

**UG0448**  
**User Guide**  
**IGLOO2 FPGA High Performance Memory Subsystem**



**Microsemi Corporate Headquarters**  
One Enterprise, Aliso Viejo,  
CA 92656 USA  
Within the USA: +1 (800) 713-4113  
Outside the USA: +1 (949) 380-6100  
Fax: +1 (949) 215-4996  
Email: [sales.support@microsemi.com](mailto:sales.support@microsemi.com)  
[www.microsemi.com](http://www.microsemi.com)

© 2017 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are 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 Corporation (Nasdaq: MSCC) 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. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. 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	2
1.5	Revision 4.0	2
1.6	Revision 3.0	2
1.7	Revision 2.0	2
1.8	Revision 1.0	2
1.9	Revision 0.0	3
<b>2</b>	<b>Embedded NVM (eNVM) Controllers</b>	<b>4</b>
2.1	Features	4
2.2	Functional Description	5
2.2.1	Memory Organization	6
2.2.2	Data Retention Time	7
2.2.3	eNVM Access Time	7
2.2.4	Theory of Operation	7
2.2.5	eNVM Command Register	9
2.2.6	Error Response	16
2.2.7	Interrupt to Fabric Master	16
2.3	Security	16
2.3.1	User-Protectable 4K Regions	17
2.3.2	eNVM Pages for Special Purpose Storage	20
2.4	How to Use eNVM	23
2.4.1	Data Storage in eNVM Using the Libero eNVM Client	23
2.4.2	HPMS Subsystem	28
2.4.3	HPMS Subsystem Connected to the FPGA Fabric Master	29
2.4.4	Reading the eNVM Block	29
2.4.5	Writing to the eNVM Block	29
2.5	SYSREG Control Registers	29
2.6	eNVM Control Registers	34
2.6.1	Status Register Bit Definitions	38
<b>3</b>	<b>Embedded SRAM (eSRAM) Controllers</b>	<b>41</b>
3.1	Features	41
3.2	Functional Description	42
3.2.1	Memory Organization	43
3.2.2	Modes of Operation	44
3.2.3	Pipeline Modes and Wait States for Read and Write Operations	45
3.3	How to Use eSRAM	47
3.3.1	Accessing eSRAM Using FPGA Fabric Master	48
3.3.2	HPMS Subsystem	51
3.3.3	HPMS Subsystem Connected to the FPGA Fabric Master	52
3.4	SYSREG Control Registers	52
<b>4</b>	<b>AHB Bus Matrix</b>	<b>61</b>
4.1	Functional Description	61
4.1.1	Architecture Overview	61
4.1.2	Timing Diagrams	64

4.1.3	Details of Operation	68
4.1.4	System Memory Map	73
4.2	How to Use AHB Bus Matrix	75
4.3	Register Map	77
<b>5</b>	<b>High Performance DMA Controller</b>	<b>78</b>
5.1	Features	78
5.2	Functional Description	79
5.2.1	Architecture Overview	79
5.2.2	Initialization	81
5.2.3	Details of Operation	82
5.3	How to Use HPDMA	83
5.3.1	Configuring HPDMA	84
5.3.2	HPMS Subsystem	87
5.3.3	HPMS Subsystem Connected to the FPGA Fabric Master	88
5.3.4	MDDR to eSRAM	88
5.4	HPDMA Controller Register Map	89
5.4.1	HPDMA Register Bit Definitions	90
5.5	SYSREG Control Register	106
<b>6</b>	<b>Peripheral DMA</b>	<b>107</b>
6.1	Features	107
6.2	Functional Description	108
6.2.1	Architecture Overview	108
6.2.2	PDMA Port List	111
6.2.3	Initialization	112
6.2.4	Details of Operations	112
6.3	How to Use PDMA	113
6.3.1	HPMS Subsystem	117
6.3.2	HPMS Subsystem Connected to the FPGA Fabric Master	118
6.3.3	SPI_0 to eSRAM_0	118
6.3.4	eNMV_0 to eSRAM_0	119
6.4	PDMA Register Map	120
6.4.1	PDMA Configuration Register Bit Definitions	123
6.5	SYSREG Control Registers	129
<b>7</b>	<b>Serial Peripheral Interface Controller</b>	<b>130</b>
7.1	Features	130
7.2	Functional Description	131
7.2.1	Architecture Overview	131
7.2.2	Interface	132
7.2.3	Initialization	142
7.2.4	Details of Operation	143
7.3	How to Use the SPI Controller	145
7.3.1	HPMS Subsystem	147
7.3.2	HPMS Subsystem Connected to the FPGA Fabric Master	148
7.3.3	Accessing the External SPI Flash Using HPMS SPI_0	148
7.4	SPI Register Map	149
7.4.1	SYSREG Configuration Register Summary	149
7.4.2	SPI Register Summary	150
7.4.3	SPI Register Details	151
<b>8</b>	<b>Communication Block</b>	<b>161</b>
8.1	Features	161
8.2	Functional Description	162

8.2.1	Architecture Overview	162
8.2.2	Frame/Command Marker	163
8.2.3	Clocks	164
8.2.4	Resets	164
8.2.5	Interrupts	164
8.2.6	CoreSysServices Soft IP	164
8.3	How to Use COMM_BLK	165
8.3.1	Configuring COMM_BLK	166
8.4	COMM_BLK Configuration Registers	171
8.5	COMM_BLK Register Interface Details	172
8.5.1	Control Register	172
8.5.2	Status Register	173
8.5.3	Interrupt Enable Register	173
8.5.4	Byte Data Register	174
8.5.5	Word Data Register	174
8.5.6	Frame/Command Byte Register	174
8.5.7	Frame/Command Word Register	175
<b>9</b>	<b>Reset Controller</b>	<b>176</b>
9.1	Functional Description	177
9.1.1	Power-On Reset Generation Sequence	177
9.1.2	Power-Up to Functional Time Sequence	180
9.1.3	Power-Up to Functional Time Data	181
9.1.4	Power-On Reset	187
9.1.5	System Reset	187
9.1.6	Block Resets	188
9.2	CoreResetP Soft Reset Controller	190
9.2.1	Reset Topology	190
9.2.2	Implementation	193
9.2.3	Timing Diagrams	193
9.3	SYSREG Control Registers	195
<b>10</b>	<b>System Register Block</b>	<b>196</b>
10.1	SYSREG Block Register Write Protection	196
10.1.1	Register Write Protect	196
10.1.2	Field Write Protect	197
10.1.3	Bit Write Protect	197
10.2	Register Types	198
10.3	Register Lock Bits Configuration	200
10.3.1	Lock Bit File	201
10.3.2	Lock Bit File Syntax	201
10.3.3	Locking and Unlocking a Register	202
10.4	Register Map	202
10.5	System Registers Behavior for M2GL005/010 devices	206
10.6	Register Details	206
10.6.1	eSRAM Latency Configuration Register	206
10.6.2	eNVM Configuration Register	207
10.6.3	eNVM FPGA Fabric Remap Base Address Register	209
10.6.4	HPMS DDR Bridge Buffer Timer Control Register	210
10.6.5	HPMS DDR Bridge Non-Bufferable Address Control Register	210
10.6.6	HPMS DDR Bridge Non-Bufferable Size Control Register	210
10.6.7	HPMS DDR Bridge Configuration Register	211
10.6.8	EDAC Configuration Register	212
10.6.9	Master Weight Configuration Register 0	212
10.6.10	Master Weight Configuration Register 1	213
10.6.11	Software Interrupt Register	213

10.6.12	Software Reset Control Register	214
10.6.13	Fabric Interface Control (FIC) Register	215
10.6.14	MDDR Configuration Register	215
10.6.15	Peripheral Clock MUX Select Control Register	216
10.6.16	MDDR I/O Calibration Control Register	216
10.6.17	EDAC Interrupt Enable Control Register	217
10.6.18	eSRAM PIPELINE Configuration Register	217
10.6.19	HPMS DDR PLL Status Low Configuration Register	218
10.6.20	HPMS DDR PLL Status High Configuration Register	219
10.6.21	HPMS DDR Fabric Alignment Clock Controller (FACC) Configuration Register 1	220
10.6.22	HPMS DDR Fabric Alignment Clock Controller Configuration Register 2	222
10.6.23	HPMS Clock Calibration Control Register	224
10.6.24	PLL Delay Line Select Control Register	224
10.6.25	Reset Source Control Register	224
10.6.26	HPMS DDR Bridge High Performance DMA Master Error Address Status Register	225
10.6.27	HPMS DDR Bridge AHB Bus Error Address Status Register	225
10.6.28	HPMS DDR Bridge Buffer Empty Status Register	225
10.6.29	HPMS DDR Bridge Disable Buffer Status Register	226
10.6.30	eSRAM0 EDAC Count	226
10.6.31	eSRAM1 EDAC Count	226
10.6.32	eSRAM0 EDAC Address Register	227
10.6.33	eSRAM1 EDAC Address Register	227
10.6.34	Security Configuration Register for Masters 4, 5, and DDR_FIC	227
10.6.35	Security Configuration Register for Masters 3 and 7	228
10.6.36	Security Configuration Register for Master 9	229
10.6.37	Device Status Register	229
10.6.38	eNVM Protect User Register	230
10.6.39	IGLOO2 eNVM Status Register	231
10.6.40	Device Version Register	231
10.6.41	HPMS PLL Status Register	231
10.6.42	eNVM Status Register	232
10.6.43	DDRB Status Register	233
10.6.44	MDDR IO Calibration Status Register	233
10.6.45	HPMS Clock Calibration Status	234
10.6.46	Fabric Protected Size Register	234
10.6.47	Fabric Protected Base Address Register	235
10.6.48	EDAC Status Register	235
10.6.49	HPMS Internal Status Register	236
10.6.50	HPMS External Status Register	236
10.6.51	Clear EDAC Counters	237
10.6.52	Flush Configuration Register	238
<b>11</b>	<b>Fabric Interface Interrupt Controller</b>	<b>239</b>
11.1	Features	239
11.2	Functional Description	240
11.2.1	Architecture Overview	240
11.2.2	FIIC Port List	242
11.3	How to Use FIIC	242
11.4	FIIC Controller Registers	243
11.5	FIIC Controller Register Bit Definitions	243
<b>12</b>	<b>Fabric Interface Controller</b>	<b>249</b>
12.1	Functional Description	250
12.1.1	Configuring FIC for Master or Slave Interface	250
12.2	FIC Interface Port List	251
12.3	Timing Diagrams	253
12.4	Implementation Considerations	256

12.5	Fabric Interface Clocks .....	256
12.6	How to Use FIC .....	257
12.6.1	FIC_1 Configuration .....	257
12.6.2	FIC_0 Configuration .....	261
12.6.3	Use Model 1: Connecting APB3 Master and Slave to FIC_1 .....	261
12.6.4	Use Model 2: Connecting AHB-Lite Master and Slave to FIC_1 .....	261
12.7	SYSREG Control Registers for FIC_0 and FIC_1 .....	262
12.8	Reference Documents .....	262
<b>13</b>	<b>FIC_2 (APB Configuration Interface) .....</b>	<b>263</b>
13.1	Functional Description .....	263
13.1.1	Architecture Overview .....	264
13.1.2	Port List .....	264
13.1.3	CoreConfig IP .....	266
13.2	How to Use FIC_2 .....	266
13.2.1	Configuring FIC_2 (Peripheral Initialization) Using Libero SoC .....	266
13.2.2	FIC_2 Interfaces for MDDR .....	268
13.2.3	FIC_2 Interfaces for SerDes .....	269

# Figures

Figure 1	eNVM Connection to AHB Bus Matrix	4
Figure 2	eNVM Controller Block Diagram	5
Figure 3	Write Path	8
Figure 4	Read Path	8
Figure 5	Timing Diagram for Single Word Read Operation	12
Figure 6	Timing Diagram for Consecutive Reads Incrementing Through Memory	12
Figure 7	eNVM Program and Verify Operations	13
Figure 8	Exclusive Register Access and Filling Data in WDBUFF	13
Figure 9	Issuing the ProgramADS Command	14
Figure 10	Completion of ProgramADS and Issue of VerifyADS Command	14
Figure 11	Completion of eNVM Verify Operation	14
Figure 12	Complete eNVM Program and Verify Operations Waveform	15
Figure 13	Exclusive Register Access and Filling Data in WDBUFF	15
Figure 14	ProgramAD Command	15
Figure 15	ProgramDA command	15
Figure 16	ProgramStart Command	16
Figure 17	eNVM Special Sectors for the M2GL050TS Device with 256 KB eNVM_0	17
Figure 18	eNVM Special Sectors for the M2GL005S Device with 128 KB eNVM_0	17
Figure 19	eNVM Special Sectors for M2GL010TS and M2GL025TS Devices with 256 KB eNVM_0	18
Figure 20	eNVM Special Sectors for the M2GL060TS Device with 256 KB eNVM_0	18
Figure 21	eNVM Special Sectors for M2GL090TS and M2GL150TS Devices with 512 KB	19
Figure 22	System Builder Window	23
Figure 23	System Builder - Device Features Tab	24
Figure 24	System Builder - Memories Tab	24
Figure 25	eNVM: Modify Core Dialog Box	25
Figure 26	Add Data Storage Client Dialog	26
Figure 27	eNVM: Modify Core Dialog Box with Two eNVM Clients	26
Figure 28	System Builder - Security Tab	27
Figure 29	System Builder - Memory Map Tab	28
Figure 30	HPMS Subsystem	28
Figure 31	HPMS Interconnection with FPGA Fabric Master	29
Figure 32	eSRAM_0 and eSRAM_1 Connection to AHB Bus Matrix	41
Figure 33	eSRAM Controller Block Diagram	42
Figure 34	System Builder Window	47
Figure 35	System Builder - Device Features Tab	48
Figure 36	System Builder - HPMS Options Tab	49
Figure 37	System Builder - SECDED Tab	49
Figure 38	System Builder - Security Tab	50
Figure 39	System Builder - Memory Map Tab	51
Figure 40	HPMS Subsystem	51
Figure 41	HPMS Interconnection with FPGA Fabric Master	52
Figure 42	AHB Bus Matrix Masters and Slaves	61
Figure 43	Master Stage and Slave Stage Interconnection	63
Figure 44	APB Destinations Connected to AHB Bus Matrix	63
Figure 45	AHB-Lite Write Transactions	64
Figure 46	AHB-Lite Read Transactions	65
Figure 47	AHB-to-AHB Write Transactions	66
Figure 48	AHB-to-AHB Read Transactions	67
Figure 49	Pure Round Robin and Fixed Priority Slave Arbitration Scheme	69
Figure 50	WRR and Fixed Priority Slave Arbitration Scheme	70
Figure 51	Slave Arbitration Flow Diagram	72
Figure 52	Default System Memory Map	73
Figure 53	System Builder Window	75
Figure 54	System Builder - HPMS Options Tab	76

Figure 55	HPDMA Interfacing With HPMSDDR Bridge and AHB Bus Matrix	78
Figure 56	HPDMA Controller Block Diagram	79
Figure 57	HPDMA Registers	80
Figure 58	DMA Controller Flow Chart	80
Figure 59	System Builder Window	83
Figure 60	System Builder - Device Features Tab	84
Figure 61	System Builder - Peripherals Tab	85
Figure 62	HPMS Options Tab - Round Robin Weight Configuration for HPDMA Master	86
Figure 63	System Builder - Memory Map Tab	87
Figure 64	HPMS Subsystem	87
Figure 65	HPMS Interconnection with FPGA Fabric Master	88
Figure 66	PDMA Interfacing with AHB Bus Matrix	107
Figure 67	PDMA Internal Architecture	108
Figure 68	Ping-Pong Operation Flow for DMA Channel	110
Figure 69	System Builder Window	113
Figure 70	System Builder - Device Features Tab	114
Figure 71	System Builder - Peripherals Tab	115
Figure 72	Configuring PDMA Weight Values	116
Figure 73	System Builder - Memory Map Tab	117
Figure 74	HPMS Subsystem	117
Figure 75	HPMS Subsystem Connected to the FPGA Fabric Master	118
Figure 76	Microcontroller Subsystem Showing SPI Peripherals	130
Figure 77	SPI Controller Block Diagram	131
Figure 78	Motorola SPI Mode 0	134
Figure 79	Motorola SPI Mode 0 Multiple Frame Transfer	134
Figure 80	Motorola SPI Mode 1	135
Figure 81	Motorola SPI Mode 2	135
Figure 82	Motorola SPI Mode 3	135
Figure 83	Write Operation Timing	137
Figure 84	Read Operation Timing	137
Figure 85	Page Program Timing	138
Figure 86	National Semiconductor MICROWAVE Single Frame Transfer	139
Figure 87	National Semiconductor MICROWIRE Multiple Frame Transfer	139
Figure 88	TI Synchronous Serial Single Frame Transfer	140
Figure 89	TI Synchronous Serial Multiple Frame Transfer	140
Figure 90	SPE Command/Data Format	141
Figure 91	System Builder Window	145
Figure 92	System Builder - Device Features Tab	146
Figure 93	System Builder - Memory Map Tab	147
Figure 94	HPMS Subsystem	147
Figure 95	HPMS Interconnection with FPGA Fabric Master	148
Figure 96	Fabric Master Accessing the External SPI Flash Using HPMS SPI_0	148
Figure 97	Interfacing of COMM_BLK with AHB Bus Matrix	162
Figure 98	Interfacing of COMM_BLK with System Controller	163
Figure 99	System Builder Window	165
Figure 100	System Builder - Device Features Tab	166
Figure 101	CoreSysServices IP to COMM_BLK Path	167
Figure 102	Clocks Configuration	168
Figure 103	System Builder - Memory Map Tab	169
Figure 104	COMM_BLK Connection with CoreSysServices IP	170
Figure 105	COMM_BLK Configuration Dialog	171
Figure 106	Reset Signals Distribution in IGLOO2 Devices	176
Figure 107	Power-On Reset Generation Block Diagram	177
Figure 108	Power-On Reset Delay Configuration	178
Figure 109	SYSRESET Macro	178
Figure 110	Power-Up to Functional Time Sequence Diagram	180
Figure 111	VDD Power-Up to Functional Time Design Setup	182
Figure 112	VDD Power-Up to Functional Timing Diagram	182
Figure 113	VDD Power-Up to Functional Time Flow	184

Figure 114	DEVRST_N Power-Up to Functional Timing Diagram	185
Figure 115	DEVRST_N Power-Up to Functional Time Flow	186
Figure 116	Reset Controller During Power-On Reset	187
Figure 117	SYSRESET_N Generation	187
Figure 118	Reset Controller During SYSRESET_N	188
Figure 119	Reset Controller With Only Block Level Resets	188
Figure 120	MDDR_AXI_RESET_N Generation	189
Figure 121	MDDR_APB_RESET_N Generation	189
Figure 122	Block Level Reset Generation	189
Figure 123	HPMS_READY Signal Generation	190
Figure 124	System Builder-Generated Design with MDDR and SERDESIF Interfaces	191
Figure 125	System Builder-Generated Design without MDDR/FDDR/SerDes Interface	191
Figure 126	CoreResetP Connectivity with Peripheral Resets	192
Figure 127	CoreResetP Connectivity with SERDES_IF Block	192
Figure 128	Timing Diagram for Reset Signals Initiated by the Assertion of POWER_N_RESET_N	193
Figure 129	Timing Diagram for Reset Signals Initiated by the Assertion of FIC_2_APB_M_PRESET_N	193
Figure 130	Timing for Reset Signals Initiated by the Assertion of EXT_RESET_IN_N	194
Figure 131	Timing for Reset Signals Initiated by the Assertion of USER_FAB_RESET_IN_N	194
Figure 132	Register Write Protect	196
Figure 133	Field Write Protect	197
Figure 134	Bit Write Protect	197
Figure 135	RW-P Type	198
Figure 136	RW Type	199
Figure 137	RO Type	199
Figure 138	RO-P Type	200
Figure 139	RO-U Type	200
Figure 140	Register Lock Bit Settings	200
Figure 141	Lock Bit Configuration File	201
Figure 142	FIIC Connection to AHB Bus Matrix	239
Figure 143	Block Diagram for Fabric Interface Interrupt Controller	240
Figure 144	Combinational Circuit for Mapping HPMS Interrupts to a HPMS_INT_M2F	240
Figure 145	FIIC Bus	242
Figure 146	HPMS to Fabric Interrupts	242
Figure 147	The FIC Connection to the AHB Bus Matrix	249
Figure 148	Fabric Interface Controller Block Diagram	250
Figure 149	Fabric Interface Controller Top-Level View	251
Figure 150	Timing Diagram for AHB-Lite Bus Signals from Fabric Master to FIC for a Write Transaction	253
Figure 151	Timing Diagram for AHB-Lite Bus Signals from Fabric Master to FIC for a Read Transaction	253
Figure 152	Timing Diagram for AHB-Lite Bus Signals from FIC to the Fabric Slave for a Write Transaction	254
Figure 153	Timing Diagram for AHB-Lite Bus Signals from FIC to the Fabric Slave for a Read Transaction	254
Figure 154	Timing Diagram for APB3 Bus Signals from Fabric Master to FIC for a Write Transaction	255
Figure 155	Timing Diagram for APB3 Bus Signals from Fabric Master to FIC for a Read Transaction	255
Figure 156	Timing Diagram for APB3 Bus Signals from FIC to the Fabric Slave for a Write Transaction	256
Figure 157	Timing Diagram for APB3 Bus Signals from FIC to the Fabric Slave for a Read Transaction	256
Figure 158	System Builder Window	257
Figure 159	System Builder- Peripherals Tab	258
Figure 160	Peripherals Tab - Configure Option for AMBA_MASTER_0	258
Figure 161	Interface Type Configuration	259
Figure 162	Clocks Tab - Fabric Interface Clocks	259
Figure 163	HPMS Options Tab with Round Robin Weight for FIC_1 Master	260
Figure 164	Memory Map Tab	260
Figure 165	Top-Level Smart Design View for Use Model 1	261
Figure 166	Top-Level Smart Design View for Use Model 2	262
Figure 167	APB Configuration Interface and Subsystems Connections with HPMS Master	263
Figure 168	System Builder Window	266
Figure 169	System Builder - Device Features Window	267
Figure 170	System Builder - Memory Map Tab	268
Figure 171	FIC_2 Interfaces for MDDR	268
Figure 172	HPMS Subsystem with APB Configuration Interface Signals	269

Figure 173 Interfacing of CoreConfig IP Mirrored APB Slave with SERDES\_IF Block . . . . . 270

# Tables

Table 1	eNVM Address Locations	5
Table 2	Memory Organization	6
Table 3	Data Retention Time	7
Table 4	AHBL Address Map to NVM	8
Table 5	Command (CMD) Register	9
Table 6	Command Table	9
Table 7	User Protection Regions	19
Table 8	Special Purpose Storage Regions	20
Table 9	Special Purpose Storage Regions for M2GL060, M2GL090 and M2GL150 Devices	20
Table 10	SYSREG Control Registers	29
Table 11	ENVM_CR	30
Table 12	SW_ENVMREMAPSIZE	32
Table 13	ENVM_REMAP_FAB_CR	32
Table 14	ENVM_PROTECT_USER	33
Table 15	ENVM_STATUS	34
Table 16	ENVM_SR	34
Table 17	eNVM Control Registers Base Address	34
Table 18	Control Registers Description	34
Table 19	Status Register Bit Definitions	38
Table 20	NV_PAGE_STATUS	39
Table 21	INTEN[10:0]	39
Table 22	NV_FREQRNG Calculations at Different HPMS_CLK Frequencies for IGLOO2 Devices	39
Table 23	CLRHint[2:0]	40
Table 24	eSRAM Block Sizes and Address Ranges	42
Table 25	SRAM Organization in SECDED-ON Mode	43
Table 26	SRAM Organization in SECDED-OFF Mode	44
Table 27	Wait States in Different Operation Modes	45
Table 28	SYSREG Control Registers	52
Table 29	ESRAM_MAX_LAT	54
Table 30	eSRAM Maximum Latency Values	54
Table 31	ESRAM1_EDAC_CNT	55
Table 32	ESRAM0_EDAC_ADR	55
Table 33	ESRAM_PIPELINE_CR	55
Table 34	ESRAM0_EDAC_CNT	55
Table 35	ESRAM1_EDAC_ADR	56
Table 36	MM4_5_DDR_FIC_SECURITY/MM4_5_FIC64_SECURITY	56
Table 37	MM3_7_SECURITY	57
Table 38	MM9_SECURITY	57
Table 39	EDAC_SR	58
Table 40	CLR_EDAC_COUNTERS	58
Table 41	EDAC_IRQ_ENABLE_CR	59
Table 42	EDAC_CR	60
Table 43	AHB Bus Matrix Connectivity	62
Table 44	WRR Masters	68
Table 45	Pure Round Robin and Fixed Priority Arbitration Scenario for eSRAM1	69
Table 46	WRR and Fixed Priority Arbitration Scenario for eNVM_0	71
Table 47	WRR Arbitration Scenario for eSRAM_0 slave	71
Table 48	Master and Slave Pairing	74
Table 49	AHB Bus Matrix Register Map	77
Table 50	HPDMA Register Map	89
Table 51	HPDMAEDR_REG	90
Table 52	HPDMAD0SAR_REG	94
Table 53	HPDMAD1SAR_REG	94
Table 54	HPDMAD2SAR_REG	94

Table 55	HPDMAD3SAR_REG	94
Table 56	HPDMAD0DAR_REG	95
Table 57	HPDMAD1DAR_REG	95
Table 58	HPDMAD2DAR_REG	95
Table 59	HPDMAD3DAR_REG	95
Table 60	HPDMAD0CR_REG	96
Table 61	HPDMAD1CR_REG	97
Table 62	HPDMAD2CR_REG	98
Table 63	HPDMAD3CR_REG	99
Table 64	HPDMAD0SR_REG	100
Table 65	HPDMAD1SR_REG	100
Table 66	HPDMAD2SR_REG	101
Table 67	HPDMAD3SR_REG	102
Table 68	HPDMAD0PTR_REG	102
Table 69	HPDMAD1PTR_REG	103
Table 70	HPDMAD2PTR_REG	103
Table 71	HPDMAD3PTR_REG	104
Table 72	HPDMAICR_REG	104
Table 73	HPDMADR_REG	105
Table 74	SYSREG Control Registers	106
Table 75	RATIOHILO Field Definition	111
Table 76	Port List	111
Table 77	IGLOO2 FPGA PDMA Register Map	120
Table 78	Ratio_HIGH_LOW	123
Table 79	BUFFER_STATUS	123
Table 80	CHANNEL_x_CONTROL	125
Table 81	PERIPHERAL_SEL	126
Table 82	CHANNEL_x_STATUS	126
Table 83	CHANNEL_x_BUFFER_A_SRC_ADDR	127
Table 84	CHANNEL_x_BUFFER_A_DST_ADDR	127
Table 85	CHANNEL_x_BUFFER_A_TRANSFER_COUNT	127
Table 86	CHANNEL_x_BUFFER_B_SRC_ADDR	128
Table 87	CHANNEL_x_BUFFER_B_DST_ADDR	128
Table 88	CHANNEL_x_BUFFER_B_TRANSFER_COUNT	128
Table 89	SYSREG Control Registers	129
Table 90	SPI Interface Signals	132
Table 91	Data Transfer Modes	133
Table 92	Summary of Master SPI Modes	133
Table 93	Behavior of the Output Enable Signal	136
Table 94	Soft Reset Bit Definitions for SPI Peripheral	143
Table 95	SYSREG Control Registers	149
Table 96	SPI Register Summary	150
Table 97	CONTROL	151
Table 98	TXRXDF_SIZE	152
Table 99	STATUS	152
Table 100	INT_CLEAR	153
Table 101	RX_DATA	154
Table 102	TX_DATA	154
Table 103	CLK_GEN	154
Table 104	CLK_MODE Example, APB Clock = 153.8 MHz	154
Table 105	SLAVE_SELECT	155
Table 106	MIS	156
Table 107	RIS	156
Table 108	CONTROL2	157
Table 109	COMMAND	158
Table 110	PKTSIZE	159
Table 111	CMD_SIZE	159
Table 112	HWSTATUS	159
Table 113	STAT8	160

Table 114	COMM_BLK Register Map	171
Table 115	CONTROL	172
Table 116	STATUS	173
Table 117	INT_ENABLE	173
Table 118	DATA8	174
Table 119	DATA32	174
Table 120	FRAME_START8	174
Table 121	FRAME_START32	175
Table 122	VDD Power-Up to Functional Time	183
Table 123	DEVRST_N Power-Up to Functional Time	185
Table 124	Switch Register Map	195
Table 125	Register Types	198
Table 126	SYSREG	202
Table 127	Subset of System Registers	206
Table 128	ESRAM_MAX_LAT	206
Table 129	eSRAM Maximum Latency Values	207
Table 130	ENVM_CR	207
Table 131	SW_ENVMREMAPSIZE	209
Table 132	ENVM_REMAP_FAB_CR	209
Table 133	DDRB_BUF_TIMER_CR	210
Table 134	DDRB_NB_ADDR_CR	210
Table 135	DDRB_NB_SIZE_CR	210
Table 136	Non-Bufferable Region	211
Table 137	DDRB_CR	211
Table 138	EDAC_CR	212
Table 139	MASTER_WEIGHT0_CR	212
Table 140	MASTER_WEIGHT1_CR	213
Table 141	Programmable Weight Values	213
Table 142	SOFT_IRQ_CR	213
Table 143	SOFT_RESET_CR	214
Table 144	FAB_IF_CR	215
Table 145	MDDR_CR	215
Table 146	PERIPH_CLK_MUX_SEL_CR	216
Table 147	MDDR_IO_CALIB_CR	216
Table 148	EDAC_IRQ_ENABLE_CR	217
Table 149	ESRAM_PIPELINE_CR	217
Table 150	HPMS_PLL_STATUS_LOW_CR	218
Table 151	FACC_PLL_RANGE	219
Table 152	HPMS_PLL_STATUS_HIGH_CR	219
Table 153	HPMS_FACC1_CR	220
Table 154	Clock Ratio	222
Table 155	HPMS_FACC2_CR	222
Table 156	HPMS_CLK_CALIB_CR	224
Table 157	PLL_DELAY_LINE_SEL_CR	224
Table 158	RESET_SOURCE_CR	224
Table 159	DDRB_HPD_ERR_ADR_SR	225
Table 160	DDRB_SW_ERR_ADR_SR	225
Table 161	DDRB_BUF_EMPTY_SR	225
Table 162	DDRB_DSBL_DN_SR	226
Table 163	ESRAM0_EDAC_CNT	226
Table 164	ESRAM1_EDAC_CNT	226
Table 165	ESRAM0_EDAC_ADR	227
Table 166	ESRAM1_EDAC_ADR	227
Table 167	MM4_5_DDR_FIC_SECURITY/MM4_5_FIC64_SECURITY	227
Table 168	MM3_7_SECURITY	228
Table 169	MM9_SECURITY	229
Table 170	DEVICE_SR	229
Table 171	ENVM_PROTECT_USER	230
Table 172	ENVM_STATUS	231

Table 173	DEVICE_VERSION .....	231
Table 174	HPMS_PLL_STATUS .....	231
Table 175	ENVM_SR .....	232
Table 176	DDR_B_STATUS .....	233
Table 177	MDDR_IO_CALIB_STATUS .....	233
Table 178	HPMS_CLK_CALIB_STATUS .....	234
Table 179	FAB_PROT_SIZE .....	234
Table 180	Region Size .....	234
Table 181	FAB_PROT_BASE .....	235
Table 182	EDAC_SR .....	235
Table 183	HPMS_INTERNAL_SR .....	236
Table 184	HPMS_EXTERNAL_SR .....	236
Table 185	CLR_EDAC_COUNTERS .....	237
Table 186	FLUSH_CR .....	238
Table 187	Interrupt Line Signal Distribution .....	241
Table 188	FIIC Port List .....	242
Table 189	IGLOO2 FPGA FIIC Register Map .....	243
Table 190	INTERRUPT_ENABLE0 .....	243
Table 191	INTERRUPT_ENABLE1 .....	245
Table 192	INTERRUPT_REASON1 .....	245
Table 193	INTERRUPT_REASON0 .....	246
Table 194	INTERRUPT_MODE .....	248
Table 195	Number of FICs Available for Use in Each Device .....	249
Table 196	Fabric Interface Controller Port List .....	251
Table 197	FAB_IF Register in the SYSREG Block .....	262
Table 198	FDDR APB Slave Configuration Interface Port List .....	264
Table 199	MDDR APB Slave Configuration Interface Port List .....	264
Table 200	SERDERIF APB Slave Configuration Interface Port List .....	265
Table 201	HPMS APB Master Configuration Interface Port List .....	265

# 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

The following changes were made in revision 8.0 of this document.

- Information about the assertion of the POWER\_ON\_RESET\_N signal was added. For more information, see [Power-Up to Functional Time Sequence](#), page 180 and [VDD Power-Up to Functional Time Data](#), page 181.
- An example was added to show how power-on reset delay settings affect VDD calculation. For more information, see [VDD Power-Up to Functional Time Data](#), page 181.
- The PDMA port list was updated and descriptions added for each port. For more information, see [PDMA Port List](#), page 111.
- Information about SMC\_FIC was added. For more information, see [Configuring HPDMA](#), page 84.

## 1.2 Revision 7.0

The following changes were made in revision 7.0 of this document.

- Added [System Registers Behavior for M2GL005/010 devices](#), page 206 and [Table 127](#), page 206 (SAR 84564).
- Edited [Software Reset Control Register](#), page 214 to add a note on reset values (SAR 86186).
- Added a note to [Table 143](#), page 214, [Table 144](#), page 215, [Table 145](#), page 215, [Table 146](#), page 216, [Table 148](#), page 217, [Table 150](#), page 218, [Table 152](#), page 219, and [Table 153](#), page 220 (SAR 84564).
- Edited [Table 173](#), page 231 (SAR 86187).
- Edited [Data Retention Time](#), page 7, [Table 3](#), page 7, [eNVM Pages for Special Purpose Storage](#), page 20, [Table 8](#), page 20, [Table 9](#), page 20, [Table 21](#), page 39, [Table 19](#), page 38 in the [Embedded NVM \(eNVM\) Controllers](#), page 4 (SAR 86089).
- Edited [Security](#), page 16, [Figure 17](#), page 17, [Figure 18](#), page 17, [Figure 19](#), page 18, [Figure 20](#), page 18, [Figure 21](#), page 19 (SAR 86276).

## 1.3 Revision 6.0

The following changes were made in revision 6.0 of this document.

- Added M2GL060 device entry in [Table 2](#), page 6 and [Table 3](#), page 7 (SAR 78895).
- Added [Power-Up to Functional Time Data](#), page 181 (SAR 81548, SAR 73079).
- Updated HPDMA transfer information in [High Performance DMA Controller](#), page 78 (SAR 51843).
- Updated PDMA transfer information in [Peripheral DMA](#), page 107 (SAR 51843).
- Updated [eNVM Pages for Special Purpose Storage](#), page 20 for M2GL060 device (SAR 78895).
- Added a note in [How to Use eNVM](#), page 23 on the support of simulation models (SAR 80669).
- Added [Figure 20](#), page 18 that depicts eNVM Protected Region for the M2GL060 Device (SAR 78895).
- Updated the NV\_FREQRNG description in [Table 11](#), page 30 (SAR 62544).
- Added the [Register Lock Bits Configuration](#), page 200 (SAR 79855).
- Updated the REQACCESS description in [Table 11](#), page 30 (SAR 61550).
- Updated [Embedded NVM \(eNVM\) Controllers](#), page 4 chapter (SAR 78895).
- Added a note on DEVREST\_N state when it is tied low in [Power-On Reset Generation Sequence](#), page 177 (SAR 58928, SAR 58927).
- Added a note in [AHB Bus Matrix](#), page 61 on the support of simulation models (SAR 80669).
- Added a note in [How to Use FIC](#), page 257 on the support of simulation models (SAR 80669).
- Updated [Peripheral DMA](#), page 107 (SAR 80669).
- Updated [Serial Peripheral Interface Controller](#), page 130 chapter (SAR 80669).
- Added a note in [How to Use PDMA](#), page 113 on the support of simulation models (SAR 80669).
- Added the 060 device entry in [Table 195](#), page 249 (SAR 78911).

## 1.4 Revision 5.0

The following changes were made in revision 5.0 of this document.

- Updated [Power-Up to Functional Time Sequence](#), page 180 (SAR 72906).
- Updated [Table 16](#), page 34 and [Table 175](#), page 232 (SAR 70182).
- Updated [Figure 17](#), page 17, [Figure 18](#), page 17, [Figure 19](#), page 18, and [Figure 21](#), page 19, and added [eNVM Pages for Special Purpose Storage](#), page 20 (SAR 66208).

## 1.5 Revision 4.0

The following changes were made in revision 4.0 of this document.

- Updated [Reset Controller](#), page 176 (SAR 66766).
- Updated SAR 67010.
- Updated [Implementation Considerations](#), page 256 (SAR 64802).

## 1.6 Revision 3.0

The following changes were made in revision 3.0 of this document.

- Updated [Embedded NVM \(eNVM\) Controllers](#), page 4 (SAR 62858).
- Updated [Fabric Interface Controller](#), page 249 (SAR 62858).
- Removed all instances of and references to M2GL100 device from the [eNVM Controllers Features](#) section, [Table 1](#), page 5, [Table 2](#), page 6, [Table 20](#), page 39, and [Table 195](#), page 249 (SAR 62858).
- Updated the latest Libero screen shots as required (SAR 54023).

## 1.7 Revision 2.0

The following changes were made in revision 2.0 of this document.

- Updated [Embedded NVM \(eNVM\) Controllers](#), page 4 (SAR 55309).
- Updated [Table 170](#), page 229 (SAR 50729).
- Updated [Embedded SRAM \(eSRAM\) Controllers](#), page 41 (SAR 55309).
- Updated [High Performance DMA Controller](#), page 78 (SAR 55309).
- Updated [Peripheral DMA](#), page 107 (SAR 55309).
- Updated [Serial Peripheral Interface Controller](#), page 130 (SAR 55309).
- Added [How to Use the SPI Controller](#), page 145 (SAR 50196).
- Updated [Table 97](#), page 151 (SAR 50168).
- Updated [Communication Block](#), page 161 (SAR 55309, SAR 50310).
- Updated [Reset Controller](#), page 176 (SAR 55309).
- Updated [System Register Block](#), page 196 (SAR 55044).
- Updated [Fabric Interface Interrupt Controller](#), page 239 (SAR 55309).
- Updated [Fabric Interface Controller](#), page 249 (SAR 55309).

## 1.8 Revision 1.0

The following changes were made in revision 1.0 of this document.

- Updated [Embedded NVM \(eNVM\) Controllers](#), page 4 (SAR 50164, 50072).
- Updated [Table 5](#), page 9 (SAR 50533).
- Added [Single Word Read](#), page 12, [Consecutive Reads Incrementing through Memory](#), page 12, and [eNVM Program and Verify Operations Timing Diagrams](#), page 13 (SAR 50533).
- Added [How to Use eSRAM](#), page 47 (SAR 50165).
- Updated [Table 97](#), page 151 (SAR 50168).
- Modified [Table 24](#), page 42 (SAR 50276).
- Added [AHB Bus Matrix](#), page 61 (SAR 49904).
- Added [How to Use HPDMA](#), page 83 (SAR 50365).
- Updated [Peripheral DMA](#), page 107 (SAR 50365).
- Updated [Serial Peripheral Interface Controller](#), page 130 (SAR 50618).
- Updated [Figure 77](#), page 131 and [Table 90](#), page 132 (SAR 50770).
- Added [How to Use FIIC](#), page 242 section (SAR 50344).
- [Table 190](#), page 243 is updated (SAR 50558).

- Updated [Communication Block](#), page 161 (SAR 50616).
- Added [How to Use COMM\\_BLK](#), page 165.
- Added [How to Use FIC](#), page 257 section (SAR 50294).
- Added [FIC\\_2 \(APB Configuration Interface\)](#), page 263.

## 1.9 **Revision 0.0**

Revision 0 was the first publication of this document.

## 2 Embedded NVM (eNVM) Controllers

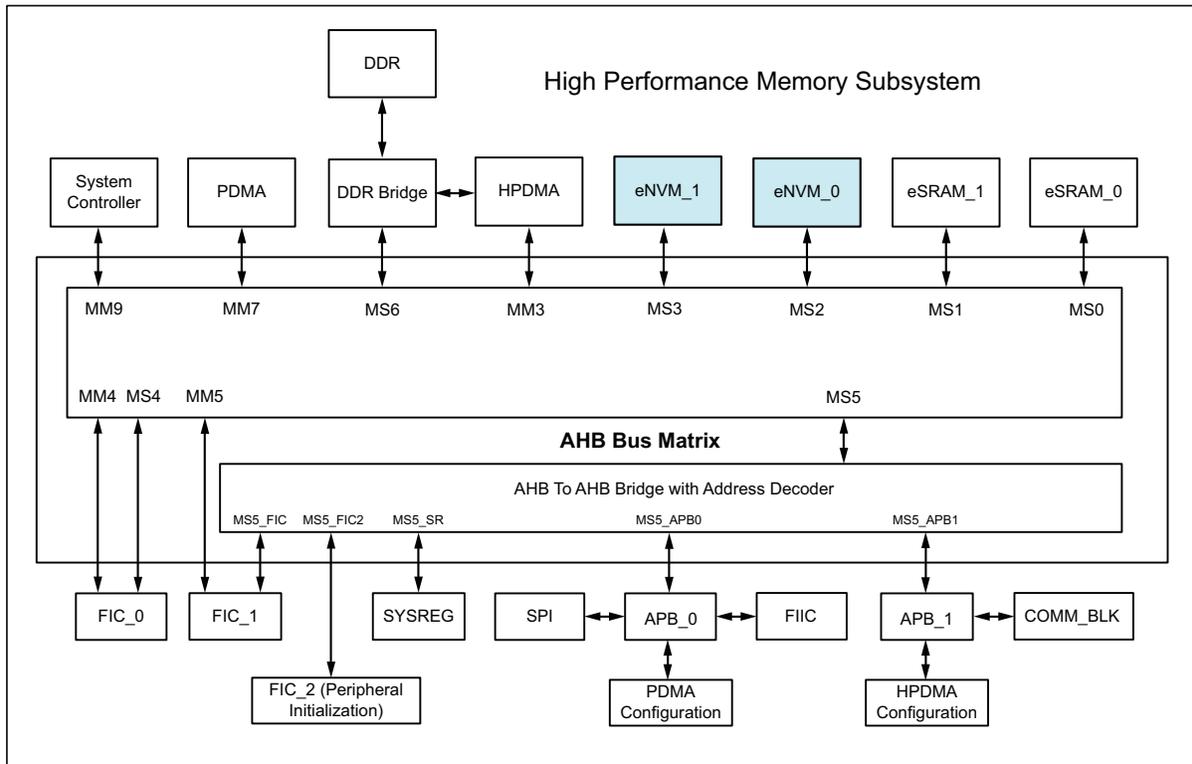
The IGLOO<sup>®</sup>2 FPGA devices have one or two embedded nonvolatile memory (eNVM) blocks (depending on the device) for user non-volatile memory. The eNVM controller interfaces these eNVM blocks to the advanced high-performance bus (AHB) bus matrix.

### 2.1 Features

- Single error correction and dual error detection (SECEDED) protected.
- Based on the selected IGLOO2 device, the total size of eNVM memory ranges from 128 KB, 256 KB, and 512 KB.
  - M2GL005 has a single block of 128 KB.
  - M2GL010, M2GL025, M2GL050, and M2GL060 have a single block of 256 KB.
  - M2GL090 and M2GL150 have two blocks of 256 KB each, the total eNVM memory size is 512 KB.
- In devices with two blocks present, any two masters can access the eNVM blocks (eNVM\_0 and eNVM\_1) in parallel, which improves the overall performance of the system.

As shown in the following figure, the eNVM block(s) is connected as slave to the AHB bus matrix.

**Figure 1 • eNVM Connection to AHB Bus Matrix**



## 2.2 Functional Description

The address range of eNVM\_0 is 0x60000000 to 0x6003FFFF and the address range of eNVM\_1 is 0x60040000 to 0x6007FFFF. The location of eNVM\_1 always follows eNVM\_0 in the system memory map. The following table gives the eNVM\_0 and eNVM\_1 addresses for different devices.

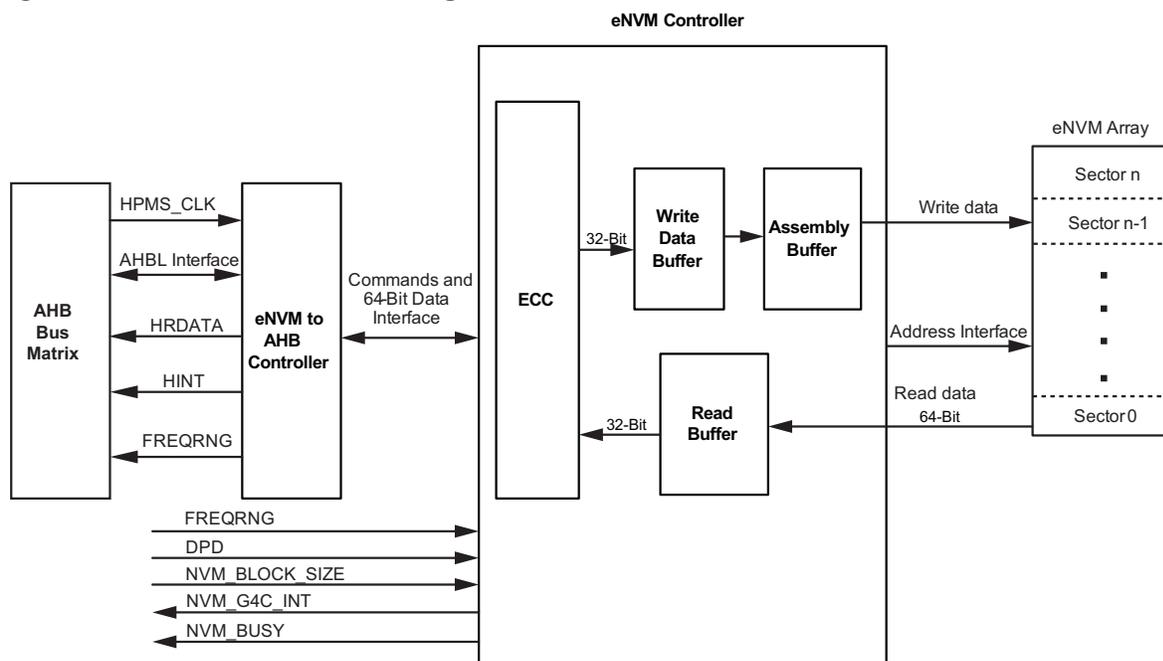
**Table 1 • eNVM Address Locations**

Device	eNVM_0	eNVM_1	Total NVM
M2GL005	0x60000000	None	128 Kbytes
M2GL010	0x60000000	None	256 Kbytes
M2GL025	0x60000000	None	256 Kbytes
M2GL050	0x60000000	None	256 Kbytes
M2GL060	0x60000000	None	256 Kbytes
M2GL090	0x60000000	0x60040000	512 Kbytes
M2GL150	0x60000000	0x60040000	512 Kbytes

Both eNVMs and embedded NVM controllers are identical and the eNVM controller consists of three components:

- eNVM Array
- eNVM Controller
- eNVM to AHB Controller

**Figure 2 • eNVM Controller Block Diagram**



HPMS\_CLK is used within the HPMS to clock the AHB bus matrix. See [UG0449: SmartFusion2 and IGL002 Clocking Resources User Guide](#) for more information on HPMS\_CLK.

**eNVM Array:** The eNVM array is connected to a 25 MHz internal oscillator. This 25 MHz internal oscillator is used during device start up to initialize the NVM controller. It is also used for eNVM program operation. For other eNVM operations (Read and Verify), the eNVM controller operates at the HPMS\_CLK. During eNVM read operations, the NVM controller uses the NV\_FREQRNG input to insert wait states to match with the eNVM array access times. The eNVM array stores the data. [Table 2](#), page 6 shows the eNVM memory organization and the total size of the eNVM.

**eNVM Controller:** Decodes all transactions from the AHBL master and issues the commands to the eNVM array.

**ECC:** The error-correcting code (ECC) block in eNVM Controller performs the SECDED. The ECC stores error correction information with each block to perform SECDED on each 64-bit data word. ECC does not consume any eNVM array bits. See [Table 19](#), page 38 for ECC status information. ECC block in eNVM Controller is enabled by default. The user has no access to control the ECC block.

**Read Data Buffer:** Contains four 64-bit data words. It functions as a small cache by reading NVM data as four consecutive 64-bit data words. Data read from the eNVM is stored in read data buffer (RDBUFF) and presented to AHB read data bus (HRDATA) corresponding to HADDR.

If the data is not available, an eNVM read cycle is invoked to retrieve data from the eNVM array. To support an 8-bit fixed length wrapping burst, four eNVM read cycles are automatically invoked and data read from the eNVM is stored in RDBUFF. Read data is presented to HRDATA when the data for the current read address becomes available.

**Assembly Buffer (AB):** The eNVM is page-based flash memory. Only one page of data (1,024 bits) can be written at a time. The assembly buffer stores thirty-two 32-bit data words for programming. During programming, the assembly buffer cannot be updated. If more than one page is to be written, the page programming function needs to be called as many times as the number of pages.

**Write Data Buffer:** The write data buffer provides a secondary 32-word data buffer. This can be updated with the next 32 words to be programmed during eNVM programming.

**eNVM to AHB Controller:** This block interfaces the eNVM Controller with the AHB-Lite (AHBL) master as shown in [Figure 2](#), page 5.

## 2.2.1 Memory Organization

The eNVM is divided into sectors based on the eNVM size. Each sector is divided into 32 pages. Each page holds 1,024 bits of data. The following table shows the total available memory and its organization.

**Table 2 • Memory Organization**

Device	NVM Size	Number of Sectors	Pages per Sector	Bytes per Page	Words per Page	64-Bit Locations per Page	Total Bytes
M2GL005	128 KB	32	32	128	32	16	131072
M2GL010	256 KB	64	32	128	32	16	262,144
M2GL025	256 KB	64	32	128	32	16	262,144
M2GL050	256 KB	64	32	128	32	16	262,144
M2GL060	256 KB	64	32	128	32	16	262,144
M2GL090	512 KB (two eNVMs, each 256 KB)	64 per NVM	32 per NVM per sector	128	32	16	262,144 per NVM
M2GL150	512 KB (two eNVMs, each 256 KB)	64 per NVM	32 per NVM per sector	128	32	16	262,144 per NVM

## 2.2.2 Data Retention Time

The following table shows the retention time of the eNVM with respect to the number of programming cycles. The same values are applicable for both commercial and industrial IGLOO2 product grades. See [DS0128: IGLOO2 and SmartFusion2 Datasheet](#) for more information on programming cycles and retention time.

**Table 3 • Data Retention Time**

Programming Cycles Per eNVM Page	Retention
< 1000	20 years
< 10000	10 years

**Note:** The eNVM is not prevented from programming, even if a page exceeds the write count threshold. The eNVM Controller generates a flag through Status register (see [Table 18](#), page 34).

## 2.2.3 eNVM Access Time

See the **Embedded NVM (eNVM) Characteristics** section from [DS0128: IGLOO2 and SmartFusion2 Datasheet](#) for eNVM Maximum Read Frequency and eNVM Page Programming Time.

## 2.2.4 Theory of Operation

The eNVM AHB Controller supports the following operations:

- Interface from AHBL for read, write, and erase operations
- Issues all eNVM commands through AHBL read and write bus operation. The data width to and from AHBL bus is 32 bits, and data to and from eNVM is 64 bits.
- Assembly buffer (AB) can be read directly from AHBL bus.
- eNVMs treated as ROM. AHBL write transactions to eNVM user data array receive errors on HRESP and write will be ignored.
- Page Program command is used to write the NVM user data array.
- AB can be written directly or loaded from the write data buffer (WDBUFF). Data can be written to WDBUFF in byte, half-word or word AHB transfers.
- Data for Page Program comes from WDBUFF or user data previously written into AB.
- Command codes in [Table 6](#), page 9 determine the NVM commands to be issued. The eNVM user data array is treated as ROM, so any program operations must be performed by submitting relevant commands to the controller. Any AHBL writes to NVM user data without a valid NVM command will cause the HRESP signal to be asserted on the AHBL bus. Any data that needs to be written into the NVM user array must be uploaded first to the WDBUFF and then written into the NVM user array through the assembly buffer. Program operation for the NVM user array occurs at the page boundaries.

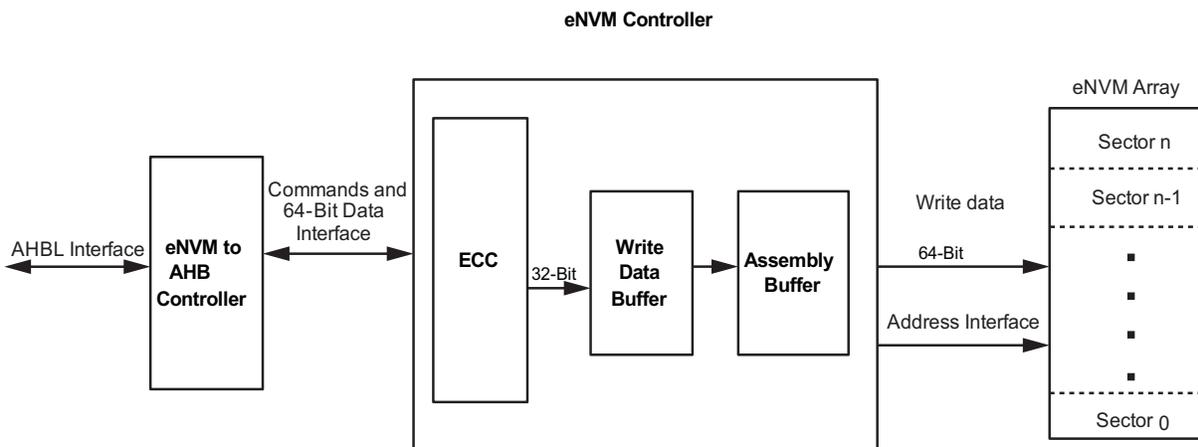
### 2.2.4.1 Write Control

The following steps describe eNVM write control.

- The data to be programmed into eNVM must first be uploaded into WDBUFF due to the width difference between the AHBL bus and the eNVM. Data can be written into WDBUFF by word, half-word, or byte from the AHBL bus. ProgramDa and ProgramADS commands take care of uploading data into AB from WDBUFF before programming eNVM.
- Data is sent to eNVM from WDBUFF in chunks of double words (64 bits). Subsequent data transfer commands to the AB and then to eNVM array, or commands such as ProgramAd, ProgramDa, and ProgramStart, must specify the page address and upload data to AB to start eNVM array programming. See [Table 6](#), page 9 for more information on commands.

The following figure shows the eNVM array write path.

**Figure 3 • Write Path**



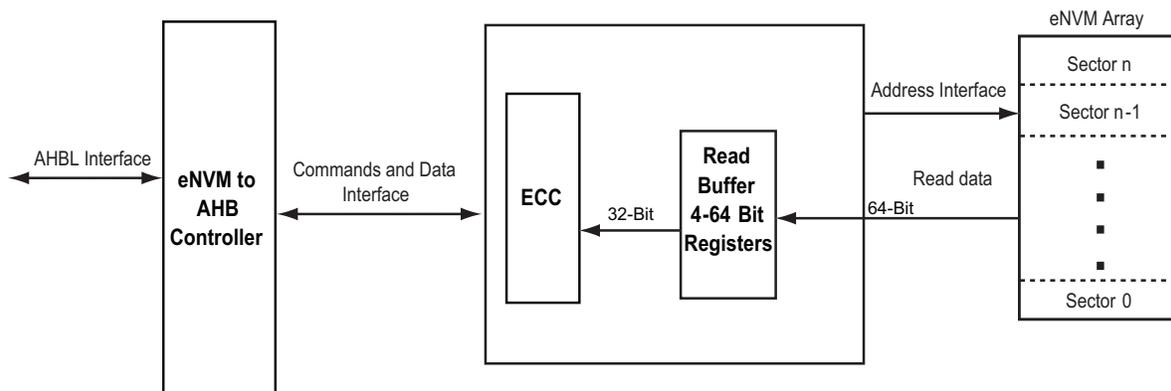
### 2.2.4.2 Read Control

The following steps describe eNVM read control.

- The read transaction from the eNVM user array to AHBL bus uses the read data buffer as a mini cache.
- If the requested 32-bit word exists in the read data buffer, it will be returned immediately on the AHB bus; otherwise a 64-bit read access of the eNVM is initiated and will take several clock cycles as configured by ENVM\_CR register (see Table 11, page 30).
- The eNVM data is stored in the read data buffer and provided to the AHB bus. Assuming that the eNVM address is incremented, the data value stored in the read data buffer is available for the next AHB read cycle.

The following figure shows the eNVM array read path.

**Figure 4 • Read Path**



In the eNVM array, the addresses are 64-bit locations; therefore each page of 1,024 bits (16 double words = 32 words) requires an AHBL address map, as specified in the following table.

**Table 4 • AHBL Address Map to NVM**

Sector Number	Page Number in Sector	Address in Page	Byte Number in 64-Bit Data
HADDR[17:12]	HADDR[11:7]	HADDR[6:3]	HADDR[2:0]

When programming the eNVM, sector and page addresses must be programmed into the command (CMD) register, as specified in Table 5, page 9.

### 2.2.4.3 eNVM Commands

The eNVM commands are explained in the [Table 6](#), page 9. The eNVM Command Register is used to program the eNVM commands. The following section explains the details of the eNVM Command Register.

### 2.2.5 eNVM Command Register

The following table shows the Command Register bit definitions.

**Table 5 • Command (CMD) Register**

Bit	Description
[31:24]	Command code
[23:0]	Address field; to supply address for NVM operation (see <a href="#">Table 6</a> , page 9)

The Command Register is located at offset 0x148 in the Control Register. See [Table 18](#), page 34 for more information. By writing to CMD when HADDR[18:0] = 0x148, any eNVM operation may be invoked. The eNVM goes into a busy state and HREADY is set High until it finishes the write operation. Any further invoking of the eNVM operation will cause HREADY to go Low until it finishes the previous operation.

The following steps describe when to write to the Command Register, decoding of commands and command execution.

- The Command Register should only be written when the NVM is non-busy (Status Register bit 0. See [Table 19](#), page 38 for the Status Register definitions.).
- If the Command Register is written when the NVM is still busy from a previous command then the logic will prevent the new command and all future commands, the access\_denied bit in the STATUS Register will be set. To recover from this state, 1 should be written to bit 1 in the CLRHINT[2:0] register (see [Table 23](#), page 40) to clear the access\_denied bit. This mechanism is used to detect the improper NVM command sequences and protect the NVM data until the firmware recovers.
- When the AHBL triggers a write transaction with HADDR[18:0] = 0x148, HWDATA is treated as a command (CMD).
- CMD[31:24] will be decoded as the eNVM operation, as mentioned in [Figure 4](#), page 8.
- The value from CMD[23:3] will be decoded as the NVM array address for the eNVM operation. Depending on the command code, some LSB bits of CMD[23:0] will be ignored. For example, to submit a program address, only the page address CMD[17:7] is significant. Therefore CMD[17:7] is taken as the NVM address and CMD[6:0] is ignored. See [Table 6](#), page 9 for more information.

For masters, which are only capable of byte access, four cycles of write may be needed to fill the Command (CMD) Register, by writing to 0x14b, 0x14a, 0x149, and 0x148.

**Table 6 • Command Table**

Name	HADDR		HWDATA		Transaction Type	Description
	18	17:0	31:24	23:0		
Read Page	0	AA	X	X	Read	
ProgramAd	1	ACMD	05	PGA	Write	Submit page address for programming. CMD[17:7] is considered as the eNVM address and CMD[6:0] is ignored.
ProgramDa	1	ACMD	06	AAB	Write	Submit data to assembly buffer for programming, up to 16 dwords can be written to the assembly buffer as specified by DWSIZE. ProgramDa must be preceded by ProgramAd. CMD[17:7] is considered as the eNVM address and CMD[6:0] is ignored.

**Table 6 • Command Table (continued)**

Name	HADDR		HWDATA		Transaction Type	Description
	18	17:0	31:24	23:0		
ProgramStart	1	ACMD	07	X	Write	Start program NVM operation
ProgramADS	1	ACMD	08	PGA	Write	Start whole program page procedure, includes sending page address, sending entire content of write data buffer to assembly buffer, then starting the NVM operation.
VerifyAd	1	ACMD	0D	PGA	Write	Submit page address for standalone verify. CMD[17:7] is taken as the eNVM address and CMD[6:0] is ignored.
VerifyDa	1	ACMD	0E	AAB	Write	Submit data to assembly buffer for standalone verify. Up to 16 dwords can be written to the assembly buffer, as specified by DWSIZE. VerifyDa must be preceded by the VerifyAd. CMD[6:3] is taken as the starting double word address and CMD[23:7] is ignored.
VerifyStart	1	ACMD	0F	X	Write	Start standalone verify NVM operation
VerifyADS	1	ACMD	10	PGA	Write	Start whole standalone verify procedure; includes sending page address, sending entire content of write data buffer to assembly buffer, and then starting NVM operation.
User Unlock			13	X	Write	Submit a User Unlock NVM command before Program NVM.

**Notes:**

- AA = NVM Array address. See [Table 1](#), page 5.
- AAB = Address of assembly buffer. See [Table 18](#), page 34 for address values.
- ACMD = Address of CMD register. The Command register is located at offset 0x148 in the Control Register. See [Table 18](#), page 34 for more information.
- PGA = Page address
- SEA = Sector address
- X = Not used

### 2.2.5.1 Read Page

Data read from eNVM is stored in the read data buffer (eight 32-bit memory blocks) and presented to HRDATA based on HADDR[2:0]. For non-sequential reads, the read data buffer is checked first. If the data is available, it is presented to HRDATA; otherwise an eNVM read cycle is invoked to read the data from the eNVM array and data is presented to HRDATA as soon as corresponding data is available. To support 8-byte fixed length burst (that is, to read the complete read data buffer, which consists of eight 32-bit memory blocks), 4 eNVM read cycles (each 64-bit) are automatically invoked. Data read from the eNVM is stored in the read data buffer.

### 2.2.5.2 Page Program

This mode allows writing the page with pre-erase. In Page Program there are three stages:

- **ProgramAd:** This command is used to submit the page address to be programmed.
- **ProgramDa:** Once the ProgramAd command is issued, data can be written to AB.
- **ProgramStart:** After ProgramAd and ProgramDa (optional), ProgramStart can be used to start the NVM operation. Once the NVM operation starts and until it finishes, any further NVM accessing AHBL transaction will result in HREADYOUT going Low until the operation is done.

If the command ProgramDa is not issued after the ProgramAd operation, the current data in the assembly buffer will be programmed to the NVM array.

#### 2.2.5.2.1 Program Page with a Single AHBL Write

- **ProgramADS:** During the command ProgramADS, a single AHBL write transaction can be used to start and complete the program page procedure. By default, all WDBUFF content is written to AB and internal program operation automatically begins.

#### 2.2.5.3 Standalone Verify

This mode allows verifying the contents of a page. In verify there are three stages:

- **VerifyAd:** This command is used to submit the page address to be verified.
- **VerifyDa:** Once the VerifyAd command is issued, data can be written to AB.
- **VerifyStart:** After VerifyAd and VerifyDa (optional), VerifyStart can be used to start the NVM operation. Once the NVM operation starts and until it finishes, any further NVM accessing AHBL transaction will result in HREADYOUT going Low until the operation is done. If the VerifyDa command is not issued after the VerifyAd operation, the current data in the assembly buffer is verified with the NVM array.

#### 2.2.5.3.1 Standalone-Verify with a Single AHBL Write

**VerifyADS:** With the command VerifyADS, a single AHBL write transaction can be used to start and complete the verify page procedure. By default, all WDBUFF content is written to AB and the internal Standalone-Verify operation automatically starts.

#### 2.2.5.4 Set Lock Bit and User Unlock Commands

There is a user page lock bit to lock the page for writing. The Control Register PAGE\_LOCK\_SET[0] is used to set the user lock bit of the page. See NV\_PAGE\_STATUS register in [Table 18](#), page 34 for more information. If PAGE\_LOCK\_SET[0] == 1, then the nv\_s\_page\_lock\_set signal will be asserted when submitting the address for Program.

To program a page, the User Unlock command must be submitted before submitting ProgramAd or ProgramADS.

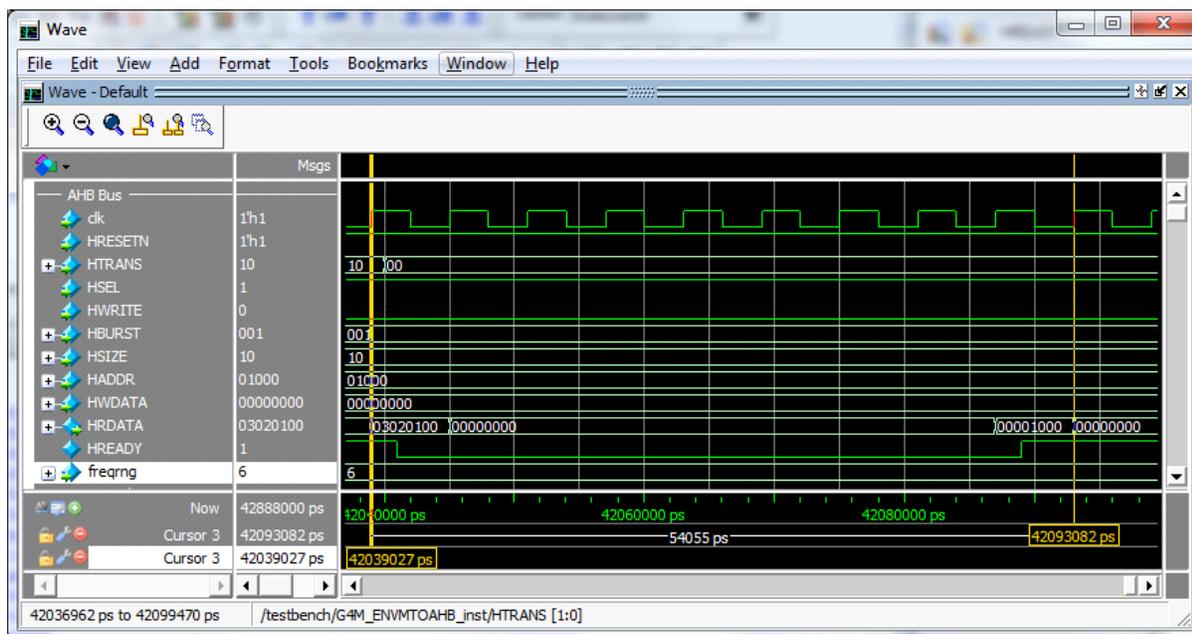
## 2.2.5.5 eNVM Read Operations with Timing Diagrams

The following are the example eNVM read operations with the HPMS clock at 166 MHz and NVM FREQRNG is set to the value of 6.

### 2.2.5.5.1 Single Word Read

The following figure shows the AHB read command to 0x60001000 starting at the first cursor, and data being returned at the second cursor 9 clock cycles later.

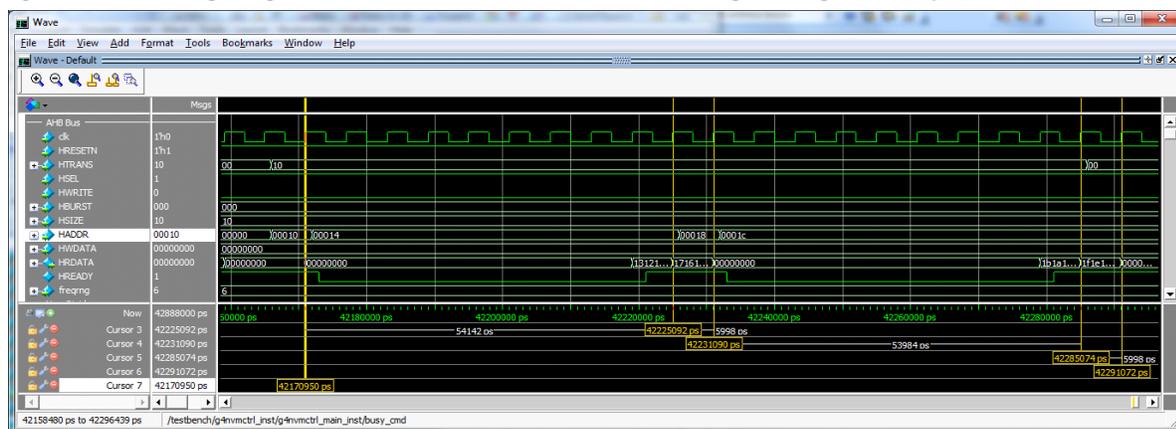
Figure 5 • Timing Diagram for Single Word Read Operation



### 2.2.5.5.2 Consecutive Reads Incrementing through Memory

In this case, four reads from addresses 0x60000010, 0x60000014, 0x60000018, and 0x6000001C are initiated by the AHB master in succession. The first word is returned 9 clock cycles later (as shown in the preceding figure), but the second word occurs in the following cycle, 9 clock cycles later the third word is provided and the fourth word occurs in the next clock cycle. This pattern is repeated as the memory is incremented as shown in the following figure.

Figure 6 • Timing Diagram for Consecutive Reads Incrementing Through Memory



### 2.2.5.6 eNVM Program and Verify Operations Timing Diagrams

Timing diagrams in this section illustrate eNVM Program and Verify operations at the AHB bus transfer level with the fabric master operating at 166 MHz. The eNVM NV\_FREQRNG is set to 15. The sample eNVM operation programs the eNVM sector 0 page 4 with random data and verifies the eNVM sector 0 page 4.

**Note:** In all the waveforms, the eNVM controller register offset is shown in AHB address line (HADDR). See [eNVM Control Registers](#), page 34 for more information.

#### 2.2.5.6.1 Sequence of eNVM Program and Verify Operations when using ProgramADS and VerifyADS Commands

The following figure shows the following sequence of eNVM ProgramADS and VerifyADS commands:

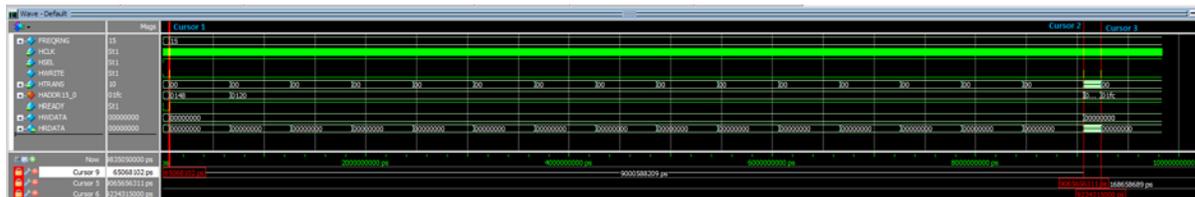
1. Fabric master requests for exclusive register access by writing 0x1 to the REQACCESS register.
2. Fills the WDBUFF (Write Data Buffer) register with the data to be written to the eNVM array.
3. Issues ProgramADS command.
4. Completes the eNVM Program operation and starts the eNVM Verification by issuing a VerifyADS command.
5. Completes the eNVM verify operation.
6. Releases the exclusive register access by writing 0x0 to the REQACCESS register.

The status of the eNVM operations are monitored by polling the Status register response.

For a description of each register, see [Table 18](#), page 34.

The following figure shows the complete eNVM program (ProgramADS) and eNVM verify (VerifyADS) operations.

**Figure 7 • eNVM Program and Verify Operations**

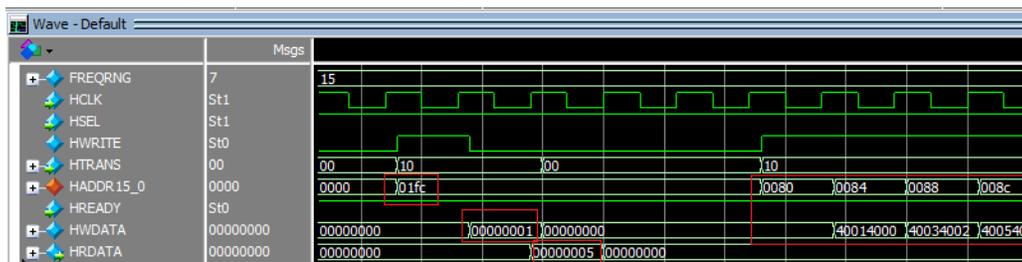


At cursor 1, step 1 and step 2 of the sequence are performed. At cursor 2, the eNVM Program operation gets completed and Verify operation gets started. At cursor 3, the verify operation is completed (see the preceding figure).

The following figures ([Figure 8](#), page 13 through [Figure 11](#), page 14) show the eNVM commands sequence in waveforms.

The fabric master gets the exclusive register access by writing 0x1 to the REQACCESS register. It reads the value 0x5 from AHB read data line (HRDATA), it means the exclusive register access is issued. Then the WDBUFF (Write Data Buffer) register is filled with the random data, as shown in the following figure.

**Figure 8 • Exclusive Register Access and Filling Data in WDBUFF**

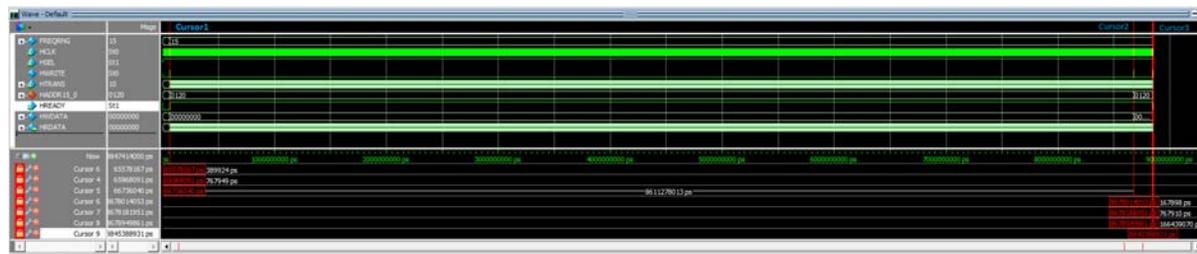




The status of the eNVM operations is monitored by polling the Status register response.

The following figure shows the complete eNVM program and eNVM verify operations.

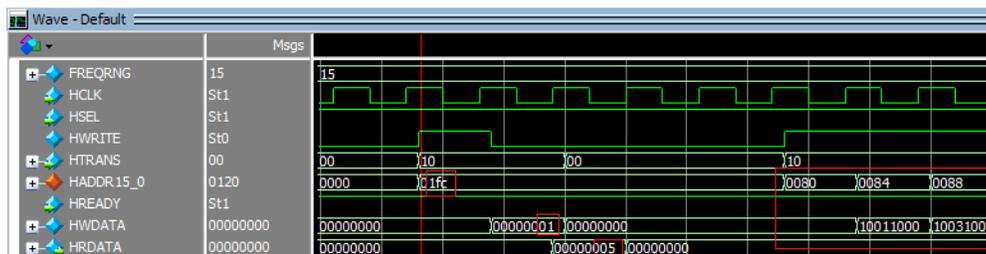
**Figure 12 • Complete eNVM Program and Verify Operations Waveform**



At cursor 1, step 1 and step 2 of the sequence are performed. At cursor 2, the eNVM ProgramStart operation is completed and VerifyAD operation is started. At Cursor 3, the verify operation is completed (see the preceding figure). The eNVM commands sequence is explained in waveforms.

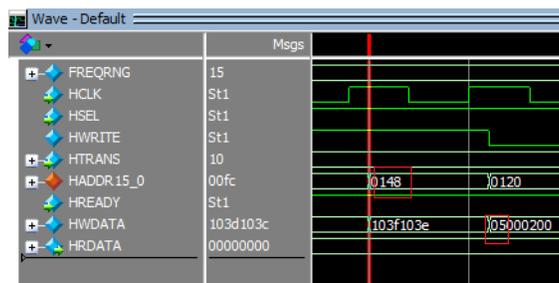
The following figure shows the fabric master requesting for exclusive register access and filling WDBUFF (Write Data Buffer).

**Figure 13 • Exclusive Register Access and Filling Data in WDBUFF**



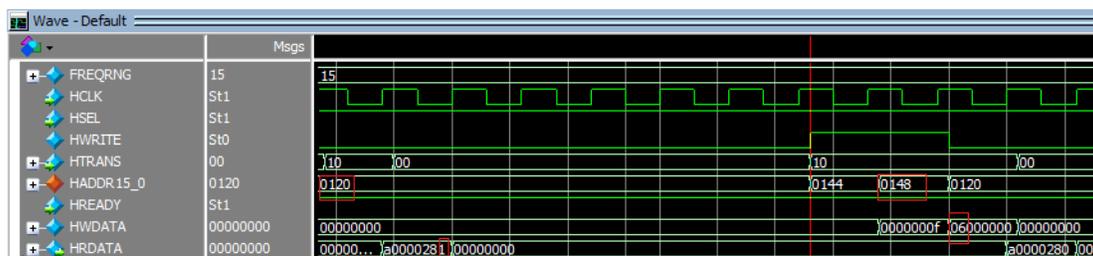
The following figure shows issue of ProgramAD command.

**Figure 14 • ProgramAD Command**



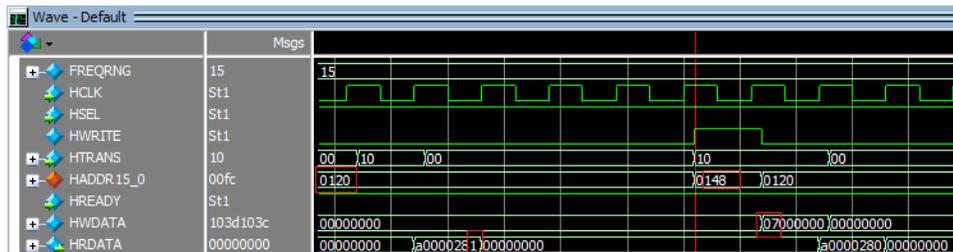
The following figure shows completion of ProgramAD command and issue of ProgramDA command.

**Figure 15 • ProgramDA command**



The following figure shows completion of ProgramDA command and issue of ProgramStart command.

**Figure 16 • ProgramStart Command**



The completion of the eNVM command is confirmed by monitoring the eNVM status register for eNVM ready and the next command in sequence is sent. VerifyAD, VerifyDA, and VerifyStart commands are issued by writing corresponding command value into CMD register.

### 2.2.6 Error Response

The error response, which is indicated by the HRESP signal, is asserted if any of the following conditions occur:

- AHBL burst read is terminated early or address sequence is not as expected. This should never occur within the system during normal operation.
- AHBL write transaction addressed to read-only user data array
- AHBL read or write transaction to a protected memory area. See [Security](#), page 16

Data on HRDATA with error response is zero. A write transaction addressed to read-only Control Register such as RD or RDT will not trigger an error response. However, the data in these registers will not be affected.

### 2.2.7 Interrupt to Fabric Master

Setting the Control Registers INTEN[10:0] as shown in [Table 18](#), page 34 allows the user to configure HINT to assert an interrupt on any active status events from eNVM, such as the assertion of any status bit from eNVM or when an internal eNVM operation ends.

After HINT is asserted, the fabric master determines the next steps. The fabric master can respond to the interrupt and then clear HINT by writing 1 to bit 0 of the write-only register CLRHINT[2:0] (HADDR = 0x158) in [Table 18](#), page 34. If the fabric master decides to ignore the interrupt (by masking it out), the interrupt is cleared if read or write continues and the interrupt-triggering events are not re-occurring. If the same triggering event happens again, HINT will remain asserted.

## 2.3 Security

The eNVM is protected using four levels of security features:

- The eNVM page protection uses two levels: factory lock and user lock. Factory lock is not accessible for the user. See [Set Lock Bit and User Unlock Commands](#), page 11.
- There are two or four special sectors per eNVM array that can be protected for read and write, depending on which entity is accessing the region as shown in [Figure 17](#), page 17 through [Figure 21](#), page 19. On devices with smaller or bigger eNVMs, the upper 4 KB special sector is aligned to the top 4 KB region of the eNVM. These user protectable 4 KB special sectors, can be configured by Libero<sup>®</sup> software. See [Figure 28](#), page 27 for more details.
- There are two private regions in M2GL060, M2GL090, and M2GL150 as shown in [Figure 20](#), page 18 and [Figure 21](#), page 19 which are reserved for storing device certificate, eNVM digest, security keys, and so on. Only system controller can access the private regions. See [eNVM Pages for Special Purpose Storage](#), page 20 for more details.
- Using AHB bus master access control, the eNVM can be protected from different masters connected on the AHB bus matrix. See [AHB Bus Matrix](#), page 61.
- User-defined regions can be protected from the FPGA fabric.

### 2.3.1 User-Protectable 4K Regions

Figure 17 • eNVM Special Sectors for the M2GL050TS Device with 256 KB eNVM\_0

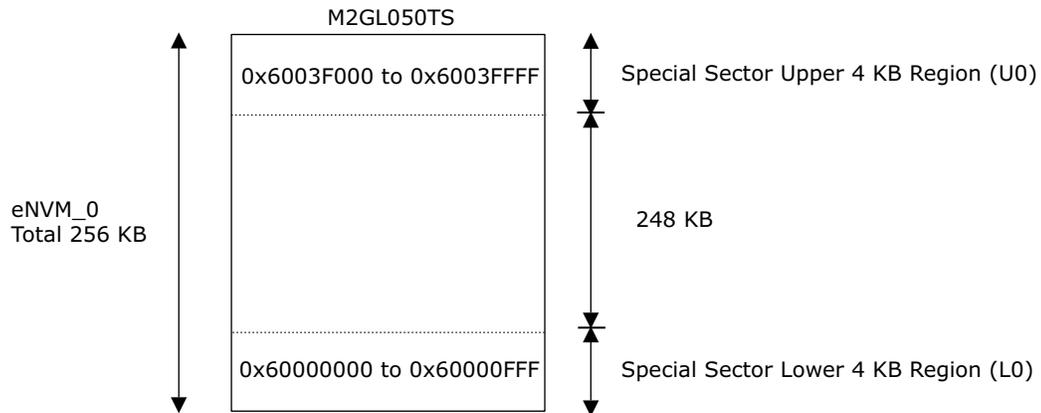
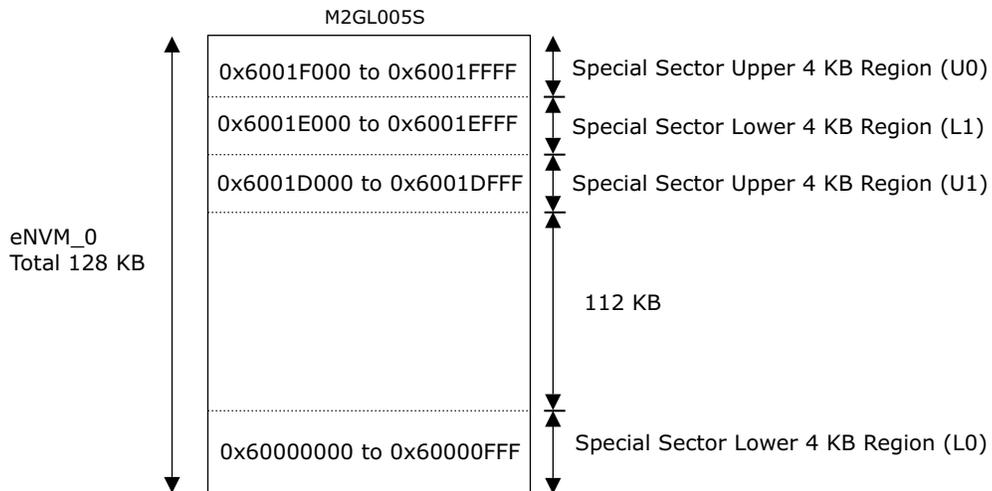
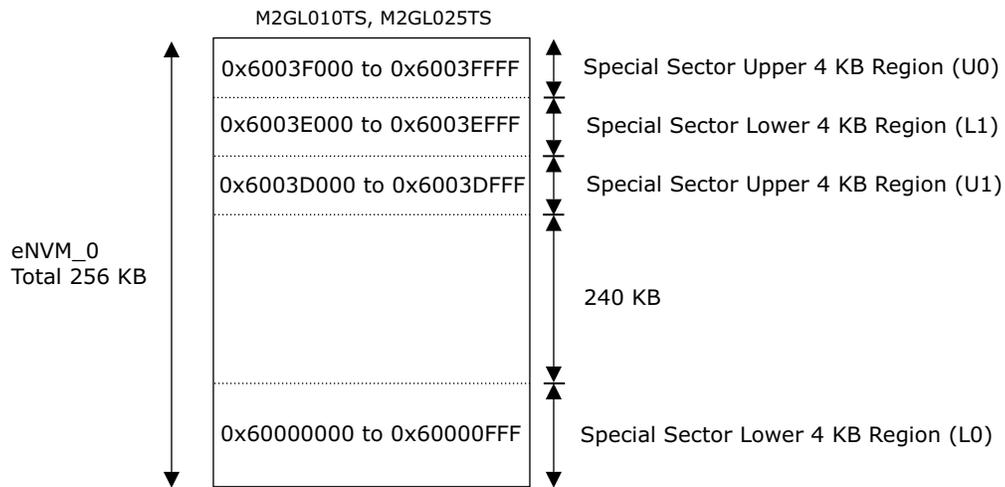


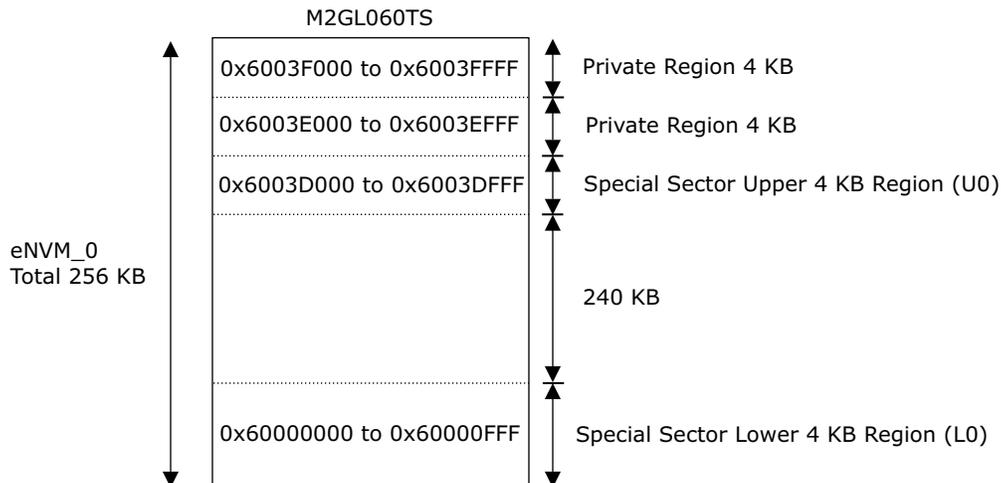
Figure 18 • eNVM Special Sectors for the M2GL005S Device with 128 KB eNVM\_0



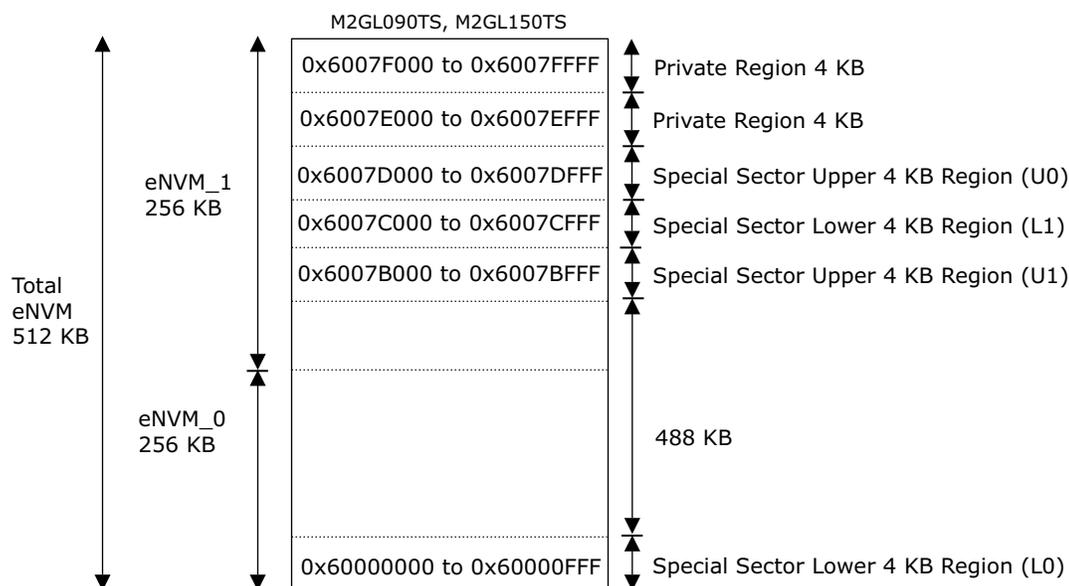
**Figure 19 • eNVM Special Sectors for M2GL010TS and M2GL025TS Devices with 256 KB eNVM\_0**



**Figure 20 • eNVM Special Sectors for the M2GL060TS Device with 256 KB eNVM\_0**



**Figure 21 • eNVM Special Sectors for M2GL090TS and M2GL150TS Devices with 512 KB**



The security configuration is provided as input to the eNVM Controller from system registers as per the ENVM\_PROTECT\_USER register described in Table 10, page 29 for configuration of upper and lower regions of NVM. The following table lists user protection regions for different masters.

**Table 7 • User Protection Regions**

Master	Function
Fabric master	FIC_0 can access the protected memory regions. Access bit defines the read accessibility. Write allowed bit indicates that the masters which have read access can also have write access.
Other masters (PDMA and HPDMA)	All other masters are allowed access. Access bit defines the read accessibility.

### 2.3.1.1 Read Protection

When AHB masters other than the system controller issue read transactions to protected regions, the address and protection configuration is checked to determine whether the read is targeted to the protected region and if the read is allowed. If the read is not allowed, the eNVM read command is not sent to the eNVM and an error is generated. For a specific AHB master to read a protected region, both the factory and user allowed bits must be set. See Table 14, page 33 for information on eNVM access controls for AHB masters.

### 2.3.1.2 Write Protection

When AHB masters other than system controller issue write transactions (which may be one of the program commands supported by this interface) to protected regions, the address and protection configuration is checked to determine whether the transaction is targeted to the protected region. If the transaction is not allowed, no command is sent to eNVM and the Status bit (see Table 18, page 34) is asserted.

### 2.3.1.3 Power-Down

During device startup, the eNVM(s) will be powered up as the fabric is powered up. As soon as the fabric is active, if the user sets the deep power down (DPD) bit, the NVM(s) will be powered down. Each eNVM block can be put into deep power down mode by configuring the SYSREG. The eNVM can permanently be switched on or switched off. See ENVM\_CR register (Table 10, page 29) for configuration settings.

During Flash\*Freeze, users may want to put the NVM(s) into deep power down mode, to save power. The user should not enter power down while the NVM is in use. DPD is not entered automatically when Flash\*Freeze is entered.

**Note:** Flash\*Freeze applies mainly to the fabric.

### 2.3.2 eNVM Pages for Special Purpose Storage

A few pages in the final sectors (N-1 and N-2) of the last eNVM module are used for special purpose storage like device certificate, eNVM digest and peripheral initialization configuration data for SerDes, FDDR and MDDR. Some special purpose pages are reserved and protected. See [Table 8](#), page 20 and [Table 9](#), page 20 for more information on eNVM special purpose storage based on IGLOO2 device density. The system controller performs read/write operations on unreserved eNVM pages using system controller services. It only reads data from reserved eNVM pages. 48 pages in the final sectors of eNVM\_0 module for M2GL005, M2GL010, M2GL025, and M2GL050 devices are used for special purpose storage as listed in the following table.

**Table 8 • Special Purpose Storage Regions**

Device	eNVM module	Sector	Page	Type	Usage
M2GL005/M2GL010/ M2GL025/M2GL050	eNVM_0	N-2	16-31	Unreserved	Peripheral initialization configuration data for SerDes, FDDR and MDDR
		N-1	0-15	Unreserved	
			16-24	Reserved	Reserved for future use
			25-30	Unreserved	Device Certificate
			31	Unreserved	Digest for eNVM_0

64 pages of eNVM in the final 2 sectors (private regions) of the last eNVM module for M2GL060, M2GL090, and M2GL150 devices are used for special purpose storage like device certificate, eNVM digest and security keys. 32 pages in N-3 sector of the last eNVM module are used as peripheral initialization configuration data for SerDes, FDDR and MDDR. For more information, see [Table 9](#), page 20. The M2GL060 device has two private regions in eNVM\_0 and the M2GL090/M2GL150 device has two private regions in eNVM\_1.

**Table 9 • Special Purpose Storage Regions for M2GL060, M2GL090 and M2GL150 Devices**

Sector in eNVM	Page	Type	Usage	Offset in page (Bytes)	Range (Bytes)
N-3	31-0	Unreserved	Peripheral initialization configuration data for SerDes, FDDR and MDDR	0	4095:0

**Table 9 • Special Purpose Storage Regions for M2GL060, M2GL090 and M2GL150 Devices (continued)**

N-2	20-0	Unreserved	User Key Code#2 to User Key Code #N. N can be maximum 58. Maximum 56 Key Codes (KC#2 to KC#58),each occupies 48 Bytes Minimum 5 Key Codes (KC#2 to KC#7), each occupies 528 Bytes	0	2687:0
	29-21	Unreserved	User Activation Code	0	1151:0
	30	Unreserved	User Activation Code (Total 1192 bytes across page 21 to page 30)	0	39:0
	30	Unreserved	User Defined (Key sizes + Exported bit + Valid bit) byte array: 56 bytes holds 56 key sizes along with exported and valid bit flags.	40	55:0
	30	Unreserved	Reserved for future use	96	31:0
	31	Unreserved	User PK-X (384-bit User PUF ECC Public Key)	0	47:0
	31	Unreserved	User PK-Y (384-bit User PUF ECC Public Key)	48	47:0
	31	Unreserved	User Activation Code exported flag (Digests Valid, Activation Code missing)	96	1 byte
	31	Unreserved	User Activation Code valid flag	97	1 byte
	31	Unreserved	User Key Code #0 exported flag (Digests Valid, Key Code missing)	98	1 byte
	31	Unreserved	User Key Code #0 valid flag	99	1 byte
	31	Unreserved	User Key Code #1 exported flag (Digests Valid, Key Code missing)	100	1 byte
	31	Unreserved	User Key Code #1 valid flag	101	1 byte
	31	Unreserved	User Public Key valid flag	102	1 byte
	31	Unreserved	Reserved for future use	103	24:0

**Table 9 • Special Purpose Storage Regions for M2GL060, M2GL090 and M2GL150 Devices (continued)**

N-1	0	Unreserved	User Key Code #0 (256-bit User AES Key)	0	43:0
	0	Unreserved	User Key Code#1 (384-bit User PUF ECC Key) (76 bytes)	44	75:0
	0	Unreserved	Reserved for future use	120	7:0
	9-1	Reserved	Factory Activation Code	0	1151:0
	10	Reserved	Factory Activation Code (Total 1192 bytes across page 1 to page 10)	0	1191:1152
	10	Reserved	Factory Key Code (384 bit Factory ECC Key Code)	40	75:0
	10	Reserved	Reserved for future use	116	11:0
	15-11	Reserved	Second ECC Key Certificate	0	639:0
21-16	Reserved	Reserved for future use	0	767:0	
22	Unreserved	eNVM_1 Private User Digest of page 0 of N-1 and all pages of N-2	0	127:0	
23	Reserved	eNVM_1 Private Factory Digest of pages from 1 to 30 of N-1 except pages 22, 23, and 24	0	127:0	
24	Unreserved	eNVM_1 Public Digest	0	127:0	
30-25	Reserved	Device Certificate	0	767:0	
31	Unreserved	eNVM_0 Digest	0	127:0	

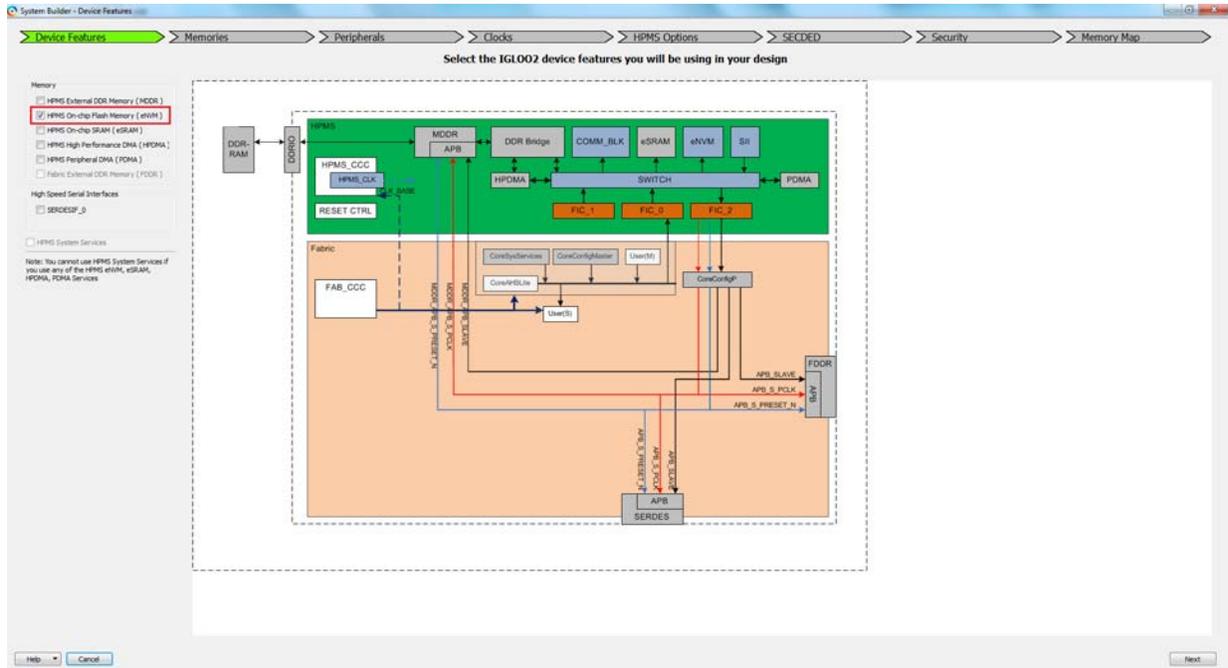
See [UG0443: SmartFusion2 SoC FPGA and IGLOO2 FPGA Security and Reliability User Guide](#) for more information on the certificates, key codes, and digests. System controller performs read/write operations on unreserved eNVM pages, and reads data from reserved eNVM pages.



