

**UG0446**  
**User Guide**  
**SmartFusion2 and IGLOO2 FPGA High Speed DDR**  
**Interfaces**



a  **MICROCHIP** company



a  MICROCHIP company

**Microsemi Headquarters**

One Enterprise, Aliso Viejo,  
CA 92656 USA

Within the USA: +1 (800) 713-4113

Outside the USA: +1 (949) 380-6100

Sales: +1 (949) 380-6136

Fax: +1 (949) 215-4996

Email: [sales.support@microsemi.com](mailto:sales.support@microsemi.com)

[www.microsemi.com](http://www.microsemi.com)

©2022 Microsemi, a wholly owned subsidiary of Microchip Technology Inc. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

### About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at [www.microsemi.com](http://www.microsemi.com).

# Contents

<b>1</b>	<b>Revision History</b>	<b>1</b>
1.1	Revision 8.0	1
1.2	Revision 7.0	1
1.3	Revision 6.0	1
1.4	Revision 5.0	1
1.5	Revision 4.0	1
1.6	Revision 3.0	1
1.7	Revision 2.0	2
1.8	Revision 1.0	2
1.9	Revision 0.0	2
<b>2</b>	<b>Overview</b>	<b>3</b>
2.1	Contents	3
2.2	Additional Documentation	3
<b>3</b>	<b>MDDR Subsystem</b>	<b>5</b>
3.1	Features	5
3.2	Memory Configurations	6
3.3	Performance	7
3.4	I/O Utilization	7
3.5	Functional Description	8
3.5.1	Architecture Overview	8
3.5.2	Port List	9
3.5.3	Initialization	16
3.5.4	Details of Operation	18
3.5.5	MDDR Subsystem Features Configuration	25
3.6	How to Use MDDR in IGLOO2 Device	32
3.6.1	Configuring MDDR	32
3.6.2	Accessing MDDR from FPGA Fabric through the AXI Interface	42
3.6.3	Accessing MDDR from FPGA Fabric Through the AHB Interface	47
3.6.4	Accessing MDDR from the HPDMA	49
3.7	Timing Diagrams	51
3.8	Timing Optimization Technique for AXI	54
3.9	DDR Memory Device Examples	57
3.9.1	Example 1: Connecting 32-Bit DDR2 to MDDR_PADs	57
3.9.2	Example 2: Connecting 32-Bit DDR3 to MDDR_PADs with SECEDED	58
3.9.3	Example 3: Connecting 16-Bit LPDDR to MDDR_PADs with SECEDED	59
3.10	Board Design Considerations	60
3.11	MDDR Configuration Registers	60
3.11.1	SYSREG Configuration Register Summary	61
3.11.2	DDR Controller Configuration Register Summary	62
3.11.3	DDR Controller Configuration Register Bit Definitions	66
3.11.4	PHY Configuration Register Summary	102
3.11.5	PHY Configuration Register Bit Definitions	102
3.11.6	DDR_FIC Configuration Registers Summary	102
3.11.7	DDR_FIC Configuration Register Bit Definitions	104
3.12	Appendix A: How to Use the MDDR in SmartFusion2	111
3.12.1	Design Flow Using System Builder	111
3.12.2	Design Flow Using SmartDesign	120

3.12.3	Use Model 1: Accessing MDDR from FPGA Fabric Through the AXI Interface	125
3.12.4	Use Model 2: Accessing MDDR from FPGA Fabric Through the AHB Interface	127
3.12.5	Use Model 3: Accessing MDDR from Cortex-M3 Processor	129
3.12.6	Use Model 4: Accessing MDDR from the HPDMA	131
<b>4</b>	<b>Fabric DDR Subsystem</b>	<b>133</b>
4.1	Features	133
4.2	Memory Configurations	134
4.3	Performance	134
4.4	I/O Utilization	135
4.5	Functional Description	135
4.5.1	Architecture Overview	135
4.5.2	Port List	137
4.6	Initialization	143
4.6.1	Reset Sequence	143
4.6.2	ZQ Calibration	144
4.6.3	Details of Operation	145
4.6.4	FDDR Subsystem Features Configuration	151
4.6.5	Memory Type	151
4.6.6	Bus Width Configurations	151
4.6.7	Burst Mode	152
4.6.8	Configuring Dynamic DRAM Constraints	152
4.6.9	Dynamic DRAM Bank Constraints	152
4.6.10	Address Mapping	154
4.7	How to Use FDDR in IGLOO2 Devices	157
4.7.1	Configuring FDDR	157
4.7.2	Accessing FDDR from FPGA Fabric through the AXI Interface	165
4.7.3	Accessing FDDR from FPGA Fabric through the AHB Interface	170
4.8	DDR Memory Device Examples	172
4.8.1	Example 1: Connecting 32-Bit DDR2 to FDDR_PADs	172
4.8.2	Example 2: Connecting 32-Bit DDR3 to FDDR_PADs with SECEDED	172
4.8.3	Example 3: Connecting 16-Bit LPDDR to FDDR_PADs with SECEDED	173
4.9	FDDR Configuration Registers	174
4.9.1	FDDR SYSREG Configuration Register Summary	175
4.9.2	FDDR SYSREG Configuration Register Bit Definitions	175
4.10	Appendix A: How to Use the FDDR in SmartFusion2 Devices	183
4.10.1	Design Flow Using System Builder	183
4.10.2	Design Flow Using SmartDesign	191
4.10.3	Use Model 1: Accessing FDDR from FPGA Fabric Through AXI Interface	195
4.10.4	Use Model 2: Accessing FDDR from FPGA Fabric Through AHB Interface	198
4.11	Appendix B: Register Lock Bits Configuration	203
4.11.1	Lock Bit File	203
4.11.2	Lock Bit File Syntax	203
4.11.3	Locking and Unlocking a Register	204
<b>5</b>	<b>DDR Bridge</b>	<b>206</b>
5.1	Functional Description	207
5.1.1	Architecture Overview	207
5.1.2	Details of Operation	208
5.2	How to Use DDR Bridge in IGLOO2 Device	211
5.2.1	Configuring the DDR Bridge	212
5.2.2	High-Speed Data Transactions from HPDMA	213
5.2.3	Selecting Non-Bufferable Region	214
5.3	SYSREG Control Registers	214
5.4	DDR Bridge Control Registers in MDDR and FDDR	215
5.5	Appendix A: How to Use DDR Bridge in SmartFusion2 Device	215



5.5.1	Use Model 1: High Speed Data Transactions from Cortex-M3 Processor	216
5.5.2	Use Model 2: Selecting Non-Bufferable Region	217
<b>6</b>	<b>Soft Memory Controller Fabric Interface Controller</b>	<b>218</b>
6.1	Functional Description	219
6.1.1	Port List	219
6.2	How to Use SMC_FIC in IGLOO2 Device	224
6.3	SYSREG Control Register for SMC_FIC	226
6.4	Appendix A: How to Use SMC_FIC in SmartFusion2 Devices	226
6.4.1	Design Flow	226
6.4.2	Use Model 1: Accessing SDRAM from MSS Through CoreSDR_AXI	227

# Figures

Figure 1	System Level MDDR Block Diagram	6
Figure 2	MDDR Subsystem Functional Block Diagram	8
Figure 3	Reset Sequence	17
Figure 4	DDR_FIC Block Diagram	18
Figure 5	AXI Transaction Controller Block Diagram	20
Figure 6	DDR Controller Block Diagram	21
Figure 7	DDR RMW Operation (32-Bit DDR Bus Width and Burst Length 8)	23
Figure 8	DDR RMW Operation (16-Bit DDR Bus Width and Burst Length 8)	23
Figure 9	DDR RMW Operation (8-Bit DDR Bus Width and Burst Length 8)	23
Figure 10	Address Mapping	28
Figure 11	System Builder—Device Features Window	32
Figure 12	MDDR Initialization Path	33
Figure 13	I/O Drive Strength Setting	35
Figure 14	Selecting I/O Standard as LVCMOS18 or LPDDR1	36
Figure 15	Memory Initialization Configuration	38
Figure 16	Memory Timing Configuration	39
Figure 17	System Builder - Peripherals Tab	40
Figure 18	MDDR_CLK Configuration	41
Figure 19	DDR_FIC_CLK Configuration	41
Figure 20	I/O Editor Window	42
Figure 21	MDDR with AXI Interfaces	43
Figure 22	System Builder - Device Features Tab	44
Figure 23	Memory Configuration	44
Figure 24	Peripherals Tab with the Master Added and Configure Icon Highlighted	45
Figure 25	AMBA Master Configuration	45
Figure 26	System Clocks Configuration	46
Figure 27	SmartDesign Connections (Top Level View)	47
Figure 28	MDDR with Single AHB-Lite Interface	48
Figure 29	MDDR with HPDMA	49
Figure 30	System Builder - Device Features Tab	49
Figure 31	Memory Configurations	50
Figure 32	Clocks Configuration	50
Figure 33	AXI Single Write Transaction and Corresponding DDR Controller Commands	51
Figure 34	DDR Controller Command Sequence for Single AXI Write Transaction	51
Figure 35	AXI Single Read Transaction and Corresponding DDR Controller Commands	52
Figure 36	AXI INCR16 Write Transaction and Corresponding DDR Controller Commands	52
Figure 37	AXI INCR16 Write Transaction	53
Figure 38	DDR Controller Command Sequence for AXI INCR16 Write Transaction	53
Figure 39	AXI INCR-16 Read Transaction and Corresponding DDR Controller Commands	53
Figure 40	DDR Controller Command Sequence for AXI INCR-16 Read Transaction	54
Figure 41	AXI Timing Optimization Logic	55
Figure 42	Timing Diagram	55
Figure 43	x16 DDR2 SDRAM Connected to MDDR	58
Figure 44	x8 DDR3 SDRAM Connection to MDDR	59
Figure 45	x16 LPDDR1 SDRAM Connection to MDDR	60
Figure 46	System Builder - Device Features Window	112
Figure 47	MSS External DDR Memory Selection	112
Figure 48	I/O Drive Strength Setting	113
Figure 49	Selecting I/O Standard as LVCMOS18 or LPDDR1	114
Figure 50	DDR Memory initialization Settings	116
Figure 51	DDR Memory Timing Settings	117
Figure 52	MSS DDR FIC Subsystem Configuration	118
Figure 53	MDDR Clock Configuration	119
Figure 54	DDR_FIC Clock Configuration	120

Figure 55	Design Flow	121
Figure 56	MDDR Configurator	121
Figure 57	Memory Interface Configuration	122
Figure 58	MSS External DDR Memory Configurator	122
Figure 59	MDDR Clock Configuration	123
Figure 60	MDDR Clock Configuration	123
Figure 61	FIC_2 Configuration	124
Figure 62	I/O Configuration	124
Figure 63	MDDR with AXI Interface	125
Figure 64	MSS External Memory Configuration	126
Figure 65	Configuring FIC_2	126
Figure 66	MDDR Clock Configuration	126
Figure 67	SmartDesign Canvas	127
Figure 68	MDDR with Single AHB Interface	128
Figure 69	MDDR with Dual AHB Interface	129
Figure 70	Accessing MDDR from Cortex-M3 Processor	130
Figure 71	MSS External Memory Configuration	130
Figure 72	Configuring MDDR_CLK	131
Figure 73	Accessing MDDR from HPDMA	132
Figure 74	System Level FDDR Block Diagram	133
Figure 75	FDDR Subsystem Functional Block Diagram	135
Figure 76	Reset Sequence	144
Figure 77	DDR_FIC Block Diagram	147
Figure 78	AXI Transaction Controller Block Diagram	148
Figure 79	DDR Controller Block Diagram	149
Figure 80	Address Mapping	154
Figure 81	System Builder - Device Features Window	157
Figure 82	System Builder - Device Features Tab	158
Figure 83	Fabric DDR Memory Configuration	159
Figure 84	Selecting I/O Standard as LVCMOS18 or LPDDR1	160
Figure 85	Memory Initialization Configuration	162
Figure 86	Memory Timing Configuration	163
Figure 87	System Builder - Peripherals Tab	164
Figure 88	FDDR Clock Configuration	165
Figure 89	I/O Editor Window	165
Figure 90	FDDR Subsystem with AXI Interface	166
Figure 91	System Builder - Device Features Tab	167
Figure 92	Memory Configuration	167
Figure 93	Fabric DDR Subsystem Configuration Dialog	168
Figure 94	AMBA Master Configuration	168
Figure 95	Clocks Configuration	169
Figure 96	SmartDesign Connections (Top Level View)	170
Figure 97	FDDR with AHB-Lite interface	171
Figure 98	x16 DDR2 SDRAM Connected to FDDR	172
Figure 99	x8 DDR3 SDRAM Connection to FDDR	173
Figure 100	x16 LPDDR1 SDRAM Connection to FDDR	174
Figure 101	System Builder - Device Features Window	184
Figure 102	MSS External DDR Memory Selection	185
Figure 103	Fabric DDR Memory Settings	186
Figure 104	Selecting I/O Standard as LVCMOS18 or LPDDR1	186
Figure 105	DDR Memory initialization Settings	188
Figure 106	DDR Memory Timing Settings	189
Figure 107	MSS DDR FIC Subsystem Configuration	190
Figure 108	FDDR Clock Configuration	191
Figure 109	Design Flow	192
Figure 110	Fabric External Memory DDR Controller Configurator	193
Figure 111	FIC Configuration	194
Figure 112	I/O Configuration	194
Figure 113	FDDR with AXI Interface	195

Figure 114	FDDR Configuration	196
Figure 115	Fabric CCC Configuration	197
Figure 116	SmartDesign Canvas	197
Figure 117	Accessing FDDR Subsystem Through Dual AHB Interface	198
Figure 118	FIC_2 Configuration	199
Figure 119	MSS CCC Configuration	199
Figure 120	FDDR Configuration	200
Figure 121	Fabric CCC Configuration	201
Figure 122	CoreConfigP IP Configuration	201
Figure 123	CoreConfigP IP Configuration	202
Figure 124	SmartDesign Canvas	203
Figure 125	Lock Bit Configuration File	204
Figure 126	Register Lock Bit Settings Window	204
Figure 127	DDR Bridges in the SmartFusion2/IGLOO2 FPGA Device	206
Figure 128	DDR Bridge Functional Block Diagram	208
Figure 129	WCB Operation	209
Figure 130	Flow Chart for Read Operation	210
Figure 131	System Builder - Device Features Window	212
Figure 132	Configuring HPMS DDR Bridge for HPDMA	213
Figure 133	Configuring HPMS DDR Bridge For Non-Bufferable Region	213
Figure 134	Configuring DDR Bridge	214
Figure 135	Configuring MSS DDR Bridge	216
Figure 136	Configuring MSS DDR Bridge for Use Model 1	217
Figure 137	Configuring MSS DDR Bridge for Use Model 2	217
Figure 138	System Level SMC_FIC Block Diagram	218
Figure 139	SMC_FIC Block Diagram	219
Figure 140	HPMS External Memory Configurator	224
Figure 141	HPMS SMC_FIC Subsystem Configuration	225
Figure 142	CoreSDR_AXI Configuration	225
Figure 143	MSS External Memory Configurator	226
Figure 144	Core_AXI Configuration	227
Figure 145	Subsystem Connections in SmartDesign	228

# Tables

Table 1	Additional Documents	3
Table 2	Supported Memory (DDR2, DDR3 and LPDDR1) Configurations	7
Table 3	DDR Speeds	7
Table 4	I/O Utilization for SmartFusion2 and IGLOO2 Devices	7
Table 5	MDDR Subsystem Interface Signals	9
Table 6	AXI Slave Interface Signals	11
Table 7	AHB Slave Interface Signals	14
Table 8	MDDR APB Slave Interface Signals	15
Table 9	MDDR_CLK to FPGA Fabric Clock Ratios	19
Table 10	Priority Level Configuration	20
Table 11	SECEDED DQ Lines at DDR	24
Table 12	Supported Bus Widths	25
Table 13	Supported Burst Modes	26
Table 14	Dynamically Enforced Bank Constraints	26
Table 15	Dynamically-Enforced Bank Constraints	27
Table 16	Dynamic DRAM Global Constraints	27
Table 17	DDR Memory Regions	31
Table 18	Accessed DDR Memory Regions Based on Mode Settings for a 2 GB Memory	31
Table 19	Accessed DDR Memory Regions Based on Mode Settings for a 1 GB Memory	31
Table 20	Supported Address Width Range for Row, Bank and Column Addressing in DDR/LPDDR	34
Table 21	DDR I/O Standard is Configured Based on I/O Drive Strength Setting	34
Table 22	MDDR Throughput (for AHB)	48
Table 23	Number of Cycles for AXI/AHB Transactions to MDDR	54
Table 24	I/O Standards and Calibration Resistance Requirements for MDDR/FDDR	60
Table 25	Address Table for Register Interfaces	61
Table 26	SYSREG Configuration Register Summary	61
Table 27	DDR Controller Configuration Register	62
Table 28	DDRC_DYN_SOFT_RESET_CR	66
Table 29	DDRC_DYN_REFRESH_1_CR	66
Table 30	DDRC_DYN_REFRESH_2_CR	66
Table 31	DDRC_DYN_POWERDOWN_CR	67
Table 32	DDRC_MODE_CR	67
Table 33	DDRC_ADDR_MAP_BANK_CR	68
Table 34	DDRC_ADDR_MAP_COL_1_CR	69
Table 35	DDRC_ADDR_MAP_COL_2_CR	69
Table 36	DDRC_ADDR_MAP_ROW_1_CR	70
Table 37	DDRC_ADDR_MAP_ROW_2_CR	71
Table 38	DDRC_INIT_1_CR	72
Table 39	DDRC_CKE_RSTN_CYCLES_1_CR	72
Table 40	DDRC_CKE_RSTN_CYCLES_2_CR	73
Table 41	DDRC_INIT_MR_CR	73
Table 42	DDRC_INIT_EMR_CR	74
Table 43	DDRC_INIT_EMR2_CR	74
Table 44	DDRC_INIT_EMR3_CR	74
Table 45	DDRC_DRAM_BANK_TIMING_PARAM_CR	74
Table 46	DDRC_DRAM_RD_WR_LATENCY_CR	75
Table 47	DDRC_DRAM_RD_WR_PRE_CR	75
Table 48	DDRC_DRAM_MR_TIMING_PARAM_CR	75
Table 49	DDRC_DRAM_RAS_TIMING_CR	76
Table 50	DDRC_DRAM_RD_WR_TRNARND_TIME_CR	76
Table 51	DDRC_DRAM_T_PD_CR	77
Table 52	DDRC_DRAM_BANK_ACT_TIMING_CR	77
Table 53	DDRC_ODT_PARAM_1_CR	77
Table 54	DDRC_ODT_PARAM_2_CR	79

Table 55	DDRC_ADDR_MAP_COL_3_CR	79
Table 56	DDRC_MODE_REG_RD_WR_CR	80
Table 57	DDRC_MODE_REG_DATA_CR	80
Table 58	DDRC_PWR_SAVE_1_CR	81
Table 59	DDRC_PWR_SAVE_2_CR	81
Table 60	DDRC_ZQ_LONG_TIME_CR	82
Table 61	DDRC_ZQ_SHORT_TIME_CR	82
Table 62	DDRC_ZQ_SHORT_INT_REFRESH_MARGIN_1_CR	82
Table 63	DDRC_ZQ_SHORT_INT_REFRESH_MARGIN_2_CR	83
Table 64	DDRC_PERF_PARAM_1_CR	83
Table 65	DDRC_HPR_QUEUE_PARAM_1_CR	85
Table 66	DDRC_HPR_QUEUE_PARAM_2_CR	85
Table 67	DDRC_LPR_QUEUE_PARAM_1_CR	85
Table 68	DDRC_LPR_QUEUE_PARAM_2_CR	86
Table 69	DDRC_WR_QUEUE_PARAM_CR	86
Table 70	DDRC_PERF_PARAM_2_CR	86
Table 71	DDRC_PERF_PARAM_3_CR	87
Table 72	DDRC_DFI_RDDATA_EN_CR	87
Table 73	DDRC_DFI_MIN_CTRLUPD_TIMING_CR	88
Table 74	DDRC_DFI_MAX_CTRLUPD_TIMING_CR	88
Table 75	DDRC_DYN_SOFT_RESET_ALIAS_CR	88
Table 76	DDRC_AXI_FABRIC_PRI_ID_CR	89
Table 77	DDRC_SR	89
Table 78	DDRC_SINGLE_ERR_CNT_STATUS_SR	90
Table 79	DDRC_DOUBLE_ERR_CNT_STATUS_SR	90
Table 80	DDRC_LUE_SYNDROME_1_SR	91
Table 81	DDRC_LUE_SYNDROME_2_SR	91
Table 82	DDRC_LUE_SYNDROME_3_SR	92
Table 83	DDRC_LUE_SYNDROME_4_SR	93
Table 84	DDRC_LUE_SYNDROME_5_SR	93
Table 85	DDRC_LUE_ADDRESS_1_SR	94
Table 86	DDRC_LUE_ADDRESS_2_SR	94
Table 87	DDRC_LCE_SYNDROME_1_SR	95
Table 88	DDRC_LCE_SYNDROME_2_SR	95
Table 89	DDRC_LCE_SYNDROME_3_SR	96
Table 90	DDRC_LCE_SYNDROME_4_SR	97
Table 91	DDRC_LCE_SYNDROME_5_SR	97
Table 92	DDRC_LCE_ADDRESS_1_SR	98
Table 93	DDRC_LCE_ADDRESS_2_SR	98
Table 94	DDRC_LCB_NUMBER_SR	98
Table 95	DDRC_LCB_MASK_1_SR	99
Table 96	DDRC_LCB_MASK_2_SR	99
Table 97	DDRC_LCB_MASK_3_SR	100
Table 98	DDRC_LCB_MASK_4_SR	100
Table 99	DDRC_ECC_INT_SR	101
Table 100	DDRC_ECC_INT_CLR_REG	101
Table 101	PHY Configuration Register Summary	102
Table 102	PHY_DATA_SLICE_IN_USE_CR	102
Table 103	DDR_FIC Configuration Register Summary	102
Table 104	DDR_FIC_NB_ADDR_CR	104
Table 105	DDR_FIC_NBRWB_SIZE_CR	104
Table 106	DDR_FIC_BUF_TIMER_CR	104
Table 107	DDR_FIC_HPD_SW_RW_EN_CR	105
Table 108	DDR_FIC_HPD_SW_RW_INVAL_CR	105
Table 109	DDR_FIC_SW_WR_ERCLR_CR	106
Table 110	DDR_FIC_ERR_INT_ENABLE	106
Table 111	DDR_FIC_NUM_AHB_MASTERS_CR	107
Table 112	DDR_FIC_HPB_ERR_ADDR_1_SR	107
Table 113	DDR_FIC_HPB_ERR_ADDR_2_SR	108

Table 114	DDR_FIC_SW_ERR_ADDR_1_SR	108
Table 115	DDR_FIC_SW_ERR_ADDR_2_SR	108
Table 116	DDR_FIC_HPD_SW_WRB_EMPTY_SR	109
Table 117	DDR_FIC_SW_HPB_LOCKOUT_SR	109
Table 118	DDR_FIC_SW_HPD_WERR_SR	110
Table 119	DDR_FIC_LOCK_TIMEOUTVAL_1_CR	110
Table 120	DDR_FIC_LOCK_TIMEOUTVAL_2_CR	110
Table 121	DDR_FIC_LOCK_TIMEOUT_EN_CR	111
Table 122	DDR_FIC_RDWR_ERR_SR	111
Table 123	DDR I/O Standard Configured Based on I/O Drive Strength Setting	113
Table 124	Supported Memory (DDR2, DDR3, and LPDDR1) Configurations	134
Table 125	DDR Speeds	134
Table 126	I/O Utilization for SmartFusion2 and IGLOO2 Devices	135
Table 127	FDDR Subsystem Interface Signals	137
Table 128	FDDR AXI Slave Interface Signals	139
Table 129	FDDR AHB Slave Interface Signals	142
Table 130	FDDR APB Slave Interface Signals	143
Table 131	FDDR_CLK to FPGA Fabric Clock Ratios	147
Table 132	SECEDED DQ Lines at DDR	150
Table 133	Supported Bus Widths	152
Table 134	Supported Burst Modes for M2S150 and M2GL150	152
Table 135	Dynamically Enforced Bank Constraints	152
Table 136	Dynamically Enforced Bank Constraints	153
Table 137	Dynamic DRAM Global Constraints	153
Table 138	Supported Address Width Range for Row, Bank and Column	158
Table 139	DDR I/O Standard is Configured based on I/O Drive Strength Setting	159
Table 140	FDDR Throughput (for AHB)	171
Table 141	Address Table for Register Interfaces	174
Table 142	FDDR SYSREG	175
Table 143	PLL_CONFIG_LOW_1	175
Table 144	PLL_CONFIG_LOW_2	176
Table 145	PLL_CONFIG_HIGH	176
Table 146	FDDR_FACC_CLK_EN	177
Table 147	FDDR_FACC_MUX_CONFIG	178
Table 148	FDDR_FACC_DIVISOR_RATIO	179
Table 149	PLL_DELAY_LINE_SEL	180
Table 150	FDDR_SOFT_RESET	180
Table 151	FDDR_IO_CALIB	180
Table 152	FDDR_INTERRUPT_ENABLE	181
Table 153	F_AXI_AHB_MODE_SEL	181
Table 154	PHY_SELF_REF_EN	182
Table 155	FDDR_FAB_PLL_CLK_SR	182
Table 156	FDDR_FPLL_CLK_SR	182
Table 157	FDDR_INTERRUPT_SR	182
Table 158	FDDR_IO_CALIB_SR	183
Table 159	FDDR_FATC_RESET	183
Table 160	Supported Address Width Range for Row, Bank and Column Addressing in DDR/LPDDR	185
Table 161	DDR I/O Standard is Configured Based on I/O Drive Strength Setting	185
Table 162	SmartFusion2 and IGLOO2 FPGA DDR Bridge Interface	207
Table 163	SYSREG Control Registers	214
Table 164	DDR Bridge Control Registers in MDDR and FDDR	215
Table 165	SMC_FIC 64-bit AXI Port List	219
Table 166	SMC_FIC 32-bit AHB-Lite Port List	223
Table 167	MDDR_CR Register	226



# 1 Revision History

---

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

## 1.1 Revision 8.0

- Decreased memory density supported by SmartFusion2 and IGLOO2 Fabric Double-Data Rate (FDDR) controller from 4 GB to 2 GB. Similarly, the memory density supported by the SmartFusion2 MSS Double-Data Rate (MDDR) controller is decreased from 4 GB to 2 GB. For more information, see [Customer Notification \(CN\)](#).

## 1.2 Revision 7.0

The following is a summary of the changes in this revision.

- Read and Write leveling is not supported. Removed information about all the Read and Write leveling registers.
- Most of the PHY registers have been reserved.

## 1.3 Revision 6.0

The following is a summary of the changes in this revision.

- Updated [I/O Utilization](#), page 7, [I/O Utilization](#), page 135, [DDRIO Calibration](#), page 16, and [DDRIO Calibration](#), page 144 (SAR 81073).

## 1.4 Revision 5.0

The following is a summary of the changes in this revision.

- Updated [MDDR Subsystem](#), page 5 and [Fabric DDR Subsystem](#), page 133 (SARs 62955 and 62858).
- Updated [Table 2](#), page 7, [Table 4](#), page 7, [Table 11](#), page 24 (SAR 78912).
- Updated [Initialization](#), page 16 and [Power Saving Modes](#), page 24 (SAR 52819).
- Updated [Table 85](#), page 94, [Table 86](#), page 94, [Table 92](#), page 98, [Table 93](#), page 98 (SAR 75057).
- Updated [Architecture Overview](#), page 135 (SAR 79005).
- Added [DDR Memory Initialization Time](#), page 18 (SAR 72725).
- Updated [Appendix B: Register Lock Bits Configuration](#), page 203 (SAR 79864).

## 1.5 Revision 4.0

The following is a summary of the changes in this revision.

- Merged SmartFusion2 and IGLOO2 User Guides.
- Updated [Additional Documentation](#), page 3 (SAR 68482).
- Updated [MDDR Subsystem](#), page 5 and [Fabric DDR Subsystem](#), page 133 (SARs 55467, 54300, 49186, 52819, 54053, 51933, 55041, 52727, 48832).
- Updated [MDDR Subsystem](#), page 5 (SARs 62441, 66225, 60914, 69568, 66860, 69611, 69261, 68400, 64575, 65164, and 69655).
- Updated [Fabric DDR Subsystem](#), page 133 (SARs 62441, 60914, 66860, 69144, and 54429).
- Updated [DDR Bridge](#), page 206.
- Updated [Soft Memory Controller Fabric Interface Controller](#), page 218.

## 1.6 Revision 3.0

The following is a summary of the changes in this revision.

- Updated the Part Numbers (M2S075 to M2S090, M2S080 to M2S100, and M2S120 to M2S150) as required (SAR 47554).



- Updated [MDDR Subsystem](#), page 5 (SARs 47919, 48832, 49947, 50561, 50732, 62858, and 62955).
- Updated [Fabric DDR Subsystem](#), page 133 (SARs 62858 and 62955).
- Updated [Soft Memory Controller Fabric Interface Controller](#), page 218 (SAR 48330).

## 1.7 Revision 2.0

The following is a summary of the changes in this revision.

- Restructured the user guide (SARs 47314, 45974, 45616, 43424, 46149, 46446).
- Updated [MDDR Subsystem](#), page 5 (SARs 55041, 58032, 51465, 58034, 58035, 58037, 51933, 58038, 57034, and 57207).
- Updated [Fabric DDR Subsystem](#), page 133 (SARs 58034, 58035, 58037, 51933, 58038, 57034, 57207, and 58038).
- Updated [Soft Memory Controller Fabric Interface Controller](#), page 218 (SAR 54036).
- Updated [MDDR Memory Map](#), page 30 (SAR 44198).
- Updated [Address Mapping](#), page 154 (SAR 45761).

## 1.8 Revision 1.0

The following is a summary of the changes in this revision.

- Restructured the user guide.
- Updated the user guide (SAR 42443).
- Updated [MDDR Subsystem](#), page 5, [Fabric DDR Subsystem](#), page 133, and [DDR Bridge](#), page 206 (SAR 50157).
- Updated [MDDR Subsystem](#), page 5 and [Fabric DDR Subsystem](#), page 133 (SAR 41901).
- Updated [MDDR Subsystem](#), page 5 (SAR 42751).
- Updated [Fabric DDR Subsystem](#), page 133 (SAR 41979).

## 1.9 Revision 0.0

The first publication of this document.

## 2 Overview

This user guide describes the high speed memory interfaces in SmartFusion<sup>®</sup>2 System-on-Chip (SoC) field programmable gate array (FPGA) and IGLOO<sup>®</sup>2 FPGA devices. The high speed interfaces microcontroller/memory subsystem double-data rate (MDDR) subsystem and fabric DDR (FDDR) subsystem provide access to DDR memories for high-speed data transfers. The DDR subsystems functionality, configurations, and their use models are discussed in this user guide.

### 2.1 Contents

This user guide contains the following chapters:

- "MDDR Subsystem"
- "Fabric DDR Subsystem"
- "DDR Bridge"
- "Soft Memory Controller Fabric Interface Controller"

### 2.2 Additional Documentation

The following table describes additional documentation available for the SmartFusion2 and IGLOO2 devices. For more information, refer to the [SmartFusion2 Documentation Page](#) and [IGLOO2 Documentation Page](#) online. (*continued*)

**Table 1 • Additional Documents**

Document	Description
PB0115: SmartFusion2 System-on-Chip FPGAs Product Brief and PB0121: IGLOO2 FPGA Product Brief	This product brief provides an overview of SmartFusion2 and IGLOO2 family, features, and development tools.
DS0128: IGLOO2 and SmartFusion2 Datasheet	This datasheet contains SmartFusion2 and IGLOO2 DC and switching characteristics.
DS0124: IGLOO2 Pin Descriptions Datasheet	This document contains IGLOO2 pin descriptions, package outline drawings, and links to pin tables in Excel format.
DS0115: SmartFusion2 Pin Descriptions Datasheet	This document contains SmartFusion2 pin descriptions, package outline drawings, and links to pin tables in Excel format.
UG0445: IGLOO2 FPGA and SmartFusion2 SoC FPGA Fabric User Guide	SmartFusion2 and IGLOO2 FPGAs integrate fourth generation flash-based FPGA fabric. The FPGA fabric is comprised of Logic Elements which consist of a 4 input look up table (LUT), includes embedded memories and Mathblocks for DSP processing capabilities. This document describes the SmartFusion2 and IGLOO2SmartFusion2 and IGLOO2 FPGA fabric architecture, embedded memories, Mathblocks, fabric routing, and I/Os.
UG0331: SmartFusion2 Microcontroller Subsystem User Guide	SmartFusion2 devices integrate a hard microcontroller subsystem (MSS). The MSS consists of a ARM Cortex-M3 processor with embedded trace macrocell (ETM), instruction cache, embedded memories, DMA engines, communication peripherals, timers, real-time counter (RTC), general purpose I/Os, and FPGA fabric interfaces. This document describes the SmartFusion2 MSS and its internal peripherals.
UG0448: IGLOO2 High Performance Memory Subsystem User Guide	IGLOO2 devices integrate a hard high performance memory subsystem (HPMS) consists of embedded memories, DMA engines, and FPGA fabric interfaces. This document describes the IGLOO2 HPMS and its internal peripherals.

**Table 1 • Additional Documents (continued)**

Document	Description
UG0447: IGLOO2 and SmartFusion2 High Speed Serial Interfaces User Guide	SmartFusion2 and IGLOO2 devices integrate hard high-speed serial interfaces (PCIe, XAUI/XGXS, SERDES). This document describes the SmartFusion2 and IGLOO2 SmartFusion2 and IGLOO2 high-speed serial interfaces.
UG0449: SmartFusion2 and IGLOO2 Clocking Resources User Guide	SmartFusion2 and IGLOO2 clocking resources include on-chip oscillators, FPGA fabric global network, and clock conditioning circuitry (CCCs) with dedicated phase-locked loops (PLLs). These clocking resources provide flexible clocking schemes to the on-chip hard IP blocks—HPMS, fabric DDR (FDDR) subsystem, and high-speed serial interfaces (PCIe, XAUI/XGXS, SERDES)—and logic implemented in the FPGA fabric.
UG0444: SmartFusion2 and IGLOO2 Low Power Design User Guide	In addition to low static power consumption during normal operation, the SmartFusion2 and IGLOO2 devices support an ultra-low-power Static mode (Flash*Freeze mode) with power consumption less than 1 mW. Flash*Freeze mode retains all the SRAM and register data which enables fast recovery to Active mode. This document describes the SmartFusion2 and IGLOO2 Flash*Freeze mode entry and exit mechanisms.
UG0443: SmartFusion2 and IGLOO2 FPGA Security and Reliability User Guide	The SmartFusion2 and IGLOO2 devices incorporate essentially all the security features that made third generation Microsemi SoC devices the gold standard for security in the PLD industry. Also included are unique design and data security features and use models new to the PLD industry. SmartFusion2 and IGLOO2 flash-based FPGA fabric has zero FIT configuration rate due to its single event upset (SEU) immunity, which is critical in reliability applications. This document describes the SmartFusion2 and IGLOO2 security features and error detection and correction (EDAC) capabilities.
UG0450: SmartFusion2 SoC and IGLOO2 FPGA System Controller User Guide	The system controller manages programming of the SmartFusion2 and IGLOO2 devices and handles system service requests. The subsystems, interfaces, and system services in the system controller are discussed in this user guide.
UG0450: SmartFusion2 SoC and IGLOO2 FPGA System Controller User Guide	Describes different programming modes supported in the SmartFusion2 and IGLOO2 devices. High level schematics of these programming methods are also provided as a reference. Important board-level considerations are discussed.
Libero SoC User Guide	Libero <sup>®</sup> System-on-Chip (SoC) is the most comprehensive and powerful FPGA design and development software available, providing start-to-finish design flow guidance and support for novice and experienced users alike. Libero SoC combines Microsemi SoC Products Group tools with such EDA powerhouses as Synplify and ModelSim. This user guide discusses the usage of the software and design flow.

## 3 MDDR Subsystem

---

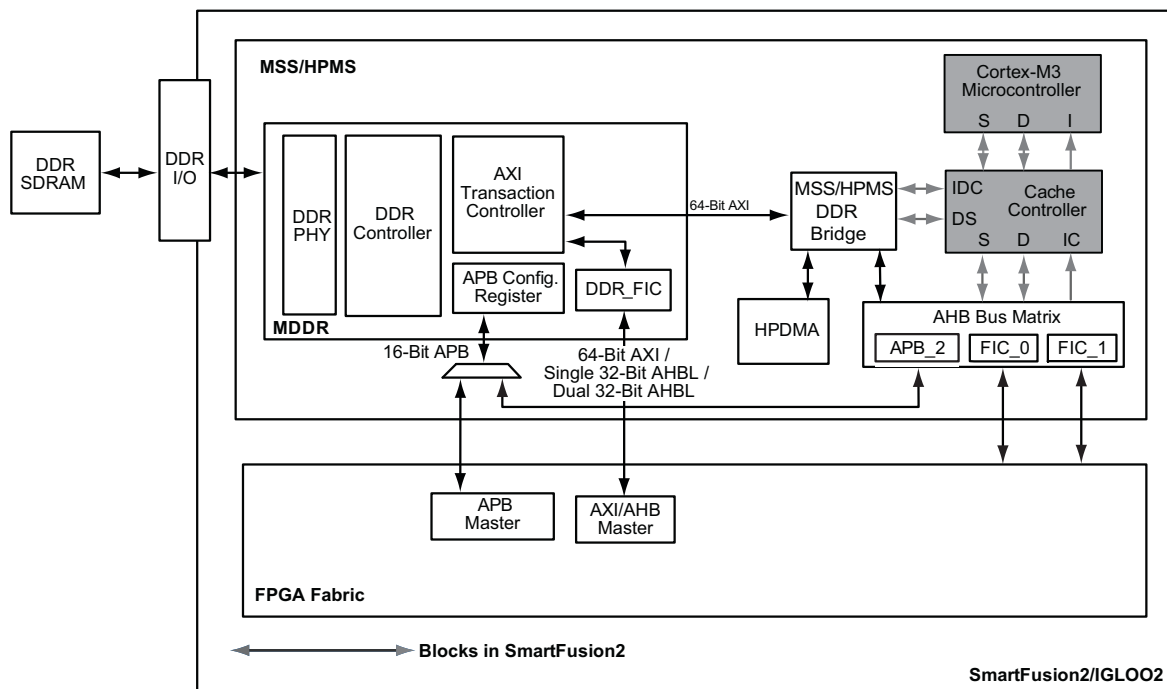
The MDDR is a hardened ASIC block for interfacing the DDR2, DDR3, and LPDDR1 memories. The MDDR subsystem is used to access DDR memories for high-speed data transfers and code execution., and includes a DDR memory controller, DDR PHY, and arbitration logic to support multiple masters. DDR memory connected to the MDDR subsystem can be accessed by the MSS/HPMS masters and master logic implemented in the FPGA fabric (FPGA fabric master).

The MSS/HPMS masters communicate with the MDDR subsystem through an MSS/HPMS DDR bridge that provides an efficient access path. FPGA fabric masters communicate with the MDDR subsystem through AXI or AHB interfaces.

### 3.1 Features

- Integrated on-chip DDR memory controller and PHY
- Capable of supporting LPDDR1, DDR2, and DDR3 memory devices
- Up to 667 Mbps (333.33 MHz DDR) performance
- Supports memory densities upto 2 GB
- Supports 8/16/32-bit DDR standard dynamic random access memory (SDRAM) data bus width modes
- Supports a maximum of 8 memory banks
- Supports single rank memory
- Single error correction and double error detection (SECDED) enable/disable feature
- Supports DRAM burst lengths of 4, 8, or 16, depending on the bus-width mode and DDR type configuration
- Support for sequential and interleaved burst ordering
- Programs internal control for ZQ short calibration cycles for DDR3 configurations
- Supports dynamic scheduling to optimize bandwidth and latency
- Supports self refresh entry and exit on command
- Supports deep power-down entry and exit on command
- Flexible address mapper logic to allow application specific mapping of row, column, bank, and rank bits
- Configurable support for 1T or 2T timing on the DDR SDRAM control signals
- Supports autonomous DRAM power-down entry and exit caused by lack of transaction arrival for programmable time

The following illustration shows the system level block diagram of the MDDR subsystem.

**Figure 1 • System Level MDDR Block Diagram**

The MDDR subsystem accepts data transfer requests from AXI or AHB interfaces. Any read/write transactions to the DDR memories can occur from the following four paths:

- High performance DMA (HPDMA) controller can access DDR memories through the MSS/HPMS DDR bridge for high speed data transactions.
- Other MSS/HPMS masters (for example, FIC\_0, FIC\_1, and PDMA) can access DDR memories through the MSS/HPMS DDR bridge.
- AXI or AHBL masters in the FPGA fabric can access DDR memories through DDR\_FIC interface.

**Note:** The Cortex-M3 processor can access DDR memories through the MSS DDR bridge for data and code execution in SmartFusion2.

**Note:** The maximum DDR3 data rate supported by MDDR is 333MHz/667Mbps. Therefore, Write Leveling is not mandatory and the interface works if the board layout includes length matching and follows [AC393 SmartFusion2 and IGLOO2 Board Design Guidelines Application Note](#). For Read Leveling, Libero SOC auto-generates pre-defined static delay ratios for MDDR initialization. These delay values are sufficient if the board layout follows the SmartFusion2/IGLOO2 board-level guidelines.

## 3.2 Memory Configurations

The SmartFusion2 and IGLOO2 FPGA MDDR subsystem supports a wide range of common memory types, configurations, and densities, as shown in the following table. If SECCED mode is enabled in the MDDR controller, the external memory module must be connected to the following:

- Data lines MDDR\_DQ\_ECC[3:0] when data width is x32
- Data lines MDDR\_DQ\_ECC[1:0] when data width is x16

- Data line MDDR\_DQ\_ECC[0] when data width is x8

**Table 2 • Supported Memory (DDR2, DDR3 and LPDDR1) Configurations**

Memory Depth	Width	Width (in SECEDED Mode)	SmartFusion2 and IGLOO2 Devices			
			M2S/M2GL 005/010/025/060/090 M2S/M2GL150-FCV484	M2S/M2GL 050 (FCS325, VF400, FG484)	M2S/M2GL 050 (FG896)	M2S/M2GL150(FC1152)
128M or Less	x32	x36	–	–	✓	✓
	x16	x18	✓	✓	✓	✓
	x8	x9	✓	–	–	✓
256M	x32	x36	–	–	✓	✓
	x16	x18	✓	✓	✓	✓
	x8	x9	✓	–	–	✓
512M	x32	x36	–	–	✓	✓
	x16	x18	✓	✓	✓	✓
	x8	x9	✓	–	–	✓
1G	x32	x36	–	–	✓	✓
	x16	x18	✓	✓	✓	✓
	x8	x9	✓	–	–	✓

### 3.3 Performance

The following table shows the maximum data rates supported by MDDR subsystem for supported memory types.

For more Information, refer to the "DDR Memory Interface Characteristics" section in [DS0128: IGLOO2 FPGA and SmartFusion2 SoC FPGA Datasheet](#).

**Table 3 • DDR Speeds**

Memory Type	Maximum Data Rate (Mbps)
LPDDR1	400 Mbps (200 MHz)
DDR2	667 Mbps (333.33 MHz)
DDR3	667 Mbps (333.33 MHz)

### 3.4 I/O Utilization

The following table lists the I/O utilization for the SmartFusion2 and IGLOO2 devices corresponding to supported bus widths. The remaining I/Os in Bank 0 can be used for general purposes.

**Table 4 • I/O Utilization for SmartFusion2 and IGLOO2 Devices**

MDDR Bus Width	M2S/M2GL005/010/025/060/090 M2S/M2GL150-FCV484	M2S/M2GL 050 (FCS325, VF400, FG484)	M2S/M2GL 050 (FG896)	M2S/M2GL 150 (FC1152)
36-bit	–	–	Bank0 (85 pins)	Bank2 (85 pins)
32-bit	–	–	Bank0 (76 pins)	Bank2 (76 pins)
18-bit	Bank0 (59 pins)	Bank0 (59 pins)	Bank0 (59 pins)	Bank2 (59 pins)

**Table 4 • I/O Utilization for SmartFusion2 and IGLOO2 Devices**

16-bit	Bank0 (53 pins)	Bank0 (53 pins)	Bank0 (53 pins)	Bank2 (53 pins)
9-bit	Bank0 (47 pins)	–	–	Bank2 (47 pins)
8-bit	Bank0 (41 pins)	–	–	Bank2 (41 pins)

**Note:** If MDDR is configured for LPDDR, one more IO also available for every 8-bit as the LPDDR does not have DQS\_N.

For general purpose use of the unused I/Os in the MDDR bank, select one of the I/O standards with the same voltage level as the DDR I/Os.

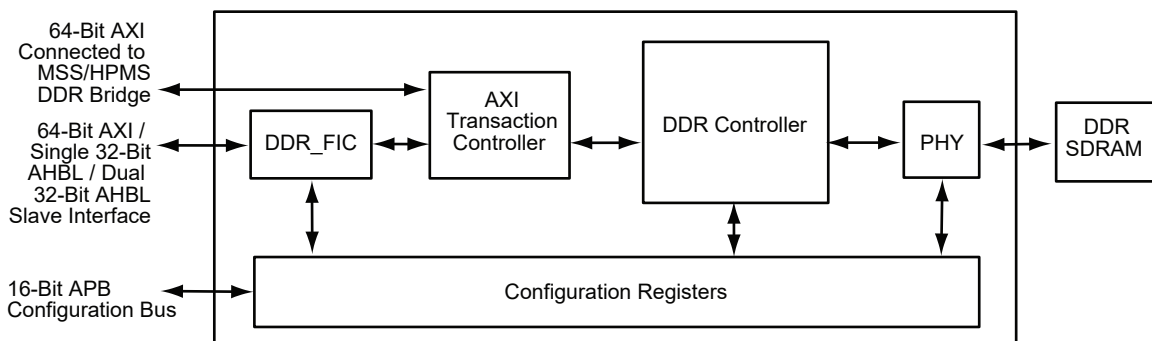
Self refresh must be disabled if the MDDR banks contain a mixed of I/Os used for DDR and for general purpose fabric I/Os. For more information, see ["Self Refresh \(DDR2, DDR3, LPDDR1\)" on page 24](#).

## 3.5 Functional Description

This section provides the functional description of the MDDR subsystem.

### 3.5.1 Architecture Overview

The following illustration shows a functional block diagram of the MDDR subsystem. The main components include the DDR fabric interface controller (DDR\_FIC), AXI transaction handler, DDR memory controller, and DDR PHY.

**Figure 2 • MDDR Subsystem Functional Block Diagram**

The DDR\_FIC facilitates communication between the FPGA fabric masters and AXI transaction controller. The DDR\_FIC can be configured to provide either one 64-bit AXI slave interface or two independent 32-bit AHB-Lite (AHBL) slave interfaces to the FPGA fabric masters.

The AXI transaction controller receives read and write requests from AXI masters (MSS/HPMS DDR bridge and DDR\_FIC) and schedules for the DDR controller by translating them into DDR controller commands.

The DDR controller receives the commands from the AXI transaction controller. These commands are queued internally and scheduled for access to the DDR SDRAM while satisfying DDR SDRAM constraints, transaction priorities, and dependencies between the transactions. The DDR controller in turn issues commands to the PHY module, which launches and captures data to and from the DDR SDRAM.

DDR PHY receives commands from the DDR controller and generates DDR memory signals required to access the external DDR memory.

The 16-bit APB configuration bus provides an interface to configure the MDDR subsystem registers. The MDDR subsystem operates on MDDR\_CLK. MSS/HPMS CCC generates the MDDR\_CLK using MPLL. For more details on MSS/HPMS CCC refer *UG0449: SmartFusion2 and IGLOO2 Clocking Resources User Guide*.

### 3.5.2 Port List

**Table 5 • MDDR Subsystem Interface Signals**

Signal Name	Type	Polarity	Description
APB_S_PCLK	In	–	APB clock. This clock drives all the registers of the APB interface.
APB_S_PRESET_N	In	Low	APB reset signal. This is an active low signal. This drives the APB interface and is used to generate the soft reset for the DDR controller as well.
MDDR_DDR_CORE_RESET_N	In	Low	Global reset. This resets the DDR_FIC/DDRC/PHY/DDRAXI logic.
MDDR_DDR_AXI_S_RMW	In	High	AXI mode only Indicates whether all bytes of a 64-bit lane are valid for all beats of an AXI transfer. 0: Indicates that all bytes in all beats are valid in the burst and the controller should default to write commands. 1: Indicates that some bytes are invalid and the controller should default to RMW commands. This is classed as an AXI write address channel sideband signal and is valid with the AWVALID signal.
HPMS_DDR_FIC_SUBSYSTEM_CLK or, MSS_DDR_FIC_SUBSYSTEM_CLK	Out	–	This output clock is derived from the MDDR_CLK and is based on the DDR_FIC divider ratio. This is the clock that should be used for the AXI or AHB slave interfaces to move data in and out of the MDDR.
HPMS_DDR_FIC_SUBSYSTEM_LOCK or, MSS_DDR_FIC_SUBSYSTEM_LOCK	Out	–	This indicates the lock from FCCC which generates HPMS_DDR_FIC_SUBSYSTEM_CLK for IGLOO2 and MSS_DDR_FIC_SUBSYSTEM_LOCK in SmartFusion2.
<b>Bus Interfaces</b>			
AXI_SLAVE <sup>1</sup>	Bus	–	AXI slave interface 1.0 bus
AHB0_SLAVE <sup>2</sup>	Bus	–	AHB0 slave interface 3.0 bus
AHB1_SLAVE <sup>3</sup>	Bus	–	AHB1 slave interface 3.0 bus
APB_SLAVE	Bus	–	APB slave interface 3.0 bus
<b>DRAM Interface</b>			
MDDR_CAS_N	Out	Low	DRAM CASN
MDDR_CKE	Out	High	DRAM CKE
MDDR_CLK	Out	–	DRAM single-ended clock – for differential pads
MDDR_CLK_N	Out	–	DRAM single-ended clock – for differential pads
MDDR_CS_N	Out	Low	DRAM CSN
MDDR_ODT	Out	High	DRAM ODT. 0: Termination Off 1: Termination On
MDDR_RAS_N	Out	Low	DRAM RASN
MDDR_RESET_N	Out	Low	DRAM reset for DDR3
MDDR_WE_N	Out	Low	DRAM WEN



**Table 5 • MDDR Subsystem Interface Signals (continued)**

Signal Name	Type	Polarity	Description
MDDR_ADDR[15:0]	Out	–	Dram address bits
MDDR_BA[2:0]	Out	–	Dram bank address
MDDR_DM_RDQS[3:0]	In/out	–	DRAM data mask – from bidirectional pads
MDDR_DQS[3:0]	In/out	–	DRAM single-ended data strobe output – for bidirectional pads
MDDR_DQS_N[3:0]	In/out	–	DRAM single-ended data strobe output – for bidirectional pads
MDDR_DQ[31:0]	In/out	–	DRAM data input/output – for bidirectional pads
MDDR_DQ_ECC[3:0]	In/out	–	DRAM data input/output for SECEDED
MDDR_DM_RDQS_ECC	In/out	High	DRAM single-ended data strobe output – for bidirectional pads
MDDR_DQS_ECC	In/out	High	DRAM single-ended data strobe output – for bidirectional pads
MDDR_DQS_ECC_N	In/out	Low	DRAM data input/output – for bidirectional pads
MDDR_DQS_TMATCH_0_IN	In	High	DQS enables input for timing match between DQS and system clock. For simulations, tie to MDDR_DQS_TMATCH_0_OUT.
MDDR_DQS_TMATCH_1_IN	In	High	DQS enables input for timing match between DQS and system clock. For simulations, tie to MDDR_DQS_TMATCH_1_OUT.
MDDR_DQS_TMATCH_0_OUT	Out	High	DQS enables output for timing match between DQS and system clock. For simulations, tie to MDDR_DQS_TMATCH_0_IN.
MDDR_DQS_TMATCH_1_OUT	Out	High	DQS enables output for timing match between DQS and system clock. For simulations, tie to MDDR_DQS_TMATCH_1_IN.
MDDR_DQS_TMATCH_ECC_IN	In	High	DQS enables input for timing match between DQS and system clock. For simulations, tie to MDDR_DQS_TMATCH_ECC_OUT.
MDDR_DQS_TMATCH_ECC_OUT	Out	High	DQS enables output for timing match between DQS and system clock. For simulations, tie to MDDR_DQS_TMATCH_ECC_IN.

**Note:** <sup>1</sup> AXI or AHB interface, depending on configuration.

<sup>2</sup> MDDR\_DQS\_N[3:0] signals are not available for LPDDR.

<sup>3</sup> TMATCH\_IN and TMATCH\_OUT pins are required to be connected together outside the device. They are used for gate training as part of the read data capture operation. The two pins create an internal DQS Enable signal that is used to calibrate the flight path. DQS needs to be gated to prevent false triggering of the FIFO write clock. This DQS Enable signal is derived from the system clock and physically matches the clock output buffer and DQS input buffer to compensate for I/O buffer uncertainty due to Process-Voltage-Temperature (PVT) changes. Without this connection, the circuit is not operable.

### 3.5.2.1 AXI Slave Interface

The following table describes the MDDR AXI slave interface signals. These signals will be available only if the MDDR interface is configured for AXI mode. For more AXI protocol details, refer to AMBA AXI v1.0 protocol specification.

**Table 6 • AXI Slave Interface Signals**

Signal Name	Direction	Polarity	Description
MDDR_DDR_AXI_S_ARREADY	Output	High	Indicates whether or not the slave is ready to accept an address and associated control signals. 1: Slave ready 0: Slave not ready
MDDR_DDR_AXI_S_AWREADY	Output	High	Indicates that the slave is ready to accept an address and associated control signals. 1: Slave ready 0: Slave not ready
MDDR_DDR_AXI_S_BID[3:0]	Output		Indicates response ID. The identification tag of the write response.
MDDR_DDR_AXI_S_BRESP[1:0]	Output		Indicates write response. This signal indicates the status of the write transaction. 00: Normal access okay 01: Exclusive access okay 10: Slave error 11: Decode error
MDDR_DDR_AXI_S_BVALID	Output	High	Indicates whether a valid write response is available. 1: Write response available 0: Write response not available
MDDR_DDR_AXI_S_RDATA[63:0]	Output		Indicates read data.
MDDR_DDR_AXI_S_RID[3:0]	Output		Read ID tag. This signal is the ID tag of the read data group of signals.
MDDR_DDR_AXI_S_RLAST	Output	High	Indicates the last transfer in a read burst.
MDDR_DDR_AXI_S_RRESP[1:0]	Output		Indicates read response. This signal indicates the status of the read transfer. 00: Normal access 01: Exclusive access 10: Slave error 11: Decode error
MDDR_DDR_AXI_S_RVALID	Output		Indicates whether the required read data is available and the read transfer can complete. 1: Read data available 0: Read data not available
MDDR_DDR_AXI_S_WREADY	Output	High	Indicates whether the slave can accept the write data. 1: Slave ready 0: Slave not ready

**Table 6 • AXI Slave Interface Signals (continued)**

Signal Name	Direction	Polarity	Description
MDDR_DDR_MDDR_DDR_AXI_S_ARADDR[31:0]	Input		Indicates initial address of a read burst transaction. <b>Note:</b> DDR_FIC AXI interface supports only 64-bit aligned addresses.
MDDR_DDR_AXI_S_ARBURST[1:0]	Input		Indicates burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. 00: FIXED - Fixed-address burst FIFO type (Not Supported) 01: INCR - Incrementing-address burst normal sequential memory 10: WRAP - Incrementing-address burst that wraps to a lower address at the wrap boundary 11: Reserved
MDDR_DDR_AXI_S_ARID[3:0]	Input		Indicates identification tag for the read address group of signals.
MDDR_DDR_AXI_S_ARLEN[3:0]	Input		Indicates burst length. The burst length gives the exact number of transfers in a burst. 0000: 1 0001: 2 0010: 3 0011: 4 0100: 5 0101: 6 0110: 7 0111: 8 1000: 9 1001: 10 1010: 11 1011: 12 1100: 13 1101: 14 1110: 15 1111: 16
MDDR_DDR_AXI_S_ARLOCK[1:0]	Input		Indicates lock type. This signal provides additional information about the atomic characteristics of the read transfer. 00: Normal access 01: Exclusive access 10: Locked access 11: Reserved
MDDR_DDR_AXI_S_ARSIZE[1:0]	Input		Indicates the maximum number of data bytes to transfer in each data transfer, within a burst. 00: 10 : Not Supported 11: 8

**Table 6 • AXI Slave Interface Signals (continued)**

Signal Name	Direction	Polarity	Description
MDDR_DDR_AXI_S_ARVALID	Input	High	Indicates the validity of read address and control information. 1: Address and control information valid 0: Address and control information not valid
MDDR_DDR_AXI_S_AWADDR[31:0]	Input		Indicates write address. The write address bus gives the address of the first transfer in a write burst transaction. <b>Note:</b> DDR_FIC AXI interface supports only 64-bit aligned addresses.
MDDR_DDR_AXI_S_AWBURST[1:0]	Input		Indicates burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. 00: FIXED - Fixed-address burst FIFO-type (Not Supported) 01: INCR - Incrementing-address burst normal sequential memory 10: WRAP - Incrementing-address burst that wraps to a lower address at the wrap boundary 11: Reserved
MDDR_DDR_AXI_S_AWID[3:0]	Input		Indicates identification tag for the write address group of signals.
MDDR_DDR_AXI_S_AWLEN[3:0]	Input		Indicates burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. 0000: 1 0001: 2 0010: 3 0011: 4 0100: 5 0101: 6 0110: 7 0111: 8 1000: 9 1001: 10 1010: 11 1011: 12 1100: 13 1101: 14 1110: 15 1111: 16
MDDR_DDR_AXI_S_AWLOCK[1:0]	Input		Indicates lock type. This signal provides additional information about the atomic characteristics of the write transfer. 00: Normal access 01: Exclusive access 10: Locked access 11: Reserved

**Table 6 • AXI Slave Interface Signals (continued)**

Signal Name	Direction	Polarity	Description
MDDR_DDR_AXI_S_AWSIZE[1:0]	Input		Indicates the maximum number of data bytes to transfer in each data transfer, within a burst. 00 to 10 : Not Supported 11: 8
MDDR_DDR_AXI_S_AWVALID	Input	High	Indicates whether or not valid write address and control information are available. 1: Address and control information available 0: Address and control information not available
MDDR_DDR_AXI_S_BREADY	Input	High	Indicates whether or not the master can accept the response information. 1: Master ready 0: Master not ready
MDDR_DDR_AXI_S_RREADY	Input	High	Indicates whether or not the master can accept the read data and response information. 1: Master ready 0: Master not ready
MDDR_DDR_AXI_S_WDATA[63:0]	Input		Indicates write data.
MDDR_DDR_AXI_S_WID[3:0]	Input		Indicates response ID. The identification tag of the write response.
MDDR_DDR_AXI_S_WLAST	Input	High	Indicates the last transfer in a write burst.
MDDR_DDR_AXI_S_WSTRB[7:0]	Input		Indicates which byte lanes to update in memory.
MDDR_DDR_AXI_S_WVALID	Input	High	Indicates whether or not valid write data and strobes are available. 1: Write data and strobes available 0: Write data and strobes not available

### 3.5.2.2 AHB Slave Interface

The following table describes the MDDR AHB slave interface signals. These signals are available only if MDDR interface is configured for single or dual AHB mode. For more AHB protocol details, refer to AMBA AHB v3.0 protocol specification.

**Table 7 • AHB Slave Interface Signals**

Signal Name	Direction	Polarity	Description
MDDR_DDR_AHBx_S_HREADYOUT	Output	High	Indicates that a transfer has finished on the bus. The signal is asserted Low to extend a transfer. Input to Fabric master.
MDDR_DDR_AHBx_S_HRESP	Output	High	Indicates AHB transfer response to Fabric master.
MDDR_DDR_AHBx_S_HRDATA[31:0]	Output		Indicates AHB read data to Fabric master.
MDDR_DDR_AHBx_S_HSEL	Input	High	Indicates AHB slave select signal from Fabric master.

**Table 7 • AHB Slave Interface Signals (continued)**

Signal Name	Direction	Polarity	Description
MDDR_DDR_AHBx_S_HADDR[31:0]	Input		Indicates AHB address initiated by Fabric master.
MDDR_DDR_AHBx_S_HBURST[2:0]	Input		Indicates AHB burst type from Fabric master. 000: Single burst 001: Incrementing burst of undefined length 010: 4-beat wrapping burst 011: 4-beat incrementing burst 100: 8-beat wrapping burst 101: 8-beat incrementing burst 110: 16-beat wrapping burst 111: 16-beat incrementing burst
MDDR_DDR_AHBx_S_HSIZE[1:0]	Input		Indicates AHB transfer size from Fabric master. 00: 8 Byte 01: 16 Halfword 10: 32 Word
MDDR_DDR_AHBx_S_HTRANS[1:0]	Input		Indicates AHB transfer type from Fabric master. 00: IDLE 01: BUSY 10: NONSEQUENTIAL 11: SEQUENTIAL.
MDDR_DDR_AHBx_S_HMASTLOCK	Input	High	Indicates AHB master lock signal from Fabric master.
MDDR_DDR_AHBx_S_HWRITE	Input	High	Indicates AHB write control signal from Fabric master.
MDDR_DDR_AHBx_S_HREADY	Input	High	Indicates that a transfer has finished on the bus. Fabric master can drive this signal Low to extend a transfer.
MDDR_DDR_AHBx_S_HWDATA[31:0]	Input		Indicates AHB write data from Fabric master.

**Note:** AHBx indicates AHB0 or AHB1.

### 3.5.2.3 APB Slave Interface

The following table describes the MDDR APB slave interface signals. For more APB protocol details, refer to AMBA APB v3.0 protocol specification.

**Table 8 • MDDR APB Slave Interface Signals**

Signal Name	Direction	Polarity	Description
MDDR_APB_S_PREADY	Output	High	Indicates APB Ready signal to Fabric master.
MDDR_APB_S_PSLVERR	Output	High	Indicates error condition on an APB transfer to Fabric master.
MDDR_APB_S_PRDATA[15:0]	Output		Indicates APB read data to Fabric master.
MDDR_APB_S_PENABLE	Input	High	Indicates APB enable from Fabric master. The enable signal is used to indicate the second cycle of an APB transfer.
MDDR_APB_S_PSEL	Input	High	Indicates APB slave select signal from Fabric master

**Table 8 • MDDR APB Slave Interface Signals (continued)**

Signal Name	Direction	Polarity	Description
MDDR_APB_S_PWRITE	Input	High	Indicates APB write control signal from Fabric master
MDDR_APB_S_PADDR[10:2]	Input		Indicates APB address initiated by Fabric master.
MDDR_APB_S_PWDATA[15:0]	Input		Indicates APB write data from Fabric master.

### 3.5.3 Initialization

After power-up, the MDDR needs to have all of the configuration registers written to establish the operating modes of the blocks. When using the System Builder design flow through Libero SoC, this is all handled for the user through the use of the System Builder module. All of the configuration register values are selected by the user and stored in a special portion of the embedded non-volatile memory (eNVM). Before the MDDR subsystem is active, it goes through an initialization phase and this process starts with a reset sequence. For DDR3 memories, the initialization phase also includes ZQ calibration and DRAM training.

#### 3.5.3.1 Reset Sequence

The following illustration shows the reset sequence for the MDDR subsystem from the power on reset stage. The MDDR subsystem comes out of reset after MPLL Lock is asserted by the MSS/HPMS\_CCC. De-assertion of MDDR\_AXI\_RESET\_N signifies the end of the reset sequence. The MDDR reset can be generated by asserting MDDR\_CTLR\_SOFTRESET bit in [SOFT\\_RESET\\_CR](#) to 1. The DDR controller performs external DRAM memory reset and initialization as per the JEDEC specification, including reset, refresh, and mode registers.

#### 3.5.3.2 DDRIO Calibration

Each DDRIO has an ODT feature, which is calibrated depending on the DDR I/O standard. DDR I/O calibration occurs after the DDR I/Os are enabled. If the impedance feature is enabled, impedance can be programmed to the desired value in three ways:

- Calibrate the ODT/driver impedance with a calibration block (recommended)
- Calibrate the ODT/driver impedance with fixed calibration codes
- Configure the ODT/driver impedance to the desired value directly

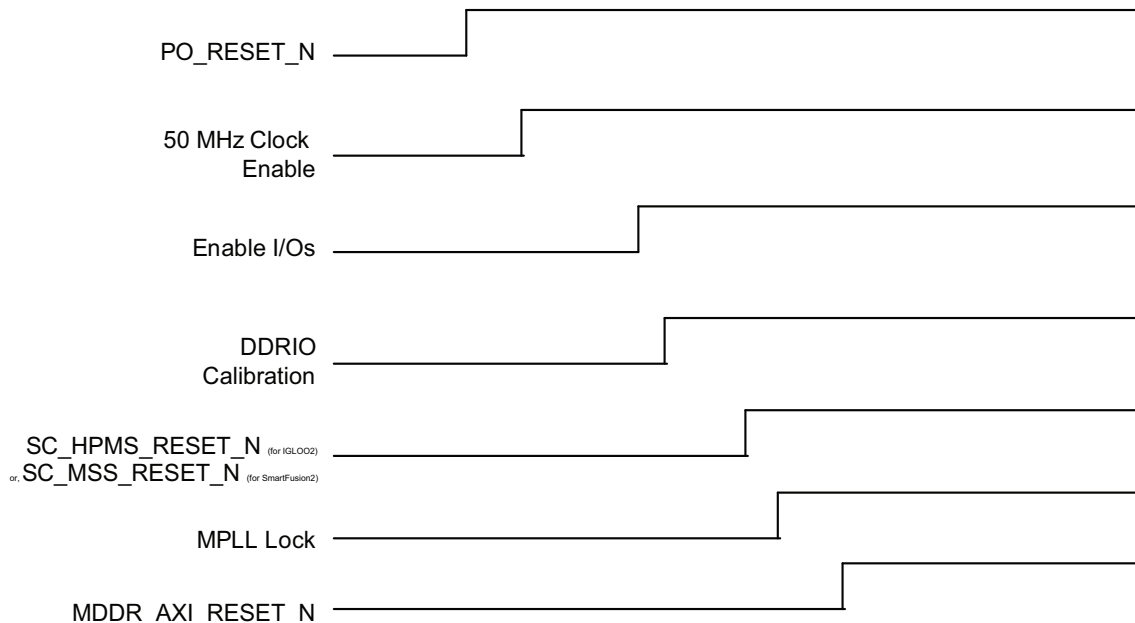
The system register, MDDR\_IO\_CALIB\_CR, can be configured for changing the ODT value to the desired value.

The I/O calibration is always enabled when the DDR subsystem is configured for DDR2 and DDR3 memories.

The I/O calibration can be disabled or enabled using the DDR configurator when the DDR subsystem is configured for LPDDR memories.

**Note:** If I/O calibration is enabled, all I/Os in the DDR bank are calibrated even though the DDR controller is not using all I/Os in the bank.

For more information on DDR I/O calibration, refer to the Configurable ODT and Driver Impedance section of the I/Os chapter in the [UG0445: IGLOO2 FPGA and SmartFusion2 SoC FPGA Fabric User Guide](#).

**Figure 3 • Reset Sequence**

### 3.5.3.3 ZQ Calibration

This is applicable for DDR3 only. The ZQ calibration command is used to calibrate DRAM output drivers ( $R_{ON}$ ) and on-die termination (ODT) values. The DDR3 SDRAM needs a longer time to calibrate  $R_{ON}$  and ODT at initialization and a relatively smaller time to perform periodic calibrations.

The DDR controller performs ZQ calibration by issuing a ZQ calibration long (ZQCL) command and ZQ calibration short (ZQCS) command.

ZQCL is used to perform initial calibration during the power-up initialization sequence. This command is allowed for a period of  $t_{ZQinit}$ , as specified by memory vendor. The value of  $t_{ZQinit}$  can be modified through register bits [REG\\_DDRC\\_T\\_ZQ\\_LONG\\_NOP](#).

The ZQCS command is used to perform periodic calibration to account for voltage and temperature variations. A shorter timing window is provided to perform calibration and transfer of values as defined by timing parameter  $t_{ZQCS}$ . The  $t_{ZQCS}$  parameter can be modified through register bits [REG\\_DDRC\\_T\\_ZQ\\_SHORT\\_NOP](#).

Other activities are not performed by the controller for the duration of  $t_{ZQinit}$  and  $t_{ZQCS}$ . All DRAM banks are precharged and  $t_{RP}$  is met before ZQCL or ZQCS commands are issued by the DDR controller.

### 3.5.3.4 DRAM Training

High Speed DDR3 memories typically requires the DDR controller to implement Write-Leveling, Read DQS Gate Training, and Read Data Eye Training. However, MDDR only supports a maximum data rate of 333 MHz/667 Mbps, which means the clock period and data window are relatively large compared to high-speed DDR3 memory interfaces. Therefore dynamic write-leveling and read training are not performed. The following sections describe how write-leveling and read training are addressed by the MDDR.

#### 3.5.3.4.1 Write Leveling

Dynamic write-leveling is not required for the MDDR controller. The board-layout needs to follow [AC393 SmartFusion2 and IGLOO2 Board Design Guidelines Application Note](#) to keep the skew between DQS and CK within the JEDEC DDR3  $t_{DQSS}$  limit of +/- 750ps at each memory device. For board layouts which do not meet the Board Design Guidelines, the MDDR controller allows static delay ratios which delays DQS for each byte lane so that the skew between DQS and CK is kept within JEDEC limits.

333 MHz/667 Mbps is the maximum DDR3 rate MDDR supports. Leveling is not mandatory and the interface will work if the board layout guidelines are followed and length matching is done.



### 3.5.3.4.2 Read Leveling

MDDR does not perform dynamic Read DQS Gate Training and Data Eye Training. Instead, these functions are achieved by using built-in static delay values automatically generated by Libero SoC for the MDDR automatic register initialization.

### 3.5.3.4.3 Read Gate

The DQS gate is aligned by the Libero SoC auto-generated MDDR initialization code containing fixed delay ratios to account for board round-trip time between FPGA and the DDR3 memory. The TMATCH\_OUT and TMATCH\_IN signals are shorted close to the FPGA balls to remove the FPGA output and input delays from the round trip delay time. Therefore, the fixed delay ratios represent only the board delay.

The fixed delay ratios work in combination with board layouts which follow the SmartFusion2/IGLOO2 Board Design Guidelines (refer [AC393: Board Design Guidelines for SmartFusion2/IGLOO2 FPGA Application Note](#)).

### 3.5.3.4.4 DQS Alignment within Data Eye

The incoming read DQS is internally centered within the read DQ data window using a static delay ratio. This static delay is applied by the Libero SoC auto-generated MDDR initialization code. The fixed delay ratios work in combination with board layouts which follow the SmartFusion2/IGLOO2 Board Design Guidelines (refer [AC393: Board Design Guidelines for SmartFusion2/IGLOO2 FPGA Application Note](#)).

**Note:** The Libero SOC auto-generated delay ratio for read DQS data eye centering is written to the required register.

### 3.5.3.5 DDR Memory Initialization Time

The time to initialize the DDR memory depends on the following factors:

- Power-up and register initialization by system controller. It depends on the power on reset delay configuration in the Libero project (**Project > Project Settings > Device settings**).
- DDR controller and PHY configuration registers initialization. In SmartFusion2 devices, the Cortex-M3 initializes these registers. In IGLOO2 devices, the ConfigMaster in the FPGA fabric initializes these registers.
- DDR memory initialization by the DDR Controller according to the JEDEC standard (mode register configuration and training).
- DDR memory settling time configured in the System Builder memory configuration window.

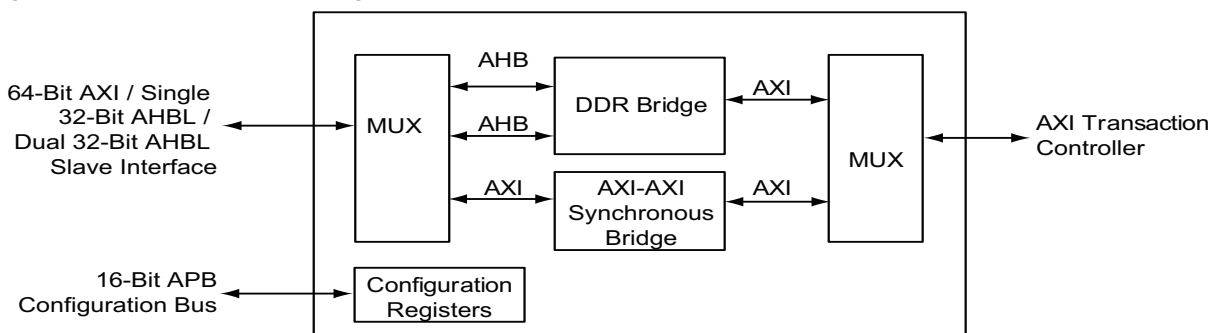
## 3.5.4 Details of Operation

This section provides a functional description of each block in the MDDR subsystem.

### 3.5.4.1 DDR\_FIC

The following illustration shows the DDR\_FIC block diagram.

**Figure 4 • DDR\_FIC Block Diagram**



Fabric masters can access the MDDR subsystem in the following ways:

- Single AXI-64 interface
- Single AHB-32 interface

- Dual AHB-32 bit interfaces

If the AXI-64 interface is selected, the DDR\_FIC acts as an AXI to AXI synchronous bridge. In this mode, DDR\_FIC provides FPGA fabric masters to access the MDDR subsystem through locked transactions. For this purpose, a user configurable 20-bit down counter keeps track of the duration of the locked transfer. If the transfer is not completed before the down counter reaches zero, a single clock cycle pulse interrupt is generated to the fabric interface.

If single or dual AHB-32 interfaces are selected, DDR\_FIC converts the single/dual 32-bit AHB master transactions from the FPGA fabric to 64-bit AXI transactions. In this mode the DDR bridge, embedded as part of the DDR\_FIC, is enabled. The DDR bridge has an arbiter, which arbitrates read and write requests from the two AHB masters on a round robin priority scheme. Refer to the ["DDR Bridge Control Registers in MDDR and FDDR" chapter on page 215](#) for a detailed description.

The DDR\_FIC input interface is clocked by the FPGA fabric clock and the MDDR is clocked by MDDR\_CLK from the MSS/HPMS CCC. Clock ratios between MDDR\_CLK and DDR\_FIC clock can vary. The following table lists supported ratios. Clock ratios can be configured through Libero System-on-Chip (SoC) software or through system register MSSDDR\_FACC1\_CR. For more information, refer to the ["MDDR Configuration Registers" section on page 60](#).

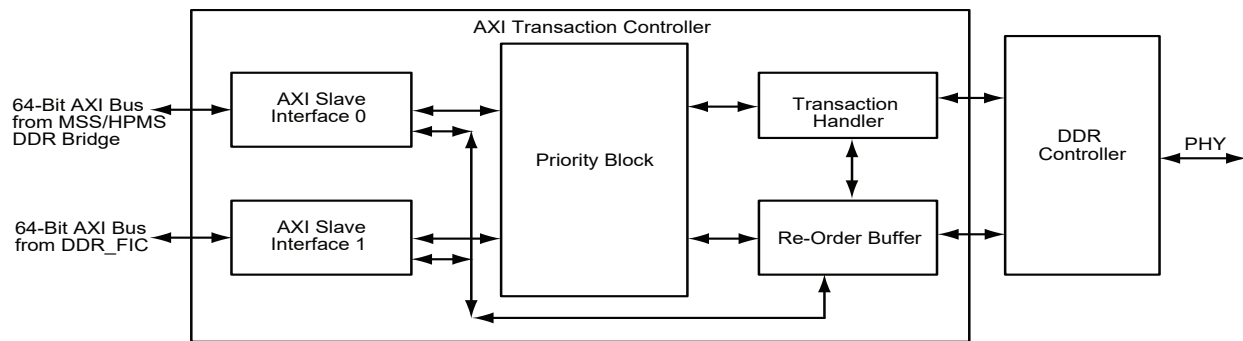
**Table 9 • MDDR\_CLK to FPGA Fabric Clock Ratios**

DIVISOR_A[1:0]	FIC64_DIVISOR[2:0]	MDDR_CLK: FPGA FABRIC Clock Ratio
00	000	1:1
00	001	2:1
00	010	4:1
00	100	8:1
00	101	16:1
01	000	2:1
01	001	4:1
01	010	8:1
01	100	16:1
11	000	3:1
11	001	6:1
11	010	12:1

### 3.5.4.2 AXI Transaction Controller

The AXI transaction controller receives 64-bit AXI transactions from various masters (MSS/HPMS DDR bridge and DDR\_FIC) and translates them into DDR controller transactions. The following illustration shows the block diagram of the AXI transaction controller interfaced with the DDR controller.

The AXI transaction controller performs arbitration of the read/write requests initiated by AXI compliant masters.

**Figure 5 • AXI Transaction Controller Block Diagram**

The AXI transaction controller comprises four major blocks:

- AXI slave interface
- Priority block
- Transaction handler
- Reorder buffer

### 3.5.4.2.1 AXI Slave Interfaces

The AXI transaction controller has two 64-bit AXI slave interfaces: one from the MSS/HPMS DDR bridge and the other from DDR\_FIC. Each of the AXI slave ports is 64 bits wide and is in compliance with the standard AXI protocol. Each transaction has an ID related to the master interface. Transactions with the same ID are completed in order, while the transactions with different read IDs can be completed in any order, depending on when the instruction is executed by the DDR controller. If a master requires ordering between transactions, the same ID should be used.

The AXI slave interface has individual read and write ports. The read port queues read AXI transactions and it can hold up to four read transactions. The write port handles only one write transaction at a time and generates the handshaking signals on the AXI interface.

### 3.5.4.2.2 Priority Block

The priority block prioritizes AXI read/write transactions and provides control to the transaction handler. AXI read transactions have higher priority. The default priority ordering is listed as follows:

1. Reads from the slave port of the MSS/HPMS DDR bridge
2. Reads from the slave port of DDR\_FIC
3. Writes from the slave port of the MSS/HPMS DDR bridge
4. Writes from the slave port of DDR\_FIC

The fabric master through DDR\_FIC can be programmed to have a higher priority by configuring the PRIORITY\_ID and PRIORITY\_ENABLE\_BIT bit fields in the [DDRC\\_AXI\\_FABRIC\\_PRI\\_ID\\_CR](#) register. Priority levels to other masters can be programmed as well, as shown in the following table.

**Table 10 • Priority Level Configuration**

Transactions	Default Priorities (Type-0)	Priorities	
		PRIORITY_ENABLE_BIT=01 (Type 1)	PRIORITY_ENABLE_BIT=10/11 (Type 2/3)
SmartFusion 2			
Reads from I - Cache	1	2	2
Reads from DSG bus	2	3	3
Reads from HPDMA/AHB bus	3	4	4
Reads from Fabric master having the ID as PRIORITY_ID	4	3	1

**Table 10 • Priority Level Configuration**

Writes from DSG bus	5	5	5
Writes from HPDMA/AHB bus	6	7	7
Writes from Fabric master having the ID as PRIORITY_ID	7	6	6
<b>IGLOO2</b>		<b>PRIORITY_ENABLE_BIT=01/10/11 (Type-1/2/3)</b>	
Reads from HPDMA/AHB bus	1		2
Reads from Fabric master having the ID as PRIORITY_ID	2		1
Writes from HPDMA/AHB bus	3		4
Writes from Fabric master having the ID as PRIORITY_ID	4		3

### 3.5.4.2.3 Transaction Handler

The transaction handler converts AXI transactions into DDR controller commands. The transaction handler works on one transaction at a time from the read/write port queue that is selected by the priority block.

The transaction handler has a write command controller and read command controller for write and read transactions.

The write command controller fetches the command from the AXI slave write port and sends a pure write instruction to the DDR controller. If SECDED is enabled, a read modified write (RMW) instruction is sent to the DDR controller.

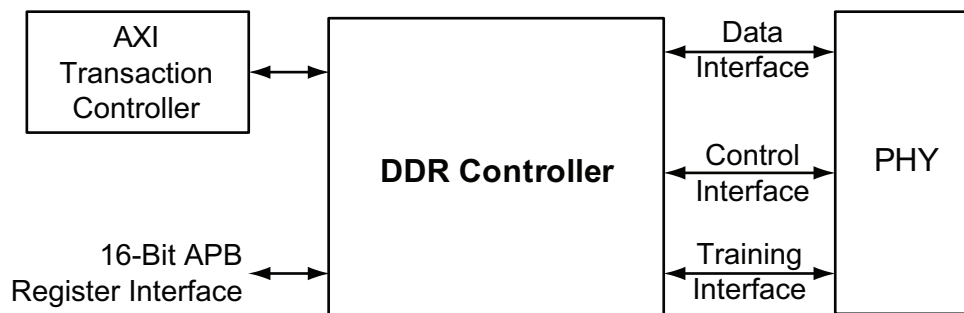
The read command controller generates read transactions to the DDR controller.

### 3.5.4.2.4 Reorder Buffer

The reorder buffer receives data from the DDR controller and orders the data as requested by the AXI master when a single AXI transaction is split into multiple DDR controller transactions, depending on the transfer size.

### 3.5.4.3 DDR Controller

The DDR controller receives requests from the AXI transaction controller, performs the address mapping from system addresses to DRAM addresses (rank, bank, row, and column), and prioritizes requests to minimize the latency of reads (especially high priority reads) and maximize page hits. It also ensures that DRAM is properly initialized, all requests are made to DRAM legally (accounting for associated DRAM constraints), refreshes are inserted as required, and the DRAM enters and exits various power-saving modes appropriately. The following illustration shows the DDR controller connections in the MDDR subsystem.

**Figure 6 • DDR Controller Block Diagram**

The following sections describe key functions of the DDR controller.

### 3.5.4.3.1 Address Mapping

Read and write requests to the DDR controller requires a system address. The controller is responsible for mapping this system address with rank, bank, row, and column address to DRAM.

The address mapper maps linear request addresses to DDR memory addresses by selecting the source bit that maps to each and every applicable DDR memory address bit. The address map interface registers can be configured to map source address bits to DRAM address (for more information, refer to "Address Mapping" section on page 27 in Configuring the MDDR features).

### 3.5.4.3.2 Transaction Scheduling

The DDR controller schedules the read and write transactions to DDR memory. The DDR controller classifies the transactions into three types, based on the commands from the AXI transaction controller:

- Low priority reads (LPR)
- High priority reads (HPR)
- Writes (WR)

Each type of transaction has a queue and the queued transactions can be in normal state or in critical state. The transactions in a queue moves from normal state to critical state when that transaction is not serviced for a count of MAX\_STARVE\_X32 clocks. The MAX\_STARVE\_X32 values for each queue can be configured using the DDR controller performance registers (refer "Performance" section on page 29). The DDR controller completes the critical transactions with high priority.

### 3.5.4.3.3 Write Combine

The DDR controller combines multiple writes to the same address into a single write to DDR memory. When a new write collides with the queued write, the DDR controller overwrites the data for the queued write with that from the new write and only performs one write transaction. The write combine functionality can be disabled by setting the register bit REG\_DDRC\_DIS\_WC to 1.

### 3.5.4.3.4 SECDED

The DDR controller supports built-in SECDED capability for correcting single-bit errors and detecting two-bit errors. The SECDED feature can be enabled in the **System Builder - memory controller configuration** window. When SECDED is enabled, the DDR controller adds 8 bits of SECDED data to every 64 bits of data.

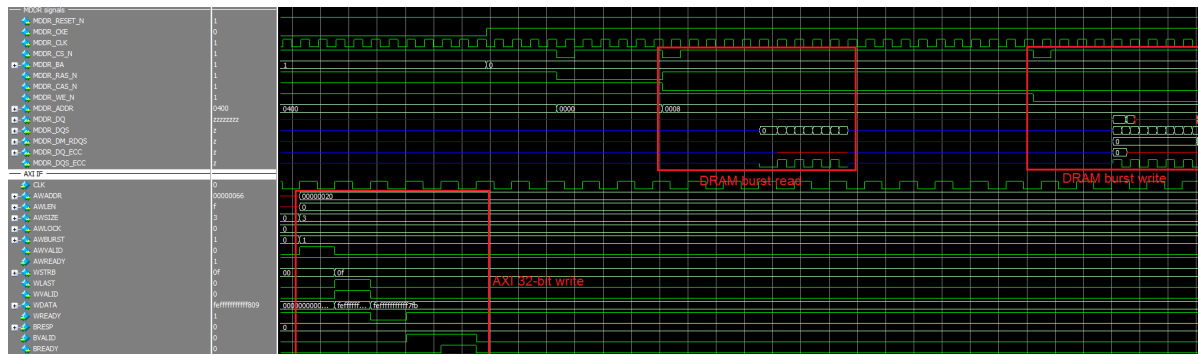
The DDR controller computes ECC for every 64-bit data. When SECDED is enabled, a write operation computes and stores a SECDED code along with the data, and a read operation reads and checks the data against the stored SECDED code. It is therefore, possible to receive single/dual bit errors when reading uninitialized memory locations. To avoid this, all the memory locations must be written before being read.

For a non 64-bit write operation, the DDR controller performs a 256-bit read modify write (RMW) operation. This read modify write operation is always performed on 256-bit aligned addresses.

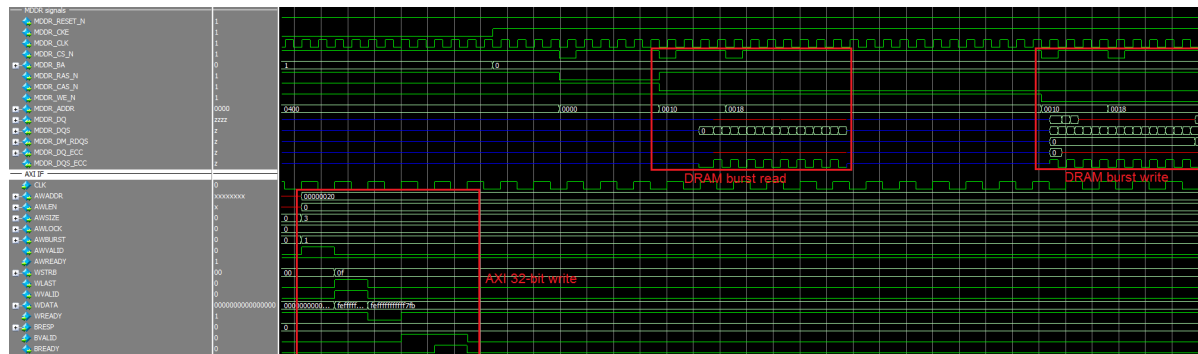
For example, if the DDR controller receives a 32-bit write operation to address 0x4, then the DDR controller performs the following operations:

1. Reads the 256-bit data from 0x0(256-bit aligned address for 0x4).
2. Modifies 32-bits (bit33 to bit64) of that 256-bit data with the user 32-bit data.
3. Computes the ECC and writes 288-bits (256-bit data + 32-bit ECC) to address 0x0.

