-Refresh Power-Down	Neither Self-Refresh nor Self-Refresh Power-Down
001	Self-Refresh 1	Self-Refresh
010	Self-Refresh Power-Down	Self-Refresh Power-Down
011	Self-Refresh 2	Self-Refresh
100 (LPDDR5 only)	Deep Sleep Mode	Deep Sleep Mode

The DDRCTL has `PWRCTL.stay_in_selfref` register to stay in Self-Refresh mode. During Self-Refresh mode, the DDRCTL can issue MRR, MRW, and SRX commands.

Figure 11-1 State Transition of Self-Refresh for LPDDR4



11.2.3.2 Entering Self-Refresh Mode

The DDRCTL puts the DDR SDRAM devices into Self-Refresh mode in the following cases:

- When the `PWRCTL.selfref_en` bit is set and no reads or writes are pending in the DDRCTL for the period specified by `PWRTMG.selfref_to_x32`. This is referred to as automatic Self-Refresh.
- When the PHY Master Interface is enabled by setting `DFIPHYMSTR.dfi_phymstr_en` bit and there is a `dfi_phymstr_req` coming from the PHY. In this case the `DDRC'shif_cmd_stall` is not driven high (the controller can accept commands on HIF) and existing controller commands in the DDRC are not executed before the entering the Self-Refresh mode sequence occurs.
- When the `PWRCTL.selfref_sw` bit is set and there are no outstanding read or write commands in the DDRC, this is referred to as the software self-refresh entry. This means that the DDRCTL cannot put SDRAM into Self-Refresh as long as write/read commands are being entered into the DDRCTL.
- When a hardware low-power entry request occurs (on `csysreq_ddrc/csysack_ddrc`) with `cactive_in_ddrc = 0`, no outstanding commands, and as long as the DDRCTL is not in init or deep sleep mode. This is referred to as an accepted hardware low-power self-refresh entry. When accepted, the `DDRC'shif_cmd_stall` is driven high to stop new commands from being accepted and existing controller commands in the DDRC are performed before the following sequence occurs.

Entering Self-Refresh mode involves the following steps:

1. If there is a self-refresh exit previously, wait for at least one refresh command (or 8 per-bank refresh commands if per-bank refresh is enabled) to all active ranks. Auto-refresh logic must be enabled, or refresh must be issued using direct software requests of refresh command through `OPCTRLCMD.rank*_refresh`.
2. Precharging (closing) all open pages. Pages are closed one-at-a-time in no specified order.
3. Waiting for tRP (row precharge) idle period. If a new command is received on the HIF during this time, the self-refresh entry is canceled.
4. Issuing the command to enter self-refresh mode. For multi-rank systems, all chip-selects are asserted so that all ranks enter self-refresh simultaneously.

If `PWRCTL.stay_in_selfref` is set to '1' before the SRE command is issued, DDRCTL does not enter the Self-Refresh Power-Down mode. In this case, right after `PWRCTL.stay_in_selfref` is set to '0', DDRCTL enters the Self-Refresh Power-Down mode. When `PWRCTL.stay_in_selfref` is '0' before SRE command issued, DDRCTL enters the Self-Refresh Power-Down mode automatically.

If the PHY Master Interface is enabled and `dfi_phymstr_req` comes before the SRE command is issued, DDRCTL does not enter the Self-Refresh Power-Down mode. In this case, right after `dfi_phymstr_req` is dropped, DDRCTL enters Self-Refresh Power-Down mode.

5. This step occurs only if DFI low-power interface for self-refresh is enabled (`DFILPCFG0.dfi_lp_en_sr`). Attempts an entry to low-power mode through DFI low-power interface with both `dfi_lp_ctrl_wakeup` and `dfi_lp_data_wakeup` set by `DFILPTMG0.dfi_lp_wakeup_sr`. The low-power entry attempt is delayed with `DFITMG0.dfi_t_ctrl_delay + DRAMSET1TMG5.t_cksre` clock cycles, this is needed to satisfy SDRAM timings related to disabling clocks when the PHY is programmed to gate the clock, to save maximum power.

Note, that `STAT.selfref_type` register field is 2'b11 if automatic self-refresh feature is the only cause of self-refresh. If software self-refresh or hardware low-power self-refresh occurs, `STAT.selfref_type = 2'b10`.

If Self-refresh entry is triggered by a PHY Master request, this step is skipped because DFI low-power interface is disabled when there is a `dfi_phymstr_req`.

Automatic self-refresh has the lowest priority followed by both software and hardware low-power self-refresh, PHY Master Interface has the highest priority. A software self-refresh entry means that a self-refresh exit occurs only if a software self-refresh exit occurs. Similarly, a hardware low-power self-refresh entry means a self-refresh exit occurs only if a hardware low-power self-refresh exit occurs. If both software and hardware low-power self-refresh entry occurs, self-refresh exit occurs only if both software and hardware low-power self-refresh exits occur. A self-refresh entry triggered by a PHY Master request means a self-refresh exit occurs only if the PHY Master request is de-asserted.

11.2.3.3 Exiting Self-Refresh

The DDRCTL takes the DDR SDRAM out of self-refresh mode under the following conditions:

- Whenever the `PWRCTL.selfref_en` input is set to '0', or new commands are received by the DDRCTL, as long as automatic Self-Refresh is only cause for self-refresh entry.
- When the PHY Master Interface is enabled by setting `DFI_PHYMSTR.dfi_phymstr_en` bit and `dfi_phymstr_req` is de-asserted by the PHY.
- Whenever the `PWRCTL.selfref_sw` bit is de-asserted. This is referred to as software Self-Refresh exit.
- When a hardware low-power exit request occurs (on `csysreq_ddrc/csysack_ddrc`). This is referred to as hardware low-power self-refresh exit.

Exiting Self-Refresh mode involves the following steps:

1. Inserting any NOP/deselect commands required to satisfy the tCKE/tCKESR/tSR requirements after entering Self-Refresh.
2. If `DFIUPD0.dis_auto_ctrlupd_srx = 0` and `DFIUPD0.ctrlupd_pre_srx = 1`, issuing a `dfi_ctrlupd_req` in accordance with the DFI specification. If Self-refresh is entered, without Power-Down, `dfi_ctrlupd_req` is not issued. This situation can occur only when there is a PHY Master request.
3. This step occurs only if DFI low-power mode entry during self-refresh entry is successful. DFI low-power mode is exited after the wakeup time specified by `DFILPTMG0.dfi_lp_wakeup_sr`, but not earlier than `DFITMG1.dfi_t_dram_clk_enable + DRAMSET1TMG5.t_cksrx` clock cycles.
4. Issuing the self-refresh exit command.

If `PWRCTL.stay_in_selfref` is set to '1' before Self-Refresh Power-Down exit command is issued, DDRCTL does not exit Self-Refresh. In this case, right after `PWRCTL.stay_in_selfref` is set to '0', DDRCTL exits Self-Refresh and enters the Idle state. When `PWRCTL.stay_in_selfref` is 0 before self-refresh power-down command issued, DDRCTL exits Self-Refresh and enters the Idle state automatically.

If the PHY Master Interface is enabled and `dfi_phymstr_req` comes before the Self-Refresh Power-Down exit command is issued, DDRCTL does not exit Self-Refresh. In this case, right after `dfi_phymstr_req` is de-asserted, DDRCTL exits Self-Refresh and enters the Idle state.

5. If `DFIUPD0.dis_auto_ctrlupd_srx = 0` and `DFIUPD0.ctrlupd_pre_srx = 0`, issuing a `dfi_ctrlupd_req` in accordance with the DFI specification. If Self-Refresh is entered, without Power-Down, `dfi_ctrlupd_req` is not issued. This situation can occur only when there is a PHY Master request.
6. Issuing NOP/deselect as determined by protocol requirements before any relevant pending command can be executed (in parallel with the `dfi_ctrlupd_req`, if applicable).



Note When `PWRCTL.stay_in_selfref` is set to '1', DDRCTL does not exit Self-Refresh. Therefore, the SoC must not set it to '1' unless SoC must issue MRR/MRW during Self-Refresh.

11.2.4 Deep Sleep Mode (LPDDR5)

Deep Sleep Mode (DSM) is applicable for LPDDR5 devices only.

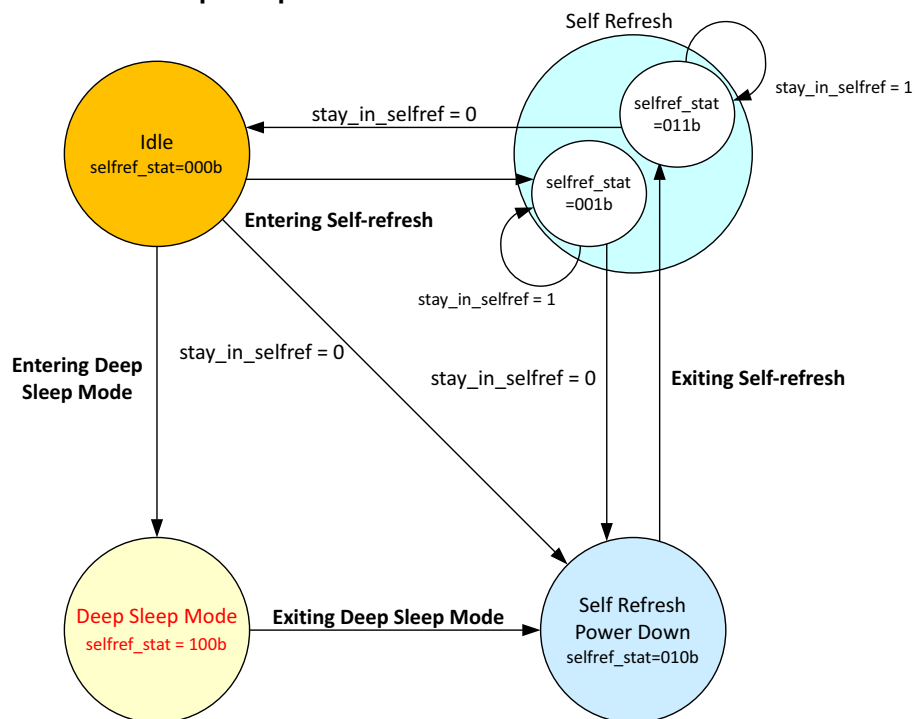
11.2.4.1 Entering Deep Sleep Mode

By setting the `PWRCTL.dsm_en` bit, you can put the LPDDR5 devices into Deep Sleep Mode, if all the following conditions are true:

- The DDRCTL is idle (except for issuing refreshes)
- `PWRCTL_selfref_sw=0`
- `PWRCTL_selfref_en=0`
- `HWLPCTL.hw_lp_en=0`

When the controller is in Deep Sleep Mode, the controller does not acknowledge the PHY master request (`dfi_phymstr_req`). Therefore, PHY master feature of the PHY must be disabled before entering Deep Sleep Mode.

Figure 11-2 State Transition of Deep Sleep Mode and Self-Refresh for LPDDR5



Entering Deep Sleep Mode involves the following steps:

1. If there is a self-refresh exit previously, wait for at least one refresh command (or 8 per-bank refresh commands if per-bank refresh is enabled) to all active ranks. Auto-refresh logic must be enabled, or refresh must be issued using direct software requests of refresh command through `OPCTRLCMD.rank*_refresh`.
2. Precharging (closing) all open pages. Pages are closed one-at-a-time (not in a specified order).
3. Waiting for tRP (row precharge) idle period.
4. Issuing the SRE command with DSM=1 to enter Deep Sleep Mode. For multi-rank systems, SRE commands must be sent to all ranks. This happens simultaneously.
5. This step occurs only if DFI low-power interface for Deep Sleep Mode is enabled (`DFILPCFG0.dfi_lp_en_dsm`). It attempts an entry to low-power mode through DFI low-power interface with both `dfi_lp_ctrl_wakeup` and `dfi_lp_data_wakeup` set by `DFILPTMG0.dfi_lp_wakeup_dsm`. The low-power entry attempt is delayed with `DFITMG0.dfi_t_ctrl_delay + DRAMSET1TMG11.t_ckmpe` clock cycles, this is needed to satisfy SDRAM timings related to disabling clocks when the PHY is programmed to gate the clock, to save maximum power.

If the DDRCTL receives a read or write request from the SoC during step 1 or step 2, the Deep Sleep Mode entry is immediately canceled. The same is true if `PWRCTL.dsm_en` is driven to '0' during step 1 or step 2. Once the Deep Sleep Mode entry command is issued, proper Deep Sleep Mode exit is required as described in the following section.

11.2.4.2 Exiting Deep Sleep Mode

Once the DDRCTL puts the DDR SDRAM devices in Deep Sleep Mode, and when the following software sequence is performed, the DDRCTL exits Deep Sleep Mode followed by Self-Refresh Power-Down mode, and eventually the DDRCTL exits Self-Refresh mode:

1. Before exiting DSM, it is assumed that `PWRCTL.dsm_en` is set to '1'.
2. Program `PWRCTL.selfref_sw = 1'b1`.
3. Program `PWRCTL.dsm_en = 1'b0` to exit power-down mode.
4. Polling until `STAT.selfref_state = 3'b010` (it means self-refresh power-down).
5. Wait for `tXDSM_XP`.
6. Program `PWRCTL.selfref_sw = 1'b0` to exit self-refresh power-down.



Note

In the DDRCTL, an exit from DFI low-power mode is performed prior to exiting the Deep Sleep Mode (occurs only if DFI low-power mode entry during Deep Sleep Mode is successful). DFI low-power mode exits after the wakeup time specified by `DFILPCFG0.dfi_lp_wakeup_dsm`, but not earlier than `DFITMG1.dfi_t_dram_clk_enable + DRAMSET1TMG5.t_cksrx` clock cycles (tCKMPX value is the same as tCKSRX).

11.2.5 Assertion of `dfi_dram_clk_disable`

Assertion of `dfi_dram_clk_disable` occurs only if `PWRCTL.en_dfi_dram_clk_disable = 1`.

`dfi_dram_clk_disable` is also dependent on the operating mode:

- `dfi_dram_clk_disable` can be asserted in the following modes:
 - Deep Sleep Mode (LPDDR5 only)
 - Self-refresh power-down
 - Power-down
 - Normal mode

This is the "Clock Stop" feature.



Note

`dfi_dram_clk_disable` cannot be asserted in WCK always on mode (`MSTR4.wck_on=1`) for LPDDR5.

The timing of the assertion and de-assertion of `dfi_dram_clk_disable` in various modes is as follows:

- In Self-Refresh and Self-Refresh power-down:
 - Asserted at least `DFITMG0.dfi_t_ctrl_delay + DRAMSET1TMG5.t_cksre - DFITMG1.dfi_t_dram_clk_disable` cycles after SRE command.
 - De-asserted at least `DFITMG1.dfi_t_dram_clk_enable + DRAMSET1TMG5.t_cksrx - DFITMG0.dfi_t_ctrl_delay` cycles before SRX command.
- In Power-down:
 - Asserted at least `DFITMG0.dfi_t_ctrl_delay + DRAMSET1TMG7.t_cksre - DFITMG1.dfi_t_dram_clk_disable` cycles after PDE command.
 - De-asserted at least `DFITMG1.dfi_t_dram_clk_enable + DRAMSET1TMG7.t_cksrx - DFITMG0.dfi_t_ctrl_delay` cycles before PDX command.
- In Normal mode (Clock Stop):
 - Asserted at least `DFITMG0.dfi_t_ctrl_delay - DFITMG0.dfi_t_dram_clk_disable` cycles after any command other than SRPDE/PDE/DSME.
 - De-asserted at least `DFITMG1.dfi_t_dram_clk_enable + DRAMSET1TMG6.t_ckcsx - DFITMG0.dfi_t_ctrl_delay` cycles before any command other than SRPDX/PDX/DSMX.

For more information about all of the modes, see `REGB_FREQf_CHc` in the "Register Descriptions" chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

11.3 Power Saving in PHY Through DFI Low-Power Control Interface

Based on whether SDRAM is in self-refresh power-down, power-down, deep sleep mode (LPDDR5 only) (see “[SDRAM Power Saving Features](#)” on page 219), the PHY can be placed in power saving modes through the DFI low-power interface. For more information, see “[Low-Power Control Interface](#)” on page 87.

11.4 Hardware Low-Power Interfaces

This topic contains the following sections:

- “Overview of Hardware Low-Power Interfaces” on page 229
- “DDRC Hardware Low-Power Interface” on page 229
- “AXI Low-Power Interface” on page 234

11.4.1 Overview of Hardware Low-Power Interfaces

Power saving opportunities from an external SoC low-power controller are driven through an external hardware low-power interface (based on AMBA 4 AXI protocol low-power control interface). There is a separate hardware low-power interface for each AXI port and dedicated hardware low-power interface for the DDRC.

11.4.2 DDRC Hardware Low-Power Interface

The hardware low-power interface can be used to enter/exit self-refresh mode. For this four-signal interface, the functionality is enabled by setting `HWLPCTL.hw_lp_en = 1`.



Note

For a single port, HIF only, `cactive_in_ddrc` is 1 bit wide and is driven by an external port. Else, `cactive_in_ddrc` is 1 bit per port, and each bit is driven by the `cactive_n` of each port.

Also, `cactive_in_ddrc` is internally used to evaluate whether self-refresh entry is allowed. It might have more registering stages than `csysreq_ddrc` to avoid CDC issues.

11.4.2.1 Entering Self-Refresh Using DDRC

Once the `HWLPCTL.hw_lp_en = 1` bit is set, the input `csysreq_ddrc` can trigger the entry into the self-refresh operating mode. A hardware low-power entry trigger is ignored/denied if any bit of the input `cactive_in_ddrc = 1`, or if the DDRC is not empty, or if a controller driven R/W/RMW command is on the controller interface (`hif_cmd_valid = 1`¹). Otherwise, it can be accepted depending on the current state of the DDRCTL. If accepted, the DDRC's `hif_cmd_stall = 1` to stop new commands from being accepted, existing controller commands in DDRC are performed before self-refresh entry occurs.

When `HWLPCTL.hw_lp_en = 1`, the outputs `csysack_ddrc` and `cactive_ddrc` provide feedback as required by the AXI low-power interface specification (this interface operation is defined by the AXI Specification). The `csysack_ddrc` signal acknowledges the request to go into self-refresh operating mode, and `cactive_ddrc` indicates when the DDRCTL is in self-refresh operating mode.

The `cactive_ddrc` output can also be used by an external low-power controller to decide when to request a transition to self-refresh. When `HWLPTMG0.hw_lp_idle_x32 > 0`, `cactive_ddrc = 0` indicates that the DDRCTL is idle for at least `HWLPTMG0.hw_lp_idle_x32 * 32 * DRAM clock cycles`.

1. `hif_cmd_valid` refers to `hif_cmd_valid` if `UMCTL2_DUAL_HIF = 0`, or `hif_rcmd_valid || hif_wcmd_valid` if `UMCTL2_DUAL_HIF = 1`.

11.4.2.2 Exiting Self-Refresh Using DDRC

In self-refresh mode, the `cactive_ddrc` output can be used by an external low-power controller to trigger a self-refresh exit. Always, `cactive_ddrc` is driven high when any bit of `cactive_in_ddrc = 1` or `hif_cmd_valid = 1` or (`dfi_phyupd_req = 1` && `DFIUPD0.dfi_phyupd_en = 1`) or (`dfi_phymstr_req = 1` && `DFIPHYMSTR.dfi_phymstr_en = 1`) or if there are outstanding commands in the DDRC. The paths from `cactive_in_ddrc`, `hif_cmd_valid`, `dfi_phymstr_req` and `dfi_phyupd_req` to `cactive_ddrc` are asynchronous. Therefore, `cactive_ddrc` asserts even if the clocks are removed. The low-power controller must re-enable the clocks when `cactive_ddrc` is driven high while in the self-refresh state.



Note

- The paths from `cactive_in_ddrc`, `hif_cmd_valid`, and `dfi_phyupd_req` to `cactive_ddrc` are asynchronous.
- `cactive_in_ddrc` and `hif_cmd_valid` are input ports of the DDRCTL in HIF only configurations. Otherwise, they are internal signals of the DDRCTL.
- `dfi_phyupd_req` is an input of the DDRCTL in all configurations. `dfi_phyupd_req = 1` generates `cactive_ddrc = 1` asynchronously (as long as `DFIUPD2.dfi_phyupd_en = 1`).

The behavior of the DDRCTL with respect to automatic clock stop, automatic precharge power-down or automatic self-refresh and `cactive_in_ddrc` is selectable through software (`HWLPCTL.hw_lp_exit_idle`). If `HWLPCTL.hw_lp_exit_idle = 1`, automatic clock stop, automatic precharge power-down and automatic self-refresh is exited (if `cactive_in_ddrc = 1`) and they do not occur until `cactive_in_ddrc = 0`. Note that it does not cause the exit of self-refresh by DDRC hardware low-power interface and/or software (`PWRCTL.selfref_sw`). This improves memory utilization performance. `cactive_in_ddrc` informs DDRC that there is RD/WR command imminent, and DDRC exits the various SDRAM power saving modes early so that it can perform the RD/WR sooner.

11.4.2.3 Hardware Removal Of Clock From DDRC

`core_ddrc_core_clk` can be removed by:

- AXI hardware low-power interface handshaking



Note

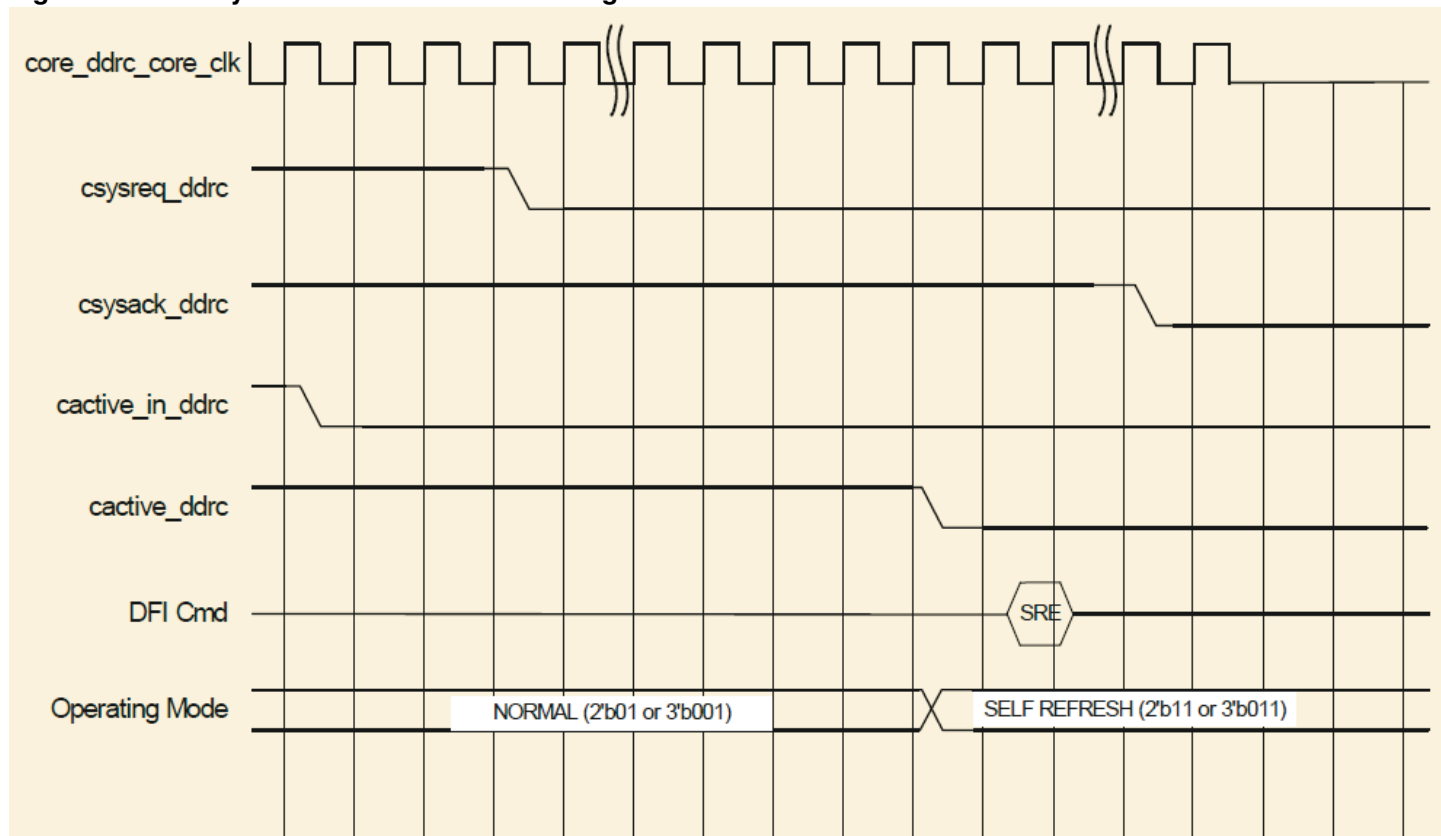
Dynamic and quasi dynamic registers can not be programmed if any of the clocks has been removed. Clocks must be turned back on before starting the programming sequence. Also the clock gating logic must ensure there are no glitches on the clocks when there are removed/enabled.

11.4.2.4 Removal of DDRC Clock Through AXI Hardware Low-Power Interface Handshaking

The `core_ddrc_core_clk` can be removed through AXI hardware low-power interface handshaking. An entry to self-refresh mode occurs as shown in [Figure 11-3](#). In this example, the system initiates the low-power state entry through de-assertion of `csysreq_ddrc`.

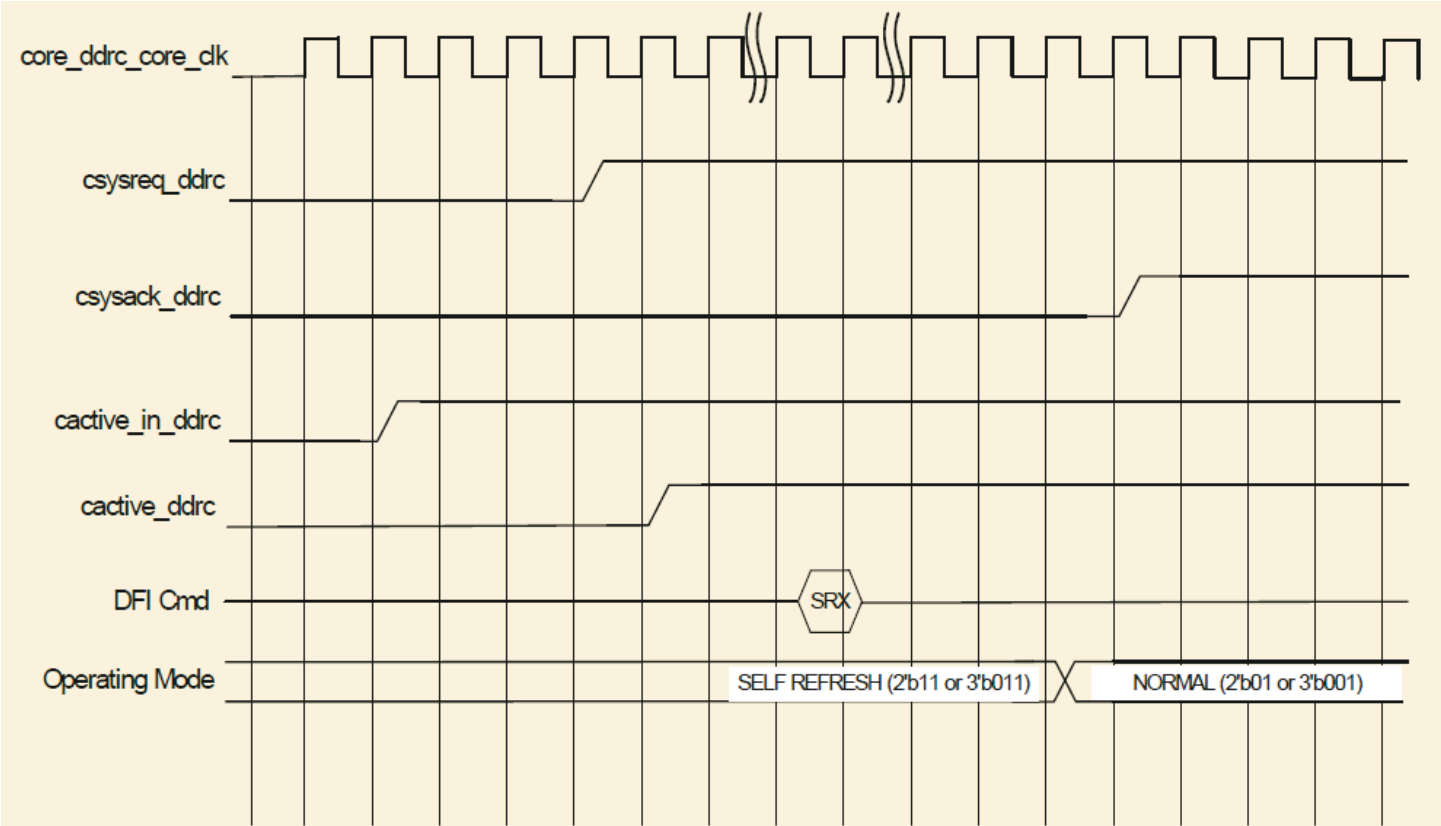
For HIF only, `core_ddrc_core_clk` can be removed from the DDRC if `cactive_ddrc = 0` when `cysack_ddrc` goes low. For more information about DDRC clock removal for non-HIF only configurations, see “[Hardware Removal of Clocks from AXI Ports](#)” on page 236.

Figure 11-3 Entry to Self-Refresh Mode Through Hardware Low-Power Interface of DDRC



An exit from self-refresh occurs as shown in [Figure 11-4](#). In this example, the system initiates the low-power state exit by asserting `csysreq_ddrc`.

Figure 11-4 Exit From Self-Refresh Mode Through Hardware Low-Power Interface of DDRC



11.4.2.5 Example of Power Saving Modes of DDRC

Table 11-2 shows an expected use case of the various power saving modes of the DDRC.

Table 11-2 Example of Power Saving Modes of the DDRC

Register or Hardware Low-Power I/F Action	Register Value	Behavior
PWRCTL.powerdown_en = 1 and PWRTMG.powerdown_to_x32 = X	Depending on the traffic profile, you can set the value of X.	<p>Puts SDRAM into power-down after X*32 cycles of idleness within DDRC.</p> <p>Note:</p> <p>This is Automatic Power-down.</p> <p>The DDRCTL puts the SDRAM into the power-down immediately, if PWRCTL.powerdown_en = 1 and PWRTMG.powerdown_to_x32 = 0.</p> <ul style="list-style-type: none"> ■ This assumes that there are no commands in progress or on the HIF. Power-down entry is based on idleness, and occurs only if the DDRCTL is idle. ■ There may be some delay before the actual power-down entry command is sent, as there may be other commands which need to be sent first (For example, precharge, refresh, ZQ, ctrlupd, phyupd). ■ The DFI low-power protocol may also need to be followed (if enabled).
PWRCTL.selfref_en = 1 and PWRTMG.selfref_to_x32 = Y	Value of Y must be greater than X. This ensures that power-down is entered prior to self-refresh.	<p>Puts SDRAM into self-refresh after Y*32 cycles of idleness within DDRC.</p> <p>This is Automatic Self-refresh.</p> <p>SDRAM moves from power-down to self-refresh.</p> <p>cactive_ddrc is not directly affected by Automatic Self-refresh.</p>
HWLPTMG0.hw_lp_idle_x32 = Z	Value of Z must be greater than Y. This ensures that system must already be in Automatic Self-refresh when cactive_ddrc = 0 occurs.	<p>cactive_ddrc = 0 occurs after Z*32 cycles of idleness within DDRC.</p>
<p>External Hardware Low-Power controller sees cactive_ddrc going low and knows that a hardware low-power request on csysreq_ddrc/csysack_ddrc must be accepted (unless the DDRC stops being idle).</p> <p>Hardware Low-Power controller sends a DDRC hardware low-power entry request by driving csysreq_ddrc = 0 and waiting for csysack_ddrc = 0 and checking cactive_ddrc value at the time.</p>	-	<p>DDRC accepts hardware low-power entry request.</p> <p>This means that if DDRC stops being idle, this does not move the SDRAM out of self-refresh.</p> <p>Once a DDRC hardware low-power entry request is accepted, self-refresh can only be exited through a hardware low-power exit request.</p>

Register or Hardware Low-Power I/F Action	Register Value	Behavior
If DDRC hardware low-power entry request is successful, the DDRC clock can be removed. If non-HIF only, this may not be the case. See “ Hardware Removal of Clocks from AXI Ports ” on page 236	-	-
If DDRC is receiving, or about to receive, a new command, due to asynchronous path from <code>cactive_in_ddrc/hif_cmd_valid^a</code> of DDRC to <code>cactive_ddrc</code> , <code>cactive_ddrc = 1</code> occurs, even if DDRC clock is removed. External Hardware Low-Power controller must observe this, re-start DDRC clock and perform DDRC hardware low-power exit request.	-	SDRAM exits self-refresh and performs RD/WR commands. idleness counters reset.

a. `hif_cmd_valid` refers to `hif_cmd_valid` if `UMCTL2_DUAL_HIF = 0`, or `hif_rcmd_valid` || `hif_wcmd_valid` if `UMCTL2_DUAL_HIF = 1`

11.4.3 AXI Low-Power Interface

The hardware low-power interface can be used to enter/exit a low-power state for the port. The input `csysreq_n` can trigger the entry into the low-power state. A hardware low-power entry trigger is accepted if the port is idle. Idle means there are no outstanding read or write commands from port `n` and there is no data accepted in the Write Data Queue. Otherwise, the request is rejected.

The outputs `csysack_n` and `cactive_n` provide feedback as required by the AXI low-power interface specification (this interface operation is defined by the AXI Specification). The port `csysack_n` acknowledges the request to go into a low-power state. The port `csysack_n` goes low for one cycle after `csysreq_n` goes low. The port `cactive_n` indicates whether the request is accepted or denied.

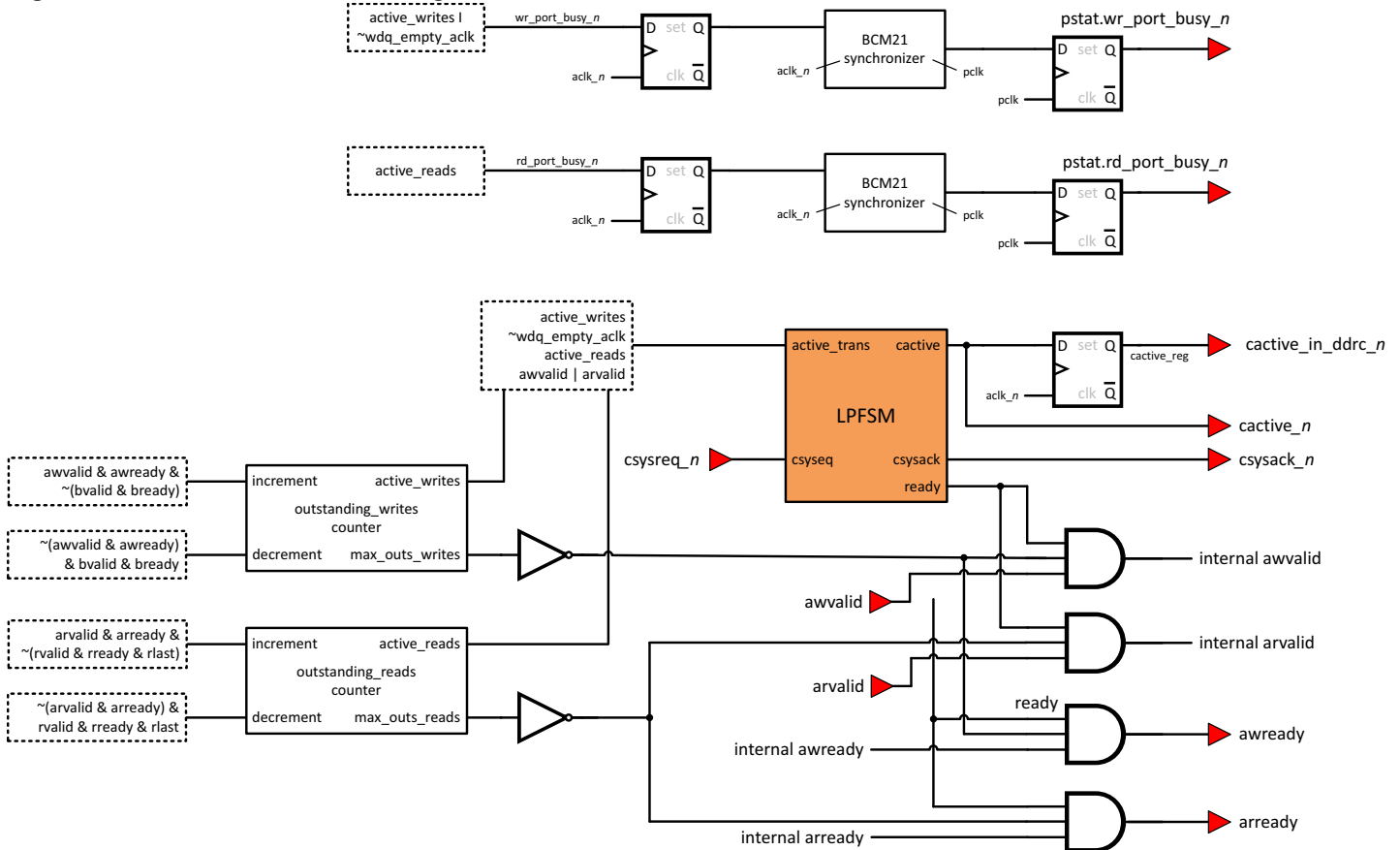
If the low-power request is accepted, the `awready_n` and `arready_n` signals are driven low and they stay low for one cycle, after `cactive_n` goes high.

The `cactive_n` output can also be used by an external low-power controller to decide when to request a transition to the low-power state for a port. When `cactive_n` is low, it indicates that there are no outstanding read or write commands from port `n`, there is no data in the Write Data Queue, and the port is idle for at least `UMCTL2_AXI_LOWPWR_NOPXCNT*acclk_n` cycles.

In the low-power state, the `cactive_n` output can be used by an external low-power controller to trigger a low-power state exit. Always, `cactive_n` is driven high when `arvalid_n` or `awvalid_n` is high, or if there are outstanding commands in the port. The paths from `arvalid_n`, `awvalid_n` and `wvalid_n` to `cactive_n` are asynchronous. Therefore, `cactive_n` asserts, even if the clocks are removed. The low-power controller must re-enable the clocks when `cactive_n` is driven high while in the low-power state.

The `cactive_n` signals are registered on `acclk_n` and input to the DDRC (`cactive_in_ddrc`).

[Figure 11-5](#) shows the block diagram of XPI LPFSM.

Figure 11-5 XPI LPFSM Block Diagram

The LPFSM has two different implementations depending on the setting of the parameter `UMCTL2_AXI_LOWPWR_NOPX_CNT`.

Figure 11-6 shows the LPFSM Implementation when `UMCTL2_AXI_LOWPWR_NOPX_CNT = 0`.

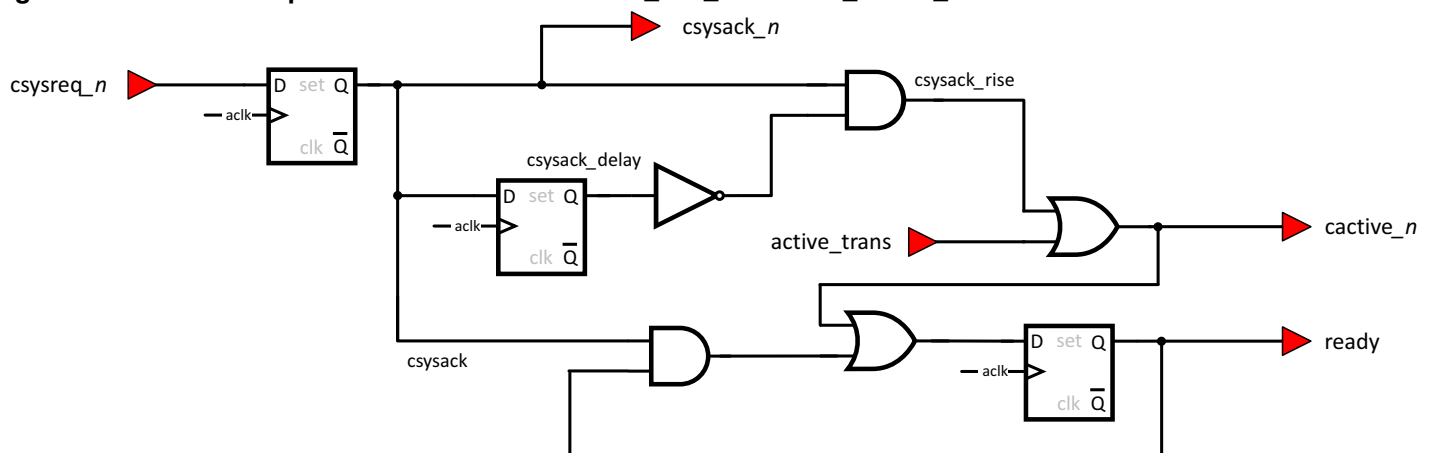
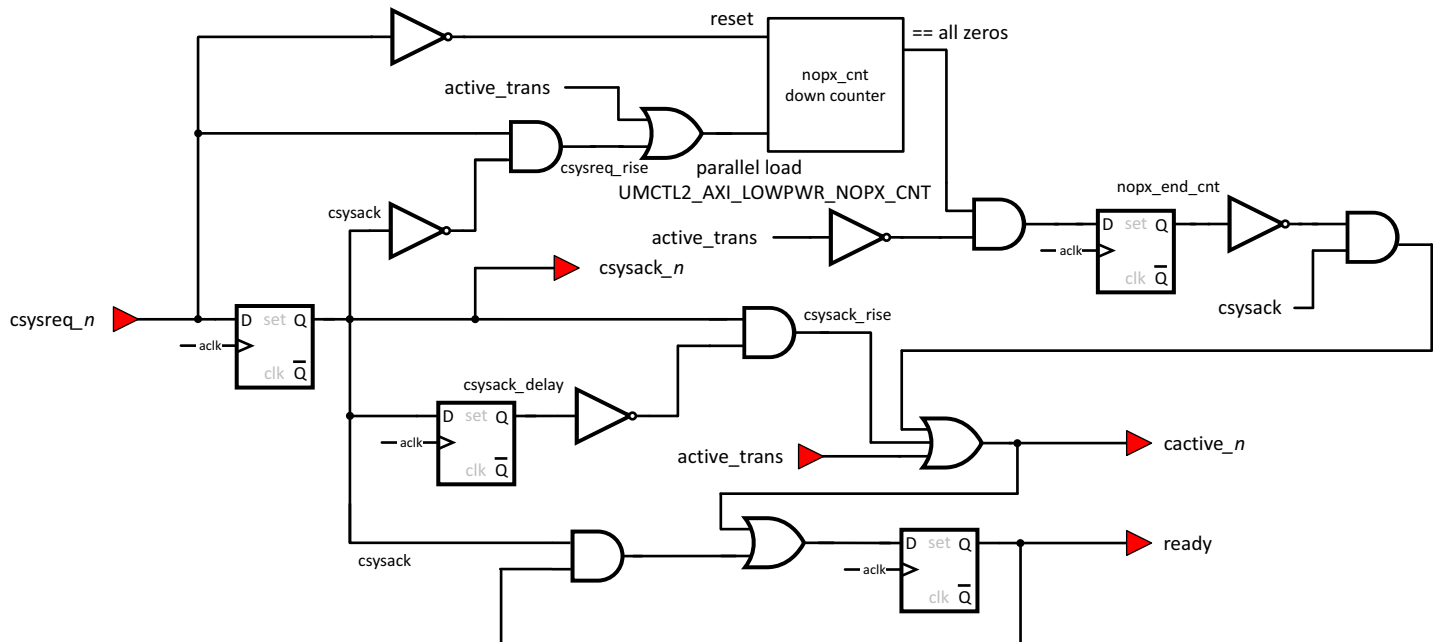
Figure 11-6 LPFSM Implementation when `UMCTL2_AXI_LOWPWR_NOPX_CNT = 0`

Figure 11-7 shows LPFSM Implementation when `UMCTL2_AXI_LOWPWR_NOPX_CNT > 0`.

Figure 11-7 LPFSM Implementation when UMCTL2_AXI_LOWPWR_NOPX_CNT > 0

11.4.3.1 Hardware Removal of Clocks from AXI Ports

There are two methods by which `aclk_n` can be removed:

- AXI hardware low-power interface handshaking



Note

Dynamic and quasi dynamic registers can not be programmed if any of the clocks has been removed. Clocks must be turned back on before starting the programming sequence. Also, the clock gating logic must ensure there are no glitches on the clocks when there are removed/enabled.

11.4.3.2 Removal of AXI Clock through AXI Hardware Low-Power Interface Handshaking

Removal of clocks using the AXI hardware low-power interface is preferred, as this method places the ports in a low-power state. `awready_n` and `arready_n` signals are driven low once the low-power request is accepted.

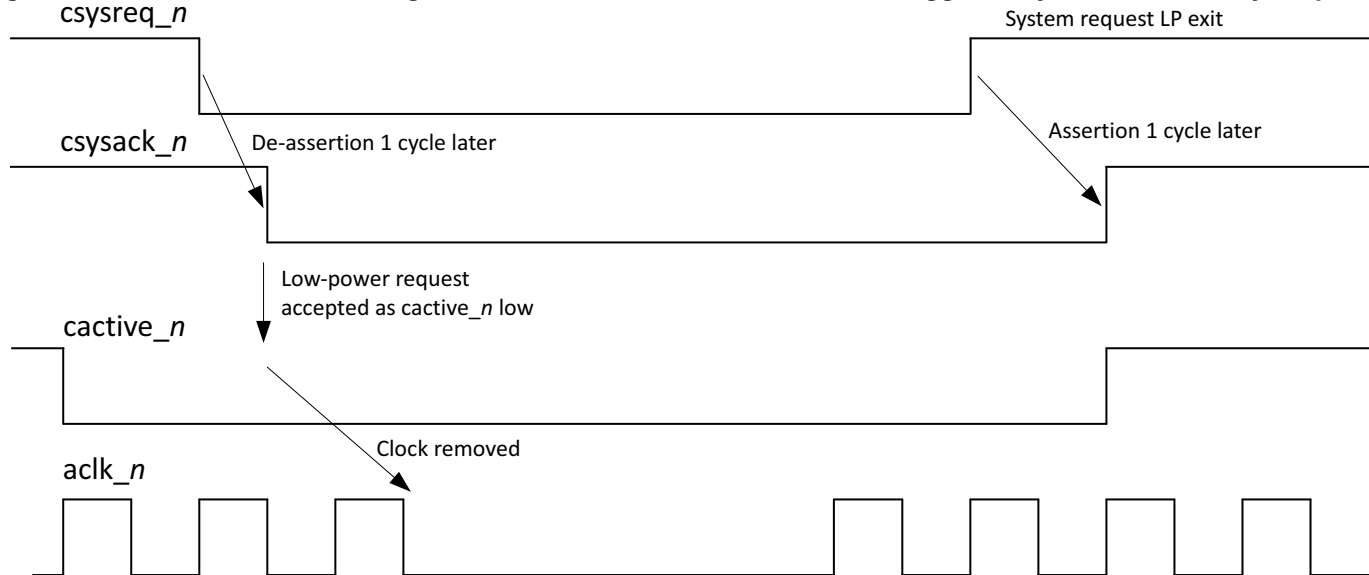
The low-power interface of all the AXI ports are independent. `aclk_n` for a port can be removed once the low-power requested has been accepted by the port.

If all the AXI ports are asynchronous (`UMCTL2_A_SYNC_n = 0`), the DDRC clock can be removed once the DDRC has accepted the low-power request.

If any of the ports are synchronous (`UMCTL2_A_SYNC_n = 1`), the DDRC clock can only be removed when `aclk_n` is removed for all synchronous ports. It must be re-enabled before any of the `aclk_n` for the synchronous ports are enabled.

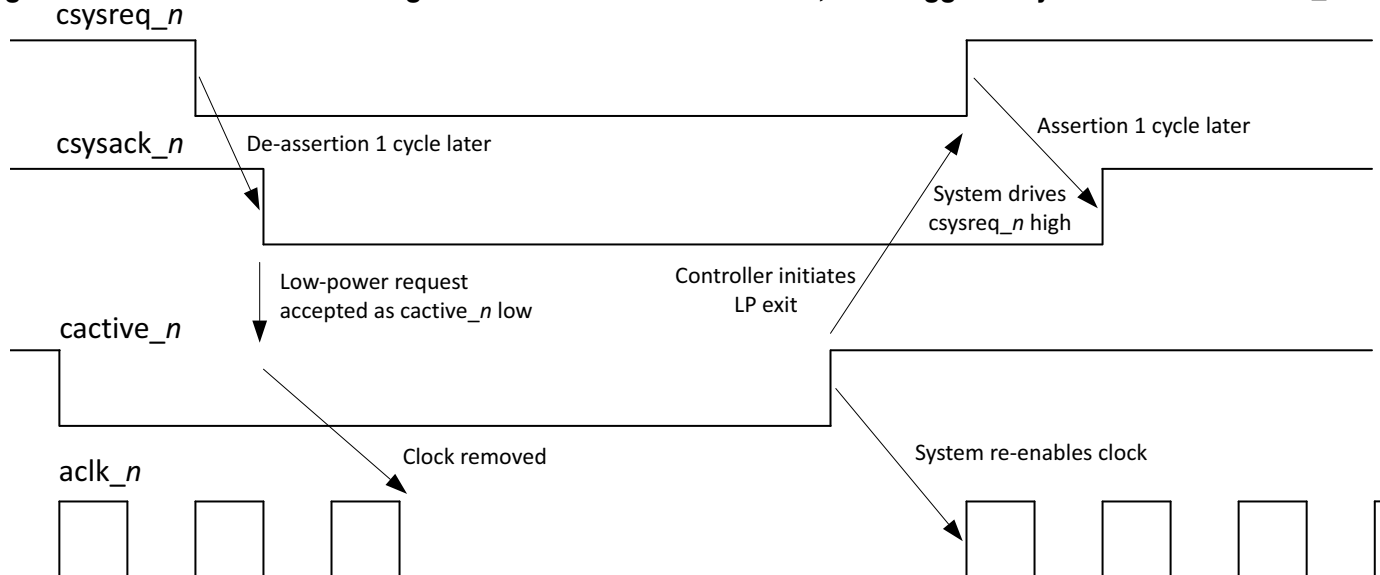
Clock removal using the AXI hardware low-power interface is shown in [Figure 11-8](#). In this example, the system initiates the low-power state exit through assertion of `csysreq_n`.

Figure 11-8 Clock Removal Using Hardware Low-Power Interface, Exit Triggered by Assertion of `csysreq_n`



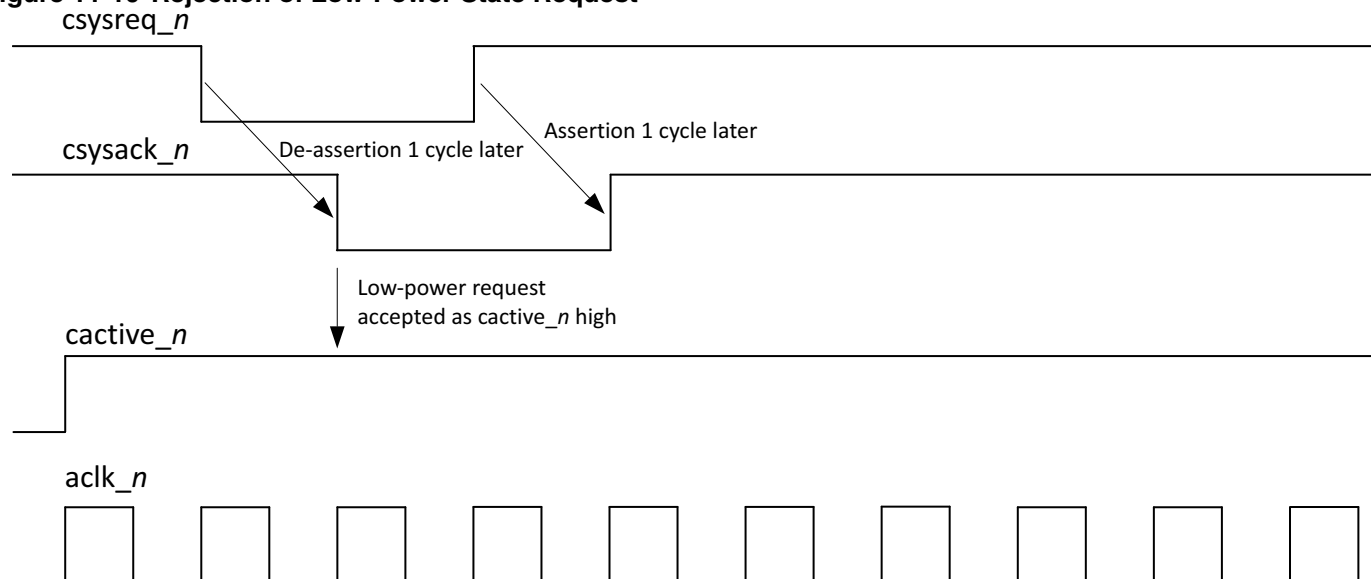
Clock removal using the AXI hardware low-power interface is shown in [Figure 11-9](#). In this example, the controller initiates the low-power state exit though assertion of `cactive_n`.

Figure 11-9 Clock Removal Using Hardware Low-Power Interface, Exit Triggered by Assertion of `cactive_n`



Rejection of the request is shown in [Figure 11-10](#). In this case, `aclk_n` cannot be removed.

Figure 11-10 Rejection of Low-Power State Request



With respect to interaction with “[DDRC Hardware Low-Power Interface](#)” on page 229, it is recommended to stagger a low-power entry of the DDRC until after all port's AXI low-power interface have successfully entered low-power. This is to allow the `caactive_n` to propagate to `caactive_ddrc_in` of DDRC hardware low-power interface.

11.5 Fast Frequency Change

The DDRCTL supports Fast Frequency Change, using up to four sets of timing registers. The alternative sets of timing registers can be found in `REGB_FREQf_CHc` registers. These registers may be written while the traffic is in progress using the first set of timing registers, thus reducing the software overhead at the time of frequency change.

The Fast Frequency Change sequence is described in the “Software Sequences” section of the “Programming” chapter in the DWC DDRCTL LPDDR5/4 Programming Guide:

11.6 Hardware Fast Frequency Change (HWFFC) Support



Note

This feature is supported only with limited configuration, and LPDDR4/LPDDR4X with single rank is only supported. That is:

- LPDDR5 is not supported.
 - Multiple rank is not supported.
 - Inline ECC, Link ECC, OCECC, REGPAR, OCCAP, OCSAP features cannot be supported together with HWFFC.
 - Data rate has to be within the range between 1066 Mbps and 4266 Mbps.
- For more information, contact Synopsys.



Note

When switching from higher frequency to lower frequency, the controller may violate the JEDEC requirement that no more than 16 refreshes must be issued within $2 \cdot t_{REFI}$. These extra refreshes are not expected to cause a problem in the SDRAM.

11.6.1 Hardware Fast Frequency Change (HWFFC) Overview

Hardware Fast Frequency Change (HWFFC) functionality in DDRCTL is intended to support clock frequency change procedure without the software intervention. HWFFC can support up to four sets of timing registers.

The following is a simplified HWFFC procedure that DDRCTL performs automatically:

1. Blocks all the AXI ports from taking further commands and data.
2. Drains commands from the controller and stops sending further transactions to SDRAMs.
3. Puts all the SDRAMs into self-refresh without powerdown (LPDDR4 only).
4. Programs several mode registers (MR) in all the SDRAMs as necessary.
5. Puts all the SDRAMs into self-refresh (or SR-Powerdown when LPDDR4).
6. Interacts with PHY by using DFI frequency change protocol.
7. Exits all the SDRAMs from self-refresh (or SR-Powerdown when LPDDR4).
8. Programs several mode registers (MR) in all the SDRAMs as necessary.
9. Exits all the SDRAMs from self-refresh without powerdown (LPDDR4 only).

During step (6), when `csysack_ddrc` becomes 0, clock frequency can be changed to a new target frequency. For more details, see “[LPDDR4 HWFFC Procedure](#)” on page 243.

The HWFFC process can be managed by a hardware low-power controller (see section “[Hardware Low-Power Interfaces](#)” on page 229) external to DDRCTL, using the DDRC Hardware Low-Power Interface. This interface is extended by the addition of three signals:

- `csysmode_ddrc`
- `csysdiscamdrain_ddrc` (This input signal needs to be fixed to 0)
- `csysfrequency_ddrc`

When the signal `csysreq_ddrc` is de-asserted, ‘1’ on `csysmode_ddrc` indicates that a frequency change of the `core_ddrc_core_clk` is required. The signal `csysfrequency_ddrc` indicates which set of timing

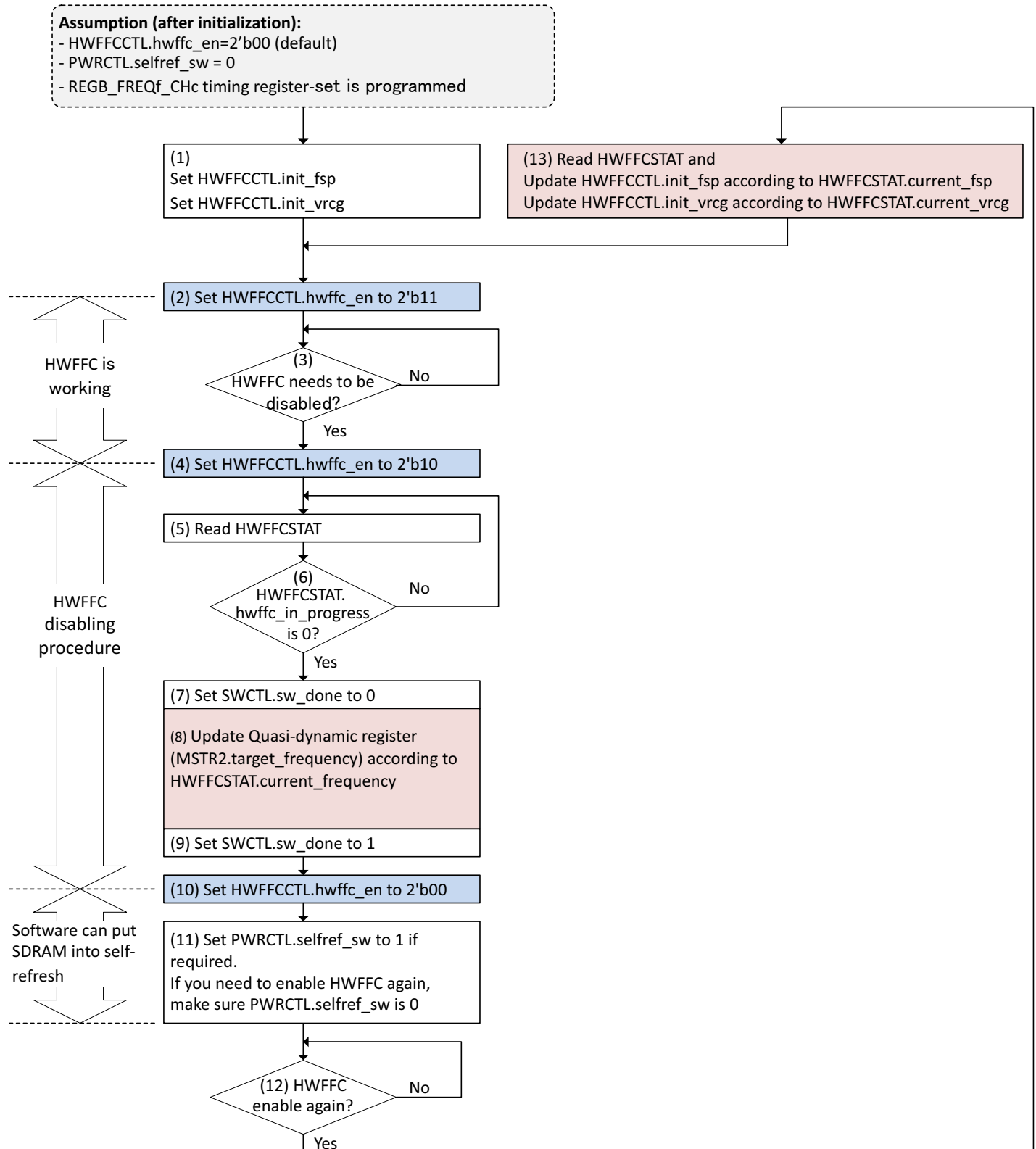
registers is used after the change of frequency. These signals must be held constant while `csysreq_ddrc` is de-asserted.

When either `HWFFCCTL.hwffc_en` or `MSTR0.lpddr4` is '0', even if the signal `csysreq_ddrc` is de-asserted and `csysmode_ddrc` is '1', then the DDRCTL does not initiate the HWFFC procedure.

Note

- A hardware low-power entry request and a HWFFC request cannot be accepted simultaneously. The signal `csysmode_ddrc` indicates which action to take place. When the signal `csysreq_ddrc` is de-asserted, '0' on `csysmode_ddrc` indicates that a hardware low-power entry request is required and the DDRCTL attempts to enter the self-refresh operating mode. See the section “[DDRC Hardware Low-Power Interface](#)” on page 229. Clocks may then be removed but HWFFC is not executed and the signal `csysfrequency_ddrc` is ignored.
- Whenever you need to put SDRAM into self-refresh mode by software when `HWFFCCTL.hwffc_en` is 2'b11, then you have to follow the procedure as shown in [Figure 11-11](#) on page 242.
- When the software updates `INITMRx`, `DRAMSETxTMGx`, or other timing registers it is strongly recommended to disable the HWFFC in advance, because the software must know which register-set (`REGB_FREQf_CHc`) is currently being used, so that the appropriate registers can be updated correctly. This prevents unexpected behavior from happening. To disable HWFFC follow the procedural steps from 4 to 10, as shown in [Figure 11-11](#) on page 242.
- When the HWFFC is requested, the DDRCTL stops new read/write commands from being accepted. All existing commands in the DDRCTL are performed before entering self-refresh.
- All the AXI clocks and the DDRC clock must be changed at the same time when `UMCTL2_A_SYNC_n = 1`. The external clock control logic must ensure there are no glitches on both the AXI and the DDRC clocks when clock frequency is changed.

[Figure 11-11](#) on page 242 shows the software sequence for disabling/enabling HWFFC. Software needs to comply with the following sequence if there is a need to disable/enable HWFFC by writing `HWFFCCTL.hwffc_en`. Software can put SDRAMs into self-refresh mode properly with the sequence.

Figure 11-11 Software Sequence for Disabling or Enabling of HWFFC

**Note**

- Updating the `HWFFCCTL.init_fsp` in Step 1 and 13 applies only for LPDDR4.
- Write-DBI/read-DBI settings can be changed only before step 2 or after step 10 (that is, `HWFFCCTL.hwffc_en` is '0').

11.6.2 Limitation of HWFFC

HWFFC supports only LPDDR4 protocol, but per-rank ODT/Vref feature is not supported.

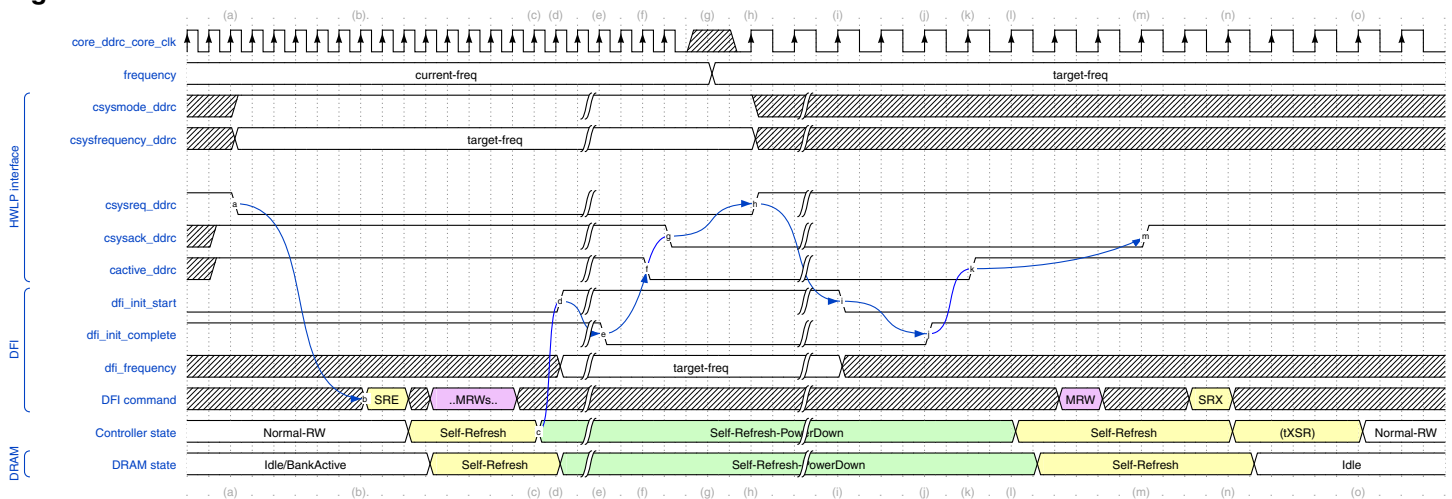
This functionality is intended to be used only with DWC LPDDR5/4/4X PHY, and works under the following conditions:

- DFI frequency change (`dfi_init_start = 1`) has priority over `dfi_phymstr_req` and `dfi_phyupd_req`.
The PHY must safely cancel the other requests when `dfi_init_start` is asserted.
- The PHY always acknowledges `dfi_init_start` (DFI frequency change request is never rejected).
- `dfi_init_start` can be asserted any time after de-assertion of `dfi_cke`.
DFI timing parameter `tinit_start_min = 1` can be assumed always.

11.6.3 LPDDR4 HWFFC Procedure

Figure 11-12 on page 243 shows the fundamental timing diagram of LPDDR4 HWFFC procedure.

Figure 11-12 LPDDR4 HWFFC Procedure



- The system requests frequency change by de-asserting `csysreq_ddrc`. `csysmode_ddrc = 1`, `csysfrequency_ddrc` and `csysdiscamdrain_ddrc` must remain at constant values while `csysreq_ddrc = 0`. `csysreq_ddrc` must be held de-asserted until `csysack_ddrc` is de-asserted.
- The DDRCTL issues SRE to put SDRAMs into self-refresh (without power-down), and then sends several MRW commands to update timing parameters for opposite side of current FSP, and lastly it switches FSP-OP to the opposite side. For more details, see ["MRW Commands Before Frequency Change"](#) on page 247.
- The DDRCTL de-asserts `dfi_cke` to put SDRAMs into SR-Powerdown.

- Figure 11-13 Variation After Frequency Change When ZQCTL2.dis_srx_zqcl=1, DFIUPD0.dis_auto_ctrlupd_srx=0, DFIUPD0.ctrlupd_pre_srx=0**

The diagram illustrates the timing of the DRAM interface signals across different phases (j) to (o). The signals are categorized into three main groups:

- core_ddrc_core_clk**: A periodic clock signal.
- frequency**: A signal indicating the target frequency.
- HWLP interface**: Signals including *csysmode_ddrc*, *csysfrequency_ddrc*, *csysreq_ddrc*, and *csysack_ddrc*.
- DFI**: Signals including *cactive_ddrc*, *dfl_init_start*, *dfl_init_complete*, *dfl_frequency*, and *dfl_ctrlupd_req*.
- DRAM**: Signals including *DFI command*, *Controller state*, and *DRAM state*.

The diagram shows the sequence of events, including the initiation of a Self-Refresh (SR) operation, the duration of the Self-Refresh period, and the subsequent return to the Normal-RW (Read/Write) state. The *DFI command* signal shows a sequence of commands: *MRW* (Memory Refresh Write) and *SRX* (Self-Refresh Exit).

Figure 11-14 Variation After Frequency Change When ZQCTL2.dis_srx_zqcl=1, DFIUPD0.dis_auto_ctrlupd_srx=0, DFIUPD0.ctrlupd_pre_srx=1

ZQCTL2.dis_srx_zqcl=1, DFIUPD0.dis_auto_ctrlupd_srx=0, DFIUPD0.ctrlupd_pre_srx=1

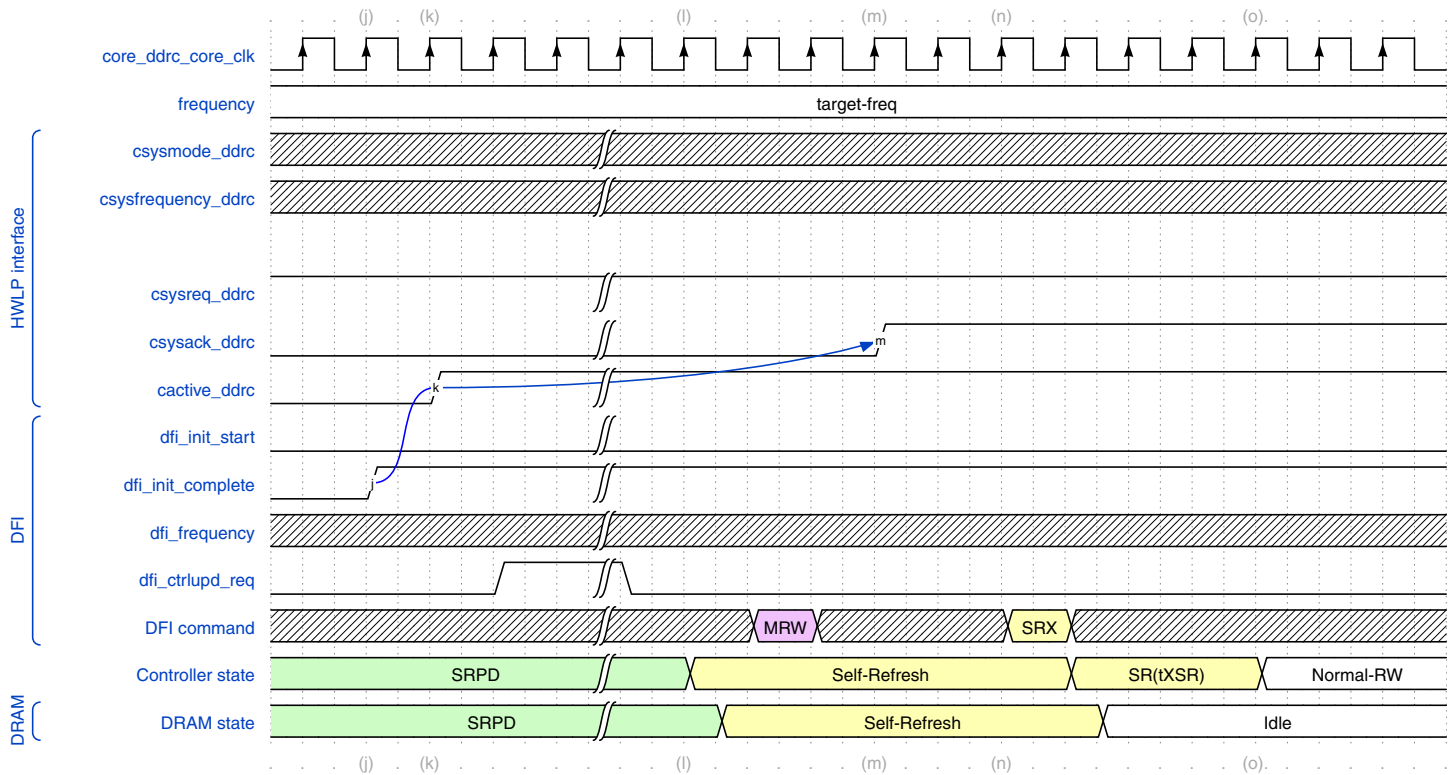
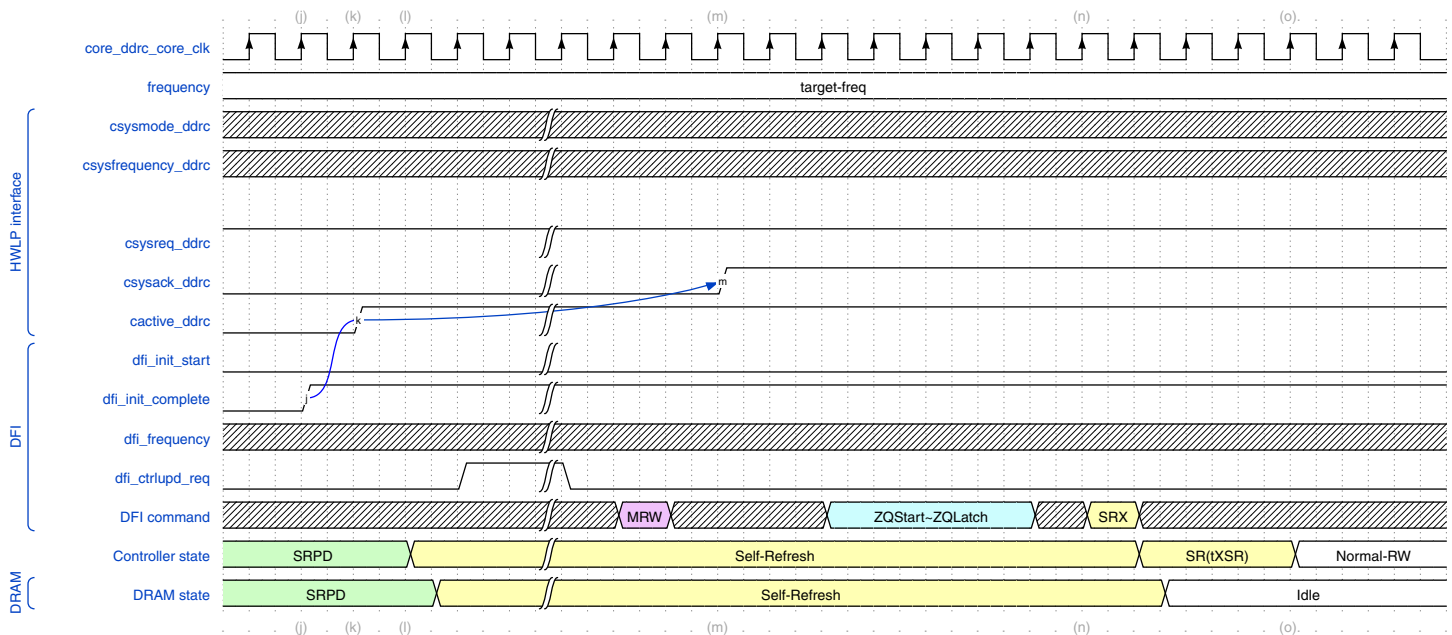


Figure 11-15 Variation After Frequency Change When ZQCTL2.dis_srx_zqcl=0, DFIUPD0.dis_auto_ctrlupd_srx=0, DFIUPD0.ctrlupd_pre_srx=0

ZQCTL2.dis_srx_zqcl=0, DFIUPD0.dis_auto_ctrlupd_srx=0, DFIUPD0.ctrlupd_pre_srx=0

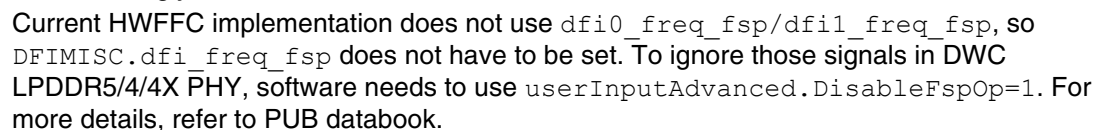


ZQCTL2.dis_srx_zqcl=0, DFIUPD0.dis_auto_ctrlupd_srx=0, DFIUPD0.ctrlupd_pre_srx=1



- FSP-WR(MR13 OP[6]) determines which frequency-set-point registers are accessed with MRW commands
- FSP-OP(MR13 OP[7]) determines which frequency-set-point register values are currently used to specify device operation

The DDRCTL uses HWFFCTL.init_fsp value as an initial value when the first HWFFC is performed to determine which FSP is currently being used in SDRAM.



11.6.3.1 MRW Commands Before Frequency Change

There are a few options on how to program LPDDR4 SDRAM as shown in the [Table 11-3](#) on page 247. MR13 is programmed twice before frequency change. The first one is to set FSP-WR, and the second one is to set VRCG.

Table 11-3 Programming LPDDR4 SDRAM

Step	MRW	MR Field	Corresponding Register Fields to be Referred	Notes
1	MR13	OP[7] FSP-OP	HWFFCCTL.init_fsp	The same value is written to all ranks.
		OP[6] FSP-WR	~HWFFCCTL.init_fsp	
		OP[3] VRCG	HWFFCCTL.init_vrcg	
		Other fields	INITMR1.emr3	
2	MR1	-	INITMR0.mr	The same value is written to all ranks.
3	MR2	-	INITMR0.emr	The same value is written to all ranks.
4	MR3	-	INITMR1.emr2	The same value is written to all ranks.
5	MR11	-	INITMR2.mr4	The same value is written to all ranks.
6	MR12	-	INITMR2.mr5	The same value is written to all ranks.
7	MR14	-	INITMR3.mr6	The same value is written to all ranks.
8	MR22	-	INITMR3.mr22	Unlike LPDDR4 SDRAM, MR22 in LPDDR4X SDRAM defines whether or not ODT is enabled. HWFFC writes the same value (INITMR3.mr22) to MR22 for all ranks. Different ranks must have different ODT settings, so HWFFC cannot be used for multiple ranks system of LPDDR4X.
9	MR13	OP[7] FSP-OP	Inverted version of what was issued in the step 1	Corresponding wait time, which is programmed with DRAMSET1TMG17.t_vrcg_enable, is inserted to ensure tVRCG_ENABLE after this MRW is issued. That wait time is necessary if VRCG has been changed from '0' to '1' when FSP-OP is changed.
		OP[6] FSP-WR	Same as what was issued in the step 1	
		OP[3] VRCG	Always set to '1'	



Note

If HWFFCCTL.skip_mr_w_odtvref is set to '1', step 5-8 (MR11, MR12, MR14, MR22) is skipped.

11.6.3.2 MRW Command After the Frequency Change

MR13 is programmed in order to set VRCG to '0' if HWFFCCTL.target_vrcg is 0.

Table 11-4 Programming LPDDR4 SDRAM (Applicable only when HWFFCCTL.target_vrcg is 0)

Step	MRW	MR Field	Corresponding Register Fields to be Referred	Notes
10	MR13	OP[7] FSP-OP	Same as what was issued in step 9	Corresponding wait time, which is programmed with DRAMSET1TMG17.t_vrcg_enable, is inserted to ensure tVRCG_DISABLE after this MRW is issued.
		OP[6] FSP-WR	Same as what was issued in step 9	
		OP[3] VRCG	Always set to '0'	

11.6.3.3 Simplified Timing Diagrams Focusing on MR13 (VRCG)

Figure 11-17 to Figure 11-18 show how DDRCTL programs MR13.OP[3] VRCG according to HWFFCCTL register fields.