The following steps describe how to generate a programming file with the eNVM client in an application using System Builder.

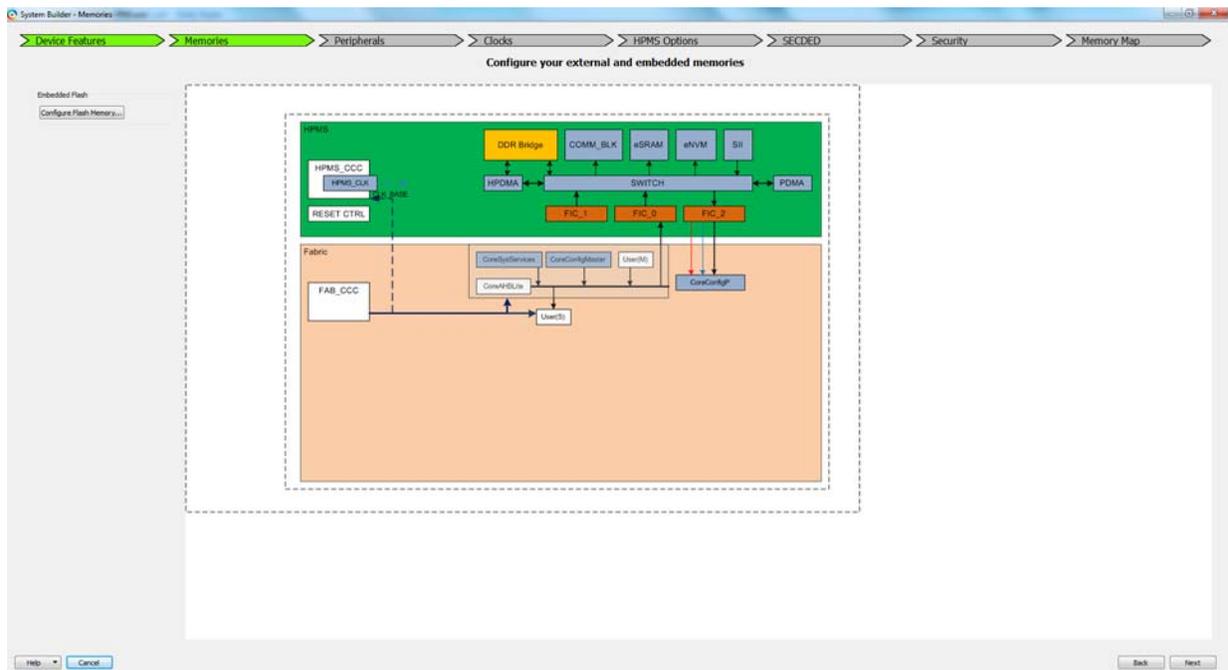
1. Check the **HPMS On-chip Flash Memory (eNVM)** check box under the **Device Features** tab and leave the other check boxes unchecked. The following figure shows the **System Builder - Device Features** tab.

**Figure 23 • System Builder - Device Features Tab**



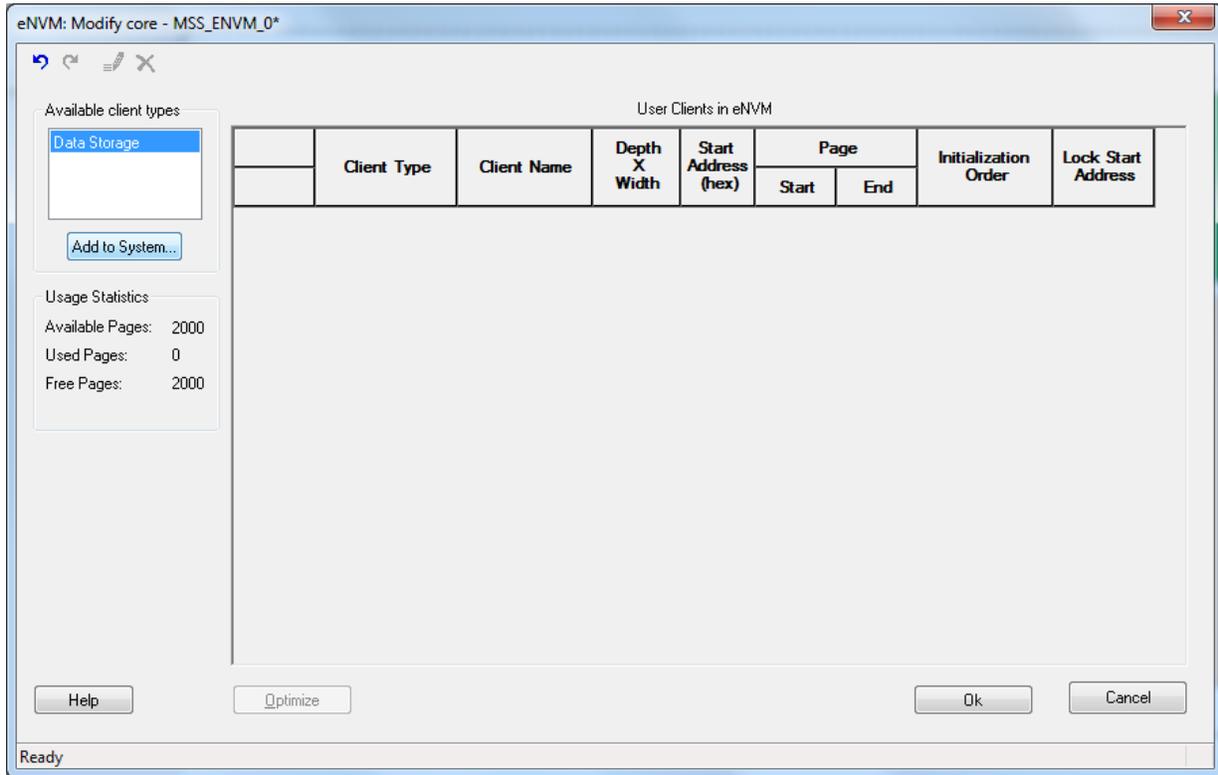
2. Click **Next** to navigate to the **Memories** tab. The following figure shows the **System Builder - Memories** tab.

**Figure 24 • System Builder - Memories Tab**



3. Click **Configure Flash Memory** to open the **eNVM: Modify core** dialog box. The following figure shows the **eNVM: Modify Core** dialog box.

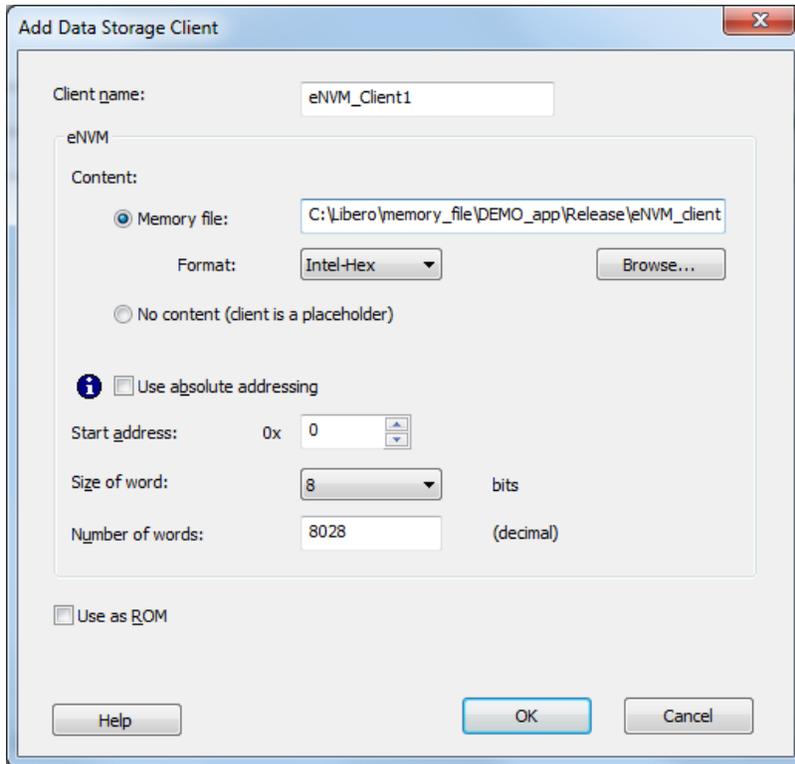
**Figure 25 • eNVM: Modify Core Dialog Box**



4. Select **Data Storage** under **Available Client Types** and click **Add to System**.
5. The following figure shows the **Add Data Storage Client** dialog. It supports the following types of file formats:
  - Intel-Hex
  - Motorola-S
  - Microsemi-Hex
  - Microsemi-Binary

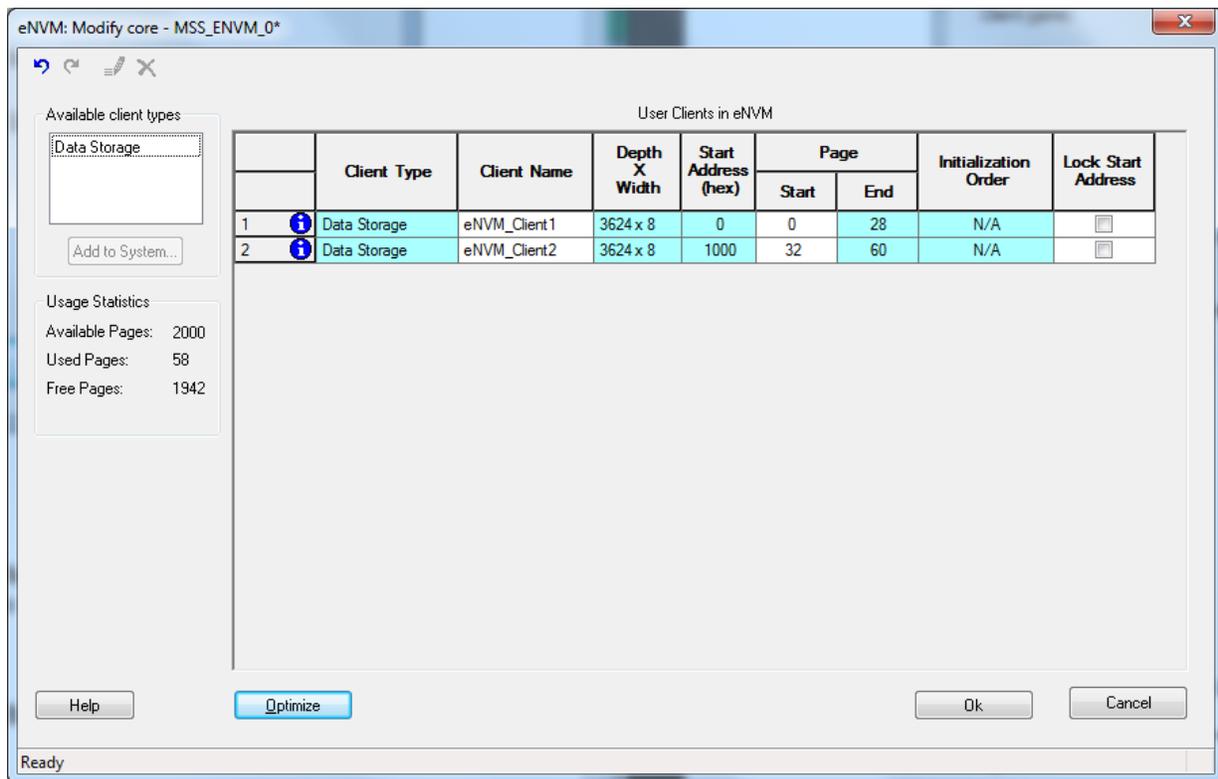
Enter the Client name, navigate to the memory file location using **Browse**, and select it. Give the rest of the parameters according to the requirements and click **OK** to add the eNVM client. For more information on **Use absolute addressing**, **Use as ROM** and other options, click **Help**.

**Figure 26 • Add Data Storage Client Dialog**



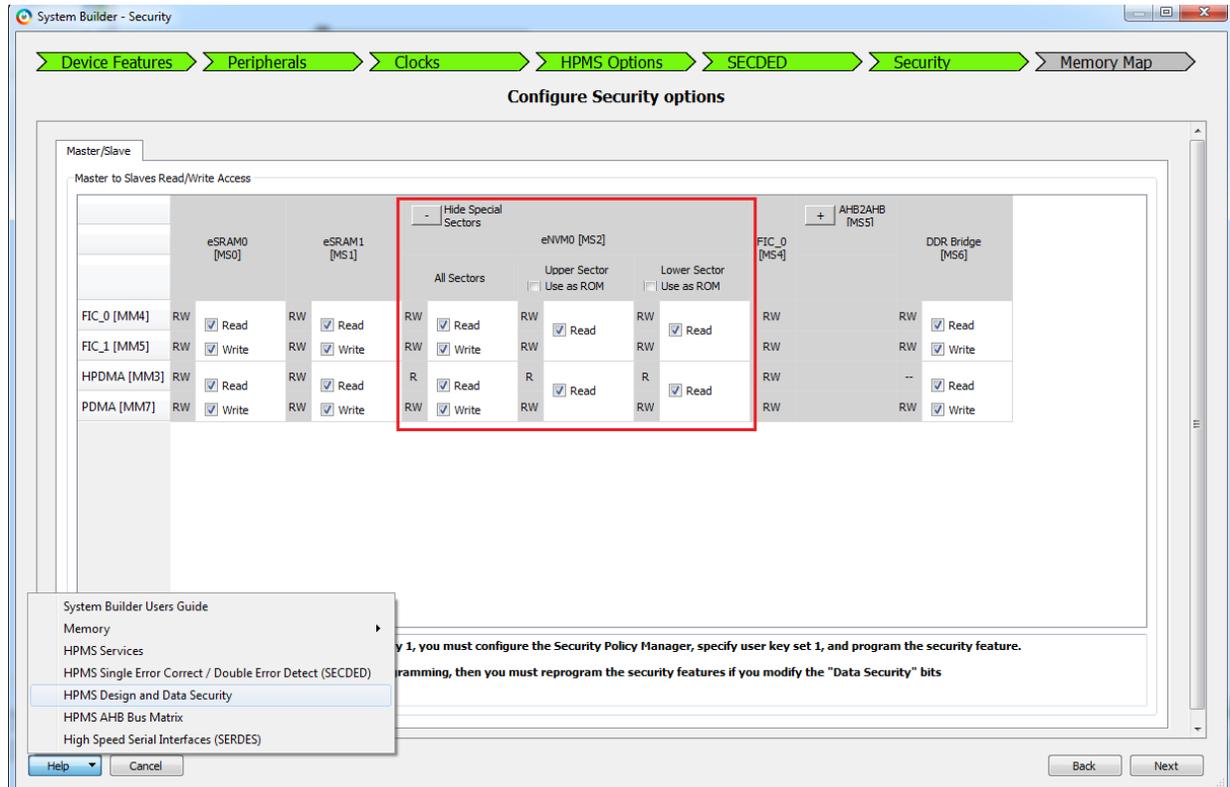
6. The eNVM client data is populated in the **eNVM: Modify core** dialog box. The following figure shows the **eNVM: Modify core** dialog box with two eNVM clients.

**Figure 27 • eNVM: Modify Core Dialog Box with Two eNVM Clients**



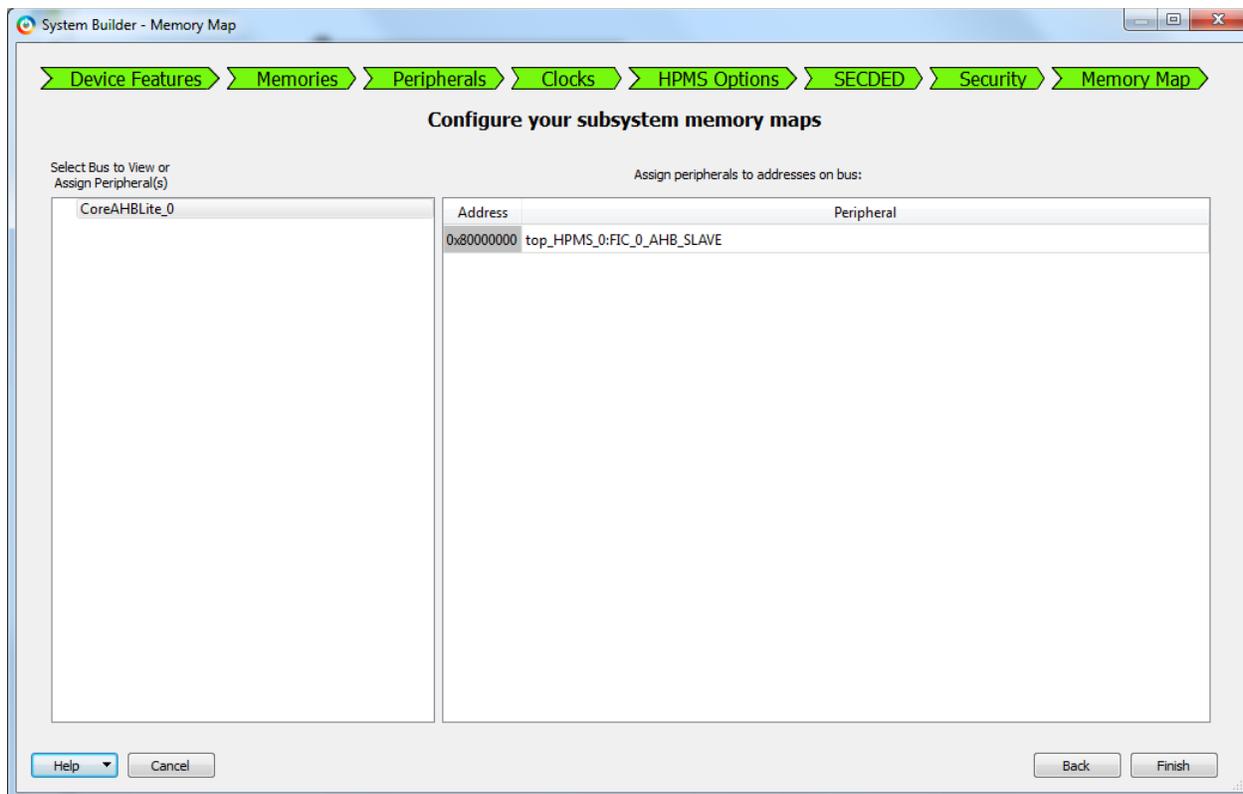
7. After adding the eNVM clients, click **OK**.
8. Navigate to the **Security** tab to select the read and write access permissions of eNVM including protected regions for different masters as shown in the following figure. For more information about eNVM Access Configuration, click **Help** and see HPMS Design and Data Security document. The read and write permission options for different masters are available for data and design security enabled devices like M2GL010TS only.

**Figure 28 • System Builder - Security Tab**



- Navigate to the **Memory Map** tab giving the required data in the rest of the **System Builder** tabs. The following figure shows the **System Builder - Memory Map** tab. Click **Finish** to proceed with creating the **HPMS Subsystem**, page 28.

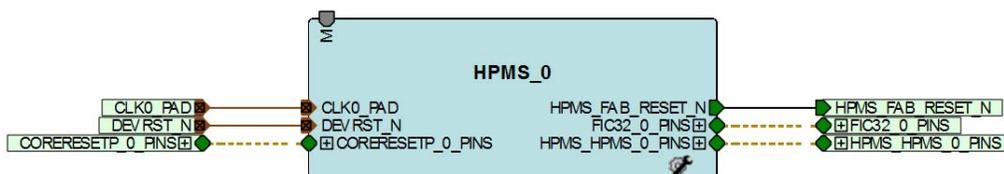
**Figure 29 • System Builder - Memory Map Tab**



## 2.4.2 HPMS Subsystem

The following figure shows an example HPMS subsystem that can be used to access the eNVM client data using the FPGA fabric master.

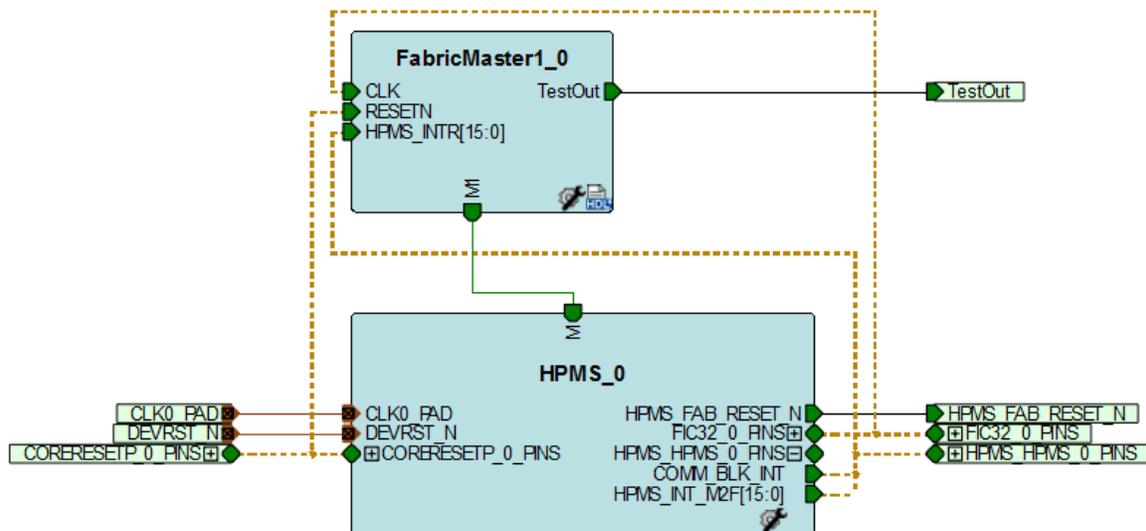
**Figure 30 • HPMS Subsystem**



### 2.4.3 HPMS Subsystem Connected to the FPGA Fabric Master

The following figure shows the FPGA fabric master connected to the AHB master port. The eNVM client data can be accessed using the FPGA fabric master that is connected to the AHB master port.

Figure 31 • HPMS Interconnection with FPGA Fabric Master



- The HPMS eNVM supports full-behavioral simulation models. See *Embedded Nonvolatile Memory (eNVM) Simulation* for information on how to simulate the eNVM operations.
- See *AC429: SmartFusion2 and IGLOO2 - Accessing eNVM and eSRAM from FPGA Fabric Application Note* for information on how to access the eNVM using FPGA fabric logic.

### 2.4.4 Reading the eNVM Block

Any master, for example, HPDMA, PDMA, or FPGA fabric connected to the AHB bus matrix can access the eNVM blocks using the address range provided in [Table 1](#), page 5 for read operations.

### 2.4.5 Writing to the eNVM Block

The FPGA fabric master can implement the user logic using the commands sequence explained in [eNVM Commands](#), page 9.

## 2.5 SYSREG Control Registers

The System Control Registers control eNVM behavior. These registers are located in the SYSREG section and are listed here for clarity. See [System Register Block](#), page 196 for more information on each register and bit.

Table 10 • SYSREG Control Registers

Register Name	Register Type	Flash Write Protect	Reset Source	Description
ENVM_CR (0x4003800C)	RW-P	Register	sysreset_n	eNVM Configuration Register. For more information, see <a href="#">Table 11</a> , page 30.
ENVM_REMAP_FAB_CR (0x40038014)	RW-P	Register	sysreset_n	eNVM remap Configuration Register for a soft processor in the FPGA. For more information, see <a href="#">Table 13</a> , page 32.

**Table 10 • SYSREG Control Registers (continued)**

Register Name	Register Type	Flash Write Protect	Reset Source	Description
ENVM_PROTECT_USER (0x40038144)	RO-U	N/A	sysreset_n	Configuration for accessibility of protected regions of eNVM_0 and eNVM_1 by different masters on the AHB bus matrix. This register gets updated by flash bit configuration set during device programming. This configuration can be done through the <b>System Builder</b> using settings on the <b>Security</b> tab. For more information, see <a href="#">Table 14</a> , page 33.
ENVM_STATUS (0x40038148)	RO-U	N/A	sysreset_n	Code shadow Status Register. For more information, see <a href="#">Table 15</a> , page 34.
ENVM_SR (0x40038158)	RO	N/A	sysreset_n	Indicates busy status for eNVM_0, eNVM_1. For more information, see <a href="#">Table 16</a> , page 34.

**Table 11 • ENVM\_CR**

Bit Number	Name	Reset Value	Description
[31:17]	Reserved	0	
16	ENVM_SENSE_ON	0	Turns on or off the sense amps for both NVM0 and NVM1. The sense amp switching feature is useful to decrease the eNVM access time. 0: Normal Operation -The sense amp switches off after every read cycle if an idle cycle follows. This saves power but slightly increases access time on the next read cycle. 1:The sense amp is turned ON. This increases power but decreases access times.
15	ENVM_PERSIST	0	Reset control for NVM0 and NVM1. 0: NVM0, NVM1 will get reset on SYSRESET_N and PORESET_N. 1: NVM0, NVM1 will get reset on PORESET_N.
14	NV_DPD1	0	Deep power-down control for the NVM1. 0: Normal operation 1: NVM deep power-down
13	NV_DPD0	0	Deep power-down control for the NVM0. 0: Normal operation 1: NVM deep power-down

**Table 11 • ENVM\_CR (continued)**

Bit Number	Name	Reset Value	Description
[12:5]	NV_FREQRNG	0x7	<p>Setting of NV_FREQRNG[8:5] or NV_FREQRNG[12:9] determines the behavior of eNVM BUSY_B with respect to the AHB Bus interface clock. NV_FREQRNG[8:5] is used for NVM0 and NV_FREQRNG[12:9] is used for NVM1. This control can be used to accommodate various frequencies of the external interface clock, HPMS_CLK, or it can be used to advance or delay the data capture due to variation of read access time of the NVM core. It sets the number of wait states to match with the fabric master operating frequency for read operations. The small counter in the NVM Controller uses this value to advance or delay the data capture before sampling data.</p> <p>0000: NOT SUPPORTED                      0001: NOT SUPPORTED                      0010: Page Read = 3, All other modes (Page program and Page verify) = 2                      0011: Page Read = 4, All other modes (Page program and Page verify) = 2                      0100: Page Read = 5, All other modes (Page program and Page verify) = 2                      0101: Page Read = 6, All other modes (Page program and Page verify) = 3                      0110: Page Read = 7, All other modes (Page program and Page verify) = 3                      0111: Page Read = 8, All other modes (Page program and Page verify) = 4                      1000: Page Read = 9, All other modes (Page program and Page verify) = 4                      1001: Page Read = 10, All other modes (Page program and Page verify) = 4                      1010: Page Read = 11, All other modes (Page program and Page verify) = 5                      1011: Page Read = 12, All other modes (Page program and Page verify) = 5                      1100: Page Read = 13, All other modes (Page program and Page verify) = 6                      1101: Page Read = 14, All other modes (Page program and Page verify) = 6                      1110: Page Read = 15, All other modes (Page program and Page verify) = 6                      1111: Page Read = 16, All other modes (Page program and Page verify) = 7</p>
[4:0]	SW_ENVMREMAPSIZE	0x11	<p>Size of the segment in eNVM, which is to be remapped to location 0x00000000. This logically splits eNVM into a number of segments, each of which may be used to store a different firmware image, for example. The region sizes are shown in <a href="#">Table 12, page 32</a>.</p>

**Table 12 • SW\_ENVMREMAPSIZE**

Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Remap Size
0	0	0	0	0	Reserved
0	0	0	0	1	Reserved
0	0	0	1	0	Reserved
0	0	0	1	1	Reserved
0	0	1	0	0	Reserved
0	0	1	0	1	Reserved
0	0	1	1	0	Reserved
0	0	1	1	1	Reserved
0	1	0	0	0	Reserved
0	1	0	0	1	Reserved
0	1	0	1	0	Reserved
0	1	0	1	1	Reserved
0	1	1	0	0	Reserved
0	1	1	0	1	16 Kbytes
0	1	1	1	0	32 Kbytes
0	1	1	1	1	64 Kbytes
1	0	0	0	0	128 Kbytes
1	0	0	0	1	256 Kbytes
1	0	0	1	0	512 Kbytes, reset value

**Table 13 • ENVM\_REMAP\_FAB\_CR**

Bit Number	Name	Reset Value	Description
[31:19]	Reserved	0	
[18:1]	SW_ENVMFABREMAPBASE	0	Offset within eNVM address space of the base address of the segment in eNVM, which is to be remapped to location 0x00000000 for use by a soft processor in the FPGA fabric. The base address of the remapped segment of eNVM is determined by the value of this register. Bit 0 of this register is defined as SW_ENVMFABREMAPENABLE. Bit 0 must be set to remap the NVM.
0	SW_ENVMFABREMAPENABLE	0	0: eNVM fabric remap not enabled for access by fabric master/soft processor. The portion of eNVM visible in the eNVM window at location 0x00000000 of a soft processor’s memory space corresponds to the memory locations at the bottom of eNVM. 1: eNVM fabric remap enabled. The portion of eNVM visible at location 0x00000000 of a soft processor’s memory space of is a remapped segment of eNVM.

**Table 14 • ENVN\_PROTECT\_USER**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0	
15	NVM1_UPPER_WRI TE_ALLOWED	0x1	When set indicates that the masters who have read access can have write access to the upper protection region of eNVM1. This is updated by the user flash row bit.
14	NVM1_UPPER_OTH ERS_ACCESS	0x1	When set indicates that the other masters can access the upper protection region of eNVM1. This is set by the user flash row bit.
13	NVM1_UPPER_FAB RIC_ACCESS	0x1	When set indicates that the fabric can access the upper protection region of eNVM1. This is set by the user flash row bit.
12	Reserved		
11	NVM1_LOWER_WRI TE_ALLOWED	0x1	When set indicates that the masters who have read access can have write access to the lower protection region of eNVM1. This is set by the user flash row bit.
10	NVM1_LOWER_OTH ERS_ACCESS	0x1	When set indicates that the other masters can access the lower protection region of eNVM1. This is set by the user flash row bit.
9	NVM1_LOWER_FAB RIC_ACCESS	0x1	When set indicates that the fabric can access the lower protection region of eNVM1. This is set by user flash row bit.
8	Reserved		
7	NVM0_UPPER_WRI TE_ALLOWED	0x1	When set indicates that the masters who have read access can have write access to the upper protection region of eNVM0. This will be set by the user flash row bit.
6	NVM0_UPPER_OTH ERS_ACCESS	0x1	When set indicates that the other masters can access the upper protection region of eNVM0.
5	NVM0_UPPER_FAB RIC_ACCESS	0x1	When set indicates that the fabric can access the upper protection region of eNVM0. This will be set by the user flash row bit.
4	Reserved		
3	NVM0_LOWER_WRI TE_ALLOWED	0x1	When set indicates that the masters who have read access can have write access to the lower protection region of eNVM0. This will be set by the user flash row bit.
2	NVM0_LOWER_OTH ERS_ACCESS	0x1	When set indicates that the other masters can access the lower protection region of eNVM0. This will be set by the user flash row bit.
1	NVM0_LOWER_FAB RIC_ACCESS	0x1	When set indicates that the fabric can access the lower protection region of eNVM0. This will be set by the user flash row bit.
0	Reserved		

See Table 7, page 19 for information on different masters.

**Table 15 • ENVM\_STATUS**

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0	
0	CODE_SHADOW_EN	0	Read by the system controller during device start-up, to indicate whether the user has configured the device such that code shadowing is to be performed by system controller firmware.

**Table 16 • ENVM\_SR**

Bit Number	Name	Reset Value	Description
[31:2]	Reserved	0	
[1:0]	ENVM_BUSY	0	Active high signals indicate a busy state per eNVM for CLK-driven operations and for internal operations triggered by the write/program/erase/transfer command. ENVM_BUSY[1] = Busy indication from ENVM1 ENVM_BUSY[0] = Busy indication from ENVM0

## 2.6 eNVM Control Registers

To perform any transaction with the NVM array, the Control Registers must be configured appropriately as per the following table.

To access or update the Control Register, the AHBL master must first get access to the register set. Without access rights, all writes to the Control Register will be ignored and the read will return zero from REQACC and the status register bit definitions (see Table 19, page 38).

This access rights system ensures that while a master is programming the NVM array, no other master can interfere or see what data is being programmed.

To obtain access rights, the master writes 0x1 to the REQACC register and then reads the register to check whether access is granted. If access is granted the Control Register is set.

The following table shows the base address of the eNVM Control Registers for eNVM\_0 and eNVM\_1.

**Table 17 • eNVM Control Registers Base Address**

eNVM Block	Control Registers Base Address
eNVM_0	0x60080000
eNVM_1	0x600C0000

**Table 18 • Control Registers Description**

OFFSET HADDR[8:0]	Register Name	Width	Type	Default	Access Rights	Description
0x000-0x07F	Assembly Buffer	1023:0 32 × 32bits	R		Exclusive access to the requested master	Reads from these addresses will return data read from assembly buffer within the NVM array

**Table 18 • Control Registers Description (continued)**

OFFSET HADDR[8:0]	Register Name	Width	Type	Default	Access Rights	Description
0x080-0x0FF	WDBUFF (Write Data Buffer)	1023:0 32 × 32bits	R/W	0	Any master on AHB bus matrix	Write data buffer. This register is cleared when exiting normal mode. This register is not cleared when the System Controller grabs ownership by writing 0x03 to REQACCESS (see <a href="#">Table 18</a> , page 34).
0x120	Status	31:0	R		Any master on AHB bus matrix	See <a href="#">Table 19</a> , page 38.
0x128	NV_PAGE_STATUS	1:0	R/W	0	Exclusive access to the requested master	See <a href="#">Table 21</a> , page 39.
0x12C	NV_FREQRNG[7:0]	7:0	R	SYSRE G	Exclusive access to the requested master	eNVM interface frequency range setting: Bits [3:0] set the number of wait cycles required for each NVM access cycles. This is read-only register. The ENVM_CR system register NV_FREQRNG field needs to be set with value as calculated below. NV_FREQRNG = roundup(40 ns / HPMS_CLK clock period in ns) The NV_FREQRNG[3:0] is for NVM0 wait states and NV_FREQRNG[7:4] is for NVM1 wait states. See <a href="#">Table 20</a> , page 39 NV_FREQRNG calculations at different HPMS_CLK frequencies for all IGLOO2 devices. Bits [7:4] are unused with the AHB-NVM block when the device has only eNVM_0. This controls the NV_FREQRNG[3:0] input on the NVMCTRL function that sets the required number of clock cycles required for NVM accesses relative to the operating frequency.
0x130	NV_DPD_B	1-bit	R	SYSRE G	Exclusive access to the requested master	NV_DPD_B[0] describes NVM deep power-down state. 0: NVM operational 1: NVM In deep power-down

**Table 18 • Control Registers Description (continued)**

OFFSET HADDR[8:0]	Register Name	Width	Type	Default	Access Rights	Description
0x134	NV_CE	2-bit	R/W	1	Exclusive access to the requested master	NV_CE[0] = 0: NVM disabled NV_CE[0] = 1: NVM enabled NV_CE [1] = 1; The internal read cache is disabled. All reads will directly read the eNVM array, AB or SFR space. When set NVM access latency will increase. By default this bit is set to '0'.
0x140	PAGE_LOCK_SET	1	R/W	0	Exclusive access to the requested master	PAGE_LOCK_SET[0] = 1: Page is locked. PAGE_LOCK_SET[0] = 0: Page is unlocked. If the page is locked, then before writing the page should be unlocked.
0x144	DWSIZE	3:0	R/W	0	Exclusive access to the requested master	Write size in number of double words, to be written to assembly buffer from Write Data buffer during NVM commands. See description for individual commands. 0000 = 1 dword 1111 = 16 dwords
0x148	CMD	31:0	R/W	0	Exclusive access to the requested master	Write to CMD and if command field in HWDATA decoded to be a command, then NVM command will be initiated. See description of <a href="#">Table 5</a> , page 9: CMD Register and individual commands.
0x154	INTEN[10:0]	10:0	R/W	0	Exclusive access to the requested master	Writing '1' to each bit will enable the corresponding interrupt. For more information, see <a href="#">Table 22</a> , page 39.
0x158	CLRHint[2:0]	2:0	W	0	Exclusive access to the requested master	Clear interrupts/flag/busy bit by writing 1 to the corresponding bit. For more information, see <a href="#">Table 23</a> , page 40.

**Table 18 • Control Registers Description (continued)**

OFFSET HADDR[8:0]	Register Name	Width	Type	Default	Access Rights	Description
0x1FC	REQACCESS	2:0	R/W	000	Any master on AHB bus matrix. This register can only be accessed using word, half or byte accesses to address 0x01FC. Accesses to addresses 0x1FD, 0x1FE, and 0x1F should not be used.	Request register access. When written with 0x01, it will request exclusive access. Read indicates whether access has granted or not or which entity currently has been granted access. Read Value [2:0] 0XX: No entity has access. The XX value indicates who had last access. 100: System controller 101: Reserved 110: Fabric 111: Other master (such as PDMA or HDMA) To release access rights, write 0x00. The System Controller may gain immediate access by writing 0x03 to this register. When access is relinquished, the WDBUFF buffer, and RDBUFF buffers are cleared.

**Note:** Addresses that are not mentioned in the register range are either reserved or exclusively for System Controller usage.

## 2.6.1 Status Register Bit Definitions

**Table 19 • Status Register Bit Definitions**

Bit	Description
[31:29]	Value of locked state of the AHB interface. These bits contain the same information as REQACCESS[2:0] (see <a href="#">Table 18</a> , page 34).
[28:20]	Reserved
19	Command when Busy. Indicates that a command was loaded while the controller was busy and has been ignored. Once set all command operations are disabled. Cleared by writing 1 to bit-2 in CLRHINT[2:0] (see <a href="#">Table 23</a> , page 40).
18	Access denied. Indicates that read or writes operations were denied due to protection systems, or that an illegal command was loaded. Once set all command operations are disabled. Cleared by writing 1 to bit 1 in the CLRHINT[2:0].
17	NVM deep power-down state, indicates NVM has entered DPD mode 0: NVM operational 1: NVM In deep power down There is delay of ~5μs for the NVM to enter power down and assert this bit from requesting power down.
[16:15]	RDBUFF3 (Read data buffer 3 = Read data buffer[255:192]) ECC status (2-bit error, 1 bit corrected) 00: no error 01: 1 bit corrected 10: 2 bit detected 11: 3 or more bits detected
[14:13]	RDBUFF2 (Read data buffer 2 = Read data buffer[191:128]) ECC status (2-bit error, 1 bit corrected) 00: no error 01: 1 bit corrected 10: 2 bit detected 11: 3 or more bits detected
[12:11]	RDBUFF1 (Read data buffer 1 = Read data buffer[127:64]) ECC status (2-bit error, 1 bit corrected) 00: no error 01: 1 bit corrected 10: 2 bit detected 11: 3 or more bits detected
[10:9]	RDBUFF0 (Read data buffer 0 = Read data buffer[63:0]) ECC status (2-bit error, 1 bit corrected) 00: no error 01: 1 bit corrected 10: 2 bit detected 11: 3 or more bits detected
8	Asserted for ECC2 (2 bit error). Valid after read and read assembly buffer.
7	Asserted for ECC1 (1 bit correction). Valid after read and read assembly buffer.
6	Asserted for refresh required. Valid after program, write only and page erase.
5	Asserted when write count is over threshold. Valid after program, verify and read page status. The threshold value per eNVM page is 1000 or 10000 depending on the data retention period. See <a href="#">DS0128: IGLOO2 and SmartFusion2 Datasheet</a> for more information on programming cycles and retention time.

**Table 19 • Status Register Bit Definitions (continued)**

Bit	Description
4	Asserted for program or erase failure due to page lock. Valid after program, page erase. Asserted for write failure when executing write only on the page with currently erased page bar (CEPB) = 1.
3	Asserted for write verify failure. Valid after program and write only.
2	Asserted for erase verify failure. Valid after program, page erase.
1	Asserted for verify failure. Valid only after verify operation.
0	NVM Ready/busy 0: Busy 1: Ready

**Table 20 • NV\_FREQRNG Calculations at Different HPMS\_CLK Frequencies for IGLOO2 Devices**

HPMS_CLK in MHz	Standard HPMS Frequencies		
	166	142	133
NV_FREQRNG[3:0]	0x7	0x6	0x 5
NV_FREQRNG[7:4]	0x7	0x6	0x 5
NV_FREQRNG[7:0]	0x77	0x66	0x55

**Note:** If HPMS\_CLK is 166 MHz, clock period is 6.024ns. NV\_FREQRNG[3:0] = roundup (40ns/6.024ns)=7.

**Table 21 • NV\_PAGE\_STATUS**

Bit	Description
1	R/W page status select
0	Reserved

**Table 22 • INTEN[10:0]**

Bit	Description
10	Command loaded when busy
9	NVM command denied by protection
8	NVM internal operation (erase/program/write only) is complete
7	ECC2 (2-bit error)
6	ECC1 (1-bit correction)
5	Refresh required
4	Write count is over threshold
3	Program or erase failure due to page lock. Write failure when executing write only on the page with currently erased page bar (CEPB) =1.
2	Write verify failure
1	Erase verify failure
0	Verify failure

**Table 23 • CLRHINT[2:0]**

<b>Bit</b>	<b>Description</b>
2	Clear the internal command when busy bit
1	Clear the internal access denied flag
0	Clear HINTERRUPT output

### 3 Embedded SRAM (eSRAM) Controllers

IGLOO2 FPGAs have two embedded SRAM (eSRAM) blocks of 32 Kbytes each for data read and write operations. These eSRAM blocks are interfaced through eSRAM controllers to the AHB bus matrix.

#### 3.1 Features

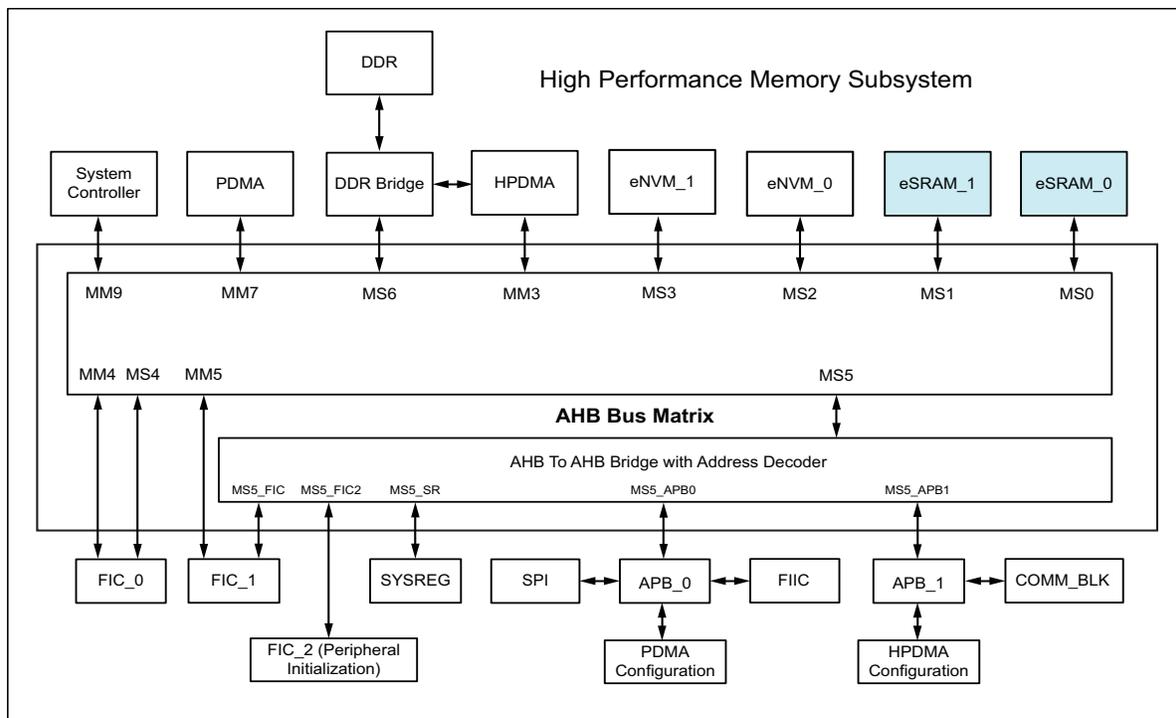
- Each eSRAM controller supports single bit error correction and dual bit error detection (SECEDED).
- Two modes of operation: SECEDED-ON and SECEDED-OFF.
- The total amount of available eSRAM in each device is 64 Kbytes in SECEDED-ON mode and 80 Kbytes in SECEDED-OFF mode.
- Each individual eSRAM block is 32 Kbytes in SECEDED-ON mode and 40 Kbytes in SECEDED-OFF mode, organized in a 2 × 4096 × 40 fashion.
- Having two blocks (eSRAM\_0 and eSRAM\_1) maximizes hardware parallelism. For example, at the same instant that the fabric master is reading from eSRAM\_0, another fabric master can read from eSRAM\_1 independently.
- The eSRAM address space is byte, half-word (16 bit), and word (32 bit) addressable.
- A pipeline is provided to address the latency issues at higher speeds of operation.

As shown in the following figure, the total available size of the eSRAM is divided into two equal-sized blocks: eSRAM\_0 and eSRAM\_1. eSRAM\_0 and eSRAM\_1 are connected to slave 0 and slave 1 on the AHB bus matrix through eSRAM controller 0 and eSRAM controller 1.

The eSRAM controller is designed to interface to an 8192 × 40 RAM, which is organized in a 2 × 4096 × 40 fashion with five 8-bit byte lanes in total. The fabric master and other masters find the eSRAMs available as one contiguous area of memory.

The following figure depicts the connectivity of eSRAM\_0 and eSRAM\_1 to the AHB bus matrix.

Figure 32 • eSRAM\_0 and eSRAM\_1 Connection to AHB Bus Matrix



## 3.2 Functional Description

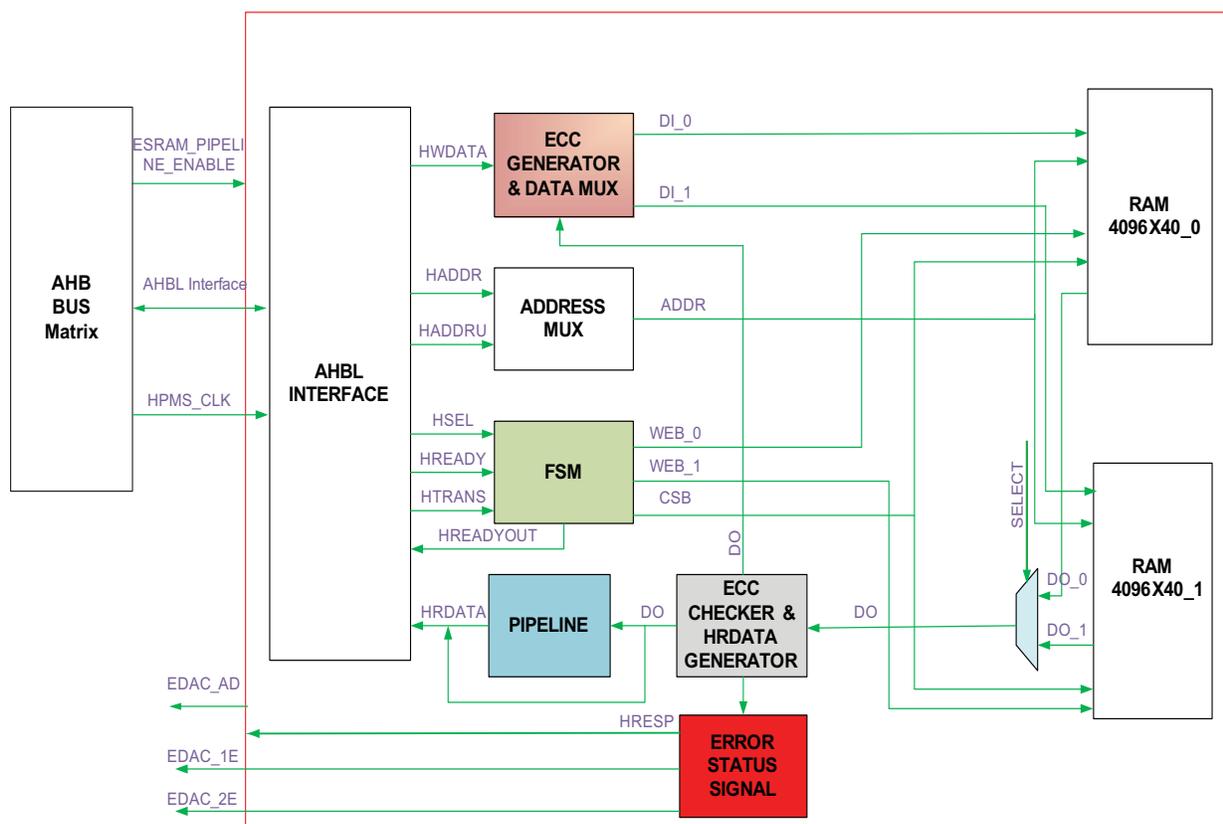
The following table shows the size of the eSRAM blocks and their address range.

**Table 24 • eSRAM Block Sizes and Address Ranges**

eSRAM Block	Physical RAM4096X40 Block	Size and Address Range with SECEDED ON	Size and Address Range with SECEDED OFF
eSRAM_0	RAM4096X40_0	16 KB from 0x20000000 to 0x20003FFF and ECC from 0x20010000 to 0x20010FFF	16 KB from 0x20000000 to 0x20003FFF and 4 KB from 0x20010000 to 0x20010FFF
	RAM4096X40_1	16 KB from 0x20004000 to 0x20007FFF and ECC from 0x20011000 to 0x20011FFF	16 KB from 0x20004000 to 0x20007FFF and 4 KB from 0x20011000 to 0x20011FFF
eSRAM_1	RAM4096X40_2	16 KB from 0x20008000 to 0x2000BFFF and ECC from 0x20012000 to 0x20012FFF	16 KB from 0x20008000 to 0x2000BFFF and 4 KB from 0x20012000 to 0x20012FFF
	RAM4096X40_3	16 KB from 0x2000C000 to 0x2000FFFF and ECC from 0x20013000 to 0x20013FFF	16 KB from 0x2000C000 to 0x2000FFFF and 4 KB from 0x20013000 to 0x20013FFF

The following figure shows the eSRAM controller blocks and their connectivity in IGLOO2 FPGAs. Both eSRAMs and eSRAM controllers are identical in all design aspects.

**Figure 33 • eSRAM Controller Block Diagram**



HPMS\_CLK is used within the HPMS to clock the AHB bus matrix. See [UG0449: SmartFusion2 and IGLOO2 Clocking Resources User Guide](#) for more information on HPMS\_CLK.

**AHBL Interface:** Each eSRAM controller is an AHB-Lite (AHBL) slave that provides access to the eSRAM block from the AHB bus matrix.

**ECC Generator and Data MUX:** In SECEDED-ON mode, the ECC Generator generates the check bits for 32-bit data. For a 32-bit write from the AHBL interface, the input data AHB write data bus (HWDATA) is used to generate check bits. These check bits are appended to HWDATA and written to the memory. For 8-bit and 16-bit writes from the AHBL interface, a read-modify-write operation is used. This reads data from the 32-bit word, corrects if necessary, and then writes the new data value and ECC check bits.

In SECEDED-OFF mode, if the memory access is within 32 Kbyte memory, HWDATA is sent directly to the memory input. If the access is for additional 8 Kbyte memory, then the address for a particular byte of HWDATA will be selected based on the shift address.

**Address MUX:** This utilizes the AHB address bus (HADDR) and HADDRU (an additional HADDR bit) for selecting upper 8 K bank of the RAM. Based on the FSM internal signals, output ADDR is generated and passed to the memory. The shifted address is also generated and used for multiplexing data.

**FSM:** This generates output signal HREADYOUT and internal signals that are used for multiplexing an address.

**Pipeline:** A pipeline stage in the read path of eSRAM and the master that accesses this path is configurable using the ESRAM\_PIPELINE\_ENABLE signal. When ESRAM\_PIPELINE\_ENABLE is High, there is an extra one clock cycle delay for the read operation to maximize operational frequency. At higher frequencies (> 100 MHz) of fabric or other masters accessing eSRAM, the eSRAM operations need an extra clock cycle for the correct data transactions.

**ECC Checker and AHB Read Data Bus (HRDATA) Generator:** In SECEDED-ON mode, the ECC Checker takes data (DO) from the memory as the input during the read or read-modify-write cycle and checks for errors. One-bit errors detected are corrected.

If errors of more than one bit are detected, they are not corrected. In SECEDED-OFF mode, the read out data is directly given as output from this block. Error Status Signals are set if any errors are detected.

**Error Status Signals:** Error bits are inputs from the ECC Checker. If one error bit is High, it causes the EDAC\_1E signal to be High. In this case, there is no HRESP as the error is corrected. If there are two-bit errors, it cause the EDAC\_2E signal to be High. In this case, HRESP is set High because the error is not corrected. The EDAC\_1E and EDAC\_2E signals are used to increment the ECC error counters within the SYSREG block (and the failing address is also passed to the SYSREG block). When the HRESET to ESRAMTOAHB is applied, it resets the EDAC address register which is maintained in ESRAMTOAHB and it does not clear the contents of SRAM. EDAC error counters are maintained in System Register which can be cleared either through same HRESET or by setting the CLR\_EDAC\_COUNTERS (see Table 40, page 58).

### 3.2.1 Memory Organization

The 40 Kbytes of eSRAM memory is divided into two banks: 32 Kbytes and 8 Kbytes, to store 32 bits of data and 7 check bits in SECEDED-ON mode. Physically, however, the memory is organized as 4096 × 40, which is 4096 × 5 bytes. When ECC is enabled, the fifth byte stores ECC values for the 32 bits of data. When ECC is disabled, the fifth byte location is used to create an additional 2 Kbytes of user memory. Four locations are used for each 32-bit word.

The following table shows the organization of 4096 × 40 bits in SECEDED-ON mode. The total size of the SRAM in the table is 40 Kbytes. The locations show the memory used for the 32 Kbyte block. ECC represents the 7-bit ECC.

**Table 25 • SRAM Organization in SECEDED-ON Mode**

Location	RAM 4096X40_1 4096 x 40 Bits					RAM 4096X40_0 4096 x 40 Bits				
	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
0	ECC	4003	4002	4001	4000	ECC	0003	0002	0001	0000
1	ECC	4007	4006	4005	4004	ECC	0007	0006	0005	0004

**Table 25 • SRAM Organization in SECDED-ON Mode (continued)**

Location	RAM 4096X40_1 4096 x 40 Bits					RAM 4096X40_0 4096 x 40 Bits				
	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
2046	ECC	5FFB	5FFA	5FF9	5FF8	ECC	1FFB	1FFA	1FF9	1FF8
2047	ECC	5FFF	5FFE	5FFD	5FFC	ECC	1FFF	1FFE	1FFD	1FFC
2048	ECC	6003	6002	6001	6000	ECC	2003	2002	2001	2000
2049	ECC	6007	6006	6005	6004	ECC	2007	2006	2005	2004
4094	ECC	7FFB	7FFA	7FF9	7FF8	ECC	3FFB	3FFA	3FF9	3FF8
4095	ECC	7FFF	7FFE	7FFD	7FFC	ECC	3FFF	3FFE	3FFD	3FFC

The following table shows the organization of 4096 × 40 bits in SECDED-OFF mode. The total size of the SRAM in the table is 40 Kbytes. The red locations show the memory used for the 32 Kbyte block. The green locations show memory used for the upper 8 Kbyte block.

**Table 26 • SRAM Organization in SECDED-OFF Mode**

Location	RAM 4096X40_1 4096 x 40 Bits					RAM 4096X40_0 4096 x 40 Bits				
	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
0	0001	4003	4002	4001	4000	0000	0003	0002	0001	0000
1	0005	4007	4006	4005	4004	0004	0007	0006	0005	0004
2046	1FF9	5FFB	5FFA	5FF9	5FF8	1FF8	1FFB	1FFA	1FF9	1FF8
2047	1FFD	5FFF	5FFE	5FFD	5FFC	1FFC	1FFF	1FFE	1FFD	1FFC
2048	0003	6003	6002	6001	6000	0002	2003	2002	2001	2000
2049	0007	6007	6006	6005	6004	0006	2007	2006	2005	2004
4094	1FFB	7FFB	7FFA	7FF9	7FF8	1FFA	3FFB	3FFA	3FF9	3FF8
4095	1FFF	7FFF	7FFE	7FFD	7FFC	1FFE	3FFF	3FFE	3FFD	3FFC

## 3.2.2 Modes of Operation

There are two modes of operation for the eSRAM controller: SECDED-ON and SECDED-OFF.

### 3.2.2.1 SECDED-ON

SECDED mode can be turned ON by configuring the EDAC\_CR register (Table 42, page 60). The total available memory for each eSRAM in this mode is 32 Kbytes. The eSRAM controller generates 7 check bits for every 32 bits of data, so for every 32 bits of data there will be 7 bits of encoded data. The 7 bits of ECC allow 1-bit correction and 2-bit detection on the user data and ECC field. The 32 data bits and 7 bits of ECC are written to the memory with zero wait states. Byte and half-word write operations are done using a read-modify-write operation. The read-modify-write operation requires an additional wait state for byte and half-word write operations.

In the case of a 1-bit error, the previous 32 bits of data and ECC value are read and correction takes place automatically. The complete 32 bits plus ECC is rewritten. For byte and half-word write operations, there is one wait state required as the ECC value is read and corrected for the byte/half-word.

When a 2-bit error is detected during a read cycle for 32-bit data, HRESP is asserted High for two clock cycles and at the same time HREADYOUT goes Low for one clock cycle to indicate an error.

When a 2-bit error is detected during the read part of a read-modify-write byte or half-word operation, HRESP is asserted High.

### 3.2.2.2 SECEDED-OFF

SECEDED mode can be turned OFF by configuring the EDAC\_CR register (see Table 42, page 60). The total available memory for each eSRAM is 40 Kbytes. 1-bit correction and 2-bit detection on the user data is not applicable in this mode.

### 3.2.3 Pipeline Modes and Wait States for Read and Write Operations

When any master on the AHB bus matrix operates at a high frequency greater than 100 MHz and is accessing eSRAM, an extra clock cycle is needed for transactions. An optional pipeline can be enabled on the Read data bus; this adds a clock cycle to all read operations. The pipeline is enabled by default in both SECEDED-ON and SECEDED-OFF modes. When the master on the AHB bus matrix operates at low frequency, less than 100 MHz, the pipeline can be turned off. See Table 31, page 55 for information on pipeline enable/disable.

The actual frequency at which this is possible is specified in the AC characteristics table of *DS0128: IGLOO2 FPGA and SmartFusion2 SoC FPGA Datasheet*. When the pipeline is disabled, the number of wait states is less, increasing throughput of read operations.

The following table describes the wait states in different operation modes. These values indicate the number of wait states inserted by eSRAM controllers and apply to the reads and writes from masters within the High Performance Memory Subsystem (HPMS). Accessing eSRAM blocks from the FPGA fabric is performed through the fabric interface controller (FIC) interfaces. The FIC interface supports Bypass mode and Pipeline mode.

In Pipeline mode, the FIC interface adds one extra clock cycle for read and write, so the overall latency for accessing the eSRAM increases in this case.

**Table 27 • Wait States in Different Operation Modes**

Pipeline	eSRAM	SECEDED Mode	Operation	Size	Number of Wait States	Number of Wait States (Reads following a Write)
Enabled	32 KB RAM	SECEDED-ON Mode	Write	32-Bit	0	1
				16-Bit	1	3
				8-Bit	1	3
			Read	32-Bit	1	2
				16-Bit	1	2
				8-Bit	1	2
	8 KB RAM	SECEDED-OFF Mode	Write	32-Bit	0	0
				16-Bit	0	0
				8-Bit	0	0
			Read	32-Bit	1	2
				16-Bit	1	2
				8-Bit	1	2

**Table 27 • Wait States in Different Operation Modes (continued)**

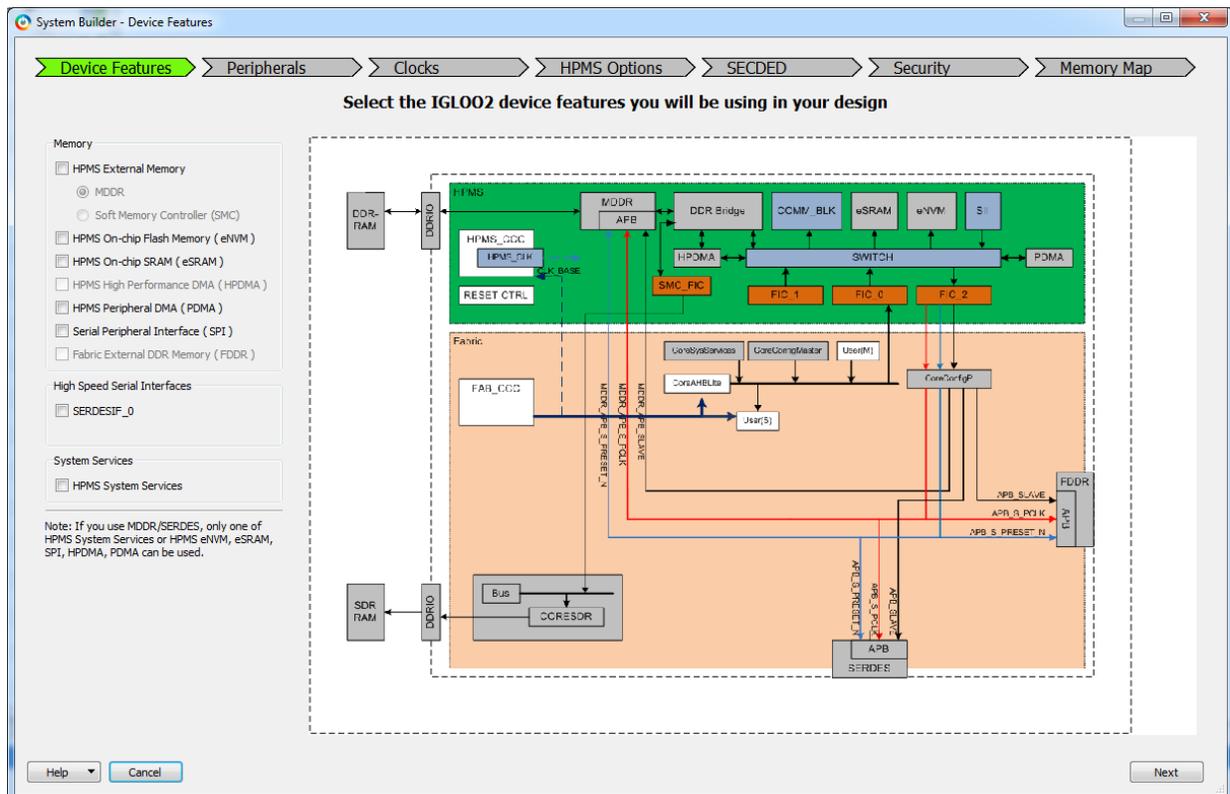
Pipeline	eSRAM	SECEDED Mode	Operation	Size	Number of Wait States	Number of Wait States (Reads following a Write)
Disabled	32 KB RAM	SECEDED-ON Mode	Write	32-Bit	0	0
				16-Bit	1	3
				8-Bit	1	3
			Read	32-Bit	0	1
				16-Bit	0	1
				8-Bit	0	1
	SECEDED-OFF Mode	Write	32-Bit	0	0	
			16-Bit	0	0	
			8-Bit	0	0	
		Read	32-Bit	0	1	
			16-Bit	0	1	
			8-Bit	0	1	
8 KB RAM	SECEDED-OFF Mode	Write	32-Bit	1	1	
			16-Bit	0	0	
			8-Bit	0	0	
		Read	32-Bit	1	2	
			16-Bit	0	1	
			8-Bit	0	1	

### 3.3 How to Use eSRAM

This section describes how to use the eSRAM 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 figure shows the initial **System Builder** window where the required device features can be selected. For details on how to launch the **System Builder** wizard and a detailed information on how to use it, See *IGLOO2 System Builder User Guide*.

**Figure 34 • System Builder Window**

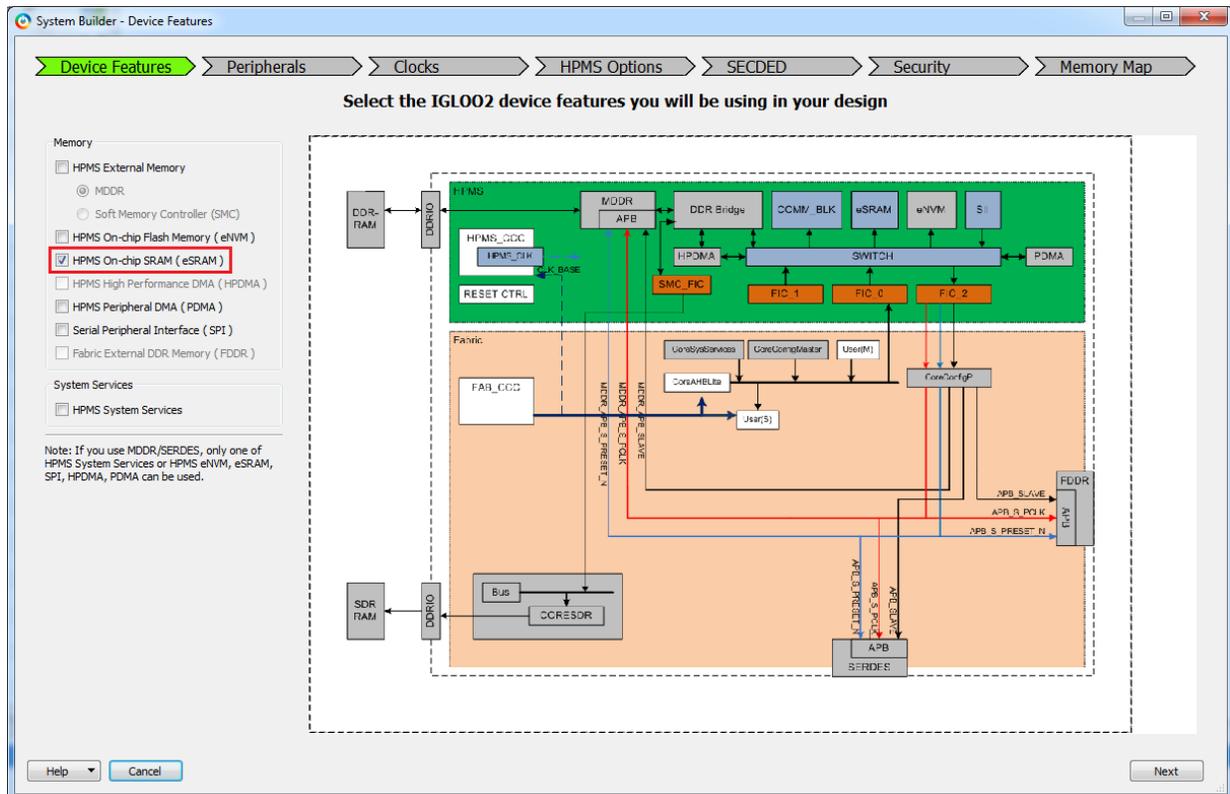


### 3.3.1 Accessing eSRAM Using FPGA Fabric Master

Any master (for example, FPGA fabric master, HPDMA, or PDMA) connected to the AHB bus matrix can access the eSRAM blocks using the address range provided in Table 24, page 42 for read and write operations. The following steps are used to enable the eSRAM blocks and eSRAM SECDED feature in the application using system builder.

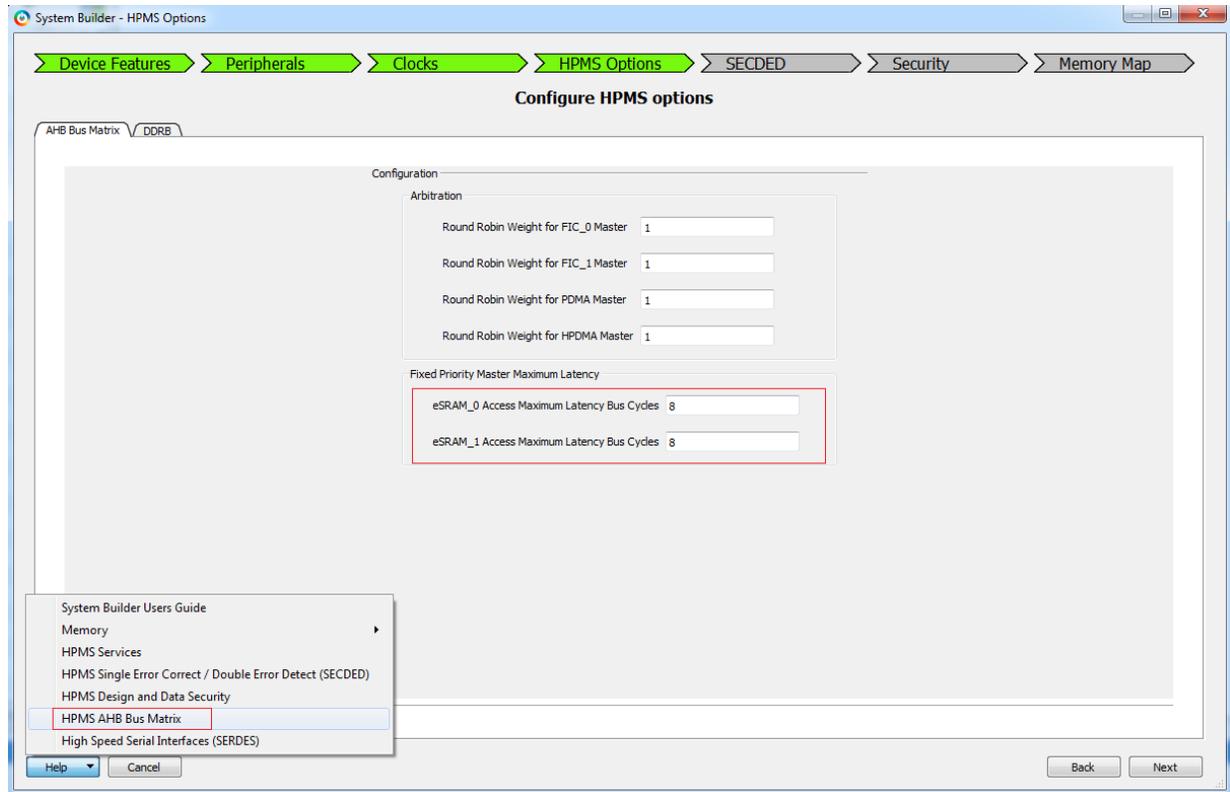
1. Check the **HPMS On-chip SRAM (eSRAM)** check box under the **Device Features** tab and leave the other check boxes unchecked. The following figure shows the **System Builder - Device Features** tab.

Figure 35 • System Builder - Device Features Tab



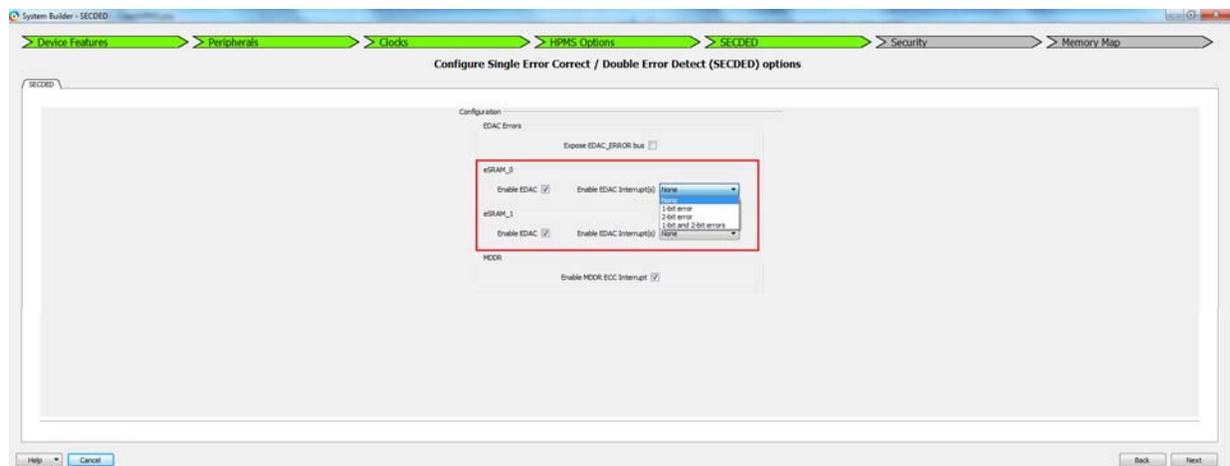
- Navigate to the **HPMS Options** tab in the System Builder to configure Programmable Slave Maximum Latency for eSRAM\_0 and eSRAM\_1. The following figure shows the **System Builder - HPMS Options** tab. For more information about the Programmable Slave Maximum Latency configuration, click **Help** and select HPMS AHB Bus Matrix document.

**Figure 36 • System Builder - HPMS Options Tab**



- Navigate to the **SECDED** tab in the **System Builder** to configure SECDED options for eSRAM\_0 and eSRAM\_1. The following figure shows the **System Builder - SECDED** tab.

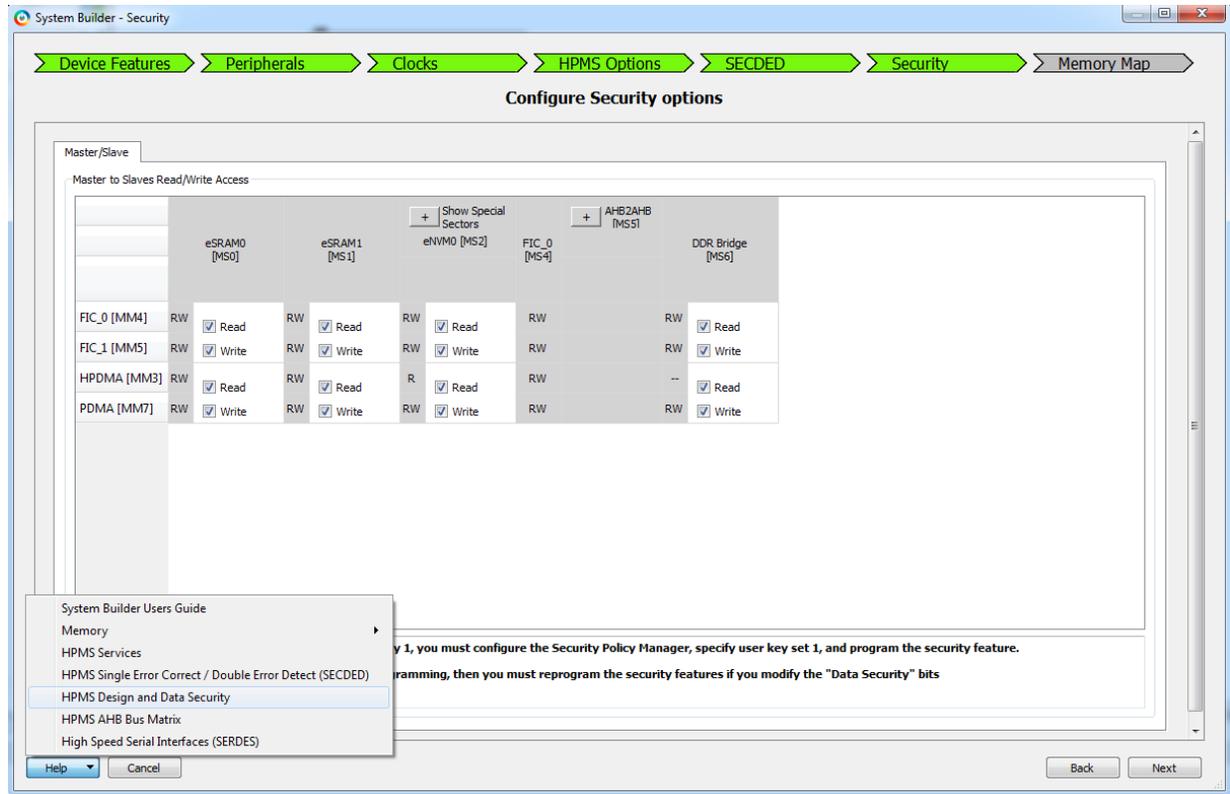
**Figure 37 • System Builder - SECDED Tab**



The SECDED feature can be enabled or disabled by selecting **Enable EDAC** for eSRAM\_0 and eSRAM\_1. The interrupts for 1-bit error or 2-bit error, or both 1-bit and 2-bit errors can be enabled. The FPGA fabric master has to implement user logic to clear these interrupts and to take the necessary actions in case of a 2-bit error.

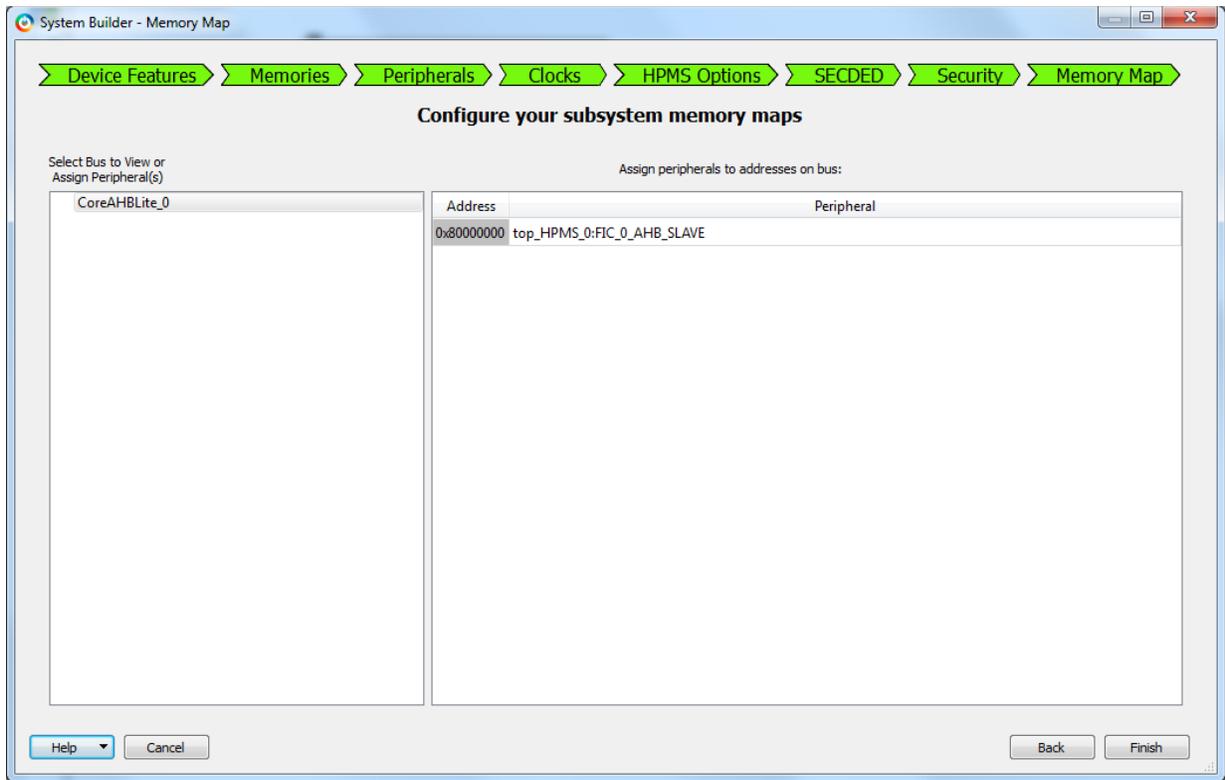
4. Navigate to the **Security** tab to select the read and write access permissions of eSRAM for different masters as shown in the following figure. For more information about eSRAM access configuration, click **Help** and see HPMS Design and Data Security document. The read and write permission options for different masters are available for data and design security enabled devices like M2GL010TS only.

**Figure 38 • System Builder - Security Tab**



5. Navigate to the **Memory Map** tab giving the required data in the rest of the **System Builder** tabs. The following figure shows the **System Builder - Memory Map** tab. Click **Finish** to proceed with creating the HPMS Subsystem (see [HPMS Subsystem](#), page 51).

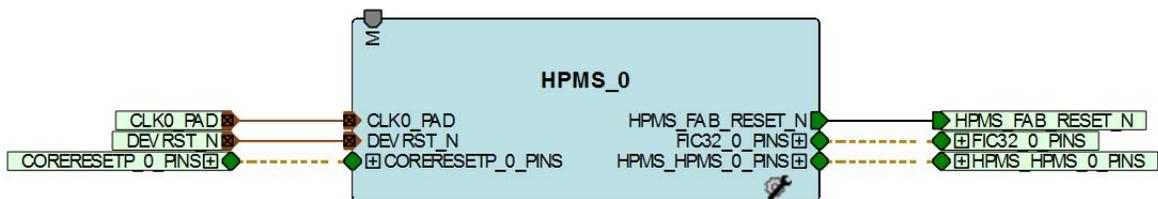
**Figure 39 • System Builder - Memory Map Tab**



### 3.3.2 HPMS Subsystem

The following figure shows the HPMS subsystem for accessing the eSRAM using FPGA fabric master.

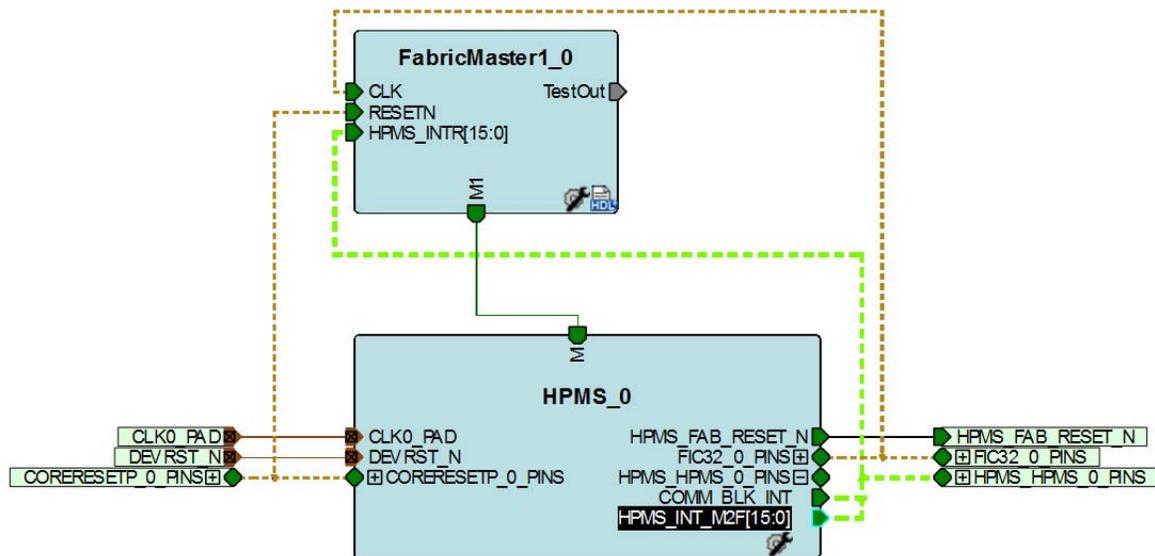
**Figure 40 • HPMS Subsystem**



### 3.3.3 HPMS Subsystem Connected to the FPGA Fabric Master

The following figure shows the FPGA fabric master connected to AHB master port. The eSRAM blocks can be accessed using FPGA fabric master connected to the AHB master port for read and write operations.

Figure 41 • HPMS Interconnection with FPGA Fabric Master



## 3.4 SYSREG Control Registers

The registers listed in the following table control the behavior of the eSRAM. These registers are detailed in SYSREG (see Table 126, page 202) and are listed here for clarity. See System Register Block, page 196 for a detailed description of each register and bit.

Table 28 • SYSREG Control Registers

Register Name	Register Type	Flash Write Protect	Reset Source	Description
ESRAM_MAX_LAT (0x40038004)	RW-P	Register	SYSRESET_N	Configuration of maximum latency for accessing eSRAM_0 and eSRAM_1 slaves. This register gets updated by flash bit configuration set during device programming. This configuration can be done through the <b>System Builder</b> also using settings on the <b>HPMS Options</b> tab. For more information, see Table 29, page 54.
ESRAM_PIPELINE_CR (0x40038080)	RW-P	Register	SYSRESET_N	Controls the pipeline present in the memory read path of eSRAM memory. For more information, see Table 31, page 55.
ESRAM0_EDAC_CNT (0x400380F0)	RO	N/A	SYSRESET_N	Represents 1-bit error count of eSRAM_0. For more information, see Table 32, page 55.

**Table 28 • SYSREG Control Registers (continued)**

Register Name	Register Type	Flash Write Protect	Reset Source	Description
ESRAM1_EDAC_CNT (0x400380F4)	RO	N/A	SYSRESET_N	Represents 1-bit error count of eSRAM_1. For more information, see <a href="#">Table 33</a> , page 55.
ESRAM0_EDAC_ADR (0x4003810C)	RO	N/A	SYSRESET_N	Address from eSRAM_0 on which 1-bit ECC error has occurred. For more information, see <a href="#">Table 34</a> , page 55.
ESRAM1_EDAC_ADR (0x40038110)	RO	N/A	SYSRESET_N	Address from eSRAM_1 on which 1-bit ECC error has occurred. For more information, see <a href="#">Table 35</a> , page 56.
MM4_5_DDR_FIC_SECUR ITY/MM4_5_FIC64_SECU RITY(0x40038128)	RO-U	N/A	SYSRESET_N	Read and Write security for Mirrored Master (MM) 4, 5, and DDR_FIC to eSRAM_0 and eSRAM_1. This register gets updated by flash bit configuration set during device programming. This configuration can be done through the <b>System Builder</b> using settings on the <b>Security</b> tab. For more information, see <a href="#">Table 36</a> , page 56.
MM3_7_SECURITY (0x4003812C)	RO-U	N/A	SYSRESET_N	Read and Write security for Mirrored Master (MM) 3 and 7 to eSRAM_0 and eSRAM_1. This register gets updated by flash bit configuration set during device programming. This configuration can be done through the <b>System Builder</b> using settings on the <b>Security</b> tab. For more information, see <a href="#">Table 37</a> , page 57.
MM9_SECURITY (0x40038130)	RO-U	N/A	SYSRESET_N	Read and Write security for Mirrored Master (MM) 9 to eSRAM_0 and eSRAM_1. This register gets updated by flash bit configuration set during device programming. This configuration can be done through the <b>System Builder</b> using settings on the <b>Security</b> tab. For more information, see <a href="#">Table 38</a> , page 57.
EDAC_SR (0x40038190)	SW1C	N/A	SYSRESET_N	Status of 1-bit ECC error detection and correction (EDAC), 2-bit ECC error detection for eSRAM_0 and eSRAM_1. Individual register bits are set ('1') when related input is asserted. Bits are individually cleared when corresponding register bit is written high. For more information, see <a href="#">Table 39</a> , page 58.
CLR_EDAC_COUNTERS (0x400381A4)	W1P	N/A	SYSRESET_N	This is used to clear the 16-bit counter value in eSRAM_0 and eSRAM_1 corresponding to the count value of EDAC 1-bit and 2-bit errors. For more information, see <a href="#">Table 40</a> , page 58.

**Table 28 • SYSREG Control Registers (continued)**

Register Name	Register Type	Flash Write Protect	Reset Source	Description
EDAC_IRQ_ENABLE_CR (0x40038078)	RW-P	Register	SYSRESET_N	Enable/disable of 1-bit error, 2-bit error status update for eSRAM_0 and eSRAM_1. This can be set by the <b>System Builder</b> also using settings on the <b>SECEDED</b> tab. For more information, see <a href="#">Table 41</a> , page 59.
EDAC_CR (0x40038038)	RW-P	Register	SYSRESET_N	EDAC enable/disable and soft reset for eSRAM_0 and eSRAM_1. This can be set by the <b>System Builder</b> also using settings on the <b>SECEDED</b> tab. For more information, see <a href="#">Table 42</a> , page 60.

**Table 29 • ESRAM\_MAX\_LAT**

Bit Number	Name	Reset Value	Description
[31:6]	Reserved	0	
[5:3]	SW_MAX_LAT_ESRAM1	0x1	Defines the maximum number of cycles the processor bus will wait for eSRAM1 when it is being accessed by a master with a weighted round robin (WRR) priority scheme. The latency values are as given in <a href="#">Table 30</a> , page 54.
[2:0]	SW_MAX_LAT_ESRAM0	0x1	Defines the maximum number of cycles the processor bus will wait for eSRAM0 when it is being accessed by a master with a WRR priority scheme. It is configurable from 1 to 8 (8 by default). The latency values are as given in <a href="#">Table 30</a> , page 54.

The following table gives eSRAM maximum latency values, where x is either 0 or 1.

**Table 30 • eSRAM Maximum Latency Values**

SW_MAX_LAT_ESRAM<X>	Latency
0	8 (default)
1	1
2	2
3	3
4	4
5	5
6	6
7	7

**Table 31 • ESRAM\_PIPELINE\_CR**

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0	
0	ESRAM_PIPELINE_ENABLE	0x1	Controls the pipeline present in the read path of eSRAM memory. Allowed values: 0: Pipeline will be bypassed. 1: Pipeline will be present in the memory read path.

**Table 32 • ESRAM0\_EDAC\_CNT**

Bit Number	Name	Reset Value	Description
[31:16]	ESRAM0_EDAC_CNT_2E	0	16-bit counter that counts the number of 2-bit uncorrected errors for eSRAM0. The counter will not roll back and will stay at its maximum value.
[15:0]	ESRAM0_EDAC_CNT_1E	0	16-bit counter that counts the number of 1-bit corrected errors for eSRAM0. The counter will not roll back and will stay at its maximum value.

**Note:** See [Table 40](#), page 58 to clear the counter.

**Table 33 • ESRAM1\_EDAC\_CNT**

Bit Number	Name	Reset Value	Description
[31:16]	ESRAM1_EDAC_CNT_2E	0	16-bit counter that counts the number of 2-bit uncorrected errors for eSRAM1. The counter will not roll back and will stay at its maximum value.
[15:0]	ESRAM1_EDAC_CNT_1E	0	16-bit counter that counts the number of 1-bit corrected errors for eSRAM1. The counter will not roll back and will stay at its maximum value.

**Note:** See [Table 40](#), page 58 to clear the counter.

**Table 34 • ESRAM0\_EDAC\_ADR**

Bit Number	Name	Reset Value	Description
[31:25]	Reserved	0	
[25:13]	ESRAM0_EDAC_2E_AD	0	Stores the address from eSRAM0 on which a 2-bit SECEDED error has occurred.
[12:0]	ESRAM0_EDAC_1E_AD	0	Stores the address from eSRAM0 on which a 1-bit SECEDED error has occurred.

**Table 35 • ESRAM1\_EDAC\_ADR**

Bit Number	Name	Reset Value	Description
[31:25]	Reserved	0	
[25:13]	ESRAM1_EDAC_2E_AD	0	Stores the address from eSRAM1 on which a 2-bit SECEDED error has occurred.
[12:0]	ESRAM1_EDAC_1E_AD	0	Stores the address from eSRAM1 on which a 1-bit SECEDED error has occurred.

**Table 36 • MM4\_5\_DDR\_FIC\_SECURITY/MM4\_5\_FIC64\_SECURITY**

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0	
9	MM4_5_DDR_FIC_MS6_ALLOWED_W	1	Write security bits for masters 4, 5, and DDR_FIC to slave 6 (HPMS DDR bridge). If not set, masters 4, 5 and DDR_FIC will not have write access to slave 6.
8	MM4_5_DDR_FIC_MS6_ALLOWED_R	1	Read security bits for masters 4, 5, and DDR_FIC to slave 6 (HPMS DDR bridge). If not set, masters 4, 5, and DDR_FIC will not have read access to slave 6.
7	MM4_5_DDR_FIC_MS3_ALLOWED_W	1	Write security bits for masters 4, 5, and DDR_FIC to slave 3 (eNVM1). If not set, masters 4, 5, and DDR_FIC will not have write access to slave 3.
6	MM4_5_DDR_FIC_MS3_ALLOWED_R	1	Read security bits for masters 4, 5, and DDR_FIC to slave 3 (eNVM1). If not set, masters 4, 5, and DDR_FIC will not have read access to slave 3.
5	MM4_5_DDR_FIC_MS2_ALLOWED_W	1	Write Security Bits for masters 4, 5, and DDR_FIC to slave 2 (eNVM0). If not set, masters 4, 5, and DDR_FIC will not have write access to slave 2.
4	MM4_5_DDR_FIC_MS2_ALLOWED_R	1	Read security bits for masters 4, 5, and DDR_FIC to slave 2 (eNVM0). If not set, masters 4, 5, and DDR_FIC will not have read access to slave 2.
3	MM4_5_DDR_FIC_MS1_ALLOWED_W	1	Write security bits for masters 4, 5, and DDR_FIC to slave 1 (eSRAM1). If not set, masters 4, 5, and DDR_FIC will not have write access to slave 1.
2	MM4_5_DDR_FIC_MS1_ALLOWED_R	1	Read security bits for masters 4, 5, and DDR_FIC to slave 1 (eSRAM1). If not set, masters 4, 5, and DDR_FIC will not have read access to slave 1.
1	MM4_5_DDR_FIC_MS0_ALLOWED_W	1	Write security bits for masters 4, 5, and DDR_FIC to slave 0 (eSRAM0). If not set, masters 4, 5, and DDR_FIC will not have write access to slave 0.
0	MM4_5_DDR_FIC_MS0_ALLOWED_R	1	Read security bits for masters 4, 5, and DDR_FIC to slave 0 (eSRAM0). If not set, masters 4, 5, and DDR_FIC will not have read access to slave 0.

**Note:** See [Figure 42](#), page 61 for more information on AHB Bus Matrix masters and slaves.

**Table 37 • MM3\_7\_SECURITY**

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0	
9	MM3_7_MS6_ALLOWED_W	1	Write security bits for masters 3, 7 to slave 6 (HPMS DDR bridge). If not set, masters 3 and 7 will not have write access to slave 6.
8	MM3_7_MS6_ALLOWED_R	1	Read security bits for masters 3, 7 to slave 6 (HPMS DDR bridge). If not set, masters 3 and 7 will not have read access to slave 6.
7	MM3_7_MS3_ALLOWED_W	1	Write security bits for masters 3, 7 to slave 3 (eNVM1). If not set, masters 3 and 7 will not have write access to slave 3.
6	MM3_7_MS3_ALLOWED_R	1	Read security bits for masters 3, 7 to slave 3 (eNVM1). If not set, masters 3 and 7 will not have read access to slave 3.
5	MM3_7_MS2_ALLOWED_W	1	Write security bits for masters 3, 7 to slave 2 (eNVM0). If not set, masters 3 and 7 will not have write access to slave 2.
4	MM3_7_MS2_ALLOWED_R	1	Read security bits for masters 3, 7 to slave 2 (eNVM0). If not set, masters 3 and 7 will not have read access to slave 2.
3	MM3_7_MS1_ALLOWED_W	1	Write security bits for masters 3, 7 to slave 1 (eSRAM1). If not set, masters 3 and 7, will not have write access to slave 1.
2	MM3_7_MS1_ALLOWED_R	1	Read security bits for masters 3, 7 to slave 1 (eSRAM1). If not set, masters 3 and 7 will not have read access to slave 1.
1	MM3_7_MS0_ALLOWED_W	1	Write security bits for masters 3, 7 to slave 0 (eSRAM0). If not set, masters 3 and 7 will not have write access to slave 0.
0	MM3_7_MS0_ALLOWED_R	1	Read security bits for masters 3, 7 to slave 0 (eSRAM0). If not set, masters 3 and 7 will not have read access to slave 0.