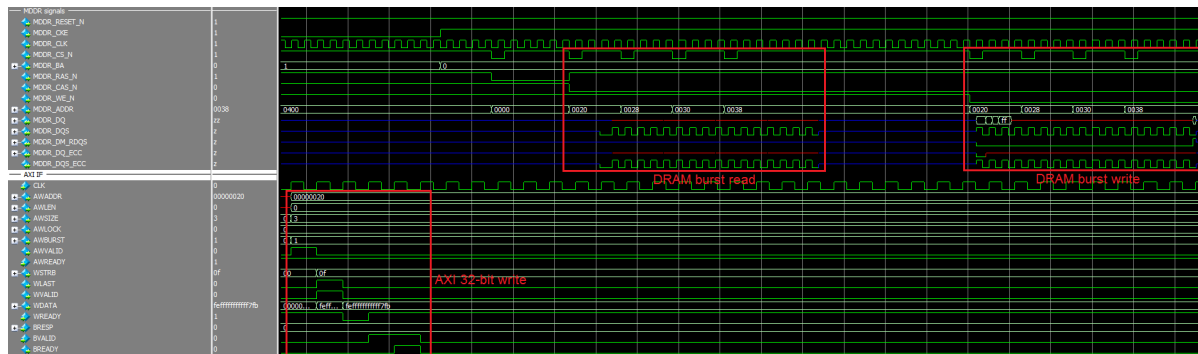
The following illustration shows the DDR controller burst transactions to DRAM for unaligned 64-bit AXI write transaction. The DDR controller is configured for DDR3 memory, 32-bit burst width, and burst length 8.

**Figure 7 • DDR RMW Operation (32-Bit DDR Bus Width and Burst Length 8)**

The following illustration shows the DDR controller burst transactions to DRAM for unaligned 64-bit AXI write transaction. The DDR controller is configured for DDR3 memory, 16-bit bust width, and burst length 8.

**Figure 8 • DDR RMW Operation (16-Bit DDR Bus Width and Burst Length 8)**

The following illustration shows the DDR controller burst transactions to DRAM for unaligned 64-bit AXI write transaction. The DDR controller is configured for DDR3 memory, 8-bit bust width, and burst length 8.

**Figure 9 • DDR RMW Operation (8-Bit DDR Bus Width and Burst Length 8)**

For more information on the SECCDED feature of SmartFusion2 MDDR, refer to the [DG0618: Error Detection and Correction on SmartFusion2 Devices using DDR Memory](#).

The SECEDED bits are interlaced with the data bits, as listed in the following table.

**Table 11 • SECEDED DQ Lines at DDR**

Mode	SECEDED Data Pins			
	M2S/M2GL005/010/025/060/090 M2S/M2GL150-FCV484	M2S/M2GL 050 (FCS325, VF400, FG484)	M2S/M2GL 050 (FG896)	M2S/M2GL 150 (FC1152)
Full bus width	—	—	MDDR_DQ_ECC[3:0]	MDDR_DQ_ECC[3:0]
Half bus width	MDDR_DQ_ECC[1:0]	MDDR_DQ_ECC[1:0]	MDDR_DQ_ECC[1:0]	MDDR_DQ_ECC[1:0]
Quarter bus width	MDDR_DQ_ECC[0]	—	—	MDDR_DQ_ECC[0]

When the controller detects a correctable SECEDED error, it does the following:

1. Generates an interrupt signal which can be monitored by reading the interrupt status register, [DDRC\\_ECC\\_INT\\_SR](#). The ECCINT interrupt is mapped to the group0 interrupt signal [MSS\\_INT\\_M2F\[12\]](#) in SmartFusion2 or [HPMS\\_INT\\_M2F\[12\]](#) in IGLOO2 of the fabric interface interrupt controller (FIIC).
2. Sends the corrected data to the read requested MSS/HPMS FPGA fabric master as part of the read data.
3. Sends the SECEDED error information to the [DDRC\\_LCE\\_SYNDROME\\_1\\_SR](#) register.
4. Performs a read-modify-write operation to correct the data present in the DRAM.

When the controller detects an uncorrectable error, it does the following:

1. Generates an interrupt signal which can be monitored by reading the interrupt status register, [DDRC\\_ECC\\_INT\\_SR](#). The ECCINT interrupt is mapped to the group0 interrupt signal [MSS\\_INT\\_M2F\[12\]](#) in SmartFusion2 or [HPMS\\_INT\\_M2F\[12\]](#) in IGLOO2 of the FIIC.
2. Sends the data with error to the read requested MSS/HPMS FPGA fabric master as part of the read data.
3. Sends the SECEDED error information to the [DDRC\\_LUE\\_SYNDROME\\_1\\_SR](#) register.

The following [SECEDED Registers](#) can be monitored for identifying the exact location of an error in the DDR SDRAM.

1. [DDRC\\_LUE\\_ADDRESS\\_1\\_SR](#) and [DDRC\\_LUE\\_ADDRESS\\_2\\_SR](#) give the row/bank/column information of the SECEDED unrecoverable error.
2. [DDRC\\_LCE\\_ADDRESS\\_1\\_SR](#) and [DDRC\\_LCE\\_ADDRESS\\_2\\_SR](#) give the row/bank/column information of the SECEDED error correction.
3. [DDRC\\_LCB\\_NUMBER\\_SR](#) indicates the location of the bit that caused the single-bit error in the SECEDED case (encoded value).
4. [DDRC\\_ECC\\_INT\\_SR](#) indicates whether the SECEDED interrupt is because of a single-bit error or double-bit error. The interrupt can be cleared by writing zeros to [DDRC\\_ECC\\_INT\\_CLR\\_REG](#).

### 3.5.4.3.5 Power Saving Modes

The DDR controller can operate DDR memories in three power saving modes:

Precharge Power-Down (DDR2, DDR3, LPDDR1)

- If power-down is enabled in the **System Builder MDDR configuration** or [REG\\_DDRC\\_POWERDOWN\\_EN](#) = 1, the DDR controller automatically keeps DDR memory in precharge power-down mode when the period specified by the power down entry time or [REG\\_DDRC\\_POWERDOWN\\_TO\\_X32](#) register has passed, while the controller is idle (except for issuing refreshes).
- The controller automatically performs the precharge power-down exit on any of the following conditions:
  - A refresh cycle is required to any rank in the system.
  - The controller receives a new request from the core logic.
  - [REG\\_DDRC\\_POWERDOWN\\_EN](#) is set to 0.

Self Refresh (DDR2, DDR3, LPDDR1)

- The DDR controller keeps the DDR memory devices in Self-refresh mode whenever the self refresh is enabled and the `REG_DDRC_SELFREF_EN` register bit is set and no reads or writes are pending in the controller.
- The controller takes the DDR memory out of Self-refresh mode whenever the `REG_DDRC_SELFREF_EN` input is deasserted or new commands are received by the controller.
- When the DDR self refresh is enabled, the DDR I/O bank may go into recalibration and a glitch may occur in the MDDR bank I/Os, which are being used for general purpose rather than for the DDR memory. The DDR I/Os ODT is periodically calibrated for PVT changes and will be effected only when the I/Os are in tri state (DDR I/Os are tri stated only in self-refresh mode).

#### Deep Power-Down (LPDDR1)

- This is supported only for LPDDR1. The DDR controller puts the DDR SDRAM devices in deep power-down mode whenever the `REG_DDRC_DEEPPowerDOWN_EN` bit is set and no reads or writes are pending in the DDR controller.
- The DDR controller automatically exits deep power-down mode and reruns the initialization sequence when the `REG_DDRC_DEEPPowerDOWN_EN` bit is reset to 0. The contents of DDR memory may be lost upon entry into deep power-down mode.

### 3.5.4.3.6 DRAM Initialization

After Reset, the DDR controller initializes DDR memories through an initialization sequence, depending on the type of DDR memory used. For more information on the initialization process, refer to the JEDEC specification.

## 3.5.5 MDDR Subsystem Features Configuration

The MDDR subsystem registers must be initialized before accessing DDR memory through the MDDR subsystem. When using the **System Builder** flow through Libero SoC, all of the necessary registers are initialized automatically by the resulting module.

This section provides the registers features of the MDDR. All registers are listed with their bit definitions in the "MDDR Configuration Registers" section on page 60 section.

### 3.5.5.1 Memory Type

`DDRC_MODE_CR` must be configured to select the memory type (DDR2, DDR3, or LPDDR1) to access from MDDR subsystem.

### 3.5.5.2 Bus Width Configurations

The MDDR supports various bus widths listed in the following table. The MDDR can be programmed to work in full, half, or quarter bus width mode by configuring the `DDRC_MODE_CR` and `PHY_DATA_SLICE_IN_USE_CR` registers when the controller is in soft reset.

**Table 12 • Supported Bus Widths**

Bus Width	M2GL005/M2GL010/M2GL025/ M2GL090	M2GL050 (FCS325, VF400, FG484)	M2GL050 (FG896)	M2GL150 (FC1152)
Full bus width	—		✓	✓
Half bus width	✓	✓	✓	✓
Quarter bus width	✓			✓

### 3.5.5.3 Burst Mode

The DDR controller performs the burst write operations to DDR memory, depending on the burst mode selection. Burst mode is selected as sequential or interleaving by configuring `REG_DDRC_BURST_MODE` to 1 or 0. Burst length can be selected as 4, 8, or 16 by configuring `REG_DDRC_BURST_RDWR`.



Supported burst modes for DDR SDRAM types and PHY widths are listed in the following table. For M2GL050 devices, only sequential burst mode and a burst length of 8 are supported.

**Table 13 • Supported Burst Modes**

Bus Width	Memory Type	Sequential/Interleaving	
		4	8
32	LPDDR1	✓	✓
	DDR2	✓	✓
	DDR3	–	✓
16	LPDDR1	–	✓
	DDR2	–	✓
	DDR3	–	✓
8	LPDDR1	–	✓
	DDR3	–	✓
	DDR2	–	

**Note:** The burst length 16 is supported for LPDDR1 if bus width is 16 except M2GL050.

### 3.5.5.4 Configuring Dynamic DRAM Constraints

Timing parameters for DDR memories must be configured according to the DDR memory specification. Dynamic DRAM constraints are subdivided into three basic categories:

- Bank constraints affect the transactions that are scheduled to a given bank.
- Rank constraints affect the transactions that are scheduled to a given rank.
- Global constraints affect all transactions.

### 3.5.5.5 Dynamic DRAM Bank Constraints

The timing constraints which affect the transactions to a bank are listed in the following table. The control bit field must be configured as per the DDR memory vendor specification.

**Table 14 • Dynamically Enforced Bank Constraints**

Timing Constraint of DDR Memory	Control Bit	Description
Row cycle time ( $t_{RC}$ )	REG_DDRC_T_RC	Minimum time between two successive activates to a given bank.
Row precharge command period ( $t_{RP}$ )	REG_DDRC_T_RP	Minimum time from a precharge command to the next command affecting that bank.
Minimum bank active time ( $t_{RAS(min)}$ )	REG_DDRC_T_RAS_MIN	Minimum time from an activate command to a precharge command to the same bank.
Maximum bank active time ( $t_{RAS(max)}$ )	REG_DDRC_T_RAS_MAX	Maximum time from an activate command to a precharge command to the same bank.
RAS-to-CAS delay ( $t_{RCD}$ )	REG_DDRC_T_RCD	Minimum time from an activate command to a Read or Write command to the same bank.
Write command period ( $t_{WR}$ )	REG_DDRC_WR2PRE	Minimum time from a Write command to a precharge command to the same bank.
Read-to-precharge delay ( $t_{RTP}$ )	REG_DDRC_RD2PRE	Minimum time from a Read command to a precharge command to the same bank. Set this to the current value of additive latency plus half of the burst length.

### 3.5.5.5.1 Dynamic DRAM Rank Constraints

The timing constraints that affect the transactions to a rank are listed in the following table. The control bit field must be configured as per the DDR memory vendor specification.

**Table 15 • Dynamically-Enforced Bank Constraints**

Timing Constraints of DDR Memory	Control Bit	Description
Nominal refresh cycle time ( $t_{RFC(nom)}$ or $t_{REFI}$ )	REG_DDRC_T_RFC_NOM_X32	Average time between refreshes for a given rank. The actual time between any two refresh commands may be larger or smaller than this; this represents the maximum time allowed between refresh commands to a given rank when averaged over a large period of time.
Minimum refresh cycle time $t_{RFC(min)}$	REG_DDRC_T_RFC_MIN	Minimum time from refresh to refresh or activate.
RAS-to-RAS delay ( $t_{RRD}$ )	REG_DDRC_T_RRD	Minimum time between activates from bank A to bank B.
RAS-to-CAS delay ( $t_{CCD}$ )	REG_DDRC_T_CCD	Minimum time between two reads or two writes (from bank A to bank B).
Four active window ( $t_{FAW}$ )	REG_DDRC_T_FAW	Sliding time window in which a maximum of: 4 bank activates are allowed in an 8-bank design. In a 4-bank design, set this register to 0x1.

### 3.5.5.5.2 Dynamic DRAM Global Constraints

The timing constraints that affect global transactions are listed in the following table. The control bit field must be configured as per the DDR memory vendor specification.

**Table 16 • Dynamic DRAM Global Constraints**

Timing Constraint	Control Bit	Description
Read-to-write turnaround time ( $t_{RTW}$ )	REG_DDRC_RD2WR	Minimum time to allow between issuing any Read command and issuing any WRITE command
Write-to-read turnaround time ( $t_{RTR}$ )	REG_DDRC_WR2RD	Minimum time to allow between issuing any Write command and issuing any Read command
Write latency	REG_DDRC_WRITE_LATENCY	Time after a Write command that write data should be driven to DRAM.

The DDR memories require delays after initializing the mode registers. The following registers must be configured for the delay requirements for the DDR memories. The DDR controller uses these delay values while initializing the DDR memories.

- DDRC\_CKE\_RSTN\_CYCLES\_1\_CR (recommended value is 0x4242)
- DDRC\_CKE\_RSTN\_CYCLES\_2\_CR (recommended value is 0x8)

### 3.5.5.6 Address Mapping

The DDR controller maps linear request addresses to DDR memory addresses by selecting the source bit that maps to each and every applicable DDR memory address bit.

Each DDR memory address bit has an associated register vector to determine its source. The source address bit number is determined by adding the internal base of a given register to the programmed value for that register, as described in [EQ 1](#).

[Internal base] + [register value] = [source address bit number]

EQ 1

For example, reading the description for REG\_DDRC\_ADDRMAP\_COLB3, the internal base is 3; so when the full data bus is in use, the column bit 4 is determined by 3+ [register value].

If this register is programmed to 2, then the source address bit is: 3+2 = 5.

The DDR configurator assigns values to the address mapping registers depending on the selected number of columns, rows and banks. The following illustration provides the default mapping of the memory row, bank, and column address to the user interface address domain.

**Figure 10 • Address Mapping**

Full bus width mode																																
AXI/AHB Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row mapping (DDR2/DDR3)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
Row mapping (LPDDR)		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0															
Bank mapping(DDR2/DDR3)																		2	1	0												
Bank mapping(LPDDR)																			1	0												
column mapping																					9	8	7	6	5	4	3	2	1	0		
Half bus width mode																																
AXI/AHB Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row mapping(DDR2/DDR3)		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0															
Row mapping (LPDDR)			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
Bank mapping(DDR2/DDR3)																			2	1	0											
Bank mapping(LPDDR)																				1	0											
column mapping																						9	8	7	6	5	4	3	2	1	0	
Quarter bus width mode																																
AXI/AHB Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row mapping(DDR2/DDR3)				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
Row mapping (LPDDR)					15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
Bank mapping(DDR2/DDR3)																				2	1	0										
Bank mapping(LPDDR)																					1	0										
column mapping																							9	8	7	6	5	4	3	2	1	0

The following are the address mapping registers:

- [DDRC\\_ADDR\\_MAP\\_BANK\\_CR](#)
- [DDRC\\_ADDR\\_MAP\\_COL\\_1\\_CR](#)
- [DDRC\\_ADDR\\_MAP\\_COL\\_2\\_CR](#)
- [DDRC\\_ADDR\\_MAP\\_COL\\_3\\_CR](#)
- [DDRC\\_ADDR\\_MAP\\_ROW\\_1\\_CR](#)
- [DDRC\\_ADDR\\_MAP\\_ROW\\_2\\_CR](#)

While configuring the registers, ensure that two DDR memory address bits are not determined by the same source address bit.

**Note:** Some registers map multiple source address bits (REG\_DDRC\_ADDRMAP\_ROW\_B0\_11).

To arrive at the right address for the DDR controller, the system address or AXI address bits [4:0] are mapped by the MDDR.

- In full bus width mode, the system address bits [4:0] are used to map the lower column address bits (C0, C1, C2).
- In half bus width mode, the system address bits [4:0] are used to map the lower column address bits (C0, C1, C2, C3).
- In quarter bus width mode, the system address bits [4:0] are used to map the lower column address bits (C0, C1, C2, C3, C4).

The MDDR configurator uses (Row, Bank, and Column) address mapping as shown in the following example.

### 3.5.5.6.1 Example

In this example, the Address map registers are configured to access a 512 MB DDR3 SDRAM memory (MT41J512M8RA) from the MDDR subsystem as shown in "Example 2: Connecting 32-Bit DDR3 to MDDR\_PADs with SECEDED" section on page 58. The 512M x 8-bit DDR3 memory module has 3 bank address lines, 16 rows, and 10 columns.

- The column address bits 3 to 9 are mapped for system address bit[5] to system address bit[11]. To map the column 3-bit (C3) to address [5], the field is configured to 3, as the base value is 2. Similarly, the other column address bits are configured:
  - DDRC\_ADDR\_MAP\_COL\_1\_CR = 0x3333
  - DDRC\_ADDR\_MAP\_COL\_2\_CR = 0x3FFF
  - DDRC\_ADDR\_MAP\_COL\_3\_CR = 0x3300
- The bank address bits 0 to 2 are mapped for system address bit[12] to system address bit[14]. To map the bank bit0 to address [12], the field is configured to A, as the base value is 2. Similarly, the other bank address bits are configured:
  - DDRC\_ADDR\_MAP\_BANK\_CR = 0xAAA
- The row address bits 0 to 15 are mapped for system address bit[15] to system address bit[27]. To map the bank bit0 to address [15], the field is configured to 9, as the base value is 6. Similarly, the other bank address bits are configured:
  - DDRC\_ADDR\_MAP\_ROW\_1\_CR = 0x9999
  - DDRC\_ADDR\_MAP\_ROW\_2\_CR = 0x9FF

**Note:** The MDDR can access the 2 GB address space (0x00000000 - 0x7FFFFFFF). But in this example, 512 MB (0x00000000 - 0x1FFFFFFF) DDR3 SDRAM is connected to the 16 address lines of MDDR. The memory visible in the other memory space is mirrored of this 512 MB memory.

### 3.5.5.7 DDR Mode Registers

After reset, the DDR controller initializes the mode registers of DDR memory with the values in the following registers. The mode registers must be configured according to the specification of the external DDR memory when the controller is in soft reset.

- DDRC\_INIT\_MR\_CR
- DDRC\_INIT\_EMR\_CR
- DDRC\_INIT\_EMR2\_CR
- DDRC\_INIT\_EMR3\_CR

The T\_MOD and T\_MRD bits in DDRC\_DRAM\_MR\_TIMING\_PARAM\_CR must be configured to the required delay values. T\_MOD and T\_MRD are delays between loading the mode registers.

### 3.5.5.8 SECEDED

To enable SECEDED mode, set the REG\_DDRC\_MODE bits to 101 in DDRC\_MODE\_CR. The PHY\_DATA\_SLICE\_IN\_USE\_CR register must be configured to enable data slice 4 of the PHY.

The register value REG\_DDRC\_LPR\_NUM\_ENTRIES in the performance register, DDRC\_PERF\_PARAM\_1\_CR, must be increased by 1 to the value used in Normal mode (without SECEDED).

**Note:** MDDR has 36 DQ lines. These data lines are split into the following data slices:

- Data slice0 represents first 8 DQ lines (DQ0 to DQ7)
- Data slice1 represents next 8 DQ lines (DQ8 to DQ15)
- Data slice2 represents next 8 DQ lines (DQ16 to DQ23)
- Data slice3 represents next 8 DQ lines (DQ24 to DQ31)
- Data slice4 represents the remaining 4 DQ lines (DQ32 to DQ35)

### 3.5.5.9 Read Write Latencies

The read and write latencies between DDR controller and DDR PHY can be configured. Configure the DDRC\_DRAM\_RD\_WR\_LATENCY\_CR register for adding latencies for read and writes.

### 3.5.5.10 Performance

The DDR controller has several performance registers which can be used to increase the speed of the read and write transactions to DDR memory.

The DDR controller has a transaction store, shared for low and high priority transactions. The [DDRC\\_PERF\\_PARAM\\_1\\_CR](#) register can be configured for allocating the transaction store between the low and high priority transactions. For example, if the [REG\\_DDRC\\_LPR\\_NUM\\_ENTRIES](#) field is configured to 0, the controller allocates more time to high priority transactions. The ratio for LPR: HPR is 1:7 (as the transaction store depth is 8).

The [DDRC\\_HPR\\_QUEUE\\_PARAM\\_1\\_CR](#), [DDRC\\_LPR\\_QUEUE\\_PARAM\\_1\\_CR](#), and [DDRC\\_WR\\_QUEUE\\_PARAM\\_CR](#) registers can be configured for the minimum clock values for treating the transactions in the HPR, LPR, and WR queue as critical and non-critical.

To force all incoming transactions to low priority, configure the [DDRC\\_PERF\\_PARAM\\_2\\_CR](#) register. By default it is configured to force all the incoming transactions to low priority.

### 3.5.5.11 Refresh Controls

The DDR controller automatically issues refresh commands to DDR memory for every tRFC (min). The DDR controller can be programmed to issue single refreshes at a time ([REG\\_DDRC\\_REFRESH\\_BURST](#) = 0) to minimize the worst-case impact of a forced refresh cycle. It can be programmed to burst the maximum number of refreshes allowed for DDR ([REFRESH\\_BURST](#) = 7, for performing 8 refreshes at a time) to minimize the bandwidth lost when refreshing the pages.

### 3.5.5.12 1T or 2T Timing

The DRAM can be used in 1T or 2T Timing mode by configuring the [DDRC\\_PERF\\_PARAM\\_3\\_CR](#) register. The address bus can be clocked using 1T or 2T clocking. With 1T, the DDR controller can issue a new command on every clock cycle. In 2T timing, the DDR controller holds the address and command bus valid for two clock cycles. This reduces the efficiency of the bus to one command per two clocks, but it doubles the amount of setup and hold time. The data bus remains the same for all of the variations in the address bus and the default configuration is 1T timing mode.

### 3.5.5.13 ODT Controls

The ODT for a specific rank of memory can be enabled or disabled by configuring the [DDRC\\_ODT\\_PARAM\\_1\\_CR](#) and [DDRC\\_ODT\\_PARAM\\_2\\_CR](#) registers. These must be configured before taking the controller out of soft reset. They are applied to every read or write issued by the controller.

### 3.5.5.14 Soft Resets

Set the [REG\\_DDRC\\_SOFT\\_RSTB](#) bit of [DDRC\\_DYN\\_SOFT\\_RESET\\_CR](#) to 0 to reset the DDR controller. To release the DDR controller from reset, set the [REG\\_DDRC\\_SOFT\\_RSTB](#) bit of [DDRC\\_DYN\\_SOFT\\_RESET\\_ALIAS\\_CR](#) to 1.

### 3.5.5.15 MDDR Memory Map

The address map to access the DDR memory from MSS/HPMS masters through MDDR is 0xA0000000-0xDFFFFFFF, which is 1 GB. But the MDDR can support up to 2 GB of memory, out of which only 1 GB of this memory is accessible at a time from the MSS/HPMS masters through the AHB bus matrix. [DDR\\_FIC](#) can access the entire 2 GB memory.

To enable MSS/HPMS masters to access 2 GB, the DDR address space (0x00000000-0x7FFFFFFF) is divided into 8 DDR regions, as shown in [Table 17 on page 31](#). Each region is 256 MB (4 regions together form 1 GB). The HPMS masters can access any of these four regions at a time, depending on the Address Space Mapping mode configured for that particular master using the [DDRB\\_CR](#) register in [SYSREG](#). For SmartFusion2, the [DDRB\\_CR](#) register has four 4-bit fields ([DDR\\_IDC\\_MAP](#), [DDR\\_SW\\_MAP](#), [DDR\\_HPD\\_MAP](#), and [DDR\\_DS\\_MAP](#)). For IGLOO2, the [DDRB\\_CR](#) register has two 4-bit fields ([DDR\\_SW\\_MAP](#), [DDR\\_HPD\\_MAP](#)) whose bits can be configured to select the DDR Address Space Mapping modes from 0 to 12.

The Address Space Mapping modes for a 2 GB memory are shown in [Table 18](#), page 31. For example, if the [DDR\\_SW\\_MAP](#) is configured as 0001, then the AHB bus matrix can access 0, 1, 2, and 3 regions of

DDR that is, the accessible DDR memory from AHB bus matrix is 0x00000000-0x4FFFFFFF which is 1 GB.

**Table 17 • DDR Memory Regions**

DDR Memory Region	DDR Memory Space
0	0x00000000-0x0FFFFFFF
1	0x10000000-0x1FFFFFFF
2	0x20000000-0x2FFFFFFF
3	0x30000000-0x3FFFFFFF
4	0x40000000-0x4FFFFFFF
5	0x50000000-0x5FFFFFFF
6	0x60000000-0x6FFFFFFF
7	0x70000000-0x7FFFFFFF

**Table 18 • Accessed DDR Memory Regions Based on Mode Settings for a 2 GB Memory**

Address Space Mapping Modes	DDR Memory Regions Visible at MSS/HPMS DDR Address Space for Different Modes			
	MSS/HPMS DDR Space 0 (0xA0000000-0xAFFFFFFF)	MSS/HPMS DDR Space 1 (0xB0000000-0xBFFFFFFF)	MSS/HPMS DDR Space 2 (0xC0000000-0xCFFFFFFF)	MSS/HPMS DDR Space 3 (0xD0000000-0xDFFFFFFF)
0000	Region 2	Region 3	Region 4	Region 5
0001	Region 0	Region 1	Region 2	Region 3
0010	Region 0	Region 1	Region 2	Region 3
0011	Region 4	Region 5	Region 6	Region 7
0110	Region 0	Region 1	Region 2	Region 3
0111	Region 0	Region 1	Region 4	Region 5
1000	Region 0	Region 1	Region 6	Region 7

If 1 GB of DDR memory is connected to MDDR, only 4 regions are available (0-4). The following table shows the DDR regions available for address mode settings.

**Table 19 • Accessed DDR Memory Regions Based on Mode Settings for a 1 GB Memory**

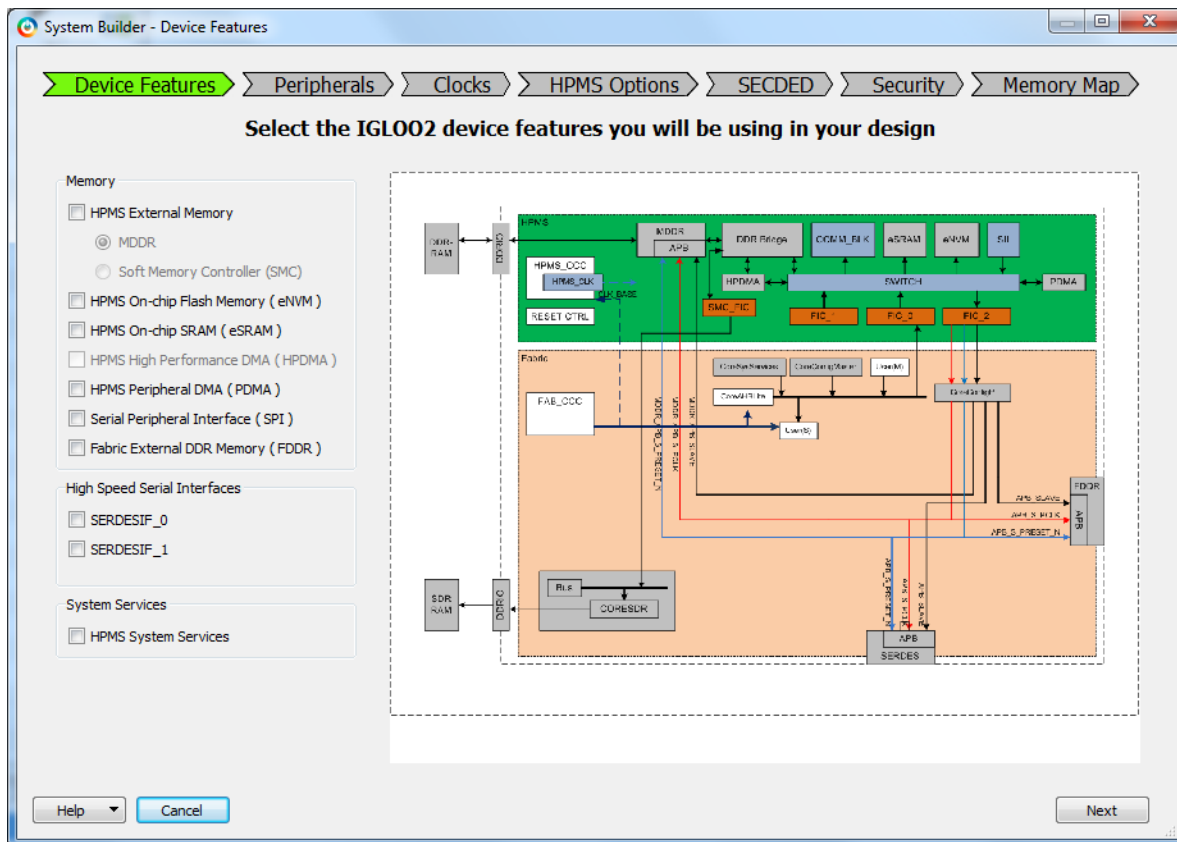
Address Space Mapping Modes	DDR Memory Regions Visible at HPMS DDR Address Space for Different Modes			
	MSS/HPMS DDR Space 0 (0xA0000000-0xAFFFFFFF)	MSS/HPMS DDR Space 1 (0xB0000000-0xBFFFFFFF)	MSS/HPMS DDR Space 2 (0xC0000000-0xCFFFFFFF)	MSS/HPMS DDR Space 3 (0xD0000000-0xDFFFFFFF)
0000	Region 2	Region 3	Region 0	Region 1
0001	Region 0	Region 1	Region 2	Region 3
0010	Region 0	Region 1	Region 2	Region 3

### 3.6 How to Use MDDR in IGL002 Device

This section describes how to use MDDR in the IGLOO2 devices. To configure the IGLOO2 device features and then build a complete system, use the System Builder graphical design wizard in the Libero Software.

The following image shows the initial System Builder window where you can select the features that you require. For details on how to launch the System Builder wizard and a detailed information on how to use it, refer the *IGLOO2 System Builder User Guide*. You can also use CoreABC based initialization as described in *Igloo2 Standalone Peripheral Initialization User Guide*.

**Figure 11 • System Builder—Device Features Window**



For more information about how to use MDDR in the SmartFusion2 devices, refer to ["Appendix A: How to Use the MDDR in SmartFusion2"](#) section on page 111.

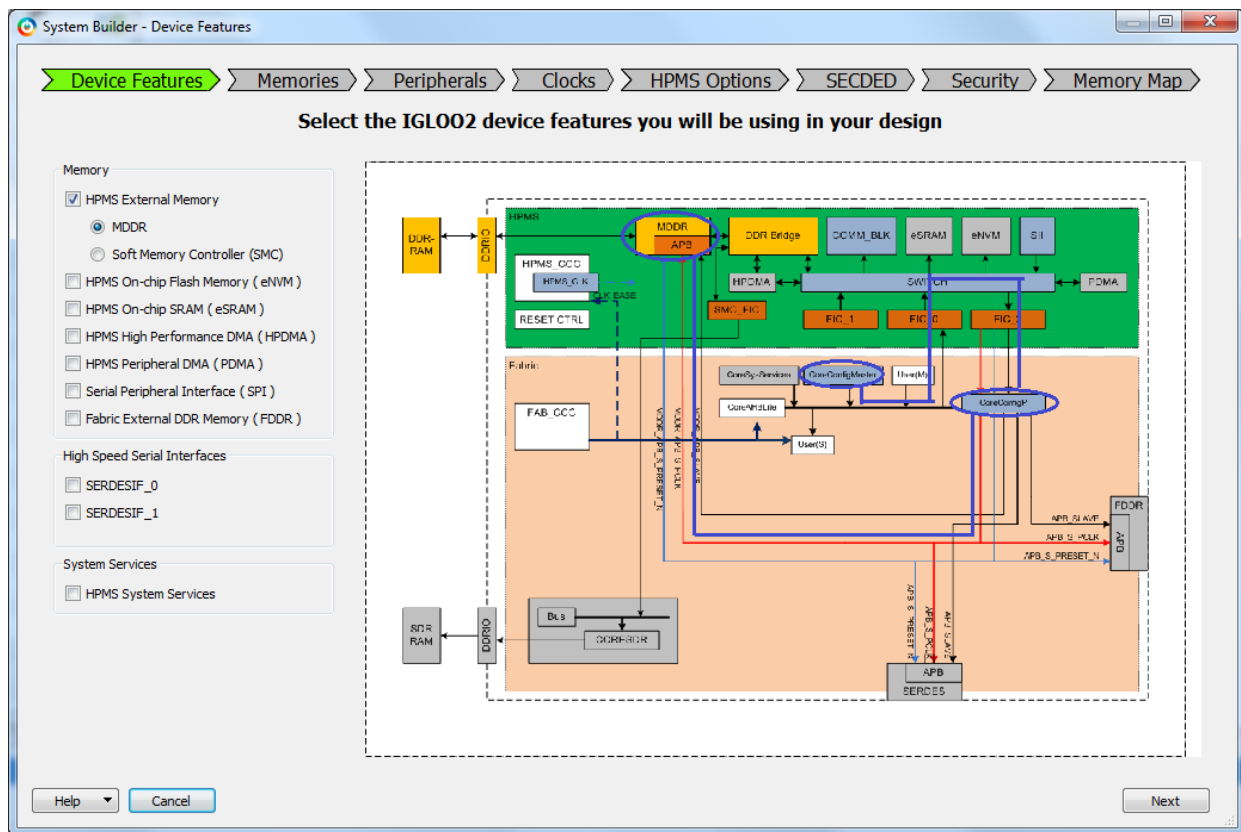
### 3.6.1 Configuring MDDR

The following steps configure the MDDR:



1. Check the HPMS External DDR Memory (MDDR) check box under the Device Features tab and leave the other check boxes unchecked. The following image shows the System Builder - Device Features tab.

**Figure 12 • MDR Initialization Path**



2. Selecting the **MDDR** under **HPMS External Memory** check box in the **System Builder** performs the following actions:
  - Instantiates the required IPs like CoreConfigMaster and CoreConfigP that initialize the MDDR Controller.
  - Establishes the initialization path:  
CoreConfigMaster → FIC\_0 → eNVM → FIC\_2 → CoreConfigP → APB bus of the MDDR subsystem
    - CoreConfigMaster (AHB Master) accesses the DDR configuration data stored in eNVM through FIC\_0.
    - The configuration data is sent to CoreConfigP through the FIC\_2 master port.
    - CoreConfigP sends the configuration data to APB bus of the MDDR subsystem.
3. Navigate to the **Memories** tab. Select the memory settings under the **General** tab depending on the application requirement, as shown in [Figure 13](#), page 35.
  - Memory type can be selected as DDR2, DDR3, or LPDDR.
  - Data width can be selected as 32-bit, 16-bit, or 8-bit. Refer to [Table 13 on page 26](#) for supported data widths for various IGLOO2 device packages.
  - SECCED (ECC) can be enabled or disabled.
  - Arbitration Scheme can be selected from Type-0 to Type-3. Refer to [Table 10 on page 20](#) for arbitration scheme details.
  - The highest priority ID of fabric master ranges from 0 to 15 if the selected arbitration scheme is other than Type-0.



- For address mapping, the register settings that perform mapping to system address bits for row, bank and column combinations are automatically computed by the configurator using the address mapping option. The following table lists the supported range for row, bank, and column.

**Table 20 • Supported Address Width Range for Row, Bank and Column Addressing in DDR/LPDDR**

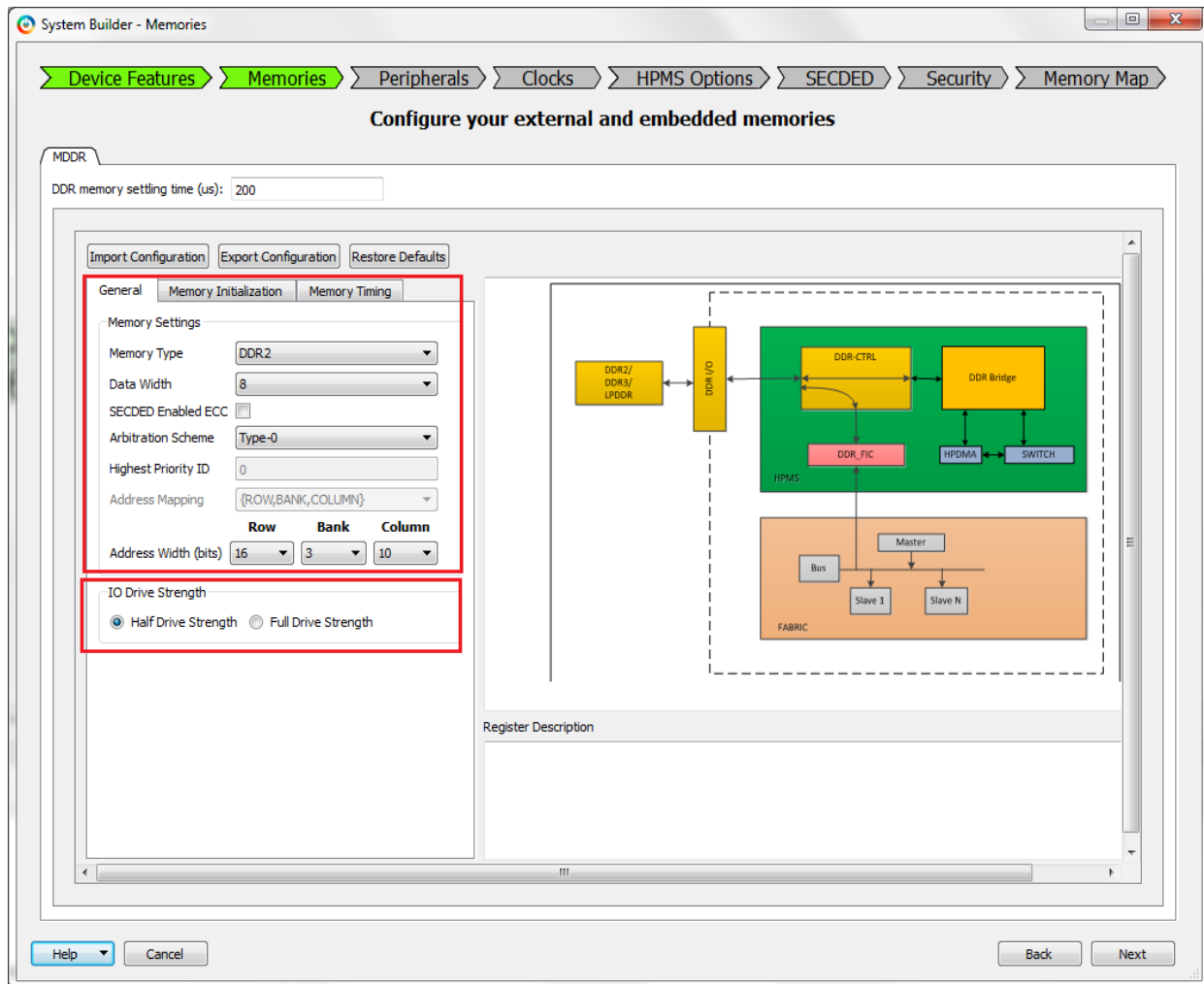
Width	DDR2	DDR3	LPDDR
Row Address	12–16	12–16	12–16
Bank Address	2–3	2–3	2–3
Column Address	9–12	9–12	9–12

For more information refer to the ["Address Mapping" section](#).

- Select the **I/O Drive Strength** as **Half Drive Strength** or **Full Drive Strength**, as shown in [Figure 13](#), page 35. The following table lists how the DDR I/O standard is configured based on this setting.

**Table 21 • DDR I/O Standard is Configured Based on I/O Drive Strength Setting**

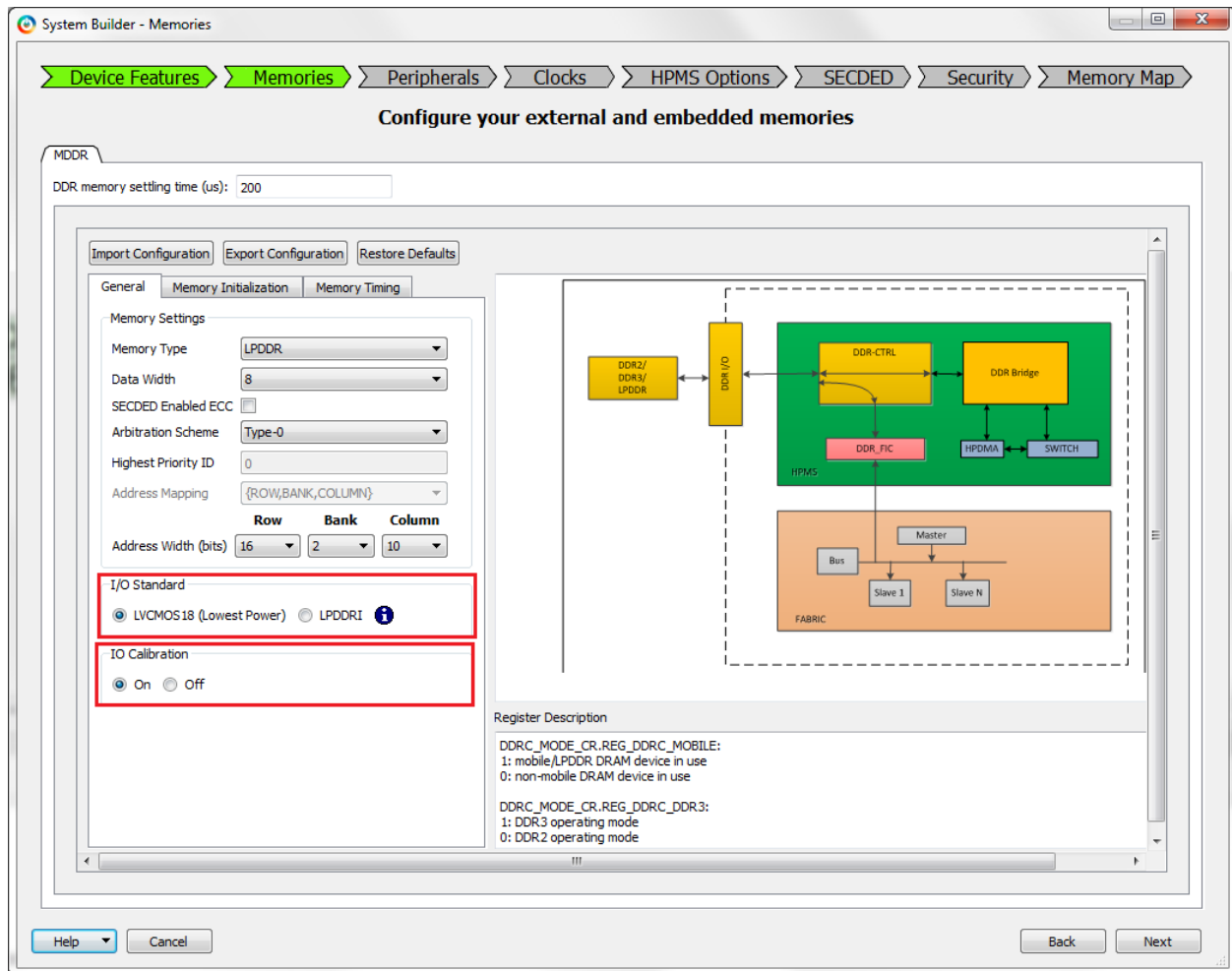
I/O Drive Strength	Memory Type	
	DDR2	DDR3
Half Drive Strength	SSTL18I	SSTL15I
Full Drive Strength	SSTL18II	SSTL15II

**Figure 13 • I/O Drive Strength Setting**

4. For only LPDDR memory, the **I/O standard** and **I/O calibration** settings are available as shown in the following illustration.
  - Select **I/O standard** as **LVC MOS18** or **LPDDR1**. For the Microsemi M2GL\_EVAL\_KIT board, select LPDDR1(SSTL18) because the board is designed to use the LPDDR1 I/O standard.

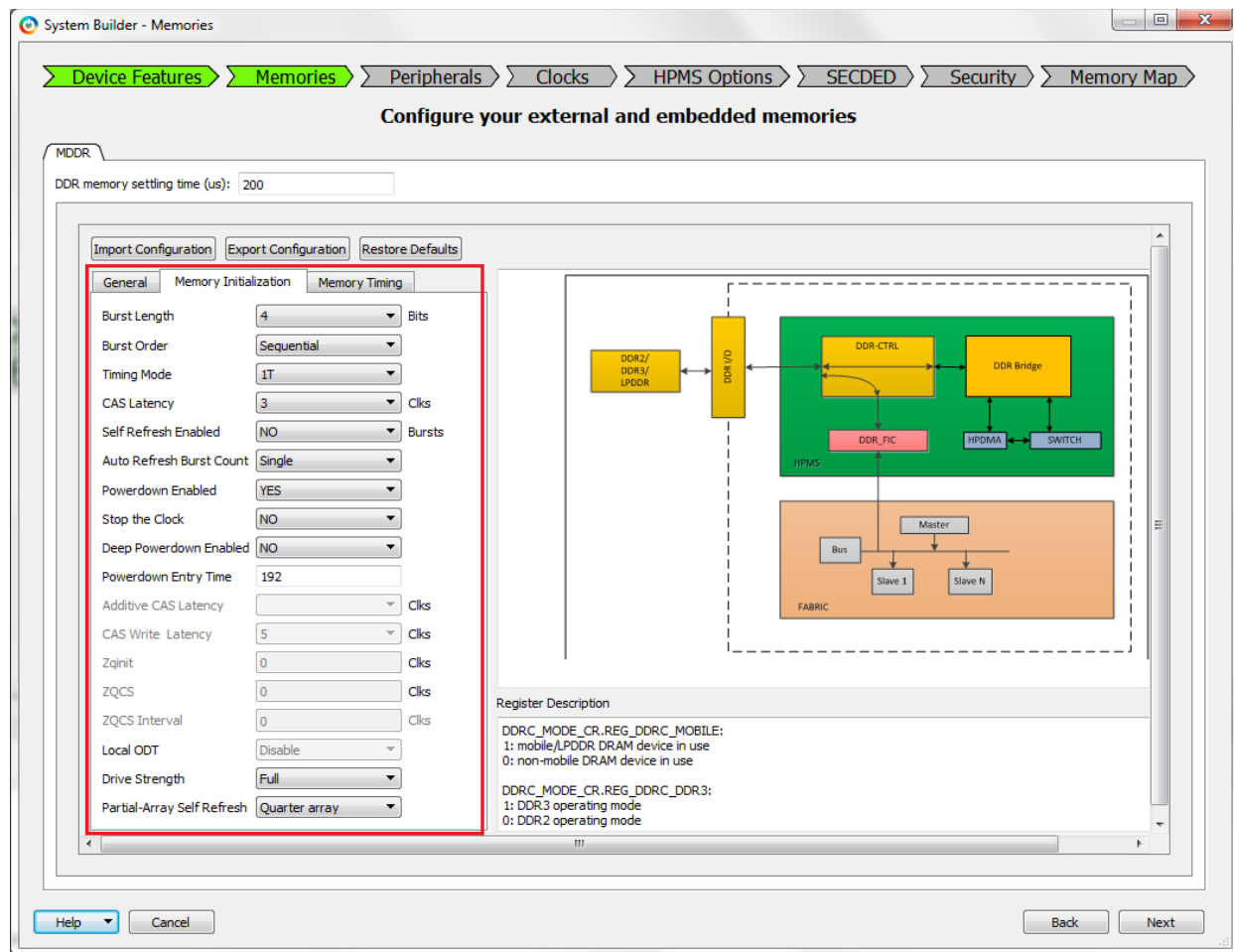
**Note:** If LVC MOS18 is selected, all I/Os are configured to LVC MOS1.8 except CLK/CLK\_N.CLK and CLK\_N, which are configured to the LPDDR1 standard because they are differential signals.

- Select **I/O calibration** as ON or OFF. If I/O calibration is selected as ON, then the IGLOO2 MDDR\_IMP\_CALIB pin must be pulled down with a resistor. For more information on resistor values, refer to the Impedance Calibration section in the DS0124: IGLOO2 Pin Descriptions Datasheet.

**Figure 14 • Selecting I/O Standard as LVCMOS18 or LPDDR1**


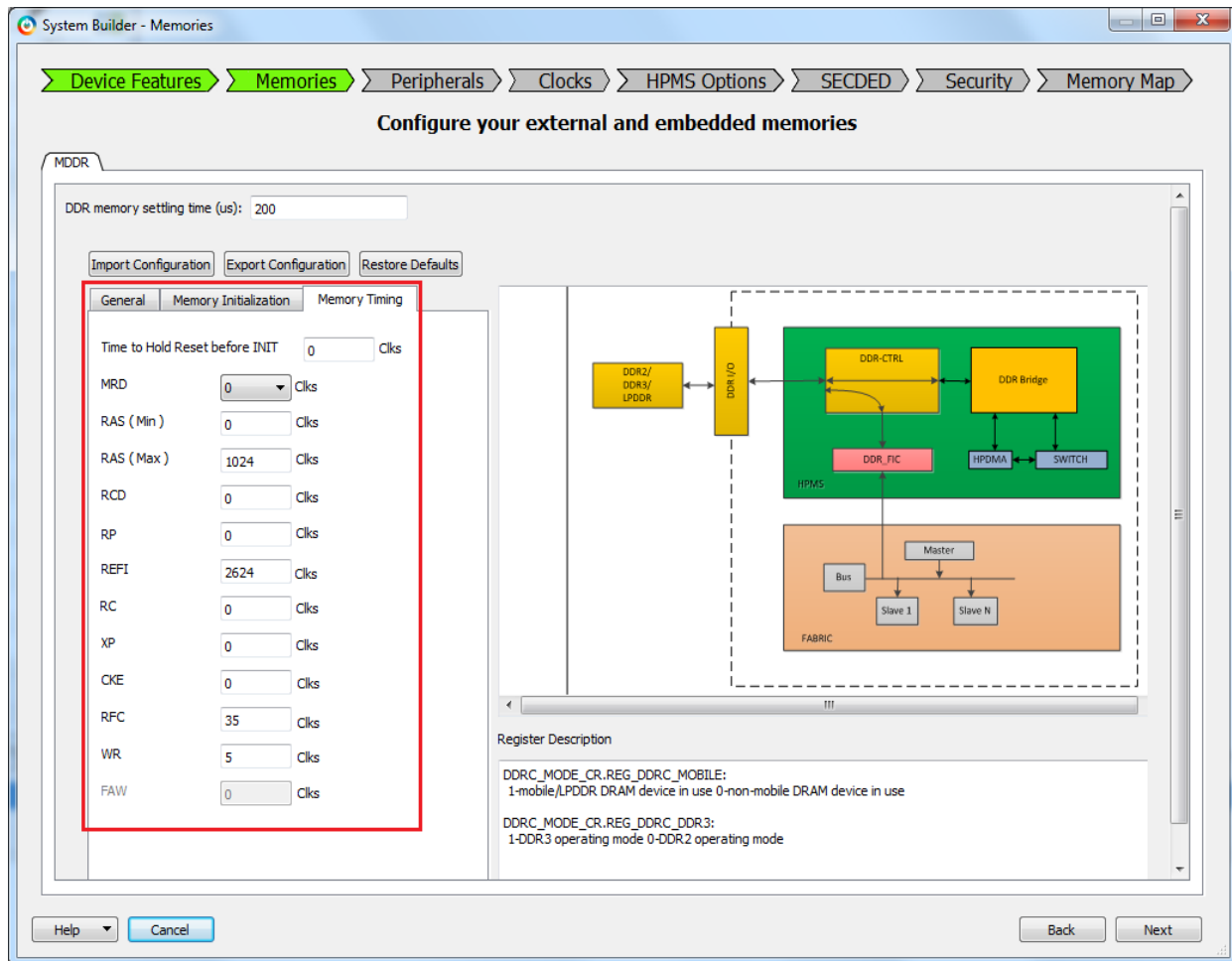
5. Depending on the application requirement, select the Memory Initialization settings under the Memory Initialization tab as shown in [Figure 15 on page 38](#).
- Select the following performance-related settings:
  - Burst length can be selected as 4, 8, or 16. Refer to [Table 13 on page 26](#) for supported burst lengths.
  - Burst order can be selected as sequential or interleaved. Refer to [Table 13 on page 26](#) for supported burst orders.
  - Timing mode can be selected as 1T or 2T. For more details, refer to "1T or 2T Timing" section on [page 30](#).
  - CAS latency is the delay in clock cycles between the internal READ command and the availability of the first bit of output data. Select the CAS latency according to the DDR memory (mode register) datasheet.
- Select the following power saving mode settings. Refer to "Power Saving Modes" section on [page 24](#) for more details.
  - Self-refresh enabled
  - Auto refresh burst count
  - Power down enabled
  - Stop the clock (supported only for LPDDR)
  - Deep power down enabled (supported only for LPDDR)
  - Power down entry time
- Select the additional performance settings for DDR3 memory.
  - Additive CAS Latency is defined by EMR[5:3] register of DDR2 memory and by MR1[4:3] register of DDR3 memory. It enables the DDR2 or DDR3 SDRAM to allow a READ or WRITE

- command from DDR Controller after the ACTIVATE command for the same bank prior to tRCD (MIN). This configuration is part of DDR2 Extended Mode register and DDR3 mode register1.
- CAS Write Latency (CWL) is defined by DDR3 MR2[5:3] and is the delay in clock cycles from the releasing of the internal write to the latching of the first data in. The overall WRITE latency (WL) is equal to CWL + AL (by default CWL is set to 5 clock cycles).
  - Select the following ZQ Calibration settings for DDR3 memory. For more details, refer to ["ZQ Calibration" section on page 17](#).
    - Zqinit
    - ZQCS
    - ZQCS Interval
  - Select the other following settings.
    - The local ODT setting is not supported for LPDDR memory. For the DDR2/DDR3 memory type, the user can choose any option for "Local ODT". User can enable or disable "LOCAL ODT" during read transaction.
    - Drive strength setting is defined by EMR[7:5] register bits of LPDDR memory with drop down options of 'Full', 'Half', 'Quarter', and 'One-eighth' drive strength; EMR[1] register bit of DDR2 memory with drop down options of 'Full' and 'Weak' drive strength; and MR1 register bits M5 and M1 of DDR3 memory with drop down options of 'RZQ/6' and 'RZQ/7'.
    - The partial array self-refresh coverage setting is defined by EMR[2:0] register bits of LPDDR memory with drop down options of 'Full', 'Quarter', 'One-eighth', and 'One-sixteenth'. This feature improves power savings by selecting the amount of memory to be refreshed during self-refresh.
    - R<sub>TT</sub> (Nominal) setting is defined by EMR[6] and EMR[2] register bits of DDR2 memory, which determines what ODT resistance is enabled with drop down options of 'RTT disabled', '50 ohms', '75 Ω', and '150 Ω', and it is defined by MR1[9], MR1[6] and MR1[2] register bits of DDR3 memory. In DDR3 memory, RTT nominal termination is allowed during standby conditions and WRITE operations, not during READ operations with drop down options of 'RZQ/2', 'RZQ/4' and 'RZQ/6'.
    - R<sub>TT\_WR</sub> (Dynamic ODT) setting is defined by MR2[10:9] register bits of DDR3 memory. This is applicable only during WRITE operations. If dynamic ODT (Rtt\_WR) is enabled, DRAM switches from normal ODT (R<sub>TT\_nom</sub>) to dynamic ODT (Rtt\_WR) when beginning WRITE burst and subsequently switches back to normal ODT at the end of WRITE burst. The drop down options provided to the user are 'off', 'RZQ/4', and 'RZQ/2'.
    - The auto self-refresh setting is defined by MR2[6] register bit of DDR3 memory with drop down options 'Manual' and 'Auto'. The self-refresh temperature setting is defined by MR2[7] register bit of DDR2 memory with drop down options of 'Normal' and 'Extended'.

**Figure 15 • Memory Initialization Configuration**

6. Select the memory timing settings under the **Memory Timing** tab according to the DDR memory vendor datasheet, as shown in the following image. For more details, refer to "Configuring Dynamic DRAM Constraints" section on page 26.

**Figure 16 • Memory Timing Configuration**



The configurator also provides the option to import and export the register configurations. The configuration settings are stored in eNVM. Configuration files for accessing LPDDR memory on IGLOO2 evaluation kit can be downloaded from:

[www.microsemi.com/soc/documents/LPDDR\\_Emcrafft\\_Config.zip](http://www.microsemi.com/soc/documents/LPDDR_Emcrafft_Config.zip).

The following is an example of MDDR register configurations for operating the LPDDR memory (MT46H64M16LF) with clock 166 MHz.

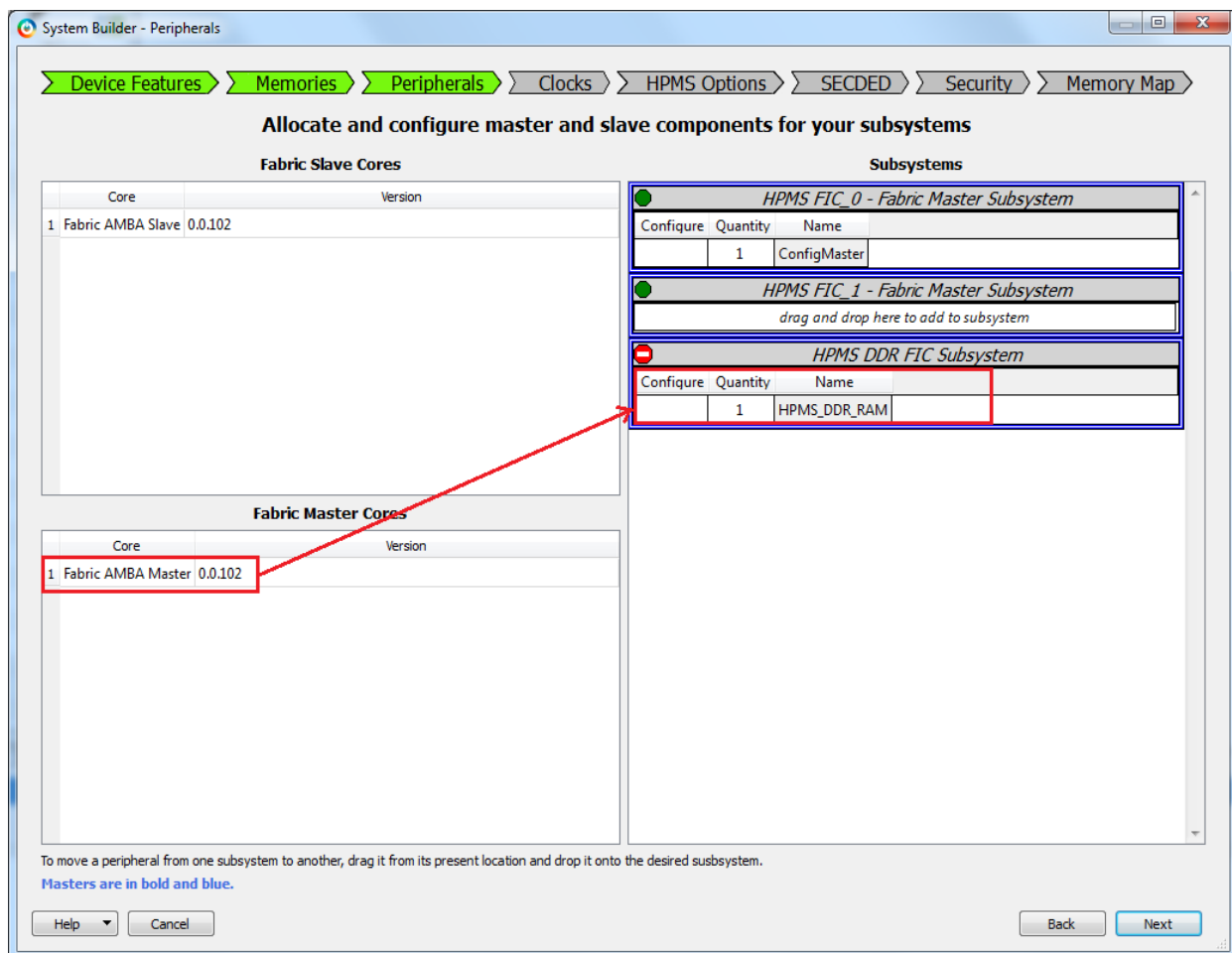
- Device Memory Settling Time ( $\mu$ s): 200

The DDR memories require settling time for the memory to initialize before accessing it. The LPDDR memory model MT46H64M16LF needs 200  $\mu$ s settling time.

- General
  - Memory Type: LPDDR
  - Data Width: 16
- Memory Initialization
  - Burst length: 8
  - Burst Order: Interleaved
  - Timing Mode: 1T
  - CAS Latency: 3
  - Self Refresh Enabled: No
  - Auto Refresh Burst Count: 8
  - PowerDown Enabled: Yes
  - Stop the clock: No

- Deep PowerDown enabled: No
  - No Activity clocks for Entry: 320
  - Memory Timing
    - Time To Hold Reset Before INIT: 67584 clks
    - MRD: 4 clks
    - RAS (Min): 8 clks
    - RAS (Max): 8192 clks
    - RCD: 6 clks
    - RP: 7 clks
    - REFI: 3104 clks
    - RC: 3 clks
    - XP: 3 clks
    - CKE: 3 clks
    - RFC: 79 clks
    - FAW: 0 clks
7. Navigate to the **Peripherals** tab. The **Peripherals** tab allows configuration of the Fabric AMBA Master and Fabric AMBA Slave required for the design. Drag and drop the required master/slave to the corresponding subsystem. The following image shows the **Peripherals** tab. Drag and drop the **Fabric Master core** to the **HPMS DDR FIC Subsystem**. This allows the interface to be configured as AXI or single AHB-Lite. On completing the configuration, the selected interface is enabled. The user logic in the FPGA fabric can access the DDR memory through the MDDR using these interfaces.

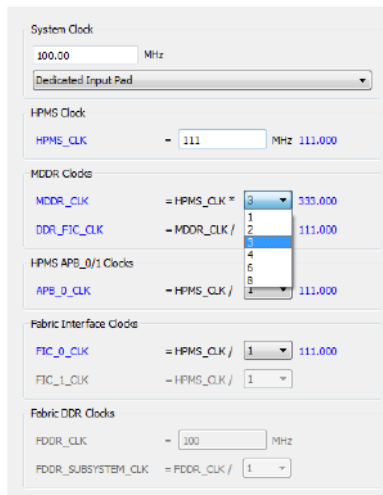
**Figure 17 • System Builder - Peripherals Tab**



8. Navigate to the **Clocks** tab. The **Clocks** tab allows configuration of the **System Clock** and subsystem clocks. The MDDR subsystem operates on MDDR\_CLK, which comes from HPMS\_CCC.

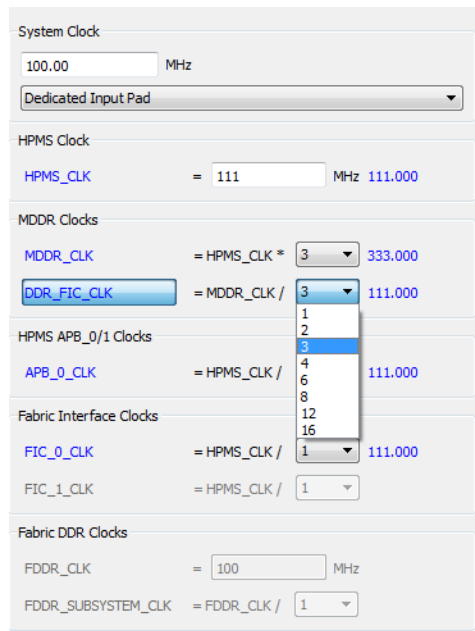
The MDDR\_CLK must be selected as multiples of 1, 2, 3, 4, 6, or 8 of HPMS\_CLK. This clock can be configured using the HPMS\_CCC configurator. The maximum frequency of MDDR\_CLK is 333.33 MHz. The following illustration shows the MDDR\_CLK configuration.

**Figure 18 • MDDR\_CLK Configuration**



DDR\_FIC\_CLK drives the DDR\_FIC slave interface and defines the frequency at which the FPGA fabric subsystem connected to this interface is intended to run. DDR\_FIC\_CLK can be configured as a ratio of MDDR\_CLK (1, 2, 3, 4, 6, 8, 12, 16, or 32) using the Clocks configurator. The maximum frequency of DDR\_FIC\_CLK is 200 MHz. The following illustration shows the DDR\_FIC\_CLK configuration.

**Figure 19 • DDR\_FIC\_CLK Configuration**



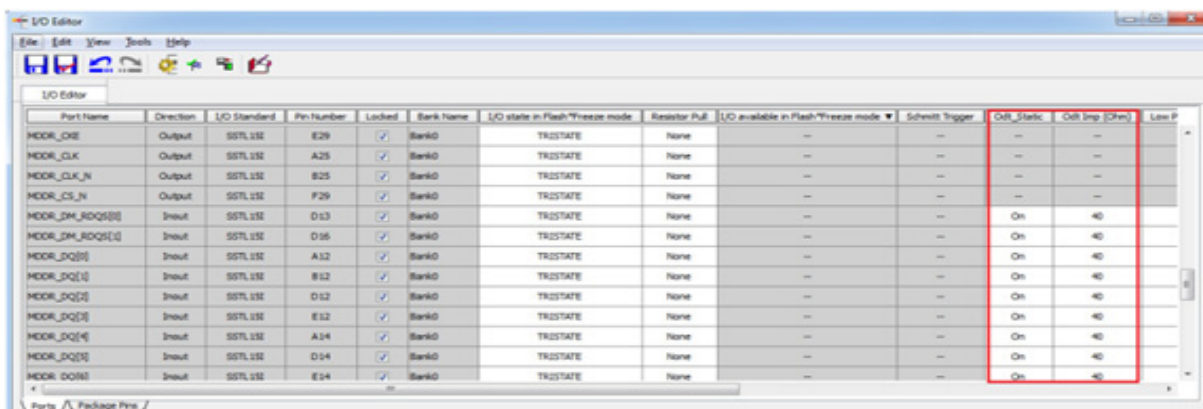
If the MDDR\_CLK ratio to HPMS\_CLK is a multiple of 3, DDR\_FIC\_CLKs ratio to MDDR\_CLK must also be a multiple of 3, and vice versa. The configuration issues an error if this requirement is not met. This limitation is imposed by the internal implementation of the HPMS CCC.

### 3.6.1.1 I/O Configuration

In the **I/O Editor** window, as shown in the following illustration, configure I/O settings such as ODT and drive strength.



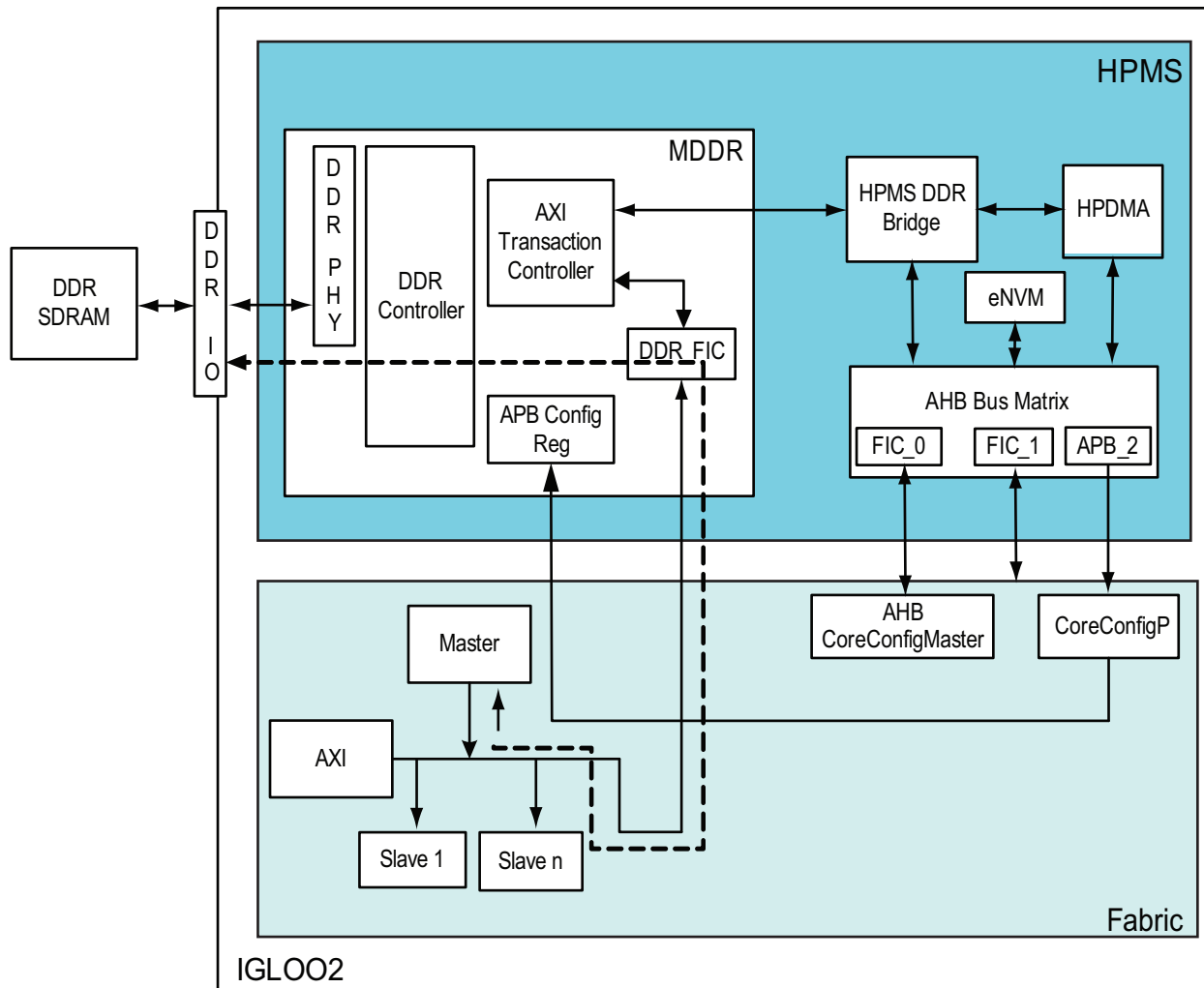
Figure 20 • I/O Editor Window



Port Name	Direction	I/O Standard	Pin Number	Locked	Bank Name	I/O state in Flash/Freeze mode	Resistor Pull	I/O available in Flash/Freeze mode	Schmitt Trigger	OdR Static	OdR Imp (Ohm)	Low P
MDDR_OE	Output	SSTL132	E29	<input checked="" type="checkbox"/>	Bank0	TRIZSTATE	None	—	—	—	—	—
MDDR_CLK	Output	SSTL132	A25	<input checked="" type="checkbox"/>	Bank0	TRIZSTATE	None	—	—	—	—	—
MDDR_CLK_N	Output	SSTL132	B25	<input checked="" type="checkbox"/>	Bank0	TRIZSTATE	None	—	—	—	—	—
MDDR_CS_N	Output	SSTL132	F29	<input checked="" type="checkbox"/>	Bank0	TRIZSTATE	None	—	—	—	—	—
MDDR_DM_A0Q5[0]	Inout	SSTL132	D13	<input checked="" type="checkbox"/>	Bank0	TRIZSTATE	None	—	—	On	40	—
MDDR_DM_A0Q5[1]	Inout	SSTL132	D16	<input checked="" type="checkbox"/>	Bank0	TRIZSTATE	None	—	—	On	40	—
MDDR_DQ[0]	Inout	SSTL132	A12	<input checked="" type="checkbox"/>	Bank0	TRIZSTATE	None	—	—	On	40	—
MDDR_DQ[1]	Inout	SSTL132	B12	<input checked="" type="checkbox"/>	Bank0	TRIZSTATE	None	—	—	On	40	—
MDDR_DQ[2]	Inout	SSTL132	D12	<input checked="" type="checkbox"/>	Bank0	TRIZSTATE	None	—	—	On	40	—
MDDR_DQ[3]	Inout	SSTL132	E12	<input checked="" type="checkbox"/>	Bank0	TRIZSTATE	None	—	—	On	40	—
MDDR_DQ[4]	Inout	SSTL132	A14	<input checked="" type="checkbox"/>	Bank0	TRIZSTATE	None	—	—	On	40	—
MDDR_DQ[5]	Inout	SSTL132	D14	<input checked="" type="checkbox"/>	Bank0	TRIZSTATE	None	—	—	On	40	—
MDDR_DO[0]	Inout	SSTL132	E14	<input checked="" type="checkbox"/>	Bank0	TRIZSTATE	None	—	—	On	40	—

### 3.6.2 Accessing MDDR from FPGA Fabric through the AXI Interface

The AXI master in the FPGA fabric accesses the DDR memory through the MDDR subsystem. The following illustration shows the MDDR subsystem with the AXI interface. The MDDR registers are configured from the FPGA fabric using the CoreConfigMaster IP through the CoreConfigP IP APB interface.

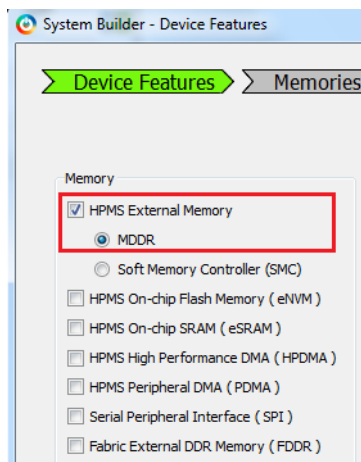
**Figure 21 • MDDR with AXI Interfaces**


Read, write, and read-modify-write transactions are initiated by the AXI master to read from or write the data to the DDR memory after initializing the MDDR registers.

The following steps describe how to access the MDDR from AXI master in the FPGA fabric:

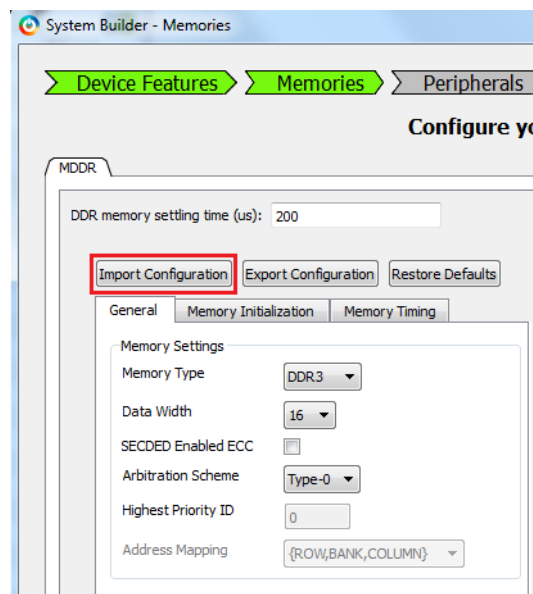
1. Go to the **System Builder - Device Features** tab, check the **HPMS External DDR Memory** check box, and select **MDDR**. Leave the rest of the check boxes unchecked. The following illustration shows the **System Builder - Device Features** tab.

**Figure 22 • System Builder - Device Features Tab**

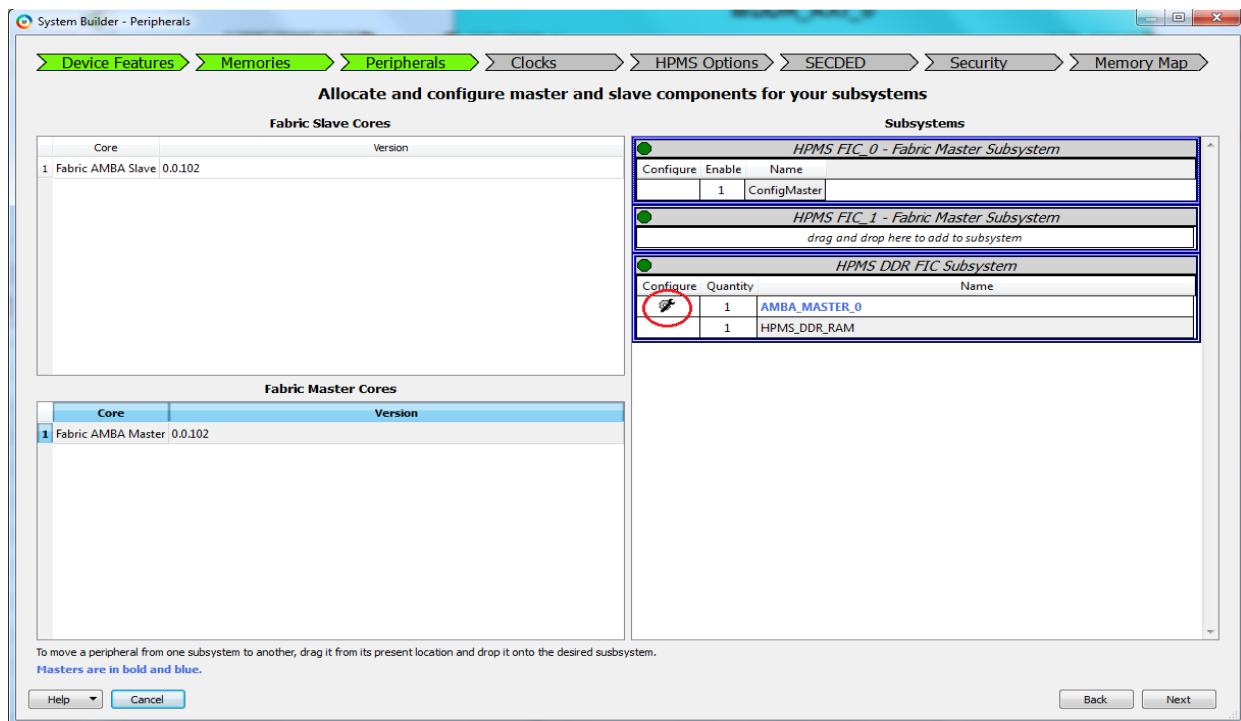


2. Configure the **HPMS External Memory** in the **Memories** tab as shown in the following illustration. In this example, the design is created to access DDR3 memory with a 32-bit data width and no ECC.
3. Set the **DDR memory settling time** to 200 us and click **Import Register Configuration**.

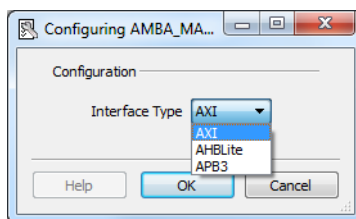
**Figure 23 • Memory Configuration**



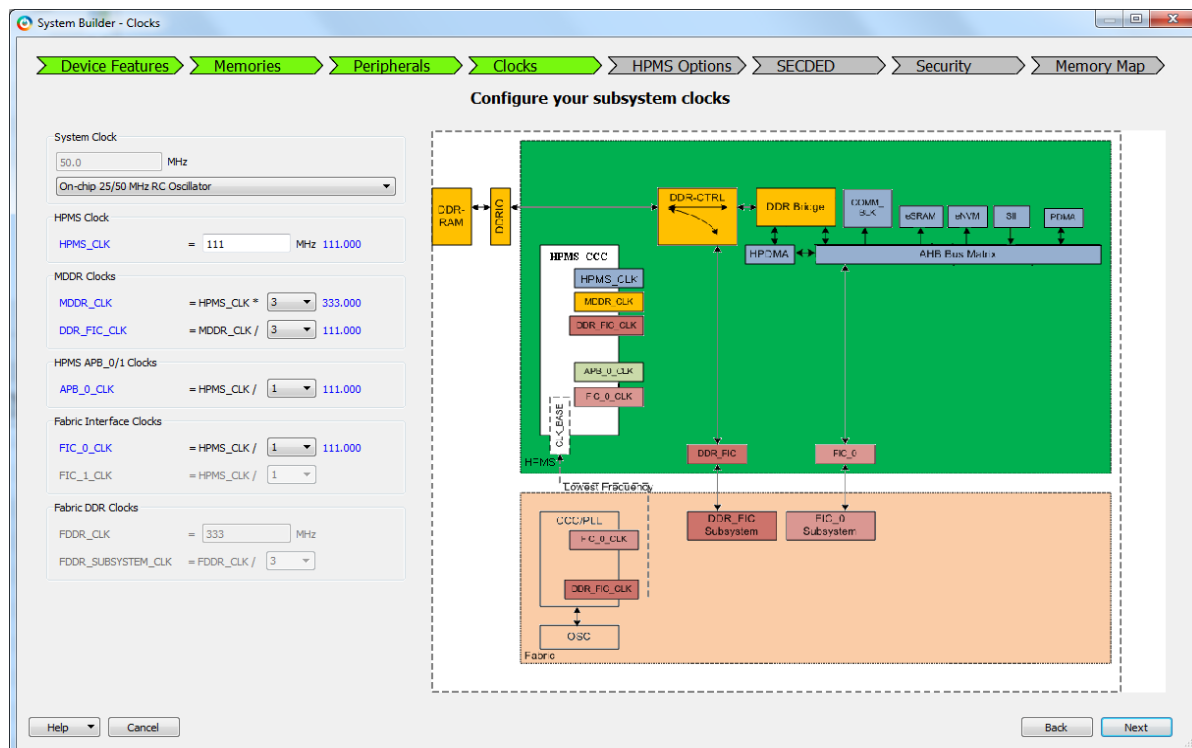
4. Navigate to the **Peripherals** tab.
5. In the **Peripherals** tab, drag the **Fabric Master Core** and drop on to the **HPMS DDR FIC Subsystem**. You can see that the master is added to the subsystem. The following image shows the **Peripherals** tab with the **AMBA\_MASTER\_0** added.
6. Click the **Configure** icon to open the **AMBA Master - Configuration** dialog. The following image shows the **Peripherals** tab with the **Configure** icon highlighted.

**Figure 24 • Peripherals Tab with the Master Added and Configure Icon Highlighted**


7. In the Configuring AMBA\_MASTER\_0 dialog, select the **Interface Type** as **AXI** and then click **OK**.  
The following image shows the **AMBA Master - Configuration** dialog.

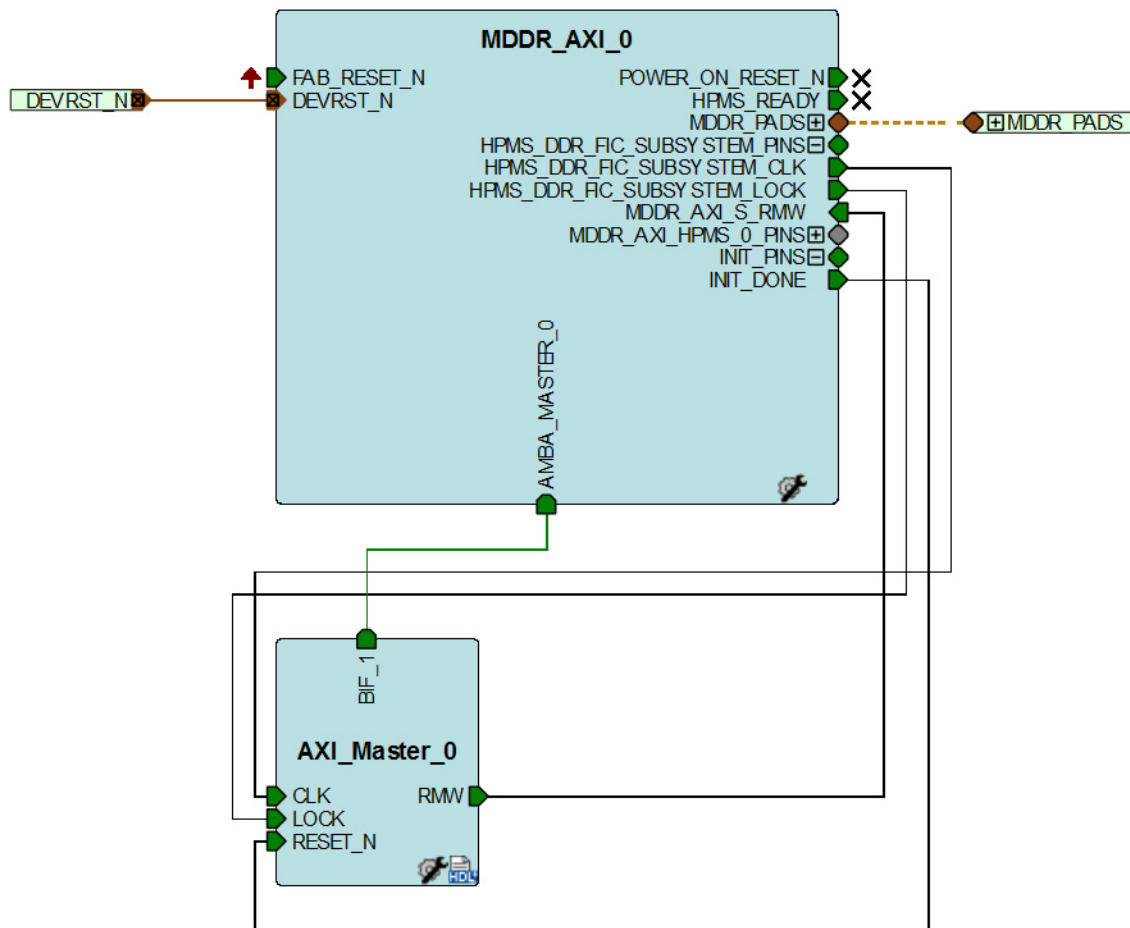
**Figure 25 • AMBA Master Configuration**


8. Configure the **System Clock** and Subsystem clocks in the **Clocks** tab. The following image shows the **Clocks** configuration dialog.
  - Select the **On-chip 25/50 MHz RC oscillator**.
  - Configure **HPMS\_CCC** for **MDDR\_CLK** and **DDR\_FIC\_CLK**.
9. Configure **HPMS\_CLK**, **DDR\_FIC\_CLK**, **APB\_0\_CLK**, **FIC\_0\_CLK** to 111 MHz and **MDDR clock** as 333 MHz.

