

**UG0567**  
**User Guide**  
**RTG4 FPGA High-Speed Serial Interfaces**



---

a  **MICROCHIP** company



a  MICROCHIP company

**Microsemi Headquarters**

One Enterprise, Aliso Viejo,  
CA 92656 USA

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

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

Sales: +1 (949) 380-6136

Fax: +1 (949) 215-4996

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

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

©2021 Microsemi, a wholly owned subsidiary of Microchip Technology Inc. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

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

### About Microsemi

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

# Contents

<b>1</b>	<b>Revision History</b>	<b>1</b>
1.1	Revision 8.0	1
1.2	Revision 7.0	1
1.3	Revision 6.0	1
1.4	Revision 5.0	1
1.5	Revision 4.0	2
1.6	Revision 3.0	2
1.7	Revision 2.0	2
1.8	Revision 1.0	2
<b>2</b>	<b>SerDes Block</b>	<b>3</b>
2.1	Introduction	3
2.2	Features	4
2.3	Device Support	4
2.4	Device Architecture	4
2.4.1	SerDes Block Serial Protocols Support	5
2.5	PMA	7
2.5.1	Features	7
2.5.2	Transmit Block	8
2.5.3	Receive Block	11
2.5.4	Clock Block	14
2.5.5	Calibration Resource Sharing	17
2.5.6	SerDes Block-I/O Signal Interface	18
2.6	PCB Design	19
<b>3</b>	<b>Physical Coding Sublayers</b>	<b>20</b>
3.1	PCI Express	20
3.1.1	Features	20
3.1.2	Device Support	21
3.1.3	RTG4 PCI Express Block	21
3.1.4	User Data Throughput	36
3.1.5	PCIe Power Management	36
3.1.6	Bridge Register Space	37
3.1.7	Information Registers	38
3.1.8	PCIe Configuration Space	63
3.1.9	TLP Contents	66
3.1.10	PCB Guidelines for PCIe	71
3.1.11	Generating PCIe Block Using Libero SoC	73
3.2	XAUI	83
3.2.1	Introduction	83
3.2.2	Features	84
3.2.3	Device Support	84
3.2.4	RTG4 XAUI Block	84
3.2.5	Reset and Clocks for XAUI	89
3.2.6	Design Considerations	92
3.2.7	Generating XAUI Block Using Libero SoC	101
3.2.8	Using High-Speed Serial Configurator for XAUI Mode	102
3.2.9	XAUI SPLL Configuration	105
3.2.10	Transmit De-Emphasis	107
3.2.11	Receive CTL Equalization	107

3.2.12	Simulating the SerDes Block with XAUI Mode	107
3.3	EPCS	108
3.3.1	Introduction	108
3.3.2	Features	108
3.3.3	Device Support	108
3.3.4	RTG4 EPCS Interface	109
3.3.5	SerDes Block Architecture in EPCS Mode	110
3.3.6	Reset and Clocks	113
3.3.7	Generating EPCS Block Using Libero SoC	122
3.3.8	Lane Configuration	124
3.3.9	EPCS Lane TX/RX Clock Selection	125
3.3.10	Reference Clock Configuration	125
3.3.11	Signal Integrity Options	126
3.3.12	Transmit De-Emphasis	127
3.3.13	Receive CTL Equalization	127
3.3.14	SerDes Edit Registers	128
3.4	Register Initialization	129
3.4.1	SerDes Initialization	129
3.4.2	Calibration Initialization	130
4	SerDes Block Register Access Map	134
4.1	Configuration of SerDes Block	134
4.1.1	SerDes Block System Register	135
4.1.2	Single-Event Upset Immunity	135
4.1.3	SerDes Block Register	153
4.1.4	SerDes Block Register Bit Definitions	157
4.2	SerDes Register Lock Bits Configuration	175
4.2.1	Lock Bit File	175
4.2.2	Lock Bit File Syntax	175
4.2.3	Locking and Unlocking a Register	176
4.2.4	Configuration Report with Lock	176
5	Debug	177
5.1	Diagnostic Loopbacks	177
5.1.1	Serial Loopbacks	177
5.1.2	Near End Serial PMA Loopback	177
5.1.3	PCS Far End PMA Receive to Transmit Loopback	178
5.2	Pseudo-Random Bit Sequences Pattern Generator	179
5.3	Pseudo-Random Bit Sequences Pattern Checker	179
5.4	Custom Pattern Generator	180
5.5	Using SmartDebug Utility for SerDes	180

# Figures

Figure 1	RTG4 SerDes Block Diagram	3
Figure 2	SerDes Block Capability Per Device	4
Figure 3	Serial Protocol Using SerDes Block and FPGA Logic	6
Figure 4	PMA Block Diagram	7
Figure 5	Transmit Output	8
Figure 6	Transmit Output Signal Parameters	9
Figure 7	Transmit Output Signal	9
Figure 8	Transmit AMP RATIO Setting as shown in Libero SoC	9
Figure 9	Pre-Cursor and Post-Cursor De-Emphasis	10
Figure 10	RX Input Diagram	11
Figure 11	RX EQ RATIO Setting as shown in the Libero SoC	12
Figure 12	Receiver Equalization Frequency Response	12
Figure 13	M, N, and F Settings as shown in the Libero SoC	15
Figure 14	SerDes Reference Clock Input Sources	17
Figure 15	Calibration Resistor Connection	17
Figure 16	RTG4 SerDes Block Diagram	21
Figure 17	SerDes Block Configuration for PCIe Single Protocol Mode	22
Figure 18	RTG4 PCIe EP Implementation	23
Figure 19	Detailed PCIe System Block Diagram	23
Figure 20	Clocks in PCIe Mode	26
Figure 21	SerDes Reference Clock for PCIe Mode	26
Figure 22	Reset Signals in PCIe Mode	27
Figure 23	PCIe IP Block Diagram	32
Figure 24	PCIe to AXI3 Master Address Translation	33
Figure 25	AXI3 Slave to PCIe Address Translation	35
Figure 26	PCIe Configuration Space	63
Figure 27	RTG4 Power Supply to the PCIe Link Implementation	71
Figure 28	PCI Express Power-Up States	72
Figure 29	Reversed Signals in PCIe Mode	72
Figure 30	Libero SoC Catalog	73
Figure 31	PCIe SerDes Block Instantiation without Initialization feature on the SmartDesign Canvas	74
Figure 32	RTG4 High Speed Serial Interface (PCIe, EPCS and XAUI) Configurator	75
Figure 33	Reference Clock Configuration	75
Figure 34	Reference Clock Configurator	76
Figure 35	Configuring PCIE	77
Figure 36	PCIe Configuration Tab	77
Figure 37	Base Address Registers	78
Figure 38	PCIE Register Options	78
Figure 39	Power Management Settings	79
Figure 40	PCIe Configuration - Master Interface Tab	80
Figure 41	PCIe Configuration - Slave Interface Tab	82
Figure 42	RTG4 SerDes Block Diagram	83
Figure 43	XAUI Implementation in RTG4	85
Figure 44	XAUI IP Block Diagram	86
Figure 45	Clocking in XAUI Mode	90
Figure 46	SerDes Reference Clock for XAUI Mode	91
Figure 47	MDIO System Block Diagram	93
Figure 48	Transmit XGMII Interface Timing Diagram	94
Figure 49	XGMII Interface Receive Timing Diagram	94
Figure 50	MDIO Interface Read Timing Diagram	95
Figure 51	MDIO Interface Read Timing Diagram	95
Figure 52	Libero SoC Catalog	102
Figure 53	XAUI Configurator	103
Figure 54	RTG4 High Speed Serial Interface Configurator—Lane Configuration	104

Figure 55	SPLL Setting for XAUI	105
Figure 56	Reference Clock Configuration	105
Figure 57	Reference Clock Configurator	105
Figure 58	Signal Integrity Options	106
Figure 59	Signal Integrity Parameters	106
Figure 60	RTG4 SerDes Block Diagram	108
Figure 61	Application using SerDes Block EPCS Interface	109
Figure 62	SerDes Block Clocking in EPCS Mode	114
Figure 63	SerDes Reference Clock Options for EPCS Modes	115
Figure 64	Flywheel FIFO	115
Figure 65	EPCS X2 Links	117
Figure 66	CDR Steady State Considerations Diagram	118
Figure 67	EPCS Startup and Operating Behavior (20-bit Width)	119
Figure 68	SerDes Block Reset Signals in EPCS Mode	121
Figure 69	Libero SoC Catalog	122
Figure 70	EPCS Block Instantiation	123
Figure 71	EPCS SerDes Block Configurator	123
Figure 72	RTG4 High Speed Serial Interface Configurator—Lane Configuration	124
Figure 73	EPCS Lane TX/RX Clock Selection	125
Figure 74	Reference Clock Configuration	125
Figure 75	Reference Clock Receiver Function Configuration	125
Figure 76	Signal Integrity Options	126
Figure 77	Signal Integrity Parameters	126
Figure 78	SerDes Edit Register GUI	128
Figure 79	PCIe Error Message	130
Figure 80	SPLL Lock Connection	131
Figure 81	CoreABC Code	132
Figure 82	FCCC Lock	133
Figure 83	SerDes Block Memory Map	134
Figure 84	Address Decoder Logic Block Diagram	135
Figure 85	Lock Bits Example Format	176
Figure 86	Register Lock Bit Settings	176
Figure 87	Configuration Report with Lock	176
Figure 88	Diagnostic Loopbacks	177
Figure 89	Near End Serial PMA Loopback	177
Figure 90	Far End PMA RX to TX Loopback	178

# Tables

Table 1	Available SerDes Blocks in RTG4 Devices	4
Table 2	SerDes Block Module Single Protocol Usage Overview	5
Table 3	Predefined Transmit Amplitude	9
Table 4	Receiver Equalization (CTLE) Settings	12
Table 5	Reference Clock Specifications (Typical)	15
Table 6	Support Reference Clock Input Standards	16
Table 7	SerDes Block I/O - PAD Interface	18
Table 8	PCIe Endpoint Blocks Available in RTG4	21
Table 9	AXI3 and Outstanding Transactions	24
Table 10	Reference Clock Signals for SerDes in PCIe Mode	27
Table 11	PCIe System AXI3 Master Interface	28
Table 12	PCIe System AXI3 Slave Interface	29
Table 13	PCIe System APB Slave Interface	30
Table 14	PCIe System Clock Signals	30
Table 15	PCIe System Reset Signals	30
Table 16	PCIe Interrupt and Power Management Interface	31
Table 17	AXI_MASTER_WINDOW Registers	32
Table 18	AXI_SLAVE_WINDOW Registers	34
Table 19	Theoretical PCIe Throughput	36
Table 20	PCIe Low Power States	37
Table 21	PCIE Information Registers	38
Table 22	PCIE Bridge Configuration Registers	38
Table 23	PCIE Bridge Power Management Registers	40
Table 24	Address Mapping Registers	41
Table 25	EP Interrupt Registers	42
Table 26	PCIe Control and Status Registers	42
Table 27	PCIE_VID_DEVID	43
Table 28	CFG_PRMSCR	43
Table 29	PCIE_CLASS_CODE_REG	43
Table 30	PCIE_BAR0	43
Table 31	PCIE_BAR1	44
Table 32	PCIE_BAR2	44
Table 33	PCIE_BAR3	44
Table 34	PCIE_BAR4	44
Table 35	PCIE_BAR5	45
Table 36	PCIE_SUBSYSTEM_ID	45
Table 37	PCIE_DEVSCR	45
Table 38	PCIE_LINKSCR	45
Table 39	PCIE_TC_VC_MAPPING	46
Table 40	PCIE_CAPTURED_BUS_DEVICE_NB	46
Table 41	PCIE_MSI_CTRL_STATUS	46
Table 42	PCIE_LTSSM	47
Table 43	PCIE_POWER_MGT_CAPABILITY	48
Table 44	PCIE_CFG_PMSCR	48
Table 45	PCIE_AER_ECRC_CAPABILITY	48
Table 46	PCIE_VC1_CAPABILITY	48
Table 47	MAX_PAYLOAD_SIZE	49
Table 48	PCIE_ASPM_L0S_CAPABILITY	49
Table 49	PCIE_ASPM_L1_CAPABILITY	49
Table 50	PCIE_TIMEOUT_COMPLETION	50
Table 51	PCIE_PM_DATA_SCALE_0	50
Table 52	PCIE_PM_DATA_SCALE_1	51
Table 53	PCIE_PM_DATA_SCALE_2	51
Table 54	PCIE_PM_DATA_SCALE_3	52

Table 55	PCIE_MSI_0	52
Table 56	PCIE_ERROR_COUNTER_0	53
Table 57	PCIE_ERROR_COUNTER_1	53
Table 58	PCIE_ERROR_COUNTER_2	53
Table 59	PCIE_ERROR_COUNTER_3	54
Table 60	PCIE_CREDIT_ALLOCATION_0	54
Table 61	PCIE_CREDIT_ALLOCATION_1	54
Table 62	PCIE_AXI_SLAVE_WINDOW0_0	54
Table 63	PCIE_AXI_SLAVE_WINDOW0_1	55
Table 64	PCIE_AXI_SLAVE_WINDOW0_2	55
Table 65	PCIE_AXI_SLAVE_WINDOW0_3	55
Table 66	PCIE_AXI_SLAVE_WINDOW1_0	55
Table 67	PCIE_AXI_SLAVE_WINDOW1_1	55
Table 68	PCIE_AXI_SLAVE_WINDOW1_2	56
Table 69	PCIE_AXI_SLAVE_WINDOW1_3	56
Table 70	PCIE_AXI_SLAVE_WINDOW2_0	56
Table 71	PCIE_AXI_SLAVE_WINDOW2_1	56
Table 72	PCIE_AXI_SLAVE_WINDOW2_2	56
Table 73	PCIE_AXI_SLAVE_WINDOW2_3	57
Table 74	PCIE_AXI_SLAVE_WINDOW3_0	57
Table 75	PCIE_AXI_SLAVE_WINDOW3_1	57
Table 76	PCIE_AXI_SLAVE_WINDOW3_2	57
Table 77	PCIE_AXI_SLAVE_WINDOW3_3	58
Table 78	PCIE_AXI_MASTER_WINDOW0_0	58
Table 79	PCIE_AXI_MASTER_WINDOW0_1	58
Table 80	PCIE_AXI_MASTER_WINDOW0_2	58
Table 81	PCIE_AXI_MASTER_WINDOW0_3	58
Table 82	PCIE_AXI_MASTER_WINDOW1_0	59
Table 83	PCIE_AXI_MASTER_WINDOW1_1	59
Table 84	PCIE_AXI_MASTER_WINDOW1_2	59
Table 85	PCIE_AXI_MASTER_WINDOW1_3	59
Table 86	PCIE_AXI_MASTER_WINDOW2_0	60
Table 87	PCIE_AXI_MASTER_WINDOW2_1	60
Table 88	PCIE_AXI_MASTER_WINDOW2_2	60
Table 89	PCIE_AXI_MASTER_WINDOW2_3	60
Table 90	PCIE_AXI_MASTER_WINDOW3_0	60
Table 91	PCIE_AXI_MASTER_WINDOW3_1	61
Table 92	PCIE_AXI_MASTER_WINDOW3_2	61
Table 93	PCIE_AXI_MASTER_WINDOW3_3	61
Table 94	PCIE_INFO	61
Table 95	RESERVED	61
Table 96	PCIE_DEV2SCR	62
Table 97	PCIE_LINK2SCR	62
Table 98	RESERVED	62
Table 99	Configuration Inputs for AXI3 Bridge	64
Table 100	PCIe Extended Capability Structure (Function 0)	64
Table 101	Type 0 Configuration Register	64
Table 102	IP Core Status Register	65
Table 103	MSI Capability Structure Register	65
Table 104	Power Management Capability Structure	65
Table 105	PCIe Capability Structure Register	66
Table 106	PCIe AER Extended Capability Structure	66
Table 107	Memory Read Request 32-bit Addressing Descriptor Format	67
Table 108	Memory Read Request-Locked 32-bit Addressing Descriptor Format	67
Table 109	Memory Read Request 64-bit Addressing Descriptor Format	67
Table 110	Memory Read Request-Locked 64-bit Addressing Descriptor Format	67
Table 111	Type 0 Configuration Read Request Descriptor Format	68
Table 112	Type 0 Configuration Read Request Descriptor Format	68
Table 113	Message (without data) Descriptor Format	68



Table 114	Completion (without data) Descriptor Format	68
Table 115	Completion Locked (without data) Descriptor Format	69
Table 116	Memory Write Request 32-bit Addressing Descriptor Format	69
Table 117	Memory Write Request 64-bit Addressing Descriptor Format	69
Table 118	Type 0 Configuration Write Request Descriptor Format	69
Table 119	Type 0 Configuration Write Request Descriptor Format	69
Table 120	Completion Locked (with data) Descriptor Format	70
Table 121	Message (with data) Descriptor Format	70
Table 122	Type 1 Configuration Write Request Descriptor Format	70
Table 123	Completion (with data) Descriptor Format	70
Table 124	SerDes Blocks in RTG4 FPGAs Supporting XAUI (PCISS and NPSS)	84
Table 125	XAUI Implementation in RTG4	85
Table 126	MDIO Interface Signals	86
Table 127	XGMII Transmit Interface Signals	87
Table 128	XAUI IP Block Miscellaneous Control Signal	88
Table 129	XGMII Receive Interface Signals	88
Table 130	APB Slave Interface—APB_SLAVE	89
Table 131	Clock Signals in XAUI Mode	91
Table 132	XAUI Mode Reset Signals	92
Table 133	Reference Clock Signals for SerDes	92
Table 134	MDIO Registers	96
Table 135	Reg01 XS Status 1 Register	97
Table 136	Reg00 XS Control 1 Register	97
Table 137	Reg02 XS Device Identifier Low Register	98
Table 138	Reg03 the XS Device Identifier High Register	98
Table 139	Reg04 XS Speed Ability Register	98
Table 140	Reg05 XS Devices in Package Low Register	98
Table 141	Reg07 XS Status 2 Register	99
Table 142	Reg08 XS Package ID Low Register	99
Table 143	Reg06 XS Devices in Package High Register	99
Table 144	Reg09 XS Package ID High Register	100
Table 145	Reg10 XGXS Lane Status register	100
Table 146	Reg11 the XGXS Test Control Register	101
Table 147	Reg12 Vendor-Specific Reset Low 1 Register	101
Table 148	Reg13 Vendor-Specific Reset Low 2 Register	101
Table 149	Reg14 Vendor-Specific Reset High 1 Register	101
Table 150	Reg15 Vendor-Specific Reset High 2 Register	101
Table 151	XAUI Protocol	103
Table 152	Reference Clock Receiver Function Configuration	106
Table 153	CTL Equalization Predefined Settings and Register Values	107
Table 154	Available SerDes Blocks in RTG4 Devices	108
Table 155	EPCS Interface Usage in Single-Protocol and Multi-Protocol Mode	109
Table 156	SerDes Block – EPCS Interface	110
Table 157	EPCS Interface and SerDes Lane Mapping in Single-Protocol Mode	110
Table 158	EPCS APB Slave Interface	113
Table 159	Clock Signals in EPCS Mode	114
Table 160	SerDes Block Reset Signals in EPCS Mode	122
Table 161	XAUI Protocol	124
Table 162	Reference Clock Receiver Function Configuration	126
Table 163	CTL Equalization Predefined Settings and Register Values	127
Table 164	SerDes Block System Registers	136
Table 165	SYSTEM_SER_PLL_CONFIG_LOW (0x2000)	138
Table 166	SYSTEM_SER_PLL_CONFIG_HIGH (0x2004)	138
Table 167	SYSTEM_SER_INTERRUPT_ENABLE (0x200C)	140
Table 168	SYSTEM_CONFIG_AXI_AHB_BRIDGE (0x2010)	140
Table 169	SYSTEM_SER_SOFT_RESET (0x2008)	140
Table 170	SYSTEM_CONFIG_PCIE_PM (0x2020)	141
Table 171	SYSTEM_CONFIG_ECC_INTR_ENABLE (0x2014)	141
Table 172	SYSTEM_CONFIG_PHY_MODE_1 (0x2028)	142

Table 173	SYSTEM_CONFIG_PHY_MODE_0 (0x2024)	142
Table 174	SYSTEM_CONFIG_PCIE_2 (0x2038)	143
Table 175	SYSTEM_CONFIG_PCIE_3 (0x203C)	143
Table 176	SYSTEM_CONFIG_PHY_MODE_2 (0x202C)	143
Table 177	SYSTEM_CONFIG_PCIE_0 (0x2030)	143
Table 178	SYSTEM_CONFIG_PCIE_1 (0x2034)	143
Table 179	SYSTEM_CONFIG_BAR_SIZE_2_3 (0x2044)	144
Table 180	SYSTEM_CONFIG_BAR_SIZE_0_1 (0x2040)	144
Table 181	SYSTEM_CONFIG_BAR_SIZE_4_5 (0x2048)	145
Table 182	SYSTEM_SER_CLK_STATUS (0x204C)	146
Table 183	SYSTEM_SER_INTERRUPT (0x2058)	146
Table 184	SYSTEM_SERDES_INTR_STATUS Register (0x205C)	146
Table 185	SYSTEM_REFCLK_SEL (0x2064)	146
Table 186	SYSTEM_EPCS_RSTN_SEL (0x206C)	147
Table 187	SYSTEM_SERDES_TEST_OUT (0x2074)	147
Table 188	SYSTEM_PCLK_SEL Register (0x2068)	147
Table 189	SYSTEM_FABRIC_CLK_ENA (0x207C)	148
Table 190	SYSTEM_CONF_AXI_MSTR_WNDW_0 (0x2084)	148
Table 191	SYSTEM_CONF_AXI_MSTR_WNDW_1 (0x2088)	148
Table 192	SYSTEM_CONF_AXI_MSTR_WNDW_2 (0x208C)	149
Table 193	SYSTEM_CONF_AXI_MSTR_WNDW_3 (0x2090)	149
Table 194	SYSTEM_CONF_AXI_SLV_WNDW_0 (0x2094)	149
Table 195	SYSTEM_CONF_AXI_SLV_WNDW_1 (0x2098)	149
Table 196	SYSTEM_CONF_AXI_SLV_WNDW_2 (0x209C)	149
Table 197	SYSTEM_CONF_AXI_SLV_WNDW_3 (0x20A0)	149
Table 198	SYSTEM_DESKEW_CONFIG (0x20A4)	149
Table 199	SYSTEM_ADVCONFIG (0x20b4)	150
Table 200	SYSTEM_ECC_ERR_INJECT (0x20bc)	151
Table 201	SYSTEM_REFCLK_MSIO_CONFIG (0x20c4)	151
Table 202	SYSTEM_ENHANCEMENT (0x20c8)	152
Table 203	SYSTEM_TXFWF_CONFIG (0x20cc)	152
Table 204	SYSTEM_FWF_STATUS(0x20d4)	153
Table 205	SYSTEM_RXFWF_CONFIG (0x20d0)	153
Table 206	SerDes Block Lane Registers	154
Table 207	CR0	157
Table 208	ERRCNT_DEC	158
Table 209	RXIDLE_MAX_ERRCNT_THR	158
Table 210	IMPED_RATIO	158
Table 211	PLL_F_PCLK_RATIO	159
Table 212	PLL_M_N	159
Table 213	CNT250NS_MAX	159
Table 214	RE_AMP_RATIO	160
Table 215	RE_CUT_RATIO	160
Table 216	TX_AMP_RATIO	160
Table 217	TX_PRE_RATIO	161
Table 218	ENDCALIB_MAX	161
Table 219	CALIB_STABILITY_COUNT	161
Table 220	TX_PST_RATIO	161
Table 221	POWERDOWN	162
Table 222	RX_OFFSET_COUNT	164
Table 223	PMA_STATUS	164
Table 224	PRBS_CTRL	164
Table 225	PHY_POWER_OVERRIDE	165
Table 226	PRBS_ERRCNT	165
Table 227	PHY_RESET_OVERRIDE	165
Table 228	CUSTOM_PATTERN_7_0	166
Table 229	CUSTOM_PATTERN_15_8	166
Table 230	CUSTOM_PATTERN_23_16	166
Table 231	CUSTOM_PATTERN_31_24	167

Table 232	CUSTOM_PATTERN_39_32 .....	167
Table 233	CUSTOM_PATTERN_47_40 .....	167
Table 234	CUSTOM_PATTERN_55_48 .....	168
Table 235	CUSTOM_PATTERN_63_56 .....	168
Table 236	CUSTOM_PATTERN_71_64 .....	168
Table 237	CUSTOM_PATTERN_79_72 .....	169
Table 238	CUSTOM_PATTERN_CTRL .....	169
Table 239	Reserved .....	169
Table 240	GEN1_TX_PLL_CCP .....	170
Table 241	GEN1_RX_PLL_CCP .....	170
Table 242	PCS_LOOPBACK_CTRL .....	170
Table 243	CDR_PLL_MANUAL_CR .....	171
Table 244	UPDATE_SETTINGS .....	171
Table 245	PRBS_ERR_CYC_FIRST_7_0 .....	171
Table 246	PRBS_ERR_CYC_FIRST_23_16 .....	172
Table 247	PRBS_ERR_CYC_FIRST_31_24 .....	172
Table 248	PRBS_ERR_CYC_FIRST_39_32 .....	172
Table 249	PRBS_ERR_CYC_FIRST_15_8 .....	172
Table 250	PRBS_ERR_CYC_FIRST_49_48 .....	173
Table 251	PRBS_ERR_CYC_FIRST_7_0 .....	173
Table 252	PRBS_ERR_CYC_FIRST_15_8 .....	173
Table 253	PRBS_ERR_CYC_FIRST_47_40 .....	173
Table 254	PRBS_ERR_CYC_FIRST_23_16 .....	174
Table 255	PRBS_ERR_CYC_FIRST_31_24 .....	174
Table 256	PRBS_ERR_CYC_FIRST_39_32 .....	174
Table 257	PRBS_ERR_CYC_FIRST_49_48 .....	175
Table 258	PRBS_ERR_CYC_FIRST_47_40 .....	175
Table 259	Diagnostic Loopback Support per Protocol .....	178
Table 260	SerDes Block PRBS Patterns .....	179

# 1 Revision History

---

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

## 1.1 Revision 8.0

The following is a summary of the changes in revision 8.0 of this document.

- Information about receiver equalization settings was added. See [Table 4](#), page 12.
- Information about AXI\_S\_RRESP[1:0] port in [Table 12](#), page 29 was updated.
- Information about byte offset in [Table 102](#), page 65 was added.
- Information about [RTG4 XAUI Block](#), page 84 was updated.
- Information about EPCS clock was updated. See [Table 157](#), page 110.
- Updated [Table 179](#), page 144, [Table 180](#), page 144, [Table 181](#), page 145, and [Table 215](#), page 160.
- Information about SYSTEM\_FABRIC\_CLK\_ENA (0x207C) bit numbers and bit names were added. See [Table 189](#), page 148.
- Information about [SerDes Register Lock Bits Configuration](#), page 175 was added.
- Removed information related to AHBLite support for PCIe.
- Information about [Fabric Interface](#), page 78 was updated.
- Information about XAUI\_PHY\_NOT\_READY port was updated. See [Table 129](#), page 88.
- Information about APB\_S\_PCLK port was updated. See [Table 130](#), page 89.
- Information about [Clocking in XAUI Mode](#), page 90 was updated.
- Information about [XAUI FPGA Clocks](#), page 92 was added.
- Information about EPCS\_0\_READY, EPCS\_1\_READY, EPCS\_2\_READY, and EPCS\_3\_READY was updated. See [Table 157](#), page 110.
- Information about AXI3 to AHBL Bridge (Master/Slave), page 25 was added.

## 1.2 Revision 7.0

The following is a summary of the changes in revision 7.0 of this document.

- Information about [PCIe Reset Network](#), page 27 was updated.
- Register Initialization and calibration was consolidated and added. See [Register Initialization](#), page 129.
- Information about [SerDes Initialization](#), page 129 covering [CN19009B](#) was updated.

## 1.3 Revision 6.0

The following is a summary of the changes in revision 6.0 of this document.

- Information about GLOBAL\_0\_OUT, GLOBAL\_1\_OUT port was updated. See [Table 157](#), page 110.
- Information about ARXSKIPBIT port was updated. See [Table 157](#), page 110.
- Information about [Multi-lane Common Clock Synchronization Considerations](#), page 117 was added.
- Information about [SerDes Edit Registers](#), page 128 was added.
- Information about PLL\_M\_N and CNT250NS\_MAX register name was updated. See [Table 206](#), page 154, [Table 212](#), page 159, and [Table 213](#), page 159.
- Information about force\_signal register was updated. See [Table 221](#), page 162.
- Information about EPCS\_X\_RESET\_N port was updated. See [Table 160](#), page 122.

## 1.4 Revision 5.0

The following is a summary of the changes in revision 5.0 of this document.

- Information about Flywheel FIFO was added. See [Flywheel FIFO](#), page 115.
- Information about [Transmit Output Buffer](#), page 8 was updated.
- Information about SerDes clock stability was added. See [SerDes Startup](#), page 18.
- Information about Register (0x207C) was corrected. See [Table 189](#), page 148.
- Information about PRBS patterns were updated. See [Table 260](#), page 179.

- Information about SYSTEM\_REFCLK\_SEL (0x2064) was updated. See [Table 185](#), page 146 and [SerDes Block Clock Network in EPCS](#), page 113.
- PRBS\_EN is replaced with PRBS\_CHK. For more information about PRBS\_CHK, see [Table 225](#), page 165.
- Information about SerDes EPCS FWF was updated. See [SerDes Block Clock Network in EPCS](#), page 113.
- Information about LVDS25 I/O Standard supports HCSL was updated. See [SerDes Reference Clock Inputs](#), page 16.

## 1.5 Revision 4.0

The following is a summary of the changes in revision 4.0 of this document.

- Updated [Table 188](#), page 147 in [SerDes Block Register Access Map](#), page 134 chapter.
- Information about SerDes reference clock inputs was added, see [Table 6](#), page 16.
- Updated [Table 11](#), page 28, [Table 12](#), page 29, [Table 126](#), page 86, [Table 129](#), page 88, [Table 166](#), page 138, [Table 206](#), page 154, and [Table 259](#), page 178.
- Added [Single-Event Upset Immunity](#), page 135 section to [SerDes Block Register Access Map](#), page 134 chapter.
- Information about LVDS33 and LVDS25 was added. For more information, see [Table 6](#), page 16.

## 1.6 Revision 3.0

Updated [Figure 15](#), page 17 in [SerDes Block](#) chapter.

## 1.7 Revision 2.0

The following is a summary of the changes in revision 2.0 of this document.

- Updated the SERDESIF name to SERDES block in the document.
- Added the [Using High-Speed Serial Interfaces Configurator in EPCS Mode](#), page 122 section.
- Deleted TX PLL and CDR PLL Operation section in SERDES Chapter.
- Updated the [Table 11](#), page 28 and [Table 12](#), page 29.
- Added the [Table 199](#), page 150, [Table 201](#), page 151, [Table 202](#), page 152, [Table 203](#), page 152, and [Table 204](#), page 153.
- Updated the [Figure 1](#), page 3, [Figure 16](#), page 21, [Figure 17](#), page 22, [Figure 42](#), page 83, [Figure 43](#), page 85, [Figure 60](#), page 108, and [Figure 61](#), page 109.
- Re-organized the "Serializer/De-serializer" chapter.

## 1.8 Revision 1.0

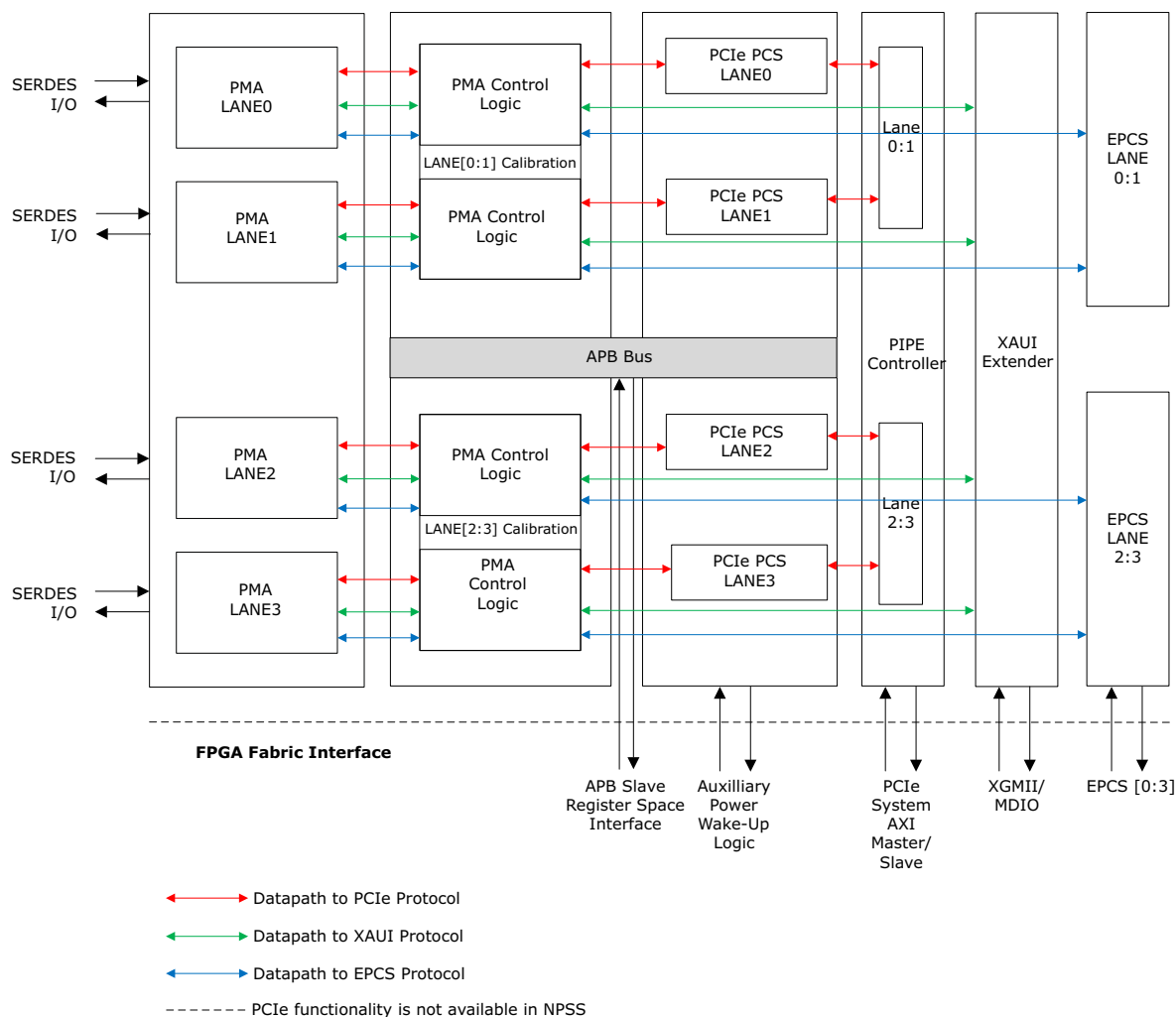
Revision 1.0 was the first publication of this document.

## 2 SerDes Block

### 2.1 Introduction

The RTG4™ FPGA device includes a serializer/de-serializer (SerDes block) that provides support for several serial communication standards. This module integrates several functional blocks to support multiple high-speed serial protocols within the device. The SerDes block has embedded functionality to integrate required features used with several protocols and is used with FPGA hosted application designs.

**Figure 1 • RTG4 SerDes Block Diagram**



## 2.2 Features

The SerDes block has the following features:

- 6 SerDes/PCS blocks (Quads/4x Lanes) - up to 24 total SerDes lanes
- x1, x2, and x4 PCIe endpoint protocol support
  - PCIe is compliant with the PCIe base specification 1.1 for Gen1 and PCIe base specification 2.1 for Gen1 or Gen2—only supports the 2.5 Gbps variant of the Gen2 specification.
- 10 Gigabit attachment unit interface (XAUI) protocol supports four lanes of 3.125Gbps (10Gbps with overhead).
- External physical coding sub-layer (EPCS) interface supports user defined high serial protocols, such as serial gigabit media independent interface (SGMII)1000-BaseX, and JESD204B protocol support up to 3 Gbps.
- SEU mitigation techniques including TMR logic triplication and self-correcting latches.
- Single or multiple serial protocol modes of operation. In multiple serial protocol modes, two protocols can be implemented on the four physical lanes of the SerDes block.
- ARM AMBA APB-3 compliant slave interface for SerDes block configuration register access.

See the *DS0131: RTG4 FPGA Datasheet* for AC and DC specifications.

## 2.3 Device Support

The following table shows the SerDes blocks available in each RTG4 device.

**Table 1 • Available SerDes Blocks in RTG4 Devices**

	RT4G150
SerDes Blocks	6
SerDes Lanes	24
PCIE Endpoints	2

**Note:** The specified number of SerDes blocks varies depending on the device package. PCIe Endpoints support x1, x2, or x4 PCIe links.

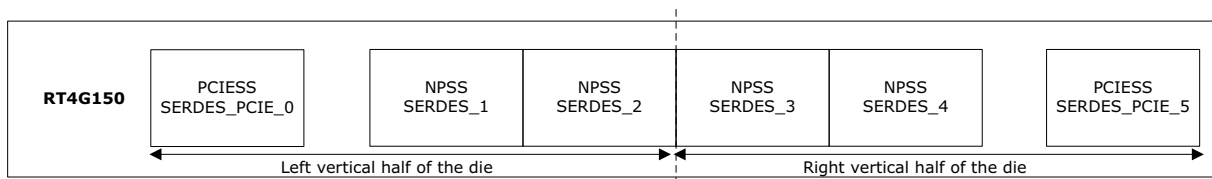
## 2.4 Device Architecture

The RTG4 device family has up to six SerDes blocks. Each SerDes block interfaces with fabric, program control, and four duplex SerDes differential I/O pads known as SerDes Lanes. Individually, the SerDes block shows the high level view of the RTG4 SerDes block. The SerDes block which is dependent on the FPGA design implemented protocol, provides an AXI3 (PCIe), XGMII (XAUI), or native SerDes clock and data (EPCS) along with the control plane interface APB.