**Note:** See [Figure 42](#), page 61 for more information on AHB Bus Matrix masters and slaves.

**Table 38 • MM9\_SECURITY**

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0	
9	MM9_MS6_ALLOWED_W	1	Write security bits for master 9 to slave 6 (HPMS DDR bridge). If not set, master 9 will not have write access to slave 6.
8	MM9_MS6_ALLOWED_R	1	Read security bits for master 9 to slave 6 (HPMS DDR bridge). If not set, master 9 will not have read access to slave 6.
7	MM9_MS3_ALLOWED_W	1	Write security bits for master 9 to slave 3 (eNVM1). If not set, master 9 will not have write access to slave 3.
6	MM9_MS3_ALLOWED_R	1	Read security bits for master 9 to slave 3 (eNVM1). If not set, master 9 will not have read access to slave 3.

**Table 38 • MM9\_SECURITY (continued)**

Bit Number	Name	Reset Value	Description
5	MM9_MS2_ALLOWED_W	1	Write security bits for master 9 to slave 2 (eNVM0). If not set, master 9 will not have write access to slave 2.
4	MM9_MS2_ALLOWED_R	1	Read security bits for master 9 to slave 2 (eNVM0). If not set, master 9 will not have read access to slave 2.
3	MM9_MS1_ALLOWED_W	1	Write security bits for master 9 to slave 1 (eSRAM1). If not set, master 9 will not have write access to slave 1.
2	MM9_MS1_ALLOWED_R	1	Read security bits for master 9 to slave 1 (eSRAM1). If not set, master 9 will not have read access to slave 1.
1	MM9_MS0_ALLOWED_W	1	Write security bits for master 9 to slave 0 (eSRAM0). If not set, master 9 will not have write access to slave 0.
0	MM9_MS0_ALLOWED_R	1	Read security bits for master 9 to slave 0 (eSRAM0). If not set, master 9 will not have read access to slave 0.

**Note:** See [Figure 42](#), page 61 for more information on AHB Bus Matrix masters and slaves.

**Table 39 • EDAC\_SR**

Bit Number	Name	Reset Value	Description
[31:14]	Reserved	0	
[13:6]	Reserved	0	
5	Reserved	0	
4	Reserved	0	
3	ESRAM1_EDAC_2E	0	Updated by the eSRAM_1 controller when a 2-bit SECEDED error has been detected for eSRAM1 memory.
2	ESRAM1_EDAC_1E	0	Updated by the eSRAM_1 Controller when a 1-bit SECEDED error has been detected and is corrected for eSRAM1 memory.
1	ESRAM0_EDAC_2E	0	Updated by the eSRAM_0 controller when a 2-bit SECEDED error has been detected for eSRAM0 memory.
0	ESRAM0_EDAC_1E	0	Updated by the eSRAM_0 controller when a 1-bit SECEDED error has been detected and is corrected for eSRAM0 memory.

**Table 40 • CLR\_EDAC\_COUNTERS**

Bit Number	Name	Reset Value	Description
[31:14]	Reserved	0	
[13:6]	Reserved	0	
5	Reserved	0	
4	Reserved	0	

**Table 40 • CLR\_EDAC\_COUNTERS (continued)**

Bit Number	Name	Reset Value	Description
3	ESRAM1_EDAC_CNTCLR_2E	0	Pulse generated to clear the 16-bit counter value in eSRAM1 corresponding to the count value of EDAC 2-bit errors. This in turn clears the upper 16 bits of the ESRAM1_EDAC_CNT register (see <a href="#">Table 33</a> , page 55).
2	ESRAM1_EDAC_CNTCLR_1E	0	Pulse generated to clear the 16-bit counter value in eSRAM1 corresponding to count value of EDAC 1-bit errors. This in turn clears the lower 16 bits of the ESRAM1_EDAC_CNT register.
1	ESRAM0_EDAC_CNTCLR_2E	0	Pulse generated to clear the 16-bit counter value in eSRAM0 corresponding to count value of EDAC 2-bit Errors. This in turn clears the upper 16 bits of the ESRAM1_EDAC_CNT register.
0	ESRAM0_EDAC_CNTCLR_1E	0	Pulse generated to clear the 16-bit counter value in eSRAM0 corresponding to the count value of EDAC 1-bit errors. This in turn clears the lower 16 bits of the ESRAM1_EDAC_CNT register.

**Table 41 • EDAC\_IRQ\_ENABLE\_CR**

Bit Number	Name	Reset Value	Description
[31:15]	Reserved	0	
[13:6]	Reserved	0	
5	Reserved	0	
4	Reserved	0	
3	ESRAM1_EDAC_2E_EN	0	Allows the 2-bit error EDAC for eSRAM1 status update to be disabled. Allowed values: 0: Disabled. 1: Enabled.
2	ESRAM1_EDAC_1E_EN	0	Allows the 1-bit error EDAC for eSRAM1 status update to be disabled. Allowed values: 0: Disabled. 1: Enabled.
1	ESRAM0_EDAC_2E_EN	0	Allows the 2-bit error EDAC for eSRAM0 status update to be disabled. Allowed values: 0: Disabled. 1: Enabled.
0	ESRAM0_EDAC_1E_EN	0	Allows the 1-bit error EDAC for eSRAM0 status update to be disabled. Allowed values: 0: Disabled. 1: Enabled.

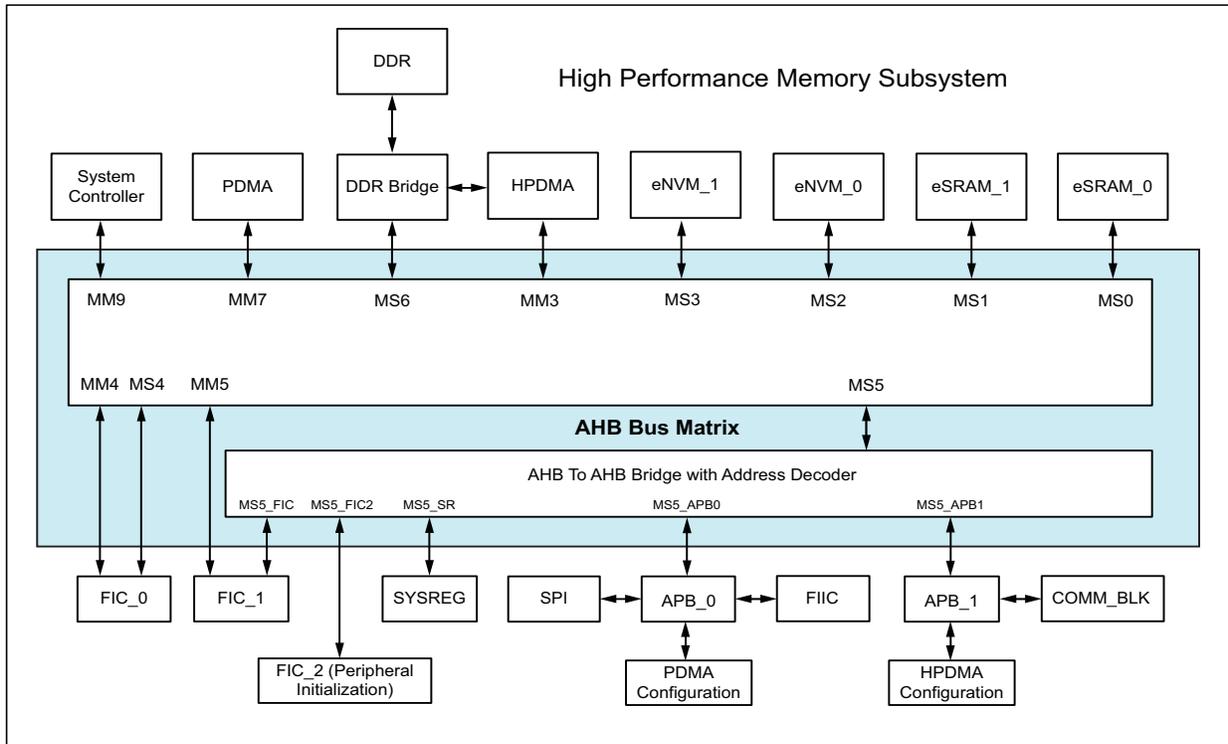
**Table 42 • EDAC\_CR**

Bit Number	Name	Reset Value	Description
[31:7]	Reserved	0	
[6:2]	Reserved	0	
1	ESRAM1_EDAC_EN	0	Allows the EDAC for eSRAM1 to be disabled. Allowed values: 0: Disabled 1: Enabled
0	ESRAM0_EDAC_EN	0	Allows the EDAC for eSRAM0 to be disabled. Allowed values: 0: Disabled 1: Enabled

# 4 AHB Bus Matrix

The AHB bus matrix is a multi-layer AHB matrix. It is not a full crossbar switch, but a customized subset of a full switch. It works purely as an AHB-Lite matrix. The IGLOO2 FPGA AHB bus matrix has five masters and seven direct slaves as depicted in the following figure. One master is permitted to access a slave at the same time another master is accessing a different slave. If more than one master is attempting to access the same slave simultaneously, arbitration for that slave is performed.

**Figure 42 • AHB Bus Matrix Masters and Slaves**



The preceding figure shows the connectivity of masters and slaves in the AHB bus matrix. Nomenclature such as MM0 and MS0 refers to a mirrored master and a mirrored slave. A mirrored master port in the matrix connects directly to an AHB master; it has the same set of signals, but the direction of the signals is described relative to the other end of the connection.

A mirrored slave port in the matrix connects directly to an AHB slave. Only a subset of the full set of theoretical paths is implemented within the AHB bus matrix. The AHB bus matrix performs the address decoding of all slaves except for slaves that connect to the AHB-to-AHB bridge.

## 4.1 Functional Description

This section provides a detailed description of the AHB bus matrix.

### 4.1.1 Architecture Overview

Figure 43, page 63 shows the interconnection between the master stage blocks and the slave stage blocks. The basic building blocks of the AHB bus matrix are the master stage block with an address decoder and the slave stage block with a slave arbiter. Each master interfaces with the master stage block and each slave interfaces with the slave stage block. The masters and slaves connect as specified in the following table.

An address decoder sub-block in each master stage generates the slave select signal to the corresponding slave. A slave arbiter sub-block in each slave stage generates the address-ready signal to the selected master.

**Table 43 • AHB Bus Matrix Connectivity**

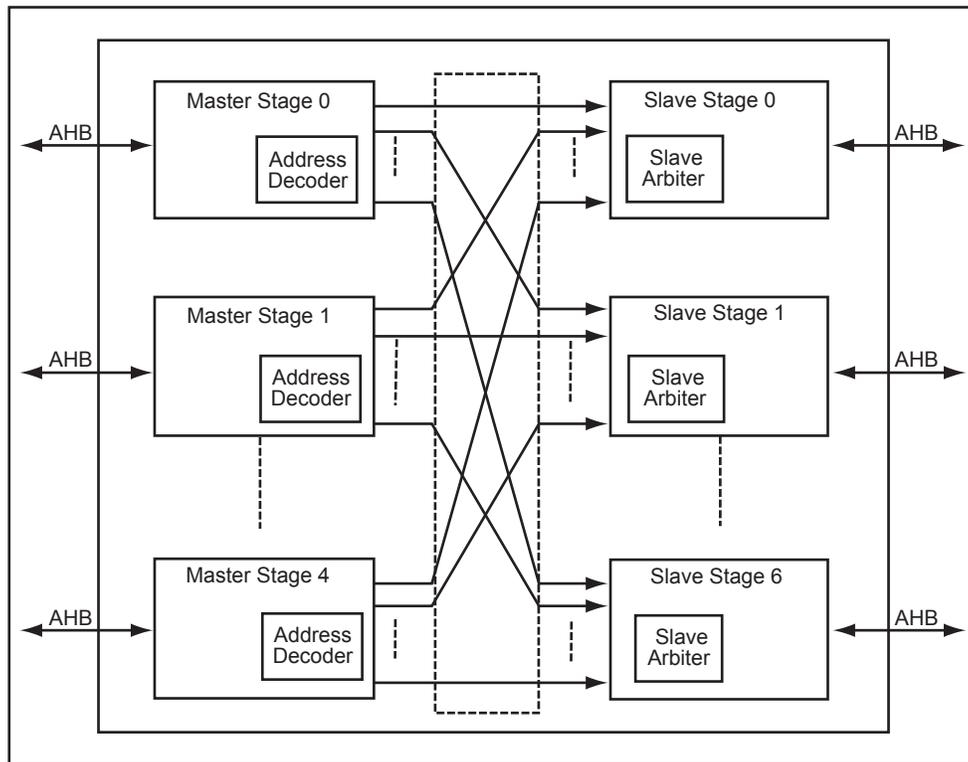
			Masters				
			System Controller	HPDMA	FIC_0	FIC_1	PDMA
			MM9	MM3	MM4	MM5	MM7
Priority			4	4	4	4	4
Arbitration			Fixed	WRR	WRR	WRR	WRR
<b>Slave</b>	eSRAM0	MS0	RW	RW	RW	RW	RW
	eSRAM1	MS1	RW	RW	RW	RW	RW
	eNVM_0	MS2	RW <sup>1</sup>	R <sup>1</sup>	RW <sup>1</sup>	RW <sup>1</sup>	RW <sup>1</sup>
	eNVM_1	MS3	RW <sup>1</sup>	R <sup>1</sup>	RW <sup>1</sup>	RW <sup>1</sup>	RW <sup>1</sup>
	FIC_0	MS4	RW	RW	RW	RW	RW
	FIC_1	MS5	RW	RW	RW	RW	RW
	SYSREG		RW		RW	RW	
	APB_0		RW		RW	RW	RW
	APB_1		RW		RW	RW	RW
	HPMS DDR Bridge	MS6	RW		RW	RW	RW

- Exercise caution when commanding the eNVM to program or erase data. Other masters in the system may not be aware that the eNVM is unavailable if it is in a program or erase cycle. Microsemi recommends using some form of software semaphore to control access.

Reads or writes to areas not allowed cause the AHB bus matrix to complete the transaction with an HRESP error indication. An error bit is set in the HPMS\_EXTERNAL\_SR[SW\_ERRORSTATUS] field. The following types of errors can occur:

- Write by an enabled master to a slave that is not RW
- Write by an enabled master to addresses not corresponding to a slave
- Write by the fabric master to the protected region
- Write by a disabled master to any location
- Read by an enabled master to any slave that is not R or RW
- Read by an enabled master to addresses not corresponding to a slave
- Read by the fabric master to the protected region
- Read by a disabled master to any location

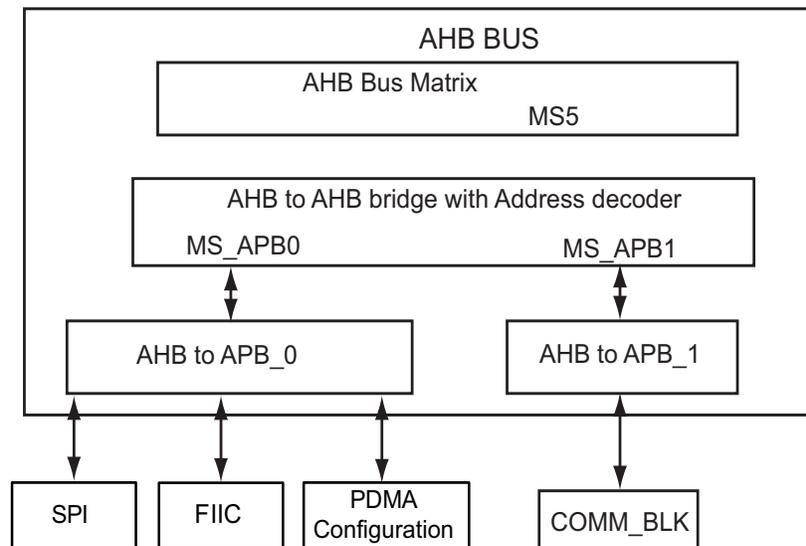
**Figure 43 • Master Stage and Slave Stage Interconnection**



To reduce the load on the AHB bus matrix, some of the low-performance peripherals are connected through the synchronous AHB-to-AHB bridge with an address decoder. The AHB bus matrix is constructed of combinatorial logic, except for the AHB-to-AHB bridge, which inserts a one-cycle delay in each direction.

The following figure shows the block diagram of all the APB peripherals connected to AHB bus matrix using the AHB-to-AHB bridge. The APB peripherals are connected through the AHB to APB bus.

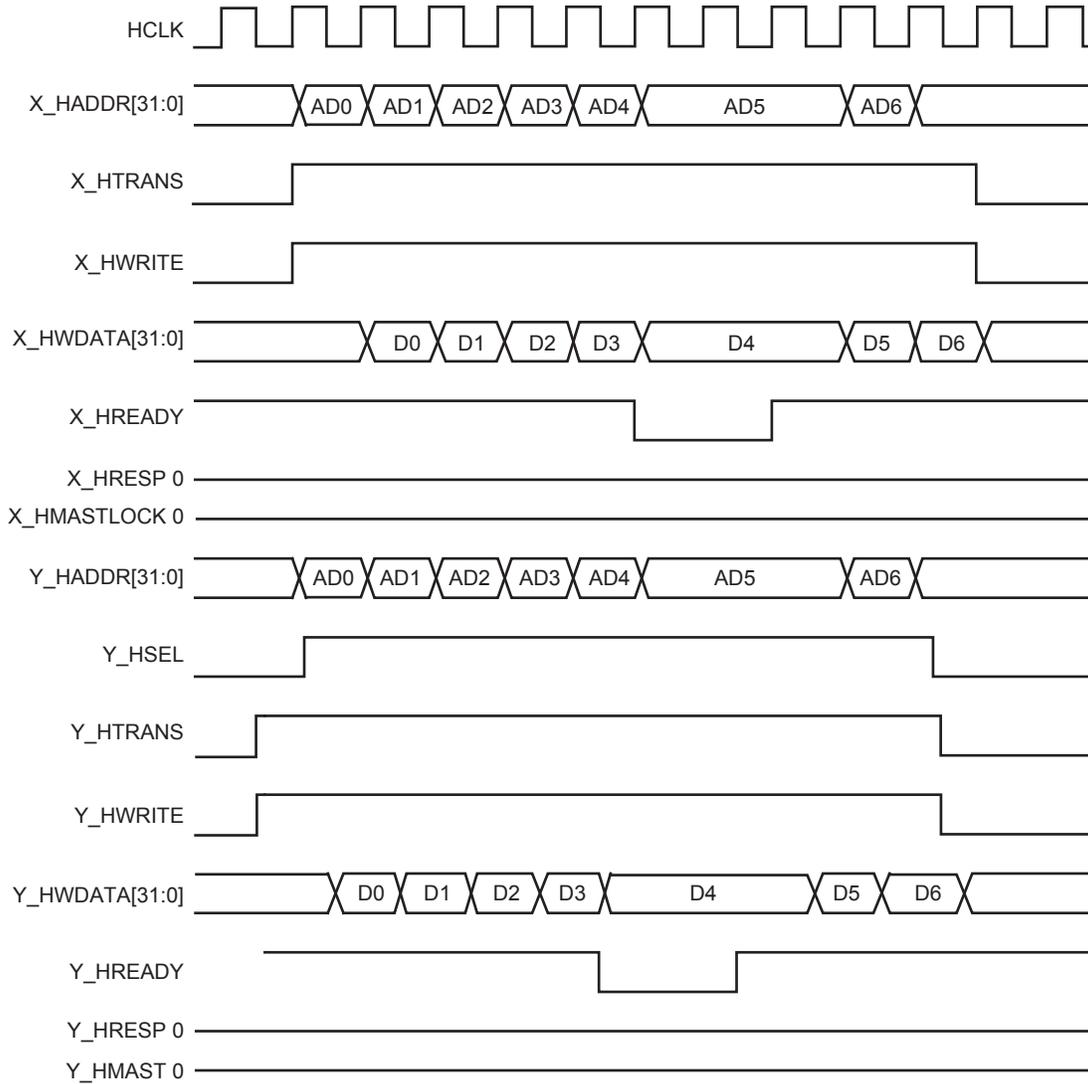
**Figure 44 • APB Destinations Connected to AHB Bus Matrix**



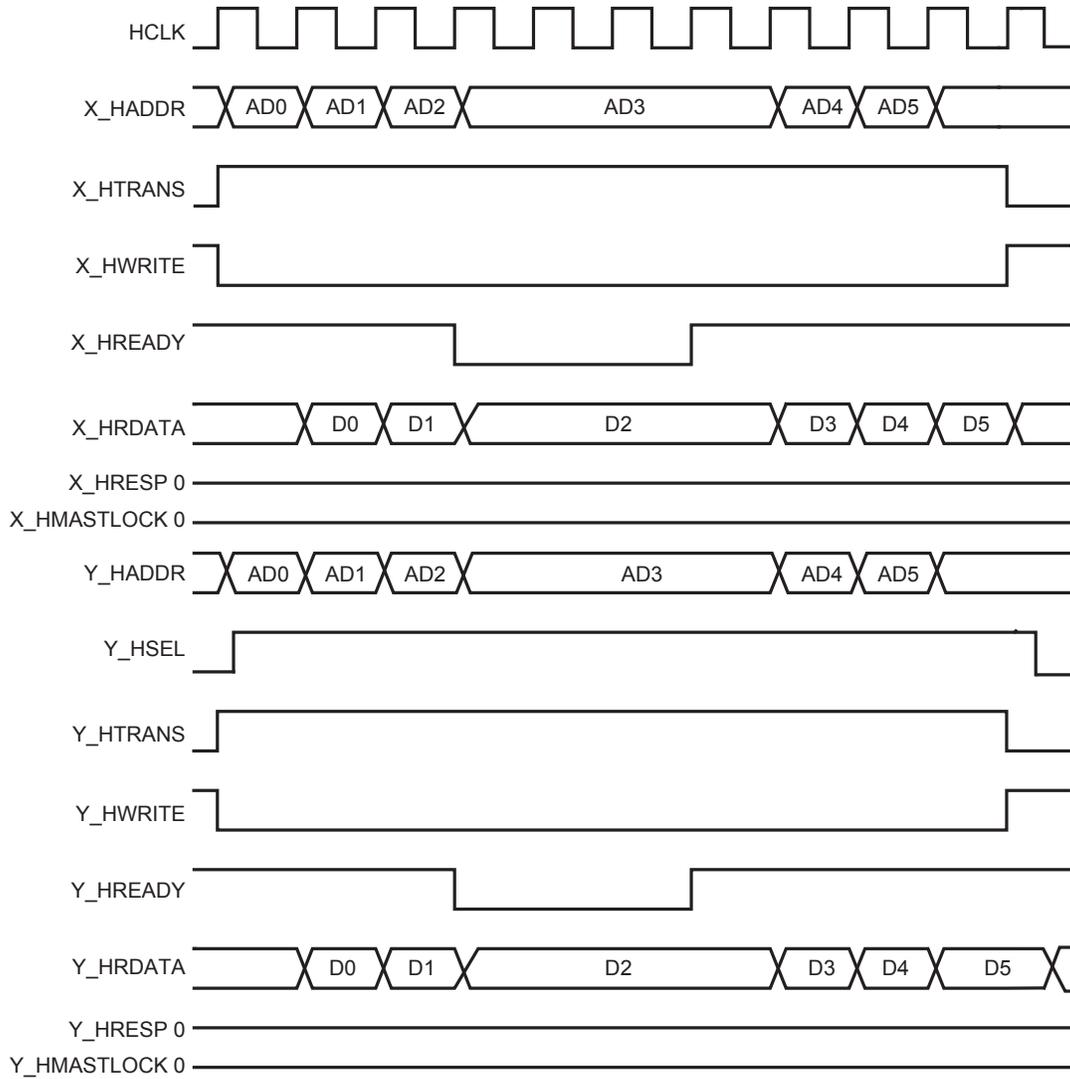
### 4.1.2 Timing Diagrams

The following figures are the functional timing diagrams for AHBL read/write transactions through the AHB bus matrix and AHB-to-AHB bridge. Signals to/from a master are denoted by X in the signal name, and signals to/from a slave are denoted with Y in the signal name. For example, if any master initiates the transactions of read/write to the eSRAM slave then the signals with X in the signal name indicates the signals of the master and signals with Y indicate slave eSRAM signals.

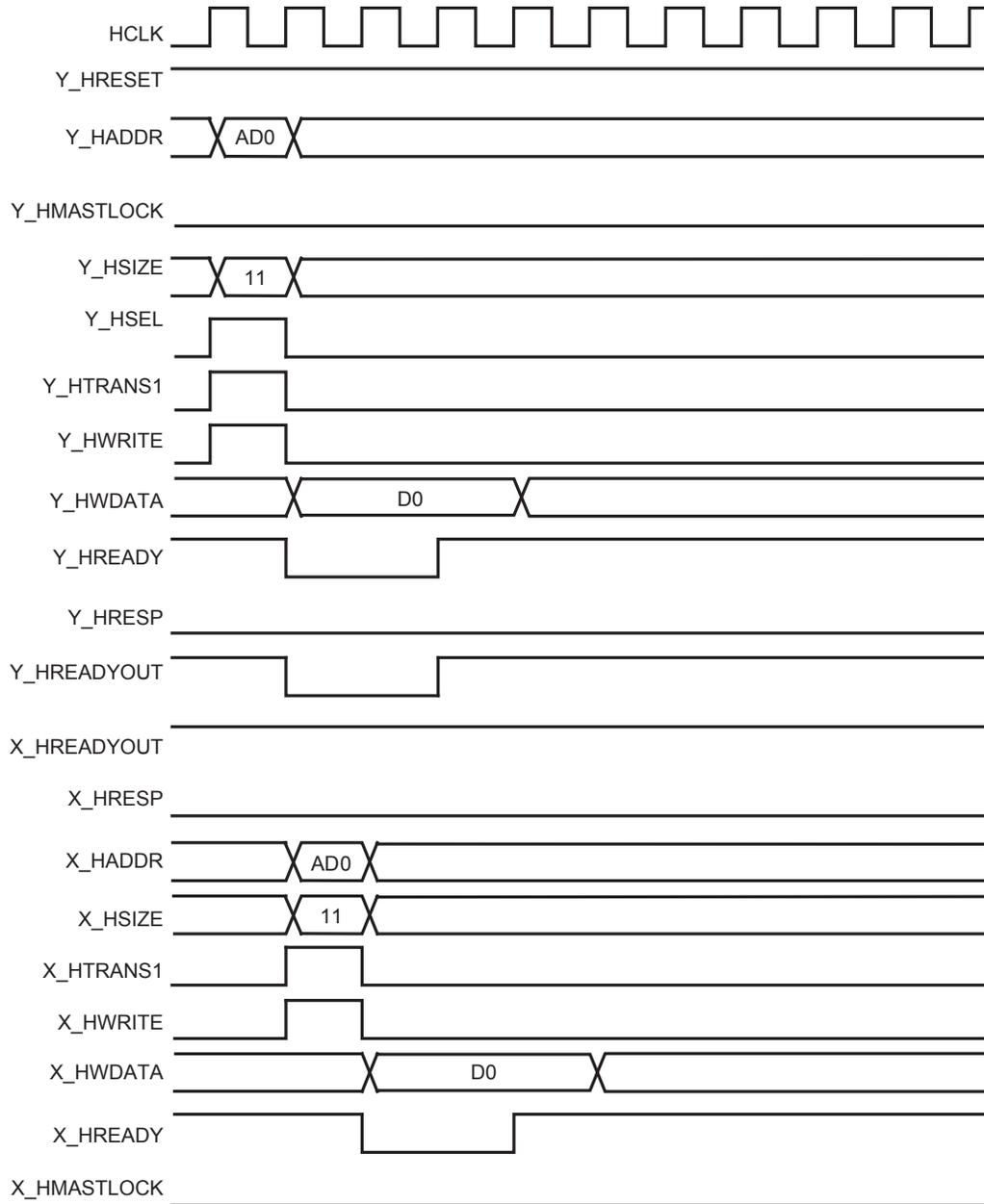
**Figure 45 • AHB-Lite Write Transactions**



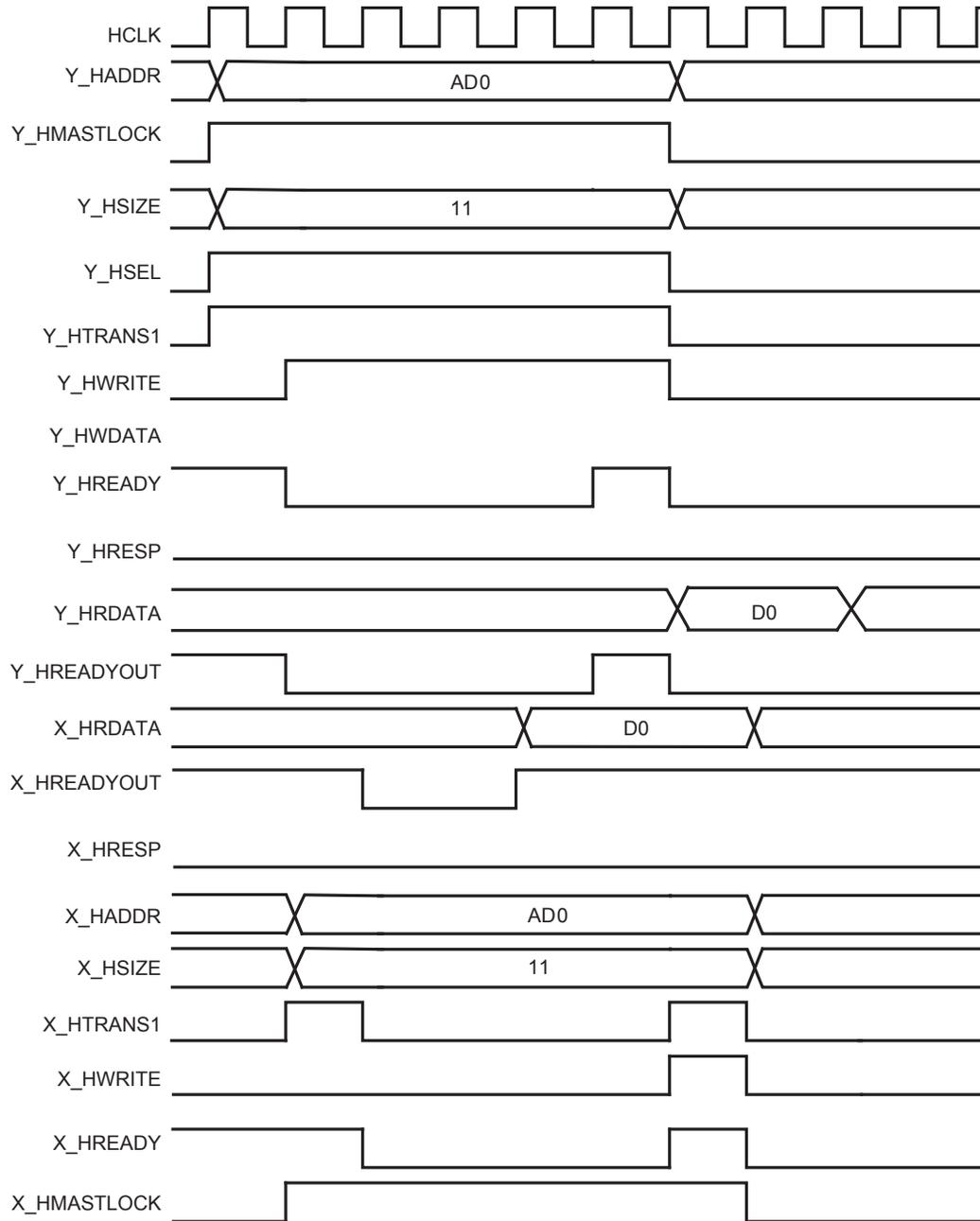
**Figure 46 • AHB-Lite Read Transactions**



**Figure 47 • AHB-to-AHB Write Transactions**



**Figure 48 • AHB-to-AHB Read Transactions**



### 4.1.3 Details of Operation

#### 4.1.3.1 Slave Arbitration

Each of the slave devices on the AHB bus matrix contains an arbiter. Arbitration is done at two levels. At the first level, the fixed higher priority master is evaluated for any access request to the slave. At the second level, the remaining masters are evaluated in round robin fashion for any access request to the slave. The priority levels of the buses with fixed priority and the buses with round robin priority are given in the following table.

**Table 44 • WRR Masters**

Masters		Priority	Arbitration
System controller	MM9	4	Fixed
HPDMA	MM3	4	WRR
FIC_0	MM4	4	WRR
FIC_1	MM5	4	WRR
PDMA	MM7	4	WRR

##### 4.1.3.1.1 Arbitration Parameters

The following slave arbitration configuration parameters are user programmable registers in the SYSREG block.

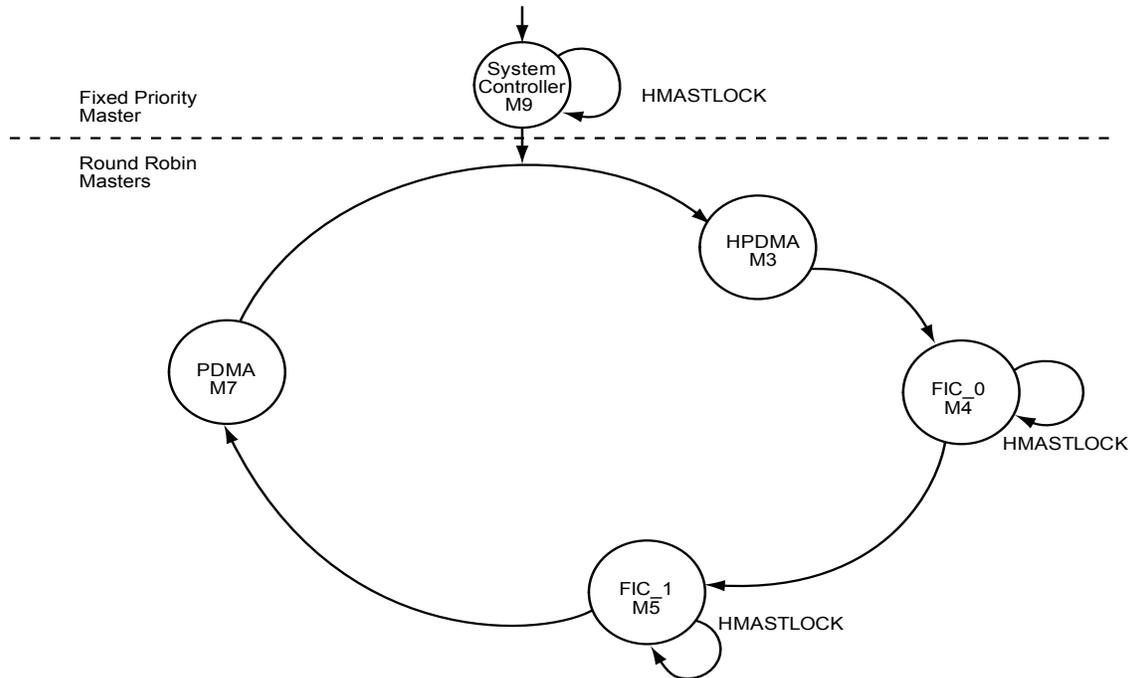
- Programmable slave maximum latency:** Slave maximum latency, `ESRAM_MAX_LAT`, decides the peak wait time for a fixed priority master arbitrating for eSRAM access while the WRR master is accessing the slave. After the defined latency period, the WRR master will have to re-arbitrate for slave access. Slave maximum latency can be configurable from one to eight clock cycles (eight by default). `ESRAM_MAX_LAT` is only supported for fixed priority masters addressing eSRAM slaves; it has no effect on WRR masters.
- Programmable weight:** `MASTER_WEIGHT0_CR` and `MASTER_WEIGHT1_CR` are 5-bit programmable registers located in the SYSREG block that define the number of consecutive transfers the weighted master can perform without being interrupted by a fixed priority master, or before moving onto the next master in the WRR cycle.

### 4.1.3.1.2 Pure Round Robin Arbitration

This is the default arbitration mode after reset. The programmable weight value of each master is set to 1, and ESRAM\_MAX\_LAT = 1.

The arbitration scheme for each slave port is identical in pure round robin arbitration, as shown in the following figure. The fixed priority master have priority over other masters. Each WRR master accessing a slave has equal priority on a round robin basis. However, if a locked transaction occurs, the master issuing the lock maintains ownership of the slave until the locked transaction completes.

**Figure 49 • Pure Round Robin and Fixed Priority Slave Arbitration Scheme**



The following table gives an example of a pure round robin and fixed priority arbitration scenario for eSRAM1. This example illustrates default AHB bus matrix behavior.

**Table 45 • Pure Round Robin and Fixed Priority Arbitration Scenario for eSRAM1**

Master	1	2	3	4	5	6
System Controller: M9		eSRAM1		eSRAM1		
HPDMA: M3	eSRAM1					
FIC_0: M4	eSRAM1					
FIC_1: M5	eSRAM1					
PDMA: M7	eSRAM1					
eSRAM1: S1	HPDMA M3	System Controller M9	FIC_0 M4	System Controller M9	FIC_1 M5	PDMA M7

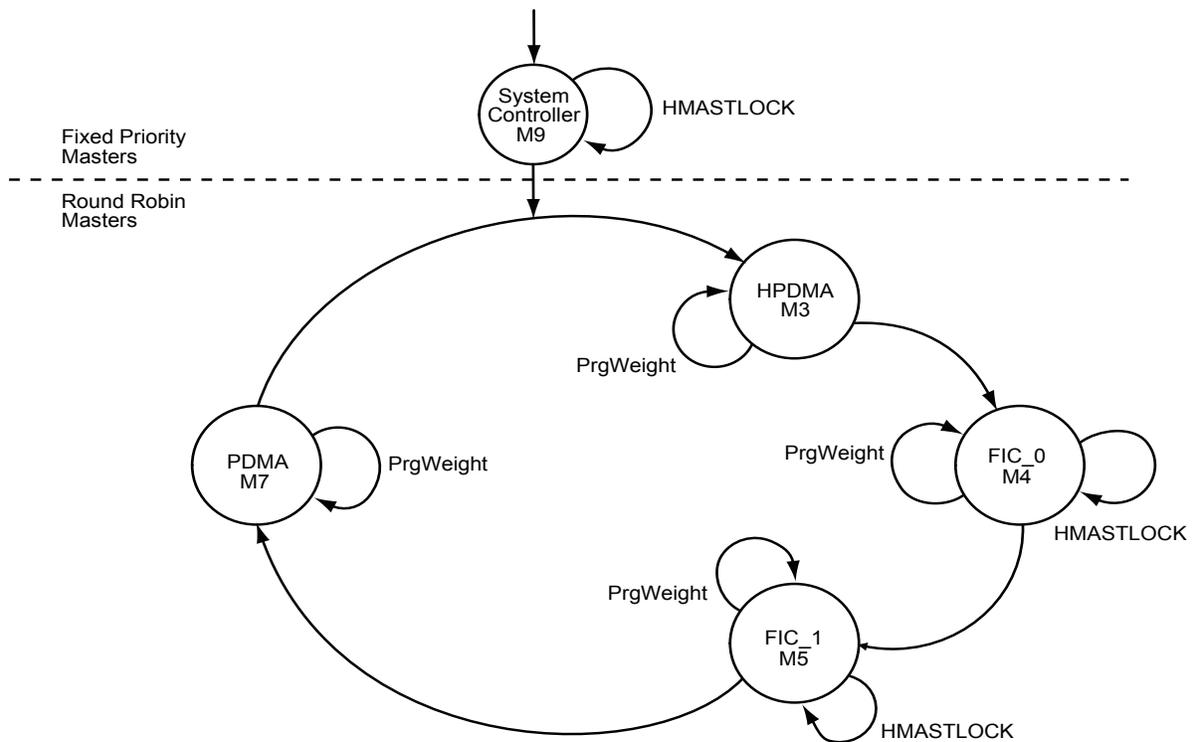
In this table, WRR masters and fixed priority master arbitrate for the S1 (eSRAM1) slave during HCLK cycle 1. The last row in the table, labeled eSRAM1: S1, shows which of the masters obtains access to the slave according to the arbitration in that clock cycle. In the first cycle, master M3 (HPDMA) is granted access, since it is the first master in the round robin scheme. In the second cycle, even though master M4 is scheduled to get access to the slave as per the round robin scheme, the M9 master (System Controller) is granted access since it has a higher priority. In the third cycle, the master M4 (FIC\_0) in the round robin scheme is granted access. In the fourth cycle, M9 (System Controller) is trying for access.

WRR masters are delayed while the fixed priority master get access to the slave. The remaining cycles are consumed by the WRR masters in order.

#### 4.1.3.1.3 WRR Arbitration

In this mode, the slave arbitration parameters, programmable weight (SW\_WEIGHT\_<master>) and eSRAM slave maximum latency (SW\_MAX\_LAT\_ESRAM<0/1> of ESRAM\_MAX\_LAT) can be configured to operate as WRR arbitration. The slave arbiter operates on a round robin basis, with each master having a maximum of N consecutive access opportunities to the slave in each round of arbitration. The value of N is determined by the programmed weight for the master and eSRAM slave maximum latency. Programmable weight values can be changed dynamically. The following figure shows the WRR slave arbitration scheme. At each stage, the arbiter checks whether that master is requesting access. If so, the master performs N transfers equal to its programmed weight and then has to re-arbitrate for the bus. For a WRR master, the WRR priority in the round robin sequence changes after the programmed number of transfers. If a locked transaction occurs, the master issuing the lock (HMASTLOCK = 1) maintains ownership of the slave until the locked transaction completes.

Figure 50 • WRR and Fixed Priority Slave Arbitration Scheme



WRR with fixed priority arbitration allows more efficient usage of slave bandwidth in cases where the slaves have a penalty when transitioning from one master to another.

The eSRAM AHB controller inserts an idle cycle every time there is a write followed by a read, enabling WRR can increase the effective eSRAM bandwidth during this time from 66% to 94% of the theoretical maximum. If a sequence of locked transfers is in progress, the locked master remains selected by the slave arbiter until the lock sequence is finished, regardless of the number of transfers.

#### 4.1.3.1.4 Arbitration for Non-eSRAM Slaves

In non-eSRAM slaves, any WRR master getting access to the slave can perform uninterrupted transactions equal to its programmed weight before re-arbitrating for the slave. Thus, for example, if FIC\_1 is programmed with a weight of 8, it can do 8 continuous transactions with the slave even if the high priority master is requesting access to the slave. Only after completing 8 transfers, the high priority master will gain access to the slave.

The following table gives an arbitration scenario for a non-eSRAM slave. In this scenario, master M5 (FIC\_1) starts a burst of twelve transfers (reads typically for accesses to eNVM) to slave S2 (eNVM\_0) in

the first clock cycle. In the second clock, fixed priority master M9 (System Controller) bus tries to access the same slave. Since the programmed weight of M5 master is 8, the M9 master does not gain access to the slave until M5 completes eight transfers. As seen in the following table, the M9 master gains access to the slave only after the M5 master completes eight transfers, which is in the 9th clock cycle. The M5 master has to re-arbitrate for the slave to complete the remaining transfers. So the maximum latency seen by the master M9 is equal to the programmed weight of 8.

**Table 46 • WRR and Fixed Priority Arbitration Scenario for eNVM\_0**

Master	HCLK															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
System controller M9		S2- B4														
FIC_1: M5	S2- B12															
eNVM_0:S2	M5- B1	M5- B2	M5- B3	M5- B4	M5- B5	M5- B6	M5- B7	M5- B8	M9- B1	M9- B2	M9- B3	M9- B4	M5- B9	M5- B10	M5- B11	M5- B12

#### 4.1.3.1.5 Arbitration for eSRAM Slaves

For eSRAM slaves, a programmable maximum latency parameter SW\_MAX\_LAT\_ESRAM<0/1> is available to optimize arbitration for the eSRAM slaves from a fixed priority master. The parameter SW\_MAX\_LAT\_ESRAM\_<0/1> sets a ceiling as to the number of cycles the fixed priority master, has to wait before accessing an eSRAM slave that is currently being accessed by a WRR master. When a WRR master has a programmable weight greater than the SW\_MAX\_LAT\_ESRAM<0/1> value, the WRR master will have to re-arbitrate for the slave after SW\_MAX\_LAT\_ESRAM<0/1> cycles. The following equation gives the maximum latency seen by a fixed priority master while accessing an eSRAM slave:

$$\text{Maximum latency seen by the fixed priority master} = \min\{\text{programmable weight (WRR master)}, \text{SW\_MAX\_LAT\_ESRAM}\langle 0/1 \rangle\}$$

For example, if SW\_WEIGHT\_HPDM is set to 18 and SW\_MAX\_LAT\_ESRAM0 is set to 4, then the maximum latency is  $\min\{18, 4\} = 4$ . Similarly, if SW\_WEIGHT\_PDMA is set to 2 and SW\_MAX\_LAT\_ESRAM1 is set to 6, then the maximum latency is  $\min\{2, 6\} = 2$ . The following table depicts a typical scenario.

**Table 47 • WRR Arbitration Scenario for eSRAM\_0 slave**

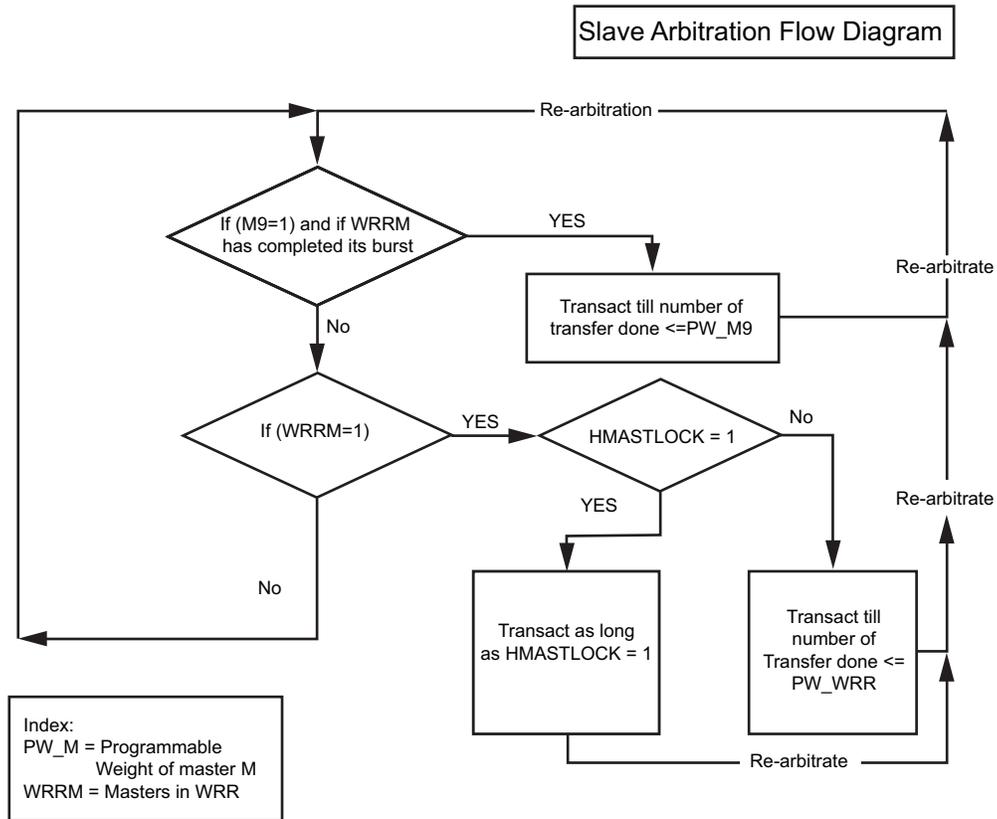
Master	HCLK											
	1	2	3	4	5	6	7	8	9	10	11	12
System controller M9		S0-B4										
PDMA: M7	S0-B8											
eSRAM_0: S0	M7-B1	M7-B2	M7-B3	M7-B4	M9-B1	M9-B2	M9-B3	M9-B4	M7-B5	M7-B6	M7-B7	M7-B8

In this scenario, the slave maximum latency is set to 4 and the master programmable weight is set to 8, so the maximum latency seen by the fixed priority master is  $\min\{4, 8\} = 4$ . When the WRR master starts transactions with the eSRAM slave, it can perform a number of transactions equal to the programmed maximum latency or the programmed weight, whichever is less, before re-arbitrating for the slave.

### 4.1.3.1.6 Slave Arbitration Flow Diagram

The following figure shows the slave arbitration flow diagram depicting the grant of access to master requesting for slave access. At each stage the arbiter checks whether that master is requesting for an access. If yes, then the master can do number of transfers equal to its programmed weight and then has to re-arbitrate for the bus. In case of WRR master, after the programmed number of transfer WRR priority changes for that master in round robin sequence.

Figure 51 • Slave Arbitration Flow Diagram



## 4.1.4 System Memory Map

The AHB bus matrix is responsible for implementing the address decoding of all masters to all slaves, so it defines the system memory map. The following figure shows the default system memory map for IGL002 devices.

Figure 52 • Default System Memory Map

Memory Map of System Controller, FPGA Fabric Master, Peripheral DMA

FPGA Fabric FIC Region5	0xF0000000 - 0xFFFFFFFF
	0xE0000000 - 0xEFFFFFFF
MDDR Space 3	0xD0000000 - 0xDFFFFFFF
MDDR Space 2	0xC0000000 - 0xCFFFFFFF
MDDR Space 1	0xB0000000 - 0xBFFFFFFF
MDDR Space 0	0xA0000000 - 0xAFFFFFFF
FPGA Fabric FIC Region4	0x90000000 - 0x9FFFFFFF
FPGA Fabric FIC Region3	0x80000000 - 0x8FFFFFFF
FPGA Fabric FIC Region2	0x70000000 - 0x7FFFFFFF
	0x60100000 - 0x6FFFFFFF
AHB-to-eNVM 1 Registers	0x600C0000 - 0x600FFFFF
AHB-to-eNVM 0 Registers	0x60080000 - 0x600BFFFF
eNVM 1	0x60040000 - 0x6007FFFF
eNVM 0	0x60000000 - 0x6003FFFF
FPGA Fabric FIC Region1	0x50000000 - 0x5FFFFFFF
	0x44000000 - 0x4FFFFFFF
	0x42000000 - 0x43FFFFFF
	0x40410000 - 0x41FFFFFF
	0x40400000 - 0x4040FFFF
	0x40044000 - 0x403FFFFF
	0x40043000 - 0x40043FFF
	0x40042000 - 0x40042FFF
	0x40041000 - 0x40041FFF
	0x40039000 - 0x40040FFF
	0x40038000 - 0x40038FFF
	0x40030000 - 0x40037FFF
Config FDDR, PCIe 0, PCIe 1, etc.	0x40020400 - 0x4002FFFF
Config MDDR	0x40020000 - 0x400203FF
	0x40018000 - 0x4001FFFF
	0x40017000 - 0x40017FFF
	0x40016000 - 0x40016FFF
	0x40015000 - 0x40015FFF
	0x40014000 - 0x40014FFF
	0x40013000 - 0x40013FFF
	0x40012000 - 0x40012FFF
	0x40011000 - 0x40011FFF
	0x40010000 - 0x40010FFF
	0x40007000 - 0x4000FFFF
	0x40006000 - 0x40006FFF
	0x40005000 - 0x40005FFF
	0x40004000 - 0x40004FFF
	0x40003000 - 0x40003FFF
	0x40002000 - 0x40002FFF
	0x40001000 - 0x40001FFF
	0x40000000 - 0x40000FFF
FPGA Fabric FIC Region0	0x30000000 - 0x3FFFFFFF
	0x24000000 - 0x2FFFFFFF
	0x22000000 - 0x23FFFFFF
	0x20014000 - 0x21FFFFFF
	0x20012000 - 0x20013FFF
ECC eSRAM 1	0x20010000 - 0x20011FFF
ECC eSRAM 0	0x20008000 - 0x2000FFFF
eSRAM 1	0x20000000 - 0x20007FFF
eSRAM 0	0x00080000 - 0x1FFFFFFF
	0x0007FFFF
eNVM (Fabric) virtual view	0x00000000

(63 K space allocation for devices outside HPMS)

### 4.1.4.1 Unimplemented Address Space

The AHB bus matrix performs address decoding based on the memory map defined in the preceding figure, to decide which slave, if any, is being addressed. Any access to memory space outside of these regions is considered unimplemented from the point of view of the AHB bus matrix. This access results in the assertion of a SW\_ERROR Status Register bit of the HPMS\_EXTERNAL\_SR Control Register, as well as the assertion of HRESP by the AHB bus matrix to the master. If any master attempts a write access to unimplemented address space, the AHB bus matrix completes the handshake to the master with an HRESP error indication. No write occurs to any slave.

If any master attempts a read access from unimplemented address space, the AHB bus matrix completes the handshake to the master with an HRESP error indication. Undefined data is returned. There may be further memory areas that are unimplemented, within individual slave memory regions. Depending on the slave, accesses may be aliased within these areas or not. Firmware should not perform writes to these locations because the aliasing may cause a write to another location within the slave. Data read from these intra-slave unimplemented regions may be undefined.

#### 4.1.4.2 Locked Transactions

HPMS supports locked accesses through its internal switch matrix to its slaves (eSRAM, DDR, FIC\_0, FIC\_1). HMASTLOCK signal is not routed to the fabric to allow a matrix to implement a lock-based arbitration system.

#### 4.1.4.3 Fabric Memory Map

There are six regions of 256 Kbytes each, which may be allocated to either FIC\_0 or FIC\_1 (fabric interrupt controller). This allows to configure large memory mapped windows into the FPGA fabric.

#### 4.1.4.4 Fabric Master Considerations

The following consideration should be taken into account while implementing fabric logic:

- Configuring the AHB bus matrix: For the mode changes (change of protection region, memory map mode, programmable weights and programmable maximum latency), all masters should be in IDLE STATE (where no data transfer is required) for a sufficient amount of time-10 IDLE cycles (ten clock cycles)-before and after the mode change.

#### 4.1.4.5 Memory Security

After reset, all master ports on the AHB bus matrix are enabled. There are separate user-defined flash configuration bits that control read and write access for each memory slave from various masters, which are organized in groups. The pairing of the masters and the slaves with respect to the bits set in the security registers are given in detail in the following table. Read access and write access can be independently controlled by separate read and write flash bits. The detailed bit configuration of these S registers is given in [System Register Block](#), page 196.

**Table 48 • Master and Slave Pairing**

SYSREG Register	Master	Slave				
		MS0	MS1	MS2	MS3	MS6
		eSRAM0	eSRAM1	eNVM_0	eNVM_1	HPMS DDR Bridge
MM4_5_FIC64_SECURITY	MM4: FIC_0	RW	RW	RW	RW	RW
	MM5: FIC_1	RW	RW	RW	RW	RW
	DDR_FIC	RW	RW	RW	RW	RW
MM3_7_SECURITY	MM3: HPDMA	RW	RW	RW	RW	RW
	MM7: PDMA	RW	RW	RW	RW	RW
MM9_SECURITY	MM9: System controller	RW	RW	RW	RW	RW

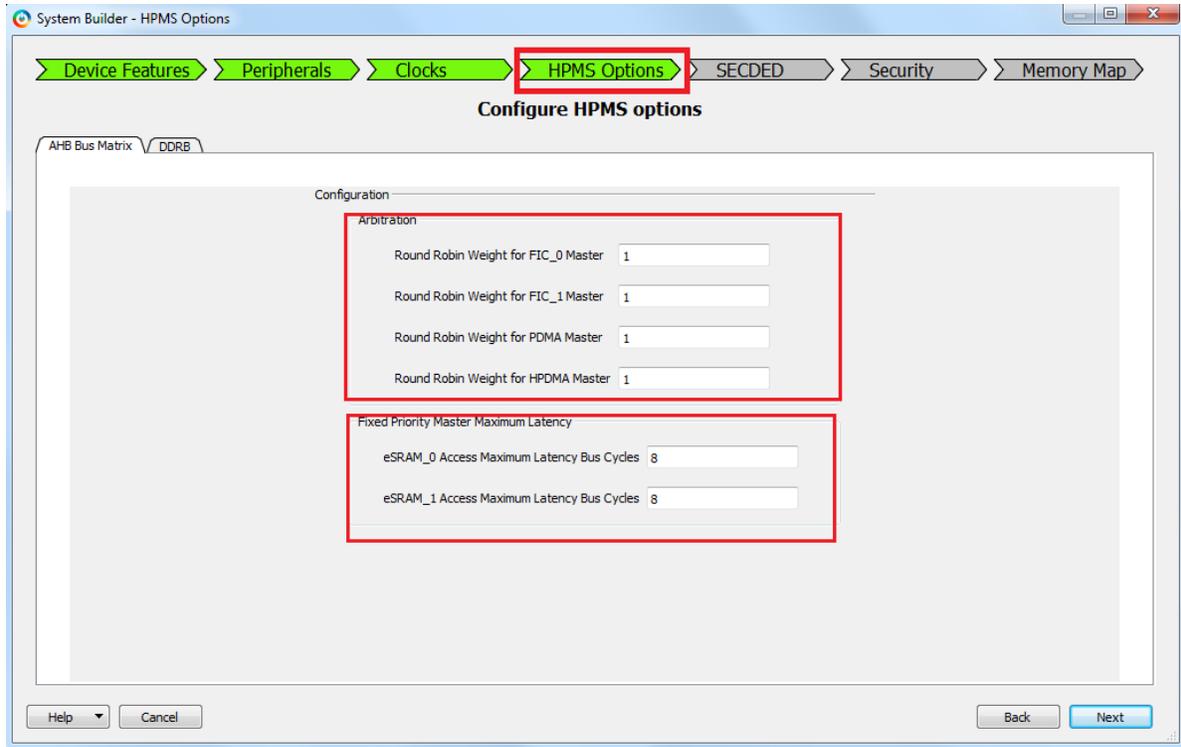
An access attempt by a master where the corresponding master port is blocked (by a flash configuration bit setting) causes the AHB bus matrix to assert HRESP to the master and terminate the transaction. If a blocked port is attempting a read access, the read data is returned as garbage. If the blocked port is attempting a write, the write of data does not occur to any slave. In both cases, one of the SW\_ERRORSTATUS bits is asserted. DDR\_FIC is not part of the AHB bus matrix but can be blocked from accessing the memory subsystem DDR (MDDR).



The following steps describe how to use the AHB bus matrix in the user application.

1. Use the **Device Features** tab and **Peripherals** tab of the **System Builder** wizard to configure the AHB bus matrix masters in the application.
2. Navigate to the **HPMS Options** tab. The following figure shows the **HPMS Options** tab.

**Figure 54 • System Builder - HPMS Options Tab**



3. Enter the programmable weight values for the FIC\_0, FIC\_1, PDMA, and HPDMA masters in **Arbitration** as shown in the preceding figure to configure the programmable weight registers, MASTER\_WEIGHT0\_CR and MASTER\_WEIGHT1\_CR with the required weight values. The weight values range is from 1 to 32.

**Note:** There is no option available in **System Builder - HPMS Options** tab to configure weight values for System Controller master. The weight values of System Controller master is 1 by default.

4. Enter the maximum latency values for the fixed priority masters to configure ESRAM\_MAX\_LAT registers that are located in the SYSREG block. Maximum latency values decide the peak wait time for a fixed priority master arbitrating the eSRAM access when the WRR master is accessing the slave. **Fixed Priority Master Maximum Latency** can be configured from 1 to 8 clock cycles. 8 clock cycles is the default value.

**Note:** **Fixed Priority Master Maximum Latency** is only supported for fixed priority master accessing the eSRAM slaves. It has no effect on WRR masters.

The HPMS AHB Bus Matrix supports full-behavioral simulation models.

## 4.3 Register Map

The following table lists the AHB bus matrix Control Registers in the SYSREG block.

**Table 49 • AHB Bus Matrix Register Map**

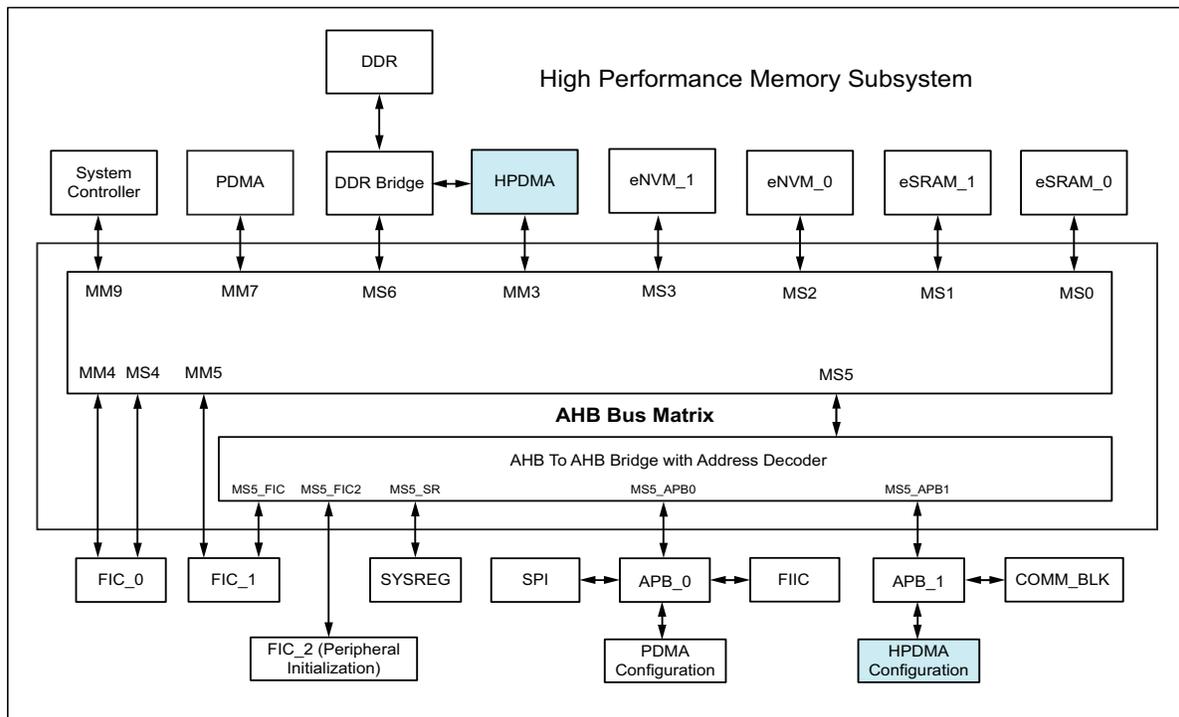
Register Name	Register Type	Flash Write Protect	Reset Source	Description
MASTER_WEIGHT0_CR	RW-P	Register	SYSRESET_N	Configures WRR master arbitration scheme for masters. For more information, see <a href="#">Table 139</a> , page 212.
MASTER_WEIGHT1_CR	RW-P	Register	SYSRESET_N	Configures WRR master arbitration scheme for masters. For more information, see <a href="#">Table 140</a> , page 213.
MM4_5_DDR_FIC_SECURITY/MM4_5_FIC64_SECURITY	RO-U	N/A	SYSRESET_N	Security bits for masters 4, 5, and DDR_FIC. For more information, see <a href="#">Table 167</a> , page 227.
MM3_7_SECURITY	RO-U	N/A	SYSRESET_N	Security bits for masters 3 and 7. For more information, see <a href="#">Table 168</a> , page 228.
MM9_SECURITY	RO-U	N/A	SYSRESET_N	Security bits for master 9. For more information, see <a href="#">Table 169</a> , page 229.
HPMS_EXTERNAL_SR	SW1C	N/A	SYSRESET_N	AHB bus matrix error status. Writing a 1 clears the status. For more information, see <a href="#">Table 184</a> , page 236.
ENVM_CR	RW-P	Register	SYSRESET_N	Configures eNVM parameters. For more information, see <a href="#">Table 130</a> , page 207.
ESRAM_MAX_LAT	RW-P	Register	SYSRESET_N	Configures maximum latency for accessing eSRAM0/1 slave. For more information, see <a href="#">Table 128</a> , page 206.
ENVM_REMAP_FAB_CR	RW-P	Register	SYSRESET_N	Configures where eNVM is mapped in fabric master space. For more information, see <a href="#">Table 132</a> , page 209.
DDRB_NB_ADDR_CR	RW-P	Register	SYSRESET_N	Base address of the non-bufferable address region. For more information, see <a href="#">Table 134</a> , page 210.
DDRB_NB_SIZE_CR	RW-P	Register	SYSRESET_N	Size of the non-bufferable address region. For more information, see <a href="#">Table 135</a> , page 210.

## 5 High Performance DMA Controller

The high performance DMA Controller (HPDMA) provides fast data transfer between the HPMS DDR bridge and HPMS memories. The HPMS memories are eSRAM0, eSRAM1, eNVM0, and eNVM1. The DDR bridge connects to external DDR memory.

The following figure shows HPDMA interfacing with AHB Bus Matrix and HPMS DDR bridge. AHB bus masters can offload the high speed memory transfers to HPDMA, making the master available for performing other tasks. All transfers by the HPDMA are full word transfers. The HPDMA controller has two AHB masters, HPMS DDR Bridge and AHB bus matrix master (MM0-MM9) which functions concurrently to enable high performance data transfers. The configuration of HPDMA is done through the APB interface.

**Figure 55 • HPDMA Interfacing With HPMSDDR Bridge and AHB Bus Matrix**



### 5.1 Features

- Faster read/write operations with two concurrent AHB masters
- 32-bit AHB operation at 166 MHz
- 32-bit APB slave interface for Control and Status Registers at 25/50/100/166 MHz
- Internal 32-bit Control, Status, and Debug Registers
- Single DMA channel with four queuing HPDMA descriptors, serviced with round robin priority
- Up to 64 Kbytes data transfer in single channel request
- 32-byte internal data buffer
- Supports word aligned data transfers
- Interrupts for DMA transfer complete and transfer errors
- DMA transfer pause
- Individual descriptor reset
- Data transfer in little-endian format

## 5.2 Functional Description

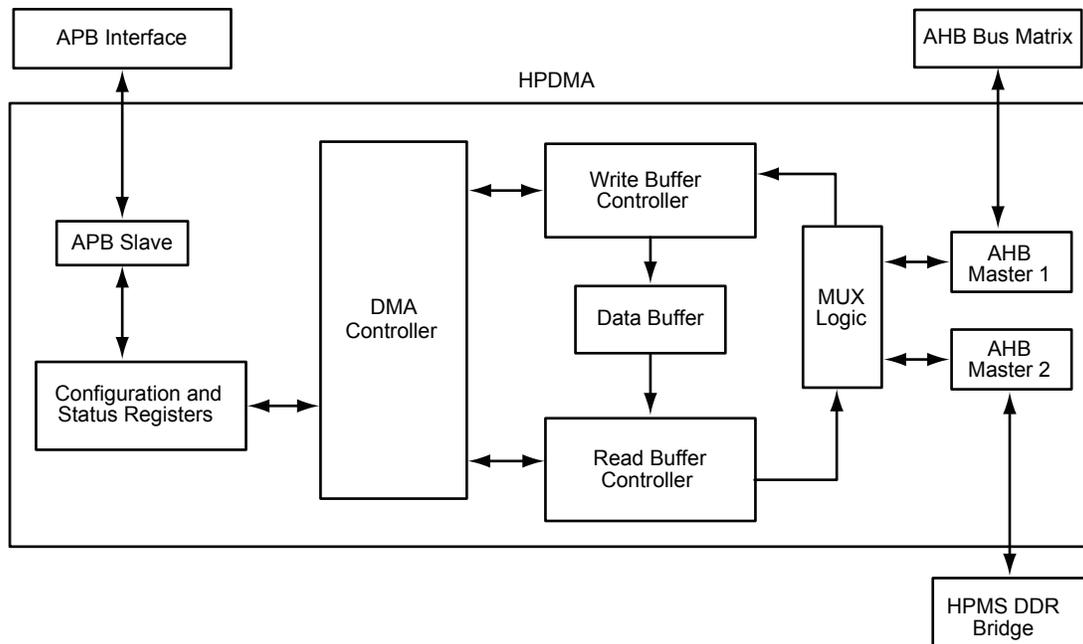
HPDMA has a single channel which can process up to four service requests (HPDMA descriptor) in a round robin fashion. To process each request, HPDMA descriptor is configured by an AHB bus matrix master (fabric master) through APB interface. The HPDMA APB interface is connected on APB\_1, which is an AHB to APB bridge as shown in the preceding figure. HPDMA then reads data from the source memory and transfers data to the destination.

### 5.2.1 Architecture Overview

HPDMA mainly consists of the following sub-blocks, as shown in the following figure:

- Interfaces
- Configuration and status registers
- DMA controller
- Write buffer controller
- Read buffer controller
- Data buffer

**Figure 56 • HPDMA Controller Block Diagram**



#### 5.2.1.1 Interfaces

There are two types of interfaces used for communicating with HPDMA.

- 32-bit APB slave interface for configuration
- Two AHB master interfaces (AHB-M1, AHB-M2) for data transfers
  - AHB master 1 does the read/write transfers at the AHB bus matrix end
  - AHB master 2 does the read/write transfers at the HPMS DDR bridge end

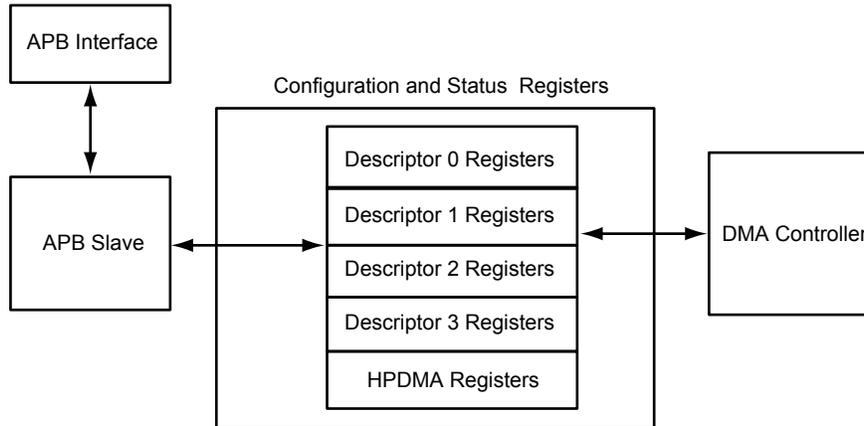
#### 5.2.1.2 Configuration and Status Registers

The Configuration and Status Registers of the HPDMA controller are accessed through a 32-bit APB slave, as shown in the following figure. In order to enable and use HPDMA services, the AHB bus matrix master must configure the 32-bit wide descriptor registers. There are four descriptors available with the HPDMA controller. Each descriptor has the following five registers.

- Source memory address register
- Destination memory address register
- Control Register

- Status Register
- Pending Transfer register

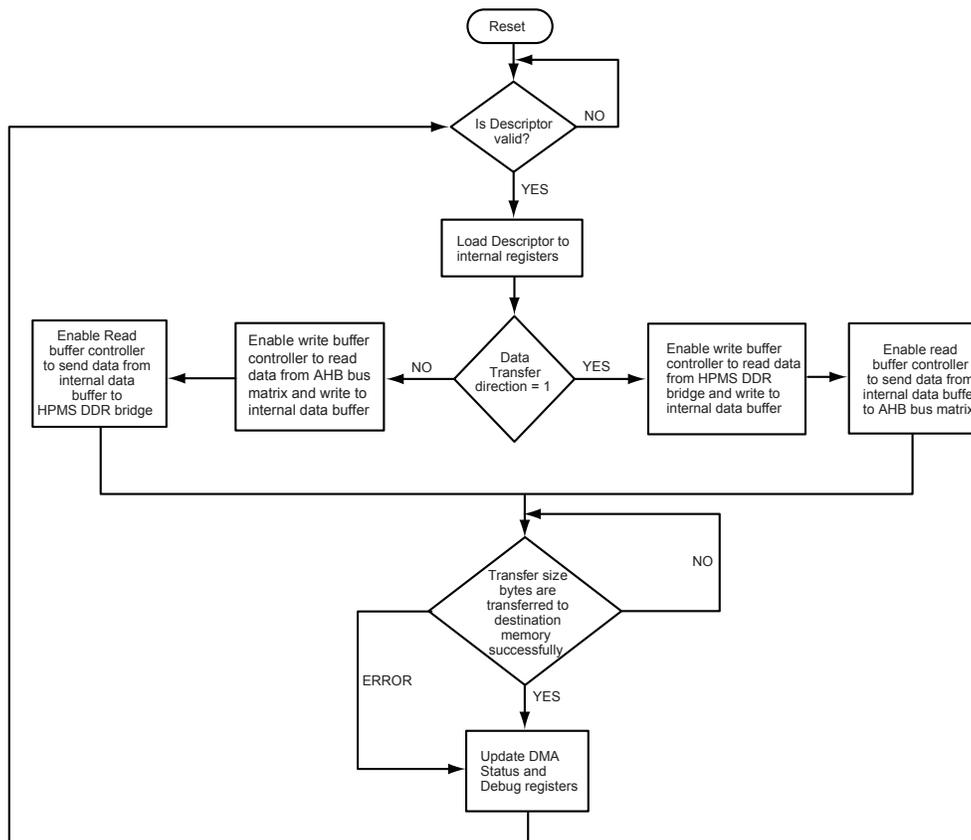
**Figure 57 • HPDMA Registers**



### 5.2.1.3 DMA Controller

The DMA controller controls and monitors transactions on the source and destination AHB master interfaces. When a descriptor is configured, the DMA controller enables the write buffer controller to read data from the appropriate source memory (AHB bus matrix or HPMS DDR bridge) and transfer it into the internal data buffer. In a similar way, the DMA controller enables read buffer controller to read the data from the internal data buffer and transfers it to the destination memory. The following figure shows the detailed DMA Controller flow.

**Figure 58 • DMA Controller Flow Chart**



### 5.2.1.4 Write Buffer Controller

The write buffer controller enables the appropriate AHB master (AHB-M1 or AHB-M2) to read the data from source memory. To initiate read transfers on the AHB bus, the write buffer controller provides the read address and asserts the ready signal. The AHB master acknowledges, and the write buffer controller writes the source memory data to the internal data buffer.

If the data buffer is full, the write buffer controller initiates idle transfers on the AHB bus, and asserts ready signal when at least one data buffer is available. The write buffer controller pauses the DMA transfers when the descriptor pause bit is enabled, and resumes the transfers as soon as the pause bit is disabled. When the last count value is reached, the AHB slave acknowledges the last transfer.

### 5.2.1.5 Read Buffer Controller

The read buffer controller places the address and asserts the ready signal to the AHB master (AHB-M1 or AHB-M2). Depending on the transfer direction, AHB-M1 or AHB-M2 initiates the data transfers from internal data buffer to destination memory.

If the data buffer is empty or if the DMA controller pause bit is enabled, then the read buffer controller initiates IDLE transfers on the AHB bus.

### 5.2.1.6 Data Buffer

The data buffer block is 32 bits wide and 8 words deep. Data buffer read/write operations are performed on the rising edge of the clock signal. There are 4-bit read and write pointers that increment on read and write.

The 3 least significant bits (LSBs) are used to address the 8 locations; the most significant bit (MSB) of the read and write pointers is used to signal the data buffer empty and full.

#### 5.2.1.6.1 Data Buffer Full and Empty

When the read pointer and write pointer are equal, the data buffer is empty. When the 3 LSBs of read pointer and write pointer are equal and the MSBs of the read pointer and write pointer are not equal, the data buffer is full.

## 5.2.2 Initialization

To initiate and setup DMA transactions, HPDMA has to be initialized. The initialization process starts with a reset sequence followed by Channel configuration and interrupt configuration.

### 5.2.2.1 Reset

The HPDMA registers are reset on power-up. The HPDMA can be reset by asserting the Bit 17 of SOFT\_RESET\_CR system register.

### 5.2.2.2 Descriptor Configuration

Before configuring each HPDMA channel, the round robin weight is specified if needed, using the MASTER\_WEIGHT\_CR register or configuring the AHB bus matrix in Libero SoC.

To configure each HPDMA descriptor, the following registers have to be set:

1. Descriptor Control Register bits:
  - Direction: bit 1 of HPDMADXCXCR\_REG (where X is 0 to 3)
  - Transfer size in bytes: bits[15:0] of HPDMADXCXCR\_REG (where X is 0 to 3)
  - Enable Interrupts: bits[22:20] of HPDMADXCXCR\_REG (where X is 0 to 3)
2. 32 bit Source memory start Address: bits[31:0] of HPDMADXSAR\_REG (where X is 0 to 3)
3. 32 bit Destination memory start Address: bits[31:0] of HPDMADXDAR\_REG (where X is 0 to 3)

### 5.2.2.3 Interrupt

There are two interrupts: HPD\_XFR\_CMP\_INT and HPD\_XFR\_ERR\_INT from the HPDMA to the Fabric master.

The interrupt signals are mapped to the dedicated interrupt signal HPMS\_INT\_M2F[9] and the HPMS\_INT\_M2F[22] of the fabric interface interrupt controller (FIIC).

This may be used to interrupt the user logic instantiated in the FPGA fabric. To enable HPDMA interrupts, the 9th bit (HPD\_XFR\_CMP\_INT\_EN) and the 22nd bit (HPD\_XFR\_CMP\_INT\_EN) of INTERRUPT\_ENABLE0 register (located at address 0x40000) need to be set to High. The status of the interrupts to FIIC can be determined by reading the 9th and 22nd bits of the INTERRUPT\_REASON0 register (located at 0x40008).

To determine the descriptor transfer status, monitor the Descriptor Status Register (HPDMADXSR, where X is 0 to 3). Before start of transaction, the enabled Descriptor interrupt bits are to be cleared. See [Table 72](#), page 104 for clearing of interrupts.

### 5.2.3 Details of Operation

After initialization, the HPDMA is ready to function in one of the two following data transfer modes:

- AHB bus matrix to HPMS DDR bridge
- HPMS DDR bridge to AHB bus matrix

For initiation of the above data transfer modes, a descriptor valid bit has to be set (that is, bit 16 of the Descriptor Control Register is asserted). If all the four descriptors are configured and set to valid, the descriptor transfer begins and executes in a round robin fashion. If any of the descriptors is paused by setting the bit 19 of Descriptor Control Register, the HPDMA stops the data transfer. HPDMA resumes the operation once the pause bit is reset. The pending transfers of the source and destination can be read from the Descriptor pending transfer register (HPDMADXPTR, where X is 0 to 3).

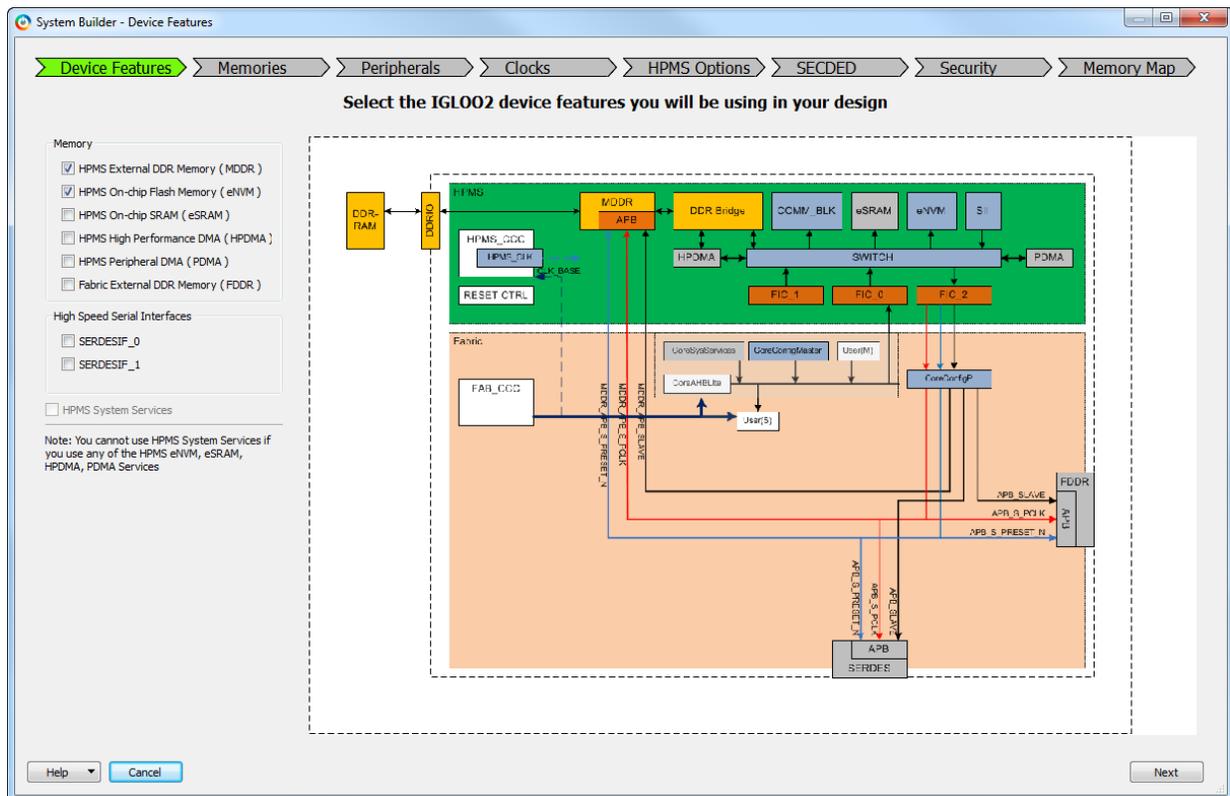
HPDMA can service the next descriptor only after the pending transfer of the current descriptor is complete. The data transfer completion interrupt is monitored using bit 20 of the Descriptor Control Register and bit 1 of the Descriptor Status Register. See [HPDMA Register Bit Definitions](#), page 90 for more information on HPDMA registers.

## 5.3 How to Use HPDMA

This section describes how to use HPDMA 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 figure shows the initial **System Builder** window where the required device features can be selected. For details on how to launch the **System Builder** wizard and a detailed information on how to use it, see *IGLOO2 System Builder User Guide*.

**Figure 59 • System Builder Window**

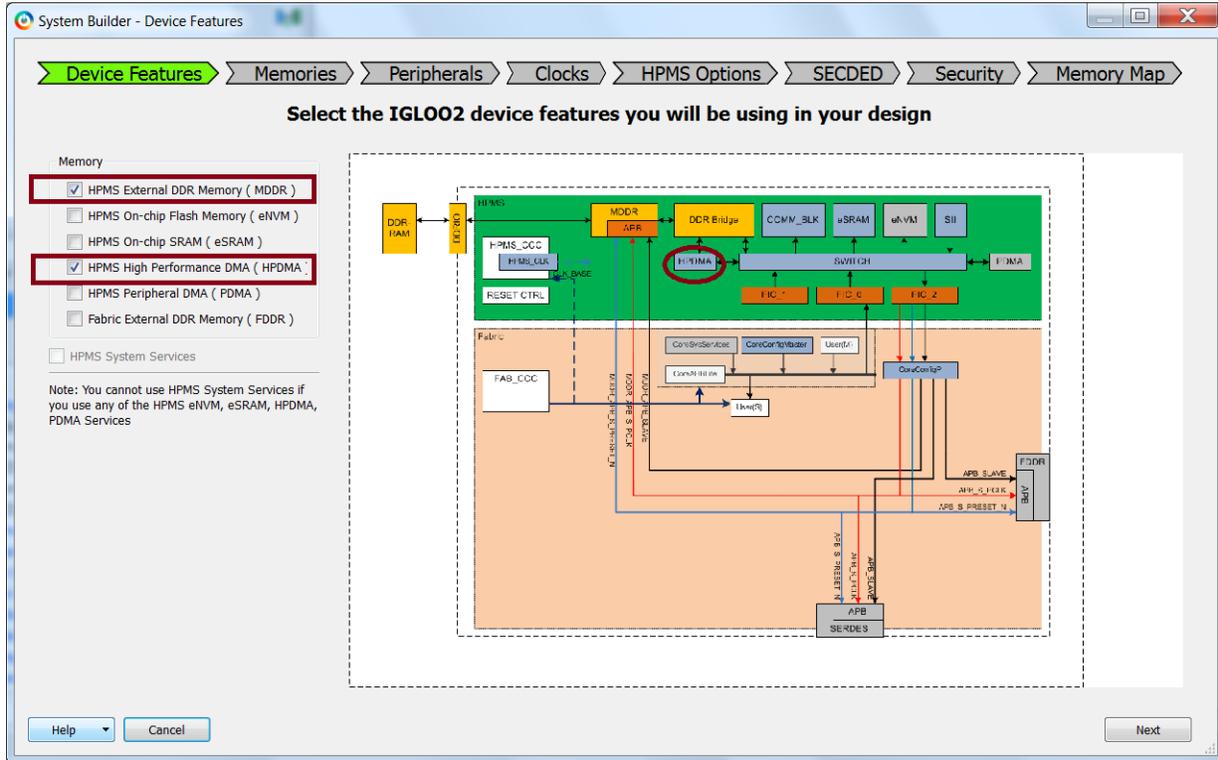


### 5.3.1 Configuring HPDMA

The following steps describe how to configure HPDMA.

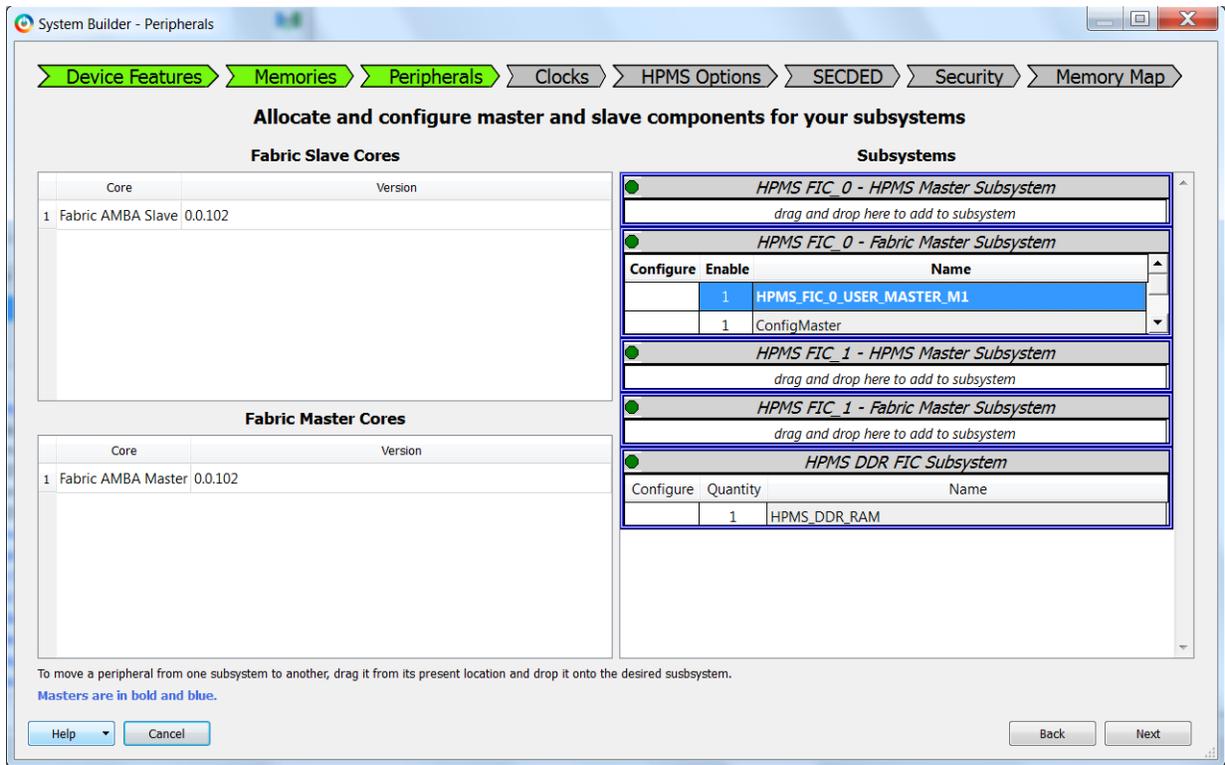
1. Check the **HPMS External DDR Memory (MDDR)**, and **HPMS High Performance DMA (HPDMA)** check boxes in the **Device Features** tab and leave the other check boxes unchecked. These other features can be used, but are outside the scope of this example. The following figure shows the **System Builder - Device Features** tab with the **HPDMA** and **MDDR** check boxes and the **HPDMA** block highlighted.

Figure 60 • System Builder - Device Features Tab



2. Navigate to the **Peripherals** tab. The following figure shows the **System Builder – Peripherals** tab. In the **Peripherals** tab, the fabric master core and slave components to HPMS\_FIC\_0 and HPMS\_FIC\_1 are added automatically by Libero SoC depending on selection on HPMS masters.

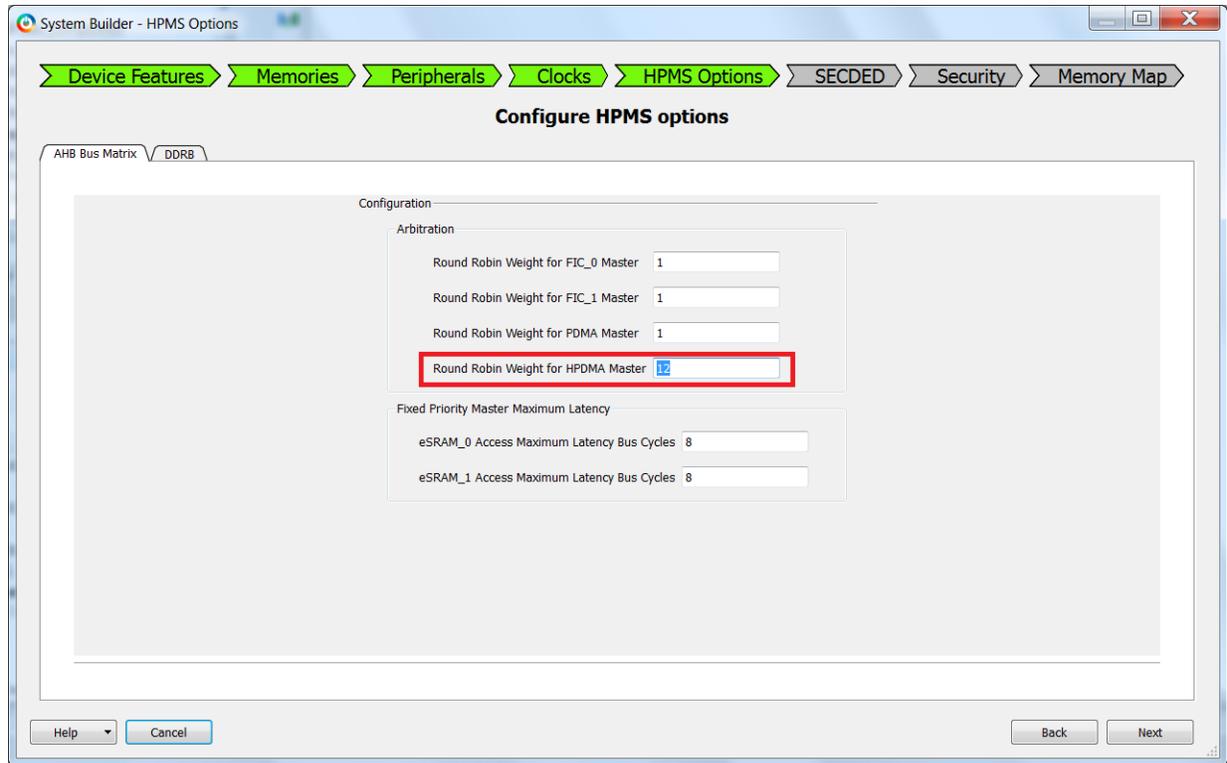
**Figure 61 • System Builder - Peripherals Tab**



3. Navigate to the **HPMS Options** in the **System Builder** wizard.

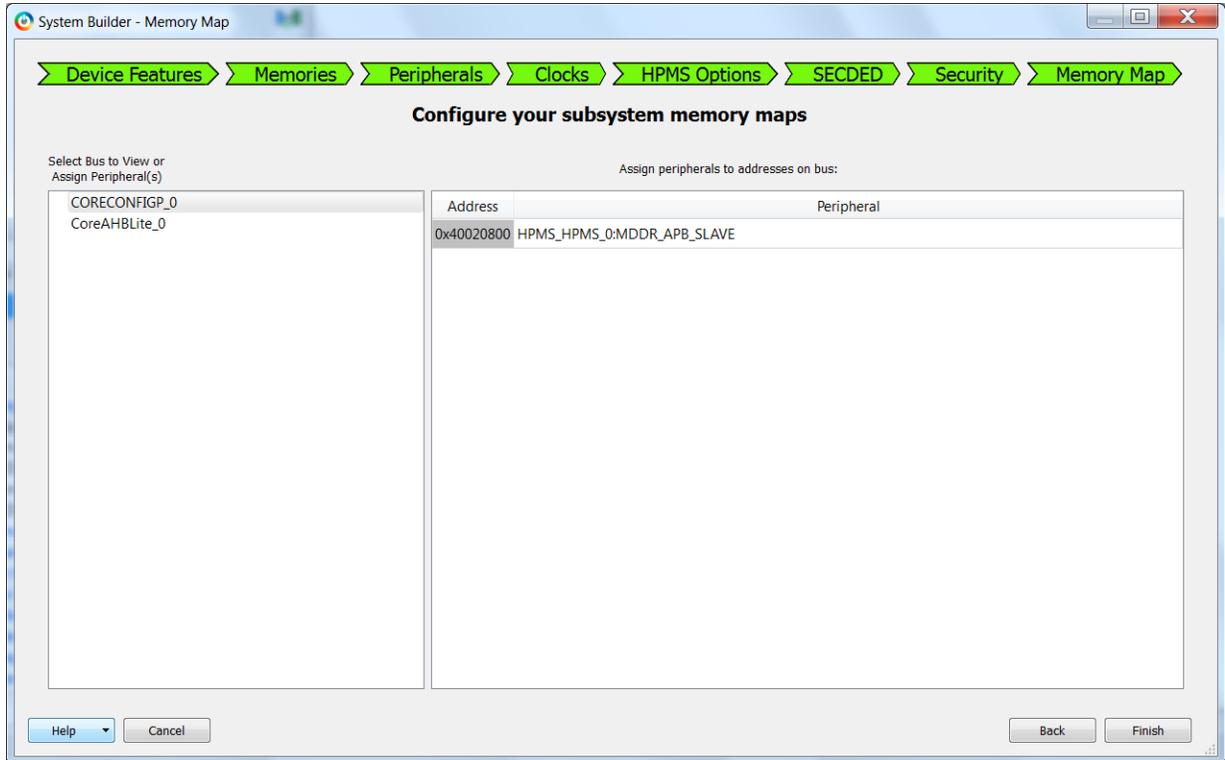
4. In the **HPMS Options** tab, configure the **Round Robin Weight for HPDMA Master**. The round robin weight is the number of consecutive transfers a master performs without being interrupted during an access. In the following figure, the configured round robin weight value for HPDMA is 12. It means that the HPDMA performs 12 consecutive transfers during its access before the next master takes control. The following figure shows the **Configuration** dialog in the **HPMS Options** tab.

**Figure 62 • HPMS Options Tab - Round Robin Weight Configuration for HPDMA Master**



5. Navigate to the **Memory Map** tab giving the required data in the rest of the **System Builder** tabs. The following figure shows the **System Builder - Memory Map** tab. The CoreCONFIGP\_0 is shown in memory map because external DDR memory is selected. The CoreCONFIGP\_0 facilitates in configuring the IGLOO2 DDR controller. Click **Finish** to proceed with creating the HPMS subsystem.

**Figure 63 • System Builder - Memory Map Tab**

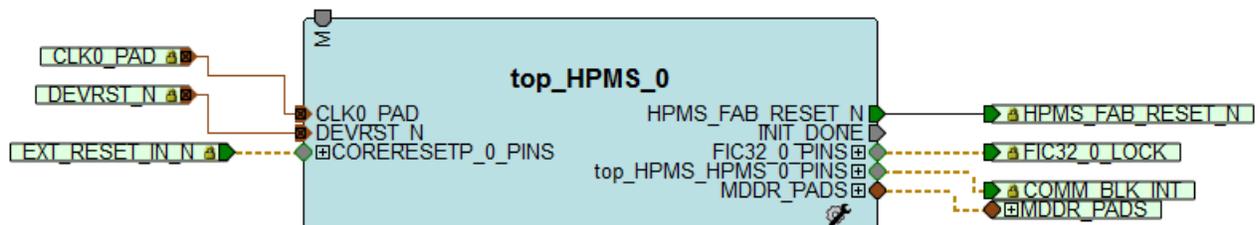


The IGLOO2 soft memory controller fabric interface controller (SMC\_FIC) helps access external bulk memories other than DDR through the FPGA fabric. The SMC\_FIC is used in conjunction with a soft memory controller to enable the HPMS to access memories such as SDRAM, flash, and SRAM. The HPMS masters communicate with the SMC\_FIC through an HPMS DDR bridge present in the HPMS. For more information about SMC\_FIC, see *UG0446: SmartFusion2 and IGLOO2 FPGA High Speed DDR Interfaces User Guide*.