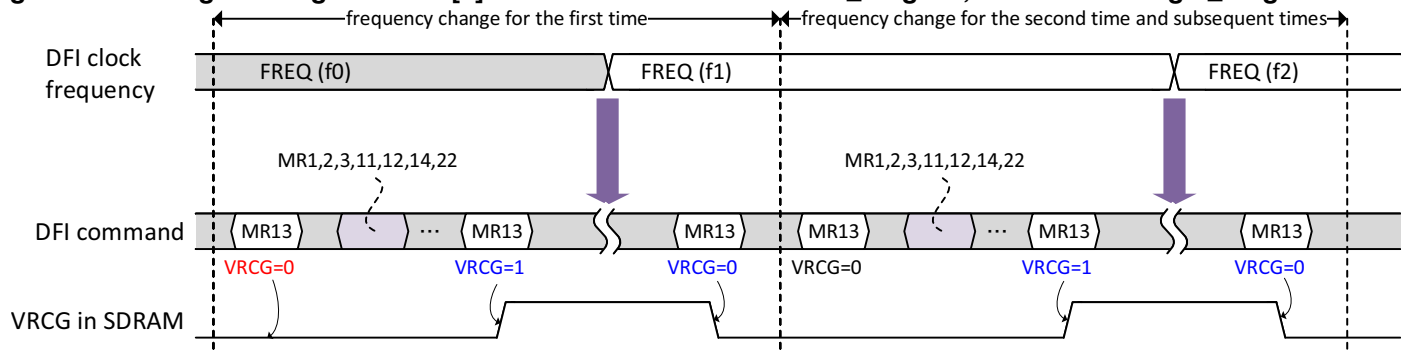
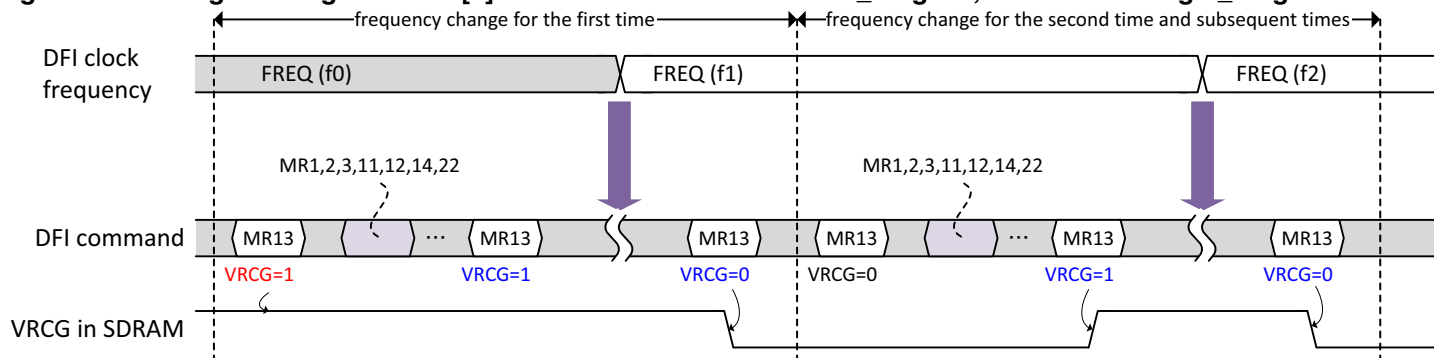
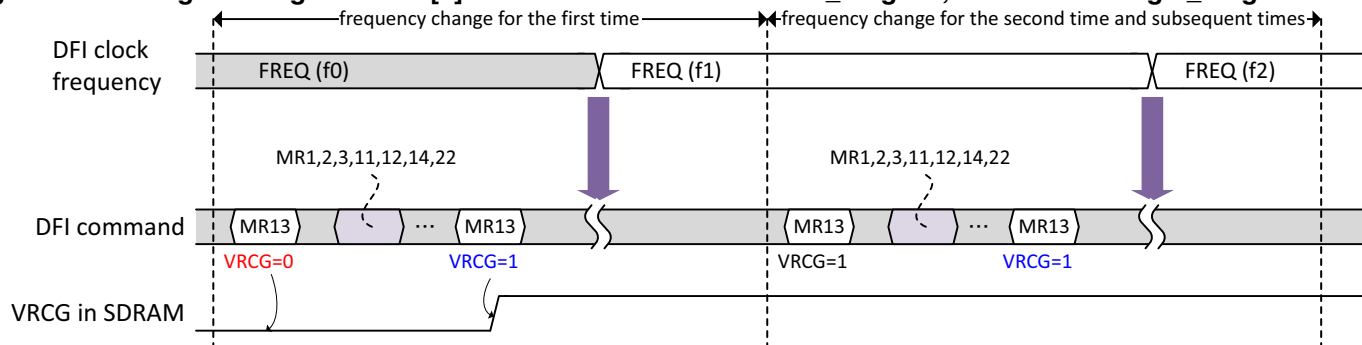
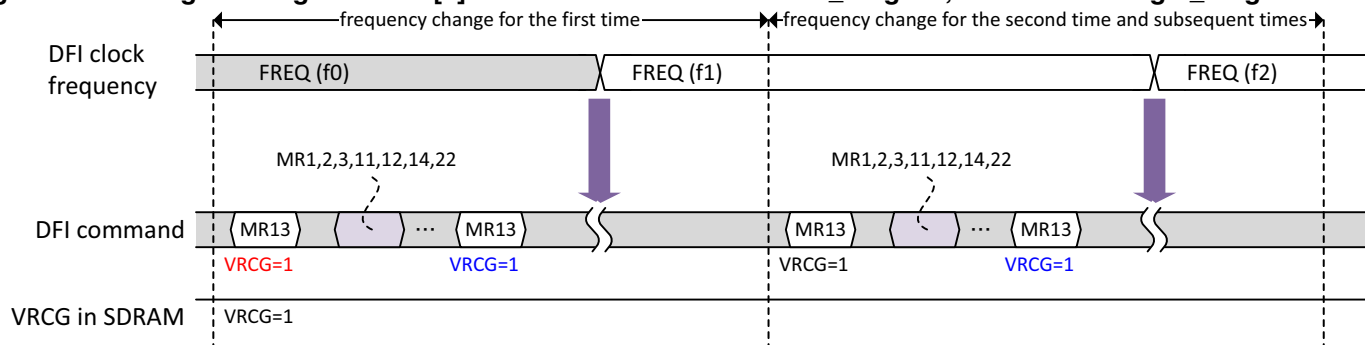
Figure 11-17 Programming MR13.OP[3] VRCG when HWFFCCTL.init_vrcg = 0, HWFFCCTL.target_vrcg = 0**Figure 11-18 Programming MR13.OP[3] VRCG when HWFFCCTL.init_vrcg = 1, HWFFCCTL.target_vrcg = 0**

Figure 11-19 Programming MR13.OP[3] VRCG when HWFFCCTL.init_vrcg = 0, HWFFCCTL.target_vrcg = 1**Figure 11-20 Programming MR13.OP[3] VRCG when HWFFCCTL.init_vrcg = 1, HWFFCCTL.target_vrcg = 1**

11.7 Low-Power Optimized Write Data for LPDDR5/4 Masked Write

The LPDDR5/4 JEDEC specification indicates that the LPDDR5/4 device masks the Write data received on the DQ inputs if the total count of '1' data bits on DQ[2:7] or DQ[10:15] (for lower or upper byte, respectively) is equal to or greater than five, and DMI signal is LOW.

A DQ in the LPDDR5/4 SDRAM consumes less power when the logic level of DQ is '0' than when it is '1'. If DFIMISC.lp_optimized_write is set to '1' and the LPDDR5/4 SDRAM device is used, dfi_wrdata per byte lane in Masked Write with enabling Write DBI is shown in Table 11-5 appears on the DFI bus. That is, setting DFIMISC.lp_optimized_write to '1' can reduce power consumption in DQ.

Table 11-5 Write DBI on DFI Bus

DFIMISC.lp_optimized_write	dfi_wrdata Per Byte Lane in Masked Write with Enabling Write DBI	Notes
0	8'b1111_1111	
1	8'b1111_1000	All the following conditions must be met: <ul style="list-style-type: none"> ■ DBICTL.wr_dbi_en = 1 ■ DBICTL.dm_en = 1 ■ DBICTL.phy_dbi_mode = 0

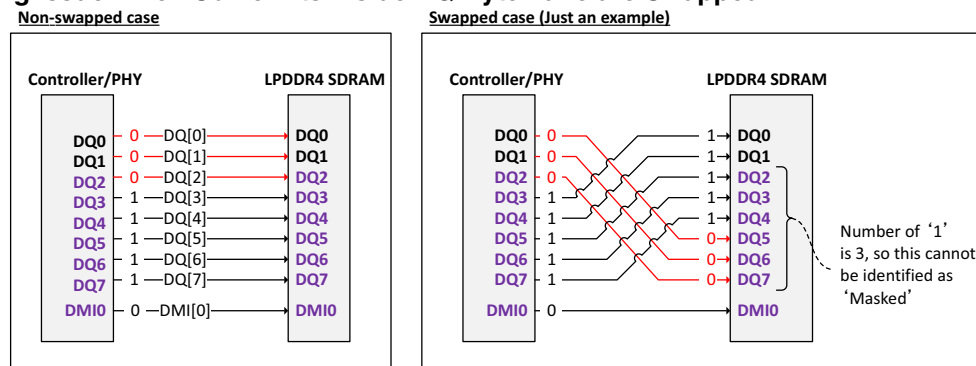


Note

This feature requires a straight-through DQ wiring between controller/PHY and SDRAM.

For example, it is possible that DQ[7:0]=8'b1111_1000 can be mapped as 8'b0001_1111 at SDRAM as shown in Figure 11-21 on page 250. If this happens because of the DQ wiring, then it cannot be identified as 'Masked' by the SDRAM. To optimize power consumption in DQs, a straight-through connection between controller/PHY and SDRAM is necessary. Otherwise, DFIMISC.lp_optimized_write must be set to '0'.

Figure 11-21 Wiring Issue When Some Bits Inside DQ Byte Lane are Swapped



11.8 BSM Clock Removal

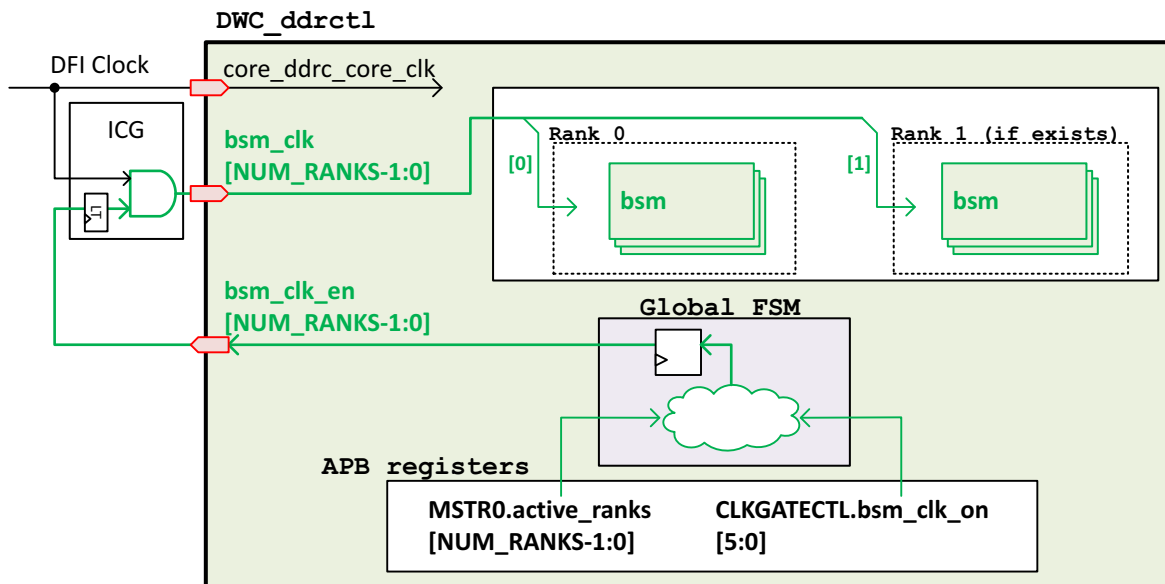
While SDRAM is in power saving mode such as DSM and SRPD, DDRCTL internal logic BSM (Bank State Machine), which manages state for each bank in SDRAM, can also save the power through removing BSM clock.

DDRCTL has two BSM clock related ports `bsm_clk` and `bsm_clk_en` which width is `NUM_RANKS`.

Regardless whether this feature is enabled or disabled, `bsm_clk` must be driven by the clock which is in the same domain as `core_ddrc_core_clk`. When BSM clock removal is enabled, `bsm_clk_en` starts to indicate when `bsm_clk` can be gated, that is, `bsm_clk[i]` can be gated only when `bsm_clk_en[i]` is 0, where 'i' represents BSM group for SDRAM rank i.

Figure 11-22 shows how to configure the behavior of `bsm_clk_en` and how to drive `bsm_clk`.

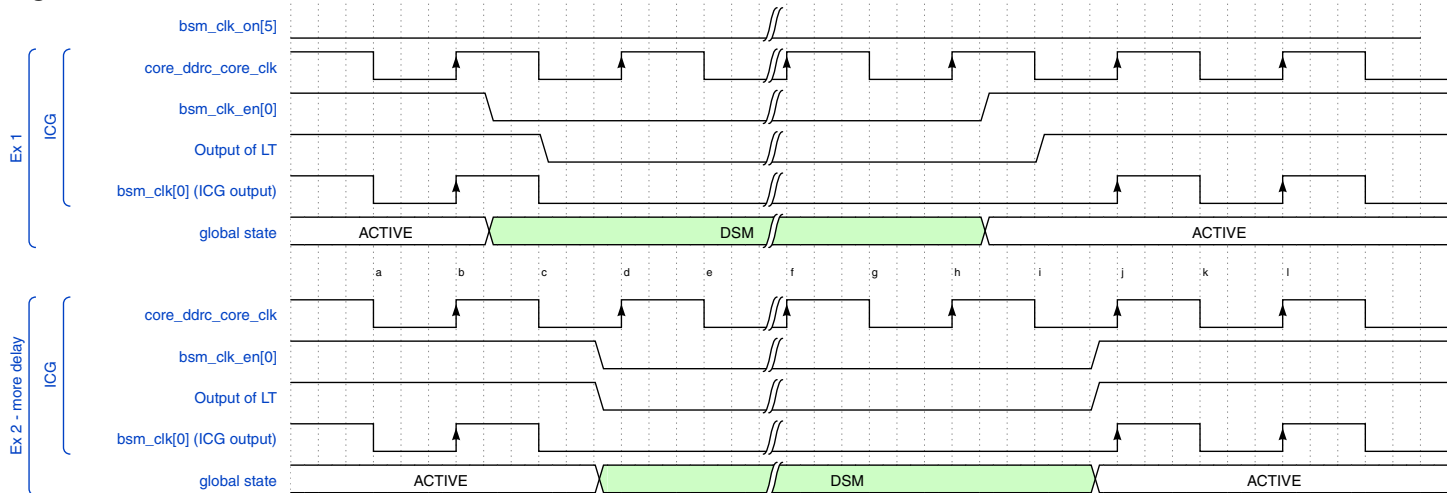
Figure 11-22 Hardware and Software Configuration to Gate BSM Clock



The register `CLKGATECTL.bsm_clk_on` represents how aggressively the BSM clock is removed. For example, setting `1'b0` to `CLKGATECTL.bsm_clk_on[1]`, which is for Deep Sleep Mode control, lets `bsm_clk_en[*]` to be de-asserted while SDRAM is in DSM state. For another example, if `CLKGATECTL.bsm_clk_on[0]` for Unpopulated rank control is `1'b0`, `bsm_clk_en[i]` is always 0, where 'i' is an unused rank according to `MSTR0.active_ranks`.

`CLKGATECTL.bsm_clk_on` is `6'b11_1111` by default, therefore `bsm_clk_en[*]` is always 1 if not configured.

To maximize power saving, this field needs to be set to `6'b00_0000` on initialization. For more information, see DWC LPDDR5/4/4X Memory Controller Programming Guide.

Figure 11-23 ICG Behavior with BSM Clock Removal

If BSM clock removal is enabled, bsm_clk might be driven by ICG (Integrated Clock Gating) cell which takes core_ddrc_core_clk and bsm_clk_en as inputs, as shown in Figure 11-22. Figure 11-23 shows how ICG behaves. When the SDRAM state transits to idle states from other active states, bsm_clk_en[*] is de-asserted. Once ICG accepts¹ bsm_clk_en de-assertion, it starts to drive bsm_clk low. Then, ICG resumes bsm_clk after bsm_clk_en assertion is accepted.

1. See the period c-d and i-j at 'Output of LT'. A latch in ICG accepts bsm_clk_en toggle only when core_ddrc_core_clk is low.

11.9 Signals Related to Power Saving

For the power saving features, `STAT.operating_mode` signal can be used to monitor the current operating mode of the DDRCTL.

The following signals are related to:

Hardware low-power interface of the DDRC:

- `csysreq_ddrc`
- `csysack_ddrc`
- `cactive_n` (input pin to the DDRC with one bit per port)
- `cactive_ddrc`

Hardware low-power interface of the AXI ports:

- `csysreq_n`
- `csysack_n`
- `cactive_n`

For more information about these signals, see [“Signal Descriptions”](#) on page 373.

12

LPDDR5 Specific Features

This chapter contains the following sections:

- [“Overview of LPDDR5 Specific Features”](#) on page 256
- [“Bank Organization”](#) on page 257
- [“WCK Clocking”](#) on page 259
- [“Link ECC”](#) on page 261

12.1 Overview of LPDDR5 Specific Features

The following are unique features of the LPDDR5 configurations:

- Bank organization:
 - LPDDR5 SDRAM supports three bank architectures which are selected by mode register.
 - The maximum data rate depends on the bank architecture.
- WCK clocking:
 - LPDDR5 SDRAM needs two clocks for command/address (CK, single data rate) and DQ (WCK, double data rate).
 - CAS with WCK sync command is necessary before READ, WRITE, MRR if DRAM is not synchronized with WCK.
- Command encoding:
 - LPDDR5 SDRAM does not have CKE pin.
 - ACTIVATE command is composed of two commands, ACT1 and ACT2. ACT2 command can be separated from ACT1 command within tAAD.
- Link ECC:
 - LPDDR5 SDRAM supports SEC/DED ECC for Link Protection.
 - Link ECC is an optional feature.

12.2 Bank Organization

LPDDR5 SDRAM supports three bank architectures to provide optimal access methods for varied system configurations. The native burst length determined by data prefetch size depends on which bank architecture is enabled.

Each architecture name is abbreviated as follows:

BG Mode = 4 banks, 4 bank groups

8B Mode = 8 banks, no bank groups

16B Mode = 16 banks, no bank groups

The supported operation data rate for each Bank/Bank Group Organization is as below.

BG Mode for more than 3200Mbps (>3200Mbps).

8B Mode for all data rate range.

16B Mode for equal or less than 3200Mbps (<=3200Mbps).

The bank architecture is selected by `DRAMSET1TMG24.bank_org`. This mode register is replicated for each frequency.

The BG and 16B modes support burst lengths of 16 or 32, while 8B mode supports only burst length 32.

The DRAM array mapping in these three modes is shown in [Table 12-1](#).

Table 12-1 Address Array Mapping

Bank Architecture	BG	BA0	BA1	BG0	BG1
	8B	BA0	BA1	BA2	B4
	16B	BA0	BA1	BA2	BA3



Note

- BA0-3: Bank Address, BG0-1: Bank Group address, B4 Burst Starting Address.
- The DDRCTL does not support 8B mode.

12.2.1 Address Mapping

LPDDR5 SDRAM introduces Burst addresses which indicates starting address within the burst. Since this address is same as lower column address for previous protocols, burst address is mapped by `addrmap_col_b3` register.



Note

- Burst address [2:0] is mapped to HIF address [2:0].
- HIF address [2:0] must be 3'b000.

For consistency between 16 bank mode and BG mode, the `addrmap_bg` register is used for BA2, BA3 for 16B mode and BG0, BG1 for BG mode.

Table 12-2 Address Mapping for BG Mode and 16B Mode

Register	BG mode	16B mode
ADDRMAP6.addrmap_col_b3	B3	B3
ADDRMAP6.addrmap_col_b4	C0	C0
ADDRMAP6.addrmap_col_b5	C1	C1
ADDRMAP6.addrmap_col_b6	C2	C2
ADDRMAP5.addrmap_col_b7	C3	C3
ADDRMAP5.addrmap_col_b8	C4	C4
ADDRMAP5.addrmap_col_b9	C5	C5
ADDRMAP3.addrmap_bank_b0	BA0	BA0
ADDRMAP3.addrmap_bank_b1	BA1	BA1
ADDRMAP4.addrmap_bg_b0	BG0	BA2
ADDRMAP4.addrmap_bg_b1	BG1	BA3

12.2.2 Burst Length

The Read/Write command behavior depends on the bank architecture. Each mode supports the following burst length.

BG mode: BL16, BL32 (interleaved)

8B mode: BL32

16B mode: BL16, BL32



Note

The DDRCTL only supports BL16.

12.3 WCK Clocking

The LPDDR5 command and address interface operates from a differential clock (CK_t and CK_c). Commands and addresses are registered single data rate (SDR) at every rising edge of CK.

LPDDR5 uses a DDR data interface. The data interface uses two differential forwarded clocks (WCK_t/WCK_c) that are source synchronous to the DQs.

WCK is used to sample DQ data for write operation and toggle DQ data for read operation. WCK must start toggle before starting write or read DQ data burst. Any commands that require DQ data burst initiate WCK2CK auto-sync sequence in LPDDR5 SDRAM.

WCK is managed by the DDRCTL using the following registers:

```
DFITMG4.dfi_twck_en_rd
DFITMG4.dfi_twck_en_wr
DFITMG4.dfi_twck_dis
DFITMG5.dfi_twck_fast_toggle
DFITMG5.dfi_twck_toggle
DFITMG5.dfi_twck_toggle_cs
DFITMG5.dfi_twck_toggle_post
DFITMG6.dfi_twck_toggle_rd
DFITMG6.dfi_twck_toggle_wr
```

The values of these registers are provided by the PHY. Refer to relevant PHY databook for more information.

LPDDR5 SDRAM utilizes two types of clock with different frequency. The frequency of WCK is four times or twice higher than the command clock.

The supported data rate for CK:WCK==1:2 mode is up to 3200Mbps.

Table 12-3 Supported CK:WCK Frequency Ratio

CK:WCK Ratio	Bank Organization	Data Rate	WCK Frequency	CK Frequency
1:2	16B mode	<= 3200 Mbps	<= 1600 MHz	<= 800 MHz
1:4	16B mode	<= 3200 Mbps	<= 1600 MHz	<= 400 MHz
	BG mode	> 3200 Mbps	>1600 MHz	> 400 MHz

The CK:WCK frequency ratio is set by `TMGCFG.frequency_ratio`.

With respect to controller clock (that is, DFI clock), the clocking relationship can be also summarized as DFI:CK:WCK which corresponds to frequency ratios of 1:1:2 and 1:1:4.

12.3.1 WCK Behavior

The controller supports two WCK modes:

- MSTR4.wck_on = 0: WCK always ON mode disabled (WCK on demand mode)
- MSTR4.wck_on = 1: WCK always ON mode enabled (WCK always on mode)

In WCK on demand mode, the controller issues CAS-WS_RD command before READ/MRR or CAS-WS_WR command before WRITE if necessary, and the controller starts to toggle WCK. When the toggling period is over, the controller stops WCK toggling. In multi-rank system, the controller issues CAS-WS command to one rank only. More than one rank cannot be in WCK2CK synchronization state simultaneously.

In WCK always on mode, the controller issues CAS-WS_RD command before READ/MRR or CAS-WS_WR command before WRITE at single rank configuration and issues CAS-WS_FS command before READ/WRITE/MRR at multi rank configuration, and the controller starts to toggle WCK. WCK is toggling until a power-down, self-refresh power-down, deep sleep mode or CAS-WS_OFF command is issued. The controller issues CAS-WS_OFF command when the following are required:

- DFI Control update
- DFI PHY update
- DFI PHY master

And also the controller issues CAS-WS_OFF command when there are no on-going Read or Write WCK2CK SYNC operations and `MSTR4.ws_off_en` is set to '1'.

In a multi-rank configuration, the controller issues CAS-WS_FS to all ranks and all ranks are in WCK2CK synchronization state simultaneously.



Note

In a single-rank configuration (`MEMC_NUM_RANKS = 1` or `MSTR0.active_ranks = 0x1`), it is recommended to use WCK on demand mode (`MSTR4.wck_on = 0`).

12.3.2 Enhanced WCK Always On Mode

If LPDDR5 device supports an enhanced WCK Always On Mode by reading out `MR0 OP[2]=1`, the controller is able to issue a CAS-WCK_SUSPEND command to reduce some of internal WCK clock net power consumption inside LPDDR5 devices. CAS-WCK_SUSPEND is a standalone command and may be issued anytime unless it interrupts on-going Read or Write WCK2CK SYNC operation when WCK always on mode is enabled. WCK SUSPEND mode keeps requiring WCK toggling input and exits automatically by following Read or Write, or Mask Write commands without issuing any additional explicit command.

The controller issues a CAS-WCK_SUSPEND only when `MSTR4.wck_on = 1` and `MSTR4.wck_suspend_en = 1`.

If LPDDR5 device does not support an enhanced WCK Always On Mode (`MR0 OP[2]=0`), `MSTR4.wck_suspend_en` must be set to '0'.

12.4 Link ECC

This topic contains the following sections:

- [“Overview of Link ECC Support” on page 261](#)
- [“Enabling Link ECC” on page 261](#)
- [“Link ECC Restrictions” on page 261](#)
- [“Controller Behavior During Read Link ECC Errors” on page 261](#)
- [“Inline ECC and Link ECC Features Combination” on page 262](#)
- [“Link ECC Error Reporting” on page 264](#)
- [“Link ECC Data Poisoning” on page 265](#)
- [“Performance Impact” on page 266](#)
- [“Registers Related to Link ECC Support” on page 266](#)

12.4.1 Overview of Link ECC Support

The DDRCTL supports Link ECC feature for Read and Write. If supported by the DRAM, Link ECC may then be enabled or disabled by the software as required by the system configuration, operating speed, or other requirements. For Read, the DDRCTL detects single/double bit error and stored the error count which can be read via APB register. If Link ECC error is single bit error, the DDRCTL corrects the data and sends to the host.

12.4.2 Enabling Link ECC

To enable Link ECC, set the hardware configuration parameter MEMC_LINK_ECC to '1'. This feature can then be enabled/disabled by software using the `LNKECCCTL0.wr/rd_link_ecc_enable` register.

12.4.3 Link ECC Restrictions

The following restrictions apply to Link ECC feature:

- Read-DBI must be disabled
- CAS(B3) must be '0' at read command
- BL16 only
- 1:1:4 mode only
- Partial Read is not supported (`hif_cmd_length` must be set to 2'b00 (Full Read))

12.4.4 Controller Behavior During Read Link ECC Errors

When the DDRCTL detects a correctable Read Link-ECC error, it performs the following:

- Sends the corrected data to the host as part of the read data.
- Sends the Link ECC error information to the APB register.
- Asserts the interrupt signal.

When the DDRCTL detects an uncorrectable Read Link-ECC error, it does the following:

- Sends the data with error to the SoC as part of the read data.
- Sends the Link ECC error information to the APB register.

- Asserts the interrupt signal.
- Assert `hif_rdata_uncorr_linkecc_err`.
- Generates a SLVERR response on the AXI interface. Note, that in case of Inline ECC overhead (RE_B) command and Read part of RMW, DDRCTL does not assert SLVERR because they do not support SLVERR.

12.4.5 Inline ECC and Link ECC Features Combination

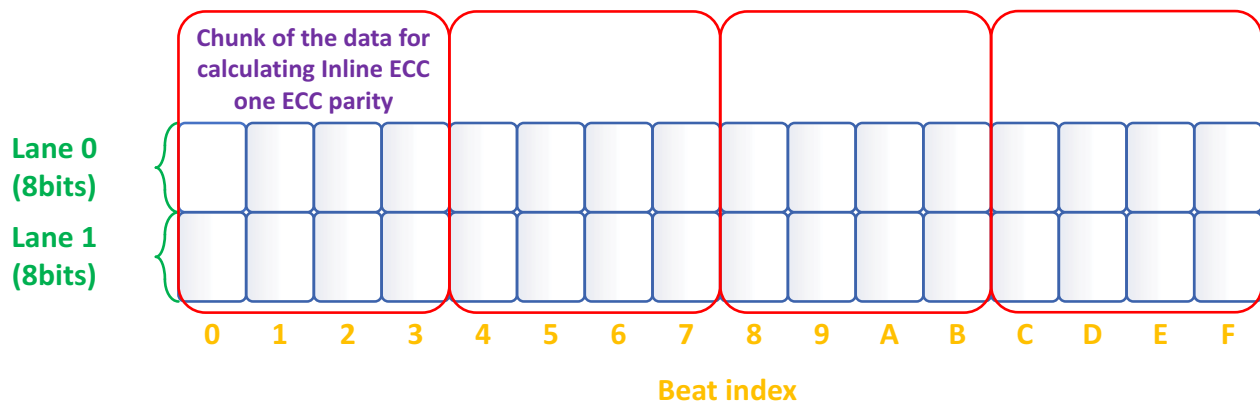
Read

When both Link ECC and Inline ECC are enabled, Link ECC check and correction is performed before Inline ECC check, correction, and detection. The Inline ECC error happens on the SDRAM devices. Link ECC parity is calculated by using SDRAM saving data even if it has already broken. This means Link ECC feature cannot detect the Inline ECC error.

If 2-bits Link ECC error happens, it cannot be corrected and this uncorrected data is propagated to Inline ECC checking mechanism.

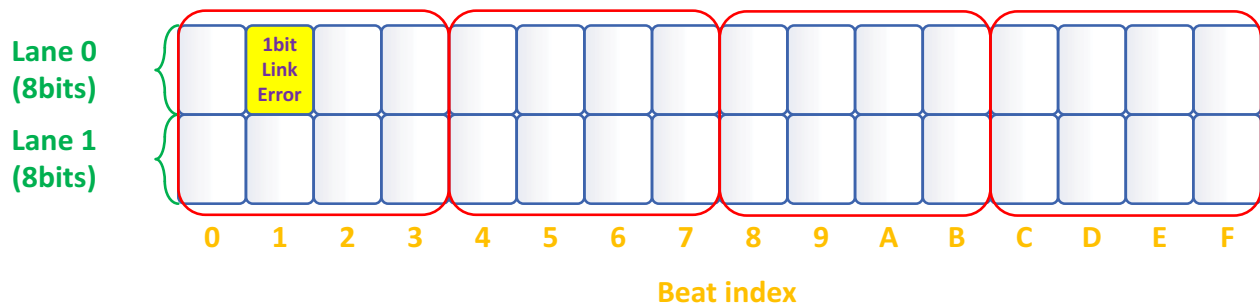
Inline ECC generates ECC parity per 8-byte data. If `MEMC_DRAM_DATA_WIDTH = 16`, 1-byte ECC parity is generated by 4-beats data.

Figure 12-1 Link ECC and Inline ECC Data Chunk Structure



If 1-bit Link ECC error happens, this error can be corrected by Link ECC parity. Then, the Inline ECC mechanism does not detect any errors.

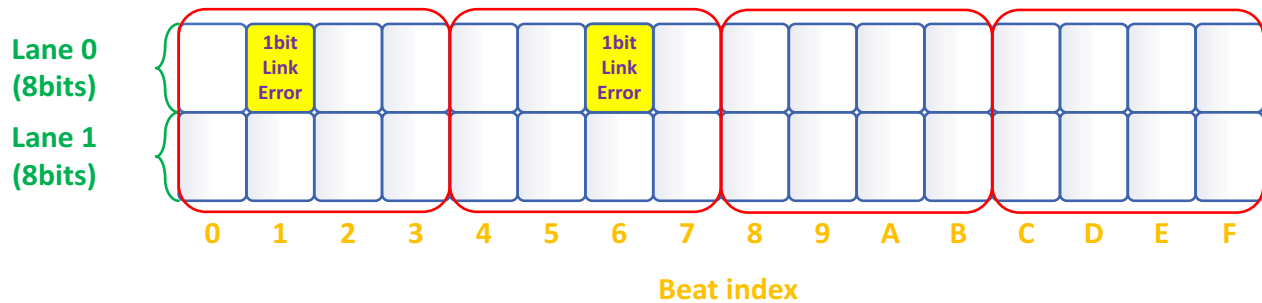
Figure 12-2 Link ECC 1-Bit Error



It is possible that Inline ECC can correct data where Link ECC indicates the uncorrected error, if the reason of Link ECC error is on the different 1-bit error on the chunk of 4-beats data. In the following case, Inline ECC judges it as 2 correctable errors. Therefore, this error is corrected by Inline ECC. Even though the Link

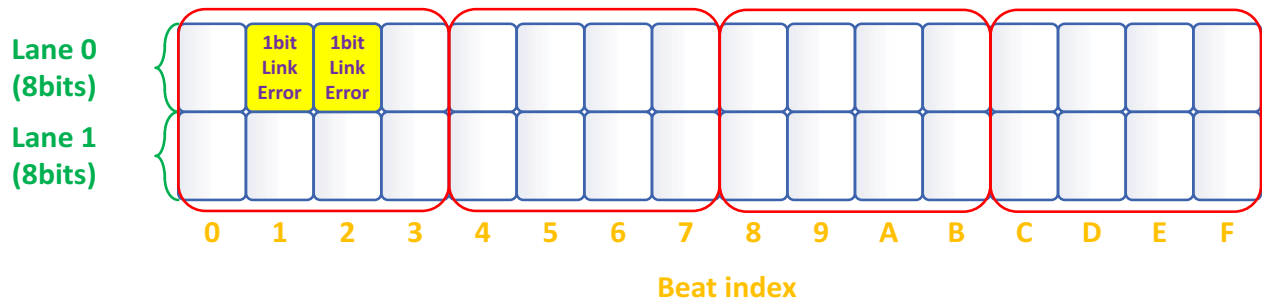
ECC error is corrected by Inline ECC, the RRESP for this data informs as SLVERR. If this type of Link ECC error happens with the read part of RMW, the corrected data is sent to SDRAM.

Figure 12-3 2-bits Link ECC Error In The Different Inline ECC Data Chunk



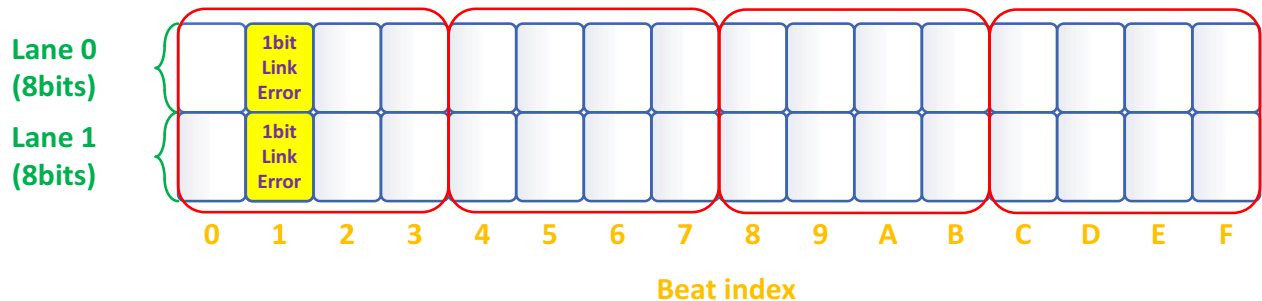
If 2-bits Link ECC error happens in the same beat as shown in [Figure 12-4](#), Inline ECC judges as uncorrected error. If this type of error happens on the read part of RMW command, the data including 2-bits error is sent to SDRAM.

Figure 12-4 2-bits Link ECC Error In The Same Inline ECC Data Chunk

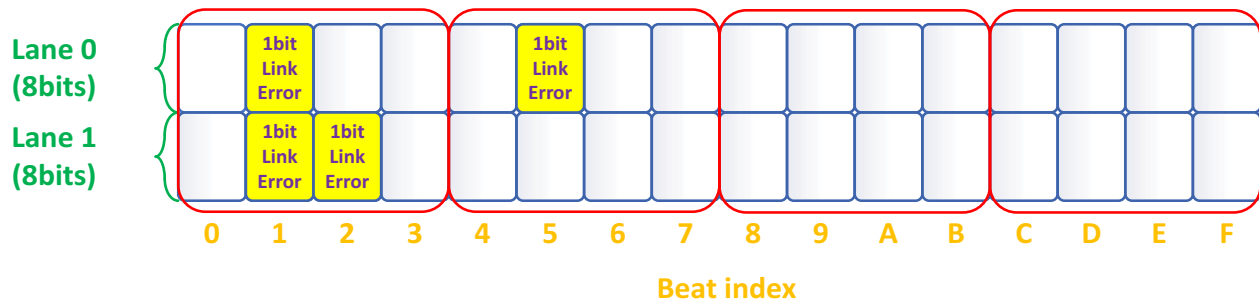


The Link ECC parity is generated for each data lane. In the following case, it is 1-bit error for each data lane from the perspective of Link ECC. This error can be fixed by Link ECC.

Figure 12-5 2-bits Link ECC Error In The Same Inline ECC Data Chunk And Different Data Lane



2-bits error for each data lane can be propagated to Inline ECC. If the link ECC 2-bits error happens on the same Inline ECC data chunk, it becomes 3-bits or 4-bits error. However, the Inline ECC feature does not detect an error greater than 2 bits. So, in this case, errors cannot be detected correctly.

Figure 12-6 2-bits Link ECC Error In The Different Data Lane**Write**

Link ECC works without distinguishing a common write data and Inline ECC data. There is no difference in the behavior depending on whether Inline ECC is enabled or not.

Poisoning

Link ECC poisoning feature poisons a DMI bit. If Link ECC poisoning feature is used, Inline ECC mechanism cannot detect any errors.

12.4.6 Link ECC Error Reporting

This section has the following sub-sections:

- [“Write Link ECC Error”](#) on page 264
- [“Read Link ECC Error”](#) on page 264

12.4.6.1 Write Link ECC Error

A counter of Single-Bit-Errors and a Double-Bit-Error flag will be maintained on the DRAM, both of which can be read through mode register MR43. The DRAM will also store both syndromes from the most recent single-bit ECC error in MR44 & MR45. These Mode Registers can be read by software via MRCTRL register to check Link ECC error status.

For more information about MRR operation, see the [“Mode Register Reads and Writes”](#) on page 212 section.

**Note**

- According to JEDEC LPDDR5 specification, these registers are cleared on Power-Down exit. It is recommended to disable automatic Power-Down/Self-Refresh entry feature.
- When performing MRR command via MRCTRL register, it is recommended to have at least 32 pclk cycles between two consecutive writes to the MRCTRL register to ensure the timing constraint “tMRR + tMRW”.

12.4.6.2 Read Link ECC Error

For Link ECC configuration, there are several indirect registers as in [Table 12-4](#). The selectors of the indirect registers are defined in LNKECCINDEX register.

**Note**

The term "indirect register" means the register is internally replicated (like a register array), and addressed through setting the register's selector to access the required one.

Table 12-4 Link-ECC Indirect Registers

Register Name	Depth	Selector (Fields of LNKECCINDEX)
LNKECCERRCNT0.rd_link_ecc_uncorr_cnt	MEMC_NUM_RANKS* (MEMC_DRAM_DATA_WIDTH/8)	LNKECCINDEX.rd_link_ecc_err_byte_sel LNKECCINDEX.rd_link_ecc_err_rank_sel
LNKECCERRCNT0.rd_link_ecc_corr_cnt	MEMC_NUM_RANKS* (MEMC_DRAM_DATA_WIDTH/8)	
LNKECCERRCNT0.rd_link_ecc_err_syndrome	MEMC_NUM_RANKS* (MEMC_DRAM_DATA_WIDTH/8)	

LNKECCINDEX is quasi-dynamic registers. To write them, set SWCTL.sw_done to '0' at the beginning of the programming sequence and set back to '1' at the end. After that, poll SWSTAT.sw_done_ack for acknowledge.

To read LNKECCERRCNT0 register indirectly, follow the steps in [Table 12-5](#).

Table 12-5 Procedure to Read from the LNKECCERRCNT0 Register indirectly

Step	Operation	Description
1	Set SWCTL.sw_done to '0'	Enable quasi-dynamic register write
2	Write appropriate values to: <ul style="list-style-type: none"> ■ LNKECCINDEX.rd_link_ecc_err_byte_sel ■ LNKECCINDEX.rd_link_ecc_err_rank_sel 	Select target byte(s) of Data Select target Rank
3	Set SWCTL.sw_done to '1'	Complete the quasi-dynamic register write
4	Write '1' to SWSTAT.sw_done_ack	Wait for acknowledge
5	Read <ul style="list-style-type: none"> ■ LNKECCERRCNT0.rd_link_ecc_uncorr_cnt ■ LNKECCERRCNT0.rd_link_ecc_corr_cnt ■ LNKECCERRCNT0.rd_link_ecc_err_syndrome 	Read out the Link-ECC Syndrome, uncorrected error count and corrected error count.

12.4.7 Link ECC Data Poisoning

In Link ECC mode, the DDRCTL provides the ability to add errors in the read/write data burst. This data burst can then be used as a test of how the system handles uncorrectable, correctable Link ECC errors.

[Table 12-6](#) shows the software sequence for the Link ECC poisoning. One sequence injects one Link ECC error. Software can repeat the sequence to inject multiple Link ECC errors.

Table 12-6 Software Sequence for the Link ECC Poisoning

Step	Description	Comment
1	Write appropriate value to: <ul style="list-style-type: none"> ■ LNKECCPOISONCTL0.linkecc_poison_byte_sel ■ LNKECCPOISONCTL0.linkecc_poison_dmi_sel ■ LNKECCPOISONCTL0.linkecc_poison_type ■ LNKECCPOISONCTL0.linkecc_poison_rw 	Select target byte(s) of Data Select target DMI(s) Single-bit error or Double bit error Select Read or Write
2	Write '1' to LNKECCPOISONCTL0.linkecc_poison_inject_en	Enable Link-ECC poisoning. A Link-ECC error is injected to the first matched data/dmi.
3	Poll LNKECCPOISONSTAT.linkecc_poison_complete=1	Once a Link-ECC error is injected, the register becomes equal to '1'. This step can be skipped if software wants to cancel the error injection.
4	Write '0' to LNKECCPOISONCTL0.linkecc_poison_inject_en	Disable Link-ECC poisoning.
5	Poll LNKECCPOISONSTAT.linkecc_poison_complete=0	Make sure the Link-ECC poisoning is completed/terminated.

12.4.8 Performance Impact

Enabling the Link ECC feature (LNKECCCTL0.wr/rd_link_ecc_enable=1) will impact the performance:

- Some timing parameters in JEDEC LPDDR5 specification will increase. Certain timing registers need to be adjusted accordingly.
- Read Latency from DFI to HIF will increase by 2 core clock cycles.

12.4.9 Registers Related to Link ECC Support

The following are the registers related to link ECC support:

- LNKECCCTL0
- LNKECCCTL1
- LNKECCPOISONCTL0
- LNKECCPOISONSTAT
- LNKECCINDEX
- LNKECCERRCNT0
- LNKECCERRSTAT

For more information about these registers, see the “Register Descriptions” chapter of the [DWC DDRCTL LPDDR5/4 Programming Guide](#).

13

RAS Features

This chapter contains the following sections:

- [“Memory ECC” on page 268](#)
- [“Scrubber” on page 301](#)
- [“On-Chip Parity \(OCPAR\)” on page 309](#)
- [“On-Chip ECC \(OCECC\)” on page 316](#)
- [“Registers Parity Protection \(REGPAR\)” on page 322](#)
- [“On-Chip Command and Address Path Protection \(OCCAP\)” on page 327](#)
- [“On-Chip External SRAM Address Protection \(OCSAP\)” on page 335](#)

13.1 Memory ECC

This topic contains the following sections:

- [“Inline ECC Support”](#) on page 268

13.1.1 Inline ECC Support

This topic contains the following sections:

- [“Overview of Inline ECC Support”](#) on page 268
- [“Enabling Inline ECC”](#) on page 269
- [“Address Map”](#) on page 269
- [“Selectable Protected Regions”](#) on page 275
- [“Locking of ECC Region”](#) on page 278
- [“Error Injection Through Software \(ECC Data Poisoning\)”](#) on page 279
- [“Related Hardware Parameters”](#) on page 283
- [“Command Flow”](#) on page 284
- [“Address Protection with ECC \(ECCAP\)”](#) on page 292
- [“Scrubbing”](#) on page 293
- [“QoS”](#) on page 293
- [“Features and Limitations”](#) on page 294
- [“Interrupts Related to Inline ECC”](#) on page 296
- [“Signals Related to Inline ECC”](#) on page 296
- [“Registers Related to Inline ECC”](#) on page 298

13.1.1.1 Overview of Inline ECC Support

The following are the features of Inline ECC:

- ECC parity (code) is stored together with the data without using a dedicated sideband memory device.
- Inline ECC is a necessity for LPDDR4/5 due to device characteristics and device topology.
 - Inline ECC is supported with LPDDR4 and LPDDR5 (BG rotation address mapping) protocols.
- The supported memory data widths are 16 and 32.
- For Inline ECC, 64/8 SECEDED Hamming code is used:
 - Data to ECC ratio is 8/1.
- Inline ECC requires Data Mask (DM) to be enabled.
- RMW command is required when a write access cannot fill one Hamming code (64 bits), For details, see [“Read-Modify-Write \(RMW\) Generation”](#) on page 46.

13.1.1.2 Enabling Inline ECC

To enable Inline ECC, set the hardware configuration parameter `MEMC_INLINE_ECC` to '1'.

For details about this option, see [“HW Configuration / DDRC Parameters”](#) on page 401.

13.1.1.3 Address Map

Definitions:

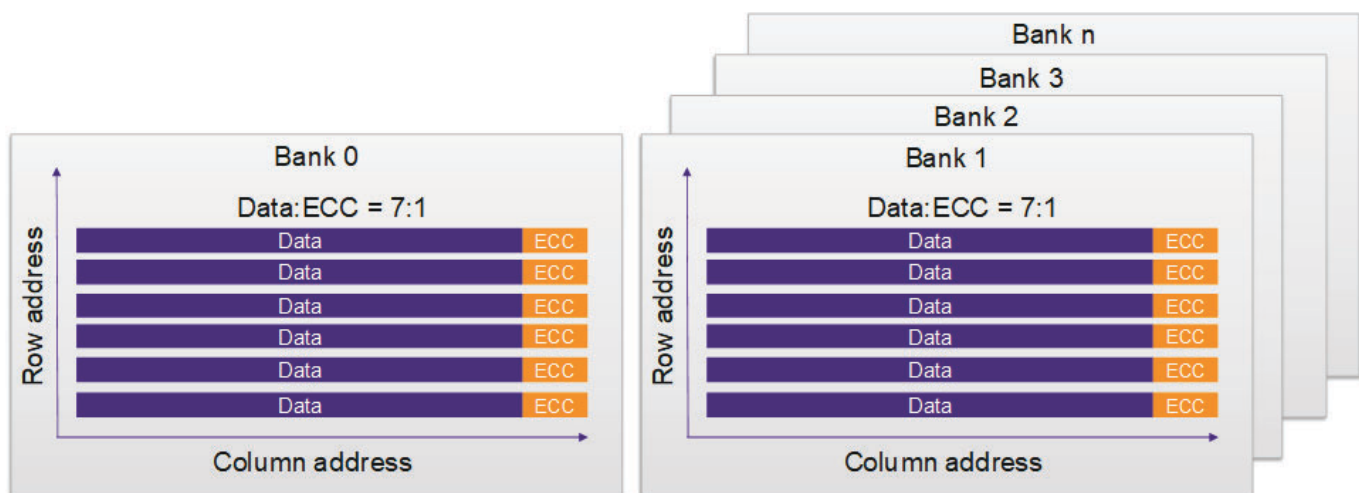
- Data Block: 8 burst accesses of data (a burst can be DRAM BL8 or BL16).
- Super-block: Block with ECC parity (9 burst accesses).
- Giant-block: Depending on address map, a transaction sequence can be divided into 8 data + 1 ECC, 16 data + 2 ECC, 32 data + 4 ECC (unit is one burst access).
- Hole: ECC parity + wasted region of the memory.
- Usable Memory Size: All the data blocks that fits.
- Usable Size Ratio: Usable memory size/Full memory size.
- Utilization Percentage: The percentage of memory used for Data and ECC.

When Inline ECC is enabled, the highest three column bits must be mapped to the highest address map position possible. The controller flexible address mapping scheme is constrained so that the highest system address space is reserved for ECC parity and waste as one region as shown in [Figure 13-1](#). For the normal data in all remaining regions, the system address is linear and continuous.

The reason for the waste is due to how the memory is divided for ease of implementation. Data to ECC ratio is 8 to 1 (divide by 9), but the memory is mapped 7 to 1 (divide by 8 which is a power of 2).

- Usable size ratio is $7/8 = 87.5\%$.
- Utilization percentage is $(7/8 + (1/8 * 7/8)) = 63/64 = 98.4\%$ and the remaining becomes the waste. The unprotected wasted area can still be accessible by the system, but not in linear and continuous fashion.

Figure 13-1 ECC Code Location – Upper 1/8 of Each Column Address



System to SDRAM Address Example

This is an example for LPDDR4 (16Mb x 16DQx 8 banks, single channel):

- Column width is 10bits. Number of columns = 1024 (0x400).
- Row width is 14bits, Number of rows = 16384 (0x4000).
- Bank width is 3bits, Number of banks = 8(0x8).
- Total SDRAM address = $1024 * 16384 * 8 = 128\text{M}$ words, DQ width is 16bits. Therefore, memory density per channel is 256M bytes.

Four instances of the above LPDDR4 device can be combined to give 64 bits of DRAM data width.

- `MEMC_DRAM_DATA_WIDTH` = 64 bits (four LPDDR4 X16 devices)¹
- `MEMC_BURST_LENGTH` = 16
- 1 page = 8192 bytes (1024 columns)
- 1 access (burst) is 128 bytes (BL16)
- 1 block = 8 accesses (1024 bytes)
- For each access, 16 bytes of ECC parity is required (1 column).

All ECC parity information for a full block (that is, 8 BL16 data accesses) is tightly packed into one ECC access (that is, 1 BL16).

Figure 13-2 shows the space usage of data and ECC code, it is a high-level description to explain what the result (where is data, where is ECC from both AXI address and SDRAM address perspective). The red zones are the wasted regions.

Figure 13-2 Inline ECC Address Translation



In the previous example:

1. {C9, C8, C7} (or highest three columns) must map to the MSB of system address.
2. No special address translation for data.
3. ECC address is in term of HIF address, that is generated by replacing `hif_addr[6:4]` (offset address in one block) with the highest three `hif_addr` (C9, C8, C7), then set the highest three `hif_addr` (C9, C8, C7)

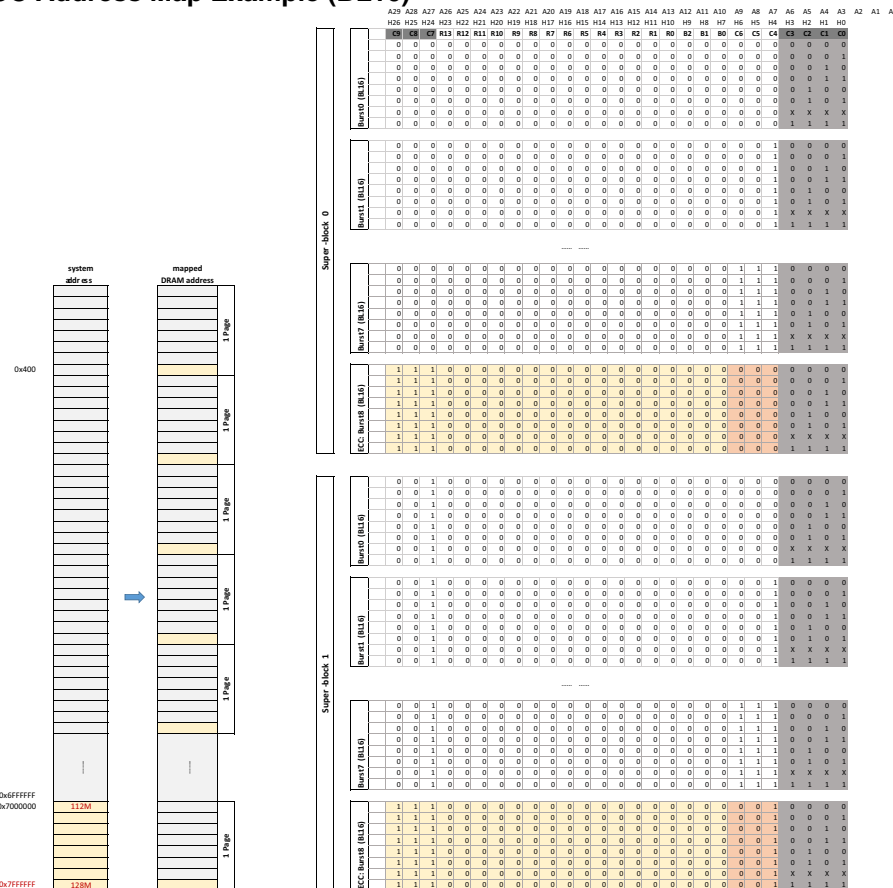
1. Currently, `MEMC_DRAM_DATA_WIDTH` must be set to '16' or '32' in LPDDR5/4/4X Controller, so this cannot be supported.

to 3'b111. Set the lowest 4 bit to 4'b0000. An example format is 3'b111, (R13...R0), (B2...B0), ({C9...C7}), 4'b0000.

4. BG/Bank address bits can be mapped anywhere between {C9, C8, C7} and {C3,...,C0} groups. It will affect Bank/Bank-Group rotation, for more information, refer to “Address Map Considerations for Inline ECC”. In this example, there is no Bank/Bank-Group rotation.
5. Rank/Row/Column address bits can be mapped with the limitations described in “Address Map Limitation”.

Figure 13-3 shows the detailed address mapping for system address (AXI) and SDRAM address in the example above. A* stands for system address, H* stands for HIF address. For more information, refer to “Application to HIF Address Mapping” on page 123.

Figure 13-3 Inline ECC Address Map Example (BL16)



Address Map Considerations for Inline ECC

The internal mapping of ECC blocks are such that data and corresponding ECC are always mapped to the same page (that is, same bank and row). This is required to achieve the highest scheduling efficiency for all protocols. Therefore, the address mapping decision for the banks and bank groups (the position of mapping) impact the way incoming sequential commands are divided into internal ECC blocks. This is done automatically by the controller. For example, if 8 (or 16) sequential commands are mapped to 2 banks, there will be 2 partial (full) ECC blocks with 2 ECC overhead comments.

To achieve good performance, when setting address mapping, you must consider whether you need banks or bank groups to rotate within 8 HIF transaction sequence, what is the size of one “transaction sequence”,

how many “transaction sequences” are interleaving. This helps to determine how many block channel resources are required.

Recommendation is to satisfy the following two conditions:

- Number of block channels $\geq (\text{Giant-block size}/8) \times (\text{number of interleaving sequences}) \times 2$
Where x2 is for read/write interleaving
- Giant-block size == (Number of ECC blocks belong to a Giant block) x 8.

If bank/bank-group is rotating within ECC block size (that is, 8 HIF commands), multiple ECC blocks are interleaved internally and then a Giant block has 2 or 4 ECC blocks, otherwise a Giant block has an ECC block.

For more details, see [Figure 13-4](#), [Figure 13-5](#), and [Figure 13-6](#), and [Table 13-1](#).

Figure 13-4 1 Giant Block = 1 ECC Block = 8 HIF Commands

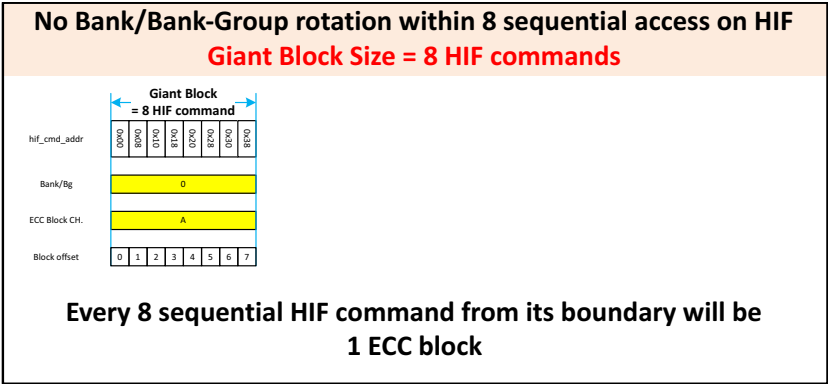


Figure 13-5 1 Giant Block = 2 ECC Blocks = 16 HIF Commands

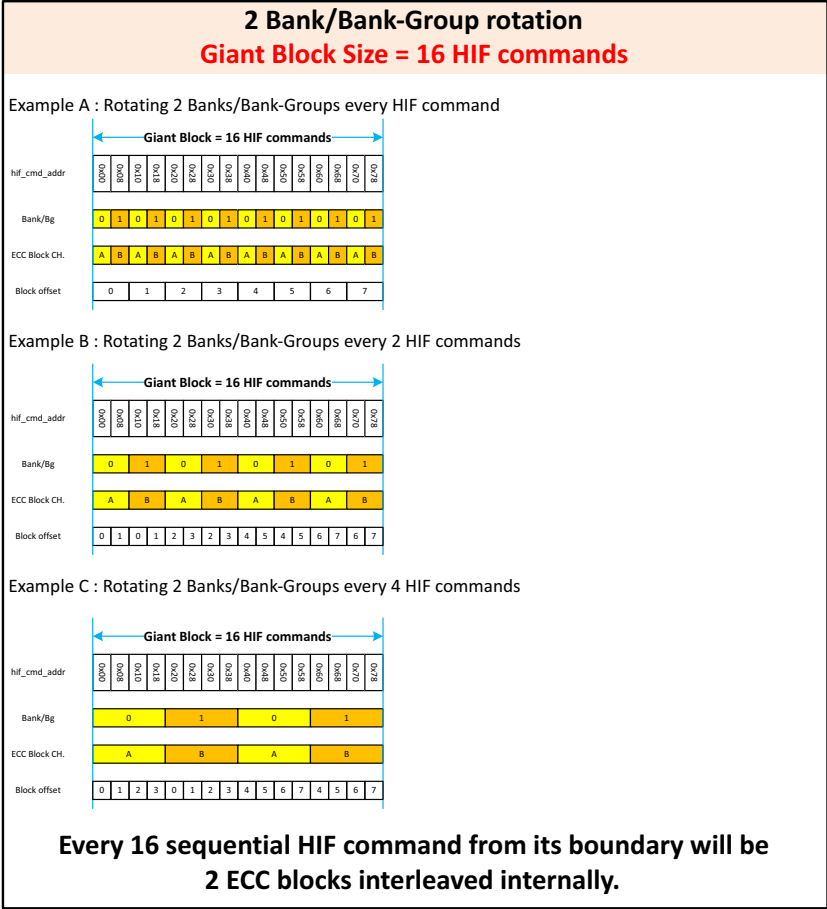
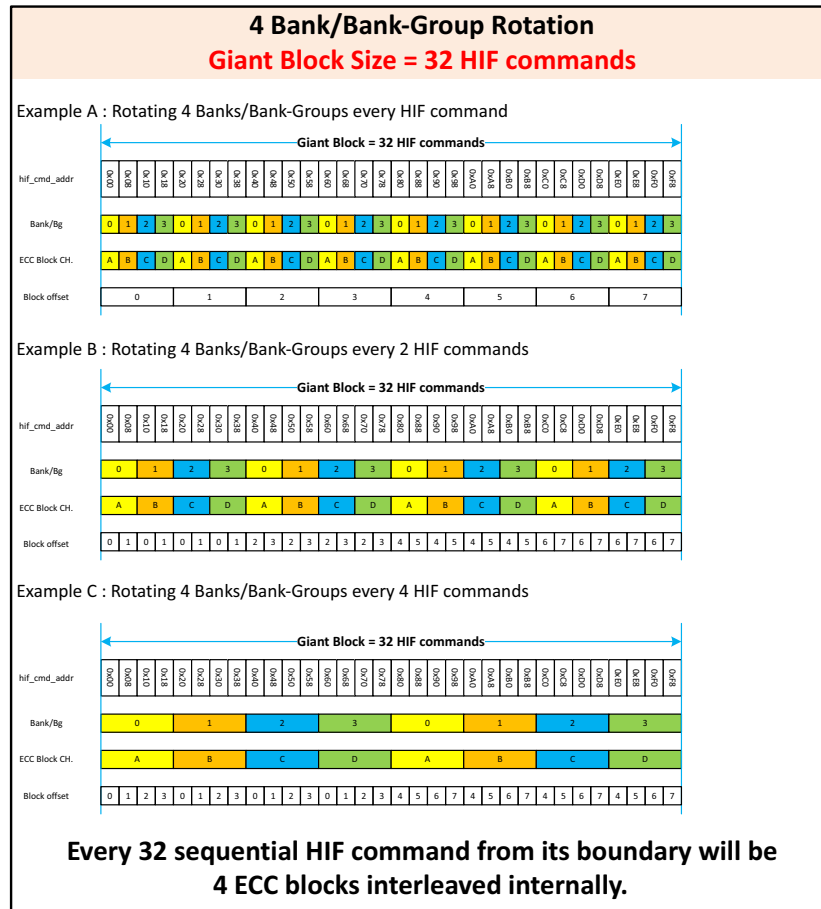


Figure 13-6 1 Giant block = 4 ECC blocks = 32 HIF Commands

- For MEMC_BURST_LENGTH = 16, burst_rdwr = BL16, LPDDR4 case.
 - For no bank rotation (within 8 HIF transaction sequence), COL[6:4] must be mapped to HIF[6:4]. In this case, one transaction sequence (size is 8 BL16) is 1 block access, which requires 1 block channel.
 - For 2 bank rotation, COL[6] must be mapped to HIF[7], COL[5:4] and BANK[0] can be mapped to anywhere within HIF[6:4] (C5, C4 relative ordering must be preserved). In this case, one transaction sequence (size is 8 BL16) is 2 interleaving block accesses, which require 2 block channels.
 - For 4 bank rotation, COL[6] must be mapped to HIF[8], COL[5:4] and BANK[1:0] can be mapped to anywhere within HIF[7:4] (C5, C4 relative ordering must be preserved). In this case, one transaction sequence (size is 8 BL16) is 4 interleaving block accesses, which require 4 block channels.
 - 8 bank rotation is not recommended. In that case, a transaction sequence (size is 8 BL16) causes 8 interleaving block accesses and consume 8 block channels. Following examples illustrate the need for many block channels which may not be available:
 - Example 1: You must issue small sized transaction sequence which generates more overhead commands and needs more block channels. If you do not have enough block channels, that generates more override channel condition.
 - Example 2: If you have more than one interleaving sequence, that needs more block channels. If you do not have enough block channels, that generates more override channel condition.

Table 13-1 Inline ECC Address Mapping

MEMC_BURST_LENGTH=16																							
LPDDR4	NO Bank rotation																						
				H0	H1	H2	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
				C9	C8	C7	R	R	R	R	BA2	BA1	BA0	C6	C5	C4	C3	C2	C1	C0			
	2 Bank rotation																						
				H0	H1	H2	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
				C9	C8	C7	R	R	R	R	BA2	BA1	BA0	C6	C5	C4	BA0	C3	C2	C1	C0		
	4 Bank rotation																						
				H0	H1	H2	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
LPDDR5	8 Bank rotation (don't need, not realistic)																						
				H0	H1	H2	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
				C9	C8	C7	R	R	R	R	C6	C5	C4	BA2	BA1	BA0	C3	C2	C1	C0			
	NO BG rotation																						
				H0	H1	H2	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
				C9	C8	C7	R	R	R	BA1	BA0	BG1	BG0	C6	C5	C4	C3	C2	C1	C0			
	2 BG rotation																						
				H0	H1	H2	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
LPDDR5				C9	C8	C7	R	R	R	BA1	BA0	BG1	C6	C5	C4	BG0	C3	C2	C1	C0			
	4 BG rotation																						
				H0	H1	H2	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
				C9	C8	C7	R	R	R	BA1	BA0	C6	C5	C4	BG1	BG0	C3	C2	C1	C0			
	8 Bank rotation (don't need, not realistic)																						
				H0	H1	H2	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
				C9	C8	C7	R	R	R	BA1	C6	C5	C4	BA0	BG1	BG0	C3	C2	C1	C0			
				C9	C8	C7	R	R	R	BA1	C6	C5	C4	BA0	BG1	BG0	C3	C2	C1	C0			

The addresses within highlighted box can be mapped in any combination with the the following rules:
 *C6 must be highest
 *C6, C5, C4 relative ordering must be preserved
 C6:4 is block offset address
 C3:0 is start address for one burst
 The others are block address

ECC address will be calculated by DRAM address as following:
 Col[Highest -: 3] <= 3'b111
 Col[6:4] <= Col[Highest -: 3]
 Col[3:0] <= 4'b0000;
 Remaining addresses are pass-through

H0, H1, H2 : highest addressable HIF based on DRAM size

13.1.1.4 Selectable Protected Regions

Enable ECC protection on the critical area of the memory. Remaining addresses can be accessed with no performance overhead.

- To maintain uninterrupted linear system address space, ECC region maps to the top of the system address, ECC region size is 1/8 memory size.
- System address space is divided into 8/16/32/64 regions. The granularity is determined by register `ECCCFG0.ecc_region_map_granu[1:0]`. Note, that highest 1/8 region is always ECC region.
- Lowest 7 regions are Selectable Protected Regions. The Selectable Protected Regions can be protected/non-protected selectively by `ECCCFG0.ecc_region_map[6:0]`. Other upper regions are protected/non-protected region, determined by `ECCCFG0.ecc_region_map_other`. Each bit of `ECCCFG0.ecc_region_map[6:0]` correspond to each of lowest 7 regions respectively. To protect a region, set the corresponding bit to '1', otherwise set to '0'. See [Figure 13-7](#) for more details.
- When the region is set to non-protected, ECC overhead commands are not injected and therefore, there is no performance drop.

Figure 13-7 Selectable Protected Region

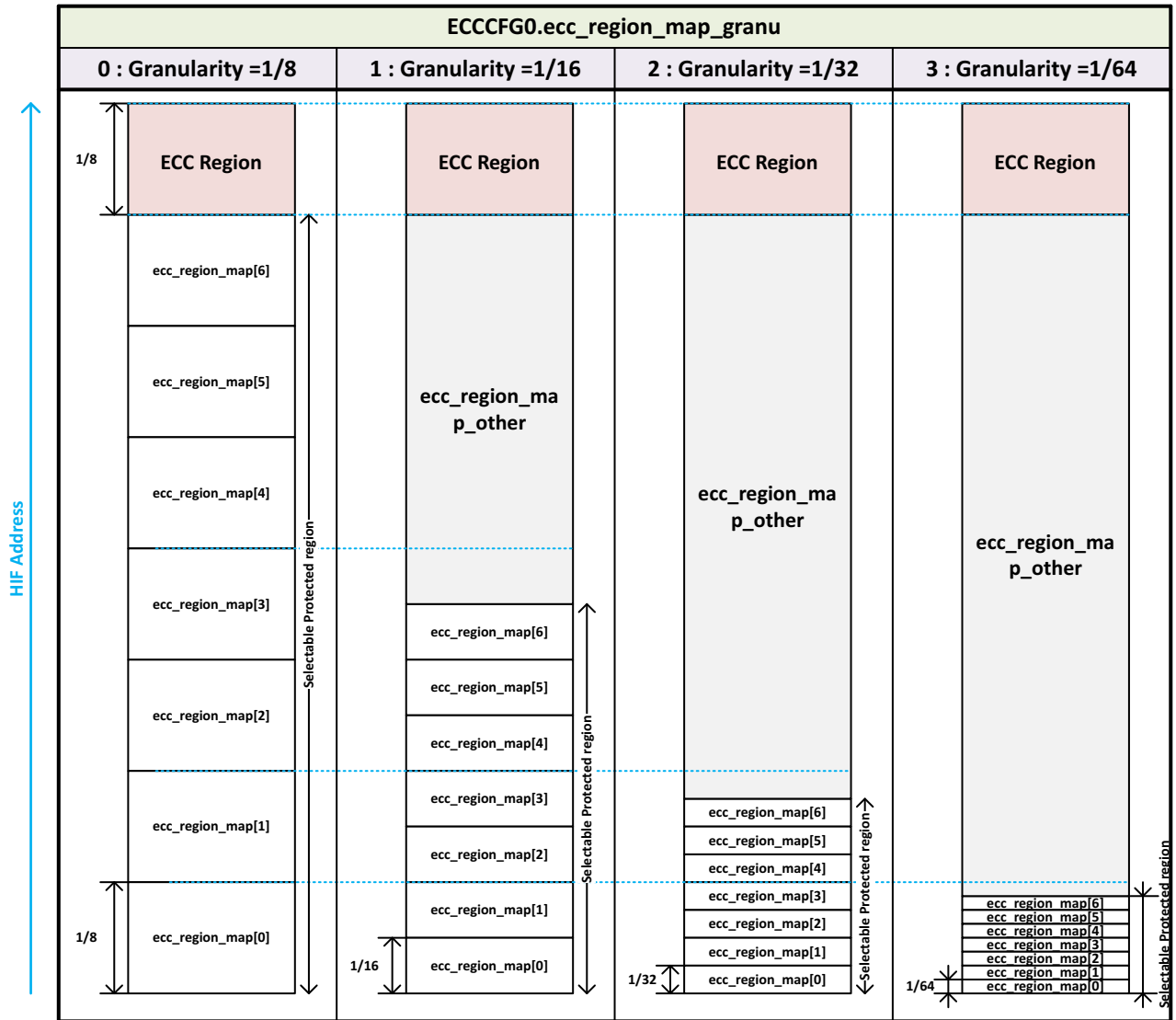


Figure 13-8 ECC Regions Example

1) ECCCFG0.ecc_region_map_granu=0 (1/8)

System Address bit	A32	A31	A30	A29	A28	A27	A26-A0		
SDRAM Address	C9	C8	C7	R13	R12	R11	Any		
0	0	0	0	X	X	X	X	Protected	ecc_region_map[0]=1
0	0	1	X	X	X	X	X	Non-Protected	ecc_region_map[1]=0
0	1	0	X	X	X	X	X	Protected	ecc_region_map[2]=1
0	1	1	X	X	X	X	X	Protected	ecc_region_map[3]=1
1	0	0	X	X	X	X	X	Non-Protected	ecc_region_map[4]=0
1	0	1	X	X	X	X	X	Non-Protected	ecc_region_map[5]=0
1	1	0	X	X	X	X	X	Protected	ecc_region_map[6]=1
1	1	1	X	X	X	X	X	ECC region	

2) ECCCFG0.ecc_region_map_granu=1 (1/16)

System Address	A32	A31	A30	A29	A28	A27	A26-A0		
SDRAM Address	C9	C8	C7	R13	R12	R11	Any		
0	0	0	0	0	X	X	X	Protected	ecc_region_map[0]=1
0	0	0	1	X	X	X	X	Non-Protected	ecc_region_map[1]=0
0	0	1	0	X	X	X	X	Protected	ecc_region_map[2]=1
0	0	1	1	X	X	X	X	Protected	ecc_region_map[3]=1
0	1	0	0	X	X	X	X	Non-Protected	ecc_region_map[4]=0
0	1	0	1	X	X	X	X	Non-Protected	ecc_region_map[5]=0
0	1	1	0	X	X	X	X	Protected	ecc_region_map[6]=1
0	1	1	1	X	X	X	X	Non-Protected	
1	0	X	X	X	X	X	X	Non-Protected	
1	1	0	X	X	X	X	X	Non-Protected	
1	1	1	X	X	X	X	X	ECC region	

3) ECCCFG0.ecc_region_map_granu=3 (1/64)

System Address	A32	A31	A30	A29	A28	A27	A26-A0		
SDRAM Address	C9	C8	C7	R13	R12	R11	Any		
0	0	0	0	0	0	0	X	Protected	ecc_region_map[0]=1
0	0	0	0	0	0	1	X	Non-Protected	ecc_region_map[1]=0
0	0	0	0	0	1	0	X	Protected	ecc_region_map[2]=1
0	0	0	0	0	1	1	X	Protected	ecc_region_map[3]=1
0	0	0	1	0	0	0	X	Non-Protected	ecc_region_map[4]=0
0	0	0	1	0	1	0	X	Non-Protected	ecc_region_map[5]=0
0	0	0	1	1	0	0	X	Protected	ecc_region_map[6]=1
0	0	0	1	1	1	1	X	Non-Protected	
0	0	1	X	X	X	X	X	Non-Protected	
0	1	X	X	X	X	X	X	Non-Protected	
1	0	X	X	X	X	X	X	Non-Protected	
1	1	0	X	X	X	X	X	Non-Protected	
1	1	1	X	X	X	X	X	ECC region	

In the three examples of Figure 13-8, where number of system (HIF) address bits is 33, the highest column address bit is column[9], ECCCFG0.ecc_region_map = 7'b1001101 and ECCCFG0.ecc_region_map_granu = 0, 1, 3 respectively.

In Figure 13-8, the:

- Blue regions are ECC protected,
- White regions are non-ECC protected, and
- Red region are ECC region.

**Note**

All regions are protected with the following setting (default):

- ECCCFG0.ecc_region_map = 7'b1111111
- ECCCFG0.ecc_region_map_granu = 0
- ECCCFG0.ecc_region_map_other = 0

13.1.1.5 Locking of ECC Region

The ECC region comprises two separate sub-regions:

1. Used sub-region:
 - ECC parity area for ECC protected regions.
2. Unused sub-region:
 - ECC parity area for unprotected regions (reserved but not used by inline ECC logic).
 - Waste region, which is never used by inline ECC logic.

The Used sub-region is locked/unlocked by the register ECCCFG1.ecc_region_parity_lock.

The Unused sub-region is locked/unlocked by the register ECCCFG1.ecc_region_waste_lock.

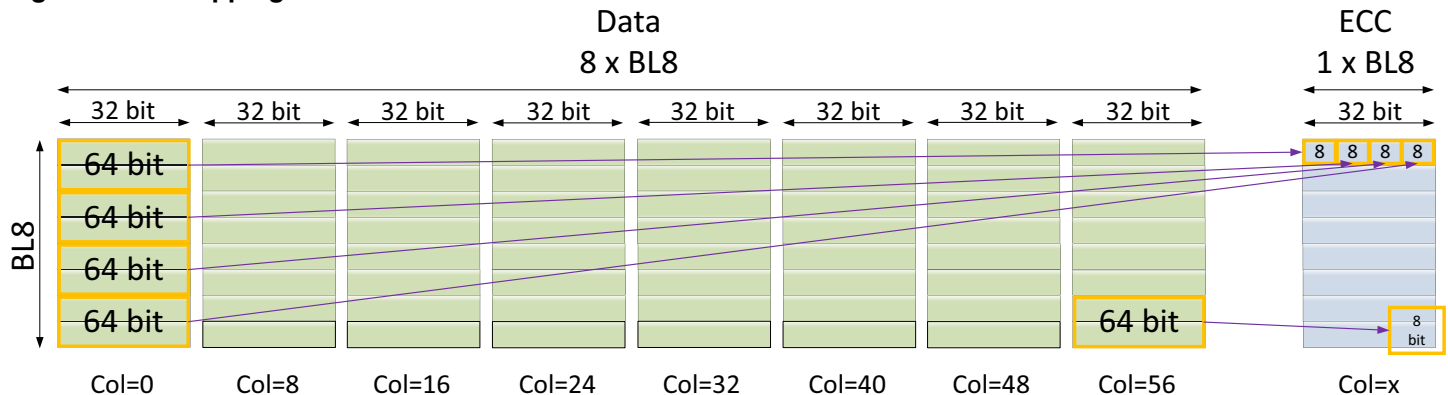
**Note**

For more information on identifying Used and Unused region, see [“Formula of Used and Unused Region”](#) on page 282.

Used and Unused regions can be identified by the HIF address.

```
// hif_addr_h3 means the highest valid 3 bits of HIF address.
assign ecc_region_map_ext = {1'b0, reg_ddrc_ecc_region_map};
assign offset_addr = `MEMC_BURST_LENGTH==8 ? col[5:3] : col[6:4];
assign hif_ecc_addr_sep = {offset_addr, hif_cmd_addr[highest_col_bit-3 -: 3]} >>
(3-reg_ddrc_ecc_region_map_granu);
assign ecc_region_used = reg_ddrc_ecc_mode==3'b000 ? 1'b0 : (&hif_addr_h3) &
~|hif_ecc_addr_sep[5:3] & ecc_region_map_ext[hif_ecc_addr_sep[2:0]];
assign ecc_region_unused = reg_ddrc_ecc_mode==3'b000 ? 1'b0 : (&hif_addr_h3) &
~(~|hif_ecc_addr_sep[5:3] & ecc_region_map_ext[hif_ecc_addr_sep[2:0]]);
```

[Figure 13-9](#) shows the mapping between the Data and the ECC code. Regardless of the memory data width, 64/8 SECCDED is used.

Figure 13-9 Mapping Between Data and ECC

In normal conditions, the user-software must not access the Used region of the ECC hole and therefore must be kept locked. However, for debugging or error injection purposes into ECC parity, it can be unlocked by the `ecc_region_parity_lock` register.

If required, for normal transfers, the user-software can access the Unused region of the ECC hole and therefore can be kept unlocked using the `ecc_region_waste_lock` register. However, it must be noted that the unused region is not consecutive in terms of system addresses and it is interleaved with the used region.

The unused region can be made consecutive by using the ECC region remap functionality. For more information, see “[ECC Region Remap](#)” on page 279.

Locking the ECC hole sub-regions from access means the following:

- Any write or RMW request to this region is discarded, while the HIF output `hif_wdata_ptr_addr_err` is asserted.
- Any read request to this region is executed to the memory, but without the ECC check and returns all zeros, while the output `hif_rdata_addr_err` is asserted.

For AXI configurations, any assertion of `hif_wdata_ptr_addr_err` or `hif_rdata_addr_err` also results in SLVERR/ERROR response.

When Inline ECC is disabled in the software, all the memory space can be accessed.

13.1.1.6 Error Injection Through Software (ECC Data Poisoning)

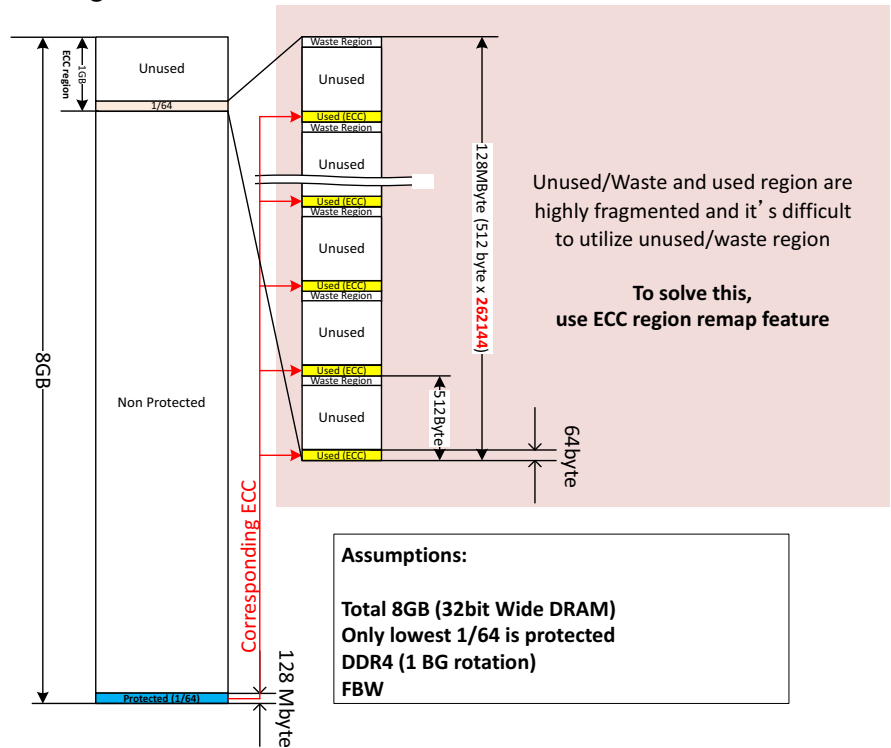
ECC error injection is useful for system-level software validation. Unlike in sideband ECC, there is no dedicated hardware support for it. However, errors can be easily injected through the software by unlocking the ECC region through the `ecc_region_parity_lock` register and overriding certain ECC parity bits. When the corresponding addresses are read from a protected memory region, ECC errors are generated as correctable or uncorrectable, depending on the type of error introduced. For more details, see section “[Locking of ECC Region](#)” on page 278.

13.1.1.7 ECC Region Remap

The unused sub-region can be used as unprotected region. However, it can be highly fragmented due to ECC mapping method (that is, ECC and Data are mapped to the same page), and it may be difficult to utilize the unused/wasted region.

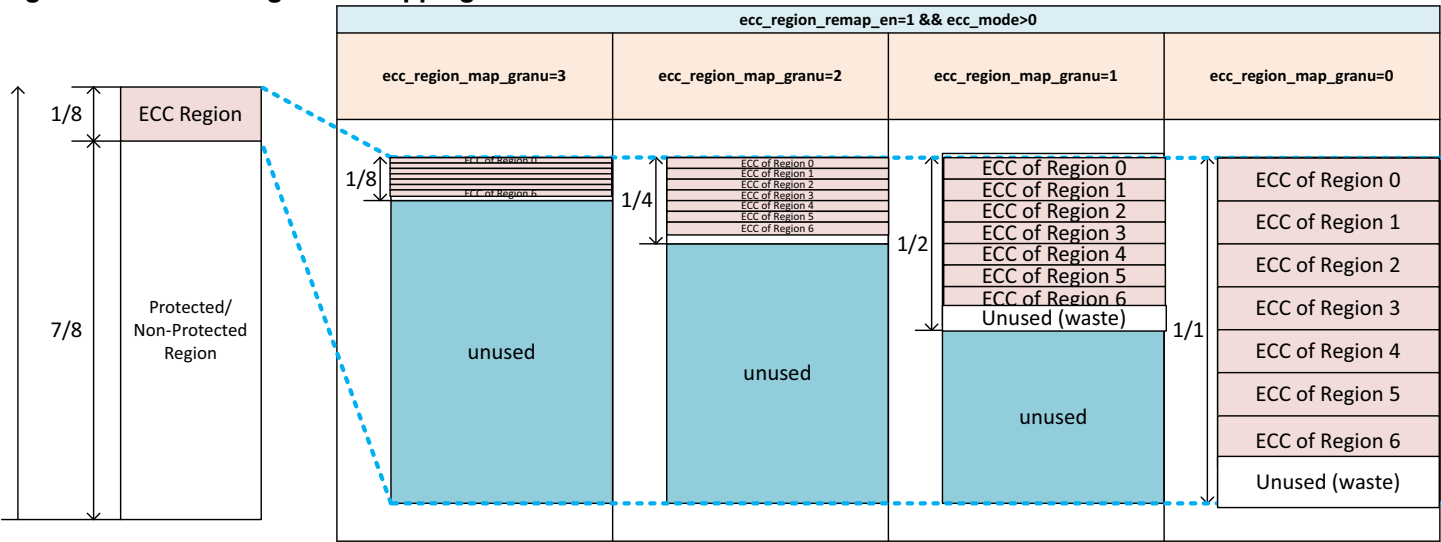
Figure 13-10 shows an example of the fragmentation.