As shown in the following figure, two of the RTG4 SerDes block modules in each device contain PCIe endpoint functionality. These PCIe Sub-system (PCIESS) modules include PCIe functionality whereas non-PCIe Sub-system (NPSS) modules do not include PCIe logic.

The following figure shows the number of PCIESS and NPSS modules per device. PCIESS modules act as a superset with all the capabilities of NPSS modules and PCIe functionality included. The north side of the device includes three SerDes blocks on the right and left vertical halves of the array.

**Figure 2 • SerDes Block Capability Per Device**





Each of the SerDes blocks include:

- **PMA:** This block implements up to four channels of high-speed I/O, the physical media attachment layer (PMA), and a physical coding sub-layer (PCS) of PCIe protocols. It also implements the PMA calibration and control logic. This PCS layer is compliant to the Intel PIPE 2.0 specification. The PCIe PCS functionality can be bypassed completely in order to use the SerDes lanes for protocols other than PCIe. This allows usage of the PMA in various PHY modes and for the implementation of different protocols in the RTG4 device.
- **PCIe System:** This block implements the x1, x2, and x4 lanes PCIe endpoint (regular and reverse lanes mode) with an AXI3 interface to the fabric.

**Note:** PCIe functionality is included only in the two PCI ESS blocks.

- **XAUI Extender:** This block is an XGMII extender to support the XAUI protocol through a FPGA IP MAC core in the FPGA fabric.
- **EPCS:** This block is a basic mode used to extend the SerDes for custom support access to the FPGA fabric.
- **SerDes Block System Register:** The SerDes block system registers control the SerDes block module for single protocol or multi-protocol support implementation. These registers can be accessed through the 32-bit APB interface, and the default values of these registers are configured using the Libero<sup>®</sup> System-on-Chip (SoC) software. The SerDes block is initially configured at power-up with parameters determined during FPGA design flow using the Libero SoC software. The SerDes block configuration can be subsequently changed by writing the related control registers through the advanced peripheral bus (APB) interface.

**Table 2 • SerDes Block Module Single Protocol Usage Overview**

Protocol	SerDes Block Description	Data Rate (Gbps)	Reference Clock (typ) Input Frequency
PCIe	SerDes block is configured to use PCIe Gen1/2 x4, x2, and x1 link mode. The PCIe link can be configured in regular or reversed modes.	2.5	100 MHz
XAUI	SerDes block is configured to use all four lanes. In XAUI mode, all lanes are used and the PCIe system is put in RESET state.	4 x 3.125	156.25 MHz
EPCS	In EPCS mode, any serial protocol can be run through the EPCS interface to the fabric using the EPCS interface. EPCS mode is used to implement many other standard protocol interfaces such as SGMII.	1 – 3.125	100 – 160 MHz

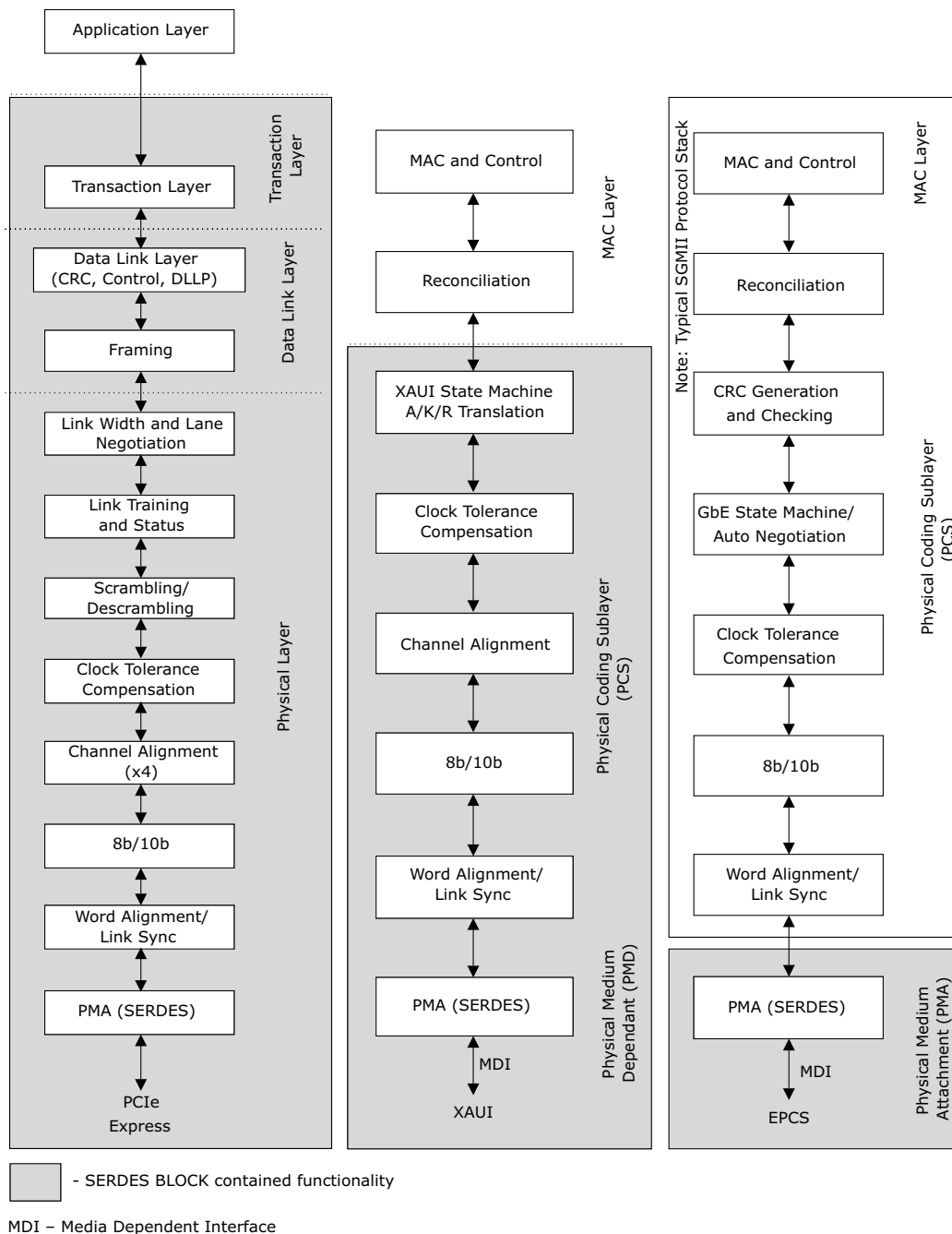
## 2.4.1 SerDes Block Serial Protocols Support

The SerDes block supports the implementation of multiple high-speed serial protocols. Although each of the serial protocols is unique, all of them are layered protocol stacks, and the implementation varies from one layer to the next layer. Typically, the physical layer consists of fixed functionality, common to multiple packet-based protocols, while the upper layers tend to be more customizable.

The advantage of connecting the FPGA logic and SerDes blocks is that it allows multiple serial protocols in the RTG4 devices. The following figure shows the implementation of PCIe, XAUI, and EPCS protocols using the SerDes block and FPGA fabric and the fixed modules contained within the SerDes block per application. As shown in the example, PCIe applications include many functional blocks within the SerDes block, whereas EPCS requires more FPGA IP blocks for complete system implementation.

The RTG4 devices contain a mix of SerDes blocks with different embedded protocol support logic. The PCIe sub-system (PCI ESS) within the SerDes blocks implements the PCIe EP specification for transaction, data link, and physical layers. The PCI ESS SerDes block also can be used for XAUI and customized EPCS protocols such as SGMII and JESD204B. The RTG4 also includes similar SerDes blocks, which have the same XAUI and EPCS capabilities without embedded PCIe logic, known as NPSS. All RTG4 devices include two PCI ESS blocks and various quantities of NPSS blocks based on the FPGA device and package sizes.



**Figure 3 • Serial Protocol Using SerDes Block and FPGA Logic**

## 2.5 PMA

The physical media attachment (PMA) of the RTG4 FPGA family is included within the SerDes block module. This section describes the fixed hardware blocks and the physical hardware capabilities of the PMA.

### 2.5.1 Features

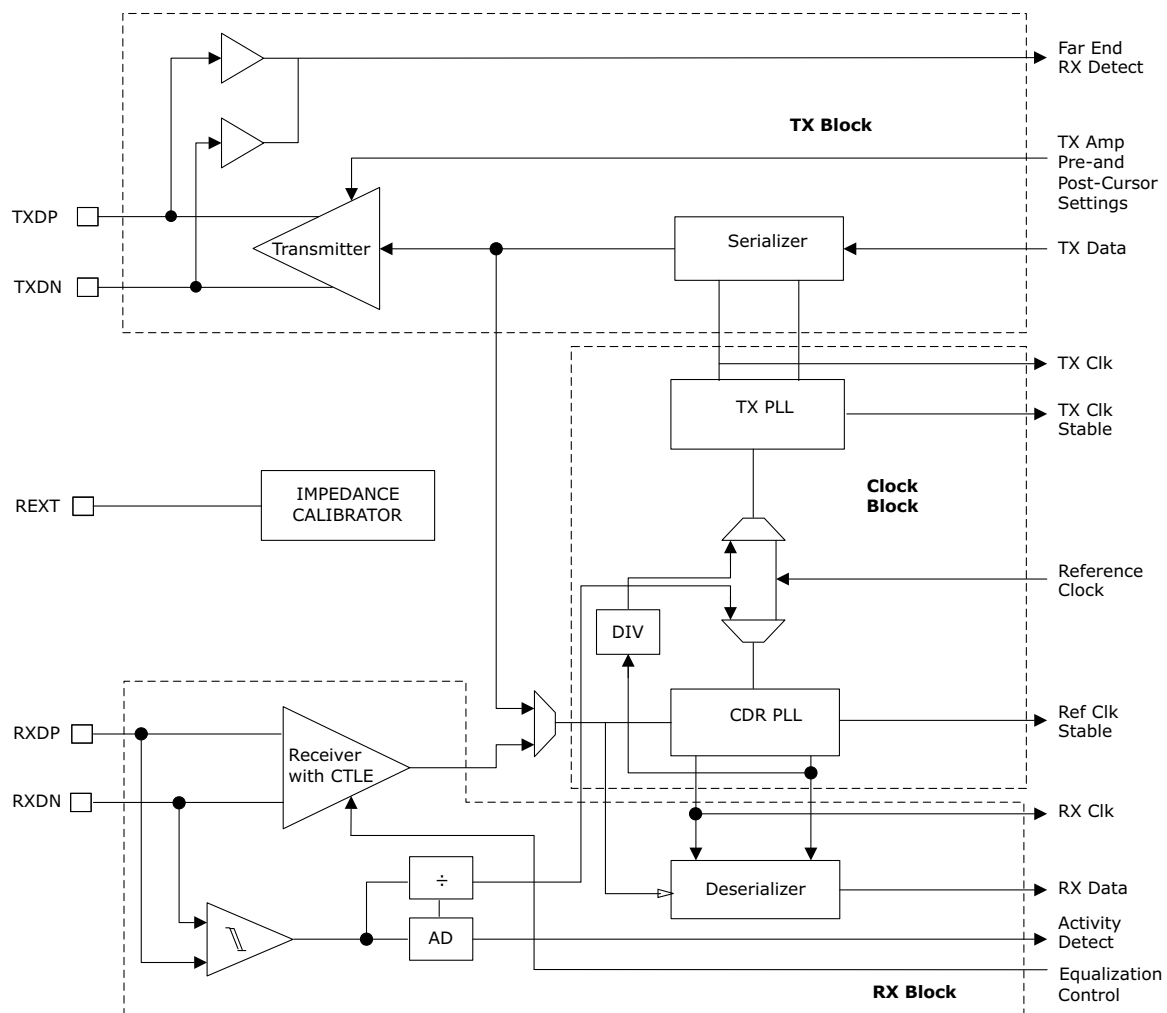
The following are the main features of the PMA:

- Transmit block includes differential impedance matching output buffers, serializer logic, transition de-emphasis, and receiver detection circuitry
- Receive block includes differential current mode logic (CML) input buffers, de-serializer logic, on-chip termination, and continuous-time linear equalization (CTLE)
- Clock block includes clock recovery circuitry and clock management logic
- Configuration control and status register access

The PMA block contains high-speed analog front-end logic, transmit PLL, clock and data recovery (CDR) PLL, calibration, and the voltage offset cancellation mechanism.

The following figure shows a simplified functional block diagram of the PMA block. Each of the PMA blocks includes three sub-functions—Transmit, Receive and Clock block. The three blocks have several primary inputs and outputs as well as control and status connections. The control and status nodes are either ports or registers used in conjunction with the implementation.

**Figure 4 • PMA Block Diagram**



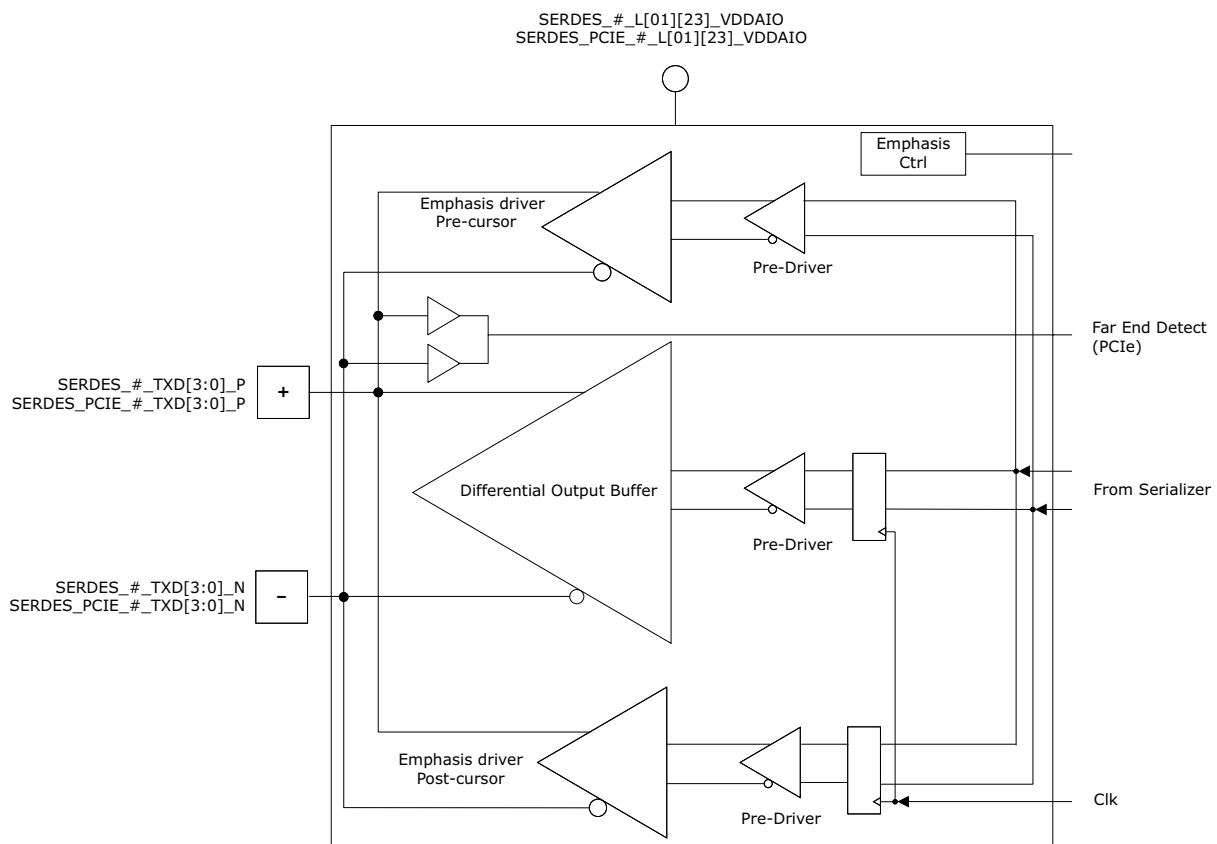
## 2.5.2 Transmit Block

The transmit macro includes a serializer, which receives a 20-bit (maximum) data-word synchronous with a transmit clock, serialized into a single stream of differential transmitted data to the lane. This macro supports multi-level output drive and multi-level transition (pre-and post-cursor) 3-tap de-emphasis while maintaining proper impedance. See the [SerDes Block-I/O Signal Interface](#), page 18 listed in Table 7, page 18 for details. The transmit outputs do not support hot-swap.

### 2.5.2.1 Transmit Output Buffer

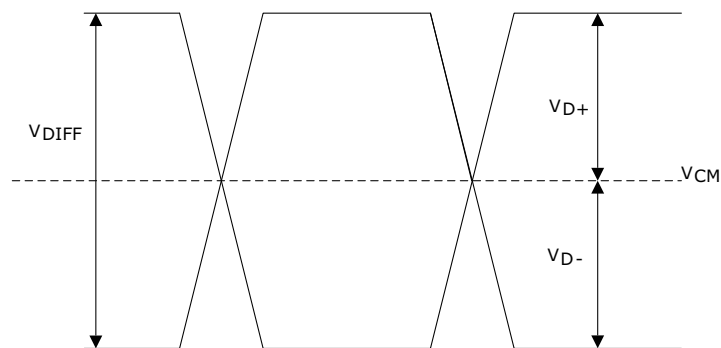
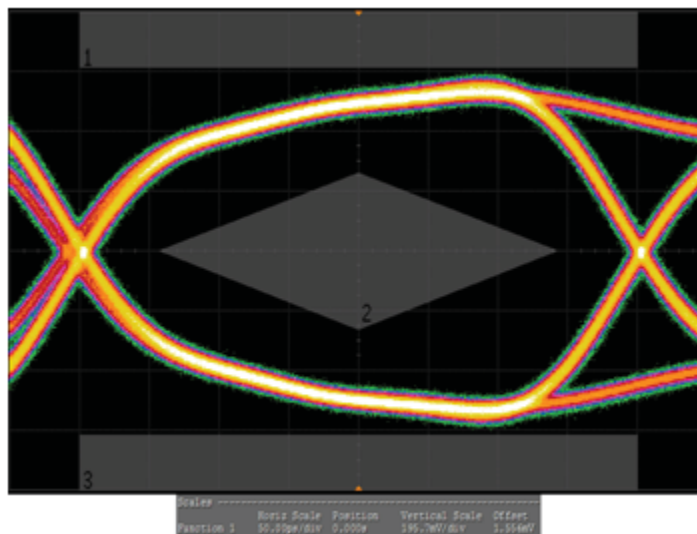
The following figure shows the transmit block with a high-speed and differential impedance matching output buffer. It supports multi-level drive, pre-cursor and post-cursor transition de-emphasis, multi-level common-mode, and calibrated output impedance. The output driver utilizes the current diversion circuitry to maintain the programmable amplitude adjustments. These parameters are predefined by the Libero software, but can be tuned by the designer. The transmitters are suitable for connections to systems using LVDS and HCSL type signaling. See the DC output specifications listed in the [DS0131: RTG4 FPGA Datasheet](#) for detailed output specifications.

**Figure 5 • Transmit Output**



The transmit output voltage levels— $V_{DIFFP-P}$  and  $V_{CM}$ , are nominally set by the Libero software based on key protocol parameters. The limits of these settings are dependent on the analog SerDes I/O supply. The output voltage parameters are defined by the following equations:

- $V_{DIFFP-P} = 2 \cdot (V_{D+} - V_{D-})$
- $V_{CM} = 0.5 \cdot (V_{D+} + V_{D-})$

**Figure 6 • Transmit Output Signal Parameters****Figure 7 • Transmit Output Signal**

The main tap of the transmit output block is programmable and controlled by the TX\_AMP\_RATIO setting. The amplitude can be set as a ratio from 100% (full-swing) to 0%. The ratio is determined as a function of the initial lane calibration. Thus the device calibrates the link and determines an optimized impedance and the ratio setting uses this calibration to baseline the needed amplitude. This allows flexibility to match receiver specifications on the far-end of the link. Valid settings for TX\_AMP\_RATIO are between 0 and 128, where 128 denotes 100% swing.

**Figure 8 • Transmit AMP RATIO Setting as shown in Libero SoC**

LANE0_TX_AMP_RATIO	0x1024	read-write	0x0	0x80	8
TX_AMP_RATIO		read-write	0x0	0x80	[7:0]

The transmit amplitude is adjusted by the TX\_AMP\_RATIO configuration settings. The TX\_AMP\_RATIO settings are controlled by Libero software that adjusts the amplitude in the hardware and includes four predefined settings as shown in the following table.

**Table 3 • Predefined Transmit Amplitude**

TX_AMP_RATIO VALUE (HEX)	Minimum	Maximum	Unit
0A	110.00	130.00	mV
2A	360.00	430.00	mV
55	740.00	850.00	mV
80	1060.00	1270.00	mV

**Note:** The ratio values in the preceding table can vary across devices. See *RTG4 FPGA Datasheet* for lower limits of absolute TX\_AMP\_RATIO\_VALUE  $\times 80$ .

### 2.5.2.2 Transmit De-Emphasis

The signal quality of a physical channel can be adjusted to match the interconnections and PCB using the integrated de-emphasis control. The post-cursor and pre-cursor nominally span from 0 dB to 20 dB. Adjustment to these controls in conjunction with the transmit amplitude allows the user to closely match the interconnect channel.

The pre- and post-cursor de-emphasis adjusts the magnitude of the output based on the prior bit values effectively attenuating the successive bits. This transition emphasis compensates for the channel losses and opens the signal eye at the far-end receiver. The following figure shows the de-emphasis settings and pre- and post-cursor responses of the transmit driver.

The pre-cursor and post-cursor attenuation is a function of the transmit amplitude ratio setting (see TX\_AMP\_RATIO) and the transmit pre- and post-cursor ratio setting.

The pre-cursor attenuation (de-emphasis in dB) is calculated using the formula:

$$\text{dB} = 20 * \log (1 - 2 * \text{precursor ratio})$$

where pre-cursor ratio is simply the TX\_PRE\_RATIO divided by the TX\_AMP\_RATIO.

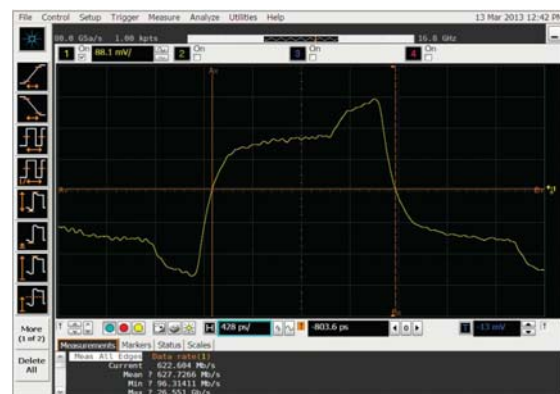
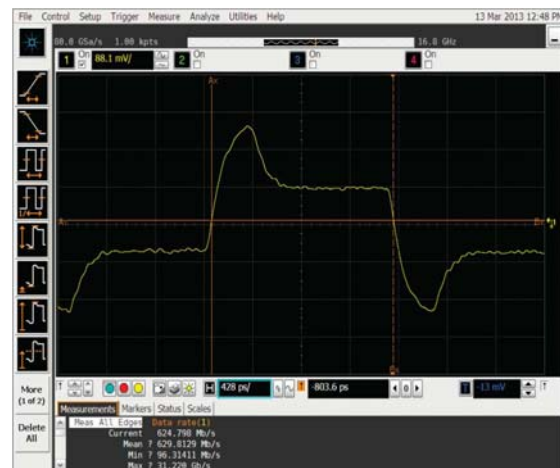
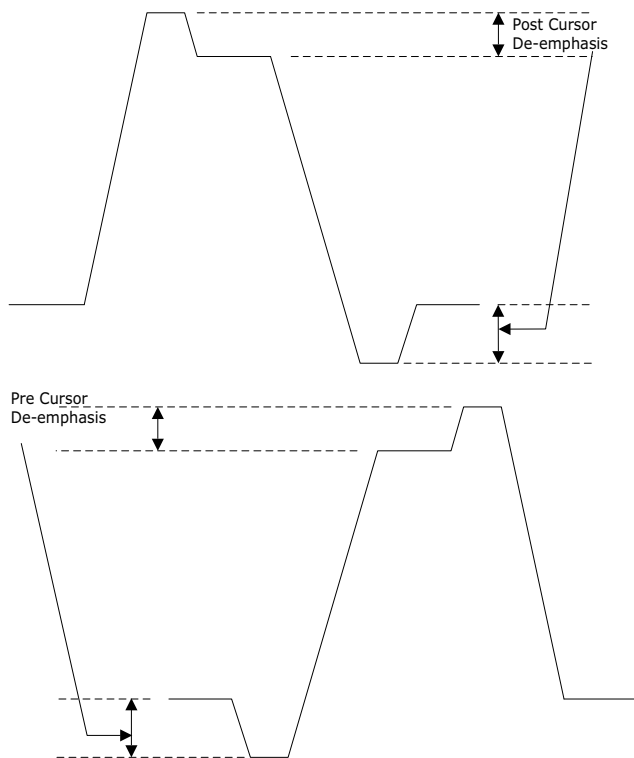
Similarly, post-cursor attenuation (de-emphasis in dB) is calculated using the formula:

$$\text{dB} = 20 * \log (1 - 2 * \text{postcursor ratio})$$

where post-cursor ratio is simply the TX\_PST\_RATIO divided by the TX\_AMP\_RATIO.

For example, when TX\_AMP\_RATIO = 128 and TX\_PST\_RATIO = 21, Post Cursor Attenuation =  $20 * \log(1 - 2 * 21/128)$ . This yields -3.5 dB attenuation.

**Figure 9 • Pre-Cursor and Post-Cursor De-Emphasis**



TX de-emphasis is supported by the Libero configurators for both XAUI and EPCS protocols.

## 2.5.3 Receive Block

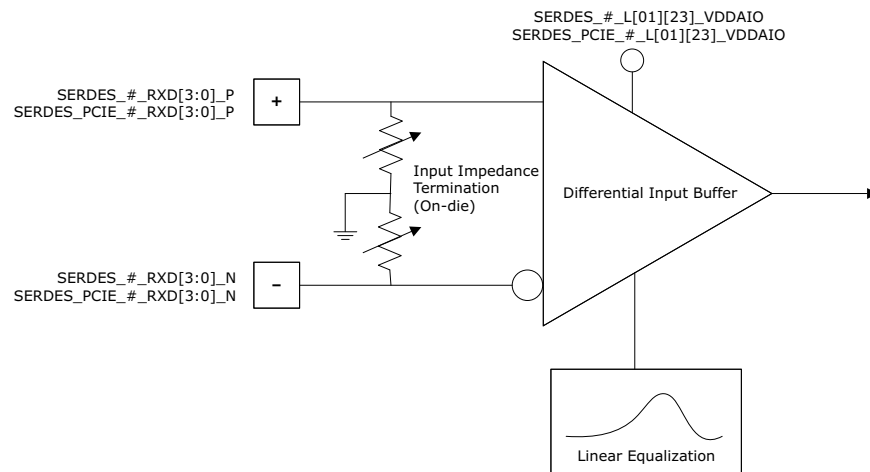
The receive block receives the serialized data from input receivers. The CDR circuit receives the signal and extracts a properly timed bit clock from the data stream. The data signal is deserialized down to a lower speed parallel, 20-bit (maximum) digital data-word (receive data). The deserialized data is synchronous to the recovered link clock (receive clock).

### 2.5.3.1 Receive Input Buffer

The receive block includes an analog front-end powered by the SerDes analog power supply. It contains a current mode, input differential buffer with on-die input impedance control. The input buffer amplifier receives the incoming differential data signal and translates the differential signal to internal logic levels, with no amplitude impairments. The input buffer amplifier rejects the common mode noise. The calibrated input impedance has typical 100  $\Omega$  differential impedance. The input impedance can be configured as per the system requirements. The receive inputs do not support hot-swap.

Jitter on the incoming data signal transfers through this stage, therefore care must be taken to ensure both the incoming timing and amplitude are clean from impairments. The integrated linear equalizer filters extraneous noise from the incoming signals.

**Figure 10 • RX Input Diagram**



### 2.5.3.2 Receive Equalization

The receive block supports a programmable CTLE. The equalizer compensates the attenuated interconnections of the system PCB by effectively using a high-pass filter component, which attenuates the lower frequency components to a degree greater than the higher frequency components. The equalizer circuitry can be tuned to compensate for the signal distortion caused due to the high frequency attenuation of the physical channel of the PCB and interconnect.

The effective receiver equalization compensates for the channel loss of the board with the added frequency response of the CTLE. The frequency response can be programmed to maximize the signal quality of the receiver to achieve the best possible bit-error rate (BER). An under-equalized channel does not adequately open the eye, whereas over-equalization can produce a channel with high jitter. Correct equalization has optimal eye opening with low noise and low jitter.

The receive block has many features that can be tuned dynamically in operation mode. When changing the transmit control registers in real-time, the changes are not updated until the specific UPDATE\_SETTINGS register are written.

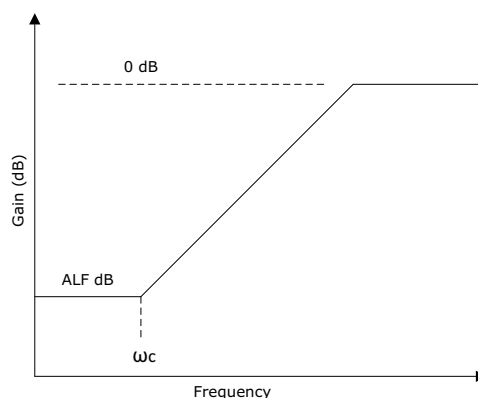
**Figure 11 • RX EQ RATIO Setting as shown in the Libero SoC**

LANE0_RE AMP RATIO	0x101c	read-write	0x0	0x00	8
RE AMP RATIO		read-write	0x0	0x00	[7:0]
LANE0 RE CUT RATIO	0x1020	read-write	0x0	0x00	8
RE CUT RATIO		read-write	0x0	0x00	[7:0]

### 2.5.3.3 CTLE

As a FIR filter, the CTLE operates on the analog input signal and is intended to equalize the incoming transmitted signal and channel by removing the ISI at the receiver. The function of the filter uses both ALF (flat band low-frequency gain) and c (relative frequency of the flat band roll-off) which can be set by the user to optimize the signal quality at the receiver. The Rx Macro can set the desired level of the filter response by setting the hardware registers RE\_AMP\_RATIO and RE\_CUT\_RATIO. The following figure illustrates the response of the CTLE.

**Figure 12 • Receiver Equalization Frequency Response**



The receive equalization is adjusted by the RE\_AMP\_RATIO and RE\_CUT\_RATIO configuration settings. The receive equalization setting is controlled by Libero software that adjusts the settings accordingly in the hardware and using predefined settings as listed in the following XAUI, page 83 and EPCS, page 108 sections.

**Table 4 • Receiver Equalization (CTLE) Settings**

rx_equalization_amp_ratio	rx_equalization_cut_frequency	High-pass Cut-off Frequency	Low-frequency dB Loss of the Filter
RE_AMP_RATIO Setting	RE_CUT_RATIO Setting	Wc (MHz)	ALF (dB) round
103	12	200.00	9.54
74	17	200.00	12.04
57	22	300.00	13.98
47	27	400.00	15.56
40	31	400.00	16.90
119	32	400.00	10.88
119	31	500.00	8.52
103	37	500.00	12.04
35	36	500.00	18.06
103	36	500.00	9.54

**Table 4 • Receiver Equalization (CTLE) Settings**

rx_equalization_amp_ratio	rx_equalization_cut_frequency	High-pass Cut-off Frequency	Low-frequency dB Loss of the Filter
RE_AMP_RATIO Setting	RE_CUT_RATIO Setting	Wc (MHz)	ALF (dB) round
91	42	500.00	13.06
31	41	500.00	19.08
91	41	600.00	10.46
81	47	600.00	13.98
27	46	600.00	20.00
81	46	600.00	11.29
74	52	600.00	14.81
25	50	600.00	20.83
74	50	700.00	12.04
70	55	700.00	15.56
23	55	700.00	21.58
67	55	800.00	12.74
64	60	800.00	16.26
21	60	800.00	22.28
62	60	800.00	13.38
60	64	800.00	16.90
19	64	800.00	22.92
128	69	900.00	11.48
57	64	900.00	13.98
55	69	900.00	17.50
18	69	900.00	23.52
120	74	900.00	12.04
53	69	900.00	14.54
52	74	900.00	18.06
5	245	900.00	24.08
112	79	1000.00	12.57
50	74	1000.00	15.07
15	254	1000.00	18.59
124	69	1100.00	8.20
106	84	1100.00	13.06
15	245	1100.00	15.56
116	74	1100.00	8.67
125	88	1100.00	11.60
35	254	1100.00	13.53
109	79	1200.00	9.12
44	249	1200.00	12.04



**Table 4 • Receiver Equalization (CTLE) Settings**

rx_equalization_amp_ratio	rx_equalization_cut_frequency	High-pass Cut-off Frequency	Low-frequency dB Loss of the Filter
RE_AMP_RATIO Setting	RE_CUT_RATIO Setting	Wc (MHz)	ALF (dB) round
103	83	1200.00	9.54
55	254	1200.00	10.88
34	252	1300.00	9.95
75	254	1400.00	9.17
82	254	1400.00	8.52
95	254	1500.00	7.96
91	254	1500.00	7.47
115	254	1600.00	7.04
120	254	1700.00	6.66
72	254	1800.00	6.02

### 2.5.3.4 AC Coupling

Each lane must be AC-coupled to remove common mode dependencies. However, AC coupling generates baseline wander if the high-speed serial data transmission is not DC-balanced. 8b/10b encoded data is an example of DC-balanced signaling used with PCIe and XAUI protocols. The addition of external capacitors for AC coupling requires careful consideration. The designer must select a capacitor based on the system requirements. The pattern-dependent jitter associated with the low frequency cutoff of the AC coupling network must be minimized. When non-return-to-zero (NRZ) data with long strings of identical 1's or 0's is applied to this high-pass filter, a voltage droop occurs that results in low-frequency, pattern-dependent jitter (PDJ). Off-chip AC coupling requires recommended capacitor values such as 10 nF for 8b/10b XAUI and 75 – 200 nF for PCIe. These example values need to be reviewed based on specific system requirements. See the [AC439: Board Design Guidelines for RTG4 FPGA Application Note](#) for more information.

## 2.5.4 Clock Block

The Clock block in the PMA contains one transmit PLL and one CDR PLL. [Figure 4](#), page 7 shows the overview of the clock block with some associated signals. The transmit PLL and receive CDR use a common input pin "RefClk". This clock signal is sourced from a reference clock input pin or from the FPGA fabric. Fabric clock is only available in EPCS modes. The power supply for the transmit PLL and CDR PLL is supplied from a dedicated 2.5 V supply.

### 2.5.4.1 Transmit PLL and CDR PLL

Each of the PLLs (TX PLL and CDR PLL) contains the necessary dividers, output high frequency (BIT\_CLK), and low frequency (TXClk and RXClk) clocks. The transmit clock (TXClk) and receive clock (RXClk) are divided down into pipelined versions of the high frequency clocks. The transmit and receive clock are complementary. Within a PMA lane, the TX PLL and CDR PLL cannot be independent, that is, the TX and RX uses the same REFCLK within its lane. Designs requiring separate RX and TX need to use an RX from one lane and TX from another lane. The exact frequencies of the clocks are determined by the reference clock (RefClk) and divide ratio settings (M, N, and F). The divide ratio settings—M, N, and F can be programmed from the APB interface on the SerDes block. See the [DS0131: RTG4 FPGA Datasheet](#) for the RefClk operating ranges.

The relationships between FREF (from RefClk clock input), FBaudClock, FBusClock, and bus width are as shown in the following equations:

$$FVCO = (FREF) \times M \times N \times F$$

$$FBaudClock = FVCO/M = FREF \times N \times F$$

$$FBusClock = FBaudClock/N = FREF \times F$$

$$Bus\ width = FBaudClock/FBusClock = N$$

**Note:** FBaudClock in transmit PLL and CDR PLL are the EPCS\_TX\_CLK and EPCS\_RX\_CLK respectively for EPCS mode and bus width is the EPCS bus width.

Transmit clock can only be present at the correct frequency if all the following conditions are true:

- Reference clock is present and at correct frequency
- M, N, and F are correctly set
- Transmit PLL is on
- Transmit clock trees are on (internal)
- Power-down mode is off and initialization is done

The receive clock can only be present at the correct frequency if all of the following conditions are true:

- Reference clock is initially present and at the correct frequency
- M, N, and F are correctly set
- Receive PLL is on, at correct frequency and transmit clock is present
- Serial bitstream is present and valid
- De-serializer circuitry is on
- Receivers are on

**Figure 13 • M, N, and F Settings as shown in the Libero SoC**

LANE0 PLL F PCLK RATIO	0x1010	read-write	0x0	0x1	8
F		read-write	0x0	0x1	[3:0]
DIV_MODE0		read-write	0x0	0x0	[5:4]
RESERVED		read-write	0x0	0x0	[7:6]
LANE0 PLL M N	0x1014	read-write	0x0	0x9	8
N_4_0		read-write	0x0	0x9	[4:0]
M_1_0		read-write	0x0	0x0	[6:5]
CNT250NS_MAX_8		read-write	0x0	0x0	[7:7]

## 2.5.4.2 Reference Clock Inputs

Each SerDes block consists of reference clock input pads. The REFCLK is multiplexed in the clock block. It optionally allows the reference clock to be sourced to the transmit PLL and/or the CDR PLL which are the dedicated input buffers used for the reference clock input. The REFCLK inputs have a dedicated power supply. VDDI powers the SerDes reference clock that is 1.8 V, 2.5 V, or 3.3 V. A common SERDES\_VDDI supply provides the required SerDes reference clocks to use the same voltage supply.