### 5.3.2 HPMS Subsystem

The following figure shows an example HPMS subsystem that can be used to access HPDMA using the FPGA fabric master.

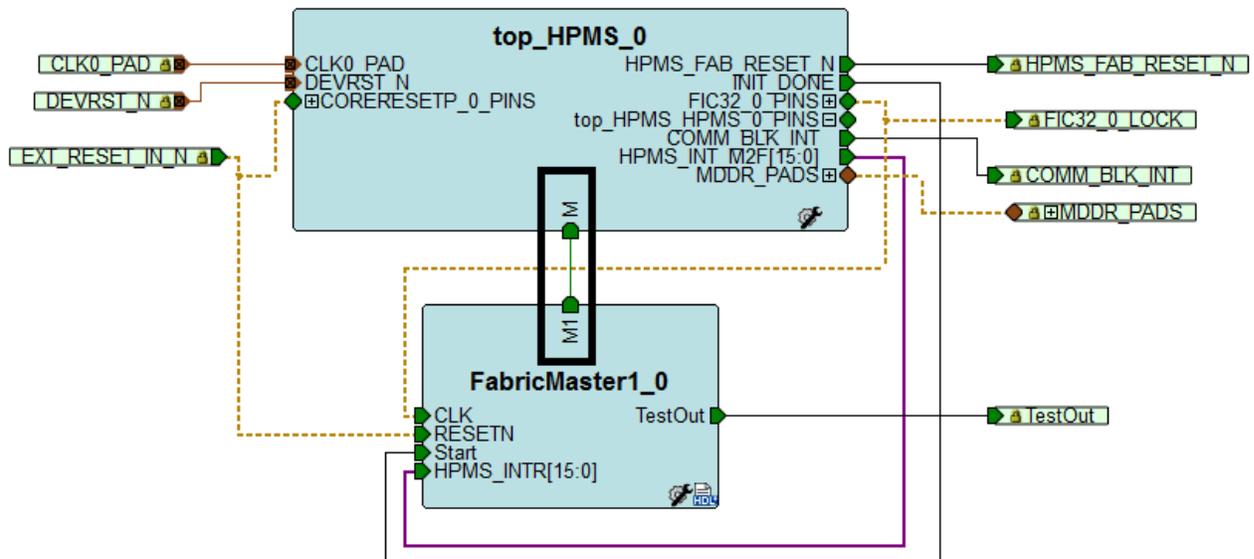
**Figure 64 • HPMS Subsystem**



### 5.3.3 HPMS Subsystem Connected to the FPGA Fabric Master

The following figure shows the FPGA fabric master connected to the AHB master port.

Figure 65 • HPMS Interconnection with FPGA Fabric Master



### 5.3.4 MDDR to eSRAM

The following steps describe how to use the Fabric master for configuring HPDMA to transfer data from MDDR to eSRAM.

1. Enable HPDMA using the **System Builder** wizard.
2. Initialize HPDMA:
  - Reset HPDMAICR\_REG register
  - Reset HPDMAEDR\_REG register
3. Configure HPDMA:
  - Load the 32-bit source memory start address register with MDDR read address. (See HPDMAD0SAR\_REG register).
  - Load the 32-bit destination memory start address register with eSRAM write address (See HPDMAD0DAR\_REG register).
  - Set the following fields in HPDMA Control Register:
    - Transfer size in bytes (Example: HPDMAD0CR[15:0] = 0x64 for 100 bytes)
    - Set the direction bit HPDMAD0CR[1] = 1 to transfer data from MDDR to eSRAM
    - For eSRAM to MDDR, set HPDMAD0CR[1] = 0
    - Enable the HPDMAD0CR[22: 20] to generate transfer complete and transfer error, non-word Align Interrupts
4. Start HPDMA by setting bit[16] of HPDMA Control Register (Example: HPDMAD0CR[16] = 1)
5. Pause the data transfers using bit[19] of HPDMA Control Register (Example: HPDMAD0CR[19] = 1)
6. Check the source and destination pending transfer bytes using HPDMA pending transfer register (See HPDMAD0PTR\_REG register).
7. Resume the data transfers using bit[19] of HPDMA Control Register (Example: HPDMAD0CR[19] = 0).
8. Check for interrupt completion by monitoring HPMS\_INT\_M2F[9] bit.
9. Select Group0 interrupt HPMS\_INT\_M2F[8] to monitor transfer error interrupt.
10. Read the status of the configured descriptor (see HPDMAD0SR\_REG register) to check the following:
  - If the current descriptor is active
  - If there are any pending interrupts, clear the interrupts by configuring the HPDMAICR\_REG register (Interrupt clear register).

## 5.4 HPDMA Controller Register Map

The following table summarizes the HPDMA controller register map. The sections that follow detail register bit descriptions of status, configuration, and debug registers. All the register bits are active high; on reset they assume default values. Register R/W corresponds to external processor accessibility. The address range of the HPDMA APB registers is x40014000 to x40014FFF. Only the 7 LSBs are considered for addressing the registers.

**Table 50 • HPDMA Register Map**

Register Name	Address Offset	Register Type	Reset Value	Description
HPDMAEDR_REG	x00	R	x0F	HPDMA Empty Descriptor register
HPDMAD0SAR_REG	x04	R/W	x00	Descriptor 0 source memory start address
HPDMAD0DAR_REG	x08	R/W	x00	Descriptor 0 destination memory start address
HPDMAD0CR_REG	x0C	R/W	x00	Descriptor 0 Control Register
HPDMAD0SR_REG	x10	R	x00	Descriptor 0 Status Register
HPDMAD0PTR_REG	x14	R	x00	Descriptor 0 Pending Transfer register
HPDMAD1SAR_REG	x18	R/W	x00	Descriptor 1 source memory start address
HPDMAD1DAR_REG	x1C	R/W	x00	Descriptor 1 destination memory start address.
HPDMAD1CR_REG	x20	R/W	x00	Descriptor 1 Control Register
HPDMAD1SR_REG	x24	R	x00	Descriptor 1 Status Register
HPDMAD1PTR_REG	x28	R	x00	Descriptor 1 Pending Transfer register
HPDMAD2SAR_REG	x2C	R/W	x00	Descriptor 2 source memory start address
HPDMAD2DAR_REG	x30	R/W	x00	Descriptor 2 destination memory start address
HPDMAD2CR_REG	x34	R/W	x00	Descriptor 2 Control Register
HPDMAD2SR_REG	x38	R	x00	Descriptor 2 Status Register
HPDMAD2PTR_REG	x3C	R	x00	Descriptor 2 Pending Transfer register
HPDMAD3SAR_REG	x40	R/W	x00	Descriptor 3 source memory start address
HPDMAD3DAR_REG	x44	R/W	x00	Descriptor 3 destination memory start address
HPDMAD3CR_REG	x48	R/W	x00	Descriptor 3 Control Register
HPDMAD3SR_REG	x4C	R	x00	Descriptor 3 Status Register
HPDMAD3PTR_REG	x50	R	x00	Descriptor 3 Pending Transfer register
HPDMAICR_REG	x54	W	x00	HPDMA Interrupt Clear register
HPDMADR_REG	x58	R	x01	HPDMA Debug register

## 5.4.1 HPDMA Register Bit Definitions

### 5.4.1.1 HPDMA Empty Descriptor Register

**Table 51 • HPDMAEDR\_REG**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	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	HPDMAEDR_DCP_NON_WORD_ERR[3]	0	Descriptor 3 non-word aligned transfer size error. 1: Descriptor 3 non-word aligned transfer size error 0: No non-word aligned transfer size error This bit is asserted High, if a non-word aligned value is configured in the descriptor 3 transfer size field. This bit clears on writing '1' to HPDMAICR_NON_WORD_INT[3] of the HPDMA Interrupt Clear register, or when the HPDMACR_DCP_VALID[3] bit of the descriptor 3 Control Register is set, or when the HPDMACR_DCP_CLR[3] bit of the HPDMA Controller register is set. In this case, HPDMA will continue the transfer by ignoring the 2 LSBs of transfer size field.
14	HPDMAEDR_DCP_NON_WORD_ERR[2]	0	Descriptor 2 non-word aligned transfer size error. 1: Descriptor 2 non-word aligned transfer size error 0: No non-word aligned transfer size error This bit is asserted High if a non-word aligned value is configured in the descriptor 2 transfer size field. This bit is cleared on writing '1' to HPDMAICR_NON_WORD_INT[2] of the HPDMA Interrupt Clear register, or when the HPDMACR_DCP_VALID[2] bit of the descriptor 2 Control Register is set, or when the HPDMACR_DCP_CLR[2] bit of the HPDMA Controller register is set. In this case, HPDMA will continue the transfer by ignoring the 2 LSBs of the transfer size field.
13	HPDMAEDR_DCP_NON_WORD_ERR[1]	0	Descriptor 1 non-word aligned transfer size error. 1: Descriptor 1 non-word aligned transfer size error 0: No non-word aligned transfer size error This bit is asserted High if a non-word aligned value is configured in the descriptor 1 transfer size field. This bit is cleared on writing '1' to HPDMAICR_NON_WORD_INT[1] of the HPDMA Interrupt Clear register, or when the HPDMACR_DCP_VALID[1] bit of the descriptor 1 Control Register is set, or when the HPDMACR_DCP_CLR[1] bit of the HPDMA Controller register is set. In this case, HPDMA will continue the transfer by ignoring the 2 LSBs of the transfer size filed.

**Table 51 • HPDMAEDR\_REG (continued)**

Bit Number	Name	Reset Value	Description
12	HPDMAEDR_DCP_NON_WORD_ERR[0]	0	<p>Descriptor 0 non-word aligned transfer size error.</p> <p>1: Descriptor 0 non-word aligned transfer size error 0: No non-word aligned transfer size error</p> <p>This bit is asserted High, if non-word aligned value is configured in descriptor 0 transfer size field. This bit is cleared on writing '1' to HPDMAICR_NON_WORD_INT[0] of the HPDMA Interrupt Clear register, or when the HPDMACR_DCP_VALID[0] bit of the descriptor 0 Control Register is set or when the HPDMACR_DCP_CLR[0] bit of the HPDMA Controller register is set.</p> <p>In this case, HPDMA will continue the transfer by ignoring the 2 LSBs of the transfer size field.</p>
11	HPDMAEDR_DCP_ERR[3]	0	<p>Descriptor 3 transfer error.</p> <p>1: Descriptor 3 transfer error 0: No descriptor 3 transfer error</p> <p>This bit is asserted High, if an error occurs during the descriptor 3 transfer at either source or destination end. This bit is cleared on writing '1' to HPDMAICR_CLR_XFR_INT[3] of the HPDMA Interrupt Clear register, or when the HPDMACR_DCP_VALID[3] bit of the descriptor 3 Control Register is set.</p>
10	HPDMAEDR_DCP_ERR[2]	0	<p>Descriptor 2 transfer error.</p> <p>1: Descriptor 2 transfer error 0: No descriptor 2 transfer error</p> <p>This bit is asserted High, if an error occurs during the descriptor 2 transfer at either source or destination end. This bit is cleared on writing '1' to the HPDMAICR_CLR_XFR_INT[2] bit of the HPDMA Interrupt Clear register, or when the HPDMACR_DCP_VALID[2] bit of the descriptor 2 Control Register is set.</p>
9	HPDMAEDR_DCP_ERR[1]	0	<p>Descriptor 1 transfer error.</p> <p>1: Descriptor 1 transfer error 0: No descriptor 1 transfer error</p> <p>This bit is asserted High, if an error occurs during the descriptor 1 transfer at either source or destination end. This bit is cleared on writing '1' to HPDMAICR_CLR_XFR_INT[1] of the HPDMA Interrupt Clear register, or when the HPDMACR_DCP_VALID[1] bit of Descriptor 1 Control Register is set.</p>

**Table 51 • HPDMAEDR\_REG (continued)**

Bit Number	Name	Reset Value	Description
8	HPDMAEDR_DCP_ERR[0]	0	Descriptor 0 transfer error. 1: Descriptor 0 transfer error 0: No descriptor 0 transfer error This bit is asserted High if an error occurs during the descriptor 0 transfer at either source or destination end. This bit is cleared on writing '1' to the HPDMAICR_CLR_XFR_INT[0] bit of the HPDMA Interrupt Clear register or when the HPDMACR_DCP_VALID[0] bit of the descriptor 0 Control Register is set.
7	HPDMAEDR_DCP_CMPLT[3]	0	Descriptor 3 transfer complete. 1: Descriptor 3 transfer completed successfully 0: Descriptor 3 transfer not completed When the descriptor 3 transfer is completed, either with transfer error or transfer done, the HPDMA controller asserts this bit High. This bit is cleared on writing '1' to the HPDMAICR_CLR_XFR_INT[3] bit of the HPDMA Interrupt Clear Register or when the HPDMACR_DCP_VALID[3] bit of the descriptor 3 Control Register is set.
6	HPDMAEDR_DCP_CMPLT[2]	0	Descriptor 2 transfer complete. 1: Descriptor 2 transfer completed successfully 0: Descriptor 2 transfer not completed When the descriptor 2 transfer is completed, either with transfer error or transfer done, the HPDMA controller asserts this bit High. This bit is cleared on writing '1' to the HPDMAICR_CLR_XFR_INT[2] bit of the HPDMA Interrupt Clear register or when the HPDMACR_DCP_VALID[2] bit of the descriptor 2 Control Register is set.
5	HPDMAEDR_DCP_CMPLT[1]	0	Descriptor 1 transfer complete. 1: Descriptor 1 transfer completed successfully. 0: Descriptor 1 transfer not completed. When the descriptor 1 transfer is completed, either with transfer error or transfer done, the HPDMA controller asserts this bit High. Cleared on writing '1' to the HPDMAICR_CLR_XFR_INT[1] bit of the HPDMA Interrupt Clear register or when the HPDMACR_DCP_VALID[1] bit of the descriptor 1 Control Register is set.

**Table 51 • HPDMAEDR\_REG (continued)**

Bit Number	Name	Reset Value	Description
4	HPDMAEDR_DCP_CMPLT[0]	0	Descriptor 0 transfer complete. 1: Descriptor 0 transfer completed successfully. 0: Descriptor 0 transfer not completed. When the descriptor 0 transfer is completed, either with transfer error or transfer done, HPDMA controller asserts this bit High. Cleared on writing '1' to the HPDMAICR_CLR_XFR_INT[0] bit of the HPDMA Interrupt Clear register or when the HPDMACR_DCP_VALID[0] bit of descriptor 0 Control Register is set.
3	HPDMAEDR_DCP_EMPTY[3]	1	Descriptor 3 is empty and ready for software configuration. 1: Descriptor 3 is empty and ready to configure. 0: Descriptor 3 is already configured and descriptor transfer is in progress/queue. At the end of the descriptor transfer, either on transfer error or transfer done, the HPDMA controller asserts this bit High.
2	HPDMAEDR_DCP_EMPTY[2]	1	Descriptor 2 is empty and ready for software configuration. 1: Descriptor 2 is empty and ready to configure. 0: Descriptor 2 is already configured and descriptor transfer is in progress/queue. At the end of the descriptor transfer, either on transfer error or transfer done, the HPDMA controller asserts this bit High.
1	HPDMAEDR_DCP_EMPTY[1]	1	Descriptor 1 is empty and ready for software configuration. 1: Descriptor 1 is empty and ready to configure. 0: Descriptor 1 is already configured and descriptor transfer is in progress/queue. At the end of the descriptor transfer, either on transfer error or transfer done, the HPDMA controller asserts this bit High.
0	HPDMAEDR_DCP_EMPTY[0]	1	Descriptor 0 is empty and ready for software configuration. 1: Descriptor 0 is empty and ready to configure. 0: Descriptor 0 is already configured and descriptor transfer is in progress/queue. At the end of the descriptor transfer, either on transfer error or transfer done, the HPDMA controller asserts this bit High.

### 5.4.1.2 Descriptor 0 Source Address Register

**Table 52 • HPDMAD0SAR\_REG**

Bit Number	Name	Reset Value	Description
[31:0]	HPDMASAR_DCP0_SRC_ADRS	0x00	Descriptor 0 source end memory start address

### 5.4.1.3 Descriptor 1 Source Address Register

**Table 53 • HPDMAD1SAR\_REG**

Bit Number	Name	Reset Value	Description
[31:0]	HPDMASAR_DCP1_SRC_ADRS	0x00	Descriptor 1 source end memory start address

### 5.4.1.4 Descriptor 2 Source Address Register

**Table 54 • HPDMAD2SAR\_REG**

Bit Number	Name	Reset Value	Description
[31:0]	HPDMASAR_DCP2_SRC_ADRS	0x00	Descriptor 2 source end memory start address

### 5.4.1.5 Descriptor 3 Source Address Register

**Table 55 • HPDMAD3SAR\_REG**

Bit Number	Name	Reset Value	Description
[31:0]	HPDMASAR_DCP3_SRC_ADRS	0x00	Descriptor 3 source end memory start address

#### 5.4.1.5.1 Notes on the Source Address Register (SAR)

- Address is word aligned at the start.
- Address increments on each successful transfer at the source end.
- HPDMA controller starts reading the data from source memory and transfers to destination memory.
- Software can write all 32-bit source address to prevent non-word aligned transfers at the start and 2 LSBs, 1:0, are masked in the hardware.
- The source address is updated when descriptor transfer is in progress.

### 5.4.1.6 Descriptor 0 Destination Address Register

**Table 56 • HPDMAD0DAR\_REG**

Bit Number	Name	Reset Value	Description
[31:0]	HPDMADAR_DCP0_DST_ADRS	0x00	Descriptor 0 destination end memory start address

### 5.4.1.7 Descriptor 1 Destination Address Register

**Table 57 • HPDMAD1DAR\_REG**

Bit Number	Name	Reset Value	Description
[31:0]	HPDMADAR_DCP1_DST_ADRS	0x00	Descriptor 1 destination end memory start address

### 5.4.1.8 Descriptor 2 Destination Address Register

**Table 58 • HPDMAD2DAR\_REG**

Bit Number	Name	Reset Value	Description
[31:0]	HPDMADAR_DCP2_DST_ADRS	0x00	Descriptor 2 destination end memory start address

### 5.4.1.9 Descriptor 3 Destination Address Register

**Table 59 • HPDMAD3DAR\_REG**

Bit Number	Name	Reset Value	Description
[31:0]	HPDMADAR_DCP3_DST_ADRS	0x00	Descriptor 3 destination end memory start address

#### 5.4.1.9.1 Notes on the Destination Address Register (DAR)

- Address is word aligned at the start.
- Address increments on each successful transfer at the destination end.
- HPDMA controller starts reading the data from source memory and transfers to destination memory.
- Software can write all 32-bit destination addresses to prevent non-word aligned transfers at the start and 2 LSBs, 1:0, are masked in the hardware.
- The destination address will be updated in the same field when the descriptor transfer is in progress.

### 5.4.1.10 Descriptor 0 Control Register

**Table 60 • HPDMAD0CR\_REG**

Bit Number	Name	Reset Value	Description
[31:23]	Reserved	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.
22	HPDMACR_NON_WORD_INT[0]	0	Non-word interrupt enable. 1: HPDMA asserts transfer error interrupt when non-word aligned transfer size is programmed in HPDMACR_DCP0_XFR_SIZE and HPDMA continues the same descriptor transfer. 0: HPDMA will not generate interrupt.
21	HPDMACR_XFR_ERR_INT[0]	0	1: HPDMA asserts transfer error interrupt on error during descriptor 0 transfers. 0: HPDMA will not generate transfer error interrupt.
20	HPDMACR_XFR_CMP_INT[0]	0	1: HPDMA asserts interrupt on completion of descriptor 0 transfers without error. 0: HPDMA will not generate transfer complete interrupt.
19	HPDMACR_DCP_PAUSE[0]	0	1: HPDMA pauses descriptor 0 transfers, does idle transfers. 0: HPDMA resumes descriptor 0 transfers from where they have stopped.
18	HPDMACR_DCP_CLR[0]	0	When this bit is set, HPDMA clears the descriptor 0 fields. HPDMA terminates the current transfer and reset descriptor status and Control Registers. This bit is always read back as zero.
17	HPDMACR_XFR_DIR[0]	0	Descriptor 0 data transfer direction. 0: AHB bus matrix to HPMS DDR bridge 1: HPMS DDR bridge to AHB bus matrix
16	HPDMACR_DCP_VALID[0]	0	1: Indicates the descriptor 0 is valid and ready to transfer. On completing descriptor 0 transfer, the HPDMA controller clears this bit. Once the descriptor valid bit is set, descriptor fields such as Source address, Destination Address, Transfer size, and Descriptor Valid bits cannot be overwritten. When this bit is set, HPDMA clears the status of the previous transfer, which includes transfer complete, transfer error interrupts, and corresponding descriptor 0 Status Register.
[15:0]	HPDMACR_DCP0_XFR_SIZE	0	Descriptor 0 transfer size in bytes. Defines number of bytes to be transferred in a descriptor 0 transfer. All zeros in this field indicates 64-Kbyte transfers. As all the transfers are word aligned, 2 LSBs 1:0 are ignored.

### 5.4.1.11 Descriptor 1 Control Register

**Table 61 • HPDMAD1CR\_REG**

Bit Number	Name	Reset Value	Description
[31:23]	Reserved	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.
22	HPDMACR_NON_WORD_INT[1]	0	Non-word interrupt enable 1: HPDMA asserts transfer error interrupt when non-word aligned transfer size is programmed in HPDMACR_DCP1_XFR_SIZE and HPDMA continues the same descriptor transfer. 0: HPDMA will not generate interrupt.
21	HPDMACR_XFR_ERR_INT[1]	0	1: HPDMA asserts transfer error interrupt on error during descriptor 1 transfers. 0: HPDMA will not generate transfer error interrupt.
20	HPDMACR_XFR_CMP_INT[1]	0	1: HPDMA asserts interrupt on completion of descriptor 1 transfers without error. 0: HPDMA will not generate transfer complete interrupt.
19	HPDMACR_DCP_PAUSE[1]	0	1: HPDMA pauses Descriptor 1 transfers, does idle transfers. 0: HPDMA resumes descriptor 1 transfers from where they have stopped.
18	HPDMACR_DCP_CLR[1]	0	When this bit is set, HPDMA clears the descriptor 1 fields. HPDMA terminates the current transfer and resets descriptor status and Control Registers. This bit is always read back as zero.
17	HPDMACR_XFR_DIR[1]	0	Descriptor 2 data transfer direction: 0: AHB bus matrix to HPMS DDR bridge 1: HPMS DDR bridge to AHB bus matrix
16	HPDMACR_DCP_VALID[1]	0	1: Indicates the descriptor 1 is valid and ready to transfer. On completing a descriptor 1 transfer, the HPDMA controller clears this bit. Once the descriptor valid bit is set, descriptor fields such as Source Address, Destination Address, Transfer Size, and Descriptor Valid bits cannot be overwritten. When this bit is set, HPDMA clears the status of the previous transfer, which includes transfer complete, transfer error interrupts, and corresponding descriptor 1 Status Register.
[15:0]	HPDMACR_DCP1_XFR_SIZE	0	Descriptor 1 transfer size in bytes. Defines number of bytes to be transferred in a descriptor 1 transfer. All zeroes in this field indicates 64-Kbyte transfers. As all the transfers are word aligned, the 2 LSBs 1:0 are ignored.

### 5.4.1.12 Descriptor 2 Control Register

**Table 62 • HPDMAD2CR\_REG**

Bit Number	Name	Reset Value	Description
[31:23]	Reserved	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.
22	HPDMACR_NON_WORD_INT[2]	0	Non-word interrupt enable. 1: HPDMA asserts transfer error interrupt when non-word aligned transfer size is programmed in HPDMACR_DCP2_XFR_SIZE and HPDMA continues the same descriptor transfer. 0: HPDMA will not generate interrupt.
21	HPDMACR_XFR_ERR_INT[2]	0	1: HPDMA asserts transfer error interrupt on error during descriptor 2 transfers. 0: HPDMA will not generate transfer error interrupt.
20	HPDMACR_XFR_CMP_INT[2]	0	1: HPDMA asserts interrupt on completion of descriptor 2 transfers without error. 0: HPDMA will not generate transfer complete interrupt.
19	HPDMACR_DCP_PAUSE[2]	0	1: HPDMA pauses the descriptor 2 transfers, does idle transfers. 0: HPDMA resumes descriptor 2 transfers from where they have stopped.
18	HPDMACR_DCP_CLR[2]	0	When this bit is set, HPDMA clears the descriptor 2 fields. HPDMA terminates the current transfer and reset descriptor status and Control Registers. This bit is always read back as zero.
17	HPDMACR_XFR_DIR[2]	0	Descriptor 2 data transfer direction. 0: AHB bus matrix to HPMS DDR bridge 1: DDR bridge to AHB bus matrix
16	HPDMACR_DCP_VALID[2]	0	1: Indicates the descriptor 2 is valid and ready to transfer. On completing descriptor 2 transfer, the HPDMA controller clears this bit. Once the descriptor valid bit is set, descriptor fields such as Source Address, Destination Address, Transfer Size, and Descriptor Valid bits cannot be overwritten. When this bit is set, HPDMA clears the status of the previous transfer, which includes transfer complete, transfer error interrupts, and corresponding descriptor 2 Status Register.
[15:0]	HPDMACR_DCP2_XFR_SIZE	0	Descriptor 2 transfer size in bytes. Defines number of bytes to be transferred in a Descriptor 2 transfer. All zeros in this field indicates 64 Kbytes transfers. As all the transfers are word aligned, 2 LSBs 1:0 are ignored.

### 5.4.1.13 Descriptor 3 Control Register

**Table 63 • HPDMAD3CR\_REG**

Bit Number	Name	Reset Value	Description
[31:23]	Reserved	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.
22	HPDMACR_NON_WORD_INT[3]	0	Non-word interrupt enable. 1: HPDMA asserts transfer error interrupt when non-word aligned transfer size is programmed in HPDMACR_DCP3_XFR_SIZE and HPDMA continues the same descriptor transfer. 0: HPDMA will not generate interrupt
21	HPDMACR_XFR_ERR_INT[3]	0	1: HPDMA asserts transfer error interrupt on error during descriptor 3 transfers. 0: HPDMA will not generate transfer error interrupt.
20	HPDMACR_XFR_CMP_INT[3]	0	1: HPDMA asserts interrupt on completion of descriptor 3 transfers without error. 0: HPDMA will not generate transfer complete interrupt.
19	HPDMACR_DCP_PAUSE[3]	0	1: HPDMA pauses the descriptor 3 transfers, does idle transfers. 0: HPDMA resumes the descriptor 3 transfers from where they have stopped.
18	HPDMACR_DCP_CLR[3]	0	When this bit is set, HPDMA clears the descriptor 3 fields. HPDMA terminates the current transfer and resets descriptor status and Control Registers. This bit is always read back as zero.
17	HPDMACR_XFR_DIR[3]	0	Descriptor 3 data transfer direction. 0: AHB bus matrix to HPMS DDR bridge 1: HPMS DDR bridge to AHB bus matrix
16	HPDMACR_DCP_VALID[3]	0	1: Indicates descriptor 3 is valid and ready to transfer. On completing descriptor 3 transfer, the HPDMA controller clears this bit. Once the descriptor valid bit is set, descriptor fields such as Source address, Destination Address, Transfer size, and Descriptor Valid bits cannot be overwritten. When this bit is set, HPDMA clears the status of the previous transfer, which includes transfer complete, transfer error interrupts, and corresponding descriptor 3 Status Register.
[15:0]	HPDMACR_DCP3_XFR_SIZE	0	Descriptor 3 transfer size in bytes. Defines number of bytes to be transferred in a descriptor 3 transfer. All zeroes in this field indicates 64-Kbytes transfers. As all the transfers are word aligned, 2 LSBs, 1:0, are ignored.

### 5.4.1.14 Descriptor 0 Status Register

**Table 64 • HPDMAD0SR\_REG**

Bit Number	Name	Reset Value	Description
[31:4]	Reserved	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	HPDMASR_DCP_DERR[0]	0	Descriptor 0 destination transfer error. 1: Descriptor 0 transfer error 0: No error at destination end during descriptor 0 transfer. This bit clears on writing '1' to HPDMAICR_CLR_XFR_INT[0] of the descriptor 0 Interrupt Clear register.
2	HPDMASR_DCP_SERR[0]	0	Descriptor 0 source transfer error. 1: Descriptor 0 transfer error occurred at source end. 0: No error at source end during descriptor 0 transfer This bit clears on writing '1' to HPDMAICR_CLR_XFR_INT[0] of the descriptor 0 Interrupt Clear register.
1	HPDMASR_DCP_CMPLT[0]	0	Descriptor 0 transfer complete. 1: Descriptor 0 transfer completed successfully 0: Descriptor 0 transfer not completed This bit clears on writing '1' to HPDMAICR_CLR_XFR_INT[0] of the descriptor 0 Control Register.
0	HPDMASR_DCP_ACTIVE[0]	0	Descriptor 0 transfer in progress. 1: Descriptor 0 transfer in progress. 0: Descriptor 0 is in queue when HPDMACR_DCP_VALID[0] bit is set in descriptor 0 Control Register.

### 5.4.1.15 Descriptor 1 Status Register

**Table 65 • HPDMAD1SR\_REG**

Bit Number	Name	Reset Value	Description
[31:4]	Reserved	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	HPDMASR_DCP_DERR[1]	0	Descriptor 1 destination transfer error. 1: Descriptor 1 transfer error 0: No error at destination end during descriptor 1 transfer This bit clears on writing '1' to HPDMAICR_CLR_XFR_INT[1] of the descriptor 1 Interrupt Clear register.
2	HPDMASR_DCP_SERR[1]	0	Descriptor 1 source transfer error. 1: Descriptor 1 transfer error occurred at source end 0: No error at source end during descriptor 1 transfer This bit clears on writing '1' to HPDMAICR_CLR_XFR_INT[1] of the descriptor 1 Interrupt Clear register.

**Table 65 • HPDMAD1SR\_REG (continued)**

Bit Number	Name	Reset Value	Description
1	HPDMASR_DCP_CMPLT[1]	0	Descriptor 1 transfer complete. 1: Descriptor 1 transfer completed successfully 0: Descriptor 1 transfer not completed This bit clears on writing '1' to HPDMAICR_CLR_XFR_INT[1] of the descriptor 1 Control Register.
0	HPDMASR_DCP_ACTIVE[1]	0	Descriptor 1 transfer in progress. 1: Descriptor 1 transfer in progress. 0: Descriptor 1 is in queue when HPDMACR_DCP_VALID[1] bit is set in Descriptor 1 Control Register.

### 5.4.1.16 Descriptor 2 Status Register

**Table 66 • HPDMAD2SR\_REG**

Bit Number	Name	Reset Value	Description
[31:4]	Reserved	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	HPDMASR_DCP_DERR[2]	0	Descriptor 2 destination transfer error. 1: Descriptor 2 transfer error 0: No error at destination end during descriptor 2 transfer This bit clears on writing '1' to HPDMAICR_CLR_XFR_INT[2] of the descriptor 2 Interrupt Clear register.
2	HPDMASR_DCP_SERR[2]	0	Descriptor 2 source transfer error. 1: Descriptor 2 transfer error occurred at source end 0: No error at source end during descriptor 2 transfer This bit clears on writing '1' to HPDMAICR_CLR_XFR_INT[2] of the descriptor 2 Interrupt Clear register.
1	HPDMASR_DCP_CMPLT[2]	0	Descriptor 2 transfer complete. 1: Descriptor 2 transfer completed successfully. 0: Descriptor 2 transfer not completed. This bit clears on writing '1' to HPDMAICR_CLR_XFR_INT[2] of the descriptor 2 Control Register.
0	HPDMASR_DCP_ACTIVE[2]	0	Descriptor 2 Transfer in progress. 1: Descriptor 2 transfer in progress. 0: Descriptor 2 is in queue when HPDMACR_DCP_VALID[2] bit is set in descriptor 2 Control Register.

### 5.4.1.17 Descriptor 3 Status Register

**Table 67 • HPDMAD3SR\_REG**

Bit Number	Name	Reset Value	Description
[31:4]	Reserved	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	HPDMASR_DCP_DERR[3]	0	Descriptor 3 destination transfer error. 1: Descriptor 3 transfer error 0: No error at destination end during descriptor 3 transfer This bit clears on writing '1' to HPDMAICR_CLR_XFR_INT[3] of the descriptor 3 Interrupt Clear register
2	HPDMASR_DCP_SERR[3]	0	Descriptor 3 source transfer error. 1: Descriptor 3 transfer error occurred at source end. 0: No error at source end during descriptor 3 transfer This bit clears on writing '1' to HPDMAICR_CLR_XFR_INT[3] of the descriptor 3 Interrupt Clear register.
1	HPDMASR_DCP_CMPLT[3]	0	Descriptor 2 transfer complete. 1: Descriptor 3 transfer completed successfully. 0: Descriptor 3 transfer not completed. This bit clears on writing '1' to HPDMAICR_CLR_XFR_INT[3] of the descriptor 3 Control Register.
0	HPDMASR_DCP_ACTIVE[3]	0	Descriptor 3 transfer in progress. 1: Descriptor 3 transfer in progress. 0: Descriptor 3 is in queue when HPDMACR_DCP_VALID[3] bit is set in descriptor 3 Control Register.

### 5.4.1.18 Descriptor 0 Pending Transfers Register

**Table 68 • HPDMAD0PTR\_REG**

Bit Number	Name	Reset Value	Description
[31:16]	HPDMAPTR_D0_DST_PNDNG	0	Descriptor 0 destination pending transfers in words. This register indicates the internal transfer size counter corresponding to the destination end of descriptor 0. At the end of the transfer, zero in this register indicates the successful transfer, and a non-zero value indicates error occurrence at the destination during the descriptor 0 transfer.
[15:0]	HPDMAPTR_D0_SRC_PNDNG	0	Descriptor 0 source pending transfers in words. This register indicates internal transfer size counter corresponding to source end of the descriptor 0. At the end of the transfer, zero in this register indicates the successful transfer, and a non-zero value indicates error occurrence at the source during the descriptor 0 transfer.

### 5.4.1.19 Descriptor 1 Pending Transfers Register

**Table 69 • HPDMAD1PTR\_REG**

Bit Number	Name	Reset Value	Description
[31:16]	HPDMAPTR_D1_DST_PNDNG	0	Descriptor 1 destination pending transfers in words. This register indicates the internal transfer size counter corresponding to the destination end of descriptor 1. At the end of the transfer, zero in this register indicates the successful transfer, and a non-zero value indicates error occurrence at the destination during descriptor 1 transfer.
[15:0]	HPDMAPTR_D1_SRC_PNDNG	0	Descriptor 1 source pending transfers in words. This register indicates the internal transfer size counter corresponding to the source end of descriptor 1. At the end of the transfer, zero in this register indicates the successful transfer, and a non-zero value indicates error occurrence at the source during descriptor 1 transfer.

### 5.4.1.20 Descriptor 2 Pending Transfers Register

**Table 70 • HPDMAD2PTR\_REG**

Bit Number	Name	Reset Value	Description
[31:16]	HPDMAPTR_D2_DST_PNDNG	0	Descriptor 2 destination pending transfers in words. This register indicates the internal transfer size counter corresponding to the destination end of descriptor 2. At the end of the transfer, zero in this register indicates the successful transfer, and a non-zero value indicates error occurrence at the destination during descriptor 2 transfer.
[15:0]	HPDMAPTR_D2_SRC_PNDNG	0	Descriptor 2 source pending transfers in words. This register indicates the internal transfer size counter corresponding to the source end of descriptor 2. At the end of the transfer, zero in this register indicates the successful transfer, and a non-zero value indicates error occurrence at the source during descriptor 2 transfer.

### 5.4.1.21 Descriptor 3 Pending Transfers Register

**Table 71 • HPDMAD3PTR\_REG**

Bit Number	Name	Reset Value	Description
[31:16]	HPDMAPTR_D3_DST_PNDNG	0	Descriptor 3 destination pending transfers in words. This register indicates the internal transfer size counter corresponding to the destination end of descriptor 3. At the end of the transfer, zero in this register indicates the successful transfer, and a non-zero value indicates error occurrence at the destination during descriptor 3 transfer.
[15:0]	HPDMAPTR_D3_SRC_PNDNG	0	Descriptor 3 source pending transfers in words. This register indicates the internal transfer size counter corresponding to the source end of descriptor 3. At the end of the transfer, zero in this register indicates the successful transfer, and a non-zero value indicates error occurrence at the source during descriptor 0 transfer.

### 5.4.1.22 HPDMA Interrupt Clear Register

**Table 72 • HPDMAICR\_REG**

Bit Number	Name	Reset Value	Description
[31:8]	Reserved	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	HPDMAICR_NON_WORD_INT[3]	0	When this bit is set, HPDMA clears the HPDMAEDR_DCP_NON_WORD_ERR[3] bit in the Empty Descriptor register. These bits always read back as 0.
6	HPDMAICR_NON_WORD_INT[2]	0	When this bit is set, HPDMA clears the HPDMAEDR_DCP_NON_WORD_ERR[2] bit in the Empty Descriptor register. These bits always read back as 0.
5	HPDMAICR_NON_WORD_INT[1]	0	When this bit is set, HPDMA clears the HPDMAEDR_DCP_NON_WORD_ERR[1] bit in the Empty Descriptor Register. These bits always read back as 0.
4	HPDMAICR_NON_WORD_INT[0]	0	When this bit is set, HPDMA clears the HPDMAEDR_DCP_NON_WORD_ERR[0] bit in the Empty Descriptor register. These bits always read back as 0.
3	HPDMAICR_CLR_XFR_INT[3]	0	When this bit is set, HPDMA clears the following register bits: Descriptor 3 Status Register HPDMASR_DCP_CMPLT[3] HPDMASR_DCP_DERR[3] HPDMASR_DCP_SERR[3] HPDMA Empty Descriptor register HPDMAEDR_DCP_NON_WORD_ERR[3]

**Table 72 • HPDMAICR\_REG (continued)**

Bit Number	Name	Reset Value	Description
2	HPDMAICR_CLR_XFR_INT[2]	0	When this bit is set, HPDMA clears the following register bits: Descriptor 2 Status Register HPDMASR_DCP_CMPLET[2] HPDMASR_DCP_DERR[2] HPDMASR_DCP_SERR[2] HPDMA Empty Descriptor register HPDMAEDR_DCP_NON_WORD_ERR[2]
1	HPDMAICR_CLR_XFR_INT[1]	0	When this bit is set, HPDMA clears the following register bits: Descriptor 1 Status Register HPDMASR_DCP_CMPLET[1] HPDMASR_DCP_DERR[1] HPDMASR_DCP_SERR[1] HPDMA Empty Descriptor register HPDMAEDR_DCP_NON_WORD_ERR[1]
0	HPDMAICR_CLR_XFR_INT[0]	0	When this bit is set, HPDMA clears the following register bits: Descriptor 0 Status Register HPDMASR_DCP_CMPLET[0] HPDMASR_DCP_DERR[0] HPDMASR_DCP_SERR[0] HPDMA Empty Descriptor register HPDMAEDR_DCP_NON_WORD_ERR[0]

### 5.4.1.23 HPDMA Debug Register

**Table 73 • HPDMADR\_REG**

Bit Number	Name	Reset Value	Description
[31:28]	Reserved	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.
[27:26]	HPDMADR_DMA_CST_DBG[1:0]	0	DMA controller FSM current state 01 – IDLE 10 – RUN
[25:22]	HPDMADR_RRBN_CST_DBG[3:0]	0	Round robin FSM current state 0001 – D0 0010 – D1 0100 – D2 1000 – D3
[21:19]	HPDMADR_RBC_CST_DBG[2:0]	0	Read buffer controller current state 001 – IDLE 010 – RUN 100 – WAIT
[18:16]	HPDMADR_WBC_CST_DBG[2:0]	0	Write buffer controller current state 001 – IDLE 010 – RUN 100 – WAIT

**Table 73 • HPDMADR\_REG (continued)**

Bit Number	Name	Reset Value	Description
[15:12]	HPDMADR_AHM2_CST_DBG[3:0]	0	Master 2 (HPMS DDR bridge) current state 0001 – IDLE 0010 – WRITE 0100 – READ 1000 – WAIT
[11:8]	HPDMADR_AHM1_CST_DBG[3:0]	0	Master 1 (AHB bus matrix) current state 0001 – IDLE 0010 – WRITE 0100 – READ 1000 – WAIT
[7:5]	HPDMADR_BFR_WR_PNTR[2:0]	0	HPDMA data buffer write pointer
[4:2]	HPDMADR_BFR_RD_PNTR[2:0]	0	HPDMA data buffer read pointer
1	HPDMADR_BFR_FULL	0	Data buffer is full; HPDMA controller initiates idle transfers on the source memory end. 1: Data buffer is full. 0: Data buffer is not full.
0	HPDMADR_BFR_EMPTY	1	Data buffer is empty; HPDMA controller initiates idle transfers on the destination memory end. 1: Data buffer is empty. 0: Data Buffer is not empty.

## 5.5 SYSREG Control Register

In addition to the specific HPDMA registers, the registers provided in the following table also control the behavior of the HPDMA peripheral. See [System Register Block](#), page 196 for a detailed description of each register and associated bits.

**Table 74 • SYSREG Control Registers**

Register Name	Register Type	Flash Write Protect	Reset Source	Description
SOFT_RESET_CR	RW-P	Bit	SYSRESET_N	Bit 17 is used for HPDMA reset '1' – Reset HPDMA '0' – Release from HPDMA reset For more information, see <a href="#">Table 143</a> , page 214.
MASTER_WEIGHT1_CR	RW-P	Register	SYSRESET_N	Bits 4:0 define round robin weight values for the HPDMA master. For more information, see <a href="#">Table 140</a> , page 213.

## 6 Peripheral DMA

The peripheral direct memory access (PDMA) is an AHB master associated with the AHB bus matrix, as shown in the following figure. The PDMA allows data transfer from various peripherals to memory, memory to peripherals, and memory to memory. There are the following three connections available to PDMA:

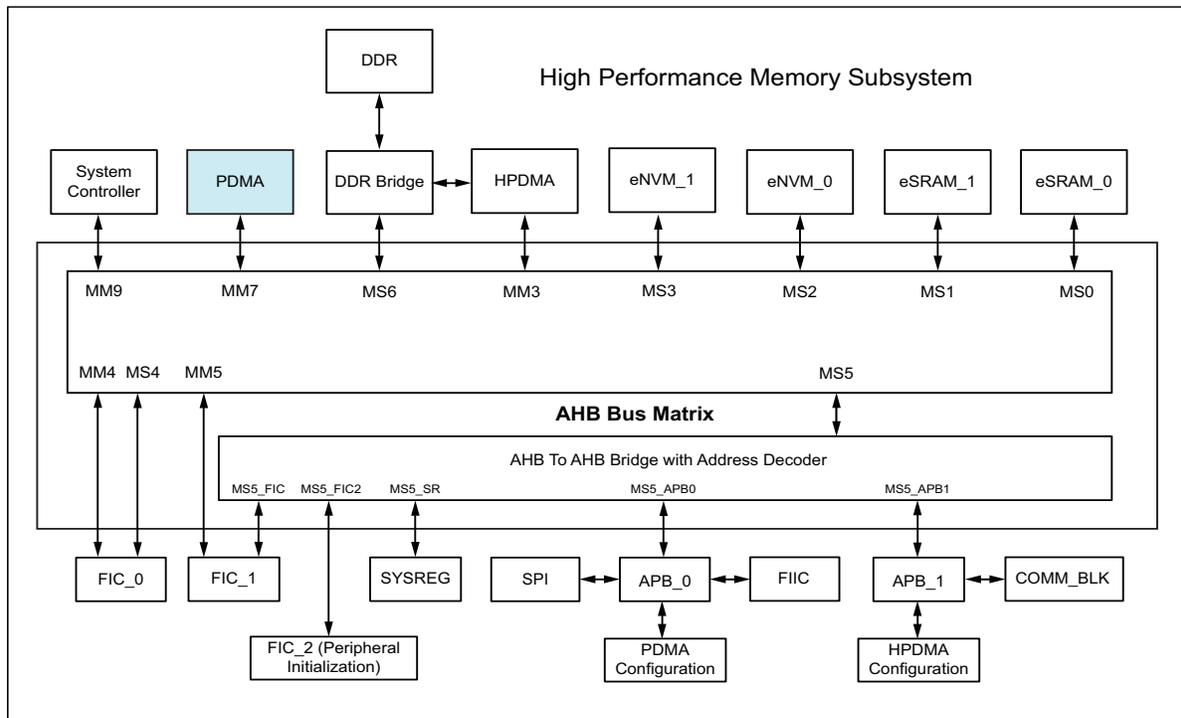
- HPMS peripheral
- HPMS memory
- FIC

HPMS peripheral can be any of the SPI or COMM\_BLK. HPMS memory can be any of the eNVM\_0, eNVM\_1, eSRAM\_0, or eSRAM\_1. FIC can connect to a fabric peripheral, a fabric internal memory (uSRAM/LSRAM), or a fabric interfaced external memory.

### 6.1 Features

- Up to 8 DMA channels
- Ping-pong mode support
- Channel priority designations
- Memory to memory DMA capable
- Interrupt capability

**Figure 66 • PDMA Interfacing with AHB Bus Matrix**



## 6.2 Functional Description

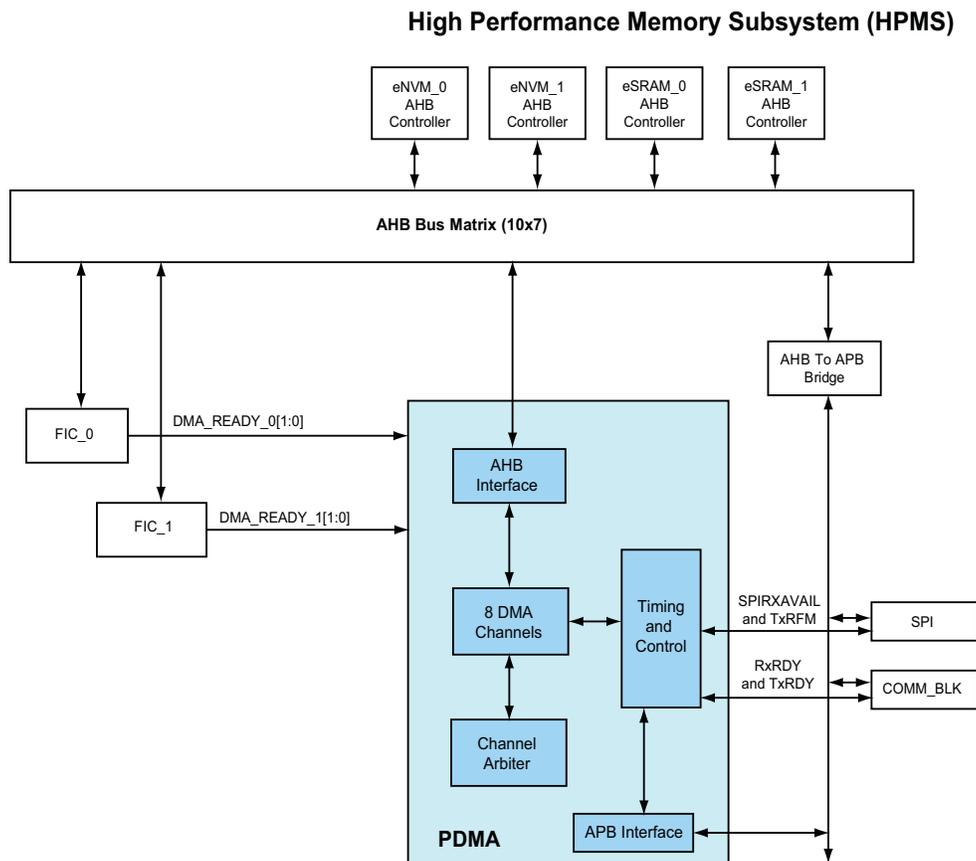
This section provides the detailed description of the PDMA.

### 6.2.1 Architecture Overview

The PDMA controller mainly consists of the following blocks, as shown in the following figure.

- AHB and APB Interfaces
- 8-Channel DMA Controller
- Timing and Control
- Channel Arbiter

**Figure 67 • PDMA Internal Architecture**



#### 6.2.1.1 AHB and APB Interfaces

As shown in the preceding figure, the PDMA has two interfaces — AHB-lite and APB. The APB interface is a 32-bit APB slave used for configuring the PDMA. DMA operations do not occur on the APB bus interface of the PDMA. The PDMA performs single cycle accesses on the AHB interface only and Burst mode is not supported.

### 6.2.1.2 8-Channel DMA Controller

The 8-channel DMA controller consists of eight instances of a single DMA channel, as shown in the preceding figure. Each channel can be configured to perform 8-bit, 16-bit, or 32-bit data transfers from the peripheral to memory, memory to peripheral, and memory to memory. Each DMA channel supports Ping-pong mode for continuous data transfer. To enable and use PDMA services, the AHB bus master matrix must configure the following 32-bit registers.

- CHANNEL\_x\_CONTROL
- CHANNEL\_x\_BUFFER\_A\_SRC\_ADDR
- CHANNEL\_x\_BUFFER\_A\_DST\_ADDR
- CHANNEL\_x\_BUFFER\_A\_TRANSFER\_COUNT
- CHANNEL\_x\_BUFFER\_B\_SRC\_ADDR
- CHANNEL\_x\_BUFFER\_B\_DST\_ADDR
- CHANNEL\_x\_BUFFER\_B\_TRANSFER\_COUNT

For more information about these registers, see [PDMA Configuration Register Bit Definitions](#), page 123.

If bidirectional DMA of peripheral to memory (receive) and memory to peripheral (transmit) is desired, two channels must be programmed appropriately. In particular, the TRANSFER\_SIZE fields in both the CHANNEL\_x\_CONTROL registers (see [Table 80](#), page 125) must be programmed identically. The PDMA performs the correct byte lane adjustments appropriate to the address being used on the AHB. Efficient use of memory storage is achieved in this manner, even if only performing byte or 16-bit accesses to or from a peripheral. For example, when the PDMA is accessing peripherals, the lowest 8 or 16 bits of the data bus are always used for 8-bit or 16-bit transfers. For 32-bit transfers, the full 32-bits are used. It is possible to configure the data width of a transfer to be independent of the address increment. The address increment at both ends of the DMA transfer can be different, which is required when reading from a peripheral holding register (single address) and writing to memory incrementally (many addresses).

Eight possible channels are available to the PDMA, as listed below.

- SPI\_0 to any HPMS memory-mapped location
- Any HPMS memory-mapped location to SPI\_0
- FPGA fabric peripheral on FIC\_0 to any HPMS memory-mapped location
- Any HPMS memory-mapped location to FPGA fabric peripheral on FIC\_0
- FPGA fabric peripheral on FIC\_1 to any HPMS memory-mapped location
- Any HPMS memory-mapped location to FPGA fabric peripheral on FIC\_1
- COMM\_BLK to any HPMS memory-mapped location
- Any HPMS memory-mapped locations to COMM\_BLK.

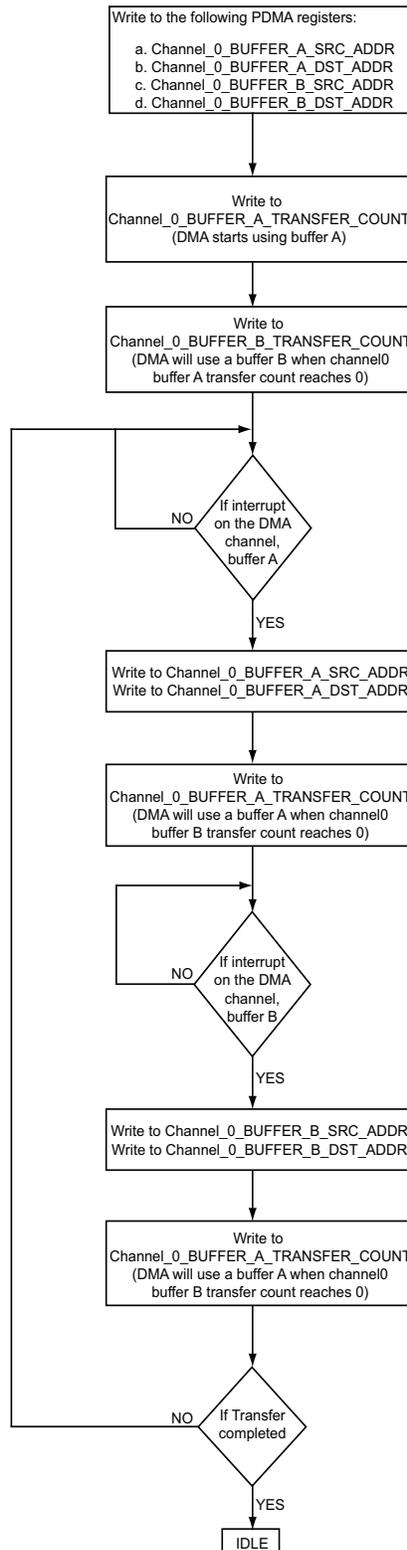
#### 6.2.1.2.1 Ping-Pong Mode

Ping-pong mode is a dual buffering scheme for continuous stream of operation. There are two buffers (Buffer A and Buffer B) associated with each DMA channel for ping-pong operation. This removes the real-time constraint on the Fabric master of having to service the DMA channel in real time, which would exist if there were only one DMA buffer per channel.

To begin a transaction, source address, destination address, and transfer size in bytes of buffer A and buffer B are to be configured by the AHB bus matrix master (such as Fabric master). The following figure shows the sequence of operations that must be performed by Fabric master for ping-pong operation on a configured DMA channel. The channel Control Register (CHANNEL\_x\_CONTROL) is configured initially before enabling ping-pong operation.

The following figure shows the ping-pong operation flow.

**Figure 68 • Ping-Pong Operation Flow for DMA Channel**



### 6.2.1.3 Timing and Control

The peripheral ready signals from the SPI and COMM\_BLK are directly connected to the PDMA. The PDMA takes care of writing or reading the receive or transmit holding registers within each peripheral using the APB interface.

The DMAREADY\_0 and DMAREADY\_1 signals correspond to the ready signals from the fabric peripheral. If the channel is configured for peripheral DMA and the direction is from the fabric peripheral to memory, this signal indicates that the fabric peripheral can write data to memory. If the channel is configured for peripheral DMA and the direction is from memory to the fabric peripheral, this signal indicates that the fabric peripheral can read data from memory. The PDMA does not support peripheral to peripheral data transfer and scatter-gather DMA.

### 6.2.1.4 Channel Arbiter

The channel arbiter is an arbitration algorithm used to service the channels based on the priority, as shown in [Figure 67](#), page 108. By default, all channels have equal priority. To configure the PDMA channel priority, `RATIO_HIGH_LOW` register must be configured by the AHB bus matrix master. The `RATIO_HIGH_LOW[RATIOHILO]` field indicates the ratio of high priority requests to low priority requests. For example, a `RATIOHILO` value of 3:1 means that a high priority DMA channel has 3 DMA access opportunities for every one access of a low priority DMA channel.

When the `RATIOHILO` value is set to 0, both high and low priority requests are serviced in a round robin fashion.

Refer to the following table and to the section [PDMA Register Map](#), page 120 for more information on configuring the register. The following table lists the valid values for `RATIOHILO`. All other values are reserved.

**Table 75 • RATIOHILO Field Definition**

Value	High:Low Ratio	Comments
0		Round robin
1	1:1	Ping-pong between high and low priority requests
3	3:1	3 high to 1 low
7	7:1	7 high to 1 low
15	15:1	15 high to 1 low
31	31:1	31 high to 1 low
63	63:1	63 high to 1 low
127	127:1	127 high to 1 low
255	255:1	255 high to 1 low
All others		Reserved

## 6.2.2 PDMA Port List

**Table 76 • Port List**

Name	Type	Polarity	Description
DMAREADY_FIC_0	Input	High	Ready signal from fabric peripheral to DMA through FIC_0
DMAREADY_FIC_1	Input	High	Ready signal from fabric peripheral to DMA through FIC_1

## 6.2.3 Initialization

To initiate and setup DMA transactions, PDMA has to be initialized. The initialization process starts with a reset sequence followed by Channel configuration and interrupt configuration.

### 6.2.3.1 Reset

The PDMA registers are reset on power-up. The PDMA can be reset by configuring the following

- Bit[5] of SOFT\_RESET\_CR system register (see [Table 143](#), page 214).
- Bit[5] of the CHANNEL\_x\_CONTROL register for channel reset (see [Table 80](#), page 125).

### 6.2.3.2 Channel Configuration

Before configuring each PDMA channel, the round robin weight is specified if needed, using the MASTER\_WEIGHT\_CR register or configuring the AHB bus matrix in Libero SoC.

To configure each PDMA channel, following fields of the channel Control Register has to be set:

- Peripheral select - CHANNEL\_x\_CONTROL[26:23]
- No. of wait states - CHANNEL\_x\_CONTROL[21:14]
- Source and/or destination address increment - CHANNEL\_x\_CONTROL[13:10]
- Channel priority - CHANNEL\_x\_CONTROL[9]
- Interrupt enable - CHANNEL\_x\_CONTROL[6]
- Transfer size - CHANNEL\_x\_CONTROL[3:2]
- Direction - CHANNEL\_x\_CONTROL[1]
- Select the data Transfer type - CHANNEL\_x\_CONTROL[0]

For CHANNEL\_x\_CONTROL register bit definitions, see [Table 80](#), page 125.

### 6.2.3.3 Interrupt

The PDMA Interrupt signal is mapped to the dedicated interrupt signal HPMS\_INT\_M2F[8] of the fabric interface interrupt controller (FIIC). This is to interrupt the user logic instantiated in the FPGA.

To determine transfer complete interrupt for each channel, the BUFFER\_STATUS\_x register bits[1:0] has to be monitored. The bit[7] and bit[8] of CHANNEL\_x\_CONTROL register are used to clear the transfer complete interrupts of the channel.

## 6.2.4 Details of Operations

After initialization, the PDMA is ready to function in any one of following transfer modes:

- Peripheral to Memory Transfers/Memory to Memory Transfers
- Posted APB Writes

### 6.2.4.1 Peripheral to Memory Transfers/Memory to Memory Transfers

For peripheral to memory or memory to memory transfer, the DMA transfer starts if BUFFER\_A\_TRANSFER\_COUNT or BUFFER\_B\_TRANSFER\_COUNT is non-zero.

Before the transfer, the source address (CHANNEL\_x\_BUFFER\_A\_SRC\_ADDR) and destination address (CHANNEL\_x\_BUFFER\_B\_DST\_ADDR) of a channel are configured; then a write to one of the transfer count registers begins the DMA transaction. Additionally, FPGA fabric master can also write to the Control Register first and turn pause on, if needed, then turn it off later.

If the PAUSE bit in the CHANNEL\_x\_CONTROL register (see [Table 80](#), page 125) is set, when a non-zero value is written to BUFFER\_A\_TRANSFER\_COUNT or BUFFER\_B\_TRANSFER\_COUNT, then the DMA transaction waits to begin the transfer until PAUSE is cleared.

If bidirectional DMA of peripheral to memory (receive) and memory to peripheral (transmit) is desired, two channels must be programmed appropriately. In particular, the TRANSFER\_SIZE fields in both the CHANNEL\_x\_CONTROL registers must be programmed identically.

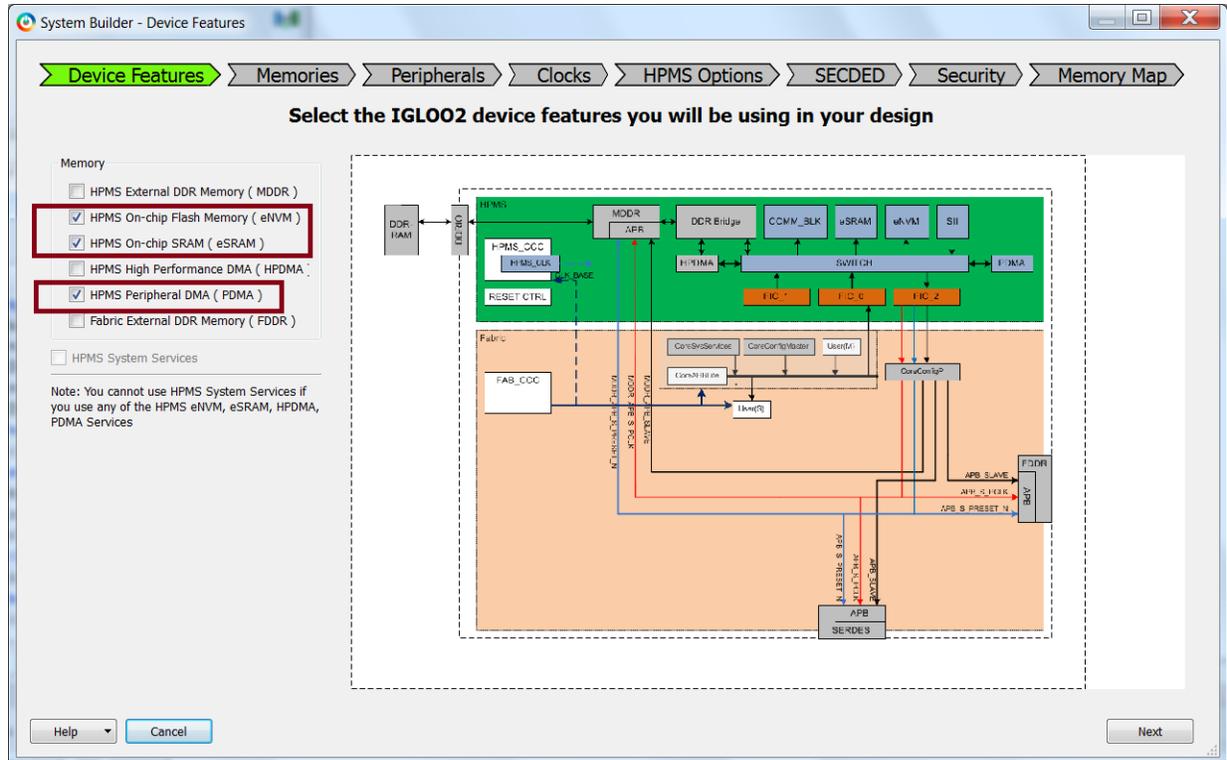
Channels can be assigned to peripherals or memory arbitrarily. For example, to receive only DMA data from one of the SPI ports, only one channel is required. In this case, the DIR bit in the CHANNEL\_x\_CONTROL register should be set to 0 (peripheral to memory) and the PERIPHERAL\_SEL field should be set to 4 (SPI receive to memory).



The following steps describe how to enable the PDMA.

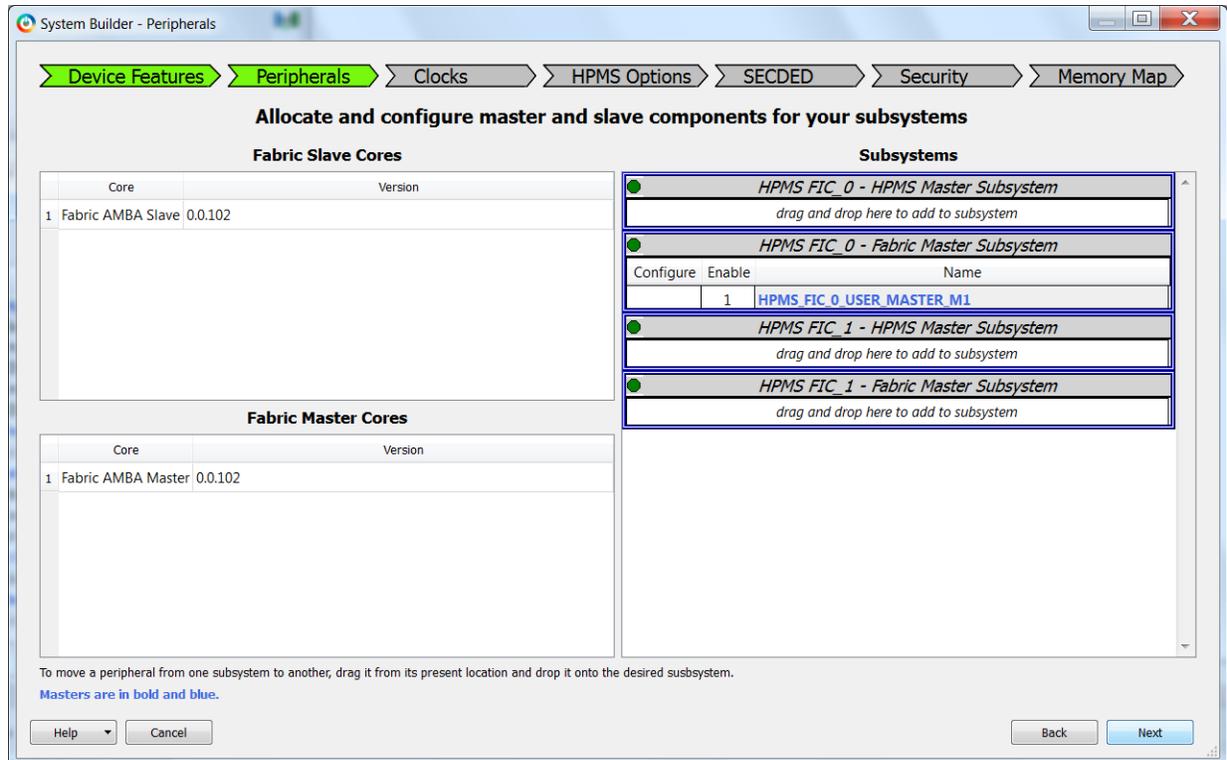
1. Check the **HPMS On-chip Flash Memory (eNVM)**, **HPMS On-chip SRAM (eSRAM)**, and **HPMS Peripheral DMA (PDMA)** check boxes in the **Device Features** tab and leave the other check boxes unchecked. The following figure shows the **System Builder - Device Features** tab.

**Figure 70 • System Builder - Device Features Tab**



- Click **Next** to navigate to the **Peripherals** tab. The following figure shows the **System Builder – Peripherals** tab. In **Peripherals** tab, the fabric master core and slave components to HPMS\_FIC\_0 and HPMS\_FIC\_1 are added automatically by Libero SoC depending on selection on HPMS masters.

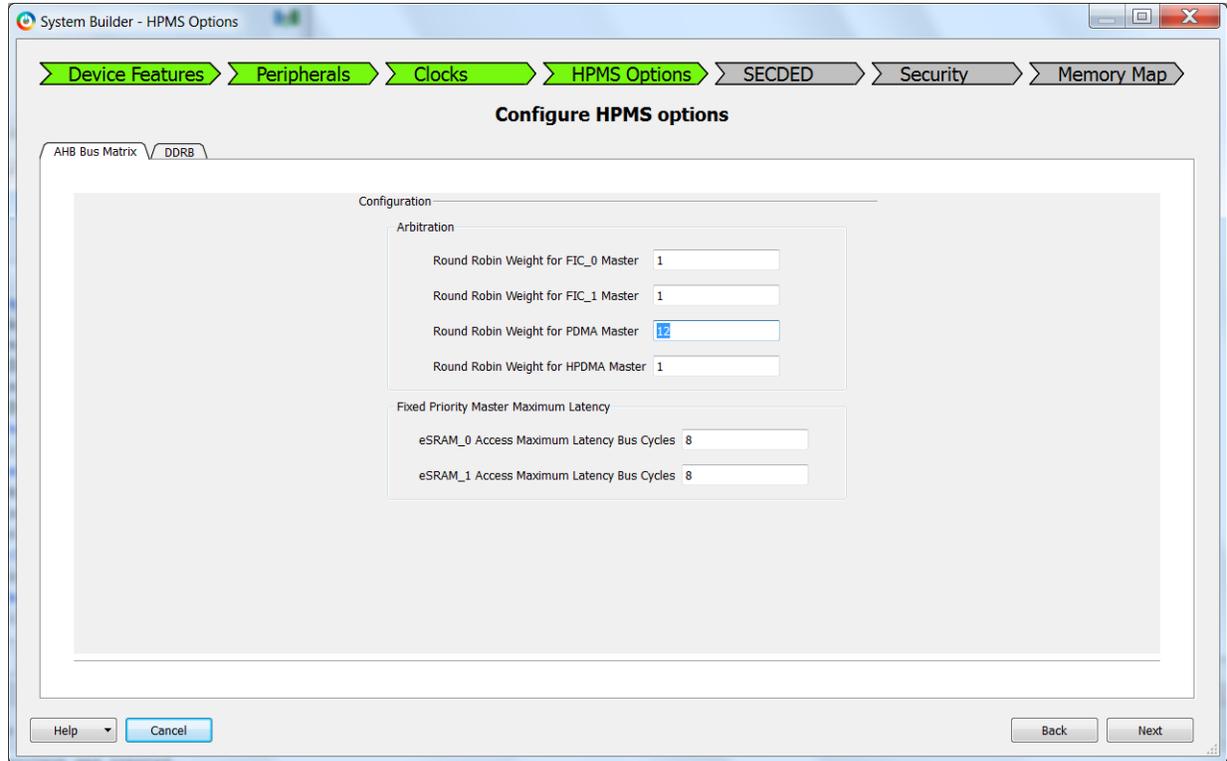
**Figure 71 • System Builder - Peripherals Tab**



- Navigate to the **HPMS Options** in the **System Builder** wizard.

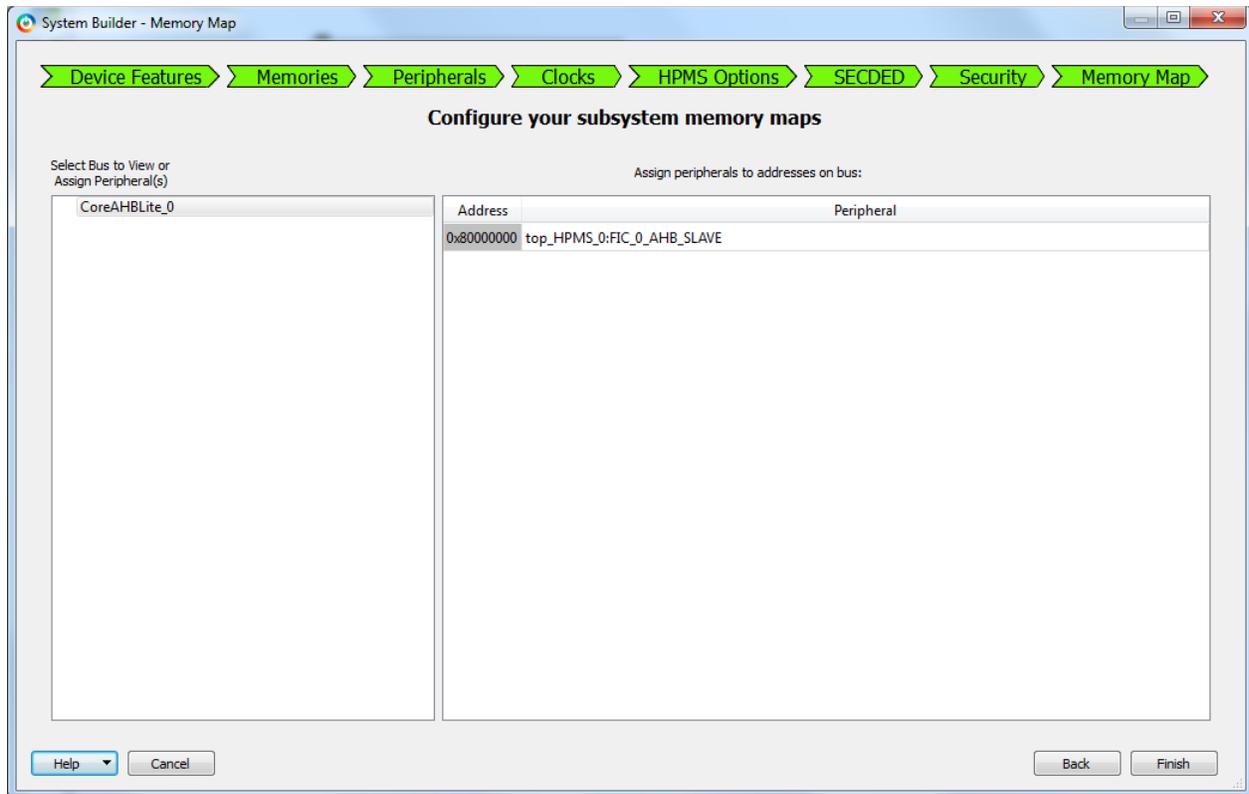
4. In the **HPMS Options** tab, configure the weight values for the PDMA master. The round robin weight is the number of consecutive transfers master performs without being interrupted during an access. In the following figure, the configured round robin weight value for PDMA is 12. It means the PDMA performs 12 consecutive transfers during its access before the next master takes control. The following figure shows the **Configuration** panel in the **HPMS Options** tab.

**Figure 72 • Configuring PDMA Weight Values**



5. Navigate to the **Memory Map** tab giving the required data in the rest of the **System Builder** tabs. The following figure shows the **System Builder - Memory Map** tab. Click **Finish** to proceed with creating the HPMS subsystem (see [HPMS Subsystem](#), page 117).

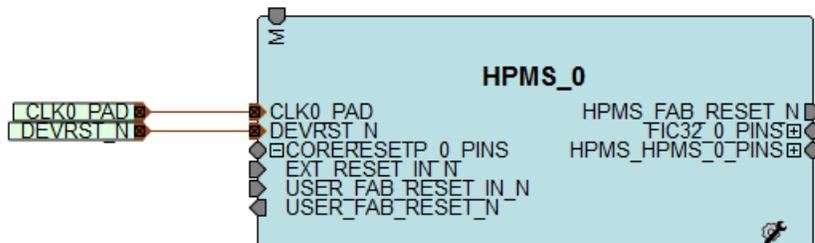
**Figure 73 • System Builder - Memory Map Tab**



### 6.3.1 HPMS Subsystem

The following figure shows an example HPMS subsystem that can be used to access PDMA using the FPGA fabric master.

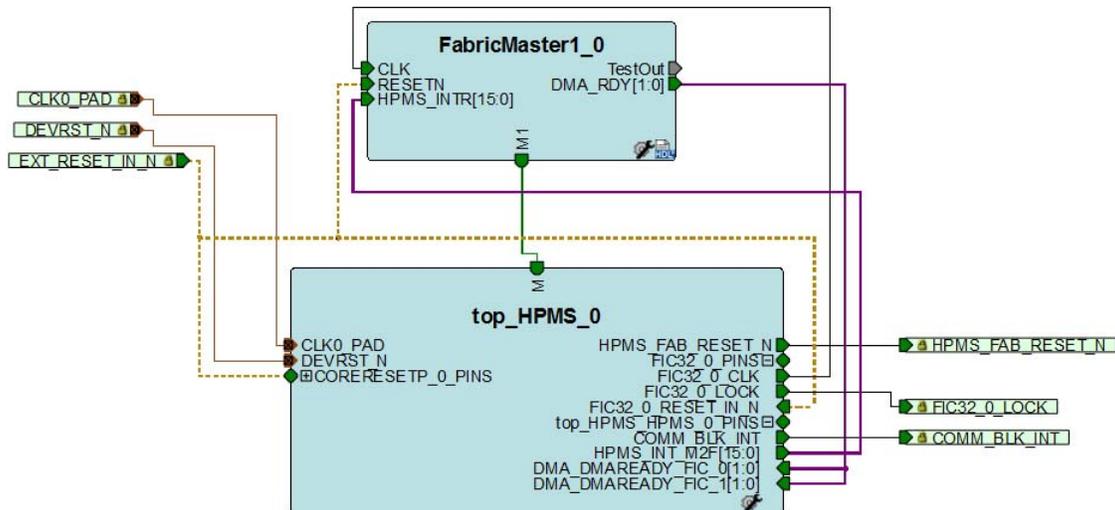
**Figure 74 • HPMS Subsystem**



## 6.3.2 HPMS Subsystem Connected to the FPGA Fabric Master

The following figure shows the FPGA fabric master connected to the AHB master port.

**Figure 75 • HPMS Subsystem Connected to the FPGA Fabric Master**



## 6.3.3 SPI\_0 to eSRAM\_0

The following steps describe how to use the Fabric master for configuring PDMA to transfer data from peripheral (SPI\_0) to memory (eSRAM\_0).

1. Enable PDMA using the **System Builder** wizard.
2. Reset the PDMA channel by asserting `CHANNEL_x_CONTROL[5] = 1`.
3. Check the interrupt status by monitoring `CHANNEL_x_STATUS[1:0]` bits and clear the interrupts using `CHANNEL_x_CONTROL[8:7]` bits.
4. Configure PDMA:
  - i. Select the SPI\_0 block by configuring `CHANNEL_x_CONTROL[26:23] = 0101`.
  - ii. Set the transfer type by configuring `CHANNEL_x_CONTROL[0] = 1`.
  - iii. Set the direction by configuring `CHANNEL_x_CONTROL[1] = 0`.
  - iv. Set the transfer size/data width using `CHANNEL_x_CONTROL[3:2]`.
  - v. Set the source and destination address increment using `CHANNEL_x_CONTROL[13:10]`.
  - vi. Set the channel priority using `CHANNEL_x_CONTROL[9]`. If all the channels are configured as low priority, requests are serviced in round robin fashion.
  - vii. Set the wait states using `WRITE_ADJ (CHANNEL_x_CONTROL[21:14])`.
5. Enable the PDMA interrupt to the Fabric by setting `CHANNEL_x_CONTROL[4]` and `PDMAINTERRUPT_ENBL` bit of `INTERRUPT_ENABLE0` register (refer to [Fabric Interface Interrupt Controller](#), page 239).
6. Start the data transfer by configuring:
  - i. Buffer A and buffer B source memory addresses with SPI\_0 address and destination addresses with eSRAM\_0 address
    - 32-bit buffer A source address (`CHANNEL_x_BUFFER_A_SRC_ADDR`)
    - 32-bit buffer A destination address (`CHANNEL_x_BUFFER_A_DST_ADDR`)
    - 32-bit buffer B source address (`CHANNEL_x_BUFFER_B_SRC_ADDR`)
    - 32-bit buffer B destination address (`CHANNEL_x_BUFFER_B_DST_ADDR`)
  - ii. Transfer count in bytes.
    - 32-bit buffer A transfer count (`CHANNEL_x_BUFFER_A_TRANSFER_COUNT = 0x64/100` bytes)
    - 32-bit buffer B transfer count (`CHANNEL_x_BUFFER_B_TRANSFER_COUNT = 0x64/100` bytes)

7. Read the Status Register BUFFER\_STATUS, to check the status on the interrupt.
8. To pause the data transfers for a channel, reset bit[6] of channel Control Register CHANNEL\_x\_CONTROL[6].
9. Check for buffer A data transfer completion by monitoring the dedicated interrupt HPMS\_INT\_M2F[8] (PDMINTERRUPT) or by polling bit[0] of the channel Status Register CHANNEL\_x\_STATUS[0].
10. When the interrupt is detected, configure the following:
  - 32-bit buffer A source address with SPI\_0 address and 32-bit buffer A destination address with eSRAM0 address
  - 32-bit buffer A transfer count
11. Check for buffer B data transfer completion by monitoring PDMINTERRUPT or by polling bit[1] of the channel Status Register CHANNEL\_x\_STATUS[1].
12. When the interrupt is detected, configure the following:
  - 32-bit buffer B source address with SPI\_0 address and 32-bit buffer B destination address with eSRAM0 address
  - 32-bit buffer B transfer count
13. Check for transfer completion by monitoring CHANNEL\_x\_STATUS[1:0] bits.
14. For more information on bi-directional DMA transfers, refer to [Details of Operations](#), page 112.

### 6.3.4 eNMV\_0 to eSRAM\_0

The following steps describe how to use the Fabric master for configuring PDMA to transfer data from a memory (eNMV\_0) to another memory (eSRAM\_0).

1. Enable PDMA using the **System Builder** wizard.
2. Reset the PDMA channel by asserting CHANNEL\_x\_CONTROL[5] = 1.
3. Check the interrupt status by monitoring CHANNEL\_x\_STATUS[1:0] bits and clear the interrupts using CHANNEL\_x\_CONTROL[8:7] bits.
4. Configure PDMA:
  - Set the transfer type by configuring CHANNEL\_x\_CONTROL[0] = 0.
  - Set the transfer size/data width using CHANNEL\_x\_CONTROL[3:2].
  - Set the source and destination address increment using CHANNEL\_x\_CONTROL[13:10].
  - Set the channel priority using CHANNEL\_x\_CONTROL[9]. If all the channels are configured as low priority, requests are serviced in round robin fashion.
5. Enable the PDMA interrupt to the Fabric by setting CHANNEL\_x\_CONTROL[4] and PDMINTERRUPT\_ENBL bit of INTERRUPT\_ENABLE0 register.
6. Start the data transfer by configuring:
  - i. Buffer A and buffer B source address with eNVM\_0 address and destination addresses with eSRAM\_0 address.
    - 32-bit buffer A source address (CHANNEL\_x\_BUFFER\_A\_SRC\_ADDR)
    - 32-bit buffer A destination address (CHANNEL\_x\_BUFFER\_A\_DST\_ADDR)
    - 32-bit buffer B source address (CHANNEL\_x\_BUFFER\_B\_SRC\_ADDR)
    - 32-bit buffer B destination address (CHANNEL\_x\_BUFFER\_B\_DST\_ADDR)
  - ii. Transfer count in bytes.
    - 32-bit buffer A transfer count (CHANNEL\_x\_BUFFER\_A\_TRANSFER\_COUNT = 0x64/100 bytes)
    - 32-bit buffer B transfer count (CHANNEL\_x\_BUFFER\_B\_TRANSFER\_COUNT = 0x64/100 bytes)
7. Read the Status Register BUFFER\_STATUS, to check the status on the interrupt.
8. To pause the data transfers for a channel, reset bit[6] of channel Control Register CHANNEL\_x\_CONTROL[6].
9. Check for buffer A data transfer completion by monitoring the dedicated interrupt HPMS\_INT\_M2F[8] (PDMINTERRUPT) or by polling bit[0] of the channel Status Register CHANNEL\_x\_STATUS[0].
10. When the interrupt is detected, configure the following:
  - 32-bit buffer A with eNVM\_0 source address and 32-bit buffer A with eSRAM\_0 destination address
  - 32-bit buffer A transfer count

11. Check for buffer B data transfer completion by monitoring PDMAINTERRUPT or by polling bit[1] of the channel Status Register CHANNEL\_x\_STATUS[1].
12. When the interrupt is detected, configure the following:
  - 32-bit buffer B with eNVM\_0 source address and 32-bit buffer B with eSRAM\_0 destination address
  - 32-bit buffer B transfer count
13. Check for transfer completion by monitoring CHANNEL\_x\_STATUS[1:0] bits.
14. For more information on bi-directional DMA transfers, see [Details of Operations](#), page 112.

**Note:** The HPMS PDMA does not support full-behavioral simulation models.

## 6.4 PDMA Register Map

The following table summarizes each of the registers covered by this document. The base address is 0x40003000.

**Table 77 • IGLOO2 FPGA PDMA Register Map**

Register Name	Address Offset	Register Type	Reset Value	Description
Ratio_HIGH_LOW	0x00	R/W	0	Ratio of high priority transfers versus low priority transfers. For more information, see <a href="#">Table 78</a> , page 123.
BUFFER_STATUS	0x04	R	0	Indicates when buffers have drained. For more information, see <a href="#">Table 79</a> , page 123.
CHANNEL_x_CONTROL (X=0)	0x20	R/W	0	Channel 0 Control Register. For more information, see <a href="#">Table 80</a> , page 125.
CHANNEL_x_STATUS (X=0)	0x24	R	0	Channel 0 Status Register. For more information, see <a href="#">Table 82</a> , page 126.
CHANNEL_x_BUFFER_A_SRC_ADDR (x=0)	0x28	R/W	0	Channel 0 buffer A source address. For more information, see <a href="#">Table 83</a> , page 127.
CHANNEL_x_BUFFER_A_DST_ADDR (x=0)	0x2C	R/W	0	Channel 0 buffer A destination address. For more information, see <a href="#">Table 84</a> , page 127.
CHANNEL_x_BUFFER_A_TRANSFER_COUNT (x=0)	0x30	R/W	0	Channel 0 buffer A transfer count. For more information, see <a href="#">Table 85</a> , page 127.
CHANNEL_x_BUFFER_B_SRC_ADDR (x=0)	0x34	R/W	0	Channel 0 buffer B source address. For more information, see <a href="#">Table 86</a> , page 128.
CHANNEL_x_BUFFER_B_DST_ADDR (x=0)	0x38	R/W	0	Channel 0 buffer B destination address. For more information, see <a href="#">Table 87</a> , page 128.
CHANNEL_x_BUFFER_B_TRANSFER_COUNT (x=0)	0x3C	R/W	0	Channel 0 buffer B transfer count. For more information, see <a href="#">Table 88</a> , page 128.
CHANNEL_1_CONTROL	0x40	R/W	0	Channel 1 Control Register.
CHANNEL_1_STATUS	0x44	R	0	Channel 1 Status Register.

**Table 77 • IGLOO2 FPGA PDMA Register Map (continued)**

Register Name	Address Offset	Register Type	Reset Value	Description
CHANNEL_1_BUFFER_A_SRC_ADDR	0x48	R/W	0	Channel 1 buffer A source address.
CHANNEL_1_BUFFER_A_DST_ADDR	0x4C	R/W	0	Channel 1 buffer A destination address.
CHANNEL_1_BUFFER_A_TRANSFER_COUNT	0x50	R/W	0	Channel 1 buffer A transfer count.
CHANNEL_1_BUFFER_B_SRC_ADDR	0x54	R/W	0	Channel 1 buffer B source address.
CHANNEL_1_BUFFER_B_DST_ADDR	0x58	R/W	0	Channel 1 buffer B destination address.
CHANNEL_1_BUFFER_B_TRANSFER_COUNT	0x5C	R/W	0	Channel 1 buffer B transfer count.
CHANNEL_2_CONTROL	0x60	R/W	0	Channel 2 Control Register.
CHANNEL_2_STATUS	0x64	R	0	Channel 2 Status Register.
CHANNEL_2_BUFFER_A_SRC_ADDR	0x68	R/W	0	Channel 2 buffer A source address.
CHANNEL_2_BUFFER_A_DST_ADDR	0x6C	R/W	0	Channel 2 buffer A destination address.
CHANNEL_2_BUFFER_A_TRANSFER_COUNT	0x70	R/W	0	Channel 2 buffer A transfer count.
CHANNEL_2_BUFFER_B_SRC_ADDR	0x74	R/W	0	Channel 2 buffer B source address.
CHANNEL_2_BUFFER_B_DST_ADDR	0x78	R/W	0	Channel 2 buffer B destination address.
CHANNEL_2_BUFFER_B_TRANSFER_COUNT	0x7C	R/W	0	Channel 2 buffer B transfer count.
CHANNEL_3_CONTROL	0x80	R/W	0	Channel 3 Control Register.
CHANNEL_3_STATUS	0x84	R	0	Channel 3 Status Register.
CHANNEL_3_BUFFER_A_SRC_ADDR	0x88	R/W	0	Channel 3 buffer A source address.
CHANNEL_3_BUFFER_A_DST_ADDR	0x8C	R/W	0	Channel 3 buffer A destination address.
CHANNEL_3_BUFFER_A_TRANSFER_COUNT	0x90	R/W	0	Channel 3 buffer A transfer count.
CHANNEL_3_BUFFER_B_SRC_ADDR	0x94	R/W	0	Channel 3 buffer B source address.
CHANNEL_3_BUFFER_B_DST_ADDR	0x98	R/W	0	Channel 3 buffer B destination address.
CHANNEL_3_BUFFER_B_TRANSFER_COUNT	0x9C	R/W	0	Channel 3 buffer B transfer count.
CHANNEL_4_CONTROL	0xA0	R/W	0	Channel 4 Control Register.
CHANNEL_4_STATUS	0xA4	R	0	Channel 4 Status Register.

**Table 77 • IGLOO2 FPGA PDMA Register Map (continued)**

Register Name	Address Offset	Register Type	Reset Value	Description
CHANNEL_4_BUFFER_A_SRC_ADDR	0xA8	R/W	0	Channel 4 buffer A source address.
CHANNEL_4_BUFFER_A_DST_ADDR	0xAC	R/W	0	Channel 4 buffer A destination address.
CHANNEL_4_BUFFER_A_TRANSFER_COUNT	0xB0	R/W	0	Channel 4 buffer A transfer count.
CHANNEL_4_BUFFER_B_SRC_ADDR	0xB4	R/W	0	Channel 4 buffer B source address.
CHANNEL_4_BUFFER_B_DST_ADDR	0xB8	R/W	0	Channel 4 buffer B destination address.
CHANNEL_4_BUFFER_B_TRANSFER_COUNT	0xBC	R/W	0	Channel 4 buffer B transfer count.
CHANNEL_5_CONTROL	0xC0	R/W	0	Channel 5 Control Register.
CHANNEL_5_STATUS	0xC4	R	0	Channel 5 Status Register.
CHANNEL_5_BUFFER_A_SRC_ADDR	0xC8	R/W	0	Channel 5 buffer A source address.
CHANNEL_5_BUFFER_A_DST_ADDR	0xCC	R/W	0	Channel 5 buffer A destination address.
CHANNEL_5_BUFFER_A_TRANSFER_COUNT	0xD0	R/W	0	Channel 5 buffer A transfer count.
CHANNEL_5_BUFFER_B_SRC_ADDR	0xD4	R/W	0	Channel 5 buffer B source address.
CHANNEL_5_BUFFER_B_DST_ADDR	0xD8	R/W	0	Channel 5 buffer B destination address.
CHANNEL_5_BUFFER_B_TRANSFER_COUNT	0xDC	R/W	0	Channel 5 buffer B transfer count.
CHANNEL_6_CONTROL	0xE0	R/W	0	Channel 6 Control Register.
CHANNEL_6_STATUS	0xE4	R	0	Channel 6 Status Register.
CHANNEL_6_BUFFER_A_SRC_ADDR	0xE8	R/W	0	Channel 6 buffer A source address.
CHANNEL_6_BUFFER_A_DST_ADDR	0xEC	R/W	0	Channel 6 buffer A destination address.
CHANNEL_6_BUFFER_A_TRANSFER_COUNT	0xF0	R/W	0	Channel 6 buffer A transfer count.
CHANNEL_6_BUFFER_B_SRC_ADDR	0xF4	R/W	0	Channel 6 buffer B source address.
CHANNEL_6_BUFFER_B_DST_ADDR	0xF8	R/W	0	Channel 6 buffer B destination address.
CHANNEL_6_BUFFER_B_TRANSFER_COUNT	0xFC	R/W	0	Channel 6 buffer B transfer count.
CHANNEL_7_CONTROL	0x100	R/W	0	Channel 7 Control Register.
CHANNEL_7_STATUS	0x104	R	0	Channel 7 Status Register.

**Table 77 • IGLOO2 FPGA PDMA Register Map (continued)**

Register Name	Address Offset	Register Type	Reset Value	Description
CHANNEL_7_BUFFER_A_SRC_ADDR	0x108	R/W	0	Channel 7 buffer A source address.
CHANNEL_7_BUFFER_A_DST_ADDR	0x10C	R/W	0	Channel 7 buffer A destination address.
CHANNEL_7_BUFFER_A_TRANSFER_COUNT	0x110	R/W	0	Channel 7 buffer A transfer count.
CHANNEL_7_BUFFER_B_SRC_ADDR	0x114	R/W	0	Channel 7 buffer B source address.
CHANNEL_7_BUFFER_B_DST_ADDR	0x118	R/W	0	Channel 7 buffer B destination address.
CHANNEL_7_BUFFER_B_TRANSFER_COUNT	0x11C	R/W	0	Channel 7 buffer B transfer count.

## 6.4.1 PDMA Configuration Register Bit Definitions

The registers listed in the following tables are present in the PDMA engine:

### 6.4.1.1 RATIO\_HIGH\_LOW Register Bit Definition

**Table 78 • Ratio\_HIGH\_LOW**

Bit Number	Name	Reset Value	Description
[31:8]	Reserved	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.
[0:7]	RATIOHIL O	0	Indicates the ratio of high priority to low priority for DMA access opportunities. This register gives the number of DMA opportunities provided by the channel arbiter to high priority channels for every one opportunity provided to a low priority channel. Only certain values are allowed, as shown in <a href="#">Table 75</a> , page 111.

### 6.4.1.2 BUFFER\_STATUS Register Bit Definition

**Table 79 • BUFFER\_STATUS**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	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	CH7BUFB	0	If CH_COMP_B for channel 7 is set and if BUF_B_SEL for channel 7 is clear, this bit is asserted.
14	CH7BUFA	0	If CH_COMP_A for channel 7 is set and if BUF_A_SEL for channel 7 is clear, this bit is asserted.
13	CH6BUFB	0	If CH_COMP_B for channel 6 is set and if BUF_B_SEL for channel 6 is clear, this bit is asserted.
12	CH6BUFA	0	If CH_COMP_A for channel 6 is set and if BUF_A_SEL for channel 6 is clear, this bit is asserted.

**Table 79 • BUFFER\_STATUS (continued)**

Bit Number	Name	Reset Value	Description
11	CH5BUFB	0	If CH_COMP_B for channel 5 is set and if BUF_B_SEL for channel 5 is clear, this bit is asserted.
10	CH5BUFA	0	If CH_COMP_A for channel 5 is set and if BUF_A_SEL for channel 5 is clear, this bit is asserted.
9	CH4BUFB	0	If CH_COMP_B for channel 4 is set and if BUF_B_SEL for channel 4 is clear, this bit is asserted.
8	CH4BUFA	0	If CH_COMP_A for channel 4 is set and if BUF_A_SEL for channel 4 is clear, this bit is asserted.
7	CH3BUFB	0	If CH_COMP_B for channel 3 is set and if BUF_B_SEL for channel 3 is clear, this bit is asserted.
6	CH3BUFA	0	If CH_COMP_A for channel 3 is set and if BUF_A_SEL for channel 3 is clear, this bit is asserted.
5	CH2BUFB	0	If CH_COMP_B for channel 2 is set and if BUF_B_SEL for channel 2 is clear, this bit is asserted.
4	CH2BUFA	0	If CH_COMP_A for channel 2 is set and if BUF_A_SEL for channel 2 is clear, this bit is asserted.
3	CH1BUFB	0	If CH_COMP_B for channel 1 is set and if BUF_B_SEL for channel 1 is clear, this bit is asserted.
2	CH1BUFA	0	If CH_COMP_A for channel 1 is set and if BUF_A_SEL for channel 1 is clear, this bit is asserted.
1	CH0BUFB	0	If CH_COMP_B for channel 0 is set and if BUF_B_SEL for channel 0 is clear, this bit is asserted.
0	CH0BUFA	0	If CH_COMP_A for channel 0 is set and if BUF_A_SEL for channel 0 is clear, this bit is asserted.

### 6.4.1.3 CHANNEL\_x\_CONTROL Register Bit Definition

**Table 80 • CHANNEL\_x\_CONTROL**

Bit Number	Name	Reset Value	Description
[31:27]	Reserved	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.
[26:23]	PERIPHERAL_SEL	0	Selects the peripheral assigned to this channel. Refer to <a href="#">Table 81</a> , page 126.
22	Reserved	0	
[21:14]	WRITE_ADJ	0	Contains a binary value, indicating the number of HPMS_CLK periods which the PDMA must wait after completion of a read or write access to a peripheral before evaluating the out-of-band status signals from that peripheral for another transfer. This is typically used to ensure that a posted write has fully completed to the peripheral in cases where the peripheral is running at a lower clock frequency than the PDMA. However, it may also be used to allow the PDMA to take account of internal latencies in the peripheral, where the ready status of a FIFO may not be available for a number of clock ticks after a read or write, due to internal synchronization delays, for example, within the peripheral. This applies particularly in the case of user-designed peripherals in the FPGA fabric.
[13:12]	DEST_ADDR_INC	00	Controls the address increment at the destination end of the DMA transfer. The values have the following meanings: 00: 0 bytes 01: 1 byte 10: 2 bytes 11: 4 bytes
[11:10]	SRC_ADDR_INC	00	Controls the address increment at the source end of the DMA transfer. The values have the following meanings: 00: 0 bytes 01: 1 byte 10: 2 bytes 11: 4 bytes
9	HI_PRIORITY	0	When asserted, this channel is treated as high priority by the arbitration state machine.
8	CLR_COMP_B	0	When asserted, clears the CH_COMP_B bit in the channel Status Register and the buffer Status Register for this buffer in this channel. This causes PDMAINTERRUPT to negate if not being held asserted by another channel. This bit always reads back as zero.
7	CLR_COMP_A	0	When asserted, clears the CH_COMP_A bit in the channel Status Register and the buffer Status Register for this buffer in this channel. This causes PDMAINTERRUPT to negate if not being held asserted by another channel. This bit always reads back as zero.
6	INTEN	0	When asserted, a DMA completion on this channel causes PDMAINTERRUPT to assert.
5	RESET	0	When asserted, resets this channel. Always reads back as zero.
4	PAUSE	0	When asserted, pauses the transfers for this channel.