**Figure 26 • System Clocks Configuration**


10. Navigate to the **Memory Map** tab giving the required data in the rest of the **System Builder** tabs.
11. Instantiate your AXI master logic in the SmartDesign canvas to access the MDDR subsystem through the AXI interface. Ensure that the AXI master logic accesses the MDDR after configuring the MDDR registers (INIT\_DONE indicates the successful MDDR initialization).
12. Connect the AXI\_Master logic signals as follows:
  - RESET\_N to INIT\_DONE
  - CLK to HPMS\_DDR\_FIC\_SUBSYSTEM\_CLK
  - LOCK to HPMS\_DDR\_FIC\_SUBSYSTEM\_LOCK
  - AXI\_S\_RMW to MDDR\_DDR\_AXI\_S\_RMW

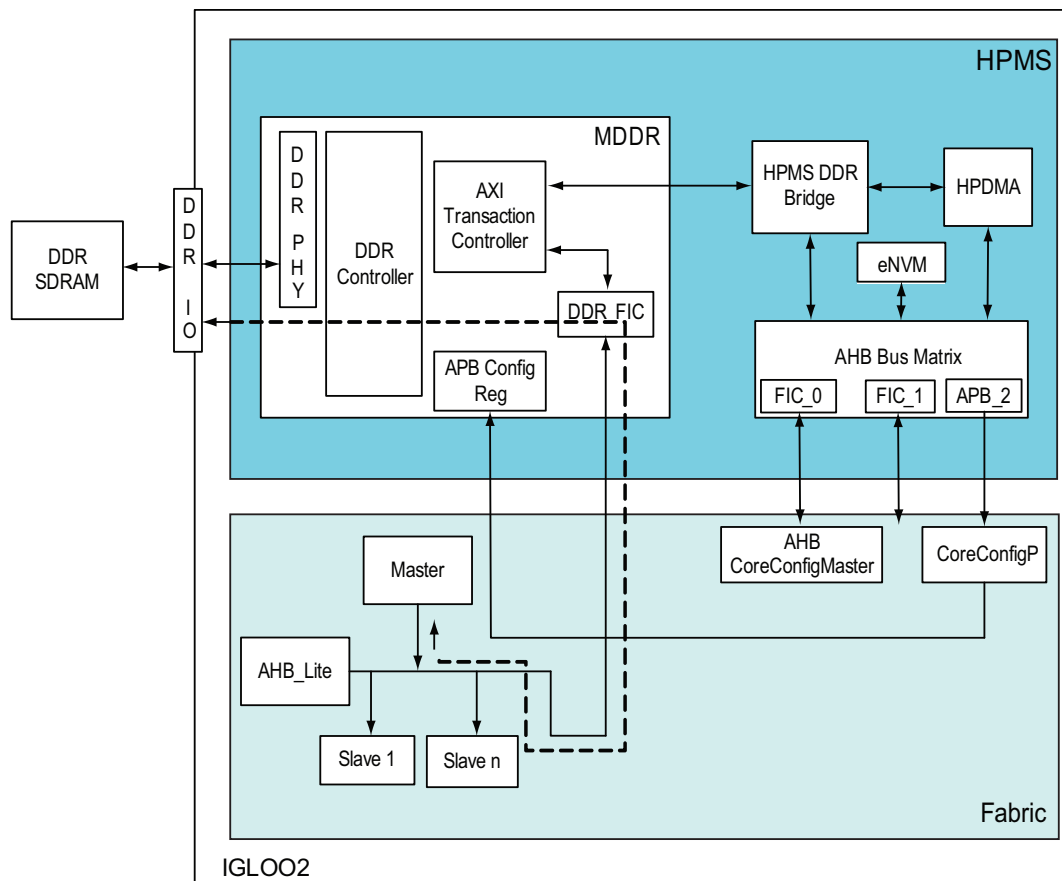
The following illustration shows the rest of the connections in the top level design.

**Figure 27 • SmartDesign Connections (Top Level View)**


For MDDR AXI throughput, see *AC422: SmartFusion2 - Optimizing DDR Controller for Improved Efficiency - Libero v11.7 Application Note*.

### 3.6.3 Accessing MDDR from FPGA Fabric Through the AHB Interface

The MDDR subsystem can be used to access the DDR memory using the AHB-Lite interface. The following illustration shows the MDDR with AHB-Lite interface.

**Figure 28 • MDDR with Single AHB-Lite Interface**

The procedure for accessing the MDDR from AHB master in the FPGA fabric is the same as in ["Accessing MDDR from FPGA Fabric through the AXI Interface"](#) section on page 42—except for the following:

- Configure the **AMBA Master Interface Type** as AHB-Lite in the **HPMS DDR FIC Subsystem** in the **Peripherals** tab of the **System Builder** wizard.

Table 22, page 48 lists the MDDR throughput for the following configuration:

- Fabric Interface: AHB
- MDDR Mode: DDR3
- Fabric Clock to MDDR Clock Ratio: 1:4
- PHY Width: 16 and 32
- Clock Frequency: 80 MHz

The other parameters are configured similar to the MDDR configuration in [AC422: SmartFusion2 - Optimizing DDR Controller for Improved Efficiency - Libero v11.7 Application Note](#).

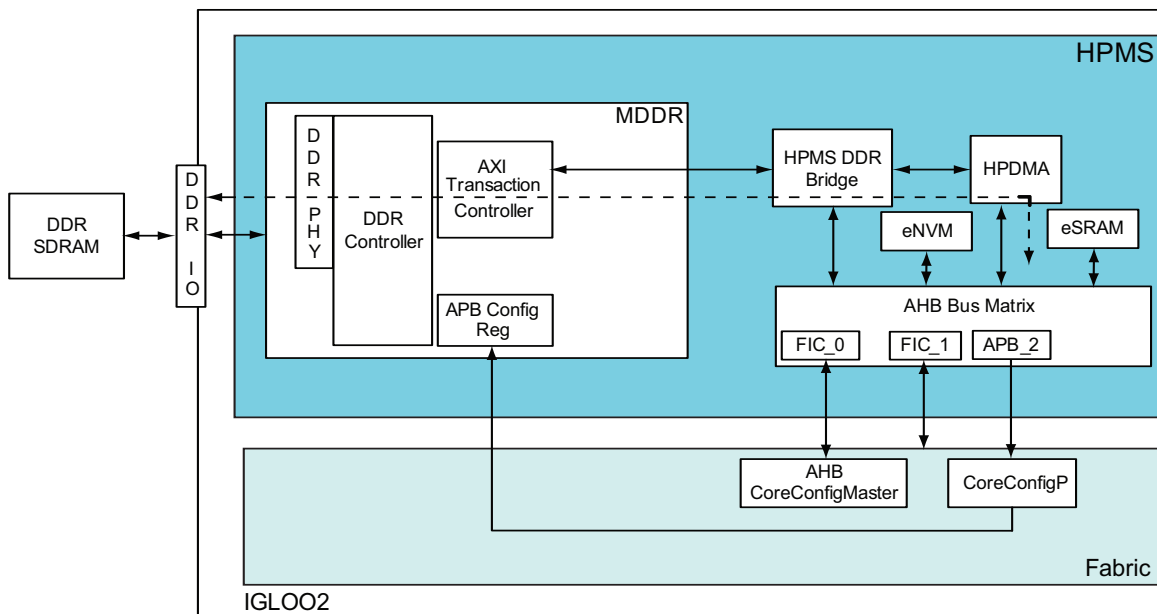
**Table 22 • MDDR Throughput (for AHB)**

MDDR-Fabric Interface-Memory	Frequency Ratio (CLK_BASE:FDDR_CLK)	PHY Width	Write Transaction BW (MB/sec)	Read Transaction BW (MB/sec)
MDDR_AHB-DDR3	1:4 80 MHz:320 MHz	PHY_16	80 MB	79 MB
		PHY_32	80 MB	79 MB

### 3.6.4 Accessing MDDR from the HPDMA

The HPDMA controller can access DDR SDRAM connected to the MDDR subsystem through the HPMS DDR bridge. The following illustration shows the MDDR with HPDMA.

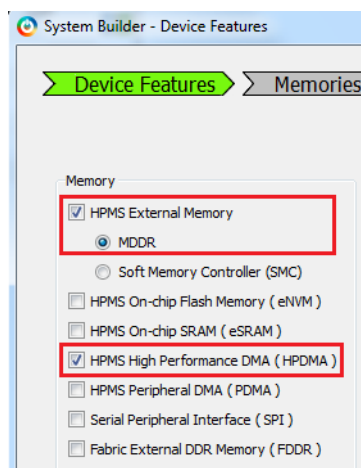
**Figure 29 • MDDR with HPDMA**



The following steps describe how to access the MDDR from HPDMA:

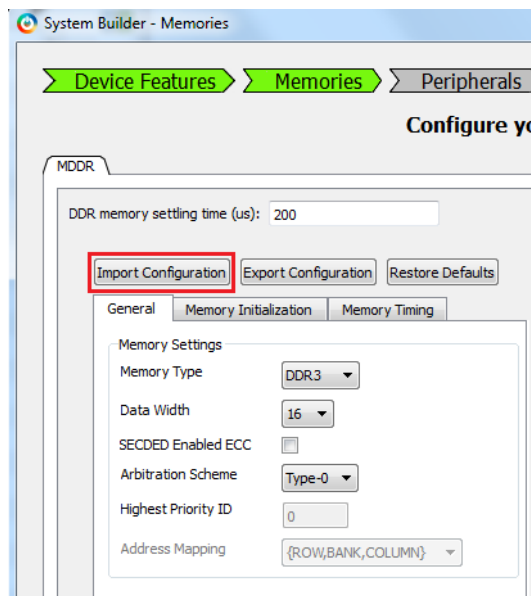
1. Open the **System Builder - Device Features** tab. Check the **HPMS External DDR Memory** check box, select **MDDR** and **HPMS High Performance DMA (HPDMA)** check boxes, leaving the rest of the check boxes unchecked. The following image shows the **System Builder - Device Features** tab.

**Figure 30 • System Builder - Device Features Tab**

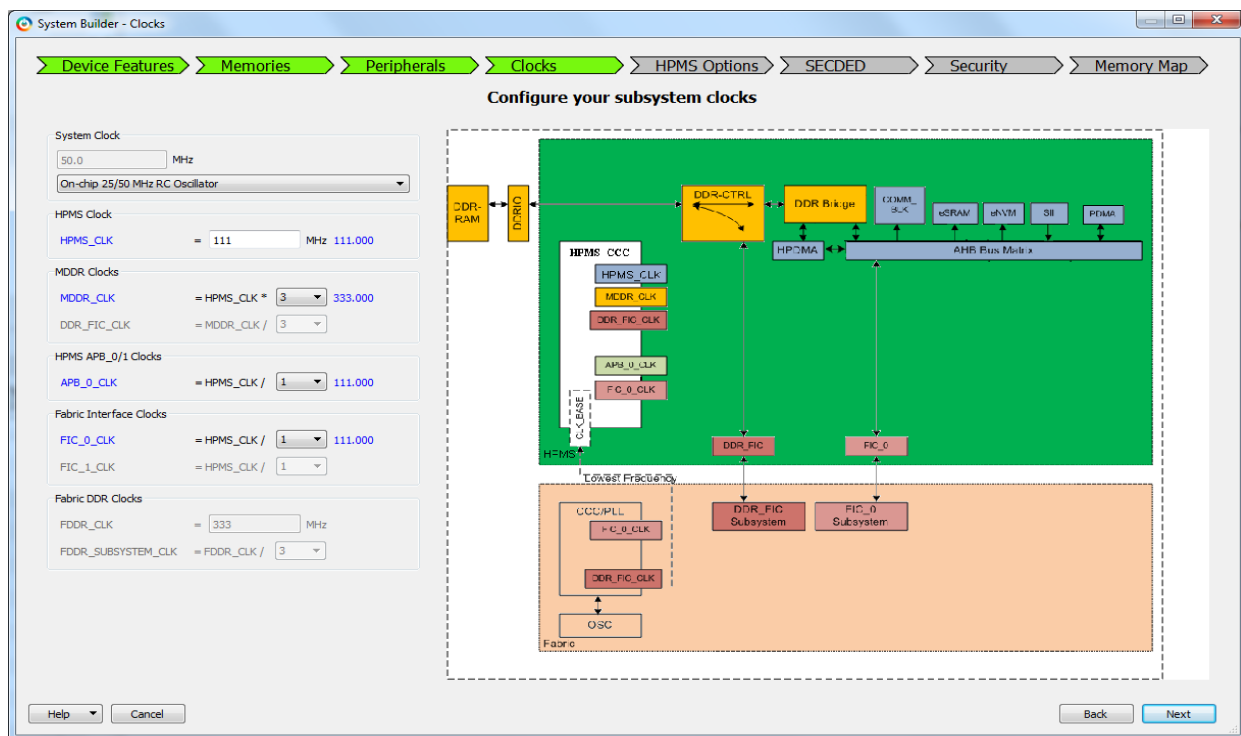


2. Configure the **HPMS External Memory in Memories** tab. as shown in the following image. In this example, the design is created to access the DDR3 memory with a 32-bit data width and no ECC.
3. Set the **DDR memory settling time** to 200 us and click **Import Register Configuration**.



**Figure 31 • Memory Configurations**

4. Configure the **System Clock** and Subsystem clocks in the **Clocks** tab. The following image shows the **Clocks** configuration dialog.
  - Select the **On-chip 25/50 MHz RC Oscillator**
  - Configure **HPMS\_CCC** for **MDDR\_CLK**
5. Configure **HPMS\_CLK**, **APB\_0\_CLK**, **FIC\_0\_CLK** clocks as 111 MHz and the **MDDR\_CLK** clock as 333 MHz.

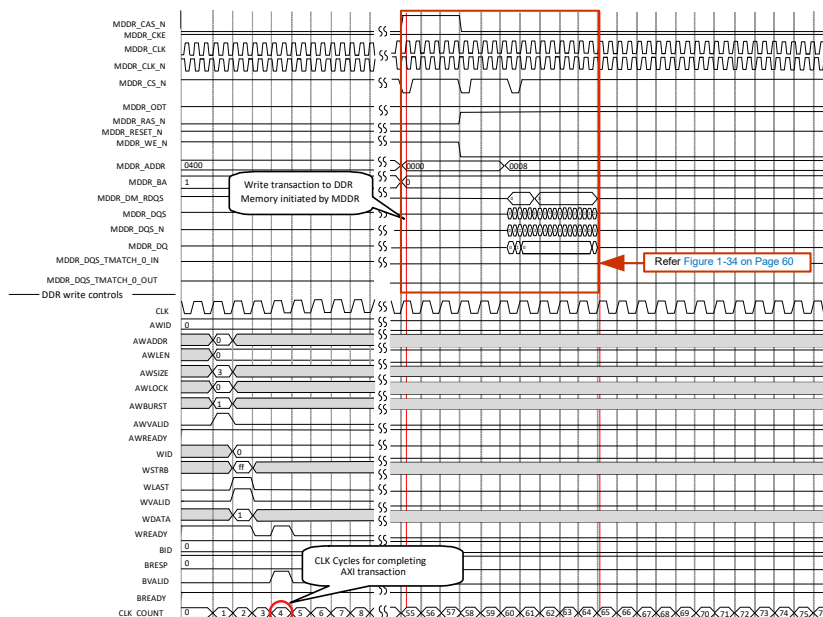
**Figure 32 • Clocks Configuration**

6. Navigate to the **Memory Map** tab giving the required data in the rest of the **System Builder** tabs. For more Information on how to use HPDMA, refer to the HPDMA chapter in UG0448: IGLOO2 High Performance Memory Subsystem User Guide.

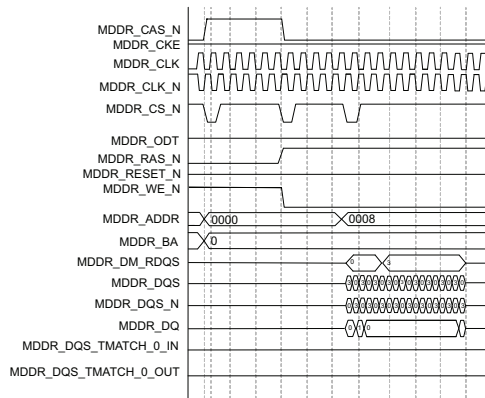
### 3.7 Timing Diagrams

This section shows the operation of the DDR controller with AXI interface with Timing diagrams. The DDR3 16-bit micron memory model is used to perform the read and write transactions from MDDR Fabric Interface (DDR\_FIC). The AXI/AHB clock is configured for 166 MHz and MDDR clock is configured for 332 MHz, that is, FIC clock to MDDR clock ratio is 1:2.

**Figure 33 • AXI Single Write Transaction and Corresponding DDR Controller Commands**



**Figure 34 • DDR Controller Command Sequence for Single AXI Write Transaction**



Timing diagram for MDDR read transaction. The diagram shows various control signals (MDDR\_CLK\_N, MDDR\_CS\_N, MDDR\_ODT, MDDR\_RAS\_N, MDDR\_RESET\_N, MDDR\_VE\_N, MDDR\_ADDR, MDDR\_BA, MDDR\_DM\_RDOS, MDDR\_DQS, MDDR\_DQS\_N, MDDR\_DQ, MDDR\_DQS\_TMATCH\_O\_IN, MDDR\_DQS\_TMATCH\_O\_OUT) and data signals (CLK, ARID, ARADDR, ARLEN, ARSIZE, ARLOCK, ARBURST, ARVALID, ARREADY, RID, RDATA, RVALID, RLAST, RREADY, RRESP, CLK\_CNT) over 20 clock cycles. A red box highlights the read transaction initiated by MDDR, showing the address (00000000), burst length (1), and data (00000000). A callout indicates "Read transaction to DDR Memory initiated by MDDR". Another callout indicates "CLK Cycles for completing transaction" from cycle 18 to 19.

Timing diagram for a DDR write transaction. The diagram shows the relationship between various signals and the completion of the transaction.

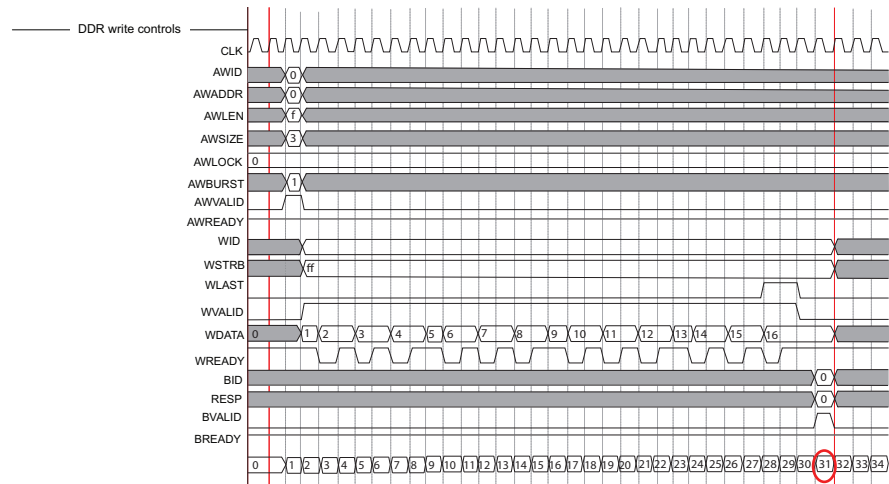
**Top Section: Write transaction initiation**

- Signals: MDDR\_CAS\_N, MDDR\_CKE, MDDR\_CLK, MDDR\_CS\_N, MDDR\_ODT, MDDR\_RAS\_N, MDDR\_RESET\_N, MDDR\_WE\_N, MDDR\_ADDR (0400), MDDR\_BA (1), MDDR\_DM\_RDQS, MDDR\_DQS, MDDR\_DQS\_N, MDDR\_DQ, MDDR\_DQS\_TMATCH\_0\_IN, MDDR\_DQS\_TMATCH\_0\_OUT, mddr\_dqs\_tmatch\_0\_out.
- Callout: "Write transaction to DDR Memory initiated"
- Red box highlights the initiation sequence, with a reference to "Refer Figure 1-34 on page 55".

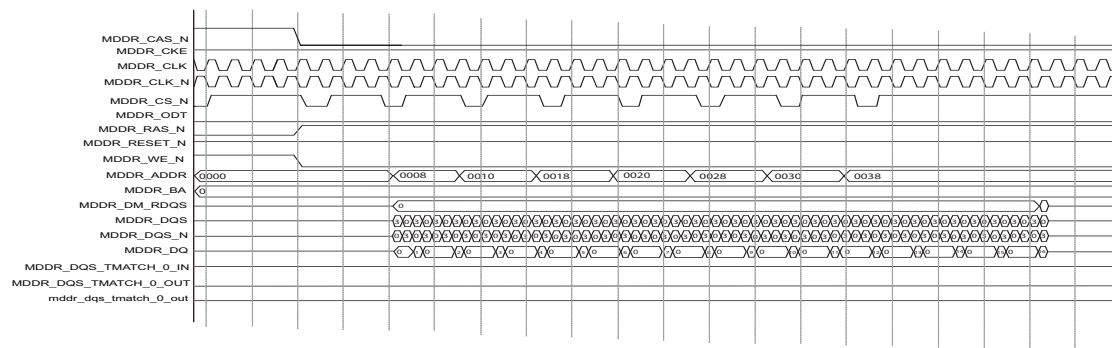
**Bottom Section: Transaction completion**

- Signals: CLK, AWID, AWADDR, AWLEN, AWSIZE, AWLOCK, AWBURST, AWVALID, AWREADY, WID, WSTRB, WLAST, WVALID, WDATA, WREADY, BID, RESP, BVALID, BREADY.
- Callout: "CLK Cycles for completing transaction"
- Red box highlights the completion sequence, with a reference to "Refer Figure 1-33 on page 55".

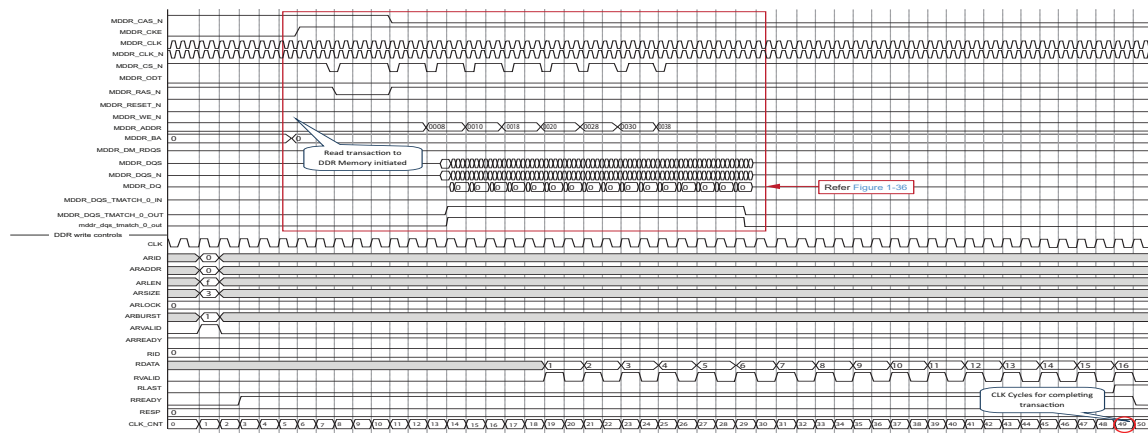
**Figure 37 • AXI INCR16 Write Transaction**

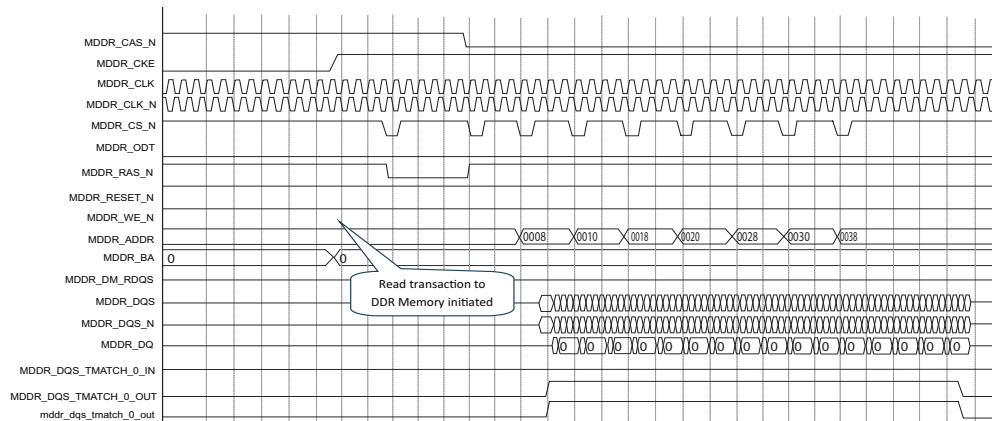


**Figure 38 • DDR Controller Command Sequence for AXI INCR16 Write Transaction**



**Figure 39 • AXI INCR-16 Read Transaction and Corresponding DDR Controller Commands**



**Figure 40 • DDR Controller Command Sequence for AXI INCR-16 Read Transaction**

The following table summarizes the number of cycles to complete the AXI/AHB transactions to MDDR.

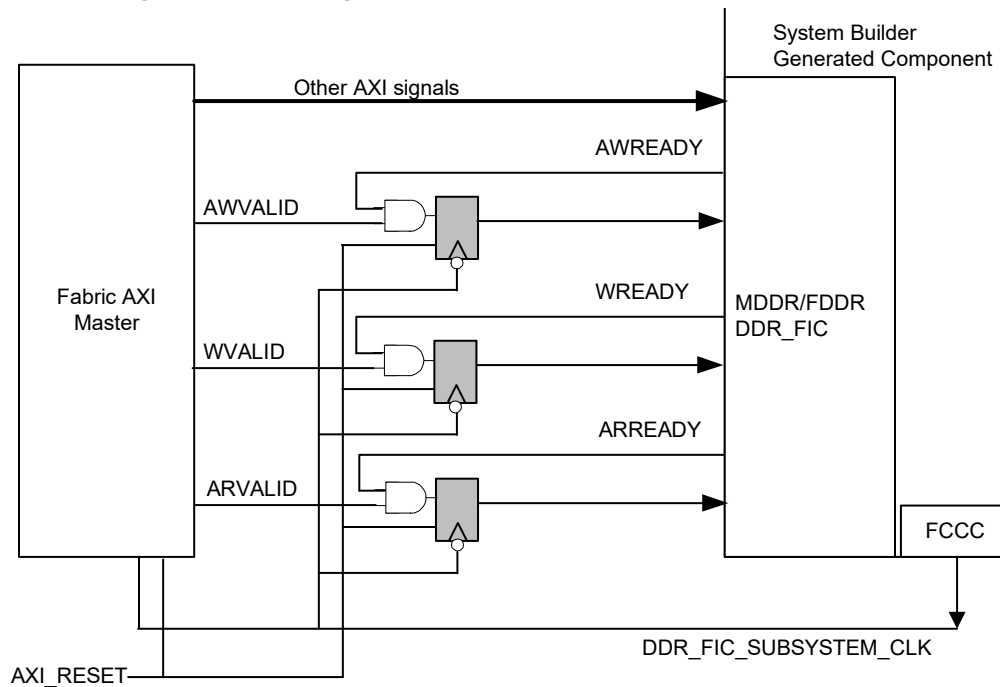
**Table 23 • Number of Cycles for AXI/AHB Transactions to MDDR**

Transaction Type	Write Cycles	Read Cycles
AXI Single	4	19
AXI INCR16 Burst	31	49

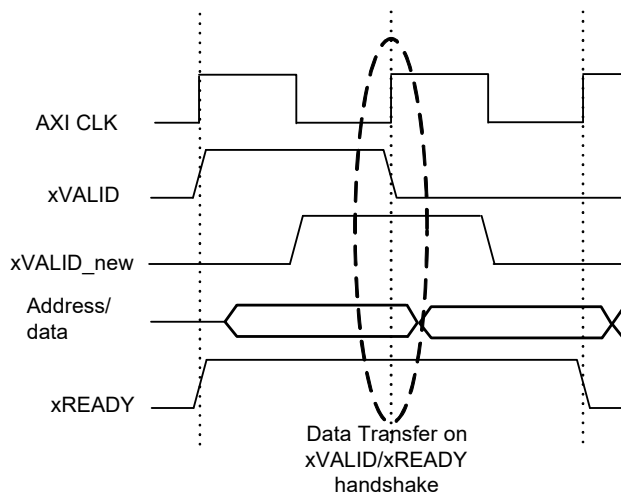
### 3.8 Timing Optimization Technique for AXI

The AXI mode of the MDDR or FDDR provides the highest throughput interface to the external memory device. The best interface ratio for clocking is 2:1 ratio which keeps the fabric clock and fabric interface running at the same rate as the external memory device. For these types of interfaces the following technique provides an optimization method for timing closure when using the 2:1 interface. Timing closure can be achieved by Timing Optimization Technique when the timing closure is not met with the design.

The optimization method can reside between an existing AXI master and the DDR\_FIC AXI slave interface and no changes are required to the AXI master design. The following illustration shows a diagram of the technique, which uses a negative edge register on the VALID lines.

**Figure 41 • AXI Timing Optimization Logic**

The AXI data lines into the DDR\_FIC can now be relaxed with additional half AXI clock cycle as the AXI valid signals are delayed by half AXI clock cycle. The following illustration shows the AXI transaction with the optimization logic.

**Figure 42 • Timing Diagram**

The following SDC constraints need to be added to the timing SDC file. It applies the proper timing relaxation on the DDR\_FIC\_AXI signals.

For FDDR:

*/\* The following constraints provide a relaxation constraint on the signals of 1.5 clock periods.*

*The user should adjust the ddr\_clock\_frequency to match their application. \*/*

```
set ddr_clock_frequency 333
```

```
set delay1 [ expr 3000/$ddr_clock_frequency ]
```

```
set_max_delay $delay1 -to [ get_pins { \
```

```

*/INST_FDDR_IP:F_ARADDR* \
*/INST_FDDR_IP:F_ARBURST* \
*/INST_FDDR_IP:F_ARID* \
*/INST_FDDR_IP:F_ARLEN* \
*/INST_FDDR_IP:F_ARLOCK* \
*/INST_FDDR_IP:F_ARSIZE* \
*/INST_FDDR_IP:F_AWADDR* \
*/INST_FDDR_IP:F_AWBURST* \
*/INST_FDDR_IP:F_AWID* \
*/INST_FDDR_IP:F_AWLEN* \
*/INST_FDDR_IP:F_AWLOCK* \
*/INST_FDDR_IP:F_AWSIZE* \
*/INST_FDDR_IP:F_WDATA* \
*/INST_FDDR_IP:F_WID* \
*/INST_FDDR_IP:F_WLAST \
*/INST_FDDR_IP:F_WSTRB* \
*/INST_FDDR_IP:F_BREADY* \
*/INST_FDDR_IP:F_RMW_AXI \
*/INST_FDDR_IP:F_RREADY* \
}]

```

*/\* The following constraints provide a relaxation constraint on the signals of 1 clock period. \*/*

```

set delay2 [ expr 2000/$ddr_clock_frequency ]
set_max_delay $delay2 -to [ get_pins { \
*/INST_FDDR_IP:F_ARVALID* \
*/INST_FDDR_IP:F_AWVALID* \
*/INST_FDDR_IP:F_WVALID \
} ]

```

For MDDR:

*/\* The following constraints provide a relaxation constraint on the signals of 1.5 clock periods.*

*The user should adjust the ddr\_clock\_frequency to match their application. \*/*

```

set ddr_clock_frequency 333
set delay1 [ expr 3000/$ddr_clock_frequency ]
set_max_delay1 $delay1 -to [ get_pins { \
*/INST_MSS_*_IP:F_ARADDR* \
*/INST_MSS_*_IP:F_ARBURST* \
*/INST_MSS_*_IP:F_ARID* \
*/INST_MSS_*_IP:F_ARLEN* \
*/INST_MSS_*_IP:F_ARLOCK* \
} ]

```

```

*/INST_MSS_*_IP:F_ARSIZE*\
*/INST_MSS_*_IP:F_AWADDR*\
*/INST_MSS_*_IP:F_AWBURST*\
*/INST_MSS_*_IP:F_AWID*\
*/INST_MSS_*_IP:F_AWLEN*\
*/INST_MSS_*_IP:F_AWLOCK*\
*/INST_MSS_*_IP:F_AWSIZE*\
*/INST_MSS_*_IP:F_WDATA*\
*/INST_MSS_*_IP:F_WID*\
*/INST_MSS_*_IP:F_WLAST\
*/INST_MSS_*_IP:F_WSTRB*\
*/INST_MSS_*_IP:F_BREADY\
*/INST_MSS_*_IP:F_RMW_AXI\
*/INST_MSS_*_IP:F_RREADY\
}]

/* The following constraints provide a relaxation constraint on the signals of 1 clock period. */
set delay2 [ expr 2000/$ddr_clock_frequency ]
set_max_delay $delay2 -to [ get_pins { \
*/INST_MSS_*_IP:F_ARVALID*\
*/INST_MSS_*_IP:F_AWVALID*\
*/INST_MSS_*_IP:F_WVALID\
}]

```

## 3.9 DDR Memory Device Examples

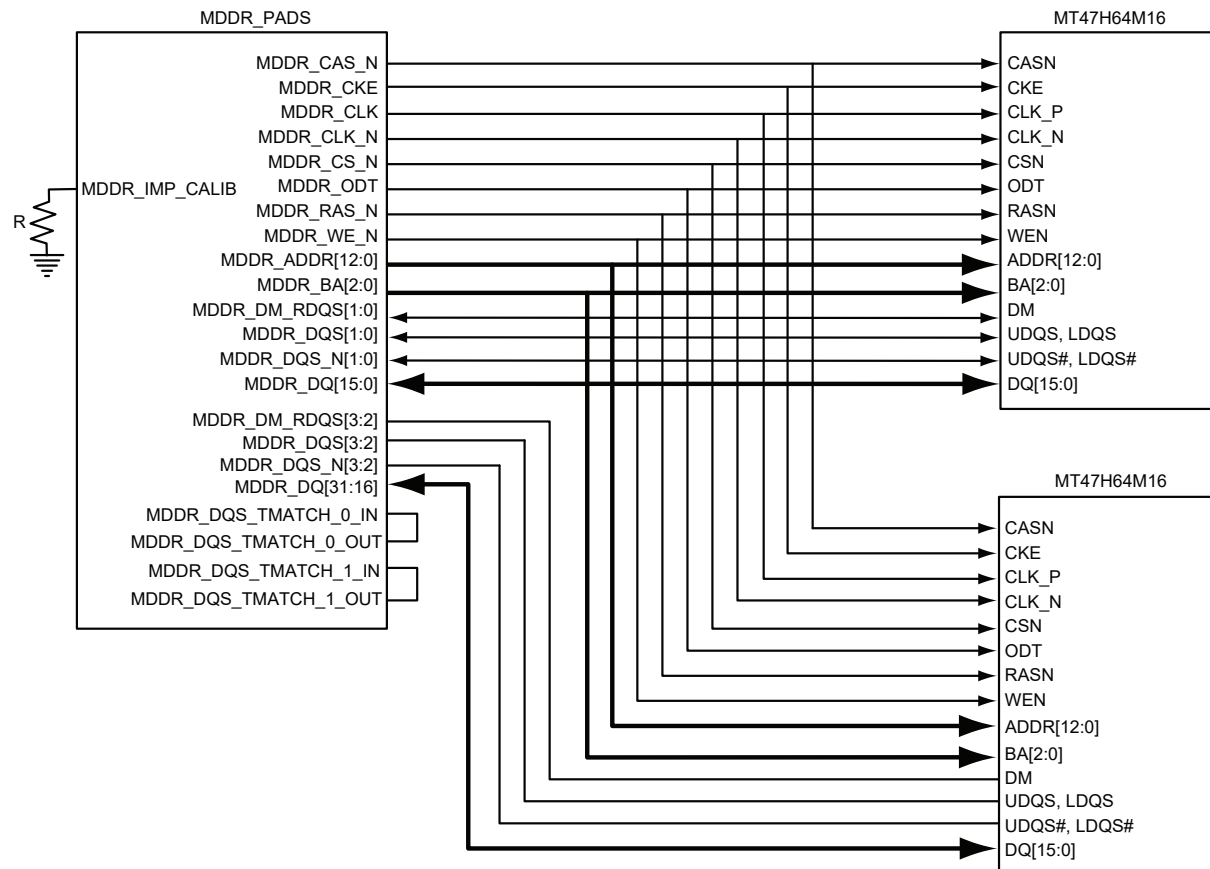
This section describes how to connect DDR memories to IGLOO2 MDDR\_PADs with examples.

**Note:** For more information on requirement of termination resistors, refer to the Datasheets/Application Notes of the memory manufacturers.

### 3.9.1 Example 1: Connecting 32-Bit DDR2 to MDDR\_PADs

The following illustration shows DDR2 SDRAM connected to the MDDR of a IGLOO2 device. Micron's MT47H64M16 is a 128 MB density device with x16 data width. The MDDR is configured in full bus width mode and without SECDED. The total amount of DDR2 memory connected to MDDR is 256 MB.

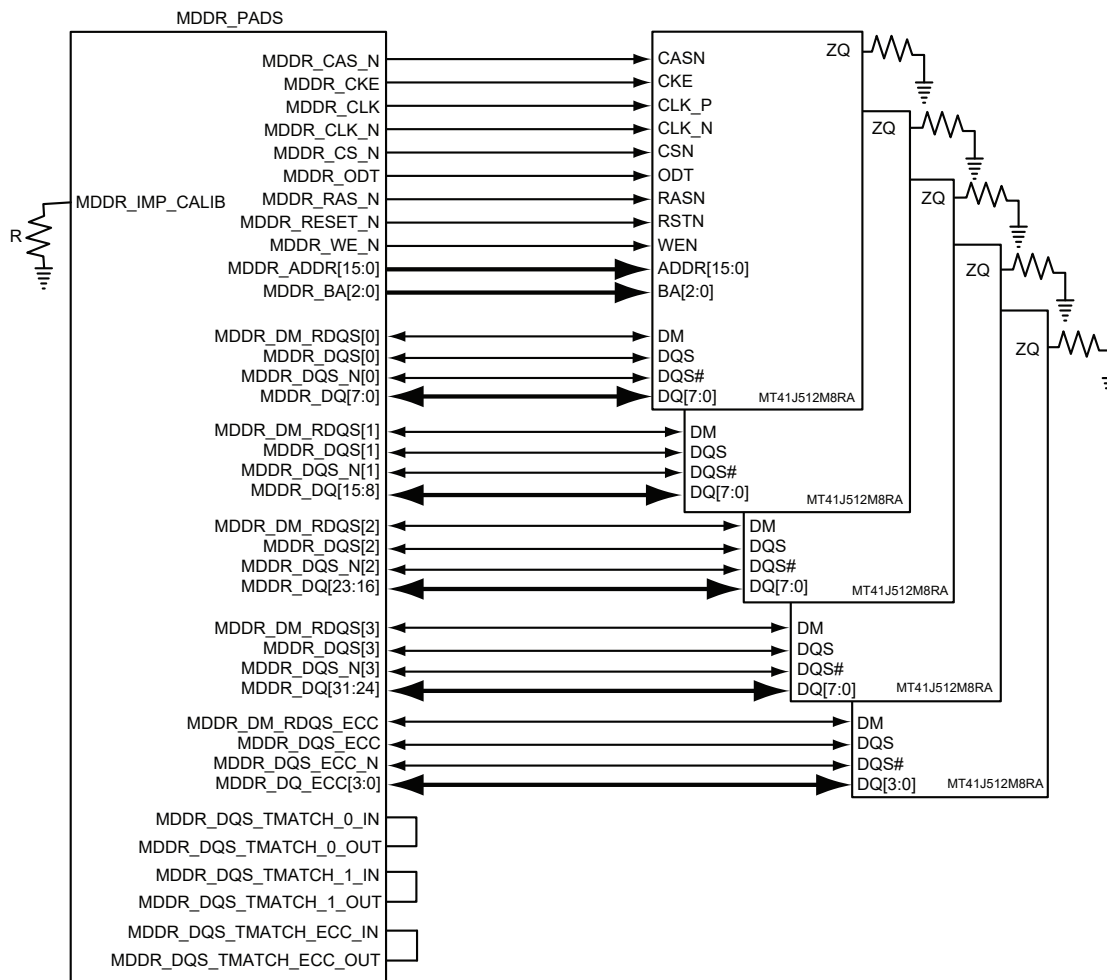


**Figure 43 • x16 DDR2 SDRAM Connected to MDDR**

### 3.9.2 Example 2: Connecting 32-Bit DDR3 to MDDR\_PADs with SECDED

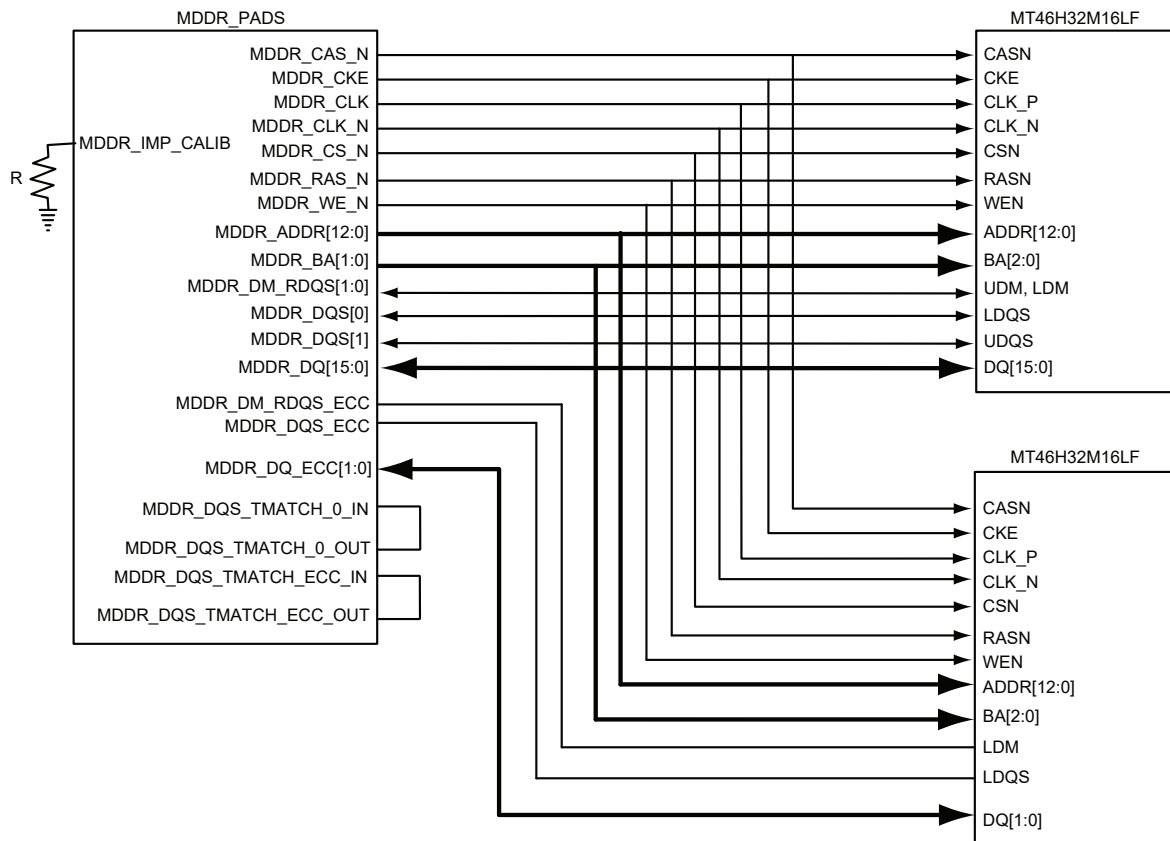
The following illustration shows DDR3 SDRAM connected to the MDDR of a IGLOO2 device. Micron's MT41J512M8RA is a 512 MB density device with x8 data width. The MDDR is configured in full bus width mode with SECDED enabled. The SDRAM connected to MDDR\_DQ\_ECC[3:0] is used to store SECDED bits. The total amount of DDR3 memory (excluding memory for SECDED) connected to MDDR is 2 GB.

**Figure 44 • ×8 DDR3 SDRAM Connection to MDDR**



### 3.9.3 Example 3: Connecting 16-Bit LPDDR to MDDR\_PADs with SECCDED

The following illustration shows LPDDR1 SDRAM connected to the MDDR of a IGLOO2 device. The micron's MT46H32M16LF is a 64 MB density device with x16 data width. The MDDR is configured in full bus width mode with SECCDED enabled. The SDRAM connected to MDDR\_DQ\_ECC[1:0] is used to store SECCDED bits. The total amount of LPDDR1 memory (excluding memory for SECCDED) connected to MDDR is 64 MB.

**Figure 45 • ×16 LPDDR1 SDRAM Connection to MDDR**

### 3.10 Board Design Considerations

MDDR/FDDR subsystems are interfaced with DDR memories through DDRIO. DDRIO is a multi-standard IO optimized for LPDDR, DDR2, and DDR3 performance. The following table lists the IO standards and calibration resistance requirements for MDDR/FDDR to interface with DDR memories.

**Table 24 • I/O Standards and Calibration Resistance Requirements for MDDR/FDDR**

Memory Type	IO Standard	Calibration Resistor
LPDDR	LVC MOS18 LPDDR1(SSTL18)	Not Required* Required
DDR2	SSTL18	Required
DDR3	SSTL15	Required

For more information on IO Standards and Calibration Resistance Requirements, refer to the [AC394: Layout Guidelines for SmartFusion2/IGLOO2-Based Board Design Application Note](#) and [AC393: Board Design Guidelines for SmartFusion2 SoC and IGLOO2 FPGA Application Note](#).

**Note:** For LVC MOS18 IO Standard, the user can optionally calibrate the IO. If calibration is desired, the user must install the appropriate resistor on the PCB.

### 3.11 MDDR Configuration Registers

This section provides MDDR subsystem registers along with the address offset, functionality, and bit definitions. The registers are categorized based on the controller blocks in the MDDR subsystem.

The following table lists the categories of registers and their offset addresses. The base address of the MDDR subsystem registers is 0x40020800.

**Table 25 • Address Table for Register Interfaces**

Registers	Address Offset Space
DDR Controller Configuration Register	0x000:0x1FC
Reserved	0x200:0x3FC
DDR_FIC Configuration Register Summary	0x400:0x4FC
Reserved	0x500:0x7FC

### 3.11.1 SYSREG Configuration Register Summary

In addition to the specific MDDR subsystem registers, the registers listed in the following table also control the behavior of the MDDR subsystem. These registers are located in the SYSREG section of the user's guide and are listed here for convenience. Refer to the "System Register Block" in the [UG0448: IGLOO2 High Performance Memory Subsystem User Guide](#) for a detailed description of each register and associated bits.

**Table 26 • SYSREG Configuration Register Summary**

Register Name	Register Type	Flash Write Protect	Reset Source	Description
MDDR_CR	RW-P	Register	PORESET_N	MDDR Configuration register
MDDR_IO_CALIB_CR	RW-P	Register	PORESET_N	MDDR I/O Calibration Control register
HPMSDDR_PLL_STATUS_LOW_CR	RW-P	Register	CC_RESET_N	Used to control the corresponding configuration input of the MPLL.
HPMSDDR_PLL_STATUS_HIGH_CR	RW-P	Register	CC_RESET_N	Used to control the corresponding configuration input of the MPLL register
HPMSDDR_FACC1_CR	RW-P	Field	CC_RESET_N	HPMS DDR Fabric Alignment Clock Controller 1 Configuration register
HPMSDDR_FACC2_CR	RW-P	Field	CC_RESET_N	HPMS DDR Fabric Alignment Clock Controller 2 Configuration register
HPMSDDR_CLK_CALIB_STATUS	RW-P	Register	SYSRESET_N	Used to start an FPGA fabric calibration test circuit.
DDR_B_CR	RW-P	Register	SYSRESET_N	HPMS DDR bridge configuration register
HPMSDDR_PLL_STATUS	RO	–	–	HPMS DDR PLL Status register
MDDR_IO_CALIB_STATUS	RO	–	PORESET_N	DDR I/O Calibration Status register
HPMSDDR_CLK_CALIB_STATUS	RO	–	SYSRESET_N	HPMS DDR Clock Calibration Status register
SOFT_RESET_CR	RW-P	Bit	SYSRESET_N	Soft reset control register

### 3.11.2 DDR Controller Configuration Register Summary

**Table 27 • DDR Controller Configuration Register**

Register Name	Address Offset	Register Type	Reset Source	Description
DDRC_DYN_SOFT_RESET_CR	0x000	RW/RO	PRESET_N	DDRC Reset register
DDRC_DYN_REFRESH_1_CR	0x008	RW	PRESET_N	DDRC Refresh Control register
DDRC_DYN_REFRESH_2_CR	0x00C	RW	PRESET_N	DDRC Refresh Control register
DDRC_DYN_POWERDOWN_CR	0x010	RW	PRESET_N	DDRC Power-Down Control register
Reserved	0x014	-	-	-
DDRC_MODE_CR	0x018	RW	PRESET_N	DDRC Mode register
DDRC_ADDR_MAP_BANK_CR	0x01C	RW	PRESET_N	DDRC Bank Address Map register
Reserved	0x020	-	-	-
DDRC_ADDR_MAP_COL_1_CR	0x024	RW	PRESET_N	DDRC Column Address Map register
DDRC_ADDR_MAP_COL_2_CR	0x028	RW	PRESET_N	DDRC Column Address Map register
DDRC_ADDR_MAP_ROW_1_CR	0x02C	RW	PRESET_N	DDRC Row Address Map register
DDRC_ADDR_MAP_ROW_2_CR	0x030	RW	PRESET_N	DDRC Row Address Map register
DDRC_INIT_1_CR	0x034	RW	PRESET_N	DDRC Initialization Control register
DDRC_CKE_RSTN_CYCLES_1_CR	0x038	RW	PRESET_N	DDRC Initialization Control register
DDRC_CKE_RSTN_CYCLES_2_CR	0x03C	RW	PRESET_N	DDRC Initialization Control register
DDRC_INIT_MR_CR	0x040	RW	PRESET_N	DDRC MR Initialization register
DDRC_INIT_EMR_CR	0x044	RW	PRESET_N	DDRC EMR Initialization register
DDRC_INIT_EMR2_CR	0x048	RW	PRESET_N	DDRC EMR2 Initialization register
DDRC_INIT_EMR3_CR	0x04C	RW	PRESET_N	DDRC EMR3 Initialization register
DDRC_DRAM_BANK_TIMING_PARAM_CR	0x050	RW	PRESET_N	DDRC DRAM Bank Timing Parameter register
DDRC_DRAM_RD_WR_LATENCY_CR	0x054	RW	PRESET_N	DDRC DRAM Write Latency register
DDRC_DRAM_RD_WR_PRE_CR	0x058	RW	PRESET_N	DDRC DRAM Read-Write Precharge Timing register

**Table 27 • DDR Controller Configuration Register (continued)**

Register Name	Address Offset	Register Type	Reset Source	Description
DDRC_DRAM_MR_TIMING_PARAM_CR	0x05C	RW	PRESET_N	DDRC DRAM Mode Register Timing Parameter register
DDRC_DRAM_RAS_TIMING_CR	0x060	RW	PRESET_N	DDRC DRAM RAS Timing Parameter register
DDRC_DRAM_RD_WR_TRNARND_TIME_CR	0x064	RW	PRESET_N	DDRC DRAM Read Write Turn-around Timing register
DDRC_DRAM_T_PD_CR	0x068	RW	PRESET_N	DDRC DRAM Power-Down Parameter register
DDRC_DRAM_BANK_ACT_TIMING_CR	0x06C	RW	PRESET_N	DDRC DRAM Bank Activate Timing Parameter register
DDRC_ODT_PARAM_1_CR	0x070	RW	PRESET_N	DDRC ODT Delay Control register
DDRC_ODT_PARAM_2_CR	0x074	RW	PRESET_N	DDRC ODT Hold/Block cycles register
DDRC_ADDR_MAP_COL_3_CR	0x078	RW	PRESET_N	Upper byte is DDRC Column Address Map register and lower byte controls debug features.
DDRC_MODE_REG_RD_WR_CR	0x07C	RW	PRESET_N	DDRC Mode Register Read/ Write Command register
DDRC_MODE_REG_DATA_CR	0x080	RW	PRESET_N	DDRC Mode Register Write Data Register
DDRC_PWR_SAVE_1_CR	0x084	RW	PRESET_N	DDRC Power Save register
DDRC_PWR_SAVE_2_CR	0x088	RW	PRESET_N	DDRC Power Save register
DDRC_ZQ_LONG_TIME_CR	0x08C	RW	PRESET_N	DDRC ZQ Long Time Calibration register
DDRC_ZQ_SHORT_TIME_CR	0x090	RW	PRESET_N	DDRC ZQ Short Time Calibration register
DDRC_ZQ_SHORT_INT_REFRESH_MARGIN_1_CR	0x094	RW	PRESET_N	DDRC ZQ Short Time Calibration register
DDRC_ZQ_SHORT_INT_REFRESH_MARGIN_2_CR	0x098	RW	PRESET_N	DDRC ZQ Short Time Calibration register
DDRC_PERF_PARAM_1_CR	0x09C	RW	PRESET_N	DDRC Performance Parameter register
DDRC_HPR_QUEUE_PARAM_1_CR	0x0A0	RW	PRESET_N	DDRC Performance Parameter register
DDRC_HPR_QUEUE_PARAM_2_CR	0x0A4	RW	PRESET_N	DDRC Performance Parameter register
DDRC_LPR_QUEUE_PARAM_1_CR	0x0A8	RW	PRESET_N	DDRC Performance Parameter register
DDRC_LPR_QUEUE_PARAM_2_CR	0x0AC	RW	PRESET_N	DDRC Performance Parameter register

**Table 27 • DDR Controller Configuration Register (continued)**

Register Name	Address Offset	Register Type	Reset Source	Description
DDRC_WR_QUEUE_PARAM_CR	0x0B0	RW	PRESET_N	DDRC Performance Parameter register
DDRC_PERF_PARAM_2_CR	0x0B4	RW	PRESET_N	DDRC Performance Parameter register
DDRC_PERF_PARAM_3_CR	0x0B8	RW	PRESET_N	DDRC Performance Parameter register
DDRC_DFI_RDDATA_EN_CR	0x0BC	RW	PRESET_N	DDRC DFI Read Command Timing register
DDRC_DFI_MIN_CTRLUPD_TIMING_CR	0x0C0	RW	PRESET_N	DDRC DFI Controller Update Min Time register
DDRC_DFI_MAX_CTRLUPD_TIMING_CR	0x0C4	RW	PRESET_N	DDRC DFI Controller Update Max Time register
Reserved	-	-	-	-
Reserved	-	-	-	-
Reserved	-	-	-	-
Reserved	-	-	-	-
Reserved	-	-	-	-
DDRC_DYN_SOFT_RESET_ALIAS_CR	0x0DC	RW	PRESET_N	DDRC reset register
DDRC_AXI_FABRIC_PRI_ID_CR	0x0E0	RW	PRESET_N	DDRC AXI Interface Fabric Priority ID Register
DDRC_SR	0x0E4	RO	PRESET_N	DDRC Status register
<b>SECDED Registers</b>				
DDRC_SINGLE_ERR_CNT_STATUS_SR	0x0E8	RO	PRESET_N	DDRC single error count Status register
DDRC_DOUBLE_ERR_CNT_STATUS_SR	0x0EC	RO	PRESET_N	DDRC double error count status register
DDRC_LUE_SYNDROME_1_SR	0x0F0	RO	PRESET_N	DDRC last uncorrected error syndrome register
DDRC_LUE_SYNDROME_2_SR	0x0F4	RO	PRESET_N	DDRC last uncorrected error syndrome register
DDRC_LUE_SYNDROME_3_SR	0x0F8	RO	PRESET_N	DDRC last uncorrected error syndrome register
DDRC_LUE_SYNDROME_4_SR	0x0FC	RO	PRESET_N	DDRC last uncorrected error syndrome register
DDRC_LUE_SYNDROME_5_SR	0x100	RO	PRESET_N	DDRC last uncorrected error syndrome register
DDRC_LUE_ADDRESS_1_SR	0x104	RO	PRESET_N	DDRC last uncorrected error address register
DDRC_LUE_ADDRESS_2_SR	0x108	RO	PRESET_N	DDRC last uncorrected error address register
DDRC_LCE_SYNDROME_1_SR	0x10C	RO	PRESET_N	DDRC last corrected error syndrome register

**Table 27 • DDR Controller Configuration Register (continued)**

Register Name	Address Offset	Register Type	Reset Source	Description
DDRC_LCE_SYNDROME_2_SR	0×110	RO	PRESET_N	DDRC last corrected error syndrome register
DDRC_LCE_SYNDROME_3_SR	0×114	RO	PRESET_N	DDRC last corrected error syndrome register
DDRC_LCE_SYNDROME_4_SR	0×118	RO	PRESET_N	DDRC last corrected error syndrome register
DDRC_LCE_SYNDROME_5_SR	0×11C	RO	PRESET_N	DDRC last corrected error syndrome register
DDRC_LCE_ADDRESS_1_SR	0×120	RO	PRESET_N	DDRC last corrected error address register
DDRC_LCE_ADDRESS_2_SR	0×124	RO	PRESET_N	DDRC last corrected error address register
DDRC_LCB_NUMBER_SR	0×128	RO	PRESET_N	DDRC last corrected bit number register
DDRC_LCB_MASK_1_SR	0×12C	RO	PRESET_N	DDRC last corrected bit mask status register
DDRC_LCB_MASK_2_SR	0×130	RO	PRESET_N	DDRC last corrected bit mask status register
DDRC_LCB_MASK_3_SR	0×134	RO	PRESET_N	DDRC last corrected bit mask status register
DDRC_LCB_MASK_4_SR	0×138	RO	PRESET_N	DDRC last corrected bit mask status register
DDRC_ECC_INT_SR	0×13C	RO	PRESET_N	DDRC SECEDED interrupt status register
DDRC_ECC_INT_CLR_REG	0×140	RW	PRESET_N	DDRC SECEDED interrupt clear register



### 3.11.3 DDR Controller Configuration Register Bit Definitions

**Table 28 • DDRC\_DYN\_SOFT\_RESET\_CR**

Bit Number	Name	Reset Value	Description
[31:3]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	AXIRESET	0×1	Set when main AXI reset signal is asserted. Reads and writes to the dynamic registers should not be carried out. This is a read only bit.
1	RESET_APB_REG	0×0	Full soft reset If this bit is set when the soft reset bit is written as 1, all APB registers reset to the power-up state.
0	REG_DDRC_SOFT_RSTB	0×0	This is a soft reset. 0: Puts the controller into reset. 1: Takes the controller out of reset. The controller should be taken out of reset only when all other registers have been programmed. Asserting this bit does NOT reset all the APB configuration registers. Once the soft reset bit is asserted, the APB register should be modified as required.

**Table 29 • DDRC\_DYN\_REFRESH\_1\_CR**

Bit Number	Name	Reset Value	Description
[31:15]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[14:7]	REG_DDRC_T_RFC_MIN	0×23	$t_{RFC(min)}$ – Minimum time from refresh to refresh or activate (specification: 75 ns to 195 ns). Unit: clocks.
6	REG_DDRC_REFRESH_UPDATE_LEVEL	0×0	Toggle this signal to indicate that the refresh register(s) have been updated. The value is automatically updated when exiting soft reset, so it does not need to be toggled initially.
5	REG_DDRC_SELFREF_EN	0×0	If 1, then the controller puts the DRAM into self refresh when the transaction store is empty.
[4:0]	REG_DDRC_REFRESH_TO_X32	0×8	Speculative refresh

**Table 30 • DDRC\_DYN\_REFRESH\_2\_CR**

Bit Number	Name	Reset Value	Description
------------	------	-------------	-------------

**Table 30 • DDRC\_DYN\_REFRESH\_2\_CR**

[31:15]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[14:3]	REG_DDRC_T_RFC_NOM_X32	0x52	$t_{REFI}$ : Average time between refreshes (specification: 7.8 $\mu$ s). Unit: multiples of 32 clocks.
[2:0]	REG_DDRC_REFRESH_BURST	0x0	<p>The programmed value plus one is the number of refresh timeouts that is allowed to accumulate before traffic is blocked and the refreshes are forced to execute. Closing pages to perform a refresh is a one-time penalty that must be paid for each group of refreshes; therefore, performing refreshes in a burst reduces the per-refresh penalty of these page closings.</p> <p>Higher numbers for burst_of_N_refresh slightly increases utilization; lower numbers decreases the worst-case latency associated with refreshes.</p> <p>0x0: Single refresh            0x1: Burst-of-2            0x7: Burst-of-8 refresh</p>

**Table 31 • DDRC\_DYN\_POWERDOWN\_CR**

Bit Number	Name	Reset Value	Description
[31:2]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	REG_DDRC_POWERDOWN_EN	0x1	If true, the controller goes into power-down after a programmable number of cycles (REG_DDRC_POWERDOWN_TO_X32). This register bit may be reprogrammed during the course of normal operation.
0	REG_DDRC_DEEPPOWERDOWN_EN	0x0	1: Controller puts the DRAM into deep power-down mode when the transaction store is empty. 0: Brings controller out of deep power-down mode. Present only in designs that have mobile support.

**Table 32 • DDRC\_MODE\_CR**

Bit Number	Name	Reset Value	Description
[31:9]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	REG_DDRC_DDR3	0x0	1: DDR3 operating mode 0: DDR2 operating mode

**Table 32 • DDRC\_MODE\_CR**

7	REG_DDRC_MOBILE	0×0	1: Mobile/LPDDR1 DRAM device in use 0: Non-mobile DRAM device in use
6	REG_DDRC_SDRAM	0×0	1: SDRAM mode 0: Non-SDRAM mode. Only present in designs that support SDRAM and/or mSDR devices.
5	REG_DDRC_TEST_MODE	0×0	1: Controller is in test mode 0: Controller is in normal mode
[4:2]	REG_DDRC_MODE	0×0	DRAM SECEDED mode 000: No SECEDED 101: SECEDED enabled All other selections are reserved.
[1:0]	REG_DDRC_DATA_BUS_WIDTH	0×0	00: Full DQ bus width to DRAM 01: Half DQ bus width to DRAM 10: Quarter DQ bus width to DRAM 11: Reserved <b>Note:</b> The half bus width modes are only supported when the DRAM bus width is a multiple of 16.

**Table 33 • DDRC\_ADDR\_MAP\_BANK\_CR**

Bit Number	Name	Reset Value	Description
[31:12]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[11:8]	REG_DDRC_ADDRMAP_BANK_B0	0×0	Selects the address bits used as bank address bit 0. Valid Range: 0 to 14 Internal Base: 2 The selected address bit for each of the bank address bits is determined by adding the internal base to the value of this field.
[7:4]	REG_DDRC_ADDRMAP_BANK_B1	0×0	Selects the address bits used as bank address bit 1. Valid Range: 0 to 14 Internal Base: 3 The selected address bit for each of the bank address bits is determined by adding the internal base to the value of this field.
[3:0]	REG_DDRC_ADDRMAP_BANK_B2	0×0	Selects the address bits used as bank address bit 2. Valid Range: 0 to 14 and 15 Internal Base: 4 The selected address bit is determined by adding the internal base to the value of this field. If set to 15, bank address bit 2 is set to 0.

**Table 34 • DDRC\_ADDR\_MAP\_COL\_1\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:12]	REG_DDRC_ADDRMAP_COL_B2	0×0	<b>Full bus width mode:</b> Selects column address bit 3. <b>Half bus width mode:</b> Selects column address bit 4. <b>Quarter bus width mode:</b> Selects column address bit 5. Valid range: 0 to 7 Internal base: 2 The selected address bit is determined by adding the internal base to the value of this field.
[11:8]	REG_DDRC_ADDRMAP_COL_B3	0×0	<b>Full bus width mode:</b> Selects column address bit 4. <b>Half bus width mode:</b> Selects column address bit 5. <b>Quarter bus width mode:</b> Selects column address bit 6. Valid range: 0 to 7 Internal base: 3 The selected address bit is determined by adding the internal base to the value of this field.
[7:4]	REG_DDRC_ADDRMAP_COL_B4	0×0	<b>Full bus width mode:</b> Selects column address bit 5. <b>Half bus width mode:</b> Selects column address bit 6. <b>Quarter bus width mode:</b> Selects column address bit 7. Valid Range: 0 to 7 Internal base: 4 The selected address bit for each of the column address bits is determined by adding the internal base to the value of this field.
[3:0]	REG_DDRC_ADDRMAP_COL_B7	0×0	<b>Full bus width mode:</b> Selects column address bit 8. <b>Half bus width mode:</b> Selects column address bit 9. <b>Quarter bus width mode:</b> Selects column address bit 11. Valid range: 0 to 7, and 15 Internal base: 7 The selected address bit is determined by adding the internal base to the value of this field. If set to 15, column address bit 9 is set to 0. <b>Note:</b> Per JEDEC DDR2 specification, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10.

**Table 35 • DDRC\_ADDR\_MAP\_COL\_2\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 35 • DDRC\_ADDR\_MAP\_COL\_2\_CR**

[15:12]	REG_DDRC_ADDRMAP_COL_B8	0×0	<p><b>Full bus width mode:</b> Selects column address bit 9.  <b>Half bus width mode:</b> Selects column address bit 11.  <b>Quarter bus width mode:</b> Selects column address bit 12.            Valid range: 0 to 7, and 15            Internal base: 8            The selected address bit is determined by adding the internal base to the value of this field. If set to 15, column address bit 9 is set to 0.</p> <p><b>Note:</b> Per JEDEC DDR2 specification, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10.</p>
[11:8]	REG_DDRC_ADDRMAP_COL_B9	0×0	<p><b>Full bus width mode:</b> Selects column address bit 11.  <b>Half bus width mode:</b> Selects column address bit 12.  <b>Quarter bus width mode:</b> Selects column address bit 13.            Valid range: 0 to 7, and 15            Internal base: 9            The selected address bit is determined by adding the internal base to the value of this field. If set to 15, column address bit 9 is set to 0.</p>
[7:4]	REG_DDRC_ADDRMAP_COL_B10	0×0	<p><b>Full bus width mode:</b> Selects column address bit 12.  <b>Half bus width mode:</b> Selects column address bit 13.  <b>Quarter bus width mode:</b> Unused. Should be set to 15.            Valid range: 0 to 7, and 15            Internal base: 10            The selected address bit is determined by adding the internal base to the value of this field. If set to 15, column address bit 10 is set to 0.</p>
[3:0]	REG_DDRC_ADDRMAP_COL_B11	0×0	<p><b>Full bus width mode:</b> Selects column address bit 13.  <b>Half bus width mode:</b> Unused. To make it unused, this should be tied to 0xF.  <b>Quarter bus width mode:</b> Unused. To make it unused, this should be tied to 0xF.            Valid range: 0 to 7, and 15            Internal base: 11            The selected address bit is determined by adding the internal base to the value of this field. If set to 15, column address bit 11 is set to 0.</p>

**Table 36 • DDRC\_ADDR\_MAP\_ROW\_1\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 36 • DDRC\_ADDR\_MAP\_ROW\_1\_CR**

[15:12]	REG_DDRC_ADDRMAP_ROW_B0	0×0	Selects the address bits used as row address bit 0. Valid range: 0 to 11 Internal base: 6 The selected address bit for each of the row address bits is determined by adding the internal base to the value of this field.
[11:8]	REG_DDRC_ADDRMAP_ROW_B1	0×0	Selects the address bits used as row address bit 1. Valid range: 0 to 11 Internal base: 7 The selected address bit for each of the row address bits is determined by adding the internal base to the value of this field.
[7:4]	REG_DDRC_ADDRMAP_ROW_B2_11	0×0	Selects the address bits used as row address bits 2 to 11. Valid Range: 0 to 11 Internal Base: 8 for row address bit 2 9 for row address bit 3 10 for row address bit 4 .... 15 for row address bit 9 16 for row address bit 10 17 for row address bit 11 The selected address bit for each of the row address bits is determined by adding the internal base to the value of this field.
[3:0]	REG_DDRC_ADDRMAP_ROW_B12	0×0	Selects the address bit used as row address bit 12. Valid Range: 0 to 11, and 15 Internal Base: 18 The selected address bit is determined by adding the internal base to the value of this field. If set to 15, row address bit 12 is set to 0.

**Table 37 • DDRC\_ADDR\_MAP\_ROW\_2\_CR**

Bit Number	Name	Reset Value	Description
[31:12]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[11:8]	REG_DDRC_ADDRMAP_ROW_B13	0×0	Selects the address bits used as row address bit 13. Valid range: 0 to 11, and 15 Internal base: 19 The selected address bit is determined by adding the internal base to the value of this field. If set to 15, row address bit 13 is set to 0.
[7:4]	REG_DDRC_ADDRMAP_ROW_B14	0×0	Selects the address bit used as row address bit 14. Valid range: 0 to 11, and 15 Internal base: 20 The selected address bit is determined by adding the internal base to the value of this field. If set to 15, row address bit 14 is set to 0.

**Table 37 • DDRC\_ADDR\_MAP\_ROW\_2\_CR**

[3:0]	REG_DDRC_ADDRMAP_ROW_B15	0×0	<p>Selects the address bit used as row address bit 15.</p> <p>Valid range: 0 to 11, and 15</p> <p>Internal base: 21</p> <p>The selected address bit is determined by adding the internal base to the value of this field.</p> <p>If set to 15, row address bit 15 is set to 0.</p>
-------	--------------------------	-----	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Table 38 • DDRC\_INIT\_1\_CR**

Bit Number	Name	Reset Value	Description
[31:12]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[11:8]	REG_DDRC_PRE_OCD_X32	0×0	<p>Wait period before driving the on chip driver calibration (OCD) Complete command to DRAM.</p> <p>Units are in counts of a global timer that pulses every 32 clock cycles.</p> <p>There is no known specific requirement for this. It may be set to zero.</p>
[7:1]	REG_DDRC_FINAL_WAIT_X32	0×0	<p>Cycles to wait after completing the DRAM initialization sequence before starting the dynamic scheduler.</p> <p>Units are in counts of a global timer that pulses every 32 clock cycles.</p> <p>There is known specific requirement for this; it may be set to zero.</p>
0	REG_DDRC_SKIP_OCD	0×1	<p>This register must be kept at 1.</p> <p>1: Indicates the controller is to skip the on chip driver calibration (OCD) adjustment step during DDR2 initialization. OCD_Default and OCD_Exit are performed instead.</p> <p>0: Not supported</p>

**Table 39 • DDRC\_CKE\_RSTN\_CYCLES\_1\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 39 • DDRC\_CKE\_RSTN\_CYCLES\_1\_CR**

[15:8]	REG_DDRC_PRE_CKE_X1024	0×0	The 10-bit REG_DDRC_PRE_CKE_X1024 [9:0] value is split across the two registers: DDRC_CKE_RSTN_CYCLES_1_CR and DDRC_CKE_RSTN_CYCLES_2_CR. [7:0] bits of REG_DDRC_PRE_CKE_X1024. Cycles to wait after reset before driving CKE High to start the DRAM initialization sequence. Units: 1,024 clock cycles. DDR2 specifications typically require this to be programmed for a delay of $\geq 200 \mu\text{s}$ .
[7:0]	REG_DDRC_DRAM_RSTN_X1024	0×0	Number of cycles to assert DRAM reset signal during initialization sequence. This is only present for implementations supporting DDR3 devices.

**Table 40 • DDRC\_CKE\_RSTN\_CYCLES\_2\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[11:2]	REG_DDRC_POST_CKE_X1024	0×0	Cycles to wait after driving CKE High to start the DRAM initialization sequence. Units: 1,024 clocks. DDR: Typically requires a 400 ns delay, requiring this value to be programmed to 2 at all clock speeds. SDR: Typically requires this to be programmed for a delay of 100 $\mu\text{s}$ to 200 $\mu\text{s}$ .
[1:0]	REG_DDRC_PRE_CKE_X1024	0×0	This field represents the upper 2 bits of the 10-bit REG_DDRC_PRE_CKE_X1024 value split across the 2 registers DDRC_CKE_RSTN_CYCLES_1_CR and DDRC_CKE_RSTN_CYCLES_2_CR. [9:8] bits of REG_DDRC_PRE_CKE_X1024. Cycles to wait from the start of reset assertion before driving CKE High to start the DRAM initialization sequence. Units: 1,024 clock cycles. DDR2 specifications typically require this to be programmed for a delay of $\geq 200 \mu\text{s}$ .

**Table 41 • DDRC\_INIT\_MR\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.



**Table 41 • DDRC\_INIT\_MR\_CR**

[15:0]	REG_DDRC_MR	0x095A	Value to be loaded into the DRAM Mode register. Bit 8 is for the DLL and the setting here is ignored. The controller sets appropriately. During DRAM initialization procedure, the controller will send the mode register setting to DRAM. The mode register sets the DRAM burst length, burst type, CAS latency (CL), and operating mode.
--------	-------------	--------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Table 42 • DDRC\_INIT\_EMR\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	REG_DDRC_EMR	0x0402	Value to be loaded into DRAM EMR registers. Bits [9:7] are for OCD and the setting in this bits is ignored. The controller sets those bits appropriately.

**Table 43 • DDRC\_INIT\_EMR2\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	REG_DDRC_EMR2	0x0	Value to be loaded into DRAM EMR2 registers.

**Table 44 • DDRC\_INIT\_EMR3\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	REG_DDRC_EMR3	0x0	Value to be loaded into DRAM EMR3 registers.

**Table 45 • DDRC\_DRAM\_BANK\_TIMING\_PARAM\_CR**

Bit Number	Name	Reset Value	Description
[31:12]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 45 • DDRC\_DRAM\_BANK\_TIMING\_PARAM\_CR**

[11:6]	REG_DDRC_T_RC	0x0	$t_{RC}$ : Minimum time between activates to same bank (specification: 65 ns for DDR2-400 and smaller for faster parts). Unit: clocks.
[5:0]	REG_DDRC_T_FAW	0x0	$t_{FAW}$ : Valid only in burst-of-8 mode. At most 4 banks must be activated in a rolling window of $t_{FAW}$ cycles. Unit: clocks

**Table 46 • DDRC\_DRAM\_RD\_WR\_LATENCY\_CR**

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[9:5]	REG_DDRC_WRITE_LATENCY	0x0	Number of clocks between the write command to write data enable PHY.
[4:0]	REG_DDRC_READ_LATENCY	0x0	Time from read command to read data on DRAM interface. Unit: clocks This signal is present for designs supporting LPDDR1 DRAM only. It is used to calculate when the DRAM clock may be stopped.

**Table 47 • DDRC\_DRAM\_RD\_WR\_PRE\_CR**

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[9:5]	REG_DDRC_WR2PRE	0x0	Minimum time between write and precharge to same bank (specifications: $WL + BL/2 + t_{WR}$ = approximately 8 cycles + 15 ns = 14 clocks @ 400 MHz and less for lower frequencies). Unit: Clocks where: WL = Write latency BL = Burst length. This must match the value programmed in the BL bit of the mode register to the DRAM. $t_{WR}$ = Write recovery time. This comes directly from the DRAM specs.
[4:0]	REG_DDRC_RD2PRE	0x0	$t_{RTP}$ – Minimum time from read to precharge of same bank (specification: $t_{RTP}$ for BL = 4 and $t_{RTP} + 2$ for BL = 8. $t_{RTP}$ = 7.5 ns). Unit: clocks.

**Table 48 • DDRC\_DRAM\_MR\_TIMING\_PARAM\_CR**

Bit Number	Name	Reset Value	Description
------------	------	-------------	-------------

**Table 48 • DDRC\_DRAM\_MR\_TIMING\_PARAM\_CR**

[31:13]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[12:3]	REG_DDRC_T_MOD	0×0	Present for DDR3 only (replaces REG_DDRC_T_MRD functionality when used with DDR3 devices). The mode register set command updates delay in number of clock cycles. This is required to be programmed even when a design that supports DDR3 is running in DDR2 mode (minimum is the larger of 12 clock cycles or 15 ns).
[2:0]	REG_DDRC_T_MRD	0×0	$t_{MRD}$ : Cycles between load mode commands. Not used in DDR3 mode.

**Table 49 • DDRC\_DRAM\_RAS\_TIMING\_CR**

Bit Number	Name	Reset Value	Description
[31:11]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[10:5]	REG_DDRC_T_RAS_MAX	0×0	$t_{RAS(max)}$ : Maximum time between activate and precharge to same bank. Maximum time that a page can be kept open (specification: 70 $\mu$ s). Minimum value of this register is 1. Zero is invalid. Unit: Multiples of 1,024 clocks.
[4:0]	REG_DDRC_T_RAS_MIN	0×0	$t_{RAS(min)}$ : Minimum time between activate and precharge to the same bank (specification: 45 ns). Unit: clocks.

**Table 50 • DDRC\_DRAM\_RD\_WR\_TRNARND\_TIME\_CR**

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[9:5]	REG_DDRC_RD2WR	0×0	$RL + BL/2 + 2 - WL$ Minimum time from READ command to WRITE command. Include time for bus turnaround and all per-bank, per-rank, and global constraints. Unit: clocks. where, WL = Write latency BL = Burst length. This must match the value programmed in the BL bit of the mode register to the DRAM. RL = Read latency = CAS latency.

**Table 50 • DDRC\_DRAM\_RD\_WR\_TRNARND\_TIME\_CR**

[4:0]	REG_DDRC_WR2RD	0×0	$WL + t_{WTR} + BL/2$ Minimum time from WRITE command to READ command. Includes time for bus turnaround and recovery times and all per-bank, per-rank, and global constraints. Unit: clocks. where, WL: Write latency. BL: Burst length. This should match the value programmed in the BL bit of the mode register to the DRAM. $t_{WTR}$ : Internal WRITE to READ command delay. This comes directly from the DRAM specifications.
-------	----------------	-----	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Table 51 • DDRC\_DRAM\_T\_PD\_CR**

Bit Number	Name	Reset Value	Description
[31:9]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[8:4]	REG_DDRC_T_XP	0×0	$t_{XP}$ : Minimum time after power-down exit to any operation. Units: clocks
[3:0]	REG_DDRC_T_CKE	0×0	Minimum number of cycles of CKE High/Low during power-down and self refresh. Unit: clocks

**Table 52 • DDRC\_DRAM\_BANK\_ACT\_TIMING\_CR**

Bit Number	Name	Reset Value	Description
[31:14]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[13:10]	REG_DDRC_T_RCD	0×0	$t_{RCD}$ : Minimum time from activate to READ or WRITE command to same bank (specification: 15 ns for DDR2-400 and lower for faster devices). Unit: clocks.
[9:7]	REG_DDRC_T_CCD	0×0	$t_{CCD}$ : Minimum time between two reads or two writes (from bank A to bank B) (specification: 2 cycles) is this value + 1. Unit: clocks.
[6:4]	REG_DDRC_T_RRD	0×0	$t_{RRD}$ : Minimum time between activates from bank A to bank B (specification: 10 ns or less). Unit: clocks.
[3:0]	REG_DDRC_T_RP	0×0	$t_{RP}$ : Minimum time from precharge to activate of same bank. Unit: clocks.

**Table 53 • DDRC\_ODT\_PARAM\_1\_CR**

Bit Number	Name	Reset Value	Description
[31:12]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 53 • DDRC\_ODT\_PARAM\_1\_CR (continued)**

Bit Number	Name	Reset Value	Description
[11:8]	REG_DDRC_RD_ODT_DELAY	0×0	The delay, in clock cycles, from issuing a READ command to setting ODT values associated with that command. Recommended value for DDR2 is CL – 4.
[7:4]	REG_DDRC_WR_ODT_DELAY	0×0	The delay, in clock cycles, from issuing a WRITE command to setting ODT values associated with that command. The recommended value for DDR2 is CL – 5. Where CL is CAS latency. DDR ODT has a 2-cycle on-time delay and a 2.5-cycle off-time delay. ODT setting should remain constant for the entire time that DQS is driven by the controller.
[3:2]	REG_DDRC_RANK0_WR_ODT	0×0	0: Indicates which remote ODTs should be turned on during a write to rank 0. Each rank has a remote ODT (in the DRAM) which can be turned on by setting the appropriate bit here. Set this bit to 1 to enable its ODT. 1: Uppermost bit is unused.
[1:0]	REG_DDRC_RANK0_RD_ODT	0×0	0: Indicates which remote ODTs should be turned on during a read to rank 0. Each rank has a remote ODT (in the DRAM) which can be turned on by setting the appropriate bit here. Set this bit to 1 to enable its ODT. 1: Uppermost bit is unused.

### 3.11.3.1 DDRC\_ODT\_PARAM\_2\_CR

**Table 54 • DDRC\_ODT\_PARAM\_2\_CR**

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[9:6]	REG_DDRC_RD_ODT_HOLD	0×0	Cycles to hold ODT for a READ command. 0: ODT signal is ON for 1 cycle. 1: ODT signal is ON for 2 cycles, and so on.
[5:2]	REG_DDRC_WR_ODT_HOLD	0×0	Cycles to hold ODT for a WRITE command. 0: ODT signal is ON for 1 cycle. 1: ODT signal is ON for 2 cycles, and so on.
[1:0]	REG_DDRC_WR_ODT_BLOCK	0×0	00: Block read/write scheduling for 1-cycle when write requires changing ODT settings. 01: Block read/write scheduling for 2 cycles when write requires changing ODT settings. 10: Block read/write scheduling for 3 cycles when write requires changing ODT settings. 11: Reserved

**Table 55 • DDRC\_ADDR\_MAP\_COL\_3\_CR**

Bit Number	Name	Reset Value	Description
[31:16] [7:6]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:12]	REG_DDRC_ADDRMAP_COL_B5	0×0	<b>Full bus width mode:</b> Selects column address bit 6. <b>Half bus width mode:</b> Selects column address bit 7. <b>Quarter bus width mode:</b> Selects column address bit 8. Valid range: 0 to 7 Internal base: 5 The selected address bit for each of the column address bits is determined by adding the internal base to the value of this field.
[11:8]	REG_DDRC_ADDRMAP_COL_B6	0×0	<b>Full bus width mode:</b> Selects column address bit 7. <b>Half bus width mode:</b> Selects column address bit 8. <b>Quarter bus width mode:</b> Selects column address bit 9. Valid range: 0 to 7 Internal base: 6 The selected address bit for each of the column address bits is determined by adding the internal base to the value of this field.
5	REG_DDRC_DIS_WC	0×0	When 1, disable write combine.

**Table 55 • DDRC\_ADDR\_MAP\_COL\_3\_CR (continued)**

Bit Number	Name	Reset Value	Description
4	REG_DDRC_DIS_ACT_BYPASS	0×0	Only present in designs supporting activate bypass. When 1, disable bypass path for high priority read activates
3	REG_DDRC_DIS_RD_BYPASS	0×0	Only present in designs supporting read bypass. When 1, disable bypass path for high priority read page hits.
2	REG_DDRC_DIS_PRE_BYPASS	0×0	Only present in designs supporting precharge bypass. When 1, disable bypass path for high priority precharges
1	REG_DDRC_DIS_COLLISION_PAGE_OP T	0×0	When this is set to '0', auto-precharge is disabled for the flushed command in a collision case. Collision cases are write followed by read to same address, read followed by write to same address, or write followed by write to same address with REG_DDRC_DIS_WC bit = 1 (where same address comparisons exclude the two address bits representing the critical word).
0	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 56 • DDRC\_MODE\_REG\_RD\_WR\_CR**

Bit Number	Name	Reset Value	Description
[31:4]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	REG_DDRC_MR_WR	0×0	When 1 is written and DDRC_REG_MR_WR_BUSY is Low, a mode register read or write operation is started. There is no need for the CPU to set this back to zero. This bit always reads as zero.
[2:1]	REG_DDRC_MR_ADDR	0×0	Address of the Mode register that is to be written to. 00: MR0 01: MR1 10: MR2 11: MR3
0	REG_DDRC_MR_TYPE	0×0	Indicates whether the Mode register operation is read or write. 1: Read 0: Write

**Table 57 • DDRC\_MODE\_REG\_DATA\_CR**

Bit Number	Name	Reset Value	Description
------------	------	-------------	-------------

**Table 57 • DDRC\_MODE\_REG\_DATA\_CR**

[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	REG_DDRC_MR_DATA	0×0	Mode register write data

**Table 58 • DDRC\_PWR\_SAVE\_1\_CR**

Bit Number	Name	Reset Value	Description
[31:13]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[12:6]	REG_DDRC_POST_SELFREF_GAP_X32	0×10	Minimum time to wait after coming out of self refresh before doing anything. This must be larger than all the constraints that exist (Specifications: maximum of $t_{XSNR}$ and $t_{XSRD}$ and $t_{XDLL}$ , which is 512 clocks). Unit: Multiples of 32 clocks.
[5:1]	REG_DDRC_POWERDOWN_TO_X32	0×06	After this many clocks of NOP or DESELECT, the controller puts the DRAM into power-down. This must be enabled in the Master Control register. Unit: Multiples of 32 clocks.
0	REG_DDRC_CLOCK_STOP_EN	0×0	1: Stops the clock to the PHY whenever a clock is not required by LPDDR1. 0: Clock will never be stopped. This is only present for implementations supporting mobile/LPDDR1 devices.

**Table 59 • DDRC\_PWR\_SAVE\_2\_CR**

Bit Number	Name	Reset Value	Description
[31:12]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	REG_DDRC_DIS_PAD_PD	0×0	1: Disable the pad power-down feature. 0: Enable the pad power-down feature. Used only in non-DFI designs.
[10:3]	REG_DDRC_DEEPPOWERDOWN_TO_X1024	0×0	Not supported.
[2:0]	REG_DDRC_PAD_PD	0×0	If pads have a power-saving mode, this is the greater of the time for the pads to enter power-down or the time for the pads to exit power-down. Used only in non-DFI designs. Unit: clocks.



**Table 60 • DDRC\_ZQ\_LONG\_TIME\_CR**

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[9:0]	REG_DDRC_T_ZQ_LONG_NOP	0×0	Number of cycles of NOP required after a ZQCL (ZQ calibration long) command is issued to DRAM. Units: Clock cycles. This is only present for implementations supporting DDR3 devices

**Table 61 • DDRC\_ZQ\_SHORT\_TIME\_CR**

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[9:0]	REG_DDRC_T_ZQ_SHORT_NOP	0×0	Number of cycles of NOP required after a ZQCS (ZQ calibration short) command is issued to DRAM. Units: Clock cycles. This is only present for implementations supporting DDR3 devices.