Figure 13-10 Example of Fragmentation



Enabling ECC region remap feature (setting `ECCCFG0.ecc_region_remap_en` to '1') can make the unused ECC region consecutive. When ECC region remap feature is enabled, ECC region is remapped as shown in Figure 13-11.

Figure 13-11 ECC Region Remapping



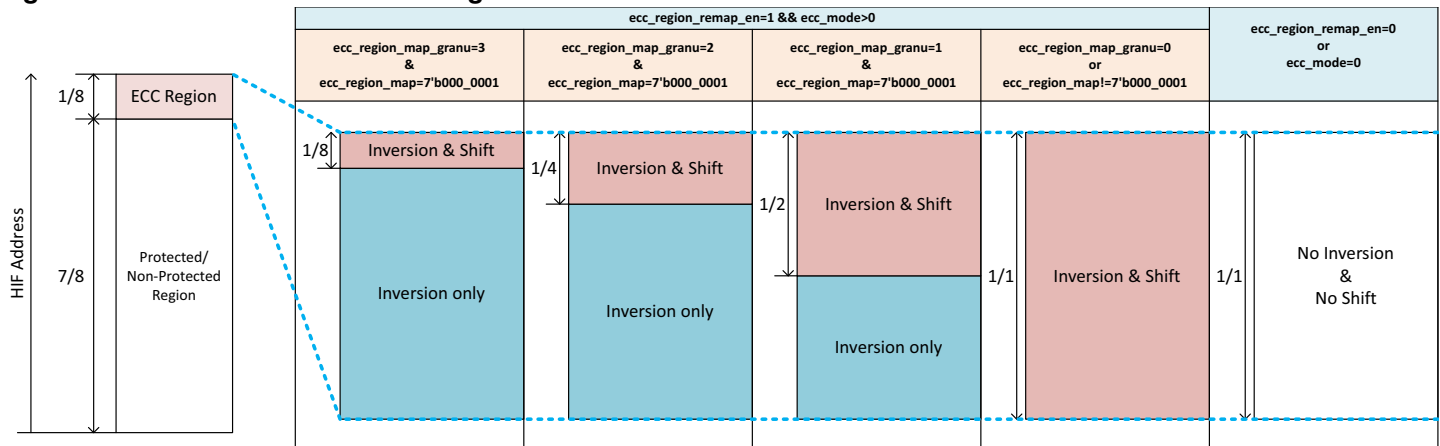
 **Note** Figure 13-11 is for `ECCCFG0.ecc_region_map_other=0`. When `ECCCFG0.ecc_region_map_other=1`, “unused” in the figure is replaced with “used”.

To optimize performance LPDDR5/4/4X Memory Controller has two types of remapping in ECC region, which are used for different parts of the ECC region.

- Inversion and Shift: Address map is not optimized for performance because C6...C4 are mapped to higher bits of HIF address with remap enabled.
- Inversion only: Performance is equivalent to other regions in terms of address map.

See Figure 13-12 performance of ECC region, this shows the mapping for “Inversion & Shift” and “Inversion only” which depends on `ECCCFG0.ecc_region_map_granu`, `ECCCFG0.ecc_region_map` and `ECCCFG0.ecc_region_remap_en`.

Figure 13-12 Performance of ECC Region



When ECC region remap is enabled (`ECCCFG0.ecc_region_remap_en=1`), ECC address can be calculated from the HIF command address (`hif_cmd_addr`) by the formula of ECC address, instead of the original calculation method in Table 13-1 on page 275:

Formula of ECC address when `ECC_region_remap_en = 1`

If $x > y$,

```
hif_ecc_addr = {3'b111, ~hif_cmd_addr[hif_highest_bit -: 6],
hif_cmd_addr[hif_highest_bit-6 : x+3], hif_cmd_addr[x-1:y], {hif_cmd_addr[x+2:x],
hif_cmd_addr[y-1:0]}>>3}
```

otherwise

```
hif_ecc_addr = {3'b111, ~hif_cmd_addr[hif_highest_bit -: 6],
hif_cmd_addr[hif_highest_bit-6 : x+3], {hif_cmd_addr[x+2:x],
hif_cmd_addr[y-1:0]}>>3}
```

Where x is derived from HIF address bits that map to column 4 and $y=4$.

`hif_highest_bit` indicates the highest valid bit of HIF address.

When ECC region remap is enabled (`ECCCFG0.ecc_region_remap_en = 1`), there is a limitation for the address mapping of Inline ECC.



Note

Block offset address must be mapped to the consecutive HIF bits. That is, C6...C4 must be mapped to consecutive HIF bits in `MEMC_BURST_LENGTH = 16`. C5...C3 must be mapped to consecutive HIF bits in `MEMC_BURST_LENGTH = 8`.

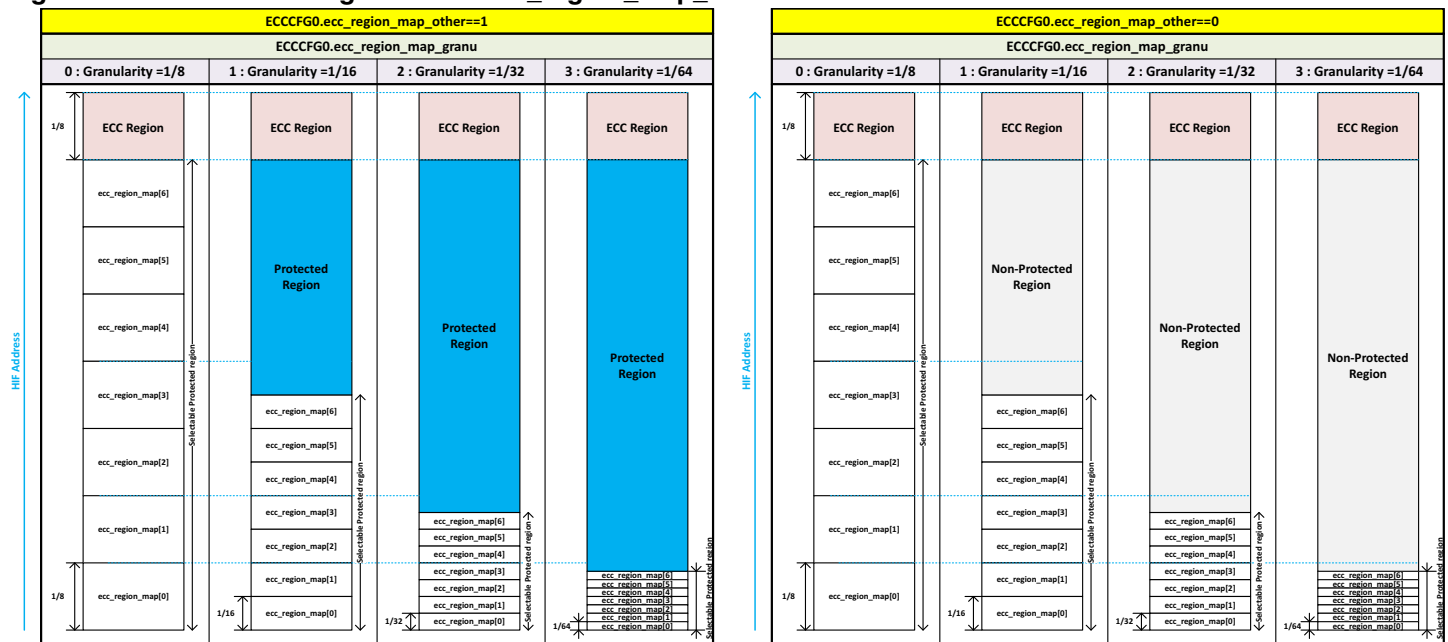
13.1.1.8 ECC Region Map Others

When `ECCCFG0.ecc_region_map_granu != 0`, there is a region that is not controlled by `ECCCFG0.ecc_region_map`. Set `ECCCFG0.ecc_region_map_other` to '1' to make this a protected region. In this way, you can choose to have a small unprotected region (such as 1/64 of the memory space), while protecting the remainder.

When `ECCCFG0.ecc_region_map_other` is 1, the ECC region remap can be enabled as well. In this case all the ECC region are remapped as "Inversion & Shift".

See Figure 13-13 about protected region when `ECCCFG0.ecc_region_map_other` is 1 or 0.

Figure 13-13 Protected Region When `ecc_region_map_other=1` or 0



13.1.1.9 Formula of Used and Unused Region

The following equations are defined in terms of the HIF address (`hif_cmd_addr`), the used and unused regions in different cases (depending on `ECCCFG0.ecc_region_remap_en` and `ECCCFG0.ecc_region_map_other`). In each case, if `ecc_region_used` evaluates as true, the HIF address is part of the used region. If `ecc_region_unused` evaluates as true, the HIF address is part of the unused region.

When `ECCCFG0.ecc_region_remap_en = 0` and `ECCCFG0.ecc_region_map_other = 0`:

```
// hif_addr_h3 means the highest valid 3 bits of HIF address. ecc_region_map_ext =
{1'b0, reg_ddrc_ecc_region_map}; assign offset_addr = `MEMC_BURST_LENGTH==8 ?
col[5:3] : col[6:4]; hif_ecc_addr_sep =
{offset_addr, hif_cmd_addr[highest_col_bit-3 -: 3]} >>
(3-reg_ddrc_ecc_region_map_granu); ecc_region_used = reg_ddrc_ecc_mode==3'b000 ?
1'b0 : (&hif_addr_h3) & ~|hif_ecc_addr_sep[6:3] &
ecc_region_map_ext[hif_ecc_addr_sep[2:0]]; ecc_region_unused =
reg_ddrc_ecc_mode==3'b000 ? 1'b0 : (&hif_addr_h3) & ~(|hif_ecc_addr_sep[6:3] &
ecc_region_map_ext[hif_ecc_addr_sep[2:0]]);
```

When `ECCCFG0.ecc_region_remap_en = 0` and `ECCCFG0.ecc_region_map_other = 1`:

```
// hif_addr_h3 means the highest valid 3 bits of HIF address. ecc_region_map_bit7
= ~|reg_ddrc_ecc_region_map_granu ? 1'b0 :
reg_ddrc_ecc_region_map_other;ecc_region_map_ext = {ecc_region_map_bit7,
reg_ddrc_ecc_region_map};offset_addr = `MEMC_BURST_LENGTH==8 ? col[5:3] : col[6:4];
hif_ecc_addr_sep = {offset_addr,hif_cmd_addr[highest_col_bit-3 -: 3]} >>
(3-reg_ddrc_ecc_region_map_granu); ecc_region_used_sep = ~|hif_ecc_addr_sep[5:3]
& ecc_region_map_ext[hif_ecc_addr_sep[2:0]];ecc_region_used_other =
|hif_ecc_addr_sep[5:3] & ~&offset_addr &
reg_ddrc_ecc_region_map_other;ecc_region_used = reg_ddrc_ecc_mode==3'b000 ? 1'b0 :
(&hif_addr_h3) & (ecc_region_used_sep | ecc_region_used_other);ecc_region_unused =
reg_ddrc_ecc_mode==3'b000 ? 1'b0 : (&hif_addr_h3) & ~(ecc_region_used_sep |
ecc_region_used_other);
```

When `ECCCFG0.ecc_region_remap_en = 1` and `ECCCFG0.ecc_region_map_other = 0`:

```
ecc_region_used = (&hif_cmd_addr[highest -: (3+granu)]) &&
reg_ddrc_ecc_region_map[~hif_ecc_addr[highest-(3+granu) -:3]];ecc_region_unused =
(&hif_cmd_addr[highest -: 3]) && ~ecc_region_used
```

When `ECCCFG0.ecc_region_remap_en = 1` and `ECCCFG0.ecc_region_map_other = 1`:

```
ecc_region_map_bit7 = ~|ecc_region_map_granu ? 1'b0 :
reg_ddrc_ecc_region_map_other;ecc_region_map_ext = {ecc_region_map_bit7,
reg_ddrc_ecc_region_map};ecc_region_used = ( ( (&hif_cmd_addr[highest -:
(3+granu)]) & ecc_region_map_ext[~hif_ecc_addr[highest-(3+granu) -:3]]) ||
( reg_ddrc_ecc_region_map_other & (&hif_cmd_addr[highest -: 3]) &
(~&hif_cmd_addr[highest-3 -: granu]) & (|hif_cmd_addr[highest-3 -:3]) );
```

13.1.1.10 ECC Data Poisoning (Error Injection Through Software)

ECC error injection is useful for system-level software validation. Unlike in Sideband ECC mode, there is no dedicated hardware support for it in Inline ECC mode. However, errors can be easily injected through the software by unlocking the ECC region through the `ecc_region_parity_lock` register and overriding certain ECC parity bits. When the corresponding addresses are read from a protected memory region, ECC errors are generated as correctable or uncorrectable, depending on the type of error introduced. For more details, see section “[Locking of ECC Region](#)” on page 278.

13.1.1.11 Related Hardware Parameters

The following are the new hardware configuration parameters that are available in the GUI:

- `MEMC_INLINE_ECC`: enables Inline ECC.
- `MEMC_SIDEBAND_ECC`: enables Sideband ECC (mutually exclusive with Inline ECC).
- `MEMC_ECCAP`: enables address protection within Inline ECC.
- `MEMC_NO_OF_BLK_CHANNEL`: interleaving depth. Indicates the number of blocks that can be interleaved at DDRC input (HIF). Valid values are 4, 8, and 16.

The following are the new hardware configuration parameters that are not available in the GUI (derived parameters):

- **MEMC_NO_OF_BLK_TOKEN:** indicates number of blocks that can be maintained in DDRC. Equals to $\text{MEMC_NO_OF_ENTRY} + \text{MEMC_NO_OF_BLK_CHANNEL}$. Though this number is optimized, it does not consider the read latency. There could be corner cases where block tokens could be full before CAM is full and cause `hif_cmd_stall` to be asserted, and this number links the read (BRT) and write (BWT) tokens.
- **MEMC_NO_OF_BRT:** indicates the read cache (ECC array) depth which is equal to $\text{MEMC_NO_OF_ENTRY}/2 + \text{MEMC_NO_OF_BLK_CHANNEL}$.
- **MEMC_NO_OF_BWT:** indicates the write cache (ECC accumulator) depth which is equal to $\text{MEMC_NO_OF_ENTRY}/2 + \text{MEMC_NO_OF_BLK_CHANNEL}$.

Debug Port

Table 13-2 shows the output signals for debugging.

Table 13-2 Output Signal for Debugging

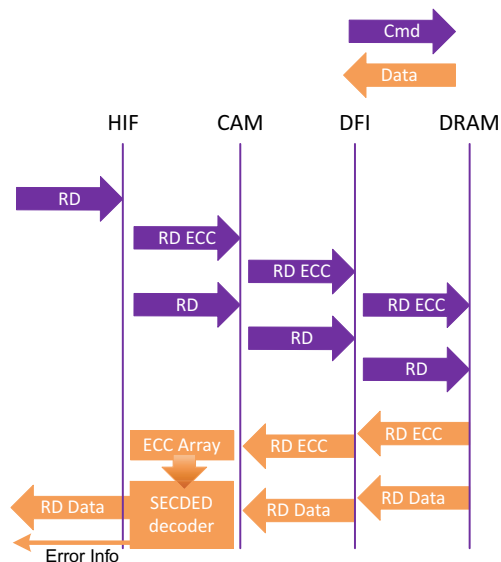
Name	Width	IO	From/To	Description
dbg_dfi_ie_cmd_type	[2:0]	O	From DFI	<p>Debug signal. This signal is valid when <code>dfi_cs</code> is valid and command is RD/RDA/WR/WRA, otherwise don't care</p> <p>Command == RD/RDA</p> <ul style="list-style-type: none"> ■ 000: RD_N (RD Data for non-protected region) ■ 001: RD_E (RD Data for protected region) ■ 010: RE_B (RD ECC in block read/write) <p>Command == WR/WRA</p> <ul style="list-style-type: none"> ■ 000: WD_N (WR Data for non-protected region) ■ 001: WD_E (WR Data for protected region) ■ 010: WE_BW (WR ECC in block write) ■ 111: MPR write (DDR4 only)

13.1.1.12 Command Flow

The controller fully manages the injection and scheduling of ECC commands which are also called the “overhead commands”. This section discusses the data structures and the flow of ECC commands.

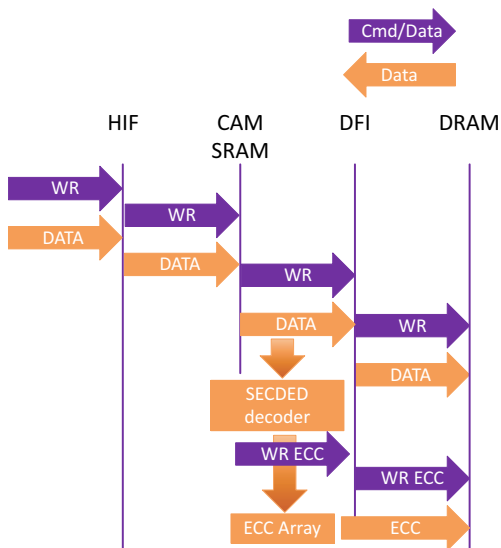
When a Read (Data) command is incoming from HIF, the Read ECC command is generated internally as necessary. For example, if the Read ECC is already stored in the ECC Array, then Read ECC command is not required. The ECC address is calculated from the HIF address. The Read ECC command must be scheduled out before the Read Data command.

Figure 13-14 Read Operation Overview



When a Write (Data) command is incoming from HIF, the ECC is calculated when it is being scheduled out, and ECC is stored into an ECC Accumulator. When Write ECC is accumulated internally, the Write ECC command becomes available in the CAM to be scheduled out.

Figure 13-15 Write Operation Overview



 **Note**

ECC is written to SDRAM using data mask if ECC Accumulator has partial ECC for an ECC block. As RMW is not used for writing ECC to SDRAM, data mask must be enabled if Inline ECC feature is used.

13.1.1.12.1 Write ECC CAM

The WR ECC commands are stored in a separate write ECC CAM. This improves the DDR efficiency especially in the worst case scenarios where ECC overhead ratio is very high.

The depth of the WR ECC CAM is always half of the WR CAM depth. [Table 13-3](#) provides the depths for different hardware configurations.

Table 13-3 Depths for Different Hardware Configurations

CAM Size (MEMC_NO_OF_ENTRY)	WR CAM	WR ECC CAM
16	16	8
32	32	16
64	64	32



Note

Since WR ECC CAM does not need Write Data SRAM, Write Data SRAM size is not changed by this.

13.1.1.12.2 Block Tokens / ECC Arrays

The DDRCTL has BTs (Block Tokens). The number of BTs are defined by a hardware parameter MEMC_NO_OF_BLK_TOKEN.

A BT can handle both read and write.

Each block must have a BT.

- When one block access only has read, the block only need BRT;
- When one block access only has write, the block only need BWT;
- When one block access has both read and write or has RMW, the block need both BRT and BWT.

Each BWT has an internal buffer (ECC_ACC) to accumulate ECC for writes in the block. Every time CAM schedules a write command, the corresponding ECC is calculated and stored in ECC_ACC. One block may have more than one write ECC command. When write ECC command is served, the associated write data is supplied directly from ECC_ACC. ECC_ACC is cleared with the last write ECC command or when the block is terminated.

Each BRT has an internal buffer (ECC_ARRAY) to store ECC that returns from read ECC command. One block can only have one read ECC command. Once the read ECC command is served, the ECC is stored in ECC_ARRAY. ECC_ARRAY is used to decode the following read commands in the block.

If a block has both BWT and BRT, and the ECC_ARRAY is ready when write ECC command is injected, the data of write ECC command is merged from both ECC_ACC and ECC_ARRAY, otherwise the write ECC command uses ECC_ACC only. The write ECC command could be a Masked Write or a Write in LPDDR4.

If a block has both BWT and BRT, and in a Read after Write case, the read data is decoded by the generated ECC in ECC_ACC, because the ECC_ARRAY could be out of date.

When access to a data block, and its block address is not in any valid block channel (see next chapter for details) a new BT is allocated to the data block. The BT is kept allocated until the following conditions are met:

- The BT is not in any valid block channel.
- All commands associated to the BT are served from CAM.

One BT only has maximum one Read ECC command and multiple Write ECC commands. The Write ECC command is the last command of the BT if needed. Once Read ECC is executed, the ECC parity is stored and used until the BT is released. The stored ECC is updated by any Write command with this BT (without writing back to SDRAM) and finally this is written to SDRAM as last command of the BT if needed.

13.1.1.12.3 Block Channels

Block channel is a data structure to support interleaving block address accesses at HIF with saving number of overhead commands.

Number of block channels is determined by a hardware parameter `MEMC_NO_OF_BLK_CHANNEL` and this means interleaving depth.

Each block channel has the following information (not necessarily all):

- Block Address (based on HIF address)
- ECC Address (based on DRAM address)
- Block Token (BT), Block Write Token (BWT), Block Read Token (BRT)
- A flag to indicate if any read is done in the channel
- A flag to indicate if any write is done in the channel
- A flag to indicate if this channel is valid
- A vector to record which offset address is written in the block
- A vector to record which offset address is read in the block

The channel structure has a write pointer to indicate the channel to be overridden with.

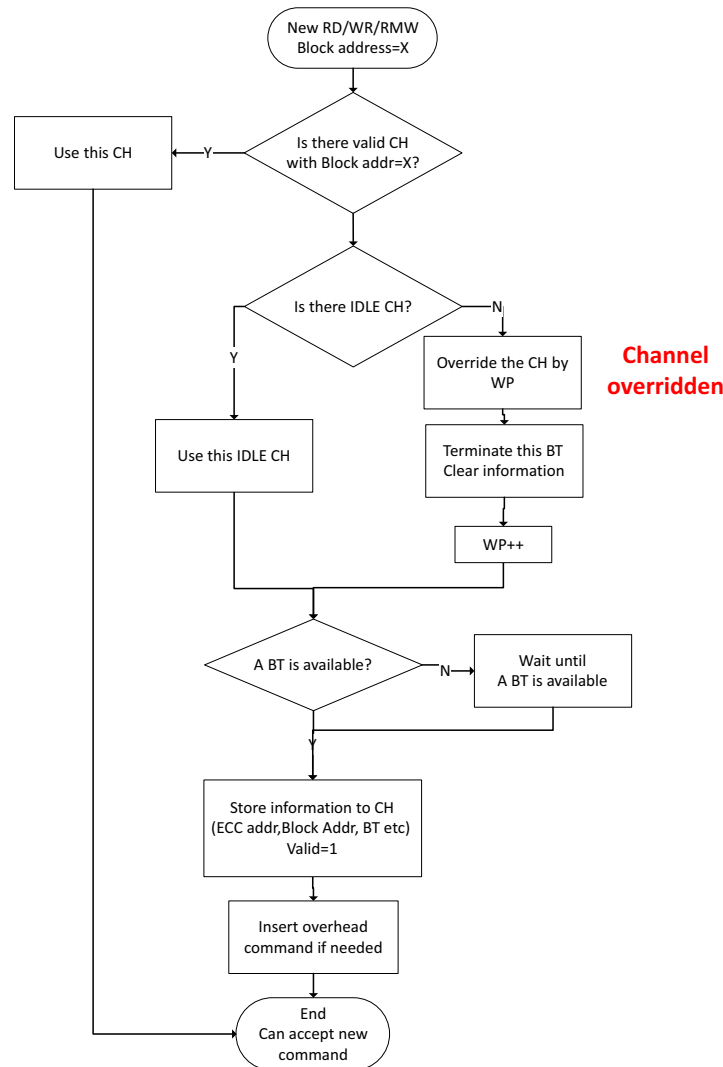
After reset, all channels are invalid (`valid=0`) and the write pointer indicates Channel 0 (CH0).

When a command (RD/WR/RMW) is incoming with block address = X, if there is a valid channel (`valid = 1`) with block address = X, use this channel and the already allocated BT, push the incoming command and overhead commands into CAM with the BT. Otherwise if there is any IDLE channel (`valid = 0`), use this IDLE channel, allocate a new BT, push the incoming command and overhead commands into CAM with the new BT, and then the channel becomes valid. Otherwise, update the channel indicated by the write pointer, allocate a new BT and push commands into CAM, then write pointer is increased by 1. If this channel is already valid, this is “channel overridden” condition and terminates the BT.

When a RD or RMW command is incoming, and this is first Read/RMW command in this channel, push an overhead RD ECC command into CAM prior to the command.

Every time a WR or RMW is incoming, push overhead WR ECC command to CAM following the command.

Figure 13-16 shows the flow for channels.

Figure 13-16 Block Channel Flow

Index:

- CH: Channel index
- WP: Write pointer to indicate CH to be overridden

Also, all block channels are invalidated (valid = 0) and BTs associated to channels are terminated by the following events.

- Before entering Self-Refresh with CAM drain
- Any read/write access to unlocked ECC parity hole sub-region, which is represented by `ECCCFG1.ecc_region_parity_lock = 0`
- `blk_channel_idle_time_x32` timer expires
- A full block read (8 access to different offset address to make all the address are read in the block)
- A full block write (8 access to different offset address to make all the address are written in the block)

When a Write follows a Read to the same address, the block is terminated before the Write command. The reason is that although CAM guarantee the order is RD and then WR, the ECC cache could be updated by

the write data before read data is returned due to the read latency being bigger than the write latency. As a result, when the read data is returned, it does not match with ECC cache which can cause decoder error.

**Note**

When one or more block channels are active, the DDRC command channel is not treated as IDLE. This affects to idleness timers `PWRTMG.selfref_to_x32`, `RFSSHSET1TMG0.refresh_to_x1_x32`, `PWRTMG.powerdown_to_x32`, and `HWLPTMG0.hw_lp_idle_x32`.

13.1.1.12.4 Overhead Commands

Each data block requires overhead commands depending on command type and is shown in a table [Table 13-4](#).

Table 13-4 Overhead Commands Depending on Command Types

Command Type (HIF)	Overhead Command	Note
RD	Read ECC (RE_B)	Read ECC is served prior to Read Data command
WR	Write ECC (WE_BW)	Write ECC is served following Write Data command
RMW	Read ECC (RE_B)	Read ECC is served prior to Read part of RMW
	Write ECC (WE_BW)	Write ECC is served following Write part of RMW

For Read ECC (RE_B), if RE_B is already inserted by a RD or RMW command, no more RE_B are inserted when RD or RMW command is received again within a channel.

For Write ECC (WE_BW), WE_BW is pushed to CAM after a WR command. For LPDDR4 protocol and if the block is not full write access, it is pushed to CAM as a Masked Write. Otherwise, it is pushed to CAM as a Write command. If Read ECC (RE_B) is already inserted within a channel and its data is returned, the Masked Write is changed to Write command in CAM.

Although Write ECC (WE_BW) is pushed to CAM after every WR command, those Write ECC commands could be combined as one command if the previous WE_BW has not been served.

[Figure 13-17](#) on page 290 gives an example sequence to explain how channel works and how overhead commands are injected (#channels = 4).

Figure 13-17 Channel Sequence Example

#Seq	From HIF	Block channels	To Scheduler	Note																						
0	After reset	<div><div>Blk Addr</div><div>BT</div><div>any rd</div><div>any wr</div><div>valid</div></div> <div><div>CH0</div><div>CH1</div><div>CH2</div><div>CH3</div></div> <table><tr><td></td><td></td><td></td><td></td><td>0</td></tr><tr><td></td><td></td><td></td><td></td><td>0</td></tr><tr><td></td><td></td><td></td><td></td><td>0</td></tr><tr><td></td><td></td><td></td><td></td><td>0</td></tr></table>					0					0					0					0		Channels are all invalid Write pointer = CH0		
				0																						
				0																						
				0																						
				0																						
1	<div>RD</div> <div>Blk A</div>	<div><div>Blk Addr</div><div>BT</div><div>any rd</div><div>any wr</div><div>valid</div></div> <div><div>CH0</div><div>CH1</div><div>CH2</div><div>CH3</div></div> <table><tr><td>A</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td></td><td></td><td></td><td></td><td>0</td></tr><tr><td></td><td></td><td></td><td></td><td>0</td></tr><tr><td></td><td></td><td></td><td></td><td>0</td></tr></table>	A	0	1	0	1					0					0					0	<div>RD ECC</div> <div>Blk A</div>	<div>RD</div> <div>Blk A</div>	There is no valid CH with blk=A, use CH0. This is first read for CH0, insert overhead RD command Increment write pointer.	
A	0	1	0	1																						
				0																						
				0																						
				0																						
2	<div>WR</div> <div>Blk B</div>	<div><div>Blk Addr</div><div>BT</div><div>any rd</div><div>any wr</div><div>valid</div></div> <div><div>CH0</div><div>CH1</div><div>CH2</div><div>CH3</div></div> <table><tr><td>A</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>B</td><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td></td><td></td><td></td><td></td><td>0</td></tr><tr><td></td><td></td><td></td><td></td><td>0</td></tr></table>	A	0	1	0	1	B	1	0	1	1					0					0	<div>WR</div> <div>Blk B</div>	<div>WR ECC</div> <div>Blk B</div>	There is no valid CH with blk=B, use CH1. This is write, insert overhead WR command. Increment write pointer.	
A	0	1	0	1																						
B	1	0	1	1																						
				0																						
				0																						
3	<div>RD</div> <div>Blk A</div>	<div><div>Blk Addr</div><div>BT</div><div>any rd</div><div>any wr</div><div>valid</div></div> <div><div>CH0</div><div>CH1</div><div>CH2</div><div>CH3</div></div> <table><tr><td>A</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>B</td><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td></td><td></td><td></td><td></td><td>0</td></tr><tr><td></td><td></td><td></td><td></td><td>0</td></tr></table>	A	0	1	0	1	B	1	0	1	1					0					0	<div>RD</div> <div>Blk A</div>		There is valid CH with blk=A, use CH0. This is read, but read is already issued for this channel, no overhead command is inserted. Do not increment write pointer.	
A	0	1	0	1																						
B	1	0	1	1																						
				0																						
				0																						
4	<div>RD</div> <div>Blk C</div>	<div><div>Blk Addr</div><div>BT</div><div>any rd</div><div>any wr</div><div>valid</div></div> <div><div>CH0</div><div>CH1</div><div>CH2</div><div>CH3</div></div> <table><tr><td>A</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>B</td><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td>C</td><td>2</td><td>1</td><td>0</td><td>1</td></tr><tr><td></td><td></td><td></td><td></td><td>0</td></tr></table>	A	0	1	0	1	B	1	0	1	1	C	2	1	0	1					0	<div>RD ECC</div> <div>Blk C</div>	<div>RD</div> <div>Blk C</div>	There is no valid CH with blk=C, use CH2. This is first read for CH2, insert overhead RD command Increment write pointer.	
A	0	1	0	1																						
B	1	0	1	1																						
C	2	1	0	1																						
				0																						
5	<div>RMW</div> <div>Blk D</div>	<div><div>Blk Addr</div><div>BT</div><div>any rd</div><div>any wr</div><div>valid</div></div> <div><div>CH0</div><div>CH1</div><div>CH2</div><div>CH3</div></div> <table><tr><td>A</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>B</td><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td>C</td><td>2</td><td>1</td><td>0</td><td>1</td></tr><tr><td>D</td><td>3</td><td>1</td><td>1</td><td>1</td></tr></table>	A	0	1	0	1	B	1	0	1	1	C	2	1	0	1	D	3	1	1	1	<div>RD ECC</div> <div>Blk D</div>	<div>RMW</div> <div>Blk D</div>	<div>WR ECC</div> <div>Blk D</div>	There is no valid CH with blk=D, use CH3. This is first read for CH3, insert overhead RD and WR command Increment write pointer.
A	0	1	0	1																						
B	1	0	1	1																						
C	2	1	0	1																						
D	3	1	1	1																						
6	<div>RD</div> <div>Blk B</div>	<div><div>Blk Addr</div><div>BT</div><div>any rd</div><div>any wr</div><div>valid</div></div> <div><div>CH0</div><div>CH1</div><div>CH2</div><div>CH3</div></div> <table><tr><td>A</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>B</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>C</td><td>2</td><td>1</td><td>0</td><td>1</td></tr><tr><td>D</td><td>3</td><td>1</td><td>1</td><td>1</td></tr></table>	A	0	1	0	1	B	1	1	1	1	C	2	1	0	1	D	3	1	1	1	<div>RD ECC</div> <div>Blk B</div>	<div>RD</div> <div>Blk B</div>	There is valid CH with blk=B, use CH1. This is first read in CH1. Insert overhead RD command. Do not increment write pointer.	
A	0	1	0	1																						
B	1	1	1	1																						
C	2	1	0	1																						
D	3	1	1	1																						
7	<div>RD</div> <div>Blk E</div>	<div><div>Blk Addr</div><div>BT</div><div>any rd</div><div>any wr</div><div>valid</div></div> <div><div>CH0</div><div>CH1</div><div>CH2</div><div>CH3</div></div> <table><tr><td>E</td><td>4</td><td>1</td><td>0</td><td>1</td></tr><tr><td>B</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>C</td><td>2</td><td>1</td><td>0</td><td>1</td></tr><tr><td>D</td><td>3</td><td>1</td><td>1</td><td>1</td></tr></table>	E	4	1	0	1	B	1	1	1	1	C	2	1	0	1	D	3	1	1	1	<div>RD ECC</div> <div>Blk E</div>	<div>RD</div> <div>Blk E</div>	There is no valid CH with blk=E, use CH0. CH0 is already valid, so this is "channel overridden". BT0 will be released once all commands belonging to BT0 are scheduled out from the scheduler. Overhead RD command is inserted for new block as this is first read. Increment write pointer.	
E	4	1	0	1																						
B	1	1	1	1																						
C	2	1	0	1																						
D	3	1	1	1																						
8	<div>WR</div> <div>Blk F</div>	<div><div>Blk Addr</div><div>BT</div><div>any rd</div><div>any wr</div><div>valid</div></div> <div><div>CH0</div><div>CH1</div><div>CH2</div><div>CH3</div></div> <table><tr><td>E</td><td>4</td><td>1</td><td>0</td><td>1</td></tr><tr><td>F</td><td>5</td><td>0</td><td>1</td><td>1</td></tr><tr><td>C</td><td>2</td><td>1</td><td>0</td><td>1</td></tr><tr><td>D</td><td>3</td><td>1</td><td>1</td><td>1</td></tr></table>	E	4	1	0	1	F	5	0	1	1	C	2	1	0	1	D	3	1	1	1	<div>WR</div> <div>Blk F</div>	<div>WR ECC</div> <div>Blk F</div>	There is no valid CH with blk=F, use CH1. CH1 is already valid, so this is "channel overridden". BT1 will be released once all commands belonging to BT1 are scheduled out from the scheduler. This is write, insert overhead WR ECC command. Increment write pointer.	
E	4	1	0	1																						
F	5	0	1	1																						
C	2	1	0	1																						
D	3	1	1	1																						
9	<div>WR</div> <div>Blk F</div>	<div><div>Blk Addr</div><div>BT</div><div>any rd</div><div>any wr</div><div>valid</div></div> <div><div>CH0</div><div>CH1</div><div>CH2</div><div>CH3</div></div> <table><tr><td>E</td><td>4</td><td>1</td><td>0</td><td>1</td></tr><tr><td>F</td><td>5</td><td>0</td><td>0</td><td>0</td></tr><tr><td>C</td><td>2</td><td>1</td><td>0</td><td>1</td></tr><tr><td>D</td><td>3</td><td>1</td><td>1</td><td>1</td></tr></table>	E	4	1	0	1	F	5	0	0	0	C	2	1	0	1	D	3	1	1	1	<div>WR</div> <div>Blk F</div>	<div>WR ECC</div> <div>Blk F</div>	Assuming there are six WR commands to Blk F after seq#8, this is 8th WR command and this WR makes it full block write. The block F is terminated and CH1 becomes IDLE channel. Valid becomes 0 any rd and any wr are cleared as well. This is write, insert overhead WR ECC command. BT5 will be released once all commands belonging to BT5 are scheduled out from the scheduler. Do not increment write pointer.	
E	4	1	0	1																						
F	5	0	0	0																						
C	2	1	0	1																						
D	3	1	1	1																						
10	<div>WR</div> <div>Blk A</div>	<div><div>Blk Addr</div><div>BT</div><div>any rd</div><div>any wr</div><div>valid</div></div> <div><div>CH0</div><div>CH1</div><div>CH2</div><div>CH3</div></div> <table><tr><td>E</td><td>4</td><td>1</td><td>0</td><td>1</td></tr><tr><td>A</td><td>6</td><td>0</td><td>1</td><td>1</td></tr><tr><td>C</td><td>2</td><td>1</td><td>0</td><td>1</td></tr><tr><td>D</td><td>3</td><td>1</td><td>1</td><td>1</td></tr></table>	E	4	1	0	1	A	6	0	1	1	C	2	1	0	1	D	3	1	1	1	<div>WR</div> <div>Blk H</div>	<div>WR ECC</div> <div>Blk H</div>	There is no valid CH with blk=A, use CH1 rather than override the CH2 (indicated by WP) because CH1 is a IDLE channel. This is write, insert overhead WR ECC command. Do not increment write pointer.	
E	4	1	0	1																						
A	6	0	1	1																						
C	2	1	0	1																						
D	3	1	1	1																						

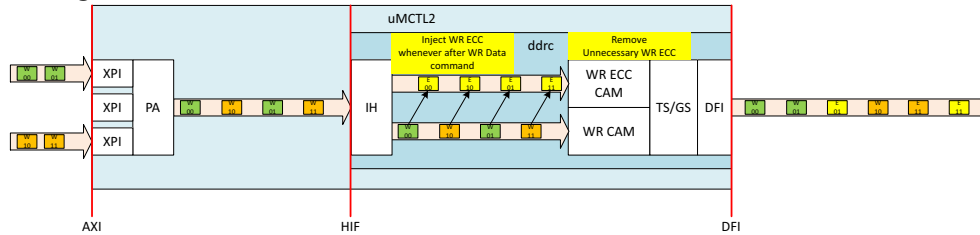
13.1.1.12.5 Block Address Collision

Block address collision means incoming command and existing commands (in CAM) have same block address (that is, use same ECC address) and have different BT (Block Token). This happens when a new block channel (block address = B) is initiated while there are entries with block address = B in CAM. If this happens, CAM does not accept incoming command until the collision is resolved by serving colliding commands in the CAM. At the same time, CAM priorities colliding commands (that is, flushes). The collision includes Write after Write (WAW), Read after Write (RAW), and Write after Read (WAR).

If incoming command and existing commands have the same BT and completely the same address, this is handled as normal address collision.

13.1.1.12.6 Scheduling Mechanism Improvement for WR

Figure 13-18 shows new mechanism to minimize performance drop by inline ECC:

Figure 13-18 Scheduling Mechanism

- Schedule WR ECC command without dedicated PRE-ACT cycle as much as possible (*):
 - Before serving any page-miss command, WR ECC is scheduled out unless there is a WR Data command to the same page/bank.
 - Last WR Data burst for a block and corresponding WR ECC burst can be scheduled back to back (at least for BL16 case).
 - This is done by the following mechanism:
 - IH always generates WR ECC command immediately after WR Data command and pushes into WR ECC CAM so that the DDRCTL can schedule WR ECC command whenever needed (one WR ECC command pushed into CAM for one WR Data command).
 - WR ECC CAM combines WR ECC commands when incoming WR ECC command from IH and existing WR ECC command in WR ECC CAM have the same Block Token (BT) to minimize number of overhead WR ECC commands.
 - WR ECC command becomes candidate to be scheduled when all WR commands belonging to a block in CAM are served.

(*) The following are exceptions:

- Critical Refresh
- RD/WR switching due to starvation/q-empty
- CAM flushing due to collision
- High priority transaction: xVPR/xVPW/HPR
- PHY Master request
- If both WR ECC command and WR Data command are candidate to be scheduled for a bank, priority is the following:
 - a. Page-hit oldest WR ECC command (corresponding BT is already terminated/overridden)
 - b. Page-hit oldest WR Data command
 - c. Page-hit oldest WR ECC command (corresponding BT is not terminated/overridden)
 - d. Page-miss oldest WR ECC command (corresponding BT is already terminated/overridden)
 - e. Page-miss oldest WR Data command
 - f. Page-miss oldest WR ECC command (corresponding BT is not terminated/overridden)
- Bank Priority is the following:
 - Oldest bank for WR ECC (corresponding BT is already terminated/overridden)
 - Oldest bank for WR Data

- Therefore, as long as WR Data command arrives at CAM before serving WR ECC command, the WR ECC command is not scheduled out. Typically, the WR ECC command is scheduled out when the block channel is terminated or before serving page-miss WR Data command.

13.1.1.12.7 Performance

The idle command latency of the controller increases by one controller clock cycle with inline ECC.

- Full block write: No additional latency
 - ECC burst is appended after 8 data bursts, the write data response is generated normally after each data burst.
 - Back to back full write blocks can be streamed with 88.9% application port efficiency and 100% DDR efficiency (this assumes single block channel).
- For block read: BL/2 additional latency (4 DDR cycles for BL8, eight DDR cycles for BL16)
 - First, read the ECC burst, then read the eight consecutive data bursts. The first read command of the block channel incurs the latency.
 - Back to back full read blocks can be streamed with 88.9% application port efficiency and 100% DDR efficiency.

Table 13-5 Performance Indicator

Condition					Performance			Latency (first access)			
Read	DM	No. burst	BL	No. burst overhead	Disabled Inline ECC (DDR clock cycles)	Inline ECC (DDR clock cycles)	Efficiency %	Disabled Inline ECC (DDR clock cycles)	Inline ECC (DDR clock cycles)	Additional (DDR clock cycles)	
Block Read	8 burst data	X	8	8	1	32	36	88.89	RL	RL+BL/2	BL/2
	7 burst data	X	7	8	1	28	32	87.50	RL	RL+BL/2	BL/2
	6 burst data	X	6	8	1	24	28	85.71	RL	RL+BL/2	BL/2
	5 burst data	X	5	8	1	20	24	83.33	RL	RL+BL/2	BL/2
	4 burst data	X	4	8	1	16	20	80.00	RL	RL+BL/2	BL/2
	3 burst data	X	3	8	1	12	16	75.00	RL	RL+BL/2	BL/2
	2 burst data	X	2	8	1	8	12	66.67	RL	RL+BL/2	BL/2
	1 burst data	X	1	8	1	4	8	50.00	RL	RL+BL/2	BL/2

13.1.1.13 Address Protection with ECC (ECCAP)

- There is no CA parity protection in LPDDR4.
- Protect the address within the ECC scheme: when you do a read and if a read address is faulty, distinguish it from data error (when possible) and the report.
- Address flips during writes can be detected within certain delay (only when read back or if the scrubber is enabled, then within a certain time window). Address errors which may appear before the ECC are detected directly.
- SECEDED on 64-bit: total of 8-bit of ECC check bits:
 - SEC part is 7-bit and there is additional one bit “overall parity” to determine 2-bit failures.
 - “The overall parity” can only detect 1, 3, 5, and 7 bit errors. Even number of errors are not detected.
- “Address Parity” is:
 - Use {cs, row, ba/bg, col} addresses to calculate it.
 - RAS/CAS or other command bits cannot be protected.
- “Error Threshold” is:
 - A certain number of “overall parity” failures within one memory burst.

- Use 64-bit SECDED.
- Calculate the address parity and invert two bits of ECC check bits when address parity = 1.

In Table 13-6, “x” indicates one-bit flip on the address. There are three error types identified and not all address failures can be detected with 100% certainty.

If there is a refresh command in between RD (ECC) and RD (DATA), then there are additional ACT (DATA) and same error types still apply.

Table 13-6 Error Types

NO	Error types	1st command ACT		2nd command RD ECC		3rd command RD DAT		Result	Address Protection
		Row	Ba/Bg/CS	Col	Ba/Bg/CS	Col	Ba/Bg/CS		
1	error type 3	x						Wrong page opened, but data and ECC are matched	ECC decoder found 2 bit uncorrectable error in all beats of a burst due to address parity. 100% address error detection using error threshold method.
2	error type 1		x					Protocol issue because RD command is to a closed bank	Unpredictable behaviour (need to confirm with DRAM vendors). If there's no DQS toggling, controller will get stuck. If there is DQS toggling, then data and ecc mismatch will occur
3	error type 2			x				Data and ECC are not matched	ECC decoder does not have the capability to handle more than 2 errors. We rely on the overall parity. The overall parity cannot identify an address error by itself. Random number of errors are detected within a memory burst. Use an error threshold (say 4 errors out of 8 within a burst) to identify as address error - therefore address error detection is statistical.
4	error type 2				x			Data and ECC are not matched if the wrong bank is already opened	
5	error type 1				x			Protocol issue because RD command is to a closed bank	
6	error type 2					x		Data and ECC are not matched	
7	error type 2						x	Data and ECC are not matched if the wrong bank is already opened	
8	error type 1						x	Protocol issue because RD command is to a closed bank	
9	error type 3			x		x		Same column bit failing together and result data and ECC are matched	
10	error type 2			x		x		Column bits failing differently, Data and ECC are not matched	
11	error type 3				x		x	Bank bits failure together, Data and ECC are matched	
12	error type 2				x		x	Bank bits failing differently and both reads go to open banks. Data and ECC are not matched	
13	error type 1				x		x	Bank bits failing differently and protocol error if any RD command is to a closed page	

13.1.1.14 Scrubbing

For more information on scrubbing functionality in Inline ECC configurations, see “[Scrubber](#)” on page 301.

13.1.1.15 QoS

The DDRCTL supports all priorities with Inline ECC feature:

- VPW/VPR
- NPW/LPR/HPR

One block must have the same priority for reads because read command cannot be served until corresponding read ECC command is served. This is managed in DDRCTL internally as:

- If the first read in a block is HPR, read ECC command is also generated as HPR and call this HPR block so that read ECC command is executed as same priority.
- If the first read/RMW in a block is LPR/VPR, read ECC command is also generated as LPR/VPR and calls this LPR/VPR block so that read ECC command is executed as same priority.

- If HPR is incoming to an existing HPR block, nothing can be done.
- If VPR read/RMW is incoming to an existing LPR/VPR block and read ECC command is still in the CAM, the read ECC command's priority is updated to be the same priority as incoming VPR command only when the incoming VPR command is higher priority (that is, smaller VPR timeout) than current read ECC command.
 - Read ECC is LPR and incoming read is LPR, nothing can be done.
 - Read ECC is LPR and incoming read is VPR with timeout = X, the read ECC is updated to VPR with timeout X, including X = 0 case which is expired VPR.
 - Read ECC is VPR with timeout = Y and incoming read is LPR, nothing special can be done.
 - Read ECC is VPR with timeout = Y, incoming read is VPR with timeout = X and X < Y, the read ECC is updated to VPR with timeout X.
 - Read ECC is VPR with timeout = Y, incoming read is VPR with timeout = X and X ≥ Y, nothing special can be done.
- If HPR read is incoming to an existing LPR/VPR block channel (meaning same block address) and read ECC command is still in the CAM, this LPR/VPR block channel is terminated and new HPR block is started.

**Note**

- If the read ECC command is already served when the HPR read is incoming, the block is not terminated.
- If any write command is in the existing LPR/VPR block channel, the block is not terminated to avoid block address collision.

- If LPR/VPR read is incoming to an existing HPR block channel (meaning same block address) and read ECC command is still in the CAM, this HPR block channel is terminated and new VLPR/VPR block is started.

**Note**

- If the read ECC command is already served when the VPR/LPR read is incoming, the block is not terminated.
- If any write command is in the existing HPR block channel, the block is not terminated to avoid block address collision.

13.1.1.16 Features and Limitations

The following are the features and limitations of Inline ECC:

- MEMC_BURST_LENGTH: 16
- Data Mask (DM) must always be enabled
- MEMC_DRAM_DATA_WIDTH = 16, 32, 64.¹
- Write combine is not supported for the block access (that is, no block write combine).
- MEMC_USE_RMW = 1 is only.

1. MEMC_DRAM_DATA_WIDTH must be set to '16' or '32' in LPDDR5/4/4X Controller.

- Scrubber block is enhanced to issue RMW scrub command when there is a correctable error (UMCTL2_SBR_EN can be set to '1'). The scrubber is not sensitive to other error types and they gets ignored.
- ECC Data Poisoning is not supported (ECCCFG1.data_poison_en must be set to '0'). There is a software method to inject errors to ECC parity bits with inline ECC. For more details, see section "[Error Injection Through Software \(ECC Data Poisoning\)](#)" on page 279.
- OCPAR is supported (UMCTL2_OCPAR_EN can be set to '1').
- Dual HIF is not supported (UMCTL2_DUAL_HIF must be set to '0').
- Supported QoS features:
 - HPR/LPR is supported.
 - VPRW is supported (UMCTL2_VPRW_EN can be set to '1').
- Unaligned reads to memory burst is not supported at the HIF interface. Unaligned writes at HIF are supported.
 - XPI in arbiter configuration always generates aligned read bursts to HIF.
- MEMC_OPT_TIMING = 1 is only supported.
- Hardware Fast Frequency Change (HWFFC) is only supported (tested) for LPDDR4.
- HIF only configuration (UMCTL2_INCL_ARB == 0) is not supported.
- MEMC_ECC_SUPPORT must be set to '1' - Advanced ECC is not supported.
- Quarter Bus Width (MEMC_QBUS_SUPPORT) is supported as hardware parameter, but when Quarter Bus Width is used, Inline ECC must be disabled by software.
- Unused sub-regions in the ECC region can be utilized as non-protected region. The entire size of the unused sub-region depends on setting of ECCCFG0.ecc_region_map and ECCCFG0.ecc_region_map_other. However, by default, the unused sub-region are fragmented in terms of HIF/AXI address. Enabling ECC region remap feature (ECCCFG0.ecc_region_remap_en=1) makes the unused sub-regions consecutive, however the performance of sequential access for the unused region can be sub-optimal compare to normal unprotected regions. For more details, see "[ECC Region Remap](#)" on page 279.

**Note**

Inline ECC cannot be enabled in software, if QBW is selected.

13.1.1.16.1 Address Map Limitation

In case of MEMC_BURST_LENGTH=16, hif_cmd_addr[6:0] can be mapped to CS or ROW from address map perspective, but this is NOT supported when Inline ECC is enabled from performance consideration.

13.1.1.16.2 Prohibit RMW Command to Unprotected Regions and ECC Region

**Note**

Only for configurations with UMCTL2_INCL_ARB == 0, which are not supported in current version of LPDDR5/4 controller.

When inline ECC is enabled (ecc_mode = 3'b100), data mask must be enabled. There is no reason to use RMW for non-protected regions and ECC region (both used and unused sub-regions).

If you use RMW to non-protected region, DDRC could have wrong behaviors as described:

There are two commands who have same address excepting of column address bit3 and are going to unprotected region. The first command is RMW command whose column bit3 0. The second command is a write command which column bit3 is 1 and is a masked write.

The second write command is combined with write part of the first RMW, and then the combined write command becomes write command instead of Mask write.

But when inline ECC is enabled, read part of RMW is partial read. Only the first half of data are read (column 0~7), the second half of data is still masked. As a result, data mask is used for a write command in LPDDR4 protocol that violates LPDDR4 specification.

13.1.1.17 Interrupts Related to Inline ECC

The following are the interrupts for Inline ECC:

- `ecc_uncorrected_err_intr`
- `ecc_corrected_err_intr`
- `ecc_uncorrected_err_intr_fault`
- `ecc_corrected_err_intr_fault`
- `ecc_ap_err_intr_fault`
- `ecc_ap_err_intr`

13.1.1.18 Signals Related to Inline ECC

This section contains the following subsections:

- [“HIF Signals”](#) on page 296
- [“Credit Counters Signals”](#) on page 297

13.1.1.18.1 HIF Signals

The following signals are generated internally by the XPI (AXI Port Interface) in AXI configurations.

- `hif_cmd_ecc_region`: this signal indicates the command belongs to the ECC protected region.
- `hif_lpr_credit[1:0]`: this signal is expanded to 2 bits. Bit 0 indicates LPR credit increment pulse from the CAM as before when the oldest entry is scheduled; Bit 1 indicates LPR credit increment pulse from the IH whenever the command does not need to inject an overhead command. Both bits can be asserted at the same time, and in that case the HIF manager needs to increment credits by 2.
- `hif_hpr_credit[1:0]`: this signal is expanded to 2 bits. Bit 0 indicates HPR credit increment pulse from the CAM as before when the oldest entry is scheduled; Bit 1 indicates HPR credit increment pulse from the IH whenever the command does not need to inject an overhead command. Both bits can be asserted at the same time, and in that case the HIF master needs to increment credits by 2.
- `hif_wr_credit[0:0]`: this signal indicates that a write request (except WR ECC command generated internally for inline ECC) and write data have been scheduled to the DDR, or that a Write Combine has occurred, and the SoC must increment the credit value associated with writes.
- `hif_wrecc_credit[1:0]`: [Inline ECC only] this signal indicates that a write ECC request and write data generated internally have been scheduled to the DDR, or that a Write Combine has

occurred, and the SoC must increment the credit value associated with writes ECC. Bit[1] is dedicated for Address error (LPDDR4) request to protected region.

- `hif_rdata_corr_ecc_err`: signal to tell scrubber that read data has corrected error. Scrubber issues a masked RMW to correct it.
- `hif_cmd_wdata_mask_full`: width of this signal is equal to the maximum number of beats allowed for that configuration, which is $\text{MEMC_BURST_LENGTH} / (2 * \text{MEMC_FREQ_RATIO})$. Each bit identifies a data beat for that write command. If the bit is set to '1', all the bytes for that beat are meant to be written, so `hif_wdata_mask` for that beat must all be set to 1s. If a bit is 0, bytes are masked.
- `hif_cmd_length`: this signal indicates for both writes and reads if the access is full or partial (can be half or quarter). This information determines the number of HIF data beats required for one HIF command.

When the HIF master or PA sends a HIF command to ECC protected region, it needs to check and consume corresponding credits according to the [Table 13-7](#).

Table 13-7 Credits Consumed When the HIF Master or PA Sends a HIF Command

	Credit			
HIF command	LPR	HPR	WR Data	WR ECC
LPR/VPR	2			
HPR		2		
WR			1	1
RMW	2		1	1

If you send LPR or RMW to protected region, you must program `SCHED.lpr_num_entries > 0`; If you send HPR to protected region, you must program `SCHED.lpr_num_entries < MEMC_NO_OF_ENTRY-2`.



Note

For non-Inline-ECC configuration, PA asserts `hif_go2critical_wr` when WR CAM is full and PA has urgent request in its pipeline. For Inline ECC configuration, PA asserts `hif_go2critical_wr` when “WR Data CAM” or “WR ECC CAM” are full and has urgent request in its pipeline. Then DDRCTL moves to write mode and WR ECC entry and WR Data entry are served.

13.1.1.18.2 Credit Counters Signals

[Table 13-8](#) shows the credit counters signals.

Table 13-8 Credit Counters Signals

Name	Width	IO	From/To	Description
<code>wrecc_credit_cnt</code>	7	O	From PA	Number of available write ECC WR CAM slots (free positions). Each slot holds a DRAM burst Synchronous to controller clock (<code>core_ddrc_core_clk</code>). Value is decremented/incremented as the commands flow in out of the write ECC CAM. Exists: <code>UMCTL2_INCL_ARB</code> & <code>MEMC_INLINE_ECC</code>

13.1.1.19 Registers Related to Inline ECC

The following are the new registers added for Inline ECC.

- ECCCFG0:
 - `ecc_region_map_granu`
 - `ecc_ap_err_threshold`
 - `blk_channel_idle_time_x32`
 - `ecc_region_map`
 - `ecc_ap_en`
 - `ecc_mode`
 - `ecc_region_remap_en`
 - `ecc_region_map_other`
- ECCCFG1:
 - `active_blk_channel`
 - `blk_channel_active_term`
 - `ecc_region_waste_lock`
 - `ecc_region_parity_lock`
- ECCCTL:
 - `ecc_ap_err_intr_force`
 - `ecc_uncorrected_err_force`
 - `ecc_corrected_err_intr_force`
 - `ecc_ap_err_intr_en`
 - `ecc_uncorrected_err_intr_en`
 - `ecc_corrected_err_intr_en`
 - `ecc_ap_err_intr_clr`
 - `ecc_uncorr_err_cnt_clr`
 - `ecc_corr_err_cnt_clr`
 - `ecc_uncorrected_err_intr_clr`
 - `ecc_corr_err_intr_clr`
- ECCAPSTAT:
 - `ecc_ap_err`
- OPCTRLCAM1
 - `dbg_wrecc_q_depth`
- SCHED5
 - `wrecc_cam_highthresh`
 - `wrecc_cam_lowthresh`

The following registers are not used for Inline ECC:

- ECCCFG1
 - data_poison_bit
 - data_poison_en
- ECCPOISONADDR0/1
- ECCPOISONPAT0/1/2
- ADVECCINDEX

13.1.1.19.1 Differences in ECC Log Registers

The widths of `ECCSTAT.ecc_corrected_err` and `ECCSTAT.ecc_uncorrected_err` are different between Sideband ECC and Inline ECC mode.

- In Inline ECC mode, `ECCSTAT.ecc_corrected_err` and `ECCSTAT.ecc_uncorrected_err` are 1 bit.
- In Sideband ECC mode, `ECCSTAT.ecc_corrected_err` and `ECCSTAT.ecc_uncorrected_err` are 2 or 4 bits according to `MEMC_FREQ_RATIO`.

The column address definition is different between Sideband ECC and Inline ECC.

- In Sideband ECC, `ecc_corr_col` is based on DFI words and not on DRAM words. Following are the details:
 - `ecc_corr_col[0]` is always 1'b0. This means it cannot identify that error is ODD column or EVEN column. It is described by "lowest bit not assigned here".
 - `ecc_corr_col[1]` is always 1'b0 also in 1:2 mode. This means it cannot identify that error is in the lowest two column bits. It is not described in the databook.
 - In other words, current `ecc_corr_col` is based on DFI word. To identify the exact DRAM column address, use "`ECCSTAT.ecc_corrected_err`" which is 2 or 4 bits.
- In Inline ECC, the ECC code is always 64 bit, and could consist of one or two or four DRAM addresses. As it is not possible to comply with previous DFI words, the new definition of `ecc_corr_col[11:0]` is:
 - If the `MEMC_DRAM_DATA_WIDTH = 64`¹, `ecc_corr_col` identifies the exact DDR column address regardless of frequency ratio (1:1 or 1:2).
 - If the `MEMC_DRAM_DATA_WIDTH = 32`, `ecc_corr_col` identifies the first DDR column address in the ECC code (where ECC is comprised of 2 column addresses), regardless of frequency ratio (1:1 or 1:2).
 - If the `MEMC_DRAM_DATA_WIDTH = 16`, `ecc_corr_col` identifies the first DDR column address in the ECC code (where ECC is comprised of 4 column addresses), regardless of frequency ratio (1:1 or 1:2).

`ECCBITMASK` is always 72-bit wide and is accumulated at the ECC decoder.

13.1.1.19.2 Differences in the Programming Flow

1. For multi-port configurations, `PCTRL.port_en` is used to enable or disable the input traffic per port.
2. The controller idleness can be polled first from the `PSTAT` register (`wr_port_busy_n` and `rd_port_busy_n` bit fields) and must read as `PSTAT == 32'b0` (not busy).

1. `MEMC_DRAM_DATA_WIDTH` must be set to '16' or '32' in LPDDR5/4/4X Controller.

3. Write 0 to `SBRCTL.scrub_en` to disable the scrubber, SBR, (required only if SBR instantiated). Poll `SBRSTAT.scrub_busy == 0`, indicates that there are no outstanding SBR read commands (required only if SBR instantiated).
4. `OPCTRL1.dis_hif` is used to enable or disable the input traffic to the DDRC part of the controller. The change here is the requirement of asserting `dis_hif` even in AXI configurations. This ensures flushing of all ECC block channels.

DDRC CAM/pipeline empty status must be polled (`(OPCTRLCAM.dbg_wr_q_empty == 1'b1) && (OPCTRLCAM.dbg_rd_q_empty == 1'b1) && (OPCTRLCAM.wr_data_pipeline_empty == 1'b1) && (OPCTRLCAM.rd_data_pipeline_empty == 1'b1)`).

13.2 Scrubber

The DDRCTL provides a comprehensive solution to automatically correct single-bit errors in the DRAM. ECC Scrubber is a block capable of generating initialization write commands, periodic read commands, periodic RMW commands, and correction RMW commands. In Normal mode, ECC Scrubber issues periodic back ground read commands to the DDRC. Periodic scrubbing is performed by the ECC Scrubber block to increase the reliability of the system by correcting single-bit ECC errors in time, before they turn into uncorrectable 2-bit errors. Each scrub command is a read of one memory burst (for example, BL8 or BL16) which is sent to the DDRC periodically with the lowest priority. The data read sent back is terminated inside scrubber and is not returned to the system. The clock and reset to the SBR block (`sbr_clk` and `sbr_resetsn`) is separated from the controller clock and reset (`core_ddrc_core_clk` and `core_ddrc_rstn`) but must be synchronous with each other. In AXI configurations the SBR can continue to function: count and request exit from low-power even if the controller clock is gated.

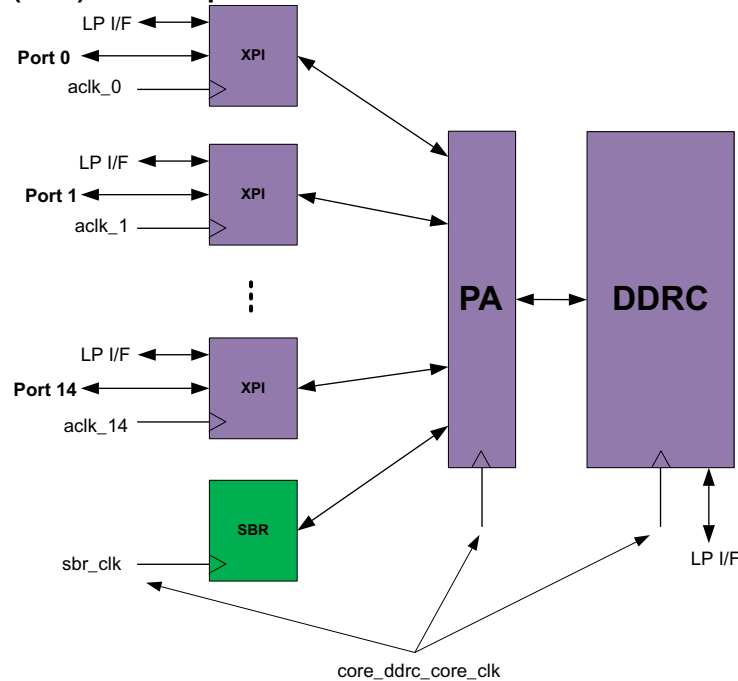
Scrubber can support three types of commands using the `SBRCTL.scrub_cmd_type` register field. The commands are periodic reads, initialization writes, and periodic RMW transactions:

- Periodic Reads are only for scrubbing. In these reads, if a correctable error is detected by the DDRCTL, then a new correction Read Modify Write will be generated.
- Initialization writes are only for memory initialization. These writes are issued back to back.
- Periodic RMWs can be used for scrubbing or for external encryption schemes.

To enable the 'ECC Scrubber', the following hardware parameters must be set to '1':

- `UMCTL2_SBR_EN`
- `UMCTL2_SYS_INTF > 0`
- `MEMC_USE_RMW`
- `MEMC_ECC_SUPPORT > 0`

Figure 13-19 shows the SBR block with respect to other controller functions. As shown in the figure, the SBR consumes one of the ports of the Port Arbiter (PA) as a requester. If the SBR is enabled, up to 15 application ports can be supported instead of 16.

Figure 13-19 ECC Scrubber (SBR) With Respect to Other Functional Blocks

In Inline ECC configurations, the SBR issues the periodic scrub read command and automatic RMW command upon detection of a single-bit ECC error. RMW command is initiated by the scrubber for every single-bit of the ECC error detected. ECC Scrubber (SBR) block is enhanced to issue fully masked (all bytes disabled) single read-modify-write command after detecting a correctable read error (a new HIF signal `hif_rdata_corr_ecc_err` is provided).

13.2.1 Scrub Period

Scrub interval (`tSCRUBI`) is defined as the number of clock cycles between two scrub read commands.

Scrub period is defined as the time needed to read every memory location of the entire SDRAM address range.

In Inline ECC configurations, the register `SBRCTL.scrub_interval[12:0]` counts in multiples of 512 `sbr_clk` cycles (see “REGB_ARB_PORTp” in “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide). Value of ‘1’ counts $512 * \text{sbr_clk}$ cycles, value of 2 counts $1024 * \text{sbr_clk}$ cycles in between each scrub command, and so on.

Value of ‘0’ is a special case where the scrub read commands are issued back-to-back.

In Inline-ECC configurations, whenever `SBRCTL.scrub_interval` is programmed to ‘0’, the scrubber does not issue any RMW command to DDRC if single bit ECC errors are detected.

In Inline ECC configurations, the internal `tSCRUBI` counter width is 22 bits due to the interval programming granularity being 9 bits ($\times 512$). This counter width (22 bits) is derived from the highest controller clock frequency of 1066 MHz, the longest scrub period of 12 hours, and the lowest memory size. The counter granularity (9 bits) is derived from the lowest controller clock frequency of 200 MHz, the shortest scrub period of 2 hours and the highest memory size. The scrub period ranges from 2 hours to 12 hours depending on the `tSCRUBI` setting, memory size, and clock frequency.



- In dual channel configurations, all register fields of SBRCTL except for SBRCTL.scrub_en and SBRCTL.scrub_en_dch1 are shared between both channels. SBRWDATA0, SBRWDATA1 are also shared.

13.2.2 Restrictions on Scrub Interval With Auto Power-down/Auto Self-refresh Idle Time

When Auto Power-down/Auto Self-refresh is enabled, it is required to make sure that the scrub interval (SBRCTL.scrub_interval) is greater than (actual time value must be greater) the Power-down idle time (PWRTMG.powerdown_to_x32) / Self-refresh idle time (PWRTMG.selfref_to_x32). Otherwise, scrubber may prevent the DDR memory from going to power-down/self-refresh by initiating scrubber periodic transactions before the idle-time expiry.

1. Power-down timer max = 1024 DRAM clock cycles - 256/512 Core clocks (based on DFI Frequency ratio 1:4/1:2)
2. Self-refresh max idle time = 8192 DRAM clock cycles - 2048/4096 Core clocks (based on DFI Frequency ratio 1:4/1:2)
3. Scrub interval granularity is 512 sbr_clk cycles in Inline ECC mode. Scrub interval=(SBRCTL.scrub_interval * 512) sbr_clk cycles. sbr_clk is the same as Core clock.
4. Based on the system's requirements, either PWRTMG.powerdown_to_x32 or PWRTMG.selfref_to_x32 can be fixed and SBRCTL.scrub_interval can be picked, or vice versa.

13.2.3 Normal Operation

The SBR is enabled by SBRCTL.scrub_en register (see “REGB_ARB_PORTp” in “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide). For Inline ECC a minimum of 8 bursts (1 block) is issued to the DDRC each time; the amount of burst sent is set by programming the SBRCTL.scrub_burst_length_nm[2:0] register. Each time tSCRUBI counter times out, a scrub command (or a burst of commands for Inline ECC) is pushed to a scrub queue (8-deep), which is then immediately issued to the DDRC. The SBR counters halt if more than 8 scrub commands are collected and pending inside the scrub queue due to DDRC being busy.

13.2.4 Address Configuration

When ECC Scrubber is enabled, the start and maximum addresses for which SBR generates commands can be programmed through the scrubber address registers.

The registers SBRSTART1.sbr_address_start_mask_1 and SBRSTART0.sbr_address_start_mask_0 have been provided to program the start address. Together, {SBRSTART1.sbr_address_start_mask_1, SBRSTART0.sbr_address_start_mask_0} is used by the SBR block internally to determine the start address.

The registers SBRRANGE1.sbr_address_range_mask_1 and SBRRANGE0.sbr_address_range_mask_0 have been provided to program the range mask. Together, {SBRRANGE1.sbr_address_range_mask_1, SBRRANGE0.sbr_address_range_mask_0} is used by the SBR block internally to determine the maximum address.

A second set of four registers is defined for dual-channel configurations for the CH1 SBR block. The registers are:

- SBRSTART0DCH1.sbr_address_start_mask_dch1_0

- `SBRSTART1DCH1.sbr_address_start_mask_dch1_1`
- `SBRRANGE0DCH1.sbr_address_range_mask_dch1_0`
- `SBRRANGE1DCH1.sbr_address_range_mask_dch1_1`

The scrubber address registers are changed only when the scrubber is disabled (`SBRCTL.scrub_en = 0`) and there are no scrubber commands in progress (`SBRSTAT.scrub_busy = 0`). For more information, see “REGB_ARB_PORTp” in “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

In Inline ECC configurations,

- The registers `SBRSTART0.sbr_address_start_mask_0`, `SBRSTART1.sbr_address_start_mask_1`, `SBRRANGE0.sbr_address_range_mask_0`, and `SBRRANGE1.sbr_address_range_mask_1` must always be set in the protected region based on the `ECCCFG0.ecc_region_map`, `ECCCFG0.ecc_region_map_granu`, and `ECCCFG0.ecc_region_map_other`.
- The minimum value for `{SBRSTART1.sbr_address_start_mask_1, SBRSTART0.sbr_address_start_mask_0}` can be '0', provided this address is in the protected region.
- The maximum value for `{SBRRANGE1.sbr_address_range_mask_1, SBRRANGE0.sbr_address_range_mask_0}` can be 'ECC region start address - 1', provided this address is in the protected region.
- For more information on the protected regions, see “[Selectable Protected Regions](#)” on page 275.



Note

The programming guidelines for `SBRSTART0.sbr_address_start_mask_0`, `SBRSTART1.sbr_address_start_mask_1`, `SBRRANGE0.sbr_address_range_mask_0`, and `SBRRANGE1.sbr_address_range_mask_1` are also applicable to `SBRSTART0DCH1.sbr_address_start_mask_dch1_0`, `SBRSTART1DCH1.sbr_address_start_mask_dch1_1`, `SBRRANGE0DCH1.sbr_address_range_mask_dch1_0`, and `SBRRANGE1DCH1.sbr_address_range_mask_dch1_1` respectively. When these registers are being programmed, the memory capacity of CH1 SDRAM is taken into consideration.

13.2.5 Address Generation

In Inline ECC configurations, ECC Scrubber (SBR) generates addresses only within protected regions, it automatically skips unprotected regions and ECC region.

- If there are no protected regions within start address `{SBRSTART1.sbr_address_start_mask_1, SBRSTART0.sbr_address_start_mask_0}` and end address `{SBRRANGE1.sbr_address_range_mask_1, SBRRANGE0.sbr_address_range_mask_0}`, SBR does not generate any command.
- If `SBRRANGE0.sbr_address_range_mask_0` or `SBRRANGE1.sbr_address_range_mask_1` or `SBRSTART0.sbr_address_start_mask_0`, or `SBRSTART1.sbr_address_start_mask_1` is programmed to values in the unprotected region or in the ECC region, then during reads or writes, ECC Scrubber will continuously loop through valid addresses.

For example, if `{SBRRANGE1.sbr_address_range_mask_1, SBRRANGE0.sbr_address_range_mask_0}` is set to $2^n - 1$ where n is `MEMC_HIF_ADDR_WIDTH`, this

address is in ECC region, for back to back scrubber reads or initialization writes ($t_{\text{SCRUBI}} = 0$), ECC Scrubber loops continuously through valid addresses.

13.2.6 Additional Note

- When `ADDRMAP12.nonbinary_device_density` is set to 3'b001 or 3'b010 or 3'b011 or 3'b100 or 3'b101, any write or RMW request with row `[MSB: MSB-1] == 2'b11` is discarded, while the HIF output `hif_wdata_ptr_addr_err` is asserted. Also, any read request with row `[MSB: MSB-1] == 2'b11` is executed, by changing the row `[MSB: MSB-1]` from 2'b11 to 2'b10. The return data for such reads are masked to zeros, while the HIF output `hif_rdata_addr_err` is asserted.
 - The ECC Scrubber (SBR) does not send transactions to the invalid addresses, but skips to the next valid address in the next cycle.
 - The signals `SBRRANGE0.sbr_address_range_mask_0`, `SBRRANGE1.sbr_address_range_mask_1`, `SBRSTART0.sbr_address_start_mask_0`, and `SBRSTART1.sbr_address_start_mask_1` must be set only to the valid addresses.
- In Inline ECC configurations, if a Scrubber periodic read transaction has both correctable and uncorrectable errors on different lanes, the controller generates a correction RMW issued to that address. This RMW will only correct the correctable error in the data. The uncorrectable error will be retained by corrupting the ECC of that data.
- In dual channel configurations with `UMCTL2_DUAL_CHANNEL = 1` & `UMCTL2_DATA_CHANNEL_INTERLEAVE_EN = 1`, there are two instances of scrubber. Both scrubber instances can be enabled and run in parallel. At the Port Arbiter, both scrubber modules can request a grant at the same time.
 - The registers `SBRSTART0`, `SBRSTART1` and `SBRSTART0DCH1`, `SBRSTART1DCH1` can have the same or different values respectively. Similarly, the registers `SBRRANGE0`, `SBRRANGE1` and `SBRRANGE0DCH1`, `SBRRANGE1DCH1` can have the same or different values.
 - To cover the entire address space, the start (`SBRSTART0`, `SBRSTART1` and `SBRSTART0DCH1`, `SBRSTART1DCH1`) and range (`SBRRANGE0`, `SBRRANGE1` and `SBRRANGE0DCH1`, `SBRRANGE1DCH1`) registers must be programmed to start and end of the HIF address space.
 - Internally each Scrubber instance will manipulate the HIF address to issue transactions only to the respective DRAM channel. Each Scrubber instance will use the HIF address bit mapped to the DCH interleave bit `ADDRMAP0.addrmap_dch_bit0` and then decide if it will issue the transaction to a particular address.

13.2.7 Status

If `SBRCTL.scrub_interval` is set to non-zero value and there are outstanding read commands on flight issued by the SBR, `SBRSTAT.scrub_busy` register is asserted. `SBRSTAT.scrub_done` is not asserted.

If `SBRCTL.scrub_interval` is set to '0',

- For reads, the ECC Scrubber continues to issue read commands and does not stop automatically after the full address range is covered. After the first round, the scrubber continues, and the controller status reads as `SBRSTAT.scrub_done = 1` and `SBRSTAT.scrub_busy = 1`.
- For writes, the ECC Scrubber stops issuing write commands after the full address range is covered. After the first round, the scrubber stops, and the controller status reads as `SBRSTAT.scrub_done = 1` and `SBRSTAT.scrub_busy = 0`.

**Note**

In Inline ECC configurations:

- If {SBRANGE1.sbr_address_range_mask_1, SBRANGE0.sbr_address_range_mask_0} is set in the unprotected region or in the ECC region and SBRCTL.scrub_interval is 0, then during reads or writes, ECC Scrubber will not generate SBRSTAT.scrub_done status.
- If {SBRSTART1.sbr_address_start_mask_1, SBRSTART0.sbr_address_start_mask_0} or {SBRANGE1.sbr_address_range_mask_1, SBRANGE0.sbr_address_range_mask_0} is set in the unprotected regions, the DDRCTL behavior is not predictable.
- For example, if {SBRANGE1.sbr_address_range_mask_1, SBRANGE0.sbr_address_range_mask_0} is set to $2^n - 1$ where n is MEMC_HIF_ADDR_WIDTH, this address is in ECC region, for back to back scrubber reads or initialization writes, the ECC scrubber will not generate SBRSTAT.scrub_done.

13.2.8 Hardware Controlled Low-Power Operation

If the DDRCTL is in one of the hardware controlled low-power modes, the SBR continues to operate automatically without any software intervention. This behavior can be disabled if SBRCTL.scrub_during_lowpower is set to '0' (see "REGB_ARB_PORTp" in "Register Descriptions" chapter of the DWC DDRCTL LPDDR5/4 Programming Guide). There are two such modes – automatically initiated by idleness and initiated by the hardware low-power interface.

The SBRCTL.scrub_burst_length_lp[2:0] register is specified to determine the number of back-to-back scrub commands, up to 1024, that can be issued together during low-power operation. In Sideband ECC configurations during normal operation mode of the controller (not power-down or self-refresh modes), scrub_burst_length_nm must be programmed to '1' and only one scrub command is generated at every scrub interval.

- **Automatically Initiated Low-Power Mode:** if SBRCTL.scrub_during_lowpower is set to '1', scrub_burst_length_lp number of scrub reads are issued every (scrub_burst_length_lp * scrub_interval) cycles when the controller is in any of the automated low-power modes (STAT.operating_mode = 10 (power-down) or 11 (self-refresh)). If in self-refresh, STAT.selfref_type must also be set to 11; then issuing of the scrub read automatically wakes up the controller. Note that the SBR cannot put the SDRAMs back to low-power; if the idleness condition continues, the automated low-power entry is automatically initiated again.
- **Low-Power Interface Initiated Low-Power Mode:** the DDRCTL can be put into self-refresh through the hardware low-power interfaces. If the low-power mode is entered through this interface, it has to be exited by the same way. If scrub_during_lowpower is set to '1', every (scrub_burst_length_lp * scrub_interval) cycles scrub_burst_length_lp number of scrub reads are attempted by asserting cactive_in_ddrc signal which in turn asserts cactive_ddrc output signal. The SBR is one of the several possible reasons for the DDRCTL to request exit from this low-power state by asserting cactive_ddrc. It is assumed that the external low-power controller initiates the exit upon receiving cactive_ddrc high.

13.2.9 Software Controlled Low-Power Operation

If the DDRCTL is in one of the software controlled low-power modes such as self-refresh power-down, it is recommended to follow the following procedure:

- Before entering into any of the software controlled low-power modes, disable the SBR by setting `SBRCTL.scrub_en` to '0' (see "REGB_ARB_PORTp" in "Register Descriptions" chapter of the DWC DDRCTL LPDDR5/4 Programming Guide).
- If scrubbing is required in this low-power mode, the whole address range can be quickly scrubbed by issuing back-to-back scrub read commands.
 - Software must wake up the DDRCTL and initialize the SDRAM/PHY as necessary.
 - To start the SBR in an accelerated rate, set `scrub_interval` to '0' and enable the SBR.
 - Once the SBR scrubs the entire SDRAM range, `sbr_done_intr` interrupt signal is asserted. In addition, `SBRSTAT.scrub_done` software status bit is set which can be polled by the software. The scrubbing operation continues until SBR is disabled. Note that `sbr_done_intr` signal and `scrub_done` register are never set if the SBR is operating in its normal mode (`scrub_interval > 0`). System can monitor `sbr_done_intr` interrupt to be set or poll the `SBRSTAT.scrub_done` bit to be 1.
 - If the software does not want to wait for the complete scrubbing operation to finish, `scrub_interval` can be set to a non-zero value anytime. The SBR re-starts the scrub commands from the start address.
 - Disabling the SBR or setting the `scrub_interval` back to its normal operating value (`tSCRUBI`) which is greater than 0 clears both the interrupt signal and the status bit.
- After software controlled self-refresh is disabled, software must poll the `STAT.operating_mode`. Once `STAT.operating_mode` changes from self-refresh to normal, then Scrubber can be enabled.

13.2.10 Initialization Writes

The previous sections refer to scrub read commands. The SBR block can be used to initialize the memories with a defined pattern. `SBRCTL.scrub_cmd_type` must be programmed to '1' (see "REGB_ARB_PORTp" in "Register Descriptions" chapter of the DWC DDRCTL LPDDR5/4 Programming Guide).

`SBRWDATA0` and `SBRWDATA1` registers (exists if `MEMC_DRAM_DATA_WIDTH == 641`) can be programmed with the desired pattern.

Scrubber gets its write data pattern from these `SBRWDATA0` and `SBRWDATA1` registers.

Since the data in these registers represents the DRAM Width data, software must ensure that in HBW only the lower half of the data has valid bytes and disabled upper half is zero padded.

Similarly in quarter bus width mode, software must ensure only the lower quarter bytes have valid data and the disabled bytes are zero padded.

In Inline ECC configurations, the registers `{SBRSTART1.sbr_address_start_mask_1, SBRSTART0.sbr_address_start_mask_0}` and `{SBRRANGE1.sbr_address_range_mask_1, SBRRANGE0.sbr_address_range_mask_0}` must be programmed in the protected region based on the `ECCCFG0.ecc_region_map`, `ECCCFG0.ecc_region_map_granu`, and `ECCCFG0.ecc_region_map_other`.

1. `MEMC_DRAM_DATA_WIDTH` must be set to '16' or '32' in LPDDR5/4/4X Controller.

- The minimum value for {SBRSTART1.sbr_address_start_mask_1, SBRSTART0.sbr_address_start_mask_0} can be 0, provided this address is in the protected region.
- The maximum value for {SBRRANGE1.sbr_address_range_mask_1, SBRRANGE0.sbr_address_range_mask_0} can be 'ECC region start address - 1', provided this address is in protected region.
- For more information on the definition of protected regions, see “[Selectable Protected Regions](#)” on page 275.