**Table 80 • CHANNEL\_x\_CONTROL (continued)**

Bit Number	Name	Reset Value	Description
[3:2]	TRANSFER_SIZE	00	Determines the data width of each DMA transfer cycle for this DMA channel. The allowed values are: 00: Byte (8 bits) 01: Halfword (16 bits) 10: Word (32 bits) 11: Reserved
1	DIR	0	If PERIPHERAL_DMA = 1, then this bit is valid. If so, then the values of this bit have the following meanings: 0: Peripheral to memory 1: Memory to peripheral
0	PERIPHERAL_DMA	0	0: Channel is configured for memory to memory DMA. 1: Channel is configured for peripheral DMA. Based on the value of DIR, the peripheral ready signal associated with this DMA channel is interpreted as initiating transfers either from memory to the peripheral or vice-versa.

The following table provides the PERIPHERAL\_SEL bits description.

**Table 81 • PERIPHERAL\_SEL**

Bit 26	Bit 25	Bit 24	Bit 23	Function
0	1	0	0	From SPI_0 receive to any HPMS memory-mapped location
0	1	0	1	From any HPMS memory-mapped location to SPI_0 transmit
1	0	0	0	To/from FPGA fabric peripheral on FIC_0 interface (DMAREADY_0[1])
1	0	0	1	To/from FPGA fabric peripheral on FIC_0 interface (DMAREADY_0[0])
1	1	0	0	To/from FPGA fabric peripheral on FIC_1 interface (DMAREADY_1[1]).
1	1	0	1	To/from FPGA fabric peripheral on FIC_1 interface (DMAREADY_1[0]).
1	1	1	0	From COMM_BLK receive to any HPMS memory-mapped location
1	1	1	1	From any HPMS memory-mapped location to COMM_BLK transmit

#### 6.4.1.4 CHANNEL\_x\_STATUS Register Bit Definition

**Table 82 • CHANNEL\_x\_STATUS**

Bit Number	Name	Reset Value	Description
[31:3]	Reserved	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	BUF_SEL	0	0: Buffer A is used 1: Buffer B is used
1	CH_COMP_B	0	Asserts when this channel completes its DMA. Cleared by writing to CLR_COMP_B, bit[8] in CHANNEL_x_CONTROL register for this channel. If INTEN is set for this channel, the assertion of CH_COMP_B causes PDMAINTERRUPT to assert.

**Table 82 • CHANNEL\_x\_STATUS (continued)**

Bit Number	Name	Reset Value	Description
0	CH_COMP_A	0	Asserts when this channel completes its DMA. Cleared by writing to CLR_COMP_A, bit[8] in CHANNEL_x_CONTROL register for this channel. If INTEN is set for this channel, the assertion of CH_COMP_A causes PDMAINTERRUPT to assert.

#### 6.4.1.5 CHANNEL\_x\_BUFFER\_A\_SRC\_ADDR Register Bit Definition

**Table 83 • CHANNEL\_x\_BUFFER\_A\_SRC\_ADDR**

Bit Number	Name	Reset Value	Description
[31:0]	BUF_A_SRC	0	Start address from which data is to be read during the next DMA transfer cycle. If PERIPHERAL_DMA = 1 and DIR = 0, this value is not incremented from one DMA transfer cycle to the next. Otherwise, it is always incremented by an amount corresponding to the TRANSFER_SIZE for this channel.

#### 6.4.1.6 CHANNEL\_x\_BUFFER\_A\_DST\_ADDR Register Bit Definition

**Table 84 • CHANNEL\_x\_BUFFER\_A\_DST\_ADDR**

Bit Number	Name	Reset Value	Description
[31:0]	BUF_A_DST	0	Start address from which data is to be write during the next DMA transfer cycle. If PERIPHERAL_DMA = 1 and DIR = 1, this value is not incremented from one DMA transfer cycle to the next. Otherwise, it is always incremented by an amount corresponding to the TRANSFER_SIZE for this channel.

#### 6.4.1.7 CHANNEL\_x\_BUFFER\_A\_TRANSFER\_COUNT Register Bit Definition

**Table 85 • CHANNEL\_x\_BUFFER\_A\_TRANSFER\_COUNT**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	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]	BUF_A_COUNT	0	Number of remaining transfers to be completed between source and destination for buffer A for this channel. This field is decremented after every DMA transfer cycle. Writing a non-zero value to this register causes the DMA to start. This must be the last register written by firmware when setting up a DMA transfer.

### 6.4.1.8 CHANNEL\_x\_BUFFER\_B\_SRC\_ADDR Register Bit Definition

**Table 86 • CHANNEL\_x\_BUFFER\_B\_SRC\_ADDR**

Bit Number	Name	Reset Value	Description
[31:0]	BUF_B_SRC	0	Start address from which data is to be read during the next DMA transfer cycle. If PERIPHERAL_DMA = 1 and DIR = 0, this value is not incremented from one DMA transfer cycle to the next. Otherwise, it is always incremented by an amount corresponding to the TRANSFER_SIZE for this channel.

### 6.4.1.9 CHANNEL\_x\_BUFFER\_B\_DST\_ADDR Register Bit Definition

**Table 87 • CHANNEL\_x\_BUFFER\_B\_DST\_ADDR**

Bit Number	Name	Reset Value	Description
[31:0]	BUF_B_DST	0	Start address from which data is to be write during the next DMA transfer cycle. If PERIPHERAL_DMA = 1 and DIR = 1, this value is not incremented from one DMA transfer cycle to the next. Otherwise, it is always incremented by an amount corresponding to the TRANSFER_SIZE for this channel.

### 6.4.1.10 CHANNEL\_x\_BUFFER\_B\_TRANSFER\_COUNT Register Bit Definition

**Table 88 • CHANNEL\_x\_BUFFER\_B\_TRANSFER\_COUNT**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	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]	BUF_B_COUNT	0	Number of remaining transfers to be completed between source and destination for buffer B for this channel. This field is decremented after every DMA transfer cycle. Writing a non-zero value to this register causes the DMA to start. This must be the last register to be written by firmware when setting up a DMA transfer.

## 6.5 SYSREG Control Registers

In addition to the specific PDMA registers found in [Table 77](#), page 120, the registers found in the following table also control the behavior of the PDMA peripheral. These registers are located in the SYSREG section of the user's guide and are listed here for convenience. Refer to [System Register Block](#), page 196 for a detailed description of each register and associated bits.

**Table 89 • SYSREG Control Registers**

Register Name	Register Type	Flash Write Protect	Reset Source	Description
SOFT_RESET_CR	RW-P	Bit	SYSRESET_N	Soft reset control. For more information, see <a href="#">Table 143</a> , page 214.
MASTER_WEIGHT1_CR	RW-P	Register	SYSRESET_N	Configures weighted round robin master arbitration scheme for masters. For more information, see <a href="#">Table 140</a> , page 213.

## 7 Serial Peripheral Interface Controller

Serial peripheral interface (SPI) is a synchronous serial data protocol that enables the FPGA fabric logic and peripheral devices to communicate with each other. The SPI controller is an APB slave in the IGL002 device that provides a serial interface compliant with the Motorola SPI, Texas Instruments synchronous serial, and National Semiconductor MICROWIRE™ formats. In addition, SPI supports interfacing with large SPI flash and EEPROM devices and a hardware-based slave protocol engine.

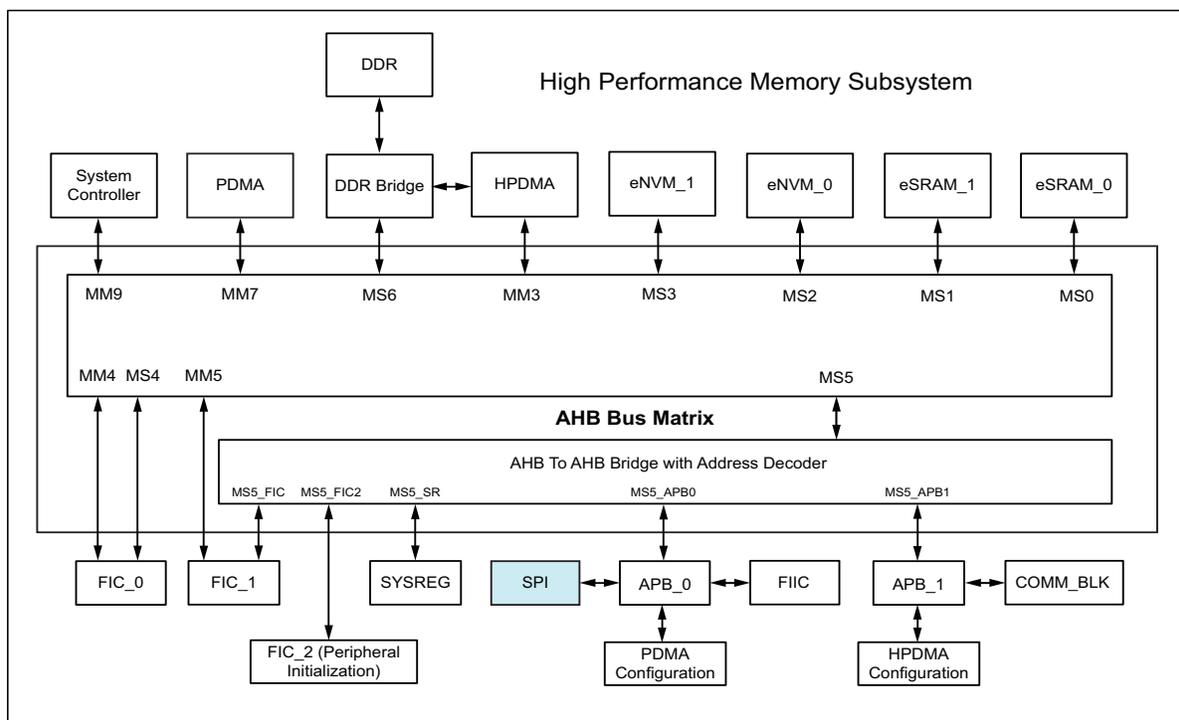
### 7.1 Features

IGLOO2 SPI peripheral supports the following features:

- Master and slave modes
- Configurable slave select operation
- Configurable clock polarity
- Separate transmit (Tx) and receive (Rx) FIFOs to reduce interrupt service loading
- Fabric logic controlled and PDMA controlled mode of data transfer

The following figure shows details of the high performance memory subsystem (HPMS). The SPI peripheral is interfaced to the AHB bus matrix through the APB\_0 interface.

**Figure 76 • Microcontroller Subsystem Showing SPI Peripherals**



## 7.2 Functional Description

This section provides the detailed description of the following SPI peripherals.

### 7.2.1 Architecture Overview

The SPI controller supports master and slave modes of an operation.

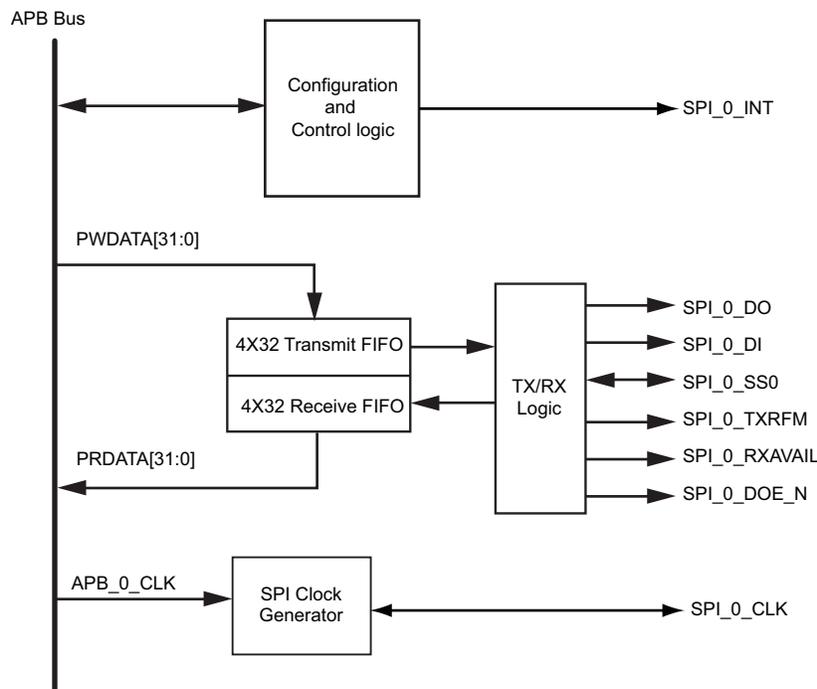
- In master mode, the SPI generates SPI\_0\_CLK, selects a slave using SPI\_0\_SS0, transmits the data on SPI\_0\_DO, and receives the data on SPI\_0\_DI.
- In slave mode, the SPI is selected by SPI\_0\_SS0. The SPI receives a clock on SPI\_0\_CLK and incoming data on SPI\_0\_DI.

The SPI peripheral consists mainly of the following components.

- Transmit and receive FIFOs
- Configuration and control logic
- SPI clock generator

The following figure shows the SPI controller block diagram.

**Figure 77 • SPI Controller Block Diagram**



**Note:**

- The SPI\_0\_DO, SPI\_0\_DI, SPI\_0\_SS0, and SPI\_0\_CLK signals are available to the FPGA fabric.
- SPI\_0\_DOE\_N is accessible through the SPI control register.
- SPI\_0\_TXRFM and SPI\_0\_RXAVAIL signals are used only by PDMA.
- SPI\_0\_INT can be routed to the FPGA fabric.

#### 7.2.1.1 Transmit and Receive FIFOs

The SPI controller embeds two 4 × 32 (depth × width) FIFOs for receive and transmit, as shown in the preceding figure. These FIFOs are accessible through RX data and TX data registers (refer to [SPI Register Details](#), page 151). Writing to the TX data register causes the data to be written to the transmit FIFO. This is emptied by the transmit logic. Similarly, reading from the RX data register causes the data to be read from the receive FIFO. The not-empty port of the receive FIFO and the not-full port of the transmit FIFO flags (of the FIFOs) are exposed as SPI\_0\_RXAVAIL (SPI has data to be read) and SPI\_0\_TXRFM (SPI has room for more data to send) ports. These are connected to the peripheral DMA

(PDMA) engine to allow continuous DMA streaming for large SPI transfers and to help free up the Fabric master.

### 7.2.1.2 Configuration and Control Logic

The SPI peripheral can be configured for master or slave mode by using the mode bit of the SPI CONTROL register (see Table 97, page 151). The type of data transfer protocol can be configured by using the TRANSFPRTL bit of the SPI CONTROL register. The control logic monitors the number of data frames to be sent/received and enables the interrupts when the data frame transmission/reception is completed. During data frames transmission/reception, if a transmit under-run error/ receive overflow error is detected, the STATUS register (see Table 99, page 152) is updated. Refer to the STAT8 register (Table 113, page 160) for bit definitions.

### 7.2.1.3 SPI Clock Generator

In master mode, the SPI clock generator generates the serial programmable clock from the APB clock. Refer to SPI Clock Requirements, page 143 for more details.

## 7.2.2 Interface

This section provides the details of the SPI interfacing ports and various data transfer protocols.

### 7.2.2.1 Port List

The following table lists the SPI signals.

**Table 90 • SPI Interface Signals**

Name	Type	Polarity/Bus Size	Description
SPI_0_DI	Input	High	Serial data input
SPI_0_DO	Output	High	Serial data output
SPI_0_CLK	Input/Output	High	Serial clock. It is a serial programmable bit rate clock out signal. Input when SPI is in the slave mode. Output when SPI is in the master mode.
SPI_0_SS0	Input/Output	Low, except for TI mode	Slave select. Input when SPI is in the slave mode. Output when SPI is in the master mode. The slave select output polarity is active Low. In TI mode the slave select output is inverted to become active High.
SPI_0_INT	Output	High	SPI interrupt
SPI_0_DOE_N	Output	High	Output enable
SPI_0_TXRFM	Output	High	SPI ready to transmit. Used only by HPMS PDMA engine
SPI_0_RXAVAIL	Output	High	SPI received data available. Used only by HPMS PDMA engine.

### 7.2.2.2 Data Transfer Protocol Details

The IGLOO2 SPI controller supports the following data transfer protocols:

- Motorola SPI Protocol
- National Semiconductor MICROWIRE Protocol
- Texas Instruments Synchronous Serial Protocol
- Slave Protocol Engine

The following sections describe the data transfer protocols, timing diagrams, signal requirements, and error case scenarios for the above protocols.

### 7.2.2.3 Motorola SPI Protocol

The Motorola SPI is a full duplex, four-wire synchronous transfer protocol which supports programmable clock polarity (SPO) and clock phase (SPH). The state of SPO and SPH control bits decides the data transfer modes as shown in the following table.

**Table 91 • Data Transfer Modes**

Data Transfer Mode	SPO	SPH
Mode 0	0	0
Mode 1	0	1
Mode 2	1	0
Mode 3	1	1

The SPH control bit determines the clock edge that captures the data.

- When SPH is Low, data is captured on the first clock transition.
  - Data is captured on the rising edge of SPI\_CLK when SPO = 0 (Figure 78, page 134).
  - Data is captured on the falling edge of SPI\_CLK when SPO = 1 (Figure 81, page 135).
- When SPH is High, data is captured on the second clock transition (rising edge if SPO = 1).
  - Data is captured on the falling edge of SPI\_CLK when SPO = 0 (Figure 80, page 135).
  - Data is captured on the rising edge of SPI\_CLK when SPO = 1 (Figure 82, page 135).

The SPO control bit determines the polarity of the clock and SPS defines the slave select behavior.

- When SPO is Low and no data is transferred, SPI\_CLK is driven to Low (Figure 79, page 134).
- When SPO is High and no data is transferred, SPI\_CLK is driven to High (Figure 78, page 134).

The following table summarizes the clock active edges in various SPI master modes.

**Table 92 • Summary of Master SPI Modes**

Mode	SPS	SPO	SPH	Clock in Idle	Sample Edge	Shift Edge	Select in Idle	Select Between Frames
Motorola	0	0	0	Low	Rising	Falling	High	Pulses between all frames
	0	1	0	High	Falling	Rising	High	
	0	0	1	Low	Falling	Rising	High	Does not pulse between back-to-back frames. Pulses if transmit FIFO empties.
	0	1	1	High	Rising	Falling	High	Does not pulse between back-to-back frames. Pulses if transmit FIFO empties.
	1	0	0	Low	Rising	Falling	High	Stays active until all the frames set by frame counter are transmitted.
	1	0	1	Low	Falling	Rising	High	
	1	1	0	High	Falling	Rising	High	
	1	1	1	High	Rising	Falling	High	
Texas Instruments	0	0	0	Low	Falling	Rising	Low	Normal operation SPI_0_CLK only generated with select and data bits.
			1	Low	Falling	Rising	Low	Removes SPI_0_SS0 on consecutive frames (back-to-back), making them appear to be big frames.
		1		Running	Falling	Rising	Low	SPI_0_CLK is free running.

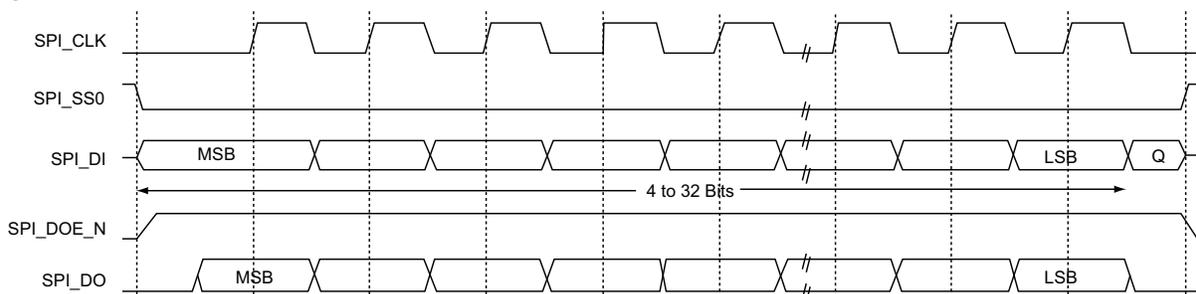
**Table 92 • Summary of Master SPI Modes (continued)**

Mode	SPS	SPO	SPH	Clock in Idle	Sample Edge	Shift Edge	Select in Idle	Select Between Frames
National Semiconductor Microwire	0	0	0	Low	Rising	Falling	High	Normal operation SPI_0_CLK only generated with select and data bits.
			1	Low	Rising	Falling	High	Forces IDLE cycles (SPI_0_SS0 deactivated) between back-to-back frames.
	1			Running	Rising	Falling	High	SPI_0_CLK is free running.
	1			Low	Rising	Falling	High	After sending the command part of the frame, the subsequent frames are concatenated to create a single large data frame (master operation only).

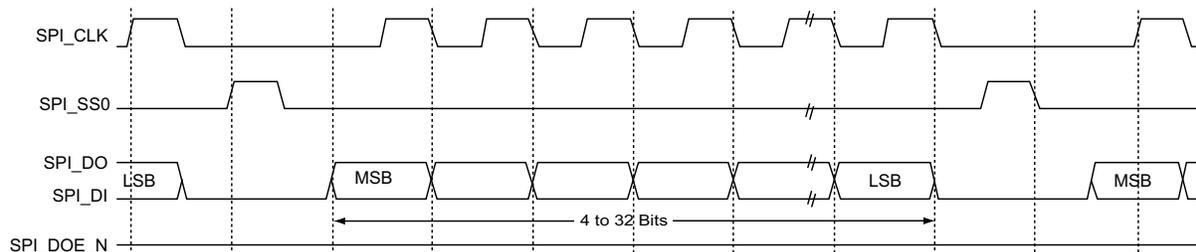
### 7.2.2.3.1 Motorola SPI Modes

Motorola SPI modes are shown in the following figures.

#### Single Frame Transfer – Mode 0: SPO = 0, SPH = 0

**Figure 78 • Motorola SPI Mode 0**

#### Multiple Frame Transfer – Mode 0: SPO = 0, SPH = 0

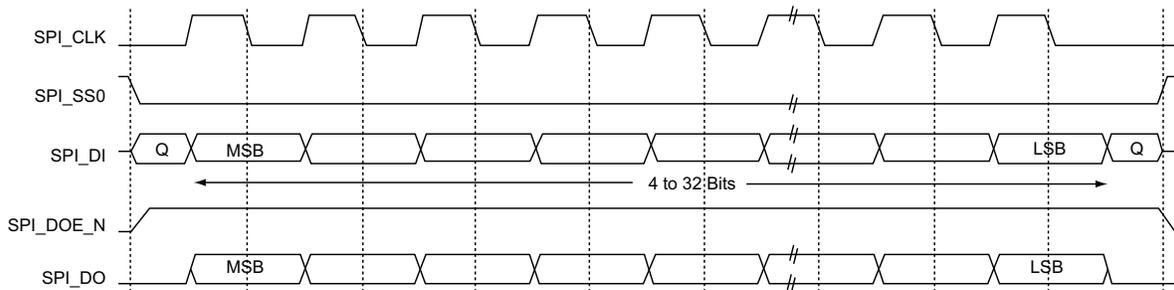
**Figure 79 • Motorola SPI Mode 0 Multiple Frame Transfer**

#### Notes:

- Between frames, the slave select (SPI\_SS0) signal is asserted for the duration of the clock pulse.
- Between frames, the clock (SPI\_CLK) is Low.
- Data is transferred to most significant bit (MSB) first.
- The output enable (SPI\_DOE\_N) signal is asserted during the transmission and deasserted at the end of the transfer (after the last frame is sent).

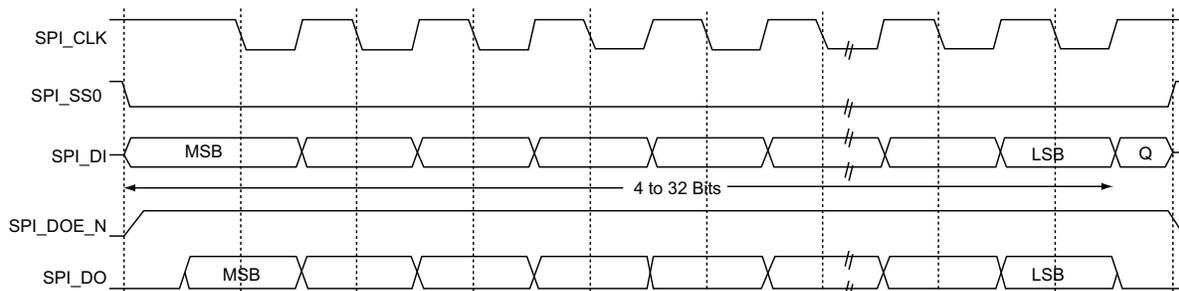
**Single Frame Transfer – Mode 1: SPO = 0, SPH = 1**

**Figure 80 • Motorola SPI Mode 1**



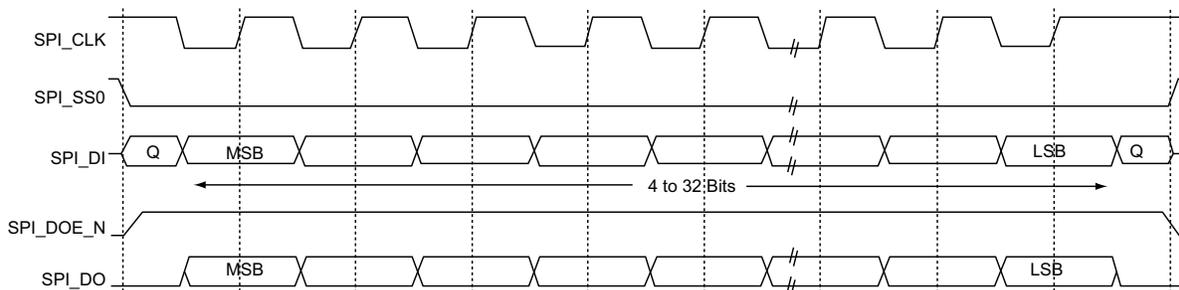
**Single Frame Transfer – Mode 2: SPO = 1, SPH = 0**

**Figure 81 • Motorola SPI Mode 2**



**Single Frame Transfer – Mode 3: SPO = 1, SPH = 1**

**Figure 82 • Motorola SPI Mode 3**



### 7.2.2.3.2 Output Enable (SPI\_0\_DOE\_N) Timing

Each SPI mode comprises two phases: transmit and receive. It is a requirement that the output enable (SPI\_0\_DOE\_N) line, which enables the output signal, should be driven so that the following occurs:

- The output signal is ready to transmit when the data is available (setup time).
- The output signal is held on long enough for the recipient to sample the data (hold time).

The minimum setup and hold time is one half SPI\_0\_CLK. In slave mode, the input clock is withdrawn at the end of the transfer. For example, consider the waveform for Mode 2 ([Single Frame Transfer – Mode 2: SPO = 1, SPH = 0](#), page 135). In this case, data is sampled on the falling edge of the clock and shifted on the rising edge of the clock. The data is sampled on the falling edge and must be held for one half SPI\_0\_CLK after the last falling edge at the end of the transmission. This means that SPI\_0\_DOE\_N must be held High for at least one half SPI\_0\_CLK after the last falling edge to satisfy the hold time requirement.

**Table 93 • Behavior of the Output Enable Signal**

Mode	Master	Slave
MOTOROLA	SPI_0_DOE_N is asserted with identical timing to that of SPI_0_SS0. This provides an additional half SPI_0_CLK cycle of data turn on and off relative to the data bit valid requirements.	The incoming SPI__SS0 signal is used to directly generate the SPI_0_DOE_N. Similar to the master case, it provides an additional half clock cycle of data turn on and off.
Texas Instruments	SPI_0_DOE_N is asserted on the negative clock edge prior to the MSB (while SPI_0_SS0 is asserted) and if the uninterrupted data is deasserted on the falling SPI_0_CLK edge following the LSB. This provides half a clock cycle of data turn on off time.	SPI_0_DOE_N is asserted on the positive SPI clock edge as the MSB is the output. SPI_0_DOE_N is deasserted on the positive SPI clock edge at the end of the LSB data bit, assuming no consecutive data.
National Semiconductor MICROWIRE	SPI_0_DOE_N is asserted with SPI_0_SS0, and then removed at the start of the ninth data bit (turn around cycle).	SPI_0_DOE_N is asserted at the start of the tenth bit as data becomes valid. SPI_0_DOE_N is deasserted at the end of the LSB, if a falling clock edge occurs or when SPI_0_SS0 is deasserted.

### 7.2.2.3.3 Motorola SPI Error Case Scenarios

The SPI protocol does not specify any error recovery strategy. The master and slave require prior knowledge of clock rates and data-frame layouts. However, there are built-in mechanisms in the SPI controller to recover from error. If the slave encounters an error, the master can toggle the slave clock until it comes to a known state. Here are three specific scenarios and error the behavior of the SPI controller in Motorola protocol mode.

- If the slave select signal is withdrawn in the middle of a transfer, the transfer continues until the end of the data frame.
- If the input clock is withdrawn, the SPI controller remains paused until the clock is restarted. It picks-up where it left off.
- If the slave select signal is withdrawn before a transfer occurs, the slave remains in the idle state (no data transfer having been initiated).

The SPI controller has no built-in timer. For applications where there is a possibility of a slave going to sleep for a long time, or in the case of very long transfers, the application should use a timer created from user logic.

### 7.2.2.3.4 SPI Data Transfer for Large Flash/EEPROM Devices in Motorola SPI Modes

Serial flash and EEPROM devices can be driven using Motorola SPI modes. Following is an outline of the interfaces to the required flash/EEPROM devices that shows how they can be driven using Motorola SPI modes. In each of these modes, the SPI controller is configured as a master with the slave select line connected to the chip select of the memory device.

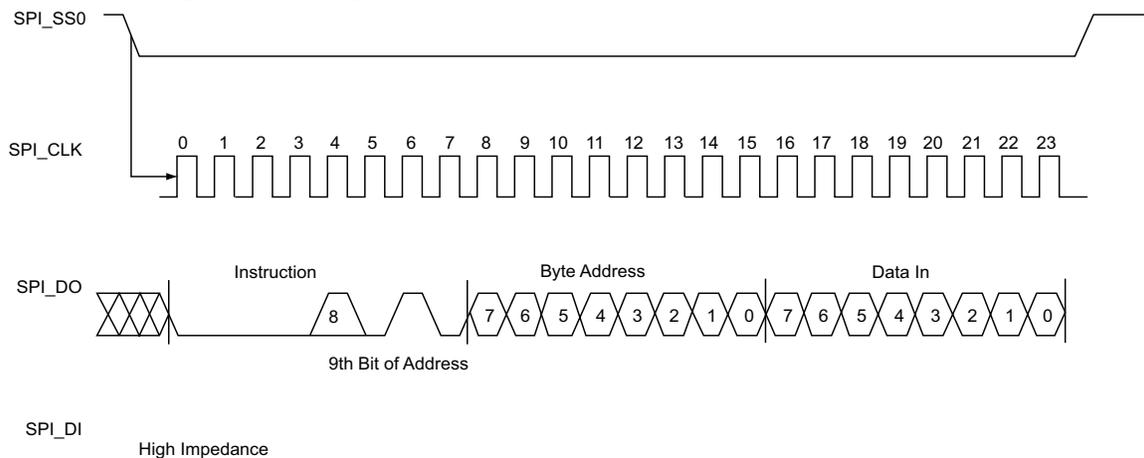
### 7.2.2.3.5 Devices Requiring Data Frame Sizes of Up to 32 Bits

Serial flash/EEPROM devices, such as the Atmel 25010/020/040, have a data frame size smaller than 32 bits and can be directly driven from SPI mode.

### 7.2.2.3.6 Write Operation for Atmel 25010/20/40 Devices

The following figure shows the write operation timing for Atmel 25010/020/040 devices. The SPI controller selects the devices using the slave select signal. The data frame size is set to 24 bits. The SPI is configured with SPO = 0, SPH = 0. The first byte is the instruction. Bit 5 of the instruction is part of the address (the 9th bit as required by the Atmel part). Bits 8-15 form a byte address. The residual 8 bits correspond to the data to be written.

**Figure 83 • Write Operation Timing**

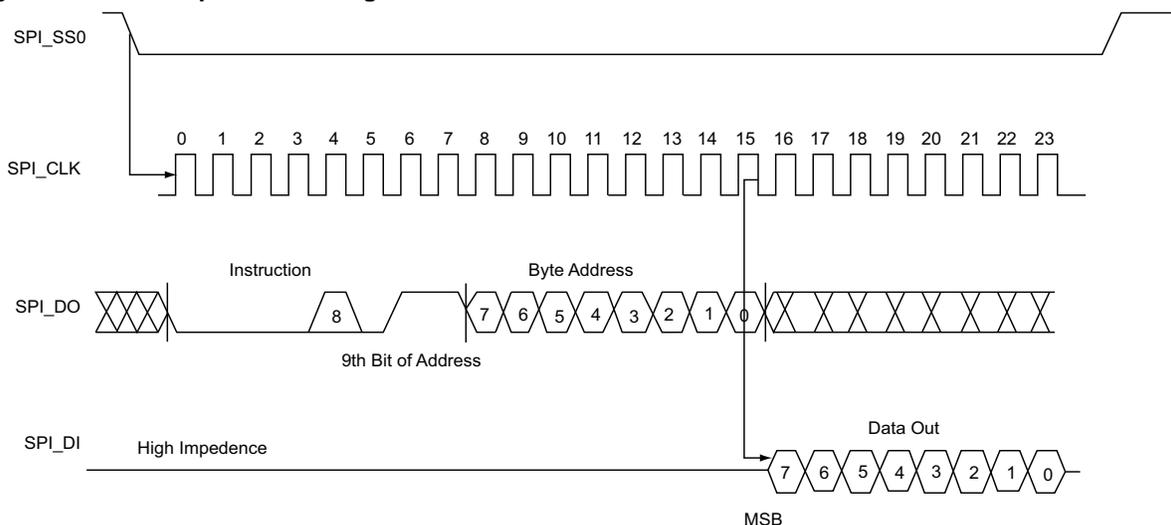


**Note:** The first byte contains the opcode that defines the operations to be performed. The opcode also contains address bit A8 in both the READ and WRITE instructions. This is mandated by the Atmel device.

### 7.2.2.3.7 Read Operation for Atmel 25010/20/40 Devices

The following figure shows the read operation timing for Atmel 25010/20/40 devices. For the read operation, the data frame size is set to 24 bits and the SPI controller is configured with SPO = 0, SPH = 0. On completing, the least significant byte of the received data frame corresponds to the data read.

**Figure 84 • Read Operation Timing**



**Note:** The first byte contains the opcode that defines the operations to be performed. The opcode also contains address bit A8 in both the read and write instructions. This is mandated by the Atmel device.

### 7.2.2.3.8 Devices Requiring Data Frame Sizes of More than 32 Bits

Serial flash devices such as the Atmel AT25DF321 which support mode 3 (SPO = 1 and SPH = 1) require more than 32 bits of frame data in some modes. To drive these devices, continuous transfers are required from the SPI interface while holding the slave select low continuously (which is connected to the chip select of the target device). This is accomplished by using the transmit FIFO from the SPI, which enforces continuous back-to-back transfers, if it is not empty. The slave select continues to be held low (active) in SPI mode 3 (SPO = 1 and SPH = 1) and not pulsed between data frames.

For example, to send 64 bits to the AT25DF321 (8-bit opcode, 24-bit address, 4 data bytes), the data frame size (see [Table 98](#), page 152) can be set to 32 and the data frame count (TXRXDFCOUNT, see [Table 97](#), page 151) set to two.

### 7.2.2.3.9 TXRXDFCOUNT Register

The SPI peripheral contains a TXRXDFCOUNT counter (found in the CONTROL register) that counts the number of transmitted and received frames. Its function varies in master and slave modes.

TXRXDFCOUNT in master mode controls the following:

- The Tx and Rx done interrupts
- Terminates the auto fill and empty operations
- Holds slave select active

TXRXDFCOUNT in slave mode controls the following:

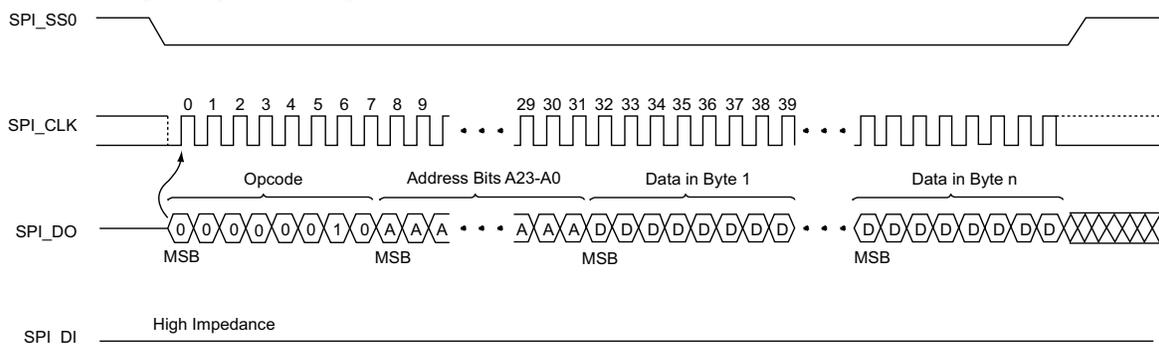
- The Tx and Rx done interrupts
- Terminates the auto fill operation

In slave operation it is possible for TXRXDFCOUNT to miscount actual transmitted and received frames if the transmit FIFO under-run condition occurs. If this is likely in an application, Microsemi recommends that TXRXDFCOUNT not be used and that it be disabled. Instead use the CMDINT and SSEND bits in the raw interrupt status (RIS) register (see [Table 107](#), page 156) to monitor operation, or simply count how many frames it is received.

### 7.2.2.3.10 Page Program for Atmel AT25DF321

The following figure shows the Page Programming Timing for Atmel AT25DF321. In this mode, the opcode, address, and data require more than 32 clock periods. To drive this device, the chip select (CS) can be connected to the slave select signal, the data frame size set to 16, and the FIFO repeatedly filled until the target flash device is programmed. As long as the data is available to transmit in the FIFO, the chip select signal (connected to slave select on the SPI controller) will be asserted Low.

**Figure 85 • Page Program Timing**



### 7.2.2.3.11 Devices That Do Not Support Mode 1 (SPO = 0 and SPH = 1) or Mode 3 (SPO = 1 and SPH = 1)

For flash devices that do not support mode 1 (SPO = 0 and SPH = 1) or mode 3 (SPO = 1 and SPH = 1), it is necessary to use a dedicated GPIO pin to drive the chip select signal.

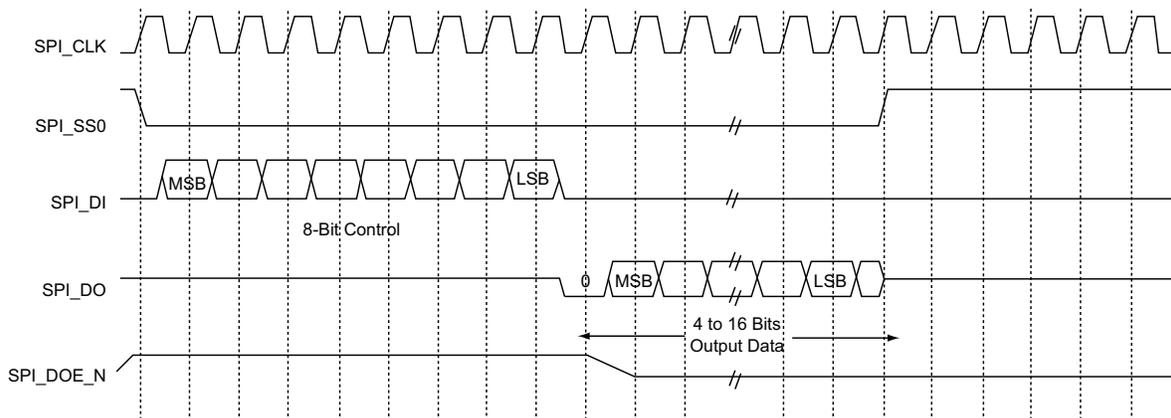
## 7.2.2.4 National Semiconductor MICROWIRE Protocol

The National Semiconductor MICROWIRE serial interface is a half-duplex protocol using a master/slave message passing technique. Each serial transmission begins with an 8-bit control word, during which time no incoming data is received. After the control word is sent, the external slave decodes it, and after waiting one serial clock cycle from the end of the control word, responds with the required data, which may be 4 to 16 bits in length.

### 7.2.2.4.1 Single Frame Transfer

In single frame transfer mode shown in the following figure, the most significant byte of the FIFO transmit word is the control byte. The total data frame size supplied must be at least 12 bits long (8 bits for the control word and a minimum of 4 bits for data payload). Only the output data is sampled and inserted in the receive FIFO.

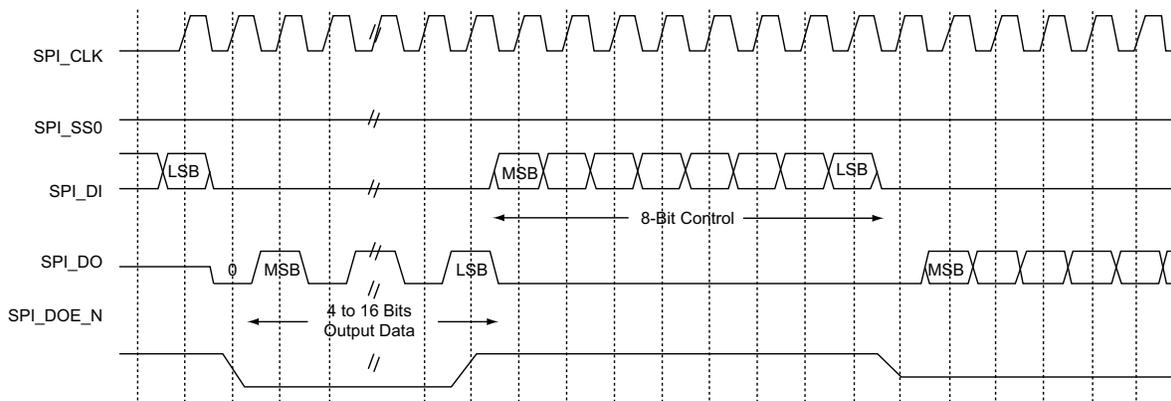
**Figure 86 • National Semiconductor MICROWAVE Single Frame Transfer**



### 7.2.2.4.2 Multiple Frame Transfer

In the multiple frame transfer shown in the following figure, the slave select signal (SPI\_0\_SS0) is continuously asserted (held Low) while SPI\_0\_DOE\_N (output Enable) is also asserted (or held Low) for the duration of each control byte. The other data transfers proceed in back-to-back manner.

**Figure 87 • National Semiconductor MICROWIRE Multiple Frame Transfer**



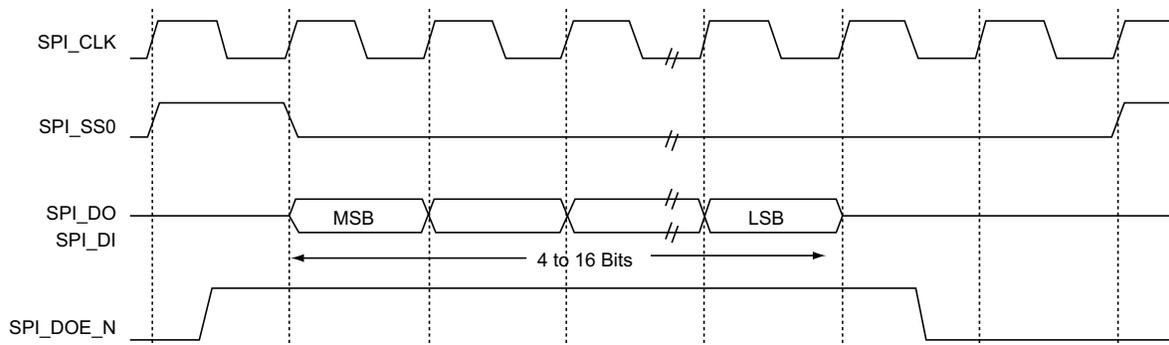
### 7.2.2.5 Texas Instruments Synchronous Serial Protocol

The Texas Instruments (TI) synchronous serial interface is based on a full duplex, four-wire synchronous transfer protocol. The transmit data pin is put in a high-impedance mode (tristated) when no data is transmitted.

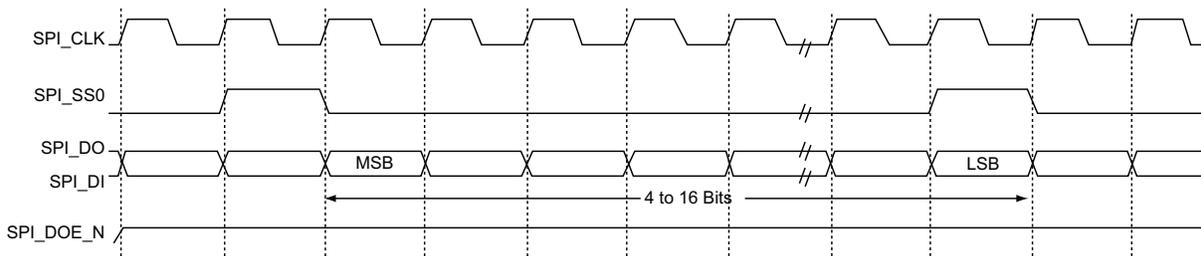
- The slave select (SPI\_0\_SS0) signal is pulsed between transfers to guarantee a high-to-low transition between each frame.
- The slave select output polarity is inverted to become active high. In an idle state, the slave select (SPI\_0\_SS0) signal is kept low.
- Data is available on the clock cycle immediately following the slave select (SPI\_0\_SS0) assertion.
- Both the SPI master and the SPI slave capture each data bit into their serial shift registers on the falling edge of the clock (SPI\_0\_CLK). The received data is latched on the rising edge of the clock (SPI\_0\_CLK).
- The output enable signal (SPI\_0\_DOE\_N) is asserted (active low) throughout the transfer.

The following figures show the TI synchronous single frame transfer and TI synchronous multiple frame transfer.

**Figure 88 • TI Synchronous Serial Single Frame Transfer**



**Figure 89 • TI Synchronous Serial Multiple Frame Transfer**



#### 7.2.2.5.1 TI Synchronous Serial Error Case Scenarios

When the SPI controller is configured for the TI synchronous serial protocol, while in slave mode, it responds to failure events. These failure events on slave select (SPI\_0\_SS0) and the slave clock (SPI\_0\_CLK) are described below:

- **Withdrawal of SPI\_0\_CLK:** In this case the device pauses and resumes on reasserting the clock.
- **Premature pulsing of slave select:** If the slave select is pulsed during a data frame transmission, it will be ignored.
- **Disconnecting the slave select before a transfer:** The transfer is not initiated unless the pulse is issued.

## 7.2.2.6 Slave Protocol Engine

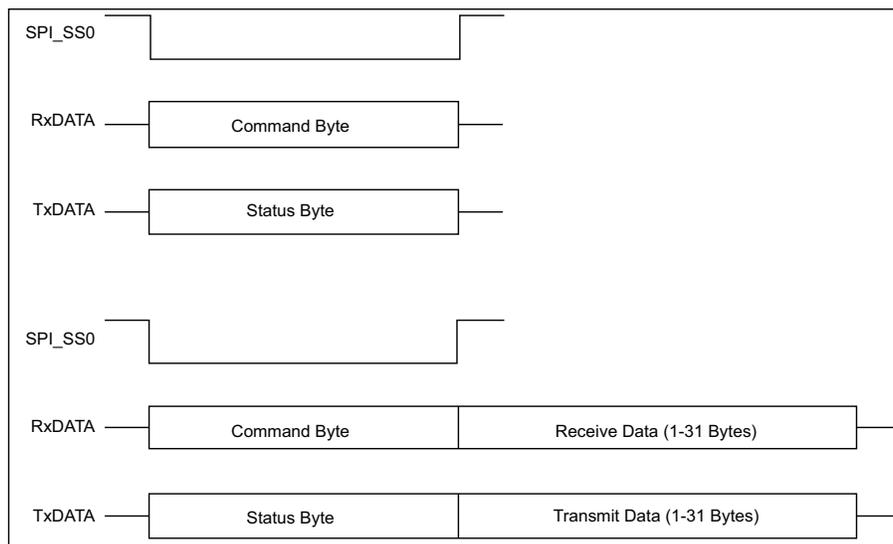
The slave protocol engine (SPE) implements a Microsemi-defined hardware protocol that allows the transfer of command and data from an SPI master to the SPI slave. The SPE controller logically sits between the SPI transmit/receive logic and the FIFOs. The SPE controller removes the command bytes and inserts status bytes from the data stream. Only one command byte is defined by Microsemi (POLL command). All other command bytes are user defined. To use the SPE, the BIGFIFO, AUTOSTATUS, AUTOPOLL, FRAMEURUN, and SPS bits should be set (refer to the SPI CONTROL register—Table 97, page 151—for bit definitions). The descriptions below assume that the frame size (TXRXDF\_SIZE[TXRXDFS] field) is set to 8 bits, although other frame size settings are acceptable (up to 32 bits).

### 7.2.2.6.1 SPI Slave Frame Format

The frame format consists of a command frame followed by 0 to 31 data frames. The size of the command frame and data frame must be equal and is defined by TXRXDFS. A typical use model would be to define the command frame as 8 bits followed by 31 bytes of data. This assumes BIGFIFO is set to 1 and TXRXDFS is set to 0x08.

The following figure shows the command and data bytes. Transmit and receive refer to the SPI peripheral as the slave. Data bytes are optional.

**Figure 90 • SPE Command/Data Format**



The first receive byte of the sequence after SPI\_0\_SS0 asserts is always a command byte. The slave always responds with a status byte, which is the contents of the HWSTATUS register (see Table 112, page 159).

**Note:** Set two bits in the HWSTATUS register to facilitate additional handshaking schemes between SPI master and SPI slave.

### 7.2.2.6.2 POLL Command

All command bytes except the POLL command are stored in the receive FIFO. Once received, the CMD interrupt is generated. The command size can be set by the CMD\_SIZE register (see Table 111, page 159) and can be 1 to 32 bits wide, although typically commands and data will be 8 bits wide. The POLL command is encoded as 0xFF and is the only encoded command. All other command byte encodings are user defined. If a POLL command is received, the contents of the HWSTATUS register are sent back to the master and the POLL command is discarded. It will not be stored in the FIFO.

### 7.2.2.6.3 Hardware Status Frame

A hardware status frame is automatically sent back by the SPE in response to every command. It provides status information back to the master. The byte contains the contents of the HWSTATUS register.

#### 7.2.2.6.4 Simple Commands

To send a command with no data to the slave, the master does the following:

1. Sends a POLL command and verifies that the slave is ready (no RXBUSY from HWSTATUS register).
2. This is repeated until the slave indicates it is ready.
3. The master sends the other command with no data. The command is queued in the receive FIFO for the slave to process.

#### 7.2.2.6.5 Data Receive Operation

To send data to the slave, the master does the following:

1. Sends a POLL command and verifies that the slave is ready and can accept the data.
2. This is repeated until the slave indicates that it is ready and can accept the data (no RXBUSY from HWSTATUS register).
3. The master sends the write command and data bytes. On receiving, the slave stores the command and data bytes in the receive FIFO. After CMDSIZE bits have been received, the CMD interrupt is generated.
4. The hardware automatically set RxBUSY, if there are less than PKTSIZE (see Table 110, page 159) storage locations left in the receive FIFO after the sequence completes.

**Note:** The slave reports under-run events having occurred, if no data is available for transmit.

#### 7.2.2.6.6 Data Transmit Operation

To receive data from the slave, the master does the following:

1. Sends a POLL command and verifies that the slave is ready and can accept the data.
2. This is repeated until the slave indicates it is ready and can accept any associated command data (no RxBUSY).
3. The master sends a read command and any associated data bytes (for example, a read address). On receiving the sequence, the slave stores the command byte and data in the receive FIFO. User logic examines the command and data bytes and puts the requested data in the transmit FIFO. As soon as it has written PKTSIZE bytes to the transmit FIFO, the TxBUSY status bit in the HWSTATUS register will be cleared.
4. The master starts polling the device until the TxBUSY bit is cleared, indicating that the data is available.
5. The master now sends a read command followed by data words. The slave will return the contents of the HWSTATUS register and required data words.

#### 7.2.2.6.7 Under-Run in Slave Mode

Under normal operating conditions, the SPI slave core in slave mode has a transmit FIFO under-run condition as the master initiates transfers when the slave transmit FIFO is empty (or attempts to transmit data faster than the slave processor loads data). The core's operation can be modified by setting FRAMEURUN (CONTROL register). Once set, the core will ignore the under-run conditions and simply transmit zero frames when the transmit FIFO is empty at the start of a series of frames. If the first data frame of a packet is read from the FIFO and transmitted, the under-run detection is enabled such that if the transmit FIFO fails to provide any of the rest of the data packet (assuming SPI\_0\_SS0 is active for the whole packet), an over-run condition is signaled.

### 7.2.3 Initialization

This section describes the SPI initialization sequence, the SPI status at reset, and clock requirements. The SPI can be initialized by configuring the SPI CONTROL register and the SOFT\_RESET\_CR system registry (see Table 143, page 214).

#### 7.2.3.1 Initialization Sequence

1. Select the type of transfer protocol by using the TRANSFPRTL bit of the SPI CONTROL register.
2. Enable SPI by writing '0' to the RESET bit of the CONTROL register.
3. Reset the transmit/ receive buffers and the data frame size.

### 7.2.3.2 SPI Status at Reset

After SPI reset, the slave select (SPI\_0\_SS0) pin is held to the default values of logic High. After selecting SPI mode and enabling the SPI controller, the SPI\_0\_SS0 line is changed to the default values for each protocol. Refer to [SPI Control Register \(CONTROL\)](#), page 151. After reset, the clock out (SPI\_0\_CLK) is at logic Low. At reset, the FIFOs are cleared and their respective read and write pointers are set to zero. Similarly, all the internal registers of the SPI controller are reset to their default values, as explained in [SPI Register Summary](#), page 150.

An option is provided to reset the SPI peripheral by writing to bit 9 in the system register, SOFT\_RESET\_CR. The soft resets are encoded in the following table. At power-up, the reset signals are asserted 1. It keeps the SPI peripheral in a reset state. The SPI peripheral becomes active when the bit is set to 0, as shown in the following table.

**Table 94 • Soft Reset Bit Definitions for SPI Peripheral**

Bit Number	Name	R/W	Reset Value	Description
9	SPI0_SOFTRESET	R/W	0x1	Controls reset input to SPI_0 0: Release SPI_0 from reset 1: Keep SPI_0 in reset

### 7.2.3.3 SPI Clock Requirements

The SPI\_0 peripherals clocked by APB\_0\_CLK on APB bus 0. APB\_0\_CLK is derived from the main HPMS clock, HPMS\_CLK. The APB clock can be programmed individually as HPMS\_CLK is divided by 1, 2, 4, or 8. Refer to the [UG0449: SmartFusion2 and IGLOO2 Clocking Resources User Guide](#) for more information.

The SPI clock in master mode is derived from APB\_0\_CLK. Master mode and slave mode SPI data rates depend on the APB clock, as given below.

- Master mode SPI data rate
  - Programmable from APB\_0\_CLK/256 to APB\_0\_CLK/2
  - Programmable from APB\_0\_CLK /65556 to APB\_0\_CLK /256 in powers of 2
  - Maximum data rate is APB\_0\_CLK/2
- Slave mode SPI data rate operates up to
  - APB\_0\_CLK for frame sizes (frame size  $\geq 8$ )
  - APB\_0\_CLK /2 for frame sizes (frame size 4 to 7)

## 7.2.4 Details of Operation

This section describes the SPI controller operation including FIFO, modes of data transfer, interrupts, and error handling.

### 7.2.4.1 SPI Transmit and Receive FIFO Flags

The SPI controller contains two, 4x32 (depth x width) FIFOs, as shown in [Figure 77](#), page 131. One is for the receive side and the other is for the transmit side. The TXFIFOFUL and TXFIFOEMP bits of the STATUS register (see [Table 99](#), page 152) indicate the full or empty status of the transmit FIFO. The RXFIFOFUL and RXFIFOEMP bits of the STATUS register indicate the full or empty status of the receive FIFO. User logic can poll these bits to obtain the status of the corresponding FIFO.

For large data transfers, the full depth of the transmit FIFO can be used by setting the number of data frames (more than one) in a burst (maximum is 64 k frames). When the interrupts are enabled, the TXDONE bit of the RIS register (see [Table 107](#), page 156) is asserted after all the data frames in the burst are sent.

For example, if the data frame size is set to 32 and the count is set to 2, the interrupt TXDONE is generated after every 2 words (each word is 32 bits). The default value for the frame count is one. The TXUNDERRUN and RXOVERFLOW bits of the STATUS register indicate that a FIFO under-run or FIFO overflow has occurred.

### 7.2.4.1.1 FIFO Under-Run Condition

If the transmit FIFO is accessed to transfer the data and there is no data in the FIFO, a transmit under-run error (TXUNDERRUN) is generated. This can be conditionally used to generate an interrupt. In this case, the transmission is assumed to have been lost and the application must catch the error and restart the transmission from the beginning. Internally, the transmit logic returns to an idle state and the entire transmission is deemed lost.

### 7.2.4.1.2 FIFO Overflow Condition

If the channel attempts to write into a receive FIFO which is already full, a receive overflow error (RXOVERFLOW) is generated. This can be conditionally used to generate an interrupt. In this case, the transmission continues but the data is now corrupted because the data frame is missing. It is assumed that the software clears the interrupt and recover; possibly by reading from the receive FIFO to clear the source of the interrupt, allowing more data to be received, or even by halting the transmission and resetting the SPI controller.

## 7.2.4.2 SPI Controller Modes of Data Transfer

There are two basic modes of transfer.

- Fabric logic controlled mode: The data transfers are controlled by a fabric logic that either polls the STATUS register or responds to interrupts.
- PDMA controlled mode: The data transfers are autonomously controlled by the PDMA engine.

### 7.2.4.2.1 Fabric Logic Controlled Mode

In this mode, the size of the data frames (set in register TXRXDF\_SIZE) and their numbers (set in the CONTROL[TXRXDFCOUNT] field) are specified. The data frame size specifies the number of bits being shifted out or being received per-frame. On completing each transfer, after a specified number of data frames (1 by default) are sent, an optional interrupt is generated. The SPI controller keeps track of the number of data frames so that special signals, like output enable, can be deactivated at the end of a transfer.

For example, to transmit one 17-bit word, the data frame size is set to 17, and the number of data frames is set to 1. Then depending on the operating mode, the 17 bits are transferred and the TXDATSENT STATUS register bit (0) is set. If enabled, an interrupt is also generated.

For example, consider the transmission of 64 kb of data to an external EEPROM from the fabric logic controlled SPI controller. The data frame size is set to eight and the number of data frames per-transfer is set to one. After each transfer, the fabric master must respond to the interrupt-transmit done-and-reload the FIFO until the 64 kb of data is sent. To improve throughput, the number of data frames per each transfer can be set to 4, in order to utilize the full depth of the transmit FIFO.

### 7.2.4.2.2 PDMA Controlled Mode

In PDMA mode, the interrupts are turned off and the PDMA controller uses SPI\_0\_TXRFM and SPI\_0\_RXAVAIL signals to govern the filling and emptying of the FIFOs. The SPI\_0\_RXAVAIL signal indicates that the data is available to be read and SPI\_0\_TXRFM indicates that the transmit is done and it is ready to receive more data.

For example, consider the transmission of 64 kb of data to an external EEPROM from a PDMA controlled SPI controller. The data frame size is set to eight and the number of data frames per-transfer is set to one. The transmit FIFO is repeatedly filled and emptied by the PDMA engine, using the SPI\_0\_TXRFM and SPI\_0\_RXAVAIL signals. In PDMA mode, the transmit done and receive data available interrupts are masked, and the PDMA engine is used to notify the application on completion.

## 7.2.4.3 Interrupts

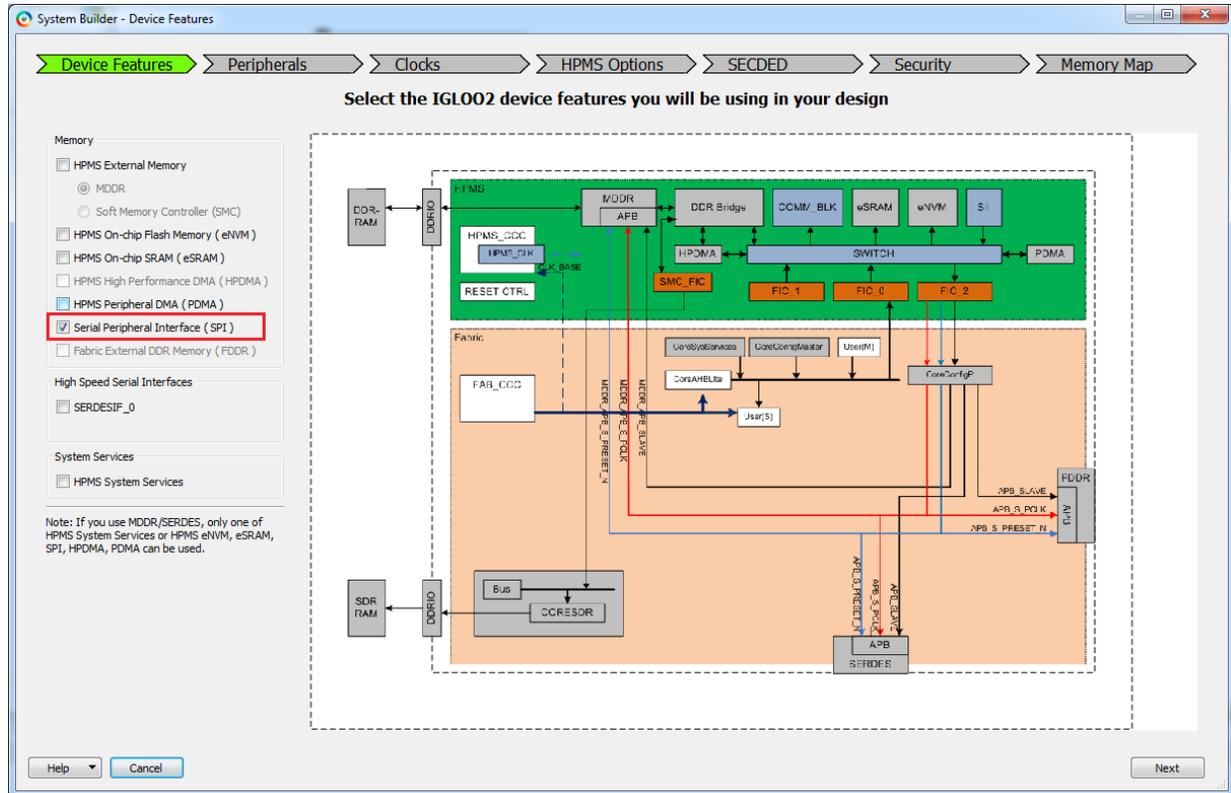
Interrupts can be set up to signal the completion of a data frame transmission or reception. There is one interrupt signal from SPI\_0 peripheral. The SPI\_0\_INT signal is generated by SPI\_0 and is handled by the fabric master. SPI\_0\_INT can be routed to the FPGA fabric through the FIIC.



The following steps describe how to enable SPI controller in an application using System Builder.

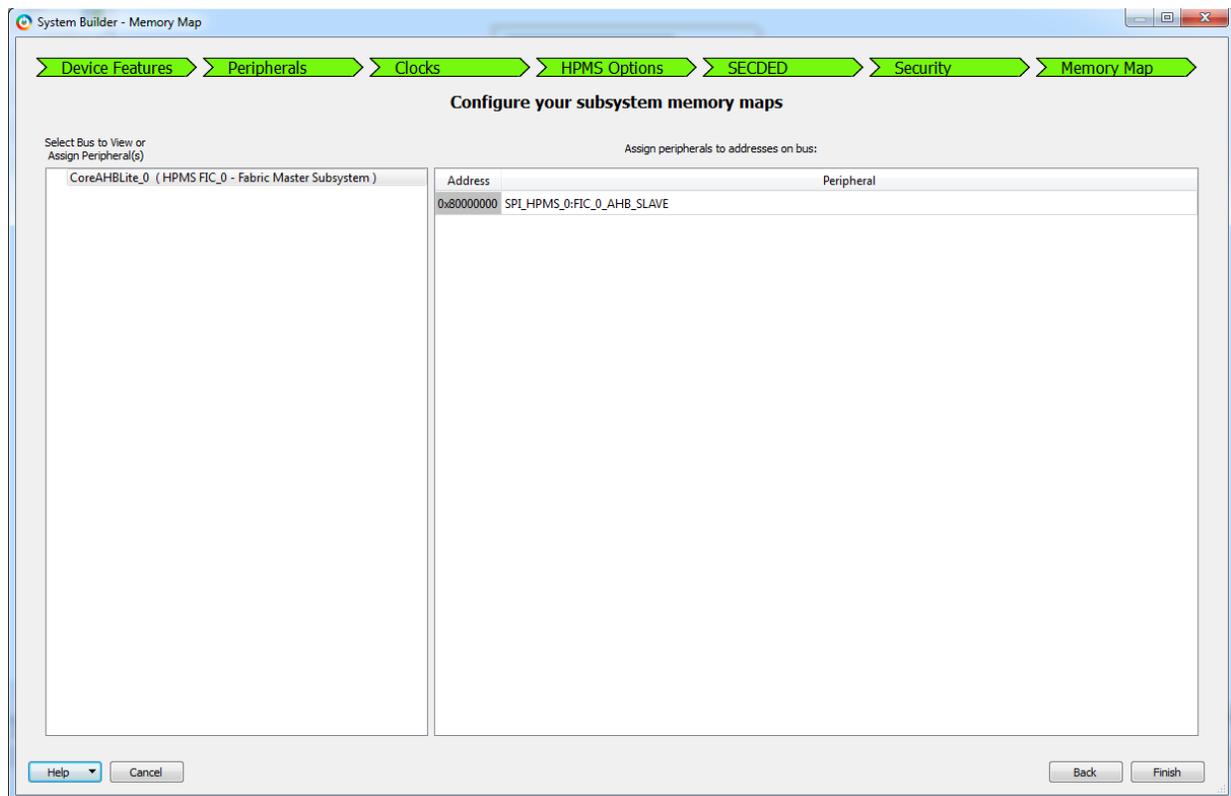
1. Check the **Serial Peripheral Interface (SPI)** box under the **Device Features** tab and leave the other check boxes unchecked. The following figure shows the **System Builder - Device Features** tab.

**Figure 92 • System Builder - Device Features Tab**



- Navigate to the **Memory Map** tab giving the required data in the rest of the System Builder tabs. The following figure shows the System Builder - Memory Map tab. Click **Finish** to proceed with creating the subsystem.

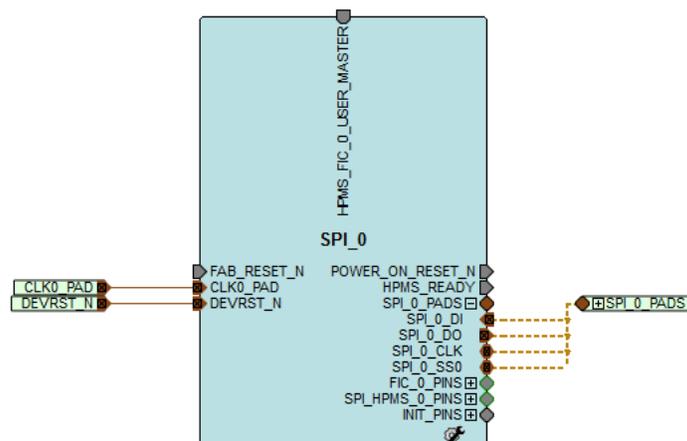
**Figure 93 • System Builder - Memory Map Tab**



### 7.3.1 HPMS Subsystem

The following figure shows an example HPMS subsystem with SPI controller enabled.

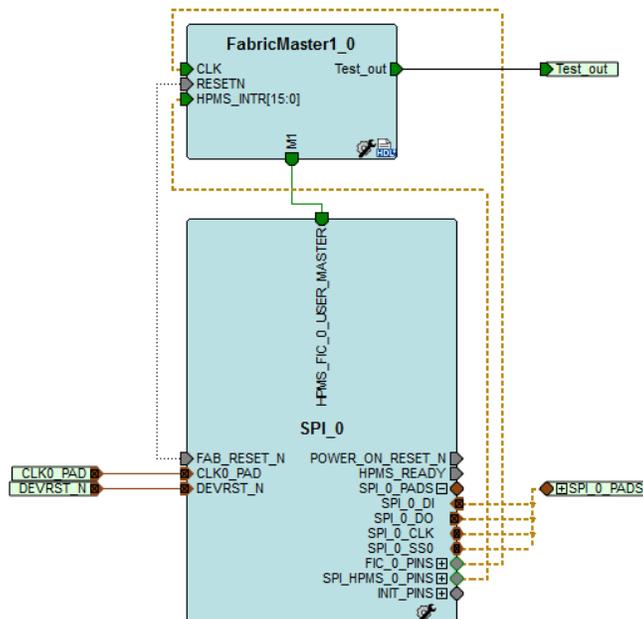
**Figure 94 • HPMS Subsystem**



### 7.3.2 HPMS Subsystem Connected to the FPGA Fabric Master

The following figure shows the FPGA fabric master connected to AHB master port.

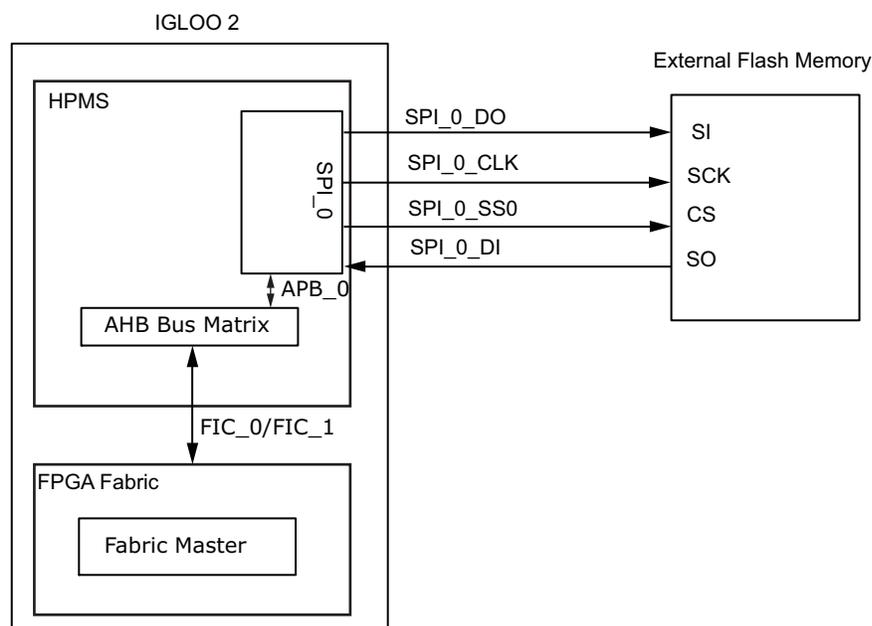
Figure 95 • HPMS Interconnection with FPGA Fabric Master



### 7.3.3 Accessing the External SPI Flash Using HPMS SPI\_0

The external SPI flash memory can be interfaced with the HPMS SPI\_0 of the IGLOO2 device. The HPMS SPI\_0 is configured as a master with the slave select line (SPI\_0\_SS0) connected to the chip select (CS) of the external SPI Flash. The following figure shows interfacing the external SPI flash to HPMS SPI\_0.

Figure 96 • Fabric Master Accessing the External SPI Flash Using HPMS SPI\_0



The following steps describe how to initialize and configure the SPI controller in master mode, write to and read from SPI flash.

### 7.3.3.1 Initializing and Configuring the SPI Controller in Master Mode

1. Configure the SPI Control Register (see [Table 97](#), page 151).
  - Enable the SPI controller.
  - Select FIFO depth, Clock mode, number of data frames to be sent or received and SPI master mode.
2. Configure SPI TxRx Data Frame Register (TXRXDF\_SIZE) with transmit and receive data size.
3. Configure SPI SCLK Generation Register to calculate the SPICLK divider.
4. Configure SPI Slave Select Register to specify the slave.

### 7.3.3.2 Writing to SPI Flash

1. Set the slave using SPI Slave Select Register.
2. Send Write Enable Command (0x06) to slave (SPI flash).
3. Send Unprotected Sector Opcode(0x39) with the SPI flash memory address to the slave.
4. Send Write Enable Command (0x06) to the slave.
5. Send program page command (0x02) with the target flash memory address to the slave.
6. Send the actual data to write to the SPI flash.
7. Reset the slave using SPI SLAVE\_SELECT Register (see [Table 105](#), page 155).

### 7.3.3.3 Reading from SPI Flash

1. Set the slave using SPI Slave Select Register.
2. Send Read Array Opcode (0x1B) with the SPI flash address to read the data from the SPI flash.

The following steps describe how to send data or command to the SPI flash for read/write operations:

1. Disable the ENABLE bit of SPI Control Register.
2. Set the frame count by writing to TXRXDFCOUNT of SPI CONTROL register.
3. Set the frame size TXRXDF\_SIZE to 8 bits.
4. Set the ENABLE bit of SPI CONTROL register.
5. Wait until the Rx FIFO is cleared by monitoring the SPI STATUS register and read the data from SPI RX\_DATA register (see [Table 101](#), page 154) if Rx FIFO is not empty.
6. Set the SPI TX\_DATA register (see [Table 101](#), page 154) with the data byte to send to the slave. Ensure that the Tx FIFO is not full by monitoring the SPI STATUS register to send the next byte of the data.
7. Repeat steps 5 and 6 until read and write transactions get completed depending upon the length of the data transfer.

**Note:** The HPMS SPI does not support full-behavioral simulation models.

## 7.4 SPI Register Map

This section provides SPI registers along with the address offset, functionality, and bit definitions.

### 7.4.1 SYSREG Configuration Register Summary

The registers in the following table control the behavior of the SPI peripheral. Refer to [System Register Block](#), page 196 for a detailed description of each register and bit.

**Table 95 • SYSREG Control Registers**

Register Name	Register Type	Flash Writer Protect	Reset Source	Description
SOFT_RESET_CR	RW-P	Bit	SYSRESET_N	Soft reset control. For more information, see <a href="#">Table 143</a> , page 214.
PERIPH_CLK_MUX_SEL_CR	RW-P	Register	PORESET_N	Peripheral clock MUX select. For more information, see <a href="#">Table 146</a> , page 216.

## 7.4.2 SPI Register Summary

The following table summarizes each of the SPI registers described in this document. The SPI\_0 base address resides at 0x40001000 and extends to address 0x40001FFF in the fabric memory map.

**Table 96 • SPI Register Summary**

Register Name	Address Offset	R/W	Reset Value	Description
CONTROL	0x00	R/W	0x80000102	Control Register
TXRXDF_SIZE	0x04	R/W	0x04	Transmit and receive data frame size
STATUS	0x08	R	0x2440	Status Register
INT_CLEAR	0x0C	W	0x0	Interrupt clear register
RX_DATA	0x10	R	0x0	Receive data register
TX_DATA	0x14	W	0x0	Transmit data register
CLK_GEN	0x18	R/W	0x07	Output clock generator (master mode)
SLAVE_SELECT	0x1C	R/W	0x0	Specifies slave selected (master mode)
MIS	0x20	R	0x0	Masked interrupt status
RIS	0x24	R	0x0	Raw interrupt status
CONTROL2	0x28	R/W	0x0	Control bits for enhanced modes
COMMAND	0x2C	R/W	0x0	Command register
PKTSIZE	0x30	R/W	0x0	Packet size
CMD_SIZE	0x34	R/W	0x0	Command size
HWSTATUS	0x38	R/W	0x0	Slave hardware status
STAT8	0x3C	R	0x44	Status Register
CTRL0	0x40	R/W	0x02	Aliased CONTROL register bits 7:0. This register allows byte operations from an 8-bit processor in the fabric.
CTRL1	0x44	R/W	0x01	Aliased CONTROL register bits 15:8. This register allows byte operations from an 8-bit processor in the fabric.
CTRL2	0x48	R/W	0x0	Aliased CONTROL register bits 23:16. This register allows byte operations from an 8-bit processor in the fabric.
CTRL3	0x4C	R/W	0x0	Aliased CONTROL register bits 25:24. This register allows byte operations from an 8-bit processor in the fabric.

## 7.4.3 SPI Register Details

This section describes the SPI registers in detail.

### 7.4.3.1 SPI Control Register (CONTROL)

The following table gives the details regarding the SPI Control Register. Using this register, the SPI mode (master/slave), the type of the protocol it uses, and the data frame count can be set.

**Table 97 • CONTROL**

Bit Number	Name	R/W	Reset Value	Description
31	RESET	R/W	1	0: SPI is enabled 1: SPI is held in Power reset state.
30	OENOFF	R/W	0	0: SPI output enable active as required 1: SPI output enable is not asserted. Allows multiple slaves to share a single slave select signal with a single master.
29	BIGFIFO	R/W	0	Alters FIFO depth when frame size is [4-8] bits. 0: FIFO depth is 4 frames. 1: FIFO depth is 32 frames when frame size is [9-16] bits FIFO depth is 16; and when frame size is [17-32] bits FIFO depth is 8.
28	CLKMODE	R/W	0	Specifies the methodology used to calculate the SPICLK divider. 0: $SPICLK = 1 / (2^{CLK\_GEN + 1})$ where $CLK\_GEN = 0$ to 15. 1: $SPICLK = 1 / (2 \times (CLK\_GEN + 1))$ where $CLK\_GEN = 0$ to 255.
27	FRAMEURUN	R/W	0	0: The under-runs are generated whenever a read is attempted from an empty transmit FIFO. 1: The under-run condition will be ignored for the complete frame, if the first data frame read resulted in a potential overflow; that is, the slave was not ready to transmit any data. If the first data frame is read from the FIFO and transmitted, an under-run will be generated, when the FIFO becomes empty for any of the remaining packet frames (that is, while SSEL is active). Master operation does not create a transmit FIFO under-run condition.
26	SPS	R/W	0	Defines slave select behavior. See <a href="#">Table 92</a> , page 133.
25	SPH	R/W	0	Clock phase
24	SPO	R/W	0	Clock polarity
[23:8]	TXRXDFCOUNT	R/W	0001	Number of data frames to be sent or received. Counts from 1. Maximum value is 0xFFFF.
7	INTTXTURUN	R/W	0	Interrupt on transmit the under-run 0: Interrupt disabled 1: Interrupt enabled
6	INTRXOVRFLO	R/W	0	Interrupt on receive overflow 0: Interrupt disable 1: Interrupt enabled
5	INTTXDATA	R/W	0	Interrupt on transmit data 0: Interrupt disabled 1: Interrupt enabled
4	INTRXDATA	R/W	0	Interrupt on receive data 0: Interrupt disabled 1: Interrupt enabled

**Table 97 • CONTROL (continued)**

Bit Number	Name	R/W	Reset Value	Description
[3:2]	TRANSFPRTL <sup>1</sup>	R/W	0	Transfer protocol Decode: 0b00: Motorola SPI 0b01: TI synchronous serial 0b10: National Semiconductor MICROWIRE 0b11: Reserved
1	MODE	R/W	1	SPI implementation 0: Slave 1: Master (default)
0	ENABLE	R/W	0	Core enable 0: Disable (default) 1: Enable The core will not respond to external signals (SPI_0_DI, SPI_0_DO) until this bit is enabled. SPI_0_CLK is driven low and SPI_0_DOE_N and SPI_0_SS (slave select) are driven inactive.

1. The transfer protocol cannot be changed while the SPI is enabled.

### 7.4.3.2 SPI TxRx Data Frame Register (TXRXDF\_SIZE)

The following table gives details regarding the Transmit Receive Data Frame register. The width of the data frame is set using this register.

**Table 98 • TXRXDF\_SIZE**

Bit Number	Name	R/W	Reset Value	Description
[31:6]	Reserved	R/W	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.
[5:0]	TXRXDFS <sup>1</sup>	R/W	0x04	Transmit and receive data size. Maximum value is 32. Number of bits shifted out and received per frame (count starts from 1). In National Semiconductor MICROWIRE mode, this is the number of shifts to be done after the control byte is sent.

1. This register must be set before SPI is enabled. Writes to this register are ignored after the SPI is enabled.

### 7.4.3.3 SPI Status Register (STATUS)

The following table gives the SPI Status Register details. This register indicates the state of SPI such as Tx/Rx FIFO, Tx under-run, and Rx overflow.

**Table 99 • STATUS**

Bit Number	Name	R/W	Reset Value	Description
[31:15]	Reserved	R/W	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	ACTIVE	R		SPI is still transmitting or receiving data.
13	SSEL	R		Current state of SPI_0_SS0

**Table 99 • STATUS (continued)**

Bit Number	Name	R/W	Reset Value	Description
12	FRAMESTART			0: SPI output enable is active as required. 1: SPI output enable is not asserted. Allows multiple slaves to share a single slave select signal with a single master.
11	TXFIFOEMPNT	R	0	Transmit FIFO empty on next read
10	TXFIFOEMP	R	1	Transmit FIFO is empty
9	TXFIFOFULNXT	R	0	Transmit FIFO full on next write
8	TXFIFOFUL	R	0	Transmit FIFO is full
7	RXFIFOEMPNT	R	0	Receive FIFO empty on next read
6	RXFIFOEMP	R	1	Receive FIFO empty
5	RXFIFOFULNXT	R	0	Receive FIFO full on next write
4	RXFIFOFUL	R	0	Receive FIFO is full
3	TXUNDERRUN	RO	0	No data available for transmission. The channel cannot read data from the transmit FIFO because the transmit FIFO is empty. Certainly this can only be raised in slave mode because the master will not attempt to transmit unless there is data in FIFO.
2	RXOVERFLOW	RO	0	Channel is unable to write to receive FIFO as it is full. Applies to master and slave modes.
1	RXDATRCED	RO	0	When set, it indicates that the number of frames specified by TXRXDFCOUNT has been received and can be read. Applies to master and slave modes.
0	TXDATSENT	RO	0	When set, it indicates that the numbers of frames specified by TXRXDFCOUNT has been sent. Applies to master and slave modes.

**Notes:**

- Bits [11:4] correspond to FIFO status.
- None of these status bits are sticky. During run-time, the status of these bits reflects the current status of SPI.
- To determine the cause of an interrupt, the masked interrupt status (MIS) register must be read.

**7.4.3.4 SPI Interrupt Clear Register (INT\_CLEAR)**

The following table describes the Interrupt Clear register. A read to this register has no effect. It returns all zeroes.

**Table 100 • INT\_CLEAR**

Bit Number	Name	R/W	Reset Value	Description
[31:6]	Reserved	W	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	SSEND	W		Write one to clear the interrupt
4	CMDINT	W		Write one to clear the interrupt
3	TXCHUNDRUN	W	0	Transmit channel under-run
2	RXCHOVRFLOW	W	0	Receive channel over flow

**Table 100 • INT\_CLEAR (continued)**

Bit Number	Name	R/W	Reset Value	Description
1	RXRDYCLR	W	0	Clears receive ready (RX_RDY)
0	TXDONECLR	W	0	Clears transmit done (TX_DONE)

### 7.4.3.5 SPI Receive Data Register (RX\_DATA)

The following table describes the Receive Data register.

**Table 101 • RX\_DATA**

Bit Number	Name	R/W	Reset Value	Description
[31:0]	RXDATA	R	0	Received data. Reading this clears the register of the received data.

### 7.4.3.6 SPI Transmit Data Register (TX\_DATA)

The following table describes the Transmit Data register.

**Table 102 • TX\_DATA**

Bit Number	Name	R/W	Reset Value	Description
[31:0]	TXDATA	W	0	Data to be transmitted. Writing to this clears the last data transmitted.

### 7.4.3.7 SPI SCLK Generation Register (CLK\_GEN)

The following table describes the clock modes used to calculate the SPICLK divider.

**Table 103 • CLK\_GEN**

Bit Number	Name	R/W	Reset Value	Description
[31:8]	Reserved	R/W	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.
[7:0]	CLK_GEN	R/W	0	Specifies the methodology used to calculate the SPICLK divider. CLK_MODE = 0: $SPICLK = 1 / (2^{CLK\_GEN + 1})$ where CLK_GEN = 0 to 15. CLK_MODE = 1: $SPICLK = 1 / (2 \times (CLK\_GEN + 1))$ where CLK_GEN = 0 to 255.

The following table describes the SPICLK rates in different modes.

**Table 104 • CLK\_MODE Example, APB Clock = 153.8 MHz**

CLK_MODE=0		CLK_MODE=1	
$SPICLK = 1 / (2^{CLKRATE + 1})$ where CLKRATE = 0 to 15		$SPICLK = 1 / (2 \times (CLKRATE + 1))$ where CLKRATE = 0 to 255	
CLKRATE	SPI Clock	CLKRATE	SPI Clock
0	76,900,000	0	76,900,000
1	38,450,000	1	38,450,000

**Table 104 • CLK\_MODE Example, APB Clock = 153.8 MHz (continued)**

CLK_MODE=0		CLK_MODE=1	
SPICLK = 1 / (2 <sup>CLKRATE</sup> + 1) where CLKRATE = 0 to 15		SPICLK = 1 / (2 × (CLKRATE + 1)) where CLKRATE = 0 to 255	
CLKRATE	SPI Clock	CLKRATE	SPI Clock
2	19,225,000	2	25,633,333.33
3	9,612,500	3	19,225,000
4	4,806,250	4	15,380,000
5	2,403,125	5	12,816,666.67
6	1,201,562.5	6	10,985,714.29
7	600,781.25	7	9,612,500
8	300,390.625	8	8,544,444.444
9	150,195.312	9	7,690,000
10	75,097.656	10	6,990,909.091
11	37,548.828	11	6,408,333.333
12	18,774.414	12	5,915,384.615
13	9,387.207	13	5,492,857.143
14	4,693.603	14	5,126,666.667
15	2,346.801	15	4,806,250
		<b>255</b>	<b>300,390.625</b>

### 7.4.3.8 SPI Slave Select Register

The following table describes the register that specifies the slave that has been selected.

**Table 105 • SLAVE\_SELECT**

Bit Number	Name	R/W	Reset Value	Description
[31:1]	Reserved	R/W	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	SLAVE SELECT	R/W	0	Specifies the slave selected. Writing one to a bit position selects the corresponding slave. SLAVESELECT[0] is available at the SPI_0_SS0 pin.

**Note:** The slave select output polarity is active low. In TI mode the slave select output is inverted to become active high.

### 7.4.3.9 SPI Masked Interrupt Status Register

The following table describes the Masked Interrupt Status (MIS) register. It is a read-only register. On a read this register gives the current masked status value of the corresponding interrupt. A write has no effect.

**Table 106 • MIS**

Bit Number	Name	R/W	Reset Value	Description
[31:6]	Reserved	R/W	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	SSEND	R		Indicates that SPI_0_SS0 has gone inactive. When this is high, the interrupt is active.
4	CMDINT	R		Indicates that the number of frames set by the CMDSIZE register has been received as a single packet of frames (SPI_0_SS0 held active). When this is high, the interrupt is active.
3	TXCHUNDDMSKINT	R	0	Masked interrupt status. Reading this returns interrupt status. Masked interrupt status = Raw interrupt status and interrupt mask (CONTROL Register). MIS = RIS and CONTROL[7:4]. Masked status of transmit channel under-run TXCHUNDDMSKINT=TXCHUNDRINT and INTTXUNRRUN
2	RXCHOVRFMSKINT	R	0	Masked status of receive channel overflow. RXCHOVRFMSKINT = RXCHOVRFINT and INTRXOVRFLO
1	RXRDYMSKINT	R	0	Masked status of receive data ready (data received in FIFO). RXRDYMSKINT = RXRDY and INTRXDATA
0	TXDONEMSKINT	R	0	Masked status of transmit done (data shifted out) TXDONEMSKINT = TXDONE and INTTXDATA

### 7.4.3.10 SPI Raw Interrupt Status Register

The following table describes the Raw Interrupt Status (RIS) register. This register returns the current raw status value, prior to masking, of the corresponding interrupt.

**Table 107 • RIS**

Bit Number	Name	R/W	Reset Value	Description
[31:6]	Reserved	R/W	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	SSEND	R/W		Indicates that SPI_0_SS0 has gone inactive.
4	CMDINT	R/W		Indicates that the number of frames set by the CMDSIZE register has been received as a single packet of frames (SPI_0_SS0 held active).
3	TXCHUNDR	R	0	RAW interrupt status. Reading this returns raw interrupt status. Raw status of transmit channel under-run
2	RXCHOVRF	R	0	Raw status of receive channel overflow

**Table 107 • RIS (continued)**

Bit Number	Name	R/W	Reset Value	Description
1	RXRDY	R	0	Receive data ready (data received in FIFO)
0	TXDONE	R	0	Raw status of transmit done (data shifted out)

### 7.4.3.11 SPI Control2 Register

The following table describes the Control2 register details as the terminal frame counter, SPI slave select, and auto status of SPI.

**Table 108 • CONTROL2**

Bit Number	Name	R/W	Reset Value	Description
[31:6]	Reserved	R/W	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	INTEN_SSEND	R/W		Indicates that SPI_0_SS0 has gone inactive.
4	INTEN_CMD	R/W		Indicates that the number of frames set by the CMDSIZE register have been received as a single packet of frames (SPI_0_SS0 held active).
3	Reserved	R/W	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	DISFRMCNT	R/W	0	0: The internal frame counter is active. When the counter reaches the programmed limit, it will pause the current SPI transfer inserting idle cycles and generate the appropriate interrupts. 1: The internal frame counter is not active. The core transmits data until the transmit FIFO empties. The FRAMECNT (CONTROL register) should also be programmed to zero.
1	AUTOPOLL	R/W	0	0: No effect 1: The first receive frame after SPI_0_SS0 is active. It is discarded (not written to the FIFO) and supports the POLL function.
0	AUTOSTATUS	R/W	0	0: No effect 1: The first transmitted frame (slave mode) contains the hardware status, not data from the transmit FIFO.

### 7.4.3.12 SPI Command Register

The following table describes the Command register..

**Table 109 • COMMAND**

Bit Number	Name	R/W	Reset Value	Description
[31:7]	Reserved	R/W	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	TXNOW	R/W	0	0: No effect 1: Writing one clears the TxBUSY bit in slave mode immediately rather than waiting for PKTSIZE frames to be available, telling the master that there is data available. This is intended to use when less than the programmed PKTSIZE data frames are being transmitted, removing the requirement to transmit PKTSIZE frames. This bit stays set until the first data frame is transmitted.
5	AUTOSTALL	R/W	0	0: No effect 1: Writing one will cause the master to delay transmission until the transmit FIFO contains the number of frames specified by the PKTSIZE register (see Table 110, page 159). This guarantee that the frames are transmitted with no idle cycles or time gaps between them. This bit will be automatically cleared as soon as the core starts transmitting the frames.
4	CLRFRAMECNT	R/W	0	0: No effect 1: Writing one clears the internal frame counter. This bit always reads as zero. The counter is also cleared when the core is disabled, CTL1, or CTL2 are written (that is, the frame count limit changed).
3	TXFIFORST	R/W	0	0: No effect 1: Writing one resets the Tx FIFO. This bit always reads as zero.
2	RXFIFORST	R/W	0	0: No effect Writing one resets the Rx FIFO. This bit always reads as zero.
1	AUTOEMPTY	R/W	0	0: No effect 1: Writing one causes the SPI core to automatically discard any further received data until the number of frames requested in the FRAMECNT register has been received or (in slave mode) SSEL goes inactive. This bit will stay set until all the frames are complete or it is cleared.
0	AUTOFILL	R/W	0	0: No effect 1: Writing one causes the SPI core to automatically fill the transmit FIFO with zeros to match the number of frames requested in the FRAMECNT register. Typically, the five command bytes must be written to the TxDATA register and then this bit must be set. Data can be read from the receive FIFO until the complete set of frames has been read. This bit will stay set until all the frames are complete or it is cleared.

### 7.4.3.13 SPI Packet Size Register

The following table provides the details of the Packet Size registers that are used to set the SPI CMD/data frame size.

**Table 110 • PKTSIZE**

Bit Number	Name	R/W	Reset Value	Description
[31:8]	Reserved	R/W	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.
[7:0]	PKTSIZE	R/W	0	Sets the size of the SPI CMD/data frame. PKTSIZE cannot be greater than the FIFO size.

### 7.4.3.14 SPI Command Size Register

The following table describes the Command size register.

**Table 111 • CMD\_SIZE**

Bit Number	Name	R/W	Reset Value	Description
[31:8]	Reserved	R/W	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.
[7:0]	CMDSIZE <sup>1</sup>	R/W	0	Number of frames after SPI_SS0 going active that the CMD interrupt should be generated.

1. This controls the RxCMD interrupt. The internal counters count frames from SPI\_SS0 going low. It automatically resets and starts counting again once SSEL goes inactive. In TI mode, back- to-back frames are counted, any gaps in data causes the counter to start counting again.

### 7.4.3.15 SPI Hardware Status Register

The following table describes the Hardware Status Register. This register allows the Fabric master to control the hardware Status Register used in the slave protocol controller.

**Table 112 • HWSTATUS**

Bit Number	Name	R/W	Reset Value	Description
[31:4]	Not used	R/W	0	These bits are undefined. The value that the slave transmits depends on the data that is queued in the transmit FIFO.
[3:2]	USER	R/W	0	These bits are set by the Fabric master. Their function is undefined but could be used to send additional status or request information to the master.
1	TXBUSY	R/W	0	0: Master may request the requested data. There are PKTSIZE frames of data in the transmit FIFO (when AUTOPOLL is set to PKTSIZE - 1) 1: Indicates not ready to transmit data.
0	RXBUSY	R/W	0	1: Indicates that the receive buffer is busy (not empty). 0: Indicates that up to PKTSIZE frames of command followed by data may be sent to the slave.

### 7.4.3.16 SPI Status 8 Register

The following table describes the SPI status 8 (STAT8) register. This register allows the important status bits to be read as a single 8-bit value. This reduces the overhead of checking the Status Register bits when an 8-bit processor is being used.

**Table 113 • STAT8**

Bit Number	Equivalent STATUS Register Bit		Name	R/W	Reset Value	Description
	Position					
[31:8]			Reserved	R/W	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	14		ACTIVE	R	0	SPI is still transmitting the data
6	13		SSEL	R	0	Current state of SPI_0_SS0
5	3		TXUNDERRUN	R	0	Transmit FIFO underflowed
4	2		RXOVERFLOW	R	0	Receive FIFO overflowed
3	8		TXFIFOFUL	R	0	Transmit FIFO is full
2	6		RXFIFOEMP	R	0	Receive FIFO is empty
1	0 and 1		DONE	R	0	The number of request frames have been transmitted and received.
0	12		FRAMESTART	R	0	Next frame in receive FIFO was received after SPI_0_SS0 went active (command frame).

## 8 Communication Block

---

The communication block (COMM\_BLK) provides a bi-directional message passing facility between the FPGA fabric master and the system controller, similar to a mailbox communication channel.