**Table 5 • Reference Clock Specifications (Typical)**

	Min	Max	Unit
Ref Clock Speed	100	160	MHz

See the [DS0131: RTG4 FPGA Datasheet](#) for detailed specifications.

The best performance of the SerDes is based on the selection of a quality clock source. The SerDes reference clock phase noise contributes to the transmit output phase noise and can decrease receiver jitter tolerance. See the [DS0131: RTG4 FPGA Datasheet](#) for more specific information.

### 2.5.4.3 SerDes Reference Clock Inputs

The PMA in the SerDes block needs a reference clock on each of its lanes for transmit and receive clock generation through the PLLs. The reference clock inputs port (REFCLK) is optionally connected to I/O pads or an additional reference clock source, `fab_ref_clk`, coming from the fabric.

Each SerDes block includes four lanes (Lane[0:3]). A dedicated reference clock block (REFCLK) is available for each SerDes block. Two I/O pads are associated with the REFCLK block. The I/O pads can accept only one differential clock signal via the REFCLK\_P and REFCLK\_N pins. The REFCLK block uses both the I/O pads to supply a single REFCLK signal to all four lanes of the SerDes block. However, an optional feature is available to supply two individual single-ended clock signals to the REFCLK\_IO[0:1] I/O pads. This feature provides two REFCLKs to the SerDes block and permits independent clocking of two lanes. In this scenario, Lane 0 and Lane 1 share one of the clock inputs, and Lane 2 and Lane 3 share the other clock input. The REFCLK is controlled by selection of the input multiplexer to the REFCLK input of the SerDes block. The Libero software correctly sets the input MUXes for the REFCLKs based on the SerDes block configurator settings.

The RTG4 SerDes device includes a multi-standard input reference clock. The reference clock is configured through the Libero SoC software configurator to support differential or single-ended signaling standards. Based on the reference clock source selection, the supported I/O standards and other options such as impedance, receiver settings, weak pull-up/pull-down settings are available.

**Table 6 • Support Reference Clock Input Standards**

SERDES_VDDI Supply	3.3 V	2.5 V	1.8 V
Supported Standards	LVTTL/LVCMOS33	LVC MOS25	LVC MOS18
	LVDS33 <sup>1</sup>	LVDS25 <sup>1</sup>	SSTL18-Class 1
	LVPECL	RS DS	SSTL18-Class 2
	RS DS	mini-LVDS	HSTL18-Class 1
	mini-LVDS	SSTL25-Class 1	–
	–	SSTL25-Class 2	–

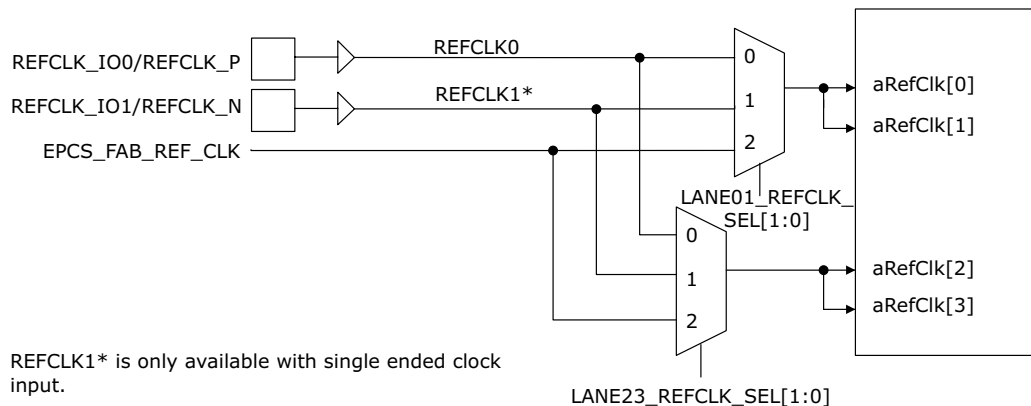
- For LVDS33 and LVDS25, designers should reference the RTG4 I/O Users Guide and [DS0131: RTG4 FPGA Datasheet](#) for correct termination and common-mode recommendations to achieve optimal jitter performance. For more information about application solution, see [Reference Clocks for RTG4 SerDes REFCLK Inputs and Interface Circuits Application Note](#).

**Note:** HCSL inputs are supported directly with LVDS I/O STD inputs from the Libero. There is no specific HCSL I/O STD available in Libero and designs requiring HCSL are supported by using the LVDS25 I/O standard.

The dedicated clock input pins are recommended to achieve optimal performance. The fabric reference clock is only available for EPCS modes. The following figure shows the reference clock selection. The user programmed clock selection is routed to the SerDes clock block RefClk input port.

The SerDes has four lanes, the two adjacent SerDes lanes share the same reference clock, as shown in the following figure. In this scheme, lane0 and lane1 share the same reference clock input. Similarly, lane3 and lane4 share the same reference clock input.

**Figure 14 • SerDes Reference Clock Input Sources**

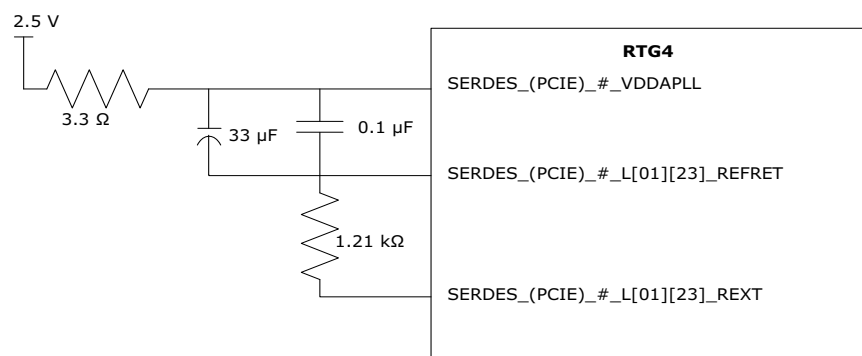


## 2.5.5 Calibration Resource Sharing

The SerDes PMA block calibration is performed to optimize the SerDes in the system. Calibration is done for the source impedance at the transmitter and at the termination of the receiver. The calibration circuitry is shared across channels. Each SerDes block contains two external reference resistor (REXT) signals—one for lane0 and lane1 and another for lane2 and lane3. The calibration interaction limits the combination of protocols/data rates per channel utilization since the adjacent channels are bonded to the same calibration circuitry. For example, if lane1 and lane0 operate and the PHY is reset on lane0, the recalibration function, which follows the reset disrupts lane1 due to the shared REXT calibration resistor.

REXT connections are required to calibrate transmit/receive termination value and internal elements. A 1.21 k-1% resistor must be connected on the PCB, as shown in the following figure. This resistor can be a 0201 or 0402 sized component, since the power dissipation through this resistor is less than 1 mW during calibration. See the [AC439: Board Design Guidelines for RTG4 FPGA Application Note](#) for more details.

**Figure 15 • Calibration Resistor Connection**



### 2.5.5.1 SerDes Startup

For SerDes to lock at power up, it is essential that the reference clock to the SerDes be at the target frequency and stable. If the clock frequency is lower or higher than the target, the SerDes link acquisition is not guaranteed or reliable. SERDES\_VDDI of the reference clock should be at its minimum or higher. Any lower SERDES\_VDDI causes higher jitter and unpredictable locking. At system startup, the user design should hold the PCS/PMA in reset until the clock stabilizes at its target frequency. The reset signals are listed for PCIe, XAUI, and EPCS modes depending on target protocols. See [Physical Coding Sublayers](#), page 20 for information about reset pin descriptions. If the PLL does not lock, a simple reset may not be sufficient and a power down (using the protocol mode related SerDes power down pin) may be required.

In EPCS modes, after power up and the fabric releases the EPCS\_#\_RESET\_N starting the Tx PLL locking process. The fabric output port, EPCS\_#\_TX\_CLK\_STABLE, is the Tx PLL lock flag. A rise on this flag begins the SerDes calibration operation. Calibration time is dependent on the transmit parallel clock. Calibration determines the PMA's optimized output impedance and receiver termination that best matches the system. The end of calibration triggers the EPCS\_#\_READY output port to rise indicating to the fabric that the transceiver is ready for operation. Serial port outputs are held at a common-mode throughout calibration. Upon completion of calibration, the transmitter is taken out of electrical-idle. This is a serial shift operation, which adds 27 parallel transmit clock cycles to complete. After which, the serial outputs reflect the applied EPCS\_#\_TX\_DATA pattern. When the serializer sends out differential data, it takes some time for the line to charge up because of capacitive coupling.

In PCIe or XAUI modes, the protocol layer manages the health and proper sequencing of the startup requiring no user logic intervention.

### 2.5.6 SerDes Block-I/O Signal Interface

The RTG4 SerDes block interfaces with differential I/O pads, the PCIe system, and the FPGA. The following table describes these signals.

**Table 7 • SerDes Block I/O - PAD Interface**

Port Name	Type	Connected to	Description
SERDES_(PCIe)_#_RXD0_P	Input	Input Pads	Receive data. SerDes differential positive input: Each SerDes block consists of 4 RX+ signals.
SERDES_(PCIe)_#_RXD1_P			
SERDES_(PCIe)_#_RXD2_P			
SERDES_(PCIe)_#_RXD3_P			
SERDES_(PCIe)_#_RXD0_N	Input	Input Pads	Receive data. SerDes differential negative input: Each SerDes block consists of 4 RX- signals.
SERDES_(PCIe)_#_RXD1_N			
SERDES_(PCIe)_#_RXD2_N			
SERDES_(PCIe)_#_RXD3_N			
SERDES_(PCIe)_#_TXD0_P	Output	Output Pads	Transmit data. SerDes differential positive output: Each SerDes block consists of 4 TX+ signals.
SERDES_(PCIe)_#_TXD1_P			
SERDES_(PCIe)_#_TXD2_P			
SERDES_(PCIe)_#_TXD3_P			
SERDES_(PCIe)_#_TXD0_N	Output	Output Pads	Transmit data. SerDes differential negative output: Each SerDes block consists of 4 TX- signals.
SERDES_(PCIe)_#_TXD1_N			
SERDES_(PCIe)_#_TXD2_N			
SERDES_(PCIe)_#_TXD3_N			

Port Name	Type	Connected to	Description
SERDES_(PCIE)_#_L01_REXT SERDES_(PCIE)_#_L23_REXT	Reference	Reference Pads	External reference resistor connection to calibrate TX/RX termination value. Each SerDes block consists of 2 REXT signals—one for lanes 0 and 1 and another for lane2 and lane3.
SERDES_(PCIE)_#_REFCLK_P SERDES_(PCIE)_#_REFCLK_IO0	Input	Input Pads	This pin acts as the positive terminal when used with a differential clock source. It is limited to driving only one clock source to the SerDes block when differential (REFCLK0) is used.
SERDES_(PCIE)_#_REFCLK_N SERDES_(PCIE)_#_REFCLK_IO1	Input	Input Pads	This pin acts as the negative terminal when used with a differential clock source. It is limited to driving only one clock source to the SerDes block when differential (REFCLK0) is used. When used with a single ended clock source, this input pin supplies a reference clock signal to the REFCLK1 input of the SerDes block.

**Note:** Here, x = the SerDes Block\_# where # varies from 0 through 5 based on the device size. The notation SERDES\_(PCIE) indicates the capability of a SerDes block to include PCIe functionality. SerDes without PCIe capability is represented without PCIe in the pin name. For example, SERDES\_3.

**Note:** For more information on SerDes pins and related power supplies, see the [DS0130: RTG4 FPGA Pin Descriptions](#).

## 2.6 PCB Design

Good board design practices are required to achieve expected performance from the PCB and RTG4 devices. High quality and reliable results depend on minimizing noise levels, preserving signal integrity, meeting impedance and power requirements, and using appropriate transceiver protocols. A good understanding of the RTG4 chip, experienced in digital and analog board layout, knowledge of transmission line theory and signal integrity are essential to build high performance PCBs. See the [AC439: Board Design Guidelines for RTG4 FPGA Application Note](#), which describes the details of PCB design for RTG4 based designs.

IBIS-AMI models recommended to simulate serial links to predict a serial link performance with an eye diagram and bit error rate. IBIS-AMI models are available for download from the Microsemi website. AC292 Application Note IBIS/IBIS-AMI Models: Background and Usage provides information for using the equalization models to optimize the design of PCB channels with the RTG4 SerDes blocks.

## 3 Physical Coding Sublayers

---

The RTG4 contains three embedded SerDes PCS blocks (PCIE, XAUI, and EPCS). These blocks are user configured in the design. The embedded PCIE and XAUI blocks support protocol required functions, without the need of FPGA hosted PCS logic. EPCS protocols require the user to include soft FPGA hosted logic to perform PCS functions.

### 3.1 PCI Express

PCIe is a high-speed, packet based, point-to-point, low pin count, and serial interconnect bus. RTG4 has a fully integrated PCIe end point (EP) implementation within the SerDes block. This section describes the architecture of the PCIe system that implements the main PCIe IP function.

The EP in PCIe refers to a type of function that can be the requester or the completer of a PCIe transaction. The PCIe sub-system (PCIESS) within the SerDes blocks implements the PCIe EP specification for transaction, data link, and physical layers. On the application side, it has one master interface and one slave interface. The master interface can be a 64-bit AXI3 master. The slave interface can be a 64-bit AXI3 slave. The PCIe link initiates transactions to the RTG4 fabric through the AXI3 master. RTG4 fabric initiates transactions towards the PCIe link through the AXI3 slave. There is an APB interface that has access to the SerDes block system registers.

#### 3.1.1 Features

The following are the main features of the PCIe EP implemented in the RTG4:

- x1, x2, or x4 lane support
- Implements native endpoint
- Compliant with PCIe base specification revision 2.0 and 1.1 for 2.5 Gbps operation
- Single-function/Single virtual channel (VC)
- Receives, transmits, and retries buffer
- AXI3 master and slave interface to the RTG4 FPGA fabric
  - AXI3 64-bit master and slave interfaces
- Advanced error reporting (AER) support
- End-to-end cyclic redundancy check (ECRC) generation and forward support
- Supports all base memory, configuration, and message transactions
- Implements type 0 configuration space for EP
- Supports three 64-bit BARs or six 32-bit BARs
- Native active state power management L0, L1, and L2 state support
- Power management event (PME) message
- Supports design time selection of the PCIe lane reversal for flexibility of lane assignments for board layout.

**Note:** A PCIe endpoint refers to the location of the connection in the PCIe topology. A PCIe endpoint can connect to switches downstream port or a root complex downstream port. As an endpoint, the PCIe can initiate and respond to transactions in the system.

### 3.1.2 Device Support

The following table shows the total number of PCIe endpoint blocks available in each RTG4 device that can be configured to support PCIe.

**Table 8 • PCIe Endpoint Blocks Available in RTG4**

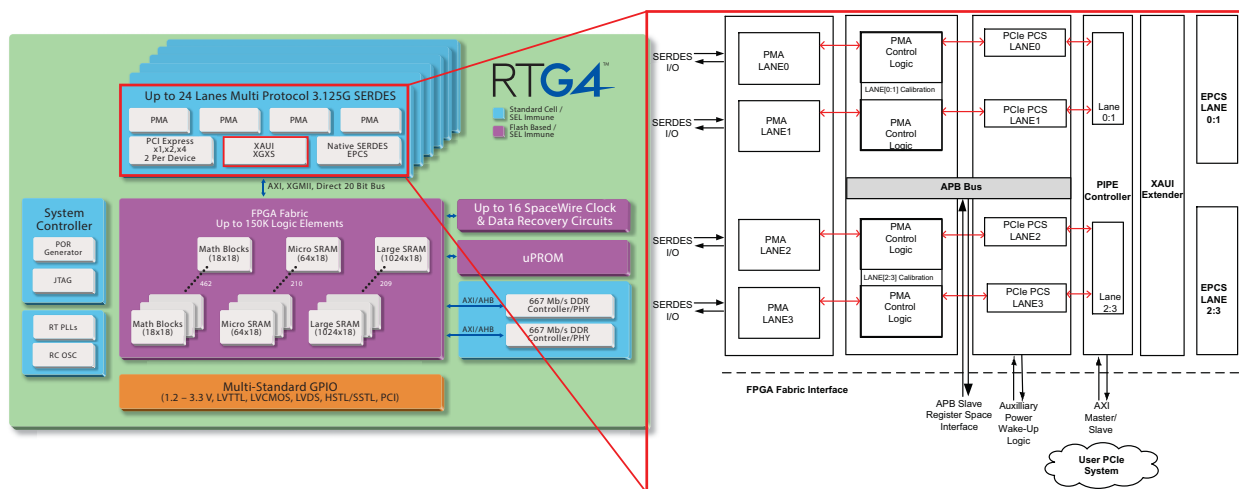
RT4G150	
PCIe EP available	2

- The specified number of PCIe EP blocks vary depending on the device package. PCI ESS can implement x1, x2, or x4 links.
- Each RTG4 device has a minimum of 2 PCI ESS endpoint blocks that allow 2 PCIe variations within an RTG4 device.

### 3.1.3 RTG4 PCI Express Block

The RTG4 family supports up to six integrated SerDes blocks in one device, and each block supports up to 4 SerDes lanes, thus supporting up to 24 SerDes lanes. The following figure shows the RTG4 device block diagram with PCIe implementation. Each PCI ESS block contains an integrated PCIe system block, also known as a PCIe system, which implements the PCIe transaction layer and data link layer. The PCI ESS block also has a SerDes PMA block that implements the physical layer. The RTG4 family includes two PCI ESS blocks per device. The PCIe system block along with the SerDes PMA block provides the complete PCIe EP solution in RTG4. The RTG4 PCIe implementation is compliant with PCIe base specification revisions 2.0 and 1.1.

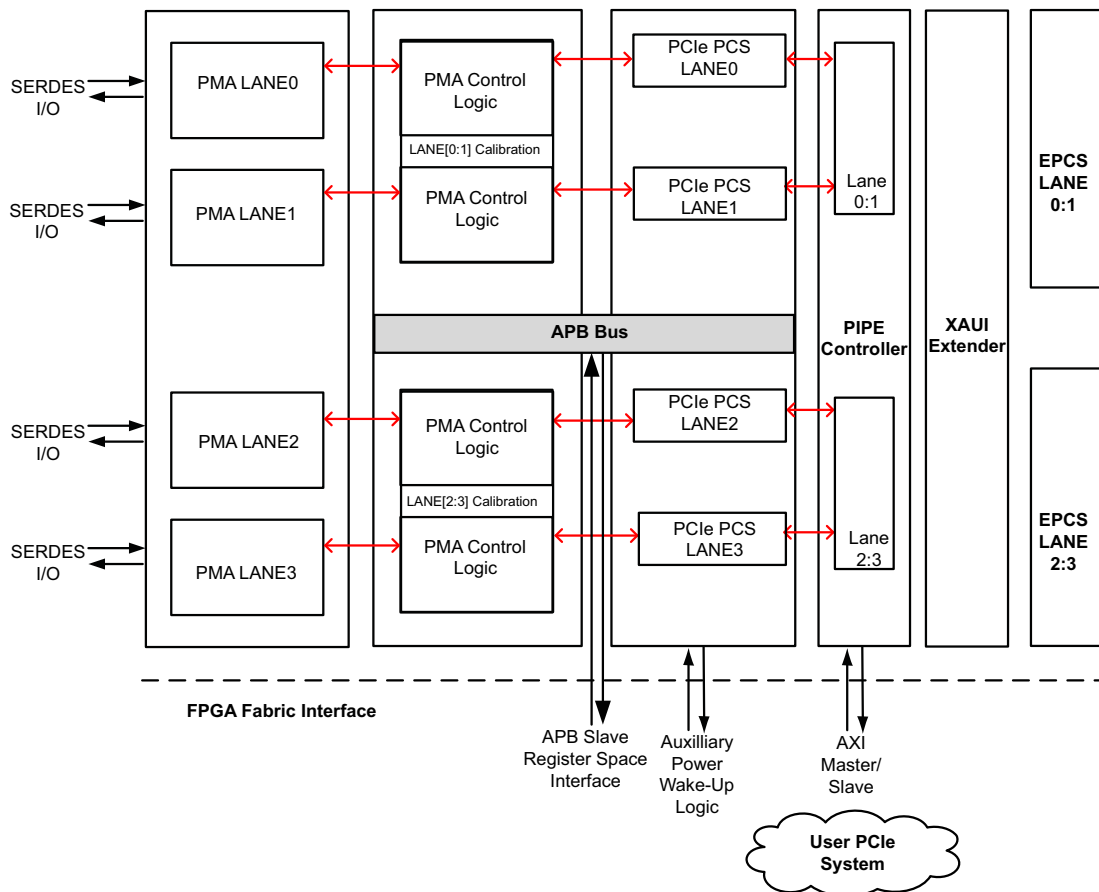
**Figure 16 • RTG4 SerDes Block Diagram**





The following figure shows a simplified view of a PCIe EP implementation in an RTG4 device. The PCIe system interfaces to the FPGA fabric on one side and the PCS/PMA SerDes block on the other side. The SerDes block interfaces to the dedicated I/O in the RTG4 device is called a SerDes I/O. Refer to the 5-SerDes Block I/O - PAD Interface for more information. The PCIe system interface to the FPGA fabric consists of an application interface and a configuration interface. In addition, it has several signals for clocking, reset, and power management.

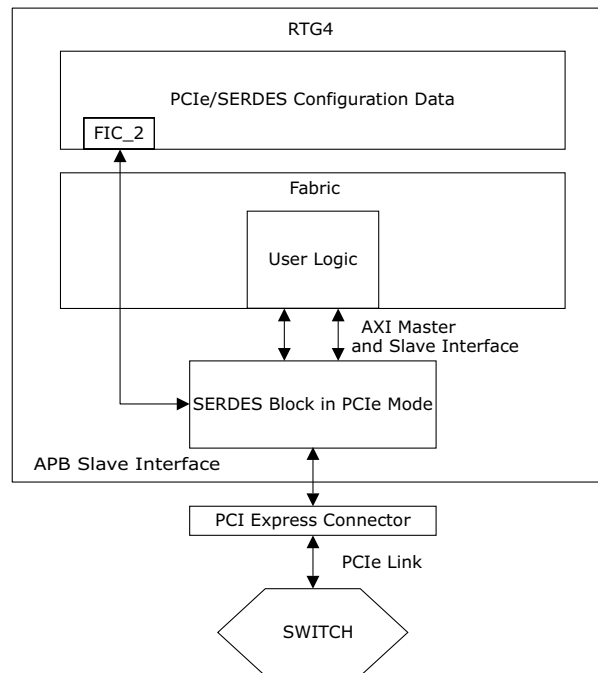
**Figure 17 • SerDes Block Configuration for PCIe Single Protocol Mode**



- **Application Interface:** The application interface is used to transfer transaction layer packets (TLP). It can be AXI3 master only, AXI3 slave only, or AXI3 master plus slave interface.
  - **AXI3 Master Interface:** The master interface can be a 64-bit AXI3 master. A typical application interface uses a master interface which is used to respond to data read requests and a slave interface which is used to initiate requests. It is also possible to use a master and/or the slave interface by itself for specific applications.
  - **AXI3 Slave Interface:** The slave interface can be a 64-bit AXI3 slave interface. The RTG4 fabric initiates PCIe transactions using the slave interface (that is, memory write TLP and memory read TLP). The data on a read request comes back to the same interface.
- **Configuration Interface:** The configuration interface uses the APB slave interface.
  - **APB Interface:** The APB interface has access to various registers, including PCIe configuration registers, AXI3 bridge register, SerDes block register and so on. The APB provides access to the memory map of the SerDes block, which includes a section for the PCIe controller.
- **Other Signals:** The PCIe system has several clocking signals, reset signals, phase-locked loop (PLL) signals, interrupts, and power management signals to the FPGA fabric.

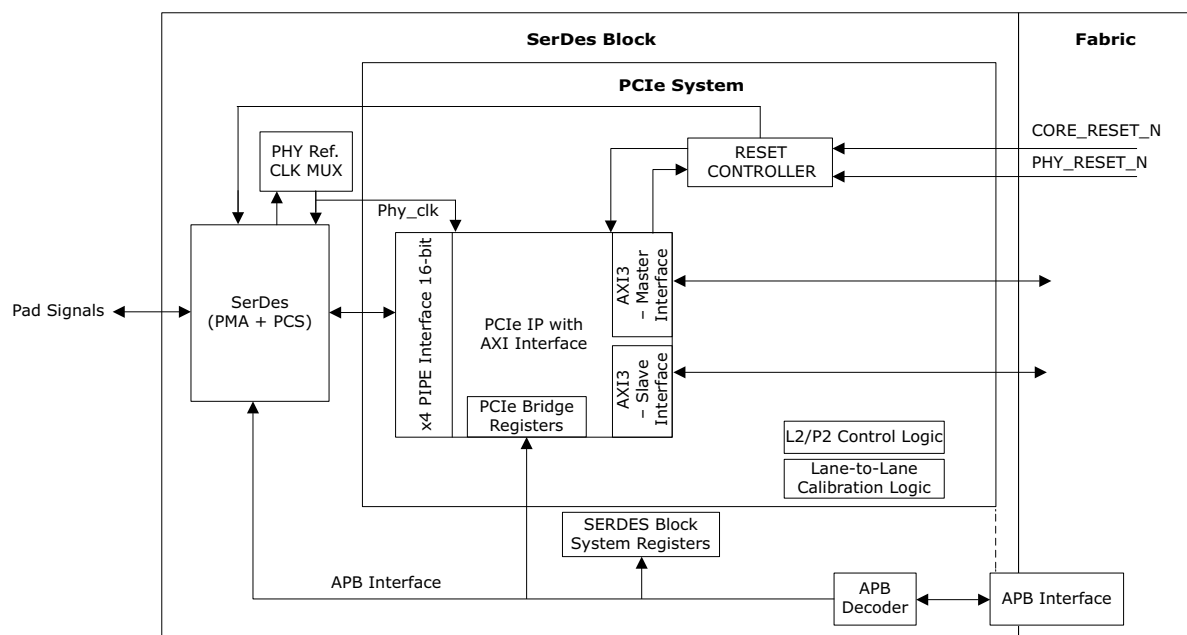
The following figure shows a simple application with a switch port connected to a PCIe EP implemented in an RTG4 device.

**Figure 18 • RTG4 PCIe EP Implementation**



The PCIe system sub-block inside the SerDes block implements the PCIe physical layer, data link layer, and transaction layer of the PCIe specification. It interfaces with the SerDes block on one side and the FPGA fabric on the other side. The following figure shows the RTG4 SerDes block in PCIE mode and various sub-blocks for the PCIe system block.

**Figure 19 • Detailed PCIe System Block Diagram**



### 3.1.3.1 AXI Master Block

The AXI3 master only supports memory read and write transactions.

#### 3.1.3.1.1 AXI Master Write Transaction Handling

- The write transaction is handled in big-endian order
- PCIe transactions can be of any size up to the configurable maximum payload size (256 bytes)
- AXI3 transactions are limited to 128 bytes, a received TLP is divided in to several AXI3 transactions.
- AXI3 master receives a write transaction, it processes the transaction as 128-byte segments (aligned on a 128-byte address boundary) until the segments in the transaction have been processed
- TLP is de-constructed and sent to the AXI3 interface, and the data is presented as little endian

#### 3.1.3.1.2 AXI Master Read Transaction Handling

- Read transactions are handled the same way as write transactions, except that before transferring the transaction to the AXI3 master read channel, the PCIe IP checks the transmit buffer for available space
- PCIe IP does not transfer the read transaction if there is no sufficient space in the transmit replay buffer to store PCIe completions
- The number of outstanding AXI3 master read transactions is therefore limited by the size of the Tx buffer
- The AXI3 master read channel can receive transactions in any order, and data can be completely interleaved. However, the PCIe IP generates completions in the order they are initiated on the link

### 3.1.3.2 AXI Slave Block

The AXI3 slave interface forwards AXI3 read and write requests from the FPGA fabric to the PCIe link.

#### 3.1.3.2.1 AXI Slave Write Transaction Handling

- Minimum 128 Bytes must be available for write transaction
- Data interleaving is not supported
- Wait states are used if buffer is full or has less than 128 Bytes of space available
- Write responses are generated as soon as the last data phase is over
- Maximum of 128 Bytes of data packet can be created
- Only four outstanding write transactions are supported
- Incrementing-address burst is supported

#### 3.1.3.2.2 AXI Slave Read Transaction Handling

- Minimum 128 Bytes must be available for read transaction
- PCIe IP generates a PCIe tag, arbitrates between write requests and completions, and then checks for available flow control (FC) credits
- Response is generated if a time-out occurs or if a completion with error status is received

#### Outstanding Requests

The AXI3 interface supports the following outstanding requests as listed in the following table.

**Table 9 • AXI3 and Outstanding Transactions**

AXI3 Transaction	Outstanding Transactions
Master Write	Limited by Tx Credits
Master Read	4
Slave Write	Limited by Rx Credits
Slave Read	4

### 3.1.3.3 AXI3 Transaction and TLP Ordering Rules

This section describes the TLP ordering rules for sending and receiving TLPs. Both AXI3 interfaces, master and slave can multiplex the transmit buffer to send packets over the PCIe link. There are priority ordering rules in PCIe which mandate the scheduling of packets and this is followed by the PCIe IP block in the case of a collision.

#### 3.1.3.3.1 AXI3 Slave Interface

The slave path does not reorder transactions other than reordering using PCIe standard ordering rules, but does arbitrate between transactions when they occur simultaneously. The order of priority for arbitrations is master read completions, slave write requests, and then slave read requests.

#### 3.1.3.3.2 AXI3 Master Interface

The master path does not reorder transactions other than reordering using PCIe standard ordering rules, but does arbitrate between transactions at the AXI3 master interface. If a transaction is currently waiting for a response phase, the transaction is completed before the read transaction that is forwarded to the AXI3 master interface.

#### 3.1.3.4 APB Slave Interface

The APB slave interface provides APB interface to SerDes block system registers.

#### 3.1.3.5 AXI3 to AHBL Bridge (Master/Slave)

There are two AHBLite IP cores available in the Libero SoC catalog to support AXI to AHBL (master) and AHBL to AXI (slave) between the SerDes block configured with an AXI3 interface and AHB hosted within the FPGA fabric. The two Direct Cores from the Libero SoC catalog are specifically developed to bridge PCIE AXI3 to AHBL transactions. RTG4 designs requiring SerDes configured as PCIe with AHBLite master should use the CorePCle\_AXItoAHBL DirectCore.

In addition, designs with both master and slave AHBLite interfaces are required to use the CorePCle\_AXItoAHBL and CorePCle\_AHBLtoAXI DirectCores. The Core IP Handbooks provides the implementation guidance.

#### 3.1.3.6 PCIe Clocking Architecture

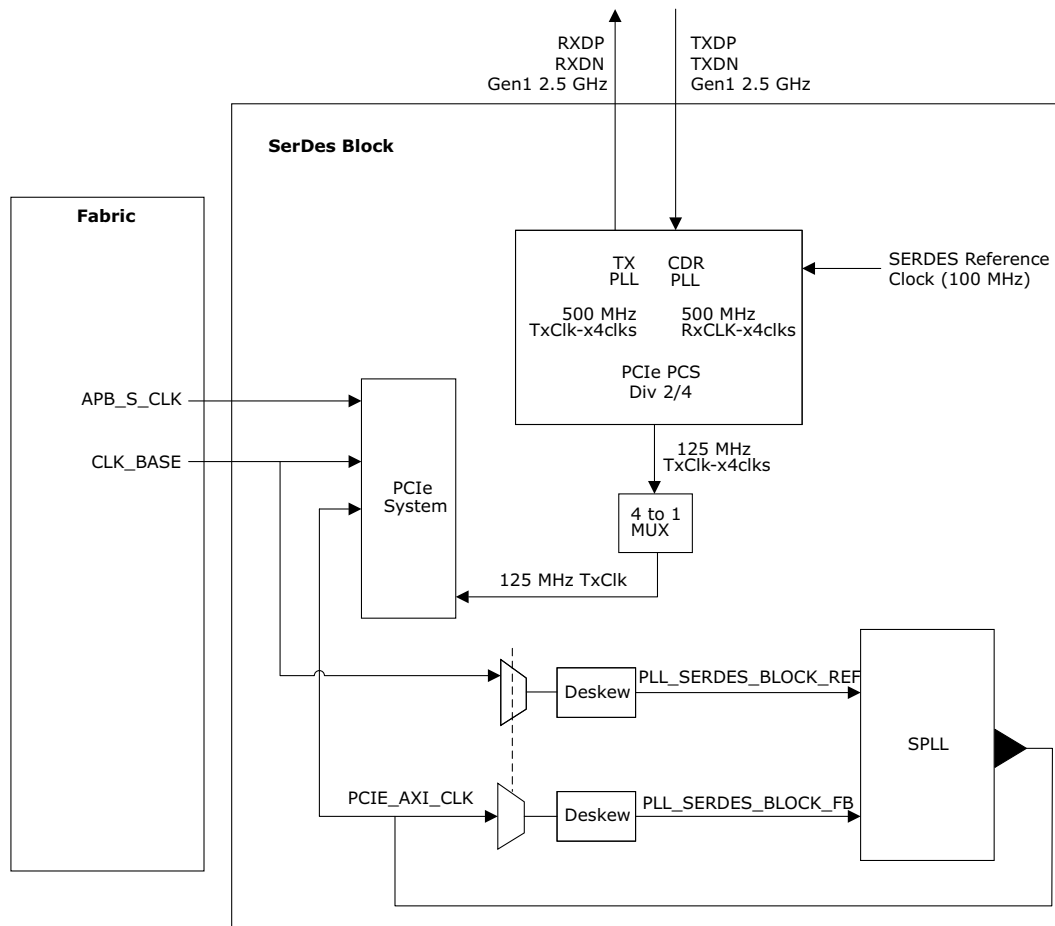
The RTG4 SerDes block, when configured in PCIe mode, uses multiple clocks inside the SerDes block. This sub-section describes the PCIe clocking architecture inside SerDes block in PCIe mode. [Figure 20](#), page 26 shows the PCIe clocking architecture in the RTG4 device. The two main clock inputs are a differential SerDes reference clock (100 MHz) for SerDes PMA, and a CLK\_BASE input for SerDes block from the FPGA fabric. In addition, there is a APB clock input for SerDes block from the FPGA fabric.

**SerDes reference clock:** The differential 100 MHz reference clock is used by SerDes (TX PLL and CDR PLL) to generate 125 MHz clock and is passed to PCIe System IP block. The setting for TX PLL and CDR PLL are calculated automatically by the Libero software. This 125 MHz clock output from SerDes is used by PCIe system.

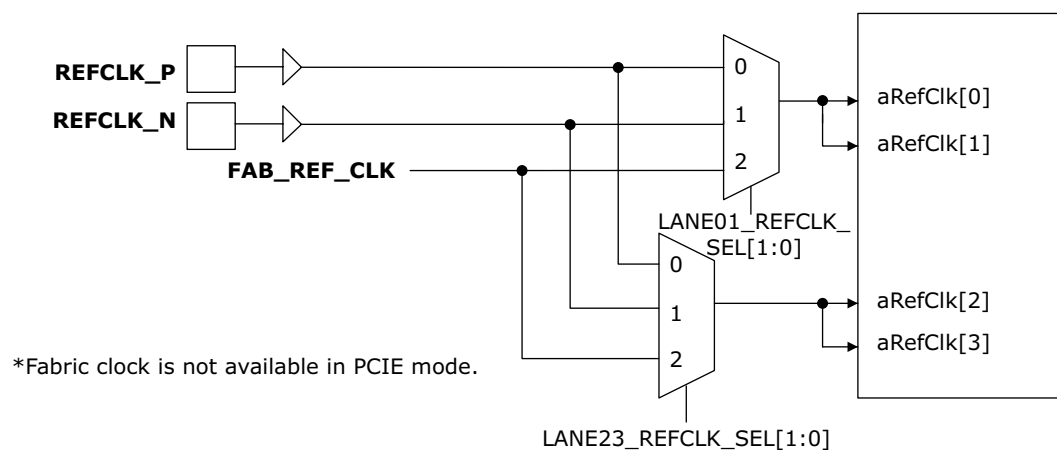
The PCIe standard specifies a 100 MHz clock (Refclk) with greater than  $\pm 300$  ppm frequency stability at both the transmitting and receiving devices. RTG4 supports two distinct clocking topologies: Common Refclk and Separate Refclk.

Common Refclk is the most widely supported clocking method in open systems where the host (switch or root) provides a clock to the end point. An advantage of this clocking architecture is that it supports spread spectrum clocking (SSC) which is useful in reducing electromagnetic interference (EMI). RTG4 supports SSC clocking in common clock systems.

Separate Refclk uses two independent clock sources. One clock for the host (switch or root) and another clock source for the endpoint. The clock sources must have  $\pm 300$  ppm frequency accuracy and cannot use any SSC.

**Figure 20 • Clocks in PCIe Mode**

The clocking architecture also uses SPLL to synchronize data between CLK\_BASE and the clock generated from SerDes (125 MHz clocks). The SPLL allows the reduction of the skew between the fabric and the RTG4 SerDes block module.