After PHY and SDRAM initialization, perform the following steps:

1. If MEMC_INLINE_ECC is defined and ECC is enabled, set ECCCFG1.ecc_parity_region_lock to '1'.
2. Ensure that PCTRL.port_en are set to '0', block the AXI ports from taking the transaction.
3. If UMCTL2_OCECC_EN is defined and On-chip ECC is enabled (OCECCCFG0.ocecc_en = 1), set OCECCCFG0.ocecc_en to '0'. This is required to avoid On-chip ECC error interrupt. For more information, refer to “[On-Chip ECC \(OCECC\)](#)” on page 316.
4. Disable Auto power-down and Auto self-refresh if it is enabled (PWRCTL.powerdown_en = 0 and PWRCTL.selfref_en = 0)
5. Set SBRCTL.scrub_cmd_type = 1.
6. Set SBRCTL.scrub_interval = 0.
7. Set the desired pattern through SBRWDATA0 and SBRWDATA1 registers.
8. Enable the SBR by programming SBRCTL.scrub_en = 1.
9. System can monitor sbr_done_intr interrupt to be set or poll the SBRSTAT.scrub_done bit to be 1.
10. Poll SBRSTAT.scrub_busy = 0 (all scrub writes data have been sent).
11. Disable SBR by programming SBRCTL.scrub_en = 0.
12. Enable Auto power-down and Auto self-refresh if required (PWRCTL.powerdown_en = 1 and PWRCTL.selfref_en = 1).
13. If UMCTL2_OCECC_EN is defined, enable On-chip ECC by programming OCECCCFG0.ocecc_en to '1'.
14. Prepare for normal scrub operation (Reads) by programming SBRCTL.scrub_cmd_type = 0 and SBRCTL.scrub_interval to tSCRUBI.
15. Enable the SBR by programming SBRCTL.scrub_en = 1.
16. Enable AXI ports by programming PCTRL.port_en = 1.



Note

You must have valid data and ECC data in the SDRAM before putting the scrubber in Read mode.



Note

SBRSTAT.scrub_done/sbr_done_intr is generated when the initialization write is completed to the scrubber end address (default end address or end address as defined by SBRRANGE1.sbr_address_range_mask_1, and SBRRANGE0.sbr_address_range_mask_0). If scrubber cannot write to this end address, scrub_done will not be asserted. You need to make sure that the end address is a valid address where scrubber can execute a write.

13.3 On-Chip Parity (OCPAR)

This topic contains the following sections:

- [“Overview of On-Chip Parity”](#) on page 309
- [“Enabling On-Chip Parity”](#) on page 310
- [“Write Data Parity”](#) on page 310
- [“Read Data Parity”](#) on page 311
- [“Read Modify Write Operation”](#) on page 313
- [“Signals Related to On-Chip Parity”](#) on page 314
- [“Registers Related to On-Chip Parity”](#) on page 315

13.3.1 Overview of On-Chip Parity

In the DDRCTL, all data paths from end-to-end are protected using parity. Address path protection is also provided at the AXI inputs. Parity has the capability to detect an odd number of errors. Once error is detected, the mechanism can neither locate nor correct the errors.

Though ECC and parity are connected to create an overlapped protection, they can be enabled independently. However, if parity is enabled and ECC is disabled, some of the parity functionality may not be available.

Parity functionality is supported only for AXI configurations.

Figure 13-20 On-Chip Parity Overview in Sideband ECC or ECC Disabled Configurations

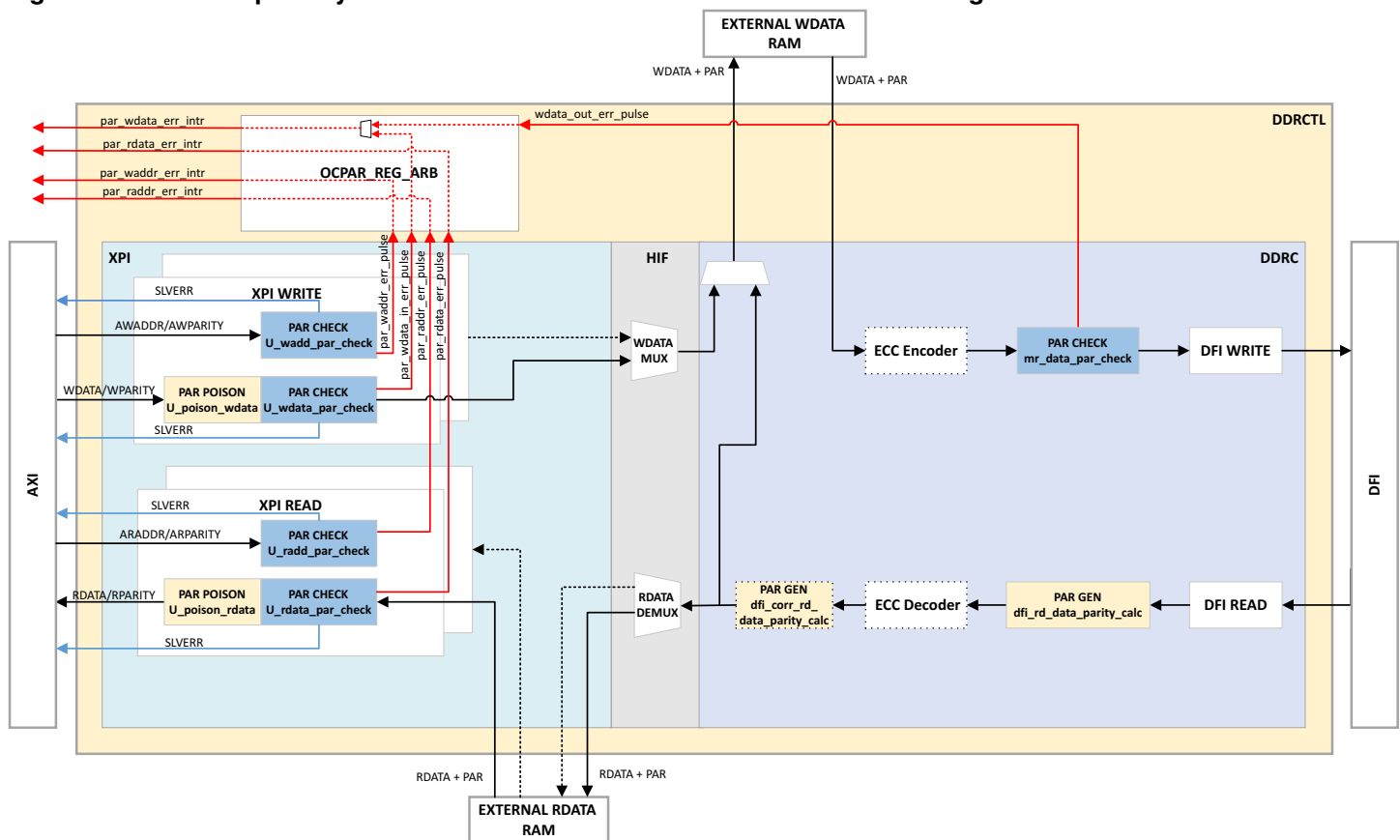


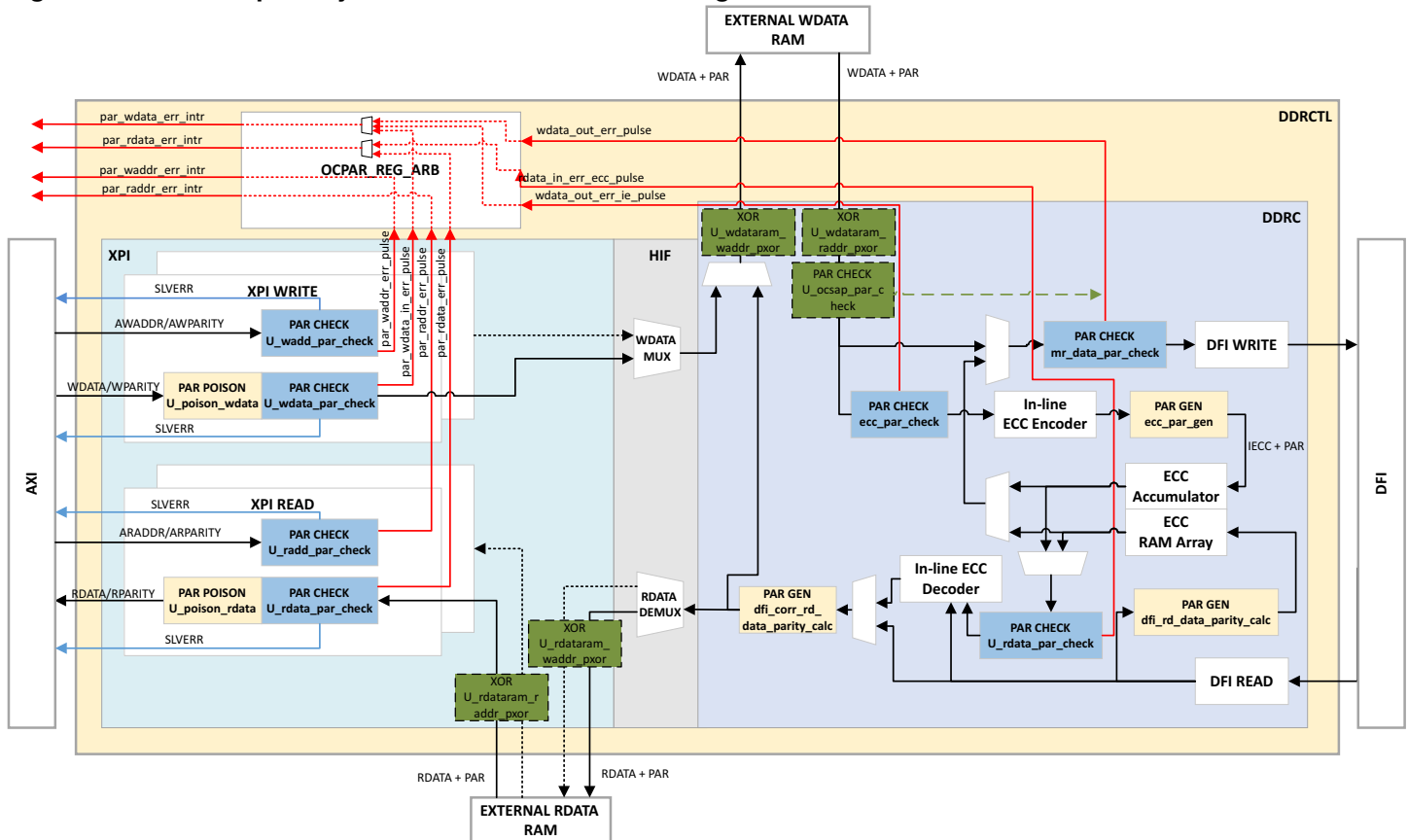
Figure 13-20 shows the on-chip parity architecture in sideband ECC configurations ($\text{MEMC_ECC_SUPPORT} > 0$ and $\text{MEMC_SIDEBAND_ECC} = 1$) or configurations without ECC support ($\text{MEMC_ECC_SUPPORT} = 0$).

**Note**

The blocks in dotted line are only present in ECC configurations (that is, $\text{MEMC_ECC_SUPPORT} > 0$).

Figure 13-21 shows the on-chip parity architecture in Inline ECC configurations ($\text{MEMC_INLINE_ECC} = 1$).

Figure 13-21 On-Chip Parity Overview in Inline ECC Configurations

**Note**

In Figure 13-20 and Figure 13-21, Parity Poisoning inside DDRC block is present after every PAR GEN blocks.

The highlighted blocks in green are present when $\text{DDRCTL_OCSAP_EN} = 1$.

13.3.2 Enabling On-Chip Parity

To enable on-chip parity, set the hardware configuration parameter UMCTL2_OCPAR_EN to '1'.

13.3.3 Write Data Parity

For every write data beat, the DDRCTL receives one bit of parity per byte. Write data parity on AXI interface is carried through a sideband signal called $\text{wparity_n}[\text{UMCTL2_PORT_NBYTES_n-1:0}]$.

Parity is checked at two places: after the AXI interface and just before the DFI interface and after the ECC encoder. The former checker can be bypassed if `OPARCFG0.par_wdata_axi_check_bypass_en` is asserted

When write data parity errors are detected, they generate maskable interrupt signal `par_wdata_err_intr`, that can be disabled with `OPARCFG0.par_wdata_err_intr_en`. `OPARSTAT1.par_wdata_err_intr[15:0]` register logs AXI port/ports on which parity error is detected. `OPARSTAT2.par_wdata_out_err_intr[0]` register logs if error detected on DFI. Additional check at the DFI is present in case Inline ECC is used (`MEMC_INLINE_ECC=1`), this protects the ECC path before the encoder; in case a parity error is detected, interrupt `par_wdata_err_intr_n` is asserted. `OPARSTAT2.par_wdata_out_err_intr[1:0]` register logs whether the error came from the data path (bit 0) and/or the ECC path (bit 1).

Write transactions with parity errors on the AXI interface are returned with a SLVERR response on AXI write response channel. Note that the DDRCTL treats all AXI transactions as bufferable regardless of the transaction attribute. In other words, if parity checker after the AXI interface is not bypassed (that is, if `OPARCFG0.par_wdata_axi_check_bypass_en=0`), the write response is generated when the last write data beat is accepted inside the DDRC CAM. Therefore, AXI SLVERR write response is generated only based on parity checking on the AXI interface. Any parity check failure detected on the DFI interface is not signaled through SLVERR but signaled through the `par_wdata_err_intr` interrupt signal that can be disabled through the register `OPARCFG0.par_wdata_err_intr_en`. The SLVERR response can be disabled through `OPARCFG0.par_wdata_slvrr_en`.

If ECC is enabled and there is a write data parity error on the DFI interface, the ECC check bits are corrupted for that write data beat. Note that write address is committed to the memory and write data is not masked if there is a parity error. It is up to the system through software to retry and write the correct data.

13.3.3.1 Output Check

The protection schemes, ECC and parity overlap, in other words, linked together. ECC is first calculated, and then the parity is terminated. To ensure this overlap, ECC encoder and the parity check are separated by a pipeline.

13.3.3.2 Input Check

The input parity check is done at the AXI interface, when both `wvalid_n` and `wready_n` is active. This provides data protection for all the upstream logic outside the controller. The status of the check is sent along with the write pointer and used for the response generation. As parity is not terminated, one bit per byte of data is appended for each beat.

13.3.4 Read Data Parity

For every read data beat, the DDRCTL generates one bit of parity per byte. Read data parity on AXI interface is carried through a sideband signal called `rparity_n[UMCTL2_PORT_NBYTES_n-1:0]`.

Parity is checked at the AXI interface. Read transactions with read data parity errors are returned with a SLVERR response which can be disabled using `OPARCFG0.par_rdata_slvrr_en`.

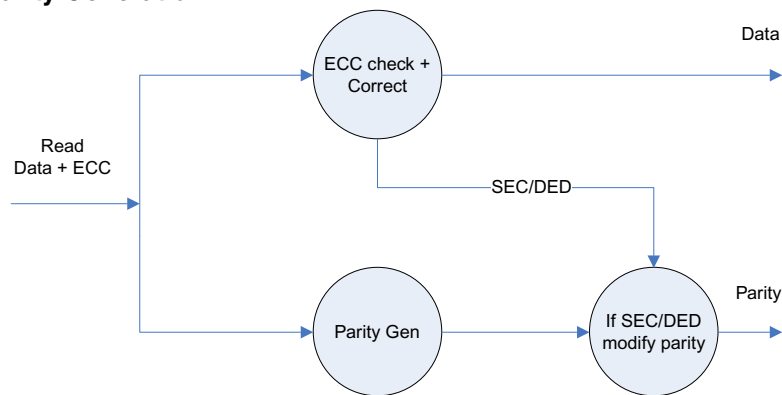
In addition to the error response, read data parity errors when detected, generate a maskable interrupt (`par_rdata_err_intr`), that can be disabled through the register

OCPARCFG0.par_rdata_err_intr_en. The OCPARSTAT1.par_rdata_err_intr_n registers log the AXI ports on which parity error is detected.

13.3.4.1 Parity Generation

Read data parity is generated from the same pipeline stage as the ECC decoder. As shown in Figure 13-22, the paths are protected by overlapping ECC and parity scheme.

Figure 13-22 Read Data Parity Generation



13.3.4.2 Parity Check

Parity check is done at the AXI interface when rvalid_n is active.

If inline ECC is used, additional check is done at the DFI, this is done before the ECC decoder. This is to protect the ECC word which comes separately from the data. In case an error is detected, interrupt par_rdata_err_intr is asserted and OCPARSTAT2.par_rdata_in_err_ecc_intr register is updated.

13.3.4.3 Address Parity

The AXI address (araddr, awaddr) is protected by one bit of parity for the entire address or by one bit of parity for each address byte, depending on hardware parameter UMCTL2_OCPAR_ADDR_PARITY_WIDTH setting. Address parity is carried by a sideband signal arparity/awparity. The address parity is checked and terminated within the XPI. Write address parity check is performed when awready_n == 1 and awvalid == 1. Likewise, read address parity check is performed when arready_n == 1 and arvalid_n == 1. Address parity errors are signaled to the master through AXI SLVERR response which can be disabled by OCPARCFG0.par_addr_slvrr_en register.

In addition to the error response, address parity errors generate maskable interrupts par_waddr_err_intr and par_raddr_err_intr respectively, for write address and read address errors. Interrupts can be disabled through registers OCPARCFG0.par_waddr_err_intr_end and OCPARCFG0.par_raddr_err_intr_en.

This feature uses the same infrastructure as “[Transaction Poisoning](#)” on page 56. Therefore, addresses with parity errors are effectively poisoned leading to similar operation.

Address parity errors are signaled through maskable interrupts (par_wdata_err_intr, par_raddr_err_intr), that can be disabled with OCPARCFG0.par_waddr_err_intr_en /

OCPARCFG0.par_raddr_err_intr_en. OCPARSTAT0.par_waddr_err_intr[15:0] and OCPARSTAT0.par_raddr_err_intr[15:0] register logs AXI port/ports on which address parity error is detected.

13.3.4.4 Embedded SRAM Protection

External SRAMs interfaces have data and byte parity.

Output parity is driven by the controller along with the data going to the external SRAMs (there is no checker inside the controller for these outputs):

- wdataram_din_par: parity for wdataram_din (write data RAM input, valid bytes according to wdataram_mask)
- rdataram_din_par_n: parity for rdataram_din_n (port n read data RAM input, all bytes valid)

You must drive the input parities to the controller based on the data read from each SRAM (controller expects the correct parity for each data byte whenever the data read is valid):

- wdataram_dout_par: parity for wdataram_dout (write data RAM output)
- rdataram_dout_par_n: parity for rdataram_dout_n (port n read data RAM output)

Note that rdataram_din_par_n and rdataram_dout_par_n are available only if the external RAM for port n is enabled; otherwise the parity is buffered inside the RRB (internal) along with the read data. In this case, you need not to perform any action.

13.3.5 Read Modify Write Operation

Read modify write operations make use of the read path and write path. As the read and write paths are properly protected, RMW operations are properly protected. In case there is an uncorrectable ECC error on the read part, the parity is corrupted (based on register setting) on the read data path. This leads to automatic ECC corruption on the write.

If parity is not enabled, controller still corrupts the write ECC for RMW directly upon detection of uncorrectable error on the read part of the transaction.

13.3.6 Parity Poisoning

The DDRCTL provides the ability to inject parity errors in the data path. This on-chip parity poisoning can be used to test the feature and system response to parity errors.

This functionality is enabled by setting the register OCPARCFG1.par_poison_en. The parity error can be injected in the following data interfaces with any combination:

- Read Data - Parity error is injected after the generation at the DFI interface enabled by OCPARCFG1.par_poison_loc_rd_dfi register. If Inline ECC is enabled, in the read data path, after the parity generation, data can go through two different paths (data or ECC). Setting the register OCPARCFG1.par_poison_loc_rd_iecc_type gives the possibility to poison either one or the other.
- Read Data - Parity error is injected at the AXI interface after the check, only one port at a time, enabled by OCPARCFG1.par_poison_loc_rd_port register. An error injected here is not captured by the controller.
- Write Data - Parity error is injected at the AXI interface before the check, only one port at a time, enabled by OCPARCFG1.par_poison_loc_wr_port register, if AXI checker is not bypassed

(OCPARCFG0.par_wdata_axi_check_bypass_en == 0). If AXI checker is bypassed (OCPARCFG0.par_wdata_axi_check_bypass_en == 1), no parity error will be injected at the AXI interface.

One-shot triggering is supported, which means that error is injected for the first valid data either after OCPARCFG1.par_poison_en is set to '1' or after the interrupt clear for that path is asserted. This means that only the parity corresponding to the first beat of the transaction on DFI/AXI is poisoned.

- OCPARCFG0.par_wdata_err_intr_clr to be asserted to re-enable poison at the Write Data AXI interface.
- OCPARCFG0.par_rdata_err_intr_clr to be asserted to re-enable poison at the Read Data DFI and AXI interfaces.

AXI traffic restrictions for on-chip parity poisoning:

- AXI burst type must be INCR.
- AXI address must be aligned to SDRAM burst (For more information on AXI to SDRAM address translation, see “[Address Mapping](#)” on page 119).
- AXI address must not access invalid LPDDR4 row address (For more information, see “[Non-binary Device Densities](#)” on page 128).
- AXI Write burst must not cause RMW on to HIF. To avoid RMW generation for AXI Write burst, see “[Read-Modify-Write \(RMW\) Generation](#)” on page 46.
- AXI address must be aligned to INLINE_ECC block boundary (When MEMC_INLINE_ECC == 1. For more details on Inline ECC block boundary, see “[Inline ECC Support](#)” on page 268).
- One AXI transaction must not access multiple ECC regions/blocks (when MEMC_INLINE_ECC == 1. For more details about Inline ECC regions/block, see “[Inline ECC Support](#)” on page 268).
- In Inline ECC configurations, for poisoning inside DDRC block, the AXI must perform read accesses to all the locations that are written by AXI from the time poisoning has been enabled.



Note

Note that all the AXI traffic restrictions apply only if the controller works in the full bus width mode, done by appropriately setting MSTR0.data_bus_width (see “[REGB_DDRC_CH0 Registers](#)” in the “[Register Descriptions](#)” chapter of the [DWC DDRCTL LPDDR5/4 Programming Guide](#)).

All automatic MRR command shall be disabled before enabling for on-chip parity poisoning as follows:

- Disable DQS oscillator by setting DQSOSCCTL0.dqsosc_enable=0 (see “[Controller Assisted Drift Tracking \(LPDDR5\)](#)” on page 198).
- Disable or Pause Automatic Temperature Derating by setting DERATECTL0.derate_enable=0 or DERATECTL0.derate_mr4_pause_fc=1 (see “[Automatic Temperature Derating](#)” on page 186).

13.3.7 Signals Related to On-Chip Parity

For more information on signals related to the on-chip parity, see:

- “[AXI Port n Write Address On-Chip Parity Signals \(for n = 0; n <= UMCTL2_A_NPORTS-1\)](#)” on page 384
- “[AXI Port n Write Data On-Chip Parity Signals \(for n = 0; n <= UMCTL2_A_NPORTS-1\)](#)” on page 387

- [“AXI Port n Read Address On-Chip Parity Signals \(for n = 0; n <= UMCTL2_A_NPORTS-1\)”](#) on page 395
- [“AXI Port n Read Data On-Chip Parity Signals \(for n = 0; n <= UMCTL2_A_NPORTS-1\)”](#) on page 398

13.3.8 Registers Related to On-Chip Parity

The following are the registers related to the on-chip parity:

- OCPARCFG0
- OCPARCFG1
- OCPARSTAT0
- OCPARSTAT1
- OCPARSTAT2

For more information about these registers, see the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

13.4 On-Chip ECC (OCECC)

**Note**

This feature is currently not supported. The information is provided for reference only.

This topic contains the following sections:

- [“Overview of On-chip ECC”](#) on page 316
- [“Enabling On-chip ECC Support”](#) on page 317
- [“ECC Encoding”](#) on page 317
- [“Write Data Protection”](#) on page 318
- [“Read Data Protection”](#) on page 318
- [“Error Detection”](#) on page 318
- [“Address Parity”](#) on page 319
- [“Embedded SRAM Protection”](#) on page 319
- [“ECC/Parity Poisoning”](#) on page 320
- [“Signals Related to On-chip ECC”](#) on page 321
- [“Registers Related to On-chip ECC”](#) on page 321

13.4.1 Overview of On-chip ECC

On-chip ECC is an alternative to on-chip parity for the data path, which is described in the section [“On-Chip Parity \(OCPAR\)”](#) on page 309.

This feature is only available with an Automotive Product (AP) License.

In On-chip ECC, the parity protection is replaced in some parts of the data path with SECDED ECC. The hardware support for the On-chip Address Parity feature is automatically enabled along with On-chip ECC. For more details, see [“Address Parity”](#) on page 312.

The data path is protected with a mix of 64/8 ECC, 8/5 ECC, and per-byte even parity.

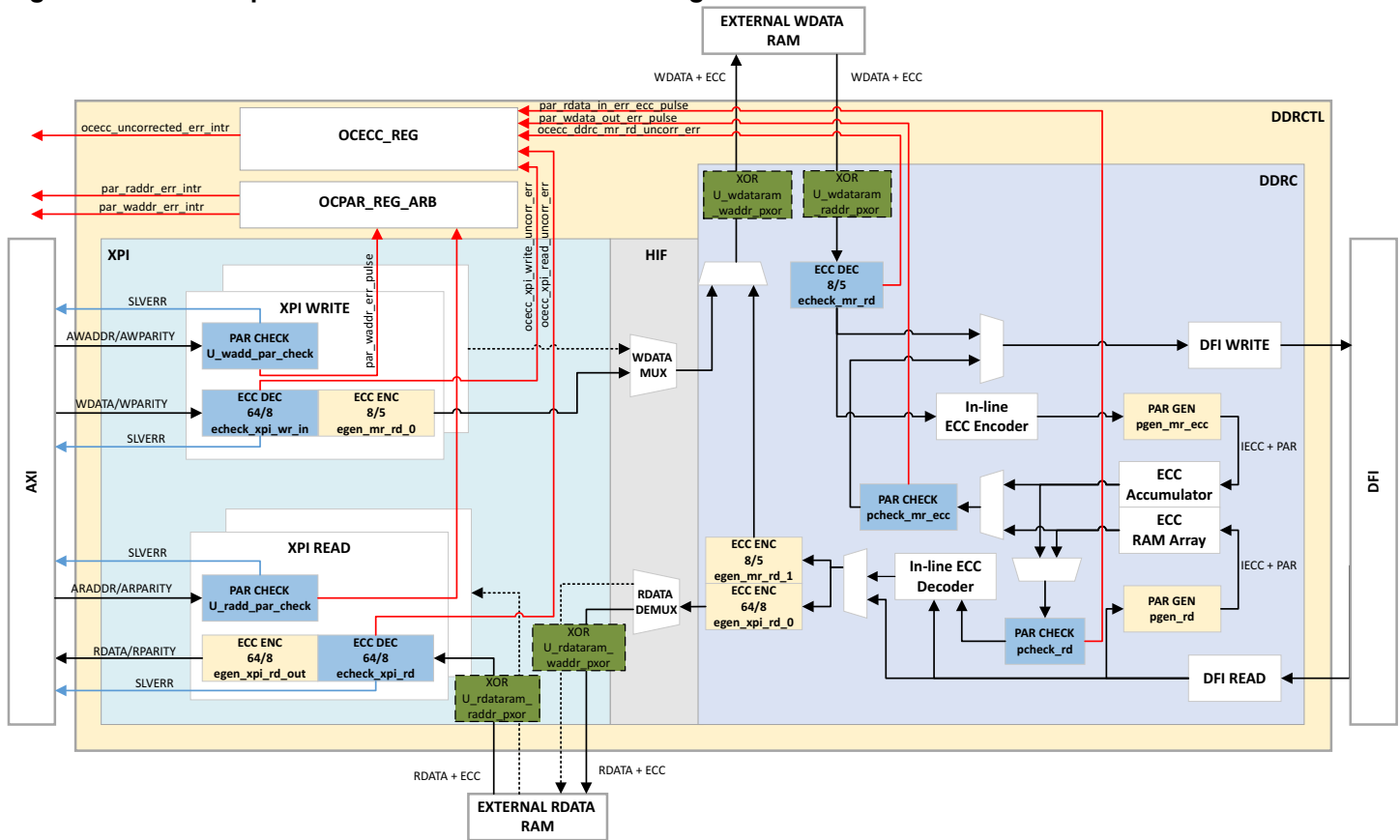
On-chip ECC functionality is supported only in AXI configurations, with Inline ECC.

The ECC travels along with the data. Whenever data are rearranged, the ECC is terminated and re-calculated. In some parts of the design, due to complexity, parity is kept and circulated instead of ECC. This applies to some logic inside DDRC. The parity used for these generators/checkers is even parity.

When ECC or parity are terminated, the controller checks for errors. If an error is detected, the controller generates an interrupt.

ECC decoders in the controller do not try to correct single-bit failures. The error interrupt is generated when two bit error or odd number of bit errors are detected within the same ECC data word by any of the ECC decoders or when parity error is detected by any of the parity checkers. See [Figure 13-23](#) for overview.

Figure 13-23 On-chip ECC Overview in Inline ECC Configurations



Note The highlighted blocks in green are present when `DDRCTL_OCSAP_EN = 1`.

13.4.2 Enabling On-chip ECC Support

To enable On-chip ECC support, set the hardware configuration parameter `UMCTL2_OCECC_EN` to '1'. This feature can then be enabled/disabled by software using the `OCECCCFG0.ocecc_en` register.

On-chip Address Parity feature can be enabled/disabled by software using the `OCPARCFG0.oc_parity_en` register.



Note In current On-chip ECC design, if Scrubber `UMCTL2_SBR_EN` is defined then On-chip ECC checkers are not available when Scrubber is in write mode (that is, `SBRCTL.scrub_en = 1` and `SBRCTL.scrub_cmd_type = 1`). Hence, you must disable On-chip ECC (by programming `OCECCCFG0.ocecc_en` to '0') during Scrubber write mode operation. Refer to steps for Initialization Writes "Scrubber" on page 301.

13.4.3 ECC Encoding

The ECC algorithm used by OCECC is a standard SECDED Hamming (72,64) code.

The OCECC encoder is implemented in the following source files:

- `src/ocecc/ocecc_enc.v`
- `src/mr/mr_secded_lane.v`

The OCECC decoder is implemented in the following source files:

- `src/ocecc/ocecc_dec.v`
- `src/rd/rd_secded_lane.v`

The ECC encoding used at the AXI interface is 64/8. For the external write data RAM, the ECC encoding is 8/5.

The ECC at the AXI interface must be provided per 64 bits. This means that, for example, for an AXI data width of 128, you must provide two 8-bit ECC words.

The minimum data width supported in the AXI interface is 64 bits.



Note

The AXI manager must provide the correct ECC for every 64 bits of AXI write data that is presented on the AXI write data channel, including AXI write data with strobes and for AXI sub-sized transfers.

13.4.4 Write Data Protection

For every write data beat, the DDRCTL receives one 8-bit ECC word per 64 data bits. Write data ECC on AXI interface is carried through a sideband signal called `wparity_n[UMCTL2_PORT_NBYTES_n-1:0]`.

The ECC is checked in two places: after the AXI interface and after the external write data RAM. After the Inline-ECC encoder, parity is used instead of ECC. The parity is checked before the DFI interface.

ECC is checked at the AXI interface when both `wvalid_n == 1` and `wready_n == 1`. Write transactions with ECC errors on the AXI interface return with a SLVERR response on AXI write response channel. Note, that the DDRCTL treats all AXI transactions as bufferable regardless of the transaction attribute. In other words, the write response is generated when the last write data beat is accepted inside the DDRC CAM. Therefore, AXI SLVERR write response is generated only based on ECC checking on the AXI interface. The SLVERR response can be disabled through `OCECCCFG0.ocecc_wdata_slvrr_en`.

The `OCECCSTAT1.ocecc_err_xpi_wr_in_n` registers log the AXI ports on which an ECC error for write data is detected.

13.4.5 Read Data Protection

For every read data beat, the DDRCTL generates one 8-bit ECC word per 64 data bits. Read data ECC on AXI interface is carried through a sideband signal called `rparity_n[UMCTL2_PORT_NBYTES_n-1:0]`.

The ECC is checked at the AXI interface when `rvalid_n == 1`. Read transactions with read data ECC errors are returned with a SLVERR response which can be disabled using `OCECCCFG0.ocecc_rdata_slvrr_en`.

The `OCECCSTAT1.ocecc_err_xpi_rd_n` registers log the AXI ports on which an ECC error for read data is detected.

13.4.6 Error Detection

An error detected in any of the ECC decoders or parity checkers is flagged by:

- Interrupt signal `ocecc_uncorrected_err_intr`, if enabled by `OCECCCFG0.ocecc_uncorrected_err_intr_en`.
- Fault signal `ocecc_uncorrected_err_intr_fault`, regardless of `OCECCCFG0.ocecc_uncorrected_err_intr_en`.
- `OCECCSTAT0.ocecc_uncorrected_err` register field, regardless of `OCECCCFG0.ocecc_uncorrected_err_intr_en`.

Additionally, the `OCECCSTAT2.ocecc_err_ddrc_mr_rd_byte_num` register logs which byte caused an ECC error at the write data RAM.

For testing or debug purposes, you can force an OCECC interrupt by setting `OCECCCFG0.ocecc_uncorrected_err_intr_force` to '1'.

In this situation, the error is flagged by:

- Interrupt signal `ocecc_uncorrected_err_intr`, if enabled by `OCECCCFG0.ocecc_uncorrected_err_intr_en`.
- `OCECCSTAT0.ocecc_uncorrected_err` register field, regardless of `OCECCCFG0.ocecc_uncorrected_err_intr_en`.

The interrupt is cleared by writing "1" to `OCECCCFG0.ocecc_uncorrected_err_intr_clr`.

For more information, see `REGB_DDRC_CH0` in the "Register Descriptions" chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

13.4.7 Address Parity

The AXI address (`araddr`, `awaddr`) is protected by one bit of parity for the entire address or by one bit of parity for each address byte, depending on hardware parameter `UMCTL2_OCPAR_ADDR_PARITY_WIDTH` setting. Address parity is carried by a sideband signal `arparity/awparity`. The address parity is checked and terminated within the XPI. Address parity errors are signaled to the manager through AXI SLVERR response which can be disabled by `OCPARCFG0.par_addr_slverr_en` register.

In addition to the error response, address parity errors generate maskable interrupts `par_waddr_err_intr` and `par_raddr_err_intr` respectively, for write address and read address errors. Interrupts can be disabled through registers `OCPARCFG0.par_waddr_err_intr_en` and `OCPARCFG0.par_raddr_err_intr_en`.

This feature uses the same infrastructure as "[Transaction Poisoning](#)" on page 55. Therefore, addresses with parity errors are effectively poisoned leading to similar operation.

Address parity errors are signaled through maskable interrupts (`par_wdata_err_intr`, `par_raddr_err_intr`), that can be disabled with `OCPARCFG0.par_waddr_err_intr_en` / `OCPARCFG0.par_raddr_err_intr_en`. `OCPARSTAT0.par_waddr_err_intr[15:0]` and `OCPARSTAT0.par_raddr_err_intr[15:0]` register logs AXI port/ports on which address parity error is detected.

13.4.8 Embedded SRAM Protection

External SRAMs interfaces have data and ECC.

Output ECC is driven by the controller along with the data going to the external SRAMs (there is no checker inside the controller for these outputs):

- `wdataram_din_par`: 8/5 ECC for `wdataram_din` (write data RAM input, valid bytes according to `wdataram_mask`)

- `rdataram_din_par_n`: 64/8 ECC for `rdataram_din_n` (port n read data RAM input, all bytes valid)

You must drive the input ECC to the controller based on the data read from each SRAM (controller expects the correct 64/8 ECC for data whenever the data read is valid):

- `wdataram_dout_par`: 8/5 ECC for `wdataram_dout` (write data RAM output)
- `rdataram_dout_par_n`: 64/8 ECC for `rdataram_dout_n` (port n read data RAM output)

Note, that `rdataram_din_par_n` and `rdataram_dout_par_n` are available only if the external RAM for port n is enabled; otherwise the ECC is buffered inside the RRB (internal) along with the read data. In this case, you need not to perform any action.

13.4.9 ECC/Parity Poisoning

The DDRCTL provides the ability to inject errors in the ECC encoders and parity generators. This poisoning can be used to test the feature and system response to ECC/parity errors. This functionality is enabled by setting the register `OCECCCFG1.ocecc_poison_en` (see the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide).

The following encoders can be poisoned:

- `egen_mr_rd_0`: poisoning is enabled by the `OCECCCFG1.ocecc_poison_egen_mr_rd_0` register. The `OCECCCFG1.ocecc_poison_egen_mr_rd_0_byte_num` register selects which byte is poisoned. The `OCECCCFG1.ocecc_poison_port_num` register selects on which port the poisoning takes place. The `OCECCCFG1.ocecc_poison_ecc_corr_uncorr` selects whether 1 or 2 bit errors are injected.
- `egen_mr_rd_1`: poisoning is enabled by the `OCECCCFG1.ocecc_poison_egen_mr_rd_1` register. The `OCECCCFG1.ocecc_poison_egen_mr_rd_1_byte_num` register selects which byte is poisoned. The `OCECCCFG1.ocecc_poison_ecc_corr_uncorr` selects whether 1 or 2 bit errors are injected.
- `egen_xpi_rd_0`: poisoning is enabled by the `OCECCCFG1.ocecc_poison_egen_xpi_rd_0` register. The `OCECCCFG1.ocecc_poison_ecc_corr_uncorr` selects whether 1 or 2 bit errors are injected.
- `egen_xpi_rd_out`: poisoning is enabled by the `OCECCCFG1.ocecc_poison_egen_xpi_rd_out` register. The `OCECCCFG1.ocecc_poison_port_num` register selects on which port the poisoning takes place.
- `pgen_mr_ecc`: poisoning is enabled by the `OCECCCFG1.poison_pgen_mr_ecc` register.
- `pgen_rd`: poisoning is enabled by the `OCECCCFG1.poison_pgen_rd` register.

The poisoning is one-shot, which means that an error is injected at the first valid data after `OCECCCFG1.ocecc_poison_en` is set or after `OCECCCFG0.ocecc_uncorrected_err_intr_clr` is set (that is, after the interrupt has been cleared).

Poisoning one of the ECC encoders or parity generators causes an error in the corresponding ECC decoder or parity checker.

AXI traffic restrictions for on-chip ECC/parity poisoning:

- AXI burst type must be INCR.
- AXI address must be aligned to SDRAM burst (for more information on AXI to SDRAM address translation, see [“Address Mapping” on page 103](#)).
- AXI address must not access invalid LPDDR4 row address (for more information, see [“Nonbinary Device Densities” on page 112](#)).
- AXI address must be aligned to INLINE_ECC block boundary (when `MEMC_INLINE_ECC == 1`. For more details on Inline ECC block boundary, see [“Inline ECC Support” on page 268](#)).
- One AXI transaction must not access multiple ECC regions/blocks (when `MEMC_INLINE_ECC == 1`. For more details on Inline ECC regions/block, see [“Inline ECC Support” on page 268](#)).
- In inline ECC configuration, for poisoning at `egen_mr_rd_1` block, AXI write burst must cause RMW on to HIF. To generate RMW for AXI Write burst, see [“Read-Modify-Write \(RMW\) Generation” on page 46](#). For poisoning at all other blocks (except `egen_mr_rd_1` block), AXI Write burst must not cause RMW on to HIF. To avoid RMW for AXI Write burst, see [“Read-Modify-Write \(RMW\) Generation” on page 46](#).
- In inline ECC configuration, for poisoning inside DDRC block, the AXI must perform read accesses to all the locations that are written by AXI from the time poisoning has enabled.

**Note**

The AXI traffic restrictions applies only if the controller works in the full bus width mode, done by appropriately setting `MSTR0.data_bus_width`.

13.4.10 Signals Related to On-chip ECC

For more information on signals related to On-chip ECC, see:

- [“AXI Port n Write Data On-Chip Parity Signals \(for n = 0; n <= UMCTL2_A_NPORTS-1\)” on page 387](#)
- [“AXI Port n Read Data On-Chip Parity Signals \(for n = 0; n <= UMCTL2_A_NPORTS-1\)” on page 398](#)
- [“AXI Port n Write Address On-Chip Parity Signals \(for n = 0; n <= UMCTL2_A_NPORTS-1\)” on page 384](#)
- [“AXI Port n Read Address On-Chip Parity Signals \(for n = 0; n <= UMCTL2_A_NPORTS-1\)” on page 395](#)

13.4.11 Registers Related to On-chip ECC

The following registers are related to On-chip ECC:

- `OCECCCFG0`
- `OCECCCFG1`
- `OCECCSTAT0`
- `OCECCSTAT1`
- `OCECCSTAT2`
- `OCPARCFG0`
- `OCPARSTAT0`

For more information, see `REGB_DDRC_CH0` in the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

13.5 Registers Parity Protection (REGPAR)

This topic contains the following sections:

- [“Overview of Registers Parity Protection \(REGPAR\)”](#) on page 322
- [“Enabling Registers Parity Protection”](#) on page 322
- [“Register Parity Generation”](#) on page 322
- [“Register Parity Checking”](#) on page 324
- [“Register Parity Poisoning”](#) on page 325
- [“Registers Related to REGPAR”](#) on page 326

13.5.1 Overview of Registers Parity Protection (REGPAR)

This feature is available for you only with the DWC-AP-LPDDR54-CONTROLLER Automotive Product (AP) license.

Register parity protection consists of the following two actions:

- Register Parity generation (on APB clock domain for configuration registers and on DDRCTL/application port clock domains for status registers).
- Register Parity checking (on APB clock, DDRCTL clock, Application port clock for configuration registers and on APB block domain for status registers).



Note

`pclk_rp` and `presetn_rp` are required for this REGPAR functionality. `presetn_rp` must have the same source as `presetn` (must be asserted at the same moment). `pclk_rp` must be driven synchronously to `pclk` and must be free-running.

13.5.2 Enabling Registers Parity Protection

To enable Registers Parity Protection, set the hardware configuration parameter `UMCTL2_REGPAR_EN` to '1'.

Support for Register parity can be enabled/disabled from software using the register field `REGPARCFG.reg_par_en`.

13.5.3 Register Parity Generation

This section contains the following subsections:

- [“Configuration Registers”](#) on page 322
- [“Status \(Read-only\) Registers”](#) on page 323
- [“Register Parity Checking”](#) on page 324
- [“Register Parity Poisoning”](#) on page 325
- [“Registers Related to REGPAR”](#) on page 326

13.5.3.1 Configuration Registers

In the following situations the parity information is generated on APB clock domain:

- At reset, based on default reset value.

- Whenever you update the register.
- Whenever you enable the register parity protection through `REGPARCFG.reg_parity_en` (see “REGB_DDRC_CH0 Registers” in the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide).
- Whenever you disable the register parity poisoning through `REGPARCFG.reg_par_poison_en` (see “REGB_DDRC_CH0 Registers” in the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide).

Parity information is generated based on 32-bit register value. The number of parity bits is controlled by the hardware parameter `UMCTL2_REGPAR_TYPE`, as following:

- `UMCTL2_REGPAR_TYPE = 0` - for every 32-bit register, one-bit parity is calculated.
- `UMCTL2_REGPAR_TYPE = 1` - for every 32-bit register, 4-bit parity is calculated, one bit for every byte of the register.

This information is appended to the payload and synchronized together with the DDRCTL clock or application port clock domain.

R/W1C and R/W1S register fields are masked when calculating parity information for the entire configuration register (for more details about register field types see the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide). This is needed to avoid false error detection caused by the fact that these register fields are automatically cleared. A separate mechanism is used for protecting these register fields. Flip-flops in R/W1S and R/W1C logic are duplicated and error is detected by comparing the outputs of original and duplicated flip-flops.

13.5.3.2 Status (Read-only) Registers

There are two types of read-only registers:

- Synchronized to the APB clock as a whole (using one CDC for all 32 bits, like configuration registers)
- Synchronized to the APB clock field by field (using separate CDC for each field)

Parity information is calculated and propagated differently for the two types of status registers.

For registers synchronized to the APB clock as a whole, the calculation mechanism is similar to the configuration register parity calculation. Parity information is generated based on 32-bit register value. The number of parity bits is controlled by the hardware parameter `UMCTL2_REGPAR_TYPE`, as following:

- `UMCTL2_REGPAR_TYPE = 0` - for every 32-bit register, one-bit parity is calculated.
- `UMCTL2_REGPAR_TYPE = 1` - for every 32-bit register, 4-bit parity is calculated, one bit for every byte of the register.

This information is appended to the payload and synchronized together with the APB clock domain.

For registers synchronized field by field to APB clock, register parity information is calculated differently. One parity bit is calculated for each field, regardless of the hardware parameter `UMCTL2_REGPAR_TYPE`. This information is appended to the field’s payload and synchronized together with the APB clock domain. After synchronization, a cumulative 1-bit error is calculated (OR between parity checking result of each). This cumulative error information is appended to the register value on APB clock domain (1 if any of the fields has parity error, 0 if none of the fields have parity error).

After synchronization, parity is checked for each field, however an error is not flagged yet. A cumulative 1-bit error is calculated based on error information of all fields (cumulative error is 1 if at least one field has

parity error, 0 if none of the fields has parity error). This 1-bit error indication is appended to the status register value.

13.5.4 Register Parity Checking

Parity checking is performed independently and executed concurrently in all the clock domains.

A parity error is flagged by the:

- Interrupt signal `reg_par_err_intr`, if enabled by `REGPARCFG.reg_par_err_intr_en`
- Interrupt signal `reg_par_err_intr_fault`, regardless of `REGPARCFG.reg_par_err_intr_en` setting
- `REGPARSTAT.reg_par_err_intr` register field, regardless of `REGPARCFG.reg_par_err_intr_en` setting

You can force the REGPAR error flagging by setting `REGPARCFG.reg_par_err_intr_force` to '1'. In this situation, a REGPAR error is flagged by:

- Interrupt signal `reg_par_err_intr`, if enabled by `REGPARCFG.reg_par_err_intr_en`
- `REGPARSTAT.reg_par_err_intr` register field, regardless of `REGPARCFG.reg_par_err_intr_en` setting

The interrupt is cleared by writing a '1' to `REGPARCFG.reg_par_err_intr_clr`. This field is automatically cleared.

13.5.4.1 Register Parity Checking on APB Interface Clock

Parity checking on the APB clock domain (excluding R/W1S and R/W1C register fields of configuration registers) is performed:

- Periodically for all configuration registers in round-robin order, one register being checked every APB clock cycle.
- Whenever a configuration register is read.

For periodic checking the existing register read multiplexing path is re-used. When APB read and register parity checking coincide, the read has a higher priority. This means that the round-robin mechanism is stopped and parity checking address is not incremented at this point. Register parity checking is instead performed on the register which is currently read.

If using this feature, before writing to a register, it is recommended to perform a spare APB read of this register prior to the APB write. This triggers a check on the register contents before being updated.

Depending on register type, parity checking is performed as follows:

- For configuration registers and status registers synchronized as a whole: When register is selected, parity is re-calculated and compared against the parity information appended to register payload.
- For status registers synchronized field by field: When the register is selected, the cumulative error information is directly used as an error indication.

Register parity error detection is flagged by the interrupt signals `reg_par_err_intr`, `reg_par_err_intr_fault`, and by the register `REGPARSTAT.reg_par_err_intr`.

Since register parity checking is performed one register at a time, errors are detected with latency. The worst case error detection latency can be calculated as:

$$\text{tregpar_err_latency} = \text{MAX_RANGE}/4 * \text{clock_period}$$

where MAX_RANGE is the biggest address range among all existing register blocks. Currently all register blocks have an address range of 0x1000 (which translates in decimal to 4096), hence the worst case error detection latency is

$$\text{tregpar_err_latency} = 4096/4 * \text{clock_period} = 1024 * \text{clock_period}.$$

Register parity error on R/W1S or R/W1C register field is immediately flagged. Note, that register parity checking of these register fields is only performed on APB clock domain.

13.5.4.2 Register Parity Checking on DDRCTL Controller Clock and Application Port Clock

Register parity checking on DDRCTL controller and application port clock domains is performed for dynamic registers if the clocks are asynchronous to the APB clock. When clocks are asynchronous, CDC logic is instantiated and mirror registers are placed on destination clocks. Parity checking is needed to detect errors in CDC logic and mirror registers.

Parity information is synchronized to `core_ddrc_core_clk` and `aclk_N` together with the register value. Parity checking is performed every clock cycle on destination clock domain in parallel for all registers being crossed. If an error is detected, error information is synchronized back to the APB clock domain, where it is flagged by interrupt signals `reg_par_err_intr`, `reg_par_err_intr_fault`, and by the register `REGPARSTAT.reg_par_err_intr` (see “REGB_DDRC_CH0 Registers” in the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide).

Error detection latency in this case is determined by the amount of time it takes for error information to be synchronized to the APB clock domain.

Note, that parity checking of R/W1S and R/W1C register fields is not performed on controller clock or application port domains, because of the self-clear behavior of these fields.

13.5.5 Register Parity Poisoning

The DDRCTL provides the ability to inject register parity errors. This can be used to test the feature and system response to register parity errors.

Register parity poisoning is enabled by setting the register `REGPARCFG.reg_par_poison_en` (see “REGB_DDRC_CH0 Registers” in the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide). When enabled, a register parity error is injected upon accessing configuration registers. There is no poisoning capability for read-only registers. You must perform the following register accesses to trigger register parity poisoning:

- Write/read for dynamic registers
- Read for quasi-dynamic registers
- Read for static registers

For details about the different register types, see the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

Later, the poisoned address is observed as an error only if checking has occurred in the APB Clock domain on that register (For more information, see the [“Register Parity Checking on APB Interface Clock”](#) on page 324).

Therefore, a poisoned address needs two APB accesses, one to poison it, and then a subsequent APB Read to the same address, to guarantee triggering of register parity error.

Poisoning is removed when `REGPARCFG.reg_par_poison_en` is set to '0' and register parity is recalculated for all configuration registers to revert false parity error introduced with poisoning. In the case of 4-bit parity (`UMCTL2_REGPAR_TYPE = 1`) only the most significant parity bit is poisoned.

**Note**

You must configure all the fields of the `REGPARCFG` register with a single write operation. It is also recommended to have at least 128 pclk cycled between two consecutive writes to the `REGPARCFG`.

13.5.6 Registers Related to REGPAR

The following are the registers related to REGPAR:

- `REGPARCFG`
- `REGPARSTAT`

For more information about these registers, see the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

13.6 On-Chip Command and Address Path Protection (OCCAP)

This topic contains the following sections:

- [“Overview of On-Chip Command and Address Path Protection \(OCCAP\)”](#) on page 327
- [“Enabling On-Chip Command and Address Path Protection”](#) on page 327
- [“Protection Mechanisms”](#) on page 327
- [“Arbiter Protection”](#) on page 330
- [“DDRC Protection”](#) on page 330
- [“Poisoning”](#) on page 332
- [“Signals Related to OCCAP”](#) on page 334
- [“Registers Related to OCCAP”](#) on page 334

13.6.1 Overview of On-Chip Command and Address Path Protection (OCCAP)

OCCAP supplements the OCPAR feature which is a similar protection mechanism for data path only.

This feature is available for you only with the following Automotive Product (AP) license:

- DWC-AP-LPDDR54-CONTROLLER

For automotive applications, reliability is important. The design needs to be robust against:

- Permanent faults (affecting both sequential and combinatorial logic)
- Transient faults (affecting sequential logic only)

The detection of single bit errors is important for both the fault types.

13.6.2 Enabling On-Chip Command and Address Path Protection

To enable On-Chip Command and Address Path Protection, set the hardware configuration parameter `UMCTL2_OCCAP_EN` to '1'.

13.6.3 Protection Mechanisms

The command and address path protection employs four mechanisms:

1. Parity protection: Parity is generated internally at the input of a block, propagated to the data, and checked at the output. This is used mainly for big registers, FIFOs, or memory arrays. An error is raised if the parity check at the output fails.
2. Duplication with comparison: The entire module is duplicated and outputs are checked every cycle, one by one, with XOR gates. An error is raised if any of the comparators fails.
3. Automotive FIFO Controllers. Enhanced versions of DesignWare BCMs of FIFO Controllers.
4. Triple Module Replication (TMR) of Registers.

The OCCAP feature is enabled by the `OCCAPCFG.occap_en` register (see “`REGB_DDRC_CH0` Registers” in the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide).

13.6.3.1 Parity Protection

Parity calculation and check are based on a standard even parity with XOR gates.

For every 8 bits, data is XOR reduced to generate the parity bit. Input to the calculation/check module is always a multiple of 8 bits. The resulting errors from the different checkers are OR'ed to generate an interrupt and the corresponding fault signal.

The parity generators/checkers can be found throughout the hierarchy in various modules. The naming convention for these are:

- *OCCAP_en*U_pgen and *OCCAP_en*U_pcheck
- *PAR_check*U_pgen and *PAR_check*U_pcheck

If UMCTL2_OCCAP_PIPELINE=1 is set, additional pipelining stages are added:

- Outputs generated by pchecks throughout the design for error reporting.

This may help with timing issues if needed.

If UMCTL2_OCCAP_PIPELINE=1 is set, includes an additional pipeline for pcheck in rd_ie_rdata_ctl module when OCPAR is enabled with Inline ECC.

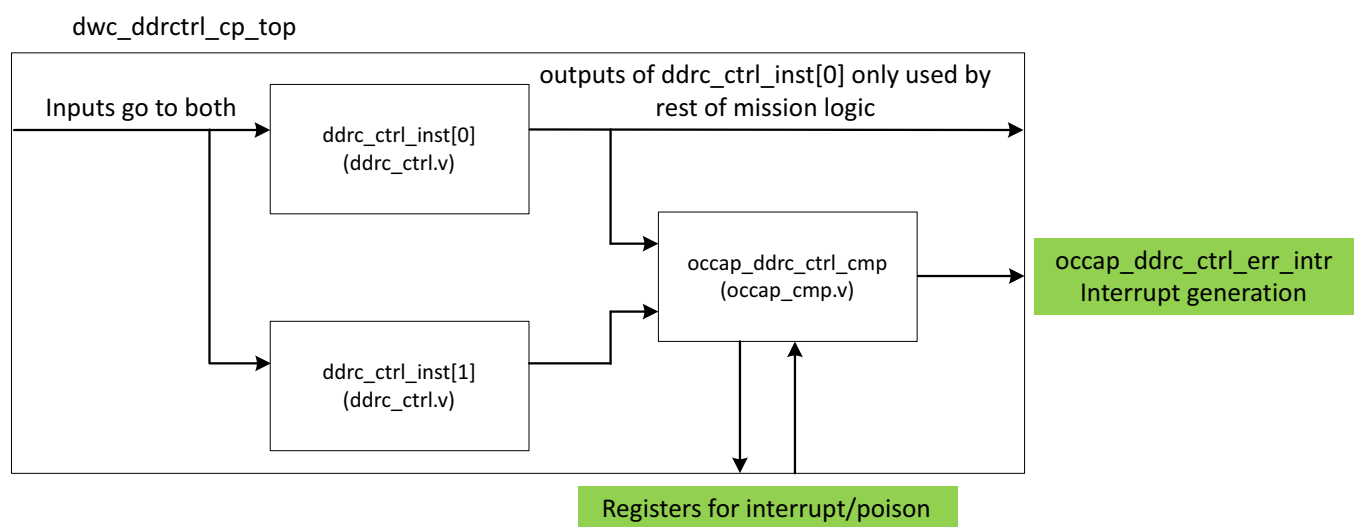
13.6.3.2 Duplication With Comparison

A replication of the module is instantiated together with the original block and a comparator, inside a *_wrapper module.

Same inputs are fed to both modules (original and replication), while both outputs are fed to the comparators. Standard comparators with XOR gates check every single output bit, and flag an error whenever there is a mismatch.

Figure 13-24 shows an example for the DDRC control (dwc_ddrctrl_cp_top) block in the DDRC.

Figure 13-24 DDRC Control Wrapper



The duplicated modules are configuration dependent and are in hierarchy at:

- *U_rp/RP_inst[1:0] (for Arbiter)
- U_wp/WP_inst[1:0] (for Arbiter)
- U_ws/WS_inst[1:0] (for Arbiter)
- U_au/AU_inst[1:0] (for Arbiter)
- *ddrc_ctrl_wrapper/ddrc_ctrl_inst[1:0] (for DDRC Control)
- *mr_wrapper/mr_inst[1:0] (for DDRC Data)
- *rd_wrapper/rd_inst[1:0] (for DDRC Data)

Note that the comparator checking of `ddrc_ctrl_inst[1:0]/mr_inst[1:0]/rd_inst[1:0]` is disabled outside of reset when `SWCTL.sw_done = 0`. This means that the other register settings are being changed.

For special requirements on these duplicated module, refer to the “OCCAP Synthesis Constraints chapter in the DesignWare Cores LPDDR54 Memory Controller User Guide.

If `UMCTL2_OCCAP_PIPELINE=1` is set, additional pipelining stages are added:

- Inputs to `occap_cmp` from the duplicated modules.
- Outputs generated by `occap_cmp` for error reporting.

This may help with timing issues if needed.

13.6.3.3 Automotive FIFO Controllers

The FIFO controllers within the DDRCTL use standard products from the DesignWare library – see “Synchronizers Used in `DWC_ddrctl`” in DesignWare Cores LPDDR54 Memory Controller User Guide. These are in following modules:

- Synchronous (Dual Clock) FIFO Controller: `DWC_ddrctl_bcm07`
- Submodule of `bcm07`: `DWC_ddrctl_bcm05`
- Synchronous (Single Clock) FIFO: `DWC_ddrctl_bcm65`
- Synchronous (Single Clock) FIFO Controller (and submodule of `bcm65`): `bcm06`

When `UMCTL2_OCCAP_EN = 1`, these are replaced with automotive versions of these existing BCM modules:

- `DWC_ddrctl_bcm07` -> `DWC_ddrctl_bcm07_atv`
- `DWC_ddrctl_bcm05` -> `DWC_ddrctl_bcm05_atv`
- `DWC_ddrctl_bcm65` -> `DWC_ddrctl_bcm65_atv`
- `DWC_ddrctl_bcm06` -> `DWC_ddrctl_bcm06_atv`

These Automotive BCMs use Triple Module Replication (TMR) techniques to be robust to transient errors in Automotive applications. In turn, some of these uses following sub-modules:

- `bcm05_atv` - Uses `DWC_ddrctl_bcm95_i` and `DWC_ddrctl_bcm21_atv`
- `bcm06_atv` - Uses `DWC_ddrctl_bcm95_i`

`DWC_ddrctl_bcm21_atv` is an automotive version of the standard `DWC_ddrctl_bcm21`, which is used for CDC. It is based on a TMR of the standard `DWC_ddrctl_bcm21`, with majority voting logic. When instantiated, each register inside the `DWC_ddrctl_bcm21_atv` has the name ‘sample_meta*’ or

'sample_sync*'. It is only instantiated in the following instances of bcm05_atv inside bcm07_atv and can be identified by:

- *U_bcm07_atv/U_PUSH_FIFOCTL*
- *U_bcm07_atv/U_POP_FIFOCTL*

DWC_ddrctl_bcm95_i is a triple module replication of a registers, with majority voting logic between the three versions. When TMR occurs, each register has the name "dw_so_reg*".

13.6.3.4 Triple Module Replication (TMR) of Registers

The DesignWare library provides a method to replace a register with a triple module replication version of the same register, with majority voting logic between the three versions of the same register. This is provided by the DWC_ddrctl_bcm95_i module. An instance of DWC_ddrctl_bcm95_i module explicitly implements a small number of important registers, and these registers can be identified by the name "dw_so_reg*".

13.6.4 Arbiter Protection

The arbiter specifically refers to the AXI Port Interface (XPI) and the Port Arbiter (PA).

The arbiter protection mechanism uses:

- Parity
- Duplication with Comparison
- Automotive FIFO Controller
- TMR of registers

An error in any of the Arbiter blocks is flagged by:

- Interrupt signal occap_arb_err_intr, if enabled by OCCAPCFG.occap_arb_intr_en
- Fault signal occap_arb_err_intr_fault, regardless of OCCAPCFG.occap_arb_intr_en setting
- OCCAPSTAT.occap_arb_err_intr register field, regardless of OCCAPCFG.occap_arb_intr_en setting

For testing or debug purposes, you can force an Arbiter OCCAP interrupt by setting OCCAPCFG.occap_arb_intr_force to '1'. In this situation, the error is flagged by:

- Interrupt signal occap_arb_err_intr, if enabled by OCCAPCFG.occap_arb_intr_en
- OCCAPSTAT.occap_arb_err_intr register field, regardless of OCCAPCFG.occap_arb_intr_en setting

The interrupt is cleared by writing "1" to OCCAPCFG.occap_arb_intr_clr. For more information, see "REGB_DDRC_CH0 Registers" in the "Register Descriptions" chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

13.6.5 DDRC Protection

This topic contains the following subsections:

- [“DDRC Control Protection”](#) on page 331
- [“DDRC Data Protection”](#) on page 331

13.6.5.1 DDRC Control Protection

The DDRC Control block (`ddrc_ctrl`) contains the address and control path of the DDRC. The DDRC Control protection uses ‘Duplication with Comparison’ mechanism.

An error in the DDRC Control block is flagged by:

- Interrupt signal `occap_ddrc_ctrl_err_intr`, if enabled by `OCCAPCFG1.occap_ddrc_ctrl_intr_en`
- Fault signal `occap_ddrc_ctrl_err_intr_fault`, regardless of `OCCAPCFG1.occap_ddrc_ctrl_intr_en` setting
- `OCCAPSTAT1.occap_ddrc_ctrl_err_intr` register field, regardless of `OCCAPCFG1.occap_ddrc_ctrl_intr_en` setting

For testing or debug purposes, you can force a DDRC Control OCCAP interrupt by setting `OCCAPCFG1.occap_ddrc_ctrl_intr_force` to ‘1’. In this situation, the error is flagged by:

- Interrupt signal `occap_ddrc_ctrl_err_intr`, if enabled by `OCCAPCFG1.occap_ddrc_ctrl_intr_en`
- `OCCAPSTAT1.occap_ddrc_ctrl_err_intr` register field, regardless of `OCCAPCFG1.occap_ddrc_ctrl_intr_en` setting

The interrupt is cleared by writing “1” to `OCCAPCFG1.occap_ddrc_ctrl_intr_clr`. For more information, see “REGB_DDRC_CH0 Registers” in the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

13.6.5.2 DDRC Data Protection

The DDRC Data refers to several modules within DDRC data path, for example, `memc_wu`, `mr`, `rd`, `rt`, and `dfi_data`. The control/address logic of the DDRC data related modules are protected.

The DDRC data protection uses:

- Parity
- Duplication with Comparison
- Automotive FIFO Controller

An error in the DDRC Data block is flagged by:

- Interrupt signal `occap_ddrc_data_err_intr`, if enabled by `OCCAPCFG1.occap_ddrc_data_intr_en`
- Fault signal `occap_ddrc_data_err_intr_fault`, regardless of `OCCAPCFG1.occap_ddrc_data_intr_en` setting
- `OCCAPSTAT1.occap_ddrc_data_err_intr` register field, regardless of `OCCAPCFG1.occap_ddrc_data_intr_en` setting

For testing or debug purposes, you can force a DDRC Data OCCAP interrupt by setting `OCCAPCFG1.occap_ddrc_data_intr_force` to ‘1’. In this situation, the error is flagged by:

- Interrupt signal `occap_ddrc_data_err_intr`, if enabled by `OCCAPCFG1.occap_ddrc_data_intr_en`
- `OCCAPSTAT1.occap_ddrc_data_err_intr` register field, regardless of `OCCAPCFG1.occap_ddrc_data_intr_en` setting

The interrupt is cleared by writing “1” to `OCCAPCFG1.occap_ddrc_data_intr_clr`. For more information, see “REGB_DDRC_CH0 Registers” in the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

13.6.6 Poisoning

The DDRCTL provides the ability to inject errors into the protection mechanisms. This can be used to test the feature and system response to OCCAP errors.

13.6.6.1 OCCAP Comparator Poisoning

All the OCCAP comparators that are used in the Duplication with Comparison mechanism can be poisoned by the software. The poisoning can be parallel (all bits of all outputs of the duplicate modules are inverted) or sequential (each bit is inverted one at a time). The sequential poisoning is performed in parallel for all involved comparators. The duration of the sequential poisoning depends on the size of the biggest comparator.

You cannot perform parallel poisoning and sequential poisoning at the same time.

Errors can be injected into the poisoning logic (either parallel or sequential) such that the XOR logic for one signal is not poisoned when expected. This is flagged by the corresponding `OCCAPSTAT*.occap_*_poison_parallel_err` or `OCCAPSTAT*.occap_*_poison_seq_err` being set.

The Arbiter comparators can be tested through the following registers:

- `OCCAPCFG.occap_arb_cmp_poison_seq`
- `OCCAPCFG.occap_arb_cmp_poison_parallel`
- `OCCAPCFG.occap_arb_cmp_poison_err_inj`
- `OCCAPSTAT.occap_arb_cmp_poison_seq_err`
- `OCCAPSTAT.occap_arb_cmp_poison_parallel_err`

The DDRC Control comparators can be tested through the following registers:

- `OCCAPCFG1.occap_ddrc_ctrl_poison_seq`
- `OCCAPCFG1.occap_ddrc_ctrl_poison_parallel`
- `OCCAPCFG1.occap_ddrc_ctrl_poison_err_inj`
- `OCCAPSTAT1.occap_ddrc_ctrl_poison_seq_err`
- `OCCAPSTAT1.occap_ddrc_ctrl_poison_parallel_err`

The DDRC Data comparators can be tested through the following registers:

- `OCCAPCFG1.occap_ddrc_data_poison_seq`
- `OCCAPCFG1.occap_ddrc_data_poison_parallel`
- `OCCAPCFG1.occap_ddrc_data_poison_err_inj`

- OCCAPSTAT1.occap_ddrc_data_poison_seq_err
- OCCAPSTAT1.occap_ddrc_data_poison_parallel_err

For more information, see “REGB_DDRC_CH0 Registers” in the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

Table 13-9 shows the software poisoning sequence example for DDRC control.

Table 13-9 Software Poisoning Sequence Example for DDRC Control

Step	Description	Comment
1	Set OCCAPCFG1.occap_ddrc_ctrl_poison_err_inj = 1. This field must not be set in the same APB access as Step 2.	Enables error injecting into the poisoning procedure. This step is optional.
2	Set OCCAPCFG1.occap_ddrc_ctrl_poison_seq = 1 Or OCCAPCFG1.occap_ddrc_ctrl_poison_parallel = 1 Do not set both at the same time.	Enables selected poisoning procedure, sequential or parallel.
3	Poll for OCCAPSTAT1.occap_ddrc_ctrl_poison_complete=1	Checks for the completion of the poisoning procedure.
4	Depending on Step 2, check corresponding: OCCAPSTAT1.occap_ddrc_ctrl_poison_seq_err Or OCCAPSTAT1.occap_ddrc_ctrl_poison_parallel_err If OCCAPCFG1.occap_ddrc_ctrl_poison_err_inj = 1, the error status must be 1, otherwise it must be 0.	Checks if the corresponding poisoning operated as expected.
5	Set OCCAPCFG1.occap_ddrc_ctrl_intr_clr	Clears the following registers: OCCAPSTAT1.occap_ddrc_ctrl_err_intr and OCCAPSTAT1.occap_ddrc_ctrl_poison_seq_err Or OCCAPSTAT1.occap_ddrc_ctrl_poison_parallel_err
6	Poll for OCCAPSTAT1.occap_ddrc_ctrl_poison_complete = 0	Makes sure poisoning has terminated.

13.6.6.2 RAQ Poisoning

Due to the complexity of the design and the unpredictability of behavior and latencies, parity poisoning is possible only for the Read Address Queues (RAQ) in each port's XPI block.

The number of RAQs in each XPI is configuration dependent: UMCTL2_XPI_USE2RAQ_n. Either 1 or 2 RAQ exist and poisoning occurs for all RAQs within a port.

When poisoning is enabled, every time the RAQ is written, all parity bits are flipped, basically an odd parity is calculated instead. An error is then flagged every time the RAQ is read.

RAQ poisoning is enabled through the register `OCCAPCFG.occap_arb_raq_poison_en`. For more information, see “REGB_DDRC_CH0 Registers” in the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

13.6.7 Signals Related to OCCAP

For information on signals related to OCCAP, see “[On-Chip Command/ Address Path Protection Signals](#)” on page 406.

13.6.8 Registers Related to OCCAP

The following are the registers related to the OCCAP:

- `OCCAPCFG`
- `OCCAPSTAT`
- `OCCAPCFG1`
- `OCCAPSTAT1`

For more information on these registers, see the “Register Descriptions” chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

13.7 On-Chip External SRAM Address Protection (OCSAP)

This topic contains the following sections:

- [“Overview of On-Chip External SRAM Address Protection”](#) on page 335
- [“Enabling On-Chip External SRAM Address Protection \(OCSAP\)”](#) on page 335
- [“On-Chip External SRAM Address Parity”](#) on page 335
- [“On-Chip External SRAM Address Parity Poisoning”](#) on page 335
- [“Signals Related to On-Chip External SRAM Address Protection”](#) on page 336
- [“Registers Related to On-Chip External SRAM Address Protection”](#) on page 336

13.7.1 Overview of On-Chip External SRAM Address Protection



Note

This feature is available only with the DWC-AP-LPDDR54-CONTROLLER Automotive Product (AP) license.

On-Chip external SRAM Address Protection (OCSAP) works in conjunction with OCPAR or OCECC support with Inline ECC. This is primarily to provide address protection for the external SRAM while end-to-end data path protection is achieved either using OCPAR or OCECC.

SRAM address has 2-bit even parity which is embedded within the data parity/ECC before writing to SRAM. The actual data parity/ECC is retrieved during the read operation of SRAM. The SRAM address parity is checked and terminated; only the retrieved data parity/ECC is further propagated along with the data. It uses the existing error reporting mechanism as followed in OCPAR or OCECC configuration. See [Figure 13-21](#) on page 310 and [Figure 13-23](#) on page 317 where OCSAP related blocks are highlighted in green.

13.7.2 Enabling On-Chip External SRAM Address Protection (OCSAP)

To enable On-Chip External SRAM Address Protection, set the hardware configuration parameter `DDRCTL_OCSAP_EN` to '1'.

This feature can then be enabled/disabled by software using the `OCSAPCFG0.ocsap_par_en` register.

13.7.3 On-Chip External SRAM Address Parity

For every write or read to external SRAM, 2-bit SRAM address even parity is generated. This 2-bit address parity is unique for four consecutive SRAM address locations and it is calculated as follows:

$$\begin{aligned}\text{Parity}[0] &= A0 \oplus A2 \oplus A4 \oplus A6 \\ \text{Parity}[1] &= A1 \oplus A3 \oplus A5 \oplus A7\end{aligned}$$

SRAM address parity, which is 2-bits, is combined with either the data path parity (in case of OCPAR) or the data path ECC (in case of OCECC) by XOR operation.

13.7.4 On-Chip External SRAM Address Parity Poisoning

The `DDRCTL` provides the ability to inject SRAM address parity errors. This poisoning can be used to test the feature and system's responses to incorrect SRAM addresses. This functionality is enabled by setting the register `OCSAPCFG0.ocsap_poison_en`.

The following SRAM address parity can be poisoned:

- **WDATARAM address** - based on `OCSAPCFG0.wdataram_addr_poison_ctl`, one of the WDATARAM address bits is corrupted during the computation of WDATARAM address parity. This eventually injects one of the WRDATARAM address parity errors. Based on `OCSAPCFG0.wdataram_addr_poison_loc`, parity is poisoned in either WDATARAM read address or write address.
- **RDATARAM address** - based on `OCSAPCFG0.rdataram_addr_poison_ctl`, one of the RDATARAM address bits is corrupted during the computation of RDATARAM address parity. This eventually injects one of the RRDATARAM address parity errors. Based on `OCSAPCFG0.rdataram_addr_poison_loc`, parity is poisoned in either RDATARAM read address or write address. Only one port at a time can be poisoned based on `OCSAPCFG0.rdataram_addr_poison_port`.

One-shot triggering is supported, which means that error is injected for the first valid address either after `OCSAPCFG0.ocsap_poison_en` is set to '1' or after the interrupt clear for that path is asserted. This means that only the parity corresponding to the first valid access of the transaction on SRAM is poisoned.

AXI traffic restrictions for on-chip external SRAM address parity poisoning:

- All the AXI traffic restrictions as defined in section “[ECC/Parity Poisoning](#)” on page 320 apply for on-chip external SRAM address parity poisoning with OCECC enable.
- All the AXI traffic restrictions as defined in section “[Parity Poisoning](#)” on page 313 apply for on-chip external SRAM address parity poisoning with OCPAR enable.



Note

Only one (`OCSAPCFG0.ocsap_poison_en`, `OCPCFG1.par_poison_en`, or `OCECCFG1.ocecc_poison_en`) is effective at one time.

13.7.5 Signals Related to On-Chip External SRAM Address Protection

OCSAP works in conjunction with either OCPAR or OCECC support. It used the existing error reporting mechanism as followed in OCPAR or OCECC configuration.

13.7.6 Registers Related to On-Chip External SRAM Address Protection

OCSAP works in conjunction with either OCPAR or OCECC support. In addition to OCPAR/OCECC related registers, the following is the only register related to the On-Chip external SRAM Address Protection:

- `OCSAPCFG0`

For more information on these registers, see the "Register Descriptions" chapter of the DWC DDRCTL LPDDR5/4 Programming Guide.

14

Parameter Descriptions

This chapter details all the configuration parameters. **You can use the coreConsultant GUI configuration reports to determine the complete configuration state of the controller.** Some expressions might refer to TCL functions or procedures (sometimes identified as **<functionof>**) that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the controller in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

The parameter descriptions in this chapter include the **Enabled:** attribute which indicates the values required to be set on other parameters before you can change the value of this parameter.

These tables define all of the configuration options for this component.