**Table 62 • DDRC\_ZQ\_SHORT\_INT\_REFRESH\_MARGIN\_1\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:4]	REG_DDRC_T_ZQ_SHORT_INTERVAL_X1024	0×0	20 bits are split into two registers. [11:0] bits of REG_DDRC_T_ZQ_SHORT_INTERVAL_X1024. Average interval to wait between automatically issuing ZQ calibration short (ZQCS) commands to DDR3 devices. Not considered if REG_DDRC_DIS_AUTO_ZQ = 1. Units: 1,024 clock cycles This is only present for implementations supporting DDR3 devices.

**Table 62 • DDRC\_ZQ\_SHORT\_INT\_REFRESH\_MARGIN\_1\_CR (continued)**

[3:0]	REG_DDRC_REFRESH_MARGIN	0x02	Threshold value in number of clock cycles before the critical refresh or page timer expires. A critical refresh is to be issued before this threshold is reached. Microsemi recommends using the default value. Unit: Multiples of 32 clocks.
-------	-------------------------	------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Table 63 • DDRC\_ZQ\_SHORT\_INT\_REFRESH\_MARGIN\_2\_CR**

Bit Number	Name	Reset Value	Description
[31:8]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[7:0]	REG_DDRC_T_ZQ_SHORT_INTERVAL_X1024	0x0	20 bits are split into two registers. [19:12] bits of REG_DDRC_T_ZQ_SHORT_INTERVAL_X1024. Average interval to wait between automatically issuing ZQ calibration short (ZQCS) commands to DDR3 devices. Not considered if REG_DDRC_DIS_AUTO_ZQ = 1. Units: 1,024 clock cycles This is only present for implementations supporting DDR3 devices.

**Table 64 • DDRC\_PERF\_PARAM\_1\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 64 • DDRC\_PERF\_PARAM\_1\_CR (continued)**

Bit Number	Name	Reset Value	Description
[15:13]	REG_DDRC_BURST_RDWR	0x0	<p>001: Burst length of 4            010: Burst length of 8            100: Burst length of 16            All other values are reserved.</p> <p>This controls the burst size used to access the DRAM. This must match the BL mode register setting in the DRAM. The DDRC and AXI controllers are optimized for a burst length of 8.</p> <p>The recommended setting is 8. A burst length of 16 is only supported for LPDDR1. Setting to 16 when using LPDDR1 in half/quarter bus mode may boost performance.</p> <p>For systems that tend to do many single cycle random transactions, a burst length of 4 may slightly improve system performance.</p>
12	Reserved	0x0	This bit must always be set to zero.
[11:5]	REG_DDRC_RDWR_IDLE_GAP	0x04	<p>When the preferred transaction store is empty for this many clock cycles, switch to the alternate transaction store if it is non-empty.</p> <p>The read transaction store (both high and low priority) is the default preferred transaction store and the write transaction store is the alternate store.</p> <p>When “Prefer write over read” is set, this is reversed.</p>
4	REG_DDRC_PAGECLOSE	0x0	<p>1: Bank is closed and kept closed if no transactions are available for it. This is different from auto-precharge:            (a) Explicit precharge commands are used, and not read/write with auto-precharge and            (b) Page is not closed after a read/write if there is another read/write pending to the same page.            0: Bank remains open until there is a need to close it (to open a different page, or for page timeout or refresh timeout).</p>
3	Reserved		This bit must always be set to zero.
[2:0]	REG_DDRC_LPR_NUM_ENTRIES	0x03	<p>Number of entries in the low priority transaction store is this value plus 1.</p> <p><math>READ\_CAM\_DEPTH - (REG\_DDRC\_LPR\_NUM\_ENTRIES + 1)</math> is the number of entries available for the high priority transaction store.</p> <p><math>READ\_CAM\_DEPTH</math> = Depth of the read transaction store, that is, 8. Setting this to maximum value allocates all entries to low priority transaction store.</p> <p>Setting this to 0 allocates 1 entry to low priority transaction store and the rest to high priority transaction store.</p> <p><b>Note:</b> In designs with ECC, number of lpr and wr credits issued to the core is 1 less than the non-ECC case. 1 entry each is reserved in wr and lpr cam for storing the RMW requests arising out of Single bit Error Correction RMW operation.</p>

**Table 65 • DDRC\_HPR\_QUEUE\_PARAM\_1\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	REG_DDRC_HPR_MAX_STARVE_X32	0×0	Lower 1 bit of REG_DDRC_HPR_MAX_STARVE_X32. Number of clocks that the HPR queue can be starved before it goes critical. Unit: 32 clocks.
[14:4]	REG_DDRC_HPR_MIN_NON_CRITICAL	0×0	Number of clocks that the HPR queue is guaranteed to be non-critical. Unit: 32 clocks.
[3:0]	REG_DDRC_HPR_XACT_RUN_LENGTH	0×0	Number of transactions that are serviced once the HPR queue goes critical is the smaller of this value and number of transactions available. Units: Transactions.

**Table 66 • DDRC\_HPR\_QUEUE\_PARAM\_2\_CR**

Bit Number	Name	Reset Value	Description
[31:11]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[10:0]	REG_DDRC_HPR_MAX_STARVE_X32	0×0	[11:1] bits of REG_DDRC_HPR_MAX_STARVE_X32. Number of clocks that the HPR queue can be starved before it goes critical. Unit: 32 clocks.

**Table 67 • DDRC\_LPR\_QUEUE\_PARAM\_1\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	REG_DDRC_LPR_MAX_STARVE_X32	0×0	12 bits are split into two registers. Lower 1 bit of REG_DDRC_LPR_MAX_STARVE_X32. Number of clocks that the LPR queue can be starved before it goes critical. Unit: 32 clocks.
[14:4]	REG_DDRC_LPR_MIN_NON_CRITICAL	0×0	Number of clocks that the LPR queue is guaranteed to be non-critical. Unit: 32 clocks.

**Table 67 • DDRC\_LPR\_QUEUE\_PARAM\_1\_CR (continued)**

Bit Number	Name	Reset Value	Description
[3:0]	REG_DDRC_LPR_XACT_RUN_LENGTH	0×0	Number of transactions that are serviced once the LPR queue goes critical is the smaller of this value and number of transactions available. Units: Transactions.

**Table 68 • DDRC\_LPR\_QUEUE\_PARAM\_2\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[10:0]	REG_DDRC_LPR_MAX_STARVE_X32	0×0	12 bits are split into two registers. [11:1] bits of REG_DDRC_HPR_MAX_STARVE_X32. Number of clocks that the LPR queue can be starved before it goes critical. Unit: 32 clocks.

**Table 69 • DDRC\_WR\_QUEUE\_PARAM\_CR**

Bit Number	Name	Reset Value	Description
[31:15]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[14:4]	REG_DDRC_W_MIN_NON_CRITICAL	0×0	Number of clocks that the write queue is guaranteed to be non-critical. Unit: 32 clocks.
[3:0]	REG_DDRC_W_XACT_RUN_LENGTH	0×0	Number of transactions that are serviced once the WR queue goes critical is the smaller of this value and number of transactions available. Units: Transactions.

**Table 70 • DDRC\_PERF\_PARAM\_2\_CR**

Bit Number	Name	Reset Value	Description
[31:12]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	REG_DDRC_BURSTCHOP	0×0	Not supported in this version of the DDRC controller always reads as zero.

**Table 70 • DDRC\_PERF\_PARAM\_2\_CR**

10	REG_DDRC_BURST_MODE	0×0	1: Interleaved burst mode 0: Sequential burst mode The burst mode programmed in the DRAM mode register and the order of the input data to the controller should both match the value programmed in the REG_DDRC_BURST_MODE register.
[9:2]	REG_DDRC_GO2CRITICAL_HYSTERESIS	0×0	Indicates the number of cycles that CO_GS_GO2CRITICAL_RD or CO_GS_GO2CRITICAL_WR must be asserted before the corresponding queue moves to the critical state in the DDRC.
1	REG_DDRC_PREFER_WRITE	0×0	If set, the bank selector prefers writes over reads.
0	REG_DDRC_FORCE_LOW_PRI_N	0×0	Active Low signal. When asserted ('0'), all incoming transactions are forced to low priority. Forcing the incoming transactions to low priority implicitly turns off bypass.

**Table 71 • DDRC\_PERF\_PARAM\_3\_CR**

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	REG_DDRC_EN_2T_TIMING_MODE	0×0	1: DDRC uses 2T timing. 0: DDRC uses 1T timing.

**Table 72 • DDRC\_DFI\_RDDATA\_EN\_CR**

Bit Number	Name	Reset Value	Description
[31:5]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[4:0]	REG_DDRC_DFI_T_RDDATA_EN	0×0	Time from the assertion of a READ command on the DFI interface to the assertion of the DDRC_DFI_RDDATA_EN signal. Program this to (RL – 1), where RL is the read latency of the DRAM. For LPDDR1 this should be set to RL. Units: Clocks

**Table 73 • DDRC\_DFI\_MIN\_CTRLUPD\_TIMING\_CR**

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[9:0]	REG_DDRC_DFI_T_CTRLUPD_MIN	0x03	Specifies the minimum number of clock cycles that the DDRC_DFI_CTRLUPD_REQ signal must be asserted. Lowest value to assign to this variable is 0x3. Units: Clocks

**Table 74 • DDRC\_DFI\_MAX\_CTRLUPD\_TIMING\_CR**

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[9:0]	REG_DDRC_DFI_T_CTRLUPD_MAX	0x40	Specifies the maximum number of clock cycles that the DDRC_DFI_CTRLUPD_REQ signal can assert. Lowest value to assign to this variable is 0x40. Units: Clocks

**Table 75 • DDRC\_DYN\_SOFT\_RESET\_ALIAS\_CR**

Bit Number	Name	Reset Value	Description
[31:3]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	AXIRESET	0x1	Set when main AXI reset signal is asserted. Reads and writes to the dynamic registers should not be carried out. This is a read only bit.

**Table 75 • DDRC\_DYN\_SOFT\_RESET\_ALIAS\_CR (continued)**

Bit Number	Name	Reset Value	Description
1	RESET_APB_REG	0×0	Full soft reset If this bit is set when the soft reset bit is written as '1', all APB registers reset to the power-up state.
0	REG_DDRC_SOFT_RSTB	0×0	This is a soft reset. 0: Puts the controller into reset. 1: Takes the controller out of reset. The controller should be taken out of reset only when all other registers have been programmed. Asserting this bit does NOT reset all the APB configuration registers. Once the soft reset bit is asserted, the APB register should be modified as required.

**Table 76 • DDRC\_AXI\_FABRIC\_PRI\_ID\_CR**

Bit Number	Name	Reset Value	Description
[31:6]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[5:4]	PRIORITY_ENABLE_BIT	0×0	This is to set the priority of the fabric master ID. 01/10/11: Indicates that the ID is higher priority. 00: None of the master IDs from the fabric has a higher priority.
[3:0]	PRIORITY_ID	0×0	If the Priority Enable bit is 1, this ID will have a higher priority over other IDs.

**Table 77 • DDRC\_SR**

Bit Number	Name	Reset Value	Description
[31:6]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.



**Table 77 • DDRC\_SR**

[5:3]	DDRC_CORE_REG_OPERATING_MODE	0×0	Operating mode. This is 3 bits wide in designs with mobile support and 2-bits in all other designs. Non-mobile designs: 000: Init 001: Normal 010: Power-down 011: Self Refresh Mobile designs: 000: Init 001: Normal 010: Power-down 011: Self refresh 1XX: Deep power-down
0	DDRC_REG_MR_WR_BUSY	0×0	1: Indicates that a mode register write operation is in progress. 0: Indicates that the core can initiate a mode register write operation. Core must initiate an MR write operation only if this signal is Low. This signal goes High in the clock after the controller accepts the write request. It goes Low when the MR write command is issued to the DRAM. Any MR write command that is received when DDRC_REG_MR_WR_BUSY is High, is not accepted.

**Table 78 • DDRC\_SINGLE\_ERR\_CNT\_STATUS\_SR**

Bit Number	Name	Reset Value	Description
[31:6]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	DDRC_SINGLE_ERR_CNT_STATUS_REG	0×0	Single error count status. If the count reaches 0xFFFF, it is held and only cleared after <b>DDRC_ECC_INT_CLR_REG</b> is written over by the system.

**Table 79 • DDRC\_DOUBLE\_ERR\_CNT\_STATUS\_SR**

Bit Number	Name	Reset Value	Description
[15:0]	DDRC_DOUBLE_ERR_CNT_STATUS_REG	0×0	Double error count status. If the count reaches 0xFFFF then it is held and only cleared after DDRC_ECC_INT_CLR_REG is written over by the system.

**Table 79 • DDRC\_DOUBLE\_ERR\_CNT\_STATUS\_SR**

[31:6]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
--------	----------	-----	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Table 80 • DDRC\_LUE\_SYNDROME\_1\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	DDRC_REG_ECC_SYNDROMES	0×0	<p>72 bits are split into five registers. [15:0] bits of DDRC_REG_ECC_SYNDROMES. First data which has SECDED error in it. 72 bits consists of the following:</p> <p>SECDED:            [71:64] – SECDED            [63:00] – Data</p> <p>In the same clock cycle, if one lane has a correctable error and the other lane has an uncorrectable error, the syndrome for the uncorrectable error is sent on this bus. If more than one data lane has an error in it, the lower data lane is selected. The priority applied when there are multiple errors in the same cycle is as follows:</p> <ul style="list-style-type: none"> <li>Uncorrectable error, lower lane</li> <li>Uncorrectable error, upper lane</li> <li>Correctable error, lower lane</li> <li>Correctable error, upper lane</li> </ul> <p>Only present in designs that support SECDED. This is cleared after DDRC_ECC_INT_CLR_REG is written over by the system.</p>

**Table 81 • DDRC\_LUE\_SYNDROME\_2\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 81 • DDRC\_LUE\_SYNDROME\_2\_SR**

[15:0]	DDRC_REG_ECC_SYNDROMES	0×0	<p>72 bits are split into five registers.            [31:16] bits of DDRC_REG_ECC_SYNDROMES.            First data which has SECEDED error in it. 72 bits consists of the following:            SECEDED:            [71:64] – SECEDED            [63:00] – Data</p> <p>In the same clock cycle, if one lane has a correctable error and the other lane has an uncorrectable error, the syndrome for the uncorrectable error is sent on this bus. If more than one data lane has an error in it, the lower data lane is selected. The priority applied when there are multiple errors in the same cycle is as follows:            Uncorrectable error, lower lane            Uncorrectable error, upper lane            Correctable error, lower lane            Correctable error, upper lane</p> <p>Only present in designs that support SECEDED.            This is cleared after            DDRC_ECC_INT_CLR_REG is written over by the system.</p>
--------	------------------------	-----	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Table 82 • DDRC\_LUE\_SYNDROME\_3\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	DDRC_REG_ECC_SYNDROMES	0×0	<p>72 bits are split into five registers.            [47:32] bits of DDRC_REG_ECC_SYNDROMES.            First data which has SECEDED error in it. 72 bits consists of the following:            SECEDED:            [71:64] – SECEDED            [63:00] – Data</p> <p>In the same clock cycle, if one lane has a correctable error and the other lane has an uncorrectable error, the syndrome for the uncorrectable error is sent on this bus. If more than one data lane has an error in it, the lower data lane is selected. The priority applied when there are multiple errors in the same cycle is as follows:            Uncorrectable error, lower lane            Uncorrectable error, upper lane            Correctable error, lower lane            Correctable error, upper lane</p> <p>Only present in designs that support SECEDED.            This is cleared after DDRC_ECC_INT_CLR_REG is written over by the system.</p>

**Table 83 • DDRC\_LUE\_SYNDROME\_4\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	DDRC_REG_ECC_SYNDROMES	0×0	<p>72 bits are split into five registers.            [63:48] bits of DDRC_REG_ECC_SYNDROMES.            First data which has SECDED error in it. 72 bits consists of the following:            SECDED:            [71:64] – SECDED            [63:00] – Data</p> <p>In the same clock cycle, if one lane has a correctable error and the other lane has an uncorrectable error, the syndrome for the uncorrectable error is sent on this bus. If more than one data lane has an error in it, the lower data lane is selected. The priority applied when there are multiple errors in the same cycle is as follows:            Uncorrectable error, lower lane            Uncorrectable error, upper lane            Correctable error, lower lane            Correctable error, upper lane</p> <p>Only present in designs that support SECDED.            This is cleared after DDRC_ECC_INT_CLR_REG is written over by the system.</p>

**Table 84 • DDRC\_LUE\_SYNDROME\_5\_SR**

Bit Number	Name	Reset Value	Description
[16:8]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 84 • DDRC\_LUE\_SYNDROME\_5\_SR**

[7:0]	DDRC_REG_ECC_SYNDROMES	0×0	<p>72 bits are split into five registers.            [71:64] bits of DDRC_REG_ECC_SYNDROMES.            First data which has SECEDED error in it. 72 bits consists of the following:            SECEDED:            [71:64] – SECEDED            [63:00] – Data</p> <p>In the same clock cycle, if one lane has a correctable error and the other lane has an uncorrectable error, the syndrome for the uncorrectable error is sent on this bus. If more than one data lane has an error in it, the lower data lane is selected. The priority applied when there are multiple errors in the same cycle is as follows:            Uncorrectable error, lower lane            Uncorrectable error, upper lane            Correctable error, lower lane            Correctable error, upper lane</p> <p>Only present in designs that support SECEDED.            This is cleared after            DDRC_ECC_INT_CLR_REG is written over by the system.</p>
-------	------------------------	-----	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Table 85 • DDRC\_LUE\_ADDRESS\_1\_SR**

Bit Number	Name	Reset Value	Description
[31:15]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[14:12]	DDRC_REG_ECC_BANK	0×0	Bank where the SECEDED error occurred. Only present in designs that support SECEDED.
[11:0]	DDRC_REG_ECC_COL	0×0	Column where the SECEDED error occurred. Col[0] is always set to 0, coming out of the controller. This bit is overwritten by the register module and indicates whether the error came from upper or lower lane. Only present in designs that support SECEDED.

**Table 86 • DDRC\_LUE\_ADDRESS\_2\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DDRC_REG_ECC_ROW	0×0	Row where the SECEDED error occurred. Only present in designs that support SECEDED.

**Table 87 • DDRC\_LCE\_SYNDROME\_1\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	DDRC_REG_ECC_SYNDROMES	0×0	<p>72 bits are split into five registers.            [15:0] bits of DDRC_REG_ECC_SYNDROMES.            First data which has SECDED error in it. 72 bits consists of the following:            SECDED:            [71:64] – SECDED            [63:00] – Data</p> <p>In the same clock cycle, if one lane has a correctable error and the other lane has an uncorrectable error, the syndrome for the uncorrectable error is sent on this bus. If more than one data lane has an error in it, the lower data lane is selected. The priority applied when there are multiple errors in the same cycle is as follows:            Uncorrectable error, lower lane            Uncorrectable error, upper lane            Correctable error, lower lane            Correctable error, upper lane</p> <p>Only present in designs that support SECDED.            This is cleared after DDRC_ECC_INT_CLR_REG is written over by the system.</p>

**Table 88 • DDRC\_LCE\_SYNDROME\_2\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 88 • DDRC\_LCE\_SYNDROME\_2\_SR**

[15:0]	DDRC_REG_ECC_SYNDROMES	0×0	<p>72 bits are split into five registers.          [31:16] bits of DDRC_REG_ECC_SYNDROMES.          First data which has SECEDED error in it. 72 bits consists of the following:          SECEDED:          [71:64] – SECEDED          [63:00] – Data</p> <p>In the same clock cycle, if one lane has a correctable error and the other lane has an uncorrectable error, the syndrome for the uncorrectable error is sent on this bus. If more than one data lane has an error in it, then the lower data lane is selected. The priority applied when there are multiple errors in the same cycle is as follows:          Uncorrectable error, lower lane          Uncorrectable error, upper lane          Correctable error, lower lane          Correctable error, upper lane</p> <p>Only present in designs that support SECEDED.          This is cleared after DDRC_ECC_INT_CLR_REG is written over by the system.</p>
--------	------------------------	-----	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Table 89 • DDRC\_LCE\_SYNDROME\_3\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	DDRC_REG_ECC_SYNDROMES	0×0	<p>72 bits are split into five registers.          [47:32] bits of DDRC_REG_ECC_SYNDROMES.          First data which has SECEDED error in it. 72 bits consists of the following:          SECEDED:          [71:64] – SECEDED          [63:00] – Data</p> <p>In the same clock cycle, if one lane has a correctable error and the other lane has an uncorrectable error, the syndrome for the uncorrectable error is sent on this bus. If more than one data lane has an error in it, the lower data lane is selected. The priority applied when there are multiple errors in the same cycle is as follows:          Uncorrectable error, lower lane          Uncorrectable error, upper lane          Correctable error, lower lane          Correctable error, upper lane</p> <p>Only present in designs that support SECEDED.          This is cleared after DDRC_ECC_INT_CLR_REG is written over by the system.</p>

**Table 90 • DDRC\_LCE\_SYNDROME\_4\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	DDRC_REG_ECC_SYNDROMES	0×0	<p>72 bits are split into five registers.            [63:48] bits of DDRC_REG_ECC_SYNDROMES.            First data which has SECEDED error in it. 72 bits consists of the following:            SECEDED:            [71:64] – SECEDED            [63:00] – Data</p> <p>In the same clock cycle, if one lane has a correctable error and the other lane has an uncorrectable error, the syndrome for the uncorrectable error is sent on this bus. If more than one data lane has an error in it, the lower data lane is selected. The priority applied when there are multiple errors in the same cycle is as follows:            Uncorrectable error, lower lane            Uncorrectable error, upper lane            Correctable error, lower lane            Correctable error, upper lane</p> <p>Only present in designs that support SECEDED.            This is cleared after DDRC_ECC_INT_CLR_REG is written over by the system.</p>

**Table 91 • DDRC\_LCE\_SYNDROME\_5\_SR**

Bit Number	Name	Reset Value	Description
[16:8]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.



**Table 91 • DDRC\_LCE\_SYNDROME\_5\_SR**

[7:0]	DDRC_REG_ECC_SYNDROMES	0×0	<p>72 bits are split into five registers.            [71:64] bits of DDRC_REG_ECC_SYNDROMES.            First data which has SECEDED error in it. 72 bits consists of the following            SECEDED:            [71:64] – SECEDED            [63:00] – Data</p> <p>In the same clock cycle, if one lane has a correctable error and the other lane has an uncorrectable error, the syndrome for the uncorrectable error is sent on this bus. If more than one data lane has an error in it, the lower data lane is selected. The priority applied when there are multiple errors in the same cycle is as follows:            Uncorrectable error, lower lane            Uncorrectable error, upper lane            Correctable error, lower lane            Correctable error, upper lane</p> <p>Only present in designs that support SECEDED.            This is cleared after DDRC_ECC_INT_CLR_REG is written over by the system.</p>
-------	------------------------	-----	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Table 92 • DDRC\_LCE\_ADDRESS\_1\_SR**

Bit Number	Name	Reset Value	Description
[31:15]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[14:12]	DDRC_REG_ECC_BANK	0×0	Bank where the SECEDED error occurred.
[11:0]	DDRC_REG_ECC_COL	0×0	Column where the SECEDED error occurred. Col[0] is always set to 0 coming out of the controller. This bit is overwritten by the register module and indicates whether the error came from upper or lower lane.

**Table 93 • DDRC\_LCE\_ADDRESS\_2\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	DDRC_REG_ECC_ROW	0×0	Row where the SECEDED error occurred.

**Table 94 • DDRC\_LCB\_NUMBER\_SR**

Bit Number	Name	Reset Value	Description
------------	------	-------------	-------------

**Table 94 • DDRC\_LCB\_NUMBER\_SR**

[31:7]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[6:0]	DDRC_LCB_BIT_NUM	0×0	Indicates the location of the bit that caused a single-bit error in SECEDED case (encoded value). If more than one data lane has an error in it, the lower data lane is selected. This register is 7 bits wide in order to handle 72 bits of the data present in a single lane. This does not indicate CORRECTED_BIT_NUM in the case of device correction SECEDED. The encoding is only present in designs that support SECEDED.

**Table 95 • DDRC\_LCB\_MASK\_1\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	DDRC_LCB_MASK	0×0	<p>64 bits are split into four registers.            [15:0] bits of DDRC_LCB_MASK.            Indicates the mask of the corrected data.            1: On any bit indicates that the bit has been corrected by the DRAM SECEDED logic.            0: On any bit indicates that the bit has NOT been corrected by the DRAM SECEDED logic.            Valid when any bit of DDRC_REG_ECC_CORRECTED_ERR is High.            This mask doesn't indicate any correction that has been made in the SECEDED check bits.            If there are errors in multiple lanes, this signal will have the mask for the lowest lane.</p>

**Table 96 • DDRC\_LCB\_MASK\_2\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 96 • DDRC\_LCB\_MASK\_2\_SR**

[15:0]	DDRC_LCB_MASK	0×0	<p>64 bits are split into four registers. [31:16] bits of DDRC_LCB_MASK.</p> <p>Indicates the mask of the corrected data.</p> <p>1: On any bit indicates that the bit has been corrected by the DRAM SECEDED logic.</p> <p>0: On any bit indicates that the bit has NOT been corrected by the DRAM SECEDED logic.</p> <p>Valid when any bit of DDRC_REG_ECC_CORRECTED_ERR is High.</p> <p>This mask does not indicate any correction that has been made in the SECEDED check bits.</p> <p>If there are errors in multiple lanes, this signal will have the mask for the lowest lane.</p>
--------	---------------	-----	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Table 97 • DDRC\_LCB\_MASK\_3\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	DDRC_LCB_MASK	0×0	<p>64 bits are split into four registers. [47:32] bits of DDRC_LCB_MASK.</p> <p>Indicates the mask of the corrected data.</p> <p>1: On any bit indicates that the bit has been corrected by the DRAM SECEDED logic.</p> <p>0: On any bit indicates that the bit has NOT been corrected by the DRAM SECEDED logic.</p> <p>Valid when any bit of DDRC_REG_ECC_CORRECTED_ERR is High.</p> <p>This mask does not indicate any correction that has been made in the SECEDED check bits.</p> <p>If there are errors in multiple lanes, this signal will have the mask for the lowest lane.</p>

**Table 98 • DDRC\_LCB\_MASK\_4\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 98 • DDRC\_LCB\_MASK\_4\_SR**

[15:0]	DDRC_LCB_MASK	0x0	<p>64 bits are split into four registers. [63:48] bits of DDRC_LCB_MASK. Indicates the mask of the corrected data.</p> <p>1: On any bit indicates that the bit has been corrected by the DRAM SECDED logic. 0: On any bit indicates that the bit has NOT been corrected by the DRAM SECDED logic.</p> <p>Valid when any bit of DDRC_REG_ECC_CORRECTED_ERR is High.</p> <p>This mask does not indicate any correction that has been made in the SECDED check bits. If there are errors in multiple lanes, this signal will have the mask for the lowest lane.</p>
--------	---------------	-----	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Table 99 • DDRC\_ECC\_INT\_SR**

Bit Number	Name	Reset Value	Description
[31:3]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[2:0]	DDRC_ECC_STATUS_SR	0x0	<p>Bit 0: 1 Indicates the SECDED interrupt is due to a single error. Bit 1: 1 Indicates the SECDED interrupt is due to a double error. Bit 2: Always 1</p>

**Table 100 • DDRC\_ECC\_INT\_CLR\_REG**

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DDRC_ECC_INT_CLR_REG	0x0	<p>This register should be written by the processor when it has read the SECDED error status information. This helps to clear all the SECDED status information, such as error counters and other SECDED registers.</p> <p>The read value of this register is always 0.</p>

### 3.11.4 PHY Configuration Register Summary

**Table 101 • PHY Configuration Register Summary**

Register Name	Offset	Type	Reset Source	Description
Reserved	0x200 to 0x22C	-	-	-
PHY_DATA_SLICE_IN_USE_CR	0x230	RW	PRESET_N	PHY data slice in use register
Reserved	0x234 to 0x3C8	-	-	-

### 3.11.5 PHY Configuration Register Bit Definitions

**Table 102 • PHY\_DATA\_SLICE\_IN\_USE\_CR**

Bit Number	Name	Reset Value	Description
[31:5]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[4:0]	REG_PHY_DATA_SLICE_IN_USE	0x0	Data bus width selection for read FIFO RE generation. One bit for each data slice. 1: Data slice is valid. 0: Read data responses are ignored. <b>Note:</b> The PHY data slice 0 must always be enabled.

### 3.11.6 DDR\_FIC Configuration Registers Summary

**Table 103 • DDR\_FIC Configuration Register Summary**

Register Name	Address Offset	R/W	Reset Source	Description
DDR_FIC_NB_ADDR_CR	0x400	RW	PRESET_N	Indicates the base address of the non-bufferable address region.
DDR_FIC_NBRWB_SIZE_CR	0x404	RW	PRESET_N	Indicates the size of the non-bufferable address region.
DDR_FIC_BUF_TIMER_CR	0x408	RW	PRESET_N	10-bit timer interface used to configure the timeout register.
DDR_FIC_HPD_SW_RW_EN_CR	0x40C	RW	PRESET_N	Enable write buffer and read buffer register for AHBL master1 and master2.
DDR_FIC_HPD_SW_RW_INVALID_CR	0x410	RW	PRESET_N	Invalidates write buffer and read buffer for AHBL master1 and master2.
DDR_FIC_SW_WR_ERCLR_CR	0x414	RW	PRESET_N	Clear bit for error status by AHBL master1 and master2 write buffer.
DDR_FIC_ERR_INT_ENABLE	0x418	RW	PRESET_N	Used for Interrupt generation.

**Table 103 • DDR\_FIC Configuration Register Summary (continued)**

Register Name	Address Offset	R/W	Reset Source	Description
DDR_FIC_NUM_AHB_MASTERS_CR	0x41C	RW	PRESET_N	Defines whether one or two AHBL 32-bit masters are implemented in fabric.
DDR_FIC_HPB_ERR_ADDR_1_SR	0x420	RO	PRESET_N	Tag of write buffer for which error response is received is placed in this register.
DDR_FIC_HPB_ERR_ADDR_2_SR	0x424	RO	PRESET_N	Tag of write buffer for which error response is received is placed in this register.
DDR_FIC_SW_ERR_ADDR_1_SR	0x428	RO	PRESET_N	Tag of write buffer for which error response is received is placed in this register.
DDR_FIC_SW_ERR_ADDR_2_SR	0x42C	RO	PRESET_N	Tag of write buffer for which error response is received is placed in this register.
DDR_FIC_HPD_SW_WRB_EMPTY_SR	0x430	RO	PRESET_N	Indicates valid data in read and write buffer for AHBL master1 and master2.
DDR_FIC_SW_HPB_LOCKOUT_SR	0x434	RO	PRESET_N	Write and read buffer status register for AHBL master1 and master2.
DDR_FIC_SW_HPD_WERR_SR	0x438	RO	PRESET_N	Error response register for bufferable write request
DDR_FIC_LOCK_TIMEOUTVAL_1_CR	0x440	RW	PRESET_N	Indicates maximum number of cycles a master can hold the bus for locked transfer.
DDR_FIC_LOCK_TIMEOUTVAL_2_CR	0x444	RW	PRESET_N	Indicates maximum number of cycles a master can hold the bus for locked transfer.
DDR_FIC_LOCK_TIMEOUT_EN_CR	0x448	RW	PRESET_N	Lock timeout feature enable register
DDR_FIC_RDWR_ERR_SR	0x460	RO	PRESET_N	Indicates read address of math error register.

### 3.11.7 DDR\_FIC Configuration Register Bit Definitions

**Table 104 • DDR\_FIC\_NB\_ADDR\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	DDR_FIC_NB_ADD	0×0	This indicates the base address of the non-bufferable address region.

**Table 105 • DDR\_FIC\_NBRWB\_SIZE\_CR**

Bit Number	Name	Reset Value	Description
[31:9]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	DDR_FIC_WCB_SZ	0×0	Configures write buffer and read buffer size as per DDR burst size. This port is common for all buffers. Buffers can be configured to 16 byte or 32 byte size. 0: Buffer size is configured to 16 bytes 1: Buffer size is configured to 32 bytes
[7:4]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[3:0]	DDR_FIC_NUBF_SZ	0×0	This signal indicates the size of the non-bufferable address region. The region sizes are as follows: 0000: None (default) 0001: 64 KB bufferable region 0010: 128 KB bufferable region 0011: 256 KB bufferable region 0100: 512 KB bufferable region 0101: 1 MB bufferable region 0110: 2 MB bufferable region 0111: 4 MB bufferable region 1000: 8 MB bufferable region 1001: 16 MB bufferable region 1010: 32 MB bufferable region 1011: 64 MB bufferable region 1100: 128 MB bufferable region 1101: 256 MB bufferable region 1110: 512 MB bufferable region 1111: 1 GB bufferable region

**Table 106 • DDR\_FIC\_BUF\_TIMER\_CR**

Bit Number	Name	Reset Value	Description
------------	------	-------------	-------------

**Table 106 • DDR\_FIC\_BUF\_TIMER\_CR**

[31:10]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[9:0]	DDR_FIC_TIMER	0×0	10-bit timer interface used to configure timeout register. Once timer reaches the timeout value, a flush request is generated by the flush controller in the DDR_FIC. This port is common for all buffers.

**Table 107 • DDR\_FIC\_HPD\_SW\_RW\_EN\_CR**

Bit Number	Name	Reset Value	Description
[31:7]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	DDR_FIC_M1_REN	0×0	1: Enable read buffer for AHBL master1. 0: Disable read buffer for AHBL master1.
5	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	DDR_FIC_M1_WEN	0×0	1: Enable write buffer for AHBL master1. 0: Disable write buffer for AHBL master1.
3	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DDR_FIC_M2_REN	0×0	1: Enable read buffer for AHBL master2. 0: Disable read buffer for AHBL master2.
1	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DDR_FIC_M2_WEN	0×0	1: Enable write buffer for AHBL master2. 0: Disable write buffer for AHBL master2.

**Table 108 • DDR\_FIC\_HPD\_SW\_RW\_INVALID\_CR**

Bit Number	Name	Reset Value	Description
[31:7]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	DDR_FIC_flushM1	0×0	1: Flush read buffer for AHBL master1. 0: Default
5	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	DDR_FIC_invalid_M1	0×0	1: Invalidate write buffer for AHBL master1. 0: Default



**Table 108 • DDR\_FIC\_HPD\_SW\_RW\_INVAL\_CR (continued)**

Bit Number	Name	Reset Value	Description
3	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DDR_FIC_flshM2	0×0	1: Flush write buffer for AHBL master2. 0: Default
1	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DDR_FIC_invalid_M2	0×0	1: Invalidate read buffer for AHBL master2. 0: Default

**Table 109 • DDR\_FIC\_SW\_WR\_ERCLR\_CR**

Bit Number	Name	Reset Value	Description
[31:9]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	DDR_FIC_LTO_CLR	0×0	Clear signal to lock timeout interrupt.
[7:5]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	DDR_FIC_M2_WR_ERCLR	0×0	Clear bit for error status of AHBL master2 write buffer. Once it goes High, error status is cleared.
[3:1]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DDR_FIC_M1_WR_ERCLR	0×0	Clear bit for error status posted by AHBL master1 write buffer. Once it goes High, error status is cleared.

**Table 110 • DDR\_FIC\_ERR\_INT\_ENABLE**

Bit Number	Name	Reset Value	Description
[31:2]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 110 • DDR\_FIC\_ERR\_INT\_ENABLE**

1	SYR_SW_WR_ERR	0×0	Status bit. Goes High when error response is received for bufferable write request. Goes Low when processor serves interrupt and makes clear bit for AHBL master1.
0	SYR_HPD_WR_ERR	0×0	Status bit. Goes High when error response is received for bufferable write request. Goes Low when processor serves the interrupt.

**Table 111 • DDR\_FIC\_NUM\_AHB\_MASTERS\_CR**

Bit Number	Name	Reset Value	Description
[31:5]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	CFG_NUM_AHB_MASTERS	0×0	Defines whether one or two AHBL 32-bit masters are implemented in the fabric. 0: One 32-bit AHB master implemented in fabric 1: Two 32-bit AHB masters implemented in fabric
[3:0]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 112 • DDR\_FIC\_HPB\_ERR\_ADDR\_1\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	DDR_FIC_M1_ERR_ADD	0×0	32 bits are split into two registers. [15:0] bits of DDR_FIC_M1_ERR_ADD Tag of write buffer for which error response is received is placed in this register. The following values are updated in this register as per buffer size: Buffer size 16 bytes: 28 bit TAG value is loaded to [31:4] and 0000 to [3:0] 32 bytes: upper 27 bits of TAG is loaded to [31:5] and 00000 to [4:0]

**Table 113 • DDR\_FIC\_HPBB\_ERR\_ADDR\_2\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	DDR_FIC_M1_ERR_ADD	0x0	32 bits are split into two registers. [31:16] bits of DDR_FIC_M1_ERR_ADD Tag of write buffer for which error response is received is placed in this register. The following values are updated in this register as per buffer size: Buffer size 16 bytes: 28 bit TAG value is loaded to [31:4] and 0000 to [3:0] 32 bytes: upper 27 bits of TAG is loaded to [31:5] and 00000 to [4:0]

**Table 114 • DDR\_FIC\_SW\_ERR\_ADDR\_1\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	DDR_FIC_M2_ERR_ADD	0x0	32 bits are split into two registers. Lower 16 bits. Tag of write buffer for which error response is received is placed in this register. The following values are updated in this register as per buffer size: Buffer size: DDR_FIC_M2_ERR_ADD[31:0] 16 bits: TAG, 0000 32 bits: TAG[27:1], 00000

**Table 115 • DDR\_FIC\_SW\_ERR\_ADDR\_2\_SR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	DDR_FIC_M2_ERR_ADD	0x0	32 bits are split into two registers. [31:16] bits of DDR_FIC_M2_ERR_ADD Tag of write buffer for which error response is received is placed in this register. The following values are updated in this register as per buffer size: Buffer size 16 bytes: 28 bit TAG value is loaded to [31:4] and 0000 to [3:0] 32 bytes: upper 27 bits of TAG is loaded to [31:5] and 00000 to [4:0]

**Table 116 • DDR\_FIC\_HPDPD\_SW\_WRB\_EMPTY\_SR**

Bit Number	Name	Reset Value	Description
[31:7]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	DDR_FIC_M1_RBEMPTY	0×0	1: Read buffer of AHBL master1 does not have valid data.
5	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	DDR_FIC_M1_WBEMPTY	0×0	1: Write buffer of AHBL master1 does not have valid data. 0: Default
3	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DDR_FIC_M2_RBEMPTY	0×0	1: Read buffer of AHBL master2 does not have valid data. 0: Default.
1	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DDR_FIC_M2_WBEMPTY	0×0	1: Write buffer of AHBL master2 does not have valid data. 0: Default

**Table 117 • DDR\_FIC\_SW\_HPB\_LOCKOUT\_SR**

Bit Number	Name	Reset Value	Description
[31:9] [7] [5] [3] [1]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	DDR_FIC_LCKTOUT	0×0	Indicates lock counter in arbiter reached its maximum value. Lock counter (20-bit) starts counting when a locked request gets access to a bus and will be cleared when the lock signal becomes logic 0.
6	DDR_FIC_M2_WDSBL_DN	0×0	High indicates AHBL master2 write buffer is disabled.
4	DDR_FIC_M2_RDSBL_DN	0×0	High indicates AHBL master2 read buffer is disabled.
2	DDR_FIC_M1_WDSBL_DN	0×0	High indicates AHBL master1 read buffer is disabled.
0	DDR_FIC_M1_RDSBL_DN	0×0	High indicates AHBL master1 write buffer is disabled.

**Table 118 • DDR\_FIC\_SW\_HPD\_WERR\_SR**

Bit Number	Name	Reset Value	Description
[31:9]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	DDR_FIC_M1_WR_ERR	0×0	Status bit. Goes High when error response is received for bufferable write request. Goes Low when the processor serves an interrupt and makes a clear bit for AHBL master1.
[7:1]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DDR_FIC_M2_WR_ERR	0×0	Status bit. Goes High when error response is received for bufferable write request. Goes Low when processor serves the interrupt.

**Table 119 • DDR\_FIC\_LOCK\_TIMEOUTVAL\_1\_CR**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[15:0]	CFGR_LOCK_TIMEOUT_REG	0×0	20 bits are split into two registers. [15:0] bits of CFGR_LOCK_TIMEOUT_REG Lock timeout 20-bit register. Indicates maximum number of cycles a master can hold the bus for locked transfer. If master holds the bus for locked transfer more than the required cycles, an interrupt is generated.

**Table 120 • DDR\_FIC\_LOCK\_TIMEOUTVAL\_2\_CR**

Bit Number	Name	Reset Value	Description
[31:4]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 120 • DDR\_FIC\_LOCK\_TIMEOUTVAL\_2\_CR**

[3:0]	CFGR_LOCK_TIMEOUT_REG	0x0	20 bits are split into two registers. [19:16] bits of CFGR_LOCK_TIMEOUT_REG Lock timeout 20-bit register. Indicates maximum number of cycles a master can hold the bus for locked transfer. If master holds the bus for locked transfer more than the required cycles, an interrupt is generated.
-------	-----------------------	-----	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Table 121 • DDR\_FIC\_LOCK\_TIMEOUT\_EN\_CR**

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	CFGR_LOCK_TIMEOUT_EN	0x0	1: Lock timeout feature is enabled and interrupt is generated. 0: Lock timeout feature is disabled and interrupt is not generated.

**Table 122 • DDR\_FIC\_RDWR\_ERR\_SR**

Bit Number	Name	Reset Value	Description
[31:6]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[5:0]	DDR_FIC_CFG_RDWR_ERR_SR	0x0	Read address of math error register.

## 3.12 Appendix A: How to Use the MDDR in SmartFusion2

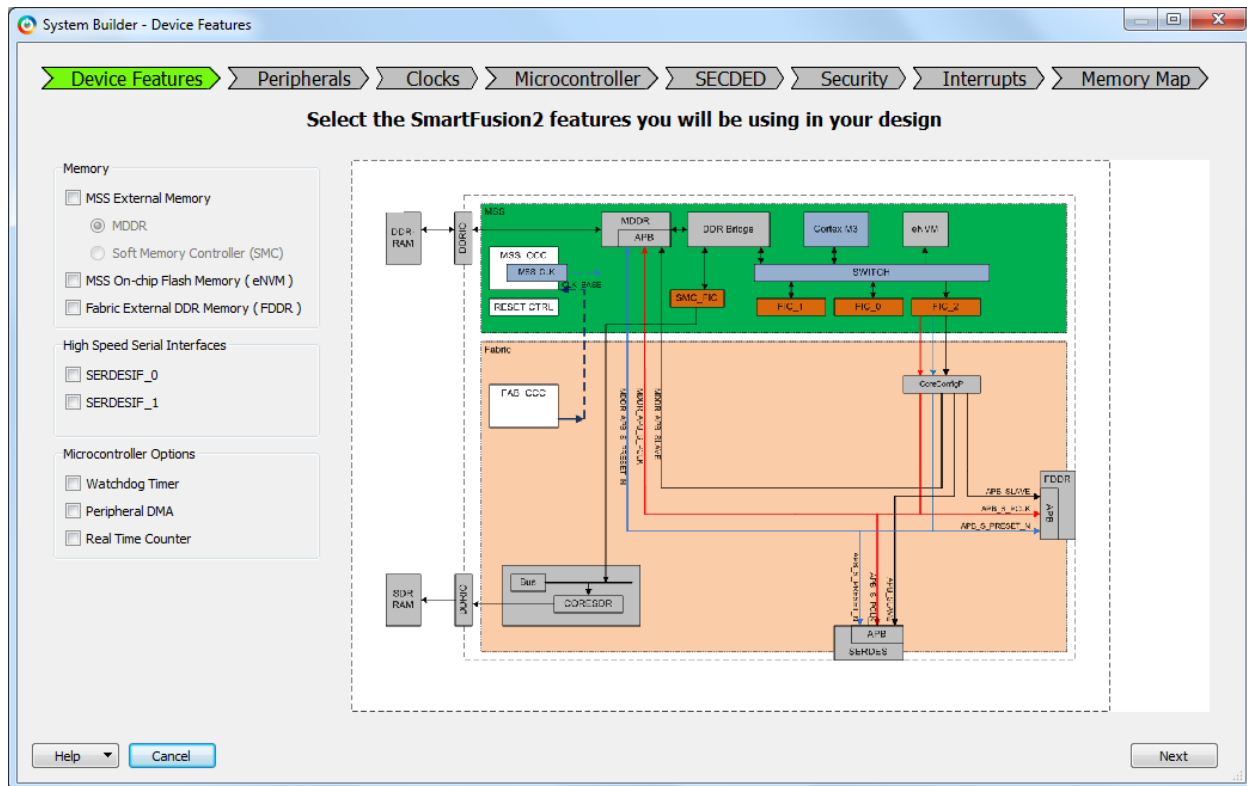
This section describes how to use the MDDR subsystem in the design. It contains the following sections:

- [Design Flow Using System Builder](#)
- [Design Flow Using SmartDesign](#)
- [Use Model 1: Accessing MDDR from FPGA Fabric Through the AXI Interface](#)
- [Use Model 2: Accessing MDDR from FPGA Fabric Through the AHB Interface](#)
- [Use Model 3: Accessing MDDR from Cortex-M3 Processor](#)
- [Use Model 4: Accessing MDDR from the HPDMA](#)

### 3.12.1 Design Flow Using System Builder

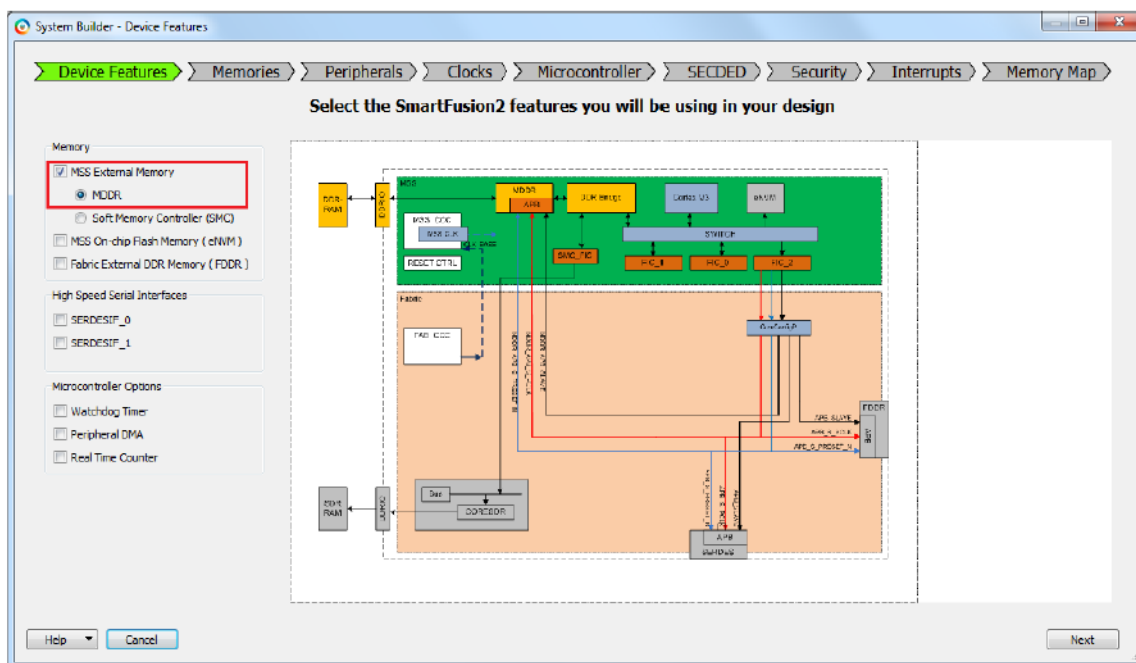
This section describes how to use MDDR in the SmartFusion2 devices using the System Builder graphical design wizard in the Libero Software.

The following image shows the initial **System Builder** window where you can select the features that you require. For details on how to launch and use the **System Builder** wizard, refer the [SmartFusion2 System Builder User Guide](#). For more information on DDR initialization, refer to the [SmartFusion2 DDR Controller and Serial High Speed Controller Initialization Methodology](#).

**Figure 46 • System Builder - Device Features Window**

The following steps describe how to configure the MDDR.

1. Check the **MSS External Memory** check box under the **Device Features** tab, select MDDR and leave the other check boxes unchecked. The following image shows the **System Builder - Device Features** tab.

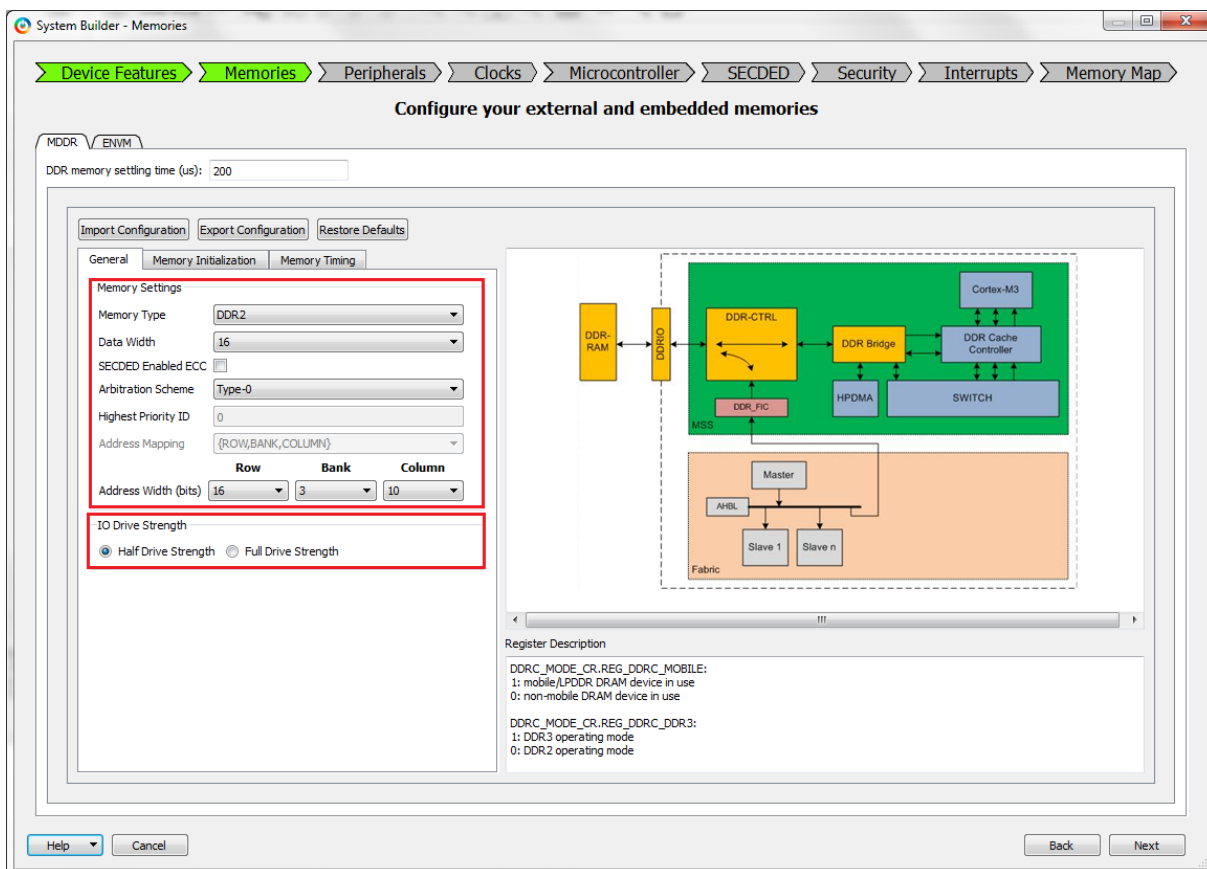
**Figure 47 • MSS External DDR Memory Selection**

2. Navigate to the **Memories** tab. Depending on the application requirement, select the memory settings under the **General** tab, as shown in the following image.
  - Memory Type can be selected as DDR2, DDR3, or LPDDR.
  - The Data width can be selected as 32-bit, 16-bit, or 8-bit. Refer to [Table 12 on page 25](#) for supported data widths for various SmartFusion2 device packages.
  - The SECDED (ECC) can be enabled or disabled.
  - Arbitration Scheme can be selected between Type-0 to Type-3. Refer Table 1-10 for details of arbitration Scheme.
  - The Highest priority ID of fabric master can be entered from 0 to 15, if the Arbitration Scheme selected other than Type-0.
  - Address Mapping - The register settings to perform mapping to system address bits for various Row, Bank, and Column combinations are automatically computed by the configurator using address mapping option. [Table 20](#), page 34 shows the supported range for Row, Bank and Column. For more information refer to the ["Address Mapping" section](#).
  - Select the **I/O Drive Strength** as **Half Drive Strength** or **Full Drive Strength** as shown in [Figure 48](#), page 113. The DDR I/O standard is configured as listed in [Figure 21](#), page 34 based on this setting.

**Table 123 • DDR I/O Standard Configured Based on I/O Drive Strength Setting**

I/O Drive Strength	Memory Type	
	DDR2	DDR3
Half Drive Strength	SSTL18I	SSTL15I
Full Drive Strength	SSTL18II	SSTL15II

**Figure 48 • I/O Drive Strength Setting**



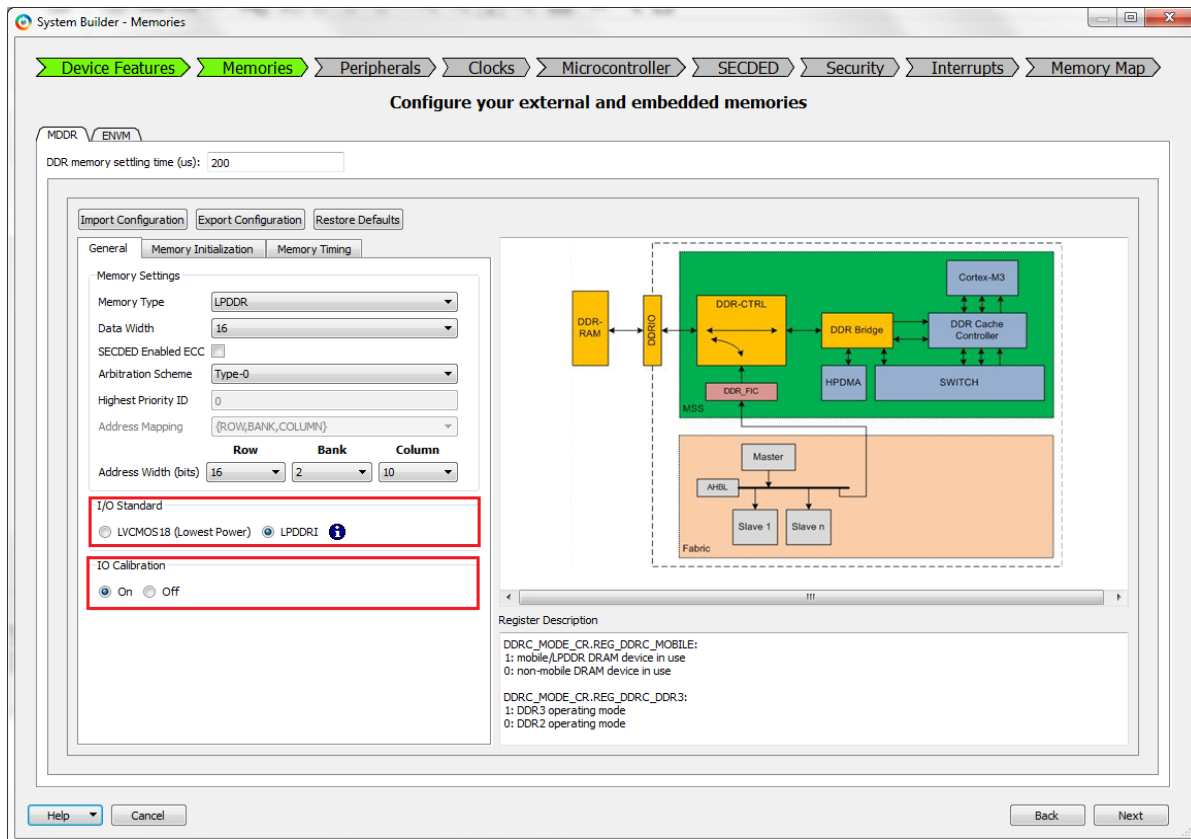


3. For only LPDDR memory, the **I/O standard** and **I/O calibration** settings are available as shown in the following image.
  - Select **I/O standard** as **LVC MOS18** or **LPDDR1**. For Microsemi M2S\_EVAL\_KIT board select LPDDR1(SSTL18) as the board is designed to use LPDDR1 I/O standard.

**Note:** If LVC MOS18 is selected, all IOs are configured to LVC MOS1.8 except CLK/CLK\_N.CLK and CLK\_N are configured to LPDDR1 standard because they are differential signals.

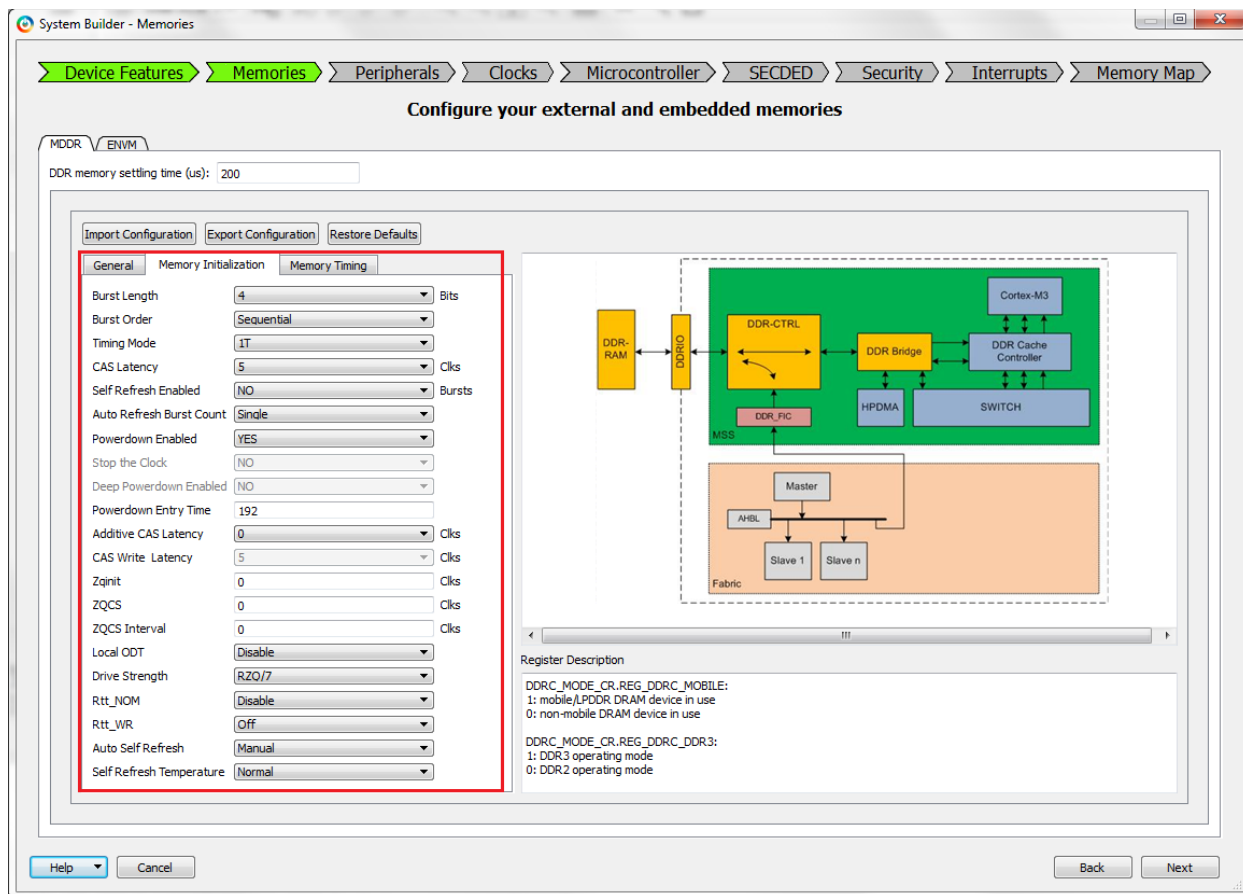
- Select **I/O calibration** as ON or OFF. If I/O calibration is selected as **ON**, then the Smartfusion2 MDDR\_IMP\_CALIB pin must be pulled down with a resistor. For information on resistor values, refer to the "Impedance Calibration section" in [DS0115: SmartFusion2 Pin Descriptions Datasheet](#).

**Figure 49 • Selecting I/O Standard as LVC MOS18 or LPDDR1**

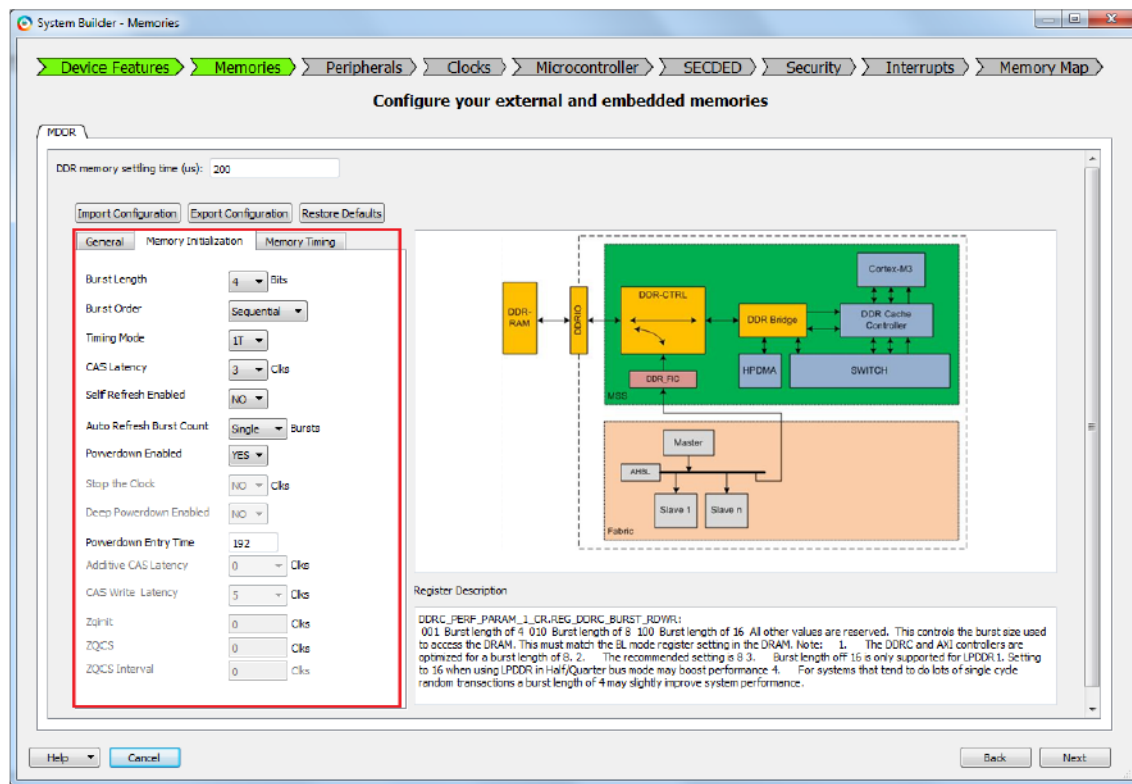


4. Depending on the application requirement, select the Memory Initialization settings under the Memory Initialization tab as shown in [Figure 50 on page 116](#).
  - Select the below performance related settings
    - Burst Length can be selected as 4, 8 or 16. [Table 12 on page 25](#) for supported burst lengths.
    - Burst order can be selected as sequential or interleaved. Refer Table 1-13 for supported burst orders.
    - Timing mode can be selected as 1T or 2T. For more details refer to "1T or 2T Timing" section on [page 30](#).
    - CAS latency is the delay, in clock cycles, between the internal READ command and the availability of the first bit of output data. Select the CAS latency according to the DDR memory (Mode register) datasheet.
  - Select the below power saving mode settings. Refer to "Power Saving Modes" on [page 24](#) for more details.
    - Self-Refresh Enabled
    - Auto Refresh Burst Count
    - Power down Enabled
    - Stop the clock: supported only for LPDDR
    - Deep Power down Enabled: supported only for LPDDR

- Power down entry time
- Select the additional performance settings for DDR3 memory.
  - Additive CAS Latency is defined by EMR[5:3] register of DDR2 memory and by MR1[4:3] register of DDR3 memory. It enables the DDR2 or DDR3 SDRAM to allow a READ or WRITE command from DDR Controller after the ACTIVATE command for the same bank prior to tRCD (MIN). This configuration is part of DDR2 Extended Mode register and DDR3 mode register1.
  - CAS Write Latency (CWL) is defined by DDR3 MR2[5:3] and is the delay, in clock cycles, from the releasing of the internal write to the latching of the first data in. The overall WRITE latency (WL) is equal to CWL + AL by default CWL is set to 5 clock cycles.
- Select the below ZQ Calibration settings for DDR3 memory. For more details refer ["ZQ Calibration" on page 17](#).
  - Zqinit
  - ZQCS
  - ZQCS Interval
- Select Other Settings
  - Local ODT setting is not supported for LPDDR memory. For DDR2/DDR3 memory type, user can choose any option for "Local ODT". User can enable or disable "LOCAL ODT" during read transaction.
  - Drive strength setting is defined by EMR[7:5] register bits of LPDDR memory with drop down options of 'Full', 'Half', 'Quarter' and 'One-eighth' drive strength, it is defined by EMR[1] register bit of DDR2 memory with drop down options of 'Full' and 'Weak' drive strength and it is defined by MR1 register bits M5 and M1 of DDR3 memory with drop down options of 'RZQ/6' and 'RZQ/7'.
  - Partial array self-refresh coverage setting is defined by EMR[2:0] register bits of LPDDR memory with drop down options of 'Full', 'Quarter', 'One-eighth' and 'One-sixteenth'. This feature helps in improving power savings during self-refresh by selecting the amount of memory to be refreshed during self-refresh.
  - RTT (Nominal) setting is defined by EMR[6] and EMR[2] register bits of DDR2 memory which determines what ODT resistance is enabled with drop down options of 'RTT disabled', '50  $\Omega$ ', '75  $\Omega$ ' and '150  $\Omega$ ' and it is defined by MR1[9], MR1[6] and MR1[2] register bits of DDR3 memory. In DDR3 memory RTT nominal termination is allowed during standby conditions and WRITE operations and NOT during READ operations with drop down options of 'RZQ/2', 'RZQ/4' and 'RZQ/6'.
  - RTT\_WR (Dynamic ODT) setting is defined by MR2[10:9] register bits of DDR3 memory. This is applicable only during WRITE operations. If dynamic ODT (Rtt\_WR) is enabled, DRAM switches from normal ODT (RTT\_nom) to dynamic ODT (Rtt\_WR) when beginning WRITE burst and subsequently switches back to normal ODT at the end of WRITE burst. The drop down options provided to the user are 'off', 'RZQ/4' and 'RZQ/2'.
  - Auto self-refresh setting is defined by MR2[6] register bit of DDR3 memory with drop down option of 'Manual' and 'Auto'. Self-refresh temperature setting is defined by MR2[7] register bit of DDR2 memory with drop down options of 'Normal' and 'Extended'.

**Figure 50 • DDR Memory initialization Settings**

5. Select the Memory Timing settings under the **Memory Timing** tab according to the DDR memory vendor datasheet as shown in the following illustration. For more details, refer to ["Configuring Dynamic DRAM Constraints"](#) section on page 26.

**Figure 51 • DDR Memory Timing Settings**

The configurator also provides the option to import and export the register configurations.

Configuration files for accessing DDR3 memory on SmartFusion2 Development kit can be downloaded from [www.microsemi.com/soc/documents/MDDR3\\_16Bit\\_SB.zip](http://www.microsemi.com/soc/documents/MDDR3_16Bit_SB.zip).

Configuration files for accessing LPDDR memory on SmartFusion2 Starter kit can be downloaded from [www.microsemi.com/soc/documents/LPDDR\\_Emcrafft\\_Config.zip](http://www.microsemi.com/soc/documents/LPDDR_Emcrafft_Config.zip).

**Note:** The firmware generated by Libero SoC stores these configurations and the MDDR subsystem registers are initialized by the Cortex-M3 processor during the system\_init phase of the firmware projects (SoftConsole/IAR/Keil projects generated by Libero SoC).

The following is an example of MDDR register configurations for operating the LPDDR memory (MT46H64M16LF) with clock 166 MHz.

- Device Memory Settling Time (us): 200

The DDR memories require settling time for the memory to initialize before accessing it. the LPDDR memory model MT46H64M16LF needs 200 us settling time.

#### General

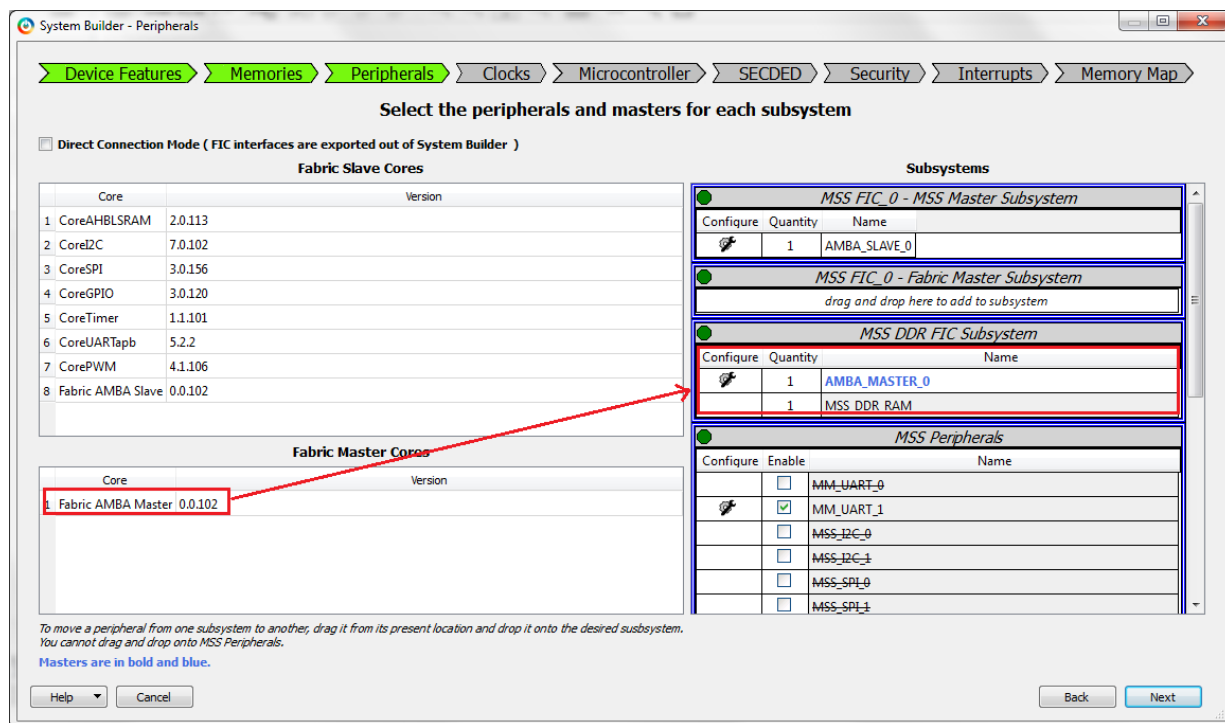
- Memory Type - Select LPDDR
- Data Width: 16
- Memory Initialization:
- Burst length - 8
- Burst Order: Interleaved
- Timing Mode: 1T
- CAS Latency: 3
- Self Refresh Enabled: No
- Auto Refresh Burst Count: 8
- PowerDown Enabled: Yes
- Stop the clock: No
- Deep PowerDown enabled: No

- No Activity clocks for Entry: 320

#### Memory Timing

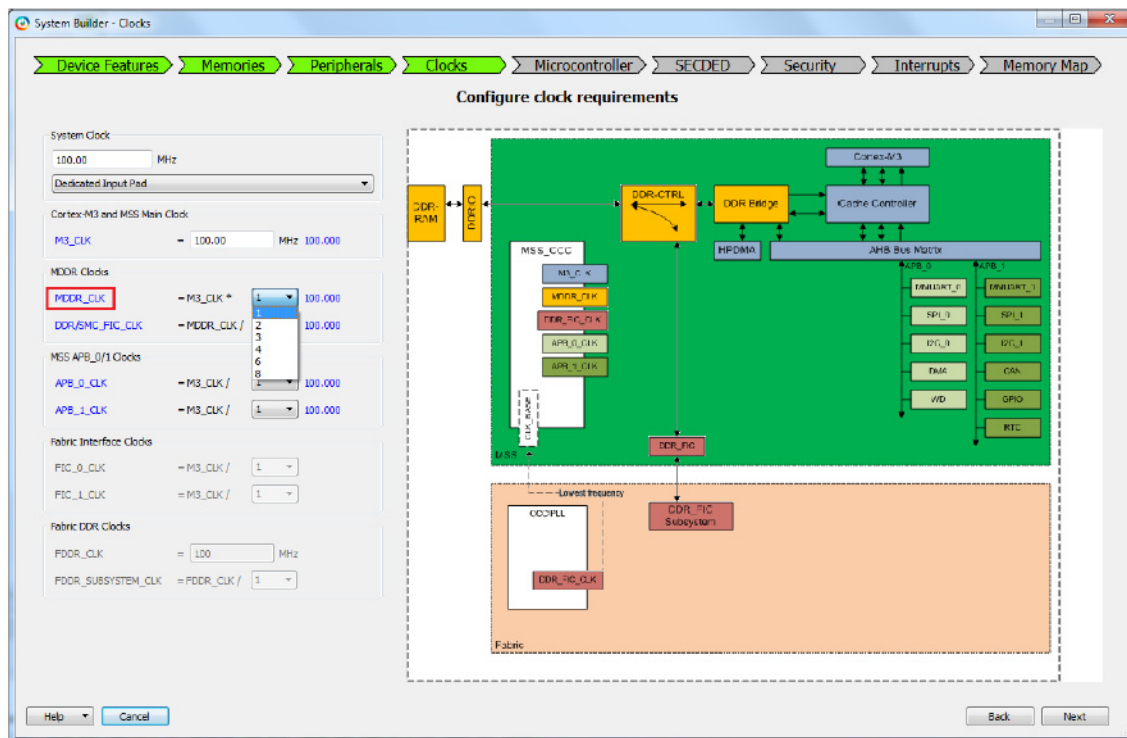
- Time To Hold Reset Before INIT - 67584 clks
  - MRD: 4 clks
  - RAS (Min): 8 clks
  - RAS (Max): 8192 clks
  - RCD: 6 clks
  - RP: 7 clks
  - REF: 3104 clks
  - RC: 3 clks
  - XP: 3 clks
  - CKE: 3 clks
  - RFC: 79 clks
  - FAW: 0 clks
6. Navigate to the **Peripherals** tab. To access the MDDR from the FPGA fabric, drag and drop the **Fabric AMBA Master** to the **MSS DDR FIC Subsystem** and click **configure** to select the type of interface as AXI or single AHB-Lite. The user logic in the FPGA fabric can access the DDR memory through the MDDR using these interfaces. The following image shows the **Peripherals** tab.

**Figure 52 • MSS DDR FIC Subsystem Configuration**



7. Navigate to the **Clocks** tab. The **Clocks** tab allows to configure the system clock and subsystem clocks. The MDDR subsystem operates on MDDR\_CLK, which comes from MSS\_CCC. The MDDR\_CLK must be selected as multiples of 1, 2, 3, 4, 6 or 8-of M3\_CLK. The maximum frequency of MDDR\_CLK is 333.33 MHz. The following image shows the MDDR\_CLK configuration.

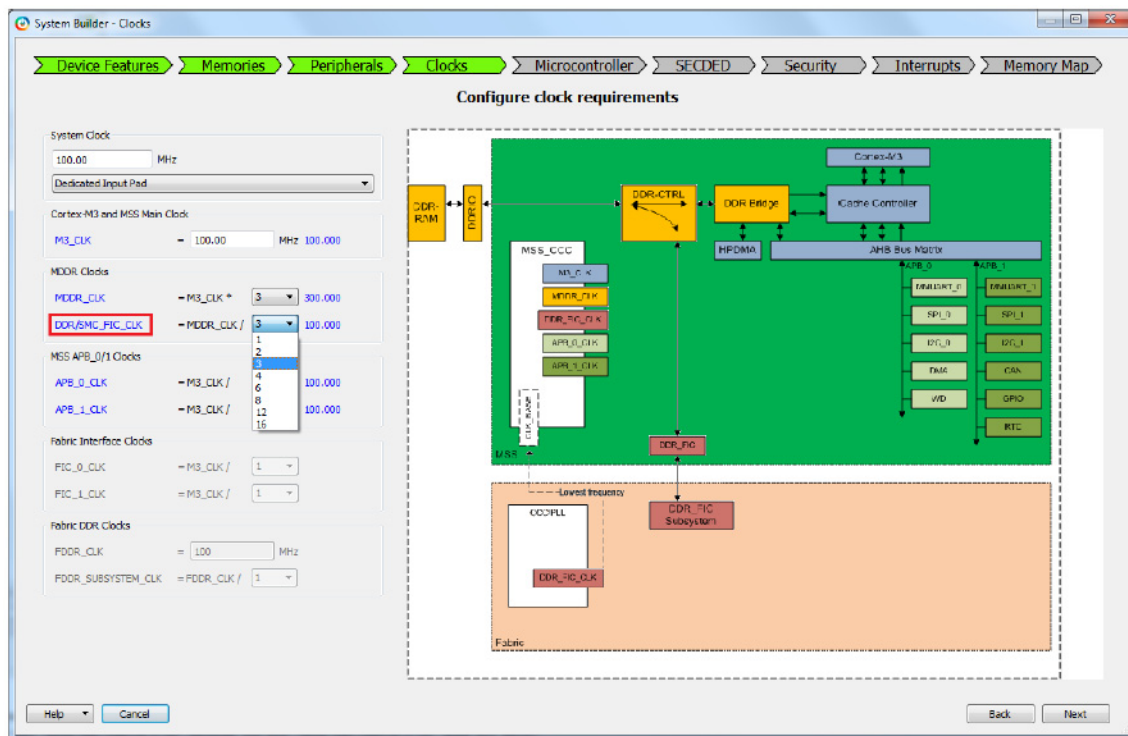
Figure 53 • MDDR Clock Configuration



DDR\_FIC\_CLK drives the DDR\_FIC slave interface and defines the frequency at which the FPGA fabric subsystem connected to this interface is intended to run. DDR\_FIC\_CLK can be configured as a ratio of MDDR\_CLK (1, 2, 3, 4, 6, 8, 12, or 16) using the Clocks configurator. The maximum frequency of DDR\_FIC\_CLK is 200 MHz. The following image shows the DDR\_FIC\_CLK configuration.

If the MDDR\_CLK ratio to M3\_CLK is a multiple of 3, DDR\_SMC\_FIC\_CLK's ratio to MDDR\_CLK must also be a multiple of 3, and vice versa. The configurator issues an error if this requirement is not met. This limitation is imposed by the internal implementation of the MSS CCC.

**Figure 54 • DDR\_FIC Clock Configuration**



### 3.12.2 Design Flow Using SmartDesign

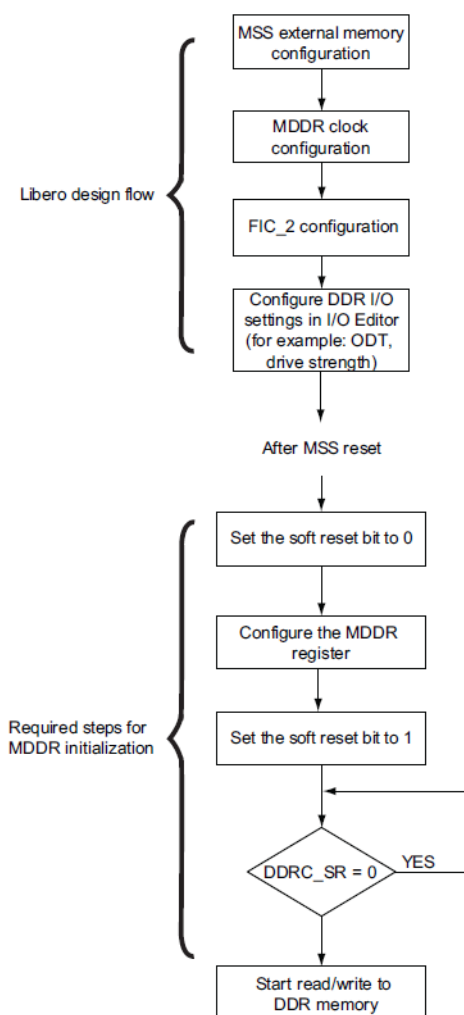
The following flow chart illustrates the design flow for using the MDDR subsystem to access external DDR memory.

The design flow consists of two parts:

- **Libero SoC flow** – This includes configuring the type of DDR memory, choosing fabric master interface type, clocking, and DDR I/O settings.
- **MDDR register initialization** – The MDDR subsystem registers can be initialized using the Cortex-M3 processor or FPGA fabric master. After MSS resets, the MDDR registers must be configured according to application and DDR memory specification. The ["MDDR Subsystem Features Configuration" section on page 25](#) provides the details of required register configuration for MDDR features. While configuring the registers, the soft reset to the DDR controller must be asserted.

After releasing the soft reset, the DDR controller performs DDR memory initialization and sets the status bits in **DDRC SR**.

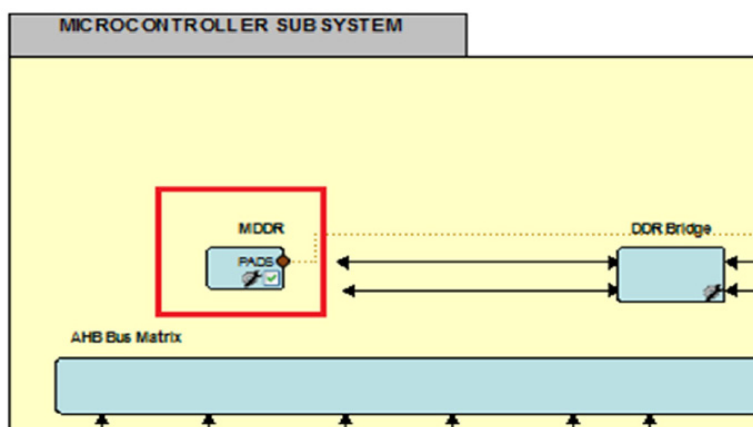


**Figure 55 • Design Flow**

The following sections explain the configuration steps in the flow chart.

### 3.12.2.1 MSS External Memory Configuration

The MDDR subsystem is configured through the MDDR configurator, which is part of the MSS configurator in the Libero SoC design software. The following image shows the MDDR configurator.

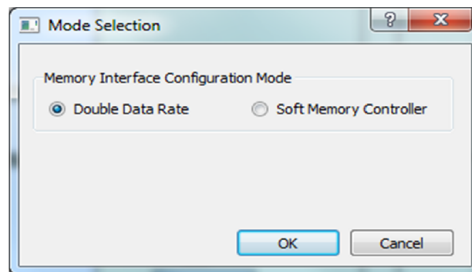
**Figure 56 • MDDR Configurator**



Double click the **MDDR Configurator**, which gives the following choices for the external memory interface type as shown in the following image.

- **Double Data Rate:** This option must be selected to access the external DDR memories (DDR2, DDR3 and LPDDR).
- **Soft Memory Controller:** This option must be selected to access the external memories through SMC\_FIC and soft memory controller in FPGA. For more information on using SMC\_FIC mode, refer to the "Soft Memory Controller Fabric Interface Controller" chapter on page 312.

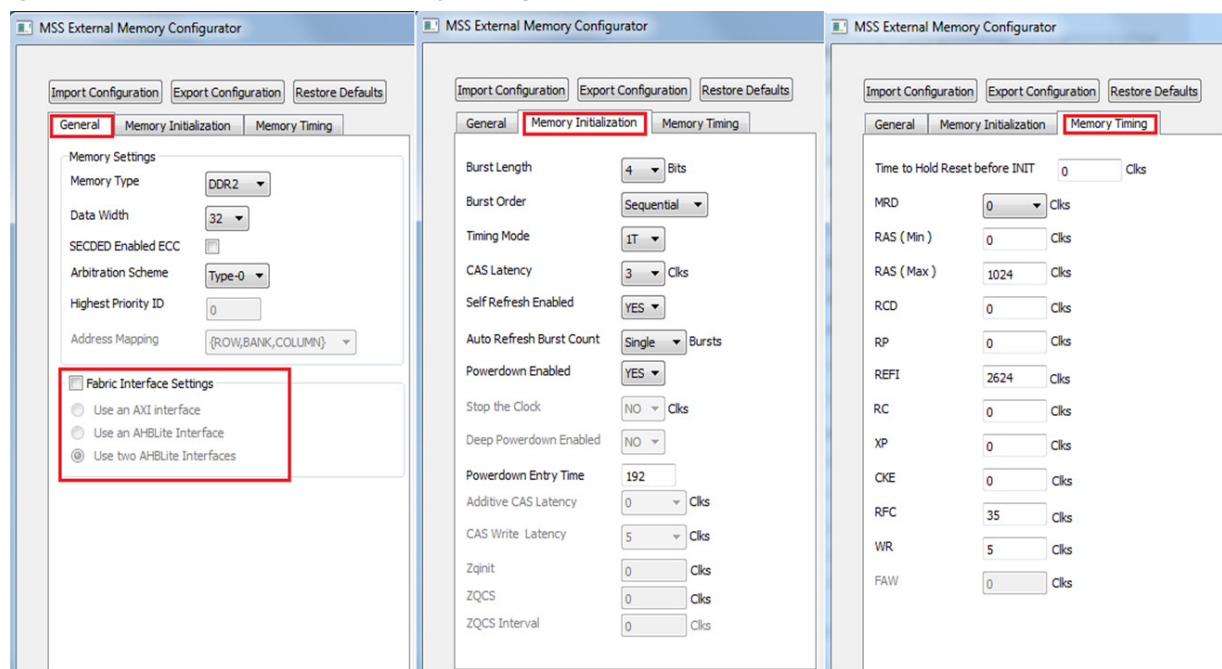
**Figure 57 • Memory Interface Configuration**



Select **Double Data Rate** and click **Ok**. The MSS External Memory Configurator will be displayed as shown in the following image. Select the memory settings as described in the steps 2, 3 and 4 in the "Design Flow Using System Builder" section on page 111.