**Figure 21 • SerDes Reference Clock for PCIe Mode**

### 3.1.3.6.1 SerDes Reference Clocks Selection

RTG4 accepts the PCIe reference clock. It is fully compliant with the PCI Express add-in card specifications and can directly receive the 100 MHz reference clock from the PCI Express Connector. The differential PCIe clock is connected to the REFCLK\_P and REFCLK\_N pins and configured correctly by the Libero software.

**Table 10 • Reference Clock Signals for SerDes in PCIe Mode**

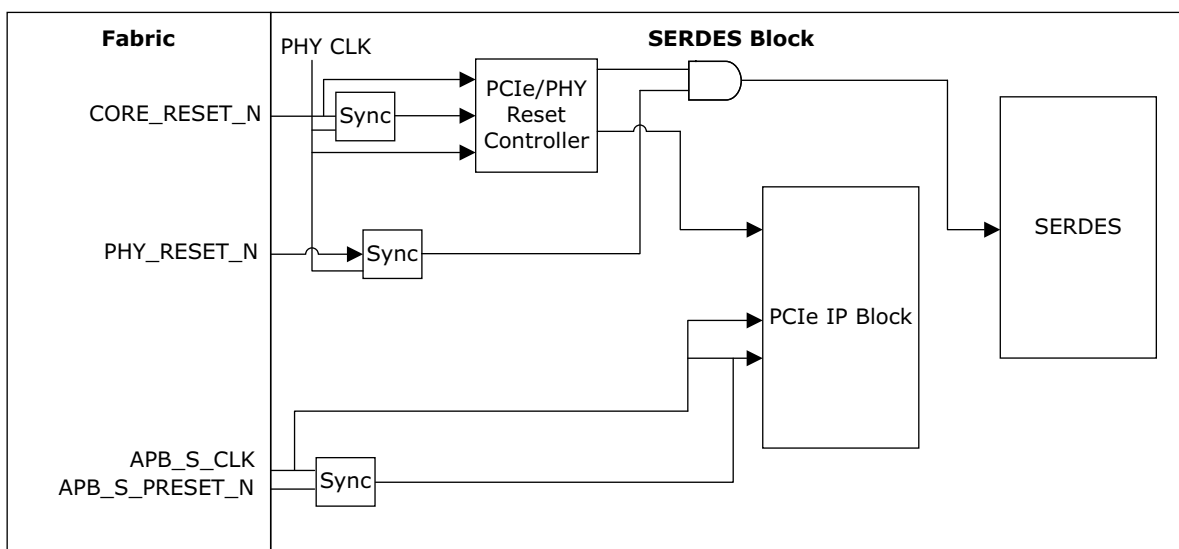
Clock Signal	Description
REFCLKP, REFCLKN	Reference clock input for SERDES_(PCIE)_#_REFCLK_P, SERDES_(PCIE)_#_REFCLK_N

The reference clock needs to be compliant with the PCIe protocol. Refer to the [DS0131: RTG4 FPGA Datasheet](#) for the specification. The Libero software permits using only REFCLK\_P or REFCLK\_N for PCIe mode.

### 3.1.3.7 PCIe Reset Network

SerDes block has different reset inputs when configured in the PCIe mode. The reset connections are required for a proper PCIe SerDes initialization solution. To build the initialization circuitry, see [RTG4 High Speed Serial Interface \(PCIe, EPCS and XAUI\) Configuration User Guide](#). The pcie\_init component has a one for one connection for the fabric signals, as shown in [Figure 22](#), page 27.

**Figure 22 • Reset Signals in PCIe Mode**



### 3.1.3.8 Fabric Interface for PCIe Block

The RTG4 PCIe block interfaces with the FPGA fabric using the following interface pins:

- PCIe system AXI3 master interface
- PCIe system AXI3 slave interface
- PCIe system APB slave interface
- PCIe system clock signals
- PCIe system reset signals
- PCIe interrupt and power management interface

**Table 11 • PCIe System AXI3 Master Interface**

Port	Type	Description
AXI_M_AWID[3:0]	Output	AXI3 master mode: AWID (not supported)
AXI_M_AWADDR[31:0]	Output	AXI3 master mode: AWADDR
AXI_M_AWLEN[3:0]	Output	AXI3 master mode: AWLEN
AXI_M_AWSIZE[1:0]	Output	AXI3 master mode: AWSIZE
AXI_M_AWBURST[1:0]	Output	AXI3 master mode: AWBURST
AXI_M_AWVALID	Output	AXI3 master mode: AWVALID
AXI_M_AWREADY	Input	AXI3 master mode: AWREADY
AXI_M_WID[3:0]	Output	AXI3 master mode: WID (not supported)
AXI_M_WSTRB[7:0]	Output	AXI3 master mode: WSTRB
AXI_M_WLAST	Output	AXI3 master mode: WLAST
AXI_M_WVALID	Output	AXI3 master mode: WVALID
AXI_M_WDATA[63:0]	Output	AXI3 master mode: WDATA
AXI_M_WREADY	Input	AXI3 master mode: WREADY
AXI_M_BID[3:0]	Input	AXI3 master mode: BID
AXI_M_BRESP[1:0]	Input	AXI3 master mode: BRESP. In response to a write the value is ignored.
AXI_M_BVALID	Input	AXI3 master mode: BVALID
AXI_M_BREADY	Output	AXI3 master mode: BREADY
AXI_M_ARID[3:0]	Output	AXI3 master mode: ARID. Used to indicate the ID of the current outstanding read completion. Read completions can be interleaved
AXI_M_ARADDR[31:0]	Output	AXI3 master mode: ARADDR
AXI_M_ARLEN[3:0]	Output	AXI3 master mode: ARLEN
AXI_M_ARSIZE[1:0]	Output	AXI3 master mode: ARSIZE (tied to 11)
AXI_M_ARBURST[1:0]	Output	AXI3 master mode: ARBURST
AXI_M_ARVALID	Output	AXI3 master mode: ARVALID
AXI_M_ARREADY	Input	AXI3 master mode: ARREADY
AXI_M_RID[3:0]	Input	AXI3 master mode: RID. Used to indicate the ID of the current outstanding read completion. Read completions can be interleaved

**Table 12 • PCIe System AXI3 Slave Interface**

Port	Type	Description
AXI_S_AWID[3:0]	Input	AXI3 slave mode: AWID (not supported)
AXI_S_AWADDR[31:0]	Input	AXI3 slave mode: AWADDR
AXI_S_AWLEN[3:0]	Input	AXI3 slave mode: AWLEN
AXI_S_AWSIZE[1:0]	Input	AXI3 slave mode: AWSIZE
AXI_S_AWBURST[1:0]	Input	AXI3 slave mode: AWBURST
AXI_S_AWVALID	Input	AXI3 slave mode: AWVALID
AXI_S_AWREADY	Output	AXI3 slave mode: AWREADY
AXI_S_AWLOCK[1:0]	Input	AXI3 slave mode: AWLOCK
AXI_S_WID[3:0]	Input	AXI3 slave mode: WID (not supported)
AXI_S_WSTRB[7:0]	Input	AXI3 slave mode: WSTRB
AXI_S_WLAST	Input	AXI3 slave mode: WLAST
AXI_S_WVALID	Input	AXI3 slave mode: WVALID
AXI_S_WDATA [63:0]	Input	AXI3 slave mode: WDATA
AXI_S_WREADY	Output	AXI3 slave mode: WREADY
AXI_S_BID[3:0]	Output	AXI3 slave mode: BID
AXI_S_BRESP[1:0]	Output	AXI3 slave mode: BRESP
AXI_S_BVALID	Output	AXI3 slave mode: BVALID
AXI_S_BREADY	Input	AXI3 slave mode: BREADY
AXI_S_ARID[3:0]	Input	AXI3 slave mode: ARID
AXI_S_ARADDR[31:0]	Input	AXI3 slave mode: ARADDR
AXI_S_ARLEN[3:0]	Input	AXI3 slave mode: ARLEN. PCIe AXI-Slave interface supports only single length transactions (S_ARLEN = 3'b000) when the size of the transfer is less than 64 bits
AXI_S_ARSIZE[1:0]	Input	AXI3 slave mode: ARSIZE. A size value of 0b11 allows all values of burst length. A size value of 0b00, 0b01, and 0b10 allow only a burst length of 1
AXI_S_ARBURST[1:0]	Input	AXI3 slave mode: ARBURST
AXI_S_ARVALID	Input	AXI3 slave mode: ARVALID
AXI_S_ARLOCK[1:0]	Input	AXI3 slave mode: ARLOCK
AXI_S_ARREADY	Output	AXI3 slave mode: ARREADY
AXI_S_RID[3:0]	Output	AXI3 slave mode: RID
AXI_S_RDATA[63:0]	Output	AXI3 slave mode: RDATA
AXI_S_RRESP[1:0]	Output	AXI3 slave mode: RRESP. The interface responds with a SLVERR under the following conditions: <ul style="list-style-type: none"> <li>– If a completion TLP has the EP (poisoned) bit set (unsupported request).</li> <li>– If a completion with error TLP is received.</li> <li>– If a completion timeout event happens.</li> <li>– If an invalid AXI3 slave transaction is encountered such as invalid burst type.</li> </ul>
AXI_S_RLAST	Output	AXI3 slave mode: RLAST
AXI_S_RVALID	Output	AXI3 slave mode: RVALID
AXI_S_RREADY	Input	AXI3 slave mode: RREADY



**Note:** Refer to AMBA AXI3 specifications for further details.

**Table 13 • PCIe System APB Slave Interface**

Port	Type	Description
APB_S_PSEL	Input	APB slave select; selects signal for register for reads or writes
APB_S_PENABLE	Input	APB strobe. This signal indicates the second cycle of an APB transfer
APB_S_PWRITE	Input	APB write or read. If high, a write occurs when an APB transfer takes place. If low, a read takes place
APB_S_PADDR[13:0]	Input	APB address bus
APB_S_PWDATA[31:0]	Input	APB write data
APB_S_PREADY	Output	APB ready. Used to insert wait states
APB_S_PRDATA[31:0]	Output	APB read data
APB_S_PSLVERR	Output	APB Error

**Table 14 • PCIe System Clock Signals**

Port	Type	Description
CLK_BASE	Input	Fabric source clock. This clock input is used for the master and slave interfaces. It is also used as the reference clock to the SPLL which is used to achieve interface timing across the fabric to SerDes block. <sup>1</sup>
APB_S_CLK	Input	PCLK for APB slave interface in the SerDes Block
SPLL_LOCK	Output	PLL Lock signal. High indicates that the frequency and phase lock are achieved.
PLL_LOCK_INT	Output	The SPLL lock status register (active high indicates locked).
PLL_LOCKLOST_INT	Output	The SPLL lock lost status register (active high indicates that the lock is lost).

1. The frequency of this clock must match the GUI option for the CLK. BASE rate to guarantee timing is met across the fabric interface.

**Table 15 • PCIe System Reset Signals**

Ports	Type	Description
CORE_RESET_N	Input	PCIe core active low reset. Top-level fundamental asynchronous RESET to the PCIe system. It affects only the SerDes lanes which are in PCIe mode. Lanes associated with the PCIe link must have one reset for all lanes.
PHY_RESET_N	Input	Active low – SerDes – reset. Top-level fundamental asynchronous RESET to the SerDes block.
APB_S_PRESET_N	Input	APB slave interface – PRESETN: Async set. APB asynchronous reset to all APB registers.

PCIE\_WAKE\_N, PCIE\_WAKE\_REQ, and PCIE\_PERST\_N ports are added optionally with L2/P2 selection. PCIE\_WAKE\_N is an output and PCIE\_WAKE\_REQ, and PCIE\_PERST\_N ports are inputs to the PCIE core.

**Table 16 • PCIE Interrupt and Power Management Interface**

Port	Type	Description
PCIE_INTERRUPT[3:0]	Input	PCIE system interrupt inputs. When using INTx legacy interrupts, PCIE_INTERRUPT[0] is used to assert/de-assert INTA. When using MSI, up to 4 MSI can be sent. Each bit sends an MSI vector with 0-bit, starting at the MSI base and each bit increments the vector asynchronously. Synchronized internally to each lane's PIPE clock.
PCIE_SYSTEM_INT	Output	PCIE system interrupt output (notsupported)
PCIE_WAKE_REQ	Input	L2/P2 implementation: (L2 requests from fabric)* Asynchronous. Input to power-management state machine.
PCIE_WAKE_N	Output	L2/P2 implementation: (L2 exit request to RP)*
PCIE_PERST_N	Input	L2/P2 implementation: (L2 exit request from RP)* Asynchronous. Synchronized to the internal 50 MHz RC Oscillator.
PCIE_LTSSM[5:0]	Output	Ports indicate the status of LTSSM state management. Equivalent to LTSSM Register (044h) [28:24]. These bits are set to LTSSM state encoding (RO)- output bits [4:0] ltssm state. Bit[5] pulses high to indicate that a hot-reset, data link up or L2 exit condition has occurred. Synchronized to the PIPE clock.
PCIE_L2P2_ACTIVE	Output	Active high output indicating that LTSSM is in low-powerstate.
PCIE_RESET_PHASE	Output	Active high output indicating that LTSSM is in resetstate.

**Note:** \* This is only available when L2/P2 option is selected in SerDes block configurator GUI.

### 3.1.3.8.1 Base Address Register Settings

The PCIe implementation supports up to six 32-bit BARs or three 64-bit BARs. The BARs can be one of the two sizes:

- **32-bit BAR:** The theoretical address space can be as small as 16 bytes or as large as 2 gigabytes.
- **64-bit BAR:** The theoretical address space can be as small as 128 bytes or as large as 8 gigabytes. Used for memory only.

Each BAR register is 32 bits, but BARs can be combined to make a 64-bit BAR. For example, BAR0 (address offset 010h) and BAR1 (address offset 014h) define the type and size of BAR01 of the PCIe native endpoint. BAR01 can be memory-mapped prefetchable (64-bit BAR) or non-prefetchable (32-bit BAR).

The SerDes block configurator in Libero provides a GUI to configure the BAR settings for the EP application. The BAR registers share the following options:

- **Width:** Width can be 32-bit or 64-bit. If an even register is selected to be 64-bit wide, then the subsequent (odd) register serves as the upper half of 64 bits. Otherwise, the width of the odd registers is restricted to 32-bit.
- **Size:** Ranges from 4 Kbytes to 1 Gbyte, with 4 Kbyte increments.
- **Prefetchable:** Prefetchable option for memory BAR.
  - A PCIe EP, requesting memory resources through a BAR must set the BAR's prefetchable bit unless the range contains locations with read side-effects or locations in which the device does not tolerate write merging. It is recommended that memory-mapped resources be designed as prefetchable. For a PCIe EP, 64-bit addressing must be supported for all BARs that have the prefetchable bit set. 32-bit addressing is permitted for all BARs that do not have the prefetchable bit set.

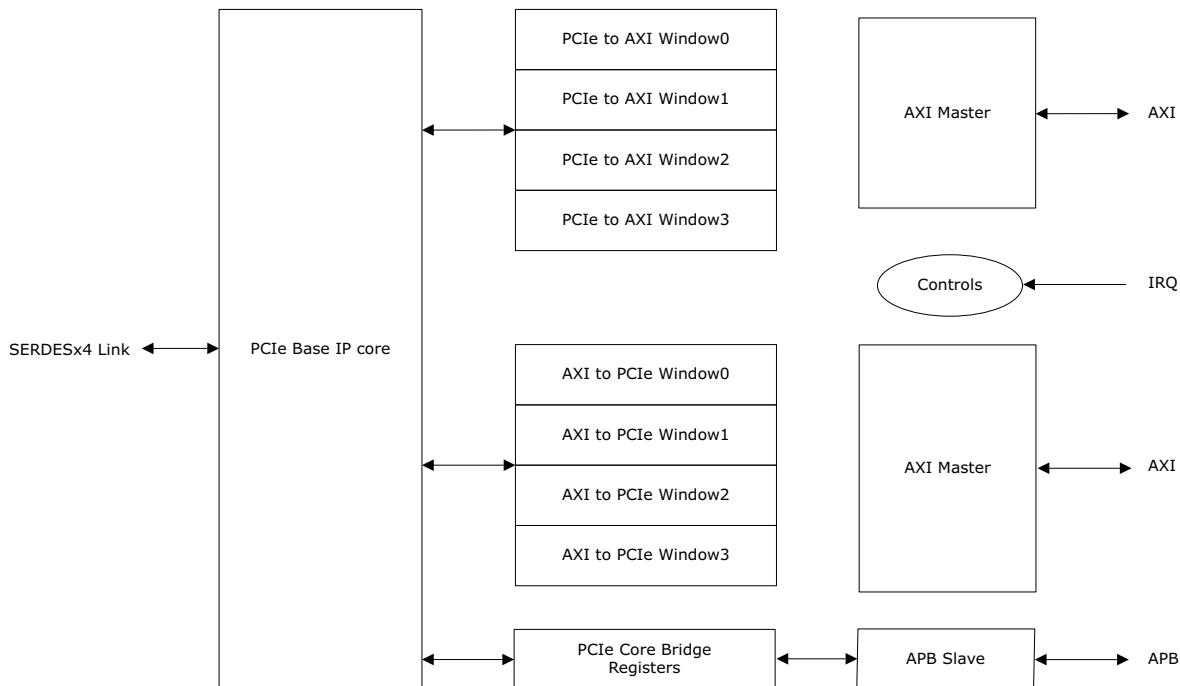
### 3.1.3.9 PCIe to AXI3 Window

The PCIe base IP receives both 32-bit address and 64-bit address PCIe requests, but only 32-bit address bits are provided to the AXI3 master. The PCIe to AXI3 address windows manage read and write requests from the PCIe link and are used to translate a PCIe 32-bit or 64-bit base address to a 32-bit AXI3 base address transaction.

### 3.1.3.10 AXI3 to PCIe Window

The AXI3 to PCIe address windows are used to translate a 32-bit AXI3 base address for a transaction to a PCIe 32-bit or 64-bit base address, to generate a PCIe TLP.

**Figure 23 • PCIe IP Block Diagram**



### 3.1.3.11 Address Translation on AXI3 Master Interface

The address space for PCIe is different from the AXI3 address space. Accessing one address space from another address space requires an address translation process.

The PCIe IP can receive both 32-bit address and 64-bit address PCIe requests, but only 32-bit address bits are provided to the AXI3 master. In order to manage address translation, the PCIe IP can implement up to 4 AXI3 master address windows, which can be mapped to 3 BARs in the main PCIe IP core.

The address mapping registers (AXI\_MASTER\_WINDOWx[x], where x can be 0, 1, 2, or 3) are shown in the following table, and are used to set the address mapping.

**Table 17 • AXI\_MASTER\_WINDOW Registers**

Bit Number	Name	Description
[31:12]	AXI_MASTER_WINDOWx[0]	Base address AXI3 master window x
[11:0]		Reserved
[31:12]	AXI_MASTER_WINDOWx[1]	Size of AXI3 master window x
[11:1]		Reserved
0		Enable bit of AXI3 master window x

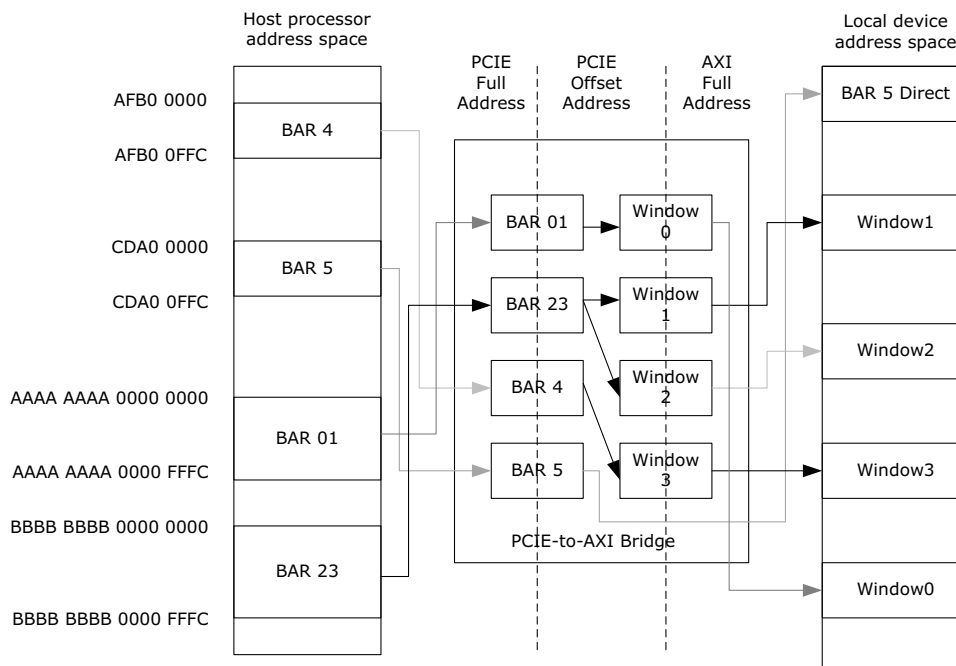
**Table 17 • AXI\_MASTER\_WINDOW Registers**

[31:12]	AXI_MASTER_WINDOWx[2]	LSB of base address PCIe window x
[11:6]		Reserved
[5:0]		These bits set the BAR. To select a BAR, set the following values: 0x01: BAR0 (32-bit BAR) or BAR0/1 (64-bit BAR) 0x02: BAR1 (32-bit BAR) only 0x04: BAR2 (32-bit BAR) or BAR2/3 (64-bit BAR) 0x08: BAR3 (32-bit BAR) only 0x10: BAR4 (32-bit BAR) or BAR4/5 (64-bit BAR) 0x20: BAR5 (32-bit BAR) only
[31:0]	AXI_MASTER_WINDOWx[3]	MSB of base address PCIe window x

**Note:** x = 0, 1, 2, and 3

Each AXI3 master address window implemented can be mapped to a BAR, and multiple address windows can be mapped to the same BAR. When transferring PCIe receive requests to the AXI3 master, the PCIe IP core, automatically removes the decoded BAR base address, then performs a windows match using the PCIe offset address. If a match is found, the IP core bridge maps the corresponding AXI3 base address. If you do not add a BAR mapping to a local address, then this causes local address to overwrite each other. Using the windows is required to shift the address to a new location to move the BAR data correctly. If you are not performing the window mapping, then you could only use a single BAR.

In Figure 24, page 33, four BARS are enabled in the bridge; two 64-bit BARS (BAR01 and BAR23), and two 32-bit BARS (BAR4 and BAR5). All AXI3 master windows are utilized: 64-bit BAR01 is mapped to AXI3 master window 0; 64-bit BAR23 is mapped to AXI3 master window 1; and AXI3 master window 2 is mapped to the upper 64 bytes of BAR23. 32-bit BAR4 is mapped to AXI3 master window 3. BAR5 is not mapped to an AXI3 window; its offset is passed directly to the AXI3 master and translation is not performed.

**Figure 24 • PCIe to AXI3 Master Address Translation**

To configure AXI\_MASTER\_WINDOWs, four APB write operations are performed to AXI\_MASTER\_WINDOW0[0], AXI\_MASTER\_WINDOW0[1], AXI\_MASTER\_WINDOW0[2] and AXI\_MASTER\_WINDOW0[3] registers:

- APB write to AXI\_MASTER\_WINDOW0[0]: APB PADDR = 100h, APB PWDATA = FFF0 0000
- APB write to AXI\_MASTER\_WINDOW0[1]: APB PADDR = 104h, APB PWDATA = FFF0 0001
- APB write to AXI\_MASTER\_WINDOW0[2]: APB PADDR = 108h, APB PWDATA = 0000 0001
- APB write to AXI\_MASTER\_WINDOW0[3]: APB PADDR = 10Ch, APB PWDATA = 0000 0000

The example is shown using relative SerDes block addressing for PADDR.

If window size is not enabled or if the PCIe offset address is located in a BAR but not in any of the windows, address translation is not performed. In this case, the PCIe base address is removed to create the AXI3 address and, for BARs larger than 4 Kbytes, MSBs are ignored.

The address translation needs to be pre-defined in the user design. This is completed using the SerDes block configurator GUI.

The PCIe AXI3 master windows are used to translate the PCIe base address domain to the local address domain.

### 3.1.3.12 AXI3 Slave Interface Address Translation

The IP core bridge can configure up to four AXI3 slave address windows to handle address translation on read/write requests initiated from the FPGA fabric. The AXI3 slave address windows are used to translate a 32-bit AXI3 base address for a transaction to a PCIe 32-bit or 64-bit base address to generate a PCIe TLP. The slave address windows can also be used to generate the following PCIe parameters:

- **TC Selection:** Indicates the PCIe traffic class in the PCIe packet header.
- **RO Bit Selection:** Generates the PCIe TLP using a selec relaxed ordering bit.
- **No Snoop Bit Selection:** Generates the PCIe TLP using a selec no snoop bit.

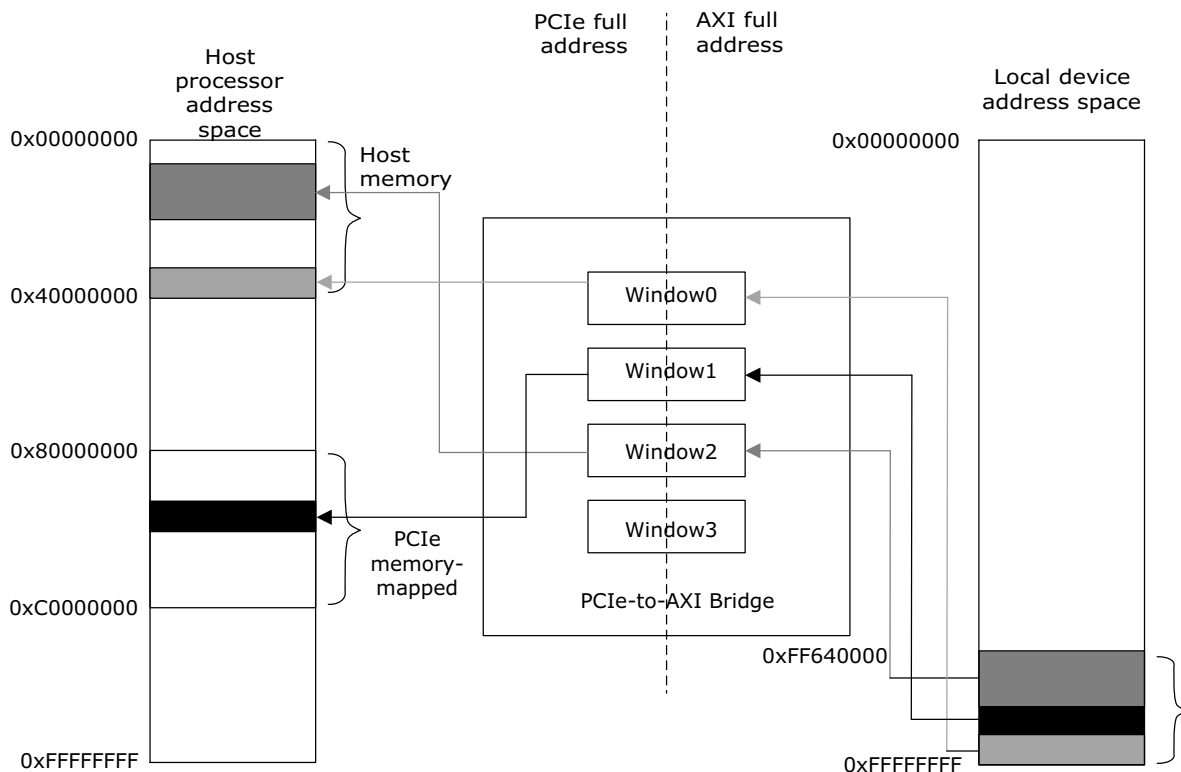
The address mapping register (AXI\_SLAVE\_WINDOWx[x], x can be 0, 1, 2, or 3) is used to set the address mapping.

**Table 18 • AXI\_SLAVE\_WINDOW Registers**

Bit Number	Name	Description
[31:12]	AXI_SLAVE_WINDOWx[0]	Base address AXI3 slave window x
[11:0]		Reserved
[31:12]	AXI_SLAVE_WINDOWx[1]	Size of AXI3 slave window x
[11:1]		Reserved
0		Enable bit of AXI3 slave window x
[31:12]	AXI_SLAVE_WINDOWx[2]	LSB of base address PCIe window 0
[11:5]		Reserved
[4:2]		AXI3 slave window 0 traffic class (TC)
1		AXI3 slave window 0 relaxed ordering (RO)
0		AXI3 slave window 0 no snoop (NS)
[31:0]	AXI_SLAVE_WINDOWx[3]	MSB of base address PCIe window x

**Note:** x = 0, 1, 2, and 3

The following figure shows address translation when AXI3 slave address window 0 and window 2 target two different regions of the host memory and the AXI3 slave address window 1 targets a PCIe memory-mapped device, enabling peer-to-peer transactions. Window 3 is not used in this example.

**Figure 25 • AXI3 Slave to PCIe Address Translation**

### 3.1.3.13 PCIe System Credit Settings

PCIe system has 2 Kbytes of receive buffer (RAM) and 1 Kbyte of transmit and replay buffer (RAM). The following sections describe the different features that impact credit processing. All the credit settings are automatically set by the Libero software based on the buffer sizes fixed in the SerDes block.

#### 3.1.3.13.1 Maximum Payload Size

The size of TLP is restricted by the capabilities of both the link partners. After the link is trained, the root complex sets the MAX\_PAYLOAD\_SIZE (maximum payload size register) value in the device control register. The possible settings are 128 and 256 bytes.

#### 3.1.3.13.2 Replay Buffer

The replay or Tx buffer, located in the data link layer, stores a copy of a transmitted TLP until the transmitted packet is acknowledged by the receiving side of the link. The replay buffer stores the read data payload from the AXI3 master as well as the write data payload from the AXI3 slave. Each stored TLP includes the header, an optional data payload, the maximum size of which is determined by the maximum payload size parameter, an optional ECRC, the sequence number, and the link CRC (LCRC) field.

#### 3.1.3.13.3 Receive Buffer

The receive buffer is located in the transaction layer and accepts incoming TLPs from the link and then sends them to the application layer for processing. The receive buffer stores TLPs based on the type of transaction and not the transaction count (TC). Types of transactions include posted transactions, non-posted transactions, and completion transactions. A transaction always has a header but does not necessarily have data. The receive buffer accounts for this distinction, maintaining separate resources for the header and data of each type of transaction. To summarize, distinct buffer resources are maintained for each of the following elements:

- Posted transactions, header (PH)
- Posted transactions, data (PD)

- Non-posted transactions, header (NPH)
- Non-posted transactions, data (NPD)
- Completion transactions, header (CPLH)
- Completion transactions, data (CPLD)

TLPs are stored in the received buffer in 64-bit addressing format, with each AXI3 slave read outstanding request consuming 16 credits (128 bytes), plus headers and data credits consuming 1 credit each (16 bytes).

### 3.1.4 User Data Throughput

The PCIe protocol is software backward-compatible with the earlier PCI and PCI-X protocols, but is significantly different from its predecessors. The performance is scalable based on the number of lanes and the generation that is implemented. The PCIe protocol specifies 2.5 giga-transfers per second for Gen1. There is a 20% overhead because the PCIe protocol uses 8b/10b encoding. Table 19, page 36 shows the aggregate bandwidth of a PCIe link.

**Table 19 • Theoretical PCIe Throughput**

	Link Width		
	x1	x2	x4
PCI Express 2.5 Gbps (1.x/2.xcompliant)	2 Gbps	4 Gbps	8 Gbps

PCIe uses credits to handle throughput balancing between both ends of the link. At the initial link-up, both sides of the link share their transmit and receive buffer sizes in terms of credits. As TLPs are sent across the link, credits are used. As user data is pulled out of the TLPs stored in the receive buffers credits are released. Information on the current state of the credits is continuously sent across the link using data link layer packets. The entire process is transparent to the user inside the PCIe core.

The [RTG4] PCIe core uses AXI3 fabric interface for user data. AXI3 slave interfaces only allow a transaction when 128 byte TLP worth of credits is available to be sent. The PCIe core can back pressure the fabric interface when credits are not available to send a TLP.

The flow control works by releasing credits to the sender as data is pulled across to the fabric. If the user is not pulling the data out fast enough, then the sender runs out of credits. When the sender sends 100% writing data to the PCIe core, only 1325 Mbps is able to go through. The credit system holds back the sender from sending more data. When the sender is pulling 100% data via a read, only 1325 Mbps is received back. In this case, PCIe core is never throttled back due to lack of credits.

In the reverse scenario, where PCIe core is the sender, for 100% write data, the fabric interface is held up at 1325 Mbps. For 100% read requests the receiver sends them back faster than 1325 Mbps, but the fabric interface only pulls them out at 1325 Mbps. The receiver is blocked due to lack of completion credits until credits are released.

The transaction size in this scenario is more efficient when TLPs are small (128 byte). Smaller packet sizes allow PCIe core to release credits faster compared to large packets. As a packet is pulled across the fabric interface, the credit is released for the sender to send another. If this happens quickly the next packet can be sent. For large packets, it takes longer to release the credit and therefore the next packet is not sent quickly.

## 3.1.5 PCIe Power Management

This section describes the power management scheme in the FPGA PCIe implementation.

### 3.1.5.1 Legacy Power Management

The PCIe bridge register space defines the capabilities of the PCIe bridge in terms of legacy power management (PME support, auxiliary current requirement and so on). The power management control and status register also contain the current power management state. The PM data and PM scale value array can define the power consumed in each power state.



### 3.1.5.2 PCIe Power Management

PCIe active state power management (ASPM) defines link power management states that a PCIe physical link is permitted to enter, in response to software-driven D-state transitions or active state link power management activities.

The PCIe protocol defines the following low power link states.

**Table 20 • PCIe Low Power States**

Low Power State	Description
L0s	Autonomous electrical idle: This state reduces power during short intervals of idle. Devices must transition to L0s independently on each direction of the link.
L1	Directed electrical idle: The L1 state reduces power when the downstream port directs the upstream ports. This state saves power in two ways: <ul style="list-style-type: none"> <li>– Shutting down the transceiver circuitry and the associated PLL</li> <li>– Significantly decreasing the number of internal core transitions</li> </ul>
L2	In this state, a WAKE# signal is required to reinitialize the Link. However, the auxiliary power is still available.
L2/L3 ready	This state prepares the PCIe link for the removal of main power and the reference clock.

PCIe EP implementation supports L0, L1, and a special version of L2.

### 3.1.5.3 PCIe Interrupts for Endpoints

The RTG4 PCIe EP implementation supports 32 MSI interrupt and INTx interrupts. It cannot support both at the same time. The user can select which interrupt model to use in the Libero SoC SerDes Configurator. The Libero software initializes the interrupts. For MSI, the user has a selection of up to 32 MSI vectors. When using MSI, the first 4 interrupts can be sent using the PCIE\_INTERRUPTS[3:0] port of the SerDes block [0] for MSI0 and [1] for MSI1. To send more than 4 interrupts, the user must use the AXI3 slave interface and send a memory write transaction to the specific address set by the root complex during interrupt negotiation.

### 3.1.5.4 ECRC Handling

ECRC ensures end-to-end data integrity. The PCIe implementation transmits a TLP with ECRC from the transmit port of the application layer. When using ECRC forwarding mode, the ECRC check and generate are performed in the application layer. The PCIE\_AER\_ECRC\_CAPABILITY register in bridge configuration registers sets the ECRC settings.

#### 3.1.5.4.1 PCIe Core Bridge Register

The PCIe core bridge registers occupy 4 KB of the configuration memory map space. These registers set the PCIe configuration and status and are initialized from flash, while configuring the high-speed serial interface generator in the Libero SoC. These registers can also be accessed through the 32-bit APB interface. The physical offset location of the PCIe core registers is 0x0000-0x0FFF from the SerDes block system memory map.

## 3.1.6 Bridge Register Space

The PCIe core bridge register space is used to configure the PCIe core settings at power-up. Initialization IP module facilitates configuration of the SerDes block in an RTG4 device. These registers are 32 bits wide and are part of the SerDes block system register. Refer to the [SerDes Block System Registers](#), page 136.

The PCIe system block registers consist of:

- Read-only registers that report control and status registers to the AXI3 side through the APB bus
- Bridge settings that must be configured at power-up, such as local interrupt mapping to MSI and test mode
- Control/status registers that can be used by the AXI3 bus to control bridge behavior during an operation



Most bridge registers are hardwired to a fixed value.

These registers are described in the next section according to their function:

- Information Registers: provide device, system, and bridge identification information.
- Bridge Configuration Registers: enable configuration of bridge functionality.
- Power Management Registers: enable configuration of the power management capabilities of the bridge.
- Address Mapping Registers: provide address mapping for AXI3 master and slave windows. These windows are used for address translation.
- EP Interrupt Registers: used in EP mode to manage interrupts.
- PCIe Control and Status Registers: These read-only registers enable the local processor to check useful information related to the PCIe interface status. This enables the local processor to detect when the bridge's PCIe interface is initialized and to monitor PCI link events.

### 3.1.7 Information Registers

The registers listed in the following table provide device, system, and the bridge identification information.

**Table 21 • PCIe Information Registers**

Register Name	Address Offset	Register Type	Initialization	Description
PCIE_VID_DEVID	000h	R/O	Flash	Identifies the manufacturer of the device or application (Table 27, page 43).
PCIE_CLASS_CODE_REG	008h	R/O	Flash	Identifies the generic function of the device and, in some cases, a specific register-level programming interface (Table 29, page 43).
PCIE_CAPTURED_BUS_DEVICE_NB	03Ch	R/O	N/A	Reports the bus and device number of the EP device for each configuration write TLP received (Table 40, page 46).
PCIE_SUBSYSTEM_ID	02Ch	R/O	Flash	Identifies the manufacturer of the device or application (Table 36, page 45).
PCIE_INFO	16Ch	R/O	N/A	Reports the bridge version (Table 94, page 61).