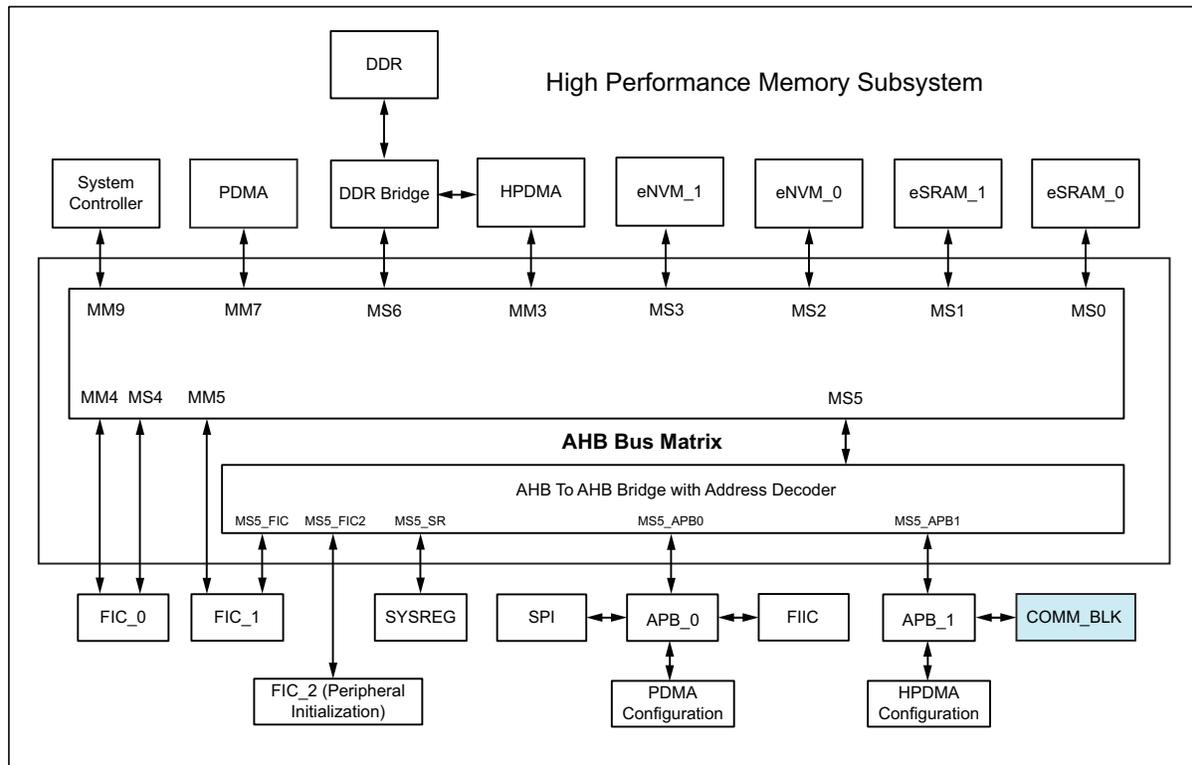
### 8.1 Features

The COMM\_BLK peripheral includes the following features:

- Bi-directional byte-wide message path
- Supports serial data rate up to 50 Mbytes/sec.
- Asynchronous clock support
  - Data clock (50 MHz RC oscillator) is different from advanced peripheral bus (APB) clock
- 8 byte transmit FIFO
- 8 byte receive FIFO
- Flow control
  - RX to TX channels between High Performance Memory Subsystem (HPMS) COMM\_BLK and system controller COMM\_BLK
  - HPMS COMM\_BLK to peripheral direct memory access (PDMA) channel
- Frame and/or command marker
  - 9th bit used as frame start or command marker
  - Allows command and data sequences to be distinguished
  - Allows incomplete sequences to be detected
  - Separate command interrupt received with programmable match logic
- Allows WORD transfers into FIFO in a single APB cycle
- Interrupts
  - RX FIFO non-empty
  - TX FIFO non-full
  - TX overflow
  - RX Underflow

The following figure depicts the connectivity of COMM\_BLK to the advanced high-performance bus (AHB) matrix.

**Figure 97 • Interfacing of COMM\_BLK with AHB Bus Matrix**



## 8.2 Functional Description

This section provides the details of the COMM\_BLK subsystem.

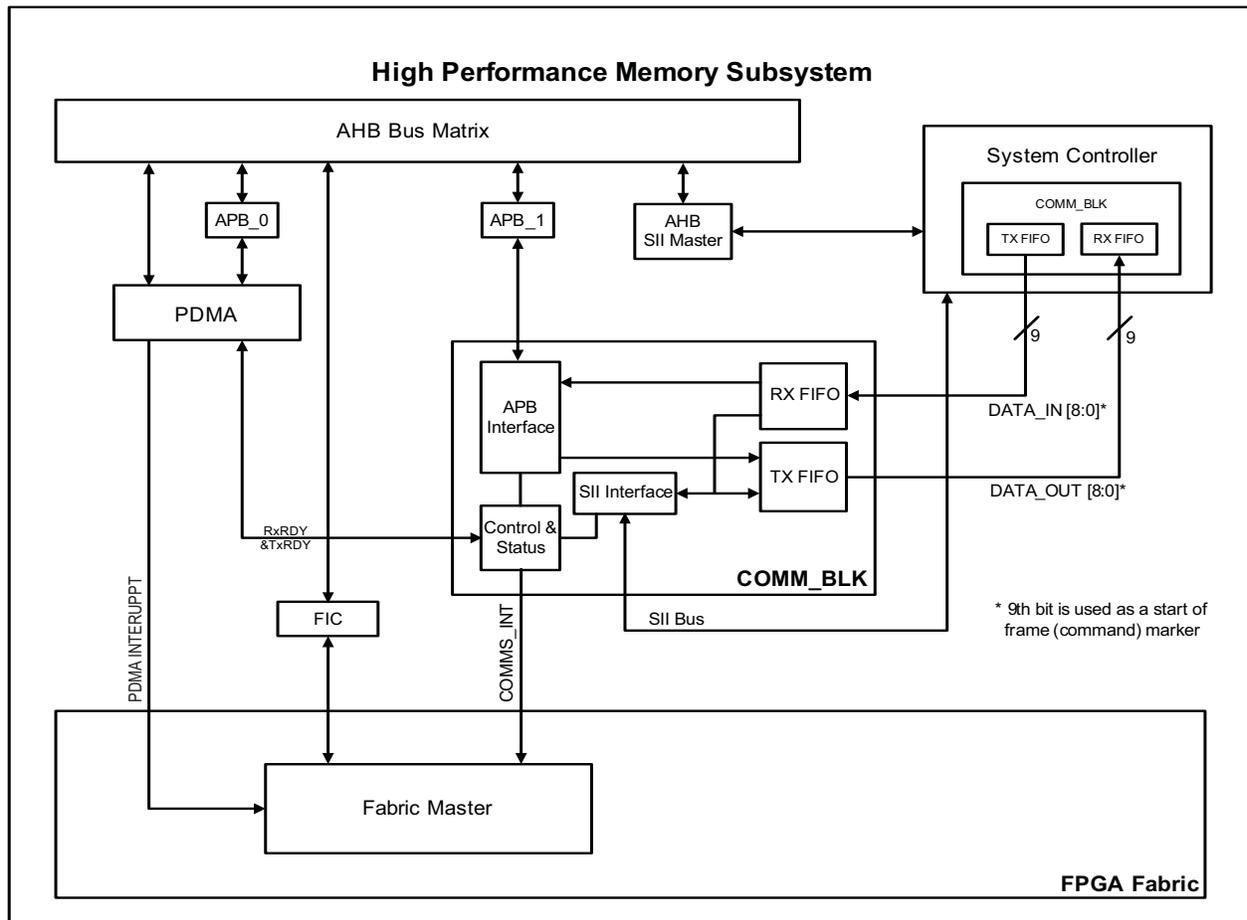
### 8.2.1 Architecture Overview

The COMM\_BLK consists of an APB interface, 8 byte transmit FIFO, and an 8 byte receive FIFO. There is one COMM\_BLK instantiated in the HPMS and one in the system controller; each can communicate with the other. Whenever the FPGA fabric master writes a character into the COMM\_BLK, it is transmitted to the receiving side of the COMM\_BLK and an interrupt is asserted to the system controller.

In the other direction, the interrupt (COMM\_BLK\_INT) goes to the FPGA fabric through the fabric interface interrupt controller (FIIC). This communication link is used as a message passing mailbox by FPGA fabric master and system controller.

The following figure shows how COMM\_BLKs are connected to create a communication channel between the FPGA fabric master and the system controller.

**Figure 98 • Interfacing of COMM\_BLK with System Controller**



The COMM\_BLK supports PDMA operation. The peripheral ready signals, RxRDY and TxRDY, are directly connected to the PDMA, and are used for flow control between the HPMS COMM\_BLK and PDMA channel. Data from the COMM\_BLK receive FIFO going to any HPMS memory mapped locations, and the data from any HPMS memory mapped locations going to the COMM\_BLK transmit FIFO can be transferred without using the FPGA fabric master or the system controller.

## 8.2.2 Frame/Command Marker

The COMM\_BLK allows the data that is being transferred to be marked as a command or data byte. To allow the receiver to correctly identify the start of a packet, the COMM\_BLK block uses a 9th bit (Bit 8 of DATA\_IN and DATA\_OUT as shown in Figure 98, page 163).

When FRAME\_START8/FRAME\_START32 register (see Table 120, page 174/Table 121, page 175) is written, the 9th bit is set. When DATA8/DATA32 register (see Table 118, page 174/Table 119, page 174) is written, the 9th bit is not set.

The STATUS register (see Table 116, page 173) bit 7 gives indication to the receiver whether the next byte that will be read out of the FIFO has the 9th bit set, and therefore indicating that it is the start of a packet.

This mechanism allows the receiver to verify that no bytes have been lost and stops it from accidentally interpreting data overruns as command. The RCVOKAY and TXTOKAY status bits must be checked in the STATUS register before reading and writing data or command.

### 8.2.3 Clocks

APB Interface, Control and Status block, and SII Interface are clocked by PCLK1 from the APB1 bus. RX FIFO and TX FIFO are clocked by data clock (50 MHz RC oscillator). PCLK is derived from the fabric aligned clock controller (FACC) output. Refer to the [UG0449: SmartFusion2 and IGLOO2 Clocking Resources User Guide](#).

### 8.2.4 Resets

The COMM\_BLK resets to zero on power-up and is held in reset until it is enabled. There is an option to reset the COMM\_BLK by writing to the system register. Specifically, this system register is SOFT\_RESET\_CR (see [System Register Block](#), page 196). The COMBLK\_SOFTRESET control bit is encoded in bit location 15 as follows:

- 0: COMM\_BLK reset released
- 1: COMM\_BLK held in reset (reset value)

At power-up, the reset signal is asserted 1. This keeps the COMM\_BLK peripheral in a reset state. If this bit is set to 0, the COMM\_BLK peripheral is allowed to become active.

### 8.2.5 Interrupts

There is one interrupt signal from the COMM\_BLK peripheral. The COMMS\_INT signal goes to the FPGA fabric through the FIIC. The interrupt in the COMM\_BLK peripheral must be enabled by setting the appropriate bits in the interrupt enable register. Clear the appropriate bit in the Interrupt Enable Register (see [Interrupt Enable Register](#), page 173) when servicing the COMMS\_INT to prevent a reassertion of the interrupt.

## COMM\_BLK Initialization

The COMM\_BLK peripheral can be initialized by configuring the COMM\_BLK Control Register and SOFT\_RESET\_CR system register. The initialization sequence is as follows:

1. Release the COMM\_BLK from reset by using SOFT\_RESET\_CR system register (refer to [Resets](#), page 164 for further details)
2. Enable COMM\_BLK by writing 1 to the ENABLE bit of Control Register.
3. Disable the loopback by writing '0' to the LOOPBACK bit in the Control Register.

### 8.2.6 CoreSysServices Soft IP

COMM\_BLK is used to call the following system services:

- Device and Design Information Services
- Flash\*Freeze Service
- Cryptographic Services
- DPA-Resistant Key-Tree Services
- Non-Deterministic Random Bit Generator (NRBG) Services
- Zeroization Service
- Programming Service
- NVM Data Integrity Check Service

Microsemi provides CoreSysServices soft IP to access the system services implemented by the System Controller. The CoreSysServices soft IP provides a user interface for each of the system services and an advanced high-performance bus (AHB)-Lite master interface on the fabric interface controller (FIC) side. The core communicates with the COMM\_BLK through the FIC\_0 interface.

CoreSysServices soft IP decodes the command received from the user logic and translates the user logic transactions to the AHB-Lite master transactions. For more information on CoreSysServices soft IP, refer to the **CoreSysServices Handbook**.

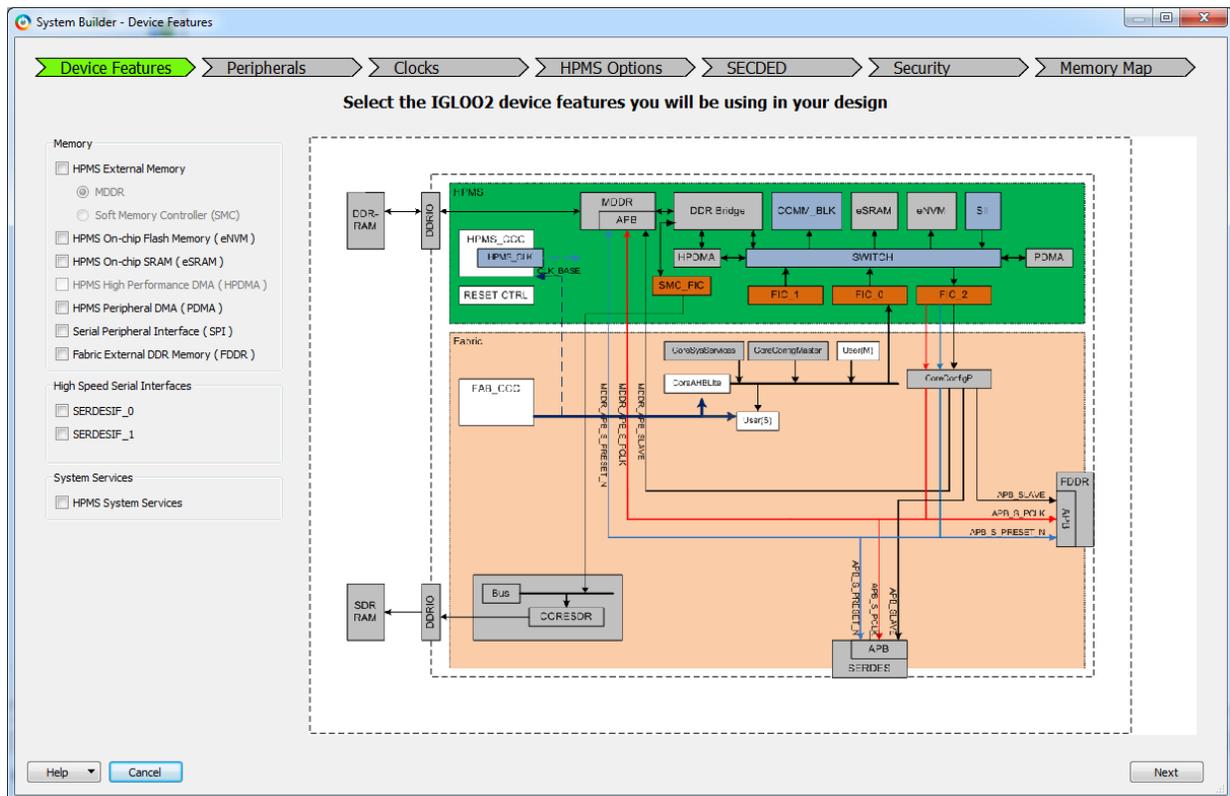
Refer to the **How to Use System Services** section in the **System Services** chapter in the [UG0450: SmartFusion2 SoC FPGA and IGLOO2 FPGA System Controller User Guide](#) to know how to implement the system services.

### 8.3 How to Use COMM\_BLK

This section describes how to use COMM\_BLK 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 figure shows the initial **System Builder** window where the required device features can be selected. 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 99 • System Builder Window**

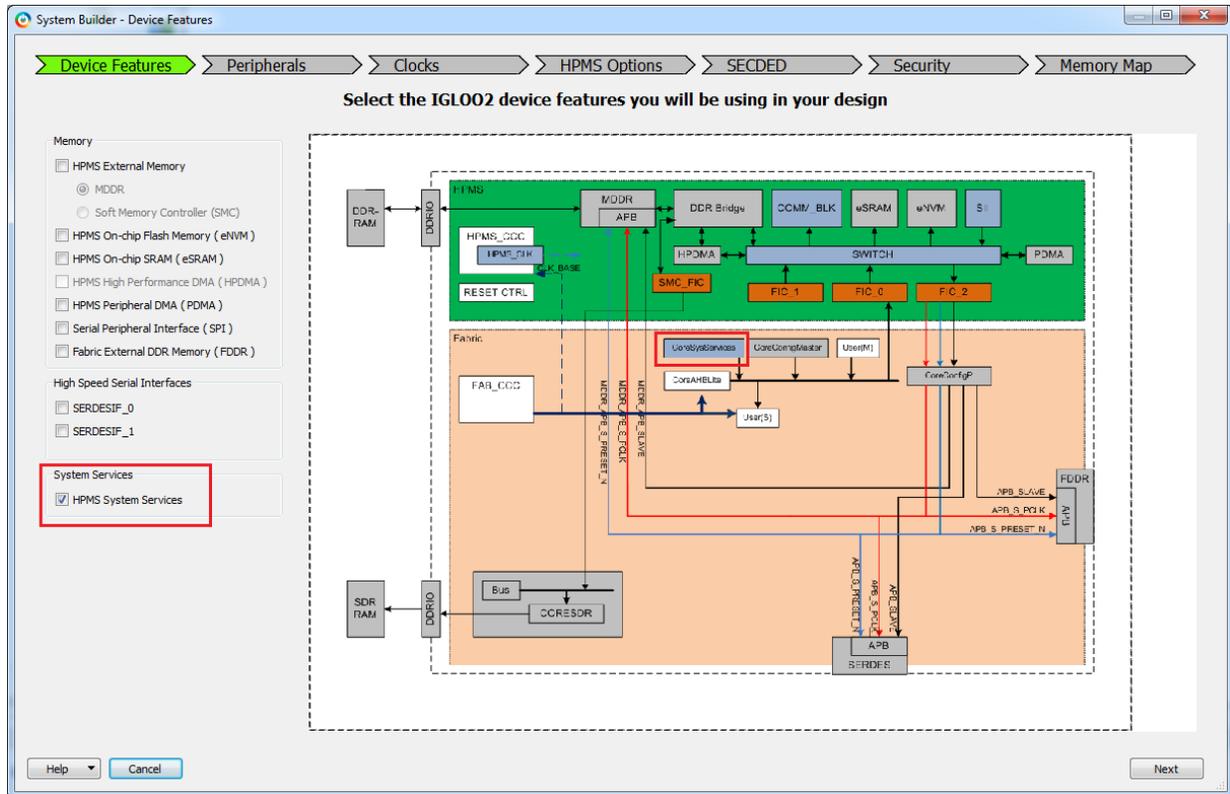


### 8.3.1 Configuring COMM\_BLK

The following steps describe how to configure COMM\_BLK.

1. Check the **HPMS System Services** check box in the **Device Features** tab and leave the other check boxes unchecked. The following figure shows the **System Builder - Device Features Tab**.

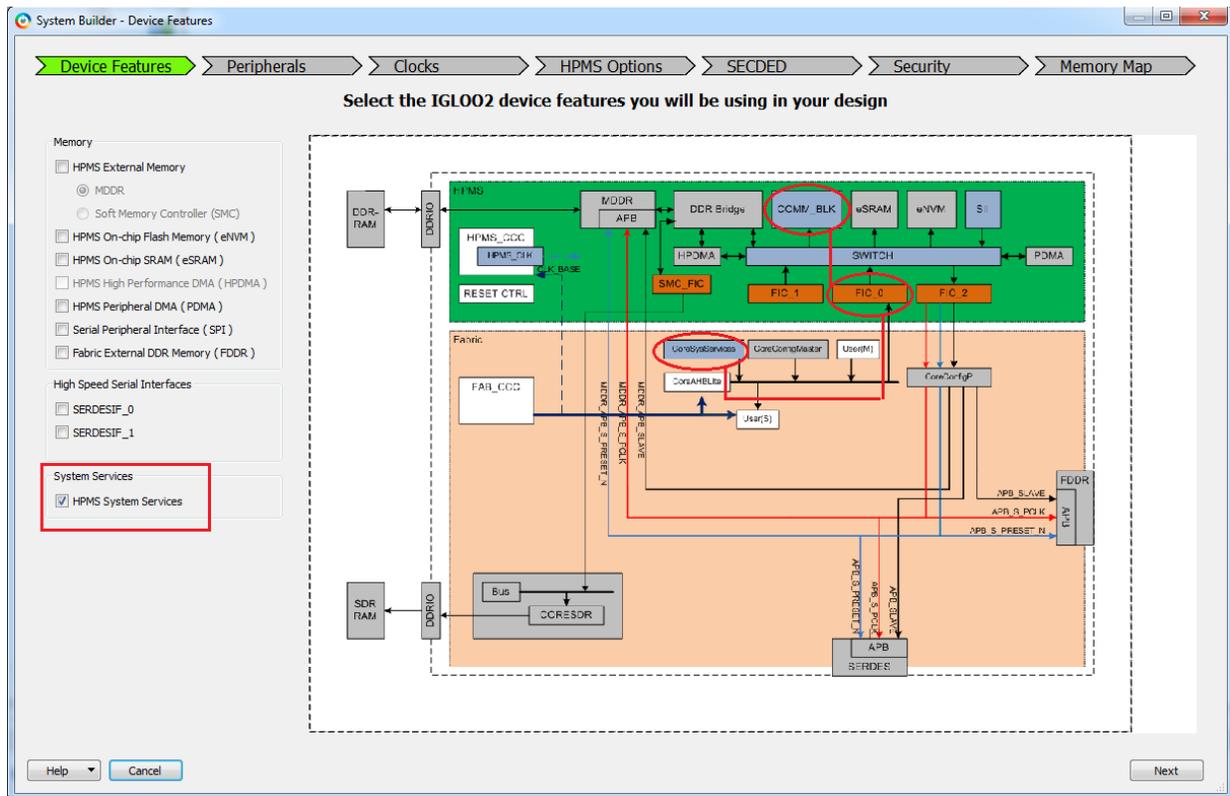
**Figure 100 • System Builder - Device Features Tab**



2. Checking the **HPMS System Services** check box establishes a path for connecting the CoreSysServices soft IP to the COMM\_BLK through the FIC\_0 interface. The enabling is indicated with a change of color in the CoreSysServices block in the System Builder. The following figure shows the path between the COMM\_BLK and the CoreSysServices soft IP.

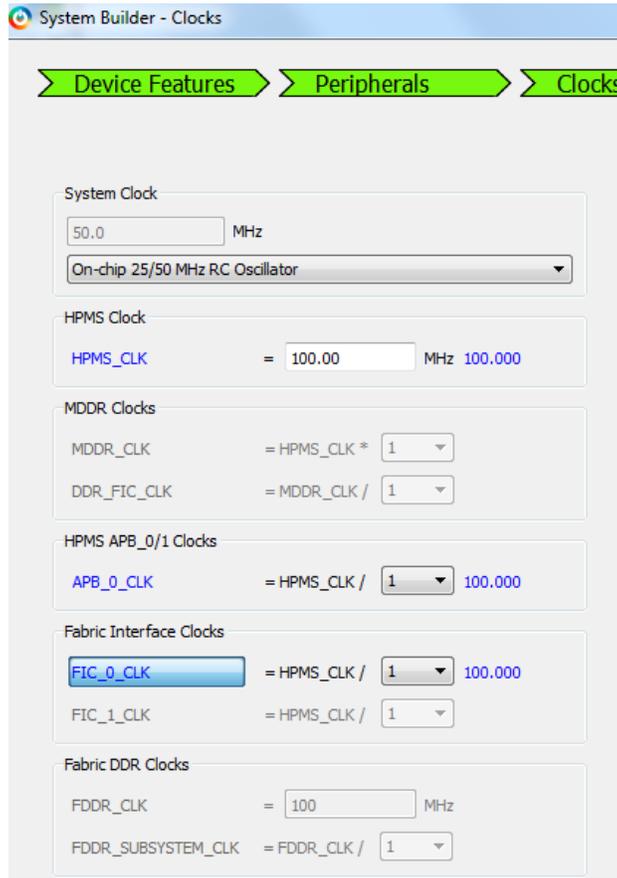
**Note:** The **System Builder** does not automatically instantiate the CoreSysServices soft IP but allows the user to connect it with the FIC\_0 master interface port.

**Figure 101 • CoreSysServices IP to COMM\_BLK Path**



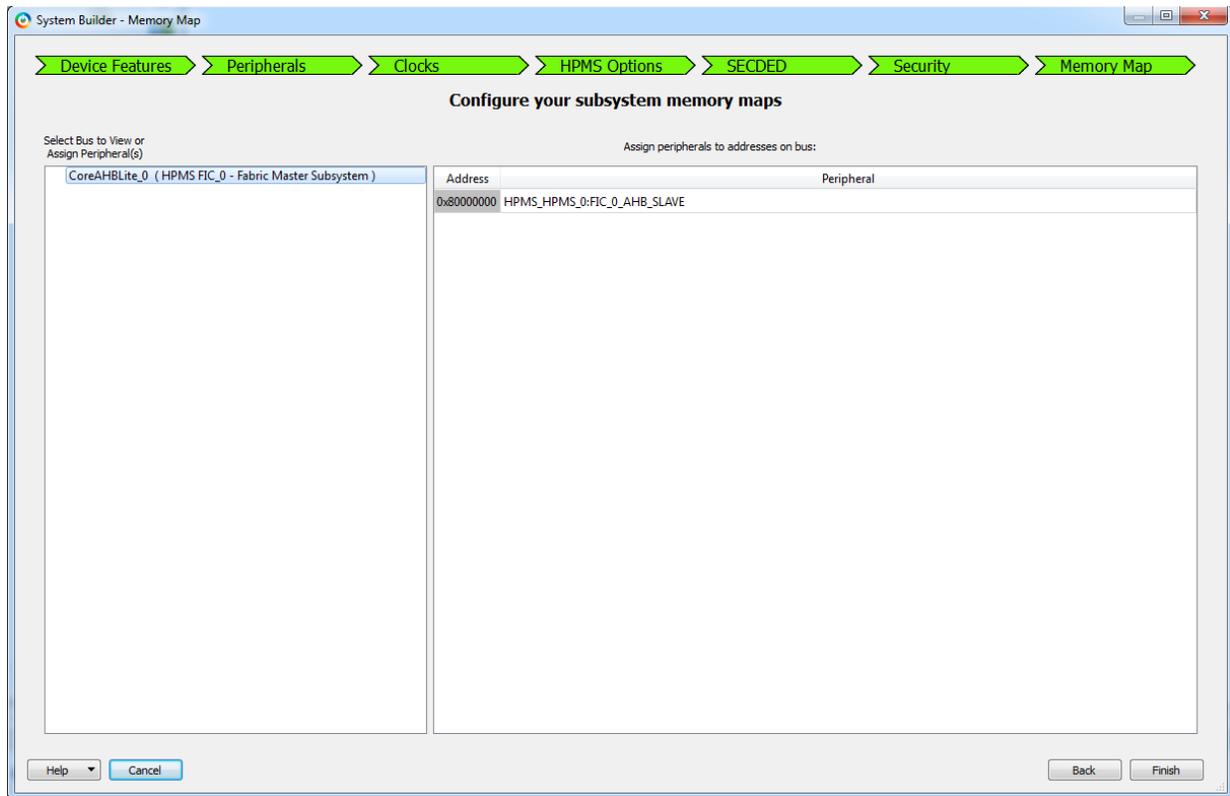
- Go to **Clocks** tab and configure the required FIC\_0 clock frequency. The following figure shows the **System Builder - Clocks** tab.

**Figure 102 • Clocks Configuration**



4. Navigate to the **Memory Map** tab giving the required data in the rest of the **System Builder** tabs. The following figure shows the **System Builder - Memory Map** tab. Click **Finish** to proceed with creating the subsystem.

**Figure 103 • System Builder - Memory Map Tab**

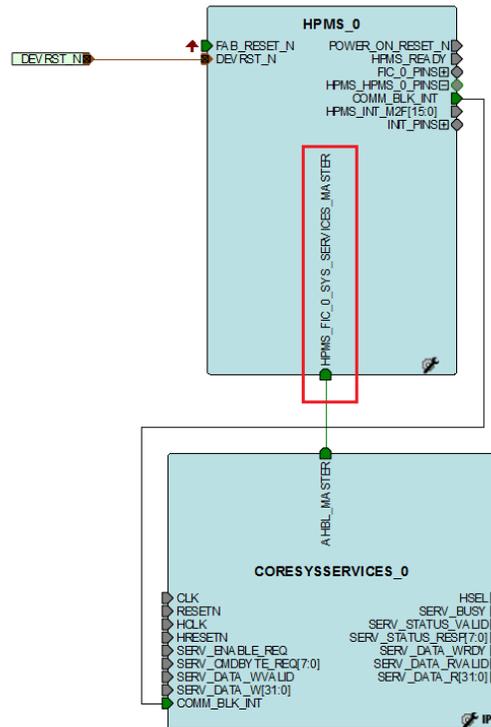


5. Instantiate the CoreSysServices soft IP in SmartDesign canvas and configure it for the required System Services features.

6. Connect:
  - The CoreSysServices master interface port to the system builder generated top-level component master interface port.
  - The COMM\_BLK\_INT signal of CoreSysServices to the COMM\_BLK\_INT signal of the system builder generated top-level component.

The following figure shows the COMM\_BLK connection to the CoreSysServices IP.

**Figure 104 • COMM\_BLK Connection with CoreSysServices IP**



- Right click on the CoreSysServices IP and select **Configure**. The following figure shows the **Configuration** dialog.

**Figure 105 • COMM\_BLK Configuration Dialog**

Configuration

Device and Design Information Services

Serial Number Service: <input type="checkbox"/>	Pointer to receive 128-bit Serial number: <input type="text" value="0x20000000"/>
User Code Service: <input type="checkbox"/>	Pointer to receive 32-bit User Code: <input type="text" value="0x20000000"/>
Device Certificate Service: <input type="checkbox"/>	Pointer to receive 768-byte Device Certificate: <input type="text" value="0x20000000"/>
User Design Version Service: <input type="checkbox"/>	Pointer to receive 16-bit Design Version: <input type="text" value="0x20000000"/>

Data Security Services

Cryptographic Service for AES128: <input type="checkbox"/>	Pointer to AES128 descriptor data: <input type="text" value="0x20000000"/>
Cryptographic Service for AES256: <input type="checkbox"/>	Pointer to AES256 descriptor data: <input type="text" value="0x20000000"/>
Pointer to input data to encrypt/decrypt: <input type="text" value="0x20000000"/>	Pointer to receive output decrypted/encrypted data: <input type="text" value="0x20000000"/>
Cryptographic Service for SHA256: <input type="checkbox"/>	Pointer to SHA256 descriptor data: <input type="text" value="0x20000000"/>
Cryptographic Service for HMAC: <input type="checkbox"/>	Pointer to HMAC descriptor data: <input type="text" value="0x20000000"/>
Pointer to receive 256-bit HASH result: <input type="text" value="0x20000000"/>	Pointer to data to be hashed: <input type="text" value="0x20000000"/>
DPA Resistant Key Tree Service: <input type="checkbox"/>	Pointer to Key Tree descriptor data: <input type="text" value="0x20000000"/>
DPA Resistant Pseudo PUF Challenge Response Service: <input type="checkbox"/>	Pointer to Pseudo PUF Challenge Response descriptor data: <input type="text" value="0x20000000"/>
Pointer to receive 32-byte result: <input type="text" value="0x20000000"/>	

Data Security Services

Non-Deterministic Random Number Generator Service:

Pointer to DRBG Instantiate structure: <input type="text" value="0x20000000"/>
Pointer to RBG Personalization string: <input type="text" value="0x20000000"/>
Pointer to DRBG Generate structure: <input type="text" value="0x20000000"/>
Pointer to receive generated random data: <input type="text" value="0x20000000"/>
Pointer to DRBG Reseed structure: <input type="text" value="0x20000000"/>
Pointer to additional input data: <input type="text" value="0x20000000"/>

NVM Data Integrity Service: <input type="checkbox"/>	Asynchronous Messages Power-on-Reset Digest Error Service: <input type="checkbox"/>
Flash*Freeze Service: <input type="checkbox"/>	Zeroization Service: <input type="checkbox"/>

Testbench:

## 8.4 COMM\_BLK Configuration Registers

The COMM\_BLK base address resides at 0x40016000 and extends to address 0x40016FFF in the memory map. The following table summarizes the control and Status Registers for the COMM\_BLK.

**Table 114 • COMM\_BLK Register Map**

Register Name	Address Offset	R/W	Reset Value	Description
CONTROL	0x00	R/W	0x00	Control Register. See <a href="#">Table 115</a> , page 172.
STATUS	0x04	R/W	0x00	Status Register. See <a href="#">Table 116</a> , page 173.
INT_ENABLE	0x08	R/W	0x00	Interrupt Enable. See <a href="#">Table 117</a> , page 173.
DATA8	0x10	R/W	0x00	Byte Data register. See <a href="#">Table 118</a> , page 174.

**Table 114 • COMM\_BLK Register Map (continued)**

DATA32	0x14	R/W	0x00000000	Word Data register. See <a href="#">Table 119</a> , page 174.
FRAME_START8	0x18	R/W	0x00	Frame/Command Byte register. See <a href="#">Table 120</a> , page 174.
FRAME_START32	0x1c	R/W	0x00000000	Frame/Command Word register. See <a href="#">Table 121</a> , page 175.

## 8.5 COMM\_BLK Register Interface Details

This section describes the COMM\_BLK registers in detail.

### 8.5.1 Control Register

**Table 115 • CONTROL**

Bit Number	Name	R/W	Reset Value	Description
[7:6]	RESERVED	R	00	
5	LOOPBACK	R/W	1	After system reset the COMM_BLK is in Loopback mode. Set LOOPBACK bit to '0' to disable the loopback (Normal operation). It is used for factory test.
4	ENABLE	R/W	0	Configure the COMM_BLK interface. 0: Disables COMM_BLK 1: Enables COMM_BLK Enable COMM_BLK before writing to the FIFO and leave it enabled if it is being used.
3	SIZERX	R/W	0	Sets the number of bytes that each APB transfer reads from the RX FIFO. 0: 1 Byte 1: 4 Bytes (32-bits) This setting effects the behavior of the RxDY signal and RCVOKAY flags. When set to 0 the flags indicate that a byte can be read and when set to 1 it indicates that a word can be read.
2	SIZETX	R/W	0	Sets the number of bytes that each APB transfer writes into the TX FIFO. 0: 1 Byte 1: 4 Bytes (32-bits) This setting effects the behavior of TxRDY signal and TXTOKAY. When set to 0 the flags indicate that a byte can be written and when set to 1 it indicates that a word can read be written.
1	FLUSHIN	R	0	Indicates FIFO flush status. 1 indicates flush process is in progress. 0 indicates that flush process is completed.
0	FLUSHOUT	R/W	0	Flush all FIFO's. Writing 1 to this bit starts the flush process. When the flush process is complete this bit returns to 0 automatically. The flush process takes several clock cycles to complete, depending on the various clock rates. Writing 0 has no effect.

## 8.5.2 Status Register

This register provides status information. R/W bits are cleared by writing 1. FIFO empty full flags will automatically clear as FIFO is full and empty.

**Table 116 • STATUS**

Bit Number	Name	R/W	Reset Value	Description
7	COMMAND	R	0	First byte queued in receive FIFO has the command marker set
6	SIERROR	R/W	0	When an SII transfer <sup>1</sup> (HPMS to SII) is in progress, the start of frame marker is set on one or more of the bytes. Write 1 to clear
5	FLUSHRCVD	R/W	0	Indicates that a FLUSH has been received. Write 1 to clear
4	SIIDONE	R/W	0	Indicated that the transfer to SII Bus is complete. Write 1 to clear
3	UNDERFLOW	R/W	0	Receive Overflow. Indicates that the receive FIFO was read when empty. Write 1 to clear
2	OVERFLOW	R/W	0	Transmit Overflow. Indicates that the Transmit FIFO was written when full. Write 1 to clear
1	RCVOKAY	R	0	RCV FIFO non empty. Indicates that 1 or 4 bytes may be read based on SIZERX.
0	TXTOKAY	R	1	TXT FIFO non full. Indicates that 1 or 4 bytes may be written depending on SIZETX.

1. The system IP interface (SII) master connects the System Controller with all the internal elements. It is used to transfer data to and from the HPMS memory space by the System Controller for System Services. It is also used for factory test but not available for customer.

## 8.5.3 Interrupt Enable Register

This register enables the COMMS\_INT to be set whenever the corresponding bit is set in the STATUS register.

**Table 117 • INT\_ENABLE**

Bit Number	Name	R/W	Reset Value	Description
[7:0]	ENABLE	R/W	0x00	Matches corresponding bit in Status Register 0: Disables Interrupt 1: Enables Interrupt

## 8.5.4 Byte Data Register

This register writes a byte to the Transmit FIFO or reads a byte from the Receive FIFO. If the Transmit FIFO is full at the time of a write, an OVERFLOW will be set in the STATUS Register. Similarly, if Receive FIFO is empty at the time of a read, an UNDERFLOW will be generated.

**Table 118 • DATA8**

Bit Number	Name	R/W	Reset Value	Description
[7:0]	DATA8	R/W	0x00	Write: Writes a byte to the HPMS COMM_BLK Transmit FIFO Read: Reads a byte from the HPMS COMM_BLK Receive FIFO

When the DATA8 register is written, the command bit (Bit 8 on DATA) is set to 0, indicating that it is data. Writes to this register automatically set the SIZETX to 0 (1 byte), and reads set the SIZERX to 0 (1 byte).

## 8.5.5 Word Data Register

This register writes a word (32 bits) to the Transmit FIFO or reads a word from the Receive FIFO. If the Transmit FIFO has less than 4 spaces available at the time of a write, an OVERFLOW will be set in the STATUS register.

Similarly, if Receive FIFO has less than 4 bytes available at the time of a read, an UNDERFLOW will be generated.

**Table 119 • DATA32**

Bit Number	Name	R/W	Reset Value	Description
[31:0]	DATA32	R/W	0x00000000	Write: Writes a word to the HPMS COMM_BLK Transmit FIFO Read: Read a word from the HPMS COMM_BLK Receive FIFO

The LSB is transferred on the DATA bus first. When the DATA32 register is written, the command bit (Bit 8 on DATA) is set to 0, indicating that it is data. Writes to this register automatically set the SIZETX to 1 (4 bytes) and reads set the SIZERX to 1 (4 bytes).

## 8.5.6 Frame/Command Byte Register

This register writes a byte to the Transmit FIFO or reads a byte from the Receive FIFO. If the Transmit FIFO is full at the time of a write, an OVERFLOW will be set in the STATUS register. Similarly, if receive FIFO is empty at the time of a read, an UNDERFLOW will be generated.

**Table 120 • FRAME\_START8**

Bit Number	Name	R/W	Reset Value	Description
[7:0]	FRAME_START8	R/W	0x00	Write: Writes byte to the HPMS COMM_BLK transmit FIFO Read: Read a byte from the HPMS COMM_BLK receive FIFO

When the FRAME\_START8 register is written, the command bit (Bit 8 on DATA) is set to 1, indicating the start of a frame, that is, the command byte. Writes to this register automatically set the SIZETX to 0 (1 byte), and reads set the SIZERX to 0 (1 byte). The STATUS register bit 7 indicates that this byte is a command.

## 8.5.7 Frame/Command Word Register

This register writes a word (32-bits) to the Transmit FIFO or reads a word from the Receive FIFO. If the Transmit FIFO has less than four spaces available at the time of a write, an OVERFLOW will be set in STATUS register. Similarly, if Receive FIFO has less than four bytes available at the time of a read, an UNDERFLOW will be generated.

**Table 121 • FRAME\_START32**

Bit Number	Name	R/W	Reset Value	Description
[31:0]	FRAME_START32	R/W	0x00000000	Write: Writes a word to the HPMS COMM_BLK transmit FIFO Read: Reads a word from the HPMS COMM_BLK receive FIFO

The least significant bit (LSB) is transferred on the DATA bus first. When the FRAME\_START32 register is written, the command bit is set to 1, indicating the start of a frame, that is, command byte. The command bit (Bit 8 on DATA) will be set on the first byte for writes.

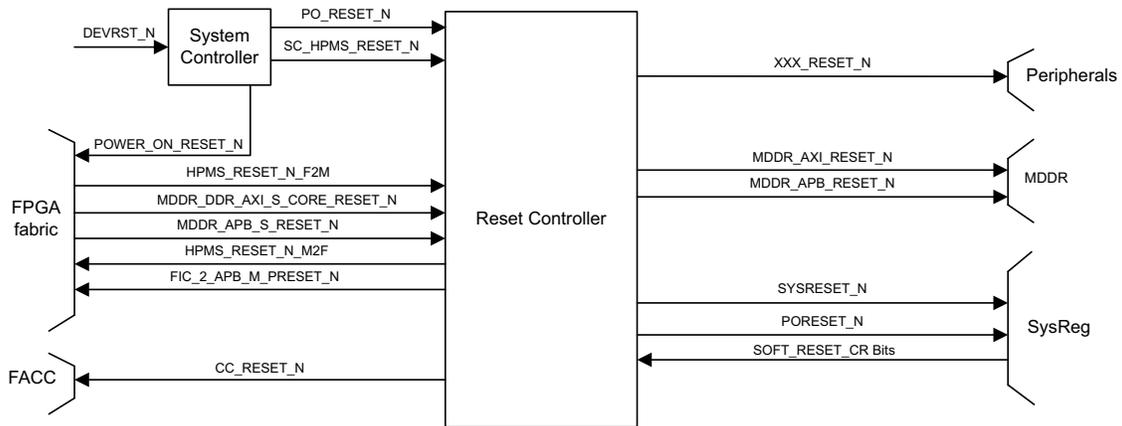
Writes to this register automatically sets the SIZETX to 1 (4 bytes) and reads set the SIZERX to 1 (4 bytes). The STATUS register bit 7 indicates that this word is a command.

## 9 Reset Controller

The reset controller manages the asynchronous reset requests coming from various sources and generates a synchronous reset for the entire High Performance Memory Subsystem (HPMS) or individual resets to the HPMS sub-blocks and user logic in the FPGA fabric.

The reset controller drives resets to various modules of the IGLOO2 devices, such as the MDDR subsystem, FPGA fabric, clock controller, SYSREG, and peripherals. The following figure shows the reset controller block diagram with various reset inputs/outputs from/to various HPMS blocks.

**Figure 106 • Reset Signals Distribution in IGLOO2 Devices**



Microsemi recommends to use the CoreResetP IP for initializing the user design in IGLOO2 devices. The CoreResetP handles sequencing of reset signals. It is available in the Libero SoC IP catalog.

System Builder is a powerful design tool within the Libero SoC Design Environment that helps the user capture the system-level requirements and produces a design implementing those requirements. A very important function of the System Builder is the automatic creation of the initialization sub-system (all required cores are instantiated, and connections are made automatically).

## 9.1 Functional Description

### 9.1.1 Power-On Reset Generation Sequence

The following figure shows the conceptual block diagram of power-on reset generation. The POR generator block in System Controller generates a power-on reset signal, PO\_RESET\_N.

Figure 107 • Power-On Reset Generation Block Diagram

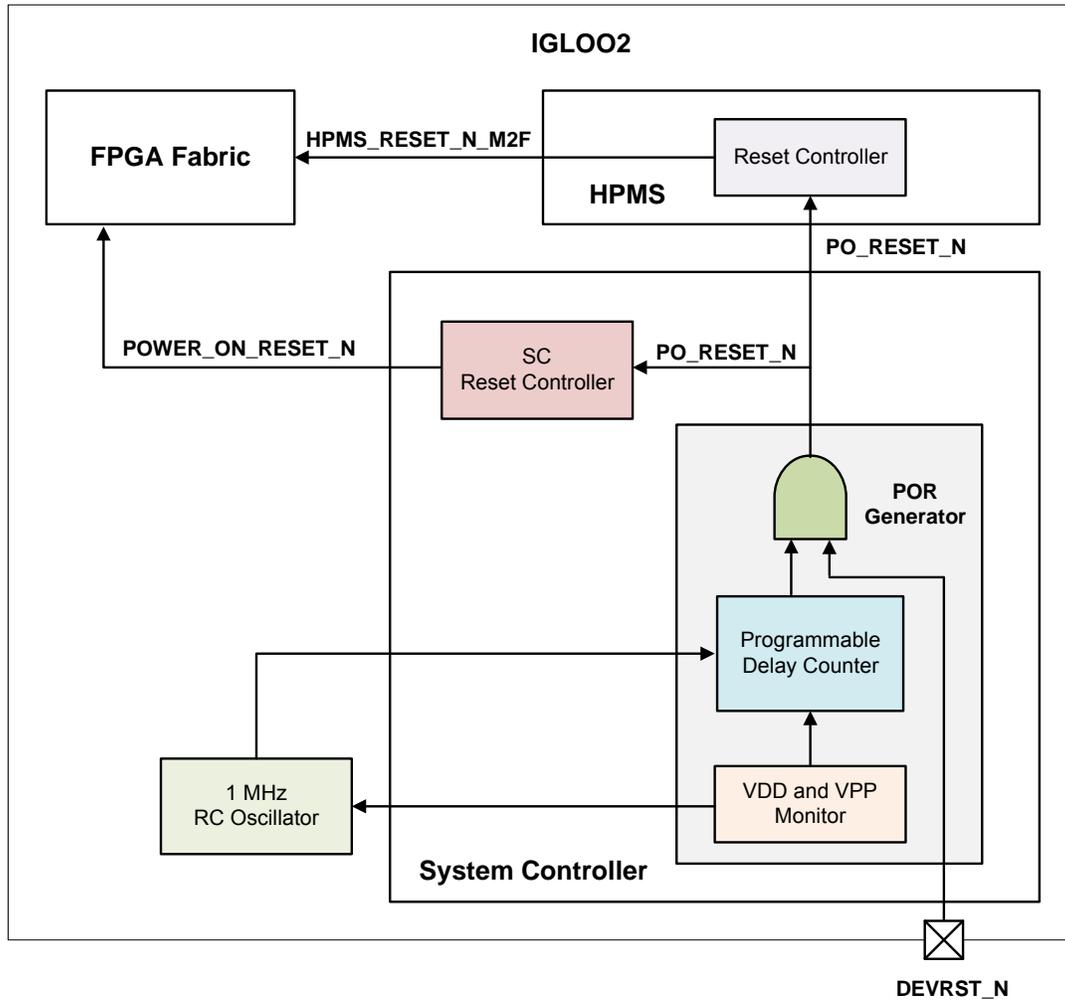
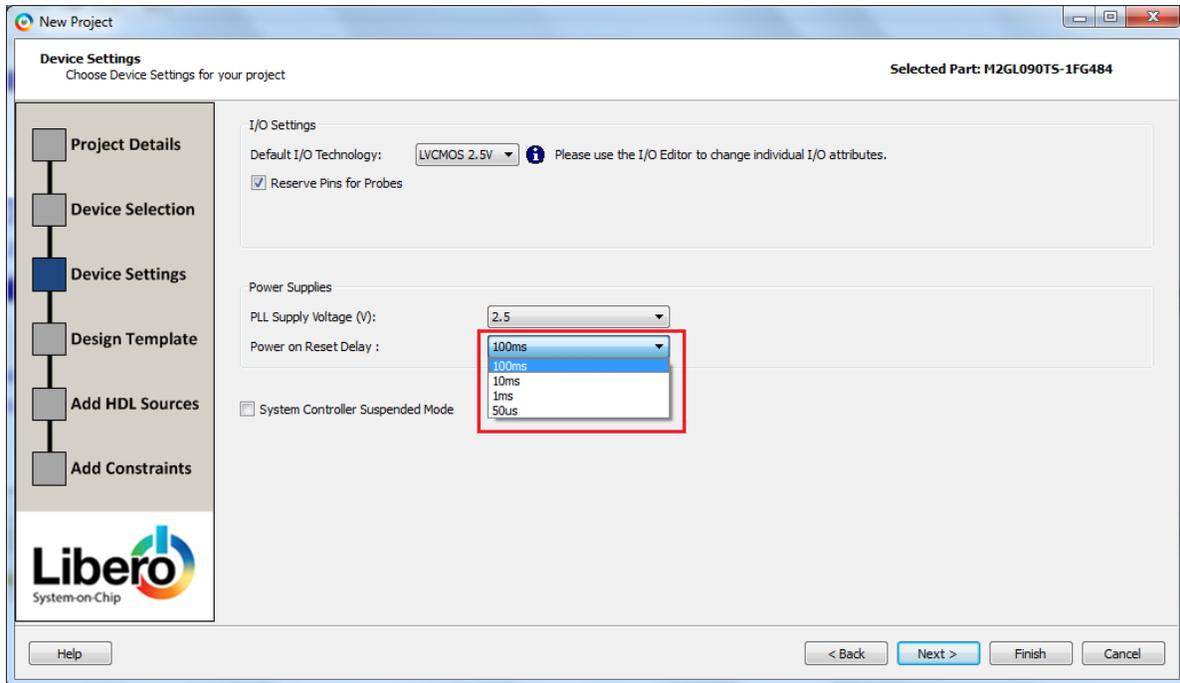


Figure 110, page 180 shows the power up to functional time sequence diagram. On power-up, the VDD and VPP monitor blocks in the POR generator block assert a power-on reset signal, PO\_RESET\_N. If the VDD and VPP supplies reach their threshold point (VDD ~ 0.9 V, VPP ~ 0.9 V), the 1 MHz RC Oscillator is turned-on, which provides the clock to the programmable delay counter.

The delay can be configured to 50  $\mu$ s, 1 ms, 10 ms, or 100 ms in the New Project window (Device Settings) while creating the Libero SoC project as shown in the following figure. You can also access and change this setting after the project has been created from the Project Settings window (Project > Project Settings...).

**Figure 108 • Power-On Reset Delay Configuration**



The delay counter is used to generate the power supply rise time. All power supplies must be stable within the configured Power on Reset Delay. When the counter reaches its maximum value, the PO\_RESET\_N signal is de-asserted. Upon de-assertion of the PO\_RESET\_N signal, the 1 MHz RC oscillator is gated off and the 50 MHz RC oscillator is enabled and the System Controller starts operating at 50 MHz clock. Then System Controller starts the initialization sequence of I/O banks, HPMS, and FPGA Fabric Subsystem.

The POWER\_ON\_RESET\_N signal is generated from the PO\_RESET\_N signal and can be used in the user design as a reset for the FPGA fabric logic. It is an active low output signal. It is made available by instantiating the SYSRESET macro from the Libero SoC IP catalog in SmartDesign or by instantiating the SYSRESET macro directly in the HDL file.

The following figure shows a block symbol of the SYSRESET macro that exposes the POWER\_ON\_RESET\_N signal.

**Figure 109 • SYSRESET Macro**



**Note:** When DEVRST\_N is low, all user I/Os are fully tri-stated. However, if the JTAG I/Os are still enabled, they cannot be used as the TAP controller in reset.

POWER\_ON\_RESET\_N asserts on the following events:

- Power-up event
- Assertion of DEVRST\_N

- Completion of programming
- Completion of zeroization

A dedicated input-only reset pad (DEVRST\_N) is present on all the IGLOO2 devices, which cause assertion to the PO\_RESET\_N signal. If an external reset circuit is connected to the DEVRST\_N pin, it increases the power up to functional time due to the delays that the external reset device does add.

DEVRST\_N is an asynchronous reset pin and must be asserted only when the device is unresponsive due to some unforeseen circumstances. It is not recommended to assert the DEVRST\_N pin during programming operation, which might cause severe consequences including corrupting the device configuration. For more details on DEVRST\_N timing information, refer to the [DS0128: IGLOO2 and SmartFusion2 Datasheet](#).

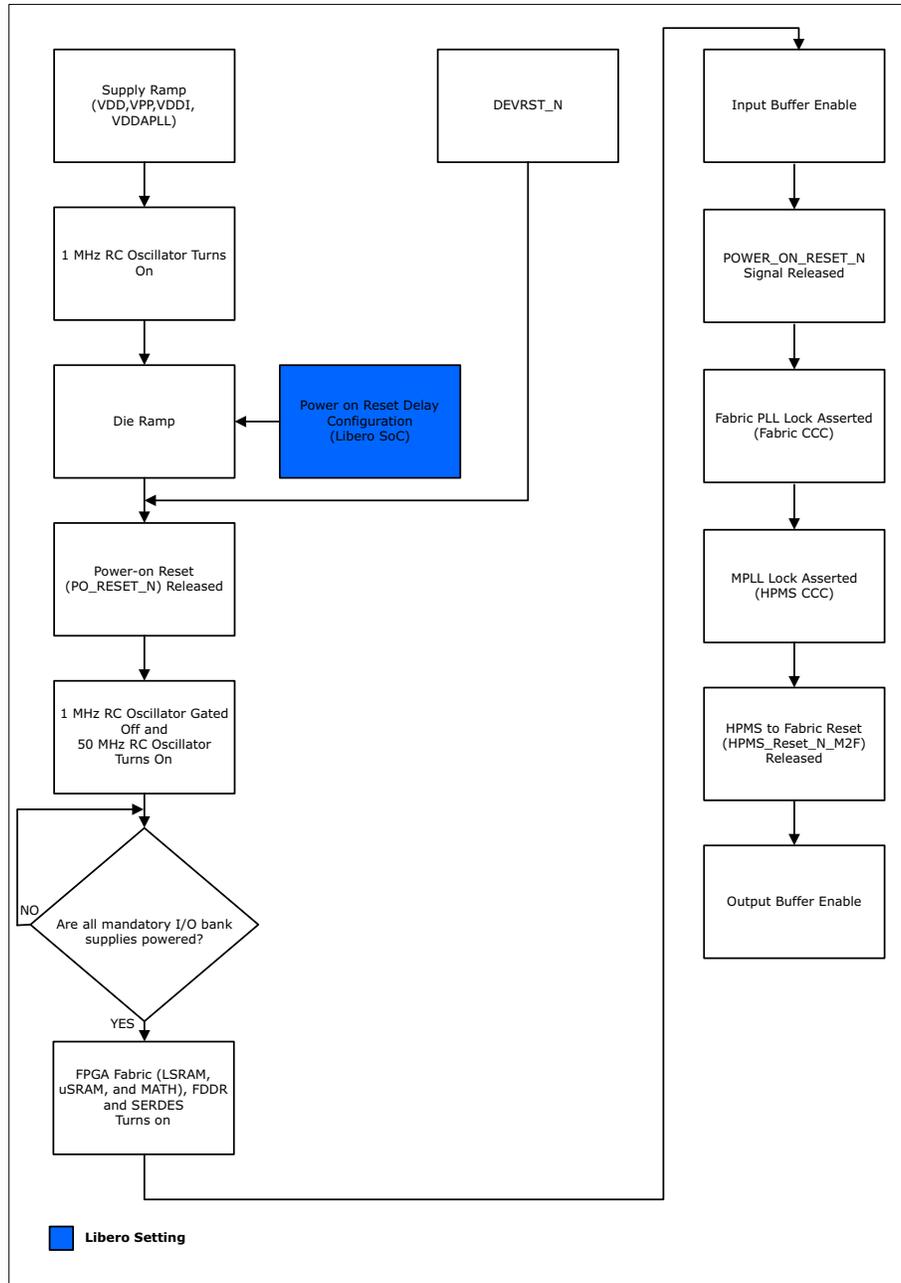
Asserting DEVRST\_N does not enable the delay counter (Power on Reset Delay) in the POR circuitry.

The delay counter is operational only at power-up. When DEVRST\_N is low, all user I/Os are fully tri-stated. Although, the JTAG I/Os are still enabled, they cannot be used as the TAP controller is in reset. The SYSRESET macro is not required to be instantiated to enable the DEVRST\_N pin in the user design. DEVRST\_N is a dedicated input-only reset pad available on all the IGLOO2 devices.

## 9.1.2 Power-Up to Functional Time Sequence

The following figure shows the power-up to functional time sequence diagram.

**Figure 110 • Power-Up to Functional Time Sequence Diagram**



The power up to functional sequence is as follows:

- Supply Ramp (VDD, VPP, VDDI, and VDDAPLL) - There is no specific power up or power down sequencing requirement for IGLOO2 devices. The I/O banks can be brought-up in any order, before or after the core voltage. However, the device is only functional if all I/O bank supplies are powered-up.
- On power-up, the POR generator block asserts the PO\_RESET\_N signal, which is not accessible for users.

- The 1 MHz RC Oscillator is turned-on, which provides the clock to the programmable delay counter. When the counter reaches its maximum value, the PO\_RESET\_N signal is de-asserted.
- The 1 MHz RC oscillator is gated off and the 50 MHz RC oscillator is enabled and the System Controller starts operating at 50 MHz clock.
- All mandatory I/O bank supplies must be powered-up. For all the IGLOO2 devices, some of the bank supplies (VDDIx) must always be powered, even if associated bank I/Os are in unused condition. For the list of mandatory I/O bank supplies, refer to Table 2 and Table 3 in the *AC393: SmartFusion2 and IGLOO2 Board Design Guidelines Application Note*.
- FPGA fabric (LSRAM, uSRAM, and MATH), FDDR, and SerDes are turned-on.
- Input buffer is enabled.
- POWER\_ON\_RESET\_N signal (generated from the PO\_RESET\_N signal) is released. This signal can be used in the design as a reset for the FPGA fabric logic.
- Fabric PLL (Fabric CCC) Lock is asserted.
- MPLL (HPMS CCC) Lock is asserted.
- HPMS to Fabric Reset (HPMS\_RESET\_N\_M2F) is released.
- Output buffer is enabled.

### 9.1.3 Power-Up to Functional Time Data

This section describes power-up to functional time data based on DEVRST\_N assertion and VDD ramp up.

#### 9.1.3.1 Parameters Used for Obtaining Power-Up to Functional Time Data

This section describes the parameters used for obtaining power-up to functional time data. Following are the test conditions:

- Power-on-reset delay setting: 1 ms
- Supply ramp rate: 5  $\mu$ s
- Measurement temperature: 25 °C

Power-on-reset delay setting indicates how long VDD takes to ramp up. The programmable delay counter starts counting based on the power-on-reset delay setting, oscillator frequency, and period variability to ensure that the supplies have reached their minimum operating levels.

Supply ramp rate indicates the ramp up rate of the on-board VDD core voltage, VPP charge pump voltage, and VDDA phase-locked loop (PLL) analog voltage supplies.

**Note:** In all the test cases, I/Os are configured as LVCMOS25, which is the default I/O standard in Libero along with the other default I/O attribute settings.

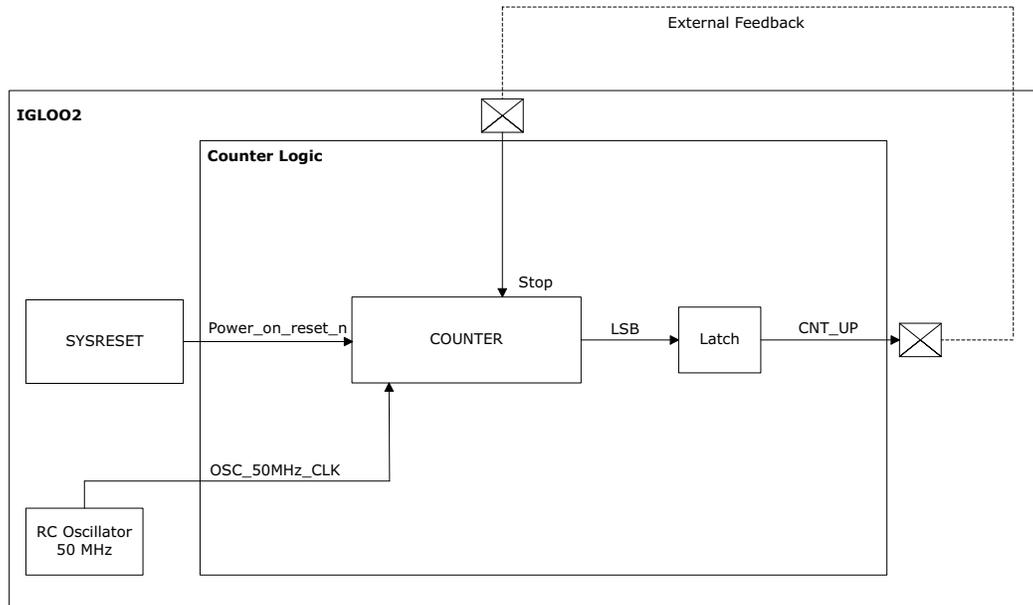
#### 9.1.3.2 VDD Power-Up to Functional Time Data

The core supply voltage VDD is connected to the appropriate source and VDD is monitored by the power-on-reset circuitry to check if it reaches the minimum threshold value and initiates the system controller to release the device from reset. This scenario provides with the power-up to functional time data only when the FPGA fabric and the FPGA I/O are used with all supplies ramped up except VDD, which is ramped up at the last. The required power-on-reset delay is set using the Libero tool.

A fabric counter is operated when POWER\_ON\_RESET\_N is de-asserted, as shown in the following figure. The least significant bit (LSB) of the counter output is connected to a latch and given to an output buffer, which is then connected to an input buffer of the fabric using an external loopback. This input is used for stopping the counter from incrementing. The counter stops as soon as the counter's LSB bit transitions to logic high. The power-up to functional time is measured from the VDD supply ramp to transition of the fabric buffer output.

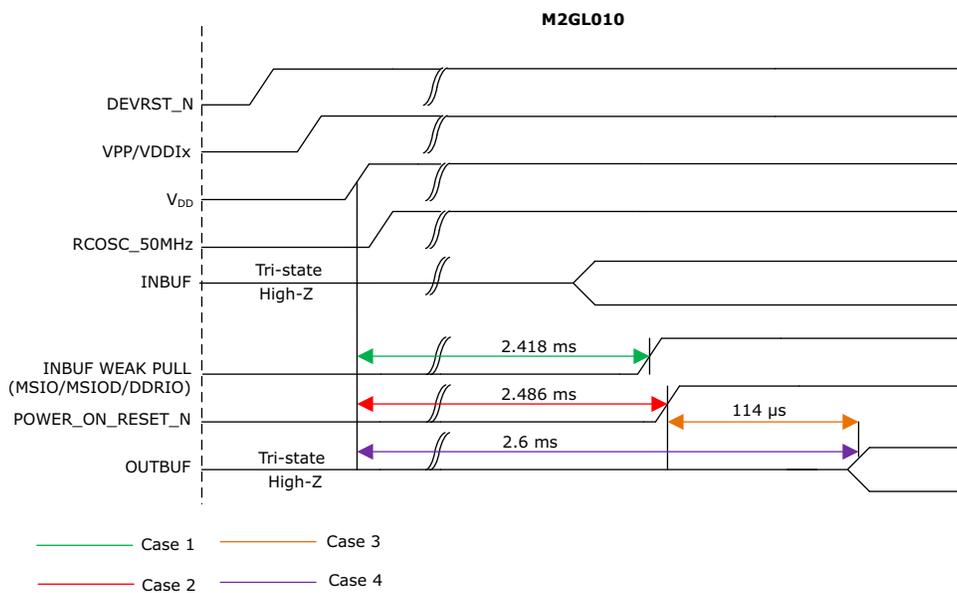
The following figure shows the characterization test design setup used for obtaining the VDD power-up to functional timing values.

**Figure 111 • VDD Power-Up to Functional Time Design Setup**



The following figure shows the behavior of different signals when VDD is ramped up with a power-on-reset delay of 1 ms from 0 V to minimum threshold level and HPMS is not used with VDD = 1.2 V, VDDI = 2.5 V, Tj = 25 °C, and power on reset delay setting = 1 ms.

**Figure 112 • VDD Power-Up to Functional Timing Diagram**



The following table lists the power-up to functional time of M2GL005, M2GL010, M2GL025, M2GL050, M2GL060, M2GL090, and M2GL150 devices.

**Table 122 • VDD Power-Up to Functional Time**

Test Cases	Start Point	End Point	Description	Power-Up to Functional Time ( $\mu$ s)						
				005	010	025	050	060	090	150
Case1	POWER_ON_ RESET_N	Output available at I/O	Fabric to output	114	114	114	113	114	114	114
Case 2	VDD	Output available at I/O	VDD at its minimum threshold level to output	2587	2600	2607	2558	2591	2600	2699
Case 3	VDD	POWER_ON_ RESET	VDD at its minimum threshold level to fabric	2474	2486	2493	2445	2477	2486	2585

**Note:** Time taken for different power on reset delay settings can be calculated as shown in the following equation.

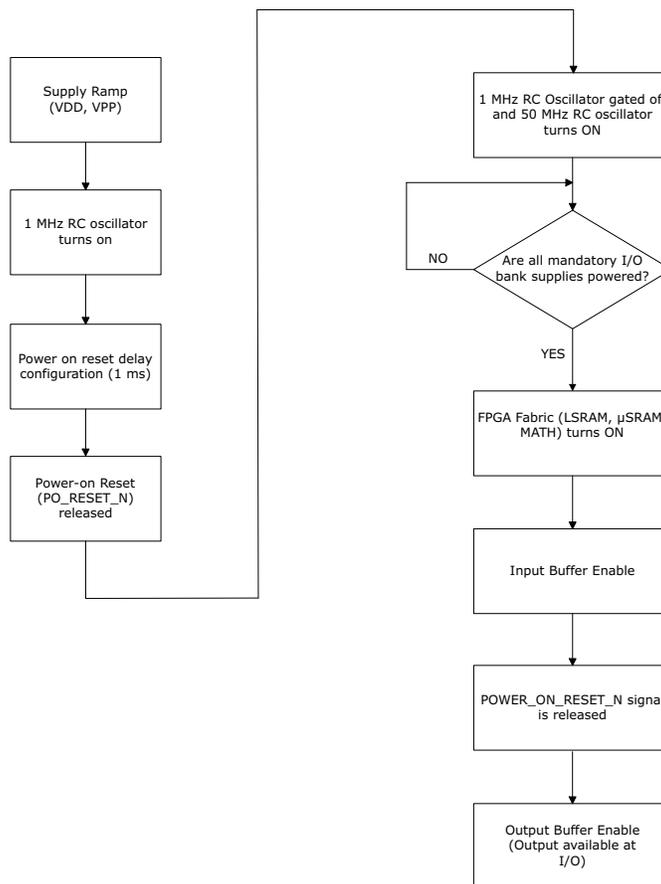
$$(\text{Test case} - 2000 \mu\text{s}) + 2 \times \text{Power on Reset Delay Setting}$$

For example, if the power-on reset delay setting of 100 ms is used in an M2S010 device, then the VDD at its minimum threshold level to output is calculated as follows:

$$(2600 - 2000 \mu\text{s}) + 2 \times 100\text{ms} = 200.6\text{ms}$$

The following figure shows the stages that contribute to VDD power-up to functional time for IGLOO2.

**Figure 113 • VDD Power-Up to Functional Time Flow**



**Note:** Power-up to functional time depends on power-on-reset delay settings, 1 MHz oscillator frequency, and period variability. At times, it is approximately equal to twice the power-on-reset delay settings.

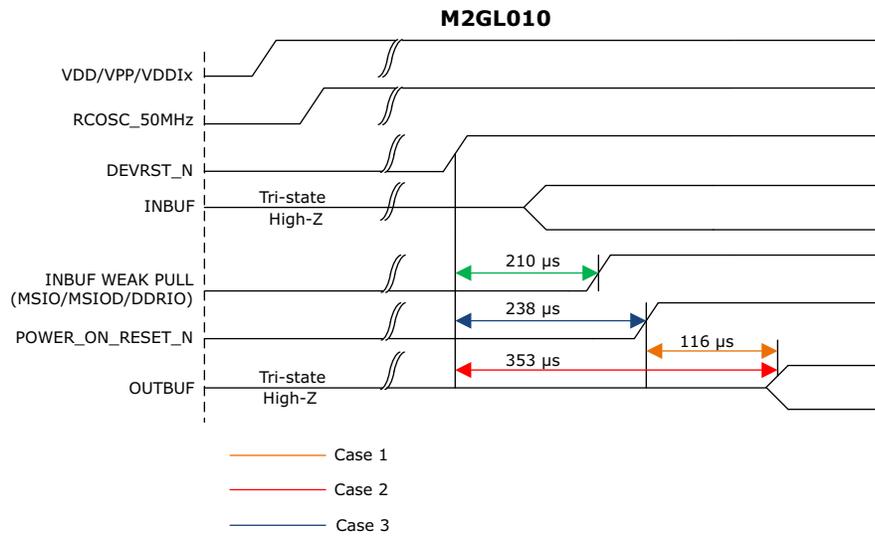
### 9.1.3.3 DEVRST\_N Power-Up to Functional Time

This scenario provides you with power-up to functional time data with respect to DEVRST\_N when the FPGA fabric, the FPGA I/O, and external oscillator are used. The design setup is same as the VDD power-up to functional time shown in [Figure 112](#), page 182.

**Note:** Assert DEVRST\_N pin during programming (including eNVM), as it corrupts the device configuration. For more information on proper usage of the DEVRST\_N pin, see the [AC393: Board Design Guidelines for SmartFusion2 SoC and IGLOO2 FPGAs Application Note](#).

The following figure shows the behavior of different signals when DEVRST\_N is asserted and HPMS is not used with VDD = 1.2 V, VDDI = 2.5 V, and Tj = 25 °C.

**Figure 114 • DEVRST\_N Power-Up to Functional Timing Diagram**



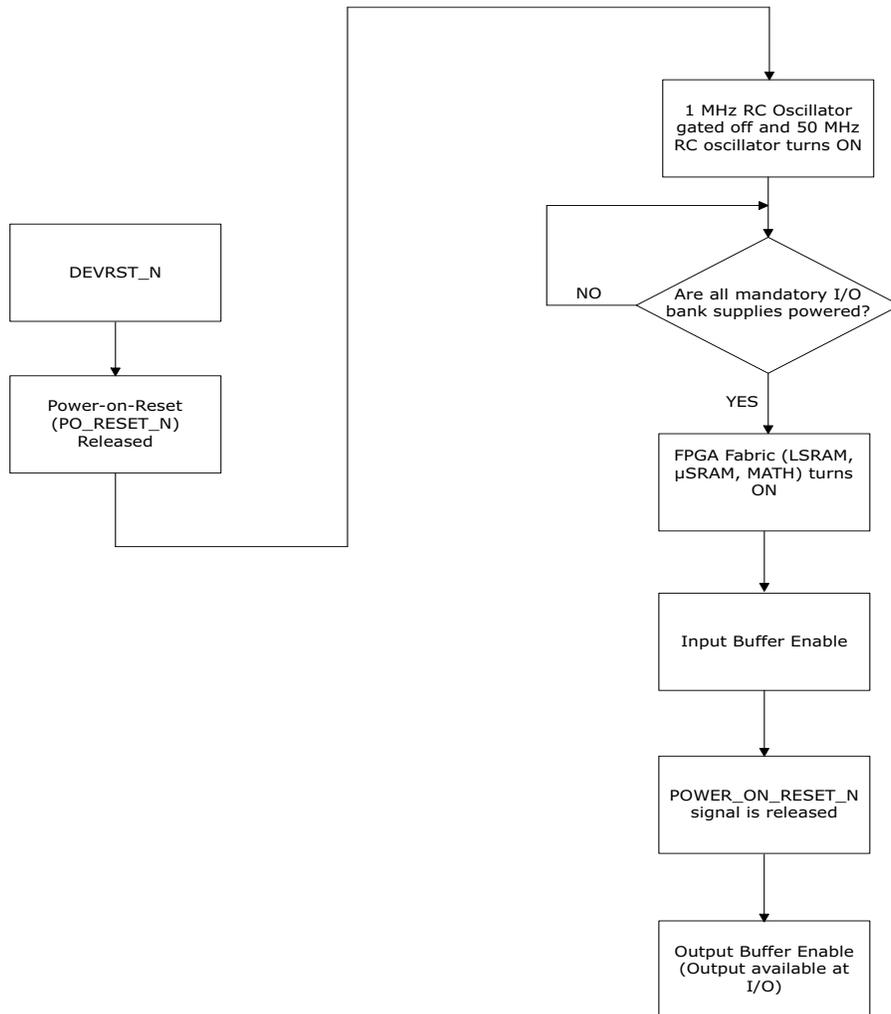
The following table lists the power-up to functional time of M2GL005, M2GL010, M2GL025, M2GL050, M2GL060, M2GL090, and M2GL150 devices without HPMS clock ranging from 3 MHz to 166 MHz.

**Table 123 • DEVRST\_N Power-Up to Functional Time**

Test Cases	Start Point	End Point	Description	Power-Up to Functional Time (μs)						
				005	010	025	050	060	090	150
Case1	POWER_ON_R ESET_N	Output available at I/O	Fabric to output	114	116	113	113	115	115	114
Case 2	DEVRST_N	Output available at I/O	DEVRST_N to Output	314	353	314	307	343	341	341
Case 3	DEVRST_N	POWER_ON_RESET_N	DEVRST_N to fabric	200	238	201	195	230	229	227

The following figure shows the stages that contribute to DEVRST\_N power-up to functional time for IGLOO2.

**Figure 115 • DEVRST\_N Power-Up to Functional Time Flow**



**Note:** All timing numbers in [Table 122](#), page 183 and [Table 123](#), page 185 are for worst case conditions.

In order to simplify the task of initializing a user design in IGLOO2 devices, Microsemi provides a CoreResetP soft reset controller IP. The CoreResetP handles sequencing of reset signals in IGLOO2 devices. The CoreResetP generates a fabric reset signal whenever POWER\_ON\_RESET\_N is asserted or HPMS\_RESET\_N\_M2F is asserted. It is available in the Libero SoC IP catalog. Refer to **CoreResetP Soft Reset Controller** for more information.

**Note:** Microsemi recommends using the System Builder which automatically creates the initialization sub-system (all required cores are instantiated, and connections are made automatically).

The reset controller receives three types of reset requests:

- Power-on reset request from system controller
- System reset request from system controller, and FPGA fabric
- Block reset requests from FPGA fabric and SYSREG

## 9.1.4 Power-On Reset

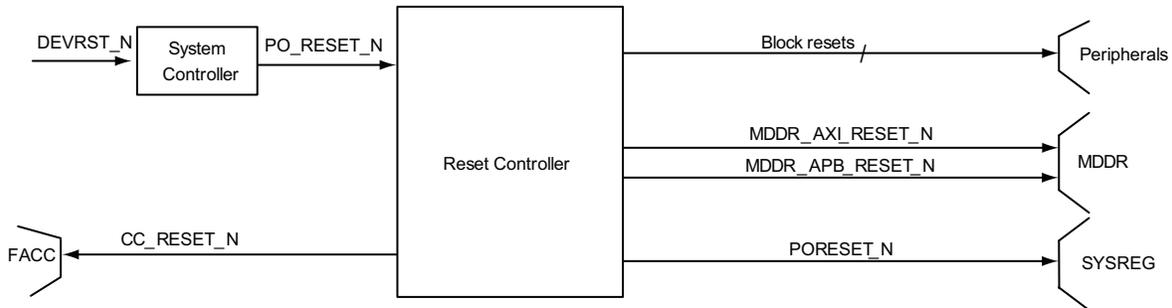
The Reset Controller receives a power-on reset signal, `PO_RESET_N`, from the System Controller, which is a cold reset signal. Its assertion initializes the IGLOO2 device to its default reset state.

`PO_RESET_N` signal is fed to the System Register block (SYSREG). The `PO_RESET_DETECT` bit of the `RESET_SOURCE_CR` register (defined in Table 124, page 195) in the SYSREG block is set or reset depending on the `PORESET_N` signal.

The `PORESET_N` signal is a synchronized version of the `PO_RESET_N` signal on `HPMS_CLK`.

The reset controller generates different synchronized resets to the HPMS and the FPGA fabric on the assertion of `PO_RESET_N`, as shown in the following figure.

**Figure 116 • Reset Controller During Power-On Reset**



The `CC_RESET_N` is generated on the assertion of `PO_RESET_N`. This is a power-on reset signal to the fabric aligned clock controller (FACC).

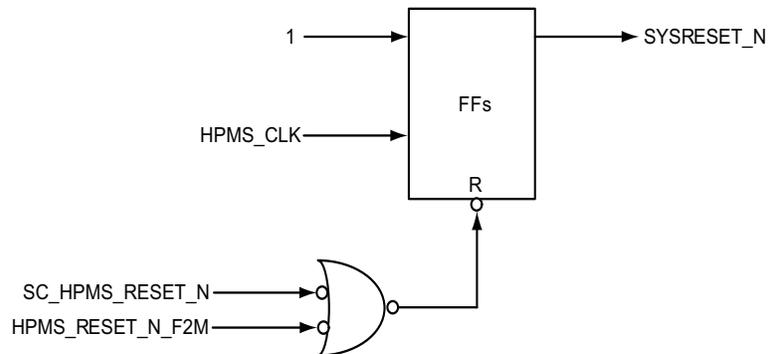
## 9.1.5 System Reset

The system reset (`SYSRESET_N`) is generated if any of the following conditions are true:

- `SC_HPMS_RESET_N` is asserted from the System Controller during the start-up sequence after power-up.
- `HPMS_RESET_N_F2M` is asserted from the FPGA fabric interface.

The generation of `SYSRESET_N` is shown in the following figure.

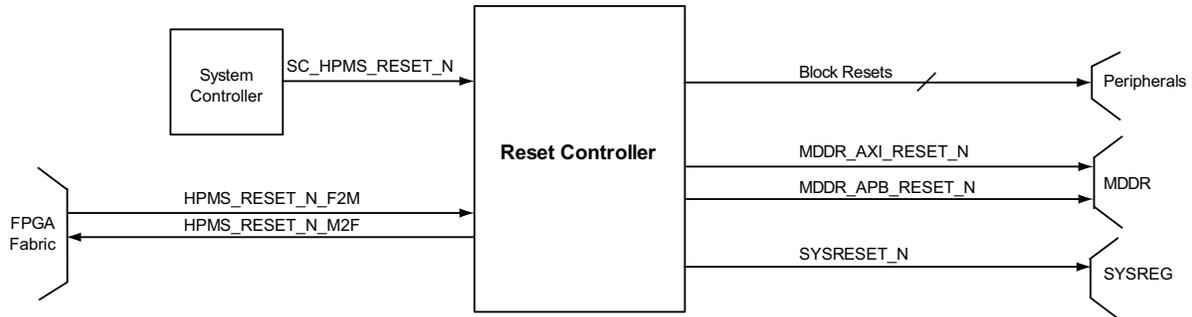
**Figure 117 • SYSRESET\_N Generation**



The inputs `SC_HPMS_RESET_N` and `HPMS_RESET_N_F2M` are first synchronized on `HPMS_CLK` and then combined. The `HPMS_RESET_N_F2M` signal can be used to reset the HPMS, independently of any resets coming from the HPMS itself. For example it may be asserted as a result of an external reset event from an off-chip reset controller, using an I/O pad to bring the reset input into the fabric.

The following figure shows the various reset signals to the HPMS blocks which are generated from reset controller on assertion of SYSRESET\_N. It also shows the reset inputs to the reset controller, which cause the generation of SYSRESET\_N.

**Figure 118 • Reset Controller During SYSRESET\_N**



SYSRESET\_N resets all modules in the HPMS. Some of the register bits within the SYSREG block are reset by SYSRESET\_N.

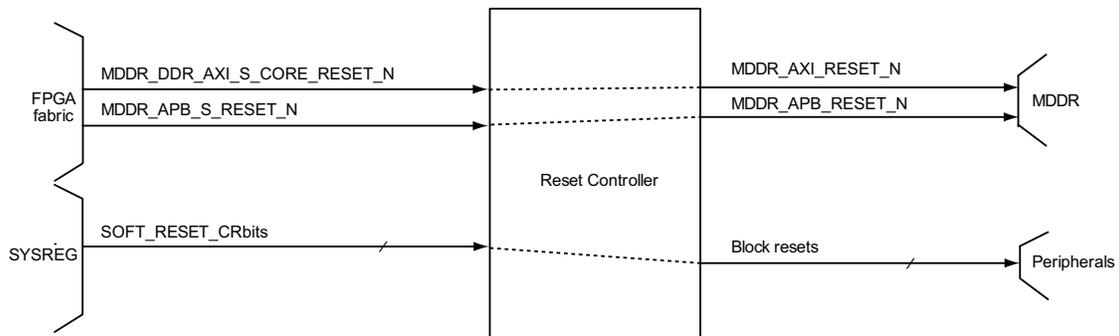
### 9.1.6 Block Resets

The reset controller generates block level resets for all modules except the AHB bus matrix, fabric interface interrupt controller (FIIC), and SYSREG.

These blocks will be reset at power-on reset or system reset. The reset controller receives block enable bits and soft reset requests (SOFT\_RESET\_CR bits) for various blocks within the HPMS from SYSREG to control the block level reset generation.

The following figure shows the block level resets from the reset controller along with the source of the resets.

**Figure 119 • Reset Controller With Only Block Level Resets**



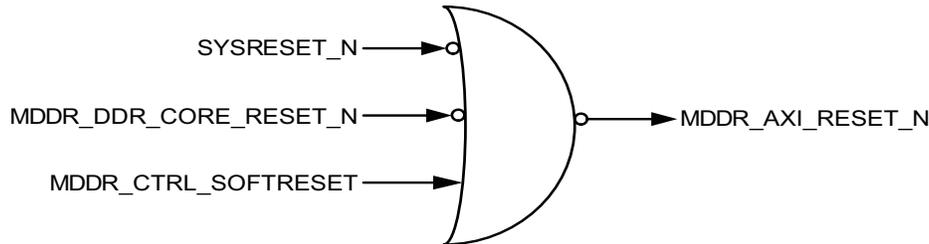
## 9.1.6.1 MDDR Resets

### 9.1.6.1.1 MDDR\_AXI\_RESET\_N

MDDR\_AXI\_RESET\_N is generated from FPGA fabric reset input (MDDR\_DDR\_CORE\_RESET\_N), SYSRESET\_N, and the MDDR soft reset (MDDR\_CTRL\_SOFTRESET) from SYSREG.

The following figure shows the generation of MDDR\_AXI\_RESET\_N.

**Figure 120 • MDDR\_AXI\_RESET\_N Generation**

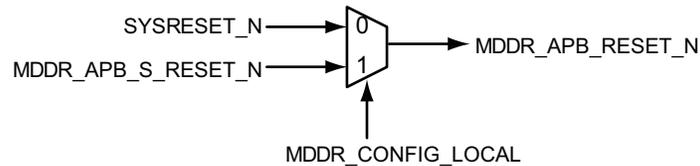


The reset controller drives a synchronized reset to AXI logic in the MDDR.

### 9.1.6.1.2 MDDR\_APB\_RESET\_N

MDDR\_APB\_RESET\_N is generated from FPGA fabric PRESET input (MDDR\_APB\_S\_RESET\_N) or SYSRESET\_N, based on the selection of MDDR\_CONFIG\_LOCAL in SYSREG. The MDDR\_CONFIG\_LOCAL bit is in the MDDR Configuration Register (MDDR\_CR as defined in [Table 124](#), page 195) of SYSREG. The following figure shows the generation of MDDR\_APB\_RESET\_N.

**Figure 121 • MDDR\_APB\_RESET\_N Generation**

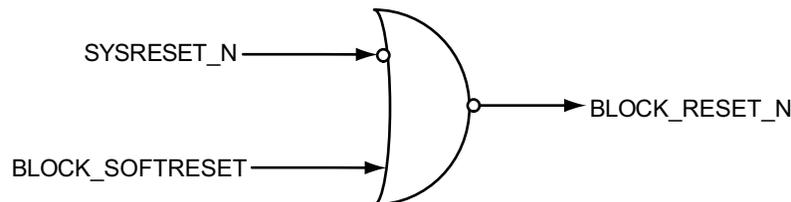


The reset controller drives a synchronized reset to the APB logic of the MDDR subsystem.

## 9.1.6.2 Reset Generation to HPMS Peripherals

The reset controller generates block level resets for the peripherals present within the HPMS. The block level reset generation is shown in the following figure.

**Figure 122 • Block Level Reset Generation**



The reset signal is asserted if any of the following conditions is true:

- SYSRESET\_N asserted
- Block level Soft reset (SOFT\_RESET\_CR) request asserted from SYSREG module.

The reset controller can generate the reset to ENVM\_0, ENVM\_1(if present), ESRAM\_0, ESRAM\_1, PDMA, HPDMA, COMM\_BLK, FIC\_0, FIC\_1 (if present), and the FPGA fabric (HPMS\_RESET\_N\_M2F reset).

## 9.2 CoreResetP Soft Reset Controller

The following Reset sub-systems in IGLOO2 devices must be sequenced properly for the overall system to function correctly.

- Chip Boot (System Controller)
- Fabric
- HPMS
- FIC sub-systems (HPMS to Fabric and Fabric to HPMS)
- Peripherals - MDDR, FDDR and SERDES\_IF

CoreResetP Soft Reset Controller gathers various reset signals from System Controller, HPMS, and FPGA fabric and generates new synchronized reset signals to handle the sequencing of reset signals of various subsystems in IGLOO2 devices. CoreResetP helps managing the following:

- FIC sub-systems resets: Both HPMS and FPGA fabric should be out of reset to establish the communication between them. CoreResetP generates HPMS\_READY signal which indicates that both HPMS and FPGA fabric are out of reset and ready for communication.
- Peripherals initialization: It generates reset signals to initialize MDDR, FDDR, and SERDES\_IF peripheral blocks.
- Peripherals reconfiguration: Individual reset controls via CoreConfigP Soft core.
- PCIe L2/P2 (in-band) and PRST# (out-band): Low power modes for all devices, except M2GL090.

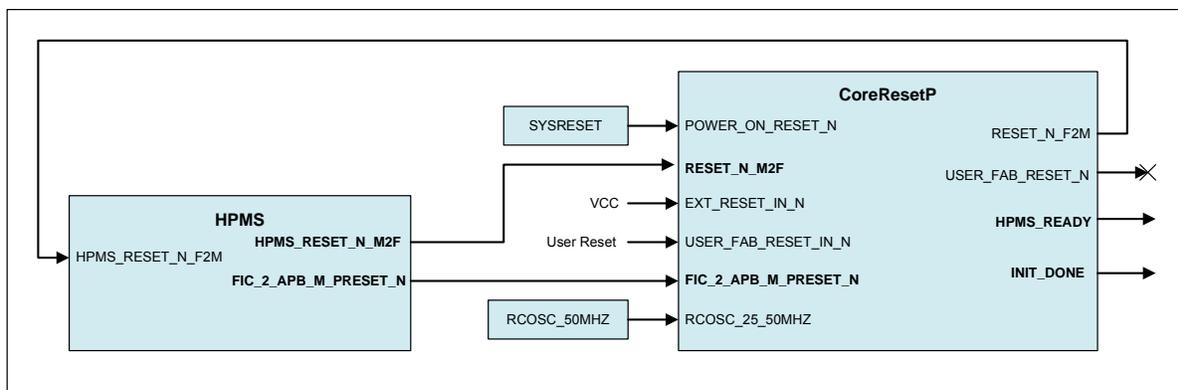
### 9.2.1 Reset Topology

This section describes the reset topology that needs to be applied to the user design.

#### 9.2.1.1 HPMS\_READY Generation

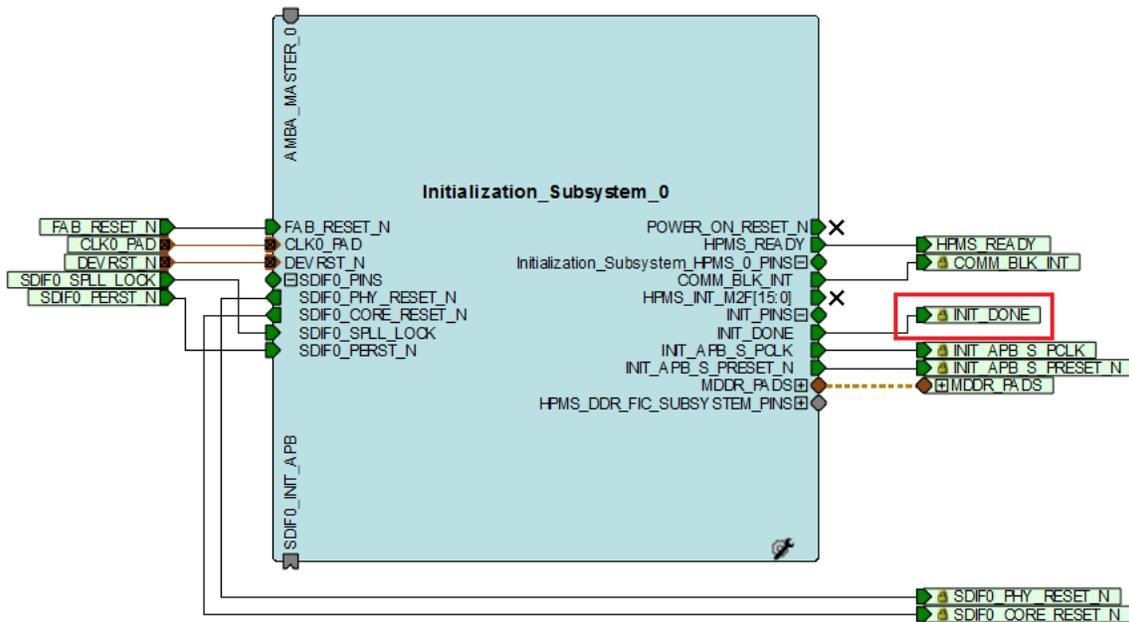
Any design which consists of HPMS and a fabric subsystem must be synchronized to establish the communication between them. When HPMS is doing any transaction, the fabric should be ready. Similarly, when fabric is doing any transaction, the HPMS should be ready. The following figure illustrates the typical FIC subsystem with CoreResetP connectivity. CoreResetP generates HPMS\_READY signal, which indicates that HPMS is ready for communication. HPMS\_READY signal is generated whenever a cold reset (power-up event or assertion of DEVRST\_N) occurs or due to HPMS reset like assertion of HPMS\_RESET\_N\_F2M. CoreResetP relies on HPMS\_RESET\_N\_M2F and FIC\_2\_APB\_M\_PRESET\_N signal to generate HPMS\_READY signal.

**Figure 123 • HPMS\_READY Signal Generation**



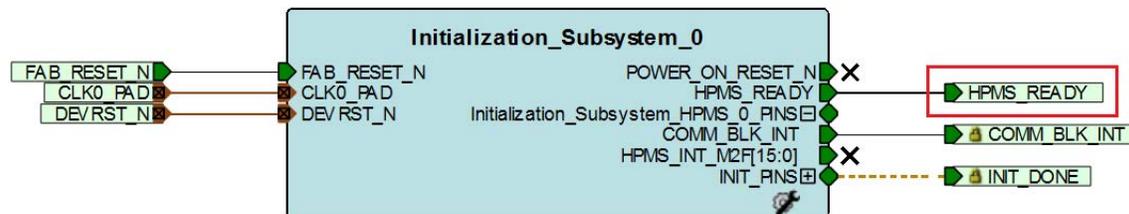
If the System Builder is used to generate the Libero project, all required cores are instantiated, and connections are made automatically. If the user logic consists of any of the two DDR controllers (FDDR or MDDR) or Serial High speed controller (SERDES\_IF), the INIT\_DONE signal should be used to reset the fabric subsystem. The following figure shows the system builder generated design with MDDR and a SERDES\_IF interfaces.

Figure 124 • System Builder-Generated Design with MDDR and SERDESIF Interfaces



If none of MDDR/FDDR/SerDes is used, the HPMS\_READY signal should be used to reset the fabric subsystem. The following figure shows the system builder generated design without MDDR/FDDR/SerDes interface.

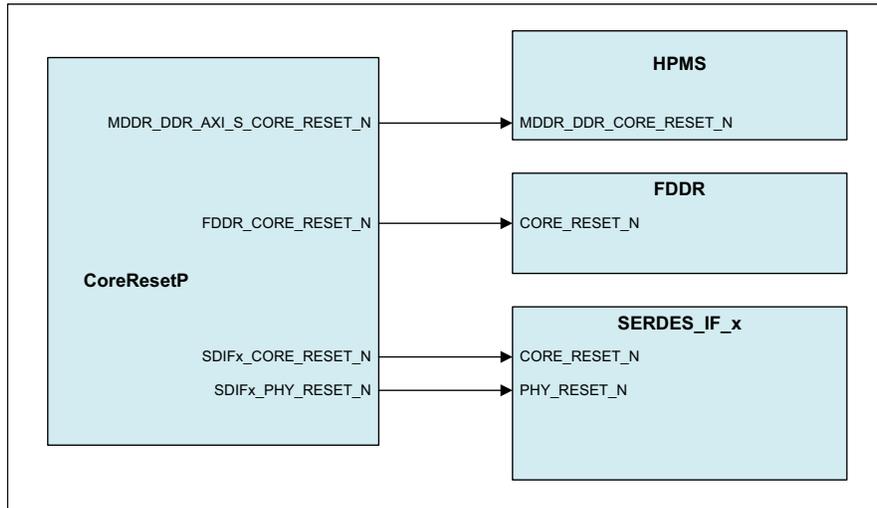
Figure 125 • System Builder-Generated Design without MDDR/FDDR/SerDes Interface



### 9.2.1.2 Peripheral Initialization

CoreResetP generates reset signals to initialize MDDR, FDDR, and SERDES\_IF peripheral blocks. The following figure shows the CoreResetP connectivity with peripheral resets. For each SERDES\_IF block, the CoreResetP generates SDIFx\_PHY\_RESET\_N and SDIFx\_CORE\_RESET\_N signals that need to be connected to SERDES\_IF macro on PHY\_RESET\_N and CORE\_RESET\_N respectively. For FDDR and MDDR, the CoreResetP generates CORE reset signals (FDDR\_CORE\_RESET\_N and MDDR\_DDR\_AXI\_S\_CORE\_RESET\_N).

**Figure 126 • CoreResetP Connectivity with Peripheral Resets**

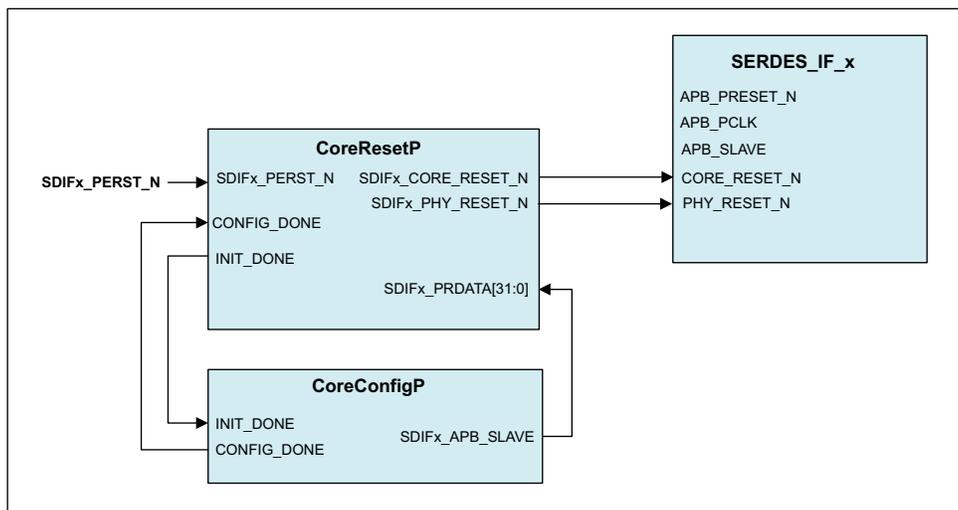


### 9.2.1.3 SerDes L2/P2, PRST#

L2 and P2 are Low power states for the Link and PHY interface in a PCI express (PCIe) system. A power management component in a PCIe system will control exit from the L2/P2 state. Part of the sequence when emerging from the Low power state involves assertion and release of the PCI Express Reset (PERST# or SDIFx\_PERST\_N in our implementation). CoreResetP monitors SDIFx\_PERST signals and L2/P2 states, and generates CORE reset and PHY reset to fulfill the Low power mode reset requirement.

The following figure shows the CoreResetP connectivity with SERDES\_IF block. If the System Builder is used to generate the Libero project, all required cores are instantiated, and connections are made automatically.

**Figure 127 • CoreResetP Connectivity with SERDES\_IF Block**



## 9.2.2 Implementation

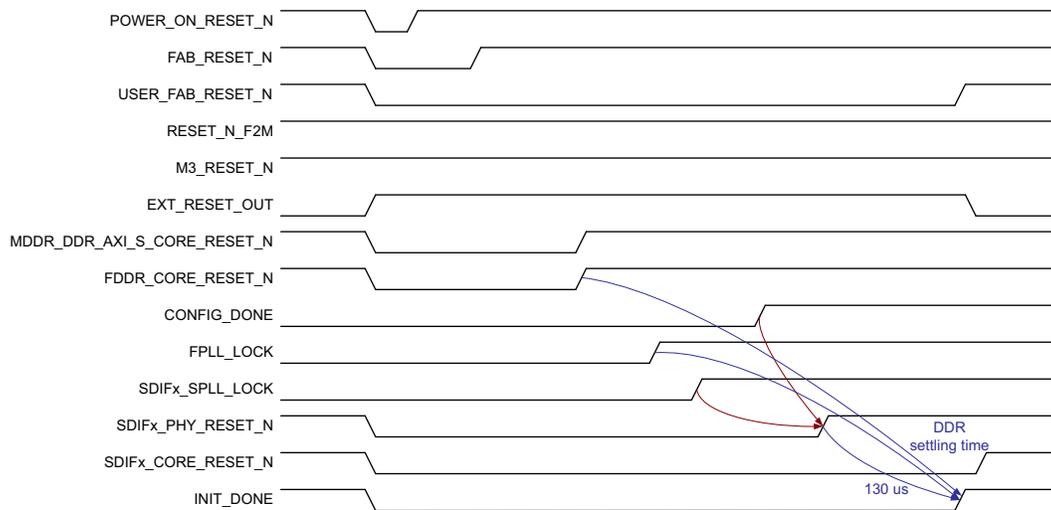
If the System Builder tool is used within the Libero SoC software to construct a design targeted at an IGLOO2 device, CoreResetP will automatically be instantiated and connected within the design if required. The user can manually instantiate and configure CoreResetP within a SmartDesign design if required. Refer to the **CoreResetP Handbook** for connecting and configuring CoreResetP in SmartDesign.