To access the MDDR from the FPGA fabric, select From Fabric Interface Settings and the type of interface as AXI, single AHBLite, or two AHBLite Interfaces. On completion of the configuration, the selected interface is exposed in SmartDesign. The user logic in the FPGA fabric can access the DDR memory through MDDR using these interfaces.

**Figure 58 • MSS External DDR Memory Configurator**

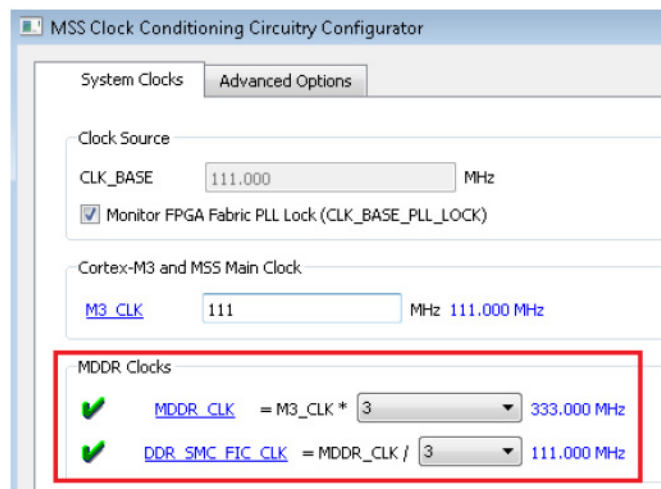


### 3.12.2.2 MDDR Clock Configuration

The MDDR subsystem operates on MDDR\_CLK, which comes from MSS\_CCC. The MDDR\_CLK must be selected as a multiple—1, 2, 3, 4, 6 or 8—of M3\_CLK. This clock value can be configured through the MSS\_CCC configurator in Libero SoC, as shown in the following figure.

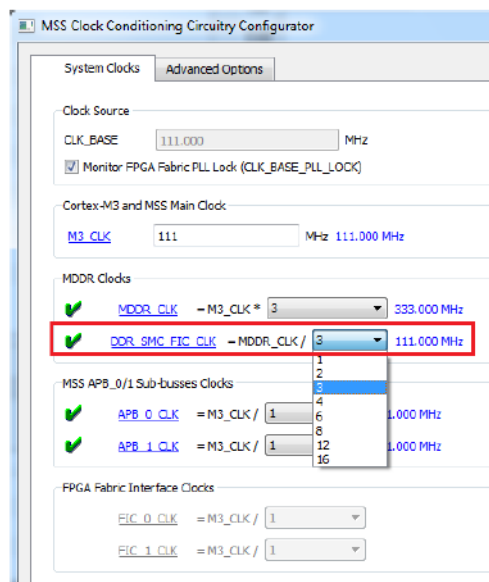
The maximum frequency of MDDR\_CLK is 333.33 MHz.

**Figure 59 • MDDR Clock Configuration**



DDR\_SMC\_FIC\_CLK drives the DDR\_FIC slave interface and defines the frequency at which the FPGA fabric subsystem connected to this interface is intended to run. DDR\_SMC\_FIC\_CLK can be configured as a ratio of MDDR\_CLK (1, 2, 3, 4, 6, 8, 12, or 16) through the MSS\_CCC configurator in Libero SoC, as shown in the following image. The maximum frequency of DDR\_SMC\_CLK is 200 MHz.

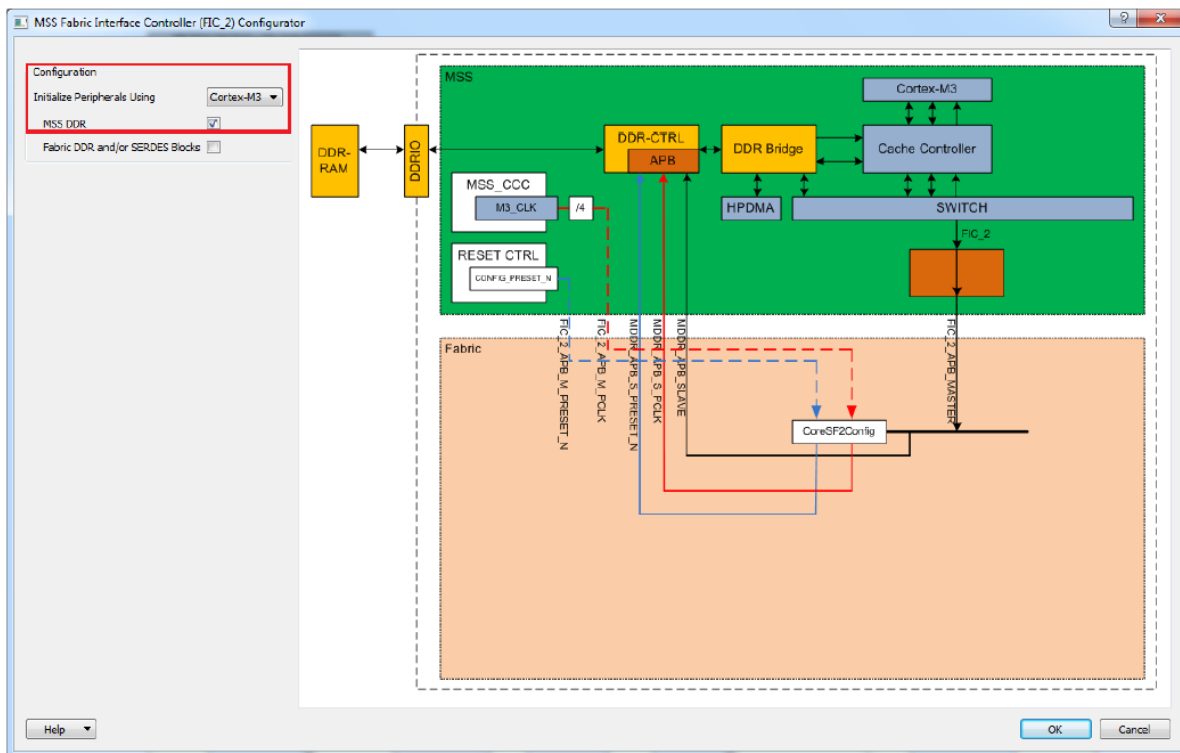
**Figure 60 • MDDR Clock Configuration**



If the MDDR\_CLK ratio to M3\_CLK is a multiple of 3, DDR\_SMC\_FIC\_CLK's ratio to MDDR\_CLK must also be a multiple of 3, and vice versa. The configurator issues an error if this requirement is not met. This limitation is imposed by the internal implementation of the MSS CCC.

### 3.12.2.2.1 FIC\_2 Configuration

This is required to initialize the MDDR registers (optional when initializing from MSS). Configure FIC\_2 (peripheral initialization) block, as shown in the following image to expose the MDDR APB interface (MDDR\_APB\_SLAVE interface) in Libero SmartDesign. Use the MDDR\_APB\_SLAVE interface to connect with the APB master logic in the FPGA fabric.

**Figure 61 • FIC\_2 Configuration**

When enabling this option, the MDDR\_APB\_S\_PCLK and FIC\_2\_APB\_M\_PCLK signals are exposed in SmartDesign. MDDR\_APB\_S\_PCLK must be connected to FIC\_2\_APB\_M\_PCLK. The FIC\_2\_APB\_M\_PCLK clock is generated from the MSS\_CCC and is identical to M3\_CLK/4.

### 3.12.2.3 I/O Configuration

I/O settings such as like ODT and drive strength can be configured as shown in the following image using the I/O Editor in the Libero design software.

**Figure 62 • I/O Configuration**

Port Name	Direction	I/O Standard	Pin Number	Locked	Bank Name	I/O state in Flash*Freeze mode	Resistor Pull	I/O available in Flash*Freeze mode	Schmitt Trigger	Odt_Static	Odt Imp (Ohm)	Low P
MDDR_OKE	Output	SSTL15I	E29	✓	Bank0	TRISTATE	None	--	--	--	--	
MDDR_CLK	Output	SSTL15I	A25	✓	Bank0	TRISTATE	None	--	--	--	--	
MDDR_CLK_IN	Output	SSTL15I	B25	✓	Bank0	TRISTATE	None	--	--	--	--	
MDDR_CS_N	Output	SSTL15I	F29	✓	Bank0	TRISTATE	None	--	--	--	--	
MDDR_DM_RDQS[0]	Inout	SSTL15I	D13	✓	Bank0	TRISTATE	None	--	--	On	40	
MDDR_DM_RDQS[1]	Inout	SSTL15I	D16	✓	Bank0	TRISTATE	None	--	--	On	40	
MDDR_DQ[0]	Inout	SSTL15I	A12	✓	Bank0	TRISTATE	None	--	--	On	40	
MDDR_DQ[1]	Inout	SSTL15I	B12	✓	Bank0	TRISTATE	None	--	--	On	40	
MDDR_DQ[2]	Inout	SSTL15I	D12	✓	Bank0	TRISTATE	None	--	--	On	40	
MDDR_DQ[3]	Inout	SSTL15I	E12	✓	Bank0	TRISTATE	None	--	--	On	40	
MDDR_DQ[4]	Inout	SSTL15I	A14	✓	Bank0	TRISTATE	None	--	--	On	40	
MDDR_DQ[5]	Inout	SSTL15I	D14	✓	Bank0	TRISTATE	None	--	--	On	40	
MDDR_DQ[6]	Inout	SSTL15I	E14	✓	Bank0	TRISTATE	None	--	--	On	40	

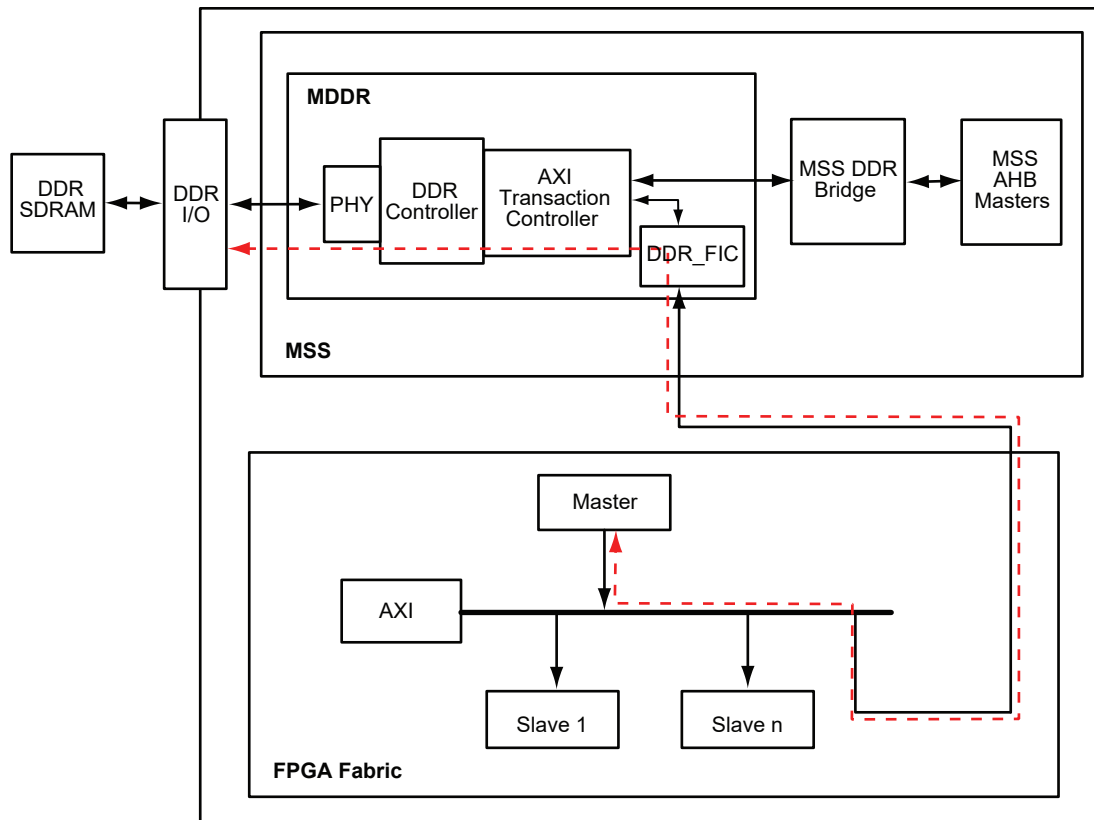
For more information about MDDR Subsystem Features Configuration, refer to the "MDDR Subsystem Features Configuration" section on page 25.

### 3.12.3 Use Model 1: Accessing MDDR from FPGA Fabric Through the AXI Interface

The MDDR subsystem can be used to access DDR memory as shown in the following illustration. This use model follows the steps "Design flow using SmartDesign" for using MDDR. The AXI master in the FPGA fabric accesses the DDR memory through the MDDR subsystem. The MDDR registers are configured from FPGA fabric through the APB interface. The APB master in the FPGA fabric asserts a ready signal to indicate that the DDR memory is successfully initialized.

The read, write, and read-modify-write transactions are initiated by the AXI master to read or write the data into the DDR memory after receiving the ready signal from APB master.

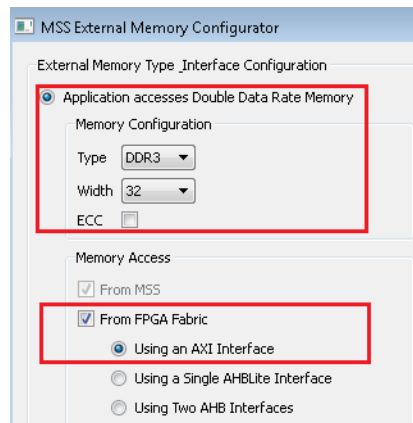
**Figure 63 • MDDR with AXI Interface**



Use the following steps to access the MDDR from the AXI master in the FPGA fabric:

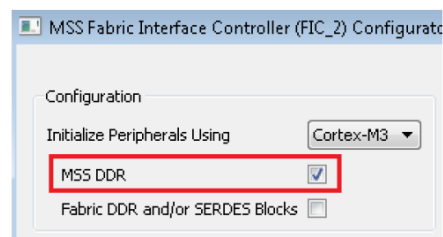
1. Instantiate the SmartFusion2 MSS component onto the SmartDesign canvas.
2. Configure the SmartFusion2 MSS peripheral components as required using the MSS configurator.
3. Configure the MDDR and select the AXI interface, as shown in the following image. In this example, the design is created to access DDR3 memory with a 32-bit data width.

**Figure 64 • MSS External Memory Configuration**



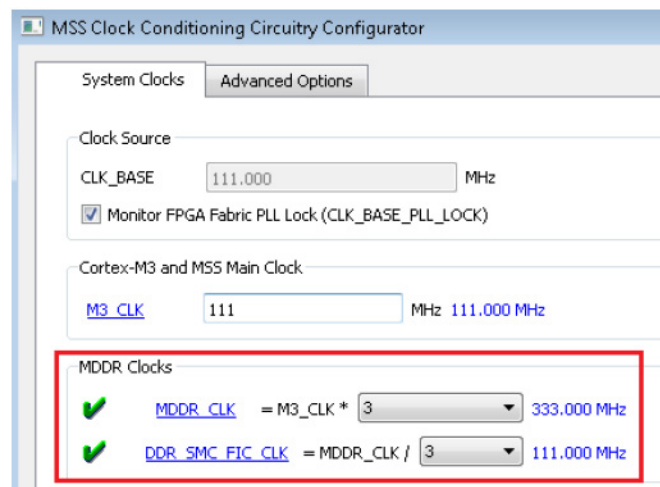
4. Configure FIC\_2, as shown in the following image, to enable the MDDR subsystem APB interface for configuring the MDDR registers using APB master in the FPGA fabric.

**Figure 65 • Configuring FIC\_2**



5. Configure the MSS\_CCC for MDDR\_CLK and DDR\_SMC\_FIC\_CLK. In the following image, the MDDR clock is configured to 333 MHz and M3\_CLK is configured to 111 MHz.

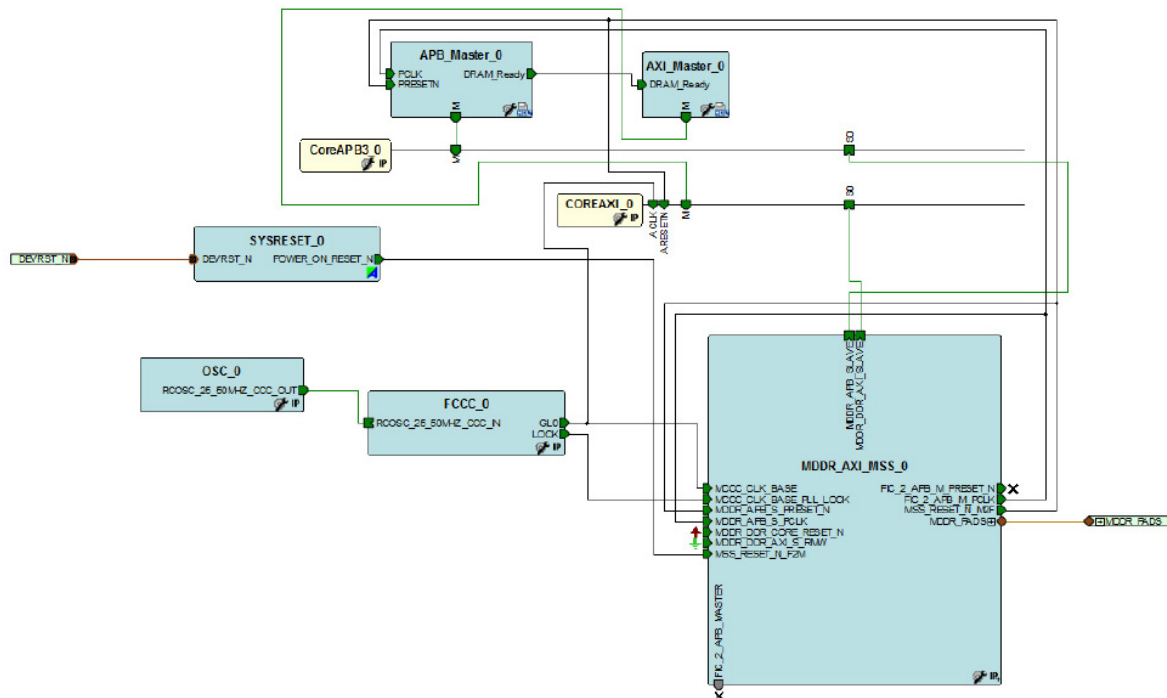
**Figure 66 • MDDR Clock Configuration**



6. Instantiate the clock resources (FCCC and chip oscillators) in the SmartDesign canvas and configure, as required.
7. Instantiate user AXI master logic in the SmartDesign canvas to access the MDDR through the AXI interface. Make sure that the AXI master logic accesses the MDDR after configuring the MDDR registers from the APB master. The AXI master clock should be same as DDR\_SMC\_FIC\_CLK.
8. Instantiate user APB master logic in the SmartDesign canvas to configure the MDDR registers through the APB interface. The APB master logic should initialize the registers after the MSS comes out of reset. The APB clock must be connected to FIC\_2\_APB\_M\_PCLK.
9. Connect the AXI master and APB master to the MSS component through CoreAXI and CoreAPB or use the auto connect option in SmartDesign.

Make the other connections in the SmartDesign canvas, as shown in the following illustration.

**Figure 67 • SmartDesign Canvas**

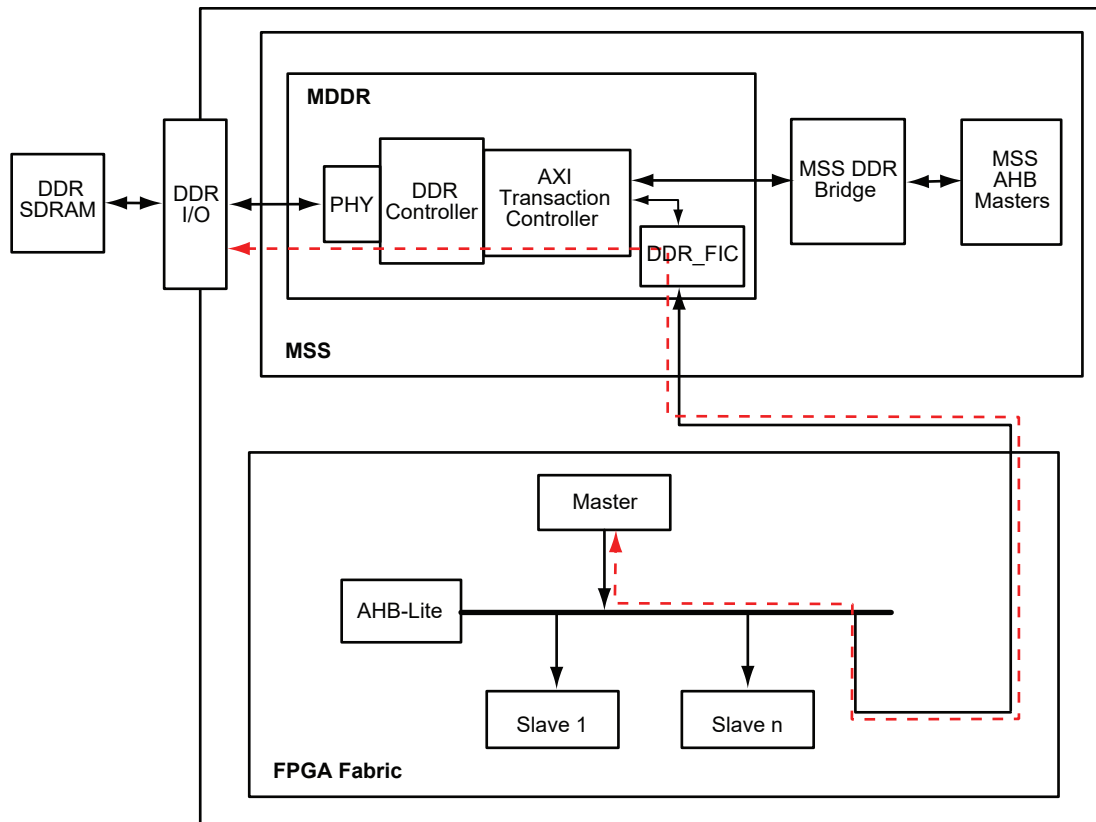


- To verify the design in Libero SoC software, create a SmartDesign testbench project and instantiate a DDR memory model provided by the DDR memory vendor. Simulate the design and observe the AXI read and write transactions.

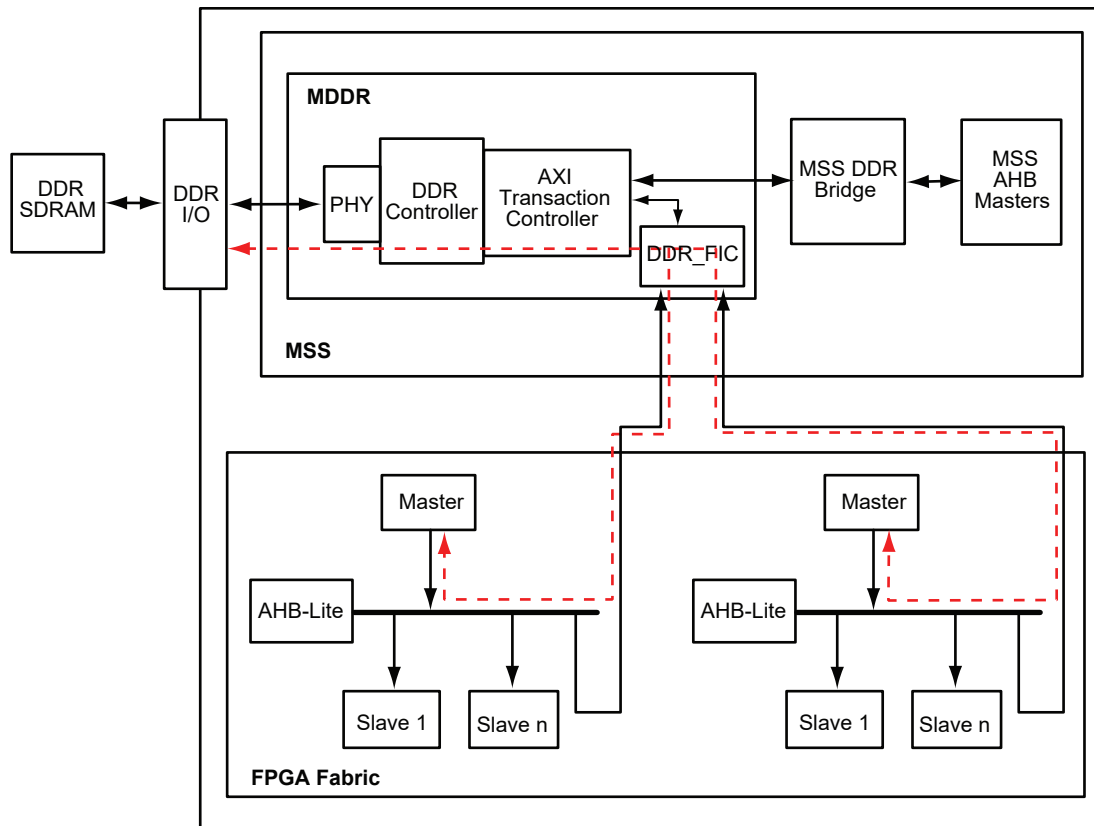
**Note:** The MDDR subsystem can be configured using the Cortex-M3 processor without having an APB master. The System Builder can be used to create the design by following steps in ["Design Flow Using System Builder" section on page 111](#). The System Builder provides "INIT\_DONE" to indicate that the DDR memory has been successfully initialized.

### 3.12.4 Use Model 2: Accessing MDDR from FPGA Fabric Through the AHB Interface

The MDDR subsystem can be used to access the DDR memory, as shown in the following illustration. The MDDR register can be configured through the MSS or user logic (AHB master) in the FPGA fabric.

**Figure 68 • MDDR with Single AHB Interface**


To use a dual rather than single AHB interface to the MDDR, set the CFG\_NUM\_AHB\_MASTERS bit in the "DDR\_FIC\_NUM\_AHB\_MASTERS\_CR" register to 1.

**Figure 69 • MDDR with Dual AHB Interface**

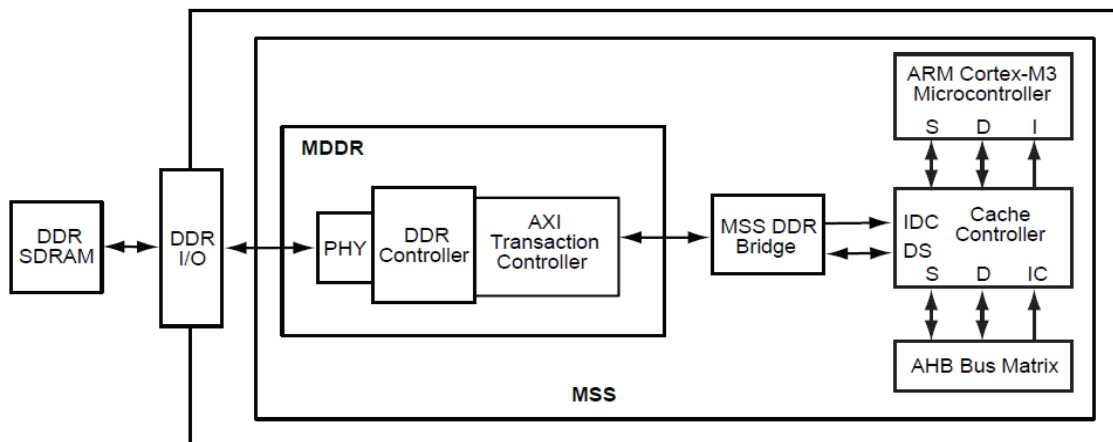
The steps for accessing the MDDR from one or two AHB masters in the FPGA fabric is the same as in "Use Model 1: Accessing MDDR from FPGA Fabric Through the AXI Interface" section on page 125 except for the following:

11. The single AHB or two AHB interfaces must be selected in the MSS external memory configurator instead of AXI master.
12. One or two AHB masters must be connected through CoreAHB's in the SmartDesign canvas.

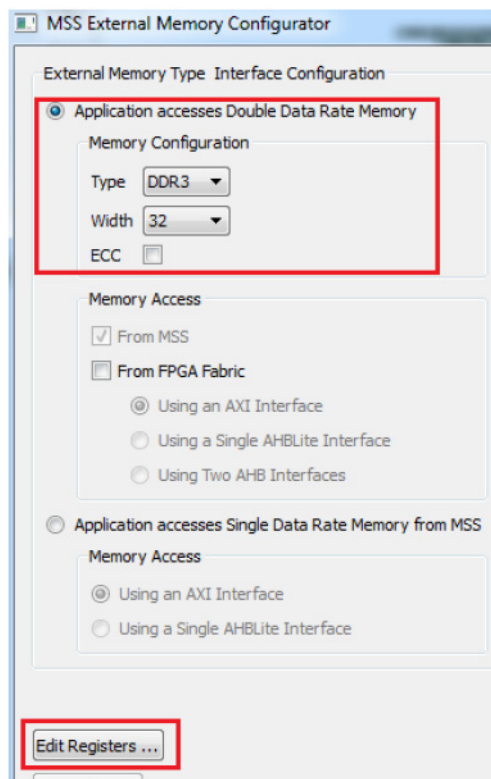
### 3.12.5 Use Model 3: Accessing MDDR from Cortex-M3 Processor

The Cortex-M3 processor can access the DDR SDRAM connected to the MDDR subsystem through the MSS DDR bridge, as shown in Figure 70, page 130. This use model follows the steps "Design Flow Using System Builder" for using MDDR. Use the following steps to access the MDDR from the Cortex-M3 processor:

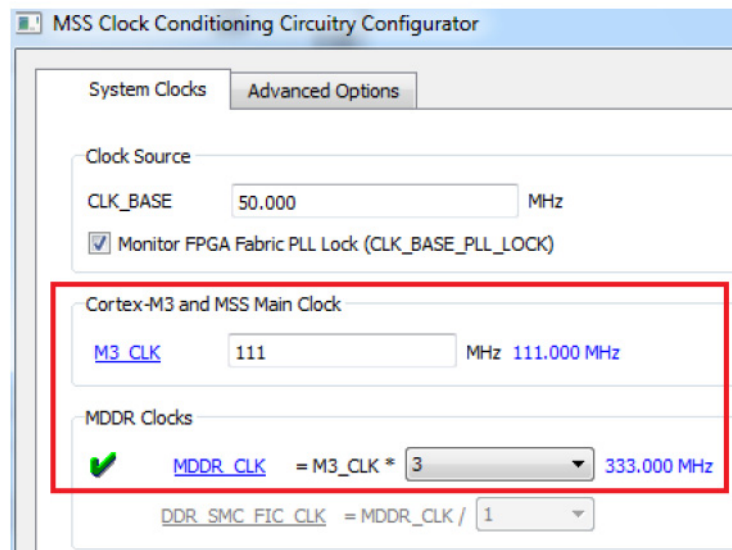


**Figure 70 • Accessing MDDR from Cortex-M3 Processor**


1. Go to the System Builder - Device Features tab and check the **MSS External Memory** check box and leave the rest of the check boxes unchecked. [Figure 16 on page 39](#) shows the **System Builder - Device Features** tab.

**Figure 71 • MSS External Memory Configuration**


2. Navigate to **Memories** tab and import the DDR configuration file or select the appropriate DDR memory settings. Refer to the "[MDDR Subsystem Features Configuration](#)" section on [page 25](#) to configure the necessary registers.
3. Navigate to **Clocks** tab to configure the MDDR\_CLK. In this example, MDDR\_CLK is configured to 333 MHz as shown in the following image.

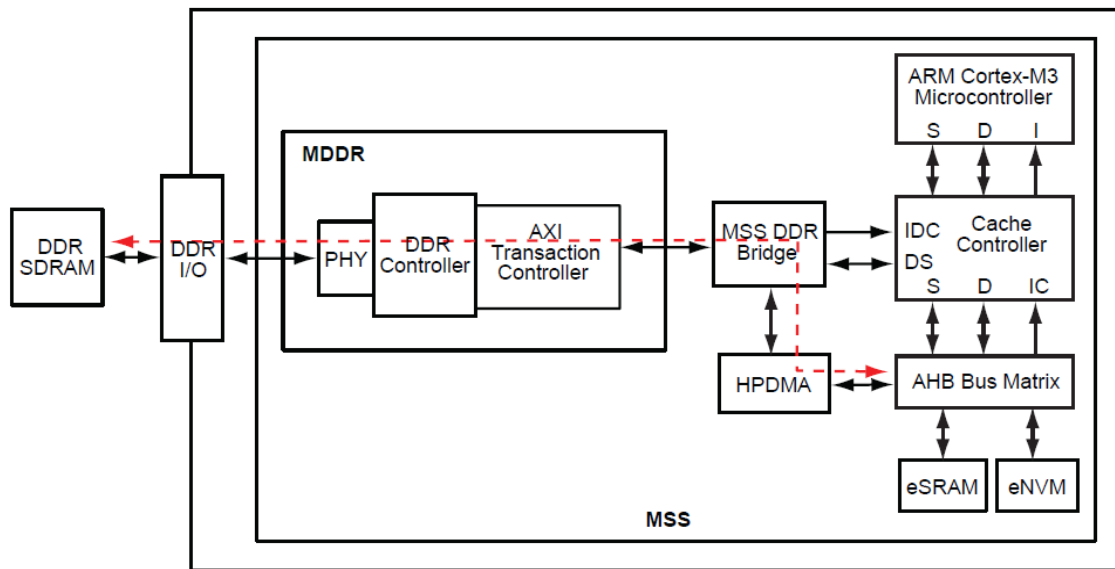
**Figure 72 • Configuring MDDR\_CLK**

4. Navigate to the **Memory Map** tab giving the required data in the rest of the **System Builder** tabs. Click **Finish**, the system builder creates the design and generates.
5. Connect the clock resources to the MSS component in the SmartDesign canvas.
6. To verify the design in Libero SoC software, create the SmartDesign testbench project and instantiate a DDR memory model provided by the DDR memory vendor.
7. Write BFM commands for read and write transactions. The MDDR\_init.bfm file will be generated by Libero SoC software, containing the BFM commands to initialize the MDDR registers.
8. Simulate the design to verify the read/write transactions to DDR memory.
9. Open I/O Attribute Editor to configure the ODT and drive strengths.
10. Program the device.
11. Use the generated firmware project to access the DDR memory from the Cortex-M3 processor through MDDR. The firmware project initializes the MDDR subsystem before executing the instructions in main() with the register settings provided in the above step 2.

Refer to the [MDDR Tutorial](#), which describes the steps to create the design for accessing the MDDR from the Cortex-M3 processor. The tutorial also explains the steps for simulating the design in Libero SoC.

### 3.12.6 Use Model 4: Accessing MDDR from the HPDMA

The HPDMA controller can access DDR SDRAM connected to the MDDR subsystem through the MSS DDR bridge, as shown in the following illustration.

**Figure 73 • Accessing MDDR from HPDMA**


The steps for accessing the MDDR from the HPDMA are the same as in ["Use Model 3: Accessing MDDR from Cortex-M3 Processor" section on page 129](#). Use the generated firmware project to access DDR memory from the HPDMA through the MDDR. The HPDMA driver has the `MSS_HPDM_start()` API to initiate memory transfers and DDR memory from and to other memory locations. This API requires the parameter's source address, destination address, and number of bytes to transfer. For more information on how to use HPDMA, refer to the HPDMA chapter in [UG0331: SmartFusion2 Microcontroller Subsystem User Guide](#).

For information on timing diagrams, refer to the ["Timing Diagrams" section on page 51](#).

## 4 Fabric DDR Subsystem

The FDDR is a hardened ASIC block for interfacing the DDR2, DDR3, and LPDDR1 memories. The FDDR subsystem is used to access DDR memories for high-speed data transfers. The FDDR subsystem includes the DDR memory controller, DDR PHY, and arbitration logic to support multiple masters.

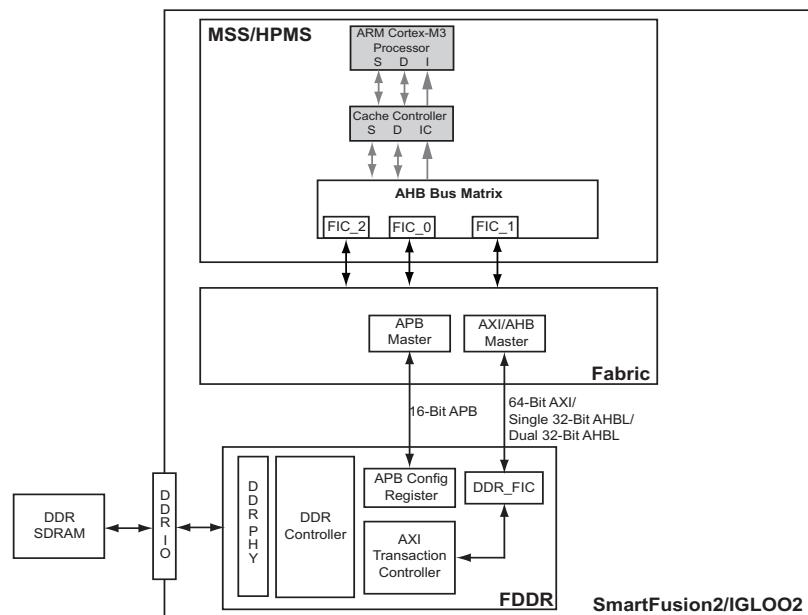
FPGA fabric masters communicate with the DDR memories interfaced to the FDDR subsystem through AXI or AHB interfaces.

### 4.1 Features

- Integrated on-chip DDR memory controller and PHY
- Configurable to support LPDDR1, DDR2, and DDR3 memory devices
- Up to 667 Mbps (333 MHz DDR) performance
- Supports memory densities up to 2 GB
- Supports 8/16/32-bit data bus width modes
- Supports a maximum of 8 memory banks
- Supports single rank memory
- Single error correction and double error detection (SECCDED) enable or disable feature
- Supports DRAM burst lengths of 4, 8, or 16, depending on the Bus-width mode and DDR type configuration
- Support for sequential and interleaved burst ordering
- Programs internal control for ZQ short calibration cycles for DDR3 configurations
- Supports dynamic scheduling to optimize bandwidth and latency
- Supports self refresh entry and exit on command
- Supports deep power-down entry and exit on command
- Flexible address mapper logic to allow application specific mapping of row, column, bank, and rank bits
- Configurable support for 1T or 2T timing on the DDR SDRAM control signals
- Supports autonomous DRAM power-down entry and exit caused by lack of transaction arrival for programmable time
- Advanced power-saving design includes toggling of command, address, and data pins

The system level block diagram of the FDDR subsystem is shown in the following illustration.

**Figure 74 • System Level FDDR Block Diagram**



*Note: Blocks and arrows marked in grey, only in SmartFusion2 MSS. Rest are all similar in MSS/HPMS.*

The FDDR subsystem accepts data transfer requests from AXI or AHB interfaces. Any read or write transactions to the DDR memories can occur through the AXI or AHBL masters in the FPGA fabric through DDR\_FIC interface.

**Note:** The maximum DDR3 data rate supported by FDDR is 333MHz/667Mbps. Therefore, Write Leveling is not mandatory and the interface works if the board layout includes length matching and follows [AC393 SmartFusion2 and IGLOO2 Board Design Guidelines Application Note](#). For Read Leveling, Libero SOC auto-generates pre-defined static delay ratios for FDDR initialization. These delay values are sufficient if the board layout follows the SmartFusion2/IGLOO2 board-level guidelines.

## 4.2 Memory Configurations

The SmartFusion2/IGLOO2 FDDR subsystem supports a wide range of common memory types, configurations, and densities, as shown in the following table. If SECDED mode is enabled in the FDDR controller, the external memory module must be connected to the following:

- Data lines FDDR\_DQ\_ECC[3:0] when data width is x32
- Data lines FDDR\_DQ\_ECC[1:0] when data width is x16
- Data line FDDR\_DQ\_ECC[0] when data width is x8

**Table 124 • Supported Memory (DDR2, DDR3, and LPDDR1) Configurations**

Memory Depth	Width	Width (in SECDED Mode)	SmartFusion2/IGLOO2 Devices		
			M2S150 (FCV484)	M2S050/M2GL050 (FG896)	M2S150/M2GL150 (FC1152)
128M or Less	x32	x36	—	√	√
	x16	x18	√	√	√
	x8	x9	√	—	√
256M	x32	x36	—	√	√
	x16	x18	√	√	√
	x8	x9	√	—	√
512M	x32	x36	—	√	√
	x16	x18	√	√	√
	x8	x9	√	—	√
1G	x32	x36	—	√	√
	x16	x18	√	√	√
	x8	x9	√	—	√

## 4.3 Performance

The following table shows the maximum data rates supported by the FDDR subsystem for supported memory types. For more information on DDR Speeds, Refer to the "DDR Memory Interface Characteristics" section in [DS0128: IGLOO2 and SmartFusion2 Datasheet](#).

**Table 125 • DDR Speeds**

Memory Type	Maximum Data Rate (Mbps)
LPDDR1	400 Mbps (200 MHz)
DDR2	667 Mbps (333 MHz)
DDR3	667 Mbps (333 MHz)

## 4.4 I/O Utilization

The following table shows the I/O utilization for SmartFusion2 and IGLOO2 devices corresponding to supported bus widths. The remaining I/Os in bank 0 can be used for general purposes.

**Table 126 • I/O Utilization for SmartFusion2 and IGLOO2 Devices**

FDDR Bus Width	M2S050/M2GL050 (FG896)	M2S150/M2GL150 (FC1152)
36-bit	Bank5 (85 pins)	Bank1 (85 pins)
32-bit	Bank5 (76 pins)	Bank1 (76 pins)
18-bit	Bank5 (59 pins)	Bank1 (59 pins)
16-bit	Bank5 (53 pins)	Bank1 (53 pins)
9-bit	–	Bank1 (47 pins)
8-bit	–	Bank1 (41 pins)

**Note:** If FDDR is configured for LPDDR, one more IO also available for every 8-bit as the LPDDR doesn't have DQS\_N.

For general purpose use of the unused I/Os in the FDDR bank, select one of the I/O standards with the same voltage level as the DDR I/Os.

Self refresh must be disabled if the FDDR banks contain a mixed of I/Os used for DDR and for general purpose fabric I/Os. For more information, see “Self Refresh” in the ["Power Saving Modes" section on page 150](#).

## 4.5 Functional Description

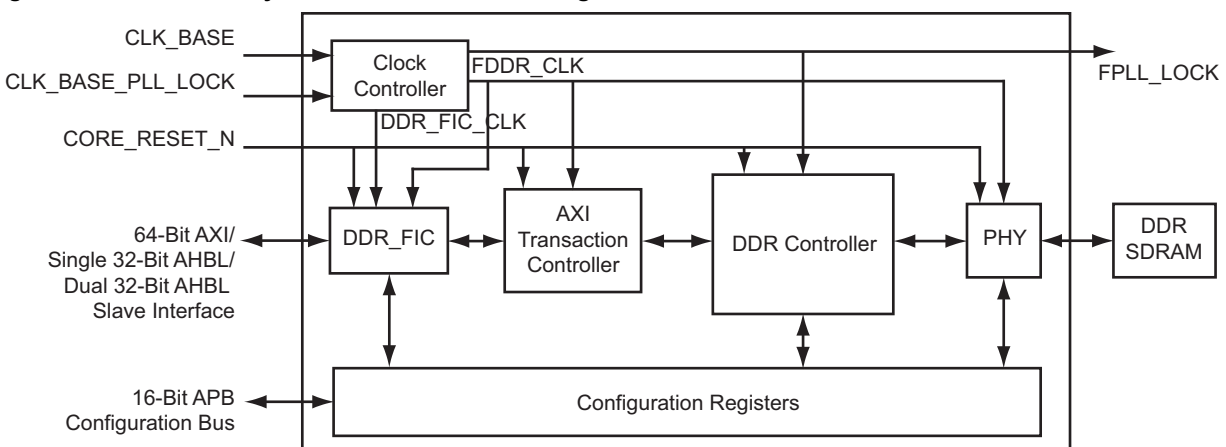
This section provides a detailed description of the FDDR subsystem with the following sub-sections:

- [Architecture Overview](#)
- [Port List](#)
- [Initialization](#)
- [Details of Operation](#)

### 4.5.1 Architecture Overview

A functional block diagram of the FDDR subsystem is shown in the following illustration. The main components include the DDR fabric interface controller (DDR\_FIC), AXI transaction handler, DDR memory controller, and DDR PHY.

**Figure 75 • FDDR Subsystem Functional Block Diagram**



The FDDR subsystem has a dedicated clock controller for generating clocks to the components of FDDR from the base clock (CLK\_BASE). The CLK\_BASE for the FDDR originates from a fabric CCC or an external source through the FPGA fabric.

The DDR\_FIC facilitates communication between the FPGA fabric masters and AXI transaction controller. The DDR\_FIC can be configured to provide either one 64-bit AXI slave interface or two independent 32-bit AHB-Lite (AHBL) slave interfaces to the FPGA fabric masters.

The AXI transaction controller receives read and write requests from AXI masters (DDR\_FIC) and schedules for the DDR controller by translating them into DDR controller commands.

The DDR controller receives the commands from the AXI transaction controller. These commands are queued internally and scheduled for access to the DDR SDRAM while satisfying DDR SDRAM constraints, transaction priorities, and dependencies between the transactions. The DDR controller in turn issues commands to the PHY module, which launches and captures data to and from the DDR SDRAM.

DDR PHY receives commands from the DDR controller and generates DDR memory signals required to access the external DDR memory.

The 16-bit APB configuration bus provides an interface for configuring the FDDR subsystem registers.

## 4.5.2 Port List

**Table 127 • FDDR Subsystem Interface Signals**

Signal Name	Type	Polarity	Description
APB_S_PCLK	In	–	APB clock. This clock drives all the registers of the APB interface.
APB_S_PRESET_N	In	Low	APB reset signal. This is an active low signal. This drives the APB interface and is used to generate the soft reset for the DDR controller as well.
CORE_RESET_N	In	Low	Global reset. This resets the DDR_FIC/DDRC/PHY/DDRAXI logic.
FDDR_SUBSYSTEM_CLK	In	–	Base clock to the FDDR clock controller. This clock is used as the reference clock to the fabric phase-locked loop (FPLL). The user sets the multiplier of the FPLL based on the rate of the AXI/AHB interface. This is done by setting the DDR_FIC divider setting in the System Builder Clocks tab.
AXI_S_RMW	In	High	AXI mode only Indicates whether all bytes of a 64-bit lane are valid for all beats of an AXI transfer. 0: Indicates that all bytes in all beats are valid in the burst and the controller should default to write commands. 1: Indicates that some bytes are invalid and the controller should default to RMW commands. This is classed as an AXI write address channel sideband signal and is valid with the AWVALID signal. Only used when SECDED is enabled.
HPMS_DDR_FIC_SUBSYSTEM_CLK <sup>3</sup>	Out	–	This output clock is derived from the FDDR_CLK and is based on the DDR_FIC divider ratio. This is the clock that should be used for the AXI or AHB slave interfaces to move data in and out of the FDDR.
HPMS_DDR_FIC_SUBSYSTEM_LOCK <sup>3</sup>	Out	–	HPMS_DDR_FIC_SUBSYSTEM_LOCK indicates the lock from FCCC which generates HPMS_DDR_FIC_SUBSYSTEM_CLK
FDDR_SUBSYSTEM_CLK_PLL_LOCK	In	High	Fabric PLL lock input
<b>Slave Interfaces</b>			
AXI_SLAVE*	Bus	–	AXI slave interface 1.0 bus
AHB0_SLAVE*	Bus	–	AHB0 slave interface 3.0 bus
AHB1_SLAVE*	Bus	–	AHB1 slave interface 3.0 bus

**Notes:**

1. \*AXI or AHB interface, depending on configuration.
2. FDDR\_DQS\_N[3:0] signals are not available for LPDDR.
3. Only in IGLOO2 Devices.
4. TMATCH\_IN and TMATCH\_OUT pins are required to be connected together outside the device. They are used for gate training as part of the read data capture operation. The two pins create an internal DQS Enable signal that is used to calibrate the flight path. DQS needs to be gated to prevent false triggering of the FIFO write clock. This DQS Enable signal is derived from the system clock and physically matches the clock output buffer and DQS input buffer to compensate for I/O buffer uncertainty due to Process-Voltage-Temperature (PVT) changes. Without this connection, the circuit is not operable.



**Table 127 • FDDR Subsystem Interface Signals (continued)**

Signal Name	Type	Polarity	Description
APB_SLAVE	Bus	–	APB slave interface 3.0 bus
<b>DRAM Interface</b>			
FDDR_CAS_N	Out	Low	DRAM CASN
FDDR_CKE	Out	High	DRAM CKE
FDDR_CLK	Out	–	DRAM single-ended clock – for differential pads
FDDR_CLK_N	Out	–	DRAM single-ended clock – for differential pads
FDDR_CS_N	Out	Low	DRAM CSN
FDDR_ODT	Out	High	DRAM ODT. 0: Termination Off 1: Termination On
FDDR_RAS_N	Out	Low	DRAM RASN
FDDR_RESET_N	Out	Low	DRAM reset for DDR3
FDDR_WE_N	Out	Low	DRAM WEN
FDDR_ADDR[15:0]	Out	–	Dram address bits
FDDR_BA[2:0]	Out	–	Dram bank address
FDDR_DM_RDQS[3:0]	In/out	–	DRAM data mask – from bidirectional pads
FDDR_DQS[3:0]	In/out	–	DRAM single-ended data strobe output – for bidirectional pads
FDDR_DQS_N[3:0]	In/out	–	DRAM single-ended data strobe output – for bidirectional pads
FDDR_DQ[31:0]	In/out	–	DRAM data input or output – for bidirectional pads
FDDR_DQ_ECC[3:0]	In/out	–	DRAM data input or output for SECDED
FDDR_DM_RDQS_ECC	In/out	High	DRAM single-ended data strobe output – for bidirectional pads
FDDR_DQS_ECC	In/out	High	DRAM single-ended data strobe output – for bidirectional pads
FDDR_DQS_ECC_N	In/out	Low	DRAM data input or output – for bidirectional pads
FDDR_DQS_TMATCH_0_IN	In	High	DQS enables input for timing match between DQS and system clock. For simulations, tie to FDDR_DQS_TMATCH_0_OUT.
FDDR_DQS_TMATCH_1_IN	In	High	DQS enables input for timing match between DQS and system clock. For simulations, tie to FDDR_DQS_TMATCH_1_OUT.

**Notes:**

1. \*AXI or AHB interface, depending on configuration.
2. FDDR\_DQS\_N[3:0] signals are not available for LPDDR.
3. Only in IGLOO2 Devices.
4. TMATCH\_IN and TMATCH\_OUT pins are required to be connected together outside the device. They are used for gate training as part of the read data capture operation. The two pins create an internal DQS Enable signal that is used to calibrate the flight path. DQS needs to be gated to prevent false triggering of the FIFO write clock. This DQS Enable signal is derived from the system clock and physically matches the clock output buffer and DQS input buffer to compensate for I/O buffer uncertainty due to Process-Voltage-Temperature (PVT) changes. Without this connection, the circuit is not operable.

**Table 127 • FDDR Subsystem Interface Signals (continued)**

Signal Name	Type	Polarity	Description
FDDR_DQS_TMATCH_0_OUT	Out	High	DQS enables output for timing match between DQS and system clock. For simulations, tie to FDDR_DQS_TMATCH_0_IN.
FDDR_DQS_TMATCH_1_OUT	Out	High	DQS enables output for timing match between DQS and system clock. For simulations, tie to FDDR_DQS_TMATCH_1_IN.
FDDR_DQS_TMATCH_ECC_IN	In	High	DQS enables input for timing match between DQS and system clock. For simulations, tie to FDDR_DQS_TMATCH_ECC_OUT.
FDDR_DQS_TMATCH_ECC_OUT	Out	High	DQS enables output for timing match between DQS and system clock. For simulations, tie to FDDR_DQS_TMATCH_ECC_IN.

**Notes:**

1. \*AXI or AHB interface, depending on configuration.
2. FDDR\_DQS\_N[3:0] signals are not available for LPDDR.
3. Only in IGLOO2 Devices.
4. TMATCH\_IN and TMATCH\_OUT pins are required to be connected together outside the device. They are used for gate training as part of the read data capture operation. The two pins create an internal DQS Enable signal that is used to calibrate the flight path. DQS needs to be gated to prevent false triggering of the FIFO write clock. This DQS Enable signal is derived from the system clock and physically matches the clock output buffer and DQS input buffer to compensate for I/O buffer uncertainty due to Process-Voltage-Temperature (PVT) changes. Without this connection, the circuit is not operable.

**4.5.2.1 AXI Slave Interface**

The following table describes the FDDR AXI slave interface signals. These signals are available only if FDDR interface is configured for AXI mode. For more details of AXI protocol, refer to [AMBA AXI v1.0 protocol specification](#).

**Table 128 • FDDR AXI Slave Interface Signals**

Signal Name	Direction	Polarity	Description
AXI_S_ARREADY	Output	High	Indicates whether the slave is ready to accept an address and associated control signals. 1: Slave ready 0: Slave not ready
AXI_S_AWREADY	Output	High	Indicates that the slave is ready to accept an address and associated control signals. 1: Slave ready 0: Slave not ready
AXI_S_BID[3:0]	Output		Indicates response ID. The identification tag of the write response.
AXI_S_BRESP[1:0]	Output		Indicates write response. This signal indicates the status of the write transaction. 00: Normal access okay 01: Exclusive access okay 10: Slave error 11: Decode error
AXI_S_BVALID	Output	High	Indicates whether a valid write response is available. 1: Write response available 0: Write response not available.

**Table 128 • FDDR AXI Slave Interface Signals (continued)**

Signal Name	Direction	Polarity	Description
AXI_S_RDATA[63:0]	Output		Indicates read data.
AXI_S_RID[3:0]	Output		Read ID tag. This signal is the ID tag of the read data group of signals.
AXI_S_RLAST	Output	High	Indicates the last transfer in a read burst.
AXI_S_RRESP[1:0]	Output		Indicates read response. This signal indicates the status of the read transfer. 00: Normal access okay 01: Exclusive access okay 10: Slave error 11: Decode error
AXI_S_RVALID	Output		Indicates whether the required read data is available and the read transfer can complete. 1: Read data available 0: Read data not available
AXI_S_WREADY	Output	High	Indicates whether the slave can accept the write data. 1: Slave ready 0: Slave not ready
AXI_S_ARADDR[31:0]	Input		Indicates initial address of a read burst transaction.
AXI_S_ARBURST[1:0]	Input		Indicates burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. 00: FIXED - Fixed-address burst FIFO type 01: INCR - Incrementing-address burst normal sequential memory 10: WRAP - Incrementing-address burst that wraps to a lower address at the wrap boundary 11: Reserved
AXI_S_ARID[3:0]	Input		Indicates identification tag for the read address group of signals.
AXI_S_ARLEN[3:0]	Input		Indicates burst length. The burst length gives the exact number of transfers in a burst. 0000: 1 0001: 2 0010: 3 0011: 4 0100: 5 0101: 6 0110: 7 0111: 8 1000: 9 1001: 10 1010: 11 1011: 12 1100: 13 1101: 14 1110: 15 1111: 16
AXI_S_ARLOCK[1:0]	Input		Indicates lock type. This signal provides additional information about the atomic characteristics of the read transfer. 00: Normal access 01: Exclusive access 10: Locked access 11: Reserved

**Table 128 • FDDR AXI Slave Interface Signals (continued)**

Signal Name	Direction	Polarity	Description
AXI_S_ARSIZE[1:0]	Input		Indicates the maximum number of data bytes to transfer in each data transfer, within a burst. 00: 1 01: 2 10: 4 11: 8
AXI_S_ARVALID	Input	High	Indicates the validity of read address and control information. 1: Address and control information valid 0: Address and control information not valid
AXI_S_AWADDR[31:0]	Input		Indicates write address. The write address bus gives the address of the first transfer in a write burst transaction.
AXI_S_AWBURST[1:0]	Input		Indicates burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. 00: FIXED - Fixed-address burst FIFO-type 01: INCR - Incrementing-address burst normal sequential memory 10: WRAP - Incrementing-address burst that wraps to a lower address at the wrap boundary 11: Reserved
AXI_S_AWID[3:0]	Input		Indicates identification tag for the write address group of signals.
AXI_S_AWLEN[3:0]	Input		Indicates burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. 0000: 1 0001: 2 0010: 3 0011: 4 0100: 5 0101: 6 0110: 7 0111: 8 1000: 9 1001: 10 1010: 11 1011: 12 1100: 13 1101: 14 1110: 15 1111: 16
AXI_S_AWLOCK[1:0]	Input		Indicates lock type. This signal provides additional information about the atomic characteristics of the write transfer. 00: Normal access 01: Exclusive access 10: Locked access 11: Reserved
AXI_S_AWSIZE[1:0]	Input		Indicates the maximum number of data bytes to transfer in each data transfer, within a burst. 00: 1 01: 2 10: 4 11: 8

**Table 128 • FDDR AXI Slave Interface Signals (continued)**

Signal Name	Direction	Polarity	Description
AXI_S_AWVALID	Input	High	Indicates whether valid write address and control information are available. 1: Address and control information available 0: Address and control information not available
AXI_S_BREADY	Input	High	Indicates whether the master can accept the response information. 1: Master ready 0: Master not ready
AXI_S_RREADY	Input	High	Indicates whether the master can accept the read data and response information. 1: Master ready 0: Master not ready
AXI_S_WDATA[63:0]	Input		Indicates write data.
AXI_S_WID[3:0]	Input		Indicates response ID. The identification tag of the write response.
AXI_S_WLAST	Input	High	Indicates the last transfer in a write burst.
AXI_S_WSTRB[7:0]	Input		Indicates which byte lanes to update in memory.
AXI_S_WVALID	Input	High	Indicates whether valid write data and strobes are available. 1: Write data and strobes available 0: Write data and strobes not available

#### 4.5.2.2 AHB Slave

The following table describes the FDDR AHB slave interface signals. These signals are available only if FDDR interface is configured for single or dual AHB mode. For more details of AHB protocol refer to [AMBA AHB v3.0 protocol specification](#).

**Table 129 • FDDR AHB Slave Interface Signals**

Signal Name	Direction	Polarity	Description
AHBx_S_HREADYOUT	Output	High	Indicates that a transfer has finished on the bus. The signal is asserted LOW to extend a transfer. Input to Fabric master.
AHBx_S_HRESP	Output	High	Indicates AHB transfer response to Fabric master.
AHBx_S_HRDATA[31:0]	Output		Indicates AHB read data to Fabric master.
AHBx_S_HSEL	Input	High	Indicates AHB slave select signal from Fabric master.
AHBx_S_HADDR[31:0]	Input		Indicates AHB address initiated by Fabric master.
AHBx_S_HBURST[2:0]	Input		Indicates AHB burst type from Fabric master. 000: Single burst 001: Incrementing burst of undefined length 010: 4-beat wrapping burst 011: 4-beat incrementing burst 100: 8-beat wrapping burst 101: 8-beat incrementing burst 110: 16-beat wrapping burst 111: 16-beat incrementing burst
AHBx_S_HSIZE[1:0]	Input		Indicates AHB transfer size from Fabric master. 00: 8 Byte 01: 16 Halfword 10: 32 Word

**Table 129 • FDDR AHB Slave Interface Signals (continued)**

Signal Name	Direction	Polarity	Description
AHBx_S_HTRANS[1:0]	Input		Indicates AHB transfer type from Fabric master. 00: IDLE 01: BUSY 10: NONSEQUENTIAL 11: SEQUENTIAL
AHBx_S_HMASTLOCK	Input	High	Indicates AHB master lock signal from Fabric master.
AHBx_S_HWRITE	Input	High	Indicates AHB write control signal from Fabric master.
AHBx_S_HREADY	Input	High	Indicates that a transfer has finished on the bus. Fabric master can drive this signal LOW to extend a transfer.
AHBx_S_HWDATA[31:0]	Input		Indicates AHB write data from Fabric master.

**Note:** AHBx indicates AHB0 or AHB1.

### 4.5.2.3 APB Slave

The following table describes the FDDR APB slave interface signals. For APB protocol details, refer to [AMBA APB v3.0 protocol specification](#).

**Table 130 • FDDR APB Slave Interface Signals**

Signal Name	Direction	Polarity	Description
APB_S_PREADY	Output	High	Indicates APB Ready signal to Fabric master.
APB_S_PSLVERR	Output	High	Indicates error condition on an APB transfer to Fabric master.
APB_S_PRDATA[15:0]	Output		Indicates APB read data to Fabric master.
APB_S_PENABLE	Input	High	Indicates APB enable from Fabric master. The enable signal is used to indicate the second cycle of an APB transfer.
APB_S_PSEL	Input	High	Indicates APB slave select signal from Fabric master
APB_S_PWRITE	Input	High	Indicates APB write control signal from Fabric master
APB_S_PADDR[10:2]	Input		Indicates APB address initiated by Fabric master.
APB_S_PWDATA[15:0]	Input		Indicates APB write data from Fabric master.

## 4.6 Initialization

After power-up, the FDDR needs to have all of the configuration registers written to establish the operating modes of the blocks. When using the **System Builder** design flow through Libero SoC, this is all handled for the user through the use of the **System Builder** module. All of the configuration register values are selected by the user and stored in a special portion of the eNVM. Before the FDDR subsystem is active, it goes through an initialization phase and this process starts with a reset sequence. For DDR3 memories, the initialization phase also includes ZQ calibration and DRAM training.

### 4.6.1 Reset Sequence

The following illustration shows the required reset sequence for FDDR subsystem from the power-on-reset stage. The CORE\_RESET\_N signal of the FDDR subsystem must be asserted after MSS\_RESET\_N\_M2F/ HPMS\_RESET\_N\_M2F and FDDR FPLL lock go High and APB register configuration is complete. Assertion of CORE\_RESET\_N signifies the end of the reset sequence. The DDR controller performs external DRAM memory reset and initialization as per the JEDEC specification, including reset, refresh, and mode registers.

### 4.6.1.1 DDRIO Calibration

Each DDRIO has an ODT feature, which is calibrated depending on the DDR I/O standard. DDR I/O calibration occurs after the DDR I/Os are enabled. If the impedance feature is enabled, impedance can be programmed to the desired value in three ways:

- Calibrate the ODT/driver impedance with a calibration block (recommended)
- Calibrate the ODT/driver impedance with fixed calibration codes
- Configure the ODT/driver impedance to the desired value directly

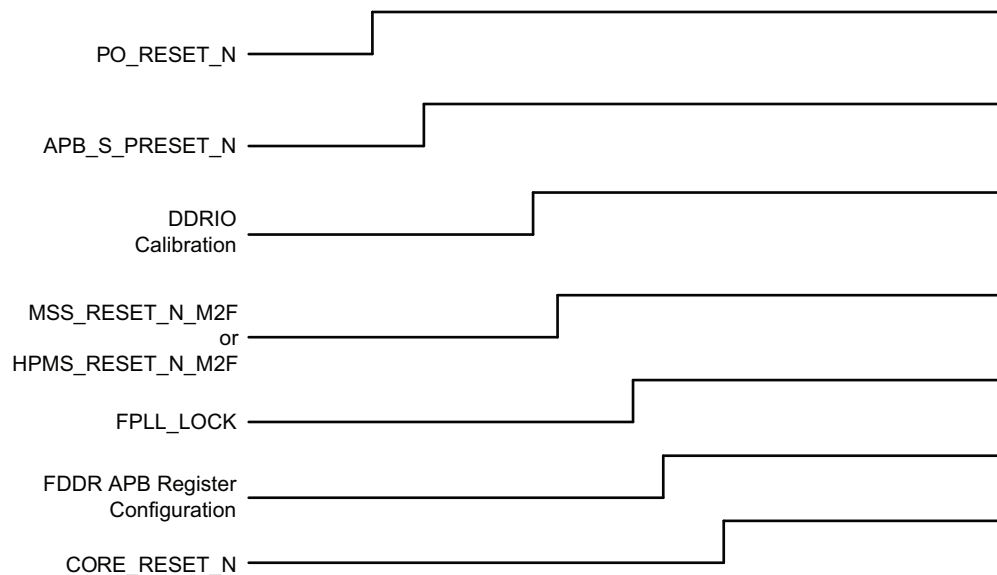
The I/O calibration is always enabled when the DDR subsystem is configured for DDR2 and DDR3 memories.

The I/O calibration can be disabled or enabled using the DDR configurator when the DDR subsystem is configured for LPDDR memories.

**Note:** If I/O calibration is enabled, all I/Os in the DDR bank are calibrated even though the DDR controller is not using all I/Os in the bank.

The FDDR\_IO\_CALIB\_CR register can be configured for changing the ODT value to the desired value. For more information on DDR I/O calibration, refer to the Configurable ODT and Driver Impedance section of the I/Os chapter in the [UG0445: IGLOO2 FPGA and SmartFusion2 SoC FPGA Fabric User Guide](#).

**Figure 76 • Reset Sequence**



### 4.6.2 ZQ Calibration

ZQ calibration is applicable for DDR3 only. This is used to calibrate DRAM output drivers ( $R_{ON}$ ) and on-die termination (ODT) values. DDR3 SDRAM needs a longer time to calibrate  $R_{ON}$  and ODT at initialization and a relatively smaller time to perform periodic calibrations.

The DDR controller performs ZQ calibration by issuing a ZQ calibration long (ZQCL) command and ZQ calibration short (ZQCS) command.

ZQCL is used to perform initial calibration during the power-up initialization sequence. This command is allowed for a period of  $t_{ZQinit}$ , as specified by memory vendor. The value of  $t_{ZQinit}$  can be modified through register bits REG\_DDRC\_T\_ZQ\_LONG\_NOP, [Table 60](#), page 82.

The ZQCS command is used to perform periodic calibration to account for voltage and temperature variations. A shorter timing window is provided to perform calibration and transfer of values as defined by timing parameter  $t_{ZQCS}$ . The  $t_{ZQCS}$  parameter can be modified through register bits REG\_DDRC\_T\_ZQ\_SHORT\_NOP, [Table 61](#), page 82.

Other activities are not performed by the controller for the duration of  $t_{ZQinit}$  and  $t_{ZQCS}$ . All DRAM banks are precharged and  $t_{RP}$  met before ZQCL or ZQCS commands are issued by the DDR controller.

#### 4.6.2.1 DRAM Training

High Speed DDR3 memories typically requires the DDR controller to implement Write-Leveling, Read DQS Gate Training, and Read Data Eye Training. However, FDDR only supports a maximum data rate of 333 MHz/667 Mbps, which means the clock period and data window are relatively large compared to high-speed DDR3 memory interfaces. Therefore dynamic write-leveling and read training are not performed. The following sections describe how write-leveling and read training are addressed by the FDDR.

##### 4.6.2.1.1 Write Leveling

Dynamic write-leveling is not required for the FDDR controller. The board-layout needs to follow AC393 SmartFusion2 and IGLOO2 Board Design Guidelines Application Note to keep the skew between DQS and CK within the JEDEC DDR3 tDQSS limit of +/- 750ps at each memory device. For board layouts which do not meet the Board Design Guidelines, the FDDR controller allows static delay ratios which delays DQS for each byte lane so that the skew between DQS and CK is kept within JEDEC limits.

##### 4.6.2.1.2 Read Leveling

FDDR does not perform dynamic Read DQS Gate Training and Data Eye Training. Instead, these functions are achieved by using built-in static delay values automatically generated by Libero SoC for the FDDR automatic register initialization.

##### 5. Read Gate

The DQS gate is aligned by the Libero SoC auto-generated FDDR initialization code containing fixed delay ratios to account for board round-trip time between FPGA and the DDR3 memory. The TMATCH\_OUT and TMATCH\_IN signals are shorted close to the FPGA balls to remove the FPGA output and input delays from the round trip delay time. Therefore, the fixed delay ratios represent only the board delay.

The fixed delay ratios work in combination with board layouts which follow the SmartFusion2/IGLOO2 Board Design Guidelines (refer [AC393: Board Design Guidelines for SmartFusion2/IGLOO2 FPGA Application Note](#)).

##### 4.6.2.1.3 DQS Alignment within Data Eye

The incoming read DQS is internally centered within the read DQ data window using a static delay ratio. This static delay is applied by the Libero SoC auto-generated FDDR initialization code. The fixed delay ratios work in combination with board layouts which follow the SmartFusion2/IGLOO2 Board Design Guidelines (refer [AC393: Board Design Guidelines for SmartFusion2/IGLOO2 FPGA Application Note](#)).

#### 4.6.2.2 DDR Memory Initialization Time

The time to initialize the DDR memory depends on the following factors:

- Power-up and register initialization by system controller. It depends on the power on reset delay configuration in the Libero project (**Project > Project Settings > Device settings**).
- DDR controller and PHY configuration registers initialization. In SmartFusion2 devices, the Cortex-M3 initializes these registers. In IGLOO2 devices, the ConfigMaster in the FPGA fabric initializes these registers.
- DDR memory initialization by the DDR Controller according to the JEDEC standard (mode register configuration and training).
- DDR memory settling time configured in the System Builder memory configuration window.

### 4.6.3 Details of Operation

This section provides a functional description of each block in the FDDR subsystem, as shown in [Figure 77 on page 147](#).

#### 4.6.3.1 Clock Controller

The FDDR subsystem has a dedicated clock controller for generating aligned clocks to all the FDDR sub-blocks for correct operation and synchronous communication with user logic in the FPGA fabric. The



base clock (FDDR\_SUBSYSTEM\_CLK) for the FDDR comes from a fabric CCC or an external source through the FPGA fabric. The FDDR clock controller is associated with a dedicated PLL (FPLL) for clock synthesis and de-skewing the internal DDR\_FIC clock from the base clock.

The FDDR clock controller consists of an FPLL and fabric alignment clock controller (FACC).

#### 4.6.3.1.1 FPLL

The FDDR\_SUBSYSTEM\_CLK from the FPGA fabric is used as a reference clock to the FPLL, and is multiplied to generate a clock frequency of up to 333 MHz. The FDDR\_SUBSYSTEM\_CLK can be generated from a fabric CCC/PLL, one of the on-chip oscillators, or directly from multi-standard user I/Os (MSIO) through FPGA fabric.

The supplies required to power the FPLL are the device core supply (VDD) for the digital section and the analog supply (FDDR\_PLL\_VDDA) for analog section. The required voltage for the FDDR\_PLL\_VDDA is 2.5 V or 3.3 V, based on the power supply availability on the board. The analog power supply voltage (2.5 V or 3.3 V) does not impact the FPLL frequency range. Refer to the DS0128: IGLOO2 and SmartFusion2 Datasheet or the FPLL operational range and characteristics.

The FPLL generates a lock signal (FPLL\_LOCK) to indicate that the FPLL is locked onto the FDDR\_SUBSYSTEM\_CLK signal. The precision of the FPLL\_LOCK discrimination can be adjusted using the lock window controls. The lock window represents the phase error window for lock assertion. The lock window can be adjusted between 500 parts per million (ppm) and 32,000 ppm in powers of 2. The integration of the lock period can be adjusted using a built-in lock counter. The lock counter or lock delay indicates the number of reference clock cycles to wait after the FPLL is locked for asserting the FPLL\_LOCK signal. The lock delay is useful for avoiding false toggling of the FPLL lock signal. The lock counter can be configured between 32 and 32,768 cycles in multiples of 2.

There are two interrupts to indicate FPLL lock assertion and deassertion.

#### 4.6.3.1.2 FACC

Within the FDDR clock controller, the FACC is responsible for interfacing with the FPLL, generating the aligned clocks required by the FDDR subsystem, and controlling the alignment of FPGA fabric interface clocks.

The clocks generated by the FACC are as follows:

- FDDR\_CLK clocks the FDDR subsystem. FDDR\_CLK can be operated up to 333 MHz, depending on the type of DDR present in the system.
- FDDR\_SUBSYSTEM\_CLK clocks the DDR\_FIC, and defines the frequency at which the connected FPGA fabric subsystem is intended to operate.
- The possible FDDR\_CLK:DDR\_FIC\_CLK ratios are 1:1, 2:1, 3:1, 4:1, 6:1, 8:1, 12:1, and 16:1.

The FACC includes no-glitch multiplexers (NGMUXs) to feed the DDR\_FIC clock with a standby clock (CK\_STANDBY) during the FPLL initialization. During initialization, the FDDR is not operational until after FPLL lock is achieved. However, the glitch-free multiplexers are still used to ensure that the clock being driven to DDR\_FIC during this time comes from the RC oscillator, avoiding the potentially high frequency output of the FPLL, which may be outside of the supported range of operation of DDR\_FIC.

#### 4.6.3.1.3 FPLL Initialization

In order to attain clock alignment between the FPGA fabric and the FDDR subsystem, it is necessary to use the FPLL to perform de-skewing of the FDDR clocks. After the FPLL is initialized, it typically takes over 500 divided reference clock cycles for lock to be achieved. The FPLL lock assertion time is also dependent on the FPLL lock parameters (lock window and lock delay). There is no provision made for operation of the FDDR subsystem before FPLL lock is achieved.

#### 4.6.3.1.4 PLL Lock Monitoring

The FDDR has an input, CLK\_BASE\_PLL\_LOCK, to monitor the fabric PLL lock. It must be connected to the lock signal generated by the fabric PLL which is being used to generate the base clock to the FDDR.

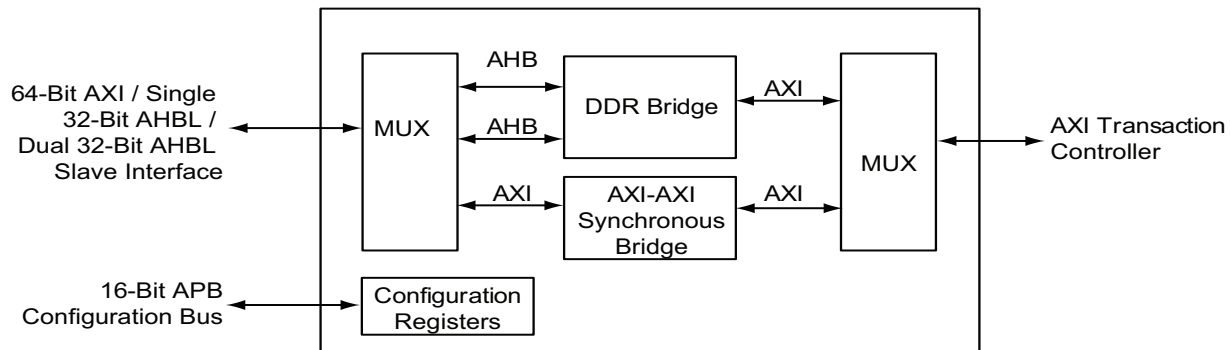
Within the FDDR subsystem, there are two interrupts related to the PLL lock. A lock interrupt, indicating FPLL lock achieved, and an FPLL lock lost interrupt. Each of these two interrupts has a corresponding interrupt enable bit in the FDDR subsystem registers. It is also possible to read the state of the two PLL lock signals through the FDDR registers.

In the event of loss of FPLL lock, even though its output is not exactly in phase lock with the reference, the FPLL still generates a clock. User logic in the FPGA fabric can use the FPLL\_LOCK signal to prevent communication with the FDDR subsystem during this time.

#### 4.6.3.2 DDR\_FIC

The following illustration shows the DDR\_FIC block diagram.

**Figure 77 • DDR\_FIC Block Diagram**



Fabric masters can access the FDDR subsystem in the following ways:

- Single AXI-64 interface
- Single AHB-32 interface
- Dual AHB-32 bit interfaces

If the AXI-64 interface is selected, the DDR\_FIC acts as an AXI to AXI synchronous bridge and also supports locked transactions. During locked transactions a user configurable 20-bit down counter keeps track of the duration of the locked transfer. If the transfer is not completed before the down counter reaches zero, a single clock cycle pulse interrupt is generated to the fabric interface.

If single or dual AHB-32 interfaces are selected, the DDR\_FIC converts the single or dual 32-bit AHBL master transactions from the FPGA fabric to 64-bit AXI transactions. The DDR bridge, which is embedded as part of the DDR\_FIC, is enabled in this case. The DDR bridge has an arbiter that uses a round robin priority scheme on read and write requests from the two AHB masters. Refer to the "[DDR Bridge](#)" chapter on page 206 for a detailed description.

The DDR\_FIC input interface is clocked by the FPGA fabric clock and the AXI transaction controller is clocked by FDDR\_CLK from the FDDR clock controller. Clock ratios between FDDR\_CLK and DDR\_FIC clock can vary. Supported ratios are shown in the following table. Clock ratios can be configured through Libero System-on-Chip (SoC) software or through the FDDR\_FACC\_DIVISOR\_RATIO register.

**Table 131 • FDDR\_CLK to FPGA Fabric Clock Ratios**

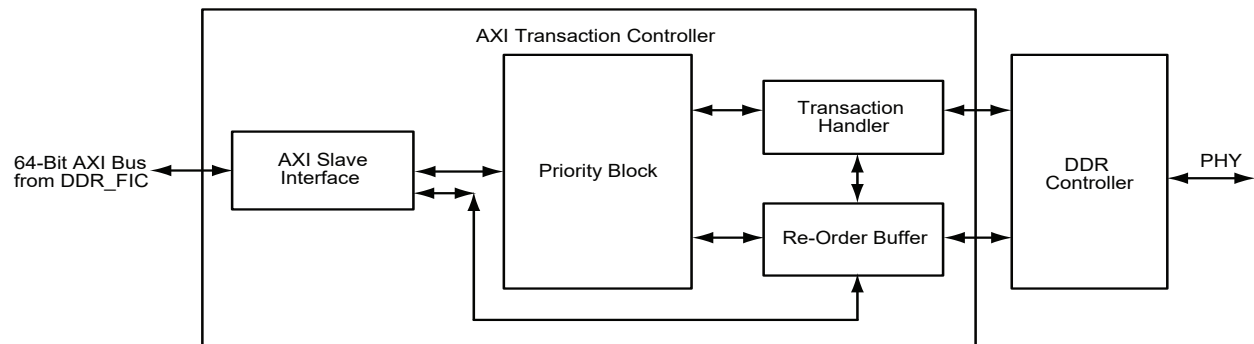
DIVISOR_A[1:0]	DDR_FIC DIVISOR[2:0]	FDDR_CLK: FPGA FABRIC Clock Ratio
00	000	1:1
00	001	2:1
00	010	4:1
00	100	8:1
00	101	16:1
01	000	2:1
01	001	4:1
01	010	8:1
01	100	16:1
11	000	3:1

**Table 131 • FDDR\_CLK to FPGA Fabric Clock Ratios (continued)**

DIVISOR_A[1:0]	DDR_FIC DIVISOR[2:0]	FDDR_CLK: FPGA FABRIC Clock Ratio
11	001	6:1
11	010	12:1

#### 4.6.3.3 AXI Transaction Controller

The AXI transaction controller receives 64-bit AXI transactions from DDR\_FIC and translates them into DDR controller transactions. The following illustration shows the block diagram of the AXI transaction controller interfaced with the DDR controller.

**Figure 78 • AXI Transaction Controller Block Diagram**

The AXI transaction controller comprises four major blocks:

- AXI slave interface
- Priority block
- Transaction handler
- Reorder buffer

##### 4.6.3.3.1 AXI Slave Interfaces

The AXI transaction controller has a 64-bit AXI slave interface from DDR\_FIC. The AXI slave port is 64 bits wide and is in compliance with the standard AXI protocol. Each transaction has an ID related to the master interface. Transactions with the same ID are completed in order, while the transactions with different read IDs can be completed in any order, depending on when the instruction is executed by the DDR controller. If a master requires ordering between the transactions, the same ID should be used.

The AXI slave interface has individual read and write ports. The read port queues read AXI transactions and it can hold up to four read transactions. The write port handles only one write transaction at a time and generates the handshaking signals on the AXI interface.

##### 4.6.3.3.2 Priority Block

The priority block prioritizes AXI read/write transactions and provides control to the transaction handler. AXI read transactions have higher priority. The fabric master through DDR\_FIC can be programmed to have a higher priority by configuring the `PRIORITY_ID` and `PRIORITY_ENABLE_BIT` bit fields in the `DDRC_AXI_FABRIC_PRI_ID_CR` register, [Table 76](#), page 89.

##### 4.6.3.3.3 Transaction Handler

The transaction handler converts AXI transactions into DDR controller commands. The transaction handler works on one transaction at a time from the read/write port queue that is selected by the priority block. The transaction handler has a write command controller and read command controller for write and read transactions.

The write command controller fetches the command from the AXI slave write port and sends a pure write instruction to the DDR controller. If `SECDED` is enabled, a read modified write (RMW) instruction is sent to the DDR controller. The read command controller generates read transactions to the DDR controller.

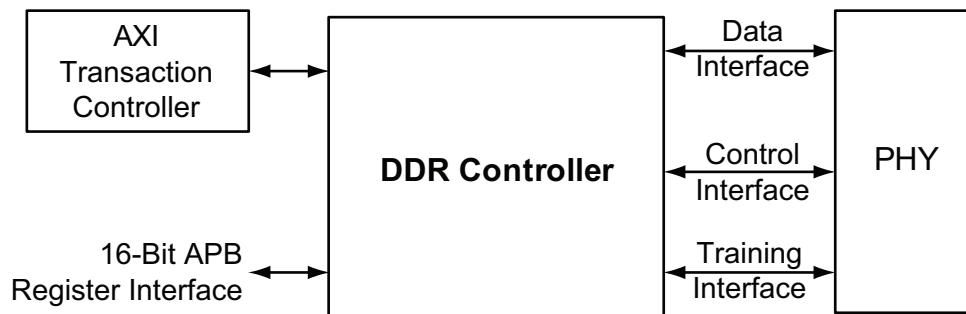
#### 4.6.3.3.4 Reorder Buffer

The reorder buffer receives data from the DDR controller and orders the data as requested by the AXI master when a single AXI transaction is split into multiple DDR controller transactions, depending on the transfer size.

#### 4.6.3.4 DDR Controller

The DDR controller receives requests from the AXI transaction controller, performs the address mapping from system addresses to DRAM addresses (rank, bank, row, and column) and prioritizes requests to minimize the latency of reads (especially high priority reads) and maximize page hits. It also ensures that DRAM is properly initialized, all requests are made to DRAM legally (accounting for associated DRAM constraints), refreshes are inserted as required, and the DRAM enters and exits various power-saving modes appropriately. The following illustration shows the DDR controller connections in the FDDR subsystem.

**Figure 79 • DDR Controller Block Diagram**



The following sections describe key functions of the DDR controller.

##### 4.6.3.4.1 Address Mapping

Read and write requests to the DDR controller requires a system address. The controller is responsible for mapping this system address with rank, bank, row, and column address to DRAM.

The address mapper maps linear request addresses to DDR memory addresses by selecting the source bit that maps to each and every applicable DDR memory address bit. The address map interface registers can be configured to map source address bits to DRAM address (for more information, refer to [Address Mapping](#), page 154 on configuring the FDDR features).

##### 4.6.3.4.2 Transaction Scheduling

The DDR controller schedules the read and write transactions to DDR memory. The DDR controller classifies the transactions into three types, based on the commands from the AXI transaction controller:

- Low priority reads (LPR)
- High priority reads (HPR)
- Writes (WR)

Each type of transaction has a queue and the queued transactions can be in normal state or in critical state. The transactions in a queue moves from normal state to critical state when that transaction is not serviced for a count of MAX\_STARVE\_X32 clocks. The MAX\_STARVE\_X32 values for each queue can be configured using the DDR controller performance registers (refer to the ["Performance" section on page 156](#)). The DDR controller completes the critical transactions with high priority.

##### 4.6.3.4.3 Write Combine

The DDR controller combines multiple writes to the same address into a single write to DDR memory. When a new write collides with the queued write, the DDR controller overwrites the data for the queued write with that from the new write and only performs one write transaction. The write combine functionality can be disabled by setting the register bit REG\_DDRC\_DIS\_WC to 1 ([Table 55](#), page 79).

#### 4.6.3.4.4 SECDED

The DDR controller supports built-in SECDED capability for correcting single-bit errors and detecting dual-bit errors. The SECDED feature can be enabled. When SECDED is enabled, the DDR controller adds 8 bits of SECDED data to every 64 bits of data.

When SECDED is enabled, a write operation computes and stores a SECDED code along with the data, and a read operation reads and checks the data against the stored SECDED code.

The SECDED bits are interlaced with the data bits, as shown in the following table.

**Table 132 • SECDED DQ Lines at DDR**

Mode	SECDED Data Pins	
	M2S050/M2GL050 (FG896)	M2S150/M2GL150 (FC1152)
Full bus width mode	FDDR_DQ_ECC[3:0]	FDDR_DQ_ECC[3:0]
Half bus width mode	FDDR_DQ_ECC[1:0]	FDDR_DQ_ECC[1:0]
Quarter bus width mode	—	FDDR_DQ_ECC[0]

When the controller detects a correctable SECDED error, it does the following:

- Generates an interrupt signal which can be monitored by reading the interrupt status register, DDRC\_ECC\_INT\_SR (Table 99, page 101). The FDDR also generates ECCINT interrupt signal, which can be monitored from FPGA fabric.
- Sends the corrected data to the read requested MSS/HPMS and FPGA fabric master as part of the read data.
- Sends the SECDED error information to the DDRC\_LCE\_SYNDROME\_1\_SR register, Table 87, page 95.
- Performs a read-modify-write operation to correct the data present in the DRAM.

When the controller detects an uncorrectable error, it does the following:

- Generates an interrupt signal that can be monitored by reading the interrupt status register DDRC\_ECC\_INT\_SR, Table 99, page 101. The FDDR also generates an ECC\_INT interrupt signal, which can be monitored from FPGA fabric.
- Sends the data with error to the read requested MSS/HPMS and FPGA fabric master as part of the read data.
- Sends the SECDED error information to the DDRC\_LUE\_SYNDROME\_1\_SR register, Table 80, page 91.

The following SECDED Registers in Table 27, page 62 can be monitored for identifying the exact location of an error in the DDR SDRAM.

- DDRC\_LUE\_ADDRESS\_1\_SR and DDRC\_LUE\_ADDRESS\_2\_SR gives the row/bank/column information of the SECDED unrecoverable error.
- DDRC\_LCE\_ADDRESS\_1\_SR and DDRC\_LCE\_ADDRESS\_2\_SR gives the row/bank/column information of the SECDED error correction.
- DDRC\_LCB\_NUMBER\_SR indicates the location of the bit that caused the single-bit error in the SECDED case (encoded value).
- DDRC\_ECC\_INT\_SR indicates whether the SECDED interrupt is because of a single-bit error or double-bit error. The interrupt can be cleared by writing zeros to DDRC\_ECC\_INT\_CLR\_REG, Table 100, page 101.