#### 3.1.7.1 Bridge Configuration Registers

The registers listed in the following table enable to configure bridge functionality.

**Table 22 • PCIe Bridge Configuration Registers**

Register Name	Byte Offset	State	Initialization	Description
RESERVED	204h	R/O	Flash	Sets the PCIe configuration (Table 95, page 61).
PCIE_BAR0	010h	R/W	Flash	Defines the type and size of BAR0 of the PCIe native endpoint. This register combines with BAR1 to define the type and size of BAR01 of the PCIe native endpoint (Table 30, page 43).
PCIE_BAR1	014h	R/O	Flash	Defines the type and size of BAR1 of the PCIe native endpoint. This register combines with BAR0 to define the type and size of BAR01 of the PCIe native endpoint (Table 31, page 44).
PCIE_BAR2	018h	R/O	Flash	Defines the type and size of BAR2 of the PCIe native endpoint. This register combines with BAR3 to define the type and size of BAR23 of the PCIe native endpoint (Table 32, page 44).

**Table 22 • PCIe Bridge Configuration Registers**

PCIE_BAR3	01Ch	R/O	Flash	Defines the type and size of BAR3 of the PCIe native endpoint. This register combines with BAR2 to define the type and size of BAR23 of the PCIe native endpoint ( <a href="#">Table 33</a> , page 44).
PCIE_BAR4	020h	R/O	Flash	Defines the type and size of BAR4 of the PCIe native endpoint. This register combines with BAR5 to define the type and size of BAR45 of the PCIe native endpoint ( <a href="#">Table 34</a> , page 44).
PCIE_BAR5	024h	R/O	Flash	Defines the type and size of BAR5 of the PCIe native endpoint. This register combines with BAR4 to define the type and size of BAR45 of the PCIe native endpoint ( <a href="#">Table 35</a> , page 45).
PCIE_AER_ECRC_CAPABILITY	050h	R/W	APB	Defines whether the bridge supports advanced error reporting (AER) and ECRC generation/check and whether AER/ECRC is implemented. ECRC generation and check bits can only be set if AER is implemented ( <a href="#">Table 45</a> , page 48).
MAX_PAYLOAD_SIZE	058h	R/O	Fixed	Negotiated maximum payload size ( <a href="#">Table 47</a> , page 49).
PCIE_CREDIT_ALLOCATION_0	0B0h	R/O	Fixed	Provides the initial credit values for posted transactions ( <a href="#">Table 60</a> , page 54).
PCIE_CREDIT_ALLOCATION_1	0B4h	R/O	Fixed	Provides the initial credit values for non-posted transactions ( <a href="#">Table 61</a> , page 54).
PCIE_ERROR_COUNTER_0	0A0h	R/W	N/A	Has four 8-bit counters for the four error sources. To clear the register content, the bridge must perform a write transaction (any value) to this register ( <a href="#">Table 56</a> , page 53).
PCIE_ERROR_COUNTER_1	0A4h	R/W	N/A	Has four 8-bit counters for the four error sources. To clear the register content, the bridge must perform a write transaction (any value) to this register ( <a href="#">Table 57</a> , page 53).
PCIE_ERROR_COUNTER_2	0A8h	R/W	N/A	Has four 8-bit counters for the four error sources. To clear the register content, the bridge must perform a write transaction (any value) to this register ( <a href="#">Table 58</a> , page 53).
PCIE_ERROR_COUNTER_3	0ACh	R/W	N/A	Has four 8-bit counters for the four error sources. To clear the register content, the bridge must perform a write transaction (any value) to this register ( <a href="#">Table 59</a> , page 54).

### 3.1.7.2 Power Management Registers

The registers listed in the following table enable to configure the power management capabilities of the bridge.

**Table 23 • PCIe Bridge Power Management Registers**

Register Name	Byte Offset	State	Description
PCIE_LTSSM	044h	R/O	Can be used to monitor the core state or to select a specific test mode on bits [31:16] and to control L2 entry on bits [15:0] (Table 42, page 47).
PCIE_POWER_MGT_CAPABILITY	048h	R/W	Enables the local processor to configure the power management capability (Table 43, page 48).
PCIE_PM_DATA_SCALE_0	070h	R/W	These four PM data and scale registers define the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field (see Table 51, page 50, Table 52, page 51, Table 53, page 51, and Table 54, page 52).
PCIE_PM_DATA_SCALE_1	074h	R/W	
PCIE_PM_DATA_SCALE_2	078h	R/W	
PCIE_PM_DATA_SCALE_3	07Ch	R/W	
PCIE_ASPM_L0S_CAPABILITY	060h	R/W	Defines the EP L0s accep latency and the number of fast training sequences (FTS) required by the SerDes to resynchronize its receiver on incoming data, depending on the clock mode configuration (separated clock or common clock). The number of FTS required in separated clock mode mustbe higher than that is required in common clock mode. The bridge automatically computes the ASPM L0s exit latency based on these two register values, and the maximum payload size of the control register. The selected NFTS field is that transmitted by the link training and status state machine (LTSSM) to the opposite component in order to define the number of FTS that the opposite component must send to be sure that the device receiver has re-locked onto the incoming data (Table 48, page 49).
Reserved	260h	R/W	–
PCIE_ASPM_L1_CAPABILITY	064h	R/W	Defines the EP L1 accep latency and the number of FTS required. The EP L1 accep latency is used to enable or disable the ASPM L1 entry by comparing its value to the maximum ASPM L1 exit latency of all components in the hierarchy (plus 1 microsecond per switch). If ASPM L1 accept latency is lower than the maximum ASPM L1 exit latency, ASPM L1 entry is not enabled (Table 49, page 49).
PCIE_TIMEOUT_COMPLETION	068h	R/W	Defines four mechanism, timeout ranges for completion timeout (Table 50, page 50).

### 3.1.7.3 Address Mapping Registers

The registers listed in the following table provide the address mapping for AXI3 master and slave windows. These windows are used for address translation.

**Table 24 • Address Mapping Registers**

Register Name	Address Offset	Register Type	Description
PCIE_AXI_SLAVE_WINDOW0_0	0C0h	R/W	These four register sets define the address mapping for AXI3 slave window 0 (see Table 62, page 54, Table 63, page 55, Table 64, page 55, and Table 65, page 55).
PCIE_AXI_SLAVE_WINDOW0_1	0C4h		
PCIE_AXI_SLAVE_WINDOW0_2	0C8h		
PCIE_AXI_SLAVE_WINDOW0_3	0CCh		
PCIE_AXI_SLAVE_WINDOW1_0	0D0h	R/W	These four register sets define the address mapping for AXI3 slave window 1 (see Table 66, page 55, Table 67, page 55, Table 68, page 56, and Table 69, page 56).
PCIE_AXI_SLAVE_WINDOW1_1	0D4h		
PCIE_AXI_SLAVE_WINDOW1_2	0D8h		
PCIE_AXI_SLAVE_WINDOW1_3	0DCh		
PCIE_AXI_SLAVE_WINDOW2_0	0E0h	R/W	These four register sets define the address mapping for AXI3 slave window 2 (see Table 70, page 56, Table 71, page 56, Table 72, page 56, and Table 73, page 57).
PCIE_AXI_SLAVE_WINDOW2_1	0E4h		
PCIE_AXI_SLAVE_WINDOW2_2	0E8h		
PCIE_AXI_SLAVE_WINDOW2_3	0ECh		
PCIE_AXI_SLAVE_WINDOW3_0	0F0h	R/W	These four register sets define the address mapping for AXI3 slave window 3 (see Table 74, page 57, Table 75, page 57, Table 76, page 57, and Table 77, page 58).
PCIE_AXI_SLAVE_WINDOW3_1	0F4h		
PCIE_AXI_SLAVE_WINDOW3_2	0F8h		
PCIE_AXI_SLAVE_WINDOW3_3	0FCh		
PCIE_AXI_MASTER_WINDOW0_0	100h	R/W	These four register sets define the address mapping for AXI3 master window 0 (see Table 78, page 58, Table 79, page 58, Table 80, page 58, and Table 81, page 58).
PCIE_AXI_MASTER_WINDOW0_1	104h		
PCIE_AXI_MASTER_WINDOW0_2	108h		
PCIE_AXI_MASTER_WINDOW0_3	10Ch		
PCIE_AXI_MASTER_WINDOW1_0	110h	R/W	These four register sets define the address mapping for AXI3 master window 1 (see Table 82, page 59, Table 83, page 59, Table 84, page 59, and Table 85, page 59).
PCIE_AXI_MASTER_WINDOW1_1	114h		
PCIE_AXI_MASTER_WINDOW1_2	118h		
PCIE_AXI_MASTER_WINDOW1_3	11Ch		
PCIE_AXI_MASTER_WINDOW2_0	120h	R/W	These four register sets define the address mapping for AXI3 master window 2 (see Table 86, page 60, Table 87, page 60, Table 88, page 60, and Table 89, page 60).
PCIE_AXI_MASTER_WINDOW2_1	124h		
PCIE_AXI_MASTER_WINDOW2_2	128h		
PCIE_AXI_MASTER_WINDOW2_3	12Ch		
PCIE_AXI_MASTER_WINDOW3_0	130h	R/W	These four register sets define the address mapping for AXI3 master window 3 (see Table 90, page 60, Table 91, page 61, Table 92, page 61, and Table 93, page 61).
PCIE_AXI_MASTER_WINDOW3_1	134h		
PCIE_AXI_MASTER_WINDOW3_2	138h		
PCIE_AXI_MASTER_WINDOW3_3	13Ch		

### 3.1.7.4 EP Interrupt Registers

The PCIe IP core can generate interrupts through the input signal. This signal may be required by a device in order to interrupt the host processor, call its device drivers, or report application layer-specific events or errors. The parameter of the IP core bridge defines the number of interrupt bits for this signal.

**Table 25 • EP Interrupt Registers**

Register Name	Address Offset	Register Type	Description
PCIE_MSI_0	080h	R/W	Defines single MSI function, with up to 32 possible MSI messages (Table 55, page 52).
PCIE_MSI_CTRL_STATUS	040h	R/W	This register sets MSI control and status. All bits are R/O except the number of MSI requested and the multiple message enable fields, which are R/W. Up to 32 MSI messages can be requested by the device, although the PCI software can allocate less than the number of MSI requested. This information can be read by the local processor through the multiple message enable field of the register (Table 41, page 46).

### 3.1.7.5 PCIe Control and Status Registers

The following registers are read-only registers that enable the local processor to check useful information related to the PCIe interface status, such as the initialization of PCIe interface and monitoring of PCI link events, as shown in the following table. A complete description of these registers can be found in the PCIe specifications.

**Table 26 • PCIe Control and Status Registers**

Register Name	Address Offset	Register Type	Description
CFG_PRMSCR	004h	R/O	The command and status register of PCI configuration space (Table 28, page 43).
PCIE_DEVSCR	030h	R/O	Reports the current value of the PCIe device control and status register. It can be monitored by the local processor when relaxed ordering and no snoop bits are enabled in the system (Table 37, page 45).
PCIE_LINKSCR	034h	R/O	Reports the current value of the PCIe link control and status register. It can be monitored by the local processor when relaxed ordering and no snoop bits are enabled in the system (Table 38, page 45).
CFG_PRMSCR	04Ch	R/O	Reports the current values of the XpressRich2 core's power management control status register (Table 28, page 43).
PCIE_SLOTCAP	154h	–	Reserved
PCIE_SLOTCSR	158h	–	Reserved
PCIE_ROOTCSR	15Ch	–	Reserved

### 3.1.7.5.1 PCIe Bridge Registers

The following sub-section describe all PCIe bridge registers in detail.

#### PCIE\_VID\_DEVID Register (000h)

**Table 27 • PCIE\_VID\_DEVID**

Bit Number	Name	Reset Value	Description
[31:16]	Device ID	0x11AA	Identifies the manufacturer of the device or application. The values are assigned by the PCI-SIG. The default value, 11AA, is the device ID for Microsemi.
[15:0]	Vendor ID	0x1556	The field Identifies the manufacturer of the device or application. The values are assigned by PCI-SIG. The default value, 1556, is the Vendor ID for Microsemi.

#### PCIE\_CFG\_PRMSCR Register (004h)

**Table 28 • CFG\_PRMSCR**

Bit Number	Name	Reset Value	Description
[31:0]	Configuration Primary Control Status Register	0x00100000	The command and status register of PCI configuration space.

#### PCIE\_CLASS\_CODE Register (008h)

**Table 29 • PCIE\_CLASS\_CODE\_REG**

Bit Number	Name	Reset Value	Description
[31:16]	PCIE_CLASS_CODE	0x0000	Identifies the manufacturer of the device or application. The values are assigned by PCI-SIG.
[15:0]	RESERVED0	0x0000	Identifies the manufacturer of the device or application. The values are assigned by PCI-SIG.

#### PCIE\_BAR0 Register (010h)

**Table 30 • PCIE\_BAR0**

Bit Number	Name	Reset Value	Description
[31:4]	BAR0_31_4	0x000000	Defines the type and size of BAR0 of the PCIe native EP.
3	BAR0_3	0x1	Identifies the ability of the memory space to be prefetched.
[2:1]	BAR0_2_1	0x10	Set to '00' to indicate anywhere in the 32-bit address space.
0	BAR0_0	0x0	Memory space indicator

**PCIE\_BAR1 Register (014h)****Table 31 • PCIE\_BAR1**

Bit Number	Name	Reset Value	Description
[31:4]	BAR1_31_4	0x000000	Defines the type and size of BAR1 of the PCIe native EP.
3	BAR1_3	0x0	Identifies the ability of the memory space to be prefetched.
[2:1]	BAR1_2_1	0x00	Set to '00' to indicate anywhere in the 32-bit address space.
0	BAR1_0	0x0	Memory space indicator.

**PCIE\_BAR2 Register (018h)****Table 32 • PCIE\_BAR2**

Bit Number	Name	Reset Value	Description
[31:4]	BAR2_31_4	0x000000	Defines the type and size of BAR2 of the PCIe native EP
3	BAR2_3	0x1	Identifies the ability of the memory space to be prefetched.
[2:1]	BAR2_2_1	0x10	Set to '00' to indicate anywhere in the 32-bit address space.
0	BAR2_0	0x0	Memory space indicator.

**PCIE\_BAR3 Register (01Ch)****Table 33 • PCIE\_BAR3**

Bit Number	Name	Reset Value	Description
[31:4]	BAR3_31_4	0x000000	Defines the type and size of BAR3 of the PCIe native EP.
3	BAR3_3	0x0	Identifies the ability of the memory space to be prefetched.
[2:1]	BAR3_2_1	0x00	Set to '00' to indicate anywhere in the 32-bit address space.
0	BAR3_0	0x0	Memory space indicator.

**PCIE\_BAR4 Register (020h)****Table 34 • PCIE\_BAR4**

Bit Number	Name	Reset Value	Description
[31:4]	BAR4_31_4	0x000000	Defines the type and size of BAR4 of the PCIe native EP.
3	BAR4_3	0x1	Identifies the ability of the memory space to be prefetched.
[2:1]	BAR4_2_1	0x10	Set to '00' to indicate anywhere in the 32-bit address space.
0	BAR4_0	0x0	Memory space indicator.

**PCIE\_BAR5 Register (024h)****Table 35 • PCIE\_BAR5**

Bit Number	Name	Reset Value	Description
[31:4]	BAR5_31_4	0x000000	Defines the type and size of BAR1 of the PCIe native EP.
3	BAR5_3	0x0	Identifies the ability of the memory space to be prefetched.
[2:1]	BAR5_2_1	0x00	Set to '00' to indicate anywhere in the 32-bit address space.
0	BAR5_0	0x0	Memory space indicator.

**PCIE\_SUBSYSTEM\_ID Register (02Ch)****Table 36 • PCIE\_SUBSYSTEM\_ID**

Bit Number	Name	Reset Value	Description
[31:16]	SUBSYSTEM_ID	0x11AA	This field further qualifies the manufacturer of the device or application and the value is typically the same as the Device ID.
[15:0]	SUBSYSTEM_VEN DOR_ID	0x1556	This field further qualifies the manufacturer of the device or application.

**PCIE\_DEVSCR Register (030h)****Table 37 • PCIE\_DEVSCR**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_DEVSCR	0x00000000	Device control and status: This register reports the current value of the PCIe device control and status register. It can be monitored by the local processor when relaxed ordering and no snoop bits are enabled in the system.

**Note:** This register is READ only.

**PCIE\_LINKSCR Register (034h)****Table 38 • PCIE\_LINKSCR**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_LINKSCR	0x00000050	This register reports the current value of the PCIe link control and status register. It can be monitored by the local processor when relaxed ordering and no snoop bits are enabled in the system.



**PCIE\_TC\_VC\_MAPPING Register(038h)****Table 39 • PCIE\_TC\_VC\_MAPPING**

Bit Number	Name	Reset Value	Description
[31:24]	TC_VC_MAPPING_31_24	0x0	Reserved
[23:21]	TC_VC_MAPPING_23_21	0x0	Mapping for TC7
[20:18]	TC_VC_MAPPING_20_18	0x0	Mapping for TC6
[17:15]	TC_VC_MAPPING_17_15	0x0	Mapping for TC5
[14:12]	TC_VC_MAPPING_14_12	0x0	Mapping for TC4
[11:9]	TC_VC_MAPPING_11_9	0x0	Mapping for TC3
[8:6]	TC_VC_MAPPING_8_6	0x0	Mapping for TC2
[5:3]	TC_VC_MAPPING_5_3	0x0	Mapping for TC1
[2:0]	TC_VC_MAPPING_2_0	0x0	Mapping for TC0 (always 0)

**Note:** This register reports the TC-to-VC mapping configured for each PCIe traffic channel.

**PCIE\_CAPTURED\_BUS\_DEVICE\_NB Register(03Ch)****Table 40 • PCIE\_CAPTURED\_BUS\_DEVICE\_NB**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_CAPTURED_BUS_DEVICE_NB	0x0	This register reports the bus and device number of the EP device for each configuration write TLP received.

**PCIE\_MSI\_CTRL\_STATUS Register (040h)****Table 41 • PCIE\_MSI\_CTRL\_STATUS**

Bit Number	Name	Reset Value	Description
[31:24]	MSI_CTRL_STATUS_31_24	0x0	These RO bits are hardwired to 00000000.
23	MSI_CTRL_STATUS_23	0x0	This RO bit is hardwired to 1.
[22:20]	MSI_CTRL_STATUS_22_20	0x0	Multiple message enable. Fabric logic/MSS checks this RO APB register to see how many MSI interrupt resources are allocated from the host side.
[19:17]	MSI_CTRL_STATUS_19_17	0x0	Number of MSI requested. Fabric logic/MSS writes this RW APB register to request the number of MSI interrupt resources needed from the host.
16	MSI_CTRL_STATUS_16	0x0	MSI is enabled. Fabric logic/MSS checks this RO APB register to see if host has enabled MSI on the PCIe bus.
[15:0]	MSI_CTRL_STATUS_15:0	0x0	These bits are hardwired to 0x7805.

**PCIE\_LTSSM Register (044h)****Table 42 • PCIE\_LTSSM**

Bit Number	Name	Reset Value	Description
[31:29]	LTSSM_31_29	0x0	Reserved
[28:24]	LTSSM_28_24	0x0	These bits set LTSSM state encoding (RO): 00000: Detect.quiet 00001: Detect.active 00010: Polling.active 00011: Polling.compliance 00100: Polling.configuration 00101: Reserved (ex polling.speed) 00110: Configuration.linkwidth.start 00111: Configuration.linkwidth.accept 01000: Configuration.lanenum.accept 01001: Configuration.lanenum.wait 01010: Configuration.complete 01011: Configuration.idle 01100: Recovery.RcvrLock 01101: Recovery.RcvrCfg 01110: Recovery.idle 01111: L0 10000: Disabled 10001: Loopback.entry 10010: Loopback.active 10011: Loopback.exit 10100: Hot reset 10101: L0s (transmit) 10110: L1.entry 10111: L1.Idle 11000: L2.idle 11001: L2.TransmitWake 11010: Recovery.speed 11011 - 11111: Reserved
[23:20]	LTSSM_23_20	0x0	Reserved
19	LTSSM_19	0x0	Forces compliance pattern (R/W).
18	LTSSM_18	0x0	Fully disables power management (R/W).
17	LTSSM_17	0x0	Sets master loopback (R/W).
16	LTSSM_16	0x0	Disables scrambling (R/W).
[15:2]	LTSSM_15_2	0x0	Reserved
1	LTSSM_1	0x0	Indicates if PME_TURN_OFF was received (RO).
0	LTSSM_0	0x0	Acknowledges PME_TURN_OFF (R/W).

**PCIE\_POWER\_MGT\_CAPABILITY Register(048h)****Table 43 • PCIE\_POWER\_MGT\_CAPABILITY**

Bit Number	Name	Reset Value	Description
[31:27]	POWER_MGT_CAPABILITY_31_27	0x0	These bits set PME support.
26	POWER_MGT_CAPABILITY_26	0x0	Sets D2 support. If this field is cleared, PME_SUPPORT bit 29 must also be cleared.
25	POWER_MGT_CAPABILITY_25	0x0	Sets D1 support. If this field is cleared, PME_SUPPORT bit 28 must also be cleared.
[24:22]	POWER_MGT_CAPABILITY_24_22	0x0	These bits set the maximum current required.
21	POWER_MGT_CAPABILITY_21	0x0	Sets device specification initialization.
[20:19]	POWER_MGT_CAPABILITY_20_19	0x0	Reserved
18	POWER_MGT_CAPABILITY_18	0x0	Sets PCI power management interface specification version; hardwired to 011b (PCIe Spec. v1.1).
[17:0]	POWER_MGT_CAPABILITY_17_0	0x0	Reserved.

**PCIE\_CFG\_PMSCR Register (04Ch)****Table 44 • PCIE\_CFG\_PMSCR**

Bit Number	Name	Reset Value	Description
31:0	CFG_PMSCR	0x0	Reports the current values of the PCIe IP core's power management control status register.

**PCIE\_AER\_ECRC\_CAPABILITY Register(050h)****Table 45 • PCIE\_AER\_ECRC\_CAPABILITY**

Bit Number	Name	Reset Value	Description
[31:3]	AER_ECRC_CAPABILITY_31_3	0x0	Reserved
2	AER_ECRC_CAPABILITY_2	0x0	Defines whether AER is implemented or not.
1	AER_ECRC_CAPABILITY_1	0x0	Defines ECRC generation.
0	AER_ECRC_CAPABILITY_0	0x0	Defines ECRC check.

**PCIE\_VC1\_CAPABILITY Register(054h)****Table 46 • PCIE\_VC1\_CAPABILITY**

Bit Number	Name	Reset Value	Description
[31:0]	Reserved	0x0	Reserved

**PCIE\_MAX\_PAYLOAD\_SIZE Register (058h)****Table 47 • MAX\_PAYLOAD\_SIZE**

Bit Number	Name	Reset Value	Description
[31:3]	Reserved	0x0	Reserved
2:0	MAX_PAYLOAD_SIZE_2_0	0x0	Negotiated maximum payload size. The EP sets its own maximum payload size to 2 KB: 000: 128 bytes 001: 256 bytes 010: 512 bytes 011: 1 Kbytes 100: 2 Kbytes

**PCIE\_ASPM\_L0S\_CAPABILITY Register(060h)****Table 48 • PCIE\_ASPM\_L0S\_CAPABILITY**

Bit Number	Name	Reset Value	Description
31	ASPM_L0S_CAPABILITY_31	0x0	NFTS_COMCLK in common clock mode
[23:16]	ASPM_L0S_CAPABILITY_23_16	0x0	NFTS_SPCLK in separated clock mode
[15:10]	ASPM_L0S_CAPABILITY_15_10	0x0	Reserved
[9:7]	ASPM_L0S_CAPABILITY_9_7	0x0	L0s exit latency for separate clock
[6:4]	ASPM_L0S_CAPABILITY_6_4	0x0	L0s exit latency for common clock
[3:1]	ASPM_L0S_CAPABILITY_3_1	0x0	EP L0s acceptlatency
0	ASPM_L0S_CAPABILITY_0	0x0	Reserved

**PCIE\_ASPM\_L1\_CAPABILITY Register(064h)****Table 49 • PCIE\_ASPM\_L1\_CAPABILITY**

Bit Number	Name	Reset Value	Description
[31:27]	ASPM_L1_CAPABILITY_31_27	0x0	Reserved
[26:24]	ASPM_L1_CAPABILITY_26_24	0x0	L1 exit latency common clock
[23:19]	ASPM_L1_CAPABILITY_23_19	0x0	Reserved
[18:16]	ASPM_L1_CAPABILITY_18_16	0x0	L1 exit latency separated clock mode. This value must be higher than for common clock mode.
[15:4]	ASPM_L1_CAPABILITY_15_4	0x0	Reserved
[3:1]	ASPM_L1_CAPABILITY_3_1	0x0	Endpoint L1 acceptable latency
[0]	ASPM_L1_CAPABILITY_0	0x0	ASPM L1 support: If this field is not set (no ASPM L1 support), all other fields must be set to 0. ASPM L1 is mandatory for ExpressCard devices.

**PCIE\_TIMEOUT\_COMPLETION Register(068Ch)****Table 50 • PCIE\_TIMEOUT\_COMPLETION**

Bit Number	Name	Reset Value	Description
[31:4]	TIMEOUT_COMPLETION_31_4	0x0	Reserved
[3:0]	TIMEOUT_COMPLETION_3_0	0x0	Completion timeout ranges supported – This field indicates device function support for the optional Completion timeout programmability mechanism: This mechanism allows system software to modify the completion timeout value. Four time value ranges are defined: Range A: 50 $\mu$ s to 10 ms Range B: 10 ms to 250 ms Range C: 250 ms to 4 s Range D: 4 s to 64 s Bits are set as per the following values to show the supported timeout value ranges. 0000b: Completion timeout programming not supported. –0000b is default setting. An error does not occur if a completion does not come back. 0001b: range A 0010b Range B 0011b: ranges A and B 0110b: ranges B and C 0111b: ranges A, B, and C 1110b: ranges B, C and D 1111b: ranges A, B, C, and D All other values are reserved.

**PCIE\_PM\_DATA\_SCALE\_0 Register (070h)****Table 51 • PCIE\_PM\_DATA\_SCALE\_0**

Bit Number	Name	Reset Value	Description
[31:24]	PM_DATA_SCALE_0_31_24	0x0	Set the register that defines Data3 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[23:16]	PM_DATA_SCALE_0_23_16	0x0	Set the register that defines Data2 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[15:8]	PM_DATA_SCALE_0_15_8	0x0	Set the register that defines Data1 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[7:0]	PM_DATA_SCALE_0_7_0	0x0	Set the register that defines Data0 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.

**PCIE\_PM\_DATA\_SCALE\_1 Register (074h)****Table 52 • PCIE\_PM\_DATA\_SCALE\_1**

Bit Number	Name	Reset Value	Description
[31:24]	PM_DATA_SCALE_1_31_24	0x0	Set the register that defines Data7 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[23:16]	PM_DATA_SCALE_1_23_16	0x0	Set the register that defines Data6 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[5:8]	PM_DATA_SCALE_1_15_8	0x0	Set the register that defines Data5 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[7:0]	PM_DATA_SCALE_1_7_0	0x0	Set the register that defines Data4 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.

**PCIE\_PM\_DATA\_SCALE\_2 Register (078h)****Table 53 • PCIE\_PM\_DATA\_SCALE\_2**

Bit Number	Name	Reset Value	Description
[31:26]	PM_DATA_SCALE_2_31_26	0x0	Reserved
[25:24]	PM_DATA_SCALE_2_25_24	0x0	Set the register that defines data scale 3 of the PM data value of the device for each possible power state, defined by the PCI power management specification, used in conjunction with the PM scale field.
[23:18]	PM_DATA_SCALE_2_23_18	0x0	Reserved
[17:16]	PM_DATA_SCALE_2_17_16	0x0	Set the register that defines data scale 2 of the PM data value of the device for each possible power state, defined by the PCI power management specification, used in conjunction with the PM scale field.
[15:10]	PM_DATA_SCALE_2_15_10	0x0	Reserved
[9:8]	PM_DATA_SCALE_2_9_8	0x0	Set the register that defines data scale 1 of the PM data value of the device for each possible power state, defined by the PCI power management specification, used in conjunction with the PM scale field.
[7:2]	PM_DATA_SCALE_2_7_2	0x0	Reserved
[1:0]	PM_DATA_SCALE_2_1_0	0x0	Set the register that defines data scale 0 of the PM data value of the device for each possible power state, defined by the PCI power management specification, used in conjunction with the PM scale field.

**PCIE\_PM\_DATA\_SCALE\_3 Register (07Ch)****Table 54 • PCIE\_PM\_DATA\_SCALE\_3**

Bit Number	Name	Reset Value	Description
[31:26]	PM_DATA_SCALE_3_31_26	0x0	Reserved
[25:24]	PM_DATA_SCALE_3_25_24	0x0	Set the register that defines data scale 7 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[23:18]	PM_DATA_SCALE_3_23_18		Reserved
[17:16]	PM_DATA_SCALE_3_17_16	0x0	Set the register that defines data scale 6 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[15:10]	PM_DATA_SCALE_3_15_10	0x0	Reserved
[9:8]	PM_DATA_SCALE_3_9_8	0x0	Set the register that defines data scale 5 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[7:2]	PM_DATA_SCALE_3_7_2	0x0	Reserved
[1:0]	PM_DATA_SCALE_3_1_0	0x0	Set the register that defines data scale 4 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.

**PCIE\_MSI\_0 Register (080h)****Table 55 • PCIE\_MSI\_0**

Bit Number	Name	Reset Value	Description
[31:27]	MSI0_31_27	0x0	These bits set MSI_Offset[4] of MSI_MAP0.
[26:24]	MSI0_26_24	0x0	These bits set MSI_TC[4] of MSI_MAP0.
[23:19]	MSI0_23_19	0x0	These bits set MSI_Offset[3] of MSI_MAP0.
[18:16]	MSI0_18_16	0x0	These bits set MSI_TC[3] of MSI_MAP0.
[15:11]	MSI0_15_11	0x0	These bits set MSI_Offset[2] of MSI_MAP0.
[10:8]	MSI0_10_8	0x0	These bits set MSI_TC[2] of MSI_MAP0.
[7:3]	MSI0_7_3	0x0	These bits set MSI_Offset[1] of MSI_MAP0.
[2:0]	MSI0_2_0	0x0	These bits set MSI_TC[1] of MSI_MAP0.

**PCIE\_ERROR\_COUNTER\_0 Register(0A0h)****Table 56 • PCIE\_ERROR\_COUNTER\_0**

Bit Number	Name	Reset Value	Description
[31:24]	ERROR_COUNTER0_31_24	0x0	8-bit counter that reports the following error source: A3: DLLP error
[23:16]	ERROR_COUNTER0_23_16	0x0	8-bit counter that reports the following error source: A2: TLP error
[15:8]	ERROR_COUNTER0_15_8	0x0	8-bit counter that reports the following error source: A1: Training error (notsupported)
[7:0]	ERROR_COUNTER0_7_0	0x0	8-bit counter that reports the following errorsource: A0: Receiver port error

**PCIE\_ERROR\_COUNTER\_1 Register(0A4h)****Table 57 • PCIE\_ERROR\_COUNTER\_1**

Bit Number	Name	Reset Value	Description
[31:24]	ERROR_COUNTER1_31_24	0x0	8-bit counter that reports the following error source: A7: Poisoned TLP received error
[23:16]	ERROR_COUNTER1_23_16	0x0	8-bit counter that reports the following error source: A6: Data link layer protocol error
[15:8]	ERROR_COUNTER1_15_8	0x0	8-bit counter that reports the following error source: A5: Replay number rollover error
[7:0]	ERROR_COUNTER1_7_0	0x0	8-bit counter that reports the following error source: A4: Replay time error

**PCIE\_ERROR\_COUNTER\_2 Register(0A8h)****Table 58 • PCIE\_ERROR\_COUNTER\_2**

Bit Number	Name	Reset Value	Description
[31:24]	ERROR_COUNTER2_31_24	0x0	8-bit counter that reports the following error source: AB: Completer abort error
[23:16]	ERROR_COUNTER2_23_16	0x0	8-bit counter that reports the following error source: AA: Completion timeout error
[15:8]	ERROR_COUNTER2_15_8	0x0	8-bit counter that reports the following errorsource: A9: Unsupported request error
[7:0]	ERROR_COUNTER2_7_0	0x0	8-bit counter that reports the following errorsource: A8: ECRC error



**PCIE\_ERROR\_COUNTER\_3 Register(0ACh)****Table 59 • PCIE\_ERROR\_COUNTER\_3**

Bit Number	Name	Reset Value	Description
[31:14]	ERROR_COUNTER3_31_24	0x0	8-bit counter that reports the following error source: AF: Malformed TLP error
[23:16]	ERROR_COUNTER3_23_16	0x0	8-bit counter that reports the following error source: AE: Flow control protocol error
[15:8]	ERROR_COUNTER3_15_8	0x0	8-bit counter that reports the following error source: AD: Receiver overflow error
[7:0]	ERROR_COUNTER3_7_0	0x0	8-bit counter that reports the following error source: AC: Unexpected completion error

**PCIE\_CREDIT\_ALLOCATION\_0 Register(0B0h)****Table 60 • PCIE\_CREDIT\_ALLOCATION\_0**

Bit Number	Name	Reset Value	Description
[31:28]	CREDIT_ALLOCATION0_31_28	0x0	Reserved
[27:16]	CREDIT_ALLOCATION0_27_16	0x0	VC0 posted header/data creditpd_cred0
[15:8]	CREDIT_ALLOCATION0_15_8	0x0	Reserved
[7:0]	CREDIT_ALLOCATION0_7_0	0x0	VC0 posted header/data creditph_cred0

**PCIE\_CREDIT\_ALLOCATION\_1 Register(0B4h)****Table 61 • PCIE\_CREDIT\_ALLOCATION\_1**

Bit Number	Name	Reset Value	Description
[31:28]	CREDIT_ALLOCATION1_31_28	0x0	Reserved
[27:16]	CREDIT_ALLOCATION1_27_16	0x0	VC0 non-posted header/data creditnph_cred0
[15:8]	CREDIT_ALLOCATION1_15_8	0x0	Reserved
[7:0]	CREDIT_ALLOCATION1_7_0	0x0	VC0 non-posted header/data creditnph_cred0

**PCIE\_AXI\_SLAVE\_WINDOW0\_0 Register(0C0h)****Table 62 • PCIE\_AXI\_SLAVE\_WINDOW0\_0**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW00_31_12	0x0	Base address AXI3 slave window 0
[11:0]	Reserved	0x0	Reserved

**PCIE\_AXI\_SLAVE\_WINDOW0\_1 Register(0C4h)****Table 63 • PCIE\_AXI\_SLAVE\_WINDOW0\_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW0_1_31_12	0x0	Size of AXI3 slave window 0
[11:1]	Reserved	0x0	Reserved
0	PCIE_AXI_SLAVE_WINDOW0_1_0	0x0	Enable bit of AXI3 slave window 0

**PCIE\_AXI\_SLAVE\_WINDOW0\_2 Register(0C8h)****Table 64 • PCIE\_AXI\_SLAVE\_WINDOW0\_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW0_2_31_12	0x0	LSB of the base address PCIe window 0
[11:5]	Reserved	0x0	Reserved
[4:2]	PCIE_AXI_SLAVE_WINDOW0_2_4_2	0x0	AXI3 slave window 0 traffic class (TC)
1	PCIE_AXI_SLAVE_WINDOW0_2_1	0x0	AXI3 Slave window 0 relaxed ordering (RO)
0	PCIE_AXI_SLAVE_WINDOW0_2_0	0x0	AXI3 Slave window 0 no snoop (NS)

**PCIE\_AXI\_SLAVE\_WINDOW0\_3 Register(0CCh)****Table 65 • PCIE\_AXI\_SLAVE\_WINDOW0\_3**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_SLAVE_WINDOW0_3_31_0	0x0	MSB of the base address PCIe window 0

**PCIE\_AXI\_SLAVE\_WINDOW1\_0 Register(0D0h)****Table 66 • PCIE\_AXI\_SLAVE\_WINDOW1\_0**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW1_0_31_12	0x0	Base address AXI3 slave window 1
[11:0]	Reserved	0x0	Reserved

**PCIE\_AXI\_SLAVE\_WINDOW1\_1 Register(0D4h)****Table 67 • PCIE\_AXI\_SLAVE\_WINDOW1\_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW1_1_31_12	0x0	Size of AXI3 slave window 1
[11:1]	Reserved	0x0	Reserved
0	PCIE_AXI_SLAVE_WINDOW1_1_0	0x0	Enable bit of AXI3 slave window 1

**PCIE\_AXI\_SLAVE\_WINDOW1\_2 Register(0D8h)****Table 68 • PCIE\_AXI\_SLAVE\_WINDOW1\_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW1_2_31_12	0x0	LSB of the base address PCIe window 1
[11:5]	Reserved	0x0	Reserved
[4:2]	PCIE_AXI_SLAVE_WINDOW1_2_4_2	0x0	AXI3 slave window 0 traffic class (TC)
1	PCIE_AXI_SLAVE_WINDOW1_2_1	0x0	AXI3 slave window 0 relaxed ordering (RO)
0	PCIE_AXI_SLAVE_WINDOW1_2_0	0x0	AXI3 slave window 0 no snoop (NS)

**PCIE\_AXI\_SLAVE\_WINDOW1\_3 Register(0DCh)****Table 69 • PCIE\_AXI\_SLAVE\_WINDOW1\_3**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_SLAVE_WINDOW1_3_31_0	0x0	MSB of the base address PCIe window 1

**PCIE\_AXI\_SLAVE\_WINDOW2\_0 Register(0E0h)****Table 70 • PCIE\_AXI\_SLAVE\_WINDOW2\_0**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW2_0_31_12	0x0	Base address AXI3 slave window 2
[11:0]	Reserved	0x0	Reserved