**Note:** CoreConfigP soft IP facilitates configuration of peripheral blocks (MDDR, FDDR, and SERDES\_IF blocks) in IGLOO2 devices. CoreConfigP is available in the Libero SoC IP Catalog. Refer to the **CoreConfigP Handbook** for port lists and their descriptions, design flows, memory maps, and Control and Status Register details.

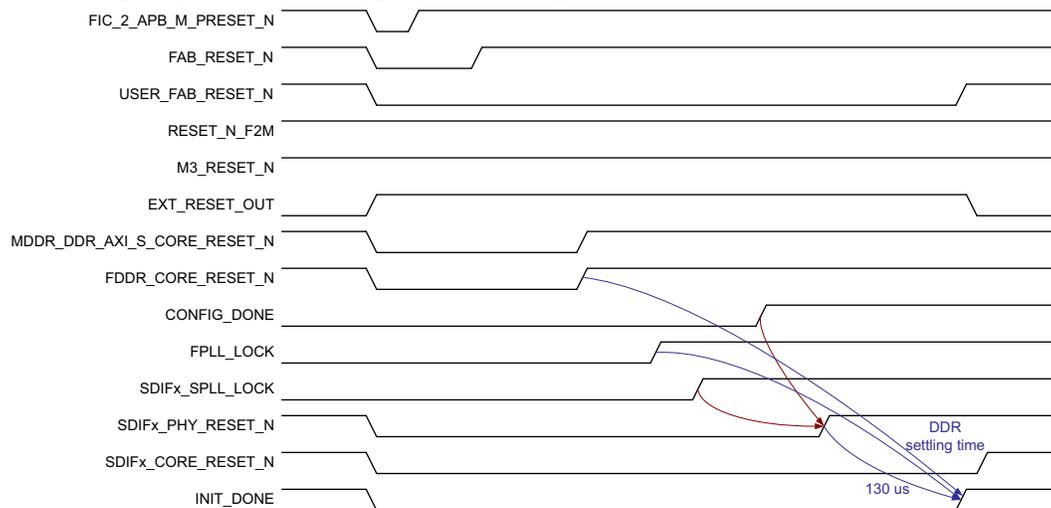
## 9.2.3 Timing Diagrams

The following figures show the timing of reset signals for reset sequences initiated by the assertion of POWER\_ON\_RESET\_N, FIC\_2\_APB\_M\_PRESET\_N, EXT\_RESET\_IN\_N, and USER\_FAB\_RESET\_IN\_N signals.

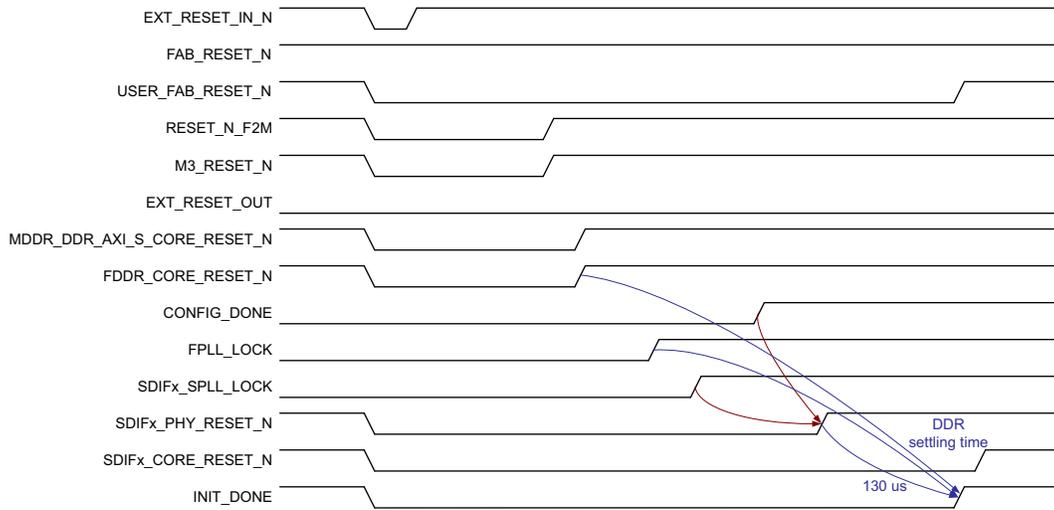
**Figure 128 • Timing Diagram for Reset Signals Initiated by the Assertion of POWER\_N\_RESET\_N**



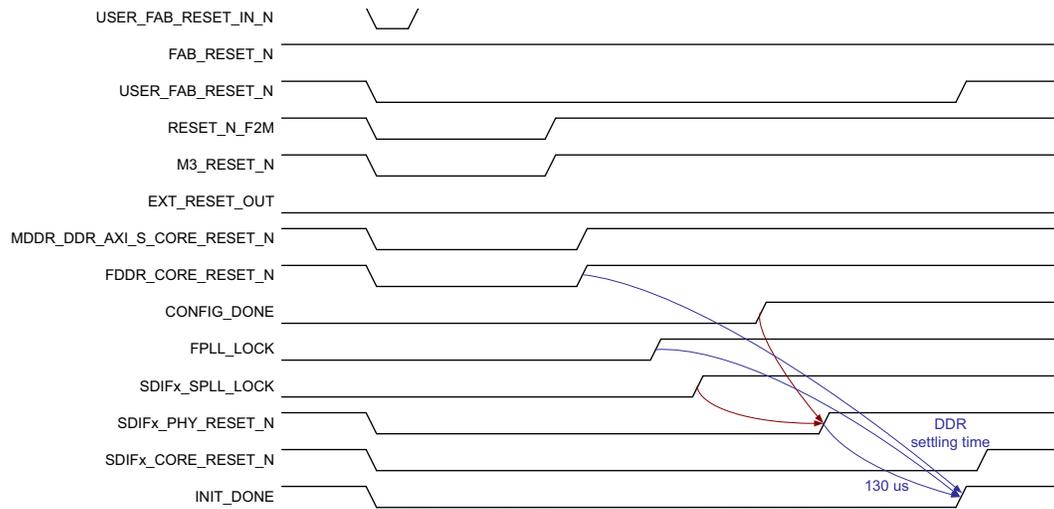
**Figure 129 • Timing Diagram for Reset Signals Initiated by the Assertion of FIC\_2\_APB\_M\_PRESET\_N**



**Figure 130 • Timing for Reset Signals Initiated by the Assertion of EXT\_RESET\_IN\_N**



**Figure 131 • Timing for Reset Signals Initiated by the Assertion of USER\_FAB\_RESET\_IN\_N**



## 9.3 SYSREG Control Registers

The following table lists the system control registers. Refer to [System Register Block](#), page 196 for a detailed description of each register and bit.

**Table 124 • Switch Register Map**

Register Name	Register Type	Flash Write Protect	Reset Source	Description
SOFT_RESET_CR	RW-P	Bit	SYSRESET_N	Generates the software control resets to the HPMS peripherals. For more information, see <a href="#">Table 143</a> , page 214.
MDDR_CR	RW-P	Register	PORESET_N	MDDR Configuration Register. For more information, see <a href="#">Table 145</a> , page 215.
HPMS_FACC1_CR	RW-P	Field	CC_RESET_N	HPMS DDR Bridge fabric alignment clock controller 1 Configuration Register. For more information, see <a href="#">Table 145</a> , page 215.

## 10 System Register Block

The System Register (SYSREG) block is an array of system-level registers that contain user configuration information used to configure the high performance memory subsystem (HPMS). The contents of these registers are initially set based on the information entered using the HPMS configurator in the Libero software. The power-up initialized state of these registers, as well as write protection bits are controlled by flash configuration bits. The configuration bits are set during device programming.

The SYSREG block is connected to the AHB bus matrix and can be accessed by all bus masters. Refer to [Figure 42](#), page 61. Write access to these registers provides the capability to modify the initialized SYSREG block register contents by the user application. There are seven types of System Registers as described in [Table 125](#), page 198 which provide different levels of read/write access by bus masters.

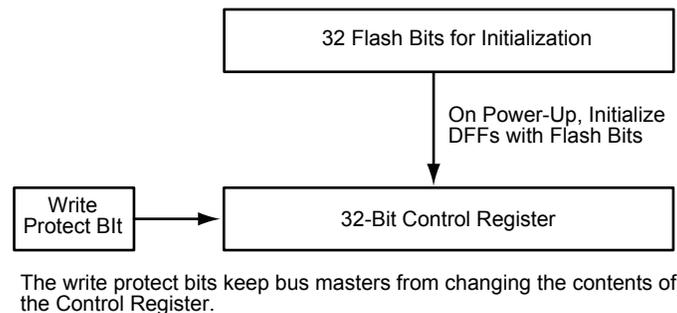
### 10.1 SYSREG Block Register Write Protection

Each SYSREG block register has dedicated write protect bits to control write access from bus masters. Write protect bits are flash configuration bits that are set based on user inputs to the HPMS configurator. These bits are defined during the device design phase and can only be modified by reprogramming the device. Users have the ability to set protection levels for the entire register, independent fields within each register, or individual bits within each register.

#### 10.1.1 Register Write Protect

One Register Write Protect bit is used to write protect entire register contents as shown in the following figure. On power-up, the register contents are initialized based on the flash configuration bits set from the HPMS configurator. If the Register Write Protect bit is set in the HPMS configurator, the initialized value of the entire register cannot be modified by the user application. If the Register Write Protect bit is not set, the contents of the register can be modified by any bus master. Register Write Protect bits can only be modified by reprogramming the FPGA and is therefore protected by the standard FPGA programming security features.

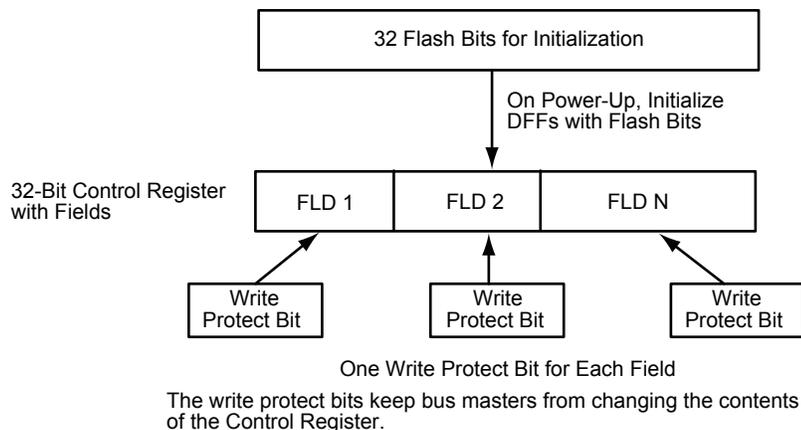
**Figure 132 • Register Write Protect**



## 10.1.2 Field Write Protect

Many System Registers contain fields of multiple bits. A Field Write Protect bit provides write protection for an entire field within a single register as shown in the following figure. Field Write Protect bits follow the same rules as Register Write Protect bits described in [Register Write Protect](#), page 196, but are instead applied to each individual field within the register. There is a Field Write Protect bit allocated for each field within the register.

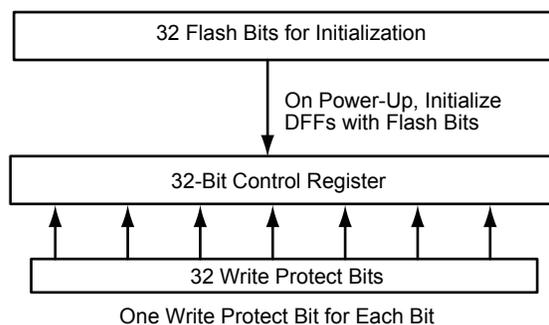
**Figure 133 • Field Write Protect**



## 10.1.3 Bit Write Protect

A Bit Write Protect bit provides write protection for each individual bit within a single register as shown in the following figure. Bit Write Protect bits follow the same rules as Register Write Protect bits described in [Register Write Protect](#), page 196, but are instead applied to each individual bit within the register. There is a Bit Write Protect bit allocated for each bit within the register.

**Figure 134 • Bit Write Protect**



The write protect bits keep bus masters from changing the contents of the Control Register.

Refer to [Register Lock Bits Configuration](#), page 200 for locking and unlocking registers.

## 10.2 Register Types

There are several register types in the SYSREG block as defined in the following table.

**Table 125 • Register Types**

Type	Function
RW-P	Supports read and write accesses via AHB bus matrix. Refer to <a href="#">Figure 135</a> , page 198. Register contents are initialized from flash configuration bits at power-up and the assertion of SYS_RESET_N. Typically used for HPMS Control Registers.
RW	Supports read and write accesses via AHB bus matrix. Refer to <a href="#">Figure 136</a> , page 199. Register contents are not initialized from flash configuration bits at power-up. The reset state is determined by the user HW design following assertion of SYS_RESET_N. Typically used for HPMS Control Registers.
RO	Supports read only accesses via AHB bus matrix. Refer to <a href="#">Figure 137</a> , page 199. Register contents are not initialized from flash configuration bits at power-up or the assertion of SYS_RESET_N. Typically used for HPMS Control Registers.
RO-P	Supports read only accesses via AHB bus matrix. Refer to <a href="#">Figure 138</a> , page 200. Register contents are initialized from flash configuration bits at power-up and the assertion of SYS_RESET_N. Typically used to return HPMS status information.
RO-U	Does not support read or write access via AHB bus matrix. Refer to <a href="#">Figure 139</a> , page 200. Register contents are initialized from flash configuration bits at power-up and the assertion of SYS_RESET_N. Typically used for HPMS Control Registers.
W1P	Write '1' to clear the register. This register is Write-Only.
SW1C	Individual register bits are set ('1') when related input is asserted. Bits are individually cleared when corresponding register bit is written high.

The following figures illustrate schematically a few of the register types of the SYSREG registers.

**Figure 135 • RW-P Type**

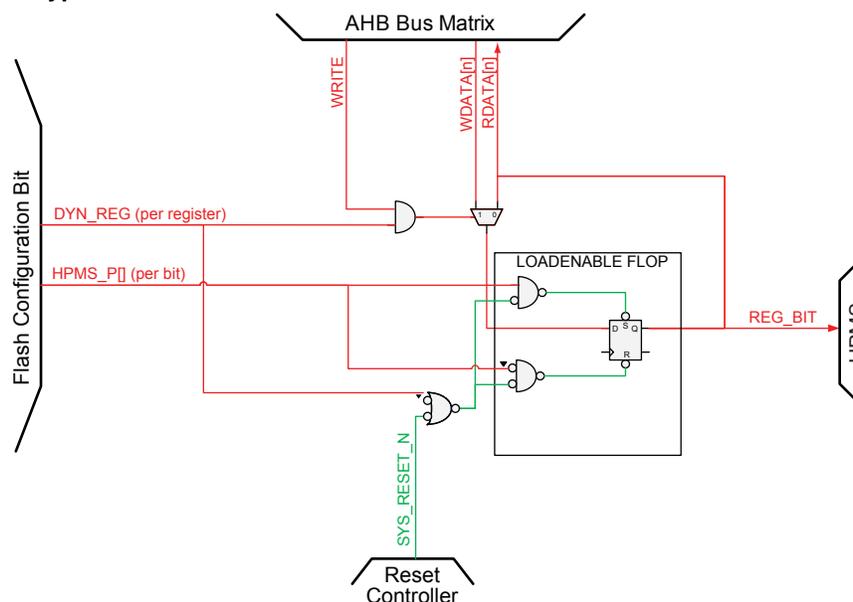


Figure 136 • RW Type

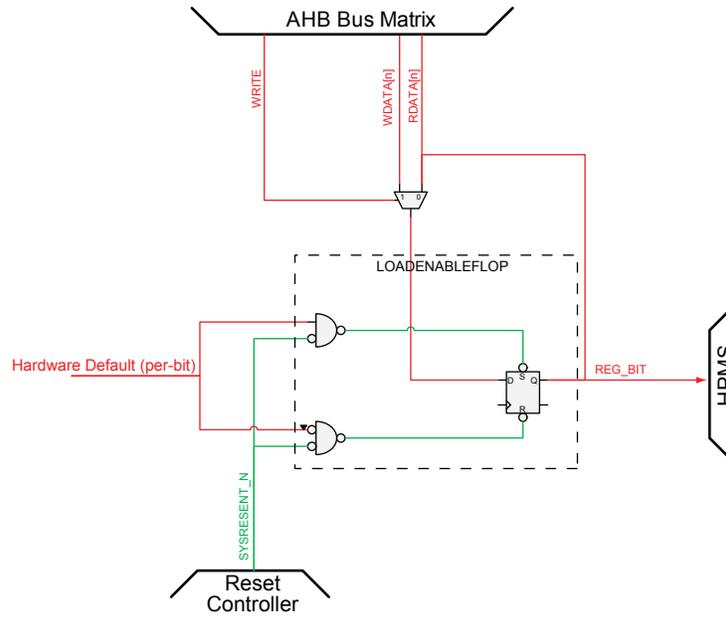


Figure 137 • RO Type

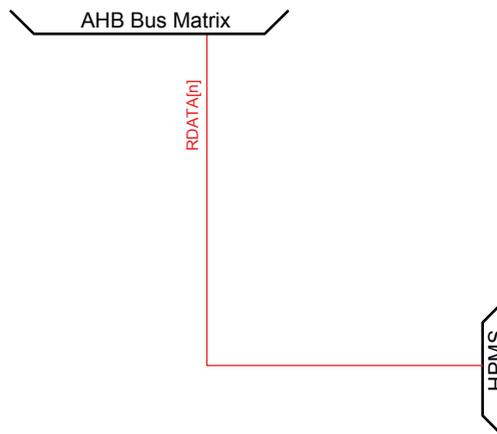


Figure 138 • RO-P Type

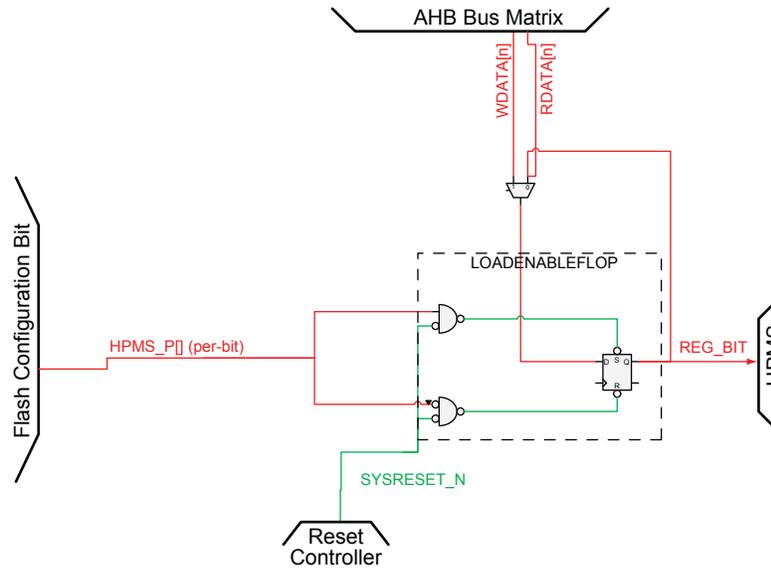
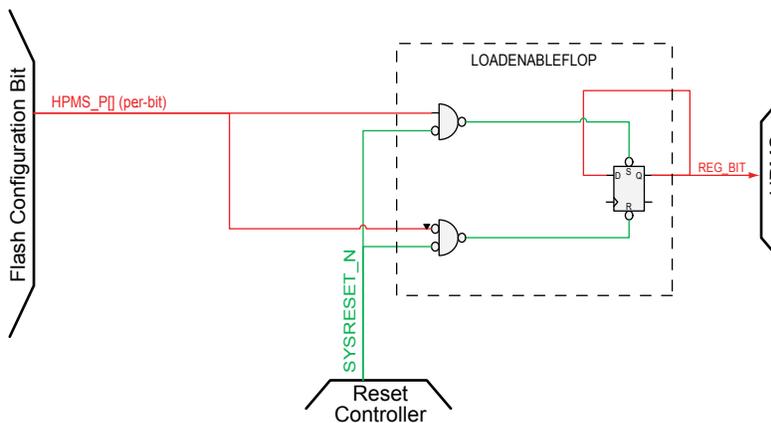


Figure 139 • RO-U Type

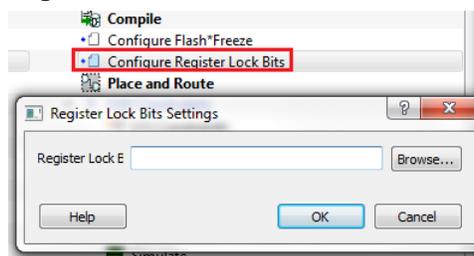


## 10.3 Register Lock Bits Configuration

The Register Lock Bits Configuration tool is used to lock HPMS, SerDes, and FDDR configuration registers of IGLOO2 devices in order 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 a IGLOO2 project.

From the **Design Flow** window, click **Configure Register Lock Bits** to open the configurator. Then, click **Browse...** to navigate to a text file (\*.txt) that contains the Register Lock Bit settings.

Figure 140 • Register Lock Bit Settings



### 10.3.1 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 in a different name to modify the text file to set lock bits.

### 10.3.2 Lock Bit File Syntax

A valid entry in the lock bit configuration file is defined as <lock\_parameters> <lock bit value> pair format.

The lock parameters are structured as follows:

- Lock bit syntax for a register: <Physical block name>\_<register name>\_LOCK
- Lock bit syntax for a specific field: <Physical block name>\_<register name>\_<field name>\_LOCK

The following are the physical block names (varies with the device family and die):

- HPMS
- FDDR
- SERDES\_IF\_x (where x is 0,1,2,3 to indicate the physical SerDes location) for IGLOO2 (M2GL010/M2GL025/M2GL050/M2GL150) devices
- SERDES\_IF2 for IGLOO2 (M2GL/M2GL090) 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.