- [“HW Configuration / Product Parameters”](#) on page 338
- [“HW Configuration / DDRC Parameters”](#) on page 340
- [“HW Configuration / Multiport Parameters”](#) on page 347
- [“HW Configuration / AXI Parameters”](#) on page 356
- [“HW Configuration / Multi-channel Parameters”](#) on page 360
- [“HW Configuration / CHI Bridge settings Parameters”](#) on page 361
- [“HW Configuration / Reliability Features Parameters”](#) on page 366
- [“HW Configuration / RTL Assertions Parameters”](#) on page 372

14.1 HW Configuration / Product Parameters

Table 14-1 HW Configuration / Product Parameters

Label	Description
Product Choice	
DDR Controller Product	<p>This parameter specifies the type of DDR memory controller. For each product, a license is required as specified:</p> <ul style="list-style-type: none"> ■ DWC LPDDR5/4/4X Controller (LPDDR54) requires DWC-LPDDR54-CONTROLLER license. ■ DWC DDR5/4 Controller (DDR54) requires DWC-DDR54-CONTROLLER license. ■ DWC AP LPDDR5/4/4X Controller (AP_LPDDR54) requires DWC-AP-LPDDR54-CONTROLLER license. ■ DWC AP DDR5/4 Controller (AP_DDR54) requires DWC-AP-DDR54-CONTROLLER license. ■ DWC LPDDR5/4/4X Controller AFP (AFP_LPDDR54) requires DWC-LPDDR54-CONTROLLER-AFP license. ■ DWC DDR5/4 Controller AFP (AFP_DDR54) requires DWC-DDR54-CONTROLLER-AFP license. ■ DWC DDR5/4 Controller with CHI (TBD) requires DWC-DDR54-CONTROLLER-CHI license. ■ DWC DDR5/4 Controller AFP with CHI (TBD) requires DWC-DDR54-CONTROLLER-AFP-CHI license. ■ DWC DDR5 Low Latency Controller AFP (AFP_DDR5_LLC) requires DWC-DDR54-LL-CONTROLLER-AFP license. <p>Values:</p> <ul style="list-style-type: none"> ■ DWC_LPDDR54_CONTROLLER (0) ■ DWC_DDR54_CONTROLLER (1) ■ DWC_AP_LPDDR54_CONTROLLER (2) ■ DWC_AP_DDR54_CONTROLLER (3) ■ DWC_LPDDR54_CONTROLLER_AFP (4) ■ DWC_DDR54_CONTROLLER_AFP (5) ■ DWC_DDR54_CONTROLLER_CHI (6) ■ DWC_DDR54_CONTROLLER_AFP_CHI (7) ■ DWC_DDR5_LL_CONTROLLER_AFP (8) <p>Default Value:</p>

Label	Description
DDR Controller Product ...(cont.)	<pre> =<DWC-LPDDR54-CONTROLLER feature authorize> ? 0 : (<DWC-DDR54-CONTROLLER feature authorize> ? 1 : (<DWC-AP-LPDDR54-CONTROLLER feature authorize> ? 2 : (<DWC-AP-DDR54-CONTROLLER feature authorize> ? 3 : (<DWC-LPDDR54-CONTROLLER-AFP feature authorize> ? 4 : (<DWC-DDR54-CONTROLLER-AFP feature authorize> ? 5 : (<DWC-DDR54-CONTROLLER-CHI feature authorize> ? 6 : (<DWC-DDR54-CONTROLLER-AFP-CHI feature authorize> ? 7 :(<DWC-DDR54-LL-CONTROLLER-AFP feature authorize> ? 8 :0))))))) Enabled: (<DWC-LPDDR54-CONTROLLER feature authorize> <DWC-DDR54-CONTROLLER feature authorize> <DWC-AP-LPDDR54-CONTROLLER feature authorize> <DWC-AP-DDR54-CONTROLLER feature authorize> <DWC-LPDDR54-CONTROLLER-AFP feature authorize> <DWC-DDR54-CONTROLLER-AFP feature authorize> <DWC-DDR54-CONTROLLER-CHI feature authorize> <DWC-DDR54-CONTROLLER-AFP-CHI feature authorize> <DWC-DDR54-LL-CONTROLLER-AFP feature authorize>) == 1 Parameter Name: DDRCTL_PRODUCT_NAME </pre>
System Interface : Include	<p>Select the System Interface for DDRC.</p> <ul style="list-style-type: none"> ■ HIF : Uses DDRC's HIF IF as system Interface ■ Arbiter (AXI) : Uses Multi-Port AXI as the System Interface ■ CHI : Uses CHI as the System Interface <p>Values:</p> <ul style="list-style-type: none"> ■ HIF (0) ■ Arbiter (AXI) (1) ■ CHI (2) <p>Default Value: (MEMC_DRAM_DATA_WIDTH == 8 MEMC_DRAM_DATA_WIDTH == 16 MEMC_DRAM_DATA_WIDTH == 32 MEMC_DRAM_DATA_WIDTH == 64) ? 1 : 0</p> <p>Enabled: Always</p> <p>Parameter Name: DDRCTL_SYS_INTF</p>

14.2 HW Configuration / DDRC Parameters

Table 14-2 HW Configuration / DDRC Parameters

Label	Description
Host Interface Configuration Options	
Enable Dual HIF	<p>Enables the support for Dual HIF command feature.</p> <ul style="list-style-type: none"> This feature converts HIF single command channel into separate HIF command channels for Read and Write commands. RMW commands are performed on the Write HIF command channel. Read/Write arbitration performed by the PA does not occur as there are separate Read and Write channels for the PA to drive. This feature can only be enabled if the logic to optimize timing over scheduling efficiency is enabled (MEMC_OPT_TIMING==1). Enabling this logic improves the SDRAM utilization, depending on your traffic profile. However, it increases the overall area due to additional logic. <p>Values: 0, 1 Default Value: 0 Enabled: MEMC_OPT_TIMING ==1 && UMCTL2_INCL_ARB == 0 Parameter Name: UMCTL2_DUAL_HIF</p>
Performance Interface Configuration Options	
Performance log on	<p>Enables performance logging interface. When enabled, the performance logging signals are added to the list of output ports of the IIP.</p> <p>Values: 0, 1 Default Value: 0 Enabled: Always Parameter Name: MEMC_PERF_LOG_ON</p>
APB Configuration options	
APB Clock Asynchronous to Core Clock	<p>This parameter defines the pclk clock to be synchronous or asynchronous with respect to the controller core_ddrc_core_clk signal. When enabled (asynchronous), the area of the controller is increased for additional instantiation of CDC components. Note: The core_ddrc_core_clk frequency has to be greater or equal to pclk frequency.</p> <p>Values:</p> <ul style="list-style-type: none"> Synchronous (0) Asynchronous (1) <p>Default Value: Asynchronous Enabled: Always Parameter Name: UMCTL2_P_ASYNC_EN</p>

Label	Description
APB Number of Synchronizers	<p>This parameter defines the number of synchronization stages for APB synchronizers.</p> <ul style="list-style-type: none"> ■ 2: Double synchronized ■ 3: Triple synchronized ■ 4: Quadruple synchronized <p>Values: 2, 3, 4 Default Value: 2 Enabled: UMCTL2_AP_ANY_ASYNC == 1 Parameter Name: UMCTL2_ASYNC_REG_N_SYNC</p>
Memory System Interface Parameters	
Memory Data Width (Bits)	<p>This parameter specifies the memory data width of the DQ signal to SDRAM in bits. For HIF configurations, this can be any multiple of 8, with a maximum of 72. For AXI configurations, it must be a power of 2 (8, 16, 32, 64).</p> <ul style="list-style-type: none"> ■ If ECC is enabled, this parameter must be set to 16, 32 or 64, and the ECC byte is additional to the width specified here. ■ If ECC is disabled, a non-power-of-2 configuration allows you to inject your own ECC at the HIF interface if required. <p>Note that this parameter must be set to 16 or 32 in LPDDR5/4/4X Controller (Other memory data width is not supported). Values: 8, 16, 24, 32, 40, 48, 56, 64, 72 Default Value: 32 Enabled: Always Parameter Name: MEMC_DRAM_DATA_WIDTH</p>
Number of Ranks Supported	<p>This parameter specifies the maximum number of ranks supported by DWC_ddrctl (that is, the maximum number of independently-controllable chip selects).</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1 (1) ■ 2 (2) ■ 4 (4) <p>Default Value: (DDRCTL_DDR == 1) ? 2 : 1 Enabled: Always Parameter Name: MEMC_NUM_RANKS</p>
Bus Width Mode support	<p>Selects the supported Bus Width Mode options. 0- Full, Half & Quarter bus width mode 1- Only Full & Half bus width modes 2- Only Full bus width mode</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Full, Half & Quarter Bus Width (0) ■ Only Full & Half Bus Width (1) ■ Only Full Bus Width (2) <p>Default Value: Full, Half & Quarter Bus Width Enabled: Always Parameter Name: DDRCTL_PBW_MODE_SUPPORT</p>

Label	Description
DFI Interface Widths	
Number of DFI Interfaces	<p>It is an internal parameter provided in the GUI for information purposes. For DWC DDR5/4 Controller This parameter specifies the number of DFI interfaces depending on UMCTL2_DUAL_CHANNEL. 1: Single channel configuration 2: Dual channel or Shared-AC configuration</p> <p>For DWC LPDDR5/4/4X Controller This parameter specifies the number of DFI interfaces depending on UMCTL2_DUAL_CHANNEL and MEMC_DRAM_DATA_WIDTH. 1: Single DDRC Single DFI configuration (MEMC_DRAM_DATA_WIDTH==16 and UMCTL2_DUAL_CHANNEL==0) 2: Single DDRC Dual DFI configuration (MEMC_DRAM_DATA_WIDTH==32 or UMCTL2_DUAL_CHANNEL==1) Note that Dual DDRC Dual DFI configuration (i.e.Dual Channel configuration) is not supported.</p> <p>Values: 1, 2</p> <p>Default Value: ((DDRCTL_1DDRC_2DFI == 1) (UMCTL2_DUAL_CHANNEL == 1)) ? 2 : 1</p> <p>Enabled: 0</p> <p>Parameter Name: UMCTL2_NUM_DFI</p>
DFI Data Width With ECC (Bits)	<p>This parameter specifies the width of DFI data bus including ECC (if any). It is an internal parameter provided in the GUI for information purposes.</p> <p>Values: 16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 192, 224, 256, 288, 320, 512, 576</p> <p>Default Value: MEMC_FREQ_RATIO * (MEMC_DRAM_DATA_WIDTH + MEMC_DRAM_ECC_WIDTH) * 2</p> <p>Enabled: 0</p> <p>Parameter Name: MEMC_DFI_TOTAL_DATA_WIDTH</p>
Width of dfi_wrdata_en/dfi_rddata_en/dfi_rddata_valid	<p>This parameter specifies the width of dfi_wrdata_en, dfi_rddata_en, and dfi_rddata_valid. It is an internal parameter provided in the GUI for information purposes.</p> <p>Values: -2147483648, ..., 2147483647</p> <p>Default Value: MEMC_DFI_TOTAL_DATA_WIDTH / 16</p> <p>Enabled: 0</p> <p>Parameter Name: MEMC_DFI_TOTAL_DATAEN_WIDTH</p>
Width of dfi_wrdata_mask	<p>This parameter specifies the width of dfi_wrdata_mask. It is an internal parameter provided in the GUI for information purposes.</p> <p>Values: -2147483648, ..., 2147483647</p> <p>Default Value: UMCTL2_DFI_MASK_PER_NIBBLE ? (MEMC_DFI_DATA_WIDTH + MEMC_DFI_ECC_WIDTH)/4 : (MEMC_DFI_DATA_WIDTH + MEMC_DFI_ECC_WIDTH)/8</p> <p>Enabled: 0</p> <p>Parameter Name: MEMC_DFI_TOTAL_MASK_WIDTH</p>

Label	Description
CAM Configuration Options	
CAM Depth	<p>This parameter specifies the depth (number of entries) of each CAM (read CAM and write CAM).</p> <p>Values: 16, 32, 64</p> <p>Default Value: 32</p> <p>Enabled: Always</p> <p>Parameter Name: MEMC_NO_OF_ENTRY</p>
Tag bits - width of token bus	<p>Specifies the width of token bus. By default, the CAM depth determines the width of the token bus. If an arbiter is present, you can increase the width of the token bus to accommodate port and AXI IDs.</p> <p>Values: UMCTL2_TOKENW, ..., 128</p> <p>Default Value: MEMC_NO_OF_ENTRY == 16 ? 4 : (MEMC_NO_OF_ENTRY == 32 ? 5 : 6)</p> <p>Enabled: UMCTL2_INCL_ARB_OR_CHB == 0</p> <p>Parameter Name: MEMC_HIF_TAGBITS</p>
Write pointer width	<p>Specifies the number of bits provided for write pointers (sent to the controller with write commands, and later returned to the interface to enable data fetches). If an arbiter is present, you can override the width of the write pointer to accommodate port and AXI IDs.</p> <p>Values: 1, ..., 128</p> <p>Default Value: 1</p> <p>Enabled: UMCTL2_INCL_ARB_OR_CHB == 0</p> <p>Parameter Name: MEMC_HIF_WDATA_PTR_BITS</p>
Write data SRAM Configuration Options	
Use External SRAM for Write Data	<p>This parameter specifies the controller to use external or internal SRAM for write data.</p> <p>Values: 0, 1</p> <p>Default Value: 1</p> <p>Enabled: Always</p> <p>Parameter Name: UMCTL2_WDATA_EXTRAM</p>

Label	Description
DDRC Internal Configuration Options	
Register DFI Outputs	<p>This parameter enables registering of all DFI output signals. By default, all the DFI output signals are registered to ensure that timing on the DFI interface is easily met. However, by setting this parameter to 0, it is possible to remove this registering stage, which improves the latency through the controller by one cycle. Set this parameter to 0 only if it can be guaranteed that the synthesis timing of the DFI output signals between controller and PHY can be met in a single cycle. If ECC support is desired (MEMC_ECC_SUPPORT > 0), the DFI outputs are required to be registered (MEMC_REG_DFI_OUT = 1). It is possible to exclude DFI Write data signals from the registering stage by setting MEMC_REG_DFI_OUT_WR_DATA to 0. For more information, see the "Latency Analysis" section in Appendix B, "Controller Performance Details".</p> <p>Values: 0, 1 Default Value: 1 Enabled: MEMC_ECC_SUPPORT == 0 Parameter Name: MEMC_REG_DFI_OUT</p>
Register DFI Write Data Outputs	<p>This parameter enables registering of DFI write data outputs. By default, all the DFI outputs are registered to ensure that timing on the DFI interface is easily met. However, by setting this parameter to 0, it is possible to remove the registering stage of the DFI Write data signals (dfi_wrddata_en, dfi_wrddata and dfi_wrddata_mask), while maintaining the registering stage of all the other DFI output signals. Set this parameter to 0 only if DFI Write Data signals can meet the single cycle synthesis timing requirement between the controller and the PHY. This parameter has a meaning only when MEMC_REG_DFI_OUT is set to 1.</p> <p>Values: 0, 1 Default Value: MEMC_REG_DFI_OUT==1 ? 1 : 0 Enabled: MEMC_REG_DFI_OUT == 1 Parameter Name: MEMC_REG_DFI_OUT_WR_DATA</p>
DDRC Number of Synchronizers	<p>This parameter specifies the number of synchronization stages for DDRC synchronizers (for asynchronous inputs directly to DDRC).</p> <ul style="list-style-type: none"> ■ 2: Double synchronized ■ 3: Triple synchronized ■ 4: Quadruple synchronized <p>Values: 2, 3, 4 Default Value: 2 Enabled: Always Parameter Name: UMCTL2_ASYNC_DDRC_N_SYNC</p>
Number of BSM Modules Required	<p>This parameter specifies the number of BSM modules required.</p> <p>Values: 8, 16, 32, 64, 128, 256 Default Value: (DDRCTL_DYN_BSM_MODE == 0) ? MEMC_NUM_TOTAL_BANKS : ((DDRCTL_DYN_BSM_MODE == 1) ? (MEMC_NUM_TOTAL_BANKS/2) : ((MEMC_NUM_TOTAL_BANKS/2 > 128) ? 128 : (MEMC_NUM_TOTAL_BANKS/2))) Enabled: DDRCTL_DYN_BSM_MODE == 2 Parameter Name: UMCTL2_NUM_BSM</p>

Label	Description
Maximum Number of Banks Supported	<p>This parameter specifies the maximum number of banks supported with a given hardware configuration.</p> <p>Values: 0x8, 0x10, 0x20, 0x40, 0x80, 0x100, 0x200, 0x400, 0x800</p> <p>Default Value: 1<<MEMC_RANKBANK_BITS</p> <p>Enabled: 0</p> <p>Parameter Name: MEMC_NUM_TOTAL_BANKS</p>
Internal Refresh Management(RFM) support	<p>DDRCTL_HW_RFM_CTRL Enables internal Refresh Management control in DDRCTL.</p> <p>Values: 0, 1</p> <p>Default Value: (MEMC_DDR5 == 1) ? 1 : 0</p> <p>Enabled: ((DDRCTL_LPDDR==1) (MEMC_DDR5==1))</p> <p>Parameter Name: DDRCTL_HW_RFM_CTRL</p>
LPDDR54 Specific Configuration Options	
Enable PPT2	<p>This parameter provides optional hardware to utilize Periodic Phase Training 2 (PPT2) which is the LPDDR54 PHY retraining feature.</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: DDRCTL_LPDDR==1</p> <p>Parameter Name: DDRCTL_PPT2</p>
Fast Frequency Change Support	
Number of Frequency Sets Supported	<p>This parameter specifies the number of operational frequencies.</p> <p>Values: 1, 2, 3, 4</p> <p>Default Value: 1</p> <p>Enabled: Always</p> <p>Parameter Name: UMCTL2_FREQUENCY_NUM</p>
Enable Hardware Fast Frequency Change	<p>This parameter provides optional hardware to enable Hardware Fast Frequency Change.</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: (DDRCTL_LPDDR==1) && (DDRCTL_PRODUCT_NAME==4) && (UMCTL2_FREQUENCY_NUM>1)</p> <p>Parameter Name: UMCTL2_HWFFC_EN</p>

Label	Description
QOS	
Enable Variable Priority Read and Write Feature	<p>This parameter enables Variable Priority Read (VPR) and Variable Priority Write (VPW) features.</p> <p>On the read side, this feature allows the use of VPR in addition to Low Priority Read (LPR) and High Priority Read (HPR) priority classes. These three priority classes are intended to be mapped to three traffic classes as follows:</p> <ul style="list-style-type: none"> ■ HPR (High Priority Read): Low Latency ■ VPR (Variable Priority Read): High Bandwidth ■ LPR (Low Priority Read): Best Effort <p>The VPR commands start out behaving like LPR traffic. But, VPR commands have down-counting latency timers associated with them. When the timer reaches 0, the commands marked with VPR are given higher priority over HPR and LPR traffic.</p> <p>On the write side, this feature allows the use of two priority classes in the controller:</p> <ul style="list-style-type: none"> ■ VPW ■ NPW <p>These two priority classes are intended to be mapped to two traffic classes as follows:</p> <ul style="list-style-type: none"> ■ VPW (Variable Priority Write) High Bandwidth ■ NPW (Normal Priority Write): Best Effort <p>The VPW traffic class commands start out behaving like NPW traffic. But, VPW commands have down-counting latency timers associated with them. When the timer reaches 0, the commands marked with VPW are given higher priority over NPW traffic.</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: (((UMCTL2_INCL_ARB==1) && (THEREIS_AXI_PORT==1)) (UMCTL2_INCL_ARB==0))</p> <p>Parameter Name: UMCTL2_VPRW_EN</p>
Timing Optimizations	
Multi-cycle path	<p>DDRCTL_MCP_INCLUDE Enables multicycle paths in synthesis. It is suggested to disable them whenever there are only single cycle paths. For example in LBIST applications</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Disable (0) ■ Enable (1) <p>Default Value: Enable</p> <p>Enabled: Always</p> <p>Parameter Name: DDRCTL_MCP_INCLUDE</p>

14.3 HW Configuration / Multiport Parameters

Table 14-3 HW Configuration / Multiport Parameters

Label	Description
Application Ports	
Number of Host Ports	<p>When specified, this parameter includes logic to implement 1 to 16 host ports. Host port 0 is always included.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1 (1) ■ 2 (2) ■ 3 (3) ■ 4 (4) ■ 5 (5) ■ 6 (6) ■ 7 (7) ■ 8 (8) ■ 9 (9) ■ 10 (10) ■ 11 (11) ■ 12 (12) ■ 13 (13) ■ 14 (14) ■ 15 (15) ■ 16 (16) <p>Default Value: DDRCTL_INCL_CHB==1 && UMCTL2_DUAL_DATA_CHANNEL==1 ? 2 : 1</p> <p>Enabled: UMCTL2_INCL_ARB == 1</p> <p>Parameter Name: UMCTL2_A_NPORTS</p>
Application Address Width	<p>Specifies the width of the application address. A minimum value equal to UMCTL2_MIN_ADDRW is required to be able to address the maximum supported memory size. If a value higher than UMCTL2_MIN_ADDRW is set and system address regions are not enabled, the exceeding MSBs are ignored.</p> <p>Values: MEMC_HIF_MIN_ADDR_WIDTH, ..., 60</p> <p>Default Value: UMCTL2_MIN_ADDRW</p> <p>Enabled: UMCTL2_INCL_ARB == 1</p> <p>Parameter Name: UMCTL2_A_ADDRW</p>
Application ID Width	<p>Specifies the width of the application ID.</p> <p>Values: 1, ..., 32</p> <p>Default Value: 8</p> <p>Enabled: UMCTL2_INCL_ARB == 1</p> <p>Parameter Name: UMCTL2_A_IDW</p>

Label	Description
Application Burst Length Width	Specifies the width of the application burst length. Values: 4, 5, 6, 7, 8 Default Value: (THEREIS_AXI4_PORT == 1) ? 8 : 4 Enabled: UMCTL2_INCL_ARB == 1 Parameter Name: UMCTL2_A_LENW
Number of AXI Exclusive Access Monitors	This parameter specifies the number of AXI Exclusive Access Monitors. <ul style="list-style-type: none"> 0: Exclusive Access Monitoring is not supported. 1-16: Exclusive Access Monitoring is supported with the selected number of monitors. Values: 0, ..., 16 Default Value: 0 Enabled: ((UMCTL2_INCL_ARB==1) && (THEREIS_AXI_PORT==1)) Parameter Name: UMCTL2_EXCL_ACCESS
Enable External Port Priorities	This parameter enables dynamic setting of port priorities externally through the AXI QoS signals (awqos_n and arqos_n). Values: 0, 1 Default Value: 0 Enabled: UMCTL2_INCL_ARB==1 && UMCTL2_A_NPORTS > 1 Parameter Name: UMCTL2_EXT_PORTPRIO
AXI Burst Address Boundary	Specifies the AXI address boundary restriction. AXI transactions must not cross 2**AXI_ADDR_BOUNDARY bytes. The default value of 12 matches the AXI specification of 4K boundary. Values: 12, ..., 32 Default Value: 12 Enabled: UMCTL2_INCL_ARB == 1 Parameter Name: UMCTL2_AXI_ADDR_BOUNDARY
AXI User Signal Width	This parameter specifies the width of the application user signals. Values: 0, ..., 8 Default Value: 0 Enabled: UMCTL2_INCL_ARB == 1 Parameter Name: UMCTL2_AXI_USER_WIDTH
Port 0	
Port n Type (for n = 0; n <= UMCTL2_A_NPORTS-1)	This parameter defines the interface type for the controller; application port n. Values: <ul style="list-style-type: none"> AXI4 (3) Default Value: AXI4 Enabled: UMCTL2_PORT_n == 1 && UMCTL2_INCL_ARB == 1 Parameter Name: UMCTL2_A_TYPE_n

Label	Description
Port n Data Width (for n = 0; n <= UMCTL2_A_NPORTS-1)	<p>This parameter defines the data width of the controller application port n. Valid ranges for AXI4 are 32 to 512.</p> <p>Values: 8, 16, 32, 64, 128, 256, 512</p> <p>Default Value: UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_n!=0 ? MEMC_DFI_DATA_WIDTH : 32</p> <p>Enabled: UMCTL2_A_NPORTS >= (n+1) && UMCTL2_INCL_ARB == 1 && UMCTL2_A_TYPE_n != 0</p> <p>Parameter Name: UMCTL2_PORT_DW_n</p>
Port n Clock (for n = 0; n <= UMCTL2_A_NPORTS-1)	<p>Defines the port n clock to be synchronous or asynchronous with respect to the controller core_ddrc_core_clk. If specified to be asynchronous, clock domain crossing logic is included in the design, which increases the latency and area. A port's clock (aclk_n or hclk_n) is considered synchronous when:</p> <ul style="list-style-type: none"> ■ It is phase aligned and ■ Equal frequency to the controller core_ddrc_core_clk <p>Values:</p> <ul style="list-style-type: none"> ■ Asynchronous (0) ■ Synchronous (1) <p>Default Value: UMCTL2_INCL_ARB==1) ? 0 : (DDRCTL_CHB_SYNC_MODE</p> <p>Enabled: UMCTL2_A_NPORTS>n && UMCTL2_INCL_ARB==1</p> <p>Parameter Name: UMCTL2_A_SYNC_n</p>
Port n Number of Synchronizers (for n = 0; n <= UMCTL2_A_NPORTS-1)	<p>This parameter defines the number of synchronization stages for the asynchronous FIFOs of port n. Applies to both the pop side and the push side.</p> <ul style="list-style-type: none"> ■ 2: Double synchronized ■ 3: Triple synchronized ■ 4: Quadruple synchronized <p>Values: 2, 3, 4</p> <p>Default Value: 2</p> <p>Enabled: UMCTL2_A_NPORTS>n && UMCTL2_A_SYNC_n==0 && UMCTL2_INCL_ARB == 1</p> <p>Parameter Name: UMCTL2_ASYNC_FIFO_N_SYNC_n</p>
Port n Static Virtual Channels Mapping (for n = 0; n <= UMCTL2_A_NPORTS-1)	<p>This parameter enables static virtual channels mapping for port n.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: No</p> <p>Enabled: UMCTL2_A_NPORTS>n && (UMCTL2_A_TYPE_n == 1 UMCTL2_A_TYPE_n == 3) && UMCTL2_INCL_ARB == 1 && UMCTL2_DATA_CHANNEL_INTERLEAVE_EN==0 && (UMCTL2_XPI_USE2RAQ_n == 0 UMCTL2_READ_DATA_INTERLEAVE_EN_n == 1)</p> <p>Parameter Name: UMCTL2_STATIC_VIR_CH_n</p>

Label	Description
Port n Number of Virtual Channels (for n = 0; n <= UMCTL2_A_NPORTS-1)	<p>This parameter defines the number of virtual channels for port n.</p> <p>Values: 1, ..., 64</p> <p>Default Value: (UMCTL2_XPI_USE2RAQ_n == 1 && UMCTL2_READ_DATA_INTERLEAVE_EN_n == 0) ? MEMC_NO_OF_ENTRY : (UMCTL2_XPI_SMALL_SIZED_PORT_n==1 && UMCTL2_XPI_USE2RAQ_n ==1) ? MEMC_NO_OF_ENTRY/2 : 32</p> <p>Enabled: UMCTL2_A_NPORTS>n && (UMCTL2_A_TYPE_n == 1 UMCTL2_A_TYPE_n == 3) && UMCTL2_INCL_ARB == 1 && (UMCTL2_XPI_USE2RAQ_n == 0 UMCTL2_READ_DATA_INTERLEAVE_EN_n == 1)</p> <p>Parameter Name: UMCTL2_NUM_VIR_CH_n</p>
Port n Enable External RAM for RRB (for n = 0; n <= UMCTL2_A_NPORTS-1)	<p>This parameter enables the external RAM for Read Reorder Buffer (RRB) of port n.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Disable (0) ■ Enable (1) <p>Default Value: (UMCTL2_DATA_CHANNEL_INTERLEAVE_EN == 1 && UMCTL2_A_NPORTS >= (n+1) && UMCTL2_INCL_ARB == 1 && UMCTL2_A_TYPE_n != 0) ? 1 : 0</p> <p>Enabled: UMCTL2_A_NPORTS >= (n+1) && UMCTL2_INCL_ARB == 1 && UMCTL2_A_TYPE_n != 0 && UMCTL2_DATA_CHANNEL_INTERLEAVE_EN == 0</p> <p>Parameter Name: UMCTL2_RRB_EXTRAM_n</p>
Port n Enable Retime for External RRB RAM (for n = 0; n <= UMCTL2_A_NPORTS-1)	<p>When selected, Retime registers are implemented to register the RRB RAM data outputs before being used elsewhere in the design.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Disabled (0) ■ Enabled (1) <p>Default Value: Disabled</p> <p>Enabled: UMCTL2_RRB_EXTRAM_n == 1</p> <p>Parameter Name: UMCTL2_RRB_EXTRAM_RETIME_n</p>
Port n Preserve Read/Write Transaction Ordering (for n = 0; n <= UMCTL2_A_NPORTS-1)	<p>If set, this parameter preserves the ordering between a read transaction and a write transaction on Port n. Additional logic is instantiated in the XPI to transport all read and write commands from the application port interface to the HIF interface in the order of acceptance.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0 (0) ■ 1 (1) <p>Default Value: 0</p> <p>Enabled: (UMCTL2_A_NPORTS>0 && (UMCTL2_A_TYPE_0 == 3) && (UMCTL2_DATA_CHANNEL_INTERLEAVE_NS_0==0))</p> <p>Parameter Name: UMCTL2_RDWR_ORDERED_n</p>

Label	Description
Port n Read Address Queues (for n = 0; n <= UMCTL2_A_NPORTS-1)	<p>This parameter enables the dual read address queue for the controller application port n. Each dual address queue XPI consumes two consecutive PA ports.</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: UMCTL2_A_AXI_n == 1 && UMCTL2_DATA_CHANNEL_INTERLEAVE_EN == 0</p> <p>Parameter Name: UMCTL2_XPI_USE2RAQ_n</p>
Port n Enable threshold based VC selection (for n = 0; n <= UMCTL2_A_NPORTS-1)	<p>Enables threshold based VC selection for Read Reorder Buffer (RRB) of port n in configurations that disable read data interleaving. RRB considers a VC for selection only when the number of HIF bursts received from DDRC exceeds the value specified in PCFGR_n.rrb_lock_threshold, or all corresponding HIF bursts for the AXI transaction are returned by DDRC. This feature provides better performance when one AXI burst is translated to multiple DDR bursts but requires more area and might impact on synthesis timing depending process and so on. The size of extra area depends on number of CAM entries. UMCTL2_NUM_VIR_CH_n > 1 is required to get benefit of the feature.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Disable (0) ■ Enable (1) <p>Default Value: (UMCTL2_A_NPORTS >= (n+1) && UMCTL2_INCL_ARB == 1 && UMCTL2_A_TYPE_n != 0 && UMCTL2_READ_DATA_INTERLEAVE_EN_n == 0 && UMCTL2_DATA_CHANNEL_INTERLEAVE_EN == 0) ? 1 : 0</p> <p>Enabled: UMCTL2_A_NPORTS >= (n+1) && UMCTL2_INCL_ARB == 1 && UMCTL2_A_TYPE_n != 0 && UMCTL2_READ_DATA_INTERLEAVE_EN_n == 0 && UMCTL2_DATA_CHANNEL_INTERLEAVE_EN == 0</p> <p>Parameter Name: UMCTL2_RRB_THRESHOLD_EN_n</p>
Port Arbiter	
Port Arbiter Type	<p>Specifies the type of optimization required for the Port Arbiter block The options are: 1) two-cycle arbitration (1 cycle of idle latency) 2) combinatorial (0 cycle of idle latency) Selecting a value of 2 for this parameter in multi-port configurations can have a significant impact on timing closure.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Two cycle (1) ■ Combinatorial (2) <p>Default Value: (UMCTL2_INT_NPORTS < 5) ? 2 : 1</p> <p>Enabled: Always</p> <p>Parameter Name: UMCTL2_PA_OPT_TYPE</p>

Label	Description
Enable Port Arbiter Pagematch feature	<p>This parameter enables the Port Arbiter (PA) pagematch feature in the hardware. This feature is not recommended if there is a timing closure challenge due to PA. For instance, when there are many ports, the pagematch feature can be disabled to improve synthesis timing.</p> <p>Values: 0, 1</p> <p>Default Value: 1</p> <p>Enabled: UMCTL2_INCL_ARB==1</p> <p>Parameter Name: UMCTL2_PAGEMATCH_EN</p>
External RAM Interface (information only)	
Write Data RAM Data Width	<p>This parameter specifies the data width of the external write data SRAM. It is an internal parameter provided in the GUI for information purposes and is derived from MEMC_DRAM_DATA_WIDTH, MEMC_FREQ_RATIO, and MEMC_ECC_SUPPORT.</p> <p>Values: -2147483648, ..., 2147483647</p> <p>Default Value: UMCTL2_INCL_ARB_OR_CHB == 0 && MEMC_SIDEBAND_ECC_EN==1 && UMCTL2_ECC_TEST_MODE_EN==1) ? (MEMC_OPT_WDATARAM == 1)? (MEMC_DRAM_DATA_WIDTH*MEMC_FREQ_RATIO*2+MEMC_DFI_ECC_WIDTH*2) : (MEMC_DRAM_DATA_WIDTH*MEMC_FREQ_RATIO*2+MEMC_DFI_ECC_WIDTH) : (MEMC_OPT_WDATARAM == 1 && MEMC_DRAM_DATA_WIDTH == 72)? ((MEMC_DRAM_DATA_WIDTH+8)*MEMC_FREQ_RATIO*2) : (MEMC_DRAM_DATA_WIDTH*MEMC_FREQ_RATIO*2</p> <p>Enabled: 0</p> <p>Parameter Name: UMCTL2_WDATARAM_DW</p>
Write Data RAM OCPAR/OCECC Width	<p>Specifies the parity length of the external write data SRAM. It is an internal parameter provided in the GUI for information purpose.</p> <p>Values: -2147483648, ..., 2147483647</p> <p>Default Value: (UMCTL2_OCPAR_EN == 1 UMCTL2_OCECC_EN == 1) ? UMCTL2_WDATARAM_PAR_DW : 0</p> <p>Enabled: 0</p> <p>Parameter Name: UMCTL2_WDATARAM_PAR_DW_GUI</p>
Write Data RAM Depth	<p>This parameter specifies the depth of the external write data SRAM. It is an internal parameter provided in the GUI for information purposes and is derived from the address width of the external write data SRAM (UMCTL2_WDATARAM_AW).</p> <p>Values: -2147483648, ..., 2147483647</p> <p>Default Value: 1<< UMCTL2_WDATARAM_AW</p> <p>Enabled: 0</p> <p>Parameter Name: UMCTL2_WDATARAM_DEPTH</p>

Label	Description
Write Data RAM Address Width	<p>This parameter specifies the address width of the external write data SRAM. It is an internal parameter provided in the GUI for information purposes and is derived from the CAM size (MEMC_NO_OF_ENTRY), MEMC_BURST_LENGTH, and MEMC_FREQ_RATIO.</p> <p>Values: -2147483648, ..., 2147483647</p> <p>Default Value: $\text{MEMC_OPT_WDATARAM} == 1 ? ((\text{MEMC_WRDATA_8_CYCLES} == 1) ? (\text{MEMC_WRCMD_ENTRY_BITS} + 2) : (\text{MEMC_WRDATA_4_CYCLES} == 1) ? (\text{MEMC_WRCMD_ENTRY_BITS} + 1) : (\text{MEMC_WRCMD_ENTRY_BITS})) : ((\text{MEMC_WRDATA_8_CYCLES} == 1) ? (\text{MEMC_WRCMD_ENTRY_BITS} + 3) : (\text{MEMC_WRDATA_4_CYCLES} == 1) ? (\text{MEMC_WRCMD_ENTRY_BITS} + 2) : (\text{MEMC_WRCMD_ENTRY_BITS} + 1))$</p> <p>Enabled: 0</p> <p>Parameter Name: UMCTL2_WDATARAM_AW</p>
Read Reorder Buffer Data Width	<p>This parameter specifies the Read Reorder Buffer (RRB) External Data RAM Interface Data Width. It is an internal parameter provided in the GUI for information purposes.</p> <p>Values: -2147483648, ..., 2147483647</p> <p>Default Value: $\text{MEMC_DRAM_DATA_WIDTH} * \text{MEMC_FREQ_RATIO} * 2$</p> <p>Enabled: 0</p> <p>Parameter Name: UMCTL2_RDATARAM_DW</p>
Read Reorder Buffer OCPAR/OCECC Width	<p>Specifies the Read Reorder Buffer (RRB) External Data RAM Interface Parity Width. It is an internal parameter provided in the GUI for information purpose.</p> <p>Values: -2147483648, ..., 2147483647</p> <p>Default Value: $(\text{UMCTL2_OCPAR_EN} == 1 \parallel \text{UMCTL2_OCECC_EN} == 1) ? \text{UMCTL2_DATARAM_PAR_DW} : 0$</p> <p>Enabled: 0</p> <p>Parameter Name: UMCTL2_DATARAM_PAR_DW_GUI</p>
Read Reorder Buffer Depth	<p>This parameter specifies the Read Reorder Buffer (RRB) External Data RAM Interface Depth. It is an internal parameter provided in the GUI for information purposes.</p> <p>Values: -2147483648, ..., 2147483647</p> <p>Default Value: $\text{MEMC_NO_OF_ENTRY} * (\text{MEMC_BURST_LENGTH} / (\text{MEMC_FREQ_RATIO} * 2))$</p> <p>Enabled: 0</p> <p>Parameter Name: UMCTL2_RDATARAM_DEPTH</p>
Read Reorder Buffer Address Width	<p>This parameter specifies the Read Reorder Buffer (RRB) External Data RAM Interface Address Width. It is an internal parameter provided in the GUI for information purposes.</p> <p>Values: -2147483648, ..., 2147483647</p> <p>Default Value: $\text{MEMC_NO_OF_ENTRY} * \text{MEMC_BURST_LENGTH} * \text{MEMC_FREQ_RATIO}$</p> <p>Enabled: 0</p> <p>Parameter Name: UMCTL2_RDATARAM_AW</p>

Label	Description
Timing Optimizations	
Enable XPI Write Address Retime	<p>This parameter enables the XPI write address retime (that is, pipelines XPI write address output to PA).</p> <p>This parameter introduces extra cycle of latency on the write address channel. It can be used for multi-port configurations to improve timing.</p> <p>A retime is automatically instantiated in the xpi RMW generator. Therefore, when RMW is used, this parameter is disabled.</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: UMCTL2_INCL_ARB == 1 && UMCTL2_XPI_USE_RMW == 0</p> <p>Parameter Name: UMCTL2_XPI_USE_WAR</p>
Enable XPI Read Address Output Retime	<p>This parameter enables the XPI read address output retime (that is, pipelines XPI write address output to PA).</p> <p>This parameter introduces an extra cycle of latency on the read address channel. It can be used for multi-port configurations to improve timing.</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: UMCTL2_INCL_ARB == 1 && THEREIS_USE2RAQ == 0</p> <p>Parameter Name: UMCTL2_XPI_USE_RAR</p>
Enable XPI Read Address Input Retime	<p>This parameter enables the XPI read address input retime (that is, pipelines XPI write address input before the QoS mapper).</p> <p>This parameter introduces an extra cycle of latency on the read address channel. It can be used to improve timing.</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: UMCTL2_INCL_ARB == 1</p> <p>Parameter Name: UMCTL2_XPI_USE_INPUT_RAR</p>
Enable XPI RRB Data Retime	<p>This parameter enables the XPI RRB data retime (that is, pipelines XPI at the output of RRB).</p> <p>This parameter introduces an extra cycle of latency on the read data channel. It can be used in dual data channel configurations to improve timing.</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: UMCTL2_INCL_ARB == 1 && UMCTL2_DATA_CHANNEL_INTERLEAVE_EN == 1</p> <p>Parameter Name: UMCTL2_XPI_USE_RDR</p>

Label	Description
Enable XPI Read Parity Retime	<p>This parameter enables the XPI read data/parity retime (that is, pipelines XPI data and parity at the AXI interface). This parameter introduces an extra cycle of latency on the read data channel. It can be used in on-chip parity configurations to improve timing.</p> <p>Values: 0, 1 Default Value: 0 Enabled: (UMCTL2_INCL_ARB == 1 && (UMCTL2_OCPAR_EN == 1 UMCTL2_OCECC_EN == 1)) Parameter Name: UMCTL2_XPI_USE_RPR</p>
Disable XPI RMW Bypass Path	<p>This parameter is used to enable/disable the bypass path for command and data in XPI RMW module. 1: The bypass path in XPI RMW is disabled. XPI RMW module introduces 1 cycle additional latency for write commands and data. 0: The Bypass path is included in XPI RMW parallel to the Store and Forward logic. Bypass path will be active if ECC is disabled, Data Mask (DM) is enabled and Programmable SnF is disabled. The bypass path is active/inactive depending on the register configuration.</p> <p>Values: 0, 1 Default Value: ((UMCTL2_PA_OPT_TYPE==1) && (UMCTL2_INT_NPORTS<3)) ? 0 : 1 Enabled: DDRCTL_SYS_INTF==1 Parameter Name: DDRCTL_XPI_USE_RMWR</p>

14.4 HW Configuration / AXI Parameters

Table 14-4 HW Configuration / AXI Parameters

Label	Description
AXI Interface Ports - Low Power	
AXI Low Power Idle Wait	<p>This parameter specifies the number of cycles after the last active transaction to de-assertion of the cactive signal.</p> <p>Values: 0, ..., 1048576</p> <p>Default Value: 0</p> <p>Enabled: UMCTL2_INCL_ARB == 1</p> <p>Parameter Name: UMCTL2_AXI_LOWPWR_NOPX_CNT</p>
AXI Interface Ports - System Address Regions	
Number of System Address Regions	<p>Specifies the number of System Address Regions.</p> <p>Specifies how many distinct address regions to be decoded in the application system address space.</p> <ul style="list-style-type: none"> ■ Minimum value 0 ■ Maximum value 4 ■ Default value 0 <p>If set to 0, no regions are specified and addresses are assumed from the address 0.</p> <p>Values: 0, ..., 4</p> <p>Default Value: 0</p> <p>Enabled: UMCTL2_INCL_ARB == 1 && UMCTL2_DATA_CHANNEL_INTERLEAVE_EN==0</p> <p>Parameter Name: UMCTL2_A_NSAR</p>
System Address Regions Minimum Block Size	<p>Specifies the minimum block size for system address regions, ranging from 256 MB to 32GB. Determines the number of most significant system address bits that are used to decode address regions. Base addresses for each region must be aligned to this minimum block size.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 256MB (1) ■ 512MB (2) ■ 1GB (3) ■ 2GB (4) ■ 4GB (5) ■ 8GB (6) ■ 16GB (7) ■ 32GB (8) <p>Default Value: 256MB</p> <p>Enabled: UMCTL2_INCL_ARB == 1 && UMCTL2_A_NSAR > 0</p> <p>Parameter Name: UMCTL2_SARMINSIZE</p>

Label	Description
Port 0	
Port n AXI Read Address Queue Depth (for n = 0; n <= UMCTL2_A_NPORTS-1)	<p>Determines how many AXI addresses can be stored in the read address buffer of Port n. Each address represents an AXI burst transaction.</p> <p>Values: 2, ..., 32</p> <p>Default Value: 4</p> <p>Enabled: UMCTL2_A_AXI_n == 1</p> <p>Parameter Name: UMCTL2_AXI_RAQD_n</p>
Port n AXI Write Address Queue Depth (for n = 0; n <= UMCTL2_A_NPORTS-1)	<p>Determines how many AXI addresses can be stored in the write address buffer of Port n. Each address represents an AXI burst transaction.</p> <p>Values: 2, ..., 32</p> <p>Default Value: 4</p> <p>Enabled: UMCTL2_A_AXI_n == 1</p> <p>Parameter Name: UMCTL2_AXI_WAQD_n</p>
Port n AXI Read Data Queue Depth (for n = 0; n <= UMCTL2_A_NPORTS-1)	<p>Determines how many AXI burst beats can be stored in the read data buffer of Port n. Set the read data buffer to an appropriate depth to allow continuous streaming of read data in the end application. If set too small, the interface will be functional, but performance might be impacted as the buffer might not have sufficient storage to permit a continuous stream of read commands.</p> <p>For configurations where UMCTL2_A_SYNC_n = 1, the minimum value to permit continuous streaming is 2. A higher value may be required depending on your application.</p> <p>For configurations where UMCTL2_A_SYNC_n = 0, the minimum value to permit continuous streaming is 10. A higher value might be required depending on the AXI to core clock ratio, as well as your application.</p> <p>Values: 2, ..., 128</p> <p>Default Value: (UMCTL2_A_SYNC_n == 1)? 2 : 10</p> <p>Enabled: UMCTL2_A_AXI_n == 1</p> <p>Parameter Name: UMCTL2_AXI_RDQD_n</p>

Label	Description
Port n AXI Write Data Queue Depth (for $n = 0$; $n \leq \text{UMCTL2_A_NPORTS}-1$)	<p>Determines how many AXI burst beats can be stored in the write data buffer of Port n. Set the write data buffer to an appropriate depth to allow continuous streaming of write data in the end application. If set too small, the interface will be functional, but performance might be impacted as the buffer might not have sufficient storage to permit a continuous stream of write commands.</p> <p>For configurations where $\text{UMCTL2_A_SYNC}_n = 1$, the minimum value to permit continuous streaming is 2. A higher value might be required depending on your application.</p> <p>For configurations where $\text{UMCTL2_A_SYNC}_n = 0$, the minimum value to permit continuous streaming is 10.</p> <p>Increase by at least one if AXI master issues AWVALID and corresponding WVALID on the same cycle.</p> <p>Increase by at least one when $\text{UMCTL2_XPI_USE_WAR}=1$. A higher value might be required depending on the AXI to core clock ratio as well as your application.</p> <p>Values: 2, ..., 128</p> <p>Default Value: $(\text{UMCTL2_A_SYNC}_n == 1)? 2 : 10$</p> <p>Enabled: $\text{UMCTL2_A_AXI}_n == 1$</p> <p>Parameter Name: UMCTL2_AXI_WDQD_n</p>
Port n AXI Write Response Queue Depth (for $n = 0$; $n \leq \text{UMCTL2_A_NPORTS}-1$)	<p>UMCTL2_AXI_WRQD_n: Determines how many AXI write responses can be stored in the write response buffer of Port n. Each entry represents a response to an AXI write burst transaction. Set the write response buffer to:</p> <ul style="list-style-type: none"> ■ 2 for configurations where $\text{UMCTL2_A_SYNC}_n = 1$. ■ 10 for configurations where $\text{UMCTL_A_SYNC}_n = 0$. <p>This allows the controller to store enough write responses in the write response buffer so that the controller does not stall a continuous stream of short write transactions (with $\text{awlen} = 0$) to wait for free storage space in the write response buffer. May be increased if additional write response buffering in the controller is required.</p> <p>If set to value less than 10, the interface will be functional, but performance might be impacted as the buffer might not have sufficient storage to permit a continuous stream of write transactions. Note: the performance impact may be hidden if awlen is greater than 0.</p> <p>Values: 2, ..., 64</p> <p>Default Value: $(\text{UMCTL2_A_SYNC}_n == 1)? 2 : 10$</p> <p>Enabled: $\text{UMCTL2_A_AXI}_n == 1$</p> <p>Parameter Name: UMCTL2_AXI_WRQD_n</p>

Label	Description
Port n Read Data Interleaving Enable (for n = 0; n <= UMCTL2_A_NPORTS-1)	<p>This parameter enables the interleaving of the read data of transactions with different ARID fields.</p> <ul style="list-style-type: none"> ■ Read data interleaving may occur at memory burst boundaries. ■ Read data interleaving can be disabled if this parameter is set to 0. If read data interleaving is disabled, read data reordering in Read Reorder Buffer may introduce further latency. For example, a short AXI burst stays in the RRB buffer and does not interrupt a longer burst that has started earlier. ■ It is recommended to enable read data interleaving for improved read data latency. <p>Values:</p> <ul style="list-style-type: none"> ■ No (0) ■ Yes (1) <p>Default Value: (UMCTL2_A_AXI_n==0) ? 0 : 1</p> <p>Enabled: ((UMCTL2_A_AXI_n==0) (UMCTL2_DATA_CHANNEL_INTERLEAVE_EN==1)) ? 0 : 1</p> <p>Parameter Name: UMCTL2_READ_DATA_INTERLEAVE_EN_n</p>

14.5 HW Configuration / Multi-channel Parameters

Table 14-5 HW Configuration / Multi-channel Parameters

Label	Description
Multi-channel Specific Configuration Options	
Enable Dual Channel Support	<p>This parameter enables Dual Channel support. This feature is under access control. For more information, contact Synopsys.</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: ((DDRCTL_DDR==1) ((DDRCTL_LPDDR==1) && (MEMC_ECC_SUPPORT==0) && (UMCTL2_SBR_EN==0) && (UMCTL2_DUAL_HIF==0) && (UMCTL2_NUM_LRANKS_TOTAL<8)))</p> <p>Parameter Name: UMCTL2_DUAL_CHANNEL</p>
Number of LPDDR4 Initialization Handshake Interface Synchronizers	<p>This parameter specifies the number of synchronization stages for LPDDR4 Initialization Handshake Interface synchronizers.</p> <ul style="list-style-type: none"> ■ 2: Double synchronized ■ 3: Triple synchronized ■ 4: Quadruple synchronized <p>Values: 2, 3, 4</p> <p>Default Value: 2</p> <p>Enabled: DDRCTL_LPDDR==1 && UMCTL2_LPDDR4_DUAL_CHANNEL==0</p> <p>Parameter Name: UMCTL2_ASYNC_LP4DCI_N_SYNC</p>
DDRCTL_DDR_DCH_HBW	<p>Enables half bus width operation for DFI data signals in DDR dual channel configurations</p> <ul style="list-style-type: none"> ■ When this is 0, output full bus width signals from both channels to DFI data interface ■ When this is 1, output half bus width signals from both channels to DFI data interface <p>Values: 0, 1</p> <p>Default Value: (DDRCTL_DDR_DUAL_CHANNEL == 1) ? 1 : 0</p> <p>Enabled: DDRCTL_DDR_DUAL_CHANNEL == 1</p> <p>Parameter Name: DDRCTL_DDR_DCH_HBW</p>
Enable Data Channel interleaving	<p>Enables the Data Channel interleaving in XPI:</p> <ul style="list-style-type: none"> ■ When enabled, each port drives dynamically both data channels based on the address. ■ When disabled, each port statically drives only one data channel based on software settings. <p>This feature is available only in designs where Shared-AC, DDR4 or LPDDR4 Dual Channel is used.</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: ((UMCTL2_DUAL_DATA_CHANNEL==1 UMCTL2_SHARED_AC==1) && (UMCTL2_INCL_ARB == 1))</p> <p>Parameter Name: UMCTL2_DATA_CHANNEL_INTERLEAVE_EN</p>

14.6 HW Configuration / CHI Bridge settings Parameters

Table 14-6 HW Configuration / CHI Bridge settings Parameters

Label	Description
CHI Interface properties	
CHI architecture revision	<p>CHI Issue compliance Flit attributes like TxnID width depend on this.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ CHI-C (2) ■ CHI-D (3) ■ CHI-E (4) <p>Default Value: CHI-D</p> <p>Enabled: Always</p> <p>Parameter Name: DDRCTL_CHB_VERSION</p>
CHI Address width	<p>CHB Req Flit Address Width.</p> <p>Values: 44, ..., 52</p> <p>Default Value: 44</p> <p>Enabled: DDRCTL_INCL_CHB == 1</p> <p>Parameter Name: DDRCTL_CHB_ADRW</p>
CHI bus Data width	<p>TxDAT and RxDAT Data Width.</p> <p>Values: 128, 256, 512</p> <p>Default Value: 256</p> <p>Enabled: 0</p> <p>Parameter Name: DDRCTL_CHB_DW</p>
NodeID width	<p>CHI Flit NodeID Width</p> <p>Values: 7, ..., 11</p> <p>Default Value: 7</p> <p>Enabled: Always</p> <p>Parameter Name: DDRCTL_CHB_NIDW</p>
MPAM Support	<p>Enable MPAM Support in DDRCTL</p> <ul style="list-style-type: none"> ■ 0 : MPAM Feature no supported by controller Controller expects MPAMID field to be absent ■ 1 : MPAM Feature supported by controller <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: DDRCTL_CHB_VERSION > 2 && DDRCTL_INCL_CHB==1</p> <p>Parameter Name: DDRCTL_CHB_MPAM_EN</p>

Label	Description
DataCheck Support	<p>Enable Data Check Flit field, i.e. DataCheck field exists in the DAT flit.</p> <p>Values: 0, 1</p> <p>Default Value: (DDRCTL_CHB_DATA_CHECK_EN DDRCTL_CHB_KBD_ECC_EN) && DDRCTL_INCL_CHB</p> <p>Enabled: DDRCTL_INCL_CHB==1</p> <p>Parameter Name: DDRCTL_CHB_DCHK_EN</p>
Poison Support	<p>Enables Poison flit field, i.e. Poison flit field exists in the DAT flit.</p> <p>Values: 0, 1</p> <p>Default Value: (DDRCTL_CHB_DATA_POIS_EN DDRCTL_CHB_KBD_ECC_EN) && DDRCTL_INCL_CHB</p> <p>Enabled: DDRCTL_INCL_CHB==1</p> <p>Parameter Name: DDRCTL_CHB_POIS_EN</p>
Poison/DataCheck granularity	<p>Defines Posion and DataCheck granularity at CHI interface Set to 64 : For CHI Spec defined DataCheck and Poison width Set to 128 : For custom 128/9/1 Data/ECC/Posion encoding scheme When set to 128, DataCheck field width reduces to [(DDRCTL_CHB_DW/128)*9bits] wide</p> <p>Values: 64, 128</p> <p>Default Value: (DDRCTL_CHB_DATA_POIS_EN==0) ? 128 : 64</p> <p>Enabled: DDRCTL_CHB_KBD_ECC_EN</p> <p>Parameter Name: DDRCTL_NUM_BITS_PER_KBD</p>
DataCheck width when transporting ECC	<p>The number of ECC bits per RXDAT/TXDAT flit or CHI data beat. ECC encoding scheme depends on DDRCTL_NUM_BITS_PER_KBD</p> <p>Values: 1, ..., 64</p> <p>Default Value: DDRCTL_NUM_BITS_PER_KBD==128) ? (DDRCTL_CHB_DW/128)*9 : (DDRCTL_CHB_DW/8</p> <p>Enabled: 0</p> <p>Parameter Name: DDRCTL_CHB_CHI_ECCW</p>
Total L-Credit supported by individual links	<p>CHI Max number of supported Lcredits by individual links. Common configuration applicable for all DDRCTL links</p> <p>Values: -2147483648, ..., 2147483647</p> <p>Default Value: 15</p> <p>Enabled: 0</p> <p>Parameter Name: DDRCTL_CHB_MAX_LCRD</p>
Transaction layer settings	
Read Protocol queue Size	<p>Protocol queues are CHB's outstanding request buffers. This parameter determines the Size/Depth of the read outstanding queue in CHB</p> <p>Values: 32, 64, 128</p> <p>Default Value: 64</p> <p>Enabled: Always</p> <p>Parameter Name: DDRCTL_CHB_RD_PROTQ_SIZE</p>

Label	Description
Write Protocol queue Size	<p>Protocol queues are CHB's outstanding request buffers. This parameter determines the Size/Depth of the write outstanding queue in CHB NOTE: The setting here impacts the size of wrb sram</p> <p>Values: 32, 64, 128</p> <p>Default Value: 64</p> <p>Enabled: Always</p> <p>Parameter Name: DDRCTL_CHB_WR_PROTQ_SIZE</p>
Trustzone configuration	
Trustzone support	<p>Trustzone for CHI</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: DDRCTL_INCL_CHB==1</p> <p>Parameter Name: DDRCTL_CHB_TZ_EN</p>
Trustzone regions	<p>Configures the number of trustzone address regions supported</p> <p>Values: 4, 8</p> <p>Default Value: 4</p> <p>Enabled: DDRCTL_CHB_TZ_EN</p> <p>Parameter Name: DDRCTL_CHB_TSZ_REG_NUM</p>
RAM Pipeline/Latency settings	
CHB Write Buffer RAM Write Address/Data Pipelining	<p>Register the CHB Write Buffer RAM write address and data interface signals. Enabling this register stage allows you to trade latency and gates for ease of timing closure.</p> <p>Values: 0, 1</p> <p>Default Value: 1</p> <p>Enabled: Always</p> <p>Parameter Name: DDRCTL_CHB_WRB_RAM_WR_REG_OUT</p>
CHB Write Buffer RAM Read Address Pipelining	<p>Register the CHB Write Buffer RAM read address interface signals. Enabling this register stage allows you to trade latency and gates for ease of timing closure.</p> <p>Values: 0, 1</p> <p>Default Value: 1</p> <p>Enabled: Always</p> <p>Parameter Name: DDRCTL_CHB_WRB_RAM_RD_ADDR_REG_OUT</p>
CHB Write Buffer RAM Read Data Pipelining	<p>Register the CHB Write Buffer RAM read data interface signals. Enabling this register stage allows you to trade latency and gates for ease of timing closure. Pipe register is implemented inside CHB</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: Always</p> <p>Parameter Name: DDRCTL_CHB_WRB_RAM_RD_DATA_REG_IN</p>

Label	Description
CHB Read Buffer RAM Write Address/Data Pipelining	Register the CHB Read Buffer RAM write address and data interface signals. Enabling this register stage allows you to trade latency and gates for ease of timing closure. Values: 0, 1 Default Value: 0 Enabled: Always Parameter Name: DDRCTL_CHB_RDB_RAM_WR_REG_OUT
CHB Read Buffer RAM Read Address Pipelining	Register the CHB Read Buffer RAM read address interface signals. Enabling this register stage allows you to trade latency and gates for ease of timing closure. Values: 0, 1 Default Value: DDRCTL_CHB_RDB_RAM_WR_REG_OUT Enabled: Always Parameter Name: DDRCTL_CHB_RDB_RAM_RD_ADDR_REG_OUT
CHB Read Buffer RAM Read Data Pipelining	Register the CHB Read Buffer RAM read data interface signals. Enabling this register stage allows you to trade latency and gates for ease of timing closure. Pipe register is implemented inside CHB Values: 0, 1 Default Value: 0 Enabled: Always Parameter Name: DDRCTL_CHB_RDB_RAM_RD_DATA_REG_IN
Clock synchronizaton settings	
Enable sync mode operation	When selected CHI clock and CORE clock are considered synchronous. DDRCTL removes synchronizers between chi_clk and core_ddrc_core_clk domains Configuration applies to all CHI ports. Values: 0, 1 Default Value: 0 Enabled: DDRCTL_INCL_CHB == 1 Parameter Name: DDRCTL_CHB_SYNC_MODE
Number of sync stages from CHI clock to CORE clock	SYNC depth when synchronizing from CHI domain to CORE domain. <ul style="list-style-type: none"> ■ 0: Reserved (used internally when in Synchronous mode) ■ 2: Double synchronized ■ 3: Triple synchronized ■ 4: Quadruple synchronize Values: 0, 2, 3, 4 Default Value: DDRCTL_CHB_SYNC_MODE ? 0 : 2 Enabled: DDRCTL_CHB_SYNC_MODE==0 Parameter Name: DDRCTL_CHB_CHI2CORE_SYNCD

Label	Description
Number of sync stages from CORE clock to CHI clock	<p>SYNC depth when synchronizing from CORE domain to CHI domain.</p> <ul style="list-style-type: none">■ 0: Reserved (used internally when in Synchronous mode)■ 2: Double synchronized■ 3: Triple synchronized■ 4: Quadruple synchronize <p>Values: 0, 2, 3, 4 Default Value: DDRCTL_CHB_SYNC_MODE ? 0 : 2 Enabled: DDRCTL_CHB_SYNC_MODE==0 Parameter Name: DDRCTL_CHB_CORE2CHI_SYNC</p>

14.7 HW Configuration / Reliability Features Parameters

Table 14-7 HW Configuration / Reliability Features Parameters

Label	Description
Memory ECC	
ECC Supported	<p>This parameter enables ECC support. This feature is available only when the DRAM bus width is 16, 32, or 64 bits. The following are the supported ECC types:</p> <ul style="list-style-type: none"> ■ SECEDED ECC ■ Advanced ECC <p>ECC is available in the following modes:</p> <ul style="list-style-type: none"> ■ Full Bus Width (FBW) ■ Half Bus Width (HBW) ■ Quarter Bus Width (QBW) <p>The following ECC codes apply for Single-beat SECEDED ECC:</p> <ul style="list-style-type: none"> ■ 64-bit SDRAM data width: SDRAM data + ECC width is 64+8 (FBW), 32+7 (HBW) and 16+6 (QBW) ■ 32-bit SDRAM data width: SDRAM data + ECC width is 32+7 (FBW), 16+6 (HBW) and 8+5 (QBW) ■ 16-bit SDRAM data width: SDRAM data + ECC width is 16+6 (FBW) and 8+5 (HBW) <p>The following ECC codes apply for Multi-beat SECEDED ECC:</p> <ul style="list-style-type: none"> ■ 64-bit SDRAM data width: SDRAM data + ECC width is 32+4 (HBW) ■ 32-bit SDRAM data width: SDRAM data + ECC width is 32+4 (FBW) <p>For Advanced ECC:</p> <ul style="list-style-type: none"> ■ The ECC code always is 256+32 for Advanced ECC X4. ■ When the SDRAM data width is less than 64, unused bytes in the data word are padded with 0 so that ECC is calculated more than 128 data bits. ■ It is applicable when sideband ECC is enabled for DDR4/DDR5 RDIMM devices <p>The advanced ECC feature is under access control. For more information, contact Synopsys.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ No ECC (0) ■ SECEDED ECC (1) ■ SECEDED ECC + Advanced ECC X4 (3) <p>Default Value: No ECC</p> <p>Enabled: MEMC_DRAM_DATA_WIDTH == 16 MEMC_DRAM_DATA_WIDTH == 32 MEMC_DRAM_DATA_WIDTH == 64</p> <p>Parameter Name: MEMC_ECC_SUPPORT</p>

Label	Description
Enable RMW	<p>This parameter enables read-modify-write commands. By default, this parameter is set for ECC configurations and unset for non-ECC configurations. If read-modify-write commands are disabled, sub-sized write accesses of sizes less than the full memory width are not allowed.</p> <ul style="list-style-type: none"> ■ For LPDDR4 HIF configurations, if MEMC_USE_RMW is disabled, only full BL16/BL8/BC4 bursts are allowed if data masks are disabled (DBICTL.dm_en = 0). ■ For LPDDR4 AXI configurations, MEMC_USE_RMW must be enabled if data masks are disabled (DBICTL.dm_en = 0). <p>Values: 0, 1 Default Value: 1 Enabled: DDRCTL_SYS_INTF !=1 Parameter Name: MEMC_USE_RMW</p>
Enable Sideband ECC	<p>This parameter enables Sideband ECC. When enabled, an additional data bus for ECC is used; therefore, the actual DRAM data width is greater than MEMC_DRAM_DATA_WIDTH.</p> <p>Values: 0, 1 Default Value: MEMC_ECC_SUPPORT>0 Enabled: MEMC_ECC_SUPPORT>0 Parameter Name: MEMC_SIDEHAND_ECC</p>
Enable Inline ECC	<p>This parameter enables Inline ECC. When enabled, an additional data bus for ECC is not required, therefore, the actual DRAM data width is equal to MEMC_DRAM_DATA_WIDTH. ECC parity is stored with the data without using a dedicated sideband memory device. This feature is under access control. For more information, contact Synopsys.</p> <p>Values: 0, 1 Default Value: 0 Enabled: MEMC_ECC_SUPPORT==1 && DDRCTL_LPDDR && UMCTL2_PARTIAL_WR==1 && UMCTL2_DUAL_HIF==0 && MEMC_BYPASS==0 && UMCTL2_DDR4_MRAM_EN==0 && MEMC_OPT_TIMING==1 Parameter Name: MEMC_INLINE_ECC</p>
Enable ECC Scrubber Block	<p>This parameter enables the ECC scrubber block. When set, this parameter instantiates the ECC scrubber block (SBR) that executes periodic background read commands to the DDRC. If enabled, SBR consumes one of the ports of the Port Arbiter (PA). Internally, SBR is always the last port. ECC support must be enabled to use this feature.</p> <p>Values: 0, 1 Default Value: 0 Enabled: UMCTL2_INCL_ARB_OR_CHB==1 && MEMC_ECC_SUPPORT>0 && MEMC_USE_RMW==1 Parameter Name: UMCTL2_SBR_EN</p>

Label	Description
SBR RMW FIFO Depth	<p>In Sideband ECC Configurations, the RMW FIFO is instantiated in the Scrubber to hold the addresses of the correctable error responses.</p> <p>Values: 4, ..., 32</p> <p>Default Value: 4</p> <p>Enabled: UMCTL2_INCL_ARB==1&&MEMC_ECC_SUPPORT>0&&MEMC_SIDEHAND_ECC==1&&UMCTL2_SBR_EN==1&&DDRCTL_UMCTL5==1</p> <p>Parameter Name: DDRCTL_SBR_RMW_FIFO_DEPTH</p>
ADVECC Decoder Pipeline Enable	<p>Instantiates logic to break the timing for ADVECC RSD decoder by adding 1 pipeline stage.</p> <p>When enabled, it introduced one more level of pipeline register for the decoded read data, thereby, increasing the read latency by 1 more clock cycle</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: ((MEMC_ECC_SUPPORT==2) (MEMC_ECC_SUPPORT==3))</p> <p>Parameter Name: DDRCTL_RSD_PIPELINES</p>
Enhanced ECC Error Reporting	<p>This parameter will enable the following:</p> <ul style="list-style-type: none"> ■ Per rank CE counters ■ Programmable CE threshold ■ Threshold based CE interrupt ■ Leaky bucket algorithm for CE counters. <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: (MEMC_ECC_SUPPORT>0)&&(MEMC_SIDEHAND_ECC_EN==1)&&(DDRCTL_DDR==1)</p> <p>Parameter Name: DDRCTL_ENH_ECC_REPORT_EN</p>
On-chip Reliability	
Enable On-Chip Parity	<p>This parameter enables the On-Chip Parity feature.</p> <p>When enabled, the controller instantiates the necessary logic to enable on-chip parity protection: address and data paths.</p> <p>This feature is available only in designs where the arbiter is used (UMCTL2_INCL_ARB=1) and external WDATA RAM is used (UMCTL2_WDATA_EXTRAM=1).</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: ((UMCTL2_INCL_ARB==1) && (UMCTL2_WDATA_EXTRAM==1))</p> <p>Parameter Name: UMCTL2_OCPAR_EN</p>

Label	Description
On-Chip Parity Address Width	<p>This parameter specifies the address parity width at AXI. The options are:</p> <ul style="list-style-type: none"> ■ Single parity bit ■ One bit per byte of address <p>Values:</p> <ul style="list-style-type: none"> ■ Single bit (0) ■ One bit per byte (1) <p>Default Value: Single bit</p> <p>Enabled: UMCTL2_OCPAR_OR_OCECC_EN_1==1</p> <p>Parameter Name: UMCTL2_OCPAR_ADDR_PARITY_WIDTH</p>
Enable On-Chip ECC	<p>This parameter enables the On-Chip ECC feature.</p> <p>When enabled, the controller instantiates necessary logic to enable on-chip ECC protection.</p> <p>This feature is available only in designs where the Arbiter is used (UMCTL2_INCL_ARB=1); there are no upsized or downsized ports, Inline-ECC (MEMC_INLINE_ECC=1) and external WDATA RAM (UMCTL2_WDATA_EXTRAM=1) are used.</p> <p>This feature is not supported if the following conditions are true:</p> <ul style="list-style-type: none"> ■ OCPAR is enabled. ■ Dual Channel is enabled. <p>This feature is under access control. For more information, contact Synopsys.</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: ((UMCTL2_INCL_ARB==1) && (THEREIS_PORT_DSIZE==0) && (THEREIS_PORT_USIZE==0) && (UMCTL2_DUAL_CHANNEL==0) && (DDRCTL_PRODUCT_NAME==2) && (UMCTL2_OCPAR_EN==0) && (MEMC_INLINE_ECC==1) && (UMCTL2_WDATA_EXTRAM==1))</p> <p>Parameter Name: UMCTL2_OCECC_EN</p>
Enable Register Parity	<p>This parameter enables the Register Parity feature.</p> <p>When enabled, the controller instantiates the necessary logic to enable register parity protection.</p> <p>This feature is under access control. For more information, contact Synopsys.</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: DDRCTL_LPDDR==1 && DDRCTL_PRODUCT_NAME==2</p> <p>Parameter Name: UMCTL2_REGPAR_EN</p>

Label	Description
Register Parity Type	<p>This parameter is the Register Parity type:</p> <ul style="list-style-type: none"> ■ 0: 1 bit parity, calculated for all 32 bits ■ 1: 4 bit parity, one parity bit for each byte <p>Values:</p> <ul style="list-style-type: none"> ■ 1 bit parity, calculated for all 32 bits (0) ■ 4 bit parity, one parity bit for each byte (1) <p>Default Value: 1 bit parity, calculated for all 32 bits</p> <p>Enabled: UMCTL2_REGPAR_EN==1</p> <p>Parameter Name: UMCTL2_REGPAR_TYPE</p>
Enable On-Chip Command/Address Protection	<p>This parameter enables the On-Chip Command/Address Path Protection feature. When enabled, the controller instantiates necessary logic to enable on-chip command and address protection.</p> <p>This feature is under access control. For more information, contact Synopsys.</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: (((UMCTL2_INCL_ARB==1 && UMCTL2_DUAL_CHANNEL==0) UMCTL2_OCCAP_DDRC_INTERNAL_TESTING==1) && DDRCTL_PRODUCT_NAME==2)</p> <p>Parameter Name: UMCTL2_OCCAP_EN</p>
Enable Pipelining for checkers of On-Chip Command/Address Protection feature	<p>Enable Pipelining for checkers of On-Chip Command/Address Protection feature</p> <p>Also includes an additional pipeline for parity checker in rd_ie_rdata_ctl module when OCPAR/OCECC is enabled with Inline ECC.</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: UMCTL2_OCCAP_EN==1 && DDRCTL_INCL_CHB==0</p> <p>Parameter Name: UMCTL2_OCCAP_PIPELINE</p>
Enable On-Chip SRAM Address Protection	<p>This parameter enables the On-Chip SRAM Address Protection .</p> <p>When enabled, the controller instantiates necessary logic to enable external On-Chip external SRAM Address Protection.</p> <p>This feature is reserved only for the automotive product with either support for OCECC (UMCTL2_OCECC_EN=1) or OCPAR (UMCTL2_OCPAR_EN=1).</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: ((DDRCTL_PRODUCT_NAME==2) && ((UMCTL2_OCPAR_EN==1) (UMCTL2_OCECC_EN==1)) && (MEMC_INLINE_ECC==1))</p> <p>Parameter Name: DDRCTL_OCSAP_EN</p>

Label	Description
Link Reliability	
Enable Link ECC	<p>This parameter enables the support for the link ECC feature.</p> <p>Values: 0, 1 Default Value: 0 Enabled: ((DDRCTL_LPDDR==1) && (MEMC_FREQ_RATIO==4)) && (DDRCTL_PRODUCT_NAME==2 DDRCTL_PRODUCT_NAME==4) Parameter Name: MEMC_LINK_ECC</p>

14.8 HW Configuration / RTL Assertions Parameters

Table 14-8 HW Configuration / RTL Assertions Parameters

Label	Description
RTL Assertions Choice	
Enable All RTL SystemVerilog Assertions	<p>This parameter enables all user executable RTL SystemVerilog assertions. This parameter is enabled by default; it is recommended to keep it enabled, especially in your testbenches. These assertions are helpful to identify unexpected input stimulus, wrong register values and bad programming sequences that can commonly occur in your environments.</p> <p>You can disable this parameter, if the RTL fails when running gate-level simulations or when using unsupported simulators.</p> <p>Values: 0, 1</p> <p>Default Value: 1</p> <p>Enabled: Always</p> <p>Parameter Name: UMCTL2_RTL_ASSERTIONS_ALL_EN</p>

15

Signal Descriptions

This chapter details all possible I/O signals in the IP. For configurable IP titles, your actual configuration might not contain all of these signals.

Inputs are on the left of the signal diagrams; outputs are on the right.

Attention: For configurable IP titles, do not use this document to determine the exact I/O footprint of the controller. It is for reference purposes only.

When you configure the controller in coreConsultant, you must access the I/O signals for your actual configuration at workspace/report/IO.html or workspace/report/IO.xml after you have completed the report creation activity. That report comes from the exact same source as this chapter but removes all the I/O signals that are not in your actual configuration. This does not apply to non-configurable IP titles. In addition, all parameter expressions are evaluated to actual values. Therefore, the widths might change depending on your actual configuration.

Some expressions might refer to TCL functions or procedures (sometimes identified as **<functionof>**) that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the controller in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

In addition to describing the function of each signal, the signal descriptions in this chapter include the following information:

- **Active State:** Indicates whether the signal is active high or active low. When a signal is not intended to be used in a particular application, then this signal needs to be tied or driven to the inactive state (opposite of the active state).
- **Registered:** Indicates whether or not the signal is registered directly inside the IP boundary without intervening logic (excluding simple buffers). A value of *No* does not imply that the signal is not synchronous, only that there is some combinatorial logic between the signal's origin or destination register and the boundary of the controller. A value of *N/A* indicates that this information is not provided for this IP title.
- **Synchronous to:** Indicates which clocks in the IP sample this input (drive for an output). This clock might not be the same as the clock that your application logic should use to clock (sample/drive) this pin. For more details, consult the clock section in the databook.
- **Exists:** Name of configuration parameter that populates this signal in your configuration.

The I/O signals are grouped as follows:

- “Clocks and Resets Signals” on page 376
- “AXI Port n Global Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)” on page 378
- “AXI Port n Write Address Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals” on page 379
- “AXI Port n Write Address On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)” on page 384
- “AXI Port n Write Data Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals” on page 385
- “AXI Port n Write Data On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)” on page 387
- “AXI Port n Write Response Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals” on page 388
- “AXI Port n Read Address Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals” on page 389
- “AXI Port n Read Address On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)” on page 395
- “AXI Port n Read Data Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals” on page 396
- “AXI Port n Read Data On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)” on page 398
- “Read Reorder Buffer Data RAM Interface (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals” on page 399
- “Write Data Parity Signals” on page 402
- “Read Data Parity Signals” on page 403
- “Write Address Parity Signals” on page 404
- “Read Address Parity Signals” on page 405
- “On-Chip Command/ Address Path Protection Signals” on page 406
- “On-Chip ECC Signals” on page 409
- “HIF Read Command Interface Signals” on page 410
- “HIF Write Command Interface Signals” on page 416
- “HIF Command Interface Signals” on page 422
- “HIF Write Data Interface Signals” on page 436
- “HIF Read Data Interface Signals” on page 439
- “Write Data RAM Interface Signals” on page 443
- “Mode Register Read/Write Signals” on page 448
- “DDRC Hardware Low Power Signals” on page 449
- “DDRC Self Refresh Signals” on page 453
- “Inline ECC Debug Signals” on page 454
- “Performance Logging Signals” on page 457
- “Credit Counters Signals” on page 476
- “Port Arbiter Signals” on page 478
- “ECC Scrubber Signals” on page 479
- “DFI Command Interface Signals” on page 480
- “DFI Write Data Interface (for n = 0; n <= 1)(for p = 0; p <= 3) Signals” on page 482

- [“DFI Read Data Interface \(for n = 0; n <= 1\)\(for p = 0; p <= 3\) Signals” on page 484](#)
- [“DFI Update Interface \(for n = 0; n <= 1\) Signals” on page 486](#)
- [“DFI Status Interface \(for n = 0; n <= 1\) Signals” on page 488](#)
- [“DFI PHY Master Interface \(for n = 0; n <= 1\) Signals” on page 490](#)
- [“DFI Low Power Interface \(for n = 0; n <= 1\) Signals” on page 492](#)
- [“DFI MC to PHY Message Interface \(for n = 0; n <= 1\) Signals” on page 495](#)
- [“DFI WCK Control Interface \(for n = 0; n <= 1\)\(for p = 0; p <= 3\) Signals” on page 496](#)
- [“Non-DFI DDRCTL PHY Sideband Interface \(for n = 0; n <= 1\)\(for p = 0; p <= 3\) Signals” on page 497](#)
- [“LPDDR4 Initialization Handshake Interface Signals” on page 498](#)
- [“Register Visibility Control Signals” on page 499](#)
- [“Interrupts Signals” on page 500](#)
- [“APB Device Interface Signals” on page 507](#)
- [“APB4 Device Interface Signals” on page 510](#)
- [“Per-bank refresh Bank number Signals” on page 511](#)
- [“Register Parity Protection Signals” on page 512](#)

15.1 Clocks and Resets Signals

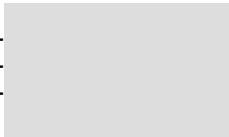


Table 15-1 Clocks and Resets Signals

Port Name	I/O	Description
core_ddrc_core_clk	I	<p>DDRC clock. The DDRC logic, including the DFI interface, runs on this clock.</p> <ul style="list-style-type: none"> For 1:2 frequency ratio mode, this is at half the frequency as the SDRAM clock. For 1:4 frequency ratio mode, this is at quarter the frequency of the SDRAM clock. <p>Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
core_ddrc_rstn	I	<p>Active low reset signal. The controller must be taken out of reset only after all registers have been programmed. Synchronous to core_ddrc_core_clk on de-assertion, asynchronous on assertion.</p> <ul style="list-style-type: none"> 0 - Resets the controller 1 - Takes the controller out of reset <p>Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: N/A Power Domain: DDRCTL_DOMAIN</p>
core_ddrc_core_clk_dch1	I	<p>DDRC DCH1 clock. The DDRC Channel 1 logic runs on this clock. Must be connected to the same clock source (synchronous to) as core_ddrc_core_clk.</p> <p>Exists: DDRCTL_DDR_DUAL_CHANNEL Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
core_ddrc_rstn_dch1	I	<p>Active low reset signal for DDRC Channel 1. DDRCTL DDRC Channel 1 must be taken out of reset only after all registers have been programmed.</p> <p>Synchronous to core_ddrc_core_clk_dch1 on de-assertion, asynchronous on assertion.</p> <ul style="list-style-type: none"> 0 - Resets the DDRCTL DDRC Channel 1 1 - Takes the DDRCTL DDRC Channel 1 out of reset. <p>Exists: DDRCTL_DDR_DUAL_CHANNEL</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: N/A</p> <p>Power Domain: DDRCTL_DOMAIN</p>
chctl_reg_ddrc_dual_channel_en	O	<p>Indicates whether shared-AC dual channel mode is enabled. Equivalent to CHCTL.dual_channel_en register. This signal can be used to gate core_ddrc_core_clk_dch1 clock.</p> <p>Exists: DDRCTL_DDR_DUAL_CHANNEL_OR_SINGLE_INST_DUALCH</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
bsm_clk[(MEMC_NUM_RANKS-1):0]	I	<p>LPDDR gated BSM clock This clock is in the same clock domain with core_ddrc_core_clk. This clock bsm_clk[i] can be gated only when bsm_clk_en[i] is 0.</p> <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
bsm_clk_en[(MEMC_NUM_RANKS-1):0]	O	<p>LPDDR BSM clock enable</p> <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.2 AXI Port n Global Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)



```

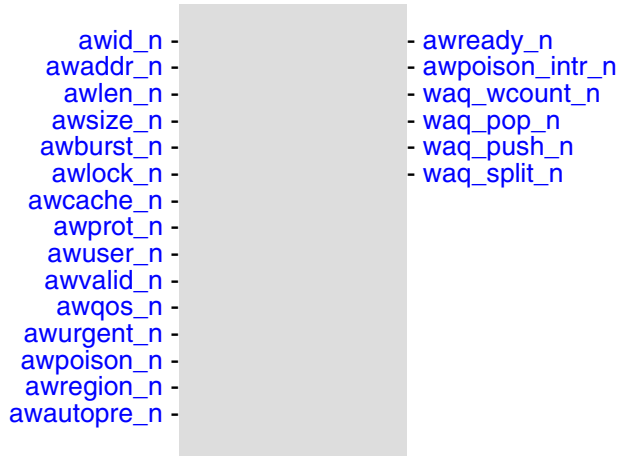
aresetn_n -
aclk_n -
csysreq_n -
csysack_n -
cactive_n -

```

Table 15-2 AXI Port n Global Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)

Port Name	I/O	Description
aresetn_n	I	AXI Asynchronous Reset. Active-low pin that asynchronously resets the AXI logic to its default state. Synchronous to aclk_n on de-assertion, asynchronous on assertion. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: N/A Power Domain: DDRCTL_DOMAIN
aclk_n	I	AXI Input Clock. All signals in the AXI host port n logic are sampled on the rising edge of this clock. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
csysreq_n	I	AXI Low-Power Request. Request from the system clock controller for the peripheral (Port n) to enter a low-power state. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
csysack_n	O	AXI Low-Power Request Acknowledge. Acknowledgement from the peripheral (Port n) of a system low-power request. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
cactive_n	O	AXI Clock Active. Indicates that the peripheral (Port n) requires its clock signal. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.3 AXI Port n Write Address Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals



awid_n -
 awaddr_n -
 awlen_n -
 awsize_n -
 awburst_n -
 awlock_n -
 awcache_n -
 awprot_n -
 awuser_n -
 awvalid_n -
 awqos_n -
 awurgent_n -
 awpoison_n -
 awregion_n -
 awautopre_n -

- awready_n
 - awpoison_intr_n
 - waq_wcount_n
 - waq_pop_n
 - waq_push_n
 - waq_split_n

Table 15-3 AXI Port n Write Address Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals

Port Name	I/O	Description
awid_n[(UMCTL2_A_IDW-1):0]	I	<p>AXI Write Address ID. Identification tag for the write address group of signals.</p> <p>Exists: UMCTL2_A_AXI_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
awaddr_n[(UMCTL2_A_ADDRW-1):0]	I	<p>AXI Write Address. The address of the first transfer in a write burst transaction.</p> <p>The associated control signals are used to determine the addresses of the remaining transfers in a burst.</p> <p>Exists: UMCTL2_A_AXI_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
awlen_n[(UMCTL2_A_LENW-1):0]	I	<p>AXI Write Burst Length. The number of transfers in a burst associated with the write address.</p> <p>Exists: UMCTL2_A_AXI_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
awsiz _n [2:0]	I	AXI Write Burst Size. The size of each transfer in a burst. Byte lane strobes indicate exactly which byte lanes to update. Exists: UMCTL2_A_AXI _n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
awburst _n [1:0]	I	AXI Write Burst Type. Coupled with the size, burst type details how the address for each transfer within a burst is calculated. Exists: UMCTL2_A_AXI _n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
awlock _n [UMCTL2_AXI_LOCK_WIDTH _n -1:0]	I	AXI Write Lock Type. Provides additional information about the atomic characteristics of the transfer. Exists: UMCTL2_A_AXI _n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
awcache _n [3:0]	I	AXI Write Cache Type. Indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction. This signal is not used by the controller. Exists: UMCTL2_A_AXI _n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
awprot _n [2:0]	I	AXI Write Protection Type. Indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. This signal is not used by the controller. Exists: UMCTL2_A_AXI _n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
awuser _n [(AXI_USERW-1):0]	I	AXI Write Address User. Travels with the write transaction and back to buser with the write response. Exists: UMCTL2_A_AXI _n && UMCTL2_AXI_USER_WIDTH > 0 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
awvalid_n	I	<p>AXI Write Address Valid. Indicates that valid write address and control information are available.</p> <p>Exists: UMCTL2_A_AXI_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
awready_n	O	<p>AXI Write Address Ready. Indicates that the subordinate is ready to accept an address and associated control signals.</p> <p>Exists: UMCTL2_A_AXI_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
awqos_n[3:0]	I	<p>AXI Write Quality of Service. Sideband signal to indicate the quality of service attributes of the write transaction.</p> <p>The awqos_n signalling is sticky, that is, it must remain stable when awvalid_n is asserted and awready_n is de-asserted.</p> <p>If enabled by UMCTL2_EXT_PORTPRIO hardware parameter, this signal determines the transaction priority for port arbitration. Higher values signify higher priority.</p> <p>For single-port configurations, this signal has no effect.</p> <p>Exists: UMCTL2_A_AXI_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
awurgent_n	I	<p>AXI Write Urgent. Sideband signal to indicate a write urgent transaction. When asserted, if wr_port_urgent_en register is set, causes the port arbiter to switch immediately to write. It can be asserted anytime, and is not associated to any particular command.</p> <p>Exists: UMCTL2_A_AXI_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
awpoison_n	I	<p>AXI Write poison. Off-band signal to indicate an invalid write transaction. When asserted, no data is written to the memory. If not needed, signal must be tied to zero.</p> <p>Exists: UMCTL2_A_AXI_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
awpoison_intr_n	O	Write transaction poisoning interrupt. Cleared by register wr_poison_intr_clr. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
awregion_n[3:0]	I	AXI 4 Write Address REGION signal. This signal is not used by the controller. Exists: UMCTL2_A_AXI4_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
waq_wcount_n[XPI_WAQD_LG2_n-1:0]	O	Number of used positions in the Write address FIFO (synchronous to core_ddrc_core_clk). Width XPI_WAQD_LG2_n = RoundUp(log2(UMCTL2_AXI_WAQD_n)). Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
waq_pop_n	O	Transaction read from the Write address FIFO (synchronous to core_ddrc_core_clk). Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
waq_push_n	O	Transaction written to the Write address FIFO (synchronous to aclk_n). Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
waq_split_n	O	First portion of a wrap burst going to the Write address FIFO (synchronous to aclk_n). Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
awautopre_n	I	<p>AXI auto-precharge signal for write command.</p> <p>This port is for write address channel signal.</p> <p>hif_cmd_autopre(hif_wcmd_autopre) is asserted when this signal is high.</p> <p>This signal is valid when awvalid_n is high. This port is available with DDR4 only. In other protocol, this should be kept low.</p> <p>Exists: UMCTL2_A_AXI_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.4

AXI Port n Write Address On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)



Table 15-4 AXI Port n Write Address On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)

Port Name	I/O	Description
awparity_n[(OCPAR_ADDR_PARITY_WIDTH-1):0]	I	AXI Write address parity. Exists: UMCTL2_A_AXI_n && UMCTL2_OCPAR_OR_OCECC_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.5 AXI Port n Write Data Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals

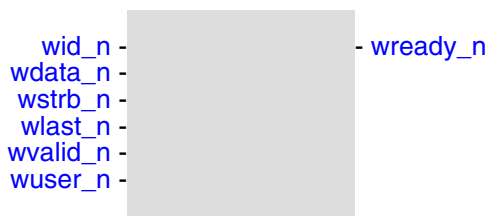


Table 15-5 AXI Port n Write Data Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals

Port Name	I/O	Description
<code>wid_n[(UMCTL2_A_IDW-1):0]</code>	I	AXI Write ID. ID tag of the write data transfer. Must match the <code>awid</code> value of the write transaction. This signal is not used by the controller. Exists: IUMCTL2_A_AXI4_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
<code>wdata_n[UMCTL2_PORT_DW_n-1:0]</code>	I	AXI Write Data. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
<code>wstrb_n[UMCTL2_PORT_NBYTES_n-1:0]</code>	I	AXI Write Strobe. Indicates which byte lanes to update in memory. There is one data strobe for each eight bits of the write bus, that is, <code>wstrb[i]</code> corresponds to <code>wdata[8*i+7:8*i]</code> . Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
<code>wlast_n</code>	I	AXI Write Last. Indicates the last transfer in a write burst. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
<code>wvalid_n</code>	I	AXI Write Valid. Indicates that valid write data and strobes are available. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
wready_n	O	AXI Write Ready. Indicates that the subordinate can accept the write data. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
wuser_n[(AXI_USERW-1):0]	I	AXI Write Data User. This signal is not used by the controller. Exists: UMCTL2_A_AXI_n && UMCTL2_AXI_USER_WIDTH > 0 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.6 AXI Port n Write Data On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_N-PORTS-1)



Table 15-6 AXI Port n Write Data On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)

Port Name	I/O	Description
wparity_n[UMCTL2_PORT_NBYTES_n-1:0]	I	AXI Write Parity/ECC. Exists: UMCTL2_A_AXI_n && UMCTL2_OCPAR_OR_OCECC_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.7 AXI Port n Write Response Channel (for n = 0; n <= UMCTL2_A_N-PORTS-1) Signals