**PCIE\_AXI\_SLAVE\_WINDOW2\_1 Register(0E4h)****Table 71 • PCIE\_AXI\_SLAVE\_WINDOW2\_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW2_1_31_12	0x0	Size of AXI3 slave window 2
[11:1]	Reserved	0x0	Reserved
0	PCIE_AXI_SLAVE_WINDOW2_1_0	0x0	Enable bit of AXI3 slave window 2

**PCIE\_AXI\_SLAVE\_WINDOW2\_2 Register(0E8h)****Table 72 • PCIE\_AXI\_SLAVE\_WINDOW2\_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW2_2_31_12	0x0	LSB of the base address PCIe window 2
[11:5]	Reserved	0x0	Reserved
[4:2]	PCIE_AXI_SLAVE_WINDOW2_2_4_2	0x0	AXI3 slave window 0 traffic class (TC)
1	PCIE_AXI_SLAVE_WINDOW2_2_1	0x0	AXI3 slave window 0 relaxed ordering (RO)

**Table 72 • PCIE\_AXI\_SLAVE\_WINDOW2\_2 (continued)**

Bit Number	Name	Reset Value	Description
0	PCIE_AXI_SLAVE_WINDOW2_2_0	0x0	AXI3 slave window 0 no snoop (NS)

**PCIE\_AXI\_SLAVE\_WINDOW2\_3 Register(0ECh)****Table 73 • PCIE\_AXI\_SLAVE\_WINDOW2\_3**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_SLAVE_WINDOW2_3_31_0	0x0	MSB of base address PCIe window 3

**PCIE\_AXI\_SLAVE\_WINDOW3\_0 Register(0F0h)****Table 74 • PCIE\_AXI\_SLAVE\_WINDOW3\_0**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW3_0_31_12	0x0	Base address AXI3 slave window 3
[11:0]	Reserved	0x0	Reserved

**PCIE\_AXI\_SLAVE\_WINDOW3\_1 Register(0F4h)****Table 75 • PCIE\_AXI\_SLAVE\_WINDOW3\_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW3_1_31_12	0x0	Size of AXI3 slave window 3
[11:1]	Reserved	0x0	Reserved
0	PCIE_AXI_SLAVE_WINDOW3_1_0	0x0	Enable bit of AXI3 slave window 3

**PCIE\_AXI\_SLAVE\_WINDOW3\_2 Register(0F8h)****Table 76 • PCIE\_AXI\_SLAVE\_WINDOW3\_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW3_2_31_12	0x0	LSB of the base address PCIe window 3
[11:5]	Reserved	0x0	Reserved
[4:2]	PCIE_AXI_SLAVE_WINDOW3_2_4_2	0x0	AXI3 slave window 0 traffic class (TC)
1	PCIE_AXI_SLAVE_WINDOW3_2_1	0x0	AXI3 Slave window 0 relaxed ordering (RO)
0	PCIE_AXI_SLAVE_WINDOW3_2_0	0x0	AXI3 Slave window 0 no snoop (NS)

**PCIE\_AXI\_SLAVE\_WINDOW3\_3 Register(0FCh)****Table 77 • PCIE\_AXI\_SLAVE\_WINDOW3\_3**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_SLAVE_WINDOW3_3_31_0	0x0	MSB of the base address PCIe window 0

**PCIE\_AXI\_MASTER\_WINDOW0\_0 Register(100h)****Table 78 • PCIE\_AXI\_MASTER\_WINDOW0\_0**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW0_0_31_12	0x0	Base address AXI3 master window 0
[11:0]	Reserved	0x0	Reserved

**PCIE\_AXI\_MASTER\_WINDOW0\_1 Register(104h)****Table 79 • PCIE\_AXI\_MASTER\_WINDOW0\_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW0_1_31_12	0x0	Size of AXI3 master window 0
[11:1]	Reserved	0x0	Reserved
0	PCIE_AXI_MASTER_WINDOW0_1_0		Enable bit of AXI3 master window 0

**PCIE\_AXI\_MASTER\_WINDOW0\_2 Register(108h)****Table 80 • PCIE\_AXI\_MASTER\_WINDOW0\_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW0_2_31_12	0x0	LSB of the base address PCIe window 0
[11:6]	Reserved	0x0	Reserved
[5:0]	PCIE_AXI_MASTER_WINDOW0_2_5_0	0x0	These bits set the BAR. To select a BAR, set the following values: 0x01: BAR0 (32-bit BAR) or BAR0/1 (64-bit BAR) 0x02: BAR1 (32-bit BAR) only 0x04: BAR2 (32-bit BAR) or BAR2/3 (64-bit BAR) 0x08: BAR3 (32-bit BAR) only 0x10: BAR4 (32-bit BAR) or BAR4/5 (64-bit BAR) 0x20: BAR5 (32-bit BAR) only

**PCIE\_AXI\_MASTER\_WINDOW0\_3 Register(10Ch)****Table 81 • PCIE\_AXI\_MASTER\_WINDOW0\_3**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_MASTER_WINDOW0_3_31_0	0x0	MSB of the base address PCIe window 0

**PCIE\_AXI\_MASTER\_WINDOW1\_0 Register(110h)****Table 82 • PCIE\_AXI\_MASTER\_WINDOW1\_0**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW1_0_31_12	0x0	Base address AXI3 master window 1
[11:0]	Reserved	0x0	Reserved

**PCIE\_AXI\_MASTER\_WINDOW1\_1 Register(114h)****Table 83 • PCIE\_AXI\_MASTER\_WINDOW1\_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW1_1_31_12	0x0	Size of the AXI3 master window 1
[11:1]	Reserved	0x0	Reserved
0	PCIE_AXI_MASTER_WINDOW1_1_0	0x0	Enable bit of the AXI3 master window 1

**PCIE\_AXI\_MASTER\_WINDOW1\_2 Register(118h)****Table 84 • PCIE\_AXI\_MASTER\_WINDOW1\_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW1_2_31_12	0x0	LSB of the base address PCIe window 1
[11:6]	Reserved	0x0	Reserved
[5:0]	PCIE_AXI_MASTER_WINDOW1_2_5_0	0x0	These bits set the BAR. To select a BAR, set the following values: 0x01: BAR0 (32-bit BAR) or BAR0/1 (64-bit BAR) 0x02: BAR1 (32-bit BAR) only 0x04: BAR2 (32-bit BAR) or BAR2/3 (64-bit BAR) 0x08: BAR3 (32-bit BAR) only 0x10: BAR4 (32-bit BAR) or BAR4/5 (64-bit BAR) 0x20: BAR5 (32-bit BAR) only

**PCIE\_AXI\_MASTER\_WINDOW1\_3 Register(11Ch)****Table 85 • PCIE\_AXI\_MASTER\_WINDOW1\_3**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_MASTER_WINDOW1_3_31_0	0x0	MSB of the base address PCIe window 1

**PCIE\_AXI\_MASTER\_WINDOW2\_0 Register(120h)****Table 86 • PCIE\_AXI\_MASTER\_WINDOW2\_0**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW2_0_31_12	0x0	Base address AXI3 master window 2
[11:0]	Reserved	0x0	Reserved

**PCIE\_AXI\_MASTER\_WINDOW2\_1 Register(124h)****Table 87 • PCIE\_AXI\_MASTER\_WINDOW2\_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW2_1_31_12	0x0	Size of the AXI3 master window 2
[11:1]	Reserved	0x0	Reserved
0	PCIE_AXI_MASTER_WINDOW2_1_0	0x0	Enable bit of the AXI3 master window 2

**PCIE\_AXI\_MASTER\_WINDOW2\_2 Register(128h)****Table 88 • PCIE\_AXI\_MASTER\_WINDOW2\_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW2_2_31_12	0x0	LSB of the base address PCIe window 2
[11:6]	Reserved	0x0	Reserved
[5:0]	PCIE_AXI_MASTER_WINDOW2_2_5_0	0x0	These bits set the BAR. To select a BAR, set the following values: 0x01: BAR0 (32-bit BAR) or BAR0/1 (64-bit BAR) 0x02: BAR1 (32-bit BAR) only 0x04: BAR2 (32-bit BAR) or BAR2/3 (64-bit BAR) 0x08: BAR3 (32-bit BAR) only 0x10: BAR4 (32-bit BAR) or BAR4/5 (64-bit BAR) 0x20: BAR5 (32-bit BAR) only

**PCIE\_AXI\_MASTER\_WINDOW2\_3 Register(12Ch)****Table 89 • PCIE\_AXI\_MASTER\_WINDOW2\_3**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_MASTER_WINDOW2_3_31_0	0x0	MSB of the base address PCIe window 3

**AXI\_MASTER\_WINDOW3\_0 Register(130h)****Table 90 • PCIE\_AXI\_MASTER\_WINDOW3\_0**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_MASTER_WINDOW3_0_31_0	0x0	Base address AXI3 master window 3

**PCIE\_AXI\_MASTER\_WINDOW3\_1 Register(134h)****Table 91 • PCIE\_AXI\_MASTER\_WINDOW3\_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW3_1_31_12	0x0	Size of the AXI3 master window 3
[11:1]	Reserved	0x0	Reserved
0	PCIE_AXI_MASTER_WINDOW3_1_0	0x0	Enable bit of the AXI3 master window 3

**PCIE\_AXI\_MASTER\_WINDOW3\_2 Register(138h)****Table 92 • PCIE\_AXI\_MASTER\_WINDOW3\_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW3_2_31_12	0x0	LSB of the base address PCIe window 3
[11:5]	Reserved	0x0	Reserved
[5:0]	PCIE_AXI_MASTER_WINDOW3_2_5_0	0x0	These bits set the BAR. To select a BAR, set the following values: 0x01: BAR0 (32-bit BAR) or BAR0/1 (64-bit BAR) 0x02: BAR1 (32-bit BAR) only 0x04: BAR2 (32-bit BAR) or BAR2/3 (64-bit BAR) 0x08: BAR3 (32-bit BAR) only 0x10: BAR4 (32-bit BAR) or BAR4/5 (64-bit BAR) 0x20: BAR5 (32-bit BAR) only

**PCIE\_AXI\_MASTER\_WINDOW3\_3 Register(13Ch)****Table 93 • PCIE\_AXI\_MASTER\_WINDOW3\_3**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW3_3_31_12	0x0	MSB of the base address PCIe window 0
[11:0]	Reserved	0x0	Reserved

**PCIE\_INFO Register (016Ch)****Table 94 • PCIE\_INFO**

Bit Number	Name	Reset Value	Description
[31:12]	INFO_31_12	0x0	Bridge version
[11:0]	INFO_11_0	0x0	Reserved

**RESERVED Register (204h)****Table 95 • RESERVED**

Bit Number	Name	Reset Value	Description
[31:0]	Reserved	0x0	Reserved



**PCIE\_DEV2SCR Register (230h)****Table 96 • PCIE\_DEV2SCR**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_DEV2SCR_31_0	0x0	Reports the current value of the PCIe device control and status register. It can be monitored by the local processor when RO and NS bits are enabled in the system.

**PCIE\_LINK2SCR Register (234h)****Table 97 • PCIE\_LINK2SCR**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_LINK2SCR_31_0	0x0	Reports the current value of the PCIe link control and status register. It can be monitored by the local processor when RO and NS bits are enabled in the system.

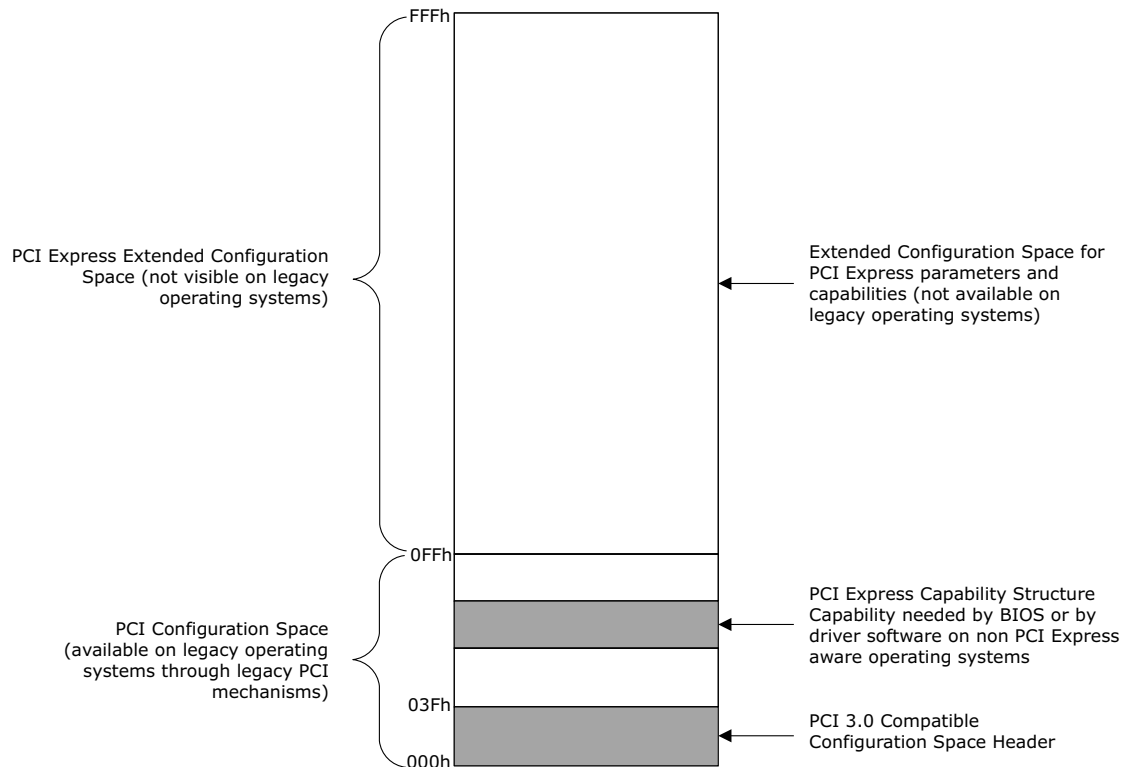
**RESERVED Register (260h)****Table 98 • RESERVED**

Bit Number	Name	Reset Value	Description
[31:0]	Reserved	0x0	Reserved

### 3.1.8 PCIe Configuration Space

The PCIe base IP core transaction layer (TL) contains the 4 Kbyte configuration space. The configuration space implements all configuration registers and associated functions. It manages BAR and window decoding, interrupt/MSI message generation, power management negotiation, and error handling. For upstream ports, the configuration space is accessed through the PCIe link using type 0 requests. Type 1 requests are forwarded to the application layer. For downstream ports, the configuration space is accessed through the application interface using Type 0 requests. Type 1 requests are forwarded to the PCIe link. The first 256 bytes of the configuration space are the function's configuration space, and the remaining configuration space is PCIe extended configuration space (see Figure 26, page 63).

**Figure 26 • PCIe Configuration Space**



### 3.1.8.1 Common Configuration Space Header

The following table lists the common configuration space header. The PCIe common configuration space includes the following registers:

- Type 0 configuration settings
- MSI capability structure
- Power management capability structure
- PCIe capability structure

For comprehensive information on these registers, refer to PCIe Base Specification Revision 1.0a, 1.1 or 2.0 specifications.

**Table 99 • Configuration Inputs for AXI3 Bridge**

31:24	23:16	15:8	7:0	Byte Offset
Type 0 configuration registers				000h...03Ch
Reserved				040h
Core version register				044h
Reserved				048h...04Ch
MSI capability structure				050...05Ch
Reserved				060h...064h
Power management capability structure				078...07Ch
PCIe capability structure				080h...0BCh
Reserved				0C8h...0FCh

### 3.1.8.2 PCIe Extended Capability Structure

The following table shows the PCIe extended capability structure. RTG4 PCIe common configuration space includes the following registers:

PCIe AER extended capability structure

**Table 100 • PCIe Extended Capability Structure (Function 0)**

31:24	23:16	15:8	7:0	Byte Offset
AER				800h.834h

### 3.1.8.3 Type 0 Configuration Settings

The following table lists the type 0 configuration settings.

**Table 101 • Type 0 Configuration Register**

31:24	23:16	15:8	7:0	Byte Offset
Device ID		Vendor ID		000h
Status		Command		004h
Class Code			Revision ID	008h
BIST	Header type	Latency timer	Cache line size	00Ch
Base address 0				010h
Base address 1				014h
Base address 2				018h

**Table 101 • Type 0 Configuration Register**

Base address 3		01Ch
Base address 4		020h
Base address 5		024h
Reserved		028h
Subsystem ID	Subsystem Vendor ID	02Ch
RESERVED		030h
Capabilities PTR		034h
Reserved		038h
Int. pin	Int. line	03Ch

### 3.1.8.4 IP Core Status Register

The following table lists the content of the IP Core Status Register.

**Table 102 • IP Core Status Register**

31:28	27:16	15:4	3:0	Byte Offset
Reserved	Core version	Signature	Reserved	044h

### 3.1.8.5 MSI Capability Structure

The following table lists the content of the MSI capability structure.

**Table 103 • MSI Capability Structure Register**

31:24	23:16	15:8	7:0	Byte Offset
Message control		Next pointer	Cap ID	050h
Message address				054h
Message upper address				058h
Reserved	Reserved	Message data		05Ch

### 3.1.8.6 Power Management Capability Structure

The following table lists the content of the power management capability structure.

**Table 104 • Power Management Capability Structure**

31:24	23:16	15:8	7:0	Byte Offset
Capabilities register		Next cap PTR	Cap ID	078h
Data	PM control/status bridge extensions	Power management status and control		07Ch

### 3.1.8.7 PCIe Capability Structure

The following table lists the content of the PCIe capability structure.

**Table 105 • PCIe Capability Structure Register**

31:24	23:16	15:8	7:0	Byte Offset
Capabilities register		Next cap PTR	cap ID	080h
Device capabilities				084h
Device status		Device control		088h
Link capabilities				08Ch
Link status		Link control		090h
Slot capabilities				094h
Slot status		Slot control		098h
Reserved		Root control		09Ch
Root status				0A0h
Device capabilities 2				0A4h
Device status 2		Device control 2		0A8h
Link capabilities 2				0ACh
Link status 2		Link control 2		0B0h

### 3.1.8.8 PCIe AER Extended Capability Structure

The following table lists the AER extended capability structure for function 0. For functions 1 - 7, the byte offset is from 100h to 134h.

**Table 106 • PCIe AER Extended Capability Structure**

31:24	23:16	15:8	7:0	Byte Offset
PCIe enhanced capability header				800h
Uncorrec error status register				804h
Uncorrec error mask register				808h
Uncorrec error severity register				80Ch
Correc error status register				810h
Correc error mask register				814h
Advanced error capabilities and control register				818h
Header log register				81Ch
Root error command				82Ch
Root error status				830h
Error source identification register		Correc error source ID register		834h

### 3.1.9 TLP Contents

The following tables describe the contents of all TLPs. The bit assignments are mapped with the MSB in the top-left and the LSB in the bottom-right of the tables.

### 3.1.9.1 Content of a TLP without a Data Payload

**Table 107 • Memory Read Request 32-bit Addressing Descriptor Format**

	+0								+1								+2								+3													
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0						
Byte 0	0	0	0	0	0	0	0	0	0	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length														
Byte 4	Requester ID																Tag								Last BE								First BE					
Byte 8	Address [31:2]																														0				0			
Byte 12	Reserved																																					

**Table 108 • Memory Read Request-Locked 32-bit Addressing Descriptor Format**

	+0								+1								+2								+3									
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		
Byte 0	0	0	0	0	0	0	0	1	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length											
Byte 4	Requester ID																Tag								Last BE								First BE	
Byte 8	Address [31:2]																														0		0	
Byte 12	Reserved																																	

**Table 109 • Memory Read Request 64-bit Addressing Descriptor Format**

	+0								+1								+2								+3															
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0								
Byte 0	0	0	1	0	0	0	0	0	0	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length																
Byte 4	Requester ID																Tag								Last BE								First BE							
Byte 8	Address [63:32]																																							
Byte 12	Address [31:2]																																0				0			

**Table 110 • Memory Read Request-Locked 64-bit Addressing Descriptor Format**

	+0								+1								+2								+3															
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0								
Byte 0	0	0	1	0	0	0	0	1	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length																	
Byte 4	Requester ID																Tag								Last BE								First BE							
Byte 8	Address[63:32]																																							
Byte 12	Address [31:2]																																0				0			

**Table 111 • Type 0 Configuration Read Request Descriptor Format**

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0 0 0 0 First BE							
Byte 8	Bus Number								Device Nb.								Func		0	0	0	0	Ext. Reg.				Register Nb.				0 0	
Byte 12	R																															

**Table 112 • Type 0 Configuration Read Request Descriptor Format**

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0 0 0 0				First BE			
Byte 8	Bus Number								Device Nb.				Func				0	0	0	0	Ext. Reg.				Register Nb.				0 0			
Byte 12	R																															

**Table 113 • Message (without data) Descriptor Format**

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	1	1	0	r	r	r	0	TC			0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	0
						2	1	0																								
Byte 4	Requester ID																Tag								Message code							
Byte 8	Vendor defined or all zeros																															
Byte 12	Vendor defined or all zeros																															

**Table 114 • Completion (without data) Descriptor Format**

	+0								+1								+2								+3												
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0					
Byte 0	0	0	0	0	1	0	1	0	0	TC			0	0	0	0	TD	EP	Attr	0	0	Length															
Byte 4	Completer ID																Status				B	Byte count															
Byte 8	Requester ID																Tag				0 Lower address																
Byte 12	R																																				

**Table 115 • Completion Locked (without data) Descriptor Format**

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	1	0	1	1	0	TC			0	0	0	0	TD	EP	Attr	0	0	Length										
Byte 4	Completer ID																Status				B Byte count											
Byte 8	Requester ID																Tag				0 Lower address											
Byte 12																																

### 3.1.9.2 Content of a TLP with a Data Payload

**Table 116 • Memory Write Request 32-bit Addressing Descriptor Format**

	+0								+1								+2								+3															
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0								
Byte 0	0	1	0	0	0	0	0	0	0	0	TC			0	0	0	0	TD	EP	Attr	0	0	Length																	
Byte 4	Requester ID																Tag								Last BE								First BE							
Byte 8	Address [31:2]																																0				0			
Byte 12	R																																							

**Table 117 • Memory Write Request 64-bit Addressing Descriptor Format**

	+0								+1								+2								+3															
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0								
Byte 0	0	1	1	0	0	0	0	0	0	TC			0	0	0	0	TD	EP	Attr	0	0	Length																		
Byte 4	Requester ID																Tag								Last BE								First BE							
Byte 8	Address [63:32]																																							
Byte 12	Address [31:2]																																0				0			

**Table 118 • Type 0 Configuration Write Request Descriptor Format**

	+0								+1								+2								+3															
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0								
Byte 0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	TD	EP	00	0	0	0	0	0	0	0	0	0	0	0	0	1								
Byte 4	Requester ID																Tag								0 0 0 0 First BE															
Byte 8	Bus Number								Device Nb.								Func		0	0	0	0	Ext. Reg.				Register Nb.								0 0					
Byte 12	R																																							

**Table 119 • Type 0 Configuration Write Request Descriptor Format**

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0



**Table 119 • Type 0 Configuration Write Request Descriptor Format**

Byte 0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID												Tag				0				0	0	0	First BE				
Byte 8	Bus Number				Device Nb.				Func		0	0	0	0	Ext. Reg.				Register Nb.				0		0			
Byte 12	R																											

**Table 120 • Type 1 Configuration Write Request Descriptor Format**

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	TD	EP	00	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag				0 0 0 0				First BE							
Byte 8	Bus Number								Device Nb.				Func		0	0	0	0	Ext. Reg.				Register Nb.				0 0					
Byte 12	R																															

**Table 121 • Completion (with data) Descriptor Format**

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	1	0	1	0	0	TC			0	0	0	0	TD	EP	Attr	0	0	Length										
Byte 4	Completer ID																Status				B	Byte count										
Byte 8	Requester ID																Tag				0	Lower address										
Byte 12	R																															

**Table 122 • Completion Locked (with data) Descriptor Format**

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	1	0	1	1	0	TC			0	0	0	0	TD	EP	Attr	0	0	Length										
Byte 4	Completer ID																Status				B	Byte count										
Byte 8	Requester ID																Tag				0	Lower address										
Byte 12																																

**Table 123 • Message (with data) Descriptor Format**

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	1	1	0	r	r	r		TC			0	0	0	0	TD	EP	0	0	0	0	Length									
						2	1	0																								
Byte 4	Requester ID												Tag				Message code															

**Table 123 • Message (with data) Descriptor Format**

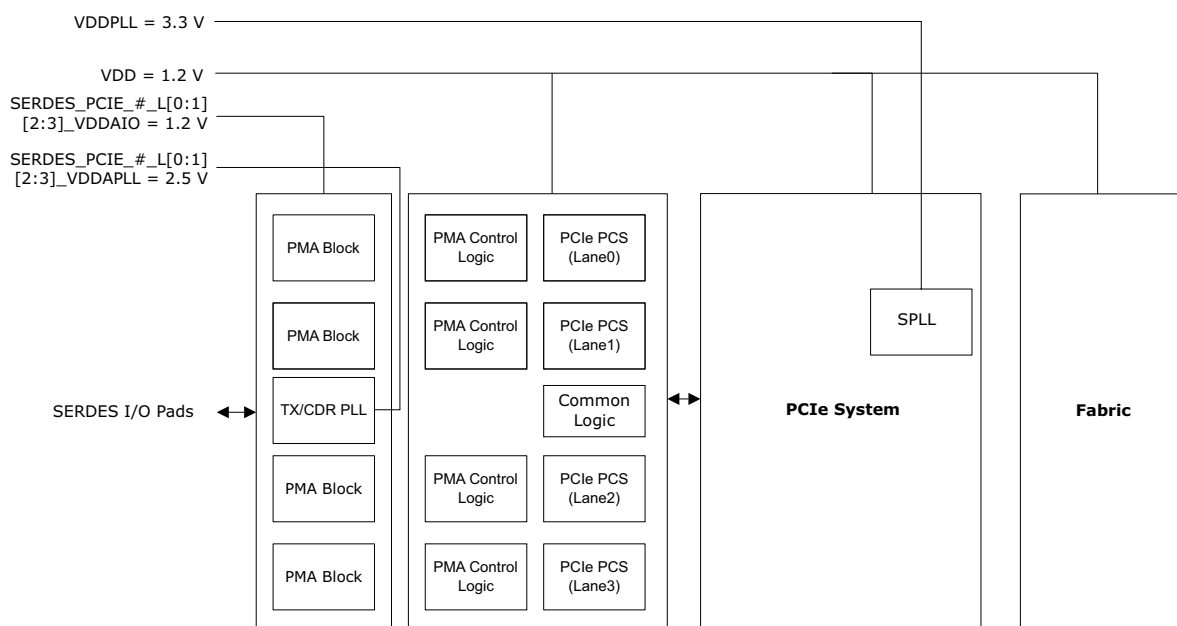
Byte 8	Vendor defined or all zeros for slot power limit
Byte 12	Vendor defined or all zeros for slots power limit

### 3.1.10 PCB Guidelines for PCIe

The following sections provide PCB guidelines for PCIe.

#### 3.1.10.1 Power Domain Implementation

In the RTG4 devices, the FPGA and PCIe link (including PMA, PCS, and PCIe controller) are combined in a single chip, so they have separate power supplies. The following figure shows the RTG4 power rails. Refer to the [AC439: Board Design Guidelines for RTG4 FPGA Application Note](#) for detailed power supply connections.

**Figure 27 • RTG4 Power Supply to the PCIe Link Implementation**

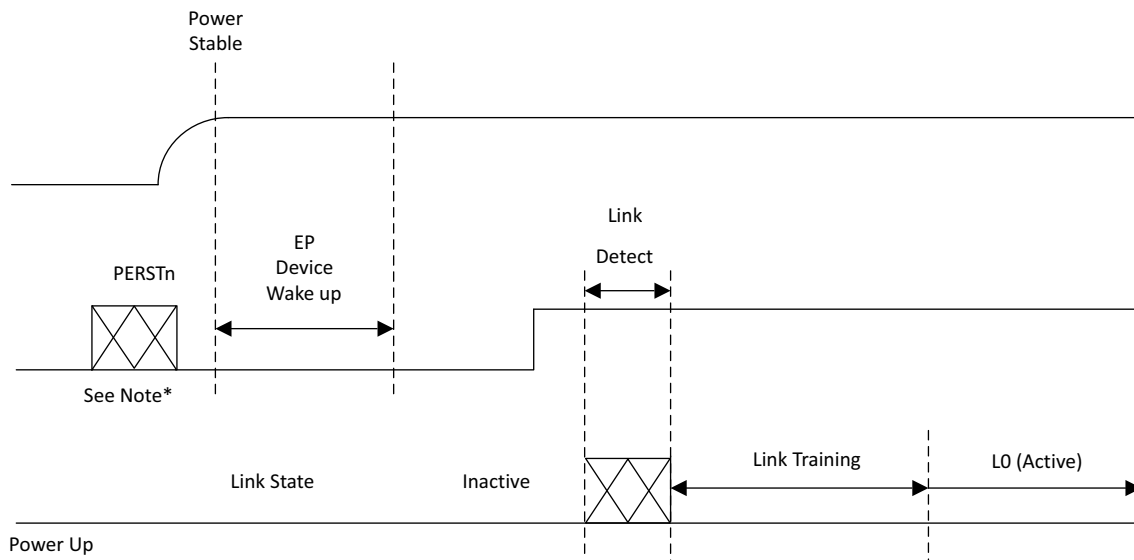
#### 3.1.10.2 PCI Express Power-Up

The PCI Express (PCIe) specification provides timing requirements for power-up. Power-up is a concern with SRAM-based FPGA endpoint devices, within these tight specifications. The PCIe specification specifies the release of the fundamental reset (PERST#) in the connector specification. The PERSTn release time (TPVPERL) of 100ms is used for the PCIe card electromechanical specification for add-in cards. From the point of power stable to at least 100 ms, the PERST# must remain asserted. Different PCIe systems hold PERSTn longer than 100 ms, but the minimum time is 100 ms.

The advantage of flash-based RTG4 device is that its wake up time is very fast in contrast to SRAM FPGA endpoints due to the configuration requirement. The semi-autonomous nature of the PCIe Core in the RTG4 device transitions from power-up to link detectable allowing the device to be detected by the root. When the device is detected by the root, it proceeds to the polling state of the LTSSM. The link now cycles through the remainder of the LTSSM.

In use cases where the root and endpoint power-up separately, the PERSTn signal must be used to handshake the link startups. The following figure shows the PCIe power-up states.

**Figure 28 • PCI Express Power-Up States**



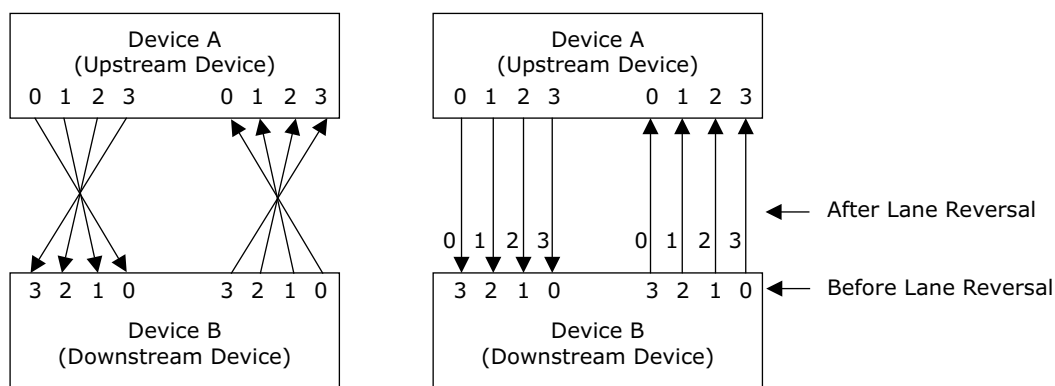
1. **EP Device Wake Up:** The internal flash loads the configuration information to the PCIe logic data of the device. If PERSTn is required, a fabric GPIO must be connected to the PERSTn of the root. This GPIO must be connected in the FPGA design to the CORE\_RESETn pin of the PCIe core. The embedded PCIe core is reset by PERSTn, and is released afterwards to start PCIe link detection and training.
2. **Link Detection:** Out-of-band pulse, looks for far-end connection.
3. The PCIe link completes the training phase and reaches the L0 state.
4. After the embedded PCIe endpoint core reaches the L0 state, the host operating system (OS) accesses the PCIe core's CSRs to perform configuration write access cycles that are part of the system enumeration process.

**Note:** PERSTn is controlled by the root. If not connected to the EP, the EP enters Link Detect as soon as the device wake up is complete.

### 3.1.10.3 Setting up Lane Reversal

Polarity swapping or inversion capability within a PCIe receive or transmit lane is not available in RTG4. PCIe system supports lane reversal capabilities and provides flexibility in the design of the board. The board layout can be chosen with reversed lane numbers and the PCIe EP continues to link train successfully and operate normally. The SerDes block configurator in Libero allows configuration of the SerDes block in reverse lane PCIe mode, as shown in the following figure.

**Figure 29 • Reversed Signals in PCIe Mode**



### 3.1.11 Generating PCIe Block Using Libero SoC

This section provides an overview of configuring the SerDes block in PCIe mode, simulating the SerDes block in PCIe mode, and implementing a PCIe EP in an RTG4 device.

The SerDes block configurator in the Libero SoC provides the configuration options for the PCIe EP implementation for PCIeSS SerDes blocks. It includes options for selecting the protocol for various SerDes lanes, serial rate settings, fabric interface, PCIe Identification registers, PCIe BAR. These settings are implemented during programming using dedicated flash bits for fast configuration or via the APB interface. These registers can be reviewed using the APB interface, which can access all the registers in the SerDes block.

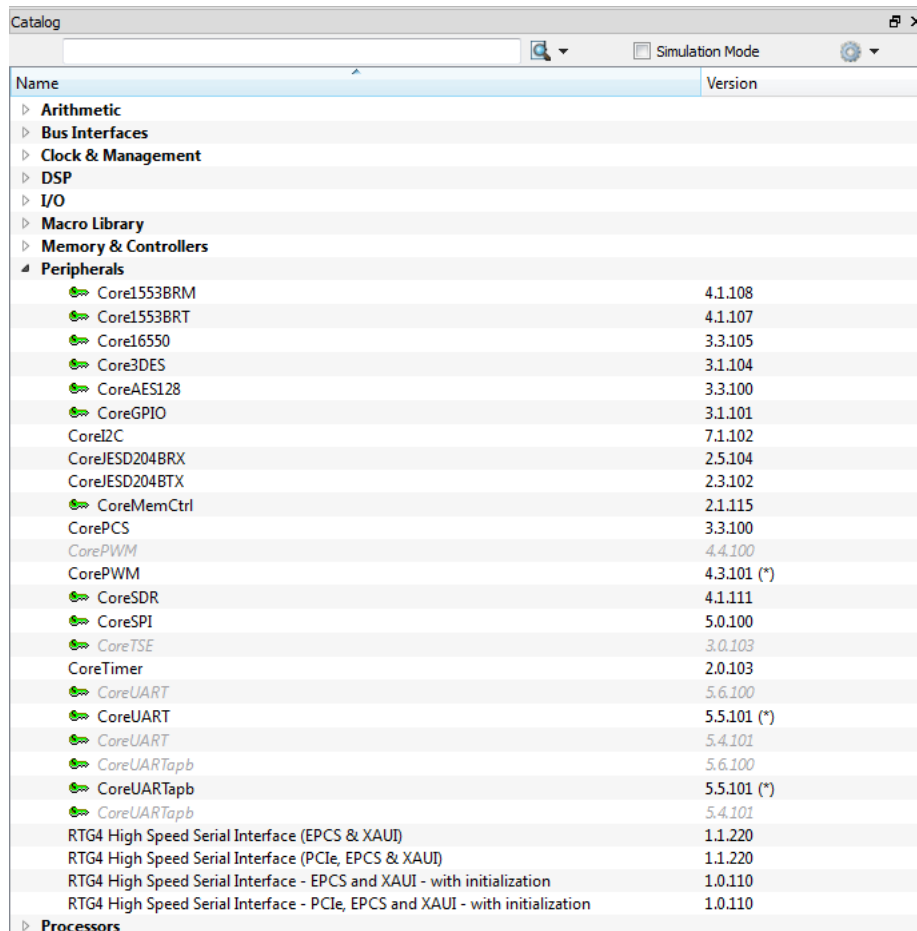
The following sub-sections show how to instantiate PCIe in a design:

- Configuring High Speed Serial Interface Configurator for PCIe
- Simulating SerDes Block in PCIe Mode
- Adding RTG4 PCIe Block to User Design

#### 3.1.11.1 Configuring High Speed Serial Interface Configurator for PCIe

This section describes configuring and generating the SerDes block for PCIe EP mode using the Libero SoC software. High speed serial interface blocks depending on its size. Refer to the *DS0131: RTG4 Datasheet*. The Libero SoC software catalog includes four SerDes cores in the peripherals subsection, as shown in the Figure 30, page 73.