#### 4.6.3.4.5 Power Saving Modes

The DDR controller can operate DDR memories in three power saving modes:

- Precharge power-down (DDR2, DDR3, LPDDR1)

If power-down is enabled in the System Builder FDDR configuration or

REG\_DDRC\_POWERDOWN\_EN = 1 (Table 31, page 67), the DDR controller automatically keeps DDR memory in Precharge power-down mode when the period specified by the power down entry time or

REG\_DDRC\_POWERDOWN\_TO\_X32 register (Table 58, page 81) has passed, while the controller is idle (except for issuing refreshes).

The controller automatically performs the precharge power-down exit on any of the following conditions:

- A refresh cycle is required to any rank in the system.
- The controller receives a new request from the core logic.
- REG\_DDRC\_POWERDOWN\_EN is set to 0.
- Self refresh (DDR2, DDR3, LPDDR1)

The DDR controller keeps the DDR memory devices in Self-refresh mode whenever the REG\_DDRC\_SELFREF\_EN register bit (Table 29, page 66) is set and no reads or writes are pending in the controller.

The DDR controller can be programmed to issue single refreshes at a time (REG\_DDRC\_REFRESH\_BURST = 0, see Table 30, page 66) to minimize the worst-case impact of a forced refresh cycle. It can be programmed to burst the maximum number of refreshes allowed for DDR (REFRESH\_BURST = 7, for performing 8 refreshes at a time) to minimize the bandwidth lost when refreshing the pages.

The controller takes the DDR memory out of Self-refresh mode whenever the REG\_DDRC\_SELFREF\_EN input is deasserted or new commands are received by the controller.

When the DDR self-refresh is enabled, the DDR I/O bank may go into recalibration and a glitch may occur in the MDDR bank I/Os, which are being used for general purpose rather than for the DDR memory. The DDR I/Os ODT is periodically calibrated and will be effected only when the I/Os are in tri-state (DDR I/Os are tri-stated only in self-refresh mode).

- Deep power-down (LPDDR1)

This is supported only for LPDDR1. The DDR controller puts the DDR SDRAM devices in Deep Power-down mode whenever the REG\_DDRC\_DEEPPOWERDOWN\_EN bit (Table 31, page 67) is set and no reads or writes are pending in the DDR controller.

The DDR controller automatically exits Deep Power-down mode and reruns the initialization sequence when the REG\_DDRC\_DEEPPOWERDOWN\_EN bit is reset to 0. The contents of DDR memory may be lost upon entry into Deep Power-down mode.

#### 4.6.3.4.6 DRAM Initialization

After Reset, the DDR controller initializes DDR memories through an initialization sequence, depending on the type of DDR memory used. For more information on the initialization process, refer to the JEDEC specification.

### 4.6.4 FDDR Subsystem Features Configuration

The FDDR subsystem registers must be initialized before accessing DDR memory through the FDDR subsystem. When using the System Builder flow through Libero SoC all of the necessary registers are initialized automatically by the resulting module. This section provides the registers features of the FDDR. All registers are listed with their bit definitions in the "FDDR Configuration Registers" section on page 174 section.

### 4.6.5 Memory Type

DDRC\_MODE\_CR (Table 32, page 67) must be configured to select the memory type (DDR2, DDR3, or LPDDR1) to access memory from the FDDR subsystem.

### 4.6.6 Bus Width Configurations

The FDDR supports various bus widths, as listed in the following table. The FDDR can be programmed to work in full, half, or quarter Bus width mode by configuring the DDRC\_MODE\_CR (Table 32, page 67) and PHY\_DATA\_SLICE\_IN\_USE\_CR registers (Table 101, page 102) when the controller is in soft reset.

**Table 133 • Supported Bus Widths**

Bus Width	M2S050/M2GL050 (FG896)	M2S150/M2GL150 (FC1152)
Full bus width	√	√
Half bus width	√	√
Quarter bus width	—	√

### 4.6.7 Burst Mode

The DDR controller performs burst write operations to DDR memory, depending on the Burst mode selection. Burst mode is selected as sequential or interleaving by configuring REG\_DDRC\_BURST\_MODE to 1 or 0 (Table 70, page 86).

Burst length can be selected as 4, 8, or 16 by configuring REG\_DDRC\_BURST\_RDWR (Table 64, page 83).

Supported burst modes for DDR SDRAM types and PHY widths are given in the following table. For M2GL050, only sequential Burst mode and a burst length of 8 is supported.

**Table 134 • Supported Burst Modes for M2S150 and M2GL150**

Bus Width	Memory Type	Sequential/Interleaving		
		4	8	16
32	LPDDR1	√	√	—
	DDR2	√	√	—
	DDR3	—	√	—
16	LPDDR1	—	√	√
	DDR2	—	√	—
	DDR3	—	√	—
8	LPDDR1	—	√	—
	DDR3	—	√	—
	DDR2	—	√	—

### 4.6.8 Configuring Dynamic DRAM Constraints

Timing parameters for DDR memories must be configured according to the DDR memory specification. Dynamic DRAM constraints are subdivided into three basic categories:

- Bank constraints affect the transactions that are scheduled to a given bank
- Rank constraints affect the transactions that are scheduled to a given rank
- Global constraints affect all transactions

### 4.6.9 Dynamic DRAM Bank Constraints

The timing constraints which affect the transactions to a bank are listed in the following table. The control bit field must be configured as per the DDR memory vendor specification.

**Table 135 • Dynamically Enforced Bank Constraints**

Timing Constraint of DDR Memory	Control Bit	Description
---------------------------------	-------------	-------------



**Table 135 • Dynamically Enforced Bank Constraints**

Row cycle time ( $t_{RC}$ )	REG_DDRC_T_RC, <a href="#">Table 45</a> , page 74	Minimum time between two successive activates to a given bank.
Row precharge command period ( $t_{RP}$ )	REG_DDRC_T_RP, <a href="#">Table 52</a> , page 77	Minimum time from a precharge command to the next command affecting that bank.
Minimum bank active time ( $t_{RAS(min)}$ )	REG_DDRC_T_RAS_MIN, <a href="#">Table 49</a> , page 76	Minimum time from an activate command to a precharge command to the same bank.
Maximum bank active time ( $t_{RAS(max)}$ )	REG_DDRC_T_RAS_MAX, <a href="#">Table 49</a> , page 76	Maximum time from an activate command to a precharge command to the same bank.
RAS-to-CAS delay ( $t_{RCD}$ )	REG_DDRC_T_RCD, <a href="#">Table 52</a> , page 77	Minimum time from an activate command to a Read or Write command to the same bank.
Write command period ( $t_{WR}$ )	REG_DDRC_WR2PRE, <a href="#">Table 47</a> , page 75	Minimum time from a Write command to a precharge command to the same bank.
Read-to-precharge delay ( $t_{RTP}$ )	REG_DDRC_RD2PRE, <a href="#">Table 47</a> , page 75	Minimum time from a Read command to a precharge command to the same bank. Set this to the current value of additive latency plus half of the burst length.

#### 4.6.9.1 Dynamic DRAM Rank Constraints

The timing constraints which affect the transactions to a rank are listed in the following table. The control bit field must be configured as per the DDR memory vendor specification.

**Table 136 • Dynamically Enforced Bank Constraints**

Timing Constraints of DDR Memory	Control Bit	Description
Nominal refresh cycle time ( $t_{RFC(nom)}$ or $t_{REFI}$ )	REG_DDRC_T_RFC_NOM_X32, <a href="#">Table 30</a> , page 66	Average time between refreshes for a given rank. The actual time between any two refresh commands may be larger or smaller than this; this represents the maximum time allowed between refresh commands to a given rank when averaged over a large period of time.
Minimum refresh cycle time $t_{RFC(min)}$	REG_DDRC_T_RFC_MIN, <a href="#">Table 29</a> , page 66	Minimum time from refresh to refresh or activate.
RAS-to-RAS delay ( $t_{RRD}$ )	REG_DDRC_T_RRD, <a href="#">Table 52</a> , page 77	Minimum time between activates from bank A to bank B.
RAS-to-CAS delay ( $t_{CCD}$ )	REG_DDRC_T_CCD, <a href="#">Table 52</a> , page 77	Minimum time between two reads or two writes (from bank A to bank B).
Four active window ( $t_{FAW}$ )	REG_DDRC_T_FAW, <a href="#">Table 45</a> , page 74	Sliding time window in which a maximum of 4 bank activates are allowed in an 8-bank design. In a 4-bank design, set this register to 0x1.

#### 4.6.9.2 Dynamic DRAM Global Constraints

The timing constraints which affect global transactions are listed in the following table. The control bit field must be configured as per the DDR memory vendor specification.

**Table 137 • Dynamic DRAM Global Constraints**

Timing Constraint	Control Bit	Description
Read-to-write turnaround time	REG_DDRC_RD2WR, <a href="#">Table 50</a> , page 76	Minimum time to allow between issuing any Read command and issuing any WRITE command



**Table 137 • Dynamic DRAM Global Constraints**

Write-to-read turnaround time	REG_DDRC_WR2RD, Table 50, page 76	Minimum time to allow between issuing any Write command and issuing any Read command
Write latency	REG_DDRC_WRITE_LATENCY, Table 46, page 75	Time after a Write command that write data should be driven to DRAM.

The DDR memories require delays after initializing the mode registers. The following registers must be configured for delay requirements for the DDR memories. The DDR controller uses these delay values while initializing the DDR memories.

- DDRC\_CKE\_RSTN\_CYCLES\_1\_CR (Table 39, page 72), recommended value is 0x4242
- DDRC\_CKE\_RSTN\_CYCLES\_2\_CR (Table 40, page 73), recommended value is 0x8

## 4.6.10 Address Mapping

The DDR controller maps linear request addresses to DDR memory addresses by selecting the source bit that maps to each and every applicable DDR memory address bit.

Each DDR memory address bit has an associated register vector to determine its source. The source address bit number is determined by adding the internal base of a given register to the programmed value for that register, as described in the following equation.

$$[\text{Internal base}] + [\text{register value}] = [\text{source address bit number}]$$

EQ 2

For example, reading the description for REG\_DDRC\_ADDRMAP\_COL\_B3, the internal base is 3; so when the full data bus is in use, the column bit 4 is determined by 3 + [register value].

If this register is programmed to 2, then the source address bit is: 3 + 2 = 5.

The DDR configurator assigns values to the address mapping registers depending on the selected number of Columns, Rows and Banks. The following illustration provides the default mapping of the memory row, bank, and column address to the user interface address domain.

**Figure 80 • Address Mapping**

Full bus width mode																																
AXI/AHB Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row mapping (DDR2/DDR3)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
Row mapping (LPDDR)			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
Bank mapping(DDR2/DDR3)																		2	1	0												
Bank mapping(LPDDR)																			1	0												
column mapping																					9	8	7	6	5	4	3	2	1	0		
Half bus width mode																																
AXI/AHB Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row mapping(DDR2/DDR3)			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
Row mapping (LPDDR)				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
Bank mapping(DDR2/DDR3)																			2	1	0											
Bank mapping(LPDDR)																				1	0											
column mapping																						9	8	7	6	5	4	3	2	1	0	
Quarter bus width mode																																
AXI/AHB Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row mapping(DDR2/DDR3)				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
Row mapping (LPDDR)					15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
Bank mapping(DDR2/DDR3)																				2	1	0										
Bank mapping(LPDDR)																					1	0										
column mapping																							9	8	7	6	5	4	3	2	1	0

The following are the address mapping registers:

- DDRC\_ADDR\_MAP\_BANK\_CR, [Table 33](#), page 68
- DDRC\_ADDR\_MAP\_COL\_1\_CR, [Table 34](#), page 69
- DDRC\_ADDR\_MAP\_COL\_2\_CR, [Table 35](#), page 69
- DDRC\_ADDR\_MAP\_COL\_3\_CR, [Table 55](#), page 79
- DDRC\_ADDR\_MAP\_ROW\_1\_CR, [Table 36](#), page 70
- DDRC\_ADDR\_MAP\_ROW\_2\_CR, [Table 37](#), page 71

While configuring the registers, ensure that two DDR memory address bits are not determined by the same source address bit.

**Note:** Some registers map multiple source address bits (REG\_DDRC\_ADDRMROW\_ROW\_B0\_11)

To arrive at the right address for the DDR controller, the system address or AXI address bits [4:0] are mapped by the FDDR.

- In Full Bus Width mode, the system address bits [4:0] are used to map the lower column address bits (C0, C1, C2).
- In Half Bus Width mode, the system address bits [4:0] are used to map the lower column address bits (C0, C1, C2, C3).
- In quarter bus width mode, the system address bits [4:0] are used to map the lower column address bits (C0, C1, C2, C3, C4).

The FDDR configurator uses {Row, Bank, Column} address mapping as shown in the following example.

#### 4.6.10.0.1 Example

In this example, the Address map registers are configured to access a 512 MB DDR3 SDRAM memory (MT41J512M8RA) from the FDDR subsystem as shown in "[Example 2: Connecting 32-Bit DDR3 to FDDR\\_PADs with SECEDED](#)" section on page 172. The 512M x 8-bit DDR3 memory module has 3 bank address lines, 16 rows, and 10 columns.

- The column address bits 3 to 9 are mapped for system address bit[5] to system address bit[11]. To map the column 3-bit (C3) to address [5], the field is configured to 3, as the base value is 2. Similarly, the other column address bits are configured:
  - DDRC\_ADDR\_MAP\_COL\_1\_CR = 0x3333
  - DDRC\_ADDR\_MAP\_COL\_2\_CR = 0x3FFF
  - DDRC\_ADDR\_MAP\_COL\_3\_CR = 0x3300
- The bank address bits 0 to 2 are mapped for system address bit[12] to system address bit[14]. To map the bank bit0 to address [12], the field is configured to A, as the base value is 2. Similarly, the other bank address bits are configured:
  - DDRC\_ADDR\_MAP\_BANK\_CR = 0xAAA
- The row address bits 0 to 15 are mapped for system address bit[15] to system address bit[27]. To map the bank bit0 to address [15], the field is configured to 9, as the base value is 6. Similarly, the other bank address bits are configured:
  - DDRC\_ADDR\_MAP\_ROW\_1\_CR = 0x9999
  - DDRC\_ADDR\_MAP\_ROW\_2\_CR = 0x9FF

**Note:** The FDDR can access the 2 GB address space (0x00000000 - 0x7FFFFFFF). But in this example, 512 MB (0x00000000 - 0x1FFFFFFF) DDR3 SDRAM is connected to the 16 address lines of FDDR. The memory visible in the other memory space is mirrored of this 512 MB memory.

#### 4.6.10.1 DDR Mode Registers

After reset, the DDR controller initializes the mode registers of DDR memory with the values in the following registers. The mode registers must be configured according to the specification of the external DDR memory when the controller is in soft reset.

- DDRC\_INIT\_MR\_CR, [Table 41](#), page 73
- DDRC\_INIT\_EMER\_CR, [Table 42](#), page 74
- DDRC\_INIT\_EMER2\_CR, [Table 43](#), page 74
- DDRC\_INIT\_EMER3\_CR, [Table 44](#), page 74

The T\_MOD and T\_MRD bits in DDRC\_DRAM\_MR\_TIMING\_PARAM\_CR ([Table 48](#), page 75) must be configured to the required delay values. T\_MOD and T\_MRD are delays between loading the mode registers.

#### 4.6.10.2 SECEDED

To enable SECEDED mode, set the REG\_DDRC\_MODE bits to 101 in DDRC\_MODE\_CR, [Table 32](#), page 67. The PHY\_DATA\_SLICE\_IN\_USE\_CR register ([Table 102](#), page 102) must be configured to enable data slice 4 of the PHY.

The register value REG\_DDRC\_LPR\_NUM\_ENTRIES in the performance register, DDRC\_PERF\_PARAM\_1\_CR ([Table 64](#), page 83), must be increased by 1 to the value used in Normal mode (without SECEDED).

#### 4.6.10.3 Read Write Latencies

The read and write latencies between DDR controller and DDR PHY can be configured. Configure the PHY\_DATA\_SLICE\_IN\_USE\_CR register for adding latencies for read and writes ([Table 102](#), page 102).

#### 4.6.10.4 Performance

The DDR controller has several performance registers which can be used to increase the speed of the read and write transactions to DDR memory.

The DDR controller has a transaction store, shared for low and high priority transactions. The DDRC\_PERF\_PARAM\_1\_CR register ([Table 64](#), page 83) can be configured for allocating the transaction store between the low and high priority transactions. For example, if the REG\_DDRC\_LPR\_NUM\_ENTRIES field ([Table 64](#), page 83) is configured to 0, the controller allocates more time to high priority transactions. The ratio for LPR: HPR is 1:7 (as the transaction store depth is 8).

The DDRC\_HPR\_QUEUE\_PARAM\_1\_CR ([Table 65](#), page 85), DDRC\_LPR\_QUEUE\_PARAM\_1\_CR ([Table 67](#), page 85), and DDRC\_WR\_QUEUE\_PARAM\_CR ([Table 69](#), page 86) registers can be configured for the minimum clock values for treating the transactions in the HPR, LPR, and WR queue as critical and non-critical.

To force all incoming transactions to low priority, configure the DDRC\_PERF\_PARAM\_2\_CR register ([Table 70](#), page 86). By default it is configured to force all the incoming transactions to low priority.

#### 4.6.10.5 Refresh Controls

The DDR controller automatically issues refresh commands to DDR memory for every tRFC (min). The DDR controller can be programmed to issue single refreshes at a time (REG\_DDRC\_REFRESH\_BURST = 0) TO MINIMIZE THE WORST-CASE IMPACT OF A FORCED REFRESH CYCLE. It can be programmed to burst the maximum number of refreshes allowed for DDR (REFRESH\_BURST = 7, for performing 8 refreshes at a time) to minimize the bandwidth lost when refreshing the pages.

#### 4.6.10.6 1T or 2T Timing

The DRAM can be used in 1T or 2T Timing mode by configuring the DDRC\_PERF\_PARAM\_3\_CR register ([Table 71](#), page 87). The address bus can be clocked using 1T or 2T clocking. With 1T, the DDR controller can issue a new command on every clock cycle. In 2T timing the DDR controller will hold the address and command bus valid for two clock cycles. This reduces the efficiency of the bus to one command per two clocks, but it doubles the amount of setup and hold time. The data bus remains the same for all of the variations in the address bus, Default configuration is 1T timing mode.

#### 4.6.10.7 ODT Controls

The ODT for a specific rank of memory can be enabled or disabled by configuring the DDRC\_ODT\_PARAM\_1\_CR ([Table 53](#), page 77) and DDRC\_ODT\_PARAM\_2\_CR ([Table 54](#), page 79) registers. These must be configured before taking the controller out of soft reset. They are applied to every read or write issued by the controller.

#### 4.6.10.8 Soft Resets

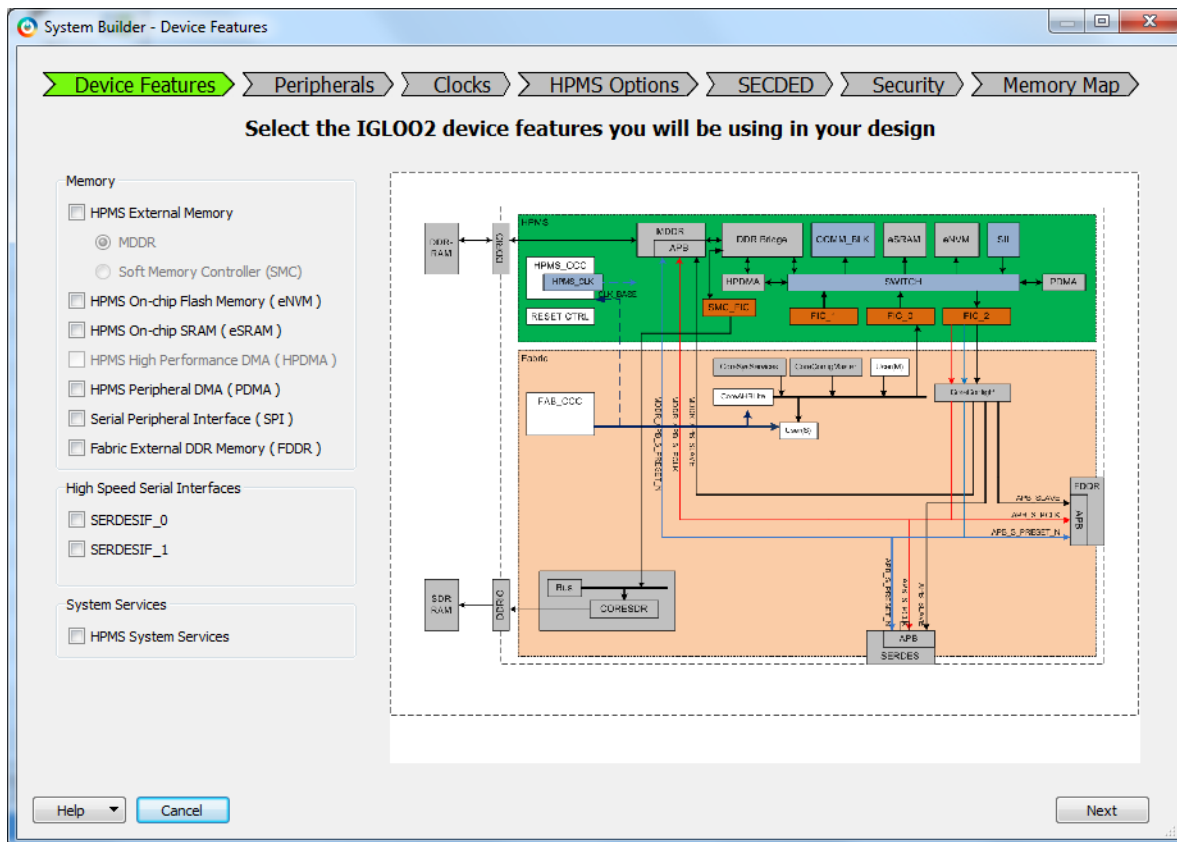
Set the REG\_DDRC\_SOFT\_RSTB bit of DDRC\_DYN\_SOFT\_RESET\_CR ([Table 28](#), page 66) to 0 to reset the DDR controller. To release the DDR controller from reset, set the REG\_DDRC\_SOFT\_RSTB bit of DDRC\_DYN\_SOFT\_RESET\_ALIAS\_CR ([Table 75](#), page 88) to 1.

## 4.7 How to Use FDDR in IGL002 Devices

This section describes how to use FDDR in the IGLOO2 devices. To configure the IGLOO2 device features and then build a complete system, use the **System Builder** graphical design wizard in the Libero Software.

The following image shows the initial **System Builder** window where you can select the features that you require. For details on how to launch the System Builder wizard and a detailed information on how to use it, refer the IGLOO2 System Builder User's Guide. You can also use CoreABC based initialization as described in the IGLOO2 Standalone Peripheral Initialization User Guide.

**Figure 81 • System Builder - Device Features Window**



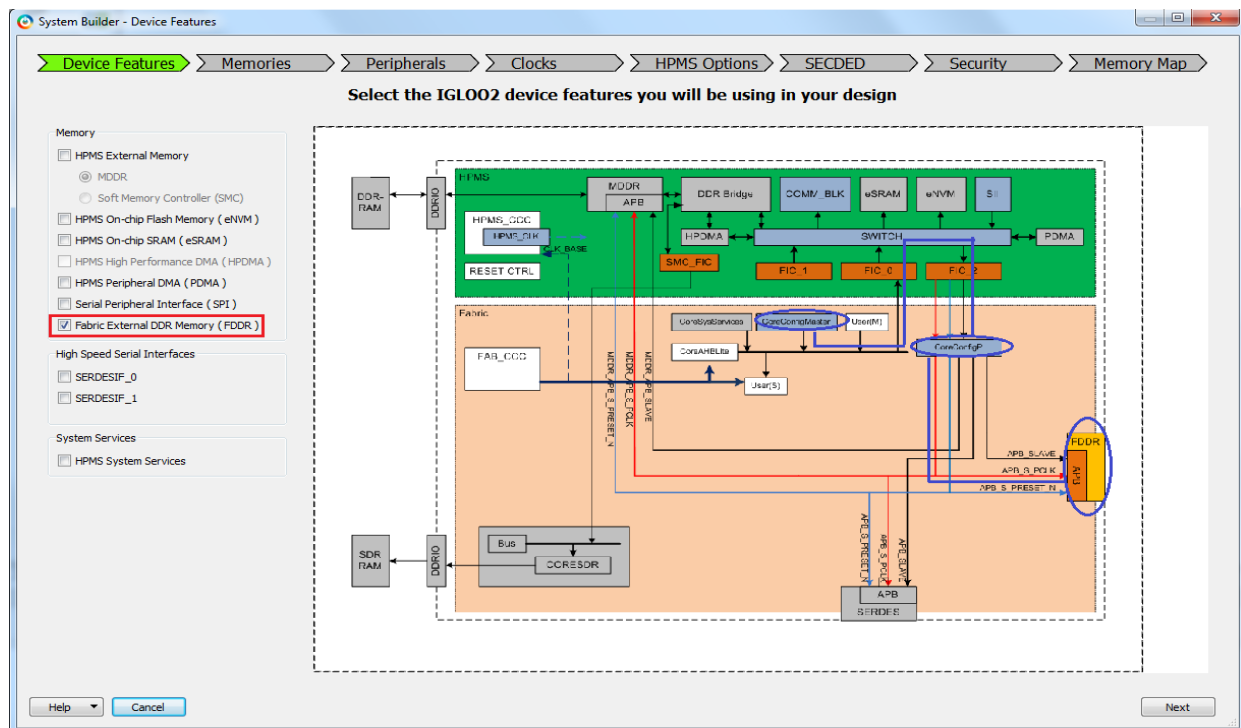
For more information about how to use MDDR in the SmartFusion2 devices, refer to ["Appendix A: How to Use the FDDR in SmartFusion2 Devices"](#) section on page 183.

### 4.7.1 Configuring FDDR

The following steps describe how to configure the FDDR:

1. Check the **Fabric External DDR Memory (FDDR)** check box under the **Device Features** tab and leave the other check boxes unchecked. The following image shows the **System Builder - Device Features** tab.

**Figure 82 • System Builder - Device Features Tab**



2. Selecting the **Fabric External DDR Memory (FDDR)** check box in the **System Builder** performs the following actions:
  - Instantiates the required IPs like CoreConfigMaster, CoreConfigP that initializes the FDDR Controller.
  - Establishes the initialization path:  
CoreConfigMaster → FIC\_0 → eNVM → FIC\_2 → CoreConfigP → APB bus of the FDDR subsystem.
  - CoreConfigMaster (AHB Master) accesses the DDR configuration data stored in eNVM through FIC\_0
  - The configuration data is sent to CoreConfigIP through the FIC\_2 master port
  - CoreConfigP sends the configuration data to APB bus of the FDDR subsystem
3. Navigate to the **Memories** tab. Depending on the application requirement, select the memory settings under the **General** tab as shown in the following image.
  - Memory Type can be selected as DDR2, DDR3 or LPDDR.
  - The Data width can be selected as 32-bit, 16-bit, or 8-bit. Refer to [Table 133](#), page 152 for supported data widths for various IGLOO2 device packages.
  - The SECDED (ECC) can be enabled or disabled.
  - Address Mapping - The register settings to perform mapping to system address bits for various Row, Bank and Column combinations are automatically computed by the configurator using address mapping option. The following table shows the supported range for Row, Bank and Column.

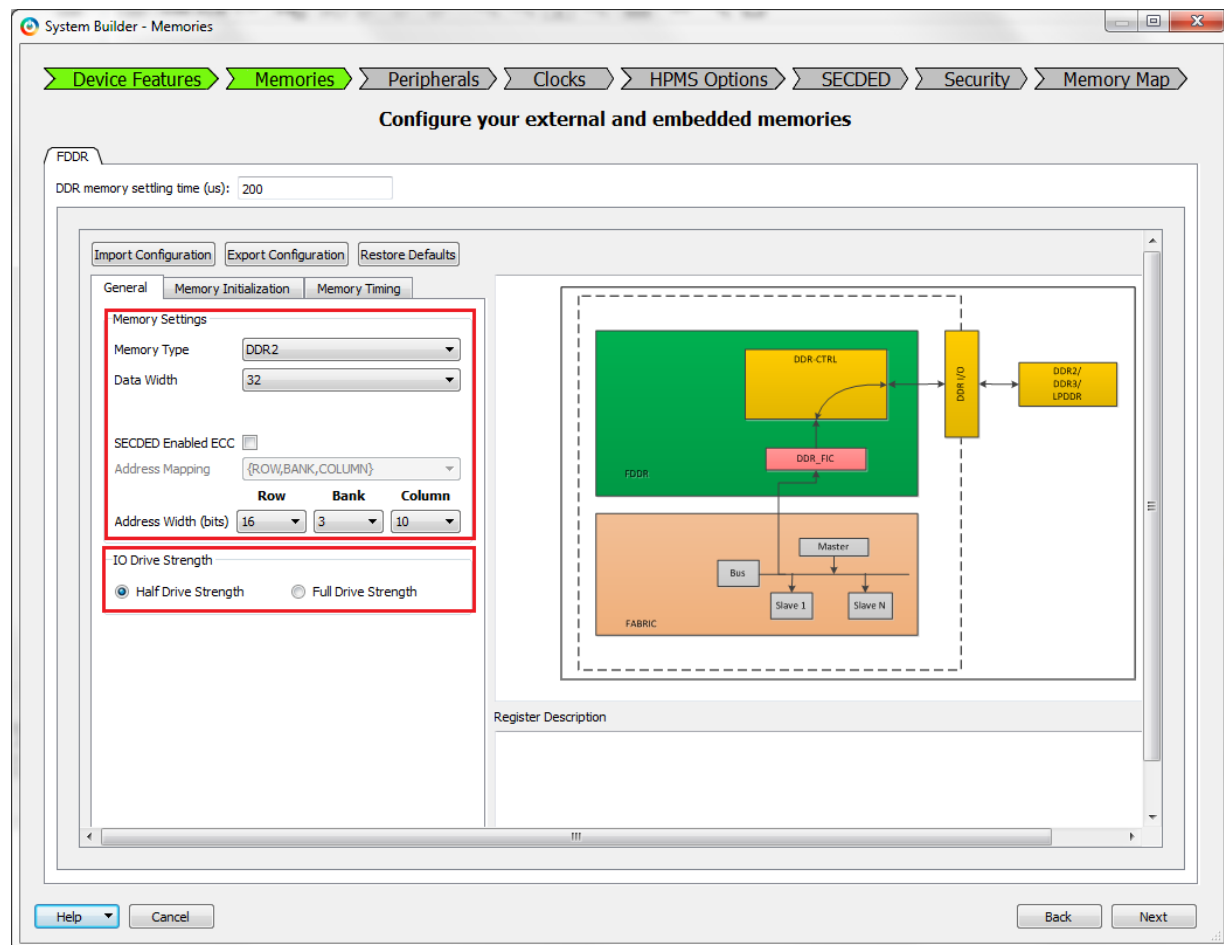
**Table 138 • Supported Address Width Range for Row, Bank and Column**

Width	<b>DDR2</b>	<b>DDR3</b>	<b>LPDDR</b>
Row Address	12-16	12-16	12-16
Bank Address	2-3	2-3	2-3
Column Address	9-12	9-12	9-12

- For more information, refer to the ["Address Mapping" section](#).
- Select the **I/O Drive Strength** as **Half Drive Strength** or **Full Drive Strength**, as shown in the following table. The DDR I/O standard is configured as listed in the following table based on this setting.

**Table 139 • DDR I/O Standard is Configured based on I/O Drive Strength Setting**

I/O Drive Strength	Memory Type	
	DDR2	DDR3
Half Drive Strength	SSTL18I	SSTL15I
Full Drive Strength	SSTL18II	SSTL15II

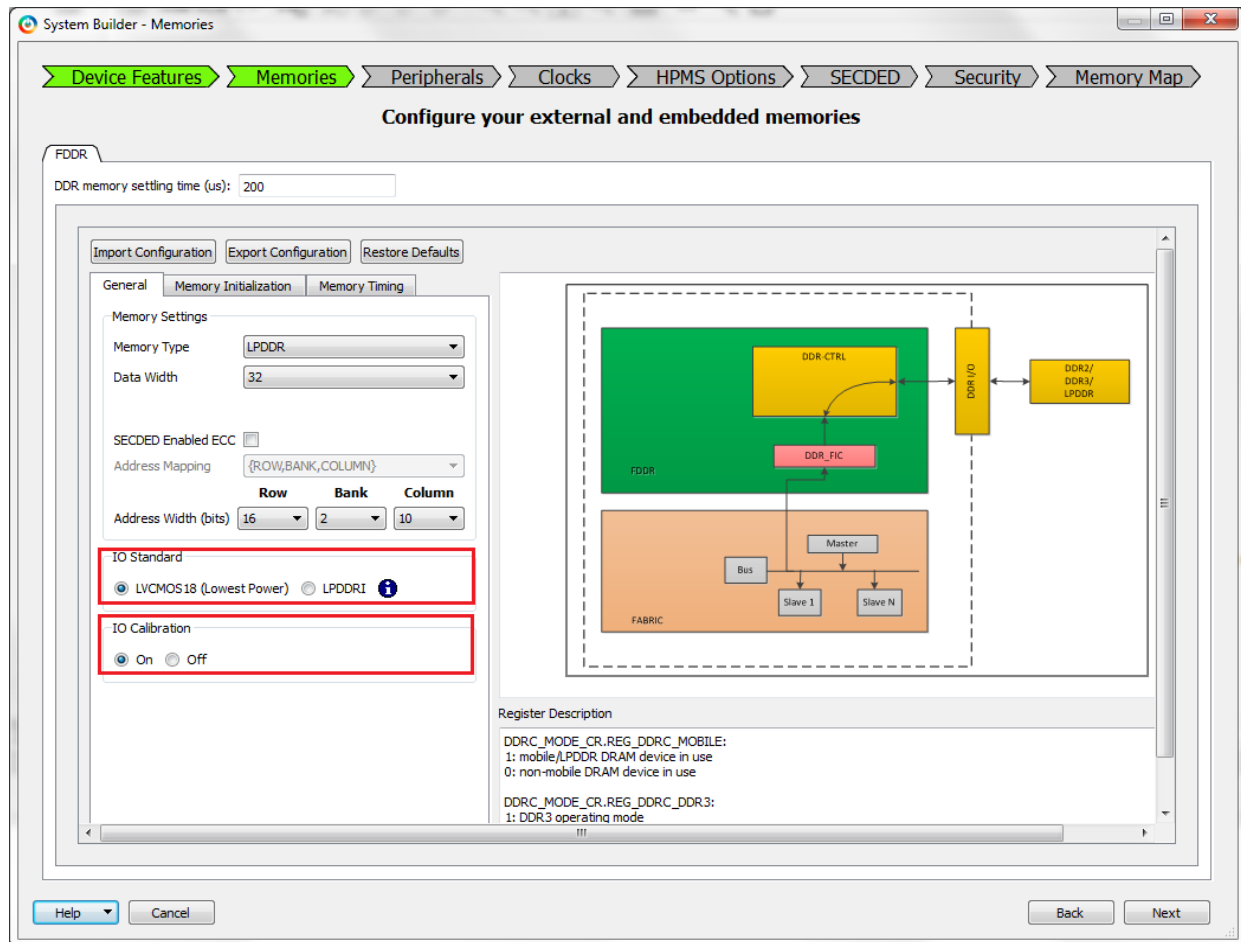
**Figure 83 • Fabric DDR Memory Configuration**


- For only LPDDR memory, the **I/O standard** and **I/O calibration** settings are available as shown in the following image.

- Select **I/O standard** as **LVC MOS18** or **LPDDR1**.

**Note:** If LVC MOS18 is selected, all IOs are configured to LVC MOS1.8 except CLK/CLK\_N.CLK and CLK\_N are configured to LPDDR1 standard as they are differential signals.

- Select **I/O calibration** as ON or OFF. If I/O calibration is selected as ON, then the IGLOO2 FDDR\_IMP\_CALIB pin must be pulled down with a resistor. For resistor values, refer to Impedance Calibration section in [DS0124: IGLOO2 Pin Descriptions Datasheet](#).

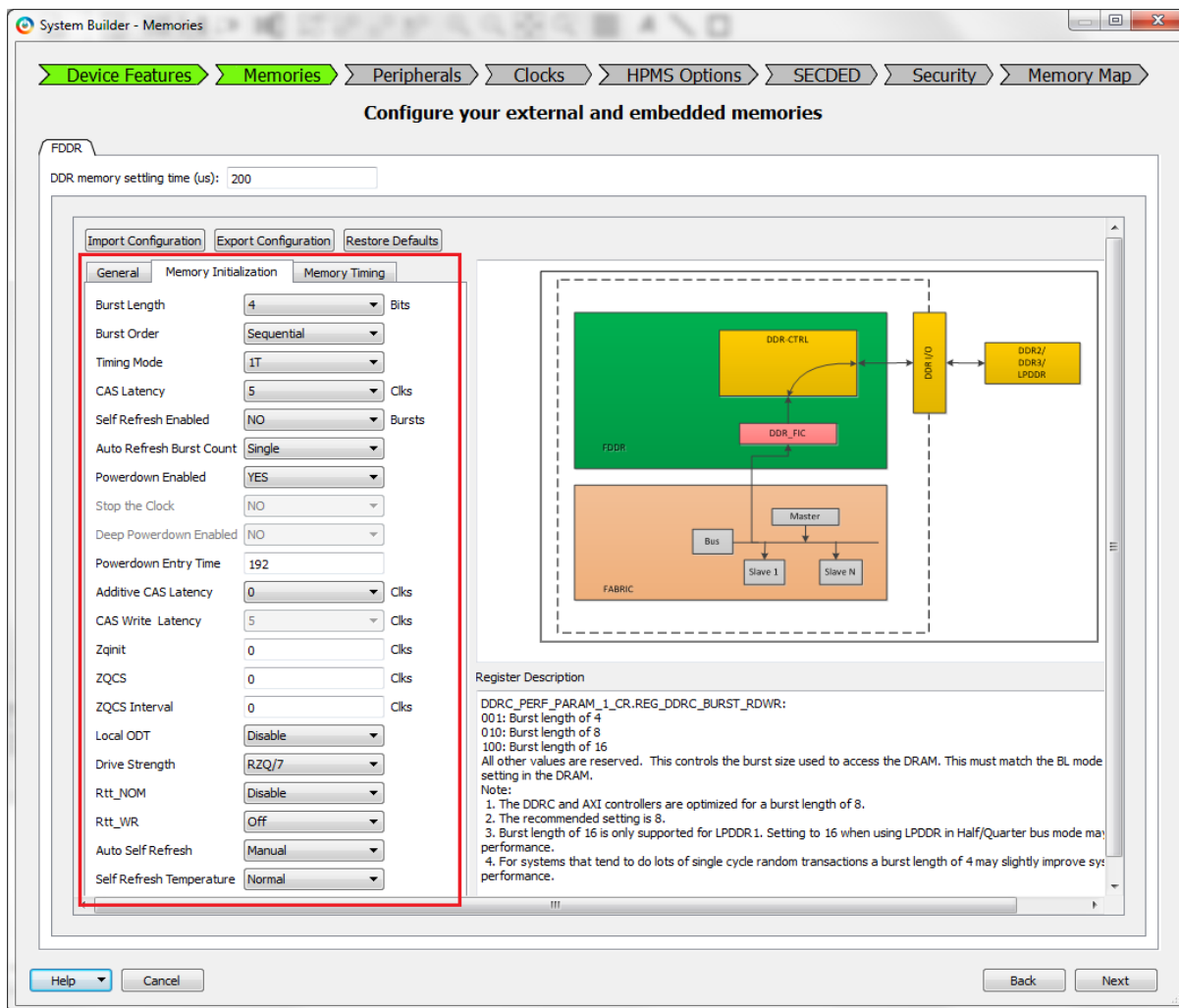
**Figure 84 • Selecting I/O Standard as LVCMOS18 or LPDDR1**

5. Depending on the application requirement, select the Memory Initialization settings under the Memory Initialization tab as shown in [Figure 85 on page 162](#).
- Select the below performance related settings
    - Burst Length can be selected as 4, 8, or 16. Refer [Table 134 on page 152](#) for supported burst lengths.
    - Burst order can be selected as sequential or interleaved. Refer [Table 134 on page 152](#) for supported burst orders.
    - Timing mode can be selected as 1T or 2T. For more details refer to "1T or 2T Timing" section on [page 156](#).
    - CAS latency is the delay, in clock cycles, between the internal READ command and the availability of the first bit of output data. Select the CAS latency according to the DDR memory (Mode register) datasheet.
  - Select the below power saving mode settings. Refer to "Power Saving Modes" section on [page 150](#) for more details.
    - Self-Refresh Enabled
    - Auto Refresh Burst Count
    - Power down Enabled
    - Stop the clock: supported only for LPDDR
    - Deep Power down Enabled: supported only for LPDDR
    - Power down entry time
  - Select the additional performance settings.
    - Additive CAS Latency is defined by EMR[5:3] register of DDR2 memory and by MR1[4:3] register of DDR3 memory. It enables the DDR2 or DDR3 SDRAM to allow a READ or WRITE command from DDR Controller after the ACTIVATE command for the same bank prior to tRCD (MIN). This configuration is part of DDR2 Extended Mode Register and DDR3 Mode Register1.

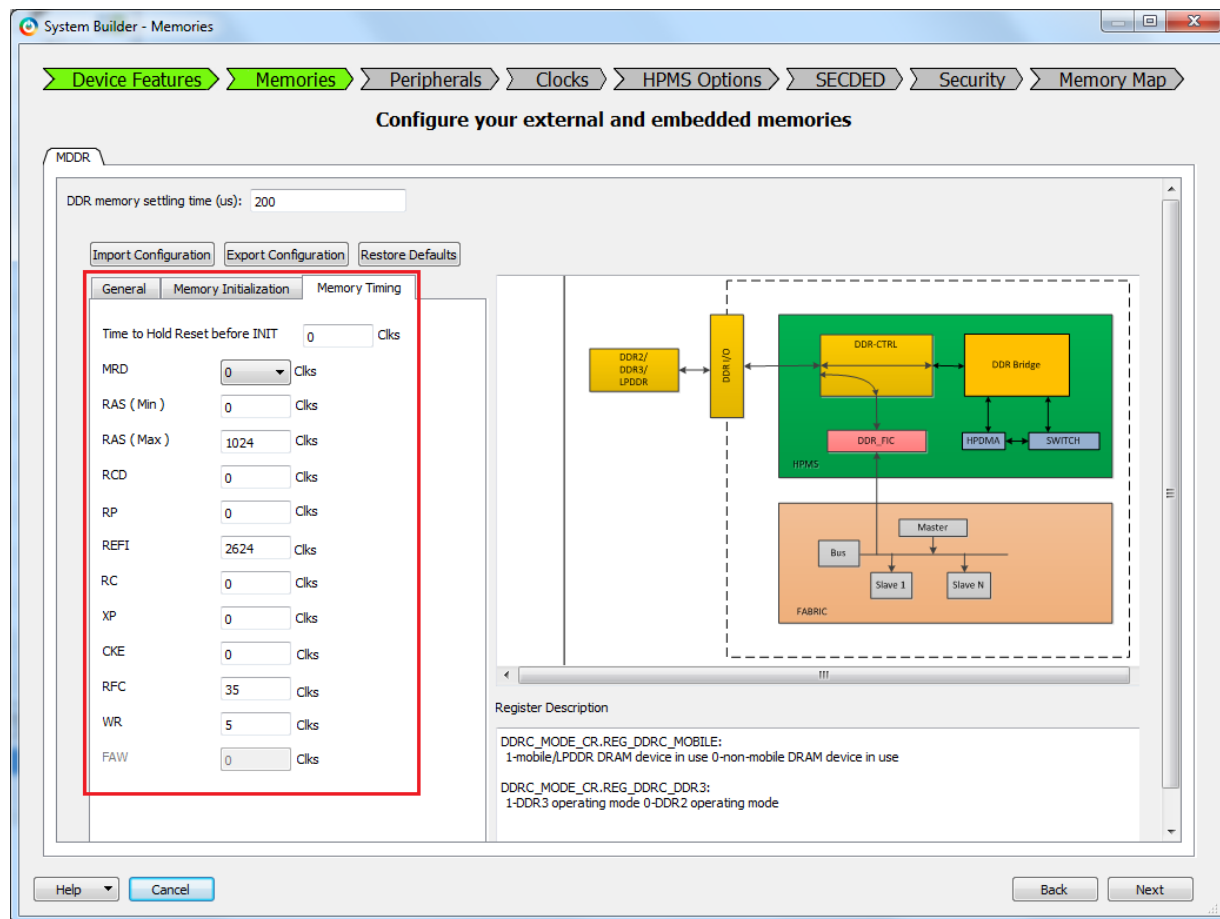


- CAS Write Latency (CWL) is defined by DDR3 MR2[5:3] and is the delay, in clock cycles, from the releasing of the internal write to the latching of the first data in. The overall WRITE latency (WL) is equal to CWL + AL, where CWL is set to 5 clock cycles by default.
- Select the below ZQ Calibration settings for DDR3 memory. For more details refer ["ZQ Calibration" section on page 144](#).
  - Zqinit
  - ZQCS
  - ZQCS Interval
- Select other settings.
  - Local ODT setting is not supported for LPDDR memory. For DDR2/DDR3 memory type, user can choose any option for "Local ODT". User can enable or disable "LOCAL ODT" during read transaction.
  - Drive strength setting is defined by EMR[7:5] register bits of LPDDR memory with drop down options of 'Full', 'Half', 'Quarter' and 'One-eighth' drive strength, it is defined by EMR[1] register bit of DDR2 memory with drop down options of 'Full' and 'Weak' drive strength and it is defined by MR1 register bits M5 and M1 of DDR3 memory with drop down options of 'RZQ/6' and 'RZQ/7'.
  - Partial array self-refresh coverage setting is defined by EMR[2:0] register bits of LPDDR memory with drop down options of 'Full', 'Quarter', 'One-eighth' and 'One-sixteenth'. This feature helps in improving power savings during self-refresh by selecting the amount of memory to be refreshed during self-refresh.
  - $R_{TT}$  (Nominal) setting is defined by EMR[6] and EMR[2] register bits of DDR2 memory which determines what ODT resistance is enabled with drop down options of 'RTT disabled', '50 ohms', '75  $\Omega$ ' and '150  $\Omega$ ' and it is defined by MR1[9], MR1[6] and MR1[2] register bits of DDR3 memory. In DDR3 memory  $R_{TT}$  nominal termination is allowed during standby conditions and WRITE operations and NOT during READ operations with drop down options of 'RZQ/2', 'RZQ/4' and 'RZQ/6'.
  - RTT\_WR (Dynamic ODT) setting is defined by MR2[10:9] register bits of DDR3 memory. This is applicable only during WRITE operations. If dynamic ODT (Rtt\_WR) is enabled, DRAM switches from normal ODT ( $R_{TT\_nom}$ ) to dynamic ODT (Rtt\_WR) when beginning WRITE burst and subsequently switches back to normal ODT at the end of WRITE burst. The drop down options provided to the user are 'off', 'RZQ/4' and 'RZQ/2'.
  - Auto self-refresh setting is defined by MR2[6] register bit of DDR3 memory with drop down option of 'Manual' and 'Auto'. Self-refresh temperature setting is defined by MR2[7] register bit of DDR2 memory with drop down options of 'Normal' and 'Extended'.



**Figure 85 • Memory Initialization Configuration**

6. Select the memory timing settings under the **Memory Timing** tab according to the DDR memory vendor data sheet as shown in the following image. For more details refer to "Configuring Dynamic DRAM Constraints" section on page 152 section.

**Figure 86 • Memory Timing Configuration**

The configurator also provides the option to import and export the register configurations. The configuration settings are stored in eNVM. Configuration files for accessing LPDDR memory on IGLOO2 Evaluation kit can be downloaded from:

[www.microsemi.com/soc/documents/LPDDR\\_Emcrafft\\_Config.zip](http://www.microsemi.com/soc/documents/LPDDR_Emcrafft_Config.zip).

An example of FDDR register configurations for operating the LPDDR memory (MT46H64M16LF) with clock 166 MHz is given below.

- Device Memory Settling Time (us): 200

The DDR memories require settling time for the memory to initialize before accessing it. the LPDDR memory model MT46H64M16LF needs 200us settling time.

#### General

- Memory Type: Select LPDDR
- Data Width: 16

#### Memory Initialization

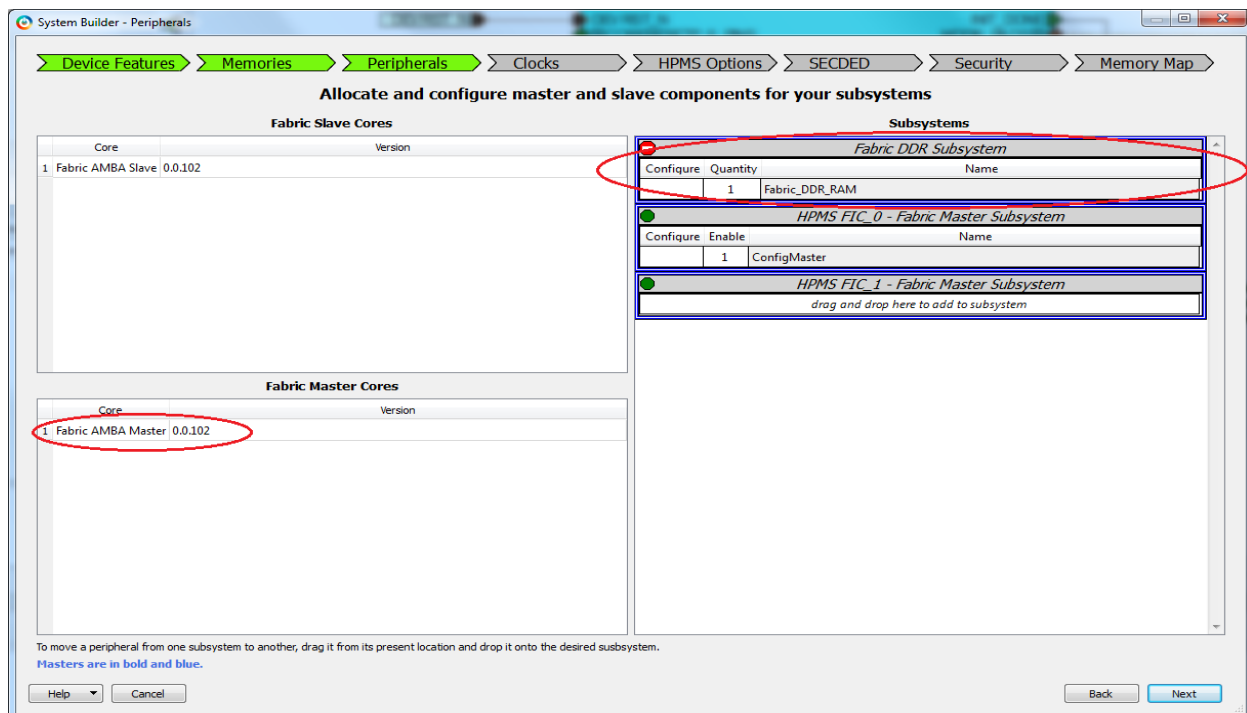
- Burst length: 8
- Burst Order: Interleaved
- Timing Mode: 1T
- CAS Latency: 3
- Self Refresh Enabled: No
- Auto Refresh Burst Count: 8
- PowerDown Enabled: Yes
- Stop the clock: No
- Deep PowerDown enabled: No

- No Activity clocks for Entry: 320

#### Memory Timing

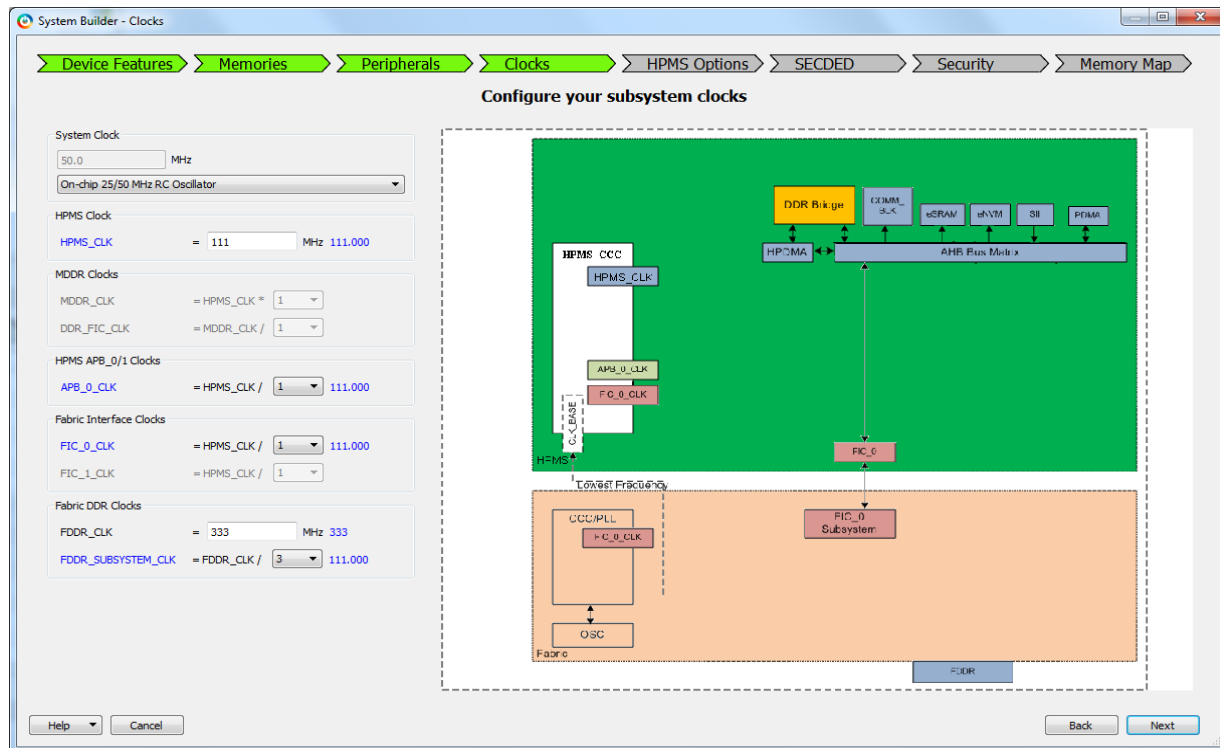
- Time To Hold Reset Before INIT – 67584 clks
  - MRD: 4 clks
  - RAS (Min): 8 clks
  - RAS (Max): 8192 clks
  - RCD: 6 clks
  - RP: 7 clks
  - REF: 3104 clks
  - RC: 3 clks
  - XP: 3 clks
  - CKE: 3 clks
  - RFC: 79 clks
  - FAW: 0 clks
7. Navigate to the **Peripherals** tab. The **Peripherals** tab allows to configure the Fabric AMBA Master and Fabric AMBA Slave required for the design. Drag and drop the required master/slave to the corresponding subsystem. The following image shows the **Peripherals** tab. Drag and drop the **Fabric Master core** to the **Fabric DDR Subsystem**. This allows to configure the type of interface as AXI, single AHB-Lite. On completing the configuration, the selected interface is enabled. The user logic in the FPGA fabric can access the DDR memory through the FDDR using these interfaces.

**Figure 87 • System Builder - Peripherals Tab**



8. Click **Next** to navigate to the **Clocks** tab. The **Clocks** tab allows to configure the **System Clock** and subsystem clocks. The FDDR subsystem operates on FDDR\_CLK frequency, which can be configured up to 333 MHz. The FDDR subsystem clock (CLK\_BASE) can be configured as a ratio-1, 2, 3, 4, 6, 8, 12, or 16 of FDDR\_CLK. The maximum frequency of FDDR subsystem clock is 200 MHz. FDDR subsystem clock has to be driven from the FPGA fabric. The following image shows the **System Builder - Clocks** tab.

**Figure 88 • FDDR Clock Configuration**



#### 4.7.1.1 I/O Configuration

In the **I/O Editor** window, configure the I/O settings such as ODT and drive strength. The following image shows the **I/O Editor** window.

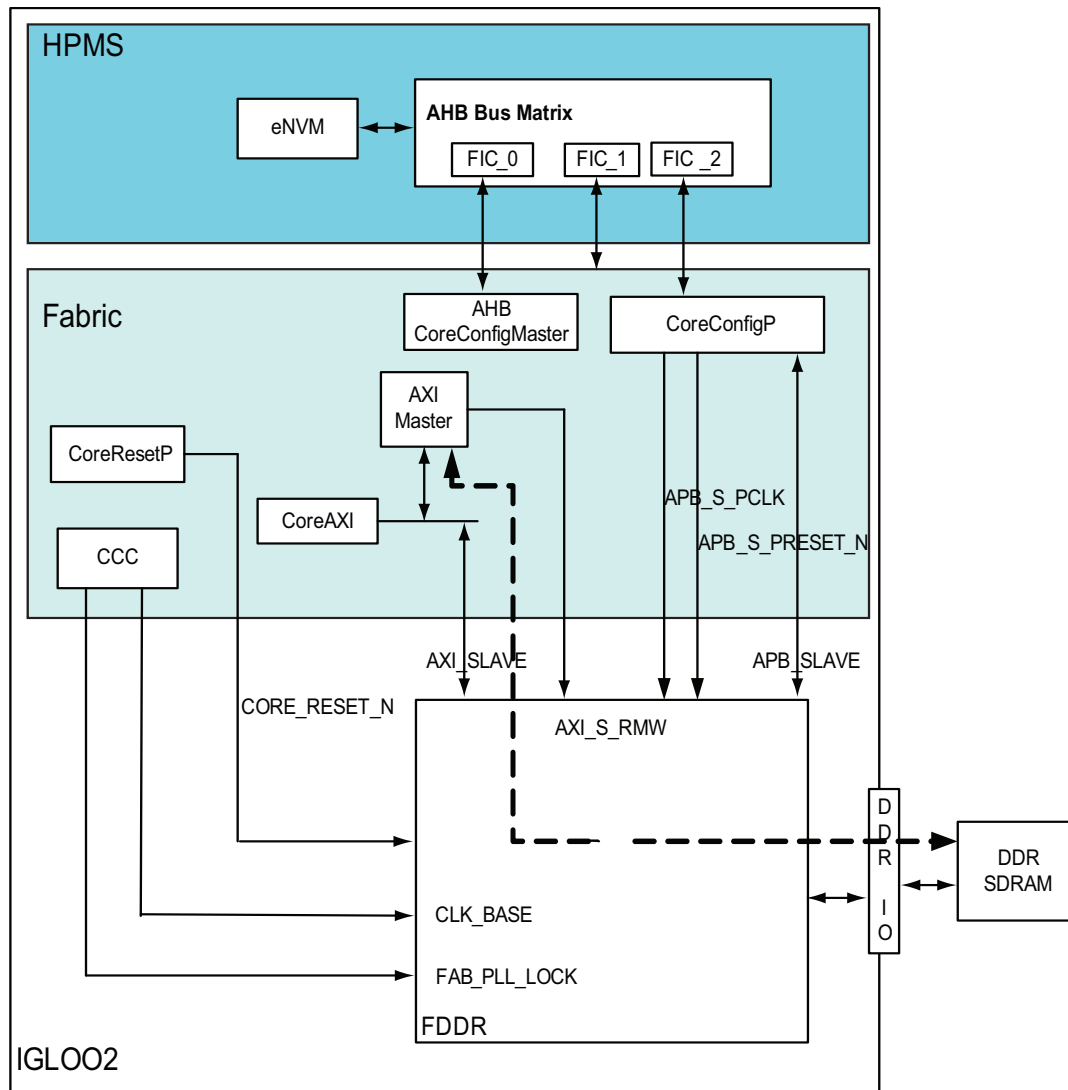
**Figure 89 • I/O Editor Window**

HW-IO Editor

Part Name	Direction	I/O Standard	Pin Number	Locked	Bank Name	I/O state in Flash-Wipe mode	Resistor Val.	I/O available in Flash-Wipe mode	Switch Trigger	Cnt. Sts.	Cnt. Jmp. (ms)	Low
PDKM_CTL_0N	Output	5VTL18E	AG29	<input checked="" type="checkbox"/>	Bank5	7825747E	None	No	—	—	—	
PDKM_CTL_0N	Output	5VTL18E	AG29	<input checked="" type="checkbox"/>	Bank5	7825747E	None	No	—	—	—	
PDKM_CTL_0N	Input	5VTL18E	AG13	<input checked="" type="checkbox"/>	Bank5	7825747E	None	No	OFF	OFF	50	
PDKM_CTL_0N	Input	5VTL18E	AG16	<input checked="" type="checkbox"/>	Bank5	7825747E	None	No	OFF	OFF	50	
PDKM_CTL_0N	Input	5VTL18E	AG19	<input checked="" type="checkbox"/>	Bank5	7825747E	None	No	OFF	OFF	50	
PDKM_CTL_0N	Input	5VTL18E	AG22	<input checked="" type="checkbox"/>	Bank5	7825747E	None	No	OFF	OFF	50	
PDKM_CTL_0N	Input	5VTL18E	AK12	<input checked="" type="checkbox"/>	Bank5	7825747E	None	No	OFF	OFF	50	
PDKM_CTL_0N	Input	5VTL18E	A312	<input checked="" type="checkbox"/>	Bank5	7825747E	None	No	OFF	OFF	50	
PDKM_CTL_0N	Input	5VTL18E	AG12	<input checked="" type="checkbox"/>	Bank5	7825747E	None	No	OFF	OFF	50	
PDKM_CTL_0N	Input	5VTL18E	AF12	<input checked="" type="checkbox"/>	Bank5	7825747E	None	No	OFF	OFF	50	
PDKM_CTL_0N	Input	5VTL18E	AK19	<input checked="" type="checkbox"/>	Bank5	7825747E	None	No	OFF	OFF	50	
PDKM_CTL_0N	Input	5VTL18E	AG14	<input checked="" type="checkbox"/>	Bank5	7825747E	None	No	OFF	OFF	50	
PDKM_CTL_0N	Input	5VTL18E	AF14	<input checked="" type="checkbox"/>	Bank5	7825747E	None	No	OFF	OFF	50	

#### 4.7.2 Accessing FDDR from FPGA Fabric through the AXI Interface

The AXI master in the FPGA fabric can access the DDR memory through the FDDR subsystem. The following illustration shows the FDDR with the AXI interface. The FDDR registers are configured from the FPGA fabric using the CoreConfigMaster IP through the CoreConfigP IP APB interface.

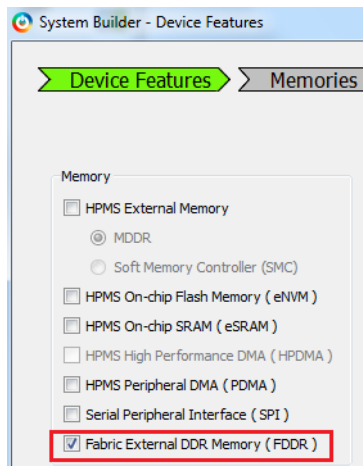
**Figure 90 • FDDR Subsystem with AXI Interface**

Read, write, and read-modify-write transactions are initiated by the AXI master to read from or write the data to the DDR memory after initializing the FDDR registers.

The following steps access the FDDR from the AXI master in the FPGA fabric:

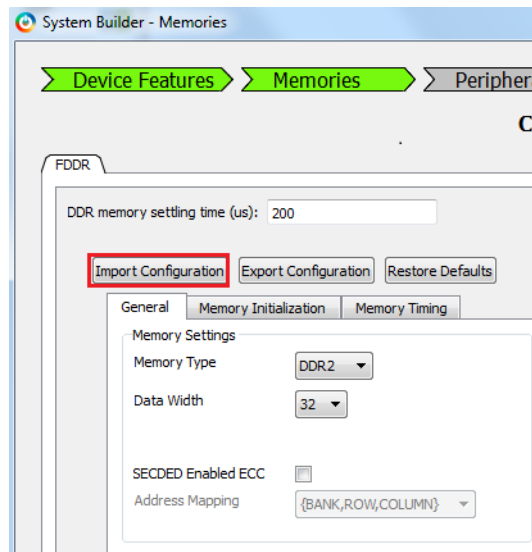
1. Go to the **System Builder - Device Features** tab and check the **Fabric External DDR Memory (FDDR)** check box and leave the rest of the check boxes unchecked. The following image shows the **System Builder - Device Features** tab.

**Figure 91 • System Builder - Device Features Tab**

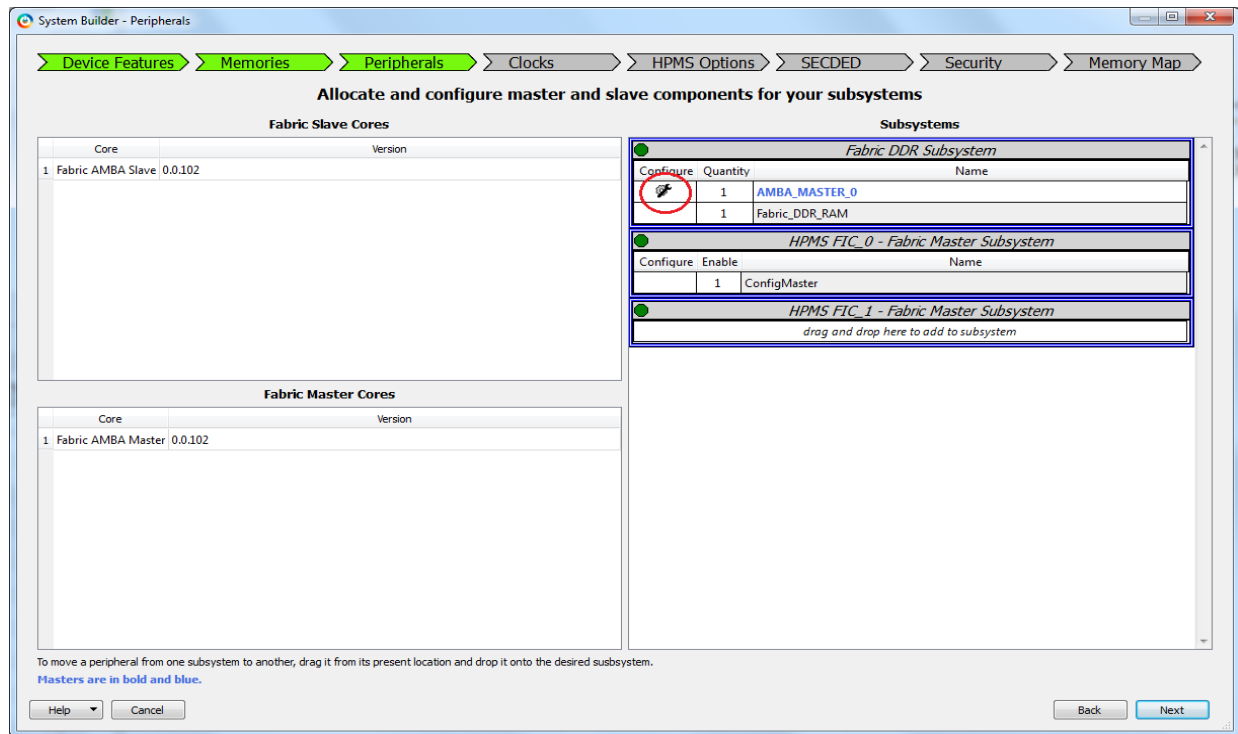


2. Configure the **HPMS External Memory** in **Memories** tab (shown in the following image). In this example, the design is created to access the DDR3 memory with a 32-bit data width and no ECC.
3. Set the **DDR memory settling time** to 200 us and then click **Import Register Configuration**.

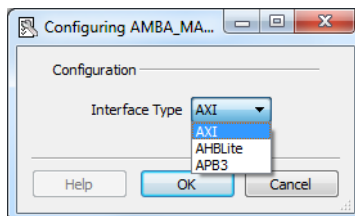
**Figure 92 • Memory Configuration**



4. Navigate to the **Peripherals** tab.
5. In the **Peripherals** tab, drag the Fabric Master Core and drop on to the **Fabric DDR Subsystem**. You can see that the master is added to the subsystem. The following image shows the **Peripherals** tab with the **AMBA\_MASTER\_0** added.
6. Click the **Configure** icon to open the **AMBA\_MASTER\_0** dialog. The following image shows the **Peripherals** tab with the **Configure** icon highlighted.

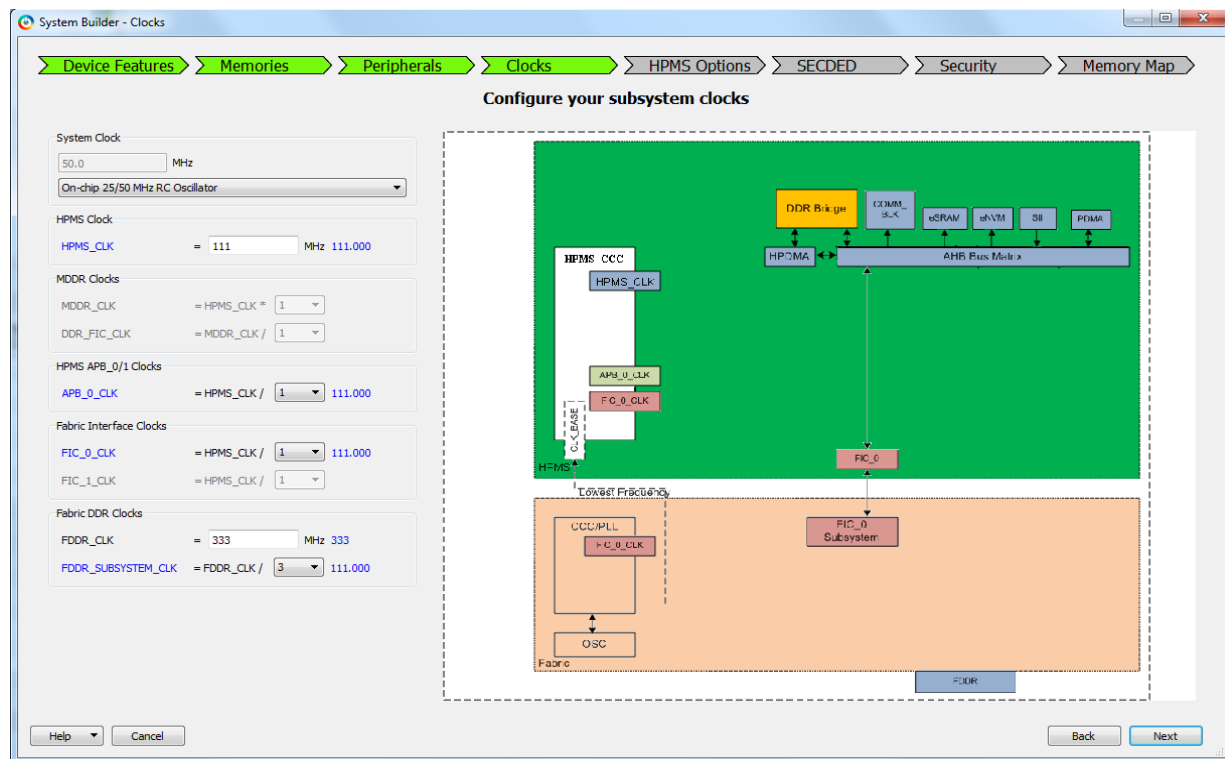
**Figure 93 • Fabric DDR Subsystem Configuration Dialog**

7. In the Configuring **AMBA\_MASTER\_0** dialog, select the **Interface Type** as **AXI** and then click **OK**. The following image shows the **AMBA Master - Configuration** dialog.

**Figure 94 • AMBA Master Configuration**

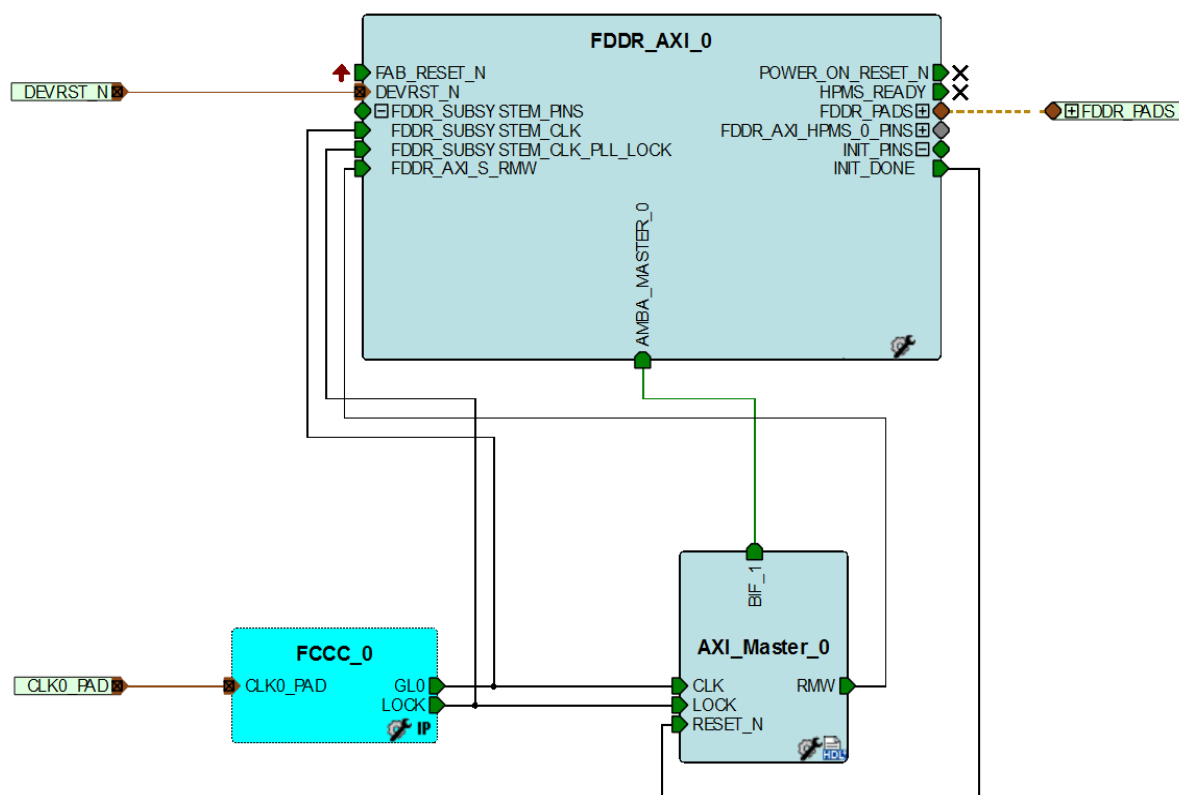
8. Configure the **System Clock** and Subsystem clocks in **Clocks** tab. The following image shows the **Clocks** configuration dialog.
  - Select the **On-chip 25/50 MHz RC Oscillator**
  - Configure **HPMS\_CCC** for **HPMS\_CLK**, **APB\_0\_CLK**, and **FIC\_0\_CLK**.
9. Configure **FDDR\_CLK** as 333 MHz and **FDDR\_SUBSYSTEM\_CLK** as **FDDR\_CLK/3**, that is, 111 MHz (need to drive this from the Fabric clock).

**Figure 95 • Clocks Configuration**



10. Navigate to the **Memory Map** tab giving the required data in the rest of the **System Builder** tabs.
11. Instantiate the user AXI master logic in the SmartDesign canvas to access the FDDR through the AXI interface. Make sure that the AXI master logic accesses the FDDR after configuring the FDDR registers.
12. Instantiate the CCC block in the SmartDesign canvas and configure it to generate 111 MHz clock.
13. Connect the AXI\_Master logic signals as follows:
  - CLK to GL0 of FCCC\_0 and FDDR\_SUBSYSTEM\_CLK
  - LOCK to LOCK of FCCC\_0 and FDDR\_SUBSYSTEM\_LOCK
  - RESET\_N to INIT\_DONE of FDDR\_AXI\_0
  - AXI\_S\_RMW to FDDR\_AXI\_S\_RMW of FDDR\_AXI\_0 block
14. The following illustration shows the rest of the connections in the top level design.



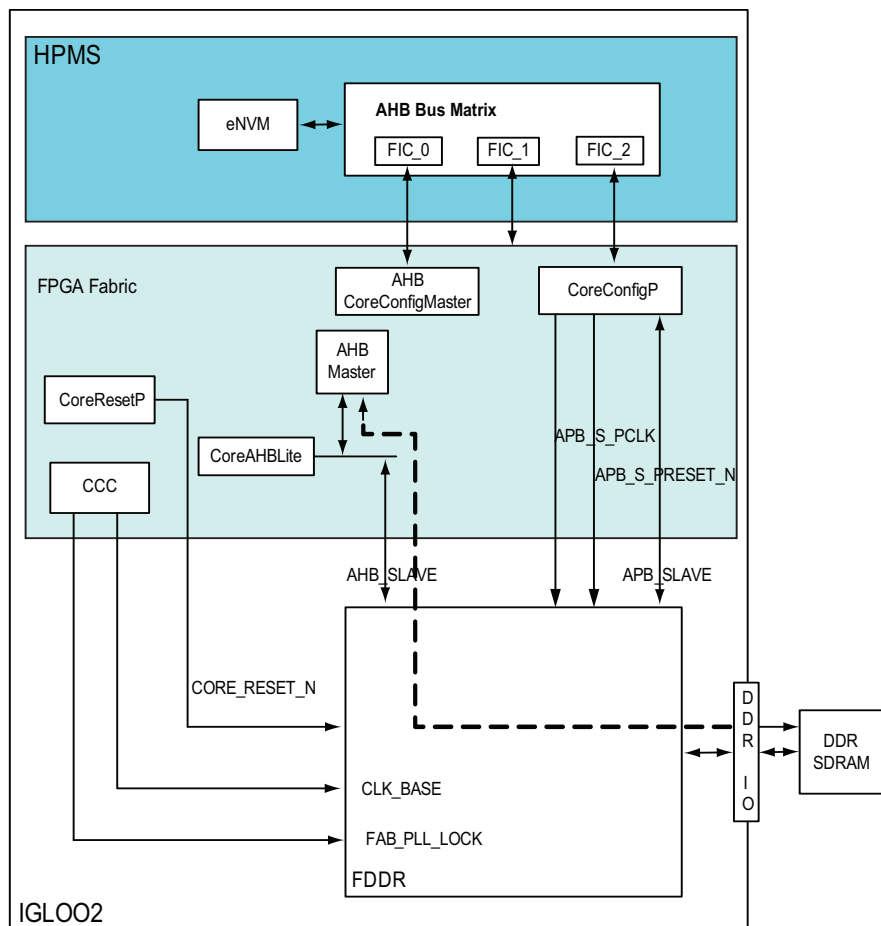
**Figure 96 • SmartDesign Connections (Top Level View)**

For FDDR AXI throughput, see *AC422: SmartFusion2 - Optimizing DDR Controller for Improved Efficiency - Libero v11.7 Application Note*.

### 4.7.3 Accessing FDDR from FPGA Fabric through the AHB Interface

The FDDR subsystem can be used to access the DDR memory using the AHB-Lite interface. The following illustration shows the **FDDR with AHB-Lite interface**.

Figure 97 • FDDR with AHB-Lite interface



The procedure for accessing the FDDR from the AHB master in the FPGA fabric is the same as "Accessing FDDR from FPGA Fabric through the AXI Interface" section on page 165, except for the following:

- Configure the **AMBA Master Interface Type** as AHB-Lite in the **Fabric DDR Subsystem** in the **Peripherals** tab of the **System Builder** wizard.

Table 140, page 171 lists the FDDR throughput for the following configuration:

- Fabric Interface: AHB
- FDDR Mode: DDR3
- Fabric Clock to FDDR Clock Ratio: 1:4
- PHY Width: 16 and 32
- Clock Frequency: 80 MHz

The other parameters are configured similar to the FDDR configuration in *AC422: SmartFusion2 - Optimizing DDR Controller for Improved Efficiency - Libero v11.7 Application Note*.

Table 140 • FDDR Throughput (for AHB)

FDDR-Fabric Interface-Memory	Frequency Ratio (CLK_BASE:FDDR_CLK)	PHY Width	Write Transaction BW (MB/sec)	Read Transaction BW (MB/sec)
FDDR_AHB-DDR3	1:4 80 MHz:320 MHz	PHY_16	80 MB	79 MB
		PHY_32	80 MB	79 MB

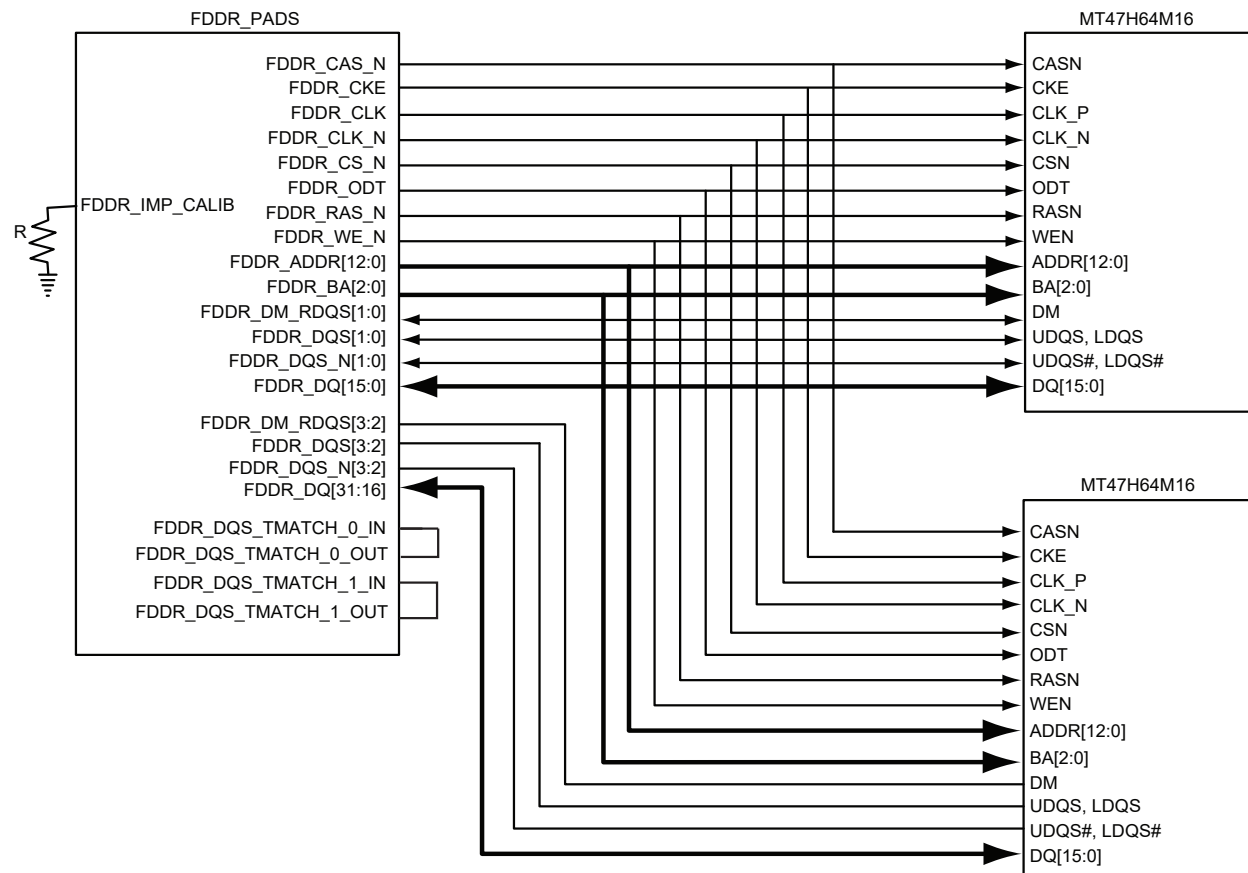
## 4.8 DDR Memory Device Examples

This section describes how to connect DDR memories to IGLOO2 FDDR\_PADs with examples.

### 4.8.1 Example 1: Connecting 32-Bit DDR2 to FDDR\_PADs

The following illustration shows DDR2 SDRAM connected to the FDDR of a IGLOO2 device. Micron's MT47H64M16 is a 128 MB density device with x16 data width. The FDDR is configured in Full Bus Width mode and without SECDED. The total amount of DDR2 memory connected to the FDDR is 256 MB.