**Figure 141 • 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_CTRL_SOFTRESET_LOCK 1
MSS_SOFTRESET_MDDR_FIC64_SOFTRESET_LOCK 1
MSS_M3_CONFIG_LOCK 1
```

### 10.3.3 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 to lock.
  - Set the lock bit value to 1 for the registers to unlock.
3. Save the file, and import the file into the project (**Design Flow window > Configure Register Lock Bits**), see [Figure 140](#), page 200.
4. Regenerate the bitstream.

## 10.4 Register Map

The following table lists all the registers in the SYSREG block. The SYSREG block is located at address 0x40038000.

**Table 126 • SYSREG**

Register Name	Addr. Offset	Register Type	Flash Write Protect	Reset Source	Description
Reserved	0x0				
ESRAM_MAX_LAT	0x4	RW-P	Register	SYSRESET_N	eSRAM0 and eSRAM1 maximum latency
Reserved	0x8				
ENVM_CR	0xC	RW-P	Register	SYSRESET_N	eNVM Configuration Register
Reserved	0x10				
ENVM_REMAP_FAB_CR	0x14	RW-P	Register	SYSRESET_N	eNVM remap Configuration Register
Reserved	0x18 to 0x24				
DDRB_BUF_TIMER_CR	0x28	RW-P	Register	SYSRESET_N	DDR write buffer timeout
DDRB_NB_ADDR_CR	0x2C	RW-P	Register	SYSRESET_N	DDR non-bufferable address region base address
DDRB_NB_SIZE_CR	0x30	RW-P	Register	SYSRESET_N	Size of non- bufferable address region
DDRB_CR	0x34	RW-P	Register	SYSRESET_N	HPMS DDR bridge Configuration Register
EDAC_CR	0x38	RW-P	Register	SYSRESET_N	EDAC Configuration Register for eSRAM0 and eSRAM1
MASTER_WEIGHT0_CR	0x3C	RW-P	Register	SYSRESET_N	Master Weight Configuration Register 0
MASTER_WEIGHT1_CR	0x40	RW-P	Register	SYSRESET_N	Master Weight Configuration Register 1
SOFT_IRQ_CR	0x44	RW-P	Register	SYSRESET_N	Enables software interrupt
SOFT_RESET_CR	0x48	RW-P	Bit	SYSRESET_N	Generates software control interrupts to the HPMS peripherals
Reserved	0x4C				

**Table 126 • SYSREG (continued)**

Register Name	Addr. Offset	Register Type	Flash Write Protect	Reset Source	Description
FAB_IF_CR	0x50	RW-P	Register	SYSRESET_N	Controls fabric interface
Reserved	0x54 to 0x5C				
MDDR_CR	0x60	RW-P	Register	PORESET_N	MDDR Configuration Register
Reserved	0x64				
PERIPH_CLK_MUX_SEL_CR	0x68	RW-P	Register	PORESET_N	Peripheral Clock MUX Select Control Register
Reserved	0x6C				
MDDR_IO_CALIB_CR	0x70	RW-P	Register	PORESET_N	MDDR I/O Calibration Control Register
Reserved	0x74				
EDAC_IRQ_ENABLE_CR	0x78	RW-P	Register	SYSRESET_N	Enables/disables 1-bit error, 2-bit error status for eSRAM0 and eSRAM1
Reserved	0x7C				
ESRAM_PIPELINE_CR	0x80	RW-P	Register	SYSRESET_N	Controls the pipeline present in the memory read path of eSRAM memory
Reserved	0x84 to 0x8C				
HPMS_PLL_STATUS_LOW_CR	0x90	RW-P	Register	CC_RESET_N	Controls the configuration input of MPLL register
HPMS_PLL_STATUS_HIGH_CR	0x94	RW-P	Register	CC_RESET_N	Controls the configuration input of the MPLL register
HPMS_FACC1_CR	0x98	RW-P	Field	CC_RESET_N	HPMS DDR bridge FACC1 Configuration Register
HPMS_FACC2_CR	0x9C	RW-P	Field	CC_RESET_N	HPMS DDR bridge FACC2 Configuration Register
HPMS_CLK_CALIB_CR	0xA4	RW-P	Register	SYSRESET_N	Starts FPGA fabric calibration test circuit
PLL_DELAY_LINE_SEL_CR	0xA8	RW-P	Register	SYSRESET_N	PLL Delay Line Select Control Register
Reserved	0xAC				
RESET_SOURCE_CR	0xB0	RW			Reset Source Control Register. The reset values are mentioned in the bit definitions
Reserved	0xB4 to 0xDC			SYSRESET_N	

**Table 126 • SYSREG (continued)**

Register Name	Addr. Offset	Register Type	Flash Write Protect	Reset Source	Description
DDRB_HPD_ERR_ADR_SR	0xE0	RO		SYSRESET_N	HPMS DDR Bridge High Performance DMA Master Error Address Status Register
DDRB_SW_ERR_ADR_SR	0xE4	RO		SYSRESET_N	HPMS DDR Bridge AHB Bus Error Address Status Register
DDRB_BUF_EMPTY_SR	0xE8	RO		SYSRESET_N	HPMS DDR Bridge Buffer Empty Status Register
DDRB_DSBL_DN_SR	0xEC	RO		SYSRESET_N	HPMS DDR Bridge Disable Buffer Status Register
ESRAM0_EDAC_CNT	0xF0	RO		SYSRESET_N	1-bit error and 2-bit error count of eSRAM0
ESRAM1_EDAC_CNT	0xF4	RO		SYSRESET_N	1-bit error and 2-bit error count of eSRAM1
Reserved	0xF8 to 0x108				
ESRAM0_EDAC_ADR	0x10C	RO		SYSRESET_N	Address from eSRAM0 on which 1-bit and 2-bit SECCED error has occurred
ESRAM1_EDAC_ADR	0x110	RO		SYSRESET_N	Address from eSRAM1 on which 1-bit and 2-bit SECCED error has occurred
Reserved	0x114 to 0x124				
MM4_5_DDR_FIC_SECURITY/ MM4_5_FIC64_SECURITY	0x128	RO-U		SYSRESET_N	Read and write security for masters 4, 5, and DDR_FIC to eSRAM0, eSRAM1, eNVM1, eNVM0, and HPMS DDR bridge
MM3_7_SECURITY	0x12C	RO-U		SYSRESET_N	Read and write security for masters 3 and 7 to eSRAM0, eSRAM1, eNVM1, eNVM0, and HPMS DDR bridge
MM9_SECURITY	0x130	RO-U		SYSRESET_N	Read and write security for master 9 to eSRAM0, eSRAM1, eNVM1, eNVM0, and HPMS DDR bridge
Reserved	0x134 to 0x13C				
DEVICE_SR	0x140	RO		SYSRESET_N	Device Status Register
ENVM_PROTECT_USER	0x144	RO-U		SYSRESET_N	Configuration for accessibility of protect regions of eNVM0 and eNVM1 by different masters on the AHB bus matrix, updated by user flash bits

**Table 126 • SYSREG (continued)**

Register Name	Addr. Offset	Register Type	Flash Write Protect	Reset Source	Description
ENVM_STATUS	0x148	RO-U		PORESET_N	Code shadow Status Register
DEVICE_VERSION	0x14C	RO			Configures device version
HPMS_PLL_STATUS	0x150	RO			HPMS DDR PLL Status Register
Reserved	0x154				
ENVM_SR	0x158	RO		SYSRESET_N	Busy status eNVM0 and eNVM1
Reserved	0x15C				
DDR_B_STATUS	0x160	RO		SYSRESET_N	HPMS DDR bridges status
MDDR_IO_CALIB_STATUS	0x164	RO		PORESET_N	DDR I/O Calibration Status Register
HPMS_CLK_CALIB_STATUS	0x168	RO		SYSRESET_N	HPMS DDR Clock Calibration Status Register
Reserved	0x16C to 0x180				
FAB_PROT_SIZE	0x184	RO-P		SYSRESET_N	Size of memory protected from fabric master
FAB_PROT_BASE	0x188	RO-P		SYSRESET_N	Base address which is protected from fabric master
Reserved	0x18C				
EDAC_SR	0x190	SW1C		SYSRESET_N	Status of 1-bit SECEDED error detection and correction, 2-bit SECEDED error detection for eSRAM0 and eSRAM1
HPMS_INTERNAL_SR	0x194	SW1C		SYSRESET_N	HPMS Internal Status Register
HPMS_EXTERNAL_SR	0x198	SW1C		SYSRESET_N	HPMS External Status Register
Reserved	0x19C to 0x1A0				
CLR_EDAC_COUNTERS	0x1A4	W1P		SYSRESET_N	Clears 16-bit counter value in eSRAM0 and eSRAM1, corresponding to count value of EDAC 1-bit and 2-bit errors
FLUSH_CR	0x1A8	W1P		SYSRESET_N	Flush Control Register
Reserved	0x1AC to 0x290				

For detailed information about each of the registers, see [Table 128](#), page 206 through [Table 186](#), page 238.

## 10.5 System Registers Behavior for M2GL005/010 devices

Application traffic across the FIC\_0 interface can cause certain bits in the SYSREG block to change state, if these bits are dynamically modified from their default values during runtime. This impacts all IGLOO2 005 and 010 devices.

The following table lists the subset of system registers and specific bit definitions that are affected. The registers/bits listed in the following table must be configured once, on power-up. Dynamically altering the contents of these registers can result in their values to be reset to the power on reset state.

**Table 127 • Subset of System Registers**

System Register	Fields
SOFT_RESET_CR	All bits
FAB_IF_CR	FAB0_AHB_BYPASS, FAB1_AHB_BYPASS, FAB0_AHB_MODE, FAB1_AHB_MODE, SW_FIC_REG_SEL
MDDR_CR	MDDR_CONFIG_LOCAL, F_AXI_AHB_MODE, PHY_SELF_REF_EN
PERIPH_CLK_MUX_SEL_CR	SPI_SCK_FAB_SEL
EDAC_IRQ_ENABLE_CR	All bits
HPMS_PLL_STATUS_LOW_CR	FACC_PLL_DIVR, FACC_PLL_DIVF, FACC_PLL_DIVQ, FACC_PLL_RANGE, FACC_PLL_LOCKWIN, FACC_PLL_LOCKCNT
HPMS_PLL_STATUS_HIGH_CR	FACC_PLL_BYPASS, FACC_PLL_MODE_1V2, FACC_PLL_MODE_3V3, FACC_PLL_FSE, FACC_PLL_PD, FACC_PLL_SSE, FACC_PLL_SSMD, FACC_PLL_SSMF
HPMS_FACC1_CR	DIVISOR_A, APB0_DIVISOR, APB1_DIVISOR, DDR_CLK_EN, HPMS_CLK_DIVISOR, FACC_GLMUX_SEL, FIC_0_DIVISOR, FIC_1_DIVISOR

## 10.6 Register Details

### 10.6.1 eSRAM Latency Configuration Register

**Table 128 • ESRAM\_MAX\_LAT**

Bit Number	Name	Reset Value	Description
[31:6]	Reserved	0	
[5:3]	SW_MAX_LAT_ESRAM1	0x1	Defines the maximum number of cycles the fixed priority master waits for eSRAM1 when it is being accessed by a master with a WRR priority scheme.
[2:0]	SW_MAX_LAT_ESRAM0	0x1	Defines the maximum number of cycles the fixed priority master waits for eSRAM0 when it is being accessed by a master with a WRR priority scheme. It is configurable from 1 to 8 (8 by default).

The following table gives eSRAM maximum latency values, where x is either 0 or 1.

**Table 129 • eSRAM Maximum Latency Values**

SW_MAX_LAT_ESRAM<X>	Latency
0	8 (default)
1	1
2	2
3	3
4	4
5	5
6	6
7	7

## 10.6.2 eNVM Configuration Register

**Table 130 • ENVM\_CR**

Bit Number	Name	Reset Value	Description
[31:17]	Reserved	0	
16	ENVM_SENSE_ON	0	Turns on or off the sense amps for both NVM0 and NVM1
15	ENVM_PERSIST	0	Reset control for NVM0 and NVM1. 0: Reset on SYSRESET_N and PORESET_N 1: Reset on PORESET_N
14	NV_DPD1	0	Deep power-down control for the NVM1. 0: Normal operation 1: NVM deep power-down
13	NV_DPD0	0	Deep power-down control for the NVM0. 0: Normal operation 1: NVM deep power-down

**Table 130 • ENVM\_CR**

[12:5]	NV_FREQRNG	0x7	<p>Setting of NV_FREQRNG[8:5] or NV_FREQRNG[12:9] determines the behavior of eNVM BUSY_B with respect to the AHB Bus interface clock. It can be used to accommodate various frequencies of the external interface clock, HPMS_CLK, or it can be used to advance or delay the data capture due to variation of read access time of the NVM core. It sets the number of wait states to match with the Fabric master operating frequency for read operations. The small counter in the NVM Controller uses this value to advance or delay the data capture before sampling data.</p> <p>0000: NOT SUPPORTED  0001: NOT SUPPORTED  0010: Page Read = 3, All other modes (Page program and Page verify) = 2  0011: Page Read = 4, All other modes (Page program and Page verify) = 2  0100: Page Read = 5, All other modes (Page program and Page verify) = 2  0101: Page Read = 6, All other modes (Page program and Page verify) = 3  0110: Page Read = 7, All other modes (Page program and Page verify) = 3  0111: Page Read = 8, All other modes (Page program and Page verify) = 4  1000: Page Read = 9, All other modes (Page program and Page verify) = 4  1001: Page Read = 10, All other modes (Page program and Page verify) = 4  1010: Page Read = 11, All other modes (Page program and Page verify) = 5  1011: Page Read = 12, All other modes (Page program and Page verify) = 5  1100: Page Read = 13, All other modes (Page program and Page verify) = 6  1101: Page Read = 14, All other modes (Page program and Page verify) = 6  1110: Page Read = 15, All other modes (Page program and Page verify) = 6  1111: Page Read = 16, All other modes (Page program and Page verify) = 7</p> <p>NV_FREQRNG[8:5] is used for NVM0 and NV_FREQRNG[12:9] is used for NVM1.</p>
4:0	SW_ENVMREMAPSIZE	0x11	<p>Size of the segment in eNVM, which is to be remapped to location 0x00000000. This logically splits eNVM into a number of segments. The region sizes are shown in <a href="#">Table 131 on page 209</a>.</p>

### 10.6.2.1 SW\_ENVMREMAPSIZE Bit Combinations

Table 131 • SW\_ENVMREMAPSIZE

Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Remap Size
0	0	0	0	0	Reserved
0	0	0	0	1	Reserved
0	0	0	1	0	Reserved
0	0	0	1	1	Reserved
0	0	1	0	0	Reserved
0	0	1	0	1	Reserved
0	0	1	1	0	Reserved
0	0	1	1	1	Reserved
0	1	0	0	0	Reserved
0	1	0	0	1	Reserved
0	1	0	1	0	Reserved
0	1	0	1	1	Reserved
0	1	1	0	0	Reserved
0	1	1	0	1	16 Kbytes
0	1	1	1	0	32 Kbytes
0	1	1	1	1	64 Kbytes
1	0	0	0	0	128 Kbytes
1	0	0	0	1	256 Kbytes
1	0	0	1	0	512 Kbytes, reset value

### 10.6.3 eNVM FPGA Fabric Remap Base Address Register

Table 132 • ENVM\_REMAP\_FAB\_CR

Bit Number	Name	Reset Value	Description
[31:19]	Reserved	0	
[18:1]	SW_ENVMFABREMAPBASE	0	Offset within eNVM address space of the base address of the segment in eNVM, which is to be remapped to location 0x00000000 for use by a soft processor in the FPGA fabric.
0	SW_ENVMFABREMAPENABLE	0	0: eNVM fabric remap not enabled for accesses by fabric master. The portion of eNVM visible in the eNVM window at location 0x00000000 of a soft processor's memory space corresponds to the memory locations at the bottom of eNVM. 1: eNVM fabric remap enabled. The portion of eNVM visible at location 0x00000000 of a soft processor's memory space of is a remapped segment of eNVM.

Bits [18:N] of this bus indicate the base address of the remapped segment. The value of N depends on the eNVM remap section size, so that the base address is aligned according to an even multiple of segment size. The power of 2 size specified by SW\_ENVMREMAPSIZE[4:0] (Table 131 on page 209) defines how many bits of base address are used. For example, if the SW\_ENVMREMAPSIZE[4:0] is 01111, this corresponds to a segment size of 64KB. 64KB is 2 to the power of 16. Therefore, the value of N in this case, is 16. So the base address of the region, in this case, is specified by SW\_ENVMREMAPSIZE[18:16].

This register should only be written by using 32-bit accesses. The behavior of the system is undefined if other size accesses are used.

## 10.6.4 HPMS DDR Bridge Buffer Timer Control Register

Table 133 • DDRB\_BUF\_TIMER\_CR

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0	
[9:0]	DDR_BUF_TIMER	0x3FF	10-bit timer interface used to configure the timeout register in the write buffer module. Once timer reaches the timeout value, a flush request is generated by the flush controller and if response has been received for previous write request from write arbiter, this request is posted to the write arbiter. This register is common for all buffers. The value in this register will be in terms of number of HPMS_CLK clocks.

## 10.6.5 HPMS DDR Bridge Non-Bufferable Address Control Register

Table 134 • DDRB\_NB\_ADDR\_CR

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0	
[15:0]	DDR_B_NB_ADDR	0xA000	Base address of a non-bufferable address region. Bits [15:(N – 1)] of this signal are compared with AHB address [31:(N + 15)] to check whether address is in non-bufferable region. The value of N depends on the non-bufferable region size, so the base address is defined according to DDRB_NB_SZ.

## 10.6.6 HPMS DDR Bridge Non-Bufferable Size Control Register

Table 135 • DDRB\_NB\_SIZE\_CR

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0	
[3:0]	DDR_B_NB_SZ	0x1	Size of non-bufferable address region [(2N – 1x64) KB]. The region sizes are as shown in Table 136, page 211.

The following table gives the size of the non-bufferable region, given by the formula  $[(2N - 1) \times 64]$  KB.

**Table 136 • Non-Bufferable Region**

Bit 0	Bit 1	Bit 2	Bit 3	N	Non-Bufferable Region
0	0	0	0	0	No non-bufferable region
0	0	0	1	1	64 KB = $(2^1 - 1 \times 64)$ KB
0	0	1	0	2	128 KB = $(2^2 - 1 \times 64)$ KB
0	0	1	1	3	256 KB = $(2^3 - 1 \times 64)$ KB
0	1	0	0	4	512 KB = $(2^4 - 1 \times 64)$ KB
0	1	0	1	5	1 MB = $(2^5 - 1 \times 64)$ KB
0	1	1	0	6	2 MB = $(2^6 - 1 \times 64)$ KB
0	1	1	1	7	4 MB = $(2^7 - 1 \times 64)$ KB
1	0	0	0	8	8 MB = $(2^8 - 1 \times 64)$ KB
1	0	0	1	9	16 MB = $(2^9 - 1 \times 64)$ KB
1	0	1	0	10	32 MB = $(2^{10} - 1 \times 64)$ KB
1	0	1	1	11	64 MB = $(2^{11} - 1 \times 64)$ KB
1	1	0	0	12	128 MB = $(2^{12} - 1 \times 64)$ KB
1	1	0	1	13	256 MB = $(2^{13} - 1 \times 64)$ KB
1	1	1	0	14	512 MB = $(2^{14} - 1 \times 64)$ KB
1	1	1	1	15	1 GB = $(2^{15} - 1 \times 64)$ KB: Entire region is non-bufferable

## 10.6.7 HPMS DDR Bridge Configuration Register

**Table 137 • DDRB\_CR**

Bit Number	Name	Reset Value	Description
[31:20]	Reserved	0	
[19:16]	DDR_SW_MAP	0	Sets the AHB bus master to DDR address space mapping mode 0–15.
[15:12]	DDR_HPDA_MAP	0	Sets the HPDA master to DDR address space mapping mode 0–15.
[11:8]	Reserved	0	
7	DDR_B_BUF_SZ	0x1	Configures the write buffer and read buffer size as per DDR burst size. This port is common for all buffers. 0: Buffer size is configured to 16 bytes 1: Buffer size is configured to 32 bytes
6	Reserved	0	
5	DDR_B_SW_REN	0x1	Allows the read buffer for AHB BUS master in HPMS DDR bridge to be disabled. Allowed values: 0: Disabled 1: Enabled

**Table 137 • DDRB\_CR (continued)**

4	DDR_B_SW_WEN	0x1	Allows the write combining buffer for AHB bus master in HPMS DDR bridge to be disabled. Allowed values: 0: Disabled 1: Enabled
3	DDR_B_HPD_REN	0x1	Allows the read buffer for high performance DMA master in HPMS DDR bridge to be disabled. Allowed values: 0: Disabled 1: Enabled
2	DDR_B_HPD_WEN	0x1	Allows the write combining buffer for high performance DMA master in HPMS DDR bridge to be disabled. Allowed values: 0: Disabled 1: Enabled
[1:0]	Reserved	0	

## 10.6.8 EDAC Configuration Register

**Table 138 • EDAC\_CR**

Bit Number	Name	Reset Value	Description
[31:2]	Reserved	0	
1	ESRAM1_EDAC_EN	0	Allows the EDAC for eSRAM1 to be disabled. Allowed values: 0: Disabled 1: Enabled
0	ESRAM0_EDAC_EN	0	Allows the EDAC for eSRAM0 to be disabled. Allowed values: 0: Disabled 1: Enabled

## 10.6.9 Master Weight Configuration Register 0

**Table 139 • MASTER\_WEIGHT0\_CR**

Bit Number	Name	Reset Value	Description
[31:30]	Reserved	0	
[29:25]	SW_WEIGHT_PDMA	0	Configures the round robin weight for peripheral DMA master. It is configurable from 1 to 32 (32 by default).
[24:20]	SW_WEIGHT_FAB_1	0	Configures the round robin weight for fabric (FIC_1) master. It is configurable from 1 to 32 (32 by default).
[19:15]	SW_WEIGHT_FAB_0	0	Configures the round robin weight for fabric (FIC_0) master. It is configurable from 1 to 32 (32 by default).
[14:0]	Reserved	0	

The weight values are as given in [Table 141](#), page 213.

## 10.6.10 Master Weight Configuration Register 1

**Table 140 • MASTER\_WEIGHT1\_CR**

Bit Number	Name	Reset Value	Description
[31:15]	Reserved	0	
[14:10]	SW_WEIGHT_G	0	Configures the round robin weight for G master. It is configurable from 1 to 32 (32 by default). The weight values are as given in <a href="#">Table 141 on page 213</a> .
[9:5]	Reserved	0	
[4:0]	SW_WEIGHT_HPDM	0	Configures the round robin weight for HPDMA master. It is configurable from 1 to 32 (32 by default). The weight values are as given in <a href="#">Table 141 on page 213</a> .

The following table lists weight values, where <master> is FIC\_0, FIC\_1, PDMA, HPDMA, or G.

**Table 141 • Programmable Weight Values**

SW_WEIGHT_<master>	Weight
0	32
1	1
2	2
3	3
.	.
.	.
.	.
28	28
29	29
30	30
31	31

## 10.6.11 Software Interrupt Register

**Table 142 • SOFT\_IRQ\_CR**

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0	
0	SOFTINTER RUPT	0	1: FIIC SOFTINTERRUPT is asserted 0: SOFTINTERRUPT signal is cleared

## 10.6.12 Software Reset Control Register

**Table 143 • SOFT\_RESET\_CR**

Bit Number	Name	Reset Value	Description
[31:27]	Reserved	0	
26	MDDR_DDRFIC_SOFTRESET	0x1	0: Releases DDR_FIC controller from reset 1: Keeps DDR_FIC controller in reset.
25	MDDR_CTLR_SOFTRESET	0x1	0: Releases MDDR controller from reset 1: Keeps MDDR controller in reset.
[24:20]	Reserved	0	
19	FIC_1_SOFTRESET	0x1	0: Releases FIC_1 from reset 1: Keeps FIC_1 in reset
18	FIC_0_SOFTRESET	0x1	0: Releases FIC_0 from reset 1: Keeps FIC_0 in reset
17	HPDMA_SOFTRESET	0x1	0: Releases HPDMA from reset 1: Keeps HPDMA n reset
16	FPGA_SOFTRESET	0x1	0: Releases FPGA from reset 1: Keeps FPGA in reset
15	COMBLK_SOFTRESET	0	0: Releases COMM_BLK from reset 1: Keeps COMMUNICATION BLOCK (COMM_BLK) in reset
[14:10]	Reserved	0	
9	SPI_SOFTRESET	0x1	0: Releases SPI from reset 1: Keeps SPI in reset
[8:6]	Reserved	0	
5	PDMA_SOFTRESET	0x1	0: Releases the PDMA from reset 1: Keeps the PDMA in reset
4	Reserved	0	
3	ESRAM1_SOFTRESET	0	0: Releases the eSRAM_1 memory controller from reset 1: Keeps the eSRAM_1 memory controller in reset
2	ESRAM0_SOFTRESET	0	0: Releases the eSRAM_0 memory controller from reset. 1: Keeps the eSRAM_0 memory controller in reset.
1	ENVM1_SOFTRESET	0	0: Releases the eNVM_1 memory controller from reset 1: Keeps the eNVM_1 memory controller in reset
0	ENVM0_SOFTRESET	0	0: Releases the eNVM_0 memory controller from reset 1: Keeps the eNVM_0 memory controller in reset

Reset values mentioned in the preceding table are the default values of the bits, when peripherals are not configured using the software. If the peripheral is enabled using the software, then the default reset value for that bit is 0x0.

**Note:** The register fields for 005 and 010 devices must not be dynamically altered. See [System Registers Behavior for M2GL005/010 devices](#), page 206

## 10.6.13 Fabric Interface Control (FIC) Register

Table 144 • FAB\_IF\_CR

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0	
[9:4]	SW_FIC_REG_SEL	0x38	Indicates whether a specific fabric region is accessible by FIC_0 or FIC_1. This register should not be changed during operation. 0: Fabric region associated with FIC_0 1: Fabric region associated with FIC_1 By default, fabric region 0, 1, 2 are accessible through FIC_0 and regions 3, 4, 5 are accessible through FIC_1. These bits are driven into the AHB bus in order to allocate a specific memory region to either FIC_0 or FIC_1.
3	FAB1_AHB_MODE	0	Controls whether the FIC_1 fabric interface supports AHB mode or APB Mode. Allowed values: 0: Supports APB mode 1: Supports AHB mode
2	FAB0_AHB_MODE	0	Controls whether FIC_0 fabric interface supports AHB mode or APB mode. Allowed values: 0: Supports APB mode 1: Supports AHB mode
1	FAB1_AHB_BYPASS	0	0: FIC_1 is configured for synchronous bridging 1: FIC_1 is configured in bypass mode, if clock ratio is 1:1 and if in AHB mode
0	FAB0_AHB_BYPASS	0	0: FIC_0 is configured for synchronous bridging 1: FIC_0 is configured in bypass mode, if clock ratio is 1:1 and if in AHB mode

**Note:** The register fields for 005 and 010 devices must not be dynamically altered. See [System Registers Behavior for M2GL005/010 devices](#)

## 10.6.14 MDDR Configuration Register

Table 145 • MDDR\_CR

Bit Number	Name	Reset Value	Description
[31:4]	Reserved	0	
3	PHY_SELF_REF_EN	0	Indicates that the DRAM has been put into self-refresh. This is used for automatic locking of the codes during intermediate runs for DDR_C. Not used in non-DDRIO modes.
2	F_AXI_AHB_MODE	0	Used by the DDR_FIC and DDR CTL to select the AXI/AHB interface in the fabric. Allowed values: 0: AHB interface is selected. 1: AXI interface is selected.

**Table 145 • MDDR\_CR (continued)**

1	Reserved	0	
0	MDDR_CONFIG_LOCAL	0x1	Configures whether the HPMS AHBTOAPB2 bridge can directly access the APB slave within the MDDR subsystem or whether the APB slave is connected to the fabric. Allowed values: 0: AHBTOAPB2 bridge cannot access MDDR APB slave 1: AHBTOAPB2 bridge can access MDDR APB slave Reset signal for this bit is CC_RESET_N.

**Note:** The register fields for 005 and 010 devices must not be dynamically altered. See [System Registers Behavior for M2GL005/010 devices](#), page 206

## 10.6.15 Peripheral Clock MUX Select Control Register

**Table 146 • PERIPH\_CLK\_MUX\_SEL\_CR**

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0	
0	SPI_SCK_FAB_SEL	0	Selects the SPI_SCK from the fabric or I/O pads. Allowed values: 0: SPI_SCK clock from I/O pads is selected and fed to SPI. 1: SPI_SCK clock from fabric is selected and fed to SPI.

**Note:** The register fields for 005 and 010 devices must not be dynamically altered. See [System Registers Behavior for M2GL005/010 devices](#), page 206.

## 10.6.16 MDDR I/O Calibration Control Register

**Table 147 • MDDR\_IO\_CALIB\_CR**

Bit Number	Name	Reset Value	Description
[31:15]	Reserved	0	
14	CALIB_LOCK	0	Used in the DDRIO calibration block as an override to lock the codes during intermediate runs. When the fabric logic 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. This bit is only read/write.
13	CALIB_START	0	Used in the DDRIO calibration block and indicates that rerun of the calibration state machine is required.
12	CALIB_TRIM		Used in the DDRIO calibration block and indicates the override of the calibration value from the PC code/programmed code values.
[11:6]	NCODE	0	Used in the DDRIO calibration block and indicates DPC override NCODE from flash. This can also be overwritten from the fabric logic.
[5:0]	PCODE	0	Used in the DDRIO calibration block and indicates PC override PODE from flash. This can also be overwritten from the fabric logic.

## 10.6.17 EDAC Interrupt Enable Control Register

**Table 148 • EDAC\_IRQ\_ENABLE\_CR**

Bit Number	Name	Reset Value	Description
[31:15]	Reserved	0	
14	MDDR_ECC_INT_EN	0	Allows the error EDAC for MDDR status update to be disabled. Allowed values: 0: Disabled 1: Enabled
[13:6]	Reserved	0	
5	Reserved	0	
4	Reserved	0	
3	ESRAM1_EDAC_2E_EN	0	Allows the 2-bit error EDAC for eSRAM1 status update to be disabled. Allowed values: 0: Disabled 1: Enabled
2	ESRAM1_EDAC_1E_EN	0	Allows the 1-bit error EDAC for eSRAM1 status update to be disabled. Allowed values: 0: Disabled 1: Enabled
1	ESRAM0_EDAC_2E_EN	0	Allows the 2-bit error EDAC for eSRAM0 status update to be disabled. Allowed values: 0: Disabled 1: Enabled
0	ESRAM0_EDAC_1E_EN	0	Allows the 1-bit error EDAC for eSRAM0 status update to be disabled. Allowed values: 0: Disabled 1: Enabled

**Note:** The register fields for 005 and 010 devices must not be dynamically altered. See [System Registers Behavior for M2GL005/010 devices](#), page 206.

## 10.6.18 eSRAM PIPELINE Configuration Register

**Table 149 • ESRAM\_PIPELINE\_CR**

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0	
0	ESRAM_PIPELINE_ENABLE	0x1	Controls the pipeline in the read path of eSRAM memory. Allowed values: 0: Pipeline is bypassed 1: Pipeline is in the memory read path

## 10.6.19 HPMS DDR PLL Status Low Configuration Register

This register is to be configured by flash bits only and the user should not write to it while the source clock is active.

**Table 150 • HPMS\_PLL\_STATUS\_LOW\_CR**

Bit Number	Name	Reset Value	Description
[31:30]	Reserved	0	
[29:26]	FACC_PLL_LOCKCNT	0	Configures the MPLL LOCK counter value given by ( $2^{\wedge}$ binary value + 5). For example, if the binary value is 0000, the LOCK counter value is 32, and if binary value is 1111, then its value is 1,048,576.
[25:23]	FACC_PLL_LOCKWIN	0	Configures the MPLL phase error window for LOCK assertion as a fraction of the divided reference period. Values are at typical PVT only and are not PVT compensated. 000: 500 ppm 100: 8000 ppm 001: 1000 ppm 101: 16000 ppm 010: 2000 ppm 110: 32000 ppm 011: 4000 ppm 111: 64000 ppm
[22:19]	FACC_PLL_RANGE	0	Configures the MPLL filter range. The bit definitions are in <a href="#">Table 151</a> , page 219.
[19:16]	FACC_PLL_DIVQ	0x2	Configures MPLL output divider value in order to generate the DDR clock. Output divider values are given by: 000: Divided by 1 001: Divided by 2 010: Divided by 4 011: Divided by 8 100: Divided by 16 101: Divided by 32 While it is possible to configure the MPLL output divider as ÷ 1, this setting must not be used when the DDR is operational. This is to ensure that the clock to the DDR has an even mark:space ratio.
[15:6]	FACC_PLL_DIVF	0x2	Configures the MPLL feedback divider value, which is given by the binary value + 1. The binary value ranges from 000000000, which is the divisor value of 1, to 111111111, which is the divisor value of 1,024.
[5:0]	FACC_PLL_DIVR	0x1	Configures the MPLL reference divider value, which is given by binary value + 1. For example, if the value is 00000, then the divisor value is 1 (00000 + 1). Both REFCLK and post-divide REFCLK must be within the range specified in the IGLOO2 datasheet.

**Note:** The register fields for 005 and 010 devices must not be dynamically altered. See [System Registers Behavior for M2GL005/010 devices](#), page 206.

### 10.6.19.1 FACC\_PLL\_RANGE

**Table 151 • FACC\_PLL\_RANGE**

Bits[23:19]	PLL 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

### 10.6.20 HPMS\_DDR\_PLL Status High Configuration Register

**Table 152 • HPMS\_PLL\_STATUS\_HIGH\_CR**

Bit Number	Name	Reset Value	Description
[31:5]	Reserved	0	
[12:8]	FACC_PLL_SSMF	0	Drives the spread spectrum modulation frequency (SSMF) input of the MPLL. The only allowable value to be programmed in this field is 0, as spread spectrum mode is not supported for the MPLL.
[7:6]	FACC_PLL_SSMD	0	Drives the spread spectrum modulation depth (SSMD) input of the MPLL. The only allowable value to be programmed in this field is 0, as spread spectrum mode is not supported for the MPLL.
5	FACC_PLL_SSE	0	Drives the SSE input of the MPLL. The only allowable value to be programmed in this field is 0, as spread spectrum mode is not supported for the MPLL.
4	FACC_PLL_PD	0	A PD signal is provided for the lowest quiescent current. When PD is asserted, the MPLL powers down and outputs will be Low. PD has precedence over all other functions.
3	FACC_PLL_FSE	0	Configures PLL internal and external feedback paths. The only allowed value to be programmed in this field is 1.
2	FACC_PLL_MODE_3V3	0x1	Configures MPLL analog operational voltage. 0: 3.3 V 1: 2.5 V
1	FACC_PLL_MODE_1V2	0x1	Configures the PLL core voltage. 1: 1.2 V Do not write to this field

**Table 152 • HPMS\_PLL\_STATUS\_HIGH\_CR (continued)**

Bit Number	Name	Reset Value	Description
0	FACC_PLL_BYPASS	0	Powers down the MPLL core and bypasses it such that PLOUT tracks REFCLK.

**Note:** The register fields for 005 and 010 devices must not be dynamically altered. See [System Registers Behavior for M2GL005/010 devices](#), page 206.

## 10.6.21 HPMS DDR Fabric Alignment Clock Controller (FACC) Configuration Register 1

**Table 153 • HPMS\_FACC1\_CR**

Bit Number	Name	Reset Value	Description
[31:28]	Reserved	0	
27	FACC_FAB_REF_SEL	0	Selects the source of the reference clock to be supplied to the MPLL. Allowed values are: 0: 50 MHz RC oscillator 1: Fabric clock (CLK_BASE)
26	CONTROLLER_PLL_INIT	0x1	Indicates whether the FACC is to be configured for PLL initialization mode. The user can write to it when it detects that the MPLL has lost lock and it wants to switch to a known good clock source until the MPLL comes back into lock. This causes the 50 MHz clock to be selected through to the HPMS. It also interrupts the System Controller, which then waits for the MPLL to come into lock before clearing this bit and thereby selecting the MPLL output as the HPMS clock source again. The allowed values of this bit are: 0: The corresponding FACC multiplexer select lines or clock gate control line comes from the normal run-time configuration signals (from relevant HPMS system register bits). 1: The corresponding FACC multiplexer select lines or clock gate control line are overridden by hardwired PLL initialization selection, as described below: – Override the four no-glitch multiplexers related to the aligned clocks, so that they select CLK_STANDBY as the source of HPMS_CLK, APB_0_CLK, APB_1_CLK and DDR_FIC_CLK. – Override the selection of the FACC standby multiplexer, so that it selects the RCOSC_25_50MHZ clock as the source of CLK_STANDBY. – Override the selection of the FACC reference multiplexer, so that it selects CLK_BASE clock as the source of MPLL_REF_CLK. – Override the value of the PLL bypass configuration signal, so that it forces the MPLL bypass path not to be used. – Force MDDR_CLK to be gated off.

**Table 153 • HPMS\_FACC1\_CR (continued)**

Bit Number	Name	Reset Value	Description
25	PERSIST_CC	0	Feeds into the HPMS Reset Controller. Based on the value of PERSIST_CC, the Reset Controller asserts a reset (CC_RESET_N) to the FACC (which inverts it and passes it on to the PLL as HPMS_PLL_RESET), either on every HPMS system reset or just on power-up reset. Configure using flash bits. Do not write to this field. The only allowable value for this bit is 1. The reset signal for this register is PORESET_N.
[24:22]	BASE_DIVISOR	0	Indicates the ratio between CLK_A and the re-generated version of CLK_BASE, called CLK_BASE_REGEN. Do not write to this field. The allowed values are listed in <a href="#">Table 154</a> , page 222.
[21:19]	DDR_FIC_DIVISOR	0	Indicates the ratio between CLK_A and DDR_FIC_CLK. The user can write to this field dynamically during run time, even when the source clock is active. The allowed values are listed in <a href="#">Table 154</a> , page 222.
[18:16]	FIC_1_DIVISOR	0	Indicates the ratio between CLK_A and the clock being used in the fabric, to clock the soft IP block which is interfacing to FIC_1 of the HPMS. The user can write to this field dynamically during run time, even when the source clock is active. The allowed ratios for CLK_A:fabric clock (FIC_1) is listed in <a href="#">Table 154</a> , page 222.
[15:13]	FIC_0_DIVISOR	0	Indicates the ratio between CLK_A and the clock being used in the fabric, to clock the soft IP block which is interfacing to FIC_0 of the HPMS. The user can write to this field dynamically during run time, even when the source clock is active. The allowed ratios for CLK_A:fabric clock (FIC_0) are listed in <a href="#">Table 154</a> , page 222.
12	FACC_GLMUX_SEL	0	Contains the select line for the four no-glitch multiplexers within the FACC, which are related to the aligned clocks. All four of these multiplexers are switched by one signal. Allowed values: 1: HPMS_CLK, APB_0_CLK, APB_1_CLK, DDR_FIC_CLK all driven from CLK_STANDBY 0: HPMS_CLK, APB_0_CLK, APB_1_CLK, DDR_FIC_CLK all driven from stage B dividers Configure this field using flash bits. Do not write to this field.
[11:9]	HPMS_CLK_DIVISOR	0	Indicates the ratio between CLK_A and HPMS_CLK. The user can write to this field dynamically during run time, even when the source clock is active.
8	DDR_CLK_EN	0	Determines whether or not the clock to the MDDR block is to be gated off. Allowed values: 0: MDDR_CLK is gated off 1: MDDR_CLK is allowed to propagate through to MDDR block Do not write to this field dynamically while the source clock is active.
[7:5]	APB1_DIVISOR	0	Indicates the ratio between CLK_A and APB_1_CLK. The user can write to this field dynamically during run time, even when the source clock is active. The allowed values are described in <a href="#">Table 154</a> , page 222.

**Table 153 • HPMS\_FACC1\_CR (continued)**

Bit Number	Name	Reset Value	Description
[4:2]	APB0_DIVISOR	0	Indicates the ratio between CLK_A and APB_0_CLK. The user can write to this field dynamically during run time, even when the source clock is active. The allowed values are described in <a href="#">Table 154</a> , page 222.
[1:0]	DIVISOR_A	0	Indicates the ratio between CLK_SRC and CLK_A. Allowed values: 00: 1:1 01: 2:1 10: 3:1 11: Reserved Configure this field statically. Do not write to this field while the source clock is active.

**Note:** The register fields for 005 and 010 devices must not be dynamically altered. See [System Registers Behavior for M2GL005/010 devices](#), page 206.

### 10.6.21.1 Clock Ratio

**Table 154 • Clock Ratio**

Bits	Clock Ratio
000	1:1
001	2:1
010	4:1
100	8:1
101	16:1
110	32:1
Other values	Reserved

## 10.6.22 HPMS DDR Fabric Alignment Clock Controller Configuration Register 2

**Table 155 • HPMS\_FACC2\_CR**

Bit Number	Name	Reset Value	Description
[31:13]	Reserved	0	
12	HPMS_XTAL_EN	0x1	Enables the signal for the main crystal oscillator. If the main crystal oscillator is selected as the HPMS Flash*Freeze clock source, this bit must be asserted at all times (even when not in Flash*Freeze mode). 1: Enable 0: Disable
11	HPMS_CLK_ENVM_EN	0x1	Enables internal eNVM RC oscillator. Configure this field statically. Do not write to this field while the source clock is active.

Table 155 • HPMS\_FACC2\_CR (continued)

Bit Number	Name	Reset Value	Description
10	HPMS_1MHZ_EN	0x1	Enables the signal for the 1 MHz RC oscillator. If the 1 MHz RC oscillator is selected as the HPMS Flash*Freeze clock source, this bit must be asserted at all times (even when not in Flash*Freeze mode). 1: Enable 0: Disable
9	HPMS_25_50MHZ_EN	0x1	Enables the signal for the 50 MHz RC oscillator. If the 50 MHz RC oscillator is selected as the HPMS Flash*Freeze clock source, this bit must be asserted at all times (even when not in Flash*Freeze mode). 1: Enable 0: Disable
[8:6]	FACC_STANDBY_SEL	0	Contains the select lines for the three 2 to 1 no-glitch multiplexers, which implement the 4 to 1 no-glitch standby MUX function. This is used to allow one of 4 possible clocks to proceed through to the HPMS during FACC PLL Initialization Time. There are two MUXes in the first rank and these feed into a third MUX in the second rank. Bit 6 feeds into one of the first rank 2 to 1 MUXes (Standby MUX 0) and is defined as follows: 0: MUX 0 output comes from RCOSC_25_50MHZ 1: MUX 0 output comes from XTLOSC_CLK Bit 7 feeds into one of the first rank 2 to 1 MUXes (Standby MUX 1) and is defined as follows: 0: MUX 1 output comes from RCOSC_1MHZ Bit 8 feeds into the second rank 2 to 1 MUX (Standby MUX 2) and is defined as follows: 0: MUX 2 output comes from MUX 0 1: MUX 2 output comes from MUX 1 Do not write to this field while the standby clock is active.
5	FACC_PRE_SRC_SEL	0	Must always be 0. Allowed values: 0: RCOSC_1MHZ is fed through to the source no-glitch clock multiplexer.
[4:2]	FACC_SRC_SEL	0	Contains the select lines for the three 2 to 1 no-glitch multiplexers, which implement a 4 to 1 no-glitch source MUX function. There are two MUXes in the first rank and these feed into a third MUX in the second rank. Bit 2 feeds into one of the first rank 2 to 1 MUXes (Source MUX 0) and is defined as follows: 0: MUX 0 output comes from RCOSC_25_50MHZ 1: MUX 0 output comes from XTLOSC_CLK Bit 3 feeds into one of the first rank 2 to 1 MUXes (Source MUX 1) and is defined as follows: 0: MUX 1 output comes from RCOSC_1MHZ 1: MUX 1 output comes from MPLL_OUT_CLK Bit 4 feeds into the second rank 2 to 1 MUX (Source MUX 2) and is defined as follows: 0: MUX 2 output comes from MUX 0 1: MUX 2 output comes from MUX 1 When switching any of the no-glitch MUXes, both the clock being switched from and the clock being switched to must be running. Do not write to this field while the source clock is active.
[1:0]	Reserved	0	

### 10.6.23 HPMS Clock Calibration Control Register

Table 156 • HPMS\_CLK\_CALIB\_CR

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0	
0	FAB_CALIB_START	0	Writing to this bit causes a one clock tick pulse to be generated on FABCALIBSTART. This is used to start an FPGA fabric calibration test circuit.

### 10.6.24 PLL Delay Line Select Control Register

Table 157 • PLL\_DELAY\_LINE\_SEL\_CR

Bit Number	Name	Reset Value	Description
[31:4]	Reserved	0	
[3:2]	PLL_FB_DEL_SEL	0	Must be programmed to a specific value by Libero SoC and never modified after that.
[1:0]	PLL_REF_DEL_SEL	0	Must be programmed to a specific value by Libero SoC and never modified after that.

### 10.6.25 Reset Source Control Register

Table 158 • RESET\_SOURCE\_CR

Bit Number	Name	Reset Value	Description
[31:7]	Reserved	0	
6	USER_RESET_DETECT	0x1	Indicates that an HPMS user reset has occurred. During the device boot sequence, this register should be cleared to arm it to detect the next reset event. Reset signal: HPMS_RESET_F2M_HPMS_CLK_N
[5:2]	Reserved	0	
1	CONTROLLER_RESET_DETECT	0x1	Indicates that an HPMS controller reset has occurred. During the device boot sequence, this register should be cleared to arm it to detect the next reset event. Reset signal: SC_HPMS_RESET_HPMS_CLK_N.
0	PO_RESET_DETECT	0x1	Indicates that a power-up reset has occurred. During the device boot sequence, this register should be cleared to arm it to detect the next reset event. The reset signal for this bit is PO_RESET_HPMS_CLK_N.

## 10.6.26 HPMS DDR Bridge High Performance DMA Master Error Address Status Register

Table 159 • DDRB\_HPD\_ERR\_ADR\_SR

Bit Number	Name	Reset Value	Description
[31:0]	DDR_B_HPD_ERR_ADD	0	If a write transfer initiated at the HPMS DDR bridge arbiter interface to empty data present in the write buffer of the HPDMA master which receives an error response, the address for which the error response is received is placed in this register. Address indicates TAG value for which error response is received. The following values are updated in this register as per buffer size: 16 bytes: DDR_B_HPD_ERR_ADR[31:4] = TAG, DDR_B_HPD_ERR_ADR[3:0] = 0000 32 bytes: DDR_B_HPD_ERR_ADR[31:5] = TAG[27:1], DDR_B_HPD_ERR_ADR[4:0] = 0000.

## 10.6.27 HPMS DDR Bridge AHB Bus Error Address Status Register

Table 160 • DDRB\_SW\_ERR\_ADR\_SR

Bit Number	Name	Reset Value	Description
[31:0]	DDR_B_SW_ERR_ADD	0	If a write transfer initiated at the HPMS DDR bridge arbiter interface to empty data present in the write buffer allocated for the AHB bus, which receives an error response, the address for which the error response is received is placed in this register. Address indicates TAG value for which the error response is received. The following values are updated in this register as per buffer size: 16 bytes: DDR_B_SW_ERR_ADR[31:4] = TAG, DDR_B_SW_ERR_ADR [3:0] = 0000 32 bytes: DDR_B_SW_ERR_ADR [31:5] = TAG[27:1], DDR_B_SW_ERR_ADR [4:0] = 0000.

## 10.6.28 HPMS DDR Bridge Buffer Empty Status Register

Table 161 • DDRB\_BUF\_EMPTY\_SR

Bit Number	Name	Reset Value	Description
[31:7]	Reserved	0	
6	Reserved	0	
5	DDR_B_HPD_RBEMPTY	0	When set to '1', indicates that the read buffer of the HPDMA master does not have valid data
4	DDR_B_HPD_WBEMPTY	0	When set to '1', indicates that the write buffer of the HPDMA master does not have valid data
3	DDR_B_SW_RBEMPTY	0	When set to '1', indicates that the read buffer of the AHB bus matrix master does not have valid data
2	DDR_B_SW_WBEMPTY	0	When set to '1', indicates that the write buffer of the AHB bus matrix master does not have valid data

**Table 161 • DDRB\_BUF\_EMPTY\_SR (continued)**

1	Reserved	0
0	Reserved	0

## 10.6.29 HPMS DDR Bridge Disable Buffer Status Register

**Table 162 • DDRB\_DSBL\_DN\_SR**

Bit Number	Name	Reset Value	Description
[31:7]	Reserved	0	
6	Reserved	0	
5	DDRB_HPD_RDSBL_DN	0	Is set to '1' once the HPDMA read buffer is disabled after getting a read buffer disable command from fabric logic.
4	DDRB_HPD_WDSBL_DN	0	Is set to '1' once the HPDMA write buffer is disabled after getting a write buffer disable command from fabric logic.
3	DDRB_SW_RDSBL_DN	0	Is set to '1' once the AHB bus matrix read buffer is disabled after getting a read buffer disable command from fabric logic.
2	DDRB_SW_WDSBL_DN	0	Is set to '1' once the AHB bus matrix write buffer is disabled after getting a write buffer disable command from fabric logic.
1	Reserved	0	
0	Reserved	0	

## 10.6.30 eSRAM0 EDAC Count

**Table 163 • ESRAM0\_EDAC\_CNT**

Bit Number	Name	Reset Value	Description
[31:16]	ESRAM0_EDAC_CNT_2E	0	16-bit counter value in eSRAM0 incremented by eSRAM0 EDAC 2-bit error. The counter will not roll back and will stay at its maximum value.
[15:0]	ESRAM0_EDAC_CNT_1E	0	16-bit counter value in eSRAM0 incremented by eSRAM0 EDAC 1-bit error. The counter will not roll back and will stay at its maximum value.

## 10.6.31 eSRAM1 EDAC Count

**Table 164 • ESRAM1\_EDAC\_CNT**

Bit Number	Name	Reset Value	Description
[31:16]	ESRAM1_EDAC_CNT_2E	0	16-bit counter value in eSRAM1 incremented by eSRAM1 EDAC 2-bit error. The counter will not roll back and will stay at its maximum value.
[15:0]	ESRAM1_EDAC_CNT_1E	0	16-bit counter value in eSRAM1 incremented by eSRAM1 EDAC 1-bit error. The counter will not roll back and will stay at its maximum value.

## 10.6.32 eSRAM0 EDAC Address Register

**Table 165 • ESRAM0\_EDAC\_ADR**

Bit Number	Name	Reset Value	Description
[31:25]	Reserved	0	
[25:13]	ESRAM0_EDAC_2E_AD	0	Stores the address from eSRAM0 on which a 2-bit SECEDED error has occurred.
[12:0]	ESRAM0_EDAC_1E_AD	0	Stores the address from eSRAM0 on which a 1-bit SECEDED error has occurred.

## 10.6.33 eSRAM1 EDAC Address Register

**Table 166 • ESRAM1\_EDAC\_ADR**

Bit Number	Name	Reset Value	Description
[31:25]	Reserved	0	
[25:13]	ESRAM1_EDAC_2E_AD	0	Stores the address from eSRAM1 on which a 2-bit SECEDED error has occurred.
[12:0]	ESRAM1_EDAC_1E_AD	0	Stores the address from eSRAM1 on which a 1-bit SECEDED error has occurred.

## 10.6.34 Security Configuration Register for Masters 4, 5, and DDR\_FIC

**Table 167 • MM4\_5\_DDR\_FIC\_SECURITY/MM4\_5\_FIC64\_SECURITY**

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0	
9	MM4_5_DDR_FIC_MS6_ALLOWED_W	1	Write security bits for masters 4, 5, and DDR_FIC to slave 6 (HPMS DDR bridge). If not set, masters 4, 5 and DDR_FIC will not have write access to slave 6.
8	MM4_5_DDR_FIC_MS6_ALLOWED_R	1	Read security bits for masters 4, 5, and DDR_FIC to slave 6 (HPMS DDR bridge). If not set, masters 4, 5, and DDR_FIC will not have read access to slave 6.
7	MM4_5_DDR_FIC_MS3_ALLOWED_W	1	Write security bits for masters 4, 5, and DDR_FIC to slave 3 (eNVM1). If not set, masters 4, 5, and DDR_FIC will not have write access to slave 3.
6	MM4_5_DDR_FIC_MS3_ALLOWED_R	1	Read security bits for masters 4, 5, and DDR_FIC to slave 3 (eNVM1). If not set, masters 4, 5, and DDR_FIC will not have read access to slave 3.
5	MM4_5_DDR_FIC_MS2_ALLOWED_W	1	Write Security Bits for masters 4, 5, and DDR_FIC to slave 2 (eNVM0). If not set, masters 4, 5, and DDR_FIC will not have write access to slave 2.
4	MM4_5_DDR_FIC_MS2_ALLOWED_R	1	Read security bits for masters 4, 5, and DDR_FIC to slave 2 (eNVM0). If not set, masters 4, 5, and DDR_FIC will not have read access to slave 2.

**Table 167 • MM4\_5\_DDR\_FIC\_SECURITY/MM4\_5\_FIC64\_SECURITY (continued)**

3	MM4_5_DDR_FIC_MS1_ALLOWED_ W	1	Write security bits for masters 4, 5, and DDR_FIC to slave 1 (eSRAM1). If not set, masters 4, 5, and DDR_FIC will not have write access to slave 1.
2	MM4_5_DDR_FIC_MS1_ALLOWED_ R	1	Read security bits for masters 4, 5, and DDR_FIC to slave 1 (eSRAM1). If not set, masters 4, 5, and DDR_FIC will not have read access to slave 1.
1	MM4_5_DDR_FIC_MS0_ALLOWED_ W	1	Write security bits for masters 4, 5, and DDR_FIC to slave 0 (eSRAM0). If not set, masters 4, 5, and DDR_FIC will not have write access to slave 0.
0	MM4_5_DDR_FIC_MS0_ALLOWED_ R	1	Read security bits for masters 4, 5, and DDR_FIC to slave 0 (eSRAM0). If not set, masters 4, 5, and DDR_FIC will not have read access to slave 0.

### 10.6.35 Security Configuration Register for Masters 3 and 7

**Table 168 • MM3\_7\_SECURITY**

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0	
9	MM3_7_MS6_ALLOWED_W	1	Write security bits for masters 3, 7 to slave 6 (HPMS DDR bridge). If not set, masters 3 and 7 will not have write access to slave 6.
8	MM3_7_MS6_ALLOWED_R	1	Read security bits for masters 3, 7 to slave 6 (HPMS DDR bridge). If not set, masters 3 and 7 will not have read access to slave 6.
7	MM3_7_MS3_ALLOWED_W	1	Write security bits for masters 3, 7 to slave 3 (eNVM1). If not set, masters 3 and 7 will not have write access to slave 3.
6	MM3_7_MS3_ALLOWED_R	1	Read security bits for masters 3, 7 to slave 3 (eNVM1). If not set, masters 3 and 7 will not have read access to slave 3.
5	MM3_7_MS2_ALLOWED_W	1	Write security bits for masters 3, 7 to slave 2 (eNVM0). If not set, masters 3 and 7 will not have write access to slave 2.
4	MM3_7_MS2_ALLOWED_R	1	Read security bits for masters 3, 7 to slave 2 (eNVM0). If not set, masters 3 and 7 will not have read access to slave 2.
3	MM3_7_MS1_ALLOWED_W	1	Write security bits for masters 3, 7 to slave 1 (eSRAM1). If not set, masters 3 and 7, will not have write access to slave 1.
2	MM3_7_MS1_ALLOWED_R	1	Read security bits for masters 3, and 7 to slave 1 (eSRAM1). If not set, masters 3 and 7 will not have read access to slave 1.
1	MM3_7_MS0_ALLOWED_W	1	Write security bits for masters 3, 7 to slave 0 (eSRAM0). If not set, masters 3 and 7 will not have write access to slave 0.
0	MM3_7_MS0_ALLOWED_R	1	Read security bits for masters 3, 7 to slave 0 (eSRAM0). If not set, masters 3 and 7 will not have read access to slave 0.

## 10.6.36 Security Configuration Register for Master 9

**Table 169 • MM9\_SECURITY**

Bit Number	Name	Reset Value	Description
[31:10]	Reserved	0	
9	MM9_MS6_ALLOWED_W	1	Write security bits for master 9 to slave 6 (HPMS DDR bridge). If not set, master 9 will not have write access to slave 6.
8	MM9_MS6_ALLOWED_R	1	Read security bits for master 9 to slave 6 (HPMS DDR bridge). If not set, master 9 will not have read access to slave 6.
7	MM9_MS3_ALLOWED_W	1	Write security bits for master 9 to slave 3 (eNVM1). If not set, master 9 will not have write access to slave 3.
6	MM9_MS3_ALLOWED_R	1	Read security bits for master 9 to slave 3 (eNVM1). If not set, master 9 will not have read access to slave 3.
5	MM9_MS2_ALLOWED_W	1	Write security bits for master 9 to slave 2 (eNVM0). If not set, master 9 will not have write access to slave 2.
4	MM9_MS2_ALLOWED_R	1	Read security bits for master 9 to slave 2 (eNVM0). If not set, master 9 will not have read access to slave 2.
3	MM9_MS1_ALLOWED_W	1	Write security bits for master 9 to slave 1 (eSRAM1). If not set, master 9 will not have write access to slave 1.
2	MM9_MS1_ALLOWED_R	1	Read security bits for master 9 to slave 1 (eSRAM1). If not set, master 9 will not have read access to slave 1.
1	MM9_MS0_ALLOWED_W	1	Write security bits for master 9 to slave 0 (eSRAM0). If not set, master 9 will not have write access to slave 0.
0	MM9_MS0_ALLOWED_R	1	Read security bits for master 9 to slave 0 (eSRAM0). If not set, master 9 will not have read access to slave 0.

## 10.6.37 Device Status Register

**Table 170 • DEVICE\_SR**

Bit Number	Name	Reset Value	Description
[31:7]	Reserved	0	
6	Reserved	0	
5	Reserved	0	
4	FLASH_VALID_SYNC	0	Asserted when FPGA fabric is valid. There is no reset signal for this bit. 0: FPGA fabric flash bits are valid and operational 1: FPGA fabric flash bits are not operational
3	Reserved	0	
2	FF_IN_PROGRESS_SYNC	0	Indicates the FF_IN_PROGRESS STATE. There is no reset signal for this bit.

**Table 170 • DEVICE\_SR (continued)**

Bit Number	Name	Reset Value	Description
1	VIRGIN_PART	0x1	Indicates the device as virgin or non-virgin type. There is no reset signal for this bit. 0: Device is not a virgin part. It has been through a programming cycle to at least configure the factory settings 1: Device is a virgin part. It has never been through any programming cycle and all internal flash bits are invalid
0	CORE_UP_SYNC	0	Indicates the status of the synchronized CORE_UP input from the system controller. There is no reset signal for this bit.

## 10.6.38 eNVM Protect User Register

**Table 171 • ENVM\_PROTECT\_USER**

Bit Number	Name	Reset Value	Description
[31:16]	Reserved	0	
15	NVM1_UPPER_WRITE_ALLOWED	0x1	When set, indicates that the masters who have read access can have write access to the upper protection region of eNVM1. This is updated by the user flash row bit.
14	NVM1_UPPER_OTHERS_ACCESS	0x1	When set, indicates that the other masters can access the upper protection region of eNVM1. This is set by the user flash row bit.
13	NVM1_UPPER_FABRIC_ACCESS	0x1	When set, indicates that the fabric can access the upper protection region of eNVM1. This is set by the user flash row bit.
12	Reserved	0	
11	NVM1_LOWER_WRITE_ALLOWED	0x1	When set, indicates that the masters who have read access can have write access to the lower protection region of eNVM1. This is set by the user flash row bit.
10	NVM1_LOWER_OTHERS_ACCESS	0x1	When set, indicates that the other masters can access the lower protection region of eNVM1. This is set by the user flash row bit.
9	NVM1_LOWER_FABRIC_ACCESS	0x1	When set, indicates that the fabric can access the lower protection region of eNVM1. This is set by user flash row bit.
8	Reserved	0	
7	NVM0_UPPER_WRITE_ALLOWED	0x1	When set, indicates that the masters who have read access can have write access to the upper protection region of eNVM0. This will be set by the user flash row bit.
6	NVM0_UPPER_OTHERS_ACCESS	0x1	When set, indicates that the other masters can access the upper protection region of eNVM0.
5	NVM0_UPPER_FABRIC_ACCESS	0x1	When set, indicates that the fabric can access the upper protection region of eNVM0. This will be set by the user flash row bit.
4	Reserved	0	

**Table 171 • ENVM\_PROTECT\_USER (continued)**

Bit Number	Name	Reset Value	Description
3	NVM0_LOWER_WRITE_ALLOWED	0x1	When set, indicates that the masters who have read access can have write access to the lower protection region of eNVM0. This will be set by the user flash row bit.
2	NVM0_LOWER_OTHERS_ACCESS	0x1	When set, indicates that the other masters can access the lower protection region of eNVM0. This will be set by the user flash row bit.
1	NVM0_LOWER_FABRIC_ACCESS	0x1	When set, indicates that the fabric can access the lower protection region of eNVM0. This will be set by the user flash row bit.
0	Reserved	0	

### 10.6.39 IGLOO2 eNVM Status Register

**Table 172 • ENVM\_STATUS**

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0	
0	CODE_SHADOW_EN	0	Read by the system controller during device start-up, to indicate whether the user has configured the device such that code shadowing is to be performed by system controller fabric logic.

### 10.6.40 Device Version Register

**Table 173 • DEVICE\_VERSION**

Bit Number	Name	Reset Value	Description
[31:20]	Reserved	0	
[19:16]	IDV	0	Internal device version.
[15:0]	IDP	0	Internal device product.

### 10.6.41 HPMS PLL Status Register

**Table 174 • HPMS\_PLL\_STATUS**

Bit Number	Name	Reset Value	Description
[31:3]	Reserved	0	
2	RCOSC_DIV2		Input from the System Controller, indicating whether the 50 MHz RC oscillator is running at 25 MHz or 50 MHz. 0: Running at 25MHz 1: Running at 50MHz

**Table 174 • HPMS\_PLL\_STATUS (continued)**

Bit Number	Name	Reset Value	Description
1	MPLL_LOCK	0	<p>MPLL lock status.</p> <p>A LOCK signal is provided to indicate that the MPLL has locked on to the incoming signal. LOCK asserts High to indicate that the MPLL has achieved frequency and phase lock. Allowed values are:</p> <p>0: MPLL is not in lock 1: MPLL is in lock</p> <p>Microsemi recommends that LOCK is only used for test and system status information, and is not used for critical system functions without thorough characterization in the host system. The precision of the LOCK discrimination can be adjusted using the LOCKWIN[2:0] controls. The integration of the LOCK period can be adjusted using the LOCKCNT[3:0] controls.</p>
0	FAB_PLL_LOCK	0	<p>If CLK_BASE is generated from a PLL in the fabric, this signal must be connected from the LOCK output of that PLL. When the FACC is going through its PLL initialization stage (either under system controller control or HPMS master control), this signal is ANDed with the LOCK output of the MPLL. Only when both PLLs are in lock, is the system considered to be ready for switching to PLL-derived clock. If CLK_BASE is not derived from a fabric PLL, then the user must ensure that this signal is tied High at the fabric interface. Allowed values:</p> <p>0: Fabric PLL is not in lock 1: Fabric PLL is in lock or CLK_BASE is not derived from a fabric PLL</p>

## 10.6.42 eNVM Status Register

**Table 175 • ENVM\_SR**

Bit Number	Name	Reset Value	Description
[31:2]	Reserved	0	
[1:0]	ENVM_BUSY	0	<p>Active high signals indicate a busy state per eNVM for CLK-driven operations and for internal operations triggered by the write/program/erase/transfer command.</p> <p>ENVM_BUSY[1] = Busy indication from eNVM1 ENVM_BUSY[0] = Busy indication from eNVM0</p>

## 10.6.43 DDRB Status Register

Table 176 • DDRB\_STATUS

Bit Number	Name	Reset Value	Description
[31:0]	DDR_DEBUG_STATUS	0x1	<p>Status of the internal ports of DDRBRIDGE. The bit definitions are as follows:</p> <p>Debug ports of the HPMS DDR bridge:</p> <p>SYR_DDRB_DP[31:30] = Reserved</p> <p>SYR_DDRB_DP[29:28] = AHB bus write buffer mode status</p> <p>SYR_DDRB_DP[27:26] = HPDMA write buffer mode status</p> <p>SYR_DDRB_DP[25:23] = Reserved</p> <p>SYR_DDRB_DP[22:20] = Reserved</p> <p>SYR_DDRB_DP[19:17] = AHB bus read buffer mode status</p> <p>SYR_DDRB_DP[16:14] = HPDMA read buffer mode status</p> <p>SYR_DDRB_DP[13] = Reserved</p> <p>SYR_DDRB_DP[12] = AHB bus write request to arbiter</p> <p>SYR_DDRB_DP[11] = HPDMA write request to arbiter</p> <p>SYR_DDRB_DP[10] = Reserved</p> <p>SYR_DDRB_DP[9] = Reserved</p> <p>SYR_DDRB_DP[8] = AHB bus read req to arbiter</p> <p>SYR_DDRB_DP[7] = HPDMA read request to arbiter</p> <p>SYR_DDRB_DP[6] = Reserved</p> <p>SYR_DDRB_DP[5] = AXI write address channel acknowledge to AHB bus write request</p> <p>SYR_DDRB_DP[4] = AXI write address channel acknowledge to HPDMA write request</p> <p>SYR_DDRB_DP[3] = Reserved</p> <p>SYR_DDRB_DP[2] = AXI write data channel acknowledge to AHB bus write request</p> <p>SYR_DDRB_DP[1] = AXI write data channel acknowledge to HPDMA write request</p> <p>SYR_DDRB_DP[0] = Lock input to arbiter from AHB bus WCB</p>

## 10.6.44 MDDR IO Calibration Status Register

Table 177 • MDDR\_IO\_CALIB\_STATUS

Bit Number	Name	Reset Value	Description
[31:15]	Reserved	0	
14	CALIB_PCOMP	0x1	State of the P analog comparator
13	CALIB_NCOMP	0x1	State of the N analog comparator
[12:6]	CALIB_PCODE	0x3F	Current PCODE value set on the MDDR DDR I/O bank
[5:1]	CALIB_NCODE	0x3F	Current NCODE value set on the MDDR DDR I/O bank
0	CALIB_STATUS	0	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 fabric logic (CALIB_LOCK).

## 10.6.45 HPMS Clock Calibration Status

**Table 178 • HPMS\_CLK\_CALIB\_STATUS**

Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0	
0	FAB_CALIB_FAIL	0	0: The currently selected CCC delay values for the HPMS_CLK and fabric Clock are such that the FPGA fabric clock calibration circuit is running correctly. 1: The FPGA fabric clock calibration circuit is failing to operate correctly. This indicates incorrectly configured delay values for HPMS_CLK and/or fabric clock in the CCC.

## 10.6.46 Fabric Protected Size Register

**Table 179 • FAB\_PROT\_SIZE**

Bit Number	Name	Reset Value	Description
[31:6]	Reserved	0	
[5:0]	SW_PROTREGIONSIZE	11110	The size of the memory region inaccessible to the FPGA fabric master is determined by the value of this bus. The region sizes are listed in <a href="#">Table 180 on page 234</a> .

### 10.6.46.1 Region Size

**Table 180 • Region Size**

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Size
0	0	0	0	0	Reserved
0	0	0	0	1	Reserved
0	0	0	1	0	Reserved
0	0	0	1	1	Reserved
0	0	1	0	0	Reserved
0	0	1	0	1	Reserved
0	0	1	1	0	128 Bytes
0	0	1	1	1	Reserved
0	1	0	0	0	Reserved
0	1	0	0	1	Reserved
0	1	0	1	0	2 Kbytes
0	1	0	1	1	Reserved
0	1	1	0	0	Reserved
0	1	1	0	1	16 Kbytes
0	1	1	1	0	32 Kbytes

**Table 180 • Region Size (continued)**

0	1	1	1	1	64 Kbytes
1	0	0	0	0	128 Kbytes
1	0	0	0	1	256 Kbytes
1	0	0	1	0	512 Kbytes
1	0	0	1	1	Reserved
1	0	1	0	0	Reserved
1	0	1	0	1	Reserved
1	0	1	1	0	8 Mbytes
1	0	1	1	1	Reserved
1	1	0	0	0	Reserved

## 10.6.47 Fabric Protected Base Address Register

**Table 181 • FAB\_PROT\_BASE**

Bit Number	Name	Reset Value	Description
[31:0]	SW_PROTREGIONBASE	0	<p>The base address of the memory region inaccessible to the FPGA fabric master is determined by the value of this bus. Bit 0 of this bus is defined as SW_PROTREGIONENABLE. This has the following meaning:</p> <p>0: Protected region not enabled. This means that a master in the FPGA fabric may access any location in the memory map, as long as the fabric master port is enabled.</p> <p>1: Protected region enabled. Any accesses attempted by a fabric master to this region of memory return an error in the bus transaction.</p> <p>Bits [31:N] of this bus indicate the base address of the protected region.</p>

The value of N depends on the protected region size, so that the base address is aligned according to an even multiple of region size. The power of 2 size specified by SW\_PROTREGIONSIZ[4:0] defines how many bits of base address are used. For example, if the SW\_PROTREGIONSIZ[4:0] is 01111, this corresponds to a protected region of 64 KB. 64 KB is 2 to the power of 16. Therefore the value of N in this case is 16. So the base address of the region, in this case, is specified by SW\_PROTREGIONBASE[31:16].

## 10.6.48 EDAC Status Register

**Table 182 • EDAC\_SR**

Bit Number	Name	Reset Value	Description
[31:6]	Reserved	0	
5	Reserved	0	
4	Reserved	0	
3	ESRAM1_EDAC_2E	0	Updated by the eSRAM_1 controller when a 2-bit SECEDED error has been detected for eSRAM1 memory.

**Table 182 • EDAC\_SR (continued)**

2	ESRAM1_EDAC_1E	0	Updated by the eSRAM_1 Controller when a 1-bit SECEDED error has been detected and is corrected for eSRAM1 memory.
1	ESRAM0_EDAC_2E	0	Updated by the eSRAM_0 controller when a 2-bit SECEDED error has been detected for eSRAM0 memory.
0	ESRAM0_EDAC_1E	0	Updated by the eSRAM_0 controller when a 1-bit SECEDED error has been detected and is corrected for eSRAM0 memory.

## 10.6.49 HPMS Internal Status Register

**Table 183 • HPMS\_INTERNAL\_SR**

Bit Number	Name	Reset Value	Description
[31:7]	Reserved	0	
6	DDR_FIC_INT	0	Indicates an interrupt from DDR_FIC.
5	MDDR_ECC_INT	0	Indicates when an SECEDED interrupt from the MDDR subsystem is asserted.
4	MDDR_IO_CALIB_INT	0	Interrupt is generated when the MDDR calibration is finished.
3	FAB_PLL_LOCKLOST_INT	0	Indicates that a falling edge event occurred on the FAB_PLL_LOCK signal. This indicates that the fabric PLL lost lock.
2	FAB_PLL_LOCK_INT	0	Indicates that a rising edge event occurred on the FAB_PLL_LOCK signal. This indicates that the fabric PLL came into lock.
1	MPLL_LOCKLOST_INT	0	Indicates that a falling edge event occurred on the MPLL_LOCK signal. This indicates that the MPLL lost lock.
0	MPLL_LOCK_INT	0	Indicates that a rising edge event occurred on the MPLL_LOCK signal. This indicates that the MPLL came into lock.

## 10.6.50 HPMS External Status Register

**Table 184 • HPMS\_EXTERNAL\_SR**

Bit Number	Name	Reset Value	Description
[31:18]	Reserved	0	
17	DDRB_LOCK_MID	0	Indicates which master (AHB bus or HPDMA) is responsible for lock timeout condition. 0: AHB bus master 1: HPDMA
16	DDRB_LCKOUT	0	Asserted when lock timeout counter reaches its maximum value. Lock time out counter (20-bit) is maintained in the HPMS DDR bridge, which starts counting when a locked transfer obtains access to the AXI bus. When the counter reaches maximum value, a DDRB_LCKOUT interrupt is generated and stays asserted until cleared by the fabric logic.

**Table 184 • HPMS\_EXTERNAL\_SR (continued)**

Bit Number	Name	Reset Value	Description
15	DDRB_HPD_WR_ERR	0	Asserted when the HPMS DDR bridge gets an error response from the DDR slave for an HPDMA write request. Address of write transaction for which error response is received is provided by DDRB_HPD_ERR_ADD.
14	DDRB_SW_WR_ERR	0	Asserted when the HPMS DDR bridge gets an error response from the DDR slave for an AHB bus master write request. Address of write transaction for which error response is received is provided by DDRB_SW_ERR_ADD.
13	Reserved	0	
[12:7]	DDRB_RDWR_ERR_REG	0	Provides the read/write address match error status generated during the following accesses: Bit 0 = 1: AHB bus and HPDMA are trying to access same address Bit [5:1] = 0
[6:0]	SW_ERRORSTATUS	0	Indicates whether any accesses by the corresponding master on the AHB bus resulted in either HRESP assertion by the slave to the AHB bus, HRESP assertion by the AHB bus to that master (in the case of blocked fabric master), or was decoded by the AHB bus as being to “unimplemented” address space. The bit definitions are as follows: Bit 0: Corresponds to an HRESP assertion being issued to the HPDMA interface Bit 1: Corresponds to an HRESP assertion being issued to FIC_0 interface Bit 2: Corresponds to an HRESP assertion being issued to FIC_1 interface Bit 4: Corresponds to an HRESP assertion being issued to the peripheral DMA engine Bit 6: Corresponds to an HRESP assertion being issued to the System Controller

## 10.6.51 Clear EDAC Counters

**Table 185 • CLR\_EDAC\_COUNTERS**

Bit Number	Name	Reset Value	Description
[31:4]	Reserved	0	
3	ESRAM1_EDAC_CNTCLR_2E	0	Pulse generated to clear the 16-bit counter value in eSRAM1 corresponding to the count value of EDAC 2-bit errors. This in turn clears the upper 16 bits of the eSRAM1_EDAC_CNT Register.
2	ESRAM1_EDAC_CNTCLR_1E	0	Pulse generated to clear the 16-bit counter value in eSRAM1 corresponding to count value of EDAC 1-bit errors. This in turn clears the lower 16 bits of the eSRAM1_EDAC_CNT Register.
1	ESRAM0_EDAC_CNTCLR_2E	0	Pulse generated to clear the 16-bit counter value in eSRAM0 corresponding to count value of EDAC 2bit Errors. This in turn clears the upper 16 bits of eSRAM0_EDAC_CNT the register.

**Table 185 • CLR\_EDAC\_COUNTERS (continued)**

Bit Number	Name	Reset Value	Description
0	ESRAM0_EDAC_CNTCLR_1E	0	Pulse generated to clear the 16-bit counter value in eSRAM0 corresponding to the count value of EDAC 1-bit errors. This in turn clears the lower 16 bits of the ESRAM0_EDAC_CNT Register.

## 10.6.52 Flush Configuration Register

**Table 186 • FLUSH\_CR**

Bit Number	Name	Reset Value	Description
[31:8]	Reserved	0	
7	DDRB_INVALID_HPD	0	Allows the read buffer allocated for the AHB bus in the HPMS DDR bridge to be invalidated. 0: No effect 1: Invalidate HPD read buffer
6	DDRB_INVALID_SW	0	Allows the read buffer for the high performance master in the HPMS DDR bridge to be invalidated. 0: No effect 1: Invalidate AHB Bus read buffer
5	Reserved	0	Reserved
4	DDRB_FLSHSW	0	Allows the write buffer for the AHB bus in the HPMS DDR bridge to be flushed. Data present in the write buffer is transferred to the HPMS DDR bridge write arbiter interface when this pulse is detected. 0: No effect 1: Flush AHB bus write buffer
3	DDRB_FLSHHPD	0	Allows the write buffer for the HPD master in the HPMS DDR bridge to be flushed. Data present in the write buffer is transferred to the HPMS DDR bridge write arbiter interface when this pulse is detected. 0: No effect 1: Flush HPD write buffer
[2:0]	Reserved	0	

# 11 Fabric Interface Interrupt Controller

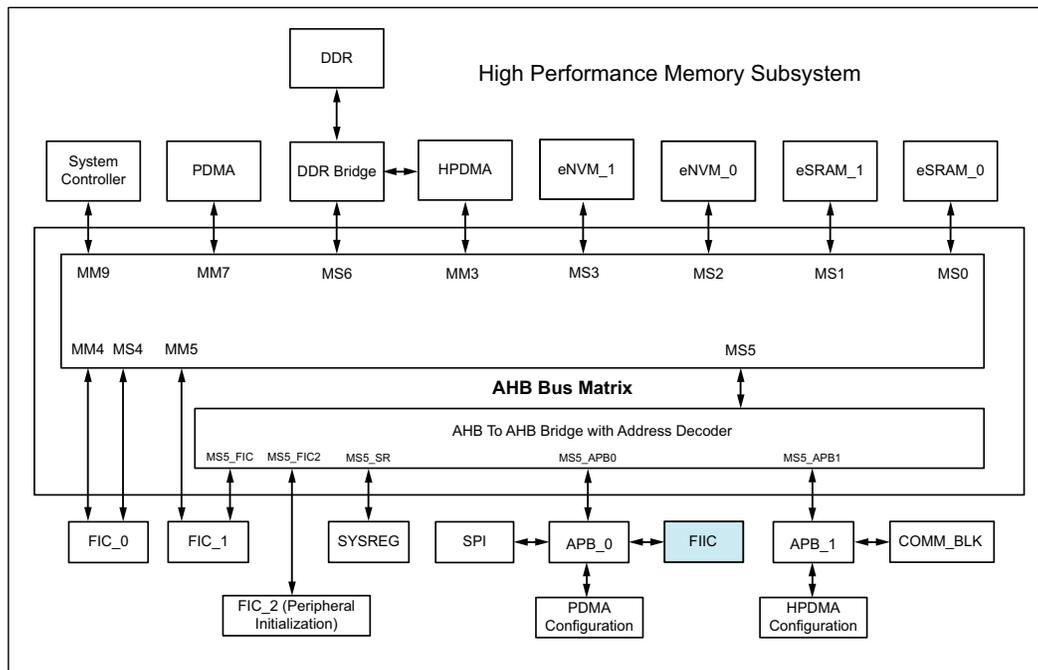
The fabric interface interrupt controller (FIIC) gathers interrupt signals from within the high performance memory subsystem (HPMS) and makes them available to the FPGA fabric. There are a number of peripherals and other blocks within the HPMS that generate interrupt signals. These interrupt signals can be used as a potential interrupt sources to a user logic within the FPGA fabric.

## 11.1 Features

- FIIC receives 24 interrupts from the HPMS as inputs
- 16 individually configurable HPMS to fabric interrupt ports

The following figure depicts the connectivity of FIIC to the AHB bus matrix.

**Figure 142 • FIIC Connection to AHB Bus Matrix**



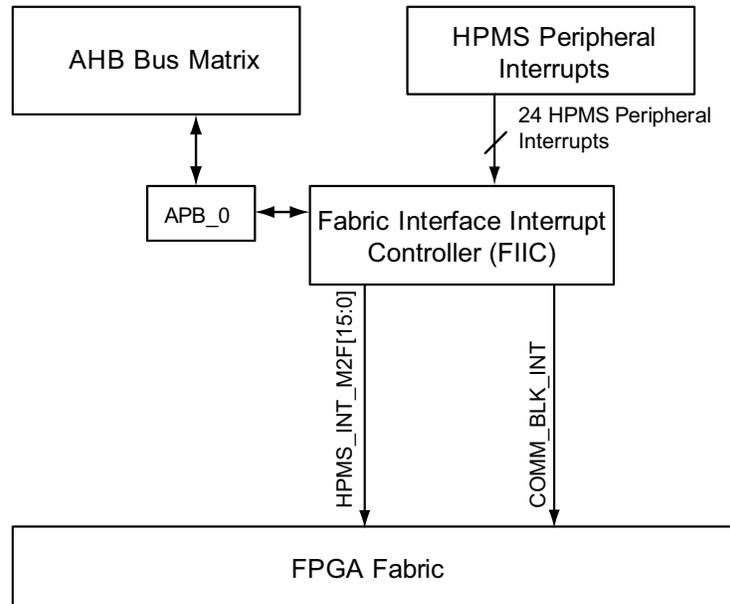
## 11.2 Functional Description

This section provides the detailed description of the FIIC subsystem.

### 11.2.1 Architecture Overview

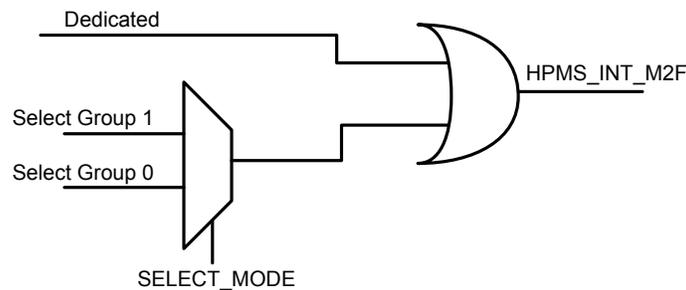
The following figure shows the interfacing of the FIIC with HPMS peripheral interrupts and FPGA fabric. The FIIC receives 24 level-sensitive active high interrupts from the HPMS as inputs. These HPMS peripheral interrupts are combined, in a predetermined fashion, into 16 M2F interrupts (HPMS\_INT\_M2F[15:0]) routed to the fabric. There is also a COMM\_BLK interrupt, COMM\_BLK\_INT.

**Figure 143 • Block Diagram for Fabric Interface Interrupt Controller**



There are 16 circuits, as shown in the following figure. Each circuit corresponds to a row in [Table 187](#), page 241. The dedicated interrupts coming from the HPMS peripherals are always connected to the 16 M2F interrupt signals.

**Figure 144 • Combinational Circuit for Mapping HPMS Interrupts to a HPMS\_INT\_M2F**



Each fabric HPMS\_INT\_M2F signal can be triggered from one of two possible scenarios:

- Dedicated interrupts
- Multiplexed group of interrupts

The selection of the HPMS interrupt to a specific HPMS\_INT\_M2F signal and making it available to the FPGA fabric is done in two stages:

- Select the group of interrupts - It can be done by setting the Select\_Mode bit of the M2F Interrupt Mode Register.
- Enable the interrupt - It can be done by writing to the appropriate FIIC INTERRUPT\_ENABLE0 and INTERRUPT\_ENABLE1 Interrupt Enable Registers.

**Table 187 • Interrupt Line Signal Distribution**

M2F Interrupt Signal	Dedicated	Select Group 0	Select Group 1
HPMS_INT_M2F[15]	HPMSDDR_PLL_LOCKLOST_INT	COMBLK_INTR	FIC64_INT
HPMS_INT_M2F[14]	Reserved	SOFTINTERRUPT	FAB_PLL_LOCKLOST_INT
HPMS_INT_M2F[13]	Reserved	Reserved	FAB_PLL_LOCK_INT
HPMS_INT_M2F[12]	Reserved	ECCINTR	Reserved
HPMS_INT_M2F[11]	Reserved	DDR_B_INTR	MDDR_IO_CALIB_INT
HPMS_INT_M2F[10]	Reserved	SW_ERRORINTERRUPT	Reserved
HPMS_INT_M2F[9]	HPD_XFR_CMP_INT	HPMSDDR_PLL_LOCK_INT	Reserved
HPMS_INT_M2F[8]	PDMAINTERRUPT	HPD_XFR_ERR_INT	Reserved
HPMS_INT_M2F[7]	Reserved	Reserved	COMM_BLK_INTR
HPMS_INT_M2F[6]	Reserved	Reserved	SOFTINTERRUPT
HPMS_INT_M2F[5]	Reserved	Reserved	Reserved
HPMS_INT_M2F[4]	Reserved	Reserved	ECCINTR
HPMS_INT_M2F[3]	Reserved	Reserved	DDR_B_INTR
HPMS_INT_M2F[2]	Reserved	Reserved	SW_ERRORINTERRUPT
HPMS_INT_M2F[1]	Reserved	ENVM_INT1	HPMSDDR_PLL_LOCK_INT
HPMS_INT_M2F[0]	SPIINT0	ENVM_INT0	HPD_XFR_ERR_INT

It is possible to overlay one interrupt signal with two interrupt sources. For example, enable a dedicated interrupt and a group 0/1 interrupt. User logic in the fabric is responsible for determining the actual source of the interrupt by reading the appropriate peripheral interrupt Status Registers and determining which interrupt has occurred. Interrupts In and Out of the FIIC are asynchronous.

All interrupts originating from HPMS blocks and fed into the FIIC are active high level sensitive signals. Once asserted, the interrupt remains asserted until the user logic clears the appropriate HPMS peripheral interrupt clear register. HPMS\_INT\_M2F interrupt signals are serviced by the FPGA fabric.

## 11.2.2 FIIC Port List

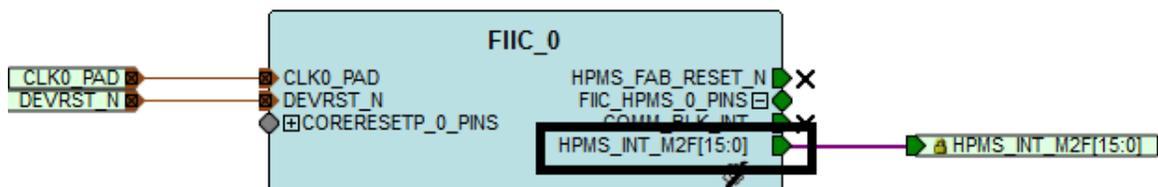
Table 188 • FIIC Port List

Port name	Direction	Polarity	Description
HPMS_INT_M2F[15:0]	Out	High	HPMS to fabric interrupts. The FIIC routes the HPMS peripheral interrupts to the fabric.
COMM_BLK_INT	Out		COMMS block interrupt.

## 11.3 How to Use FIIC

FIIC is enabled by default while creating the Libero Project using System Builder. The following figure shows a basic HPMS with the HPMS\_INT\_M2F[15:0] bus highlighted.

Figure 145 • FIIC Bus

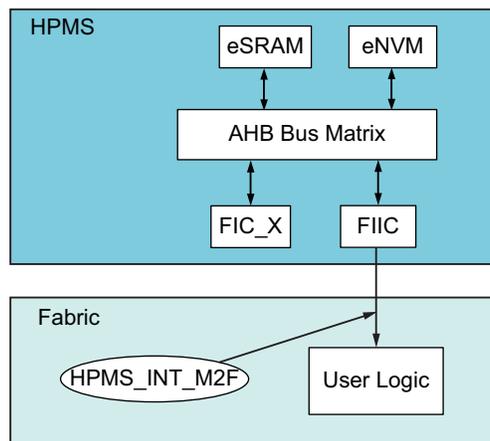


### 11.3.0.1 HPMS to the Fabric Interrupt

The following figure shows the HPMS FIIC connected to user logic. The user logic can monitor the HPMS\_INT\_M2F[15:0] signals and process according to the application requirement.

For the usage of HPMS interrupting the fabric in conjunction with PDMA, refer to [HPMS Subsystem Connected to the FPGA Fabric Master](#), page 118.

Figure 146 • HPMS to Fabric Interrupts



## 11.4 FIIC Controller Registers

The register set contains two interrupt enable registers, two interrupt Status Registers and an interrupt mode register.

The following table summarizes each of the registers covered by this chapter. The base address of the FIIC block is 0x40000.

**Table 189 • IGLOO2 FPGA FIIC Register Map**

Register Name	Address Offset	Register Type	Reset Value	Description
INTERRUPT_ENABLE0	0x00	R/W	0x0	Enables HPMS to fabric interrupts
INTERRUPT_ENABLE1	0x04	R/W	0x0	Enables HPMS to fabric interrupts
INTERRUPT_REASON0	0x08	RO	0x0	Indicates which interrupts are active
INTERRUPT_REASON1	0x0C	RO	0x0	Indicates which interrupts are active
INTERRUPT_MODE	0x10	R/W	0x0	Indicates select group 0 or select group 1

For more information on these registers, see [Table 190](#), page 243 through [Table 194](#), page 248.

## 11.5 FIIC Controller Register Bit Definitions

The following tables give the bit definitions for registers in the FIIC.

**Table 190 • INTERRUPT\_ENABLE0**

Bit Number	Name	Reset Value	Description
[31:30]	Reserved	0	
29	COMBLK_INTR_ENBL	0	COMBLK_INTR interrupt from the COMM_BLK block to fabric 1: Enable 0: Mask
28	SOFTINTERRUPT_ENBL	0	SOFTINTERRUPT interrupt from the SYSREG block to fabric 1: Enable 0: Mask
27	Reserved	0	
26	ECCINTR_ENBL	0	ECCINTR interrupt from ESRAM0, ESRAM1, CAN, and MDDR to fabric 1: Enable 0: Mask The ECCINTR interrupt is asserted when an SECEDED error has been detected in ESRAM0, ESRAM1, CAN, or MDDR memories.

**Table 190 • INTERRUPT\_ENABLE0 (continued)**

Bit Number	Name	Reset Value	Description
25	DDRB_INTR_ENBL	0	HPMS DDR bridge DDRB_INTR to fabric 1: Enable 0: Mask DDRB_INTR input indicates that either of the following interrupts are asserted from the HPMS DDR bridge. –DDRB_ERROR interrupts –DDRB_DISABLEDONE interrupts –DDRB_LOCKTIMEOUT interrupts
24	SW_ERRORINTERRUPT_ENBL	0	SW_ERRORINTERRUPT interrupt from the SYSREG block to fabric 1: Enable 0: Mask HRESP from AHB bus matrix assertion to the master in case of blocked fabric master or for the unimplemented address space results in SW_ERRORINTERRUPT signal. In case of above error condition the following signals are ORed together in SYSREG to create the SW_ERRORINTERRUPT signal. 1. HRESP assertion being issued to the HPDMA. 2. HRESP assertion being issued to FIC_0. 3. HRESP assertion being issued to FIC_1. 4. HRESP assertion being issued to the peripheral DMA engine. 5. HRESP assertion being issued to the system controller.
23	HPMSDDR_PLL_LOCK_INT_ENBL	0	HPMSDDR_PLL_LOCK_INT interrupt from the MPLL block to fabric 1: Enable 0: Mask
22	HPD_XFR_ERR_INT_ENBL	0	HPD_XFR_ERR_INT interrupt from the HPMS HPDMA block to fabric 1: Enable 0: Mask
[21:18]	Reserved	0	
17	ENVM_INT1_ENBL	0	ENVM_INT1 interrupt from HPMS ENVM1 block to fabric 1: Enable 0: Mask
16	ENVM_INT0_ENBL	0	ENVM_INT0 interrupt from the HPMS ENVM0 block to fabric 1: Enable 0: Mask
15	HPMSDDR_PLL_LOCKLOST_INT_ENBL	0	HPMSDDR_PLL_LOCKLOST_INT interrupt from MPLL to the fabric 1: Enable 0: Mask
[14:10]	Reserved	0	

**Table 190 • INTERRUPT\_ENABLE0 (continued)**

Bit Number	Name	Reset Value	Description
9	HPD_XFR_CMP_INT_ENBL	0	HPD_XFR_CMP_INT interrupt from the HPMS HPDMA block to fabric 1: Enable 0: Mask
8	PDMAINTERRUPT_ENBL	0	PDMAINTERRUPT interrupt from the HPMS peripheral DMA block to fabric 1: Enable 0: Mask
[7:1]	Reserved	0	
0	SPIINT0_ENBL	0	SPIINT0 interrupt from the HPMS SPI_0 block to fabric 1: Enable 0: Mask

**Table 191 • INTERRUPT\_ENABLE1**

Bit Number	Name	Reset Value	Description
[31:8]	Reserved	0	
7	FIC64_INT_ENBL	0	1: Enables the FIC64_INT interrupt from the DDR_FIC block. 0: Mask the FIC64_INT interrupt from the DDR_FIC block.
6	FAB_PLL_LOCKLOST_INT_ENBL	0	1: Enables the FAB_PLL_LOCKLOST_INT interrupt from FAB_PLL. 0: Mask the FAB_PLL_LOCKLOST_INT interrupt from FAB_PLL.
5	FAB_PLL_LOCK_INT_ENBL	0	1: Enables the FAB_PLL_LOCK_INT interrupt from FAB_PLL. 0: Masks the FAB_PLL_LOCK_INT interrupt from FAB_PLL.
4	Reserved	0	
3	MDDR_IO_CALIB_INT_ENBL	0	1: Enables the MDDR_IO_CALIB_INT interrupt from the MDDR block to fabric. 0: Masks the MDDR_IO_CALIB_INT interrupt from the MDDR block to fabric.
[2:0]	Reserved	0	

**Table 192 • INTERRUPT\_REASON1**

Bit Number	Name	Reset Value	Description
[31:8]	Reserved	0	
7	FIC64_INT_STATUS	0	Set if the interrupt source for FIC64_INT is asserted and the FIC64_INT_ENBL interrupt enable bit in INTERRUPT_ENABLE1 is High.

**Table 192 • INTERRUPT\_REASON1 (continued)**

Bit Number	Name	Reset Value	Description
6	FAB_PLL_LOCKLOST_INT_STATUS	0	Set if the interrupt source for FAB_PLL_LOCKLOST_INT is asserted and the FAB_PLL_LOCKLOST_INT_ENBL interrupt enable bit in INTERRUPT_ENABLE1 is High.
5	FAB_PLL_LOCK_INT_STATUS	0	Set if the interrupt source for FAB_PLL_LOCK_INT is asserted and the FAB_PLL_LOCK_INT_ENBL interrupt enable bit in INTERRUPT_ENABLE1 is High.
4	Reserved	0	
3	MDDR_IO_CALIB_INT_STATUS	0	Set if the interrupt source for MDDR_IO_CALIB_INT is asserted and the MDDR_IO_CALIB_INT_ENBL interrupt enable bit in INTERRUPT_ENABLE1 is High.
[2:0]	Reserved	0	

**Table 193 • INTERRUPT\_REASON0**

Bit Number	Name	Reset Value	Description
[31:30]	Reserved	0	
29	COMBLK_INTR_STATUS	0	Set if the interrupt source for COMBLK_INTR is asserted and the COMBLK_INTR_ENBL interrupt enable bit in INTERRUPT_ENABLE0 is High.
28	SOFTINTERRUPT_STATUS	0	Set if the interrupt source for SOFTINTERRUPT is asserted and the SOFTINTERRUPT_ENBL interrupt enable bit in INTERRUPT_ENABLE0 is High.
27	Reserved	0	
26	ECCINTR_STATUS	0	Set if the interrupt source for ECCINTR is asserted and the ECCINTR_ENBL interrupt enable bit in INTERRUPT_ENABLE0 is High.
25	DDRB_INTR_STATUS	0	Set if the interrupt source for DDRB_INTR is asserted and the DDRB_INTR_ENBL interrupt enable bit in INTERRUPT_ENABLE0 is High.
24	SW_ERRORINTERRUPT_STATUS	0	Set if the interrupt source for SW_ERRORINTERRUPT is asserted and the SW_ERRORINTERRUPT_ENBL interrupt enable bit in INTERRUPT_ENABLE0 is High.

**Table 193 • INTERRUPT\_REASON0 (continued)**

Bit Number	Name	Reset Value	Description
23	HPMSDDR_PLL_LOCK_INT_STATUS	0	Set if the interrupt source for HPMSDDR_PLL_LOCK_INT is asserted and the HPMSDDR_PLL_LOCK_INT_ENBL interrupt enable bit in INTERRUPT_ENABLE0 is High. HPMSDDR_PLL_LOCK_INT interrupt is asserted when MPLL achieves lock.
22	HPD_XFR_ERR_INT_STATUS	0	Set if the interrupt source for HPD_XFR_ERR_INT is asserted and the HPD_XFR_ERR_INT_ENBL interrupt enable bit in INTERRUPT_ENABLE0 is High.
[21:18]	Reserved	0	
17	ENVM_INT1_STATUS	0	Set if the interrupt source for ENVM_INT1 is asserted and the ENVM_INT1_ENBL interrupt enable bit in INTERRUPT_ENABLE0 is High.
16	ENVM_INT0_STATUS	0	Set if the interrupt source for ENVM_INT0 is asserted and the ENVM_INT0_ENBL interrupt enable bit in INTERRUPT_ENABLE0 is High.
15	HPMSDDR_PLL_LOCKLOST_INT_STATUS	0	Set if the interrupt source for HPMSDDR_PLL_LOCKLOST_INT is asserted and the HPMSDDR_PLL_LOCKLOST_INT_ENBL interrupt enable bit in INTERRUPT_ENABLE0 is High.
[14:10]	Reserved	0	
9	HPD_XFR_CMP_INT_STATUS	0	Set if the interrupt source for HPD_XFR_CMP_INT is asserted and the HPD_XFR_CMP_INT_ENBL interrupt enable bit in INTERRUPT_ENABLE0 is High.
8	PDMAINTERRUPT_STATUS	0	Set if the interrupt source for PDMAINTERRUPT is asserted and the PDMAINTERRUPT_ENBL interrupt enable bit in INTERRUPT_ENABLE0 is High.
[7:1]	Reserved	0	
0	SPIINT0_STATUS	0	Set if the interrupt source for SPIINT0 is asserted and the SPIINT0_ENBL interrupt enable bit in INTERRUPT_ENABLE0 is High.

**Table 194 • INTERRUPT\_MODE**

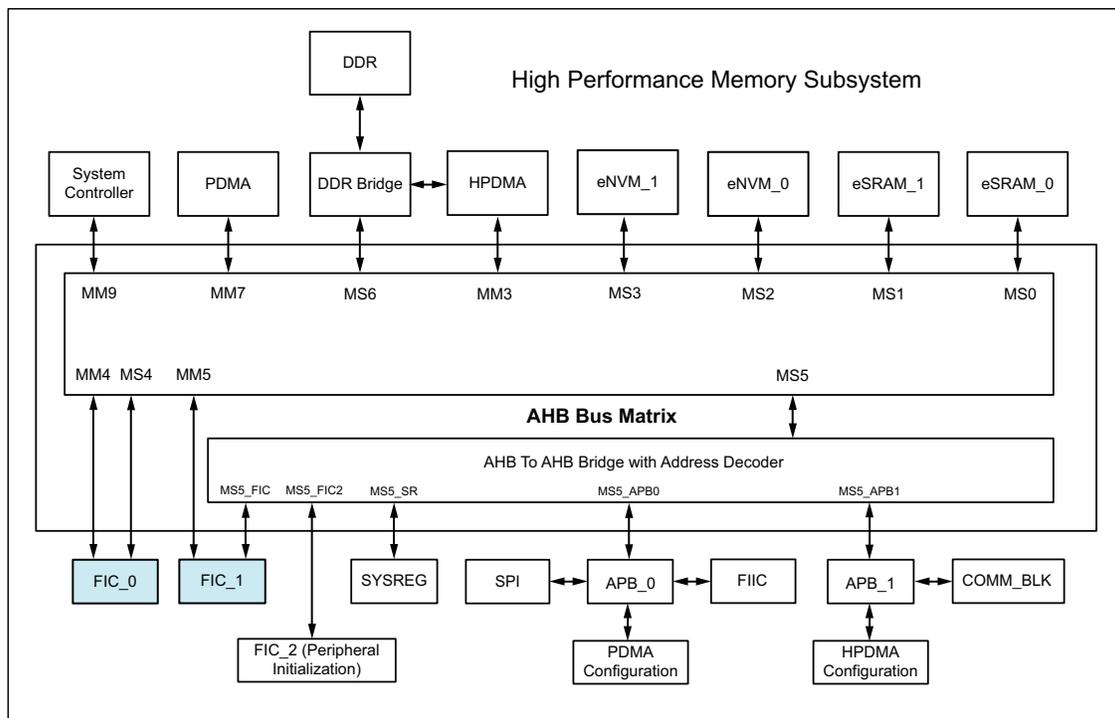
Bit Number	Name	Reset Value	Description
[31:1]	Reserved	0	
0	SELECT_MODE	0	The following are the valid values for this bit: 0: Select group 0 1: Select group 1

## 12 Fabric Interface Controller

The HPMS fabric interface controller (FIC) performs a bridging function for AHB-Lite to APB3 or APB3 to AHB-Lite between the AHB bus matrix and the FPGA fabric. There are up to two 32-bit FIC blocks in IGLOO2 devices, referred to as FIC\_0 and FIC\_1. Each FIC block provides a master interface (AHB-Lite or APB3) and a slave interface (AHB-Lite or APB3). But both master and slave interfaces need to operate in the same bus protocol, either AHB-Lite or APB3. However, it is possible to have master and slave at the same time. Each FIC block can operate on a different clock frequency, defined as a ratio of the HPMS main clock, HPMS\_CLK.

The IGLOO2 architecture imposes a certain number of rules related to clocking domains between the fabric interfaces and the FPGA fabric. This document provides guidance on how to properly construct such systems. The following figure depicts the connectivity of FIC\_0 and FIC\_1 to the AHB bus matrix.

**Figure 147 • The FIC Connection to the AHB Bus Matrix**



The following table lists the number of FICs available for use in each device.

**Table 195 • Number of FICs Available for Use in Each Device**

Device	FIC Blocks
M2GL005	1 <sup>1</sup>
M2GL010	
M2GL025	
M2GL050	2
M2GL060	1 <sup>1</sup>
M2GL090	
M2GL150	2

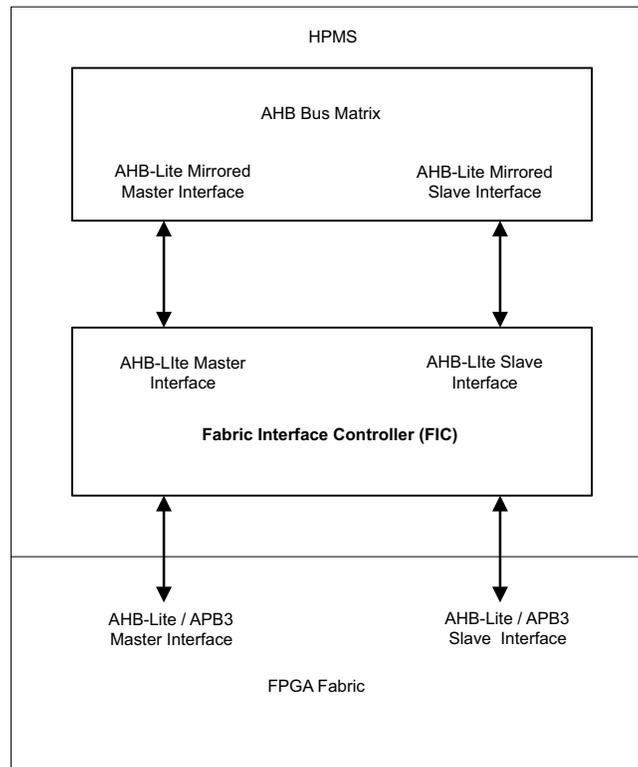
1. Only FIC\_0 is available.

## 12.1 Functional Description

This section provides a detailed description of the FIC subsystem.

The following figure shows a block diagram for the FIC. The FIC is a hard block; enabling or disabling it will not use any user logic.

**Figure 148 • Fabric Interface Controller Block Diagram**



### 12.1.1 Configuring FIC for Master or Slave Interface

FIC\_0 and FIC\_1 can be configured independently from each other by using System Builder in the Libero SoC Software. There are two options:

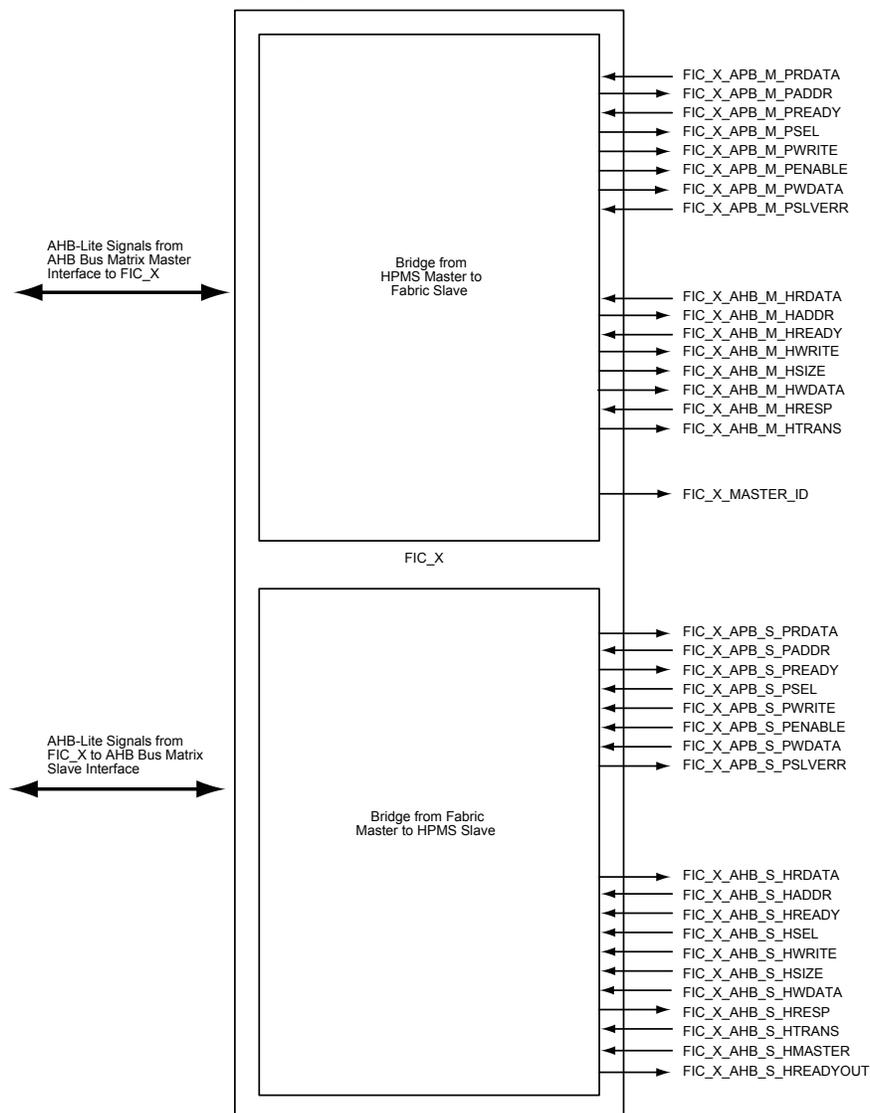
- The HPMS is the master and the fabric has the slave (HM - hard master)
- The fabric has the master and the HPMS is the slave (FM - fabric master)

Since FIC\_0 and FIC\_1 have an AMBA interface, user logic should implement the AMBA AHB-Lite or APB3 protocol in order to communicate with the FIC. Microsemi provides numerous AHB and APB v 3.0 compliant cores in the Libero SoC IP catalog for easy instantiation into the FPGA fabric. Instantiate CoreAHBLite and CoreAPB3 soft IP into the fabric to allow further instantiation of soft AHB-Lite and APB3 masters and slaves.

## 12.2 FIC Interface Port List

The following figure shows FIC top-level view. The bridge from the HPMS master to the fabric slave translates the AHB-Lite to APB3; and the bridge from the fabric master to the HPMS slave translates APB3 to AHB-Lite.

**Figure 149 • Fabric Interface Controller Top-Level View**



The following table lists the FIC ports.

**Table 196 • Fabric Interface Controller Port List**

Port Name	Direction	Description
FIC_X_MASTER_ID [1:0]	Out	Current master performing the transfer.
FIC_X_APB_S_PRDATA [31:0]	Out	APB3 read data to the fabric master.
FIC_X_APB_S_PADDR [31:0]	In	APB3 address initiated by the fabric master.
FIC_X_APB_S_PREADY	Out	APB3 ready signal to the fabric master.
FIC_X_APB_S_PSEL	In	APB3 slave select signal from the fabric master.

**Table 196 • Fabric Interface Controller Port List (continued)**

Port Name	Direction	Description
FIC_X_APB_S_PWRITE	In	APB3 write control signal from the fabric master.
FIC_X_APB_S_PENABLE	In	APB3 enable from the fabric master.
FIC_X_APB_S_PWDATA [31:0]	In	APB3 write data from the fabric master.
FIC_X_APB_S_PSLVERR	Out	Error condition on an APB3 transfer to the fabric master.
FIC_X_APB_M_PRDATA [31:0]	In	APB3 read data from the fabric slave.
FIC_X_APB_M_PADDR [31:0]	Out	APB3 address to the fabric slave.
FIC_X_APB_M_PREADY	In	APB3 ready signal from the fabric slave.
FIC_X_APB_M_PSEL	Out	APB3 slave select signal to the fabric slaves.
FIC_X_APB_M_PWRITE	Out	APB3 write control signal to the fabric slaves.
FIC_X_APB_M_PENABLE	Out	APB3 enable to the fabric slave.
FIC_X_APB_M_PWDATA [31:0]	Out	APB3 write data to the fabric slave.
FIC_X_APB_M_PSLVERR	In	Error condition on an APB3 transfer from the fabric slave.
FIC_X_AHB_S_HRDATA [31:0]	Out	AHB read data to the fabric master.
FIC_X_AHB_S_HADDR [31:0]	In	AHB address initiated by the fabric master.
FIC_X_AHB_S_HREADY	In	Transfer has completed on the bus. The fabric master can drive this signal Low to extend a transfer.
FIC_X_AHB_S_HWDATA [31:0]	In	AHB write data from the fabric master.
FIC_X_AHB_S_HWRITE	In	AHB write control signal from the fabric master.
FIC_X_AHB_S_HRESP	Out	AHB transfer response to the fabric master.
FIC_X_AHB_S_HSIZE [1:0]	In	AHB transfer size from the fabric master.
FIC_X_AHB_S_HTRANS [1:0]	In	AHB transfer type from the fabric master.
FIC_X_AHB_S_HMASTLOCK	In	AHB master lock signal from the fabric master.
FIC_X_AHB_S_HSEL	In	AHB slave select signal from the fabric master.
FIC_X_AHB_S_HREADYOUT	Out	Transfer has completed on the bus. The signal is asserted Low to extend a transfer. Input to the fabric master.
FIC_X_AHB_M_HWRITE	Out	AHB write control signal to the fabric slave.
FIC_X_AHB_M_HADDR [31:0]	Out	AHB address to the fabric slave.
FIC_X_AHB_M_HREADY	In	Transfer has completed on the bus. The fabric slave can drive this signal Low to extend a transfer.
FIC_X_AHB_M_HWDATA [31:0]	Out	AHB-Lite write data to the fabric slave.
FIC_X_AHB_M_HRDATA [31:0]	In	AHB-Lite read data from the fabric slave.
FIC_X_AHB_M_HRESP	In	AHB-Lite transfer response from the fabric slave.

**Table 196 • Fabric Interface Controller Port List (continued)**

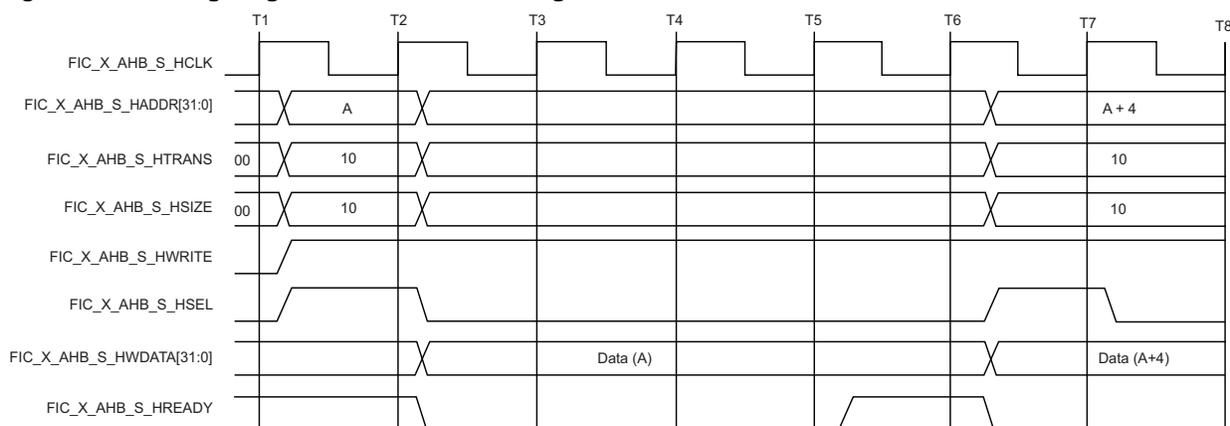
Port Name	Direction	Description
FIC_X_AHB_M_HSIZE [1:0]	Out	AHB-Lite transfer size to the fabric slave.
FIC_X_AHB_M_HTRANS [1:0]	Out	AHB-Lite transfer type to the fabric slave.

## 12.3 Timing Diagrams

The following timing diagrams illustrate AHB-Lite non-sequential transfers with 32 bits as the transfer size.

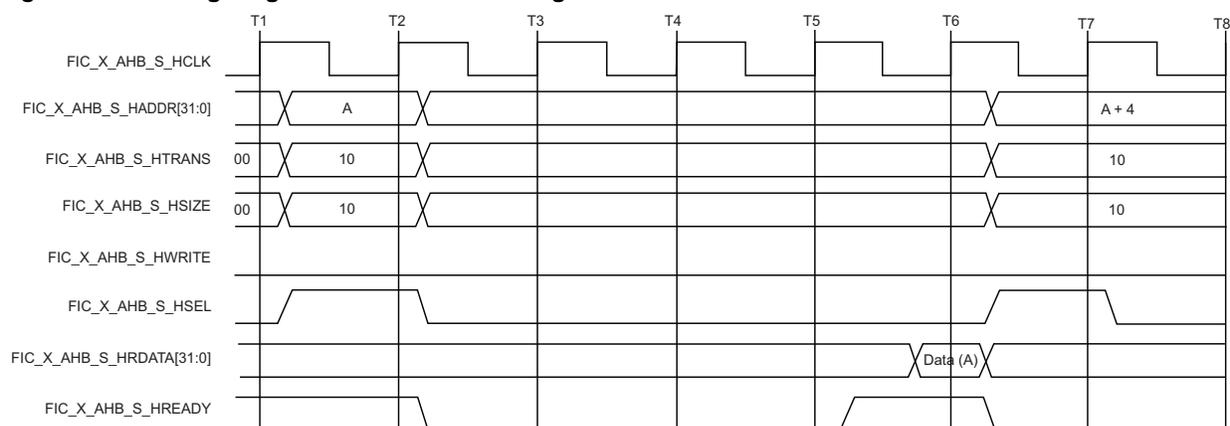
The following figure illustrates the AHB-Lite bus signals from the fabric master to the fabric interface controller for write transactions. Generation of pipelined requests depends on the efficiency of the master in the fabric to generate it.

**Figure 150 • Timing Diagram for AHB-Lite Bus Signals from Fabric Master to FIC for a Write Transaction**



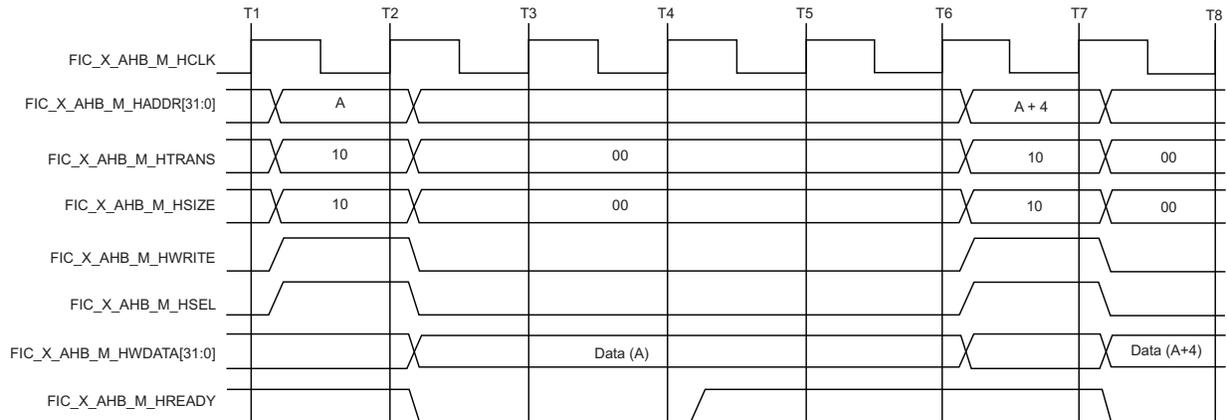
The following figure illustrates the AHB-Lite bus signals from the fabric master to the fabric interface controller for read transactions. Generation of pipelined requests depends on the efficiency of the master in the fabric to generate it.

**Figure 151 • Timing Diagram for AHB-Lite Bus Signals from Fabric Master to FIC for a Read Transaction**



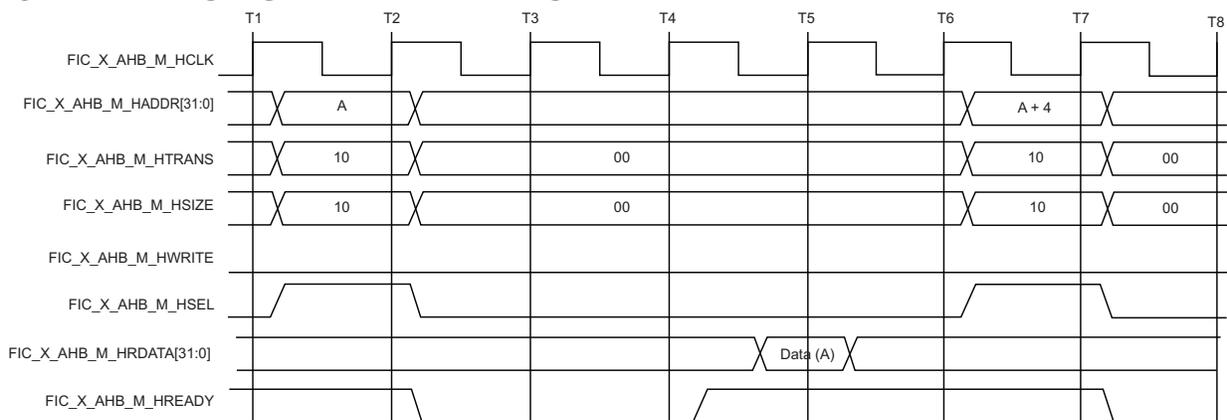
The following figure illustrates the AHB-Lite bus signals from the fabric interface controller to the fabric slave for write transaction.

**Figure 152 • Timing Diagram for AHB-Lite Bus Signals from FIC to the Fabric Slave for a Write Transaction**



The following figure illustrates the AHB-Lite bus signals from the fabric interface controller to the fabric slave for read transaction.

**Figure 153 • Timing Diagram for AHB-Lite Bus Signals from FIC to the Fabric Slave for a Read Transaction**

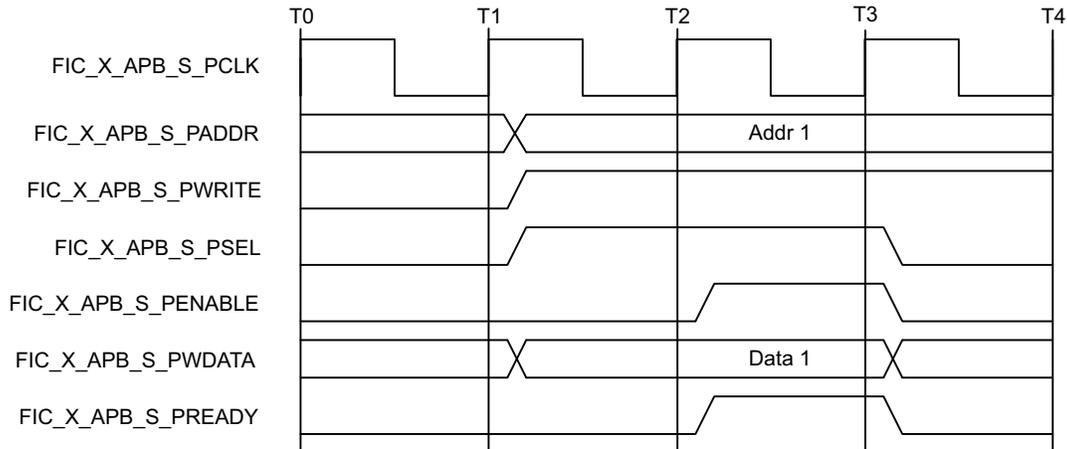


The following timing diagrams illustrate the APB3 write and read transfers.

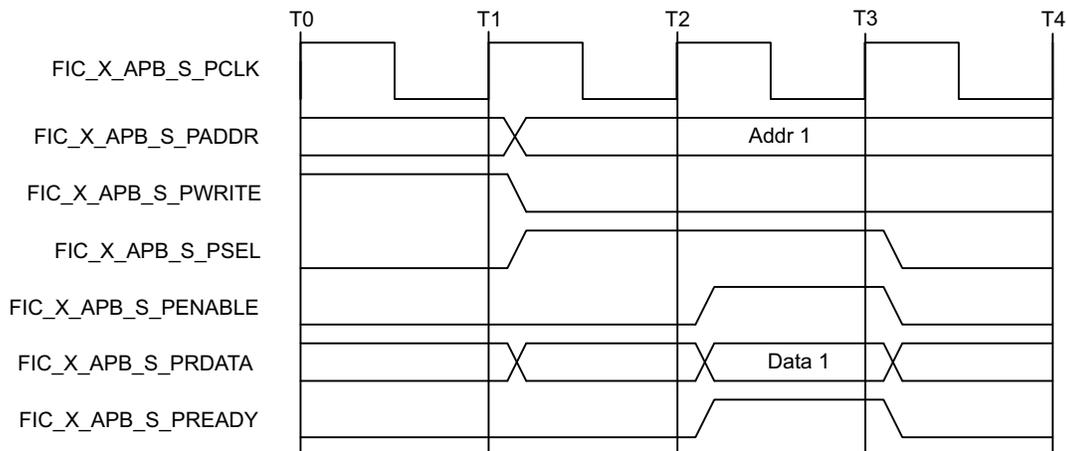
**Figure 154**, page 255 and **Figure 156**, page 256 show an example of basic write transfer with no wait states. The APB3 write transfer starts with the address, PADDR (FIC\_X\_APB\_M\_PADDR/FIC\_X\_APB\_S\_PADDR), write data, PWDATA (FIC\_X\_APB\_M\_PWDATA/FIC\_X\_APB\_S\_PWDATA), write signal, PWRITE (FIC\_X\_APB\_M\_PWRITE/FIC\_X\_APB\_S\_PWRITE), and select signal PSEL (FIC\_X\_APB\_M\_PSEL/FIC\_X\_APB\_S\_PSEL) - all changing after the rising edge of the PCLK (FIC\_X\_APB\_M\_PCLK/FIC\_X\_APB\_S\_PCLK). In the next clock edge, the enable signal is asserted.

PENABLE (FIC\_X\_APB\_M\_PENABLE/FIC\_X\_APB\_S\_PENABLE) indicates that the Access phase is taking place. The address, data, and control signals all remain valid throughout this Access phase. The transfer completes at the end of this cycle and PENABLE is deasserted at the end of the transfer. PSEL also goes Low unless the transfer is to be followed immediately by another transfer to the same peripheral. During an Access phase, when PENABLE is High, the transfer can be extended by driving PREADY (FIC\_X\_APB\_M\_PREADY/FIC\_X\_APB\_S\_PREADY) Low.

**Figure 154 • Timing Diagram for APB3 Bus Signals from Fabric Master to FIC for a Write Transaction**

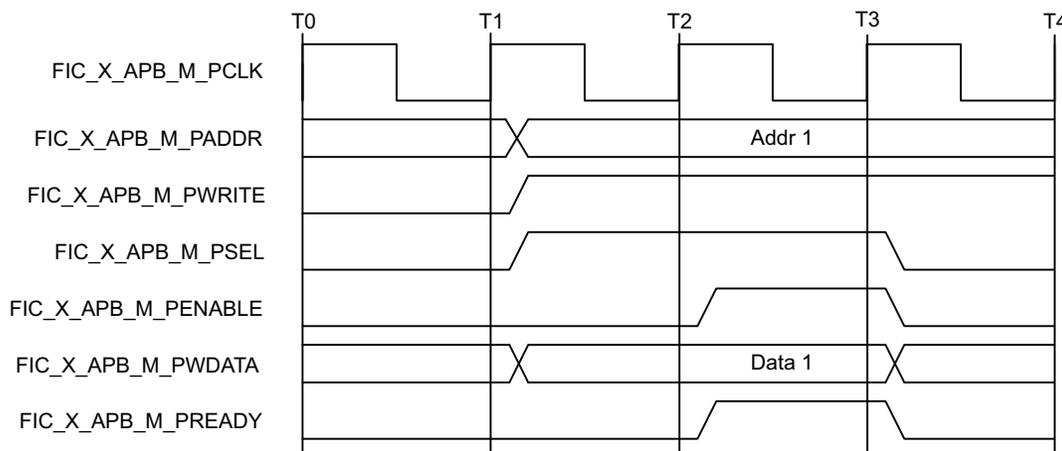


**Figure 155 • Timing Diagram for APB3 Bus Signals from Fabric Master to FIC for a Read Transaction**

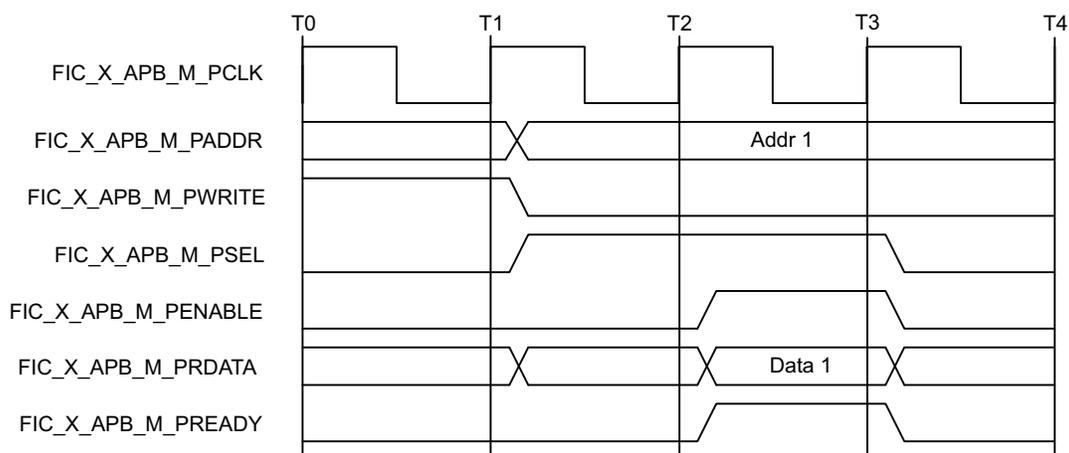


During a read transfer, the timing of PADDR, PWRITE, PSEL, and PENABLE signals are as described in Write transfers. The slave must provide the data before the end of the read transfer. The transfer is extended if PREADY is driven Low during an Access phase. [Figure 155](#), page 255 and [Figure 157](#), page 256 show an example of basic read transfer with no wait states.

**Figure 156 • Timing Diagram for APB3 Bus Signals from FIC to the Fabric Slave for a Write Transaction**



**Figure 157 • Timing Diagram for APB3 Bus Signals from FIC to the Fabric Slave for a Read Transaction**



## 12.4 Implementation Considerations

In AHB mode, the user may perform byte, half word, and word accesses from the fabric to HPMS. However, in APB16 mode, the user can only cause a word access to occur to an HPMS slave. This is done by two accesses over the APB16, one of which is to write a 16-bit holding register (in the case of writes) or to read a 16-bit holding register in the case of reads.

## 12.5 Fabric Interface Clocks

The HPMS clocks should be aligned with fabric clock to establish the synchronous communication between the HPMS peripheral and the user logic. The fabric alignment clock controller (FACC) within the HPMS CCC generates the various aligned clocks required by the HPMS sub-blocks, and controlling the alignment of the FPGA fabric interface clocks. The following rules of the IGLOO2 architecture must be followed for synchronous communication between the HPMS and FPGA fabric FIC subsystems.

- The HPMS and FPGA fabric FIC clocks must have matching frequencies for each FIC subsystem.
- The FPGA fabric FIC subsystem clock with the smallest frequency must drive the HPMS CLK\_BASE.
- Align all the FPGA fabric FIC subsystem clocks precisely; the clocks could be of different frequencies, but align the rising-edges of the slower clocks to the rising-edges of the faster clocks.

Refer to the [UG0449: SmartFusion2 SoC FPGA and IGLOO2 FPGA Clocking Resources User Guide](#) for more details on the alignment of fabric clocks and derived clocks in the HPMS.

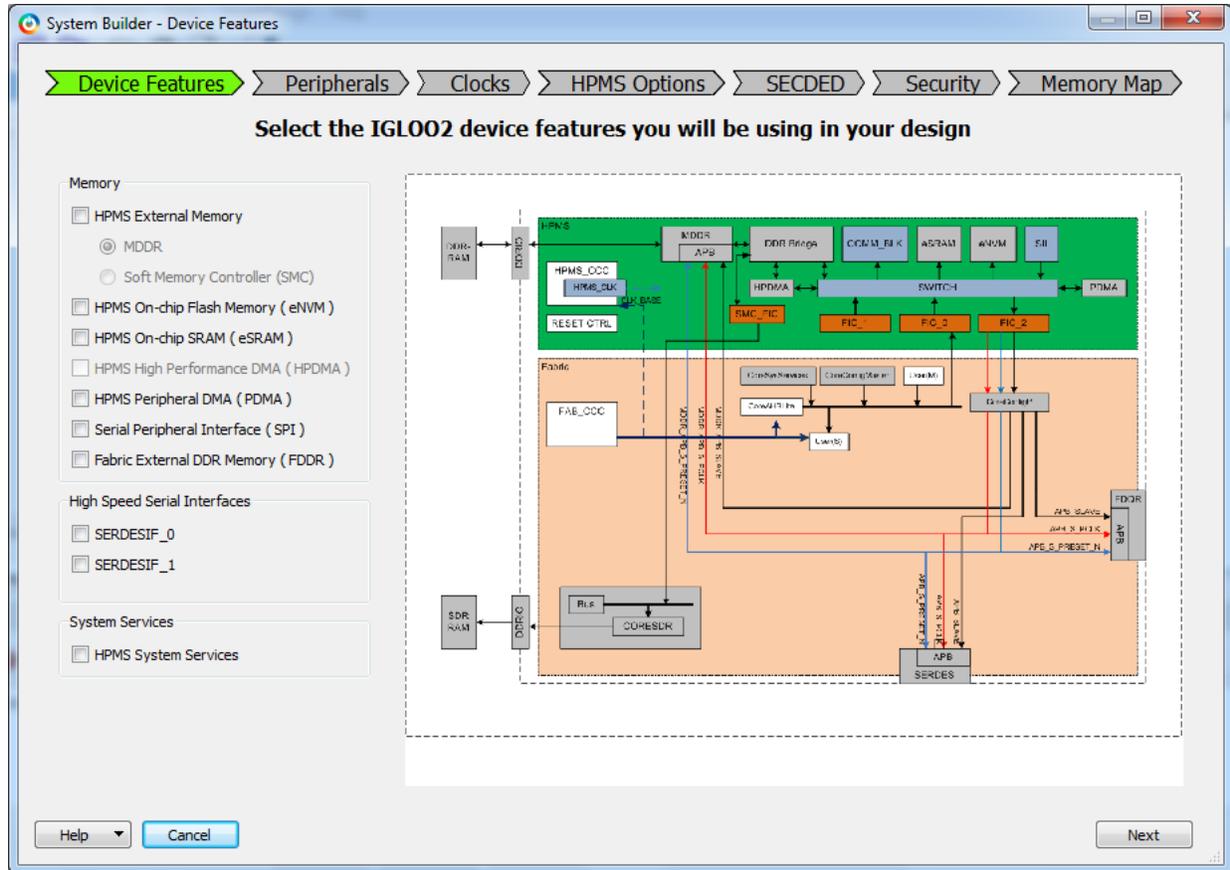
## 12.6 How to Use FIC

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

### 12.6.1 FIC\_1 Configuration

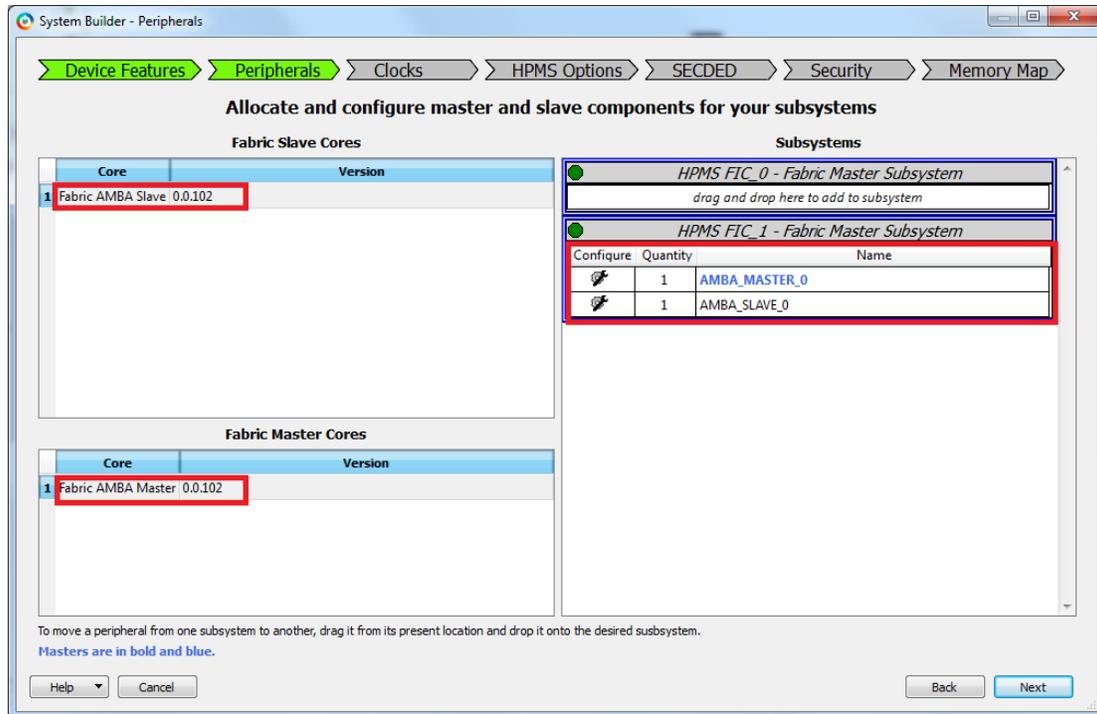
The following figure shows the initial **System Builder** window where the required device features can be selected. For details on how to launch the **System Builder** wizard and detailed information on how to use it, refer the *IGLOO2 System Builder User Guide*.

**Figure 158 • System Builder Window**



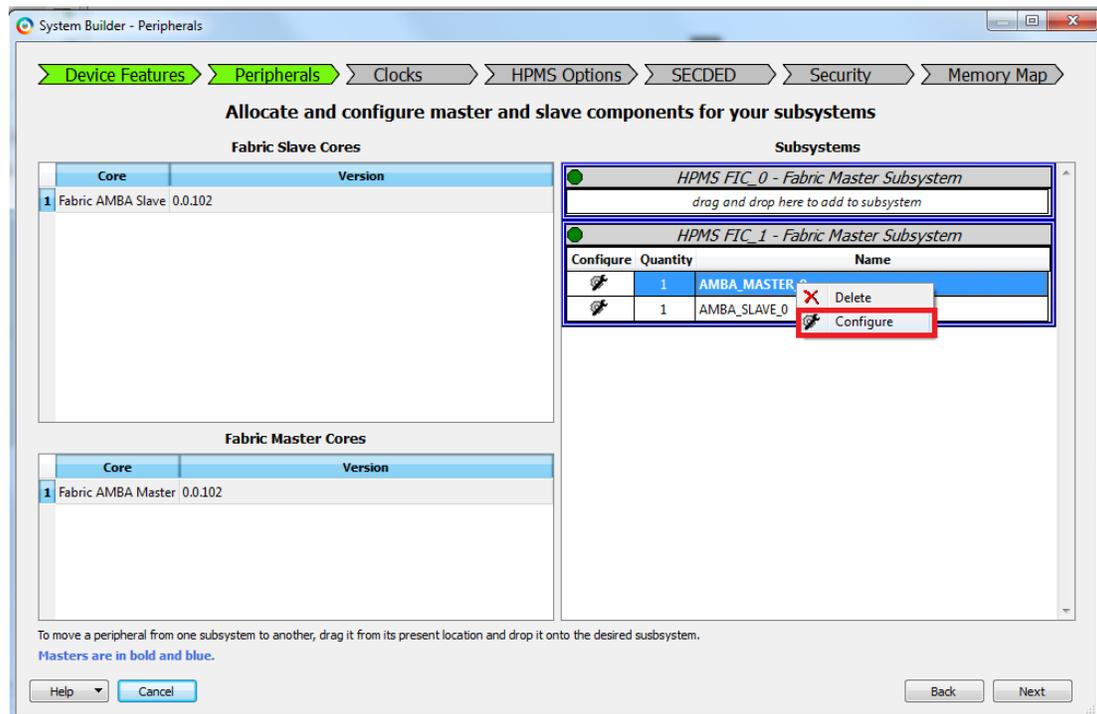
1. Click **Next** in the **Device Features** tab to navigate to the **Peripherals** tab.
2. In the **Peripherals** tab, drag and drop the **Fabric AMBA Slave** and **Fabric AMBA Master** on to **HPMS FIC\_1 - Fabric Master Subsystem**. The following figure shows the Peripherals tab with the master and slave components configured in the **HPMS FIC\_1 - Fabric Master Subsystem**.

**Figure 159 • System Builder- Peripherals Tab**



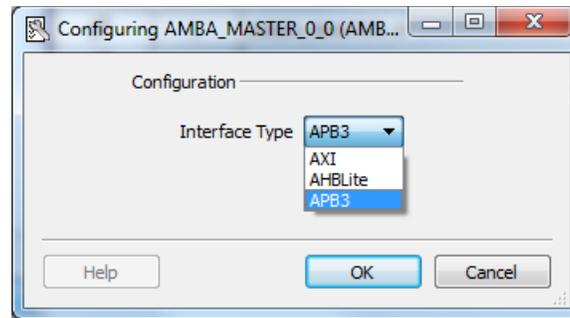
3. To select the interface type for the fabric master, right-click on the master and select **Configure**. The following figure shows the Configure option.

**Figure 160 • Peripherals Tab - Configure Option for AMBA\_MASTER\_0**



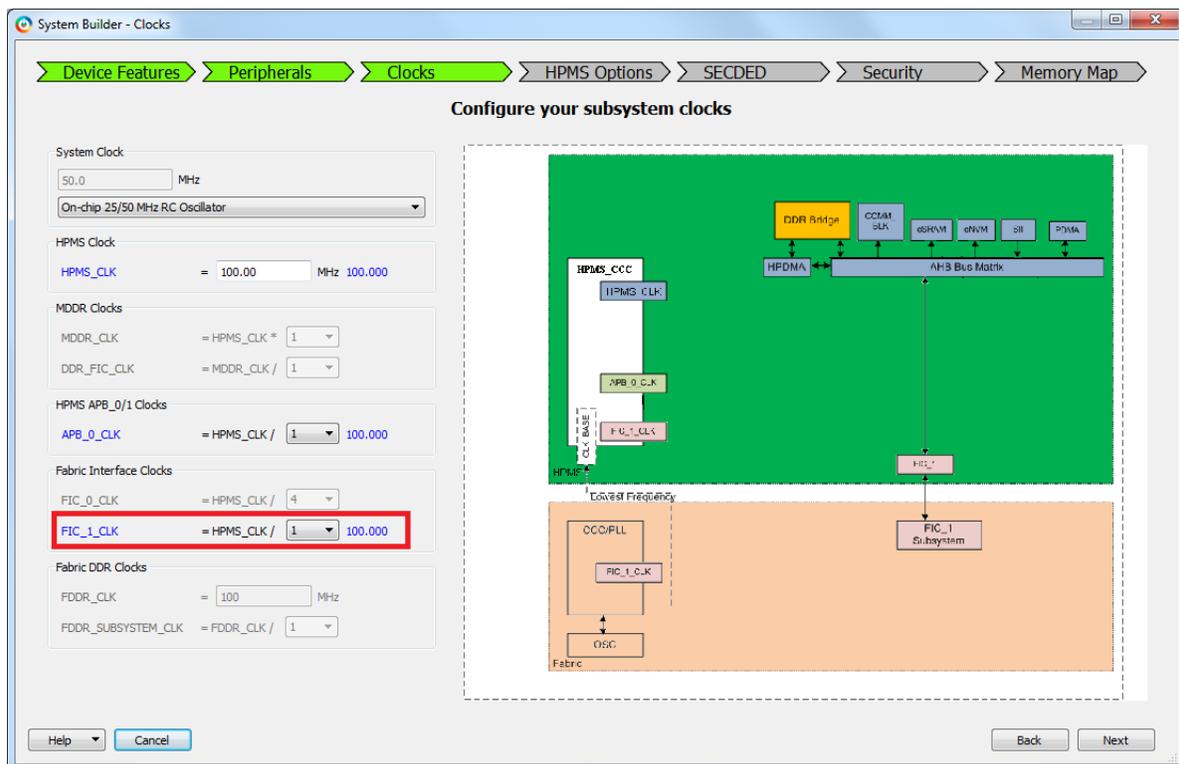
- From the Configuring AMBA Master dialog, select the **Interface Type** according to your requirement, as shown in the following figure.  
Repeat step 3 and step 4 to configure fabric slave interface type.

**Figure 161 • Interface Type Configuration**



- Click **Next** to navigate to the **Clocks** tab.
- Select the ratio of the FIC\_1\_CLK against the HPMS\_CLK. The following figure shows the Clocks tab with the FIC\_1\_CLK configuration option highlighted.

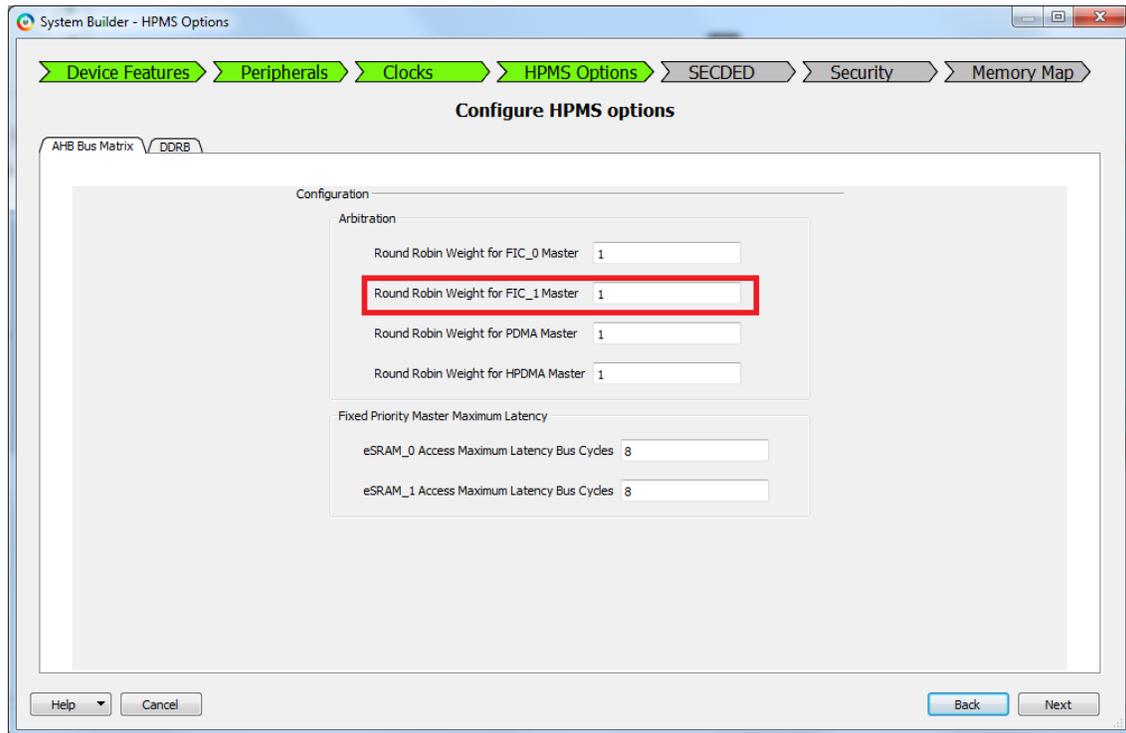
**Figure 162 • Clocks Tab - Fabric Interface Clocks**



- Click **Next** to navigate to the **HPMS Options** tab.

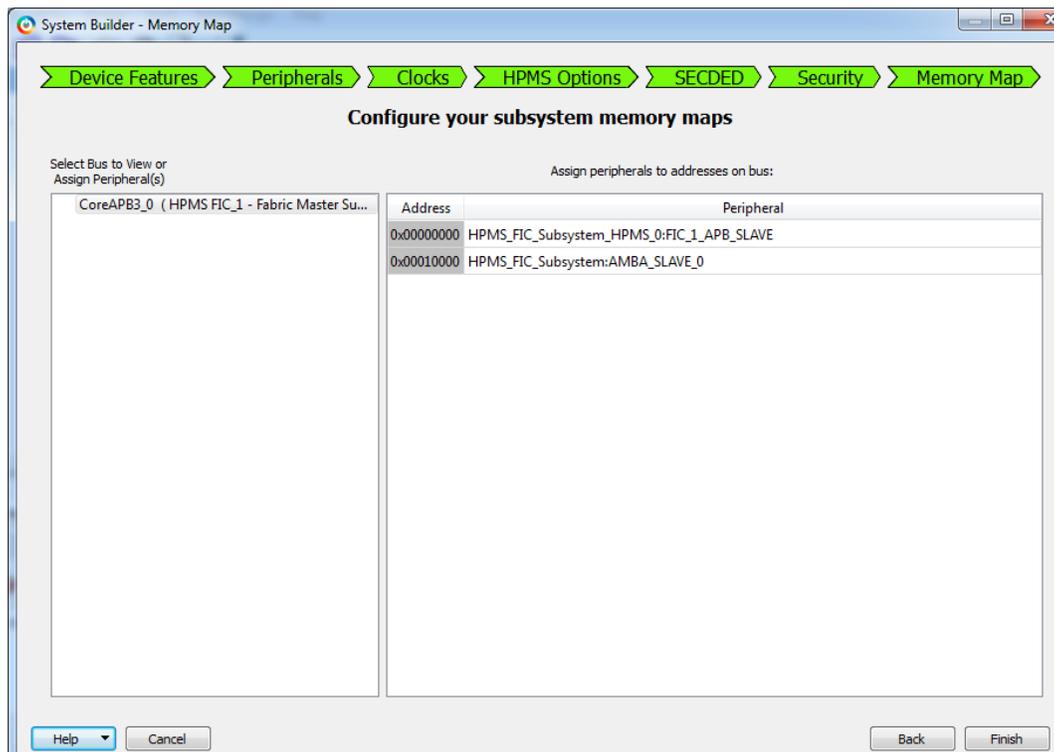
- Enter the weight value for the **Round Robin Weight for FIC\_1 Master** as shown in the following figure.

**Figure 163 • HPMS Options Tab with Round Robin Weight for FIC\_1 Master**



- Navigate to the **Memory Map** tab giving the required data in the rest of the **System Builder** tabs.
- Click **Finish** to create the subsystem as shown in the following figure.

**Figure 164 • Memory Map Tab**



## 12.6.2 FIC\_0 Configuration

The System Builder configures FIC\_0 as master interface based on the following user selection:

- If one of the eNVM, eSRAM, PDMA, or HPDMA is selected, the System Builder configures FIC\_0 as AHB-Lite slave. A Fabric AHB-Lite master can be connected to do the data transfer.
- If HPMS System Services is selected, the system builder configures FIC\_0 as AHB-Lite slave. But it is meant to connect the CoreSysServices IP only.

If the user has enabled PDMA or HPDMA and wants to transfer data to/from the HPMS DDR/eNVM /eSRAM to/from Fabric AMBA slaves, drag-and-drop a fabric AMBA slave core from the available cores panel to FIC\_0 - HPMS master subsystem. The System Builder then configures FIC\_0 slave interface as well.

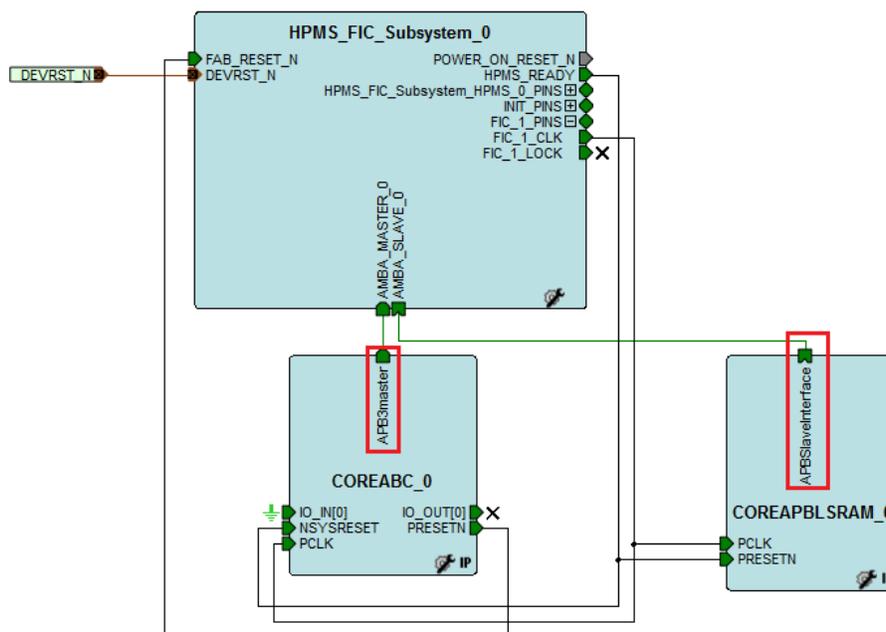
For more information on configuring Peripherals refer to “IGLOO2 Design Subsystems” section of *IGLOO2 System Builder User Guide*.

For more information on configuring FIC interface signals refer to “Generating Your System” section of *IGLOO2 System Builder User Guide*.

## 12.6.3 Use Model 1: Connecting APB3 Master and Slave to FIC\_1

1. Navigate to the **Peripherals** tab in the **System Builder** wizard. Add AMBA master and slave to FIC\_1 subsystem.
2. Select the interface type as APB3 for FIC\_1 master and slave.
3. Proceed with the rest of the configurations in the **System Builder** wizard and create the subsystem.
4. Instantiate the user APB3 master logic and slave logic in the SmartDesign canvas and connect them to FIC\_1 master and slave interfaces as shown in the following figure.

**Figure 165 • Top-Level Smart Design View for Use Model 1**

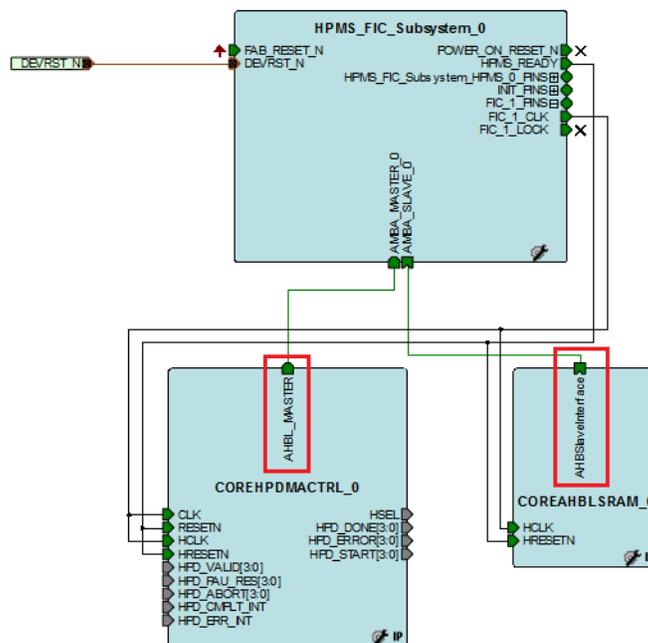


## 12.6.4 Use Model 2: Connecting AHB-Lite Master and Slave to FIC\_1

1. Navigate to the **Peripherals** tab in the **System Builder** wizard. Add AMBA master and slave to FIC\_1 subsystem.
2. Select the interface type as AHB-Lite for FIC\_1 master and slave.
3. Proceed with the rest of the configurations in the **System Builder** wizard and create the subsystem.

4. Instantiate user AHB-Lite master logic and slave logic in the Smart Design canvas and connect them to FIC\_1 master and slave interfaces as shown in the following figure.

Figure 166 • Top-Level Smart Design View for Use Model 2



**Note:** The HPMS FIC supports full behavioral simulation models.

## 12.7 SYSREG Control Registers for FIC\_0 and FIC\_1

Refer to the [System Register Block](#), page 196 for a detailed description of each register and bit. The following table gives the Control Registers for FIC\_0 and FIC\_1 from the SYSREG block.

Table 197 • FAB\_IF Register in the SYSREG Block

Register name	Register Type	Flash Write Protect	Reset Source	Description
FAB_IF_CR	RW-P	Register	SYSRESET_N	Control Register for fabric interface.
SOFT_RESET_CR	RW-P	Bit	SYSRESET_N	Generates software control interrupts to the HPMS peripherals.
HPMSDDR_FACC1_CR	RW-P	Field	CC_SYSRESET_N	HPMS DDR fabric alignment clock controller 1 Configuration Register
HPMSDDR_CLK_CALIB_STATUS	RO		SYSRESET_N	HPMS DDR clock calibration Status Register

## 12.8 Reference Documents

AMBA 3 AHB-Lite Protocol Specification:  
<http://infocenter.arm.com/help/topic/com.arm.doc.ih0033a/index.html>

AMBA 3 APB Protocol Specification:  
<http://infocenter.arm.com/help/topic/com.arm.doc.ih0024b/index.html>



## 13.1.1 Architecture Overview

The preceding figure shows the APB configuration interfaces and SerDes and DDR subsystems connections with the HPMS master. The FIC\_2 port in the AHB bus matrix routes the APB configuration interface to the FPGA fabric. The SerDes and DDR subsystems are connected through CoreConfig IP. CoreConfig IP must be instantiated in the FPGA fabric to allow configuration of FDDR, SERDESIF, and MDDR.

The following tables (Table 198, page 264 through Table 201, page 265) lists the APB configuration interface signals and descriptions.

The APB configuration space is divided into multiple partitions; each partition is reserved to a specific module or type of functionality. The APB addresses are word aligned.

The base address of FDDR, SERDESIF0, and SERDESIF1 configuration address space resides at 0x40020400 and extends to address 0x4002FFFF in the memory map.

- Refer to the “Fabric DDR Subsystem” chapter in the *UG0446: SmartFusion2 SoC FPGA and IGLOO2 FPGA High Speed DDR Interfaces User Guide* for FDDR register map details and address space partition.
- Refer to the “Serializer/Deserializer” chapter of the *UG0447: IGLOO2 FPGA and SmartFusion2 SoC FPGA High Speed Serial Interfaces User Guide* for SerDes register map details and address space partition.

The base address of the MDDR configuration address space resides at 0x40020000 and extends to address 0x400203FF in the memory map.

- Refer to the “MDDR Subsystem” chapter of the *UG0446: SmartFusion2 SoC FPGA and IGLOO2 FPGA High Speed DDR Interfaces User Guide* for MDDR register map details and address space partition.

## 13.1.2 Port List

**Table 198 • FDDR APB Slave Configuration Interface Port List**

Port Name	Direction	Polarity	Description
APB_S_PSEL	In	High	Indicates APB slave select
APB_S_PENABLE	In	High	Indicates APB enable
APB_S_PWRITE	In	High	APB write control signal. Indicates read when Low and write when High.
APB_S_PADDR [10:2]	In		Indicates APB address. Addresses are word aligned.
APB_S_PWDATA [15:0]	In		Indicates APB write data
APB_S_PRDATA [15:0]	Out		Indicates APB read data
APB_S_PREADY	Out		Indicates APB PREADY signal and is used to extend an APB transfer.
APB_S_PSLVERR	Out	High	Indicates a transfer failure
APB_S_PCLK	In		Indicates APB clock
APB_S_PRESET_N	In	Low	Indicates APB active low reset

**Table 199 • MDDR APB Slave Configuration Interface Port List**

Port Name	Direction	Polarity	Description
MDDR_APB_S_PSEL	In	High	Indicates APB slave select
MDDR_APB_S_PENABLE	In	High	Indicates APB enable
MDDR_APB_S_PWRITE	In	High	APB write control signal. Indicates read when Low and write when High.

**Table 199 • MDDR APB Slave Configuration Interface Port List (continued)**

Port Name	Direction	Polarity	Description
MDDR_APB_S_PADDR [10:2]	In		Indicates APB address. Addresses are word aligned.
MDDR_APB_S_PWDATA [15:0]	In		Indicates APB write data
MDDR_APB_S_PRDATA [15:0]	Out		Indicates APB read data
MDDR_APB_S_PREADY	Out		Indicates APB PREADY signal and is used to extend an APB transfer.
MDDR_APB_S_PSLVERR	Out	High	Indicates a transfer failure
MDDR_APB_S_PCLK	In		Indicates APB clock
MDDR_APB_S_PRESET_N	In	Low	Indicates APB active low reset

**Table 200 • SERDERIF APB Slave Configuration Interface Port List**

Port Name	Direction	Polarity	Description
APB_S_PSEL	In	High	Indicates APB slave select
APB_S_PENABLE	In	High	Indicates APB enable
APB_S_PWRITE	In	High	Indicates APB write control signal. Indicates read when Low and write when High.
APB_S_PADDR [13:2]	In		Indicates APB address. Addresses are word aligned.
APB_S_PWDATA [31:0]	In		Indicates APB write data
APB_S_PRDATA [31:0]	Out		Indicates APB read data
APB_S_PREADY	Out		Indicates APB PREADY signal and is used to extend an APB transfer.
APB_S_PSLVERR	Out	High	Indicates a transfer failure
APB_S_PCLK	In		Indicates APB clock
APB_S_PRESET_N	In	Low	Indicates APB active low reset

**Table 201 • HPMS APB Master Configuration Interface Port List**

Port Name	Direction	Polarity	Description
FIC_2_APB_M_PSEL	Out	High	Indicates APB slave select
FIC_2_APB_M_PENABLE	Out	High	Indicates APB enable
FIC_2_APB_M_PWRITE	Out	High	APB write control signal. Indicates read when Low and write when High.
FIC_2_APB_M_PADDR [15:2]	Out		Indicates APB address. Addresses are word aligned.
FIC_2_APB_M_PWDATA [31:0]	Out		Indicates APB write data
FIC_2_APB_M_PRDATA [31:0]	In		Indicates APB read data
FIC_2_APB_M_PREADY	In		Indicates APB PREADY signal and used to extend an APB transfer.
FIC_2_APB_M_PSLVERR	In	High	Indicates a transfer failure
FIC_2_APB_M_PCLK	In		Indicates APB clock
FIC_2_APB_M_PRESET_N	In	Low	Indicates APB active low reset

### 13.1.3 CoreConfig IP

CoreConfig IP facilitates configuration of peripheral blocks (MDDR, FDDR, and SERDESIF blocks) in an IGLOO2 device. The CoreConfig IP has a mirrored master APB port and several mirrored slave APB ports. The mirrored master port must be connected to the FIC\_2\_APB\_MASTER port of the HPMS and the mirrored slave ports must be connected to the APB slave ports of the blocks to be configured.

CoreConfig IP can be configured using the Libero Software and is available in the IP Catalog of the Libero Software. Refer to the **CoreConfig Handbook** for port lists and their descriptions, design flows, memory maps, and Control and Status Register details.

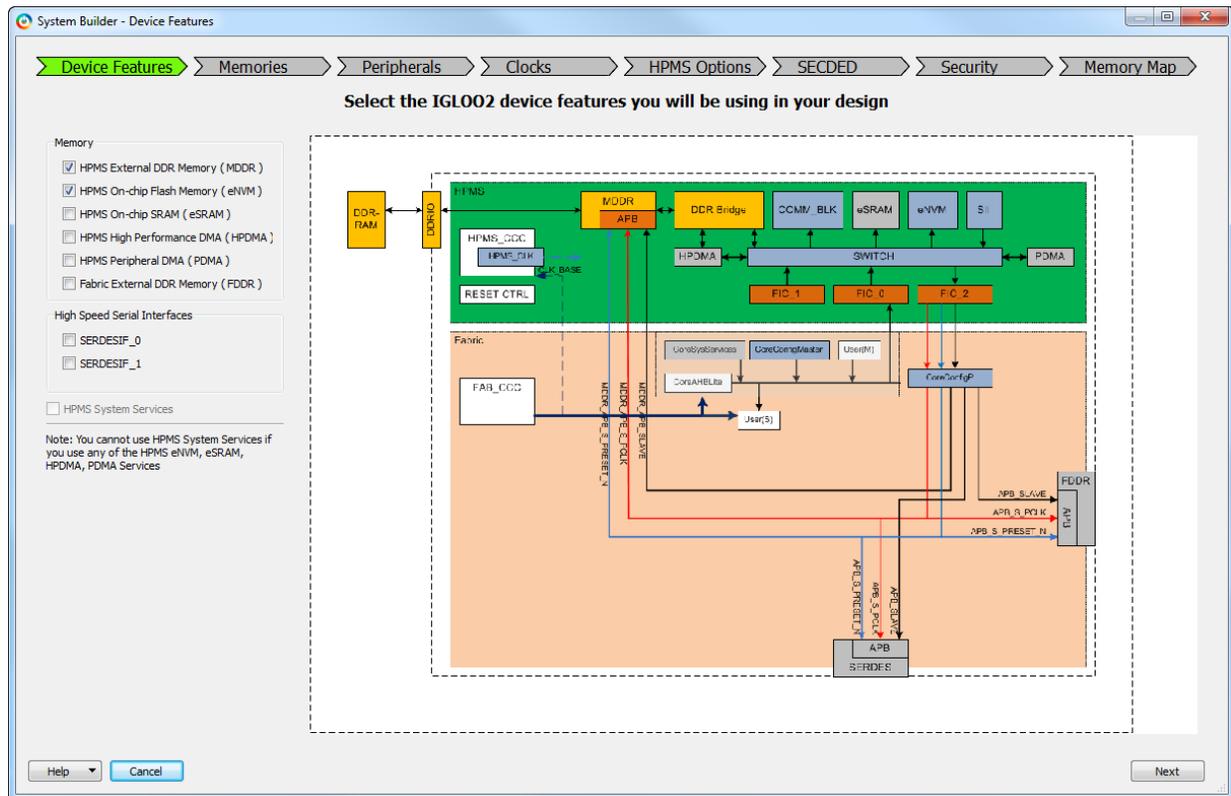
## 13.2 How to Use FIC\_2

This section describes how to use the FIC\_2 (Peripheral Initialization) subsystem 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.

### 13.2.1 Configuring FIC\_2 (Peripheral Initialization) Using Libero SoC

The following figure shows the initial **System Builder** window where the required device features can be selected. For details on how to launch the **System Builder** wizard and a detailed information on how to use it, refer to the *IGLOO2 System Builder User Guide*.

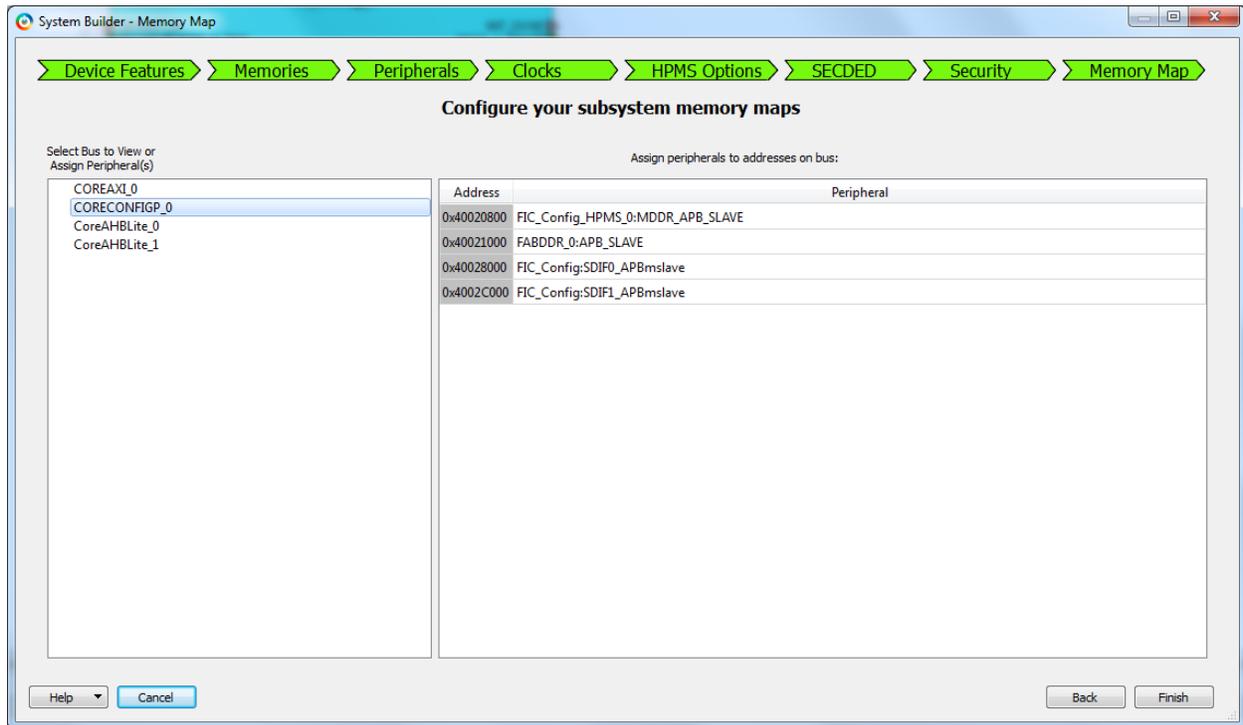
Figure 168 • System Builder Window





- Navigate to the **Memory Map** tab. The following figure shows the **System Builder - Memory Map** tab. Click **Finish** to proceed with creating the HPMS Subsystem.

**Figure 170 • System Builder - Memory Map Tab**

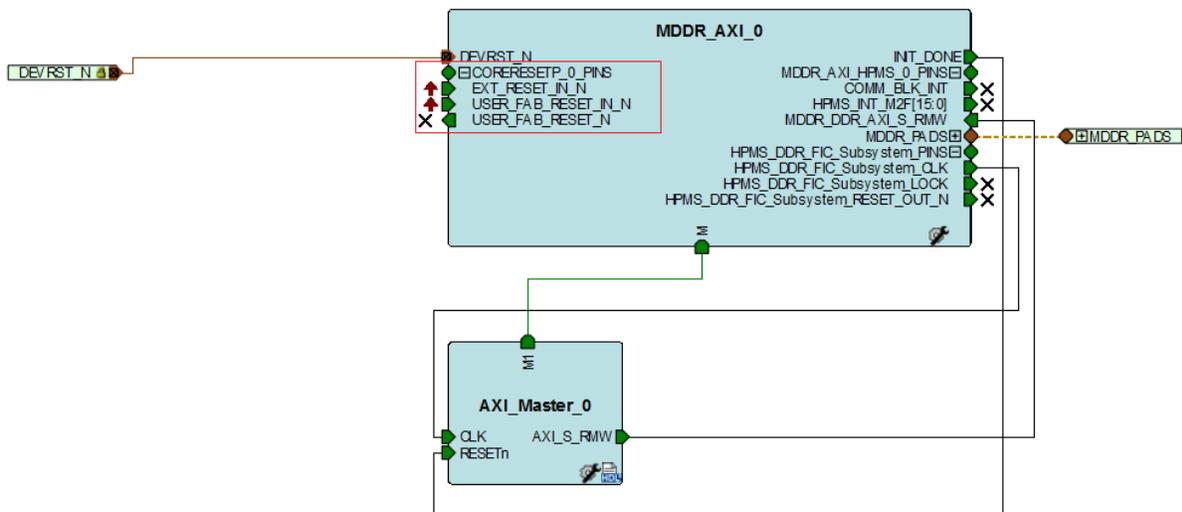


### 13.2.2 FIC\_2 Interfaces for MDDR

- Enable MDDR in the System Builder wizard and create a subsystem.
- Instantiate user AXI master logic in the SmartDesign canvas to access the MDDR subsystem through the AXI interface.

The following figure shows the AXI fabric master connected to AXI master port.

**Figure 171 • FIC\_2 Interfaces for MDDR**



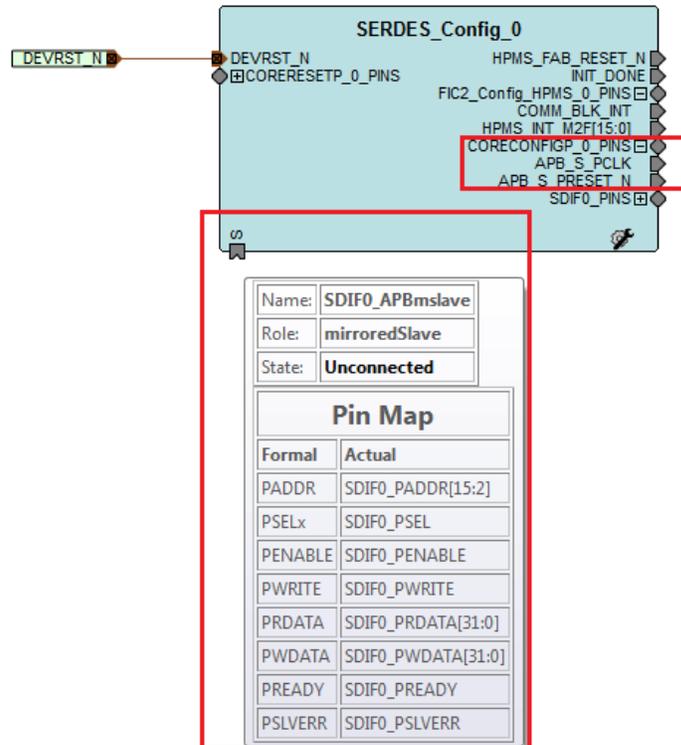
The CoreConfig APB slave signals and CoreConfig IP mirrored APB slave port are connected internally to MDDR by System Builder.

CoreReset handles sequencing reset signals in IGLOO2 devices. It focus on resets related to the peripheral blocks, MDDR, FDDR, and SERDESIF. CoreReset soft IP is available in the Libero SoC IP Catalog. Refer to the **CoreReset Handbook** for port lists, their descriptions and design flow.

### 13.2.3 FIC\_2 Interfaces for SerDes

1. Enable SERDESIF\_0 in the System Builder wizard and create a subsystem. The following figure shows the HPMS subsystem.

**Figure 172 • HPMS Subsystem with APB Configuration Interface Signals**



The preceding figure shows the CoreConfig APB slave signals and the CoreConfig IP mirrored APB slave port that should be connected to the APB slave port of the SerDes block to be configured.

- Instantiate the high-speed serial interface (SERDES\_IF) macro in SmartDesign, and connect the CoreConfig IP mirrored APB slave port with APB slave port of the SERDES\_IF block, as shown in the following figure.

**Figure 173 • Interfacing of CoreConfig IP Mirrored APB Slave with SERDES\_IF Block**