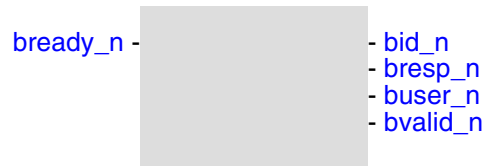


Table 15-7 AXI Port n Write Response Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals

Port Name	I/O	Description
<code>bid_n[(UMCTL2_A_IDW-1):0]</code>	O	AXI Write Response ID. Must match the <code>awid</code> value of the write transaction to which the subordinate is responding. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
<code>bresp_n[(AXI_RESPW-1):0]</code>	O	AXI Write Response. Indicates the status of the write transaction. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
<code>buser_n[(AXI_USERW-1):0]</code>	O	AXI Write Response User. Mirror of <code>awuser</code> sent with write transaction. Exists: UMCTL2_A_AXI_n && UMCTL2_AXI_USER_WIDTH > 0 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
<code>bvalid_n</code>	O	AXI Write Response Valid. Indicates that a valid write response is available. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
<code>bready_n</code>	I	AXI Write Response Ready. Indicates that the master can accept the write response information. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.8 AXI Port n Read Address Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals

arid_n	-	arready_n
araddr_n	-	arpoison_intr_n
arlen_n	-	raqb_wcount_n
arsize_n	-	raqr_wcount_n
arburst_n	-	raqb_pop_n
arlock_n	-	raqb_push_n
arcache_n	-	raqr_pop_n
arprot_n	-	raqr_push_n
aruser_n	-	raq_wcount_n
arvalid_n	-	raq_pop_n
arqos_n	-	raq_push_n
arpoison_n	-	raq_split_n
arregion_n		
arurgentb_n		
arurgentn_n		
arurgent_n		
arautopre_n		

Table 15-8 AXI Port n Read Address Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals

Port Name	I/O	Description
arid_n[(UMCTL2_A_IDW-1):0]	I	AXI Read Address ID. Identification tag for the read address group of signals. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
araddr_n[(UMCTL2_A_ADDRW-1):0]	I	AXI Read Address. The address of the first transfer in a read burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in a burst. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
arlen_n[(UMCTL2_A_LENW-1):0]	I	AXI Read Burst Length. The number of transfers in a burst associated with the read address. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
arsize_n[(AXI_SIZEW-1):0]	I	AXI Read Burst Size. The size of each transfer in a burst. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
arburst_n[(AXI_BURSTW-1):0]	I	AXI Read Burst Type. Coupled with the size, burst type details how the address for each transfer within a burst is calculated. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
arlock_n[UMCTL2_AXI_LOCK_WIDTH_n-1:0]	I	AXI Read Lock Type. Provides additional information about the atomic characteristics of the transfer. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
arcache_n[(AXI_CACHEW-1):0]	I	AXI Read Cache Type. Provides additional information about the cacheable characteristics of the transfer. This signal is not used by the controller. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
arprot_n[(AXI_PROTW-1):0]	I	AXI Read Protection Type. Provides protection unit information for the transaction. This signal is not used by the controller. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
aruser_n[(AXI_USERW-1):0]	I	AXI Read Address User. Travels with the read transaction and back to ruser with the read response. Exists: UMCTL2_A_AXI_n && UMCTL2_AXI_USER_WIDTH > 0 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
arvalid_n	I	<p>AXI Read Address Valid. Indicates that valid read address and control information are available.</p> <p>Exists: UMCTL2_A_AXI_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
arready_n	O	<p>AXI Read Address Ready. Indicates that the subordinate is ready to accept an address and associated control signals.</p> <p>Exists: UMCTL2_A_AXI_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
arqos_n[(AXI_QOSW-1):0]	I	<p>AXI Read Quality of Service. Sideband signal to indicate the quality of service attributes of the read transaction.</p> <p>The arqos_n signalling is sticky, that is, it must remain stable when arvalid_n is asserted and arready_n is de-asserted.</p> <p>If enabled by UMCTL2_EXT_PORTPRIO hardware parameter, this signal determines the transaction priority for port arbitration. Higher values signify higher priority.</p> <p>This signal determines also the priority of the read transfer going into the CAM depending on the programming of the PCFGQOS0_n register.</p> <p>Exists: UMCTL2_A_AXI_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
arpoison_n	I	<p>AXI Read poison. Sideband signal to indicate an invalid read transaction.</p> <p>When asserted, all zeros are returned at the output. If not needed, signal must be tied to zero.</p> <p>Exists: UMCTL2_A_AXI_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
arpoison_intr_n	O	<p>Read transaction poisoning interrupt. Cleared by register rd_poison_intr_clr.</p> <p>Exists: UMCTL2_A_AXI_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
arregion_n[(UMCTL2_AXI_REGION_WIDTH-1):0]	I	<p>AXI 4 Read Address REGION signal. This signal is not used by the controller.</p> <p>Exists: UMCTL2_A_AXI4_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
arurgentb_n	I	<p>AXI Read Urgent Blue Queue. Off-band signal to indicate a blue queue urgent transaction.</p> <p>When asserted, if rd_port_urgent_en register is set, causes the port arbiter to switch immediately to read.</p> <p>It can be asserted anytime, it is not associated to any particular command.</p> <p>Exists: UMCTL2_XPI_USE2RAQ_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
arurgentn_n	I	<p>AXI Read Urgent Red Queue. Off-band signal to indicate a red queue urgent transaction.</p> <p>When asserted, if rd_port_urgent_en register is set, causes the port arbiter to switch immediately to read.</p> <p>It can be asserted anytime, it is not associated to any particular command.</p> <p>Exists: UMCTL2_XPI_USE2RAQ_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
raqb_wcount_n[XPI_RAQD_LG2_n-1:0]	O	<p>Number of used positions in the Blue Read address FIFO (synchronous to core_ddrc_core_clk).</p> <p>Width XPI_RAQD_LG2_n = RoundUp(log2(UMCTL2_AXI_RAQD_n)).</p> <p>Exists: UMCTL2_XPI_USE2RAQ_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
raqr_wcount_n[XPI_RAQD_LG2_n-1:0]	O	<p>Number of used positions in the Red Read address FIFO (synchronous to core_ddrc_core_clk).</p> <p>Width XPI_RAQD_LG2_n = RoundUp(log2(UMCTL2_AXI_RAQD_n)).</p> <p>Exists: UMCTL2_XPI_USE2RAQ_n</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
raqb_pop_n	O	Transaction read from the Blue Read address FIFO (synchronous to core_ddrc_core_clk). Exists: UMCTL2_XPI_USE2RAQ_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
raqb_push_n	O	Transaction written to the Blue Read address FIFO (synchronous to aclk_n). Exists: UMCTL2_XPI_USE2RAQ_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
raqr_pop_n	O	Transaction read from the Red Read address FIFO (synchronous to core_ddrc_core_clk). Exists: UMCTL2_XPI_USE2RAQ_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
raqr_push_n	O	Transaction written to the Red Read address FIFO (synchronous to aclk_n). Exists: UMCTL2_XPI_USE2RAQ_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
arurgent_n	I	AXI Read Urgent. Off-band signal to indicate a read urgent transaction. When asserted, if rd_port_urgent_en register is set, causes the port arbiter to switch immediately to read. It can be asserted anytime, it is not associated to any particular command. Exists: UMCTL2_A_AXI_n && UMCTL2_XPI_USE2RAQ_n == 0 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
raq_wcount_n[XPI_RAQD_LG2_n-1:0]	O	Number of used positions in the Read address FIFO (synchronous to core_ddrc_core_clk). XPI_RAQD_LG2_n = RoundUp(log2('UMCTL2_AXI_RAQD_n)). Exists: UMCTL2_A_AXI_n && UMCTL2_XPI_USE2RAQ_n == 0 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
raq_pop_n	O	Transaction read from the Read address FIFO (synchronous to core_ddrc_core_clk). Exists: UMCTL2_A_AXI_n && UMCTL2_XPI_USE2RAQ_n == 0 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
raq_push_n	O	Transaction written to the Read address FIFO (synchronous to aclk_n). Exists: UMCTL2_A_AXI_n && UMCTL2_XPI_USE2RAQ_n == 0 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
raq_split_n	O	First portion of a wrap burst going to the Read address FIFO (synchronous to aclk_n). Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
arautopre_n	I	AXI auto-precharge signal for read command. This port is for read address channel signal. The hif_cmd_autopre(hif_rcmd_autopre) is asserted when this signal is high. This signal is valid when arvalid_n is high. This port is available with DDR4 only. In other protocol, this should be kept low. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.9 AXI Port n Read Address On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)



Table 15-9 AXI Port n Read Address On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)

Port Name	I/O	Description
arparity_n[(OCPAR_ADDR_PARITY_WIDTH-1):0]	I	AXI Read address parity. Exists: UMCTL2_A_AXI_n && UMCTL2_OCPAR_OR_OCECC_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.10 AXI Port n Read Data Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals



Table 15-10 AXI Port n Read Data Channel (for n = 0; n <= UMCTL2_A_NPORTS-1) Signals

Port Name	I/O	Description
<code>rid_n[(UMCTL2_A_IDW-1):0]</code>	O	AXI Read ID. Must match the arid value of the read transaction to which the subordinate is responding. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
<code>rdata_n[UMCTL2_PORT_DW_n-1:0]</code>	O	AXI Read Data. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
<code>rresp_n[(AXI_RESPW-1):0]</code>	O	AXI Read Response. Indicates the status of the read transfer. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
<code>ruser_n[(AXI_USERW-1):0]</code>	O	AXI Read Response User. Mirror of aruser sent with read transaction. Exists: UMCTL2_A_AXI_n && UMCTL2_AXI_USER_WIDTH > 0 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
<code>rlast_n</code>	O	AXI Read Last. Indicates the last transfer in a read burst. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
rvalid_n	O	AXI Read Valid. Indicates that the required read data is available and the read transfer can complete. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
rready_n	I	AXI Read Ready. Indicates that the master can accept the read data and response information. Exists: UMCTL2_A_AXI_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

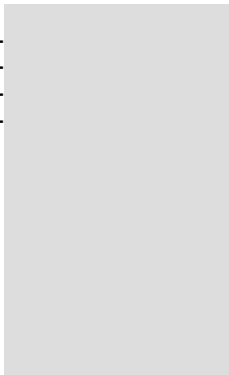
15.11 AXI Port n Read Data On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_N-PORTS-1)



Table 15-11 AXI Port n Read Data On-Chip Parity Signals (for n = 0; n <= UMCTL2_A_NPORTS-1)

Port Name	I/O	Description
rparity_n[UMCTL2_PORT_NBYTES_n-1:0]	O	AXI Read parity/ECC. Exists: UMCTL2_A_AXI_n && UMCTL2_OCPAR_OR_OCECC_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.12 Read Reorder Buffer Data RAM Interface (for $n = 0; n \leq \text{UMCTL2_A_NPORTS-1}$) Signals



rdataram_dout_n -
 rdataram_dout_par_n -
 rdataram_dout_dch1_n -
 rdataram_dout_par_dch1_n -

- rdataram_din_n
 - rdataram_wr_n
 - rdataram_re_n
 - rdataram_raddr_n
 - rdataram_waddr_n
 - rdataram_din_par_n
 - rdataram_din_dch1_n
 - rdataram_wr_dch1_n
 - rdataram_re_dch1_n
 - rdataram_raddr_dch1_n
 - rdataram_waddr_dch1_n
 - rdataram_din_par_dch1_n

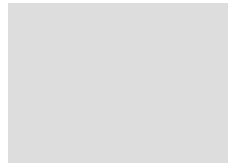
Table 15-12 Read Reorder Buffer Data RAM Interface (for $n = 0; n \leq \text{UMCTL2_A_NPORTS-1}$) Signals

Port Name	I/O	Description
rdataram_dout_n[(UMCTL2_RDATARAM_DW-1):0]	I	Read data coming from the Read Reorder Buffer Data RAM. Valid data comes out one clock after rdataram_re = 1. Exists: UMCTL2_RRB_EXTRAM_ENABLED_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
rdataram_din_n[(UMCTL2_RDATARAM_DW-1):0]	O	Data for the write operation. Valid with rdataram_wr = 1. Exists: UMCTL2_RRB_EXTRAM_ENABLED_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
rdataram_wr_n	O	Write signal to the RRB data RAM. Exists: UMCTL2_RRB_EXTRAM_ENABLED_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
rdataram_re_n	O	Read signal to the RRB data RAM. Exists: UMCTL2_RRB_EXTRAM_ENABLED_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
rdataram_raddr_n[(UMCTL2_RDATARAM_AW-1):0]	O	Read address to the RRB data RAM. Valid with rdataram_re = 1. Exists: UMCTL2_RRB_EXTRAM_ENABLED_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
rdataram_waddr_n[(UMCTL2_RDATARAM_AW-1):0]	O	Write address for the write operation. Valid with rdataram_wr = 1. Exists: UMCTL2_RRB_EXTRAM_ENABLED_n Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
rdataram_dout_par_n[(UMCTL2_DATARAM_PAR_DW-1):0]	I	Read data parity/ECC for rdataram_dout. Exists: UMCTL2_RRB_EXTRAM_ENABLED_n && UMCTL2_OCPAR_OR_OCECC_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
rdataram_din_par_n[(UMCTL2_DATARAM_PAR_DW-1):0]	O	Read data parity/ECC for rdataram_din. Exists: UMCTL2_RRB_EXTRAM_ENABLED_n && UMCTL2_OCPAR_OR_OCECC_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
rdataram_dout_dch1_n[(UMCTL2_RDATARAM_DW-1):0]	I	Read data coming from the Read Reorder Buffer Data RAM (channel 1). Valid data comes out one clock after rdataram_re_dch1. Exists: UMCTL2_RRB_EXTRAM_ENABLED_n && UMCTL2_DATA_CHANNEL_INTERLEAVE_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
rdataram_din_dch1_n[(UMCTL2_RDATARAM_DW-1):0]	O	Data for the write operation (channel 1). Valid with rdataram_wr_dch1 = 1. Exists: UMCTL2_RRB_EXTRAM_ENABLED_n && UMCTL2_DATA_CHANNEL_INTERLEAVE_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
rdataram_wr_dch1_n	O	Write signal to the RRB data RAM (channel 1). Exists: UMCTL2_RRB_EXTRAM_ENABLED_n && UMCTL2_DATA_CHANNEL_INTERLEAVE_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
rdataram_re_dch1_n	O	Read signal to the RRB data RAM (channel 1). Exists: UMCTL2_RRB_EXTRAM_ENABLED_n && UMCTL2_DATA_CHANNEL_INTERLEAVE_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
rdataram_raddr_dch1_n[(UMCTL2_RDATARAM_AW-1):0]	O	Read address to the RRB data RAM (channel 1). Valid with rdataram_re_dch1 = 1. Exists: UMCTL2_RRB_EXTRAM_ENABLED_n && UMCTL2_DATA_CHANNEL_INTERLEAVE_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
rdataram_waddr_dch1_n[(UMCTL2_RDATARAM_AW-1):0]	O	Write address for the write operation (channel 1). Valid with rdataram_wr_dch1 = 1. Exists: UMCTL2_RRB_EXTRAM_ENABLED_n && UMCTL2_DATA_CHANNEL_INTERLEAVE_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
rdataram_dout_par_dch1_n[(UMCTL2_DATARAM_PAR_DW-1):0]	I	Read data parity for rdataram_dout_dch1 (channel 1). Exists: UMCTL2_RRB_EXTRAM_ENABLED_n && UMCTL2_DATA_CHANNEL_INTERLEAVE_EN_1 && UMCTL2_OCPAR_OR_OCECC_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
rdataram_din_par_dch1_n[(UMCTL2_DATARAM_PAR_DW-1):0]	O	Read data parity for rdataram_din_dch1 (channel 1). Exists: UMCTL2_RRB_EXTRAM_ENABLED_n && UMCTL2_DATA_CHANNEL_INTERLEAVE_EN_1 && UMCTL2_OCPAR_OR_OCECC_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.13 Write Data Parity Signals

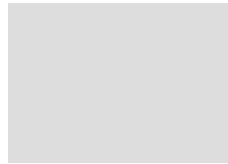


- [par_wdata_err_intr](#)
- [par_wdata_err_intr_fault](#)
- [par_wdata_err_intr_dch1](#)
- [par_wdata_err_intr_fault_dch1](#)

Table 15-13 Write Data Parity Signals

Port Name	I/O	Description
par_wdata_err_intr	O	On-Chip Write data parity error interrupt. Exists: UMCTL2_OCPAR_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
par_wdata_err_intr_fault[1:0]	O	On-Chip Write data parity error fault. This is a version of par_wdata_err_intr which can not be disabled or forced through register. It is a 2-bit antivalent signal with encoding of <ul style="list-style-type: none"> ■ 01 - No Fault ■ 10 - Fault Detected Exists: UMCTL2_OCPAR_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
par_wdata_err_intr_dch1	O	On-Chip Write data parity error interrupt (channel 1). Exists: (UMCTL2_OCPAR_EN_1) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
par_wdata_err_intr_fault_dch1[1:0]	O	On-Chip Write data parity error fault (channel 1). This is a version of par_wdata_err_intr_dch1 which can not be disabled or forced through register. It is a 2-bit antivalent signal with encoding of <ul style="list-style-type: none"> ■ 01 - No Fault ■ 10 - Fault Detected Exists: (UMCTL2_OCPAR_EN_1) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.14 Read Data Parity Signals



- par_rdata_err_intr
- par_rdata_err_intr_fault
- par_rdata_err_intr_dch1
- par_rdata_err_intr_fault_dch1

Table 15-14 Read Data Parity Signals

Port Name	I/O	Description
par_rdata_err_intr	O	On-Chip Read data parity error interrupt. Exists: UMCTL2_OCPAR_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
par_rdata_err_intr_fault[1:0]	O	On-Chip Read data parity error fault. This is a version of par_rdata_err_intr which can not be disabled or forced through register. It is a 2-bit antivalent signal with encoding of <ul style="list-style-type: none"> ■ 01 - No Fault ■ 10 - Fault Detected Exists: UMCTL2_OCPAR_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
par_rdata_err_intr_dch1	O	On-Chip Read data parity error interrupt (channel 1). Exists: (UMCTL2_OCPAR_EN_1) && (UMCTL2_DUAL_CHANNEL) && (MEMC_INLINE_ECC) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
par_rdata_err_intr_fault_dch1[1:0]	O	On-Chip Read data parity error fault (channel 1). This is a version of par_rdata_err_intr_dch1 which can not be disabled or forced through register. It is a 2-bit antivalent signal with encoding of <ul style="list-style-type: none"> ■ 01 - No Fault ■ 10 - Fault Detected Exists: (UMCTL2_OCPAR_EN_1) && (UMCTL2_DUAL_CHANNEL) && (MEMC_INLINE_ECC) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.15 Write Address Parity Signals



- [par_waddr_err_intr](#)
- [par_waddr_err_intr_fault](#)

Table 15-15 Write Address Parity Signals

Port Name	I/O	Description
par_waddr_err_intr	O	AXI Write address parity error interrupt. Exists: UMCTL2_OCPAR_OR_OCECC_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
par_waddr_err_intr_fault[1:0]	O	AXI Write address parity error fault. This is a version of par_waddr_err_intr which can not be disabled or forced through register. It is a 2-bit antivalent signal with encoding of <ul style="list-style-type: none"> ■ 01 - No Fault ■ 10 - Fault Detected Exists: UMCTL2_OCPAR_OR_OCECC_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.16 Read Address Parity Signals

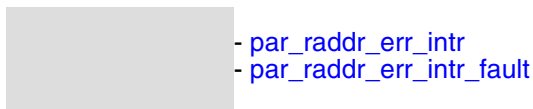


Table 15-16 Read Address Parity Signals

Port Name	I/O	Description
<code>par_raddr_err_intr</code>	O	AXI Read address parity error interrupt. Exists: UMCTL2_OCPAR_OR_OCECC_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
<code>par_raddr_err_intr_fault[1:0]</code>	O	AXI Read address parity error fault. This is a version of <code>par_raddr_err_intr</code> which can not be disabled or forced through register. It is a 2-bit antivalent signal with encoding of <ul style="list-style-type: none"> ■ 01 - No Fault ■ 10 - Fault Detected Exists: UMCTL2_OCPAR_OR_OCECC_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.17 On-Chip Command/Address Path Protection Signals

	- occap_arb_err_intr
	- occap_arb_err_intr_fault
	- occap_ddrc_ctrl_err_intr
	- occap_ddrc_ctrl_err_intr_fault
	- occap_ddrc_ctrl_err_intr_dch1
	- occap_ddrc_ctrl_err_intr_fault_dch1
	- occap_ddrc_data_err_intr
	- occap_ddrc_data_err_intr_fault
	- occap_ddrc_data_err_intr_dch1
	- occap_ddrc_data_err_intr_fault_dch1
	- occap_dfiic_err_intr
	- occap_dfiic_err_intr_fault

Table 15-17 On-Chip Command/Address Path Protection Signals

Port Name	I/O	Description
occap_arb_err_intr	O	On-Chip Command/Address Path Protection Arbiter interrupt. Exists: (UMCTL2_OCCAP_EN_1) && (UMCTL2_INCL_ARB) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
occap_arb_err_intr_fault[1:0]	O	On-Chip Command/Address Path Protection Arbiter fault. This is a version of occap_arb_err_intr which can not be disabled or forced through register. It is a 2-bit antivalent signal with encoding of <ul style="list-style-type: none"> ■ 01 - No Fault ■ 10 - Fault Detected Exists: (UMCTL2_OCCAP_EN_1) && (UMCTL2_INCL_ARB) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
occap_ddrc_ctrl_err_intr	O	On-Chip Command/Address Path Protection DDRC_CTRL interrupt. Exists: UMCTL2_OCCAP_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
occap_ddrc_ctrl_err_intr_fault[1:0]	O	On-Chip Command/Address Path Protection DDRC_CTRL fault. This is a version of occap_ddrc_ctrl_err_intr which can not be disabled or forced through register. It is a 2-bit antivalent signal with encoding of <ul style="list-style-type: none"> ■ 01 - No Fault ■ 10 - Fault Detected Exists: UMCTL2_OCCAP_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
occap_ddrc_ctrl_err_intr_dch1	O	On-Chip Command/Address Path Protection DDRC_CTRL interrupt (channel 1). Exists: (UMCTL2_OCCAP_EN_1) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
occap_ddrc_ctrl_err_intr_fault_dch1[1:0]	O	On-Chip Command/Address Path Protection DDRC_CTRL fault (channel 1). This is a version of occap_ddrc_ctrl_err_intr_dch1 which can not be disabled or forced through register. It is a 2-bit antivalent signal with encoding of <ul style="list-style-type: none"> ■ 01 - No Fault ■ 10 - Fault Detected Exists: (UMCTL2_OCCAP_EN_1) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
occap_ddrc_data_err_intr	O	On-Chip Command/Address Path Protection DDRC Data interrupt. Exists: UMCTL2_OCCAP_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
occap_ddrc_data_err_intr_fault[1:0]	O	On-Chip Command/Address Path Protection DDRC Data fault. This is a version of occap_ddrc_data_err_intr which can not be disabled or forced through register. It is a 2-bit antivalent signal with encoding of <ul style="list-style-type: none"> ■ 01 - No Fault ■ 10 - Fault Detected Exists: UMCTL2_OCCAP_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
occap_ddrc_data_err_intr_dch1	O	On-Chip Command/Address Path Protection DDRC Data interrupt (channel 1). Exists: (UMCTL2_OCCAP_EN_1) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
occap_ddrc_data_err_intr_fault_dch1[1:0]	O	On-Chip Command/Address Path Protection DDRC Data fault (channel 1). This is a version of occap_ddrc_data_err_intr_dch1 which can not be disabled or forced through register. It is a 2-bit antivalent signal with encoding of <ul style="list-style-type: none"> ■ 01 - No Fault ■ 10 - Fault Detected Exists: (UMCTL2_OCCAP_EN_1) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
occap_dfiic_err_intr	O	On-Chip Command/Address Path Protection DFI interconnect interrupt. Exists: UMCTL2_OCCAP_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
occap_dfiic_err_intr_fault[1:0]	O	On-Chip Command/Address Path Protection DFI interconnect fault. This is a version of occap_dfiic_err_intr which can not be disabled or forced through register. It is a 2-bit antivalent signal with encoding of <ul style="list-style-type: none"> ■ 01 - No Fault ■ 10 - Fault Detected Exists: UMCTL2_OCCAP_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.18 On-Chip ECC Signals



- ocecc_uncorrected_err_intr
- ocecc_uncorrected_err_intr_fault

Table 15-18 On-Chip ECC Signals

Port Name	I/O	Description
ocecc_uncorrected_err_intr	O	<p>On-Chip ECC uncorrected error interrupt. Asserted when a 2-bit error or an odd number of bit errors is detected by any of the ECC decoders or when a parity error is detected by any of the parity checkers.</p> <p>Exists: UMCTL2_OCECC_EN_1</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
ocecc_uncorrected_err_intr_fault[1:0]	O	<p>On-Chip ECC uncorrected error fault. This is a version of ocecc_uncorrected_err_intr which can not be disabled or forced via register. It is a 2-bit antivalent signal with encoding of</p> <ul style="list-style-type: none"> ■ 01 - No Fault ■ 10 - Fault Detected <p>Exists: UMCTL2_OCECC_EN_1</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.19 HIF Read Command Interface Signals

hif_rcmd_valid	-	hif_rcmd_stall
hif_rcmd_type	-	hif_rcmd_stall_dch1
hif_rcmd_addr		
hif_rcmd_pri		
hif_rcmd_latency		
hif_rcmd_token		
hif_rcmd_length		
hif_rcmd_wdata_ptr		
hif_rcmd_autopre		
hif_rcmd_valid_dch1		
hif_rcmd_type_dch1		
hif_rcmd_addr_dch1		
hif_rcmd_pri_dch1		
hif_rcmd_latency_dch1		
hif_rcmd_token_dch1		
hif_rcmd_length_dch1		
hif_rcmd_wdata_ptr_dch1		
hif_rcmd_autopre_dch1		

Table 15-19 HIF Read Command Interface Signals

Port Name	I/O	Description
hif_rcmd_valid	I	Valid Read command request to the controller. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rcmd_type[1:0]	I	Valid when hif_rcmd_valid is high. Determines the type of command being issued: <ul style="list-style-type: none"> 00 - Reserved 01 - Read 10 - Reserved 11 - Reserved Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
hif_rcmd_addr[(MEMC_HIF_ADDR_WIDTH_MAX-1):0]	I	<p>Valid when hif_rcmd_valid is high. Word address of the read or write request being made. Word size (in bits) is defined as the configuration parameter MEMC_DRAM_DATA_WIDTH. This bus is MEMC_HIF_ADDR_WIDTH_MAX bits into the DDRCTL, though several of the upper-most bits are generally be unused. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
hif_rcmd_pri[UMCTL2_VPR_EN_VAL:0]	I	<p>Valid when hif_rcmd_valid is high. Determines the priority of command being issued. If UMCTL2_VPRW_EN=0, valid when hif_rcmd_type indicates a read. Don't care for writes. Must be set to 0 for RMWs.</p> <ul style="list-style-type: none"> ■ 0 - Low priority read ■ 1 - High priority read ■ If UMCTL2_VPRW_EN=1 <p>Indicates the priority of a read/write request. hif_rcmd_pri[1] must be set to 0 for RMWs.</p> <ul style="list-style-type: none"> ■ 00 - Low priority read/write ■ 01 - Variable priority read/write ■ 10 - High priority read ■ 11 - Reserved <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
hif_rcmd_latency[(HIF_RQOS_TW-1):0]	I	<p>Valid when hif_rcmd_valid is high and hif_rcmd_pri indicates variable priority read (VPR). Don't care for other priority types. Specifies the timeout value of VPR request. The controller start down counting VPR timer when the request is accepted in the controller. If VPR timeout value is set to 0, the VPR request expires immediately as it enter the controller, thereby making it highest priority transaction class within the device. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1) && (UMCTL2_VPRW_EN) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>

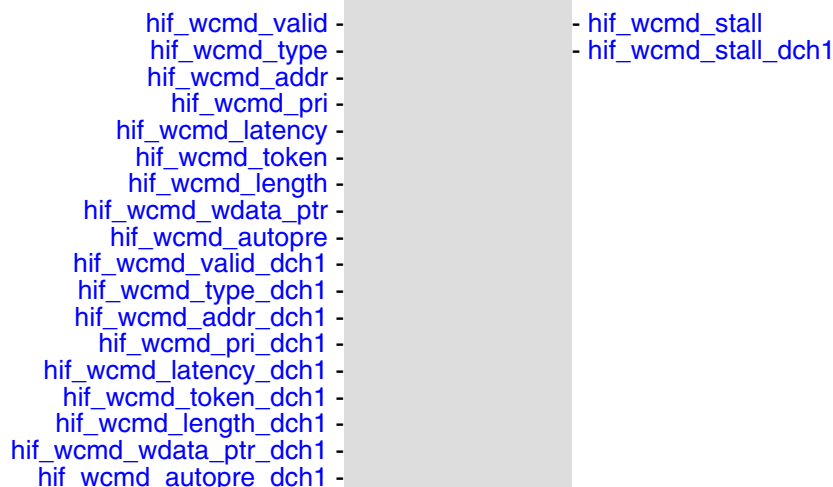
Port Name	I/O	Description
hif_rcmd_token[(MEMC_TAGBITS-1):0]	I	<p>Valid when hif_rcmd_valid is high and hif_rcmd_type indicates a read. Don't care for writes.</p> <p>Token bits are provided with read requests and returned later with read data.</p> <p>For DDRCTL, common uses for these bits include identifying the requester and re-ordering read data which may be returned out-of-order.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_rcmd_length[(UMCTL2_CMD_LEN_BITS-1):0]	I	<p>Valid when hif_rcmd_valid is high and hif_rcmd_type indicates a Read. This signal is ignored for Write or RMW.</p> <p>Indicates the number of requested words</p> <p>If MEMC_BURST_LENGTH = 16 and MEMC_FREQ_RATIO = 4</p> <ul style="list-style-type: none"> ■ 2'b00 - Full RD or WR of 2 HIF data words ■ 2'b10 - Partial (Half) RD or WR of 1 HIF data words ■ 2'b11 - Invalid <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_rcmd_wdata_ptr[(MEMC_WDATA_PTR_BITS-1):0]	I	<p>Don't care for reads.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_rcmd_autopre	I	<p>Valid when hif_rcmd_valid is high. Indicates that the command should be issued to memory with an auto-precharge.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_rcmd_stall	O	<p>HIF Read commands are not accepted starting from the first clock edge after the stall goes high until the clock edge on which stall is detected low.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_rcmd_valid_dch1	I	Valid Read command request to the controller (channel 1). Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rcmd_type_dch1[1:0]	I	Valid when hif_rcmd_valid_dch1 is high. Determines the type of command being issued (channel 1): <ul style="list-style-type: none"> 00 - Reserved 01 - Read 10 - Reserved 11 - Reserved Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rcmd_addr_dch1[(MEMC_HIF_ADDR_WIDTH_MAX-1):0]	I	Valid when hif_rcmd_valid=1. Word address of the read or write request being made (channel 1). Word size (in bits) is defined as the configuration parameter MEMC_DRAM_DATA_WIDTH. This bus is MEMC_HIF_ADDR_WIDTH_MAX bits into the controller, though several of the upper-most bits are generally be unused. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rcmd_pri_dch1[UMCTL2_VPR_EN_VAL:0]	I	For DDRCTL, Valid when hif_rcmd_valid_dch1 is high and hif_rcmd_type_dch1 indicates a read (channel 1). Don't care for writes. Must be set to 0 for RMWs Indicates the priority of a read request. <ul style="list-style-type: none"> 1 - High priority read 0 - Low priority read Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
hif_rcmd_latency_dch1[(HIF_RQOS_TW-1):0]	I	<p>Valid when hif_rcmd_valid_dch1 is high and hif_rcmd_pri_dch1 indicates variable priority read/write (VPR/VPW) (channel 1). Don't care for other priority types.</p> <p>Specifies the timeout value of VPR/VPW request. The controller start down counting VPR/VPW timer when the request is accepted in the controller. If VPR/VPW timeout value is set to 0, the VPR/VPW request expires immediately as it enter the controller, thereby making it highest priority transaction class within the device.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1) && (UMCTL2_VPRW_EN)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_rcmd_token_dch1[(MEMC_TAGBITS-1):0]	I	<p>Valid when hif_rcmd_valid_dch1 and hif_rcmd_type_dch1 indicates a read (channel 1). Don't care for writes.</p> <p>Token bits are provided with read requests and returned later with read data. Common uses for these bits include identifying the requester and re-ordering read data which may be returned out-of-order.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_rcmd_length_dch1[(UMCTL2_CMD_LEN_BITS-1):0]	I	<p>Valid when hif_rcmd_valid_dch1 is high and hif_rcmd_type_dch1 indicates a Read.</p> <p>This signal is ignored for Write or RMW.</p> <p>Indicates the number of requested words</p> <p>If MEMC_BURST_LENGTH = 16 and MEMC_FREQ_RATIO = 4</p> <ul style="list-style-type: none"> ■ 2'b00 - Full RD or WR of 2 HIF data words ■ 2'b10 - Partial (Half) RD or WR of 1 HIF data words ■ 2'b11 - Invalid <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_rcmd_wdata_ptr_dch1[(MEMC_WDATA_PTR_BITS-1):0]	I	<p>Valid when hif_rcmd_valid_dch1 is high and hif_rcmd_type indicates a write. Pointer bits are provided with write requests and returned later with write pointer return corresponding to the write command issued (channel 1).</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_rcmd_autopre_dch1	I	<p>Valid when hif_rcmd_valid_dch1 is high. Indicates that the command should be issued to memory with an auto-precharge (channel 1).</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_rcmd_stall_dch1	O	<p>HIF Read commands are not accepted starting from the first clock edge after the stall goes high until the clock edge on which stall is detected low (channel 1).</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.20 HIF Write Command Interface Signals



hif_wcmd_valid - hif_wcmd_stall
hif_wcmd_type - hif_wcmd_stall_dch1
hif_wcmd_addr
hif_wcmd_pri
hif_wcmd_latency
hif_wcmd_token
hif_wcmd_length
hif_wcmd_wdata_ptr
hif_wcmd_autopre
hif_wcmd_valid_dch1
hif_wcmd_type_dch1
hif_wcmd_addr_dch1
hif_wcmd_pri_dch1
hif_wcmd_latency_dch1
hif_wcmd_token_dch1
hif_wcmd_length_dch1
hif_wcmd_wdata_ptr_dch1
hif_wcmd_autopre_dch1

Table 15-20 HIF Write Command Interface Signals

Port Name	I/O	Description
hif_wcmd_valid	I	Valid Write command request to the controller. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_wcmd_type[1:0]	I	Valid when hif_wcmd_valid is high. Determines the type of command being issued: <ul style="list-style-type: none"> ■ 00 - Write ■ 01 - Reserved ■ 10 - RMW (DDRCTL only) ■ 11 - Reserved Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
hif_wcmd_addr[(MEMC_HIF_ADDR_WIDTH_MAX-1):0]	I	<p>Valid when hif_wcmd_valid is high. Word address of the read or write request being made. Word size (in bits) is defined as the configuration parameter MEMC_DRAM_DATA_WIDTH. This bus is MEMC_HIF_ADDR_WIDTH_MAX bits into the DDRCTL, though several of the upper-most bits are generally be unused. hif_wcmd_addr[3:0] refer to the critical word if MEMC_BURST_LENGTH=16. When critical word is not zero, interleaved/sequential ordering is used for writing data or returning read responses. See JEDEC specification for information on interleaved/sequential data ordering. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
hif_wcmd_pri[UMCTL2_VPR_EN_VAL:0]	I	<p>Valid when hif_wcmd_valid is high. Determines the priority of command being issued. If UMCTL2_VPRW_EN=0, this signal is meaningless. If UMCTL2_VPRW_EN=1, indicates the priority of a write request. hif_wcmd_pri[1] must be set to 0 for RMWs.</p> <ul style="list-style-type: none"> ■ 00 - Normal priority write ■ 01 - Variable priority write ■ 10 - Reserved ■ 11 - Reserved <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
hif_wcmd_latency[(HIF_RQOS_TW-1):0]	I	<p>Valid when hif_wcmd_valid is high and hif_wcmd_pri indicates variable priority write (VPW). Don't care for other priority types. Specifies the timeout value of VPW request. The controller start down counting VPW timer when the request is accepted in the controller. If VPW timeout value is set to 0, the VPW request expires immediately as it enter the controller, thereby making it highest priority transaction class within the device. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1) && (UMCTL2_VPRW_EN) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_wcmd_token[(MEMC_TAGBITS-1):0]	I	<p>Valid when hif_wcmd_valid is high and hif_wcmd_type indicates a read. Don't care for writes.</p> <p>Token bits are provided with read requests and returned later with read data.</p> <p>For DDRCTL, common uses for these bits include identifying the requester and re-ordering read data which may be returned out-of-order.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wcmd_length[(UMCTL2_CMD_LEN_BITS-1):0]	I	<p>Valid when hif_wcmd_valid is high and hif_wcmd_type indicates a Read.</p> <p>This signal is ignored for Write or RMW.</p> <p>Indicates the number of requested words</p> <p>If MEMC_BURST_LENGTH = 16 and MEMC_FREQ_RATIO = 4</p> <ul style="list-style-type: none"> ■ 2'b00 - Full RD or WR of 2 HIF data words ■ 2'b10 - Partial (Half) RD or WR of 1 HIF data words ■ 2'b11 - Invalid <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wcmd_wdata_ptr[(MEMC_WDATA_PTR_BITS-1):0]	I	<p>Valid when hif_wcmd_valid is high and hif_wcmd_type indicates a write. Pointer bits are provided with write requests and returned later with write pointer return corresponding to the write command issued.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wcmd_autopre	I	<p>Valid when hif_wcmd_valid is high. Indicates that the command should be issued to memory with an auto-precharge.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wcmd_stall	O	<p>HIF Write commands are not accepted starting from the first clock edge after the stall goes high until the clock edge on which stall is detected low.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_wcmd_valid_dch1	I	Valid Write command request to the controller (channel 1). Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_wcmd_type_dch1[1:0]	I	Valid when hif_wcmd_valid_dch1 is high. Determines the type of command being issued (channel 1): <ul style="list-style-type: none"> 00 - Write 01 - Reserved 10 - RMW (DDRCTL only) 11 - Reserved Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_wcmd_addr_dch1[(MEMC_HIF_ADDR_WIDTH_MAX-1):0]	I	Valid when hif_wcmd_valid=1. Word address of the read or write request being made (channel 1). Word size (in bits) is defined as the configuration parameter MEMC_DRAM_DATA_WIDTH. This bus is MEMC_HIF_ADDR_WIDTH_MAX bits into the controller, though several of the upper-most bits are generally be unused. hif_wcmd_addr[3:0] refer to the critical word if MEMC_BURST_LENGTH=16. When critical word is not zero, interleaved/sequential ordering is used for writing data or returning read responses. See JEDEC specification for information on interleaved/sequential data ordering. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_wcmd_pri_dch1[UMCTL2_VPR_EN_V AL:0]	I	For DDRCTL, Valid when hif_wcmd_valid_dch1 is high and hif_wcmd_type_dch1 indicates a read (channel 1). Don't care for writes. Must be set to 0 for RMWs. Indicates the priority of a read request. <ul style="list-style-type: none"> 1 - High priority read 0 - Low priority read Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
hif_wcmd_latency_dch1[(HIF_RQOS_TW-1):0]	I	<p>Valid when hif_wcmd_valid_dch1 is high and hif_wcmd_pri_dch1 indicates variable priority read/write (VPR/VPW) (channel 1). Don't care for other priority types.</p> <p>Specifies the timeout value of VPR/VPW request. The controller start down counting VPR/VPW timer when the request is accepted in the controller.</p> <p>If VPR/VPW timeout value is set to 0, the VPR/VPW request expires immediately as it enter the controller, thereby making it highest priority transaction class within the device.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1) && (UMCTL2_VPRW_EN)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wcmd_token_dch1[(MEMC_TAGBITS-1):0]	I	<p>Valid when hif_wcmd_valid_dch1 and hif_wcmd_type_dch1 indicates a read (channel 1). Don't care for writes.</p> <p>Token bits are provided with read requests and returned later with read data. Common uses for these bits include identifying the requester and re-ordering read data which may be returned out-of-order.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wcmd_length_dch1[(UMCTL2_CMD_LEN_BITS-1):0]	I	<p>Valid when hif_wcmd_valid_dch1 is high and hif_wcmd_type_dch1 indicates a Read.</p> <p>This signal is ignored for Write or RMW.</p> <p>Indicates the number of requested words</p> <p>If MEMC_BURST_LENGTH = 16 and MEMC_FREQ_RATIO = 4</p> <ul style="list-style-type: none"> ■ 2'b00 - Full RD or WR of 2 HIF data words ■ 2'b10 - Partial (Half) RD or WR of 1 HIF data words ■ 2'b11 - Invalid <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_wcmd_wdata_ptr_dch1[(MEMC_WDATA_PTR_BITS-1):0]	I	<p>Valid when hif_wcmd_valid_dch1 is high and hif_wcmd_type indicates a write. Pointer bits are provided with write requests and returned later with write pointer return corresponding to the write command issued (channel 1).</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wcmd_autopre_dch1	I	<p>Valid when hif_wcmd_valid_dch1 is high. Indicates that the command should be issued to memory with an auto-precharge (channel 1).</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wcmd_stall_dch1	O	<p>HIF Write commands are not accepted starting from the first clock edge after the stall goes high until the clock edge on which stall is detected low (channel 1).</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.21 HIF Command Interface Signals

hif_cmd_valid	-	hif_cmd_stall
hif_cmd_type	-	hif_wdata_ptr
hif_cmd_addr	-	hif_wdata_ptr_valid
hif_cmd_pri	-	hif_wdata_ptr_addr_err
hif_cmd_latency	-	hif_lpr_credit
hif_cmd_token	-	hif_wr_credit
hif_cmd_length	-	hif_hpr_credit
hif_cmd_wdata_ptr	-	hif_wrecc_credit
hif_cmd_autopre	-	hif_cmd_q_not_empty
hif_cmd_wdata_mask_full	-	hif_cmd_stall_dch1
hif_go2critical_hpr	-	hif_wdata_ptr_dch1
hif_go2critical_lpr	-	hif_wdata_ptr_valid_dch1
hif_go2critical_wr	-	hif_wdata_ptr_addr_err_dch1
hif_cmd_valid_dch1	-	hif_lpr_credit_dch1
hif_cmd_type_dch1	-	hif_wr_credit_dch1
hif_cmd_addr_dch1	-	hif_hpr_credit_dch1
hif_cmd_pri_dch1	-	hif_wrecc_credit_dch1
hif_cmd_latency_dch1	-	hif_cmd_q_not_empty_dch1
hif_cmd_token_dch1	-	
hif_cmd_length_dch1	-	
hif_cmd_wdata_ptr_dch1	-	
hif_cmd_autopre_dch1	-	
hif_cmd_wdata_mask_full_dch1	-	
hif_go2critical_hpr_dch1	-	
hif_go2critical_lpr_dch1	-	
hif_go2critical_wr_dch1	-	

Table 15-21 HIF Command Interface Signals

Port Name	I/O	Description
hif_cmd_valid	I	Valid command request to the controller. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (!UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_cmd_type[1:0]	I	Valid when hif_cmd_valid is high. Determines the type of command being issued: <ul style="list-style-type: none"> ■ 00 - Write ■ 01 - Read ■ 10 - RMW ■ 11 - Reserved Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (!UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
hif_cmd_addr[(MEMC_HIF_ADDR_WIDTH_MAX-1):0]	I	<p>Valid when hif_cmd_valid is high.</p> <p>Word address of the read or write request being made. Word size (in bits) is defined as the configuration parameter MEMC_DRAM_DATA_WIDTH.</p> <p>This bus is MEMC_HIF_ADDR_WIDTH_MAX bits into the DDRCTL, though several of the upper-most bits are generally be unused.</p> <p>hif_cmd_addr[3:0] refer to the critical word if MEMC_BURST_LENGTH=16.</p> <p>When critical word is not zero, interleaved/sequential ordering is used for writing data or returning read responses. See JEDEC specification for information on interleaved/sequential data ordering.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (!UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_cmd_pri[UMCTL2_VPR_EN_VAL:0]	I	<p>Valid when hif_cmd_valid is high.</p> <p>Determines the priority of command being issued.</p> <p>If UMCTL2_VPRW_EN=0, valid when hif_cmd_type indicates a read. Don't care for writes. Must be set to 0 for RMWs.</p> <ul style="list-style-type: none"> ■ 0 - Low priority read ■ 1 - High priority read ■ If UMCTL2_VPRW_EN=1, indicates the priority of a read/write request <p>hif_cmd_pri[1] must be set to 0 for RMWs</p> <ul style="list-style-type: none"> ■ 00 - Low priority read/write ■ 01 - Variable priority read/write ■ 10 - High priority read ■ 11 - Reserved <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (!UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_cmd_latency[(HIF_RQOS_TW-1):0]	I	<p>Valid when hif_cmd_valid is high and hif_cmd_pri indicates variable priority read/write (VPR/VPW). Don't care for other priority types. Specifies the timeout value of VPR/VPW request. The controller start down counting VPR/VPW timer when the request is accepted in the controller. If VPR/VPW timeout value is set to 0, the VPR/VPW request expires immediately as it enter the controller, thereby making it highest priority transaction class within the device.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (!UMCTL2_DUAL_HIF_1) && (UMCTL2_VPRW_EN)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_cmd_token[(MEMC_TAGBITS-1):0]	I	<p>Valid when hif_cmd_valid is high and hif_cmd_type indicates a read. Don't care for writes. Token bits are provided with read requests and returned later with read data. For DDRCTL, common uses for these bits include identifying the requester and re-ordering read data which may be returned out-of-order.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (!UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_cmd_length[(UMCTL2_CMD_LEN_BIT S-1):0]	I	<p>Valid when hif_cmd_valid is high and hif_cmd_type indicates a Read. This signal is ignored for Write or RMW. Indicates the number of requested words</p> <p>If MEMC_BURST_LENGTH = 16 and MEMC_FREQ_RATIO = 4</p> <ul style="list-style-type: none"> ■ 2'b00 - Full RD or WR of 2 HIF data words ■ 2'b10 - Partial (Half) RD or WR of 1 HIF data words ■ 2'b01 and 2'b11 - Invalid <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (!UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_cmd_wdata_ptr[(MEMC_WDATA_PTR _BITS-1):0]	I	<p>Valid when hif_cmd_valid is high and hif_cmd_type indicates a write. Pointer bits are provided with write requests and returned later with write pointer return corresponding to the write command issued.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (!UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_cmd_autopre	I	<p>Valid when hif_cmd_valid is high. Indicates that the command should be issued to memory with an auto-precharge.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (!UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_cmd_wdata_mask_full[(WRDATA_CYCLES-1):0]	I	<p>When 1, all bytes for the relevant HIF data beat are to be written. When 0, some or all of the bytes for the relevant HIF data beat are masked.</p> <p>Each bit corresponds to each HIF data beat in turn that is, hif_cmd_wdata_mask_full[0] corresponds to the 1st HIF data beat, hif_cmd_wdata_mask_full[1] corresponds to the 2nd HIF data beat, and so on.</p> <p>Used to determine if Masked Write (MWR) command is required or not. Valid only if LPDDR4 enabled, Valid only if posted write is enabled, Valid if Inline ECC enabled.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (MEMC_HIF_CMD_WDATA_MASK_FULL_EN)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_go2critical_hpr	I	<p>For DDRCTL, asserting this causes the HPR queue to go to the Critical state immediately, bypassing the starvation mechanism. This also causes the Controller to switch to Read mode, if it is currently in Write mode. This signal is provided to externally control the Read/Write switching in the Controller. This can be used by the external logic to switch the Read/Write mode if the read or write command queues outside the Controller start to fill up. See the section "Transaction Service Controls" for more details on the read queue and write queue.</p> <p>Exists: !UMCTL2_INCL_ARB_OR_CHB</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_go2critical_lpr	I	<p>For DDRCTL, asserting this causes the LPR queue to go to the Critical state immediately, bypassing the starvation mechanism.</p> <p>This also causes the controller to switch to Read mode, if it is currently in Write mode.</p> <p>This signal is provided to externally control the Read/Write switching in the controller. This can be used by the external logic to switch the Read/Write mode if the read or write command queues outside the controller start to fill up.</p> <p>For more information on the read queue and write queue, see "Transaction Service Controls" section.</p> <p>Exists: !UMCTL2_INCL_ARB_OR_CHB</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_go2critical_wr	I	<p>For DDRCTL, asserting this causes the WR queue to go to the Critical state immediately, bypassing the starvation mechanism.</p> <p>This does not explicitly cause the controller to switch to Write mode, if it is currently in Read mode. The switch happens only if the LPR and HPR queues are not currently in critical state. This signal is provided to externally control the Read/Write switching in the controller.</p> <p>This can be used by the external logic to switch the Read/Write mode if the read or write command queues outside the controller start to fill up.</p> <p>For more information on the read queue and write queue,, see "Transaction Service Controls" section.</p> <p>Exists: !UMCTL2_INCL_ARB_OR_CHB</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_cmd_stall	O	<p>HIF commands are not accepted starting from the first clock edge after the stall goes high until the clock edge on which stall is detected low.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (!UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wdata_ptr[(MEMC_WDATA_PTR_BITS -1):0]	O	<p>Valid when hif_wdata_ptr_valid is high.</p> <p>Write pointer bits are returned to indicate that SoC can send the corresponding write data.</p> <p>Exists: !UMCTL2_INCL_ARB_OR_CHB</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_wdata_ptr_valid	O	<p>Valid indication for the hif_wdata_ptr signal. This goes high when a write command is received.</p> <p>Exists: !UMCTL2_INCL_ARB_OR_CHB</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wdata_ptr_addr_err	O	<p>Valid when hif_wdata_ptr_valid is high. A 1 on this signal indicates that the HIF write request, denoted by pointer hif_wdata_ptr, has invalid address.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (MEMC_ADDR_ERR_EN)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_lpr_credit[(MEMC_HIF_CREDIT_BITS-1):0]	O	<p>Indicates that a low priority read request has been scheduled to the DDR and the SoC must increment the credit value associated with low priority reads.</p> <p>Note:In ECC configurations, the numbers of write and low priority read credits issued is one less than in the non-ECC case. One entry each is reserved in the write and low-priority read CAMs for storing the RMW requests arising out of single bit error correction RMW operation.</p> <p>In Inline ECC configuration, it is expand to 2 bits. Bit0 indicate that credit increment pulse from the CAM; Bit1 indicate credit increment pulse from the IH whenever the command does not need to inject an overhead command.</p> <p>Both bits can be assert at the same time, and in that case the SoC needs to increment credits by 2.</p> <p>Exists: !UMCTL2_INCL_ARB_OR_CHB</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wr_credit	O	<p>Indicates that a write request and write data have been scheduled to the DDR, or that a Write Combine has occurred, and the SoC must increment the credit value associated with writes.</p> <p>Note:In ECC configurations, the numbers of write and low priority read credits issued is one less than in the non-ECC case. One entry each is reserved in the write and low-priority read CAMs for storing the RMW requests arising out of single bit error correction RMW operation.</p> <p>Exists: !UMCTL2_INCL_ARB_OR_CHB</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_hpr_credit[(MEMC_HIF_CREDIT_BIT S-1):0]	O	<p>For DDRCTL, indicates that a high priority read request has been scheduled to the DDR and the SoC must increment the credit value associated with high priority reads.</p> <p>In Inline ECC configuration, it is expand to 2 bits.</p> <ul style="list-style-type: none"> ■ Bit0 indicates that credit increment pulse from the CAM. ■ Bit1 indicates that credit increment pulse from the IH whenever the command does not need to inject an overhead command. <p>Both bits can be assert at the same time, and in that case the SoC needs to increment credits by 2.</p> <p>Exists: !UMCTL2_INCL_ARB_OR_CHB</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wrecc_credit[1:0]	O	<p>Indicates that an ECC write request has been scheduled to the DDR and the SoC must increment the credit value associated with ECC writes.</p> <p>This is there only with Inline ECC configuration, it is 2 bits.</p> <ul style="list-style-type: none"> ■ Bit0 indicates that credit increment pulse from the CAM. ■ Bit1 indicates that credit increment pulse from the IH whenever the command does not need to inject an overhead command due to address error. <p>Both bits can be assert at the same time, and in that case the SoC needs to increment credits by 2.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (MEMC_INLINE_ECC)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_cmd_q_not_empty	O	<p>If asserted, indicates that the write and/or read CAMs in the controller are not empty.</p> <p>Exists: !UMCTL2_INCL_ARB_OR_CHB</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_cmd_valid_dch1	I	<p>Valid command request to the controller (channel 1).</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (!UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_cmd_type_dch1[1:0]	I	<p>Valid when hif_cmd_valid_dch1 is high. Determines the type of command being issued (channel 1):</p> <ul style="list-style-type: none"> ■ 00 - Write ■ 01 - Read ■ 10 - RMW (DDRCTL only) ■ 11 - Reserved <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (!UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
hif_cmd_addr_dch1[(MEMC_HIF_ADDR_WIDTH_MAX-1):0]	I	<p>Valid when hif_cmd_valid=1. Word address of the read or write request being made (channel 1). Word size (in bits) is defined as the configuration parameter MEMC_DRAM_DATA_WIDTH. This bus is MEMC_HIF_ADDR_WIDTH_MAX bits into the controller, though several of the upper-most bits are generally be unused. hif_cmd_addr[3:0] refer to the critical word if MEMC_BURST_LENGTH=16. When critical word is not zero, interleaved/sequential ordering is used for writing data or returning read responses. See JEDEC specification for information on interleaved/sequential data ordering.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (!UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
hif_cmd_pri_dch1[UMCTL2_VPR_EN_VAL:0]	I	<p>For DDRCTL, Valid when hif_cmd_valid_dch1 is high and hif_cmd_type_dch1 indicates a read (channel 1). Don't care for writes. Must be set to 0 for RMWs Indicates the priority of a read request.</p> <ul style="list-style-type: none"> ■ 1 - High priority read ■ 0 - Low priority read <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (!UMCTL2_DUAL_HIF_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_cmd_latency_dch1[(HIF_RQOS_TW-1):0]	I	<p>Valid when hif_cmd_valid_dch1 is high and hif_cmd_pri_dch1 indicates variable priority read/write (VPR/VPW) (channel 1). Don't care for other priority types.</p> <p>Specifies the timeout value of VPR/VPW request. The controller start down counting VPR/VPW timer when the request is accepted in the controller. If VPR/VPW timeout value is set to 0, the VPR/VPW request expires immediately as it enter the controller, thereby making it highest priority transaction class within the device.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (!UMCTL2_DUAL_HIF_1) && (UMCTL2_VPRW_EN)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_cmd_token_dch1[(MEMC_TAGBITS-1):0]	I	<p>Valid when hif_cmd_valid_dch1 and hif_cmd_type_dch1 indicates a read (channel 1). Don't care for writes.</p> <p>Token bits are provided with read requests and returned later with read data. Common uses for these bits include identifying the requester and re-ordering read data which may be returned out-of-order.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (!UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_cmd_length_dch1[(UMCTL2_CMD_LEN_BITS-1):0]	I	<p>Valid when hif_cmd_valid_dch1 is high and hif_cmd_type_dch1 indicates a Read.</p> <p>This signal is ignored for Write or RMW.</p> <p>Indicates the number of requested words</p> <p>If MEMC_BURST_LENGTH = 16 and MEMC_FREQ_RATIO = 4</p> <ul style="list-style-type: none"> ■ 2'b00 - Full RD or WR of 2 HIF data words ■ 2'b10 - Partial (Half) RD or WR of 1 HIF data words ■ 2'b01 and 2'b11 - Invalid <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (!UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_cmd_wdata_ptr_dch1[(MEMC_WDATA_PTR_BITS-1):0]	I	<p>Valid when hif_cmd_valid_dch1 is high and hif_cmd_type_dch1 indicates a write.</p> <p>Pointer bits are provided with write requests and returned later with write pointer return corresponding to the write command issued (channel 1).</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (!UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_cmd_autopre_dch1	I	<p>Valid when hif_cmd_valid_dch1 is high. Indicates that the command should be issued to memory with an auto-precharge (channel 1).</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (!UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_cmd_wdata_mask_full_dch1[(WRDATA_CYCLES-1):0]	I	<p>When 1, all bytes for the relevant HIF data beat are to be written (channel 1).</p> <p>When 0, some or all of the bytes for the relevant HIF data beat are masked.</p> <p>Each bit corresponds to each HIF data beat in turn that is, hif_cmd_wdata_mask_full_dch1[0] corresponds to the 1st HIF data beat, hif_cmd_wdata_mask_full_dch1[1] corresponds to the 2nd HIF data beat, and so on.</p> <p>Used to determine if Masked Write (MWR) command is required or not.</p> <p>Valid only if LPDDR4 enabled,</p> <p>Valid only if posted write is enabled,</p> <p>Valid if Inline ECC enabled.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (MEMC_HIF_CMD_WDATA_MASK_FULL_EN)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

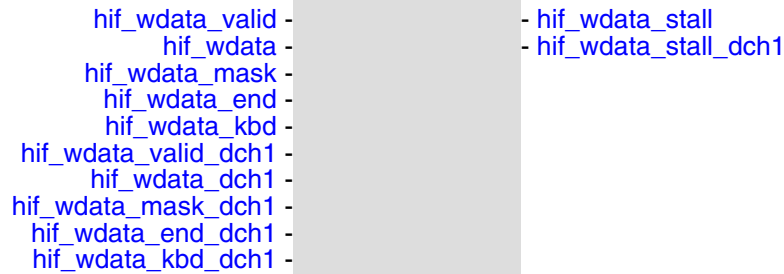
Port Name	I/O	Description
hif_go2critical_hpr_dch1	I	<p>For DDRCTL, asserting this causes the HPR queue to go to the Critical state immediately, bypassing the starvation mechanism (channel 1). This also causes the controller to switch to Read mode, if it is currently in Write mode.</p> <p>This signal is provided to externally control the Read/Write switching in the controller. This can be used by the external logic to switch the Read/Write mode if the read or write command queues outside the controller start to fill up.</p> <p>For more information on the read queue and write queue, see the section "Transaction Service Controls" section.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_go2critical_lpr_dch1	I	<p>For DDRCTL, asserting this causes the LPR queue to go to the Critical state immediately, bypassing the starvation mechanism (channel 1). This also causes the controller to switch to Read mode, if it is currently in Write mode.</p> <p>This signal is provided to externally control the Read/Write switching in the controller. This can be used by the external logic to switch the Read/Write mode if the read or write command queues outside the controller start to fill up.</p> <p>For more information on the read queue and write queue, see the section "Transaction Service Controls" section.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_go2critical_wr_dch1	I	<p>For DDRCTL, asserting this causes the WR queue to go to the Critical state immediately, bypassing the starvation mechanism (channel 1). This does not explicitly cause the controller to switch to Write mode, if it is currently in Read mode.</p> <p>The switch happens only if the LPR and HPR queues are not currently in critical state. This signal is provided to externally control the Read/Write switching in the controller.</p> <p>This can be used by the external logic to switch the Read/Write mode if the read or write command queues outside the controller start to fill up.</p> <p>For more information on the read queue and write queue, see the section "Transaction Service Controls" section.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_cmd_stall_dch1	O	<p>HIF commands are not accepted starting from the first clock edge after the stall goes high until the clock edge on which stall is detected low (channel 1).</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (!UMCTL2_DUAL_HIF_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wdata_ptr_dch1[(MEMC_WDATA_PTR_BITS-1):0]	O	<p>Valid when hif_wdata_ptr_valid_dch1 is high (channel 1). Write pointer bits are returned to indicate that SoC can send the corresponding write data.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wdata_ptr_valid_dch1	O	<p>Valid indication for the hif_wdata_ptr_dch1 signal (channel 1). This goes high when a write command is received.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wdata_ptr_addr_err_dch1	O	<p>Valid when hif_wdata_ptr_valid_dch1 is high. A 1 on this signal indicates that the HIF write request, denoted by pointer hif_wdata_ptr, has invalid address (channel 1).</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (MEMC_ADDR_ERR_EN)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_lpr_credit_dch1[(MEMC_HIF_CREDIT_BITS-1):0]	O	<p>Indicates that a low priority read request has been scheduled to the DDR and the SoC must increment the credit value associated with low priority reads (channel 1).</p> <p>Note:In ECC configurations, the numbers of write and low priority read credits issued is one less than in the non-ECC case. One entry each is reserved in the write and low-priority read CAMs for storing the RMW requests arising out of single bit error correction RMW operation. In Inline ECC configuration, it is expend to 2 bits.</p> <ul style="list-style-type: none"> ■ Bit0 indicates that the credit increment pulse from the CAM ■ Bit1 indicates that the credit increment pulse from the IH whenever the command does not need to inject an overhead command. <p>Both bits can be assert at the same time, and in that case the SoC needs to increment credits by 2.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wr_credit_dch1	O	<p>Indicates that a write request and write data have been scheduled to the DDR, or that a Write Combine has occurred, and the SoC must increment the credit value associated with writes (channel 1).</p> <p>Note:In ECC configurations, the numbers of write and low priority read credits issued is one less than in the non-ECC case. One entry each is reserved in the write and low-priority read CAMs for storing the RMW requests arising out of single bit error correction RMW operation. In Inline ECC configuration, it is expend to 2 bits.</p> <ul style="list-style-type: none"> ■ Bit0 indicates that the credit increment pulse from the CAM. ■ Bit1 indicates that the credit increment pulse from the IH whenever the command does not need to inject an overhead command. <p>Both bits can be assert at the same time, and in that case the SoC needs to increment credits by 2.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_hpr_credit_dch1[(MEMC_HIF_CREDIT_BITS-1):0]	O	<p>Indicates that a high priority read request has been scheduled to the DDR and the SoC must increment the credit value associated with high priority reads (channel 1). In Inline ECC configuration, it is expand to 2 bits.</p> <ul style="list-style-type: none"> ■ Bit0 indicates that the credit increment pulse from the CAM. ■ Bit1 indicates that the credit increment pulse from the IH whenever the command does not need to inject an overhead command. <p>Both bits can be assert at the same time, and in that case the SoC needs to increment credits by 2.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wrecc_credit_dch1[1:0]	O	<p>Indicates that an ECC write request has been scheduled to the DDR and the SoC must increment the credit value associated with ECC writes (channel 1). This is there only with Inline ECC configuration, it is 2 bits.</p> <ul style="list-style-type: none"> ■ Bit0 indicates that the credit increment pulse from the CAM. ■ Bit1 indicates that the credit increment pulse from the IH whenever the command does not need to inject an overhead command due to address error. <p>Both bits can be assert at the same time, and in that case the SoC needs to increment credits by 2.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (MEMC_INLINE_ECC)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_cmd_q_not_empty_dch1	O	<p>If asserted, indicates that the write and/or read CAMs in the controller are not empty (channel 1).</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.22 HIF Write Data Interface Signals



hif_wdata_valid -
 hif_wdata -
 hif_wdata_mask -
 hif_wdata_end -
 hif_wdata_kbd -
 hif_wdata_valid_dch1 -
 hif_wdata_dch1 -
 hif_wdata_mask_dch1 -
 hif_wdata_end_dch1 -
 hif_wdata_kbd_dch1 -

- hif_wdata_stall
 - hif_wdata_stall_dch1

Table 15-22 HIF Write Data Interface Signals

Port Name	I/O	Description
hif_wdata_valid	I	Valid for the write data bus (hif_wdata) Exists: !UMCTL2_INCL_ARB_OR_CHB Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_wdata[(((MEMC_DRAM_DATA_WIDTH * MEMC_FREQ_RATIO) * 2) - 1) : 0]	I	Write data bus. Valid when hif_wdata_valid is high. Exists: !UMCTL2_INCL_ARB_OR_CHB Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_wdata_mask[(((MEMC_DRAM_DATA_WIDTH * MEMC_FREQ_RATIO) * 2) / 8) - 1) : 0]	I	Write data mask bus. One bit of hif_wdata_mask signal is for each byte of data. <ul style="list-style-type: none"> hif_wdata_mask[0] is for hif_wdata[7:0] hif_wdata_mask[1] is for hif_wdata[15:8] (and so on.) <ul style="list-style-type: none"> 1: The byte is written to SDRAM 0: The byte is not written to SDRAM Valid when hif_wdata_valid = 1 When DDR4 SDRAM is used and data mask is disabled, this signal must set to all 1 in Write command. Exists: !UMCTL2_INCL_ARB_OR_CHB Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
hif_wdata_end	I	<p>Valid when hif_wdata_valid is high. Indicates the final write data phase for a write command. For MEMC_BURST_LENGTH = 16, MEMC_FREQ_RATIO = 4, this is asserted on the second clock for a normal write command, or the first clock for a partial write command.</p> <p>Exists: !UMCTL2_INCL_ARB_OR_CHB Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
hif_wdata_kbd[(DDRCTL_HIF_KBD_WIDTH-1):0]	I	<p>Write data KBD. Valid when hif_wdata_valid is high.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (DDRCTL_KBD_SBECC_EN_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
hif_wdata_stall	O	<p>Data and mask presented on the data interface is accepted by the controller, when hif_wdata_valid is detected high and hif_wdata_stall is detected low on the same positive edge of the clock.</p> <ul style="list-style-type: none"> For DDRCTL, hif_wdata_stall is asserted during a RMW operation. When the DDRC writes the read data for the RMW into the write data SRAM, it blocks the HIF from sending any write data during those cycles. <p>Exists: !UMCTL2_INCL_ARB_OR_CHB Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
hif_wdata_valid_dch1	I	<p>Valid for the write data bus (hif_wdata) (channel 1)</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
hif_wdata_dch1[(((MEMC_DRAM_DATA_WIDTH*MEMC_FREQ_RATIO)*2)-1):0]	I	<p>Write data bus (channel 1). Valid when hif_wdata_valid_dch1 is high.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
hif_wdata_mask_dch1[(((MEMC_DRAM_DATA_WIDTH*MEMC_FREQ_RATIO)*2)/8)-1):0]	I	<p>Write data mask bus (channel 1). One bit of hif_wdata_mask_dch1 signal is for each byte of data.</p> <ul style="list-style-type: none"> hif_wdata_mask_dch1[0] is for hif_wdata_dch1[7:0] hif_wdata_mask_dch1[1] is for hif_wdata_dch1[15:8] (and so on.) 1: The byte is written to SDRAM 0: The byte is not written to SDRAM <p>Valid when hif_wdata_valid_dch1 = 1. When DDR4 SDRAM is used and data mask is disabled, this signal must set to all 1 in Write command.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wdata_end_dch1	I	<p>Valid when hif_wdata_valid_dch1 is 1. Indicates the final write data phase for a write command (channel 1). For MEMC_BURST_LENGTH = 16, MEMC_FREQ_RATIO = 4, this is asserted on the second clock for a normal write command, or the first clock for a partial write command.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wdata_kbd_dch1[(DDRCTL_HIF_KBD_WIDTH-1):0]	I	<p>Write data KBD (channel 1). Valid when hif_wdata_valid_dch1 is high.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (DDRCTL_KBD_SBECC_EN_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_wdata_stall_dch1	O	<p>Data and mask presented on the data interface is accepted by the controller, when hif_wdata_valid_dch1 is detected high and hif_wdata_stall_dch1 is detected low on the same positive edge of the clock (channel 1).</p> <ul style="list-style-type: none"> For uMCLT2, hif_wdata_stall_dch1 is asserted during a RMW operation. When the DDRC writes the read data for the RMW into the write data SRAM, it blocks the HIF from sending any write data during those cycles. <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.23 HIF Read Data Interface Signals

	<ul style="list-style-type: none"> - hif_rdata_valid - hif_rdata_end - hif_rdata_token - hif_rdata - hif_rdata_uncorr_ecc_err - hif_rdata_crc_err - hif_rdata_eapar_err - hif_rdata_uncorr_linkecc_err - hif_rdata_addr_err - hif_rdata_kbd - hif_rdata_valid_dch1 - hif_rdata_end_dch1 - hif_rdata_token_dch1 - hif_rdata_dch1 - hif_rdata_uncorr_ecc_err_dch1 - hif_rdata_crc_err_dch1 - hif_rdata_eapar_err_dch1 - hif_rdata_addr_err_dch1 - hif_rdata_kbd_dch1
--	---

Table 15-23 HIF Read Data Interface Signals

Port Name	I/O	Description
hif_rdata_valid	O	Valid for the read response data. Exists: !UMCTL2_INCL_ARB_OR_CHB Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rdata_end	O	Valid when hif_rdata_valid is high. Indicates the final write data phase for a read command. For MEMC_BURST_LENGTH = 16, MEMC_FREQ_RATIO = 4, this is asserted on the second clock for a normal read command, or the first clock for a partial read command. Exists: !UMCTL2_INCL_ARB_OR_CHB Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rdata_token[(MEMC_TAGBITS-1):0]	O	Token associated with the read command and the read data that is going out. Valid with hif_rdata_valid. Exists: !UMCTL2_INCL_ARB_OR_CHB Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
hif_rdata[(((MEMC_DRAM_DATA_WIDTH* MEMC_FREQ_RATIO)*2)-1):0]	O	Read data. Valid when hif_rdata_valid is high. Exists: !UMCTL2_INCL_ARB_OR_CHB Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rdata_uncorr_ecc_err	O	A 1 on this signal indicates that the read data returned on 'hif_rdata_rdata' bus has uncorrectable ECC error on it. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (MEMC_ECC) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rdata_crc_err	O	A 1 on this signal indicates that the read data returned on 'hif_rdata' bus has CRC error on it. <ul style="list-style-type: none"> ■ For 1:4 frequency ratio mode, valid when hif_rdata_valid is high. This signal indicates status of each HIF data beat. ■ For 1:2 frequency ratio mode, valid when hif_rdata_valid and hif_rdata_end are high. This signal indicates status of HIF data burst. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (MEMC_DDR5) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rdata_eapar_err	O	A 1 on this signal indicates that the read data returned on 'hif_rdata_rdata' bus has EAPAR error on it. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (DDRCTL_EAPAR_EN_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rdata_uncorr_linkecc_err	O	A 1 on this signal indicates that the read data returned on 'hif_rdata' bus has LinkECC uncorrectable error on it. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (MEMC_LINK_ECC) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rdata_addr_err	O	Valid when hif_rdata_valid is high. A 1 on this signal indicates that the read token returned on hif_rdata_token was associated with an invalid address. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (MEMC_ADDR_ERR_EN) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
hif_rdata_kbd[(DDRCTL_HIF_KBD_WIDT H-1):0]	O	Read data KBD. Valid when hif_rdata_valid is high. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (DDRCTL_KBD_SBECC_EN_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rdata_valid_dch1	O	Valid for the read response data (channel 1). Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rdata_end_dch1	O	Indicates the end of the read data for a read command (channel 1). For MEMC_BURST_LENGTH = 16, MEMC_FREQ_RATIO = 4, this is asserted on the second clock for a full read command, or the first clock for a partial (half) read command. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rdata_token_dch1[(MEMC_TAGBITS-1):0]	O	Token associated with the read command and the read data that is going out (channel 1). SoC must use this for re-assembling the read data in order. Valid with hif_rdata_valid_dch1. Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rdata_dch1[(((MEMC_DRAM_DATA_WIDTH*MEMC_FREQ_RATIO)*2)-1):0]	O	Read data. Valid when hif_rdata_valid_dch1 is high (channel 1). Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
hif_rdata_uncorr_ecc_err_dch1	O	A 1 on this signal indicates that the read data returned on 'hif_rdata_rdata' bus has uncorrectable ECC error on it (channel 1). Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (MEMC_ECC) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
hif_rdata_crc_err_dch1	O	<p>A 1 on this signal indicates that the read data returned on 'hif_rdata' bus has CRC error on it (channel 1).</p> <ul style="list-style-type: none"> ■ For 1:4 frequency ratio mode, valid when hif_rdata_valid is high. This signal indicates status of each HIF data beat. ■ For 1:2 frequency ratio mode, valid when hif_rdata_valid and hif_rdata_end are high. This signal indicates status of HIF data burst. <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (MEMC_DDR5)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_rdata_eapar_err_dch1	O	<p>A 1 on this signal indicates that the read data returned on 'hif_rdata_rdata' bus has EAPAR error on it (channel 1).</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (DDRCTL_EAPAR_EN_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_rdata_addr_err_dch1	O	<p>Valid when hif_rdata_valid_dch1 is high.</p> <p>A 1 on this signal indicates that the read token returned on hif_rdata_token was associated with an invalid address (channel 1).</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (MEMC_ADDR_ERR_EN)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_rdata_kbd_dch1[(DDRCTL_HIF_KBD_WIDTH-1):0]	O	<p>Read data KBD (channel 1). Valid when hif_rdata_valid_dch1 is high.</p> <p>Exists: (!UMCTL2_INCL_ARB_OR_CHB) && (UMCTL2_DUAL_CHANNEL) && (DDRCTL_KBD_SBECC_EN_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.24 Write Data RAM Interface Signals

wdataram_dout	-	wdataram_din
wdataram_dout_par	-	wdataram_mask
wdataram_dout_dch1	-	wdataram_wr
wdataram_dout_par_dch1	-	wdataram_waddr
		wdataram_re
		wdataram_raddr
		wdataram_din_par
		wdataram_din_dch1
		wdataram_mask_dch1
		wdataram_wr_dch1
		wdataram_waddr_dch1
		wdataram_re_dch1
		wdataram_raddr_dch1
		wdataram_din_par_dch1

Table 15-24 Write Data RAM Interface Signals

Port Name	I/O	Description
wdataram_din[(UMCTL2_WDATARAM_D W-1):0]	O	<p>Data for the write operation. Valid with wdataram_wr. In HIF configurations with ECC enabled, the width of this bus increases to include the ECC byte.</p> <p>Exists: UMCTL2_WDATA_EXTRAM</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
wdataram_dout[(UMCTL2_WDATARAM_D W-1):0]	I	<p>Read data coming from the write data RAM. Valid data comes out one clock after wdataram_re = 1. In HIF configurations with ECC enabled, the width of this bus increases to include the ECC byte.</p> <p>Exists: UMCTL2_WDATA_EXTRAM</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

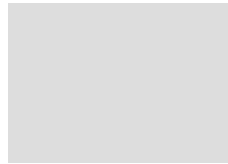
Port Name	I/O	Description
wdataram_mask[((UMCTL2_WDATARAM_DW/8)-1):0]	O	<p>Mask for the write operation. One bit of wdataram_mask signal is for each byte of data.</p> <p>wdataram_mask[0] is for wdataram_din[7:0] wdataram_mask[1] is for wdataram_din[15:8] (and so on.)</p> <p>For On Chip Parity configurations (UMCTL2_OCPAR_EN==1) wdataram_mask[0] is for wdataram_din_par[0] wdataram_mask[1] is for wdataram_din_par[1] (and so on.)</p> <p>For On Chip Parity configurations (UMCTL2_OCPAR_EN==1) with DDRCTL_OCSAP_EN=1 wdataram_mask[0] is for wdataram_din_par[1:0] wdataram_mask[1] is for wdataram_din_par[3:2] (and so on.)</p> <p>For On Chip ECC configurations (UMCTL2_OCECC_EN==1) wdataram_mask[0] is for wdataram_din_par[4:0] wdataram_mask[1] is for wdataram_din_par[9:5] (and so on.)</p> <ul style="list-style-type: none"> ■ 1: The byte should be written to the write data RAM ■ 0: The byte should not be written to the write data RAM <p>Valid with wdataram_wr = 1. In HIF configurations with ECC enabled, the width of this bus increases to include the mask for the ECC byte.</p> <p>Exists: UMCTL2_WDATA_EXTRAM</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
wdataram_wr	O	<p>Write signal to write data RAM.</p> <p>Exists: UMCTL2_WDATA_EXTRAM</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
wdataram_waddr[((UMCTL2_WDATARAM_AW-1):0]	O	<p>Write address for the write operation. Valid with wdataram_wr = 1.</p> <p>Exists: UMCTL2_WDATA_EXTRAM</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
wdataram_re	O	<p>Read signal to the write data RAM.</p> <p>Exists: UMCTL2_WDATA_EXTRAM</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
wdataram_raddr[(UMCTL2_WDATARAM_AW-1):0]	O	Read address to the write data RAM. Valid with wdataram_re = 1. Exists: UMCTL2_WDATA_EXTRAM Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
wdataram_din_par[(UMCTL2_WDATARAM_PAR_DW_EXT-1):0]	O	Parity/ECC of the data for the write operation. Exists: UMCTL2_OCPAR_OR_OCECC_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
wdataram_dout_par[(UMCTL2_WDATARAM_PAR_DW_EXT-1):0]	I	Parity/ECC of the read data coming from the write data RAM. Exists: UMCTL2_OCPAR_OR_OCECC_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
wdataram_din_dch1[(UMCTL2_WDATARAM_DW-1):0]	O	Data for the write operation (data channel 1). Valid with wdataram_wr_dch1. In HIF configurations with ECC enabled, the width of this bus increases to include the ECC byte. Exists: (UMCTL2_DUAL_CHANNEL) && (UMCTL2_WDATA_EXTRAM) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
wdataram_dout_dch1[(UMCTL2_WDATARAM_DW-1):0]	I	Read data coming from the write data RAM (data channel 1). Valid data comes out one clock after wdataram_re_dch1 = 1. In HIF configurations with ECC enabled, the width of this bus increases to include the ECC byte. Exists: (UMCTL2_DUAL_CHANNEL) && (UMCTL2_WDATA_EXTRAM) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
wdataram_mask_dch1[((UMCTL2_WDATA_RAM_DW/8)-1):0]	O	<p>Mask for the write operation. One bit of wdataram_mask_dch1 signal is for each byte of data and corresponding parity if any.</p> <p>wdataram_mask_dch1[0] is for wdataram_din_dch1[7:0] wdataram_mask_dch1[1] is for wdataram_din_dch1[15:8] (and so on.)</p> <p>For On Chip Parity configurations (UMCTL2_OCPAR_EN==1) wdataram_mask_dch1[0] is for wdataram_din_par_dch1[0] wdataram_mask_dch1[1] is for wdataram_din_par_dch1[1] (and so on.)</p> <ul style="list-style-type: none"> ■ 1: The byte should be written to the write data RAM ■ 0: The byte should not be written to the write data RAM <p>Valid with wdataram_wr_dch1 = 1. In HIF configurations with ECC enabled, the width of this bus increases to include the mask for the ECC byte.</p> <p>Exists: (UMCTL2_DUAL_CHANNEL) && (UMCTL2_WDATA_EXTRAM)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
wdataram_wr_dch1	O	<p>Write signal to write data RAM (data channel 1).</p> <p>Exists: (UMCTL2_DUAL_CHANNEL) && (UMCTL2_WDATA_EXTRAM)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
wdataram_waddr_dch1[(UMCTL2_WDATA_RAM_AW-1):0]	O	<p>Write address for the write operation (data channel 1). Valid with wdataram_wr_dch1 = 1.</p> <p>Exists: (UMCTL2_DUAL_CHANNEL) && (UMCTL2_WDATA_EXTRAM)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
wdataram_re_dch1	O	<p>Read signal to the write data RAM (data channel 1).</p> <p>Exists: (UMCTL2_DUAL_CHANNEL) && (UMCTL2_WDATA_EXTRAM)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
wdataram_raddr_dch1[(UMCTL2_WDATA_RAM_AW-1):0]	O	Read address to the write data RAM (data channel 1). Valid with wdataram_re_dch1 = 1. Exists: (UMCTL2_DUAL_CHANNEL) && (UMCTL2_WDATA_EXTRAM) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
wdataram_din_par_dch1[(UMCTL2_WDATA_RAM_PAR_DW_EXT-1):0]	O	Parity of the data for the write operation (data channel 1). Exists: (UMCTL2_DUAL_CHANNEL) && (UMCTL2_OCPAR_EN_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
wdataram_dout_par_dch1[(UMCTL2_WDATA_RAM_PAR_DW_EXT-1):0]	I	Parity of the read data coming from the write data RAM (data channel 1). Exists: (UMCTL2_DUAL_CHANNEL) && (UMCTL2_OCPAR_EN_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.25 Mode Register Read/Write Signals

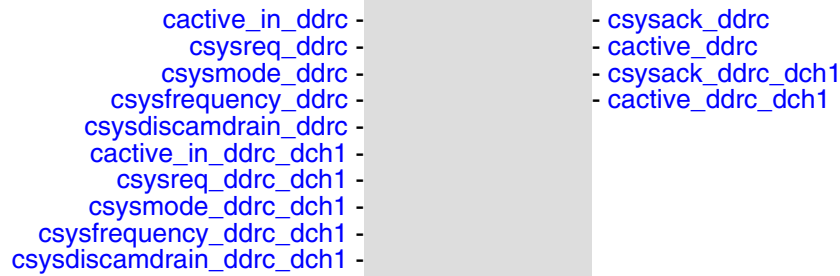


- hif_mrr_data
- hif_mrr_data_valid
- hif_mrr_data_dch1
- hif_mrr_data_valid_dch1