**Figure 30 • Libero SoC Catalog**



Name	Version
Arithmetic	
Bus Interfaces	
Clock & Management	
DSP	
I/O	
Macro Library	
Memory & Controllers	
Peripherals	
Core1553BRM	4.1.108
Core1553BRT	4.1.107
Core16550	3.3.105
Core3DES	3.1.104
CoreAES128	3.3.100
CoreGPIO	3.1.101
CoreI2C	7.1.102
CoreJESD204BRX	2.5.104
CoreJESD204BTX	2.3.102
CoreMemCtrl	2.1.115
CorePCS	3.3.100
CorePWM	4.4.100
CorePWM	4.3.101 (*)
CoreSDR	4.1.111
CoreSPI	5.0.100
CoreTSE	3.0.103
CoreTimer	2.0.103
CoreUART	5.6.100
CoreUART	5.5.101 (*)
CoreUART	5.4.101
CoreUARTapb	5.6.100
CoreUARTapb	5.5.101 (*)
CoreUARTapb	5.4.101
RTG4 High Speed Serial Interface (EPCS & XAUI)	1.1.220
RTG4 High Speed Serial Interface (PCIe, EPCS & XAUI)	1.1.220
RTG4 High Speed Serial Interface - EPCS and XAUI - with initialization	1.0.110
RTG4 High Speed Serial Interface - PCIe, EPCS and XAUI - with initialization	1.0.110
Processors	

There are two specific cores that can be selected for PCIe applications. These cores can only target one of the two RTG4 SerDes blocks with PCIe functionality.

This core is used where the user design includes all the needed functionality to configure and initialize the SerDes block. In addition to specifying SerDes configuration register values the PCIe SerDes initialization solution requires, building the configuration and initialization circuitry in SmartDesign for the PCIe SerDes. To help build the initialization circuitry, Microsemi provides the **pcie\_init** component ready for import as a block into the Libero SoC project. The core is located in <LiberoSoC\_installation>/Designer/templates/rtg4/pcie\_init.cxz.

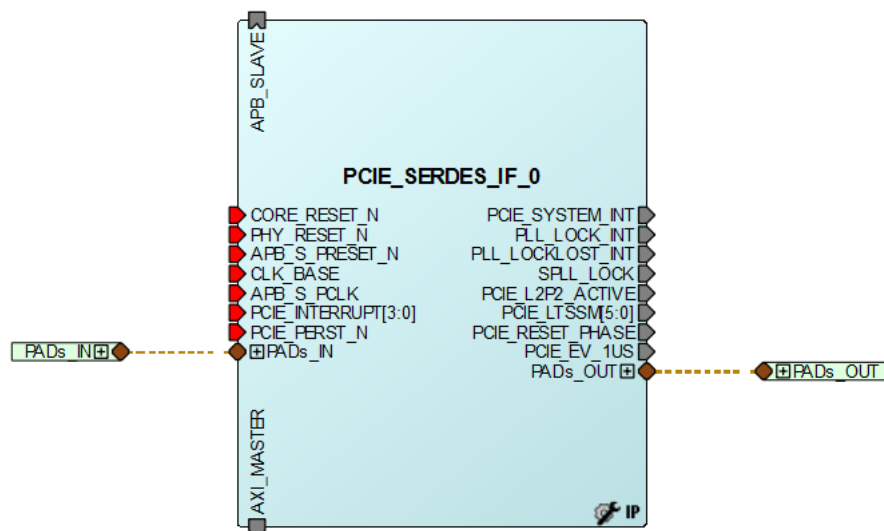
The pcie\_init component consists of:

- CoreABC soft IP core
- CoreAPB3 bus soft IP core

To access the high speed serial interface (PCIe, EPCS and XAUI) configurator:

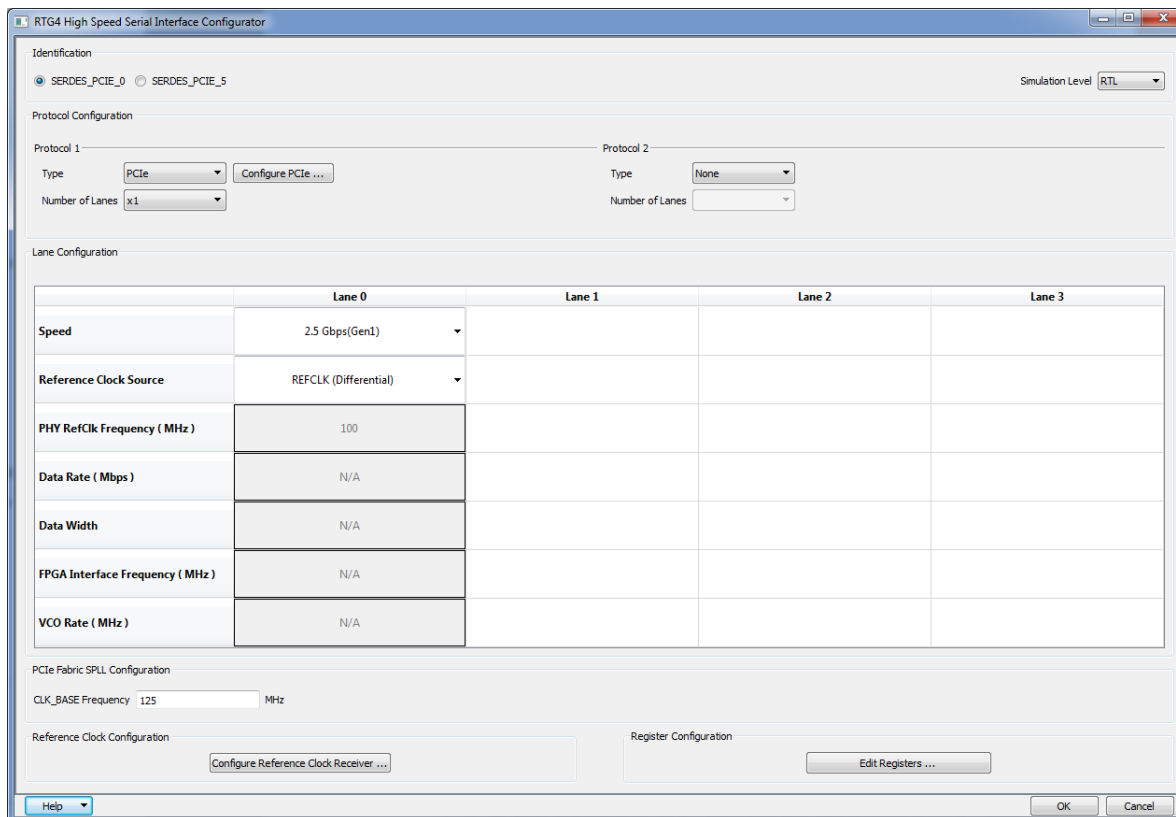
1. Instantiate the RTG4 high speed serial interface (PCIe, EPCS and XAUI) core from the catalog under the peripherals group into the SmartDesign canvas.

**Figure 31 • PCIe SerDes Block Instantiation without Initialization feature on the SmartDesign Canvas**



- Double-clicking the PCIe SerDes block on the canvas opens the configurator permitting customization of the PCIe features.

**Figure 32 • RTG4 High Speed Serial Interface (PCIe, EPCS and XAUI) Configurator**



The configurator window is titled "RTG4 High Speed Serial Interface Configurator". It contains several sections:

- Identification:** Radio buttons for `SERDES_PCIE_0` (selected) and `SERDES_PCIE_5`. A "Simulation Level" dropdown is set to "RTL".
- Protocol Configuration:**
  - Protocol 1:** Type is "PCIe" (dropdown), with a "Configure PCIe ..." button. Number of Lanes is "x1" (dropdown).
  - Protocol 2:** Type is "None" (dropdown). Number of Lanes is empty.
- Lane Configuration:** A table with columns for Lane 0, Lane 1, Lane 2, and Lane 3.
 

	Lane 0	Lane 1	Lane 2	Lane 3
Speed	2.5 Gbps(Gen1)			
Reference Clock Source	REFCLK (Differential)			
PHY RefClk Frequency ( MHz )	100			
Data Rate ( Mbps )	N/A			
Data Width	N/A			
FPGA Interface Frequency ( MHz )	N/A			
VCO Rate ( MHz )	N/A			
- PCIe Fabric SPLL Configuration:** CLK\_BASE Frequency is 125 MHz.
- Reference Clock Configuration:** A "Configure Reference Clock Receiver ..." button.
- Register Configuration:** An "Edit Registers ..." button.

Buttons at the bottom include "Help", "OK", and "Cancel".

For the RTG4 high speed serial interface (PCIe, EPCS and XAUI), PCIe is available in the following SerDes blocks:

- `SERDES_PCIE_0`
- `SERDES_PCIE_5`

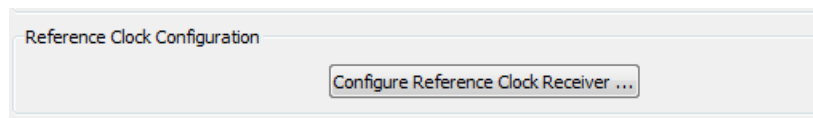
**Note:** The SerDes block names are dependent on the number of blocks present in the device. The SerDes blocks shown are based on the RT4G150 device.

### 3.1.11.1.1 Protocol Selection

The following settings are used for protocol selection.

- Protocol Type 1: Select **PCIe** from the drop-down menu.
- Number of Lanes: Select **number of lanes** used.
- Protocol 1 PHY Reference Clock: Select the inputs for the PHY reference clock selection. Only **REFCLK (Differential)** is specified for PCIe applications. Users should customize the reference clock using the **Reference Clock Configuration** feature.

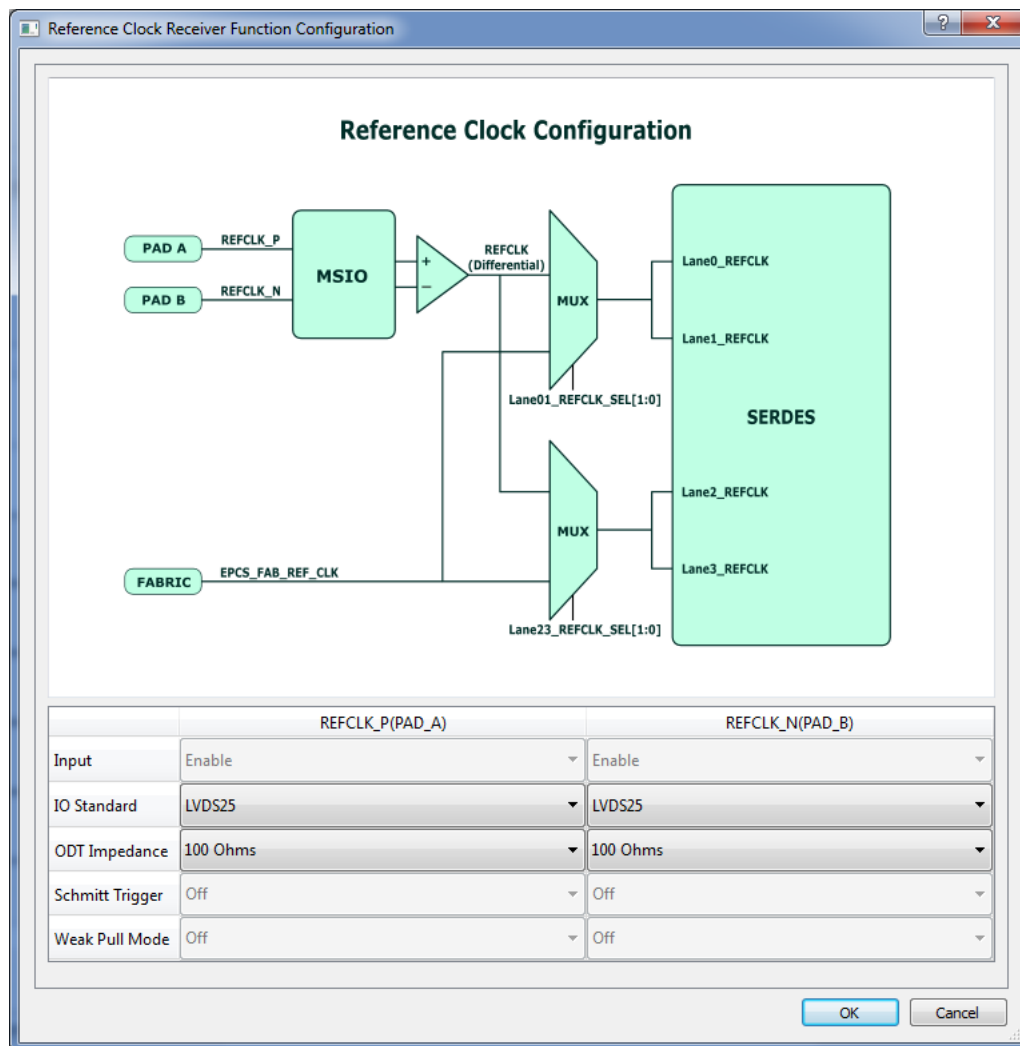
**Figure 33 • Reference Clock Configuration**



The dialog box is titled "Reference Clock Configuration". It contains a single button labeled "Configure Reference Clock Receiver ...".

As shown in the following figure, the reference clock configurator GUI is launched using the clicking the Configure Reference Clock Receiver. This GUI allows the user to select various reference clock input features. For PCIe protocol, users are recommended to use the setting shown.

**Figure 34 • Reference Clock Configurator**



The high speed serial interface configurator (SerDes block configurator) in Libero allows configuration of the SerDes block in PCIe mode. In addition to the protocol settings, various other options for PCIe implementation are preset. Following is a brief description of the various protocol configuration options.

### 3.1.11.1.2 PCIe SPLL Configuration

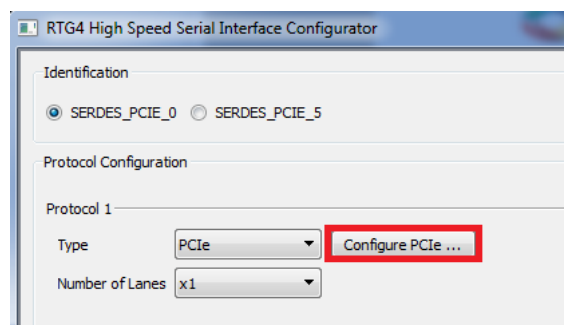
These settings are used for SPLL that synchronizes data between the AXI3 interface to the SerDes block.

- **CLK\_BASE:** This is an AXI3 clock setting.
- The SPLL is a PLL embedded in the SerDes block to manage the clock phase, used for transfers across the SerDes block to FPGA fabric interface. The SPLL configuration fields are relevant for PCIe protocol, 125 MHz is a typical value for PCIe applications.
- **CLK\_BASE** frequency must be set to the same frequency of the APB interface as the operating frequency. These settings select the PCIe system interfaces to the fabric. It can be AXI3 bus as master only, slave only, or both master and slave.

### 3.1.11.1.3 Configure PCIe

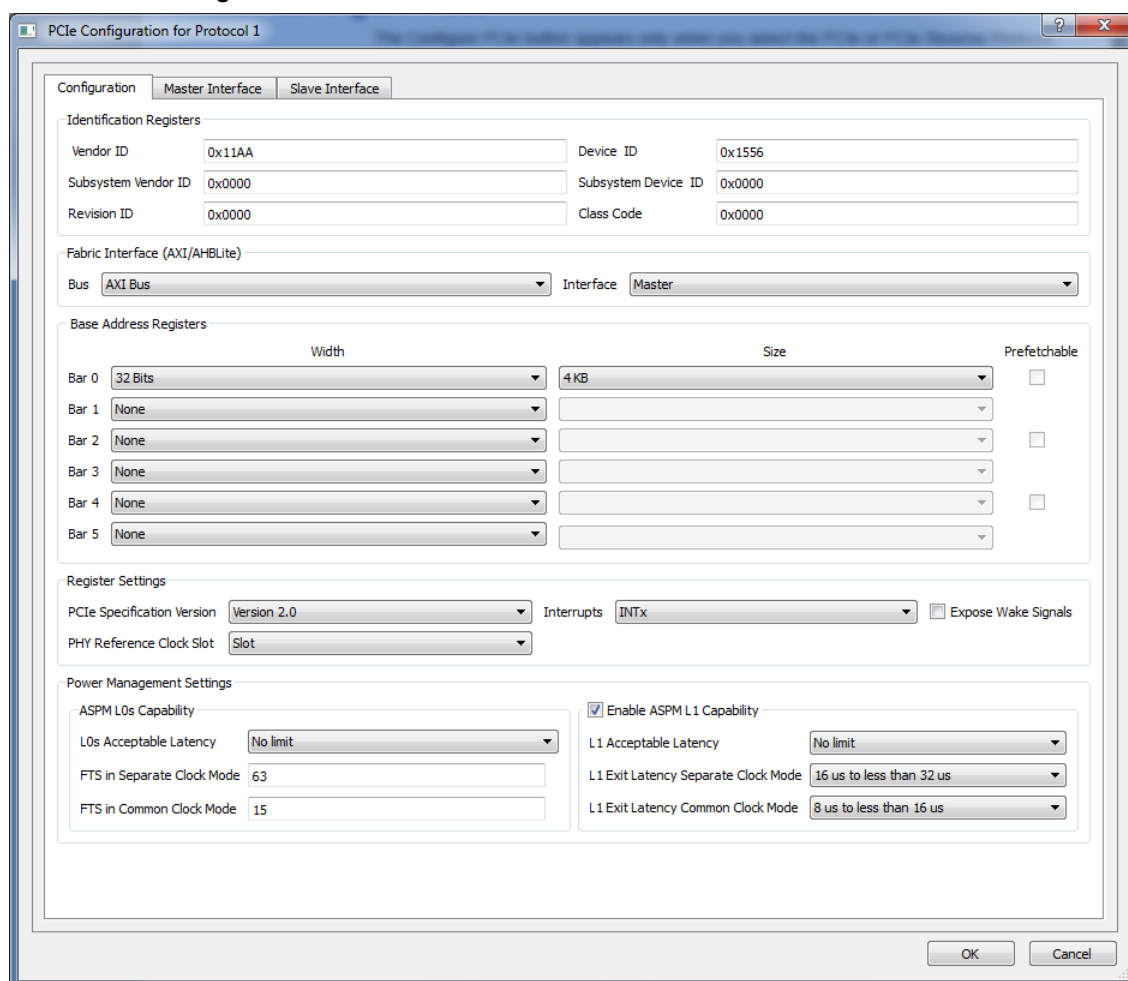
The **Configure PCIe...** as shown in the following figure appears only when the PCIe or PCIe reverse protocol is selected.

**Figure 35 • Configuring PCIe**



The **Configuration** tab sets the identification registers, fabric interface, base address registers.

**Figure 36 • PCIe Configuration Tab**





### 3.1.11.1.4 Identification Registers

User can assign 16-bit hexadecimal signatures to the following six identification registers for PCIe:

- Vendor ID - 0x11AA is the vendor ID assigned to Microsemi by PCI-SIG.
- Subsystem vendor ID - Card manufacturer's ID.
- Device ID - Manufacturer's assigned part number assigned by the vendor.
- Revision ID - Revision number, if available.
- Subsystem Device ID - Assigned by the subsystem vendor.
- Class Code - PCIe device's generic function.

### 3.1.11.1.5 Fabric Interface

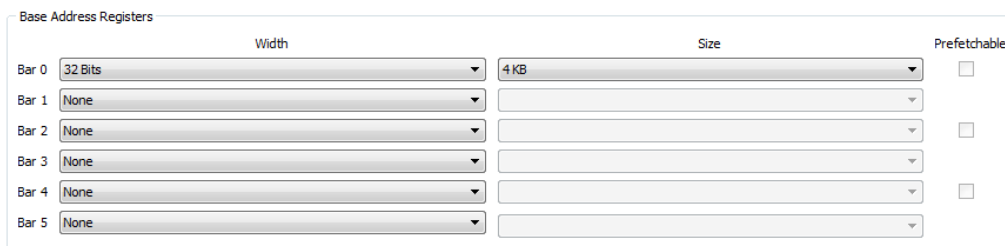
Use this field to configure the bus standard (AXI) and the Interface (Master Only, Slave Only, or both) for the PCIe protocol.

In PCIe mode, the SerDes block can act as an AXI master. Microsemi recommends that a COREAXI or CoreAHBLite bus is instantiated into the SmartDesign canvas and the Master and/or Slave Bus Interface (BIF) of the SerDes is connected to the Master and/ or Slave BIF of the COREAXI bus or COREAHBLite bus.

Microchip recommends new and on-going RTG4 designs requiring SerDes configured as PCIe with AHBLite master should use the CorePCle\_AXItoAHBL DirectCore. In addition, designs with both master and slave AHBLite interfaces are required to use the CorePCle\_AXItoAHBL and CorePCle\_AHBLtoAXI DirectCores.

## 3.1.11.2 Base Address Registers

**Figure 37 • Base Address Registers**



The figure shows a configuration window titled "Base Address Registers". It contains a table with six rows, one for each BAR (Bar 0 to Bar 5). Each row has three columns: "Width", "Size", and "Prefetchable".

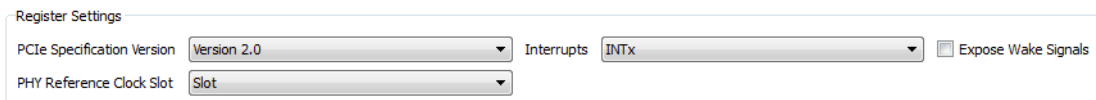
	Width	Size	Prefetchable
Bar 0	32 Bits	4 KB	<input type="checkbox"/>
Bar 1	None		<input type="checkbox"/>
Bar 2	None		<input type="checkbox"/>
Bar 3	None		<input type="checkbox"/>
Bar 4	None		<input type="checkbox"/>
Bar 5	None		<input type="checkbox"/>

The individual fields of the six BARs (Bar 0 through Bar 5) can be configured as follows:

- Width - The width on even registers can be 32 bit or 64 bit. If an even register is set to 64 bits wide, the subsequent (odd) register serves as the upper half of 64 bits. Otherwise, the width of the odd registers is restricted to 32 bits
- Size - Ranges from 4 KB to 2 GB. Some devices may go up to 1 GB. Refer to device datasheet (*DS0131: RTG4 FPGA Datasheet*) for more information
- Prefetchable - Enabled only on even registers with 64-bit width

### 3.1.11.3 PCIe Register Options

**Figure 38 • PCIe Register Options**



The figure shows a configuration window titled "Register Settings". It contains three main settings:

- PCIe Specification Version: Version 2.0
- Interrupts: INTx
- Expose Wake Signals: ☐
- PHY Reference Clock Slot: Slot

- PHY Reference Clock Slot - Sets the reference clock to slot or independent.  
 PHY Reference Clock Slot: Select this option if the PHY reference clock is sourced from a PCIe slot or it is generated separately  
 Slot refers to a clock source that is shared in the PCIe system between both ends of the link. The other setting, Independent, is used in a system which uses independent clock sources on either side of the link. This setting changes the PCIe configuration space register to advertise to the system root, which clocking topology is used. It makes no other functional changes to the endpoint. Expose wake signals: Selecting this option adds PCIE\_WAKE\_N, PCIE\_WAKE\_REQ, and PCIE\_PERST\_N ports to control the L2/P2 state.

Entry of L2P2 can only be requested by the host and when requested, the RTG4 PCIe core responds to the protocol request and transitions to L2 state. Its response is in conjunction with the CFGR\_L2\_P2\_ENABLE bit of the CONFIG\_PCIE\_PM register. If the bit is configured as CFGR\_L2\_P2\_ENABLE=0 or =1, the device goes into L2, and needs a fundamental reset (PCIE\_PERST\_N) to get out of L2.

If CFGR\_L2\_P2\_ENABLE=1, the device shows much lower power when it enters L2. In either case the host needs to perform enumeration.

PCIE\_WAKE\_REQ is an input to the RTG4 PCIe reset controller. When PCIE\_WAKE\_REQ is asserted, the PCIe reset controller drives out the PCIE\_WAKE to RC as WAKE# side band signal

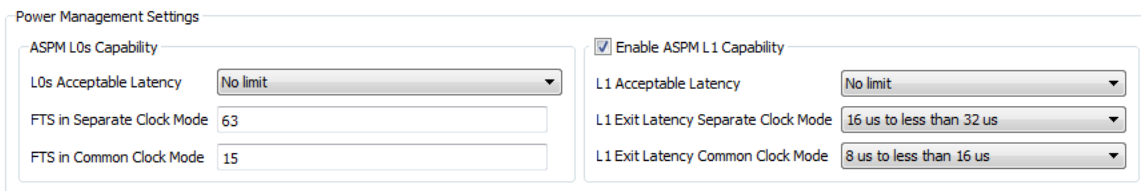
- PCIe specification version - Sets the specification version to 1.0, 1.1 or 2.0.
- Interrupt - Sets the Interrupt to the following values:
  - MSI 1
  - MSI 2
  - MSI 4
  - MSI 8
  - MSI 16
  - MSI 32
  - INTx

The interrupt selection sets bit 16 of the PCIE\_MSI\_CTRL\_STATUS register and bits [19:17] of the PCIE\_MSI\_CTRL\_STATUS register. RTG4 does not support MSI-X.

### 3.1.11.4 Power Management Settings

Use the Power Management Settings to configure the active state power management (ASPM) settings. The configurator sets the correct register values for the SerDes block based on the selections made.

**Figure 39 • Power Management Settings**



The screenshot shows the 'Power Management Settings' window. It is divided into two main sections. The left section, titled 'ASPM L0s Capability', contains three settings: 'L0s Acceptable Latency' set to 'No limit', 'FTS in Separate Clock Mode' set to '63', and 'FTS in Common Clock Mode' set to '15'. The right section, titled 'Enable ASPM L1 Capability' (which is checked), contains three settings: 'L1 Acceptable Latency' set to 'No limit', 'L1 Exit Latency Separate Clock Mode' set to '16 us to less than 32 us', and 'L1 Exit Latency Common Clock Mode' set to '8 us to less than 16 us'.

#### ASPM L0s Capability

This is mandatory. Settings available are:

- L0s Acceptable Latency - Opens the pull-down list to choose one of the following latency values:
  - Maximum of 64 ns
  - Maximum of 128 ns
  - Maximum of 256 ns
  - Maximum of 512 ns
  - Maximum of 1  $\mu$ s
  - Maximum of 2  $\mu$ s
  - Maximum of 4  $\mu$ s
  - No Limit
- FTS in Separate Clock Mode - Enter the number of Fast Training Sequences (FTS) required in separate clock mode. Valid values are from 0 through 255.
- FTS in Common Clock Mode - Enter the number of FTS required in common clock mode. Valid values are from 0 through 255.

### ASPM L1 Capability

By default, the ASPM L1 Capability is enabled. Configure the settings for ASPM L1 as follows:

- L1 Acceptable Latency - Click the pull-down list to choose one of the of following for latency values:
  - Maximum of 1  $\mu$ s
  - Maximum of 2  $\mu$ s
  - Maximum of 4  $\mu$ s
  - Maximum of 8  $\mu$ s
  - Maximum of 16  $\mu$ s
  - Maximum of 32  $\mu$ s
  - Maximum of 64  $\mu$ s
  - No Limit
- L1 Exit Latency Separate Clock/Common Clock Mode - Click the pull-down menu to choose one of the following values for the Exit Latency value of Separate/Common Clock Mode:
  - Less than 1 us
  - 1 us to less than 2  $\mu$ s
  - 2 us to less than 4  $\mu$ s
  - 4 us to less than 8  $\mu$ s
  - 8 us to less than 16  $\mu$ s
  - 16 us to less than 32  $\mu$ s
  - 32 us to 64  $\mu$ s
  - More than 64  $\mu$ s

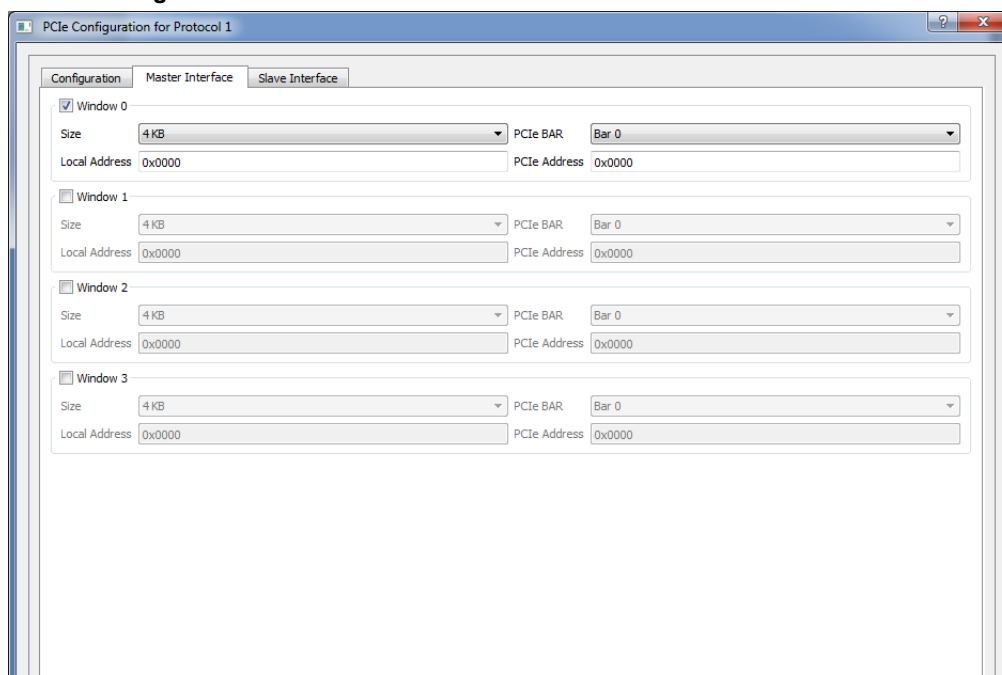
**Note:** The Exit Latency value for the Common Clock Mode must be smaller than the value for the separate clock mode.

### 3.1.11.5 Master Interface

The **Master Interface** tab allows to set up local AXI addresses on the master interface mapped to or from a PCIe BAR and the address offset of the BAR. PCIe or PCIe reverse protocol enables to use the Master Interface tab to configure up to four PCI windows (Window 0 through Window 3).

- Size
- PCIe BAR
- Local Address
- PCIe Address

**Figure 40 • PCIe Configuration - Master Interface Tab**



The screenshot shows the 'PCIe Configuration for Protocol 1' window with the 'Master Interface' tab selected. The 'Configuration' tab is also visible. The 'Slave Interface' tab is disabled. The 'Window 0' section is active, showing a checked checkbox, a 'Size' dropdown set to '4 KB', a 'PCIe BAR' dropdown set to 'Bar 0', a 'Local Address' text field set to '0x0000', and a 'PCIe Address' text field set to '0x0000'. Below this, there are three more sections for 'Window 1', 'Window 2', and 'Window 3', each with an unchecked checkbox and the same default settings for Size, PCIe BAR, Local Address, and PCIe Address.

### 3.1.11.5.1 Size

For each of the windows 0 through 3, select one of the available window sizes: 4KB, 8KB, 16KB, 32KB, 64KB, 128KB, 256KB, 512KB, 1MB, 2MB, 4MB, 8MB, 16MB, 32MB, 64MB, 128MB, 256MB, 512MB, and 1GB.

The default selection is 4KB. The size selected is mapped to bits [31:12] of WindowsX\_1 where X can be 0, 1, 2, or 3.

### 3.1.11.5.2 PCIe BAR

Select one of the following BARs:

- BAR0
- BAR1
- BAR2
- BAR3
- BAR4
- BAR5
- BAR0/1
- BAR2/3
- BAR4/5

Bar size is mapped to bits [5:0] of WindowsX\_2, where X can be 0, 1, 2 or 3.

### 3.1.11.5.3 Local Address

The address set in this field is the base address where transactions matching the window are mapped. Local Address is 20 bits wide and covers the AXI address [31:12]. The lower order AXI [11:0] are all zero in this control.

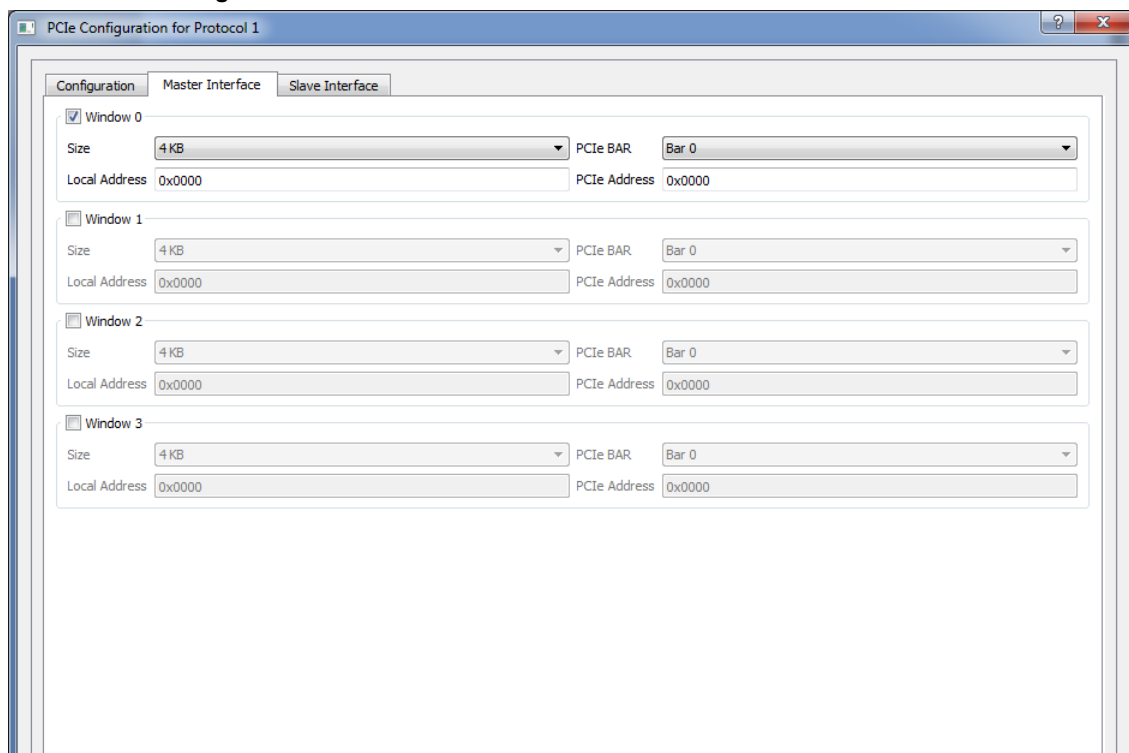
### 3.1.11.5.4 PCIe Address

The address set in this field is the offset address inside the selected BAR to which the transaction address is matched.

### 3.1.11.6 Slave Interface

- The Slave Interface tab allows to set a local AXI address on the slave interface that is mapped to a PCIe address.
- Size
- Local Address
- PCIe Address
- Traffic Class: Selects the PCIe traffic class in the PCIe packet header.
- Relaxed Ordering: Enables the generation of the PCIe TLP using a selectable relaxed ordering bit.
- No Snoop: Enables the generation of the PCIe TLP using a selectable no snoop bit.

**Figure 41 • PCIe Configuration - Slave Interface Tab**



**Note:** The Size, Local Address and PCIe Address options are the same as those for the Master Interface.

#### 3.1.11.6.1 Traffic Class

Traffic class enables to set the Traffic Class and corresponding register bits. The traffic class is required by the PCI Express packet header. Its value determines the relative priority of a given transaction as it traverses the PCIe link. The following traffic class values are used to create a priority scheme for different packets.

- TC 0 (Default)
- TC 1
- TC 2
- TC 3
- TC 4
- TC 5
- TC 6
- TC 7

### 3.1.11.7 Simulating SerDes Block in PCIe Mode

- BFM\_PCIe - In this mode, the APB bus is accessed similar to the BFM\_CFG mode. In addition, communication with the SerDes block through the master and slave AXI bus interfaces can be done. This mode is intended for the validation of the fabric interfaces to the SerDes block and is only available for the PCIe mode of operation.

**Note:** In the simulation mode, the physical interface of the SerDes IP block is inactive.

- **RTL** - This mode enables to fully simulate the SerDesIF block from the fabric interface to the serial I/O interface. This simulation mode is available for PCIe operation and requires verification IP models and test benches. The simulation runtime for this mode is longer than the runtime for the other SerDesIF simulation modes.

Refer to the *SERDESIF BFM Simulation Guide* for more information on simulation detail.

### 3.1.11.8 Adding RTG4 PCIe Block to User Design

Libero promotes the SerDes I/Os to the top level and exposes the AXI3 (based on user settings) and APB interface to the FPGA fabric. In addition, the SerDes block exposes the clocks, resets, PLL locks, and power management signals for PCIe implementation. The appropriate setup is done using the SerDes block configurator in the Libero software tools.

The user logic block implements an AXI3 slave interface to transfer data to the PCIe link and transaction requests come through the AXI3 master interface to receive the data from the PCIe link. A fabric clock conditioning circuit (SPLL) generates the clock for the AXI3 interface on the port CLK\_BASE.

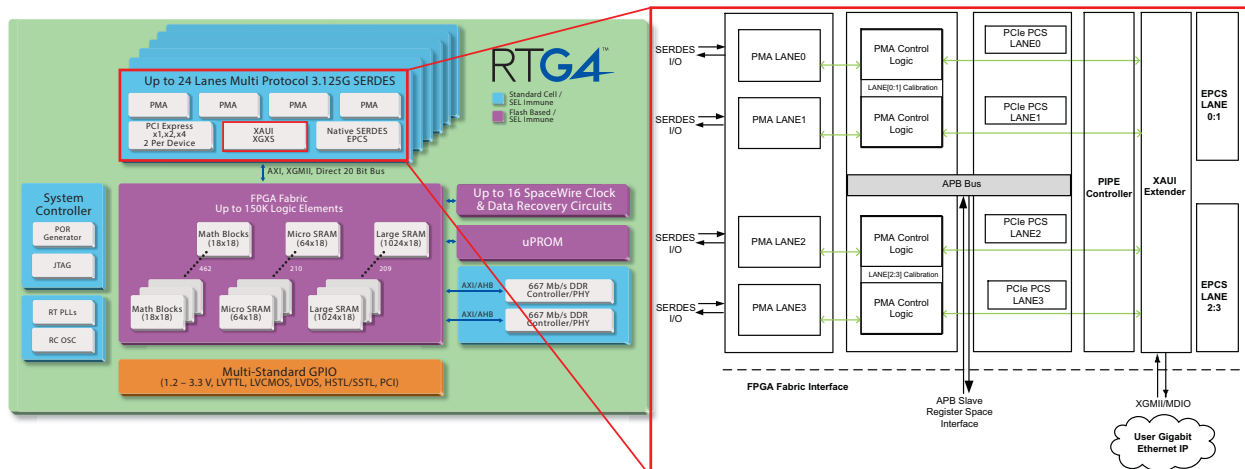
## 3.2 XAUI

### 3.2.1 Introduction

This chapter describes implementing XAUI in the RTG4 FPGA devices using the XAUI extender block inside the high-speed serial Interface SerDes block. XAUI is a standard for extending the 10 Gb media independent interface (XGMII) between the media access control (MAC) and PHY layer of 10 Gb ethernet (10 GbE). The RTG4 high-speed serial block implements the integrated XAUI, which can be connected to a 10 Gb ethernet FPGA IP core in the FPGA fabric for a complete solution.