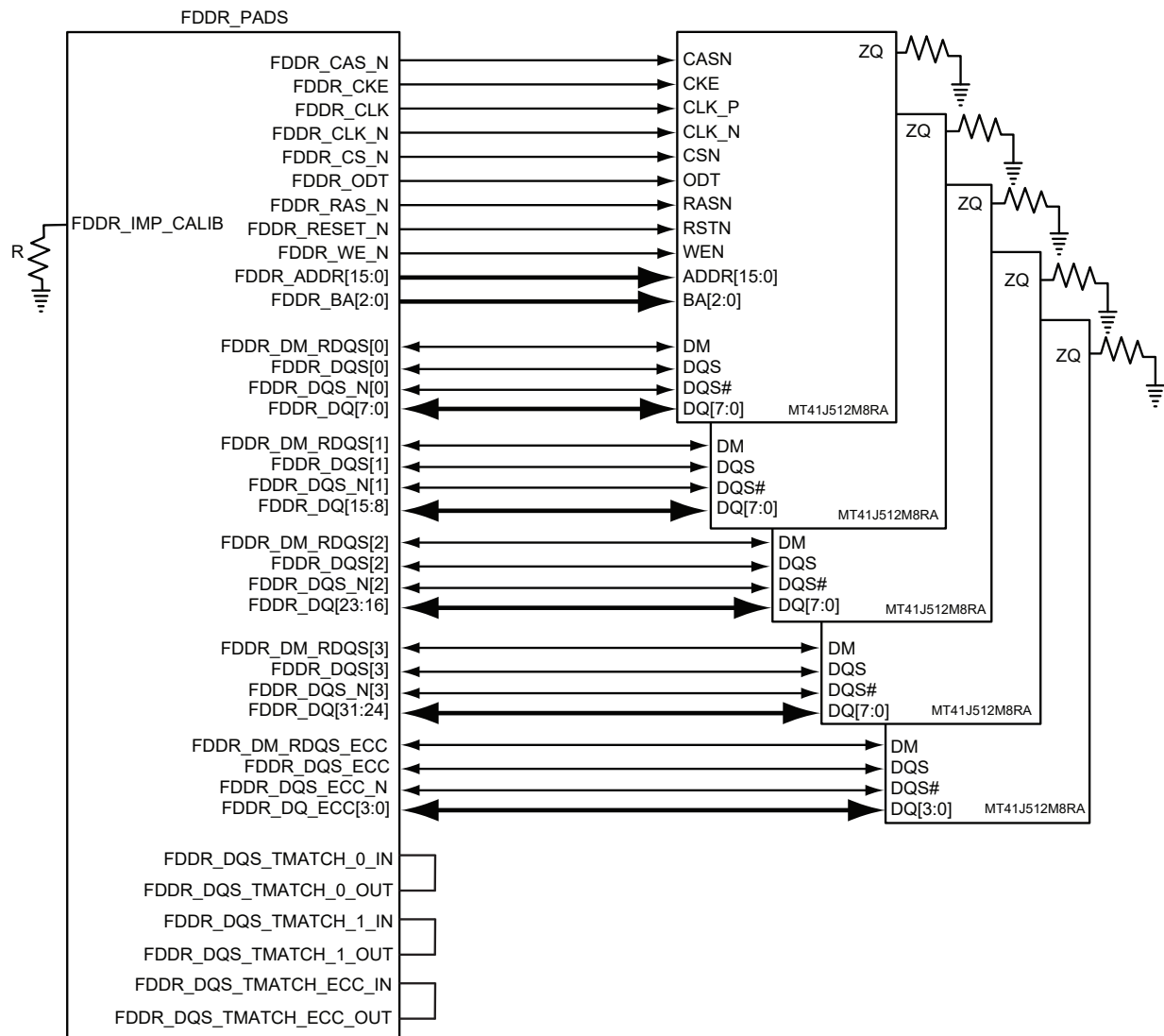
**Figure 98 • x16 DDR2 SDRAM Connected to FDDR**



### 4.8.2 Example 2: Connecting 32-Bit DDR3 to FDDR\_PADs with SECDED

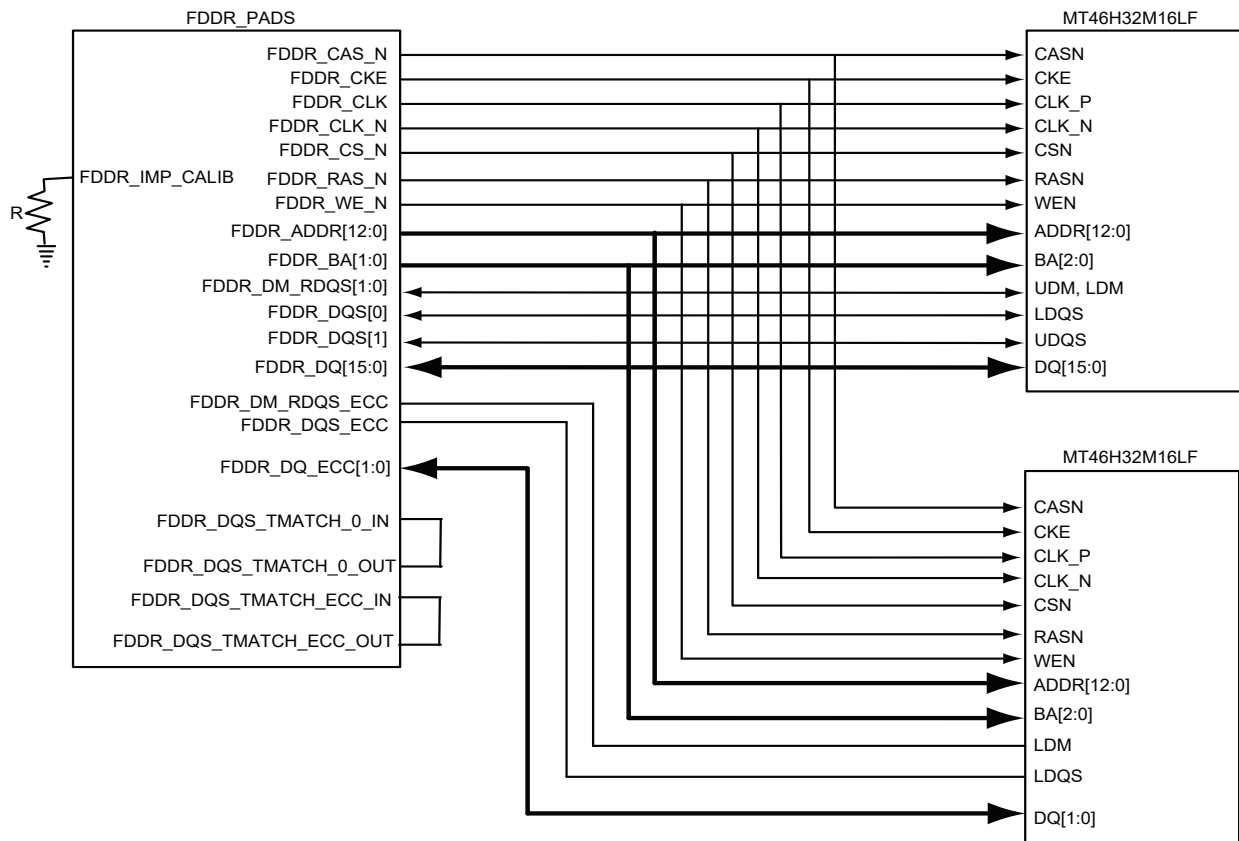
The following illustration shows DDR3 SDRAM connected to the FDDR of a IGLOO2 device. Micron's MT41J512M8RA is a 512 MB density device with x8 data width. The FDDR is configured in Full Bus Width mode with SECDED enabled. The SDRAM connected to FDDR\_DQ\_ECC[3:0] is used to store SECDED bits. The total amount of DDR3 memory (excluding memory for SECDED) connected to FDDR is 2 GB.

**Figure 99 • x8 DDR3 SDRAM Connection to FDDR**



### 4.8.3 Example 3: Connecting 16-Bit LPDDR to FDDR\_PADS with SECCDED

The following illustration shows LPDDR1 SDRAM connected to the FDDR of a IGLOO2 device. The Micron's MT46H32M16LF is a 64 MB density device with x16 data width. The FDDR is configured in Full Bus Width mode with SECCDED enabled. The SDRAM connected to FDDR\_DQ\_ECC[1:0] is used to store SECCDED bits. The total amount of LPDDR1 memory (excluding memory for SECCDED) connected to FDDR is 64 MB.

**Figure 100 • x16 LPDDR1 SDRAM Connection to FDDR**

## 4.9 FDDR Configuration Registers

This section provides FDDR subsystem registers along with the address offset, functionality, and bit definitions. The registers are categorized based on the controller blocks in the FDDR subsystem.

The following table lists the categories of registers and their offset addresses.

**Table 141 • Address Table for Register Interfaces**

Registers	Address Offset Space
DDR Controller Configuration Register, <a href="#">Table 27</a> , page 62	0x000:0x1FC
PHY Configuration Register Summary, <a href="#">Table 101</a> , page 102	0x200:0x3FC
DDR_FIC Configuration Register Summary, <a href="#">Table 103</a> , page 102	0x400:0x4FC
FDDR SYSREG Configuration Register Summary, <a href="#">Table 142</a> , page 175	0x500:0x5FC
Reserved	0x600:0x7FC

**Note:** The FDDR SYSREG configuration registers can be locked to prevent them from being overwritten by the masters that have access to these registers. For information on how to lock/unlock these registers, see "[Appendix B: Register Lock Bits Configuration](#)" on page 203.

## 4.9.1 FDDR SYSREG Configuration Register Summary

**Table 142 • FDDR SYSREG**

Register Name	Address Offset	Register Type	Flash	Reset Source	Description
PLL_CONFIG_LOW_1	0x500	RW	P	PRESETN	Comes from SYSREG. Controls the corresponding configuration input of the FPLL.
PLL_CONFIG_LOW_2	0x504	RW	P	PRESETN	Comes from SYSREG. Controls the corresponding configuration input of the FPLL.
PLL_CONFIG_HIGH	0x508	RW	P	PRESETN	Comes from SYSREG. Controls the corresponding configuration input of the FPLL.
FDDR_FACC_CLK_EN	0x50C	RW	P	PRESETN	Enables the clock to the DDR memory controller.
FDDR_FACC_MUX_CONFIG	0x510	RW	P	PRESETN	Selects the standby glitch-free multiplexers within the fabric alignment clock controller (FACC).
FDDR_FACC_DIVISOR_RATIO	0x514	RW	P	PRESETN	Selects the ratio between CLK_A and CLK_DDR_FIC.
PLL_DELAY_LINE_SEL	0x518	RW	P	PRESETN	Selects the delay values to be added to the FPLL.
FDDR_SOFT_RESET	0x51C	RW	P	PRESETN	Soft reset register for FDDR
FDDR_IO_CALIB_CR	0x520	RW	P	PRESETN	Configurations register for DDRIO calibration block
FDDR_INTERRUPT_ENABLE	0x524	RW	P	PRESETN	Interrupt enable register
F_AXI_AHB_MODE_SEL	0x528	RW	P	PRESETN	Selects AXI/AHB interface in the fabric.
PHY_SELF_REF_EN	0x52C	RW	P	PRESETN	Automatic calibration lock is enabled.
FDDR_FAB_PLL_CLK_SR	0x530	RO	–	PRESETN	Indicates the lock status of the fPLL.
FDDR_FPLL_CLK_SR	0x534	RO	–	PRESETN	Indicates the lock status of the fabric PLL.
FDDR_INTERRUPT_SR	0x53C	RO	–	PRESETN	Interrupt status register
FDDR_IO_CALIB_SR	0x544	RO	–	PRESETN	I/O calibration status register
FDDR_FATC_RESET	0x548	RW	P	PRESETN	Reset to fabric portion of the fabric alignment test circuit

## 4.9.2 FDDR SYSREG Configuration Register Bit Definitions

**Table 143 • PLL\_CONFIG\_LOW\_1**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

**Table 143 • PLL\_CONFIG\_LOW\_1**

[15:6]	PLL_FEEDBACK_DIVISOR	0×2	Can be configured to control the corresponding configuration input of the FPLL. Feedback divider value (SSE = 0) (binary value + 1: 00000000 = ÷1, .... 111111111 = ÷ 1,024) Feedback divider value (SSE = 1) (binary value + 1: 0000000 = ÷1, .... 1111111 = ÷ 128)
[5:0]	PLL_REF_DIVISOR	0×1	Can be configured to control the corresponding configuration input of the FPLL. Reference divider value (binary value + 1: 000000 = ÷ 1)

**Table 144 • PLL\_CONFIG\_LOW\_2**

Bit Number	Name	Reset Value	Description
[31:4]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	FDDR_PLL_RESET	0×1	This is only for FDDR in M2S/M2GL 150 device. 1: FDDR PLL held in reset 0: FDDR PLL is not in reset
[2:0]	PLL_OUTPUT_DIVISOR	0×2	Configures the amount of division to be performed on the internal (multiplied) PLL clock, in order to generate the DDR clock. Output divider value 000: ÷1 001: ÷2 010: ÷4 011: ÷8 100: ÷16 101: ÷32 It is possible to configure the PLL output divider as ÷1; this setting must not be used when the DDR is operational.

**Table 145 • PLL\_CONFIG\_HIGH**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	PLL_PD	0×0	When PD is asserted, the PLL will power down and outputs will be Low. PD has precedence over all other functions.
14	PLL_FSE	0×0	Chooses between internal and external input paths. 0: FB pin input 1: Internal feedback FB should be tied off (High or Low) and not left floating when FSE is High. FB should connect directly or through the clock tree to PLLOUT when FSE is Low. SSE is ineffective when FSE = 0.

**Table 145 • PLL\_CONFIG\_HIGH (continued)**

Bit Number	Name	Reset Value	Description
13	PLL_MODE_3V3	0×1	Analog voltage selection 1: 3.3 V 0: 2.5 V
12	PLL_MODE_1V2	0×1	Core voltage selection 1: 1.2 V 0: 1.0 V The wrong selection (when operating at 1 V, the jitter is not within the required limit for operation of DDR) may cause the PLL not to function, but will not damage the PLL.
11	PLL_BYPASS	0×1	If 1, powers down the PLL core and bypasses it such that PLLOUT tracks REFCK. BYPASS has precedence over RESET. Microsemi recommends that either BYPASS or RESET are asserted until all configuration controls are set in the desired working value, and the power supply and reference clock are stable within operating range.
[10:7]	PLL_LOCKCNT	0×F	Configured to control the corresponding configuration input of the FPLL. LOCK counter Value $2^{(\text{binary value} + 5)}$ 0000: 32 1111: 1048576 For the number of reference cycles before LOCK is asserted from LOCK being detected.
[6:4]	PLL_LOCKWIN	0×0	000: 500 ppm 100: 8000 ppm 001: 1000 ppm 101: 16000 ppm 010: 2000 ppm 110: 32000 ppm 011: 4000 ppm 111: 64000 ppm Phase error window for Lock assertion as a fraction of divided reference period. Values are at typical PVT only and are not PVT compensated.
[3:0]	PLL_FILTER_RANGE	0×9	PLL filter range 0000: BYPASS 0111: 18–29 MHz 0001: 1–1.6 MHz 1000: 29–46 MHz 0010: 1.6–2.6 MHz 1001: 46–75 MHz 0011: 2.6–4.2 MHz 1010: 75–120 MHz 0100: 4.2–6.8 MHz 1011: 120–200 MHz 0101: 6.8–11 MHz 0110: 11–18 MHz

**Table 146 • FDDR\_FACC\_CLK\_EN**

Bit Number	Name	Reset Value	Description
------------	------	-------------	-------------



**Table 146 • FDDR\_FACC\_CLK\_EN**

[31:1]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DDR_CLK_EN	0×1	Enables the clock to the DDR memory controller.

**Table 147 • FDDR\_FACC\_MUX\_CONFIG**

Bit Number	Name	Reset Value	Description
[31:9]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	FACC_FAB_REF_SEL	0×0	Selects the source of the reference clock to be supplied to the FPLL. 0: 25/50 MHz RC oscillator selected as the reference clock for the FPLL. 1: Fabric clock (FDDR_SUBSYSTEM_CLK) selected as the reference clock for the FPLL.
7	FACC_GLMUX_SEL	0×1	Selects the four glitch-free multiplexers within the FACC, which are related to the aligned clocks. All four of these multiplexers are switched by one signal. Allowed values: 0: HPMS_CLK, PCLK0, PCLK1, CLK_DDR_FIC, all driven from stage 2 dividers (from CLK_SRC) 1: HPMS_CLK, PCLK0, PCLK1, CLK_DDR_FIC, all driven from CLK_STANDBY
6	FACC_PRE_SRC_SEL	0×0	Selects whether CLK_1MHZ or ccc2asic is to be fed into the source glitch-free multiplexer. 0: CLK_1MHZ is fed into the source glitch-free multiplexer. 1: ccc2asic is fed into the source glitch-free multiplexer.
[5:3]	FACC_SRC_SEL	0×0	Selects the source multiplexer within the FACC. This is used to allow one of four possible clocks to proceed through the FACC dividers, for generation of normal functional (run-time) FDDR subsystem clocks. There are three individual 2 to 1 glitch-free multiplexers in the 4 to 1 source glitch-free multiplexer. FACC_SRC_SEL[0] is used to select the lower source MUX. 0: CLK_SRC driven from CLK_25_50 MHZ 1: CLK_SRC driven from clk_xtal FACC_SRC_SEL[1] is used to select the upper source MUX. 0: CLK_SRC driven from output of PRE_SRC_MUX (either clk_1mhz or ccc2asic) 1: CLK_SRC driven from FDDR_PLL_OUT_CLK FACC_SRC_SEL[2] is used to select output source MUX. 0: CLK_SRC driven from output of lower source MUX 1: CLK_SRC driven from output of upper source MUX

**Table 147 • FDDR\_FACC\_MUX\_CONFIG (continued)**

Bit Number	Name	Reset Value	Description
[2:0]	FACC_STANDBY_SEL	0×0	<p>Selects the standby glitch-free multiplexers within the FACC. This is used to allow one of four possible clocks to proceed through to the FDDR subsystem during FACC PLL initialization time (before the FPLL comes into lock).</p> <p>FACC_STANDBY_SEL[0] is used to select the lower standby MUX.            0: CLK_STANDBY driven from CLK_25_50 MHZ            1: CLK_STANDBY driven from CLK_XTAL</p> <p>FACC_STANDBY_SEL[1] is used to select upper standby MUX.            0: CLK_STANDBY driven from CLK_1 MHZ            1: CLK_STANDBY driven from ccc2asic</p> <p>FACC_STANDBY_SEL[2] is used to select the output standby MUX.            0: CLK_STANDBY driven from output of lower standby MUX            1: CLK_STANDBY driven from output of upper standby MUX</p>

**Table 148 • FDDR\_FACC\_DIVISOR\_RATIO**

Bit Number	Name	Reset Value	Description
[31:8]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[7:5]	BASE_DIVISOR	0×0	<p>Selects the ratio between CLK_A and the regenerated version of CLK_BASE, called CLK_BASE_REGEN. Allowed values:</p> <p>000: CLK_A: CLK_BASE_REGEN ratio is 1:1            001: CLK_A: CLK_BASE_REGEN ratio is 2:1            010: CLK_A: CLK_BASE_REGEN ratio is 4:1            100: CLK_A: CLK_BASE_REGEN ratio is 8:1            101: CLK_A: CLK_BASE_REGEN ratio is 16:1            110: CLK_A: CLK_BASE_REGEN ratio is 32:1            Other values: Reserved</p>
[4:3]	DIVISOR_A	0×0	<p>Selects the ratio between CLK_SRC and CLK_A, which is an intermediate clock within the FACC.</p> <p>00: CLK_SRC:CLK_A ratio is 1:1            01: CLK_SRC:CLK_A ratio is 2:1            10: CLK_SRC:CLK_A ratio is 3:1            11: Reserved</p>
[2:0]	DDR_FIC_DIVISOR	0×0	<p>Selects the ratio between CLK_A and CLK_DDR_FIC.</p> <p>000: CLK_A: CLK_DDR_FIC ratio is 1:1            001: CLK_A: CLK_DDR_FIC ratio is 2:1            010: CLK_A: CLK_DDR_FIC ratio is 4:1            100: CLK_A: CLK_DDR_FIC ratio is 8:1            101: CLK_A: CLK_DDR_FIC ratio is 16:1            110: CLK_A: CLK_DDR_FIC ratio is 32:1            Other values: Reserved</p>

**Table 149 • PLL\_DELAY\_LINE\_SEL**

Bit Number	Name	Reset Value	Description
[31:4]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[3:2]	PLL_FB_DEL_SEL	0×0	Selects the delay values that are added to the FPLL feedback clock before being output to the FPLL. 00: No buffer delay 01: One buffer delay 10: Two buffers delay 11: Three buffers delay
[1:0]	PLL_REF_DEL_SEL	0×0	Selects the delay values that are added to the FPLL reference clock before being output to the FPLL. 00: No buffer delay 01: One buffer delay 10: Two buffers delay 11: Three buffers delay

**Table 150 • FDDR\_SOFT\_RESET**

Bit Number	Name	Reset Value	Description
[31:2]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	FDDR_DDR_FIC_SOFTRESET	0×1	When 1, holds the DDR_FIC (AXI/AHB) interface controller in reset.
0	FDDR_CTLR_SOFTRESET	0×1	When 1, holds the FDDR subsystem in reset.

**Table 151 • FDDR\_IO\_CALIB**

Bit Number	Name	Reset Value	Description
[31:15]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	CALIB_TRIM	0×0	Indicates override of the calibration value from the pc code / programmed code values in the DDRIO calibration block.

**Table 151 • FDDR\_IO\_CALIB**

13	CALIB_LOCK	0×0	Used in the DDRIO calibration block as an override to lock the codes during intermediate runs. When the firmware receives CALIB_INTRPT, it may choose to assert this signal by prior knowledge of the traffic without going through the process of putting the DDR into self refresh.
12	CALIB_START	0×0	Indicates that rerun of the calibration state machine is required in the DDRIO calibration block.
[11:6]	NCODE	0×0	Indicates the DPC override NCODE from flash in DDRIO calibration. This can also be overwritten from the firmware.
[5:0]	PCODE	0×0	Indicates the PC override PCODE from flash in the DDRIO calibration block. This is also be overwritten from the firmware.

**Table 152 • FDDR\_INTERRUPT\_ENABLE**

Bit Number	Name	Reset Value	Description
[31:7]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	DDR_FIC_INT_ENABLE	0×0	Masking bit to enable DDR_FIC interrupt
5	IO_CALIB_INT_ENABLE	0×0	Masking bit to enable DDR I/O calibration interrupt
4	FDDR_ECC_INT_ENABLE	0×0	Masking bit to enable ECC error interrupt
3	FABRIC_PLL_LOCKLOST_INT_ENABLE	0×0	Masking bit to enable FAB_PLL_LOCK_LOST interrupt
2	FABRIC_PLL_LOCK_INT_ENABLE	0×0	Masking bit to enable FAB_PLL_LOCK interrupt
1	FPLL_LOCKLOST_INT_ENABLE	0×0	Masking bit to enable FPLL_LOCK_LOST interrupt
0	FPLL_LOCK_INT_ENABLE	0×0	Masking bit to enable FPLL_LOCK interrupt

**Table 153 • F\_AXI\_AHB\_MODE\_SEL**

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	F_AXI_AHB_MODE	0×0	1: AXI interface in the fabric will be selected. 0: AHB interface in the fabric will be selected.

**Table 154 • PHY\_SELF\_REF\_EN**

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	PHY_SELF_REF_EN	0×0	If 1, automatic calibration lock is enabled.

**Table 155 • FDDR\_FAB\_PLL\_CLK\_SR**

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	FAB_PLL_LOCK	0×0	Indicates the lock status of the Fabric PLL.

**Table 156 • FDDR\_FPLL\_CLK\_SR**

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	FPLL_LOCK	0×0	Indicates the lock status of the FPLL (PLL in FDDR).

**Table 157 • FDDR\_INTERRUPT\_SR**

Bit Number	Name	Reset Value	Description
[31:5]	Reserved	0×0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	DDR_FIC_INT	0×0	Indicates interrupt from DDR_FIC.
3	IO_CALIB_INT	0×0	The interrupt is generated when the calibration is finished. For the calibration after reset, this typically would be followed by locking the codes directly. For in-between runs during functional operation of DDR, the assertion of an interrupt does not guarantee lock because the state machine would wait for the ideal time (DRAM self refresh) for locking. This can be used by firmware to insert the ideal time and provides an indication that locked codes are available.

**Table 157 • FDDR\_INTERRUPT\_SR**

2	FDDR_ECC_INT	0x0	Indicates when the ECC interrupt from the FDDR subsystem is asserted.
1	PLL_LOCKLOST_INT	0x0	This bit indicates that a falling edge event occurred on the FPLL_LOCK signal. This indicates that the FPLL lost lock.
0	PLL_LOCK_INT	0x0	This bit indicates that a rising edge event occurred on the FPLL_LOCK signal. This indicates that the FPLL came into lock.

**Table 158 • FDDR\_IO\_CALIB\_SR**

Bit Number	Name	Reset Value	Description
31	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	CALIB_PCOMP	0x01	The state of the P analog comparator
13	CALIB_NCOMP	0x01	The state of the N analog comparator
[12:7]	CALIB_PCODE	0x3F	The current PCODE value set on the FDDR DDR I/O bank
[6:1]	CALIB_NCODE	0x3F	The current NCODE value set on the FDDR DDR I/O bank
0	CALIB_STATUS	0x0	This is 1 when the codes are actually locked. For the first run after reset, this would be asserted 1 cycle after CALIB_INTRPT. For in-between runs, this would be asserted only when the DRAM is put into self refresh or there is an override from the firmware (CALIB_LOCK).

**Table 159 • FDDR\_FATC\_RESET**

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	FATC_RESET	0x1	Reset to the fabric portion of the fabric alignment test circuit. 1: Reset active

## 4.10 Appendix A: How to Use the FDDR in SmartFusion2 Devices

This section describes how to use the FDDR subsystem in a design. It contains the following sections:

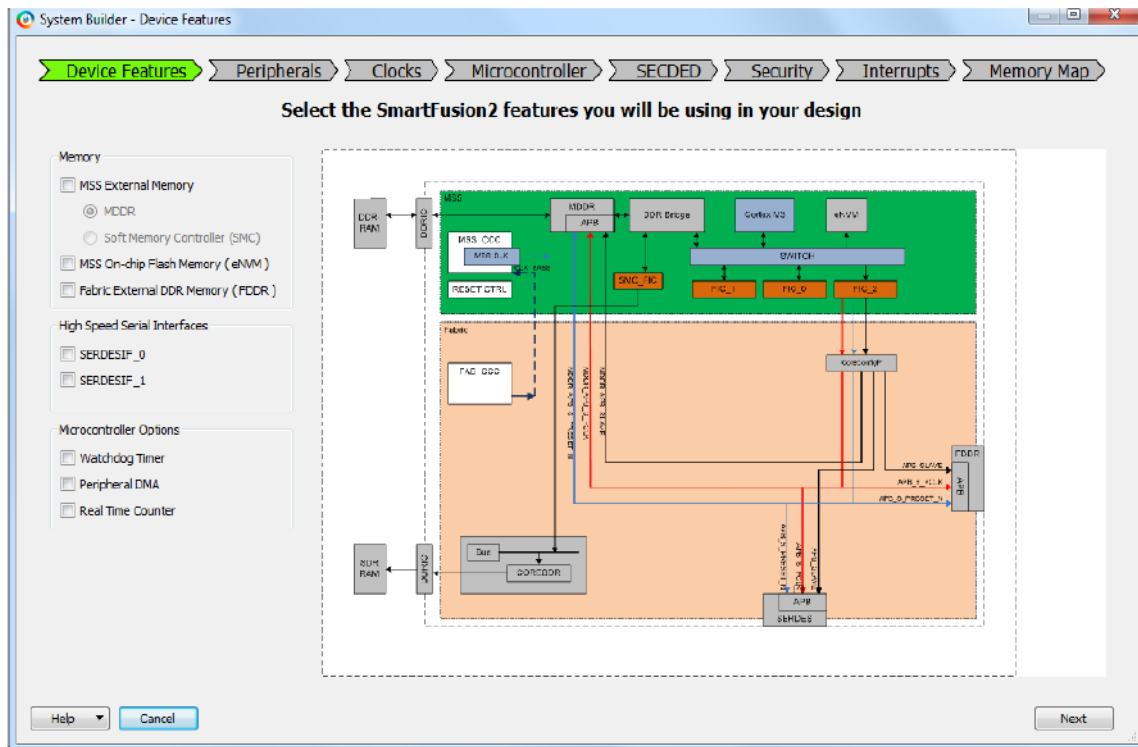
- [Design Flow Using System Builder](#)
- [Design Flow Using SmartDesign](#)
- [Use Model 1: Accessing FDDR from FPGA Fabric Through AXI Interface](#)
- [Use Model 2: Accessing FDDR from FPGA Fabric Through AHB Interface](#)

### 4.10.1 Design Flow Using System Builder

This section describes how to use FDDR in the SmartFusion2 devices using the System Builder graphical design wizard in the Libero Software.

The following image shows the initial **System Builder** window where you can select the features that you require. For details on how to launch the **System Builder** wizard and detailed information on how to use it, refer the [SmartFusion2 System Builder User Guide](#). For more information on DDR initialization, refer to the [SmartFusion2 DDR Controller and Serial High Speed Controller Initialization Methodology](#).

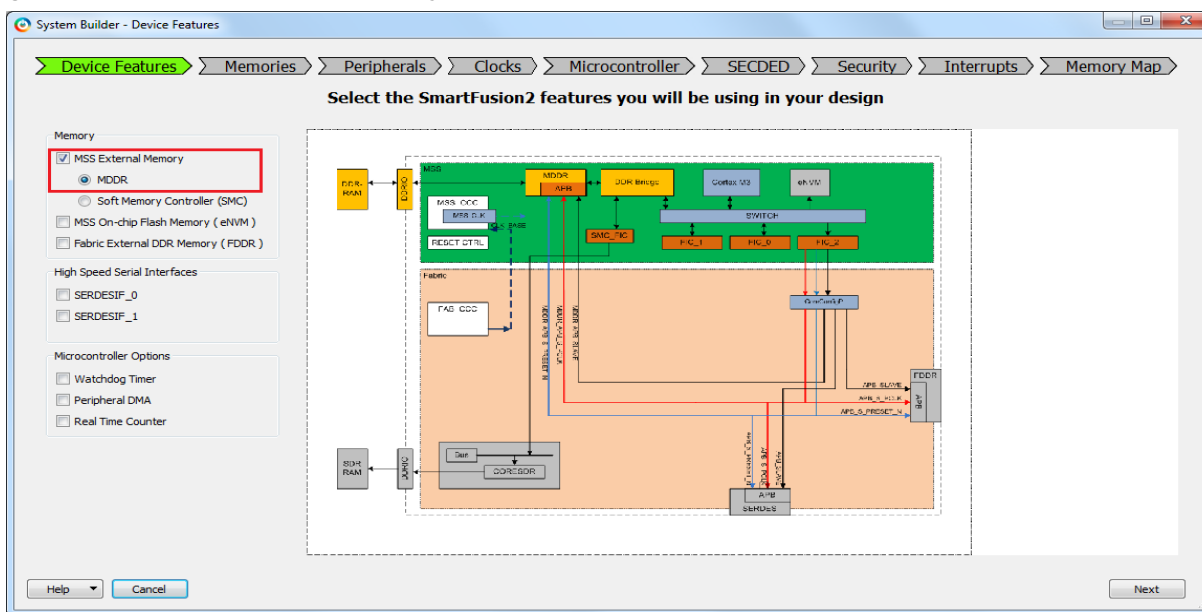
**Figure 101 • System Builder - Device Features Window**



The following steps describe how to configure the FDDR.

1. Check the **Fabric External Memory (FDDR)** check box under the **Device Features** tab and leave the other check boxes unchecked. The following image shows the **System Builder - Device Features** tab.

**Figure 102 • MSS External DDR Memory Selection**



2. Navigate to the **Memories** tab. Depending on the application requirement, select the memory settings under the **General** tab as shown in the following image.
  - Memory Type can be selected as DDR2, DDR3 or LPDDR.
  - The Data width can be selected as 32-bit, 16-bit, or 8-bit. Refer [Table 133 on page 152](#) for supported data widths for various SmartFusion2 device packages.
  - The SECCED (ECC) can be enabled or disabled.
  - Address Mapping - The register settings to perform mapping to system address bits for various Row, Bank and Column combinations are automatically computed by the configurator using address mapping option. [Table 160](#), page 185 shows the supported range for Row, Bank and Column.

**Table 160 • Supported Address Width Range for Row, Bank and Column Addressing in DDR/LPDDR**

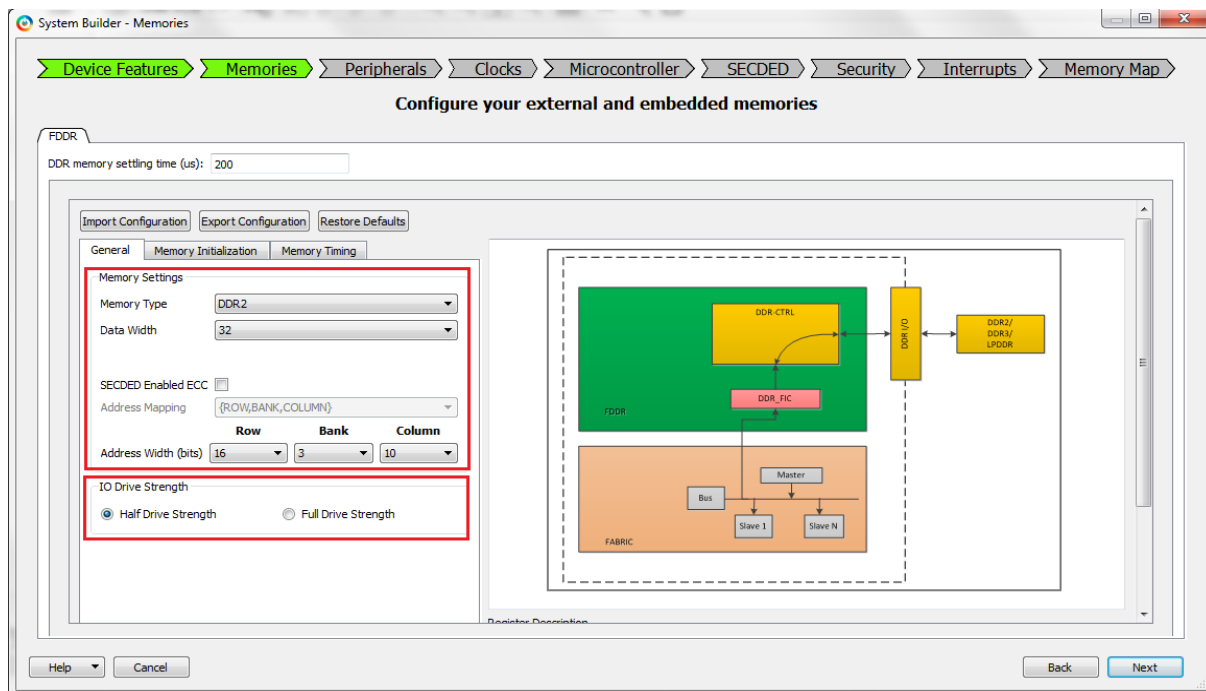
Width	DDR2	DDR3	LPDDR
Row Address	12-16	12-16	12-16
Bank Address	2-3	2-3	2-3
Column Address	9-12	9-12	9-12

- Select the **I/O Drive Strength** as **Half Drive Strength** or **Full Drive Strength** as shown in [Figure 3](#), page 17. The DDR I/O standard is configured as listed in [Table 139](#), page 159 based on this setting.

**Table 161 • DDR I/O Standard is Configured Based on I/O Drive Strength Setting**

I/O Drive Strength	Memory Type	
	DDR2	DDR3
Half Drive Strength	SSTL18I	SSTL15I
Full Drive Strength	SSTL18II	SSTL15II



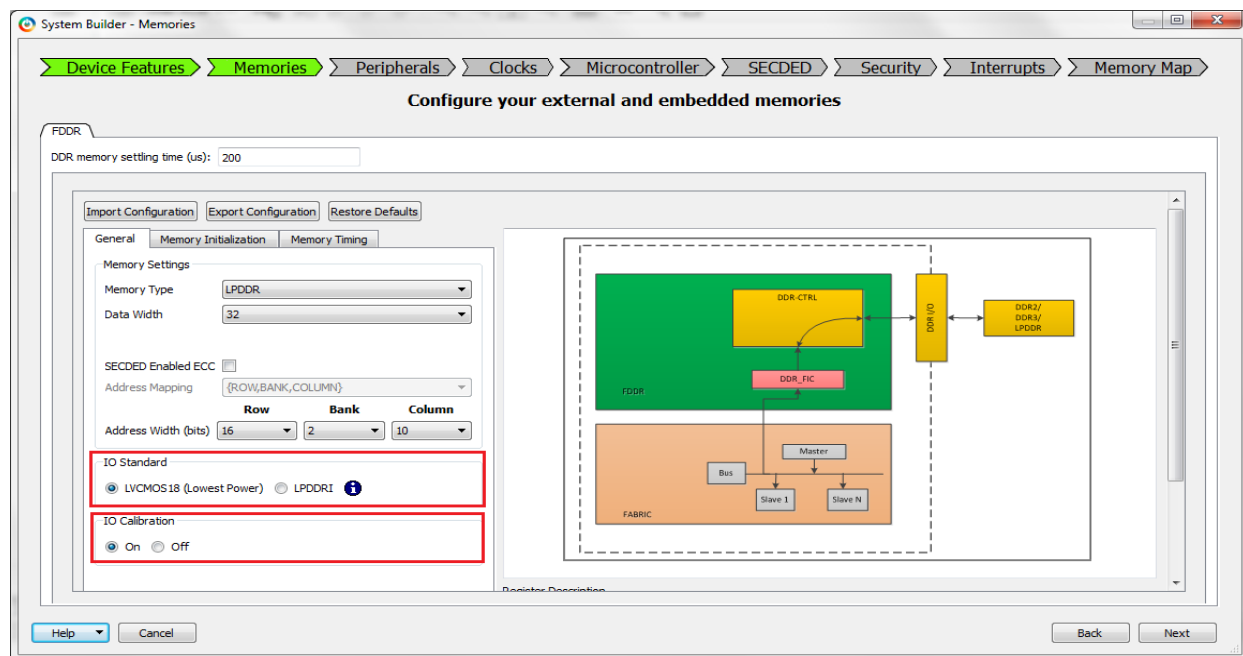
**Figure 103 • Fabric DDR Memory Settings**

- For only LPDDR memory, the **I/O standard** and **I/O calibration** settings are available as shown in [Figure 84 on page 160](#).

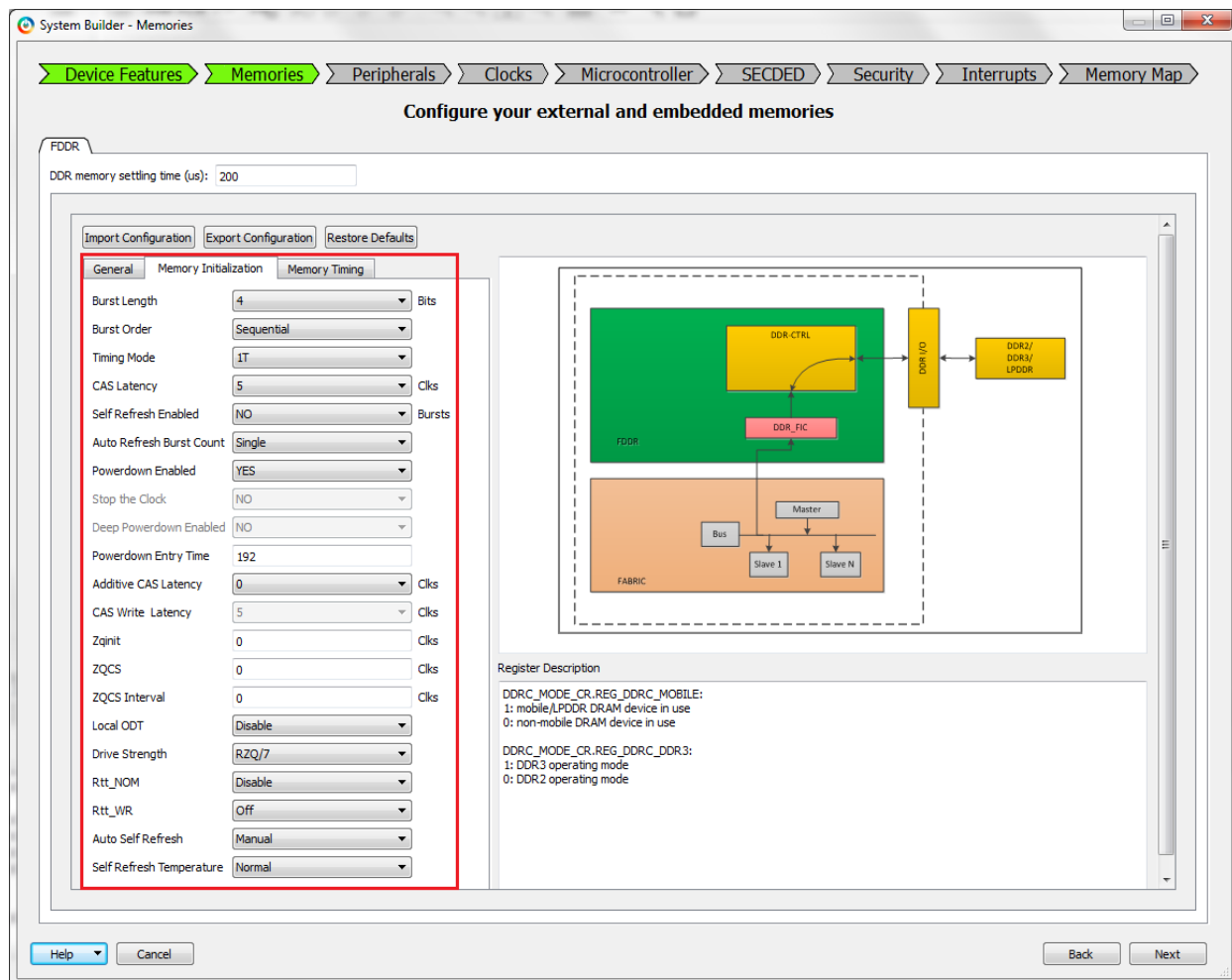
- Select **I/O standard** as **LVC MOS18** or **LPDDR1**.

**Note:** If LVC MOS18 is selected, all IOs are configured to LVC MOS1.8 except CLK/CLK\_N. CLK and CLK\_N are configured to LPDDR1 standard as they are differential signals.

- Select **I/O calibration** as ON or OFF. If I/O calibration is selected as ON, then the SmartFusion2 FDDR\_IMP\_CALIB pin must be pulled down with a resistor. For resistor values refer to Impedance Calibration section in [DS0115: SmartFusion2 Pin Descriptions Datasheet](#).

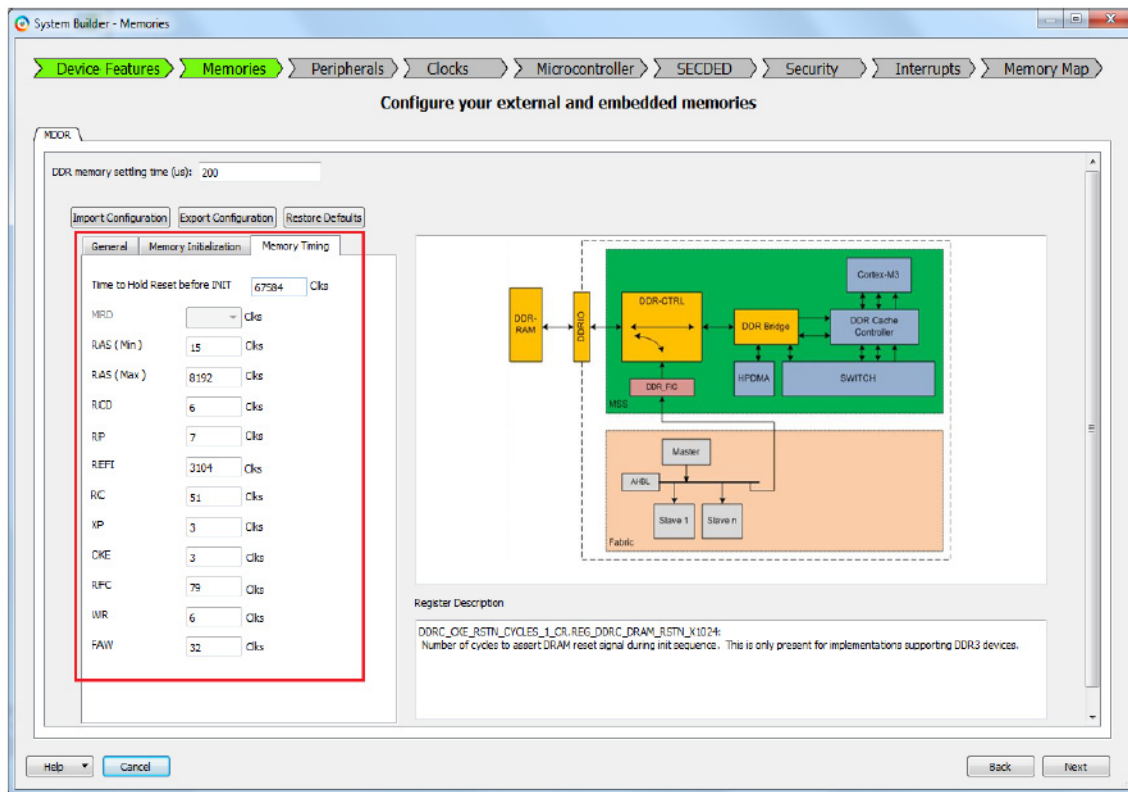
**Figure 104 • Selecting I/O Standard as LVC MOS18 or LPDDR1**

4. Depending on the application requirement; select the **Memory Initialization** settings under the **Memory Initialization** tab as shown in the following image.
  - Select the below performance related settings
    - Burst Length can be selected as 4, 8 or 16. [Table 133 on page 152](#) for supported burst lengths.
    - Burst order can be selected as sequential or interleaved. Refer [Table 133 on page 152](#) for supported burst orders.
    - Timing mode can be selected as 1T or 2T. For more details refer to ["1T or 2T Timing" section on page 156](#).
    - CAS latency is the delay, in clock cycles, between the internal READ command and the availability of the first bit of output data. Select the CAS latency according to the DDR memory (Mode register) datasheet.
  - Select the below power saving mode settings. Refer to ["Power Saving Modes" on page 150](#) for more details.
    - Self-Refresh Enabled
    - Auto Refresh Burst Count
    - Power down Enabled
    - Stop the clock: supported only for LPDDR
    - Deep Power down Enabled: supported only for LPDDR
    - Power down entry time
  - Select the additional performance settings.
    - Additive CAS Latency is defined by EMR[5:3] register of DDR2 memory and by MR1[4:3] register of DDR3 memory. It enables the DDR2 or DDR3 SDRAM to allow a READ or WRITE command from DDR Controller after the ACTIVATE command for the same bank prior to tRCD (MIN). This configuration is part of DDR2 Extended Mode register and DDR3 Mode register1.
    - CAS Write Latency (CWL) is defined by DDR3 MR2[5:3] and is the delay, in clock cycles, from the releasing of the internal write to the latching of the first data in. The overall WRITE latency (WL) is equal to CWL + AL, where CWL is set to 5 clock cycles by default.
  - Select the below ZQ Calibration settings for DDR3 memory. For more details refer ["ZQ Calibration" section on page 144](#).
    - Zqinit
    - ZQCS
    - ZQCS Interval
  - Select other settings.
    - Local ODT setting is not supported for LPDDR memory. For DDR2/DDR3 memory type, user can choose any option for "Local ODT". User can enable or disable "LOCAL ODT" during read transaction.
    - Drive strength setting is defined by EMR[7:5] register bits of LPDDR memory with drop down options of 'Full', 'Half', 'Quarter' and 'One-eighth' drive strength, it is defined by EMR[1] register bit of DDR2 memory with drop down options of 'Full' and 'Weak' drive strength and it is defined by MR1 register bits M5 and M1 of DDR3 memory with drop down options of 'RZQ/6' and 'RZQ/7'.
    - Partial array self-refresh coverage setting is defined by EMR[2:0] register bits of LPDDR memory with drop down options of 'Full', 'Quarter', 'One-eighth' and 'One-sixteenth'. This feature helps in improving power savings during self-refresh by selecting the amount of memory to be refreshed during self-refresh.
    - R<sub>TT</sub> (Nominal) setting is defined by EMR[6] and EMR[2] register bits of DDR2 memory which determines what ODT resistance is enabled with drop down options of 'RTT disabled', '50 ohms', '75  $\Omega$ ' and '150  $\Omega$ ' and it is defined by MR1[9], MR1[6] and MR1[2] register bits of DDR3 memory. In DDR3 memory RTT nominal termination is allowed during standby conditions and WRITE operations and NOT during READ operations with drop down options of 'RZQ/2', 'RZQ/4' and 'RZQ/6'.
    - R<sub>TT\_WR</sub> (Dynamic ODT) setting is defined by MR2[10:9] register bits of DDR3 memory. This is applicable only during WRITE operations. If dynamic ODT (R<sub>tt\_WR</sub>) is enabled, DRAM switches from normal ODT (R<sub>TT\_nom</sub>) to dynamic ODT (R<sub>tt\_WR</sub>) when beginning WRITE burst and subsequently switches back to normal ODT at the end of WRITE burst. The drop down options provided to the user are 'off', 'RZQ/4' and 'RZQ/2'.
    - Auto self-refresh setting is defined by MR2[6] register bit of DDR3 memory with drop down option of 'Manual' and 'Auto'. Self-refresh temperature setting is defined by MR2[7] register bit of DDR2 memory with drop down options of 'Normal' and 'Extended'.

**Figure 105 • DDR Memory initialization Settings**

5. Select the Memory Timing settings under the **Memory Timing** tab according to the DDR memory vendor datasheet as shown in the following image. For more details refer to "Configuring Dynamic DRAM Constraints" section on page 152.

**Figure 106 • DDR Memory Timing Settings**



The configurator also provides the option to import and export the register configurations.

Configuration files for accessing DDR3 memory on SmartFusion2 Development kit can be downloaded from [www.microsemi.com/soc/documents/FDDR3\\_16Bit\\_SB.zip](http://www.microsemi.com/soc/documents/FDDR3_16Bit_SB.zip).

Configuration files for accessing LPDDR memory on SmartFusion2 Starter kit can be downloaded from [www.microsemi.com/soc/documents/LPDDR\\_Emcrafft\\_Config.zip](http://www.microsemi.com/soc/documents/LPDDR_Emcrafft_Config.zip).

**Note:** The firmware generated by Libero SoC stores these configurations and the FDDR subsystem registers are initialized by the Cortex-M3 processor during the system\_init phase of the firmware projects (SoftConsole/IAR/Keil projects generated by Libero SoC).

An example of FDDR register configurations for operating the LPDDR memory (MT46H64M16LF) with clock 166 MHz is shown below.

Device Memory Settling Time (us): 200

The DDR memories require settling time for the memory to initialize before accessing it. the LPDDR memory model MT46H64M16LF needs 200us settling time.

#### General

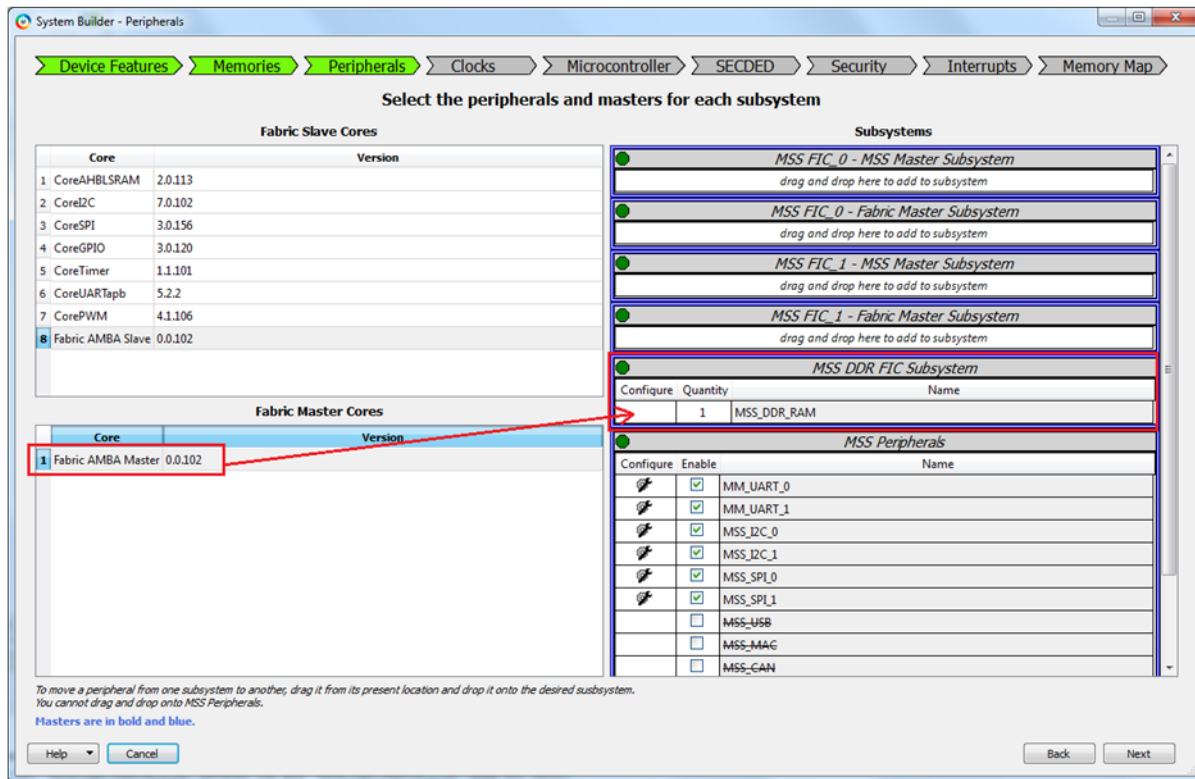
- Memory Type - Select LPDDR
- Data Width: 16
- Memory Initialization:
- Burst length - 8
- Burst Order: Interleaved
- Timing Mode: 1T
- CAS Latency: 3
- Self Refresh Enabled: No
- Auto Refresh Burst Count: 8
- PowerDown Enabled: Yes
- Stop the clock: No

- Deep PowerDown enabled: No
- No Activity clocks for Entry: 320

### Memory Timing

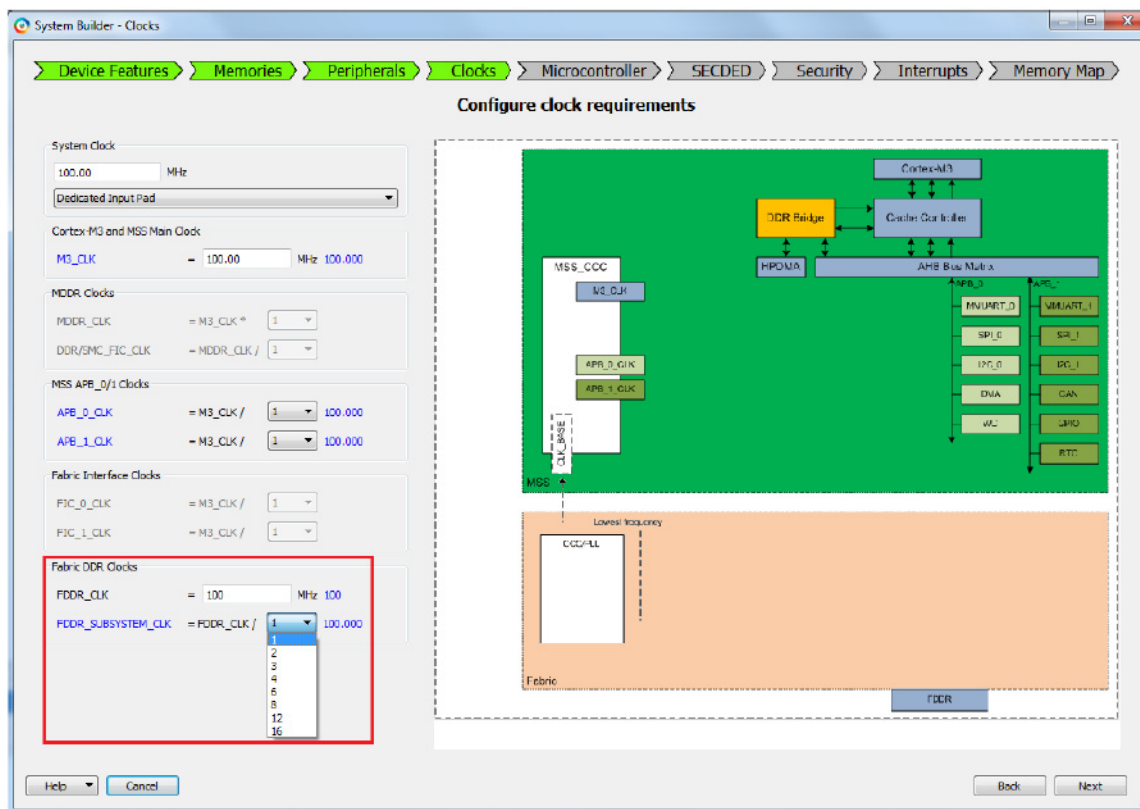
- Time To Hold Reset Before INIT - 67584 clks
  - MRD: 4 clks
  - RAS (Min): 8 clks
  - RAS (Max): 8192 clks
  - RCD: 6 clks
  - RP: 7 clks
  - REFI: 3104 clks
  - RC: 3 clks
  - XP: 3 clks
  - CKE: 3 clks
  - RFC: 79 clks
  - FAW: 0 clks
6. Navigate to the **Peripherals** tab. To access the FDDR from the FPGA fabric, drag and drop the **Fabric AMBA Master** to the **MSS DDR FIC Subsystem** and click **configure** to select the type of interface as AXI or single AHB-Lite. The user logic in the FPGA fabric can access the DDR memory through the FDDR using these interfaces. the following image shows the **Peripherals** tab.

**Figure 107 • MSS DDR FIC Subsystem Configuration**



7. Navigate to the **Clocks** tab. The **Clocks** tab allows to configure the system clock and subsystem clocks. The FDDR subsystem operates on FDDR\_CLK, which comes from MSS\_CCC. The FDDR\_CLK must be selected as multiples of 1, 2, 3, 4, 6 or 8-of M3\_CLK. The maximum frequency of FDDR\_CLK is 333.33 MHz.
- FDDR\_SUBSYSTEM\_CLK drives the DDR\_FIC slave interface and defines the frequency at which the FPGA fabric subsystem connected to this interface is intended to run. DDR\_FIC\_CLK can be configured as a ratio of FDDR\_CLK (1, 2, 3, 4, 6, 8, 12, or 16) using the Clocks configurator. The maximum frequency of FDDR\_SUBSYSTEM\_CLK is 200 MHz. The following image shows the FDDR\_CLK configuration.

**Figure 108 • FDDR Clock Configuration**

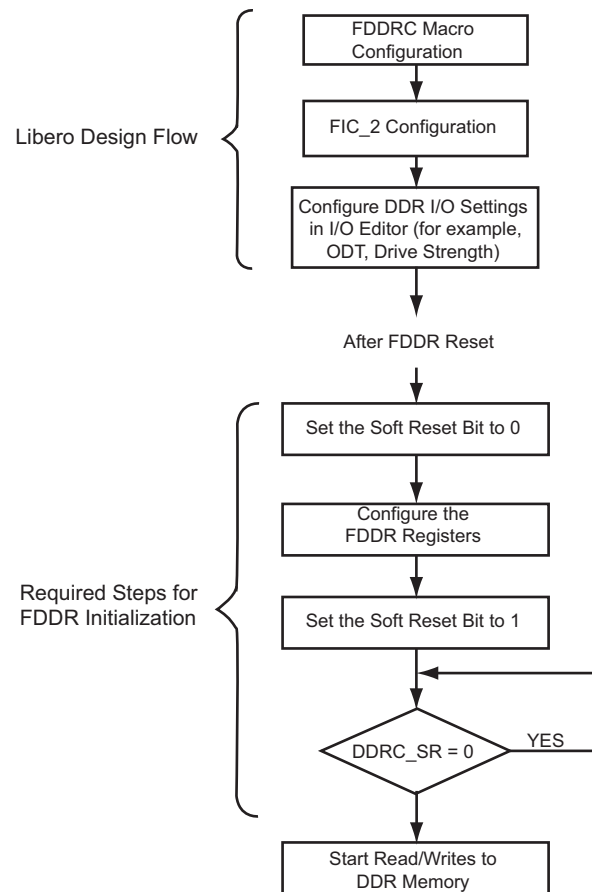


### 4.10.2 Design Flow Using SmartDesign

The following illustration shows the design flow for using the FDDR subsystem to access external DDR memory.

The design flow consists of two parts:

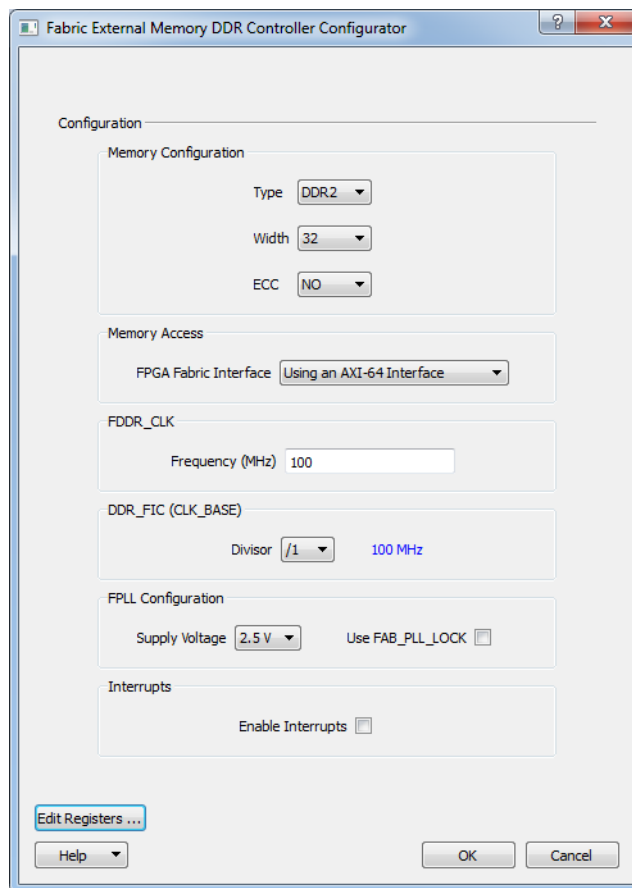
- **Libero flow:** This includes configuring the type of DDR memory, choosing fabric master interface type, clocking, and DDR I/O settings.
- **FDDR register initialization:** FDDR subsystem registers can be initialized using the ARM Cortex-M3 processor or FPGA fabric master. After MSS reset, the FDDR registers have to be configured according to application and DDR memory specification. The ["FDDR Subsystem Features Configuration" section on page 151](#) provides the details of required register configuration for FDDR features. While configuring the registers, the soft reset to the DDR controller must be asserted. After releasing the soft reset, the DDR controller performs DDR memory initialization and sets the status bits in DDRC\_SR.

**Figure 109 • Design Flow**

The configuration steps in the flow chart are explained in detail in the following sections.

#### 4.10.2.1 DDR Memory Controller Macro Configuration

The DDR Memory Controller macro in the Libero IP Catalog has to be instantiated in SmartDesign to access the external DDR memory through the DDR Memory Controller subsystem. The FDDRC macro configurator shown in the following image enables configuration of the DDR Memory Controller subsystem.

**Figure 110 • Fabric External Memory DDR Controller Configurator**

Depending on the application requirement, select the memory settings under the General tab as shown in the image.

- Memory Type can be selected as DDR2, DDR3 or LPDDR.
- The Data width can be selected as 32-bit, 16-bit, or 8-bit. Refer Table 1-13 for supported data widths for various SmartFusion2 device packages.
- Clock Frequency can be selected between 20 MHz to 333MHz. The FDDR subsystem operates on this clock (FDDR\_CLK) frequency
- The SECDED (ECC) can be enabled or disabled.

Select **FPGA Fabric Interface** type as AXI, single AHBLite, or two AHBLite. On completion of the configuration, the selected interface is exposed in SmartDesign. User logic in the FPGA fabric can access DDR memory through the FDDR using these interfaces.

The DDR\_FIC clock drives the DDR\_FIC slave interface and defines the frequency at which the FPGA fabric subsystem connected to this interface is intended to run. DDR\_FIC clock can be configured using FDDR CLOCK Divisor—1, 2, 3, 4, 6, 8, 12, or 16—of FDDR\_CLK. The maximum frequency of DDR\_FIC clock is 200 MHz. The DDR\_FIC clock has to be driven from FPGA fabric.

The FPLL LOCK signal can be exposed to the FPGA fabric to monitor the health of the PLL (loss of lock requires special handling by the application).

The interrupts in the FDDR subsystem can be exposed in SmartDesign by selecting the **Enable Interrupts** check box.

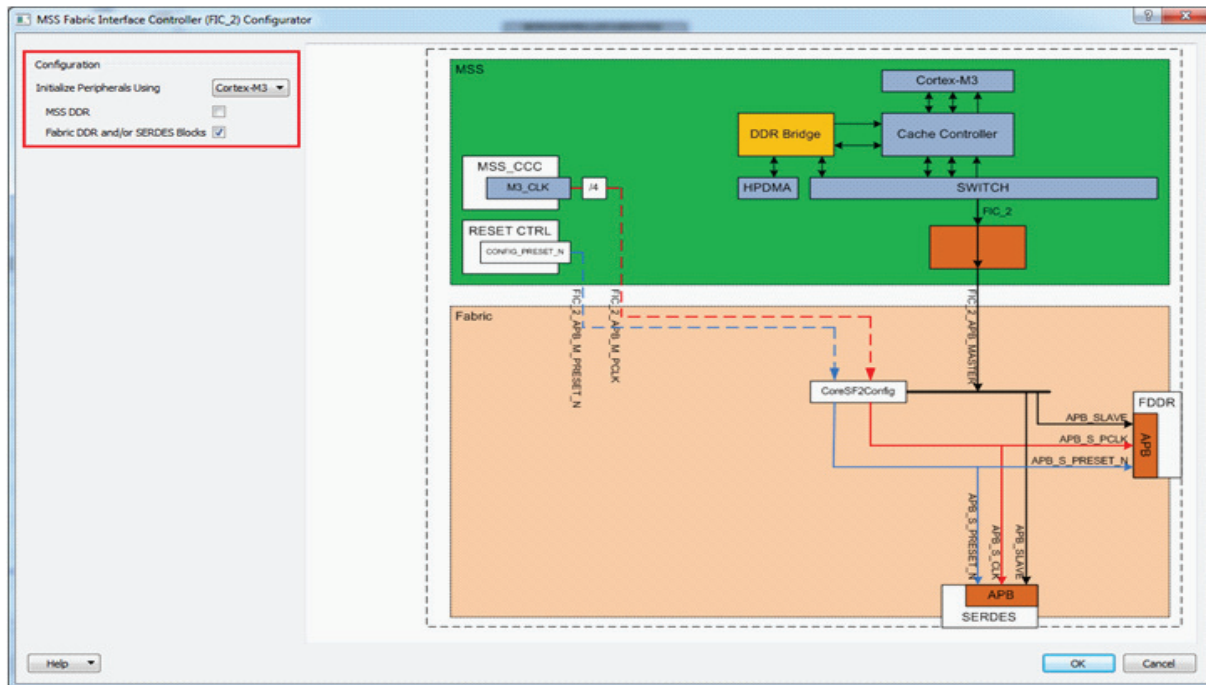
Select the memory settings under **Memory Initialization** tab and **Memory Timing** tab as described in the steps 3 and 4 in the "Design Flow Using System Builder" section on page 183.



#### 4.10.2.2 FIC\_2 Configuration

This is required for initializing the FDDR registers from Cortex-M3 processor. Configure the FIC\_2 (Peripheral Initialization) block as shown in the following image to expose the FIC\_2\_APB\_MASTER interface in Libero SmartDesign. CoreConfigP must be instantiated in SmartDesign and make the connections illustrated in the FIC\_2 Configurator. The following image shows the connectivity between the APB configuration interface and FDDR subsystem.

Figure 111 • FIC Configuration

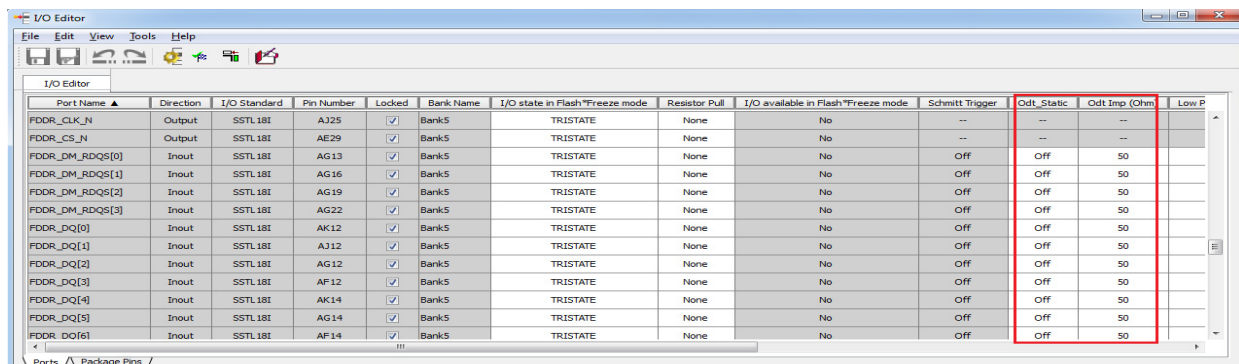


While enabling this option, the APB\_S\_PCLK and FIC\_2\_APB\_M\_PCLK signals are exposed in SmartDesign. The FDDR's APB\_S\_PCLK and APB\_S\_PRESET\_N have to be connected to FIC\_2\_APB\_M\_PCLK and FIC\_2\_APB\_M\_PRESET\_N. The FIC\_2\_APB\_M\_PCLK clock is generated from MSSCCC and is identical to M3\_CLK/4.

#### 4.10.2.3 I/O Configuration

I/O settings such as ODT and drive strength can be configured as shown in the following image using the I/O Editor in Libero SoC.

Figure 112 • I/O Configuration



Port Name	Direction	I/O Standard	Pin Number	Locked	Bank Name	I/O state in Flash/Freeze mode	Resistor Pull	I/O available in Flash/Freeze mode	Schmitt Trigger	Odt_Static	Odt Imp (Ohm)	Low P
FDDR_CLK_N	Output	SSTL18I	AJ25	✓	Bank5	TRISTATE	None	No	Off	Off	50	
FDDR_CS_N	Output	SSTL18I	AE29	✓	Bank5	TRISTATE	None	No	Off	Off	50	
FDDR_DM_RDQS[0]	Inout	SSTL18I	AG13	✓	Bank5	TRISTATE	None	No	Off	Off	50	
FDDR_DM_RDQS[1]	Inout	SSTL18I	AG16	✓	Bank5	TRISTATE	None	No	Off	Off	50	
FDDR_DM_RDQS[2]	Inout	SSTL18I	AG19	✓	Bank5	TRISTATE	None	No	Off	Off	50	
FDDR_DM_RDQS[3]	Inout	SSTL18I	AG22	✓	Bank5	TRISTATE	None	No	Off	Off	50	
FDDR_DQ[0]	Inout	SSTL18I	AK12	✓	Bank5	TRISTATE	None	No	Off	Off	50	
FDDR_DQ[1]	Inout	SSTL18I	AJ12	✓	Bank5	TRISTATE	None	No	Off	Off	50	
FDDR_DQ[2]	Inout	SSTL18I	AG12	✓	Bank5	TRISTATE	None	No	Off	Off	50	
FDDR_DQ[3]	Inout	SSTL18I	AF12	✓	Bank5	TRISTATE	None	No	Off	Off	50	
FDDR_DQ[4]	Inout	SSTL18I	AK14	✓	Bank5	TRISTATE	None	No	Off	Off	50	
FDDR_DQ[5]	Inout	SSTL18I	AG14	✓	Bank5	TRISTATE	None	No	Off	Off	50	
FDDR_DO[6]	Inout	SSTL18I	AF14	✓	Bank5	TRISTATE	None	No	Off	Off	50	

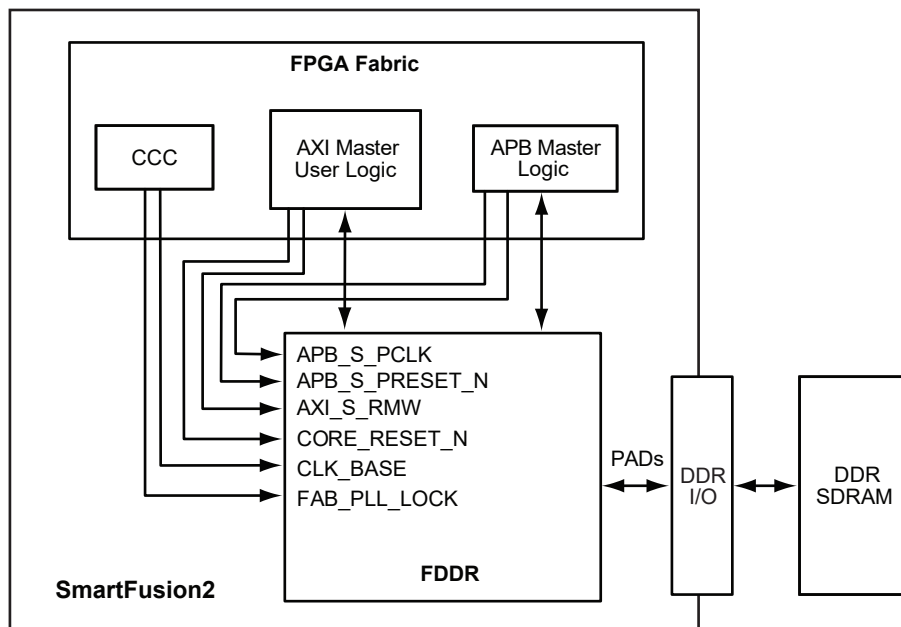
For more information about FDDR Subsystem Features Configuration, refer to "FDDR Subsystem Features Configuration" section on page 151.

### 4.10.3 Use Model 1: Accessing FDDR from FPGA Fabric Through AXI Interface

The AXI master in the FPGA fabric can access the DDR memory through the FDDR subsystem, as shown in the following illustration. The FDDR registers are configured from FPGA fabric through the APB interface. The APB master in the FPGA fabric asserts a ready signal to the AXI master, indicating successful initialization of the DDR memory.

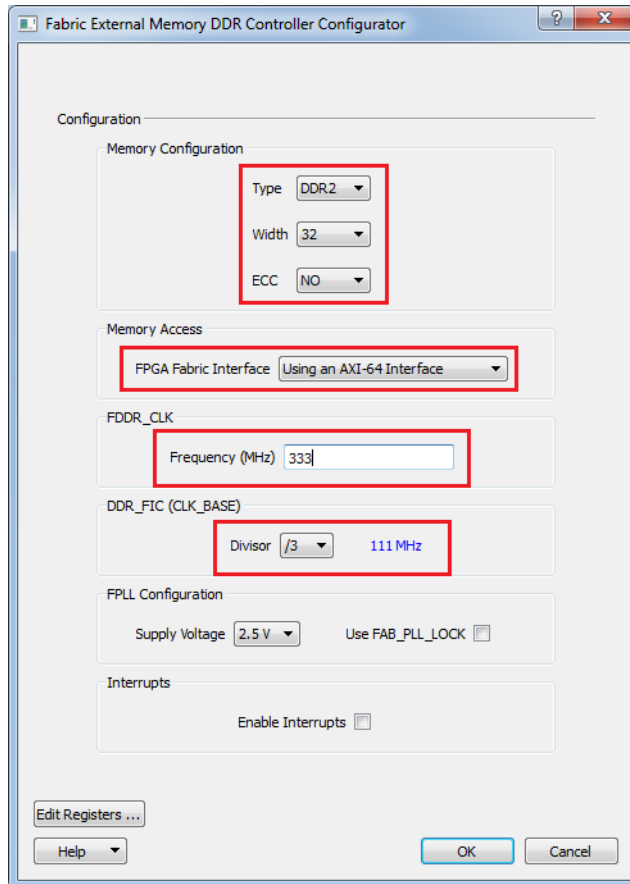
Read, write, and read-modify-write transactions are initiated by the AXI master to read or write the data into the DDR memory after receiving a ready signal from the APB master.

**Figure 113 • FDDR with AXI Interface**

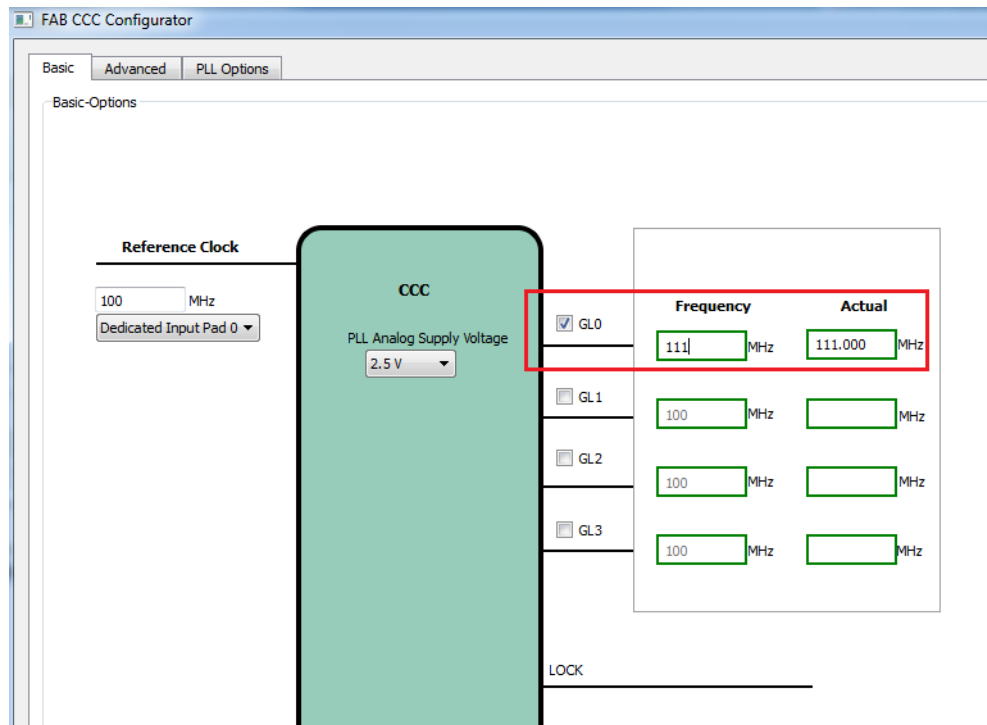


Use the following steps to access the FDDR from the AXI master in the FPGA fabric:

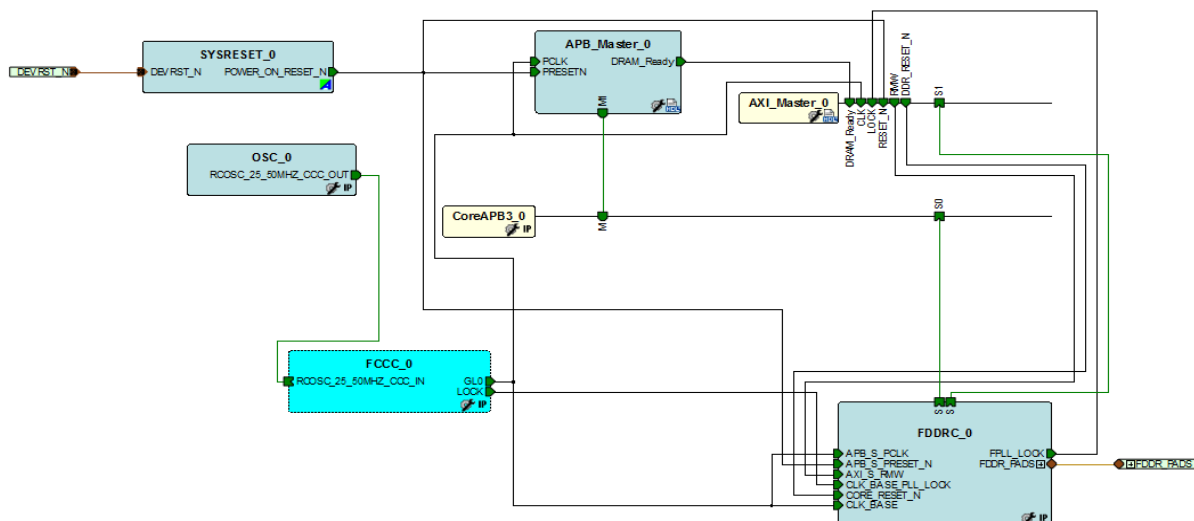
1. Instantiate the DDR Memory Controller macro in the SmartDesign canvas.
2. Configure the FDDR and select the AXI interface, as shown in the following image. In this example, the design is created to access DDR3 memory with a 32-bit data width. The FDDR clock is configured to 333 MHz and DDR\_FIC is configured to 111 MHz.

**Figure 114 • FDDR Configuration**

3. Instantiate the clock resources (FAB\_CCC and chip oscillators) in the SmartDesign canvas and configure, as required. In this example, the fabric CCC is configured to generate 111 MHz, as shown in the following image.

**Figure 115 • Fabric CCC Configuration**

4. Instantiate user AXI master logic in the SmartDesign canvas to access the FDDR through the AXI interface. Ensure that the AXI master logic accesses the FDDR after configuring the FDDR registers from the APB master. The AXI master clock frequency should be same as FDDR DDR\_FIC clock frequency.
5. Instantiate user APB master logic in the SmartDesign canvas to configure the FDDR registers through the APB interface.
6. Connect the AXI master to the FDDR AXI slave interface. Connect the APB master to the FDDR APB slave interface through CoreAPB.
7. Make the other connections in the SmartDesign canvas, as shown in the following image.

**Figure 116 • SmartDesign Canvas**

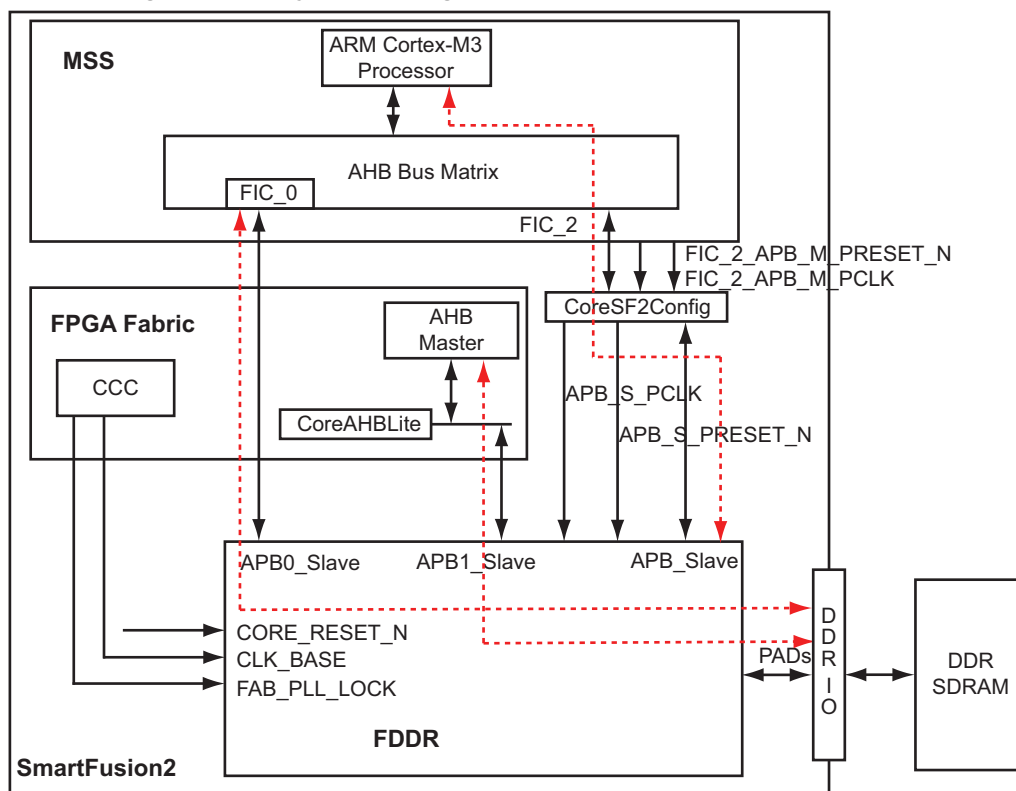
- To verify the design in Libero SoC, create a SmartDesign testbench project and instantiate a DDR memory model provided by the DDR memory vendor. Simulate the design and observe the AXI read and write transactions.

**Note:** The FDDR subsystem can be configured using the Cortex-M3 processor without having an APB master in the FPGA fabric. The System Builder can be used to create the design by following steps in ["Design Flow Using System Builder"](#). The System Builder provides "INIT\_DONE" to indicate that the DDR memory has been successfully initialized.

#### 4.10.4 Use Model 2: Accessing FDDR from FPGA Fabric Through AHB Interface

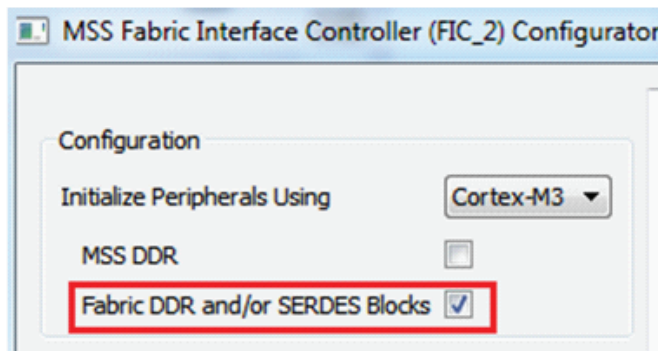
This use model shows an example of accessing DDR memory through the FDDR subsystem from two AHB masters (refer to the following illustration). FIC\_0 is used as AHB master 0 and user logic in the fabric is used as AHB master 1. The FDDR registers are configured from the Cortex-M3 processor through CoreConfigP. The read, write, and read-modify-write transactions are initiated by the AXI master to read or write the data into the DDR memory after receiving the ready signal from the APB master.

**Figure 117 • Accessing FDDR Subsystem Through Dual AHB Interface**

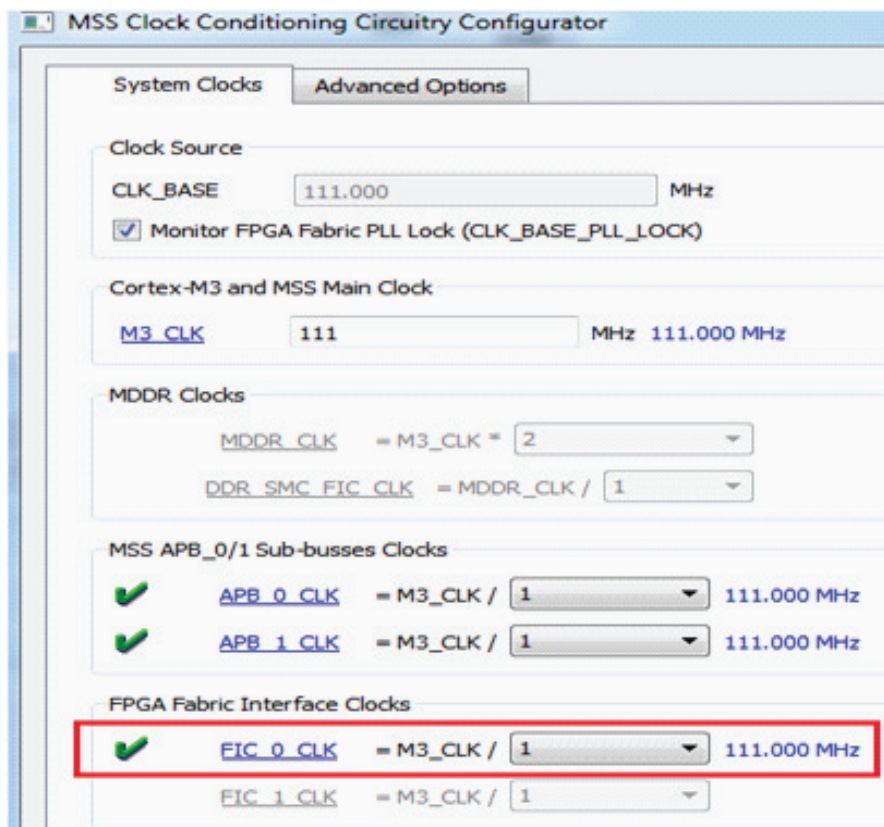


Use the following steps to access the FDDR from the AXI master in the FPGA fabric:

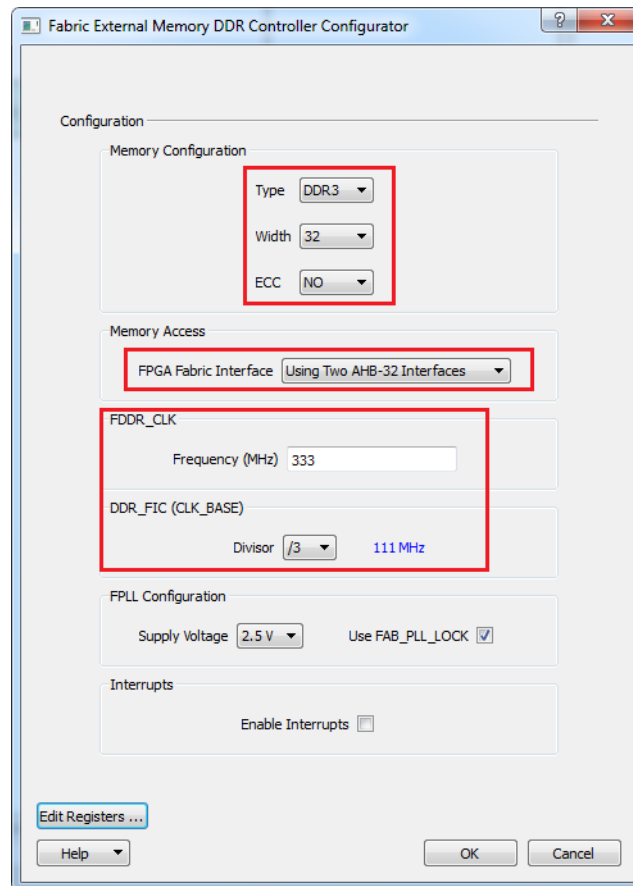
- Instantiate the SmartFusion2 MSS component in the SmartDesign canvas.
- Configure the SmartFusion2 MSS peripheral components as required using the MSS configurator. Configure FIC\_0 as the AHB master.
- Configure FIC\_2 to enable the FIC\_2 APB interface for configuring the FDDR subsystem registers from the Cortex-M3 processor, as shown in the following image.