Table 15-25 Mode Register Read/Write Signals

Port Name	I/O	Description
hif_mrr_data[(MEMC_MRR_DATA_TOTAL_DATA_WIDTH-1):0]	O	<p>LPDDR4/5: Mode register read data. This signal is valid when hif_mrr_data_valid is high and present only in designs configured to support LPDDR4/5.</p> <p>Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
hif_mrr_data_valid	O	<p>When asserted high, this signal indicates that data on hif_mrr_data is valid. This signal is present only in designs configured to support LPDDR4/5.</p> <p>Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
hif_mrr_data_dch1[(MEMC_DFI_TOTAL_DATA_WIDTH-1):0]	O	<p>LPDDR4/5: Mode register read data. This signal is valid when hif_mrr_data_valid_dch1 is high and present only in designs configured to support LPDDR4/5 for Channel 1.</p> <p>Exists: UMCTL2_DUAL_CHANNEL Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
hif_mrr_data_valid_dch1	O	<p>When asserted high, this signal indicates that data on hif_mrr_data_dch1 is valid for Channel 1. This signal is present only in designs configured to support LPDDR4/5.</p> <p>Exists: UMCTL2_DUAL_CHANNEL Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>

15.26 DDRC Hardware Low Power Signals



cactive_in_ddrc - csysack_ddrc
 csysreq_ddrc - cactive_ddrc
 csysmode_ddrc - csysack_ddrc_dch1
 csysfrequency_ddrc - cactive_ddrc_dch1
 csysdiscamdrain_ddrc
 cactive_in_ddrc_dch1
 csysreq_ddrc_dch1
 csysmode_ddrc_dch1
 csysfrequency_ddrc_dch1
 csysdiscamdrain_ddrc_dch1

Table 15-26 DDRC Hardware Low Power Signals

Port Name	I/O	Description
cactive_in_ddrc[(NPORTS-1):0]	I	<p>DDRC Hardware Low-Power Clock Active In. External asynchronous signal from the system that flags if a hardware low power request can be accepted or should always be denied.</p> <ul style="list-style-type: none"> 0 - Accepted 1 - Denied <p>Can also be used to exit the auto clock stop, power down, self refresh states modes. Driving c_active_in to 1 exits these modes. Note PWRCTL.hw_lp_exit_idle_en needs to be programmed to 1 to enable this functionality. For a system that does not generate this signal, it is recommended that this signal should be tied low to avoid issues due to undriven input. As this signal is an asynchronous input, it is required that it be glitch free.</p> <p>Exists: !UMCTL2_INCL_ARB_OR_CHB</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
csysreq_ddrc	I	<p>DDRC Hardware Low-Power Request. Request from the system clock controller for the peripheral (DDRC) to enter a low-power state.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
csysmode_ddrc	I	<p>DDRC Hardware Fast Frequency Change mode.</p> <ul style="list-style-type: none"> 0 - Hardware Low-Power is requested 1 - Hardware Fast Frequency Change is requested <p>This signal should remain stable during csysreq_ddrc=0.</p> <p>Exists: UMCTL2_HWFFC_EN</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
csysfrequency_ddrc[(TARGET_FREQUENCY_WIDTH-1):0]	I	<p>Target frequency for DDRC Hardware Fast Frequency Change. This signal is effective only when Hardware Fast Frequency Change is requested (that is, csysreq_ddrc=0 and csysmode_ddrc=1).</p> <p>Exists: UMCTL2_HWFFC_EN</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
csysdiscamdrain_ddrc	I	<p>Disable CAM draining for DDRC Hardware Fast Frequency Change. When asserted, Self-Refresh can be entered without draining CAM. This signal is effective only when Hardware Fast Frequency Change is requested (that is, csysreq_ddrc=0 and csysmode_ddrc=1).</p> <p>Exists: UMCTL2_HWFFC_EN</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
csysack_ddrc	O	<p>DDRC Hardware Low-Power Request Acknowledge. Acknowledgement from the peripheral (DDRC) of a system low-power request.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
cactive_ddrc	O	<p>DDRC Hardware Low-Power Clock Active. Indicates that the peripheral (DDRC) requires its clock signal.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
cactive_in_ddrc_dch1[(INT_NPORTS_DATA-1):0]	I	<p>DDRC Hardware Low-Power Clock Active In (channel 1). External asynchronous signal from the system that flags if a hardware low power request can be accepted or should always be denied.</p> <ul style="list-style-type: none"> 0 - Accepted 1 - Denied <p>Can also be used to exit the auto clock stop, power down, self refresh states modes. Driving c_active_in to 1 exits these modes. Note PWRCTL.hw_lp_exit_idle_en needs to be programmed to 1 to enable this functionality. For a system that does not generate this signal, it is recommended that this signal should be tied low to avoid issues due to undriven input. As this signal is an asynchronous input, it is required that it be glitch free.</p> <p>Exists: (UMCTL2_DUAL_CHANNEL) && (!UMCTL2_INCL_ARB_OR_CHB)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
csysreq_ddrc_dch1	I	<p>DDRC Hardware Low-Power Request. Request from the system clock controller for the peripheral (DDRC_dch1) to enter a low-power state.</p> <p>Exists: UMCTL2_DUAL_CHANNEL</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
csysmode_ddrc_dch1	I	<p>DDRC Hardware Fast Frequency Change mode (channel 1).</p> <ul style="list-style-type: none"> 0 - Hardware Low-Power is requested 1 - Hardware Fast Frequency Change is requested <p>This signal should remain stable during csysreq_ddrc_dch1=0.</p> <p>Exists: (UMCTL2_DUAL_CHANNEL) && (UMCTL2_HWFFC_EN)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
csysfrequency_ddrc_dch1[(TARGET_FREQUENCY_WIDTH-1):0]	I	<p>Target frequency for DDRC Hardware Fast Frequency Change. This signal is effective only when Hardware Fast Frequency Change is requested (that is, csysreq_ddrc_dch1=0 and csysmode_ddrc_dch1=1).</p> <p>Exists: (UMCTL2_DUAL_CHANNEL) && (UMCTL2_HWFFC_EN)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
csysdiscamdrain_ddrc_dch1	I	<p>Disable CAM draining for DDRC Hardware Fast Frequency Change. When asserted, Self-Refresh can be entered without draining CAM. This signal is effective only when Hardware Fast Frequency Change is requested (that is, csysreq_ddrc_dch1=0 and csysmode_ddrc_dch1=1).</p> <p>Exists: (UMCTL2_DUAL_CHANNEL) && (UMCTL2_HWFFC_EN)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
csysack_ddrc_dch1	O	<p>DDRC Hardware Low-Power Request Acknowledge. Acknowledgement from the peripheral (DDRC_dch1) of a system low-power request.</p> <p>Exists: UMCTL2_DUAL_CHANNEL</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
cactive_ddrc_dch1	O	<p>DDRC Hardware Low-Power Clock Active. Indicates that the peripheral (DDRC_dch1) requires its clock signal.</p> <p>Exists: UMCTL2_DUAL_CHANNEL</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

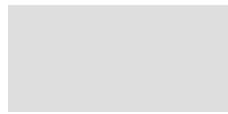
15.27 DDRC Self Refresh Signals



Table 15-27 DDRC Self Refresh Signals

Port Name	I/O	Description
stat_ddrc_reg_selfref_type[(SELFREF_TY PE_WIDTH-1):0]	O	DDRC 0 Self Refresh status and type. Equivalent to STAT.selfref_type register. Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
stat_ddrc_dch1_reg_selfref_type[(SELFR EF_TYPE_WIDTH-1):0]	O	DDRC 1 Self Refresh status and type. Equivalent to STAT.selfref_type register. Exists: UMCTL2_DUAL_CHANNEL Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.28 Inline ECC Debug Signals



- dbg_dfi_ie_cmd_type
- dbg_dfi_ie_cmd_type_dch1

Table 15-28 Inline ECC Debug Signals

Port Name	I/O	Description
dbg_dfi_ie_cmd_type[2:0]	O	<p>DDRC 0 Inline ECC Debug signal. This signal is valid when dfi command/address is RD/RDA/WR/WRA, otherwise don't care.</p> <p>With RD/RDA commands</p> <ul style="list-style-type: none"> ■ 000: RD_N (RD Data for non-ECC region) ■ 001: RD_E (RD Data for ECC region) ■ 010: RE_B (RD ECC in block read/write) ■ 111: MPR read (DDR4 only) <p>With WR/WRA commands</p> <ul style="list-style-type: none"> ■ 000: WD_N (WR Data for non-ECC region) ■ 001: WD_E (WR Data for ECC region) ■ 010: WE_BW (WR ECC in block write) ■ 111: MPR write (DDR4 only) <p>DEBUG ONLY</p> <p>Exists: MEMC_INLINE_ECC</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dbg_dfi_ie_cmd_type_dch1[2:0]	O	<p>DDRC 1 Inline ECC Debug signal. This signal is valid when dfi command/address is RD/RDA/WR/WRA, otherwise don't care.</p> <p>With RD/RDA commands</p> <ul style="list-style-type: none"> ■ 000: RD_N (RD Data for non-ECC region) ■ 001: RD_E (RD Data for ECC region) ■ 010: RE_B (RD ECC in block read/write) ■ 111: MPR read (DDR4 only) <p>With WR/WRA commands</p> <ul style="list-style-type: none"> ■ 000: WD_N (WR Data for non-ECC region) ■ 001: WD_E (WR Data for ECC region) ■ 010: WE_BW (WR ECC in block write) ■ 111: MPR write (DDR4 only) <p>DEBUG ONLY</p> <p>Exists: (MEMC_INLINE_ECC) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.29 Performance Logging Signals

- perf_hif_rd_or_wr
- perf_hif_wr
- perf_hif_rd
- perf_hif_rmw
- perf_hif_hi_pri_rd
- perf_read_bypass
- perf_act_bypass
- perf_dfi_wr_data_cycles
- perf_dfi_rd_data_cycles
- perf_hpr_xact_when_critical
- perf_lpr_xact_when_critical
- perf_wr_xact_when_critical
- perf_op_is_activate
- perf_op_is_rd_or_wr
- perf_op_is_rd_activate
- perf_op_is_rd
- perf_op_is_wr
- perf_op_is_mwr
- perf_op_is_cas
- perf_op_is_cas_ws
- perf_op_is_cas_ws_off
- perf_op_is_cas_wck_sus
- perf_op_is_enter_dsm
- perf_op_is_rfm
- perf_op_is_precharge
- perf_precharge_for_rdwr
- perf_precharge_for_other
- perf_rdwr_transitions
- perf_write_combine
- perf_write_combine_noecc
- perf_write_combine_wrecc
- perf_war_hazard
- perf_raw_hazard
- perf_waw_hazard
- perf_ie_blk_hazard
- perf_op_is_enter_selfref
- perf_op_is_enter_powerdown
- perf_selfref_mode
- perf_op_is_refresh
- perf_op_is_crit_ref
- perf_op_is_spec_ref
- perf_op_is_load_mode
- perf_rank
- perf_bank
- perf_bg
- perf_cid
- perf_hpr_req_with_nocredit
- perf_lpr_req_with_nocredit
- perf_bsm_alloc
- perf_bsm_starvation
- perf_num_alloc_bsm
- perf_visible_window_limit_reached_rd
- perf_visible_window_limit_reached_wr
- perf_op_is_dqsosc_mpc
- perf_op_is_dqsosc_mrr
- perf_op_is_tcr_mrr
- perf_op_is_zqstart
- perf_op_is_zqlatch
- perf_hif_rd_or_wr_dch1

- perf_hif_wr_dch1
- perf_hif_rd_dch1
- perf_hif_rmw_dch1
- perf_hif_hi_pri_rd_dch1
- perf_read_bypass_dch1
- perf_act_bypass_dch1
- perf_dfi_wr_data_cycles_dch1
- perf_dfi_rd_data_cycles_dch1
- perf_hpr_xact_when_critical_dch1
- perf_lpr_xact_when_critical_dch1
- perf_wr_xact_when_critical_dch1
- perf_op_is_activate_dch1
- perf_op_is_rd_or_wr_dch1
- perf_op_is_rd_activate_dch1
- perf_op_is_rd_dch1
- perf_op_is_wr_dch1
- perf_op_is_mwr_dch1
- perf_op_is_cas_dch1
- perf_op_is_cas_ws_dch1
- perf_op_is_cas_ws_off_dch1
- perf_op_is_cas_wck_sus_dch1
- perf_op_is_enter_dsm_dch1
- perf_op_is_rfm_dch1
- perf_op_is_precharge_dch1
- perf_precharge_for_rdwr_dch1
- perf_precharge_for_other_dch1
- perf_rdwr_transitions_dch1
- perf_write_combine_dch1
- perf_write_combine_noecc_dch1
- perf_write_combine_wrecc_dch1
- perf_war_hazard_dch1
- perf_raw_hazard_dch1
- perf_waw_hazard_dch1
- perf_ie_blk_hazard_dch1
- perf_op_is_enter_selfref_dch1
- perf_op_is_enter_powerdown_dch1
- perf_selfref_mode_dch1
- perf_op_is_refresh_dch1
- perf_op_is_crit_ref_dch1
- perf_op_is_spec_ref_dch1
- perf_op_is_load_mode_dch1
- perf_rank_dch1
- perf_bank_dch1
- perf_bg_dch1
- perf_cid_dch1
- perf_hpr_req_with_nocredit_dch1
- perf_lpr_req_with_nocredit_dch1
- perf_bsm_alloc_dch1
- perf_bsm_starvation_dch1
- perf_num_alloc_bsm_dch1
- perf_visible_window_limit_reached_rd_dch1
- perf_visible_window_limit_reached_wr_dch1
- perf_op_is_dqsosc_mpc_dch1
- perf_op_is_dqsosc_mrr_dch1
- perf_op_is_tcr_mrr_dch1
- perf_op_is_zqstart_dch1
- perf_op_is_zqlatch_dch1

Table 15-29 Performance Logging Signals

Port Name	I/O	Description
perf_hif_rd_or_wr	O	Asserts for every Read or Write command send to DDRC. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_hif_wr	O	Asserts for every Write command send to DDRC. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_hif_rd	O	Asserts for every Read command send to DDRC. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_hif_rmw	O	Asserts for every RMW command send to DDRC. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_hif_hi_pri_rd	O	Asserts for every High-Priority Read command send to DDRC. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_read_bypass	O	Asserts for every Read command that is send through the Bypass path. Exists: (MEMC_PERF_LOG_ON) && (MEMC_RD_BYPASS) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_act_bypass	O	Asserts for every Activate command that is send through the Bypass path. Exists: (MEMC_PERF_LOG_ON) && (MEMC_ACT_BYPASS) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
perf_dfi_wr_data_cycles	O	Asserts for every write data beat transfer on the DFI interface going to DRAM. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_dfi_rd_data_cycles	O	Asserts for every read data beat transfer on the DFI interface coming from DRAM. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_hpr_xact_when_critical	O	Asserts for every High Priority Read transaction that is scheduled when the High Priority Queue is in Critical state. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_lpr_xact_when_critical	O	Asserts for every Low Priority Read transaction that is scheduled when the Low Priority Queue is in Critical state. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_wr_xact_when_critical	O	Asserts for every Write transaction that is scheduled when the Write Queue is in Critical state. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_activate	O	Asserts for every Activate that is issued for commands going through CAM. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
perf_op_is_rd_or_wr	O	Asserts for every Read or Write that is issued for commands going through CAM. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_rd_activate	O	Asserts for every Read Activate that is issued for commands going through CAM. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_rd	O	Asserts for every Read that is issued for commands going through CAM. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_wr	O	Asserts for every Write that is issued for commands going through CAM. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_mwr	O	Asserts for every Masked Write that is issued for commands going through CAM. Not Applicable for DDR4 DRAM. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_cas	O	Asserts for every CAS command that is issued by the controller. Exists: (MEMC_PERF_LOG_ON) && (DDRCTL_LPDDR) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_cas_ws	O	Asserts for every CAS-WS_RD/CAS-WS_WR/CAS-WS_FS that is issued by the controller. Exists: (MEMC_PERF_LOG_ON) && (DDRCTL_LPDDR) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
perf_op_is_cas_ws_off	O	Asserts for every CAS-WS_OFF that is issued by the controller. Exists: (MEMC_PERF_LOG_ON) && (DDRCTL_LPDDR) && (DDRCTL_ENHANCED_WCK) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_cas_wck_sus	O	Asserts for every CAS-WCK_SUSPEND that is issued by the controller. Exists: (MEMC_PERF_LOG_ON) && (DDRCTL_LPDDR) && (DDRCTL_ENHANCED_WCK) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_enter_dsm	O	Asserts for every entry into Deep Sleep Mode. Exists: (MEMC_PERF_LOG_ON) && (DDRCTL_LPDDR) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_rfm	O	Asserts for every RFM that is issued by the controller. Exists: (MEMC_PERF_LOG_ON) && (DDRCTL_LPDDR) && (DDRCTL_HW_RFM_CTRL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_precharge	O	Asserts for every Precharge that is issued by the controller. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_precharge_for_rdwr	O	Asserts for every Precharge that is issued for Read or Write commands. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_precharge_for_other	O	Asserts for every Precharge that is issued due to requests other than Read or Write (for example: Refresh, ZQ Calib, MRW, MRR, tRAS(max)). Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
perf_rdwr_transitions	O	<p>Asserts for every Read to Write and Write to Read bus-turn-around that happens in the controller.</p> <p>Exists: MEMC_PERF_LOG_ON</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_write_combine	O	<p>Asserts for every Write Combine operation that happens in the controller.</p> <p>Exists: MEMC_PERF_LOG_ON</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_write_combine_noecc	O	<p>Asserts for every Write Combine (for Normal Write) operation that happens in the controller.</p> <p>Exists: (MEMC_PERF_LOG_ON) && (MEMC_INLINE_ECC)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_write_combine_wrecc	O	<p>Asserts for every Write Combine (for WRECC Overhead Combine) operation that happens in the controller.</p> <p>Exists: (MEMC_PERF_LOG_ON) && (MEMC_INLINE_ECC)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_war_hazard	O	<p>Asserts for every Write-after-Read collision that happens in the controller.</p> <p>Exists: MEMC_PERF_LOG_ON</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_raw_hazard	O	<p>Asserts for every Read-after-Write collision that happens in the controller.</p> <p>Exists: MEMC_PERF_LOG_ON</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
perf_waw_hazard	O	<p>Asserts for every Write-after-Write collision that happens in the controller. Valid only when Write Combine is turned off.</p> <p>Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
perf_ie_blk_hazard	O	<p>Asserts for every BLock Address collision (Inline ECC) that happens in the controller. When this is asserted, perf_raw/war/waw_hazard are don't care.</p> <p>Exists: (MEMC_PERF_LOG_ON) && (MEMC_INLINE_ECC) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
perf_op_is_enter_selfref[(MEMC_NUM_RANKS-1):0]	O	<p>Asserts for every entry into Self-Refresh mode.</p> <p>Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
perf_op_is_enter_powerdown[(MEMC_NUM_RANKS-1):0]	O	<p>Asserts for every entry into Power Down mode.</p> <p>Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
perf_selfref_mode[(MEMC_NUM_RANKS-1):0]	O	<p>Asserts for the entire duration for which the controller stays in Self-Refresh mode.</p> <p>Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
perf_op_is_refresh	O	<p>Asserts for every Refresh command that is issued by the controller after initialization is complete (that is, when STAT.operating_mode != 0).</p> <p>Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
perf_op_is_crit_ref	O	Asserts for every critical Refresh command that is issued by the controller. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_spec_ref	O	Asserts for every speculative Refresh command that is issued by the controller. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_load_mode	O	Asserts for every Load Mode operation (MRW or MRR) that is issued by the controller after initialization is complete (that is, when STAT.operating_mode != 0). Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_rank[(MEMC_RANK_BITS-1):0]	O	Gives the rank number for every scheduled command in DDRC through CAM path. Validated by the signals: perf_op_is_rd_or_wr or perf_op_is_rd or perf_op_is_wr. Exists: (MEMC_PERF_LOG_ON) && (MEMC_NUM_RANKS_GT_1) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_bank[(MEMC_BANK_BITS-1):0]	O	Gives the bank number for every scheduled command in DDRC through CAM path. Validated by the signals: perf_op_is_rd_or_wr or perf_op_is_rd or perf_op_is_wr. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_bg[(MEMC_BG_BITS-1):0]	O	Gives the bank group number for every scheduled command in DDRC through CAM path. Validated by the signals: perf_op_is_rd_or_wr or perf_op_is_rd or perf_op_is_wr. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
perf_cid[(CID_WIDTH-1):0]	O	<p>Gives the chip ID number for every scheduled command in DDRC through CAM path.</p> <p>Validated by the signals: perf_op_is_rd_or_wr or perf_op_is_rd or perf_op_is_wr.</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_CID_EN)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_hpr_req_with_nocredit	O	<p>Asserts when there is a High Priority Read (HPR) request (from XPI to PA) not served due to no available credit.</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_INCL_ARB)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_lpr_req_with_nocredit	O	<p>Asserts when there is a Low Priority Read (LPR) request (from XPI to PA) not served due to no available credit.</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_INCL_ARB)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_bsm_alloc	O	<p>Asserts for every BSM allocation that happens in the controller.</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DYN_BSM)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_bsm_starvation	O	<p>Asserts for every BSM starvation that happens in the controller.</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DYN_BSM)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_num_alloc_bsm[UMCTL2_BSM_BIT S:0]	O	<p>Gives the number of allocated BSMs.</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DYN_BSM)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
perf_visible_window_limit_reached_rd	O	Indicates that at least one RD CAM entry reaches visible window limit at this cycle. Exists: (MEMC_PERF_LOG_ON) && (MEMC_ENH_CAM_PTR) && (UMCTL2_VPRW_EN) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_visible_window_limit_reached_wr	O	Indicates that at least one WR CAM entry reaches visible window limit at this cycle. Exists: (MEMC_PERF_LOG_ON) && (MEMC_ENH_CAM_PTR) && (UMCTL2_VPRW_EN) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_dqsosc_mpc	O	Asserted for every DQSOSC MPC command issued by the controller. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_dqsosc_mrr	O	Asserted for every DQSOSC MRR command issued by the controller. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_tcr_mrr	O	Asserted for every TCR MRR command issued by the controller. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_zqstart	O	Asserts for every ZQcal start command that is issued by the controller after initialization is complete (that is, when STAT.operating_mode != 0). This signal is applicable for DDR5/LPDDR4. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
perf_op_is_zqlatch	O	Asserts for every ZQcal latch Short command that is issued by the controller after initialization is complete (that is, when STAT.operating_mode != 0). This signal is applicable for DDR5/LPDDR4/LPDDR5. Exists: MEMC_PERF_LOG_ON Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_hif_rd_or_wr_dch1	O	Asserts for every Read or Write command send to DDRC (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_hif_wr_dch1	O	Asserts for every Write command send to DDRC (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_hif_rd_dch1	O	Asserts for every Read command send to DDRC (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_hif_rmw_dch1	O	Asserts for every RMW command send to DDRC (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_hif_hi_pri_rd_dch1	O	Asserts for every High-Priority Read command send to DDRC (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_read_bypass_dch1	O	Asserts for every Read command that is send through the Bypass path (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (MEMC_RD_BYPASS) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
perf_act_bypass_dch1	O	Asserts for every Activate command that is send through the Bypass path (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (MEMC_ACT_BYPASS) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_dfi_wr_data_cycles_dch1	O	Asserts for every write data beat transfer on the DFI interface going to DRAM (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_dfi_rd_data_cycles_dch1	O	Asserts for every read data beat transfer on the DFI interface coming from DRAM (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_hpr_xact_when_critical_dch1	O	Asserts for every High Priority Read transaction that is scheduled when the High Priority Queue is in Critical state (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_lpr_xact_when_critical_dch1	O	Asserts for every Low Priority Read transaction that is scheduled when the Low Priority Queue is in Critical state (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_wr_xact_when_critical_dch1	O	Asserts for every Write transaction that is scheduled when the Write Queue is in Critical state (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
perf_op_is_activate_dch1	O	Asserts for every Activate that is issued for commands going through CAM (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_rd_or_wr_dch1	O	Asserts for every Read or Write that is issued for commands going through CAM (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_rd_activate_dch1	O	Asserts for every Read Activate that is issued for commands going through CAM (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_rd_dch1	O	Asserts for every Read that is issued for commands going through CAM (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_wr_dch1	O	Asserts for every Write that is issued for commands going through CAM (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_mwr_dch1	O	Asserts for every Masked Write that is issued for commands going through CAM (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
perf_op_is_cas_dch1	O	Asserts for every CAS command that is issued by the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (DDRCTL_LPDDR) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_cas_ws_dch1	O	Asserts for every CAS-WS_RD/CAS-WS_WR/CAS-WS_FS that is issued by the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (DDRCTL_LPDDR) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_cas_ws_off_dch1	O	Asserts for every CAS-WS_OFF that is issued by the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (DDRCTL_LPDDR) && (DDRCTL_ENHANCED_WCK) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_cas_wck_sus_dch1	O	Asserts for every CAS-WCK_SUSPEND that is issued by the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (DDRCTL_LPDDR) && (DDRCTL_ENHANCED_WCK) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_enter_dsm_dch1	O	Asserts for every entry into Deep Sleep Mode (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (DDRCTL_LPDDR) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_rfm_dch1	O	Asserts for every RFM that is issued by the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (DDRCTL_LPDDR) && (DDRCTL_HW_RFM_CTRL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
perf_op_is_precharge_dch1	O	Asserts for every Precharge that is issued by the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_precharge_for_rdwr_dch1	O	Asserts for every Precharge that is issued for Read or Write commands (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_precharge_for_other_dch1	O	Asserts for every Precharge that is issued due to requests other than Read or Write (for example, Refresh, ZQ Calib, MRW, MRR, tRAS(max)) (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_rdwr_transitions_dch1	O	Asserts for every Read to Write and Write to Read bus-turn-around that happens in the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_write_combine_dch1	O	Asserts for every Write Combine operation that happens in the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_write_combine_noecc_dch1	O	Asserts for every Write Combine (for Normal Write) operation that happens in the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (MEMC_INLINE_ECC) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
perf_write_combine_wrecc_dch1	O	Asserts for every Write Combine (for WRECC Overhead Combine) operation that happens in the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (MEMC_INLINE_ECC) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_war_hazard_dch1	O	Asserts for every Write-after-Read collision that happens in the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_raw_hazard_dch1	O	Asserts for every Read-after-Write collision that happens in the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_waw_hazard_dch1	O	Asserts for every Write-after-Write collision that happens in the controller (Channel 1). Is valid only when Write Combine is turned off. Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_ie_blk_hazard_dch1	O	Asserts for every BLock Address collision (Inline ECC) that happens in the controller (Channel 1). When this is asserted, perf_raw/war/waw_hazard are don't care. Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (MEMC_INLINE_ECC) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_enter_selfref_dch1[(MEMC_NUM_RANKS-1):0]	O	Asserts for every entry into Self-Refresh mode (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

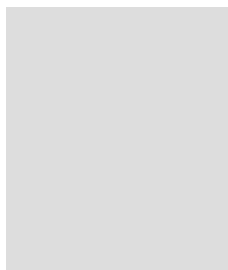
Port Name	I/O	Description
perf_op_is_enter_powerdown_dch1[(MEMC_NUM_RANKS-1):0]	O	Asserts for every entry into PowerDown mode (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_selfref_mode_dch1[(MEMC_NUM_RANKS-1):0]	O	Asserts for the entire duration for which the controller stays in Self-Refresh mode (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_refresh_dch1	O	Asserts for every Refresh command that is issued by the controller after initialization is complete (that is, when STAT.operating_mode != 0) (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_crit_ref_dch1	O	Asserts for every critical Refresh command that is issued by the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_spec_ref_dch1	O	Asserts for every speculative Refresh command that is issued by the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_op_is_load_mode_dch1	O	Asserts for every Load Mode operation (MRW or MRR) that is issued by the controller after initialization is complete (that is, when STAT.operating_mode != 0) (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
perf_rank_dch1[(MEMC_RANK_BITS-1):0]	O	<p>Gives the rank number for every scheduled command in DDRC through CAM path (Channel 1). Validated by the signals: perf_op_is_rd_or_wr_dch1 or perf_op_is_rd_dch1 or perf_op_is_wr_dch1.</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (MEMC_NUM_RANKS_GT_1)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_bank_dch1[(MEMC_BANK_BITS-1):0]	O	<p>Gives the bank number for every scheduled command in DDRC through CAM path (Channel 1). Validated by the signals: perf_op_is_rd_or_wr_dch1 or perf_op_is_rd_dch1 or perf_op_is_wr_dch1.</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_bg_dch1[(MEMC_BG_BITS-1):0]	O	<p>Gives the bank group number for every scheduled command in DDRC through CAM path (Channel 1). Validated by the signals: perf_op_is_rd_or_wr_dch1 or perf_op_is_rd_dch1 or perf_op_is_wr_dch1.</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_cid_dch1[(CID_WIDTH-1):0]	O	<p>Gives the chip ID number for every scheduled command in DDRC through CAM path (Channel 1). Validated by the signals: perf_op_is_rd_or_wr_dch1 or perf_op_is_rd_dch1 or perf_op_is_wr_dch1.</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_CID_EN)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_hpr_req_with_nocredit_dch1	O	<p>Asserts when there is a High Priority Read (HPR) request (from XPI to PA) not served due to no available credit (Channel 1).</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_INCL_ARB)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
perf_lpr_req_with_nocredit_dch1	O	Asserts when there is a Low Priority Read (LPR) request (from XPI to PA) not served due to no available credit (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_INCL_ARB) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_bsm_alloc_dch1	O	Asserts for every BSM allocation that happens in the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DYN_BSM) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_bsm_starvation_dch1	O	Asserts for every BSM starvation that happens in the controller (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DYN_BSM) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_num_alloc_bsm_dch1[UMCTL2_BSM_BITS:0]	O	Gives the number of allocated BSMs (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (UMCTL2_DYN_BSM) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_visible_window_limit_reached_rd_dch1	O	Indicates that at least one RD CAM entry reaches visible window limit at this cycle (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (MEMC_ENH_CAM_PTR) && (UMCTL2_VPRW_EN) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
perf_visible_window_limit_reached_wr_dch1	O	Indicates that at least one WR CAM entry reaches visible window limit at this cycle (Channel 1). Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL) && (MEMC_ENH_CAM_PTR) && (UMCTL2_VPRW_EN) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
perf_op_is_dqsosc_mpc_dch1	O	<p>Asserted for every DQSOSC MPC command issued by the controller (Channel 1).</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_op_is_dqsosc_mrr_dch1	O	<p>Asserted for every DQSOSC MRR command issued by the controller (Channel 1).</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_op_is_tcr_mrr_dch1	O	<p>Asserted for every TCR MRR command issued by the controller (Channel 1).</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_op_is_zqstart_dch1	O	<p>Asserts for every ZQcal start command that is issued by the controller after initialization is complete (that is, when STAT.operating_mode != 0) (Channel 1). This signal is applicable for DDR5/LPDDR4.</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
perf_op_is_zqlatch_dch1	O	<p>Asserts for every ZQcal latch Short command that is issued by the controller after initialization is complete (that is, when STAT.operating_mode != 0) (Channel 1). This signal is applicable for DDR5/LPDDR4/LPDDR5.</p> <p>Exists: (MEMC_PERF_LOG_ON) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.30 Credit Counters Signals



- lpr_credit_cnt
- hpr_credit_cnt
- wr_credit_cnt
- wrecc_credit_cnt
- lpr_credit_cnt_dch1
- hpr_credit_cnt_dch1
- wr_credit_cnt_dch1
- wrecc_credit_cnt_dch1

Table 15-30 Credit Counters Signals

Port Name	I/O	Description
lpr_credit_cnt[6:0]	O	<p>Indicates the number of available Low priority read CAM slots (free positions). Each slots holds a DRAM burst. Value is decremented/incremented as the commands flow in out of the read CAM (LPR store). Exists: UMCTL2_INCL_ARB Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
hpr_credit_cnt[6:0]	O	<p>Indicates the number of available High priority read CAM slots (free positions). Each slots holds a DRAM burst. Value is decremented/incremented as the commands flow in out of the read CAM (HPR store). Exists: UMCTL2_INCL_ARB Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
wr_credit_cnt[6:0]	O	<p>Indicates the number of available write CAM slots (free positions). Each slots holds a DRAM burst. Value is decremented/incremented as the commands flow in out of the write CAM. Exists: UMCTL2_INCL_ARB Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
wrecc_credit_cnt[6:0]	O	<p>Indicates the number of available write ECC CAM slots (free positions). Each slots holds a DRAM burst. Value is decremented/incremented as the commands flow in out of the write ECC CAM.</p> <p>Exists: (UMCTL2_INCL_ARB) && (MEMC_INLINE_ECC)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
lpr_credit_cnt_dch1[6:0]	O	<p>Indicates the number of available Low priority read CAM slots (free positions). Each slots holds a DRAM burst (Channel 1). Value is decremented/incremented as the commands flow in out of the read CAM (LPR store).</p> <p>Exists: (UMCTL2_INCL_ARB) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hpr_credit_cnt_dch1[6:0]	O	<p>Indicates the number of available High priority read CAM slots (free positions). Each slots holds a DRAM burst (Channel 1). Value is decremented/incremented as the commands flow in out of the read CAM (HPR store).</p> <p>Exists: (UMCTL2_INCL_ARB) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
wr_credit_cnt_dch1[6:0]	O	<p>Indicates the number of available write CAM slots (free positions). Each slots holds a DRAM burst (Channel 1). Value is decremented/incremented as the commands flow in out of the write CAM.</p> <p>Exists: (UMCTL2_INCL_ARB) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
wrecc_credit_cnt_dch1[6:0]	O	<p>Indicates the number of available write ECC CAM slots (free positions). Each slots holds a DRAM burst (Channel 1). Value is decremented/incremented as the commands flow in out of the write ECC CAM.</p> <p>Exists: (UMCTL2_INCL_ARB) && (UMCTL2_DUAL_CHANNEL) && (MEMC_INLINE_ECC)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.31 Port Arbiter Signals



Table 15-31 Port Arbiter Signals

Port Name	I/O	Description
pa_rmask[((2*NPORTS)-1):0]	I	<p>When asserted (active high), this signal masks (prevents) the corresponding application port read address channel from requesting to the PA.</p> <p>An external agent can control the port arbitration by throttling certain XPI ports based on traffic class, queue status and other dynamic criteria.</p> <p>There are 2 bits for each port, first one for the blue queue, second for the red queue (Bit 0 drives blue queue of Port 0, Bit 1 drives red queue of Port 0, Bit 2 drives blue queue of Port 1 and so on.).</p> <p>If dual queue is not used for a specific port, red input is unused.</p> <p>If not used, tie-off all bits to 0.</p> <p>Exists: UMCTL2_INCL_ARB</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
pa_wmask[(NPORTS-1):0]	I	<p>When asserted (active high), this signal masks (prevents) the corresponding application port write address channel from requesting to the PA.</p> <p>An external agent can control the port arbitration by throttling certain XPI ports based on traffic class, queue status and other dynamic criteria.</p> <p>Each bit position corresponds to a port index (Bit 0 drives Port 0 and so on).</p> <p>If not used, tie-off all bits to 0.</p> <p>Exists: UMCTL2_INCL_ARB</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.32 ECC Scrubber Signals

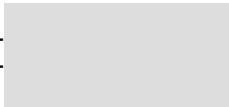
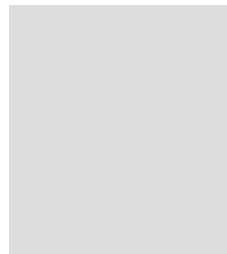
sbr_clk -  - sbr_done_intr
 sbr_resetrn -  - sbr_done_intr_dch1

Table 15-32 ECC Scrubber Signals

Port Name	I/O	Description
sbr_clk	I	<p>Scrubber Clock. Same clock (synchronous to) as core_ddrc_core_clk. The SBR can continue to function that is, count and request exit from low power - even if the core controller clock is gated.</p> <p>Exists: UMCTL2_SBR_EN_1</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
sbr_resetrn	I	<p>Scrubber Reset. Same as core_ddrc_rstn. Active-low pin that asynchronously resets the SBR logic to its default state. Synchronous to sbr_clk on de-assertion, asynchronous on assertion.</p> <p>Exists: UMCTL2_SBR_EN_1</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: N/A</p> <p>Power Domain: DDRCTL_DOMAIN</p>
sbr_done_intr	O	<p>Scrubber interrupt indicating one full address range sweep; only asserted in certain conditions. Behaves identical to SBRSTAT.scrub_done register.</p> <p>Exists: UMCTL2_SBR_EN_1</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
sbr_done_intr_dch1	O	<p>Scrubber interrupt indicating one full address range sweep. Only asserted in certain conditions. Behaves identical to SBRSTAT.scrub_done register (Channel 1).</p> <p>Exists: (UMCTL2_SBR_EN_1) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.33 DFI Command Interface Signals



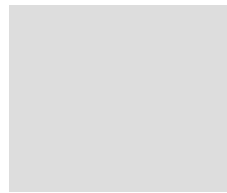
- [dfin_address_Pp](#) (for $n = 0; n \leq 1$)(for $p = 0; p \leq 3$)
- [dfin_cke_Pp](#) (for $n = 0; n \leq 1$)(for $p = 0; p \leq 3$)
- [dfin_cs_Pp](#) (for $n = 0; n \leq 1$)(for $p = 0; p \leq 3$)
- [dfin_dram_clk_disable_Pp](#) (for $n = 0; n \leq 1$)(for $p = 0; p \leq 3$)
- [dfi_reset_n](#)

Table 15-33 DFI Command Interface Signals

Port Name	I/O	Description
dfin_address_Pp [(MEMC_DFI_ADDR_WIDTH-1):0] (for $n = 0; n \leq 1$)(for $p = 0; p \leq 3$)	O	<p>This signal is the controller-to-PHY address on phase p for DFIn channel. This signal is 14 bits for $p=0$. 6 bits for $p=1,2,3$. The CA bus for DFIn channel is mapped to dfin_address.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfin_cke_Pp [(DDRCTL_DFI0_CS_WIDTH-1):0] (for $n = 0; n \leq 1$)(for $p = 0; p \leq 3$)	O	<p>This signal is the controller-to-PHY clock enable on phase p for the DFIn channel. It is active high.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfin_cs_Pp [(DDRCTL_DFI0_CS_WIDTH-1):0] (for $n = 0; n \leq 1$)(for $p = 0; p \leq 3$)	O	<p>This signal is the controller-to-PHY chip select on phase p for DFIn channel. This signal is active high or low, depending on the SDRAM device in use. When LPDDR4/LPDDR5 SDRAM device is used, this signal is active high. When other SDRAM device is used, this signal is active low.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
dfin_dram_clk_disable_Pp[(MEMC_NUM_CLKS-1):0] (for n = 0; n <= 1)(for p = 0; p <= 3)	O	<p>This signal is the SDRAM clock disable signal on phase <i>p</i> for the DFI_n channel.</p> <p>When active, this indicates to the PHY that the clocks to the SDRAM devices must be disabled such that the clock signals hold a constant value. When the dfin_dram_clk_disable signal is inactive, the SDRAMs must be clocked normally.</p> <p>One bit per rank.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfi_reset_n[(UMCTL2_RESET_WIDTH-1):0]	O	<p>This signal is the controller-to-PHY reset (active low) for memory. This signal is always reset to its active value (LOW).</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.34 DFI Write Data Interface (for $n = 0; n \leq 1$)(for $p = 0; p \leq 3$) Signals



- dfin_wrdata_Pp
- dfin_wrdata_en_Pp
- dfin_wrdata_mask_Pp
- dfin_wrdata_cs_Pp
- dfin_wrdata_link_ecc_Pp

Table 15-34 DFI Write Data Interface (for $n = 0; n \leq 1$)(for $p = 0; p \leq 3$) Signals

Port Name	I/O	Description
dfin_wrdata_Pp[(DDRCTL_DFI_DATA_WIDTH-1):0]	O	<p>This signal is the write data on phase p for the DFIn channel. The write data stream is valid for the number of cycles that the dfin_wrdata_en_Pp signal is asserted. The latency between dfin_wrdata_en_Pp and dfin_wrdata_Pp is defined by DFITMG0.dfi_tphy_wrdata.</p> <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfin_wrdata_en_Pp[(DDRCTL_DFI_DATA_EN_WIDTH-1):0]	O	<p>This signal is the write data and data mask valid signal on phase p for the DFIn channel. This signal indicates the number of cycles of data and data mask to be sent on the DFI interface. The latency between the write command and dfin_wrdata_en_Pp is defined by DFITMG0.dfi_tphy_wrlat.</p> <p>When the dfin_wrdata_en_Pp signal is asserted, it remains asserted for the number of contiguous cycles of write data passed through the DFI write data interface.</p> <p>There is a single dfin_wrdata_en bit for each slice of memory data. The dfin_wrdata_en_Pp[0] signal corresponds with the lowest segment of dfin_wrdata_Pp signals.</p> <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
dfin_wrddata_mask_Pp[(DDRCTL_DFI_MASK_WIDTH-1):0]	O	<p>This signal is used to transfer either the write data mask or the write data inversion (DBI) information on phase <i>p</i> for the DFI_n channel, depending on the system and DRAM settings.</p> <p>When the DM feature is enabled:</p> <ul style="list-style-type: none"> Write data byte mask signal. The timing is the same as for the dfin_wrddata_Pp bus. The dfin_wrddata_mask_Pp[0] signal defines masking for dfin_wrddata_Pp[7:0]; dfin_wrddata_mask[1] defines masking for dfin_wrddata_Pp[15:8]; and so on. This signal is active high or low, depending on the SDRAM device in use. <p>When the Write DBI feature is enabled:</p> <ul style="list-style-type: none"> Write DBI signal. The timing is the same as for the dfi0_wrddata_P0 bus. The dfi0_wrddata_mask_P0[0] signal defines the DBI information for dfi0_wrddata_P0[7:0]; dfi0_wrddata_mask_P0[1] defines DBI information for dfi0_wrddata_P0[15:8]; and so on. This signal is active high or low, depending on SDRAM device in use. When DDR4/DDR5 SDRAM device is used, this signal is active low. When LPDDR4/LPDDR5 SDRAM device is used, this signal is active high. <p>When the DM and Write DBI features are enabled:</p> <ul style="list-style-type: none"> If the total count of '1' data bits on dfi0_wrddata_P0[2:7] is equal to or greater than five and dfi0_wrddata_mask_P0[0] is low, write data is masked. If the count of '1' is less than five, dfi0_wrddata_mask_P0[0] defines the DBI information for dfi0_wrddata_P0[7:0], and so on. <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfin_wrddata_cs_Pp[((DDRCTL_DFI_DATA_EN_WIDTH*DDRCTL_DFI0_CS_WIDTH)-1):0]	O	<p>This signal is the DFI write data chip select on phase <i>p</i> for the DFI_n channel. The dfin_wrddata_cs_Pp signal is asserted DFITMG2.dfi_tphy_wrslat cycles after the assertion of a write command on the DFI control interface. The polarity of this signal is set by DFIMISC.dfi_data_cs_polarity.</p> <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfin_wrddata_link_ecc_Pp[((DDRCTL_DFI_DATA_WIDTH/8)-1):0]	O	<p>This signal is used to transfer the write Link-ECC code, depending on the DRAM setting. This signal is valid for the number of cycles that the dfi_wrddata_en signal is asserted. The latency between dfi_wrddata_en and this signal is defined by DFITMG0.dfi_tphy_wrddata.</p> <p>Exists: MEMC_LINK_ECC</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.35 DFI Read Data Interface (for $n = 0; n \leq 1$)(for $p = 0; p \leq 3$) Signals

dfin_rddata_Wp - dfin_rddata_cs_Pp
 $\text{dfin_rddata_dbi_Wp}$ - dfin_rddata_en_Pp
 $\text{dfin_rddata_valid_Wp}$ -

Table 15-35 DFI Read Data Interface (for $n = 0; n \leq 1$)(for $p = 0; p \leq 3$) Signals

Port Name	I/O	Description
$\text{dfin_rddata_Wp}[(\text{DDRCTL_DFI_DATA_WIDTH}-1):0]$	I	<p>This signal is the read data on phase p from the DFIn channel.</p> <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
$\text{dfin_rddata_cs_Pp}[(\text{DDRCTL_DFI_DATA_EN_WIDTH} * \text{DDRCTL_DFI0_CS_WIDTH}) - 1):0]$	O	<p>This signal is the DFI read data chip select on phase p for the DFIn channel. The dfin_rddata_cs_Pp signal is asserted DFITMG2.dfi_tphy_rdcslat cycles after the assertion of a read command on the DFI control interface. The polarity of this signal is set by DFIMISC.dfi_data_cs_polarity.</p> <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
$\text{dfin_rddata_dbi_Wp}[(\text{DDRCTL_DFI_DATA_A_WIDTH}/8 - 1):0]$	I	<p>This signal is sent with DFI read data (dfin_rddata_Wp) on phase p from the DFIn channel, indicating data bus inversion functionality. It is valid with $\text{dfin_rddata_valid_Pp}$.</p> <p>There is 1 bit of this signal for every 8 bits of the read data bus. This signal is active high or low, depending on SDRAM device in use. When a LPDDR4/LPDDR5 SDRAM device is used, this signal is active high.</p> <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
$\text{dfin_rddata_en_Pp}[(\text{DDRCTL_DFI_DATA_EN_WIDTH}-1):0]$	O	<p>This signal is the read data enable on phase p for the DFIn channel. The dfin_rddata_en_Pp signal is asserted DFITMG.dfi_t_rddata_en cycles after the assertion of a read command on the DFI control interface and remains valid for the duration of read data expected on the dfin_rddata_Pp bus.</p> <p>There is a single dfin_rddata_en_Pp bit for each slice of memory data. The dfin_rddata_en_Pp [0] signal corresponds with the lowest segment of dfin_rddata_Pp signals.</p> <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
dfin_rddata_valid_Wp[(DDRCTL_DFI_DATAEN_WIDTH-1):0]	I	<p>This signal is the read data valid indicator on phase <i>p</i> from the DFI_n channel. The dfin_rddata_valid_Wp signal is asserted with the read data for the number of cycles that data is being sent. The timing is the same as dfin_rddata_Wp bus.</p> <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.36 DFI Update Interface (for n = 0; n <= 1) Signals

dfin_ctrlupd_ack -
 dfin_phyupd_req -
 dfin_phyupd_type -

Table 15-36 DFI Update Interface (for n = 0; n <= 1) Signals

Port Name	I/O	Description
dfin_ctrlupd_req	O	<p>This signal is used to trigger a controller-initiated update for the DFI_n channel. This indicates that the DFI is idle for some time, during which the PHY may perform an update. The dfin_ctrlupd_req signal is asserted for a minimum of DFIUPDTMG0.dfi_t_ctrlup_min cycles and a maximum of DFIUPDTMG0.dfi_t_ctrlup_max cycles.</p> <p>A dfin_ctrlupd_req signal assertion is an invitation for the PHY to update and does not require a response. The behavior of the dfin_ctrlupd_req signal is dependent on the dfin_ctrlupd_ack signal:</p> <ul style="list-style-type: none"> ■ If the update is acknowledged by the PHY, the dfin_ctrlupd_req signal remains asserted as long as the dfin_ctrlupd_ack signal is asserted, but is de-asserted before DFIUPDTMG0.dfi_t_ctrlup_max expires. While this signal is asserted, the DFI bus remains idle other than any transactions specifically associated with the update process. ■ If the update is not acknowledged, the dfin_ctrlupd_req signal is de-asserted at any time after DFIUPDTMG0.dfi_t_ctrlup_min and before DFIUPDTMG0.dfi_t_ctrlup_max. <p>The maximum number of clock cycles that the controller may wait between assertions of the dfin_ctrlupd_req is determined by DFIUPD1.dfi_ctrlupd_interval_max_x1024.</p> <p>Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
dfin_ctrlupd_ack	I	<p>This signal is asserted to acknowledge a controller-initiated update request from the DFI₀ channel. The PHY is not required to acknowledge this request.</p> <p>While this signal is asserted, the DFI bus remains idle other than any transactions specifically associated with the update process.</p> <ul style="list-style-type: none"> ■ If the PHY acknowledges the request, the dfin_ctrlupd_ack signal must be asserted before the dfin_ctrlupd_req signal de-asserts. ■ If PHY ignores the request, the dfin_ctrlupd_ack signal must remain de-asserted until the dfin_ctrlupd_req signal is de-asserted. <p>The dfin_ctrlupd_req signal is guaranteed to be asserted for at least DFIUPDTMG0.dfi_t_ctrlup_min cycles.</p> <p>Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
dfin_ctrlupd_type[1:0]	O	<p>This signal is the DFI controller-initiated update select from the DFI channel. This signal indicates which one of the four types of DFI MC-Initiated Update is being requested by the dfin_ctrlupd_req signal. The valid values are as follows:</p> <ul style="list-style-type: none"> ■ 00: ctrlupd_type0 ■ 01: ctrlupd_type1 ■ 10: ctrlupd_type2 (not used) ■ 11: ctrlupd_type3 (not used) <p>This signal is not yet in DFI standard. Possibly renamed later.</p> <p>Exists: DDRCTL_PPT2</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfin_phyupd_req	I	<p>This signal is the DFI PHY-initiated update request from the DFI channel.</p> <p>If set, this signal indicates that the PHY requires the DFI to be idle; that is, for the DFI command, read data channel and write data channel are inactive for a specified period of time.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfin_phyupd_type[1:0]	I	<p>This signal is the DFI PHY-initiated update select from the DFI channel. This signal indicates which one of the four types of PHY update times is being requested by the dfin_phyupd_req signal. The valid values are as follows:</p> <ul style="list-style-type: none"> ■ 00: Tphyupd_type0 ■ 01: Tphyupd_type1 (not used) ■ 10: Tphyupd_type2 (not used) ■ 11: Tphyupd_type3 (not used) <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfin_phyupd_ack	O	<p>This signal is the DFI PHY-initiated update acknowledge for the DFI channel: This signal indicates that the DFI is idle and will remain so until the dfin_phyupd_req signal de-asserts.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.37 DFI Status Interface (for n = 0; n <= 1) Signals



Table 15-37 DFI Status Interface (for n = 0; n <= 1) Signals

Port Name	I/O	Description
dfin_freq_ratio[1:0]	O	<p>This signal indicates the frequency ratio for the DFI channel.</p> <ul style="list-style-type: none"> 01: 1:2 Frequency Ratio 10: 1:4 Frequency Ratio 00/11: Reserved <p>Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
dfin_init_complete	I	<p>The signal is the PHY initialization complete for the DFI channel. The dfin_init_complete signal indicates that the PHY is able to respond to any proper stimulus on the DFI. All DFI signals are held at their default values until the dfin_init_complete signal asserts. This signal will be ignored by the controller if DFIMISC.dfi_init_complete_en is set to 0. This signal is also used as part of the DFI frequency change protocol.</p> <p>Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
dfin_init_start	O	<p>The signal is the PHY initialization start and DFI frequency change request for DFI channel from the DFIMISC.dfi_init_start APB register. When asserted, this signal triggers the PHY initialization start and DFI frequency change request and then the DFISTAT.dfi_init_complete flag is polled to know when the initialization and frequency change is done.</p> <p>Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
dfin_frequency[4:0]	O	<p>This signal indicates the operating frequency of the system for the DFIn channel from the DFIMISC.dfi_frequency APB register. The number of supported frequencies and the mapping of signal values to clock frequencies are defined by the PHY.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfin_freq_fsp[1:0]	O	<p>This signal indicates the FSP-OP for the DFIn channel from the DFIMISC.dfi_freq_fsp APB register.</p> <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.38 DFI PHY Master Interface (for n = 0; n <= 1) Signals

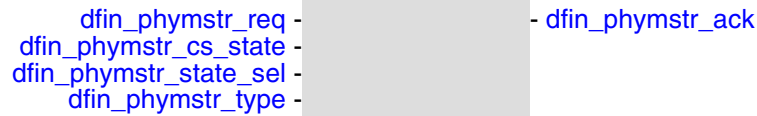


Table 15-38 DFI PHY Master Interface (for n = 0; n <= 1) Signals

Port Name	I/O	Description
dfm_phymstr_req	I	<p>This signal is the DFI PHY master request for the DFI channel. If set, this signal indicates that the PHY requests control on the DFI bus.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfm_phymstr_cs_state[(MEMC_NUM_RANKS-1):0]	I	<p>This signal is the DFI PHY master CS state for the DFI channel. This signal indicates the state of the DRAM when the PHY becomes the master:</p> <ul style="list-style-type: none"> 0: The PHY specifies the required state, using the dfm_phymstr_state_sel signal. 1: The PHY does not specify the state. <p>This signal is valid only when dfm_phymstr_req is asserted. Each memory rank uses one bit. Note: This input port is not used internally as the DDRCTL puts all ranks of SDRAM into Self Refresh in response to dfm_phymstr_req.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfm_phymstr_state_sel	I	<p>This signal is the DFI PHY master state select for the DFI channel. This signal indicates the state requested by the PHY:</p> <ul style="list-style-type: none"> 0: IDLE 1: Self-Refresh <p>Note: This input port is not used internally as the DDRCTL puts all ranks of SDRAM into Self Refresh in response to dfm_phymstr_req.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
dfin_phymstr_type[1:0]	I	<p>This signal is the DFI PHY master type for the DFI channel. This signal indicates which of the four types of PHY master interface times the dfin_phymstr_req signal is requesting:</p> <ul style="list-style-type: none"> ■ 00: tphymstr_type0 ■ 01: tphymstr_type1 ■ 10: tphymstr_type2 ■ 11: tphymstr_type3 <p>Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
dfin_phymstr_ack	O	<p>This signal is the DFI PHY master acknowledge for the DFI channel: When asserted, the PHY is the master of the DRAM bus.</p> <p>Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>

15.39 DFI Low Power Interface (for $n = 0$; $n \leq 1$) Signals

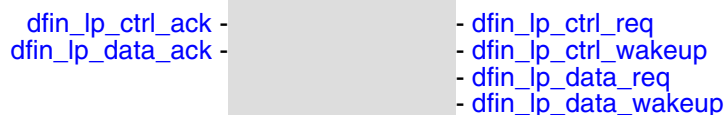


Table 15-39 DFI Low Power Interface (for $n = 0$; $n \leq 1$) Signals

Port Name	I/O	Description
dfm_lp_ctrl_req	O	<p>This signal is the DFI low-power control request for the DFI_n channel: This controller uses this signal to inform the PHY of an opportunity to switch to a low-power mode.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfm_lp_ctrl_ack	I	<p>This signal is the DFI low-power control acknowledge for the DFI_n channel: This PHY asserts this signal to acknowledge the controller low-power opportunity request. The PHY is not required to acknowledge the request.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
dfin_lp_ctrl_wakeup[(DFI_LP_WAKEUP_P D_WIDTH-1):0]	O	<p>This signal is the DFI low-power control wake-up for the DFI channel: This signal indicates which one of the 16 wake-up times the controller is requesting the PHY. The valid values for the PHY and the PHY respective low-power action are as follows:</p> <ul style="list-style-type: none"> ■ 00000: 1 cycle ■ 00001: 2 cycles ■ 00010: 4 cycles ■ 00011: 8 cycles ■ 00100: 16 cycles ■ 00101: 32 cycles ■ 00110: 64 cycles ■ 00111: 128 cycles ■ 01000: 256 cycles ■ 01001: 512 cycles ■ 01010: 1024 cycles ■ 01011: 2048 cycles ■ 01100: 4096 cycles ■ 01101: 8192 cycles ■ 01110: 16384 cycles ■ 01111: 32768 cycles ■ 10000: 65536 cycles ■ 10001: 131072 cycles ■ 10010: 262144 cycles ■ 10011: Unlimited <p>Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
dfin_lp_data_req	O	<p>This signal is the DFI low-power data request the for DFI channel: This controller uses this signal to inform the PHY of an opportunity to switch to a low-power mode.</p> <p>Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
dfin_lp_data_ack	I	<p>This signal is the DFI Low Power Data Acknowledge for DFI channel: DFI low power data acknowledge. This signal is asserted by the PHY to acknowledge the Controller controller low power opportunity request. The PHY is not required to acknowledge the request</p> <p>Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
dfin_lp_data_wakeup[(DFI_LP_WAKEUP_PD_WIDTH-1):0]	O	<p>This signal is the DFI low-power data wake-up for the DFI_n channel. This signal indicates which one of the 16 wake-up times the controller is requesting for the PHY. The valid values for the PHY and the PHY respective low-power action are as follows:</p> <ul style="list-style-type: none"> ■ 00000: 1 cycle ■ 00001: 2 cycles ■ 00010: 4 cycles ■ 00011: 8 cycles ■ 00100: 16 cycles ■ 00101: 32 cycles ■ 00110: 64 cycles ■ 00111: 128 cycles ■ 01000: 256 cycles ■ 01001: 512 cycles ■ 01010: 1024 cycles ■ 01011: 2048 cycles ■ 01100: 4096 cycles ■ 01101: 8192 cycles ■ 01110: 16384 cycles ■ 01111: 32768 cycles ■ 10000: 65536 cycles ■ 10001: 131072 cycles ■ 10010: 262144 cycles ■ 10011: Unlimited <p>Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>

15.40 DFI MC to PHY Message Interface (for n = 0; n <= 1) Signals

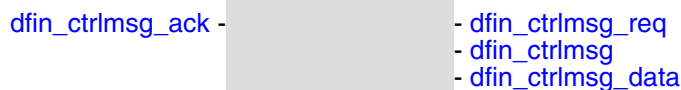


Table 15-40 DFI MC to PHY Message Interface (for n = 0; n <= 1) Signals

Port Name	I/O	Description
dfi_ctrlmsg_req	O	<p>This signal is the DFI controller message request for the DFI channel. When asserted, this signal indicates a valid MC-to-PHY message. If acknowledged, the request must remain asserted until the dfi_ctrlmsg_ack signal. If not acknowledged within dfi_t_ctrlmsg_resp cycles, the request is de-asserted.</p> <p>Exists: DDRCTL_DFI_CTRLMSG</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfi_ctrlmsg[7:0]	O	<p>This signal is the DFI controller message command for the DFI channel. It is valid only when the dfi_ctrlmsg_req signal is asserted. This signal encodes messages from the MC to the PHY.</p> <p>Exists: DDRCTL_DFI_CTRLMSG</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfi_ctrlmsg_data[15:0]	O	<p>This signal is the DFI Controller message data for the DFI channel. It is valid only when the dfi_ctrlmsg_req signal is asserted. Data associated with the information command transfers from the MC to the PHY.</p> <p>Exists: DDRCTL_DFI_CTRLMSG</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfi_ctrlmsg_ack	I	<p>This signal is the DFI Controller message acknowledge for the DFI channel: When asserted, this signal indicates that the PHY received the MC message. If the message is to be acknowledged, the dfi_ctrlmsg_ack signal must assert within dfi_t_ctrlmsg_resp clock cycles. Once asserted, the dfi_ctrlmsg_ack signal must de-assert within dfi_t_ctrlmsg_max clock cycles.</p> <p>Exists: DDRCTL_DFI_CTRLMSG</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.41 DFI WCK Control Interface (for $n = 0; n \leq 1$)(for $p = 0; p \leq 3$) Signals

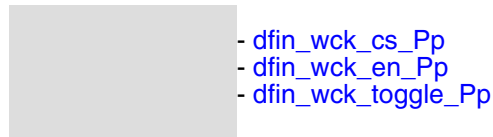


Table 15-41 DFI WCK Control Interface (for $n = 0; n \leq 1$)(for $p = 0; p \leq 3$) Signals

Port Name	I/O	Description
dfm_wck_cs_Pp[(MEMC_NUM_RANKS-1):0]	O	<p>This signal is the WCK chip select on phase p for the DFIn channel. This signal indicates which chip selects currently have the WCK active. More than one chip select can be active at a time. There is one bit per chip select. This signal is only valid when the dfm_wck_en_Pp signal is asserted.</p> <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfm_wck_en_Pp[((DDRCTL_DFI_DATA_WIDTH/16)-1):0]	O	<p>This signal is the WCK clock enable on phase p for the DFIn channel. This signal defines when the WCK clock is driven or disabled (tri-state).</p> <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfm_wck_toggle_Pp[1:0]	O	<p>This signal is the WCK toggle on phase p for the DFIn channel. This is a 2-bit encoded value defining the state of the WCK clock. This signal is only valid when the dfm_wck_en_Pp signal is asserted.</p> <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.42 Non-DFI DDRCTL PHY Sideband Interface (for n = 0; n <= 1)(for p = 0; p <= 3) Signals

 - `dwc_ddrphyn_snoop_en_Pp`

Table 15-42 Non-DFI DDRCTL PHY Sideband Interface (for n = 0; n <= 1)(for p = 0; p <= 3) Signals

Port Name	I/O	Description
<code>dwc_ddrphyn_snoop_en_Pp[((DDRCTL_DFI_DATAEN_WIDTH*4)-1):0]</code>	O	<p>This signal is used to indicate when the PHY snoops the MR read data on phase <i>p</i> for the DFI_n channel. The <code>dwc_ddrphyn_snoop_en_Pp</code> is 4-bit wide for each bit of <code>dfin_rddata_en_Pp</code> and follows the timing of the corresponding <code>dfin_rddata_en_Pp</code>. Each bit in every 4-bit of <code>dwc_ddrphyn_snoop_en_Pp</code> signals indicates the MR register to be snooped as follows:</p> <ul style="list-style-type: none"> ■ <code>dwc_ddrphyn_snoop_en_Pp[4*i]</code>: Asserted when MR18(LPDDR4) or MR35(LPDDR5) is read by controller ■ <code>dwc_ddrphyn_snoop_en_Pp[4*i+1]</code>: Asserted when MR19(LPDDR4) or MR36(LPDDR5) is read by controller ■ <code>dwc_ddrphyn_snoop_en_Pp[4*i+2]</code>: Asserted when MR38(LPDDR5) is read by controller ■ <code>dwc_ddrphyn_snoop_en_Pp[4*i+3]</code>: Asserted when MR39(LPDDR5) is read by controller <p>Exists: DDRCTL_LPDDR</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.43 LPDDR4 Initialization Handshake Interface Signals

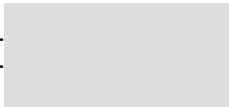
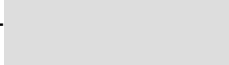
dfi_reset_n_in -  - dfi_reset_n_ref
 init_mr_done_in -  - init_mr_done_out

Table 15-43 LPDDR4 Initialization Handshake Interface Signals

Port Name	I/O	Description
dfi_reset_n_in	I	<p>This signal is the DFI reset reference signal and is active low. Connect this signal to the dfi_reset_n_ref port of the other controller. If not used, tie the signal to 1. This signal is supported in LPDDR4 mode only.</p> <p>Exists: (DDRCTL_LPDDR) && (!UMCTL2_LPDDR4_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
dfi_reset_n_ref	O	<p>This signal is the DFI reset reference signal and is active low. Connect this signal to the dfi_reset_n_in port of the other controller. This signal is supported in LPDDR4 mode only.</p> <p>Exists: (DDRCTL_LPDDR) && (!UMCTL2_LPDDR4_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
init_mr_done_in	I	<p>This signal indicates the MRW section of the initialization is done. It is active high. Connect this signal to the init_mr_done_out port of the other controller. If not used, tie the signal to 1. This signal is supported in LPDDR4 mode only.</p> <p>Exists: (DDRCTL_LPDDR) && (!UMCTL2_LPDDR4_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
init_mr_done_out	O	<p>This signal indicates the MRW section of the initialization is done. It is active high. Connect this signal to the init_mr_done_in port of the other controller. This signal is supported in LPDDR4 mode only.</p> <p>Exists: (DDRCTL_LPDDR) && (!UMCTL2_LPDDR4_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.44 Register Visibility Control Signals

dis_regs_ecc_syndrome - 

Table 15-44 Register Visibility Control Signals

Port Name	I/O	Description
dis_regs_ecc_syndrome	I	<p>Signal used to hide the value of ECCCSYN* and ECCUSYN* registers, e.g. for security purposes. When this value is set to 1, reading registers ECCCSYN*/ECCUSYN* returns value 0 always, otherwise it returns appropriate value. If this feature is not used, this port can be tied to 0. The value of dis_regs_ecc_syndrome signal cannot change outside of reset (presetn=0 && core_ddrc_core_rstn=0).</p> <p>Exists: MEMC_ECC</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.45 Interrupts Signals

- ecc_corrected_err_intr
- ecc_corrected_err_intr_fault
- ecc_uncorrected_err_intr
- ecc_uncorrected_err_intr_fault
- ecc_ap_err_intr
- ecc_ap_err_intr_fault
- capar_retry_limit_reached_intr_fault
- rd_linkecc_uncorr_err_intr
- rd_linkecc_uncorr_err_intr_fault
- rd_linkecc_corr_err_intr
- rd_linkecc_corr_err_intr_fault
- capar_retry_limit_reached_intr_dch1_fault
- ecc_corrected_err_intr_dch1
- ecc_corrected_err_intr_dch1_fault
- ecc_uncorrected_err_intr_dch1
- ecc_uncorrected_err_intr_dch1_fault
- ecc_ap_err_intr_dch1
- ecc_ap_err_intr_dch1_fault
- rd_linkecc_uncorr_err_intr_dch1
- rd_linkecc_uncorr_err_intr_fault_dch1
- rd_linkecc_corr_err_intr_dch1
- rd_linkecc_corr_err_intr_fault_dch1
- derate_temp_limit_intr
- derate_temp_limit_intr_fault
- derate_temp_limit_intr_dch1
- derate_temp_limit_intr_fault_dch1

Table 15-45 Interrupts Signals

Port Name	I/O	Description
ecc_corrected_err_intr	O	<p>This signal is the ECC corrected error interrupt. This interrupt is asserted when a correctable ECC error is detected at the DFI.</p> <p>Exists: MEMC_ECC</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
ecc_corrected_err_intr_fault[1:0]	O	<p>This signal is the ECC corrected error fault. This is a version of ecc_corrected_err_intr which can not be disabled or forced through register. It is a 2-bit antivalent signal with encoding as follows:</p> <ul style="list-style-type: none"> ■ 01: No Fault ■ 10: Fault Detected <p>Exists: MEMC_ECC</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
ecc_uncorrected_err_intr	O	<p>This signal is the ECC uncorrected error interrupt. This interrupt is asserted when an uncorrectable ECC error is detected.</p> <p>Exists: MEMC_ECC</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
ecc_uncorrected_err_intr_fault[1:0]	O	<p>This signal is the ECC uncorrected error fault. This is a version of ecc_uncorrected_err_intr which can not be disabled or forced through a register. It is a 2-bit antivalent signal with encoding as follows:</p> <ul style="list-style-type: none"> ■ 01: No Fault ■ 10: Fault Detected <p>Exists: MEMC_ECC</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
ecc_ap_err_intr	O	<p>This signal is the ECC address protection interrupt. This interrupt is asserted when the number of ECC errors within a burst exceeds the value set in ECCCFG1.ecc_ap_err_threshold.</p> <p>Exists: MEMC_ECCAP</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
ecc_ap_err_intr_fault[1:0]	O	<p>This signal is the ECC address protection fault. This is a version of ecc_ap_err_intr which can not be disabled or forced through a register. It is a 2-bit antivalent signal with encoding as follows:</p> <ul style="list-style-type: none"> ■ 01: No Fault ■ 10: Fault Detected <p>Exists: MEMC_ECCAP</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
capar_retry_limit_reached_intr_fault[1:0]	O	<p>This signal is a version of capar_retry_limit_reached_intr which can not be disabled or forced through a register. It is a 2-bit antivalent signal with encoding as follows:</p> <ul style="list-style-type: none"> ■ 01: No Fault ■ 10: Fault Detected <p>Exists: (DDRCTL_DDR) && (DDRCTL_CAPAR_RETRY)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
rd_linkecc_uncorr_err_intr	O	<p>This signal is the Read Link-ECC uncorrected error interrupt. This interrupt is asserted when an uncorrectable Link-ECC error is detected.</p> <p>Exists: MEMC_LINK_ECC</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
rd_linkecc_uncorr_err_intr_fault[1:0]	O	<p>This signal is the Read Link-ECC uncorrected error fault. This is a version of rd_linkecc_uncorr_err_intr, which can not be disabled or forced through a register. It is a 2-bit antivalent signal with encoding as follows:</p> <ul style="list-style-type: none"> ■ 01: No Fault ■ 10: Fault Detected <p>Exists: MEMC_LINK_ECC</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
rd_linkecc_corr_err_intr	O	<p>This signal is the Read Link-ECC corrected error interrupt. This interrupt is asserted when a correctable Link-ECC error is detected.</p> <p>Exists: MEMC_LINK_ECC</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
rd_linkecc_corr_err_intr_fault[1:0]	O	<p>This signal is the Read Link-ECC corrected error fault. This is a version of rd_linkecc_corr_err_intr which can not be disabled or forced through a register. It is a 2-bit antivalent signal with encoding of</p> <ul style="list-style-type: none"> ■ 01: No Fault ■ 10: Fault Detected <p>Exists: MEMC_LINK_ECC</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
capar_retry_limit_reached_intr_dch1_fault[1:0]	O	<p>This signal is a version of capar_retry_limit_reached_intr_dch1 which can not be disabled or forced through a register. It is a 2-bit antivalent signal with encoding as follows:</p> <ul style="list-style-type: none"> ■ 01: No Fault ■ 10: Fault Detected <p>Exists: (UMCTL2_DUAL_CHANNEL) && (DDRCTL_DDR) && (DDRCTL_CAPAR_RETRY)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
ecc_corrected_err_intr_dch1	O	<p>This signal is the ECC corrected error interrupt (Channel 1). This interrupt is asserted when a correctable ECC error is detected at the DFI.</p> <p>Exists: (UMCTL2_DUAL_CHANNEL) && (MEMC_ECC)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
ecc_corrected_err_intr_dch1_fault[1:0]	O	<p>This signal is the ECC corrected error fault (Channel 1). This is a version of ecc_corrected_err_intr_dch1 which can not be disabled or forced through a register. It is a 2-bit antivalent signal with encoding as follows:</p> <ul style="list-style-type: none"> ■ 01: No Fault ■ 10: Fault Detected <p>Exists: (UMCTL2_DUAL_CHANNEL) && (MEMC_ECC)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
ecc_uncorrected_err_intr_dch1	O	<p>This signal is the ECC uncorrected error interrupt (Channel 1). This interrupt is asserted when an uncorrectable ECC error is detected at the DFI.</p> <p>Exists: (UMCTL2_DUAL_CHANNEL) && (MEMC_ECC)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
ecc_uncorrected_err_intr_dch1_fault[1:0]	O	<p>This signal is the ECC uncorrected error fault (Channel 1). This is a version of ecc_uncorrected_err_intr_dch1 which can not be disabled or forced through a register. It is a 2-bit antivalent signal with encoding as follows:</p> <ul style="list-style-type: none"> ■ 01: No Fault ■ 10:: Fault Detected <p>Exists: (UMCTL2_DUAL_CHANNEL) && (MEMC_ECC)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
ecc_ap_err_intr_dch1	O	<p>This signal is the ECC address protection interrupt (Channel 1). This interrupt is asserted when a number of ECC errors within a burst exceeds the value set in ECCCFG1.ecc_ap_err_threshold.</p> <p>Exists: (UMCTL2_DUAL_CHANNEL) && (MEMC_ECCAP)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
ecc_ap_err_intr_dch1_fault[1:0]	O	<p>This signal is the ECC address protection fault (Channel 1). This is a version of ecc_ap_err_intr_dch1 which can not be disabled or forced through a register. It is a 2-bit antivalent signal with encoding as follows:</p> <ul style="list-style-type: none"> ■ 01: No Fault ■ 10: Fault Detected <p>Exists: (UMCTL2_DUAL_CHANNEL) && (MEMC_ECCAP) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
rd_linkecc_uncorr_err_intr_dch1	O	<p>This signal is the Read Link-ECC uncorrected error interrupt (Channel 1). This interrupt is asserted when a uncorrectable Link-ECC error is detected.</p> <p>Exists: (UMCTL2_DUAL_CHANNEL) && (MEMC_LINK_ECC) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
rd_linkecc_uncorr_err_intr_fault_dch1[1:0]	O	<p>This signal is the Read Link-ECC uncorrected error fault (Channel 1). This is a version of rd_linkecc_uncorr_err_intr_dch1 which can not be disabled or forced through a register. It is a 2-bit antivalent signal with encoding as follows:</p> <ul style="list-style-type: none"> ■ 01: No Fault ■ 10: Fault Detected <p>Exists: (UMCTL2_DUAL_CHANNEL) && (MEMC_LINK_ECC) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
rd_linkecc_corr_err_intr_dch1	O	<p>This signal is the Read Link-ECC corrected error interrupt (Channel 1). This interrupt is asserted when a correctable Link-ECC error is detected.</p> <p>Exists: (UMCTL2_DUAL_CHANNEL) && (MEMC_LINK_ECC) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
rd_linkecc_corr_err_intr_fault_dch1[1:0]	O	<p>This signal is the Read Link-ECC corrected error fault (Channel 1). This is a version of rd_linkecc_corr_err_intr_dch1 which can not be disabled or forced through a register. It is a 2-bit antivalent signal with encoding as follows:</p> <ul style="list-style-type: none"> 01: No Fault 10: Fault Detected <p>Exists: (UMCTL2_DUAL_CHANNEL) && (MEMC_LINK_ECC) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
derate_temp_limit_intr	O	<p>This signal is the derate temperature limit interrupt indicating that the LPDDR4/5 SDRAM temperature operating limit is exceeded. It is cleared by the DERATECTL.derate_temp_limit_intr_clr register.</p> <p>Exists: DDRCTL_DDR_OR_MEMC_LPDDR4 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
derate_temp_limit_intr_fault[1:0]	O	<p>This signal is the derate temperature limit fault. This is a version of derate_temp_limit_intr, which can not be disabled or forced via a register. It is a 2-bit antivalent signal with encoding as follows:</p> <ul style="list-style-type: none"> 01: No Fault 10: Fault Detected <p>Exists: DDRCTL_DDR_OR_MEMC_LPDDR4 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>
derate_temp_limit_intr_dch1	O	<p>This signal is the derate temperature limit interrupt indicating that the LPDDR4/5 SDRAM temperature operating limit is exceeded (channel1). This signal is cleared by the DERATECTL.derate_temp_limit_intr_clr register.</p> <p>Exists: (DDRCTL_DDR_OR_MEMC_LPDDR4) && (UMCTL2_DUAL_CHANNEL) Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
derate_temp_limit_intr_fault_dch1[1:0]	O	<p>This signal is the derate temperature limit fault (channel 1). This is a version of derate_temp_limit_intr which can not be disabled or forced by a register. It is a 2-bit antivalent signal with encoding as follows:</p> <ul style="list-style-type: none">■ 01: No Fault■ 10: Fault Detected <p>Exists: (DDRCTL_DDR_OR_MEMC_LPDDR4) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.46 APB Device Interface Signals

scanmode - pready
 scan_resetrn - prdata
 pclk - pslverr
 presetn
 pclk_rp
 presetn_rp
 paddr
 pwrdata
 pwrite
 psel
 penable

Table 15-46 APB Device Interface Signals

Port Name	I/O	Description
scanmode	I	This signal indicates that the controller is in scan mode. Exists: UMCTL2_USE_SCANMODE Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
scan_resetrn	I	This signal is the reset used when the controller is in scan mode. Exists: UMCTL2_USE_SCANMODE Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
pclk	I	This signal is the APB clock. It is used in the APB interface to program registers. Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
presetn	I	APB reset. Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: N/A Power Domain: DDRCTL_DOMAIN
pclk_rp	I	This signal is the free-running APB clock. It is used in Register Parity Protection. Exists: UMCTL2_REGPAR_EN_1 Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

Port Name	I/O	Description
presetn_rp	I	<p>This signal is the free-running APB reset. It is used in Register Parity Protection.</p> <p>Exists: UMCTL2_REGPAR_EN_1</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: N/A</p> <p>Power Domain: DDRCTL_DOMAIN</p>
paddr[(UMCTL2_APB_AW-1):0]	I	<p>This signal is the APB address.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
pdata[(UMCTL2_APB_DW-1):0]	I	<p>This signal is the APB write data.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
pwrite	I	<p>This signal is the APB direction.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
psel	I	<p>This signal is the APB peripheral select.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
penable	I	<p>This signal is the APB enable.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
pready	O	<p>This signal is the APB ready.</p> <p>Exists: Always</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

Port Name	I/O	Description
prdata[(UMCTL2_APB_DW-1):0]	O	This signal is the APB read data. Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
pslverr	O	This signal is the APB error. This signal is asserted only when APB address is out of range. Exists: Always Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.47 APB4 Device Interface Signals

pprot -
pstrb -



Table 15-47 APB4 Device Interface Signals

Port Name	I/O	Description
pprot[2:0]	I	APB4 PPROT input. Exists: DDRCTL_APB4_EN Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN
pstrb[((UMCTL2_APB_DW/8)-1):0]	I	APB4 PSTRB input. Exists: DDRCTL_APB4_EN Synchronous To: ::RCE_TPUBS::getSyncToClk Registered: * Varies Power Domain: DDRCTL_DOMAIN

15.48 Per-bank refresh Bank number Signals



- [hif_refresh_req_bank](#)
- [hif_refresh_req_bank_dch1](#)

Table 15-48 Per-bank refresh Bank number Signals

Port Name	I/O	Description
hif_refresh_req_bank[(((MEMC_NUM_RANKS*MEMC_BANK_BITS)-1):0]	O	<p>This signal indicates the next bank that is refreshed; for multi-rank configurations, the bank number is reported independently for each rank, and the information for all ranks is concatenated to form this signal. DEBUG ONLY</p> <p>Exists: MEMC_LPDDR4</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
hif_refresh_req_bank_dch1[(((MEMC_NUM_RANKS*MEMC_BANK_BITS)-1):0]	O	<p>This signal indicates the next bank that is refreshed (channel 1); for multi-rank configurations, the bank number is reported independently for each rank, and the information for all ranks is concatenated to form this signal. DEBUG ONLY</p> <p>Exists: (MEMC_LPDDR4) && (UMCTL2_DUAL_CHANNEL)</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

15.49 Register Parity Protection Signals



- [reg_par_err_intr](#)
- [reg_par_err_intr_fault](#)

Table 15-49 Register Parity Protection Signals

Port Name	I/O	Description
reg_par_err_intr	O	<p>This signal is the register parity error interrupt. This interrupt is asserted when a register parity error is detected. It may be cleared by writing to the REGPARCFG.reg_par_err_intr_clr register.</p> <p>Exists: UMCTL2_REGPAR_EN_1</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>
reg_par_err_intr_fault[1:0]	O	<p>This signal is the register parity error fault. This is a version of reg_par_err_intr which cannot be disabled or forced through a register. It is a 2-bit anti-valent signal with encoding as follows:</p> <ul style="list-style-type: none"> ■ 01: No Fault ■ 10: Fault Detected <p>Exists: UMCTL2_REGPAR_EN_1</p> <p>Synchronous To: ::RCE_TPUBS::getSyncToClk</p> <p>Registered: * Varies</p> <p>Power Domain: DDRCTL_DOMAIN</p>

A

Data Lane Mapping Examples

**Note**

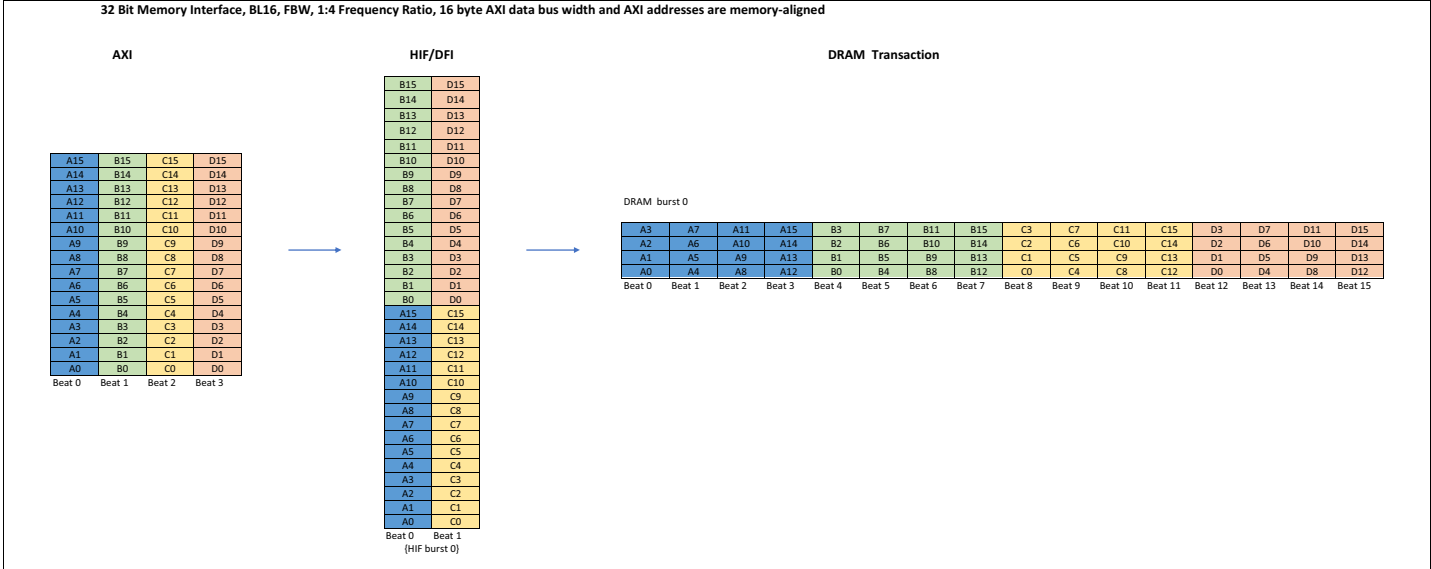
All the data lane mapping examples in this appendix are applicable for both read and write operations.

In figures:

1:4 Frequency Ratio means `MEMC_FREQ_RATIO=4`

BL16 means register field `MSTR0.burst_rdwr = 5'b01000`

Figure A-1 32 Bit Memory Interface, BL16, FBW, 1:4 Frequency Ratio, 16 byte AXI data bus width and AXI addresses are memory-aligned



B

Controller Performance Details

This appendix provides power consumption, gate count, and latency information. It contains the following sections:

- [“Timing”](#) on page [516](#)
- [“Area and Power”](#) on page [517](#)
- [“Latency Analysis”](#) on page [518](#)

B.1 Timing

The maximum achievable system clock (`core_ddrc_core_clk`) frequency is between 400 MHz and 1067 MHz depending on the following:

- Complexity/Size of DDRCTL configuration
- Frequency ratio selected
- Process technology
- Library PVT corner used for timing closure

Therefore, the supported SDRAM speed grades depend on the timing closure frequency. For example, in a slow technology, where the DDRCTL can reach a clock frequency of 400 MHz, data rates up to 1600 Mbps can be supported in DFI 1:2 frequency ratio mode, but 3200 Mbps can only be supported in DFI 1:4 frequency ratio mode.

If you need 4267 Mbps with LPDDR4 SDRAMs in DFI 1:2 frequency ratio mode, clock frequency must be 1067 MHz. If you need 6400 Mbps with LPDDR5 SDRAMs, clock frequency must be 800 MHz, and LPDDR5-6400 can only be supported in DFI 1:4 frequency ratio mode.

B.2 Area and Power

The following area is generated using Synopsys Design Compiler Reference Methodology (DCRM) in the topographical mode using the industry standard 7 nm high-speed library with a combination of SVT, LVT and uLVT cells (uLVT limited to 2%). Gate count is also reported in NAND2-equivalent gates, in Area column. These results include scan chains logic. The “Design for Test” options “Test Ready Compile” and “Insert Dft” are enabled in the coreConsultant “Synthesis” activity. A synthesis clock frequency of 600 MHz is used for `core_ddrc_core_clk`.



Note

Contact Synopsys if power estimate is required.

Stimulus Scenario	CAM Entry	DRAM Data Width	Frequency Ratio (MEMC_FREQ_RATIO)	Host Ports	Number of Ranks	Area (um2)	Gate Count
LPDDR4	64	32	4	1	1	289227	3765981
LPDDR5							
LPDDR4	64	32	4	2	1	355046	4622994
LPDDR5							
LPDDR4	32	32	4	2	1	266218	3466381

B.3 Latency Analysis

This section includes the following topics:

- “Write Latency With Controller Idle” on page 518
- “Read Latency With Controller Idle” on page 519

B.3.1 Write Latency With Controller Idle

The various elements of the write latency which are under the control of the DDRCTL are shown in [Figure B-2](#) on page 519. Note, that these are the best-case latencies. Actual latencies depend on other activities in progress at the time of each transaction. [Figure B-1](#) shows the components of write latency for which DDRCTL is responsible.

Figure B-1 Write Latency Components

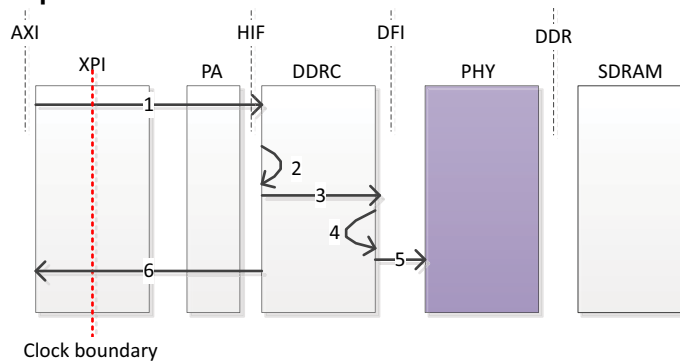


Figure B-2 Write Latency Components

Path	From	To	Hardware Parameter Settings	Register Settings	Write Latency Through DDRCTL for Open Page	Comment
1 - Address latency - AXI	Command presented on AXI: awvalid = 1	Command presented on HIF: hif_smd_valid = 1	<div>UMCTL2_A_NPO RTs</div> <div>UMCTL2_A_SYN C_n</div> <div>MEMC_ECC_SUP PORT</div> <div>MEMC_USE_RM W</div> <div>UMCTL2_XPI_US E_WAR</div> <div>ECCCFG0.ecc_mo de</div>		<div>TMIN = 1 core clock cycle</div> <div>TMIN + 1 core clock cycle</div> <div>TMIN + 1 core clock cycle</div> <div>MAX((TMIN + 1), (MSTR.burst_rdw/MEMC_FR EQ_RATIO))</div> <div>TMIN + 1 aclk cycle + UMCTL2_ASYNC_FIFO_N_SYNC C_n core cycles</div> <div>TMIN = 1 core clock cycle</div>	Note that all the parameter settings are not mutually exclusive; latency accumulates if more than one parameter is enabled. When ECC is enabled in software, the additional latency is due to store and forward operation on the write data with the assumption that (DDR_bytes==AXI_bytes). Default UMCTL2_ASYNC_FIFO_N_SYNC_n=2
			1 1 0 0 0 3'b000			
			1 1 0 0 1 3'b000			
			1 1 1 1 0 3'b100			
			1 0 0 0 0 3'b000			
			2 or more 1 0 0 0 3'b000			
2 - Write data pointer return	Command presented on HIF: hif_cmd_valid=1	hif_wdata_ptr_valid=1			2 core clock cycles	If there is address collision, this latency can be more, depending on how long it takes for the colliding entry to flush.
3 - Address latency through - DDRC	Last write data word on HIF: hif_wdata_end=1 and hif_wdata_valid=1	Write column command on DFI: dfi_cs=0 dfi_cas_n=0 dfi_ras_n=1 dfi_we_n=0	<div>MEMC_OPT_TIM ING</div> <div>MEMC_REG_DFI _OUT</div> <div>MEMC_INLINE_E CC</div>	<div>wdata_delay</div>		Write column command encoding on DFI is applicable to DDRx. For other protocols, refer to the DFI specification. Further command latency (for both write and read) can be incurred if DDR4 CAL mode is enabled. In that case, DFITMG1.dfi_t_cmd_lat specifies the number of DFI PHY clocks between the dfi_cs signal is asserted and when the associated address/ command is driven.
			0 0 0	>2	TMIN = 3 core clock cycles	
			0 0 0	2	TMIN + 1 core clock cycle	
			0 0 0	1	TMIN + 2 core clock cycle	
			0 0 0	0	TMIN + 3 core clock cycle	
			0 1 0	>2	TMIN + 1 core clock cycle	
			0 1 0	2	TMIN + 2 core clock cycle	
			0 1 0	1	TMIN + 3 core clock cycle	
			0 1 0	0	TMIN + 4 core clock cycle	
			1 N/A N/A	X	+1 core clock cycle	
4 - DFI write data enable latency	Write column command on DFI: dfi_cs=0 dfi_cas_n=0 dfi_ras_n=1	dfi_wdata_en=1			DFITMG0.dfi_tphy_wrlat	Determined by the PHY
5 - DFI write data latency	dfi_wdata_en=1	Write data sent on DFI interface			DFITMG0.dfi_tphy_wrdata	Determined by the PHY
6 - Write response latency - AXI	hif_wdata_ptr_valid=1	bvalid=1	<div>UMCTL2_DATA _CHANNEL_INT ERLEAVE_EN</div>			
			0 1		<div>TMIN = 2 core clock cycles</div> <div>TMIN + 1 core clock cycle</div>	

The unit of `DFITMG0.dfi_tphy_wrdata` is DRAM data clock cycles.

The unit of `DFITMG0.dfi_tphy_wrlat` is DRAM data clock cycles.

B.3.2 Read Latency With Controller Idle

The various elements of the read latency which are under the control of the DDRCTL are shown in [Figure B-4](#). Note, that these are the best-case latencies. Actual latencies depend on other activities in progress at the time of each transaction. [Figure B-3](#) shows the components of read latency for which the DDRCTL is responsible.

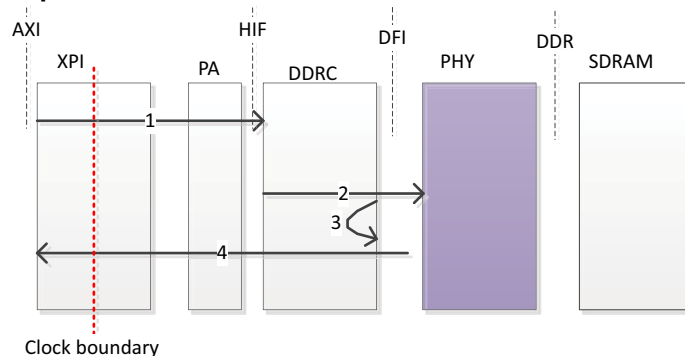
Figure B-3 Read Latency Components

Figure B-4 Read Idle Latency

Path	From	To	Parameter Settings						Read Latency Through DDRCTL for Open Page	Comment
1 - Address latency - AXI	Command presented on AXI: arvalid = 1	Command presented on HIF: hif_smd_valid = 1	UMCTL2_A_N_PORTS	UMCTL2_A_SYNC_n	UMCTL2_XPI_USE_INPUT_RAR	UMCTL2_XPI_USE_RAR				Note that all the parameter settings are not mutually exclusive; latency accumulates if more than one parameter is enabled. Default UMCTL2_ASYNC_FIFO_N_SYNC_n=2
			1	1	0	0			TMIN = 1 core clock cycle	
			1	1	0	1			TMIN + 1 core clock cycle	
			1	1	1	0			TMIN + 1 adk cycle	
			1	0	0	0			TMIN + 1 adk cycle + UMCTL2_ASYNC_FIFO_N_SYNC_n core cycles	
			2 or more	1	0	0			TMIN + 1 core clock cycle	
2 - Address latency through - DDRC	Command presented on HIF: hif_smd_valid = 1	Read column command on DFI: dfi_cs=0 dfi_cas_n=0 dfi_ras_n=1 dfi_we_n=1	MEMC_BYPASS*	MEMC_REG_DFI_OUT	MEMC_INLINE_ECC					Read column command encoding on DFI is applicable to DDRx and mDDR. For other protocols, refer to the DFI specification. The latency reduction is only for some eligible high-priority read (HPR) commands. MEMC_INLINE_ECC require MEMC_REG_DFI_OUT=1
			0	0	0				TMIN = 4 core clock cycles	
			0	1	0				TMIN + 1 core clock cycle	
			1	0	0				TMIN - 2 core clock cycles	
			0	1	1				TMIN + 2 core clock cycles	
3 - DFI read data enable latency	Read column command on DFI: dfi_cs=0 dfi_cas_n=0 dfi_ras_n=1 dfi_we_n=1	dfi_rddata_en=1							DFITMG0.dfi_t_rddata_en	Determined by the PHY
4 - Read response (data) latency - AXI	dfi_rddata_valid = 1	Read data received on AXI: rvalid = 1	UMCTL2_RRB_EXTRAM_n	UMCTL2_A_SYNC_n	MEMC_ECC_SUPPORT	UMCTL2_XPI_USE_RPR	UMCTL2_XPI_USE_RDR	UMCTL2_RRB_EXTRAM_RETUNE_n	UMCTL2_RRB_THRESHOLD_PPL_n	(1) In MEMC_FREQ_RATIO=2 cases, where upper and lower half of dfi_rddata_valid are not equal (dfi_rddata_valid_w0=dfi_rddata_valid_w1), there is an additional cycle of latency. (2) UMCTL2_RRB_THRESHOLD_PPL exists when UMCTL2_RRB_THRESHOLD_EN = 1. When both UMCTL2_RRB_THRESHOLD_EN and UMCTL2_RRB_THRESHOLD_PPL are enabled, there is an additional cycle of latency.
			0	1	0	0	0	0	0	
			0	1	0	0	1	0	0	
			0	1	0	1	0	0	0	
			0	1	> 0	0	0	0	0	
			0	0	0	0	0	0	0	
			1	1	0	0	0	0	0	
			1	1	0	0	0	1	0	
			0	1	0	0	0	0	1	

The unit of DFITMG0.dfi_t_rddata_en is DRAM data clock cycles.

C

DWC_ddrctl Automotive Safety

DWC_ddrctl is certified ISO 26262 ASIL B ready. This section describes the automotive features supported by the controller and enabled by the `DDRCTL_PRODUCT_NAME==2` parameter when you have the corresponding automotive license. The following topics are discussed:

- [“Overview of Automotive Safety Feature”](#) on page 522
- [“Automotive Safety Package Documents”](#) on page 523
- [“Automotive Specific Features”](#) on page 524

C.1 Overview of Automotive Safety Feature

To make an IP better suited for applications in the automotive space, functional safety needs to be considered throughout the IP development process. ISO 26262 addresses functional safety, and requires the hardware components to be analyzed with respect to their ability to detect and/or control the effects of random hardware faults. Random hardware faults can occur at any point in time during the life-cycle of the hardware component. They can take the form of permanent faults, or transient faults, and require safety mechanisms to be in place to ensure that severe consequences do not occur as a result of those faults.

The DDRCTL includes a broad range of internal safety mechanisms which assist in handling random hardware faults. These complement other mechanism already defined in the DDRCTL.

ASIL B Ready Certification

The DDRCTL IP has been analyzed in the scope of ISO 26262 targeting ASIL B for a particular reference configuration.

Reference Configuration

Users of the DDRCTL must understand that the ASIL B analysis uses a reference configuration with major features such as, REGPAR, OCCAP, OCPAR, OCSAP, and Inline ECC. The exact reference configuration is described in the accompanying Safety Manual.

The reference configuration internal safety mechanisms monitor the IP logic and report any detected errors to the application. The application should take an action to handle the error conditions and ensure safe operation of the hardware component.

Furthermore, ASIL B Ready status is dependent on several external diagnostics being implemented by you. These are described in the accompanying Safety Manual.

FMEDA

As part of the ASIL B Ready analysis, a Failure Mode, Effects, and Diagnostic Analysis (FMEDA) analysis has been performed for the DDRCTL IP. The analysis utilizes a per-transistor failure rate derived in accordance to IEC TR -62380. This, in conjunction with transistor count information allows one to determine the failure rate for the IP which can then be distributed between the modules of the design according to their relative area.

Safety Goals are defined for the IP. A list of failure modes and their effects is derived for each module and each of the failure modes is assigned a portion of the failure rate of the module they belong to. Failure modes are reviewed in the context of which Safety Goals they may cause to be violated, and if the violation is of type single-point or multi-point. If there is a diagnostic capability for each failure mode a Diagnostic Coverage number is assigned. The process is repeated for all failure modes and finally results are accumulated and IP safety metrics are collected in the FMEDA.

C.2 Automotive Safety Package Documents



Note

This section is under development.

The automotive safety package contains a number of automotive specific document as outlined in [Table C-1](#). You enable the safety package by setting the `DDRCTL_PRODUCT_NAME==2`. You can access the automotive documents in your workspace directory at `workspace/automotive`.

Table C-1 Description of Documents in Automotive Safety Package

Filename	Description	Contents
Certificate_ASIL-B_Random_DWC_ddrctl_lpddr54.pdf	Certificate	ASIL B certificate from SGS TUV SAAR
Certification_Report_ASIL-B_Random_DWC_ddrctl_lpddr54.pdf	Certification Report	Certification report from SGS TUV SAAR
WKPR17_DFMEA_DWC_ddrctl_lpddr54.xlsm	Safety Manual	Contains all safety relevant information regarding the use of the IP in safety related applications. Explains FMEDA worksheet.
WKPR22_FMEDA_DWC_ddrctl_lpddr54.xlsm	FMEDA Worksheet	Calculates FIT rates for each module. Enumerates diagnostic capabilities for failure modes.
WKPR41_Safety_Manual_DWC_ddrctl_lpddr54.pdf	DFMEA Worksheet	Design Failure Mode and Effects Analysis. This is a systematic group of activities to evaluate potential design failures.
SG_Quality_Manual.pdf	Solutions Group (SG) Quality Manual	This manual documents the Solutions Group Quality Management System Policies and key processes. It is compliant with ISO 9001:2015 and applicable IATF 16949:2016 requirements

C.3 Automotive Specific Features

DDRCTL_PRODUCT_NAME==2 is required to access automotive safety package and automotive specific features. If you do not have a valid license, this setting is not allowed in coreConsultant for you. The valid license is DWC-AP-LPDDR54-CONTROLLER. Only with this license, and configured with DDRCTL_PRODUCT_NAME==2 are the following automotive specific features available:

- [“Registers Parity Protection \(REGPAR\)” on page 322](#)
- [“On-Chip Command and Address Path Protection \(OCCAP\)” on page 327](#)
- [“On-Chip Parity \(OCPAR\)” on page 309](#)
- [“On-Chip External SRAM Address Protection \(OCSAP\)” on page 335](#)

D

Internal Parameter Descriptions

Provides a description of the internal parameters that might be indirectly referenced in expressions in the Signals, Parameters, or Registers chapters. These parameters are not visible in the coreConsultant GUI and most of them are derived automatically from visible parameters. **You must not set any of these parameters directly.**

Some expressions might refer to TCL functions or procedures (sometimes identified as **function_of**) that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the core in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

Table D-1 Internal Parameters

Parameter Name	Equals To
AXI_ADDR_BOUNDARY	UMCTL2_AXI_ADDR_BOUNDARY
AXI_BURSTW	2
AXI_CACHEW	4
AXI_PROTW	3
AXI_QOSW	4
AXI_RESPW	2
AXI_SIZEW	3
AXI_USERW	UMCTL2_AXI_USER_WIDTH_INT
BL16	0x8
BL8	0x4
CID_WIDTH	((UMCTL2_CID_WIDTH!=0)?UMCTL2_CID_WIDTH:1)
DDRCTL_1DDRC_2DFI	((DDRCTL_LPDDR == 1) && (MEMC_DRAM_DATA_WIDTH > 16)) ? 1 : 0)

Parameter Name	Equals To
DDRCTL_APB4_EN	(DDRCTL_CHB_MPAM_EN == 1 DDRCTL_CHB_TZ_EN == 1)
DDRCTL_CHB_BIDW	(DDRCTL_CHB_DRBIDW > DDRCTL_CHB_PRBIDW ? DDRCTL_CHB_DRBIDW : DDRCTL_CHB_PRBIDW)+1
DDRCTL_CHB_CHID_EN	(DDRCTL_CHB_VERSION == 3) && (DDRCTL_INCL_CHB==1)
DDRCTL_CHB_CHI_DW_GT_HIF_DW	DDRCTL_CHB_DW > UMCTL2_A_DW
DDRCTL_CHB_CHI_DW_LT_HIF_DW	DDRCTL_CHB_DW < UMCTL2_A_DW
DDRCTL_CHB_CHIE_EN	(DDRCTL_CHB_VERSION == 4) && (DDRCTL_INCL_CHB==1)
DDRCTL_CHB_CHI_TO_HIF_DW_RATIO	DDRCTL_CHB_CHI_DW_GT_HIF_DW ? DDRCTL_CHB_DW/UMCTL2_A_DW : 1
DDRCTL_CHB_DATA_CHECK_EN	0
DDRCTL_CHB_DATA_POIS_EN	0
DDRCTL_CHB_DIDW	(DDRCTL_CHB_XFRSZ_128B_EN==1) ? 3 : 2
DDRCTL_CHB_DRBIDW	[function_of: MEMC_NO_OF_ENTRY]
DDRCTL_CHB_HIF_BS_MAX_NUM_CHUNKS	DDRCTL_CHB_NUM_BEATS_BL*DDRCTL_CHB_HIF_MAX_NUM_CHUNKS_PER_BEAT
DDRCTL_CHB_HIF_BS_MAX_NUM_CHUNKS_CLOG2	[function_of: DDRCTL_CHB_HIF_BS_MAX_NUM_CHUNKS]
DDRCTL_CHB_HIF_MAX_NUM_CHUNKS_PER_BEAT	4
DDRCTL_CHB_HIF_MAX_NUM_CHUNKS_PER_BEAT_CLOG2	[function_of: DDRCTL_CHB_HIF_MAX_NUM_CHUNKS_PER_BEAT]
DDRCTL_CHB_HIF_TO_CHI_DW_RATIO	DDRCTL_CHB_CHI_DW_LT_HIF_DW ? UMCTL2_A_DW/DDRCTL_CHB_DW : 1
DDRCTL_CHB_KBD_ECC_EN	((DDRCTL_KBD_ECC_EN==1) (DDRCTL_KBD_ECC_BYP_EN==1))
DDRCTL_CHB_MAX_NUM_HIF_BUF_PER_CLN	((DDRCTL_CHB_XFRSZ*8)/(MEMC_DFI_DATA_WIDTH*DDRCTL_CHB_NUM_BEATS_BL8))
DDRCTL_CHB_MAX_PRC_LINES	8
DDRCTL_CHB_NUM_BEATS_BL	MEMC_BURST_LENGTH==32 ? DDRCTL_CHB_NUM_BEATS_BL32 : (MEMC_BURST_LENGTH==16 ? DDRCTL_CHB_NUM_BEATS_BL16 : DDRCTL_CHB_NUM_BEATS_BL8)

Parameter Name	Equals To
DDRCTL_CHB_NUM_BEATS_BL16	$16 / (\text{MEMC_FREQ_RATIO} * 2)$
DDRCTL_CHB_NUM_BEATS_BL32	$32 / (\text{MEMC_FREQ_RATIO} * 2)$
DDRCTL_CHB_NUM_BEATS_BL8	$8 / (\text{MEMC_FREQ_RATIO} * 2)$
DDRCTL_CHB_NUM_BEATS_BL_LOG2	[function_of: DDRCTL_CHB_NUM_BEATS_BL]
DDRCTL_CHB_NUM_CHI_BEATS_BL16	$(\text{DDRCTL_CHB_CHI_DW_GT_HIF_DW} ? ((\text{DDRCTL_CHB_DW} \geq (16 * \text{MEMC_DRAM_DATA_WIDTH})) ? 1 : ((16 * \text{MEMC_DRAM_DATA_WIDTH}) / \text{DDRCTL_CHB_DW})) : ((16 * \text{MEMC_DRAM_DATA_WIDTH}) / \text{DDRCTL_CHB_DW}))$
DDRCTL_CHB_NUM_CHI_BEATS_BL16W	$\text{DDRCTL_CHB_NUM_CHI_BEATS_BL16} > 1 ? 1 + [\text{function_of: DDRCTL_CHB_NUM_CHI_BEATS_BL16}] : 1$
DDRCTL_CHB_NUM_CHI_BEATS_BL32	$(\text{DDRCTL_CHB_CHI_DW_GT_HIF_DW} ? ((\text{DDRCTL_CHB_DW} \geq (32 * \text{MEMC_DRAM_DATA_WIDTH})) ? 1 : ((32 * \text{MEMC_DRAM_DATA_WIDTH}) / \text{DDRCTL_CHB_DW})) : ((32 * \text{MEMC_DRAM_DATA_WIDTH}) / \text{DDRCTL_CHB_DW}))$
DDRCTL_CHB_NUM_CHI_BEATS_BL32W	$\text{DDRCTL_CHB_NUM_CHI_BEATS_BL32} > 1 ? 1 + [\text{function_of: DDRCTL_CHB_NUM_CHI_BEATS_BL32}] : 1$
DDRCTL_CHB_NUM_CHI_BEATS_BL8	$\text{DDRCTL_CHB_CHI_DW_GT_HIF_DW} ? ((\text{DDRCTL_CHB_DW} \geq (8 * \text{MEMC_DRAM_DATA_WIDTH})) ? 1 : ((8 * \text{MEMC_DRAM_DATA_WIDTH}) / \text{DDRCTL_CHB_DW})) : ((8 * \text{MEMC_DRAM_DATA_WIDTH}) / \text{DDRCTL_CHB_DW})$
DDRCTL_CHB_NUM_CHI_BEATS_BL8W	$\text{DDRCTL_CHB_NUM_CHI_BEATS_BL8} > 1 ? 1 + [\text{function_of: DDRCTL_CHB_NUM_CHI_BEATS_BL8}] : 1$
DDRCTL_CHB_PER_SCRUB_EN	$((\text{DDRCTL_DFI_HIF_CMD_WDATA_PTR_EN} == 1) \&\& (\text{UMCTL2_SBR_EN} == 1))$
DDRCTL_CHB_PRBDW	[function_of: [function_of: DDRCTL_CHB_MAX_PRC_LINES*DDRCTL_CHB_MAX_NUM_HIF_BUF_PER_CLN]]
DDRCTL_CHB_TXIDW	$(\text{DDRCTL_CHB_CHIE_EN} == 1) ? 12 : (\text{DDRCTL_CHB_CHID_EN} == 1) ? 10 : 8$
DDRCTL_CHB_WDATA_PTR_BITS	$((\text{DDRCTL_CHB_PER_SCRUB_EN} == 1) ? \text{DDRCTL_DFI_HIF_CMD_WDATA_PTR_RANGE} : 0) + \text{DDRCTL_CHB_TXIDW} + \text{DDRCTL_CHB_WDPTR_DIDAW} + \text{DDRCTL_CHB_WDPTR_DIDMW} + \text{DDRCTL_CHB_WDPTR_NDIDW} + 1 + 1 + 1 + \text{DDRCTL_CHB_HIF_BS_MAX_NUM_CHUNKS_CLOG2} + 1 + \text{DDRCTL_CHB_HIF_MAX_NUM_CHUNKS_PER_BEAT_CLOG2} + 1 + ((\text{DDRCTL_CHB_WRZERO_EN} == 1) ? 1 : 0) + ((\text{DDRCTL_CHB_TZ_EN} == 1) ? 1 : 0)$

Parameter Name	Equals To
DDRCTL_CHB_WDPTR_BL16_DIDMW	DDRCTL_CHB_CHI_DW_LT_HIF_DW ? DDRCTL_CHB_HIF_TO_CHI_DW_RATIO*DDRCTL_CHB_NUM_BEATS_BL16 : DDRCTL_CHB_CHI_DW_GT_HIF_DW ? DDRCTL_CHB_CHI_TO_HIF_DW_RATIO*DDRCTL_CHB_NUM_CHI_BEATS_BL16 : DDRCTL_CHB_NUM_CHI_BEATS_BL16
DDRCTL_CHB_WDPTR_BL32_DIDMW	DDRCTL_CHB_CHI_DW_LT_HIF_DW ? DDRCTL_CHB_HIF_TO_CHI_DW_RATIO*DDRCTL_CHB_NUM_BEATS_BL32 : DDRCTL_CHB_CHI_DW_GT_HIF_DW ? DDRCTL_CHB_CHI_TO_HIF_DW_RATIO*DDRCTL_CHB_NUM_CHI_BEATS_BL32 : DDRCTL_CHB_NUM_CHI_BEATS_BL32
DDRCTL_CHB_WDPTR_BL8_DIDMW	DDRCTL_CHB_CHI_DW_LT_HIF_DW ? DDRCTL_CHB_HIF_TO_CHI_DW_RATIO*DDRCTL_CHB_NUM_BEATS_BL8 : DDRCTL_CHB_CHI_DW_GT_HIF_DW ? DDRCTL_CHB_CHI_TO_HIF_DW_RATIO*DDRCTL_CHB_NUM_CHI_BEATS_BL8 : DDRCTL_CHB_NUM_CHI_BEATS_BL8
DDRCTL_CHB_WDPTR_DIDAW	DDRCTL_CHB_DIDW
DDRCTL_CHB_WDPTR_DIDMW	MEMC_BURST_LENGTH==32 ? DDRCTL_CHB_WDPTR_BL32_DIDMW : MEMC_BURST_LENGTH==16 ? DDRCTL_CHB_WDPTR_BL16_DIDMW : DDRCTL_CHB_WDPTR_BL8_DIDMW
DDRCTL_CHB_WDPTR_NDIDW	((MEMC_BURST_LENGTH==32)?(DDRCTL_CHB_NUM_CHI_BEATS_BL32W?(MEMC_BURST_LENGTH==16):DDRCTL_CHB_NUM_CHI_BEATS_BL16W):DDRCTL_CHB_NUM_CHI_BEATS_BL8W)
DDRCTL_CHB_WRZERO_EN	(DDRCTL_CHB_CHIE_EN == 1)
DDRCTL_CHB_XFRSZ	64
DDRCTL_CHB_XFRSZ_128B_EN	(DDRCTL_CHB_XFRSZ == 128)
DDRCTL_DDR	((DDRCTL_PRODUCT_NAME == 1 DDRCTL_PRODUCT_NAME == 3 DDRCTL_PRODUCT_NAME == 5 DDRCTL_PRODUCT_NAME == 6 DDRCTL_PRODUCT_NAME == 7 DDRCTL_PRODUCT_NAME == 8) ? 1 : 0)
DDRCTL_DDRC_CID_WIDTH	UMCTL2_CID_WIDTH > 2 ? 2 : UMCTL2_CID_WIDTH

Parameter Name	Equals To
DDRCTL_DDR_DRAM_DATA_WIDTH	((MEMC_DRAM_DATA_WIDTH % 16) ? (MEMC_DRAM_DATA_WIDTH-8) : MEMC_DRAM_DATA_WIDTH)
DDRCTL_DDR_DRAM_ECC_WIDTH	((MEMC_DRAM_DATA_WIDTH % 16) ? 8 : MEMC_DRAM_ECC_WIDTH)
DDRCTL_DDR_DUAL_CHANNEL	((DDRCTL_DDR==1) && (UMCTL2_DUAL_CHANNEL==1))
DDRCTL_DDR_DUAL_CHANNEL_EN	(DDRCTL_DDR_DUAL_CHANNEL==1)
DDRCTL_DDR_DUAL_CHANNEL_OR_SINGLE_INST_DUALCH	(DDRCTL_DDR_DUAL_CHANNEL==1 DDRCTL_SINGLE_INST_DUALCH==1)
DDRCTL_DDR_EN	(DDRCTL_DDR)
DDRCTL_DDR_OR_MEMC_LPDDR4	(MEMC_DDR5==1 DDRCTL_LPDDR==1) ? 1 : 0
DDRCTL_DFI0_CS_WIDTH	((((MEMC_NUM_RANKS << UMCTL2_SHARED_AC_EN) > 4) ? 4 : (MEMC_NUM_RANKS << UMCTL2_SHARED_AC_EN))
DDRCTL_DFI_CTRLMSG	(DDRCTL_LPDDR==1)
DDRCTL_DFI_DATAEN_WIDTH	(DDRCTL_DFI_DATA_WIDTH / 16)
DDRCTL_DFI_DATA_WIDTH	(DDRCTL_DDR == 1) ? (((DDRCTL_DDR_DRAM_DATA_WIDTH >> DDRCTL_DDR_DCH_HBW) + DDRCTL_DDR_DRAM_ECC_WIDTH) << (UMCTL2_DUAL_CHANNEL_EN + 1)) : ((MEMC_DRAM_TOTAL_DATA_WIDTH*2) / UMCTL2_NUM_DFI)
DDRCTL_DFI_HIF_CMD_WDATA_PTR_EN	0
DDRCTL_DFI_HIF_CMD_WDATA_PTR_RANGE	(UMCTL2_INCL_ARB_OR_CHB==1) ? 1 : MEMC_HIF_WDATA_PTR_BITS
DDRCTL_DFI_MASK_WIDTH	(DDRCTL_DFI_DATA_WIDTH / 8)
DDRCTL_EAPAR_EN	0
DDRCTL_EAPAR_EN_1	(DDRCTL_EAPAR_EN==1)
DDRCTL_ENHANCED_WCK	((DDRCTL_LPDDR==1) && (MEMC_NUM_RANKS>1))
DDRCTL_FREQUENCY_BITS	((UMCTL2_FREQUENCY_NUM <= 2) ? 1 : (UMCTL2_FREQUENCY_NUM <= 4) ? 2 : (UMCTL2_FREQUENCY_NUM <= 8) ? 3 : 4)
DDRCTL_HIF_KBD_WIDTH	(MEMC_DFI_DATA_WIDTH/DDRCTL_NUM_BITS_PER_KBD)

Parameter Name	Equals To
DDRCTL_INCL_CHB	DDRCTL_SYS_INTF == 2
DDRCTL_KBD_ECC_BYP_EN	0
DDRCTL_KBD_ECC_EN	0
DDRCTL_KBD_SBECC_EN	((DDRCTL_KBD_ECC_EN==1) (DDRCTL_KBD_ECC_BYP_EN==1))
DDRCTL_KBD_SBECC_EN_1	(DDRCTL_KBD_SBECC_EN==1)
DDRCTL_LLC	((DDRCTL_PRODUCT_NAME==8) ? 1 : 0)
DDRCTL_LPDDR	((DDRCTL_PRODUCT_NAME == 0 DDRCTL_PRODUCT_NAME == 2 DDRCTL_PRODUCT_NAME == 4) ? 1 : 0)
DDRCTL_LPDDR_RFM	((DDRCTL_LPDDR == 1 && DDRCTL_HW_RFM_CTRL == 1) ? 1 : 0)
DDRCTL_LPDDR_RFMSBC	0
DDRCTL_LUT_ADDRMAP_CS_WIN_BITS	5
DDRCTL_LUT_ADDRMAP_EN	(DDRCTL_LUT_ADDRMAP)
DDRCTL_OCSAP_EN_1	(DDRCTL_OCSAP_EN == 1 ? 1 : 0)
DDRCTL_SINGLE_INST_DUALCH	0
DDRCTL_UMCTL5	1
DDRCTL_VER_NUMBER_VAL	0x3131302a
DDRCTL_VER_TYPE_VAL	0x736f3031
DDRCTL_XPI_PROG_SNF	0
DFI_LP_WAKEUP_PD_WIDTH	5
FBW	0x0
HBW	0x1
HIF_RQOS_TW	UMCTL2_XPI_RQOS_TW
INT_NPORTS_DATA	UMCTL2_INT_NPORTS_DATA
LPDDR45_DQSOSC	((DDRCTL_LPDDR==1) && (MEMC_FREQ_RATIO==4))
LPDDR45_DQSOSC_EN	(LPDDR45_DQSOSC)
LPDDR54_DQOSC_EN_OR_MEMC_DDR5	((LPDDR45_DQSOSC) (MEMC_DDR5==1))
MEMC_ACT_BYPASS	MEMC_BYPASS
MEMC_ADDR_ERR_EN	1

Parameter Name	Equals To
MEMC_BANK_BITS	(DDRCTL_DDR == 1 ? 2 : 3)
MEMC_BG_BANK_BITS	(DDRCTL_LPDDR == 1 MEMC_DDR4_BG_BITS2_INTERNAL_TESTING == 1 ? 4 : 5)
MEMC_BG_BITS	(DDRCTL_LPDDR == 1 MEMC_DDR4_BG_BITS2_INTERNAL_TESTING == 1 ? 2 : 3)
MEMC_BURST_LENGTH	16
MEMC_BYPASS	0
MEMC_DDR4_BG_BITS2_INTERNAL_TESTING	((DDRCTL_DDR == 1 && MEMC_BURST_LENGTH == 8) ? 1 : 0)
MEMC_DDR4_EN	(DDRCTL_DDR)
MEMC_DDR5	((((DDRCTL_DDR==1)&& (MEMC_BURST_LENGTH==16)&& (MEMC_ENH_RDWR_SWITCH==1))) ? 1 : 0)
MEMC_DFI_ADDR_WIDTH	(DDRCTL_LPDDR == 1) ? 14 : (DDRCTL_DDR == 1) ? 18 : 16
MEMC_DFI_ADDR_WIDTH_P0	(DDRCTL_LPDDR == 1) ? 14 : (DDRCTL_DDR == 1) ? MEMC_DFI_ADDR_WIDTH : MEMC_DFI_ADDR_WIDTH
MEMC_DFI_DATA_WIDTH	(MEMC_FREQ_RATIO == 2) ? MEMC_DRAM_DATA_WIDTH*4 : MEMC_DRAM_DATA_WIDTH*8
MEMC_DFI_ECC_WIDTH	(MEMC_FREQ_RATIO == 2 ? MEMC_DRAM_ECC_WIDTH * 4 : MEMC_DRAM_ECC_WIDTH * 8)
MEMC_DRAM_DATA_WIDTH_64_OR_MEMC_INLINE_ECC	(MEMC_DRAM_DATA_WIDTH==64 MEMC_DRAM_DATA_WIDTH==32 MEMC_INLINE_ECC_EN==1) ? 1 : 0
MEMC_DRAM_ECC_WIDTH	(MEMC_SIDEHAND_ECC_EN==1 ? 8 : 0)
MEMC_DRAM_NBYTES	(MEMC_DRAM_DATA_WIDTH/8)
MEMC_DRAM_NBYTES_LG2	[function_of: MEMC_DRAM_NBYTES]
MEMC_DRAM_TOTAL_DATA_WIDTH	MEMC_DRAM_DATA_WIDTH + MEMC_DRAM_ECC_WIDTH+0
MEMC_ECC	(MEMC_ECC_SUPPORT > 0)
MEMC_ECCAP	(MEMC_INLINE_ECC == 1 ? 1 : 0)
MEMC_ENH_CAM_PTR	1

Parameter Name	Equals To
MEMC_ENH_RDWR_SWITCH	(MEMC_ENH_CAM_PTR+0)
MEMC_FREQ_RATIO	(DDRCTL_LLC==1 ? 2 : 4)
MEMC_HIF_ADDR_WIDTH	(UMCTL2_LRANK_BITS + UMCTL2_DATA_CHANNEL_INTERLEAVE_EN + (MEMC_DDR4_BG_BITS2_INTERNAL_TESTING == 1 ? 33 : (DDRCTL_DDR == 1) ? 34 : (DDRCTL_LPDDR == 1) ? 32 : MEMC_HIF_MIN_ADDR_WIDTH))
MEMC_HIF_ADDR_WIDTH_MAX	(UMCTL2_DATA_CHANNEL_INTERLEAVE_EN==1 && UMCTL2_NUM_LRANKS_TOTAL==64) ? 41 : 40
MEMC_HIF_CMD_WDATA_MASK_FULL_EN	((UPCTL2_POSTED_WRITE_EN==1 && DDRCTL_LPDDR==1) MEMC_INLINE_ECC_EN==1) ? 1 : 0
MEMC_HIF_CREDIT_BITS	(MEMC_INLINE_ECC_EN==1) ? 2 : 1
MEMC_HIF_MIN_ADDR_WIDTH	32
MEMC_INLINE_ECC	MEMC_INLINE_ECC_EN
MEMC_INLINE_ECC_EN	(MEMC_INLINE_ECC+0)
MEMC_LPDDR4	(DDRCTL_LPDDR)
MEMC_LPDDR4_OR_UMCTL2_CID_EN	(DDRCTL_LPDDR==1 UMCTL2_CID_EN==1) ? 1 : 0
MEMC_MAX_INLINE_ECC_PER_BURST	((MEMC_DRAM_DATA_WIDTH * MEMC_BURST_LENGTH)/64)
MEMC_MAX_INLINE_ECC_PER_BURST_BITS	((MEMC_MAX_INLINE_ECC_PER_BURST == 2) ? 1 : ((MEMC_MAX_INLINE_ECC_PER_BURST == 4) ? 2 : ((MEMC_MAX_INLINE_ECC_PER_BURST == 8) ? 3 : ((MEMC_MAX_INLINE_ECC_PER_BURST == 16) ? 4 : 5))))
MEMC_MRR_DATA_TOTAL_DATA_WIDTH	(DDRCTL_DDR == 1 ? MEMC_DFI_TOTAL_DATA_WIDTH : MEMC_DRAM_TOTAL_DATA_WIDTH)
MEMC_NTT_UPD_ACT	(MEMC_ENH_CAM_PTR == 1) ? 1 : 0
MEMC_NTT_UPD_PRE	(MEMC_ENH_CAM_PTR == 1) ? 1 : 0
MEMC_NUM_CLKS	(DDRCTL_DDR == 1 ? 4 : 1)
MEMC_NUM_RANKS_GT_1	(MEMC_NUM_RANKS > 1) ? 1 : 0
MEMC_NUM_RANKS_GT_1_OR_DDRCTL_DDRC_CID_WIDTH_GT_0	(DDRCTL_DDRC_CID_WIDTH > 0 MEMC_NUM_RANKS > 1) ? 1 : 0
MEMC_OPT_TIMING	1
MEMC_OPT_WDATARAM	(DDRCTL_DDR_DCH_HBW == 1 ? 1 : 0)

Parameter Name	Equals To
MEMC_PAGE_BITS	(DDRCTL_DDR_EN == 1) ? 18 : 18
MEMC_PROG_FREQ_RATIO	1
MEMC_QBUS_SUPPORT	((MEMC_DRAM_DATA_WIDTH%32==0) (MEMC_DRAM_DATA_WIDTH==40) (MEMC_DRAM_DATA_WIDTH==72)) && (DDRCTL_PBW_MODE_SUPPORT==0)
MEMC_RANKBANK_BITS	UMCTL2_LRANK_BITS + MEMC_BG_BANK_BITS
MEMC_RANK_BITS	((MEMC_NUM_RANKS == 1) ? 0 : ((MEMC_NUM_RANKS == 2) ? 1 : 2))
MEMC_RD_BYPASS	MEMC_BYPASS
MEMC_RDCMD_ENTRY_BITS	(MEMC_NO_OF_ENTRY == 16 ? 4 : (MEMC_NO_OF_ENTRY == 32 ? 5 : 6))
MEMC_RDWR_SWITCH_POL_SEL	0
MEMC_SECDED_SIDEHAND_ECC	((MEMC_ECC_SUPPORT == 1 MEMC_ECC_SUPPORT == 3) && MEMC_SIDEHAND_ECC_EN==1)
MEMC_SIDEHAND_ECC	MEMC_SIDEHAND_ECC_EN
MEMC_SIDEHAND_ECC_EN	(MEMC_SIDEHAND_ECC+0)
MEMC_TAGBITS	(UMCTL2_INCL_ARB_OR_CHB == 0) ? MEMC_HIF_TAGBITS : (DDRCTL_INCL_CHB == 1) ? UMCTL2_CHB_TAGBITS : UMCTL2_MAX_AXI_TAGBITS
MEMC_WDATA_PTR_BITS	UMCTL2_INCL_ARB_OR_CHB==0 ? MEMC_HIF_WDATA_PTR_BITS:UMCTL2_INCL_ARB==1 ? UMCTL2_AXI_WDATA_PTR_BITS : DDRCTL_CHB_WDATA_PTR_BITS
MEMC_WRCMD_ENTRY_BITS	(MEMC_NO_OF_ENTRY == 16 ? 4 : (MEMC_NO_OF_ENTRY == 32 ? 5 : 6))
MEMC_WRDATA_4_CYCLES	(MEMC_FREQ_RATIO == 2 && MEMC_BURST_LENGTH == 16) ? 1 : 0
MEMC_WRDATA_8_CYCLES	0
MEMC_WRDATA_CYCLES	(MEMC_BURST_LENGTH/(MEMC_FREQ_RATIO*2))
NPORTS	UMCTL2_A_NPORTS
OCPAR_ADDR_PARITY_WIDTH	UMCTL2_OCPAR_ADDR_PARITY_W
QBW	0x2
SELFREF_TYPE_WIDTH	((DDRCTL_DDR_EN==1)?(MEMC_NUM_RANKS*2):2)
TARGET_FREQUENCY_WIDTH	DDRCTL_FREQUENCY_BITS

Parameter Name	Equals To
THEREIS_AXI4_PORT	<pre> ((UMCTL2_INCL_ARB == 1) && ((UMCTL2_A_TYPE_0 == 3 (UMCTL2_A_TYPE_1 == 3 && (UMCTL2_A_NPORTS > 1)) (UMCTL2_A_TYPE_2 == 3 && (UMCTL2_A_NPORTS > 2)) (UMCTL2_A_TYPE_3 == 3 && (UMCTL2_A_NPORTS > 3)) (UMCTL2_A_TYPE_4 == 3 && (UMCTL2_A_NPORTS > 4)) (UMCTL2_A_TYPE_5 == 3 && (UMCTL2_A_NPORTS > 5)) (UMCTL2_A_TYPE_6 == 3 && (UMCTL2_A_NPORTS > 6)) (UMCTL2_A_TYPE_7 == 3 && (UMCTL2_A_NPORTS > 7)) (UMCTL2_A_TYPE_8 == 3 && (UMCTL2_A_NPORTS > 8)) (UMCTL2_A_TYPE_9 == 3 && (UMCTL2_A_NPORTS > 9)) (UMCTL2_A_TYPE_10 == 3 && (UMCTL2_A_NPORTS > 10)) (UMCTL2_A_TYPE_11 == 3 && (UMCTL2_A_NPORTS > 11)) (UMCTL2_A_TYPE_12 == 3 && (UMCTL2_A_NPORTS > 12)) (UMCTL2_A_TYPE_13 == 3 && (UMCTL2_A_NPORTS > 13)) (UMCTL2_A_TYPE_14 == 3 && (UMCTL2_A_NPORTS > 14)) (UMCTL2_A_TYPE_15 == 3 && (UMCTL2_A_NPORTS > 15))))) ? 1 : 0 </pre>
THEREIS_AXI_PORT	<pre> ((UMCTL2_INCL_ARB == 1) && (((UMCTL2_A_TYPE_0 == 1 UMCTL2_A_TYPE_0 == 3) ((UMCTL2_A_TYPE_1 == 1 UMCTL2_A_TYPE_1 == 3) && (UMCTL2_A_NPORTS > 1)) ((UMCTL2_A_TYPE_2 == 1 UMCTL2_A_TYPE_2 == 3) && (UMCTL2_A_NPORTS > 2)) ((UMCTL2_A_TYPE_3 == 1 UMCTL2_A_TYPE_3 == 3) && (UMCTL2_A_NPORTS > 3)) ((UMCTL2_A_TYPE_4 == 1 UMCTL2_A_TYPE_4 == 3) && (UMCTL2_A_NPORTS > 4)) ((UMCTL2_A_TYPE_5 == 1 UMCTL2_A_TYPE_5 == 3) && (UMCTL2_A_NPORTS > 5)) ((UMCTL2_A_TYPE_6 == 1 UMCTL2_A_TYPE_6 == 3) && (UMCTL2_A_NPORTS > 6)) ((UMCTL2_A_TYPE_7 == 1 UMCTL2_A_TYPE_7 == 3) && (UMCTL2_A_NPORTS > 7)) ((UMCTL2_A_TYPE_8 == 1 UMCTL2_A_TYPE_8 == 3) && (UMCTL2_A_NPORTS > 8)) ((UMCTL2_A_TYPE_9 == 1 UMCTL2_A_TYPE_9 == 3) && (UMCTL2_A_NPORTS > 9)) ((UMCTL2_A_TYPE_10 == 1 UMCTL2_A_TYPE_10 == 3) && (UMCTL2_A_NPORTS > 10)) ((UMCTL2_A_TYPE_11 == 1 UMCTL2_A_TYPE_11 == 3) && (UMCTL2_A_NPORTS > 11)) ((UMCTL2_A_TYPE_12 == 1 UMCTL2_A_TYPE_12 == 3) && (UMCTL2_A_NPORTS > 12)) ((UMCTL2_A_TYPE_13 == 1 UMCTL2_A_TYPE_13 == 3) && (UMCTL2_A_NPORTS > 13)) ((UMCTL2_A_TYPE_14 == 1 UMCTL2_A_TYPE_14 == 3) && (UMCTL2_A_NPORTS > 14)) ((UMCTL2_A_TYPE_15 == 1 UMCTL2_A_TYPE_15 == 3) && </pre>

Parameter Name	Equals To
THEREIS_AXI_PORT ...(cont.)	(UMCTL2_A_NPORTS > 15))))) ? 1 : 0
THEREIS_PORT_DSIZE	((UMCTL2_INCL_ARB == 1) && ((UMCTL2_PORT_DSIZE_0 == 1 (UMCTL2_PORT_DSIZE_1 == 1 && (UMCTL2_A_NPORTS > 1)) (UMCTL2_PORT_DSIZE_2 == 1 && (UMCTL2_A_NPORTS > 2)) (UMCTL2_PORT_DSIZE_3 == 1 && (UMCTL2_A_NPORTS > 3)) (UMCTL2_PORT_DSIZE_4 == 1 && (UMCTL2_A_NPORTS > 4)) (UMCTL2_PORT_DSIZE_5 == 1 && (UMCTL2_A_NPORTS > 5)) (UMCTL2_PORT_DSIZE_6 == 1 && (UMCTL2_A_NPORTS > 6)) (UMCTL2_PORT_DSIZE_7 == 1 && (UMCTL2_A_NPORTS > 7)) (UMCTL2_PORT_DSIZE_8 == 1 && (UMCTL2_A_NPORTS > 8)) (UMCTL2_PORT_DSIZE_9 == 1 && (UMCTL2_A_NPORTS > 9)) (UMCTL2_PORT_DSIZE_10 == 1 && (UMCTL2_A_NPORTS > 10)) (UMCTL2_PORT_DSIZE_11 == 1 && (UMCTL2_A_NPORTS > 11)) (UMCTL2_PORT_DSIZE_12 == 1 && (UMCTL2_A_NPORTS > 12)) (UMCTL2_PORT_DSIZE_13 == 1 && (UMCTL2_A_NPORTS > 13)) (UMCTL2_PORT_DSIZE_14 == 1 && (UMCTL2_A_NPORTS > 14)) (UMCTL2_PORT_DSIZE_15 == 1 && (UMCTL2_A_NPORTS > 15))))) ? 1 : 0

Parameter Name	Equals To
THEREIS_PORT_USIZE	$((\text{UMCTL2_INCL_ARB} == 1) \&\& ((\text{UMCTL2_PORT_USIZE_0} == 1 \parallel (\text{UMCTL2_PORT_USIZE_1} == 1 \&\& (\text{UMCTL2_A_NPORTS} > 1)) \parallel (\text{UMCTL2_PORT_USIZE_2} == 1 \&\& (\text{UMCTL2_A_NPORTS} > 2)) \parallel (\text{UMCTL2_PORT_USIZE_3} == 1 \&\& (\text{UMCTL2_A_NPORTS} > 3)) \parallel (\text{UMCTL2_PORT_USIZE_4} == 1 \&\& (\text{UMCTL2_A_NPORTS} > 4)) \parallel (\text{UMCTL2_PORT_USIZE_5} == 1 \&\& (\text{UMCTL2_A_NPORTS} > 5)) \parallel (\text{UMCTL2_PORT_USIZE_6} == 1 \&\& (\text{UMCTL2_A_NPORTS} > 6)) \parallel (\text{UMCTL2_PORT_USIZE_7} == 1 \&\& (\text{UMCTL2_A_NPORTS} > 7)) \parallel (\text{UMCTL2_PORT_USIZE_8} == 1 \&\& (\text{UMCTL2_A_NPORTS} > 8)) \parallel (\text{UMCTL2_PORT_USIZE_9} == 1 \&\& (\text{UMCTL2_A_NPORTS} > 9)) \parallel (\text{UMCTL2_PORT_USIZE_10} == 1 \&\& (\text{UMCTL2_A_NPORTS} > 10)) \parallel (\text{UMCTL2_PORT_USIZE_11} == 1 \&\& (\text{UMCTL2_A_NPORTS} > 11)) \parallel (\text{UMCTL2_PORT_USIZE_12} == 1 \&\& (\text{UMCTL2_A_NPORTS} > 12)) \parallel (\text{UMCTL2_PORT_USIZE_13} == 1 \&\& (\text{UMCTL2_A_NPORTS} > 13)) \parallel (\text{UMCTL2_PORT_USIZE_14} == 1 \&\& (\text{UMCTL2_A_NPORTS} > 14)) \parallel (\text{UMCTL2_PORT_USIZE_15} == 1 \&\& (\text{UMCTL2_A_NPORTS} > 15))) ? 1 : 0$
THEREIS_SAR	$(\text{UMCTL2_A_NSAR} > 0) ? 1 : 0$
THEREIS_USE2RAQ	$(\text{UMCTL2_TOT_USE2RAQ} > 0) ? 1 : 0$
UMCTL2_A_AXI	$(\text{THEREIS_AXI_PORT} == 1) ? 1 : 0$
UMCTL2_A_AXI_0	$((\text{UMCTL2_A_NPORTS} \geq (0+1)) \&\& (\text{UMCTL2_A_TYPE_0} == 1 \parallel \text{UMCTL2_A_TYPE_0} == 3) \&\& (\text{UMCTL2_INCL_ARB} == 1)) ? 1 : 0$
UMCTL2_A_AXI_1	$((\text{UMCTL2_A_NPORTS} \geq (1+1)) \&\& (\text{UMCTL2_A_TYPE_1} == 1 \parallel \text{UMCTL2_A_TYPE_1} == 3) \&\& (\text{UMCTL2_INCL_ARB} == 1)) ? 1 : 0$
UMCTL2_A_AXI_10	$((\text{UMCTL2_A_NPORTS} \geq (10+1)) \&\& (\text{UMCTL2_A_TYPE_10} == 1 \parallel \text{UMCTL2_A_TYPE_10} == 3) \&\& (\text{UMCTL2_INCL_ARB} == 1)) ? 1 : 0$
UMCTL2_A_AXI_11	$((\text{UMCTL2_A_NPORTS} \geq (11+1)) \&\& (\text{UMCTL2_A_TYPE_11} == 1 \parallel \text{UMCTL2_A_TYPE_11} == 3) \&\& (\text{UMCTL2_INCL_ARB} == 1)) ? 1 : 0$

Parameter Name	Equals To
UMCTL2_A_AXI_12	((UMCTL2_A_NPORTS >= (12+1)) && (UMCTL2_A_TYPE_12 == 1 UMCTL2_A_TYPE_12 == 3) && (UMCTL2_INCL_ARB == 1)) ? 1 : 0
UMCTL2_A_AXI_13	((UMCTL2_A_NPORTS >= (13+1)) && (UMCTL2_A_TYPE_13 == 1 UMCTL2_A_TYPE_13 == 3) && (UMCTL2_INCL_ARB == 1)) ? 1 : 0
UMCTL2_A_AXI_14	((UMCTL2_A_NPORTS >= (14+1)) && (UMCTL2_A_TYPE_14 == 1 UMCTL2_A_TYPE_14 == 3) && (UMCTL2_INCL_ARB == 1)) ? 1 : 0
UMCTL2_A_AXI_15	((UMCTL2_A_NPORTS >= (15+1)) && (UMCTL2_A_TYPE_15 == 1 UMCTL2_A_TYPE_15 == 3) && (UMCTL2_INCL_ARB == 1)) ? 1 : 0
UMCTL2_A_AXI_2	((UMCTL2_A_NPORTS >= (2+1)) && (UMCTL2_A_TYPE_2 == 1 UMCTL2_A_TYPE_2 == 3) && (UMCTL2_INCL_ARB == 1)) ? 1 : 0
UMCTL2_A_AXI_3	((UMCTL2_A_NPORTS >= (3+1)) && (UMCTL2_A_TYPE_3 == 1 UMCTL2_A_TYPE_3 == 3) && (UMCTL2_INCL_ARB == 1)) ? 1 : 0
UMCTL2_A_AXI_4	((UMCTL2_A_NPORTS >= (4+1)) && (UMCTL2_A_TYPE_4 == 1 UMCTL2_A_TYPE_4 == 3) && (UMCTL2_INCL_ARB == 1)) ? 1 : 0
UMCTL2_A_AXI_5	((UMCTL2_A_NPORTS >= (5+1)) && (UMCTL2_A_TYPE_5 == 1 UMCTL2_A_TYPE_5 == 3) && (UMCTL2_INCL_ARB == 1)) ? 1 : 0
UMCTL2_A_AXI_6	((UMCTL2_A_NPORTS >= (6+1)) && (UMCTL2_A_TYPE_6 == 1 UMCTL2_A_TYPE_6 == 3) && (UMCTL2_INCL_ARB == 1)) ? 1 : 0
UMCTL2_A_AXI_7	((UMCTL2_A_NPORTS >= (7+1)) && (UMCTL2_A_TYPE_7 == 1 UMCTL2_A_TYPE_7 == 3) && (UMCTL2_INCL_ARB == 1)) ? 1 : 0
UMCTL2_A_AXI_8	((UMCTL2_A_NPORTS >= (8+1)) && (UMCTL2_A_TYPE_8 == 1 UMCTL2_A_TYPE_8 == 3) && (UMCTL2_INCL_ARB == 1)) ? 1 : 0
UMCTL2_A_AXI_9	((UMCTL2_A_NPORTS >= (9+1)) && (UMCTL2_A_TYPE_9 == 1 UMCTL2_A_TYPE_9 == 3) && (UMCTL2_INCL_ARB == 1)) ? 1 : 0
UMCTL2_A_DW	(MEMC_FREQ_RATIO == 4) ? (8*MEMC_DRAM_DATA_WIDTH):(MEMC_FREQ_RATIO == 2) ? (4*MEMC_DRAM_DATA_WIDTH) : (2*MEMC_DRAM_DATA_WIDTH))
UMCTL2_A_ID_MAPW	UMCTL2_A_IDW

Parameter Name	Equals To
UMCTL2_A_NPORTS_LG2	[function_of: UMCTL2_INT_NPORTS_DATA]
UMCTL2_AP_ANY_ASYNC	(UMCTL2_USE_SCANMODE == 1)?1:0
UMCTL2_APB_AW	24
UMCTL2_APB_DW	32
UMCTL2_A_RDWR_ORDERED_0	(UMCTL2_RDWR_ORDERED_0 == 1) ? 1 : 0
UMCTL2_A_RRB_THRESHOLD_EN_0	(UMCTL2_RRB_THRESHOLD_EN_0 == 1) ? 1 : 0
UMCTL2_A_SAR_0	(UMCTL2_A_NSAR >= (0+1)) ? 1 : 0
UMCTL2_A_USE2RAQ_0	(UMCTL2_XPI_USE2RAQ_0 == 1) ? 1 : 0
UMCTL2_AXI_ADDRW	((UMCTL2_A_ADDRW < UMCTL2_MIN_ADDRW) ? UMCTL2_MIN_ADDRW : UMCTL2_A_ADDRW)
UMCTL2_AXI_REGION_WIDTH	4
UMCTL2_AXI_SAR_BW	(THEREIS_SAR==1) ? (UMCTL2_A_ADDRW - UMCTL2_SAR_MIN_ADDRW) : 1
UMCTL2_AXI_SAR_REG_BW	(UMCTL2_AXI_SAR_BW==1) ? 2 : UMCTL2_AXI_SAR_BW
UMCTL2_AXI_SAR_SW	(THEREIS_SAR==1) ? 8 : 1
UMCTL2_AXI_USER_WIDTH_INT	(UMCTL2_AXI_USER_WIDTH > 0) ? UMCTL2_AXI_USER_WIDTH : 1
UMCTL2_AXI_WDATA_PTR_BITS	((DDRCTL_CHB_PER_SCRUB_EN==1) ? DDRCTL_DFI_HIF_CMD_WDATA_PTR_RANGE : 0) + 1 + UMCTL2_A_IDW + UMCTL2_A_NPORTS_LG2 + 5 + 2 + UMCTL2_AXI_USER_WIDTH_INT)
UMCTL2_BSM_BITS	[function_of: UMCTL2_NUM_BSM]
UMCTL2_CHB_TAGBITS	(1 + DDRCTL_CHB_HIF_BS_MAX_NUM_CHUNKS_CLOG2+1 + DDRCTL_CHB_NUM_BEATS_BL_LOG2 + DDRCTL_CHB_HIF_MAX_NUM_CHUNKS_PER_BEAT_C LOG2 + DDRCTL_CHB_BIDW + 1 + (DDRCTL_CHB_TZ_EN ? 1 : 0))
UMCTL2_CID_EN	(UMCTL2_CID_WIDTH>0)
UMCTL2_CMD_LEN_BITS	(MEMC_BURST_LENGTH == 16) ? 2 : 1
UMCTL2_DATA_CHANNEL_INTERLEAVE_EN_1	((UMCTL2_DATA_CHANNEL_INTERLEAVE_EN == 1) ? 1 : 0)

Parameter Name	Equals To
UMCTL2_DATA_CHANNEL_INTERLEAVE_NS_0	((UMCTL2_DATA_CHANNEL_INTERLEAVE_EN == 1) && ((UMCTL2_A_DW*UMCTL2_NUM_DATA_CHANNEL) == UMCTL2_PORT_DW_0)) ? 1 : 0)
UMCTL2_DATARAM_PAR_DW	(MEMC_DFI_DATA_WIDTH)/8
UMCTL2_DATARAM_PAR_DW_LG2	[function_of: UMCTL2_DATARAM_PAR_DW]
UMCTL2_DDR4_MRAM_EN	0
UMCTL2_DFI_MASK_PER_NIBBLE	0
UMCTL2_DUAL_CHANNEL_EN	(UMCTL2_DUAL_CHANNEL)
UMCTL2_DUAL_DATA_CHANNEL	((UMCTL2_NUM_DATA_CHANNEL == 2) ? 1 : 0)
UMCTL2_DUAL_HIF_1	(UMCTL2_DUAL_HIF == 1) ? 1 : 0
UMCTL2_DYN_BSM	((DDRCTL_DYN_BSM_MODE==1 DDRCTL_DYN_BSM_MODE==2) ? 1 : 0)
UMCTL2_ECC_TEST_MODE_EN	(UMCTL2_INCL_ARB_OR_CHB == 0 && MEMC_SIDEBAND_ECC_EN==1) ? 1 : 0
UMCTL2_HET_RANK	((UMCTL2_DDR4_MRAM_EN && (MEMC_NUM_RANKS > 1)) (DDRCTL_LUT_ADDRMAP_EN == 1)) ? 1 : 0
UMCTL2_INCL_ARB	DDRCTL_SYS_INTF == 1
UMCTL2_INCL_ARB_OR_CHB	DDRCTL_SYS_INTF != 0
UMCTL2_INT_NPORTS	(UMCTL2_A_NPORTS + UMCTL2_TOT_USE2RAQ + UMCTL2_SBR_EN)
UMCTL2_INT_NPORTS_DATA	(UMCTL2_A_NPORTS + UMCTL2_SBR_EN)
UMCTL2_LPDDR4_DUAL_CHANNEL	((DDRCTL_LPDDR==1) && (UMCTL2_DUAL_CHANNEL==1))
UMCTL2_LRANK_BITS	((UMCTL2_NUM_LRANKS_TOTAL == 1) ? 0 : (UMCTL2_NUM_LRANKS_TOTAL == 2) ? 1 : (UMCTL2_NUM_LRANKS_TOTAL == 4) ? 2 : (UMCTL2_NUM_LRANKS_TOTAL == 8) ? 3 : (UMCTL2_NUM_LRANKS_TOTAL == 16) ? 4 : (UMCTL2_NUM_LRANKS_TOTAL == 32) ? 5 : 6)
UMCTL2_MAX_AXI_TAGBITS	[function_of: UMCTL2_A_NPORTS]
UMCTL2_MAX_XPI_PORT_DW	[function_of: UMCTL2_A_NPORTS]
UMCTL2_MAX_XPI_PORT_DW_GTEQ_512	(UMCTL2_MAX_XPI_PORT_DW >= 512 ? 1 : 0)
UMCTL2_MIN_ADDRW	(MEMC_HIF_ADDR_WIDTH+MEMC_DRAM_NBYTES_LG 2)
UMCTL2_NUM_DATA_CHANNEL	((UMCTL2_DUAL_CHANNEL == 1) ? 2 : 1)

Parameter Name	Equals To
UMCTL2_OCCAP_DDRC_INTERNAL_TESTING	0
UMCTL2_OCCAP_EN_1	(UMCTL2_OCCAP_EN == 1 ? 1 : 0)
UMCTL2_OCECC_EN_1	(UMCTL2_OCECC_EN == 1 ? 1 : 0)
UMCTL2_OCPAR_ADDR_LOG_HIGH_WIDTH	((UMCTL2_AXI_ADDRW > UMCTL2_OCPAR_ADDR_LOG_LOW_WIDTH) ? (UMCTL2_AXI_ADDRW - UMCTL2_OCPAR_ADDR_LOG_LOW_WIDTH) : 1)
UMCTL2_OCPAR_ADDR_LOG_LOW_WIDTH	32
UMCTL2_OCPAR_ADDR_LOG_USE_MSB	((UMCTL2_AXI_ADDRW > UMCTL2_OCPAR_ADDR_LOG_LOW_WIDTH) ? 1 : 0)
UMCTL2_OCPAR_ADDR_PARITY_W	((UMCTL2_OCPAR_ADDR_PARITY_WIDTH == 0) ? 1 : [function_of:])
UMCTL2_OCPAR_EN_1	(UMCTL2_OCPAR_EN == 1 ? 1 : 0)
UMCTL2_OCPAR_OR_OCECC_EN_1	((UMCTL2_OCPAR_EN == 1) (UMCTL2_OCECC_EN == 1))
UMCTL2_OCPAR_WDATA_OUT_ERR_WIDTH	(MEMC_INLINE_ECC_EN+1)
UMCTL2_PARTIAL_WR	(MEMC_BURST_LENGTH==8 && MEMC_FREQ_RATIO==4) ? 0 : 1
UMCTL2_PORT_0	(UMCTL2_A_NPORTS >= (0+1)) ? 1 : 0
UMCTL2_PORT_1	(UMCTL2_A_NPORTS >= (1+1)) ? 1 : 0
UMCTL2_PORT_10	(UMCTL2_A_NPORTS >= (10+1)) ? 1 : 0
UMCTL2_PORT_11	(UMCTL2_A_NPORTS >= (11+1)) ? 1 : 0
UMCTL2_PORT_12	(UMCTL2_A_NPORTS >= (12+1)) ? 1 : 0
UMCTL2_PORT_13	(UMCTL2_A_NPORTS >= (13+1)) ? 1 : 0
UMCTL2_PORT_14	(UMCTL2_A_NPORTS >= (14+1)) ? 1 : 0
UMCTL2_PORT_15	(UMCTL2_A_NPORTS >= (15+1)) ? 1 : 0
UMCTL2_PORT_2	(UMCTL2_A_NPORTS >= (2+1)) ? 1 : 0
UMCTL2_PORT_3	(UMCTL2_A_NPORTS >= (3+1)) ? 1 : 0
UMCTL2_PORT_4	(UMCTL2_A_NPORTS >= (4+1)) ? 1 : 0
UMCTL2_PORT_5	(UMCTL2_A_NPORTS >= (5+1)) ? 1 : 0
UMCTL2_PORT_6	(UMCTL2_A_NPORTS >= (6+1)) ? 1 : 0
UMCTL2_PORT_7	(UMCTL2_A_NPORTS >= (7+1)) ? 1 : 0

Parameter Name	Equals To
UMCTL2_PORT_8	$(UMCTL2_A_NPORTS \geq (8+1)) ? 1 : 0$
UMCTL2_PORT_9	$(UMCTL2_A_NPORTS \geq (9+1)) ? 1 : 0$
UMCTL2_PORT_CH0_0	$(UMCTL2_NUM_VIR_CH_0 > 0 \ \&\& \ UMCTL2_STATIC_VIR_CH_0 == 1) ? 1 : 0$
UMCTL2_PORT_DSIZE_0	$(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB == 1 \ \&\& \ UMCTL2_A_TYPE_0 != 0 \ \&\& \ ((UMCTL2_PORT_DW_0 > UMCTL2_A_DW) \ \&\& \ (DDRCTL_UMCTL5 == 1 \ \&\& \ (UMCTL2_PORT_DW_0 > (UMCTL2_A_DW/4))))) ? 1 : 0) : 5)$
UMCTL2_PORT_DSIZE_1	$(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB == 1 \ \&\& \ UMCTL2_A_TYPE_1 != 0 \ \&\& \ ((UMCTL2_PORT_DW_1 > UMCTL2_A_DW) \ \&\& \ (DDRCTL_UMCTL5 == 1 \ \&\& \ (UMCTL2_PORT_DW_1 > (UMCTL2_A_DW/4))))) ? 1 : 0) : 5)$
UMCTL2_PORT_DSIZE_10	$(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB == 1 \ \&\& \ UMCTL2_A_TYPE_10 != 0 \ \&\& \ ((UMCTL2_PORT_DW_10 > UMCTL2_A_DW) \ \&\& \ (DDRCTL_UMCTL5 == 1 \ \&\& \ (UMCTL2_PORT_DW_10 > (UMCTL2_A_DW/4))))) ? 1 : 0) : 5)$
UMCTL2_PORT_DSIZE_11	$(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB == 1 \ \&\& \ UMCTL2_A_TYPE_11 != 0 \ \&\& \ ((UMCTL2_PORT_DW_11 > UMCTL2_A_DW) \ \&\& \ (DDRCTL_UMCTL5 == 1 \ \&\& \ (UMCTL2_PORT_DW_11 > (UMCTL2_A_DW/4))))) ? 1 : 0) : 5)$
UMCTL2_PORT_DSIZE_12	$(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB == 1 \ \&\& \ UMCTL2_A_TYPE_12 != 0 \ \&\& \ ((UMCTL2_PORT_DW_12 > UMCTL2_A_DW) \ \&\& \ (DDRCTL_UMCTL5 == 1 \ \&\& \ (UMCTL2_PORT_DW_12 > (UMCTL2_A_DW/4))))) ? 1 : 0) : 5)$
UMCTL2_PORT_DSIZE_13	$(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB == 1 \ \&\& \ UMCTL2_A_TYPE_13 != 0 \ \&\& \ ((UMCTL2_PORT_DW_13 > UMCTL2_A_DW) \ \&\& \ (DDRCTL_UMCTL5 == 1 \ \&\& \ (UMCTL2_PORT_DW_13 > (UMCTL2_A_DW/4))))) ? 1 : 0) : 5)$

Parameter Name	Equals To
UMCTL2_PORT_DSIZE_14	(UMCTL2_INCL_ARB== 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_14!=0 && ((UMCTL2_PORT_DW_14>UMCTL2_A_DW) (DDRCTL_UMCTL5==1 && (UMCTL2_PORT_DW_14>(UMCTL2_A_DW/4))))) ? 1 : 0) : 5)
UMCTL2_PORT_DSIZE_15	(UMCTL2_INCL_ARB== 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_15!=0 && ((UMCTL2_PORT_DW_15>UMCTL2_A_DW) (DDRCTL_UMCTL5==1 && (UMCTL2_PORT_DW_15>(UMCTL2_A_DW/4))))) ? 1 : 0) : 5)
UMCTL2_PORT_DSIZE_2	(UMCTL2_INCL_ARB== 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_2!=0 && ((UMCTL2_PORT_DW_2>UMCTL2_A_DW) (DDRCTL_UMCTL5==1 && (UMCTL2_PORT_DW_2>(UMCTL2_A_DW/4))))) ? 1 : 0) : 5)
UMCTL2_PORT_DSIZE_3	(UMCTL2_INCL_ARB== 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_3!=0 && ((UMCTL2_PORT_DW_3>UMCTL2_A_DW) (DDRCTL_UMCTL5==1 && (UMCTL2_PORT_DW_3>(UMCTL2_A_DW/4))))) ? 1 : 0) : 5)
UMCTL2_PORT_DSIZE_4	(UMCTL2_INCL_ARB== 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_4!=0 && ((UMCTL2_PORT_DW_4>UMCTL2_A_DW) (DDRCTL_UMCTL5==1 && (UMCTL2_PORT_DW_4>(UMCTL2_A_DW/4))))) ? 1 : 0) : 5)
UMCTL2_PORT_DSIZE_5	(UMCTL2_INCL_ARB== 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_5!=0 && ((UMCTL2_PORT_DW_5>UMCTL2_A_DW) (DDRCTL_UMCTL5==1 && (UMCTL2_PORT_DW_5>(UMCTL2_A_DW/4))))) ? 1 : 0) : 5)
UMCTL2_PORT_DSIZE_6	(UMCTL2_INCL_ARB== 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_6!=0 && ((UMCTL2_PORT_DW_6>UMCTL2_A_DW) (DDRCTL_UMCTL5==1 && (UMCTL2_PORT_DW_6>(UMCTL2_A_DW/4))))) ? 1 : 0) : 5)

Parameter Name	Equals To
UMCTL2_PORT_DSIZE_7	(UMCTL2_INCL_ARB== 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_7!=0 && ((UMCTL2_PORT_DW_7>UMCTL2_A_DW) (DDRCTL_UMCTL5==1 && (UMCTL2_PORT_DW_7>(UMCTL2_A_DW/4))))) ? 1 : 0) : 5)
UMCTL2_PORT_DSIZE_8	(UMCTL2_INCL_ARB== 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_8!=0 && ((UMCTL2_PORT_DW_8>UMCTL2_A_DW) (DDRCTL_UMCTL5==1 && (UMCTL2_PORT_DW_8>(UMCTL2_A_DW/4))))) ? 1 : 0) : 5)
UMCTL2_PORT_DSIZE_9	(UMCTL2_INCL_ARB== 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_9!=0 && ((UMCTL2_PORT_DW_9>UMCTL2_A_DW) (DDRCTL_UMCTL5==1 && (UMCTL2_PORT_DW_9>(UMCTL2_A_DW/4))))) ? 1 : 0) : 5)
UMCTL2_PORT_EN_RESET_VALUE	0
UMCTL2_PORT_USIZE_0	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_0!=0 && UMCTL2_PORT_DW_0<UMCTL2_A_DW) ? 1 : 0) : 5)
UMCTL2_PORT_USIZE_1	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_1!=0 && UMCTL2_PORT_DW_1<UMCTL2_A_DW) ? 1 : 0) : 5)
UMCTL2_PORT_USIZE_10	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_10!=0 && UMCTL2_PORT_DW_10<UMCTL2_A_DW) ? 1 : 0) : 5)
UMCTL2_PORT_USIZE_11	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_11!=0 && UMCTL2_PORT_DW_11<UMCTL2_A_DW) ? 1 : 0) : 5)
UMCTL2_PORT_USIZE_12	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_12!=0 && UMCTL2_PORT_DW_12<UMCTL2_A_DW) ? 1 : 0) : 5)
UMCTL2_PORT_USIZE_13	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_13!=0 && UMCTL2_PORT_DW_13<UMCTL2_A_DW) ? 1 : 0) : 5)
UMCTL2_PORT_USIZE_14	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_14!=0 && UMCTL2_PORT_DW_14<UMCTL2_A_DW) ? 1 : 0) : 5)

Parameter Name	Equals To
UMCTL2_PORT_USIZE_15	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_15!=0 && UMCTL2_PORT_DW_15<UMCTL2_A_DW) ? 1 : 0) : 5)
UMCTL2_PORT_USIZE_2	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_2!=0 && UMCTL2_PORT_DW_2<UMCTL2_A_DW) ? 1 : 0) : 5)
UMCTL2_PORT_USIZE_3	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_3!=0 && UMCTL2_PORT_DW_3<UMCTL2_A_DW) ? 1 : 0) : 5)
UMCTL2_PORT_USIZE_4	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_4!=0 && UMCTL2_PORT_DW_4<UMCTL2_A_DW) ? 1 : 0) : 5)
UMCTL2_PORT_USIZE_5	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_5!=0 && UMCTL2_PORT_DW_5<UMCTL2_A_DW) ? 1 : 0) : 5)
UMCTL2_PORT_USIZE_6	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_6!=0 && UMCTL2_PORT_DW_6<UMCTL2_A_DW) ? 1 : 0) : 5)
UMCTL2_PORT_USIZE_7	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_7!=0 && UMCTL2_PORT_DW_7<UMCTL2_A_DW) ? 1 : 0) : 5)
UMCTL2_PORT_USIZE_8	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_8!=0 && UMCTL2_PORT_DW_8<UMCTL2_A_DW) ? 1 : 0) : 5)
UMCTL2_PORT_USIZE_9	(UMCTL2_INCL_ARB == 1 ? ((UMCTL2_INCL_ARB==1 && UMCTL2_A_TYPE_9!=0 && UMCTL2_PORT_DW_9<UMCTL2_A_DW) ? 1 : 0) : 5)
UMCTL2_REF_ZQ_IO	0
UMCTL2_REGPAR_EN_1	(UMCTL2_REGPAR_EN == 1 ? 1 : 0)
UMCTL2_REG_SCRUB_INTERVALW	13
UMCTL2_RESET_WIDTH	1
UMCTL2_SAR_MIN_ADDRW	(UMCTL2_SARMINSIZE + 27)
UMCTL2_SBR_EN_1	(UMCTL2_SBR_EN == 1 ? 1 : 0)
UMCTL2_SHARED_AC_EN	(UMCTL2_SHARED_AC)
UMCTL2_TOKENW	(MEMC_NO_OF_ENTRY == 16 ? 4 : (MEMC_NO_OF_ENTRY == 32 ? 5 : 6))

Parameter Name	Equals To
UMCTL2_TOT_USE2RAQ	(UMCTL2_XPI_USE2RAQ_0 + UMCTL2_XPI_USE2RAQ_1 + UMCTL2_XPI_USE2RAQ_2 + UMCTL2_XPI_USE2RAQ_3 + UMCTL2_XPI_USE2RAQ_4 + UMCTL2_XPI_USE2RAQ_5 + UMCTL2_XPI_USE2RAQ_6 + UMCTL2_XPI_USE2RAQ_7 + UMCTL2_XPI_USE2RAQ_8 + UMCTL2_XPI_USE2RAQ_9 + UMCTL2_XPI_USE2RAQ_10 + UMCTL2_XPI_USE2RAQ_11 + UMCTL2_XPI_USE2RAQ_12 + UMCTL2_XPI_USE2RAQ_13 + UMCTL2_XPI_USE2RAQ_14 + UMCTL2_XPI_USE2RAQ_15)
UMCTL2_USE_SCANMODE	[function_of: UMCTL2_A_NPORTS UMCTL2_P_ASYNC_EN]
UMCTL2_VPR_EN	(UMCTL2_VPRW_EN == 1 ? 1 : 0)
UMCTL2_VPR_EN_VAL	(UMCTL2_VPR_EN)
UMCTL2_VPW_EN	(UMCTL2_VPRW_EN == 1 ? 1 : 0)
UMCTL2_WDATARAM_PAR_DW	(UMCTL2_OCECC_EN == 1) ? 5*(UMCTL2_WDATARAM_DW)/8 : (UMCTL2_WDATARAM_DW)/8
UMCTL2_WDATARAM_PAR_DW_EXT	(UMCTL2_OCECC_EN == 1) ? 5*(UMCTL2_WDATARAM_DW)/8 : (DDRCTL_OCSAP_EN == 1) ? (UMCTL2_WDATARAM_DW)/4 : (UMCTL2_WDATARAM_DW)/8
UMCTL2_XPI_RQOS_MLW	4
UMCTL2_XPI_RQOS_RW	2
UMCTL2_XPI_RQOS_TW	(UMCTL2_XPI_VPT_EN==1 (UMCTL2_INCL_ARB==0 && UMCTL2_VPRW_EN==1)) ? 11 : 1
UMCTL2_XPI_USE_RMW	((MEMC_ECC_SUPPORT>0 DDRCTL_DDR==1 DDRCTL_LPDDR==1) && MEMC_USE_RMW==1 && UMCTL2_INCL_ARB==1) ? 1 : 0)
UMCTL2_XPI_VPR_EN	(UMCTL2_INCL_ARB==1 && UMCTL2_VPR_EN==1) ? 1 : 0
UMCTL2_XPI_VPT_EN	(UMCTL2_XPI_VPR_EN==1 UMCTL2_XPI_VPW_EN==1) ? 1 : 0
UMCTL2_XPI_VPW_EN	(UMCTL2_INCL_ARB==1 && UMCTL2_VPW_EN==1) ? 1 : 0
UMCTL2_XPI_WQOS_MLW	4

Parameter Name	Equals To
UMCTL2_XPI_WQOS_RW	2
UMCTL2_XPI_WQOS_TW	(UMCTL2_XPI_VPT_EN==1 (UMCTL2_INCL_ARB==0 && UMCTL2_VPRW_EN==1)) ? 11 : 1
UPCTL2_POSTED_WRITE_EN	0
WRDATA_CYCLES	MEMC_WRDATA_CYCLES

E

Password Protected Features

This chapter describes the features supported by the controller that can be enabled when you have the corresponding license.

Table E-1 **Features Protected by Password**

Feature	Hardware Parameter Enabling the Feature	License Required for the Feature
ECC Supported	MEMC_ECC_SUPPORT	DWC-LPDDR54-CONTROLLER
Enable Inline ECC	MEMC_INLINE_ECC	DWC-AP-LPDDR54-CONTROLLER DWC-LPDDR54-CONTROLLER-AFP
Enable On-Chip ECC feature	UMCTL2_OCECC_EN	DWC-AP-LPDDR54-CONTROLLER
Enable On-Chip Command/Address Protection feature	UMCTL2_OCCAP_EN	DWC-AP-LPDDR54-CONTROLLER
Interleaving Support with Heterogeneous DRAM Densities	DDRCTL_HET_INTERLEAVE	DWC-LPDDR54-CONTROLLER-AFP

**Note**

- If you want to use these features, contact [Synopsys Customer Support](#).
- These features have dependencies on other hardware parameters which must be also enabled. Check the databook and cC GUI.