**Figure 118 • FIC\_2 Configuration**

4. Configure MSSCCC for the FIC\_0 clock, as shown in the following image. The FIC\_0 clock is configured to 111 MHz.

**Figure 119 • MSS CCC Configuration**

5. Instantiate the DDR Memory Controller macro in the SmartDesign canvas.
6. Configure the FDDR and select the dual AHB interface, as shown in the following image. In this example, the design is created to access DDR3 memory with a 32-bit data width. The FDDR clock is configured to 333 MHz and DDR\_FIC is configured to 111 MHz.

**Figure 120 • FDDR Configuration**


**Fabric External Memory DDR Controller Configurator**

Configuration

Memory Configuration

Type: DDR3

Width: 32

ECC: NO

Memory Access

FPGA Fabric Interface: Using Two AHB-32 Interfaces

FDDR\_CLK

Frequency (MHz): 333

DDR\_FIC (CLK\_BASE)

Divisor: /3 111 MHz

FPLL Configuration

Supply Voltage: 2.5 V Use FAB\_PLL\_LOCK: ☒

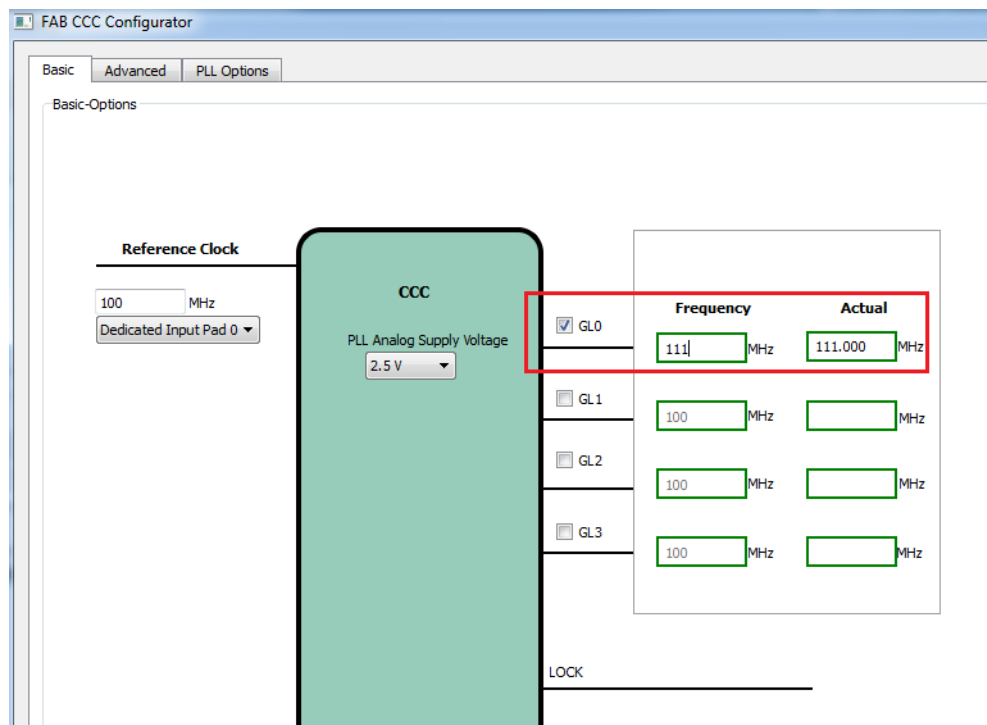
Interrupts

Enable Interrupts: ☐

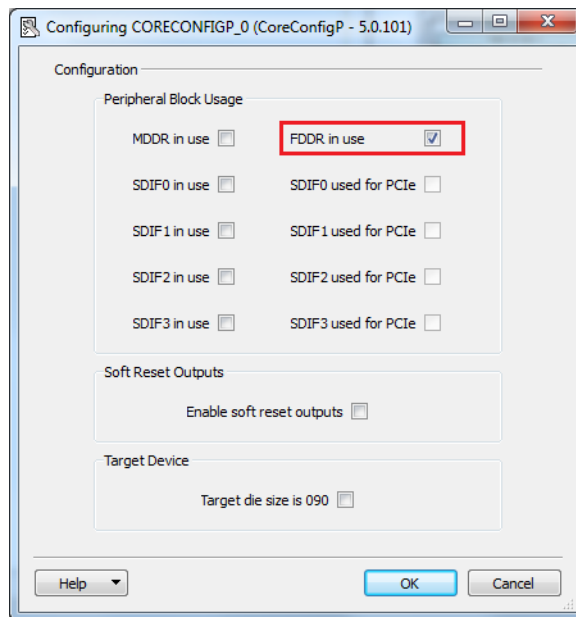
Edit Registers ...

Help OK Cancel

7. Depending on the application requirement select the memory settings. For more details refer to 3 and 4 in the ["Design Flow Using System Builder"](#).
8. Instantiate the clock resources (FCCC and chip oscillators) in the SmartDesign canvas and configure, as required. In this example, the fabric CCC is configured to generate 111 MHz, as shown in the following image.

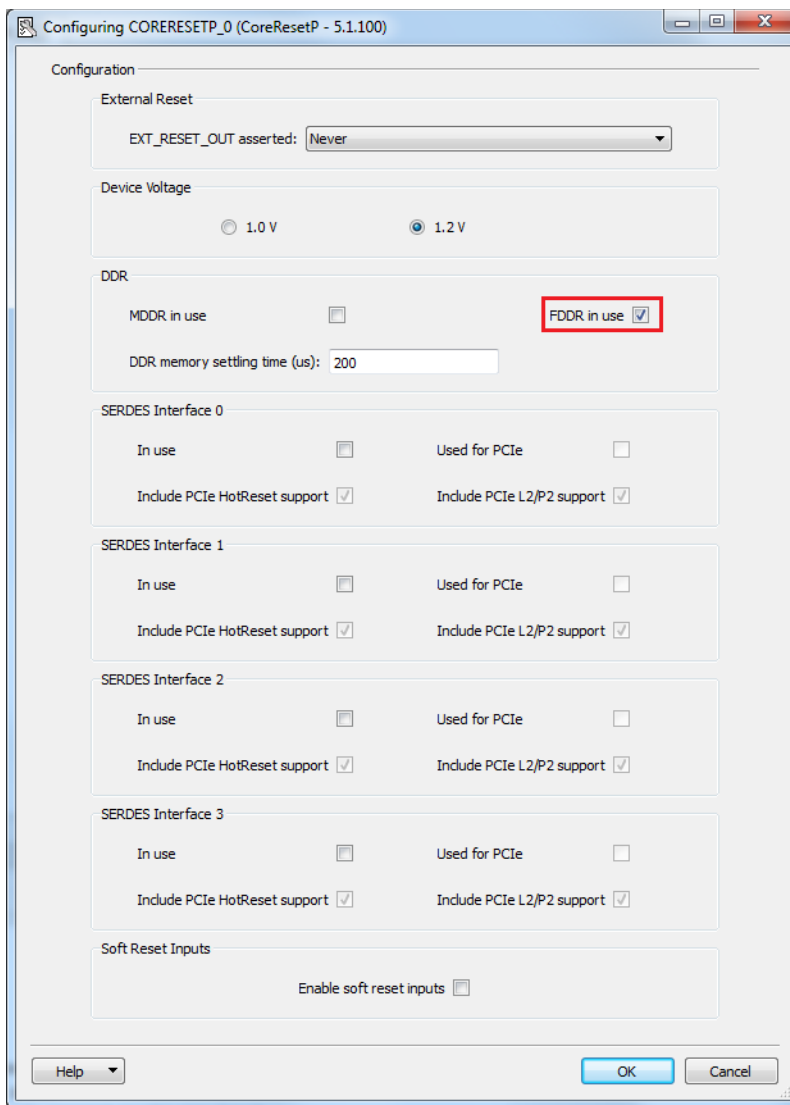
**Figure 121 • Fabric CCC Configuration**

9. Instantiate CoreConfigP in the SmartDesign canvas and configure for FDDR, as shown in the following image. Make the FIC\_2 and FDDR APB interface connections to CoreConfigP.

**Figure 122 • CoreConfigP IP Configuration**

10. Instantiate CoreResetP in the SmartDesign canvas and configure for FDDR, as shown in the following image. Make the connections to CoreResetP and CoreConfigP accordingly.



**Figure 123 • CoreConfigP IP Configuration**


Configuring CORERESETP\_0 (CoreResetP - 5.1.100)

**Configuration**

External Reset

EXT\_RESET\_OUT asserted:

Device Voltage

☐ 1.0 V ☒ 1.2 V

DDR

MDDR in use ☐ **FDDR in use ☒**

DDR memory settling time (us):

SERDES Interface 0

In use ☐ Used for PCIe ☐

Include PCIe HotReset support ☒ Include PCIe L2/P2 support ☒

SERDES Interface 1

In use ☐ Used for PCIe ☐

Include PCIe HotReset support ☒ Include PCIe L2/P2 support ☒

SERDES Interface 2

In use ☐ Used for PCIe ☐

Include PCIe HotReset support ☒ Include PCIe L2/P2 support ☒

SERDES Interface 3

In use ☐ Used for PCIe ☐

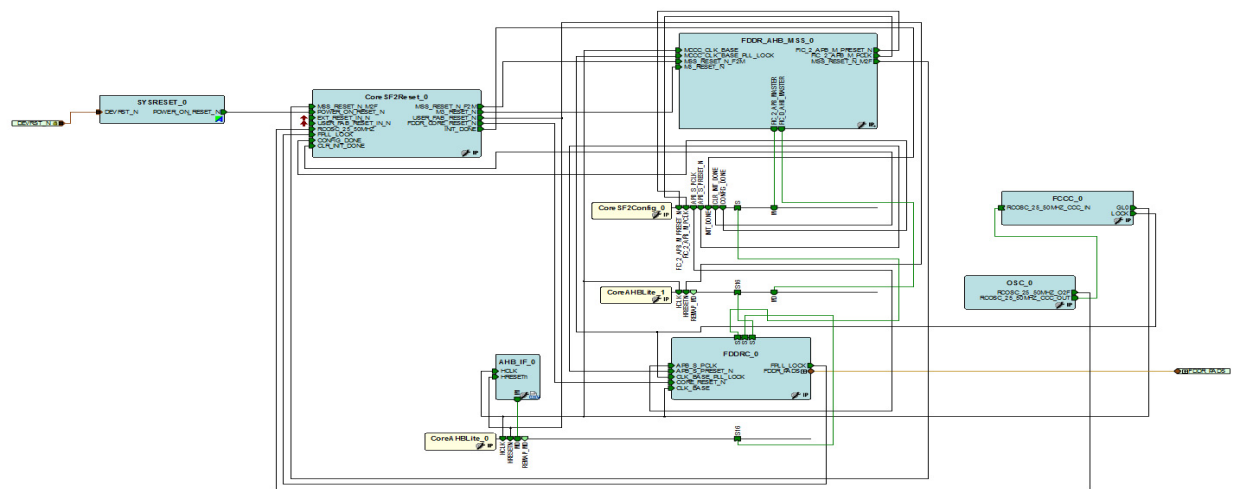
Include PCIe HotReset support ☒ Include PCIe L2/P2 support ☒

Soft Reset Inputs

Enable soft reset inputs ☐

Help OK Cancel

11. Instantiate user AHB master logic in the SmartDesign canvas to access the FDDR through the AHB interface. The AHB master clock frequency should be the same as the FDDR DDR\_FIC clock frequency.
12. Connect the AHB master to the FDDR AHB slave0 interface through CoreAHBLite. Connect the FIC\_0 master to the FDDR AHB slave1 interface through CoreAHBLite.
13. Make the other connections in the SmartDesign canvas, as shown in the following illustration.

**Figure 124 • SmartDesign Canvas**

14. To verify the design in Libero SoC, create a SmartDesign testbench project and instantiate a DDR memory model provided by the DDR memory vendor. Simulate the design and observe the AHB read and write transactions.

**Note:** Microsemi provides the System Builder tool to simplify design creation. To use System Builder, select **Use System Builder** while creating a new project from the Design Templates and Creators panel in Libero SoC. Follow the steps in the **System builder - Device Features** GUI and generate the design.

## 4.11 Appendix B: Register Lock Bits Configuration

The Register Lock Bits Configuration tool is used to lock MSS, SerDes, and FDDR configuration registers of SmartFusion2 devices to prevent them from being overwritten by masters that have access to these registers. Register lock bits are set in a text (\*.txt) file, which is then imported into the SmartFusion2 project.

### 4.11.1 Lock Bit File

An initial, default lock bit file can be generated by clicking **Generate FPGA Array Data** in the **Design Flow** window.

The default file located at <proj\_location>/designer/<root>/<root>\_init\_config\_lock\_bits.txt can be used to make the required changes.

**Note:** Save the file using a different name if you modify the text file to set the lock bits.

### 4.11.2 Lock Bit File Syntax

A valid entry in the lock bit configuration file is defined as a <lock\_parameters> <lock bit value> pair format.

The lock parameters are structured as follows:

- Lock bits syntax for a register: <Physical block name>\_<register name>\_LOCK
- Lock bits syntax for a specific field: <Physical block name>\_<register name>\_<field name>\_LOCK

The following are the physical block names (varies with device family and die):

- MSS
- FDDR
- SERDES\_IF\_x (where x is 0,1,2,3 to indicate the physical SERDES location) for SmartFusion2 and IGLOO2 (010/025/050/150) devices
- SERDES\_IF2 for SmartFusion2 and IGLOO2 (060/090) devices (only one SERDES block per device)

Set the lock bit value to 1 to indicate that the register can be written to (unlocked) and to 0 to indicate that the register cannot be written to (locked).

Lines starting with # or ; are comments. Empty lines are allowed in the lock bit configuration file.

The following figure shows the lock bit configuration file.

**Figure 125 • Lock Bit Configuration File**

```
# Register Lock Bits Configuration File for MSS, SERDES(s) and Fabric DDR
# Microsemi Corporation - Microsemi Libero Software Release v11.7 SP1 (Version 11.7.1.2)
# Date: Tue Mar 29 13:24:54 2016

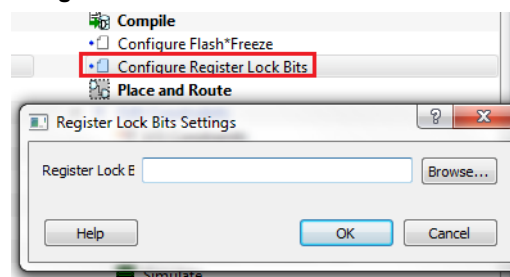
# sb_sb_0/sb_sb_MSS_0/MSS_ADLIB_INST/INST_MSS_050_IP
MSS_ESRAM_CONFIG_LOCK          0
MSS_ESRAM_MAX_LAT_LOCK         1
MSS_DDR_CONFIG_LOCK            1
MSS_ENVM_CONFIG_LOCK           0
MSS_ENVM_REMAP_BASE_LOCK       1
MSS_ENVM_FAB_REMAP_LOCK        1
MSS_CC_CONFIG_LOCK             0
MSS_CC_CACHEREGION_LOCK        1
MSS_CC_LOCKBASEADDR_LOCK       1
MSS_CC_FLUSHINDX_LOCK          0
MSS_DDRB_BUF_TIMER_LOCK        1
MSS_DDRB_NB_ADR_LOCK           1
MSS_DDRB_NB_SIZE_LOCK          0
MSS_DDRB_CONFIG_LOCK           1
MSS_EDAC_ENABLE_LOCK           1
MSS_MASTER_WEIGHT_CONFIG0_LOCK 1
MSS_MASTER_WEIGHT_CONFIG1_LOCK 1
MSS_SOFT_INTERRUPT_LOCK        1
MSS_SOFTRESET_ENVM0_SOFTRESET_LOCK 1
MSS_SOFTRESET_ENVM1_SOFTRESET_LOCK 1
MSS_SOFTRESET_ESRAM0_SOFTRESET_LOCK 1
MSS_SOFTRESET_ESRAM1_SOFTRESET_LOCK 1
MSS_SOFTRESET_MAC_SOFTRESET_LOCK 1
MSS_SOFTRESET_PDMA_SOFTRESET_LOCK 1
MSS_SOFTRESET_TIMER_SOFTRESET_LOCK 1
MSS_SOFTRESET_MMUART0_SOFTRESET_LOCK 1
MSS_SOFTRESET_MMUART1_SOFTRESET_LOCK 1
MSS_SOFTRESET_G4SPI0_SOFTRESET_LOCK 1
MSS_SOFTRESET_G4SPI1_SOFTRESET_LOCK 1
MSS_SOFTRESET_I2C0_SOFTRESET_LOCK 1
MSS_SOFTRESET_I2C1_SOFTRESET_LOCK 1
MSS_SOFTRESET_CAN_SOFTRESET_LOCK 1
MSS_SOFTRESET_USB_SOFTRESET_LOCK 1
MSS_SOFTRESET_COMBLK_SOFTRESET_LOCK 1
MSS_SOFTRESET_FPGA_SOFTRESET_LOCK 1
MSS_SOFTRESET_HPDMA_SOFTRESET_LOCK 1
MSS_SOFTRESET_FIC32_0_SOFTRESET_LOCK 1
MSS_SOFTRESET_FIC32_1_SOFTRESET_LOCK 1
MSS_SOFTRESET_MSS_GPIO_SOFTRESET_LOCK 1
MSS_SOFTRESET_MSS_GPOUT_7_0_SOFT_RESET_LOCK 1
MSS_SOFTRESET_MSS_GPOUT_15_8_SOFT_RESET_LOCK 1
MSS_SOFTRESET_MSS_GPOUT_23_16_SOFT_RESET_LOCK 1
MSS_SOFTRESET_MSS_GPOUT_31_24_SOFT_RESET_LOCK 1
MSS_SOFTRESET_MDDR_CTLR_SOFTRESET_LOCK 1
MSS_SOFTRESET_MDDR_FIC64_SOFTRESET_LOCK 1
MSS_M3_CONFIG_LOCK            1
```

### 4.11.3 Locking and Unlocking a Register

A register can be locked or unlocked by setting the appropriate lock bit value in the lock bit configuration .txt file.

1. Browse to locate the lock bit configuration .txt file.
2. Do one or both of the following:
  - Set the lock bit value to 0 for the registers you want to lock.
  - Set the lock bit value to 1 for the registers you want to unlock.
3. Save the file, and import the file into the project (**Design Flow** window > **Configure Register Lock Bits**).

**Figure 126 • Register Lock Bit Settings Window**

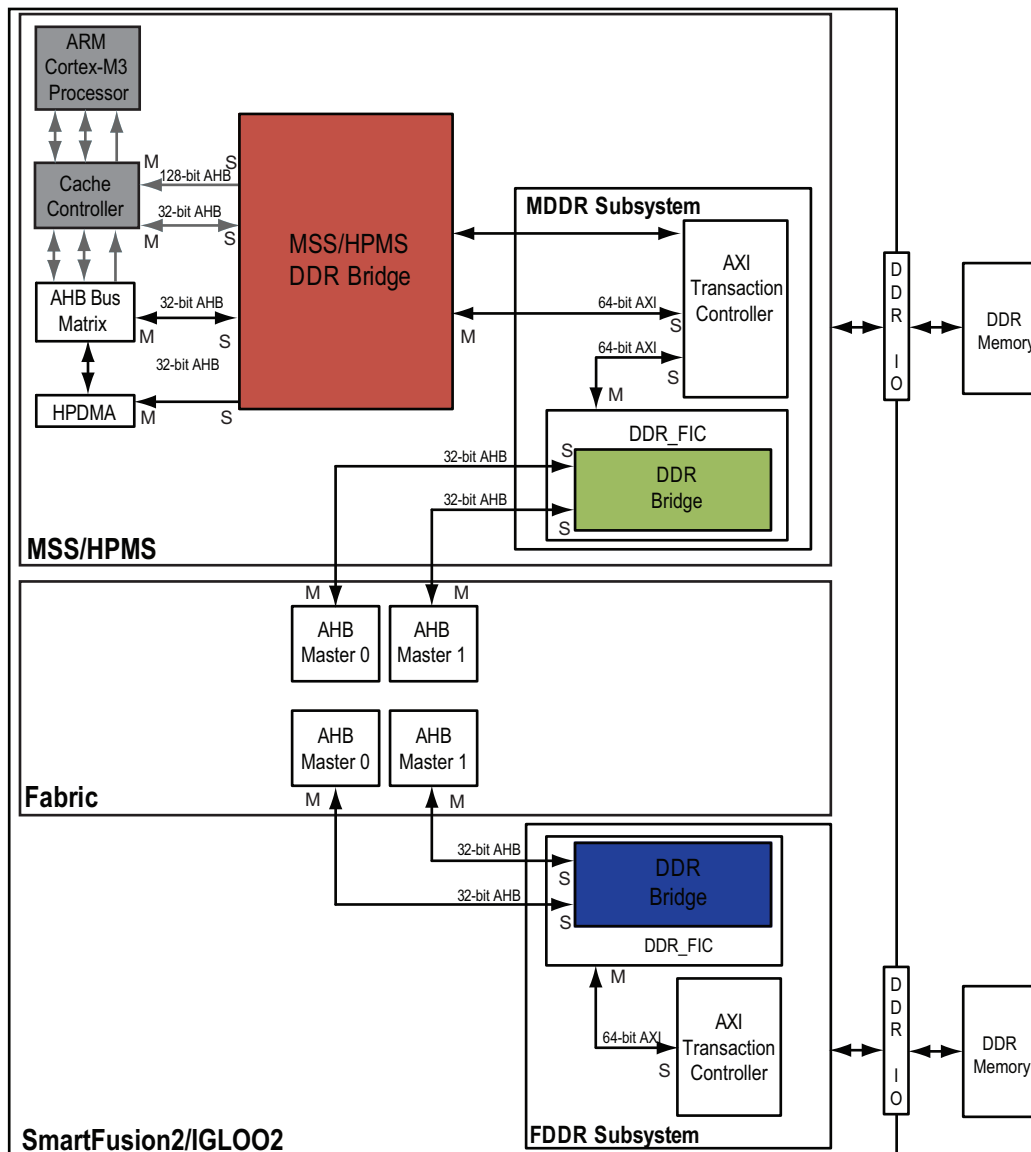


4. Regenerate the bitstream.

## 5 DDR Bridge

The DDR bridge facilitates multiple AHB bus masters to access a single AXI slave and optimizes read and write operations from multiple AHB masters to a single external DDR memory. The SmartFusion2 and IGLOO2 devices have three instances of the DDR bridge, one each for the MSS/HPMS, FDDR, and MDDR subsystems, as shown in the following image. The DDR bridge implemented in the MSS/HPMS (shown in red) provides an interface between AHB masters within the MSS/HPMS for accessing DDR memory. The DDR bridge implemented in the MDDR subsystem (shown in green) provides an interface between the user implemented AHB masters in the FPGA fabric for accessing DDR memory. Similarly, the DDR bridge in the FDDR (shown in blue) subsystem facilitates fabric masters to access DDR memory.

**Figure 127 • DDR Bridges in the SmartFusion2/IGLOO2 FPGA Device**



**Note:** Grey blocks and arrows indicate the steps happen only in MSS. Rest are same in SmartFusion2 and IGLOO2.

The DDR bridge supports a single 64-bit AXI and up to four 32-bit AHB interfaces. For SmartFusion2 the four MSS AHB masters are fixed, as shown in the following table. For Igloo2, the two HPMS AHB

masters are fixed, as shown in the following table. The DDR bridges in the MDDR and FDDR subsystems support only two AHB interfaces out of four and these can be used for user implemented AHB masters.

**Table 162 • SmartFusion2 and IGLOO2 FPGA DDR Bridge Interface**

Sub-System	DDR Bridge				
	AHB Interface 0 Read Only	AHB Interface 1 R/W	AHB Interface 2 R/W	AHB Interface 3 R/W	AXI Interface
HPMS	Not available—	Not available—	AHB bus matrix	HPDMA	MDDR subsystem
MSS	Cache Controller IDC	Cache Controller DS	AHB bus matrix	HPDMA	MDDR subsystem
MDDR	Not available	Not available	AHB master interface 0	AHB master interface 1	MDDR subsystem
FDDR	Not available	Not available	AHB master interface 0	AHB master interface 1	FDDR subsystem

**Note:** If the AXI bus is selected as the interface between the FPGA fabric and the MDDR/ FDDR subsystem, the DDR bridge in these subsystems is not used.

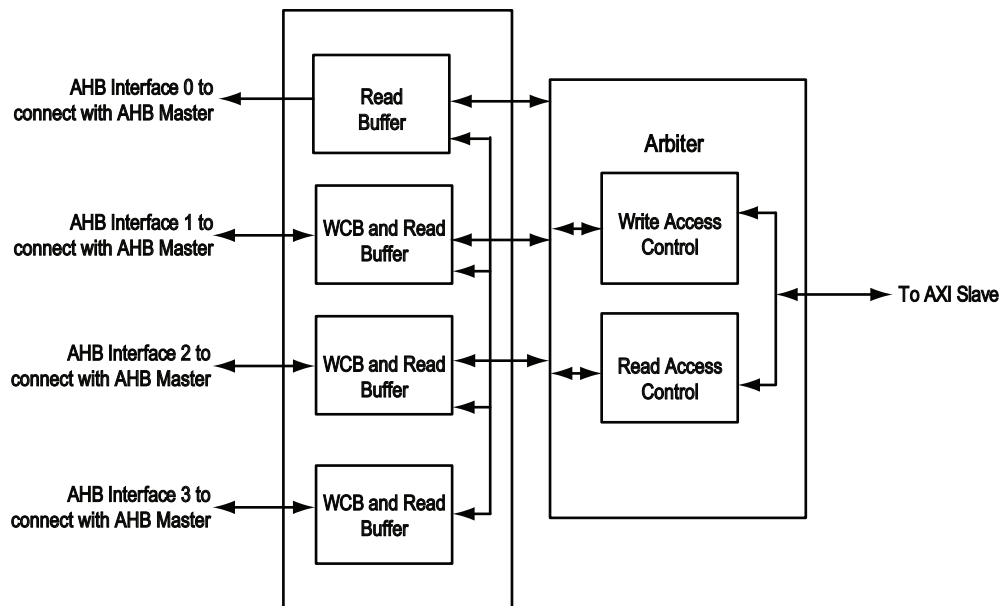
## 5.1 Functional Description

This section provides the detailed description of the DDR Bridge, which contains the following sections:

- [Architecture Overview](#)
- Details of Operation

### 5.1.1 Architecture Overview

The DDR bridge consists of two main components: read and write combining buffers (WCB), and an arbiter, as shown in the following illustration. The DDR bridge buffers AHB write transactions into write combining buffers before bursting out to external DDR memory. It also includes read buffers for AHB masters to efficiently read data from the external DDR memory. All buffers within the DDR bridge are implemented with latches and hence are not subject to single event upsets (SEUs). The external DDR memory regions can be configured to be non-bufferable. If a master interface requests a write or read to a non-bufferable region, the DDR bridge is essentially bypassed. The size of the non-bufferable address space can also be configured.

**Figure 128 • DDR Bridge Functional Block Diagram**

Arbitration between the four AHB interfaces is handled as follows:

- Fixed priority between AHB Interfaces 0 and 1, with 0 having the highest priority
- Round robin arbitration between interfaces 2 and 3

## 5.1.2 Details of Operation

This section provides a functional description of each block in the DDR Bridge, as shown in the previous illustration.

### 5.1.2.1 Write Combining Buffer

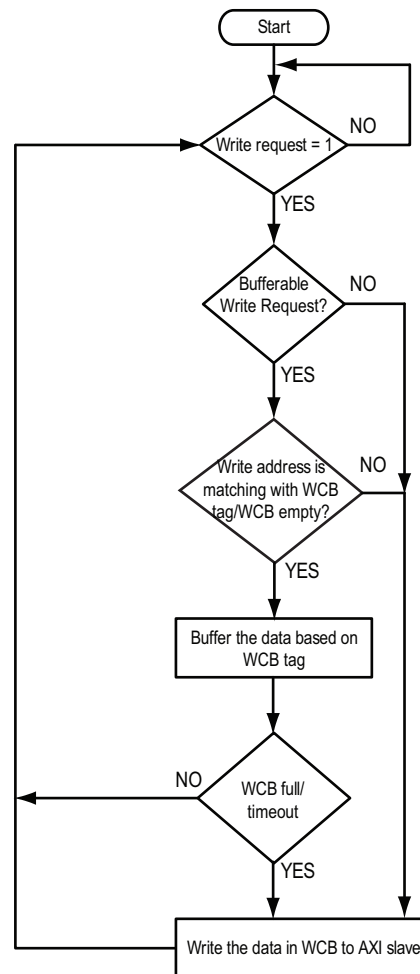
The write combining buffer (WCB) combines multiple write transactions from the AHB master into AXI burst transactions. The WCB has a user configurable burst size of 16 or 32 bytes. Each WCB maintains a base address tag that stores the base address of the data to be combined in the buffer.

For each write transaction, the address is compared with the WCB tag. If the address matches the tag, data is combined into the buffer. The WCB writes to the correct byte location based on the offset address of the data. WCB can also be disabled, if buffering is not required.

The WCB has a 10-bit timer (down counter), which starts when the first bufferable write data is loaded into the WCB. The timer starts decrementing its value at every positive edge of the AHB clock and when it reaches zero, the data in the WCB is written to the AXI slave.

The WCB checks for any other master that has initiated a read to the same address for which data is already present in a write buffer or for which a write operation is ongoing. If the address for a read request matches the write buffer tag, the read request is held until the buffer is written completely to the AXI slave.

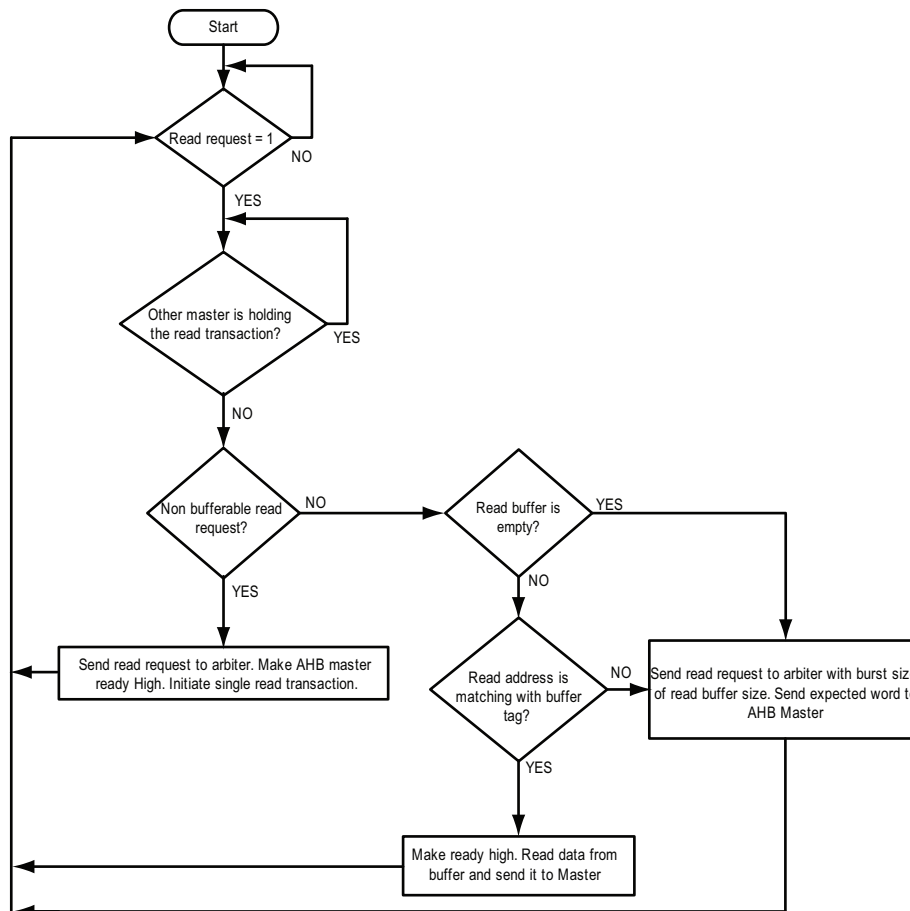
the following illustration shows the flowchart for WCB operation.

**Figure 129 • WCB Operation**

### 5.1.2.2 Read Buffer

The DDR bridge has a read buffer for each master to hold the fetched DDR burst data. Each read buffer has a configurable burst size of 16 or 32 bytes. The read buffer initiates a DDR burst size request for reads in the bufferable region, regardless of the size of request from the master. Each read buffer is associated with one specific master for reading; it does not check the read addresses of other masters to determine whether that data can be read from the read buffer—there is no cross buffer read access. The following illustration shows the flow chart for read operation.



**Figure 130 • Flow Chart for Read Operation**

The read buffer is invalidated under the following conditions:

- If the address from the master is outside the TAG region, the current data in the read buffer is invalidated (TAG mismatch).
- To ensure proper data coherency, every master's write address is tracked. If an address matches that of the read buffer TAG, the read entry is invalidated.
- A non-bufferable or locked transaction is initiated by any master.
- An Invalidate command is issued.
- A buffer disable command is issued.
- An error response from DDR for the expected word read.

### 5.1.2.3 Arbiter

The DDR bridge arbiter includes two independent arbitration controllers for read and write requests.

#### 5.1.2.3.1 Write Access Controller

The write access controller (WAC) arbitrates write requests from the WCBs and grants access to one of the requesting masters based on its priority. All transactions from a single master have a dedicated master ID.

Combinations of fixed and round robin priorities are assigned to the following masters:

- Master Interface 1: Fixed first priority (Master Interface 0 is read only)
- Round robin between Master Interface 2 and Master Interface 3 for second and third priorities

Once a burst transaction is initiated to the external DDR memory, the transactions are completed without an interruption. No other master, even a high priority master, can interrupt this process. Subsequent write requests from the same master are held until the previous write transactions are completed to the external DDR memory. Subsequent write requests from other masters can be accepted and allowed to

write into WCB, but the DDR bridge does not write this data until the previous write transactions are completed to the external DDR memory.

#### 5.1.2.3.2 Read Access Controller

The read access controller (RAC) arbitrates read requests from read buffers and grants access to one of the requesting masters depending on its priority.

Combinations of fixed and round robin priorities are assigned to the masters as below:

- Master Interface 0 and Master Interface 1 have fixed first and second priority
- Round robin between Master Interface 2 and Master Interface 3 for second and third priority

The RAC also routes the read data from the AXI slave (MDDR or FDDR) to the corresponding master based on the Read data ID.

#### 5.1.2.3.3 Locked Transactions

The DDR bridge masters can initiate locked transfers by asserting the HMASTLOCK signal of the corresponding AHB interface. These locked transactions are initiated only after all the pending write and read transactions are completed.

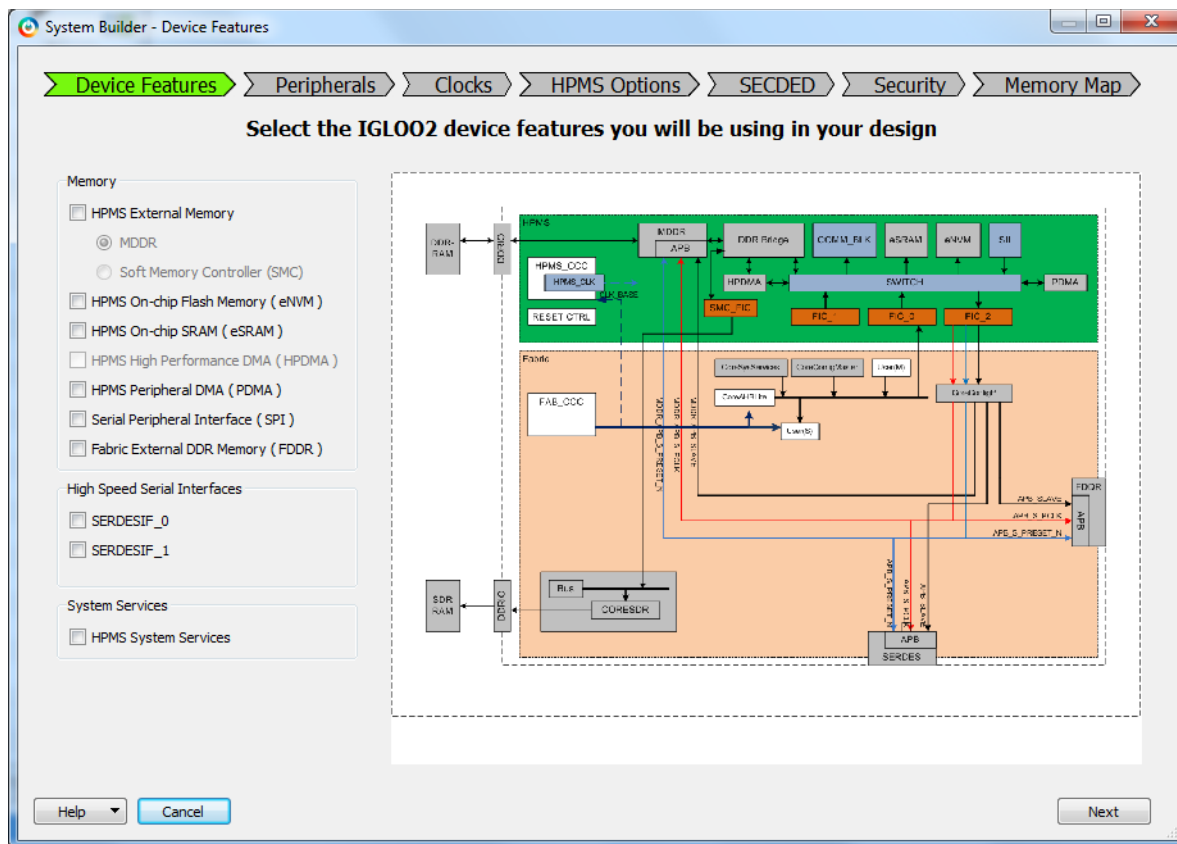
The arbiter has a 20-bit up counter for detecting a lock timeout condition. The counter starts counting when a locked transaction is initiated on the bus. When the counter reaches its maximum value, an interrupt is generated. The interrupt can be cleared by setting the DDR\_LOCKOUT bit in the MSS\_EXTERNAL\_SR from the SYSREG block. In SmartFusion2 When the counter reaches its maximum value, an interrupt is generated to the Cortex-M3 processor. The error routine has to be stored in either eNVM or eSRAM for the Cortex-M3 processor to fetch the interrupt service routine (ISR) without going through the DDR bridge. As part of the ISR, the Cortex-M3 processor reads the SYSREG registers to identify the master and take appropriate action to release the arbiter from dead lock. If the interrupt is cleared and the lock signal is still asserted, the counter will start counting again.

## 5.2 How to Use DDR Bridge in IGLOO2 Device

This section describes how to use DDR bridge. To configure the IGLOO2 device features and then build a complete system, use the **System Builder** graphical design wizard in the Libero SoC software.

The following illustration shows the initial **System Builder** window where you can select the features that you require. For details on how to launch the **System Builder** wizard and a detailed information on how to use it, refer the *IGLOO2 System Builder User's Guide*.

**Figure 131 • System Builder - Device Features Window**



Navigate to the **HPMS Options** tab in the **System Builder** wizard.

For more information about how to use MDDR in the SmartFusion2 devices, refer to ["Appendix A: How to Use DDR Bridge in SmartFusion2 Device"](#) section on page 215.

### 5.2.1 Configuring the DDR Bridge

The following sections describe the configuration options for the HPMS DDR Bridge.

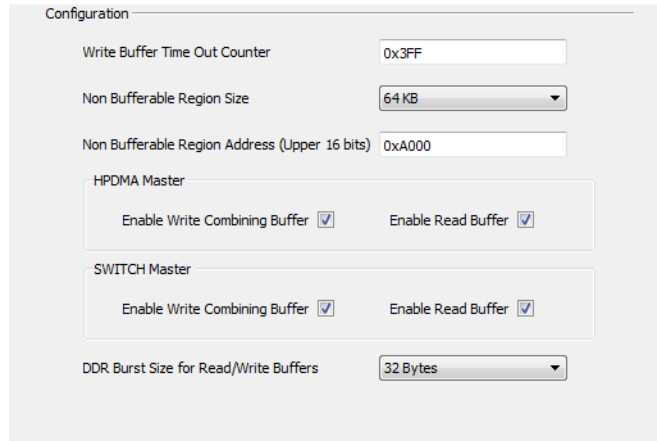
### 5.2.1.1 HPMS DDR Bridge Configurations

The HPMS DDR bridge is statically configured through the DDR bridge configurator in the System Builder wizard. The following illustration shows the configuration options.

- **Write buffer time out counter:** This allows the user to configure the 10-bit timer of write buffer for time out value. By default this is configured for maximum wait time (0x3FF) to buffer the write transactions. For configuring to other values enter a 10-bit hexadecimal value in the provided field of DDR bridge configurator. Select timeout value to a non zero value for buffering the write transactions.
  - **Non-bufferable region size:** The size of non-bufferable memory region can be selected from a drop-down menu in the DDR bridge configurator. The menu has the options to select the region from 64 KB to 1 GB. It also has an option **None** to select the complete memory as bufferable. The default selection is 64 KB.
  - **Non-bufferable region address:** The base address of the non-bufferable memory region can be selected by configuring this field. The value must be configured as a 16-bit hexadecimal address. The default address is 0xA000. If the non-bufferable region size and address is left as default then the 64 KB memory from 0xA0000000 address to 0xA0010000 address will be non-bufferable.
5. Enable or disable respective buffers allocated for each master. The selection of disabling the write/read buffer makes all the transactions without buffering. By default buffering is enabled. Select the DDR burst size for

read/write buffers. The DDR bridge configurator allows to select the size of read/write buffers as 32 bytes or 16 bytes.

**Figure 132 • Configuring HPMS DDR Bridge for HPDMA**



### 5.2.1.2 Configurations for the DDR Bridge in the MDDR or FDDR Subsystems

The DDR bridge in the MDDR or FDDR subsystem can be configured using the DDR\_FIC registers listed in [Table 164 on page 215](#). The possible configurations and corresponding registers are:

- Enable or disable the write and read buffers of the DDR bridge using the DDR\_FIC\_HPD\_SW\_RW\_EN\_CR register
- Configure buffer size to 32 bytes or 16 bytes using the DDR\_FIC\_NBRWB\_SIZE\_CR register
- Configure the non-bufferable address using the DDR\_FIC\_NB\_ADD register
- Configure the non-bufferable size using the DDR\_FIC\_NBRWB\_SIZE\_CR register
- Configure the timeout value for each write buffer using the DDR\_FIC\_LOCK\_TIMEOUTVAL\_1\_CR and DDR\_FIC\_LOCK\_TIMEOUTVAL\_2\_CR registers.

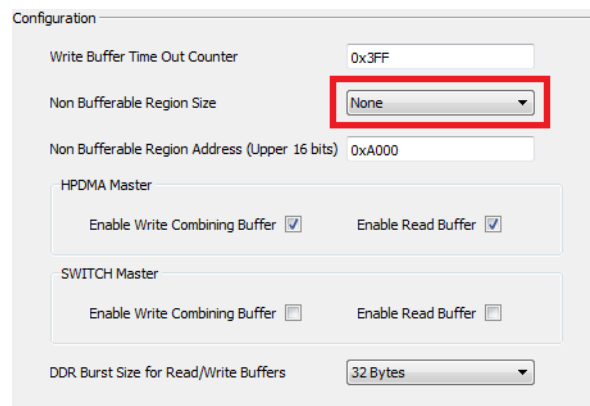
Set the timeout value to maximum or a non- zero value.

The configuration registers for the MDDR bridge and FDDR bridge are also listed under the "[DDR\\_FIC Configuration Registers Summary](#)" section on [page 102](#) section in the MDDR and FDDR chapters.

## 5.2.2 High-Speed Data Transactions from HPDMA

This section describes the use of the DDR bridge to increase the throughput from the HPDMA to the external DDR memories. The HPDMA performs only the single read and write transactions and not the burst transactions. The DDR bridge converts these single transactions into burst transactions and further increases the throughput. The HPDMA buffers are enabled for this, and the non-bufferable size is selected as **None**, as shown in the following image.

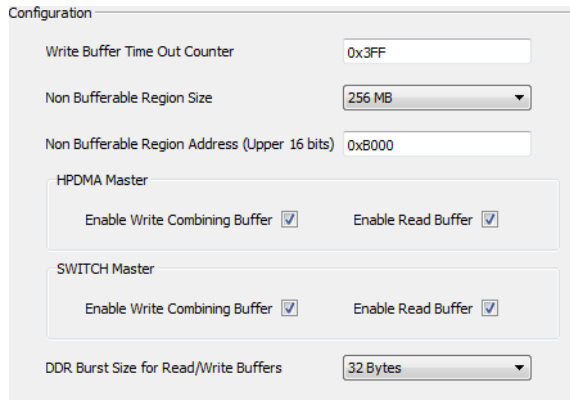
**Figure 133 • Configuring HPMS DDR Bridge For Non-Bufferable Region**



### 5.2.3 Selecting Non-Bufferable Region

This section describes the use of the non-bufferable region selection in the DDR bridge. The buffering creates more latency in the applications which access non-continuous memory locations. In such cases non-bufferable region selection provides high throughput than bufferable. The application uses only 256 MB of memory segment (0xB000\_0000 to 0XBFFF\_FFFF) as non-bufferable and the other memory region as bufferable. The following image shows the selection of the non-bufferable region.

**Figure 134 • Configuring DDR Bridge**



## 5.3 SYSREG Control Registers

The following table lists HPMS DDR bridge Control registers in the SYSREG block. Refer to the **System Register Map** chapter of the *UG0448: IGLOO2 High Performance Memory Subsystem User Guide* for a detailed description of each register and bit.

**Table 163 • SYSREG Control Registers**

Register Name	Register Type	Flash Write Protect	Reset Source	Description
DDRB_BUF_TIMER_CR	RW-P	Register	SYSRESET_N	Uses a 10-bit timer interface to configure the timeout register in the write buffer module.
DDRB_NB_ADDR_CR	RW-P	Register	SYSRESET_N	Indicates the base address of the non-bufferable address region.
DDRB_NB_SIZE_CR	RW-P	Register	SYSRESET_N	Indicates the size of the non-bufferable address region.
DDRB_CR	RW-P	Register	SYSRESET_N	HPMS DDR bridge configuration register
DDRB_HPD_ERR_ADR_SR	RO	–	SYSRESET_N	HPMS DDR bridge high performance DMA master error address status register
DDRB_SW_ERR_ADR_SR	RO	–	SYSRESET_N	HPMS DDR bridge switch error address status register
DDRB_BUF_EMPTY_SR	RO	–	SYSRESET_N	HPMS DDR bridge buffer empty status register
DDRB_DSBL_DN_SR	RO	–	SYSRESET_N	HPMS DDR bridge disable buffer status register
DDRB_STATUS	RO	–	SYSRESET_N	Indicates HPMS DDR bridge status
MSS_EXTERNAL_SR	SW1C	–	SYSRESET_N	HPMS external status register

**Table 163 • SYSREG Control Registers (continued)**

Register Name	Register Type	Flash Write Protect	Reset Source	Description
MSSDDR_FACC1_CR	RW-P	Field	CC_RESET_N	HPMS DDR fabric alignment clock controller 1 configuration register.

## 5.4 DDR Bridge Control Registers in MDDR and FDDR

The following table lists HPMS DDR bridge control registers in the MDDR and FDDR. Refer to the "MDDR Subsystem" chapter on page 5 and the "Fabric DDR Subsystem" chapter on page 133 for a detailed description of each register and bit.

**Table 164 • DDR Bridge Control Registers in MDDR and FDDR**

Register Name	Address Offset	R/W	Reset Source	Description
DDR_FIC_NB_ADDR_CR, Table 104, page 104	0x400	RW	PRESET_N	Indicates the base address of the non-bufferable address region.
DDR_FIC_NBRWB_SIZE_CR, Table 105, page 104	0x404	RW	PRESET_N	Indicates the size of the non-bufferable address region.
DDR_FIC_BUF_TIMER_CR, Table 106, page 104	0x408	RW	PRESET_N	10-bit timer interface used to configure the timeout register.
DDR_FIC_HP_D_SW_RW_EN_CR, Table 107, page 105	0x40C	RW	PRESET_N	Enable write buffer and read buffer register for AHB-Lite (AHBL) master1 and master2.
DDR_FIC_HP_D_SW_RW_INVAL_CR, Table 108, page 105	0x410	RW	PRESET_N	Invalidates write buffer and read buffer for AHBL master1 and master2.
DDR_LOCK_TIMEOUTVAL_1_CR, Table 119, page 110	0x440	RW	PRESET_N	Indicates maximum number of cycles a master can hold the bus for a locked transfer.
DDR_LOCK_TIMEOUTVAL_2_CR, Table 120, page 110	0x444	RW	PRESET_N	Indicates maximum number of cycles a master can hold the bus for a locked transfer.

## 5.5 Appendix A: How to Use DDR Bridge in SmartFusion2 Device

This section describes how to use DDR Bridge in an application and contains the following sub-sections:

- MSS DDR Bridge Configurations
- Use Model 1: High Speed Data Transactions from Cortex-M3 Processor
- Use Model 2: Selecting Non-Bufferable Region

### 5.5.0.1 MSS DDR Bridge Configurations

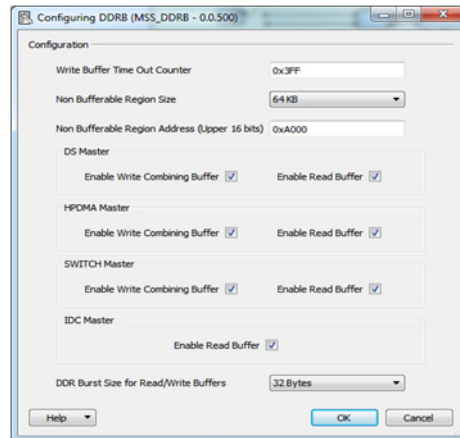
The MSS DDR bridge is statically configured through the DDR bridge configurator of the MSS configurator in Libero SoC, as shown in the following image. Configurable parameters are as follows:

- **Write buffer time out counter:** This allows to configure the 10-bit timer of write buffer for time out value. By default this is configured for maximum wait time (0x3FF) to buffer the write transactions. For configuring to other values enter a 10-bit hexadecimal value in the provided field of DDR bridge configurator. Select timeout value to a non zero value for buffering the write transactions.
- **Non-bufferable region size:** The size of non-bufferable memory region can be selected from a drop-down menu in the DDR bridge configurator. The menu has the options to select the region from 64 KB to 1 GB. It also has an option "none" to select the complete memory as bufferable. The default selection is 64 KB.
- **Non-bufferable region address:** The base address of the non-bufferable memory region can be selected by configuring this field. The value must be configured as a 16-bit hexadecimal address.

The default address is 0xA000. If the non-bufferable region size and address is left as default then the 64 KB memory from 0xA0000000 address to 0xA0010000 address will be non-bufferable.

- **Enable or disable respective buffers allocated for each master:** The selection of disabling the write/read buffer makes all transactions without buffering. By default buffering is enabled.
- **DDR burst size for read/write buffers:** The DDR bridge configurator allows to select the size of read/write buffers as 32 bytes or 16 bytes.

**Figure 135 • Configuring MSS DDR Bridge**



### 5.5.0.2 MDDR/FDDR DDR Bridge Configurations

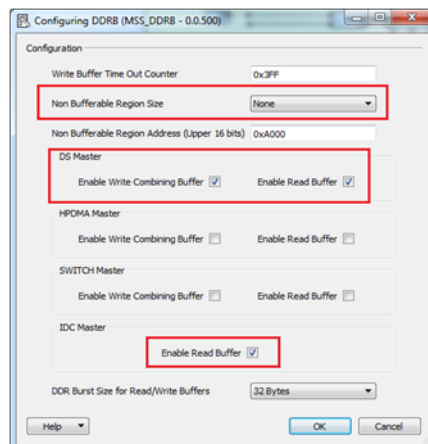
The DDR bridge in the MDDR or FDDR subsystem can be configured through the DDR\_FIC registers shown in [Table 164 on page 215](#). The possible configurations and corresponding registers are as follows:

- Enable or disable the write and read buffers of the DDR bridge using the DDR\_FIC\_HPD\_SW\_RW\_EN\_CR register.
- Configure buffer size to 32 bytes or 16 bytes using the DDR\_FIC\_NBRWB\_SIZE\_CR register.
- Configure the non-bufferable address using the DDR\_FIC\_NB\_ADD register.
- Configure the non-bufferable size using the DDR\_FIC\_NBRWB\_SIZE\_CR register.
- Configure the timeout value for each write buffer using the DDR\_FIC\_LOCK\_TIMEOUTVAL\_1\_CR and DDR\_FIC\_LOCK\_TIMEOUTVAL\_2\_CR registers. Set the timeout value to maximum or a non-zero value.

The configuration registers for the MDDR DDR bridge and FDDR DDR bridge are also listed under the **DDR FIC registers** section in the **MDDR and FDDR chapters**.

### 5.5.1 Use Model 1: High Speed Data Transactions from Cortex-M3 Processor

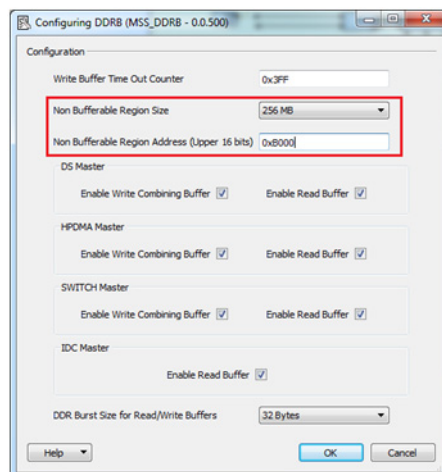
This use model shows the use of the DDR bridge for increasing throughput from the Cortex-M3 processor to external DDR memories. The Cortex-M3 processor performs only the single read and write transactions-not the burst transactions. The DDR bridge converts these single transactions into burst transactions and further increases the throughput. The buffers for DS and IDC masters are enabled for this, and the non-bufferable size is selected as **None**, as shown in the following image.

**Figure 136 • Configuring MSS DDR Bridge for Use Model 1**


## 5.5.2 Use Model 2: Selecting Non-Bufferable Region

This use model shows the use of the non-bufferable region selection in the DDR bridge. The buffering creates more latency in the applications which access non-continuous memory locations. In such cases non-bufferable region selection provides high throughput than bufferable. For example, when Cortex-M3 processor fetches the data from data region that is, stack and the application has bulk data transactions then keeping the data region as bufferable and code region as non-bufferable is preferred.

In this use model, the application uses only 256 MB of memory segment (0xB000\_0000 to 0XBFFF\_FFFF) as non-bufferable and the other memory region as bufferable. The following image shows the selection of the non-bufferable region.

**Figure 137 • Configuring MSS DDR Bridge for Use Model 2**




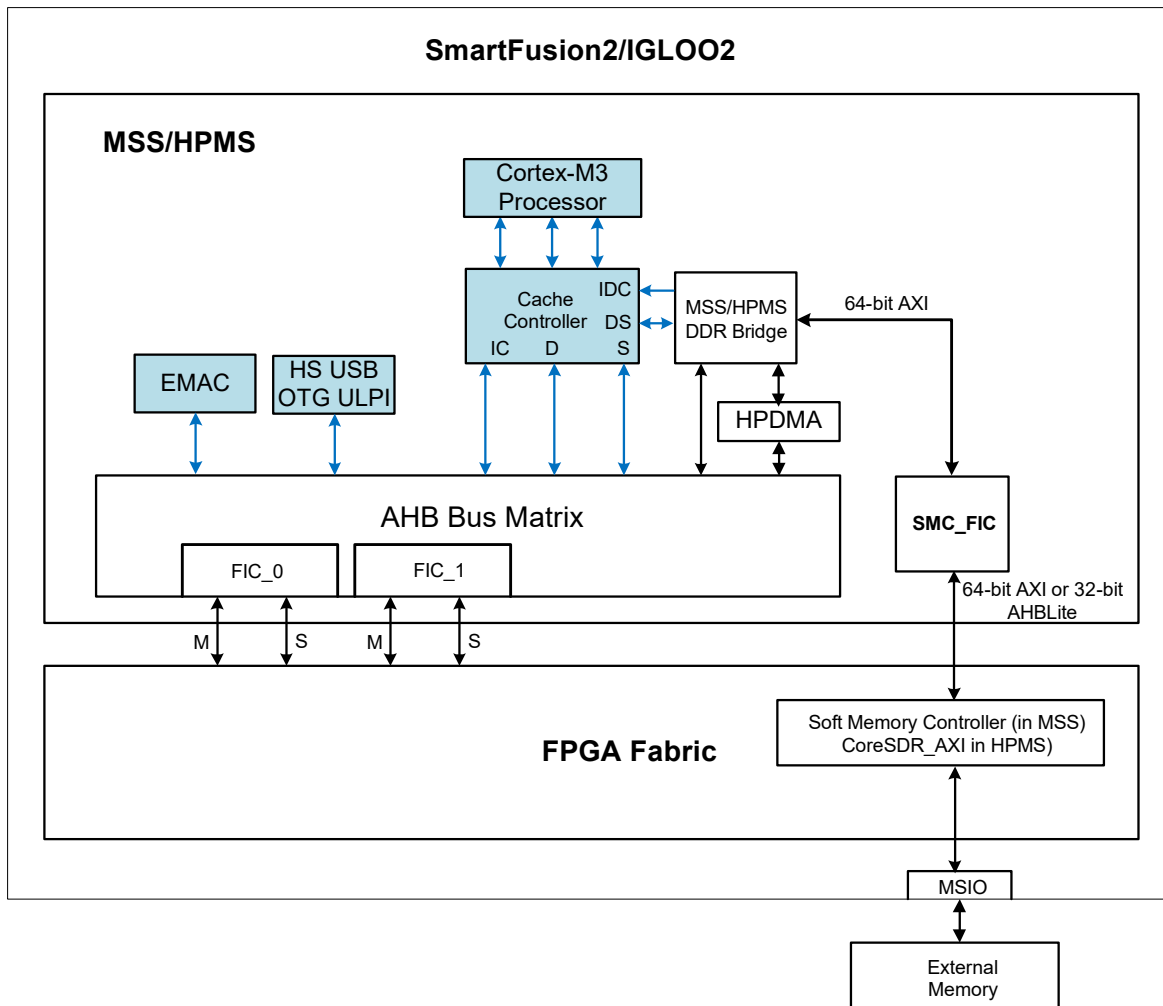
## 6 Soft Memory Controller Fabric Interface Controller

The SmartFusion2 and IGLOO2 soft memory controller fabric interface controller (SMC\_FIC) is used to access external bulk memories other than DDR through the FPGA fabric. The SMC\_FIC can be used with a soft memory controller for the MSS/HPMS to access memories such as SDRAM, flash, and SRAM. MSS/HPMS masters communicate with the SMC\_FIC through an MSS/HPMS DDR bridge present in the MSS/HPMS.

If the SMC\_FIC is enabled, the MDDR subsystem will not be available. In SMC\_FIC mode, the DDRIOs associated with the MDDR subsystem are available for user applications.

The following illustration shows a soft memory controller instantiated in the FPGA fabric for interfacing with external memory.

**Figure 138 • System Level SMC\_FIC Block Diagram**



**Note:** Blue arrows and blocks refer to the flow only in MSS. Rest are similar in MSS and HPMS.

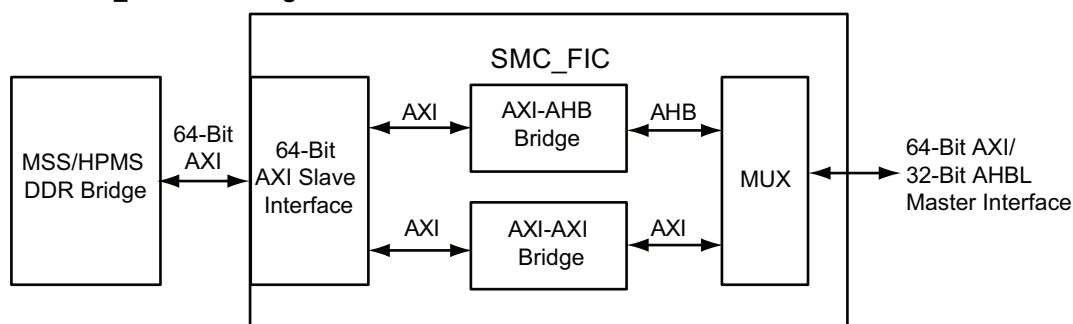
## 6.1 Functional Description

The SMC\_FIC receives 64-bit AXI transactions from the MSS/HPMS DDR bridge and converts them into 64-bit AXI or 32-bit AHB-Lite transactions to the SMC in the FPGA fabric. The following illustration shows the block diagram of the SMC\_FIC. The SMC\_FIC has two bridges:

- The AXI-AHB bridge converts 64-bit AXI transactions into 32-bit AHB transactions. It implements the AXI master to AHBL master protocol translator. This bridge is enabled when the SMC\_FIC is configured for a 32-bit AHB interface.
- The AXI-AXI bridge facilitates 64-bit AXI transactions from the MSS/HPMS DDR bridge to the 64-bit AXI FPGA fabric interface. This bridge is enabled when the SMC\_FIC is configured for a 64-bit AXI interface.

The SMC\_FIC receives a clock from the MSS/HPMS CCC that is identical to M3\_CLK/HPMS\_CLK. HPMS peripherals can access the external memory with the address space 0xA0000000 to 0xD0000000.

**Figure 139 • SMC\_FIC Block Diagram**



**Note:** The Libero 11.2 System Builder configures the SMC\_FIC in AHB mode for the devices M2GL005, M2GL010, and M2GL025. For other devices it configures the SMC\_FIC in AXI mode.

### 6.1.1 Port List

The following two tables show the 64-bit AXI and 32-bit AHBL port lists.

**Note:** The SMC\_FIC in M2S005, M2S010, M2S025, M2GL005, M2GL010, and M2GL025 devices provides only one 32-bit AHB-Lite interface.

The AXI interface has the following limitations:

- Supports only 64-bit read/write transactions on the AXI slave interface.
- Exclusive access cycles are not supported

The SMC\_AXI FIC AXI Read transactions can only be any of the following:

- Single transfer of 64 bit only aligned to 64-bit addresses.
- Wrap Transactions of 64-bit size and Wrap burst length of 2 which are aligned to 128-bit (16 byte) addresses.
- Wrap Transactions of 64-bit size and Wrap burst length of 4 which are aligned to 256-bit (32 byte) addresses.

**Table 165 • SMC\_FIC 64-bit AXI Port List**

Signal	Direction	Polarity	Description
MDDR_SMC_AXI_M_WLAST	Output	High	Indicates the last transfer in a write burst.
MDDR_SMC_AXI_M_WVALID	Output	High	Indicates whether or not valid write data and strobes are available. 1: Write data and strobes available 0: Write data and strobes not available

**Table 165 • SMC\_FIC 64-bit AXI Port List (continued)**

Signal	Direction	Polarity	Description
MDDR_SMC_AXI_M_BREADY	Output	High	Indicates whether or not the master can accept the response information. 1: Master ready 0: Master not ready
MDDR_SMC_AXI_M_AWVALID	Output	High	Indicates whether or not valid write address and control information are available. 1: Address and control information available 0: Address and control information not available
MDDR_SMC_AXI_M_ARVALID	Output	High	Indicates whether or not valid read address and control information are available. 1: Address and control information valid 0: Address and control information not valid
MDDR_SMC_AXI_M_RREADY	Output	High	Indicates whether or not the master can accept the read data and response information. 1: Master ready 0: Master not ready
MDDR_SMC_AXI_M_AWREADY	Input	High	Indicates that the slave is ready to accept an address and associated control signals. 1: Slave ready 0: Slave not ready
MDDR_SMC_AXI_M_WREADY	Input	High	Indicates whether or not the slave can accept the write data. 1: Slave ready 0: Slave not ready
MDDR_SMC_AXI_M_BVALID	Input	High	Indicates whether or not a valid write response is available. 1: Write response available 0: Write response not available.
MDDR_SMC_AXI_M_ARREADY	Input	High	Indicates whether or not the slave is ready to accept an address and associated control signals. 1: Slave ready 0: Slave not ready
MDDR_SMC_AXI_M_RLAST	Input	High	Indicates the last transfer in a read burst.
MDDR_SMC_AXI_M_RVALID	Input	High	Indicates whether or not the required read data is available and the read transfer can complete. 1: Read data available 0: Read data not available

**Table 165 • SMC\_FIC 64-bit AXI Port List (continued)**

Signal	Direction	Polarity	Description
MDDR_SMC_AXI_M_AWLEN[3:0]	Output		Indicates burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. 0000: 1 0001: 2 0010: 3 0011: 4 0100: 5 0101: 6 0110: 7 0111: 8 1000: 9 1001: 10 1010: 11 1011: 12 1100: 13 1101: 14 1110: 15 1111: 16
MDDR_SMC_AXI_M_AWBURST[1:0]	Output		Indicates burst type. The burst type, coupled with the size information, provides details on how the address for each transfer within the burst is calculated. 00: FIXED – Fixed-address burst, FIFO-type 01: INCR – Incrementing-address burst, normal sequential memory 10: WRAP – Incrementing-address burst that wraps to a lower address at the wrap boundary 11: Reserved
MDDR_SMC_AXI_M_AWID[3:0]	Output		Indicates identification tag for the write address group of signals.
MDDR_SMC_AXI_M_WDATA[63:0]	Output		Indicates write data.
MDDR_SMC_AXI_M_WID[3:0]	Output		Indicates ID tag of the write data transfer. The SMC_AXI64_WID value must match the SMC_AXI64_AWID value of the write transaction.
MDDR_SMC_AXI_M_WSTRB[7:0]	Output		Indicates which byte lanes to update in memory.
MDDR_SMC_AXI_M_ARID[3:0]	Output		Indicates identification tag for the read address group of signals.
MDDR_SMC_AXI_M_ARADDR[31:0]	Output		Indicates initial address of a read burst transaction.

**Table 165 • SMC\_FIC 64-bit AXI Port List (continued)**

Signal	Direction	Polarity	Description
MDDR_SMC_AXI_M_ARLEN[3:0]	Output		Indicates burst length. The burst length gives the exact number of transfers in a burst. 0000: 1 0001: 2 0010: 3 0011: 4 0100: 5 0101: 6 0110: 7 0111: 8 1000: 9 1001: 10 1010: 11 1011: 12 1100: 13 1101: 14 1110: 15 1111: 16
MDDR_SMC_AXI_M_ARSIZE[1:0]	Output		Indicates the maximum number of data bytes to transfer in each data transfer, within a burst. 00: 1 01: 2 10: 4 11: 8
MDDR_SMC_AXI_M_ARBURST[1:0]	Output		Indicates burst type. The burst type, coupled with the size information, provides details on how the address for each transfer within the burst is calculated. 00: FIXED – Fixed-address burst, FIFO type 01: INCR – Incrementing-address burst, normal sequential memory 10: WRAP – Incrementing-address burst that wraps to a lower address at the wrap boundary 11: Reserved
MDDR_SMC_AXI_M_AWADDR[31:0]	Output		Indicates write address. The write address bus gives the address of the first transfer in a write burst transaction.
MDDR_SMC_AXI_M_AWSIZE[1:0]	Output		Indicates the maximum number of data bytes to transfer in each data transfer, within a burst. 00: 1 01: 2 10: 4 11: 8
MDDR_SMC_AXI_M_AWLOCK[1:0]	Output		Indicates lock type. This signal provides additional information about the atomic characteristics of the write transfer. 00: Normal access 01: Exclusive access 10: Locked access 11: Reserved

**Table 165 • SMC\_FIC 64-bit AXI Port List (continued)**

Signal	Direction	Polarity	Description
MDDR_SMC_AXI_M_ARLOCK[1:0]	Output		Indicates lock type. This signal provides additional information about the atomic characteristics of the read transfer. 00: Normal access 01: Exclusive access 10: Locked access 11: Reserved
MDDR_SMC_AXI_M_BID[3:0]	Input		Indicates response ID. The identification tag of the write response. The MDDR_SMC_AXI_M_BID value must match the MDDR_SMC_AXI_M_AWID value of the write transaction to which the slave is responding.
MDDR_SMC_AXI_M_RID[3:0]	Input		Read ID tag. This signal is the ID tag of the read data group of signals. The MDDR_SMC_AXI_M_RID value is generated by the slave and must match the MDDR_SMC_AXI_M_ARID value of the read transaction to which it is responding.
MDDR_SMC_AXI_M_RRESP[1:0]	Input		Indicates read response. This signal indicates the status of the read transfer. 00: Normal access okay 01: Exclusive access okay 10: Slave error 11: Decode error
MDDR_SMC_AXI_M_BRESP[1:0]	Input		Indicates write response. This signal indicates the status of the write transaction. 00: Normal access okay 01: Exclusive access okay 10: Slave error 11: Decode error
MDDR_SMC_AXI_M_RDATA[63:0]	Input		Indicates read data.

**Table 166 • SMC\_FIC 32-bit AHB-Lite Port List**

Signal	Direction	Polarity	Description
MDDR_SMC_AHB_M_HMASTLOCK	Output	High	Indicates that the current master is performing a locked sequence of transfers.
MDDR_SMC_AHB_M_HWRITE	Output	High	Indicates write control signal. When High, this signal indicates a write transfer, and when Low, a read transfer.
MDDR_SMC_AHB_M_HRESP	Input	High	The transfer response indicates the status of transfer.
MDDR_SMC_AHB_M_HREADY	Input	High	When High, the signal indicates that a transfer has been completed on the bus. This signal may be driven Low to extend a transfer.
MDDR_SMC_AHB_M_HBURST[1:0]	Output		Indicates the burst type.
MDDR_SMC_AHB_M_HTRANS[1:0]	Output		Indicates the type of the current transfer. 00: Idle 01: Busy 10: Non-sequential 11: Sequential

**Table 166 • SMC\_FIC 32-bit AHB-Lite Port List (continued)**

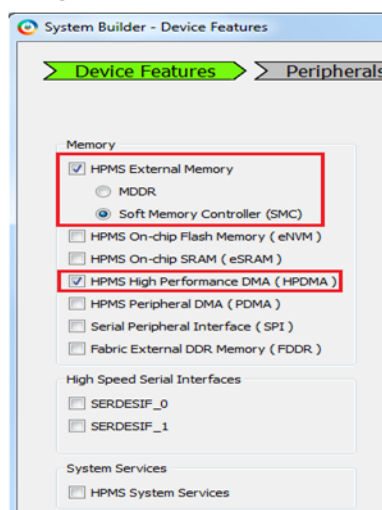
Signal	Direction	Polarity	Description
MDDR_SMC_AHB_M_HSIZE[1:0]	Output		Indicates the size of the transfer. 00: Byte 01: Half word 10: Word
MDDR_SMC_AHB_M_HWDATA[31:0]	Output		The write data bus is used to transfer data during write operations.
MDDR_SMC_AHB_M_HADDR[31:0]	Output		Indicates address bus.
MDDR_SMC_AHB_M_HRDATA[31:0]	Input		The read data bus is used to transfer data from bus slaves to the bus master during read operations.

## 6.2 How to Use SMC\_FIC in IGLOO2 Device

This section describes how to use SMC\_FIC for accessing external SDR memory. The SMC\_FIC can be enabled and configured using the System Builder in the Libero SoC design software. The System Builder uses the CoreSDR\_AXI and connects to SMC\_FIC interface. The CoreSDR\_AXI IP is an AXI based SDR memory controller. The steps provided below are required to access the external SDR memory from CoreSDR\_AXI.

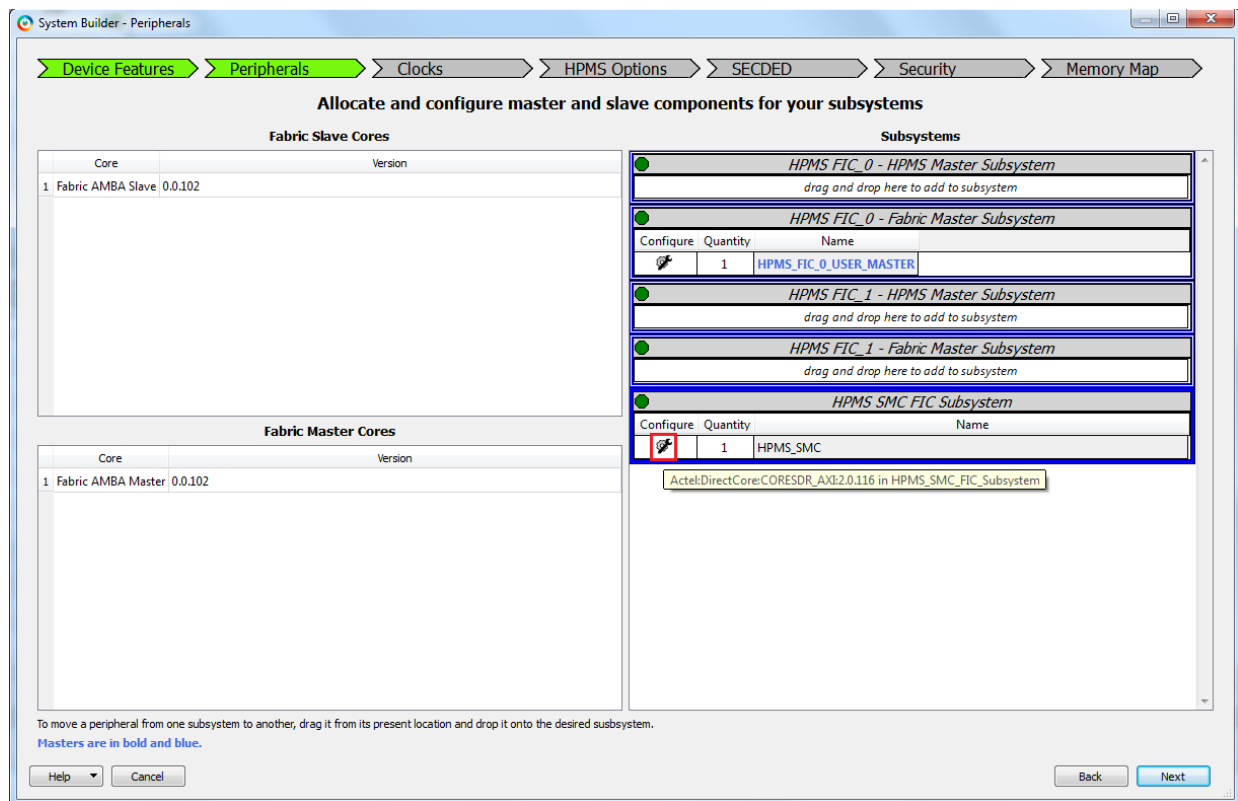
1. Select the **HPMS External Memory, Soft Memory Controller (SMC)** and **HPDMA** in the **System Builder - Device Features** window as shown in the following image.

For details on how to launch the **System Builder** wizard and a detailed information on how to use it, refer the [IGLOO2 System Builder User Guide](#).

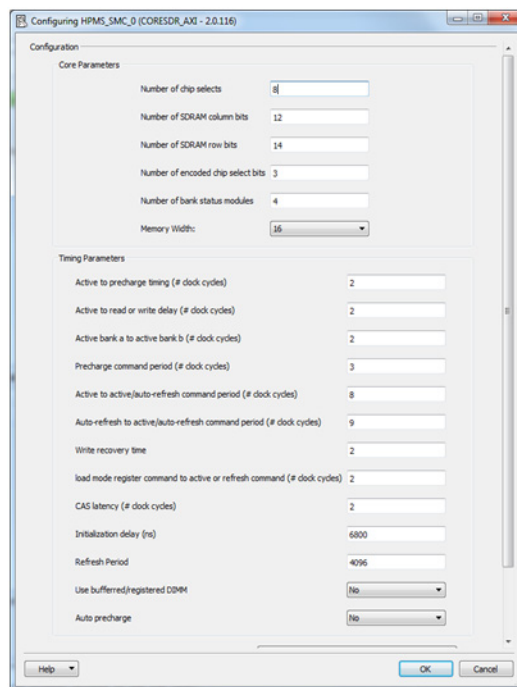
**Figure 140 • HPMS External Memory Configurator**

For more information on how to use SMC\_FIC in SmartFusion2 Device, refer to ["Appendix A: How to Use SMC\\_FIC in SmartFusion2 Devices"](#).

2. Click **Next** to get the **Peripherals** window. Click **configure** under **HPMS SMC\_FIC subsystem** as shown in the following image.

**Figure 141 • HPMS SMC\_FIC Subsystem Configuration**

3. Configure CoreSDR\_AXI to match the external memory parameters.

**Figure 142 • CoreSDR\_AXI Configuration**

4. Navigate to the **Memory Map** tab giving the required data in the rest of the **System Builder** tabs. Click Finish.



5. The System Builder creates a SmartDesign with CoreSDR\_AXI connected to SMC\_FIC and exposes the AHB mirrored master interface which is connected to FIC\_0 to access the HPDMA configuration registers.
6. Microsemi provides CoreHPDMActrl IP to configure the HPDMA. Connect the CoreHPDMActrl IP to the AHB mirrored master interface of System Builder created design or connect user AHB master logic to configure the HPDMA to perform the DMA transactions from SDRAM.

## 6.3 SYSREG Control Register for SMC\_FIC

Complete descriptions of each register and bit are located in the “System Register Map” chapter of the *IGLOO2 High Performance Memory Subsystem User Guide* and are listed as follows for clarity.

**Table 167 • MDDR\_CR Register**

Register Name	Register Type	Flash Write Protect	Reset Source	Description
MDDR_CR	RW-P	Register	PORESET_N	MDDR configuration register

## 6.4 Appendix A: How to Use SMC\_FIC in SmartFusion2 Devices

This section describes how to use SMC\_FIC in an application and contains the following sub-sections:

- Design Flow
- Use Model 1: Accessing SDRAM from MSS Through CoreSDR\_AXI

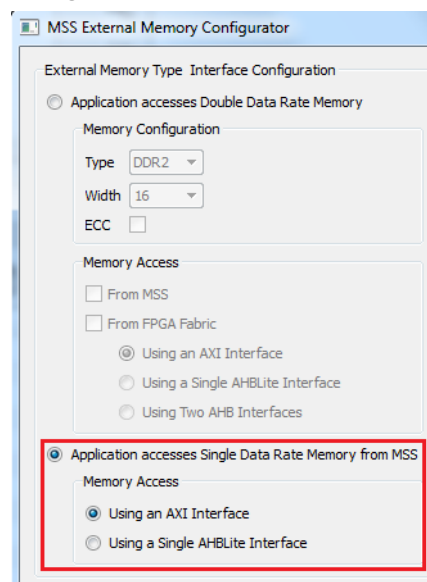
### 6.4.1 Design Flow

The SMC\_FIC can be enabled and configured through the MSS external memory configurator, which is part of the MSS configurator in the Libero SoC design software. The following image shows the MSS external memory configurator. The external memory type interface must be selected as “**Application Accesses Single Data Rate Memory from MSS**” to enable the SMC\_FIC.

Select the type of interface as AXI or AHB-32. After completing the configuration, the selected interface is exposed in SmartDesign. This interface must be connected to the SMC through CoreAXI or CoreAHB.

Microsemi provides CoreSDR\_AHB and CoreSDR\_AXI SMC IPs for interfacing with external SDRAM. Any other custom soft memory controller can also be implemented in the FPGA fabric to access the external memories.

**Figure 143 • MSS External Memory Configurator**

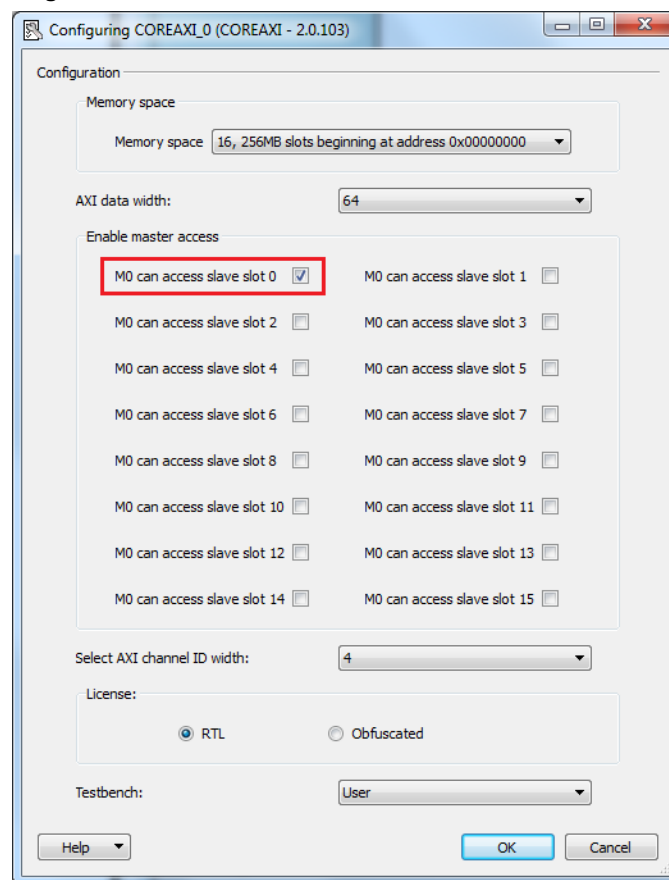


## 6.4.2 Use Model 1: Accessing SDRAM from MSS Through CoreSDR\_AXI

This use model describes how to use the SMC\_FIC to access external SDR memory from MSS. It uses the AXI interface of SMC\_FIC to connect to CoreSDR\_AXI. CoreSDR\_AXI is an AXI-based SDR memory controller. The steps provided below are required to access the external SDR memory from CoreSDR\_AXI.

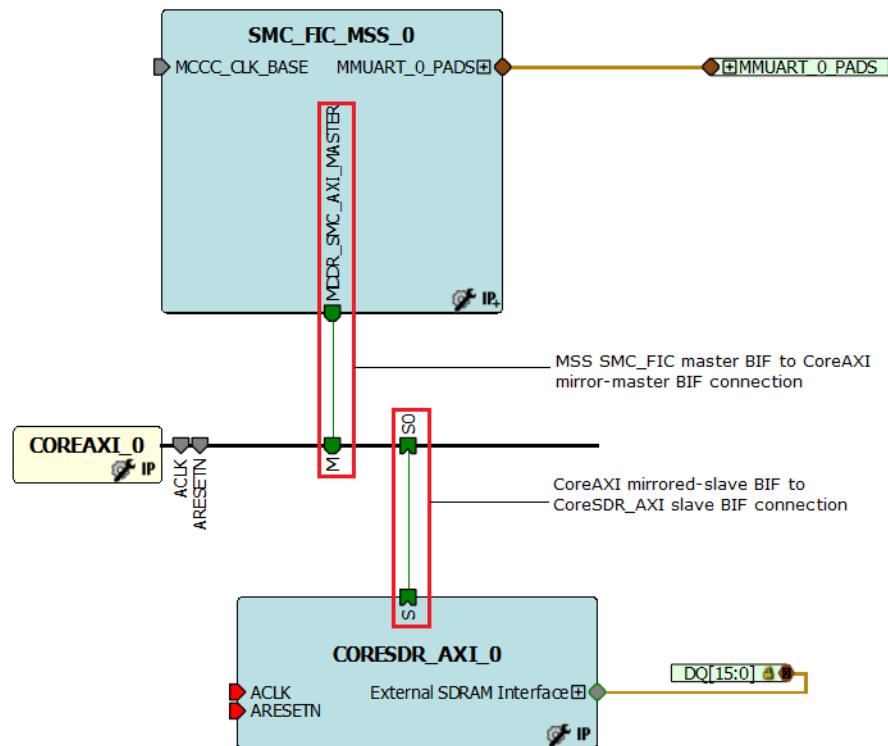
1. Instantiate the SmartFusion2 MSS component onto the SmartDesign canvas.
2. Configure the SmartFusion2 MSS peripheral components to meet application needs using MSS configurator.
3. Configure the external memory interface type and select **Using an AXI Interface**, as shown in the previous image.
4. Instantiate and configure CoreAXI so that the master slot M0 is enabled for the slave slot S0, as shown in the following image. The slot size selection must be matched with the amount of external memory space.

**Figure 144 • Core\_AXI Configuration**



5. Instantiate and configure CoreSDR\_AXI to match the external memory parameters.
6. Connect the subsystem together as shown in the following image. Connect the MSS SMC\_FIC master interface port, MDDR\_SMC\_AXI\_MASTER, to the CoreAXI bus mirrored-master M0. Connect the CoreAXI mirrored-slave bus interface (BIF) port S0 to the slave BIF port of the CoreSDR\_AXI core instance.

**Figure 145 • Subsystem Connections in SmartDesign**



Refer to the [Accessing External SDRAM through Fabric](#) tutorial, which describes the steps for creating a design that accesses external SDR memory from the Cortex-M3 processor. The tutorial also explains the steps for simulating the design in Libero SoC.