The RTG4 SerDes block (PCISS and NPSS blocks) integrates the functionality of supporting multiple high-speed serial protocols. The following figure shows the PCIe 2.0, XAUI, and SGMII. All SerDes blocks can be configured in various modes, including XAUI. XAUI is a standard for extending the 10 Gb media independent interface (XGMII) between the MAC and PHY layer of 10 Gb ethernet (10 GbE).

**Figure 42 • RTG4 SerDes Block Diagram**



### 3.2.2 Features

The XAUI implementation in the RTG4 devices offers the following features:

- Full compliance with IEEE 802.3
- IEEE 802.3ae- clause 45 MDIO interface
- IEEE 802.3ae- clause 48 state machines
- Pseudo-random idle insertion (PRBS Polynomial  $X^7 + X^3 + 1$ )
- FPGA interface clock frequency of 156.25 MHz
- Double-width 64-bit single data rate (SDR) interface
- Comma alignment function
- Low power mode
- PHY-XS and DTE-XS loopback
- IEEE 802.3ae- annex 48A jitter test pattern support
- IEEE 802.3 clause 36 8B/10B encoding compliance
- Tolerance of lane skew up to 16 ns (50 UI)
- IEEE 802.3 PICs compliance matrix
- Simple signal mapping to the XGMII
- Independent transmit and receive data paths
- Four lanes conveying the XGMII 64-bit data and control
- Differential signaling with low voltage swing (1600 mV (p-p))
- Self-timed interface allows jitter control to the PCS
- Shared technology with other 10 Gbps interfaces
- Shared functionality with other 10 Gbps ethernet block
- Utilization of 8b/10b encoding

The conversion between the XGMII and XAUI interfaces occurs at the XGXS (XAUI extender sublayer).

### 3.2.3 Device Support

The following table lists the total number of SerDes Blocks in each RTG4 device that can be configured to support XAUI.

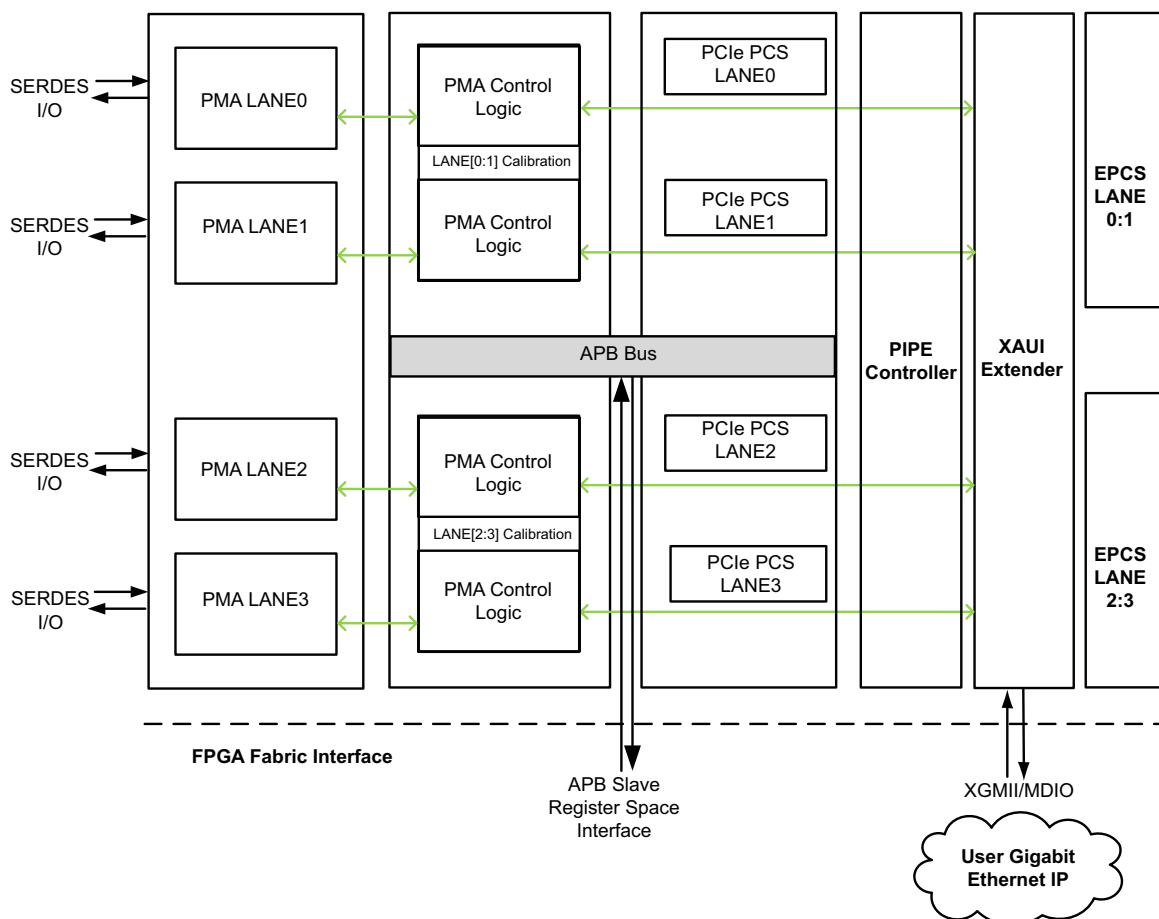
**Table 124 • SerDes Blocks in RTG4 FPGAs Supporting XAUI (PCISS and NPSS)**

	RT4G150
SerDes block available for XAUI	Up to 6

**Note:** XAUI uses the entire SerDes block (4-Lanes).

### 3.2.4 RTG4 XAUI Block

The RTG4 FPGA has an integrated XAUI implementation. The RTG4 high-speed interface (SerDes block) has a XAUI IP block (XAUI extender) and SerDes block. The following figure shows an application example; the XAUI IP extending the 10 Gb soft IP in the fabric. The XAUI IP block in SerDes block provides the XGXS functionality and the SerDes block provides the physical layer. The XAUI IP block connects a 10 Gb ethernet MAC to SerDes physical medium attachment (PMA) logic. Third party MAC IP hosted in the FPGA fabric is responsible for providing IEEE 802.3ae functionality including generating XAUI compliant idle sequence and insertion of  $||K||$ ,  $||R||$ ,  $||A||$  ordered sets at TXD/TXC interface. In addition, the XAUI IP block has a MDIO interface allowing an MDIO manageable device to program the MDIO registers. The SerDes block is configured to PMA only mode and requires a reference clock of 156.25 MHz to operate at a line rate of 3.125 Gbps.

**Figure 43 • XAUI Implementation in RTG4**

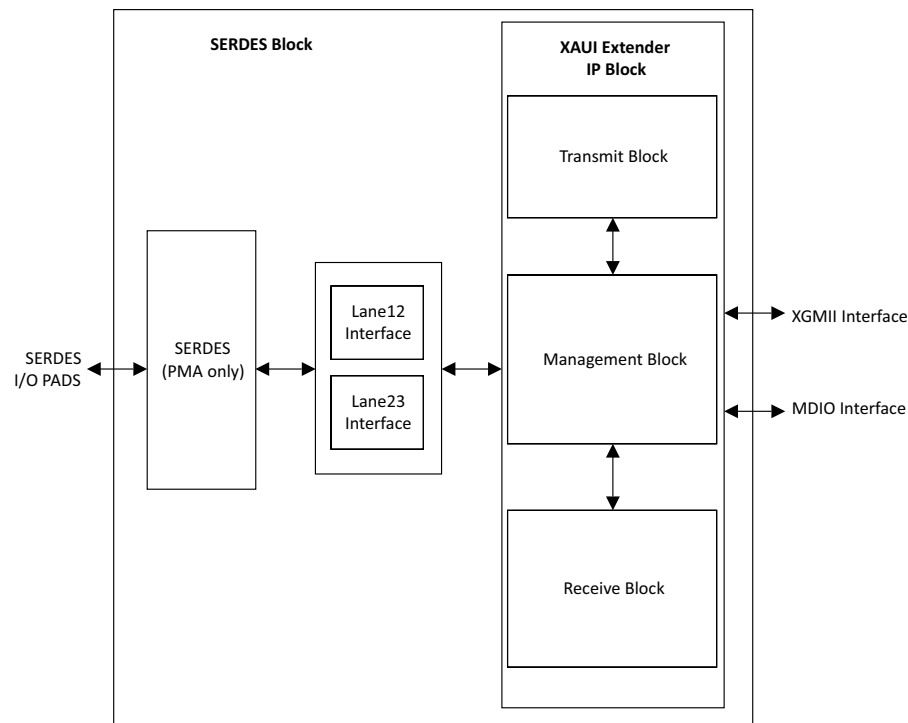
The high-speed serial interface can be configured to support multiple serial protocols. However, when using the XAUI protocol, only one protocol can be implemented as XAUI uses all four SerDes lanes of the Quad. The following table lists the lane speed of four physical SerDes lanes when using XAUI.

**Table 125 • XAUI Implementation in RTG4**

XAUI Protocol	Lane0		Lane1		Lane2		Lane3	
	Protocol	Speed	Protocol	Speed	Protocol	Speed	Protocol	Speed
Single protocol PHY	XAUI	3.125 G	XAUI	3.125 G	XAUI	3.125 G	XAUI	3.125 G

The SerDes block in XAUI mode has SerDes I/O pads on one side, an XGMII interface and MDIO interface on the FPGA fabric side. The preceding table lists the RTG4 SerDes lane assignments in XAUI mode. Refer to [DS0130: RTG4 Pin Description](#) for other SerDes required pins.



**Figure 44 • XAUI IP Block Diagram**

### 3.2.4.1 Transmit Block

This block is responsible for encoding the XGMII data (using 8B/10B). The output to the transmit block is an 80-bit interface (20 bits per lane). The PMA in the SerDes receives this 80-bit data and transmits it to the XAUI bus.

### 3.2.4.2 Receive Block

This block receives 8B/10B encoded data and four recovered clocks from an external XAUI SerDes PMA. The receive block performs comma alignment on the data, phase-aligns the four lanes of data, and performs the 8B/10B decode function.

The transmit and receive FPGA interface frequencies are set at 156.25 MHz.

### 3.2.4.3 Management Block

The management block is the MDIO interface to the design registers.

### 3.2.4.4 XAUI IP Fabric Interface

The RTG4 SerDes block in XAUI mode interfaces with the fabric and differential I/O pads.

**Table 126 • MDIO Interface Signals**

Port	Type	Description
XAUI_MMD_MDC	Input	MDIO I/F clock. 40 MHz or less. MDC is required with or without MDIO functionality.
XAUI_MMD_MDI	Input	MDIO data input from bidirectional pad.
XAUI_MMD_MDI_EXT	Input	Serial data output of another block that responds to a host read transaction.
XAUI_MMD_MDO	Output	MDIO data output to bidirectional pad.
XAUI_MMD_MDOE	Output	MDIO data output enable. This is used to control bidirectional pad. It is active high.
XAUI_MMD_MDOE_IN	Input	MDIO data output enable input. This is used to force MMD_MDI high in an idle state. It is active high.

**Table 126 • MDIO Interface Signals (continued)**

XAUI_MMD_PRTAD[4:0]	Input	A static signal that defines the port address of the XAUI extender block instantiated. Access to the MDIO registers is granted only if the port address specified in the MDI stream matches this input.
XAUI_MMD_DEVID[4:0]	Input	A static signal that defines the device ID of the XAUI extender block instantiated. Access to the MDIO registers is granted only if the device ID (DEVID) specified in the MMD_MDI stream matches this input. For the PHY- XS, this value must be 04h. For the DTE-XS, this value must be 05h.
XAUI_VNDRRESLO[7:0]	Output	A general purpose register for vendor use, reset Low. The output of two 16-bit registers (address 0x8000 and 0x8001) that are set Low on reset for general purpose use.
XAUI_VNDRRESHI[7:0]	Output	General purpose register for vendor use, reset High. The output of two 16-bit registers (address 0x8002 and 0x8003) that are set High on reset for general use.

**Table 127 • XGMII Transmit Interface Signals**

Port	Type	Description
XAUI_TXD[63:0]	Input	Transmit data input from the XGMII. The signal has the following lane definitions: Lane0, row0: txd[7:0] Lane1, row0: txd[15:8] Lane2, row0: txd[23:16] Lane3, row0: txd[31:24] Lane0, row1: txd[39:32] Lane1, row1: txd[47:40] Lane2, row1: txd[55:48] Lane3, row1: txd[63:56] The row0 lanes are leading the row1 lanes in time. Refer to IEEE 802.3ae, clause 46, for a complete definition.
XAUI_TXC[7:0]	Input	Transmit data lane control signals. The signal has the following lane definitions: Lane0, row0: txc[0] Lane1, row0: txc[1] Lane2, row0: txc[2] Lane3, row0: txc[3] Lane0, row1: txc[4] Lane1, row1: txc[5] Lane2, row1: txc[6] Lane3, row1: txc[7] The row0 lanes are leading the row1 lanes in time. Refer to IEEE 802.3ae, clause 46, for a complete definition.

**Table 128 • XGMII Receive Interface Signals**

Port	Type	Description
XAUI_RXD[63:0]	Output	Receive data output to the XGMII. The signal has the following lane definitions: Lane0, row0: rxd[7:0] Lane1, row0: rxd[15:8] Lane2, row0: rxd[23:16] Lane3, row0: rxd[31:24] Lane0, row1: rxd[39:32] Lane1, row1: rxd[47:40] Lane2, row1: rxd[55:48] Lane3, row1: rxd[63:56] The row0 lanes are leading the row1 lanes in time. Refer to IEEE 802.3ae, clause 46, for a complete definition.
XAUI_RXC[7:0]	Output	Receive lane data control signals. The signal has the following lane definitions: Lane0, row0: rxc[0] Lane1, row0: rxc[1] Lane2, row0: rxc[2] Lane3, row0: rxc[3] Lane0, row1: rxc[4] Lane1, row1: rxc[5] Lane2, row1: rxc[6] Lane3, row1: rxc[7] The row0 lanes are leading the row1 lanes in time. Refer to IEEE 802.3ae, clause 46, for a complete definition.

**Table 129 • XAUI IP Block Miscellaneous Control Signal**

Port	Type	Description
XAUI_LOOPBACK_OUT	Output	Loopback mode enable out. This signal is asserted when the XAUI extender block is placed in loopback. Typically, this signal is shunted back into the input XAUI_LOOPBACK_IN port. In this case, loopback is implemented in the XAUI extender block.
XAUI_LOOPBACK_IN	Input	Loopback mode enable in. When asserted, the XAUI PMA output data signals are shunted back into the input signals. For loopback to function appropriately, the XGMII transmit clock TX_CLK must be shunted back into the PMA recovered clock inputs.
XAUI_LOWPOWER	Output	SerDes low power status. When set to 1, the SerDes block signals to the fabric that the SerDes block is in a low power state.
XAUI_PHY_NOT_READY	Output	This signal goes low when all 4- SerDes lanes within the XAUI block have completed the calibration sequence indicating the entire PMA is ready for operation. Completed calibration corresponds to the two conditions completed by the PMA control logic: Receiver detects incoming data and the CDR PLL is locked to the input bit stream in fine grain state for all lanes. This pin will go high if any of the SerDes lanes go down.
XAUI_LANE01_RX_ERR	Input	XAUI Interface Lane[0:1] receiver error is detected when using the external logic. When there are many receive errors such as, invalid 8b/10b code or disparity error, then the asynchronous signal can be used to cause the CDRPLL to switch back to the frequency lock phase. These pins can be hardwired to 0, and rely only on Electrical Idle detection to switch the CDR PLL back to frequency lock state.

**Table 129 • XAUI IP Block Miscellaneous Control Signal (continued)**

Port	Type	Description
XAUI_LANE23_RX_ERR	Input	XAUI Interface Lane[2:3] receiver error is detected when using the external logic. When there are many receive errors such as, invalid 8b/10b code or disparity error, then the asynchronous signal can be used to cause the CDRPLL to switch back to the frequency lock phase. These pins can be hardwired to 0, and rely only on Electrical Idle detection to switch the CDR PLL back to frequency lock state.
XAUI_PWRDN	Input	When this input is driven high it powers down the XAUI Core. Tie low for normal operation.
XAUI_TX_OOB	Input	Unused for XAUI. Tie Low for normal operation.
PLL_LOCKLOST_INT	Output	Output of SPLL Lock lost status register (Active high indicates that the lock is lost). The SPLL manages clock domain data transfers skew between the FABRIC and XAUI block module.
SPLL_LOCK	Output	"SPLL Lock signal. High indicates that the frequency and phase lock are achieved. The SPLL manages clock domain data transfers skew between the FABRIC and XAUI block module.

**Table 130 • APB Slave Interface—APB\_SLAVE**

Clock Signal	Type	Description
APB_S_PCLK	Input	APB interface clock signal input. Controlled by CLK_BASE of the Libero configurator. For XAUI, the CLK_BASE is the same frequency as the 156.25 MHz reference clock. This value is required to perform accurate timing driven place and route and analysis.
APB_S_PENABLE	Input	APB strobe. This signal indicates the second cycle of an APB transfer.
APB_S_PWRITE	Input	APB write or read. If High, a write occurs when an APB transfer takes place. If low, a read takes place.
APB_S_PADDR[13:0]	Input	APB address bus
APB_S_PWDATA[31:0]	Input	APB write data
APB_S_PREADY	Output	APB ready. Used to insert wait states
APB_S_PRDATA[31:0]	Output	APB read data
APB_S_PSLVERR	Output	APB Error

### 3.2.5 Reset and Clocks for XAUI

This section covers the functional aspects of the reset and clock circuitry inside the high-speed serial interface block for XAUI mode. It has the following components:

- XAUI Mode Clocking
- XAUI Mode Reset Network

#### 3.2.5.1 XAUI Mode Clocking

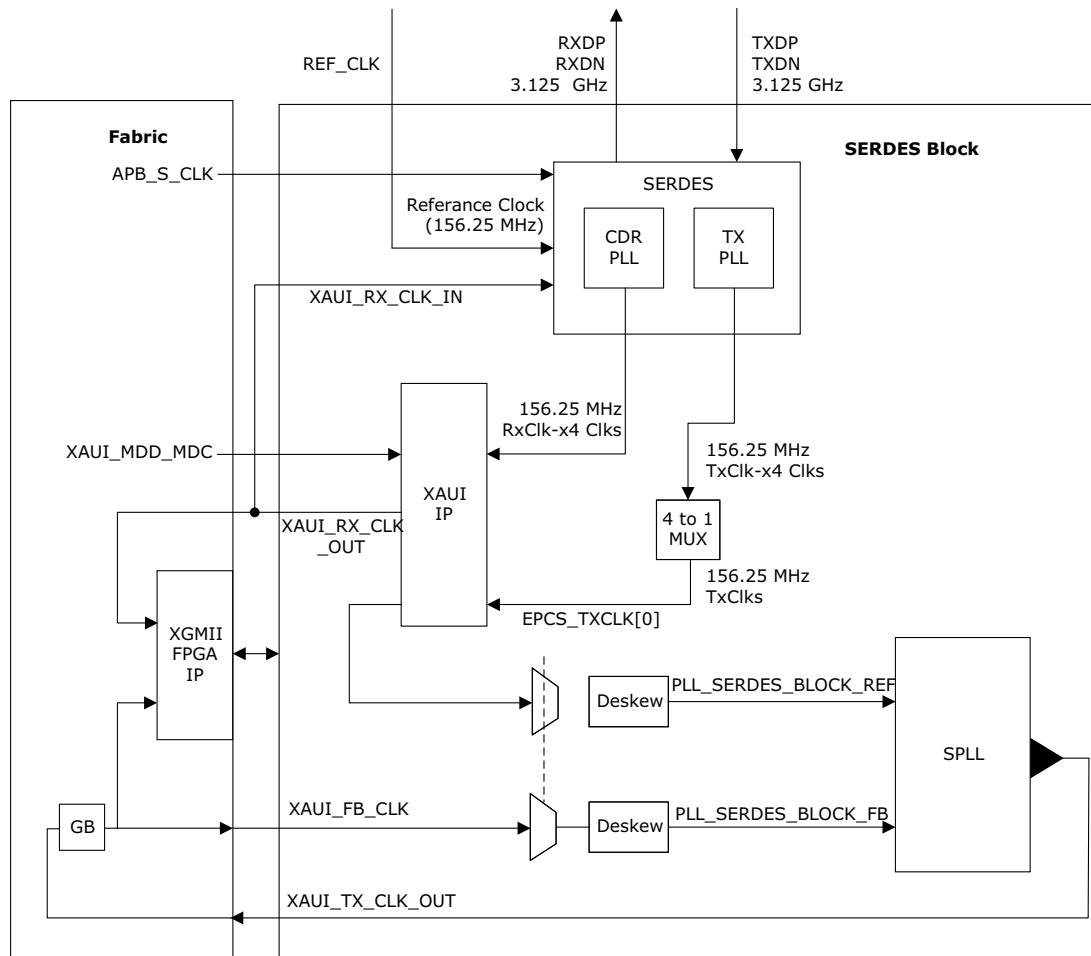
When the SerDes block is configured in XAUI mode, it has multiple clock inputs and outputs. This section describes the XAUI clocking scheme.

In XAUI mode, data is exchanged from FPGA IP in the fabric and XAUI IP. The following figure shows the clocking of the XAUI IP block. The 156.25 MHz reference clock is used by the SerDes PMA (Tx PLL and CDR PLL). The PLLs generate 156.25 MHz clocks and send 4 Rx and 4 Tx clocks through the EPCS interface. The Lane0 Tx clock is fed in to the reference clock of SPLL and XAUI extender block. This SPLL is used to reduce the skew between the fabric and RTG4 SerDes block module. The Libero SOC automatically connects the XAUI\_TX\_CLK\_OUT signal with the XAUI\_FB\_CLK signal in the FPGA fabric through the global network, as shown in the following figure. The 4 Rx clocks are fed in to the XAUI

extender block, where lane de-skewing is done and only one Rx clock is given out to the FPGA fabric. The XAUI\_TX\_CLK\_OUT and XAUI\_RX\_CLK signals are used by XGMII FPGA IP. The APB clock (APB\_S\_PCLK) is an asynchronous clock used for SerDes block configuration register access.

In XAUI only mode, the Tx clock is generated from the PMA. The lane0 Tx clock is used for this purpose. The Rx clock for all four lanes is passed to the XGXS receiver block with gating logic in between to low power operation.

**Figure 45 • Clocking in XAUI Mode**



**Note:** For information about REF\_CLK, see [Figure 46](#), page 91.

The following table lists the various clocks in the XAUI mode.

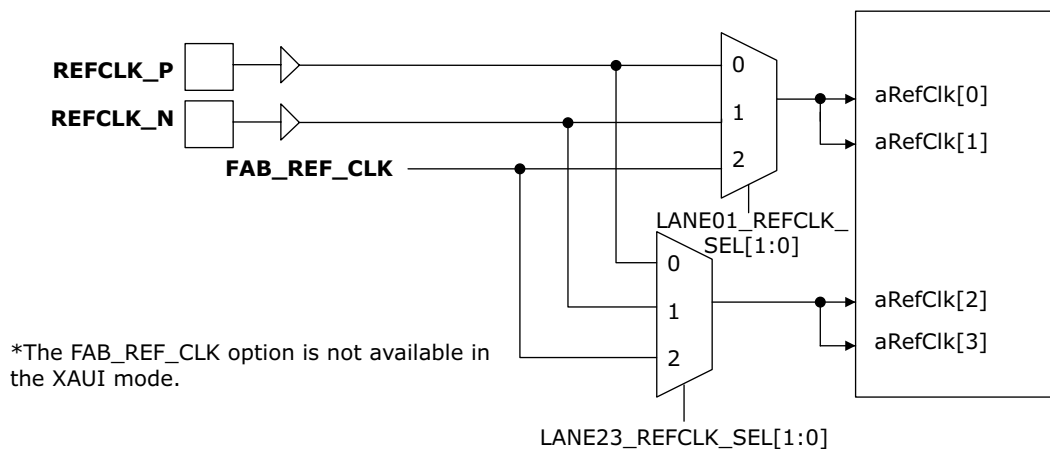
**Table 131 • Clock Signals in XAUI Mode**

Clock Signal	Type	Description
XAUI_RX_CLK_OUT	Output	Receive clock synchronous with Rxd, clock synchronous with the output XGMII data Rxd. This clock operates nominally at 156.25 MHz. Refer to IEEE 802.3ae, clause 46, for a complete definition.
XAUI_TX_CLK_OUT	Output	Transmit clock to be used for transmit data. This clock operates nominally at 156.25 MHz. Refer to IEEE 802.3ae, clause 46, for a complete definition.
XAUI_RX_CLK_IN	Input	Receive Interface FIFO read clock within XAUI Core. This is the read clock of the receive Fly-Wheel-FIFO used to decouple the phase of the FPGA clock from the SerDes receiver clock. The Fly-Wheel-FIFO makes FPGA timing closure easier on the RX_DATA[n:0] by using the same clock to launch and capture data into the fabric. This input is typically connected to the XAUI_RX_CLK_OUT.
XAUI_FB_CLK	Input	Transmit Interface FIFO write clock within XAUI Core. Write clock to the transmit Fly-Wheel-FIFO used to decouple the phase of the FPGA clock from the SerDes transmit clock. Use of the Fly-Wheel-FIFO makes the FPGA timing closure easier on the TX_DATA[n:0] by removing the round trip time of the clock from the SerDes to the fabric clock resource and back to the SerDes to capture the data. This input is typically connected to the XAUI_TX_CLK_OUT.

### 3.2.5.1.1 SerDes Reference Clocks Selection

The PMA in the SerDes block needs a reference clock on each of its lanes for Tx and Rx clock generation through PLLs. The following figure shows reference clock selection in the high-speed serial interface generator available in the Libero SoC. In XAUI mode, the protocol expects a differential clock to be input to the REFCLK\_P and REFCLK\_N pads and the software correctly sets the input MUX for the refclk inputs to the SerDes block. The FAB\_REF\_CLK option is not available in XAUI mode. The reference clock pads are differential input. In XAUI mode, one reference clock is chosen for all 4 lanes.

**Figure 46 • SerDes Reference Clock for XAUI Mode**



**Table 132 • Reference Clock Signals for SerDes**

Clock Signal	Description
REFCLK_P, REFCLK_N	Differential
REFCLK0	Voltage Referenced—input signal requires a reference voltage (SERDES_VREF) rather than to ground such as HSTL or SSTL input standard types.
REFCLK1	Voltage Referenced—input signal requires a reference voltage (SERDES_VREF) rather than to ground such as HSTL or SSTL input standard types.
REFCLK0	Single-ended—clocks that references the input signal to ground such as LVTTTL and LVCMOS.
REFCLK1	Single-ended—clocks that references the input signal to ground such as LVTTTL and LVCMOS.

### 3.2.5.1.2 XAUI FPGA Clocks

The XAUI\_TX\_CLK\_OUT and XAUI\_RX\_CLK\_OUT from the XAUI block requires global fabric clock resources. These clock resources require use of the eight half-row global routes on the top row of the FPGA routing array. Routing considerations to other clocks such as APB\_S\_CLK also require global clock resources. Attention to design practice may be required to reduce clock congestion with other high-fanout clock nets.

### 3.2.5.1.3 XAUI Mode Reset

The XAUI core reset is a function of device power-on and fundamental XAUI core (CORE\_RESET\_N). The XAUI mode reset is internally synchronized to the release of the CORE\_RESET\_N. However, the PHY\_RESET\_N must also be released for proper operation of the XAUI high-speed serial interface.

The following table lists the reset signals and recommended connections.

**Table 133 • XAUI Mode Reset Signals**

Port	Type	Description
CORE_RESET_N	Input	Active-low reset for XAUI Core.
PHY_RESET_N	Input	SERDES-PHY-Active low reset.
APB_S_PRESET_N	Input	Asynchronous active-low reset signal for APB slave interface.

## 3.2.6 Design Considerations

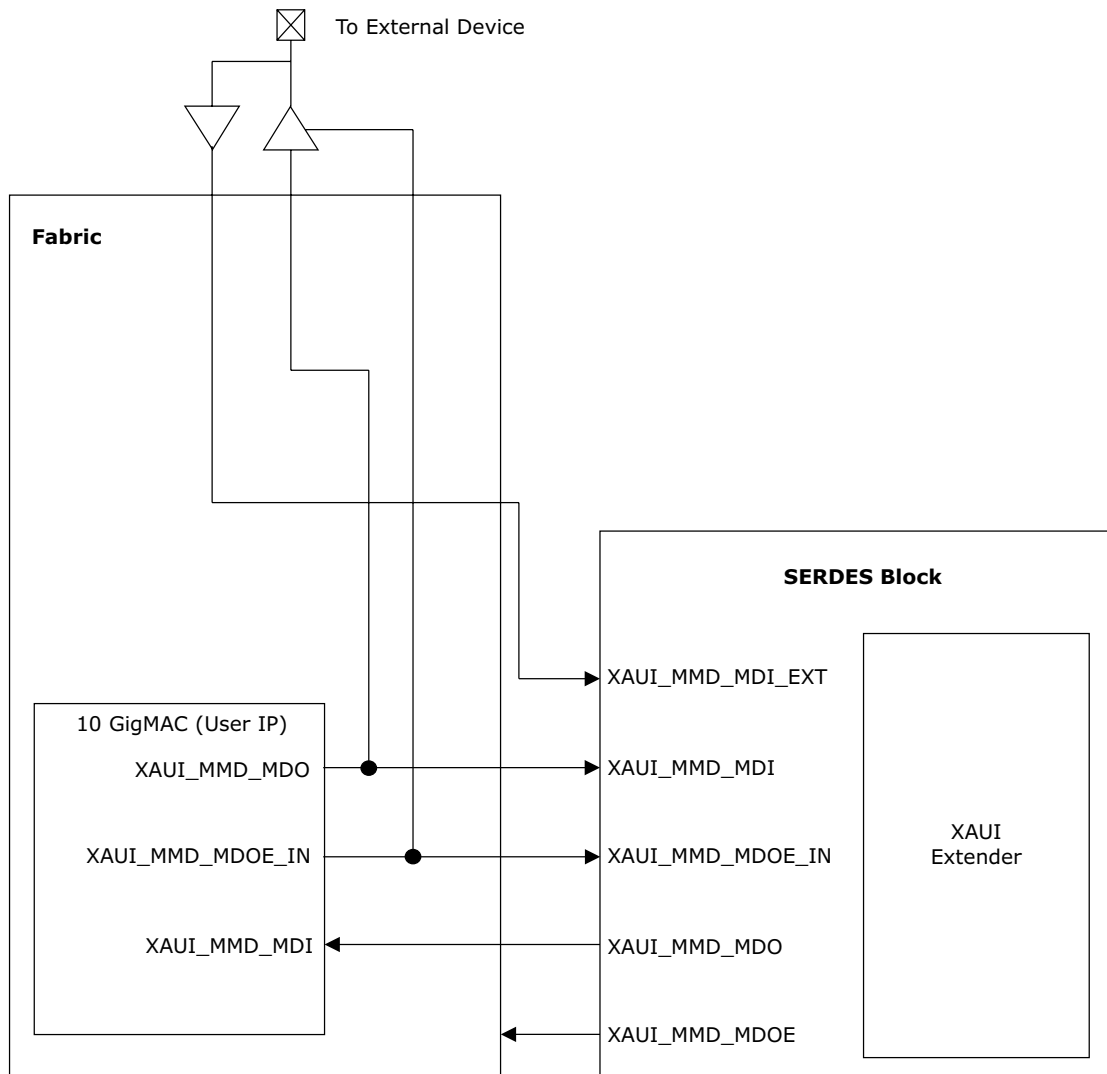
This section provides instructions for implementing XAUI in the RTG4 devices. It includes the following sections:

- Using MDIO Interface
- XAUI IP Block Timing Diagram
- XAUI Mode Loopback Test Operation
- Using MMD Status Registers

### 3.2.6.1 Using MDIO Interface

The MDIO interface allows users to access the MDIO registers. The following figure shows a system block diagram for connecting XAUI IP and an MDIO manageable device (MMD) to a station management entity (STA). In this case, the STA is A-XGMAC. If the MDIO is not required for the user application, the MDIO ports can be tied off inactive.

**Figure 47 • MDIO System Block Diagram**



### 3.2.6.2 XAUI IP Block Timing Diagram

The following sections show the timing relations between clock and data for the three interfaces of the XAUI extender.

- Transmit Interface
- Receive Interface
- MMD Read Timing
- MMD Write Timing

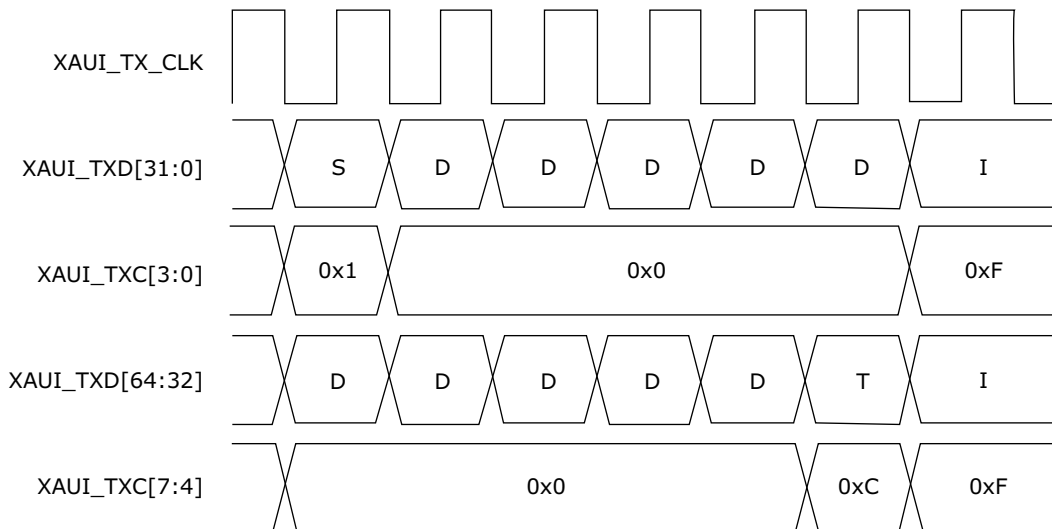
Refer to the *DS0131: RTG4 FPGA Datasheet* for the detailed timing numbers.



### 3.2.6.2.1 Transmit Interface

The following figure shows the XGMII transmit timing diagram. The transmit data and control signals are source centered on the transmit clock as per the requirements of IEEE 802.3ae, clause 46. All four lanes of data are synchronous with a common clock.

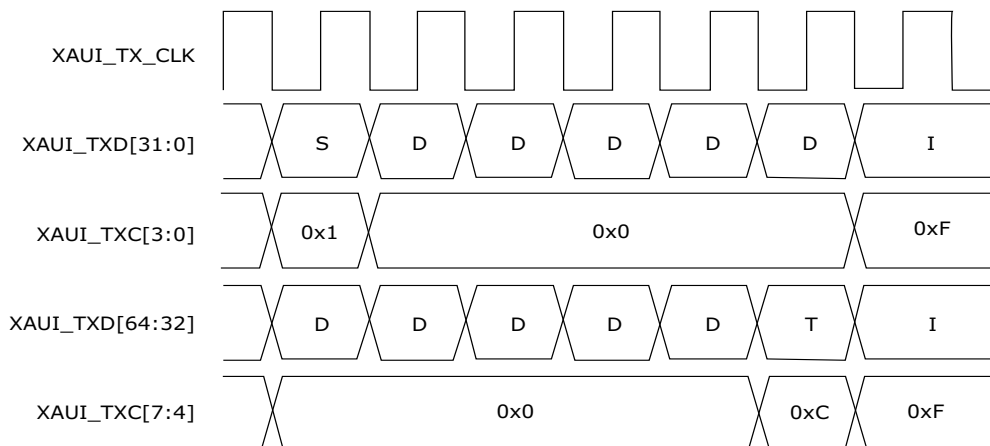
**Figure 48 • Transmit XGMII Interface Timing Diagram**



### 3.2.6.2.2 Receive Interface

The following figure shows the XGMII receive timing diagram. The receive data and control signals are edge-aligned with the receive clock XAUI\_RX\_CLK. To be fully compliant with IEEE 802.3ae, the data and control signals are normally source centered on XAUI\_RX\_CLK. However, in the RTG4 FPGA, the XAUI extender is interfaced with a FPGA 10G MAC in the fabric within the same device, eliminating the need to source center the data. All four lanes of data are synchronous with the common clock XAUI\_RX\_CLK. The user must check timing and ensure that the data is captured by the FPGA 10G MAC in the fabric.

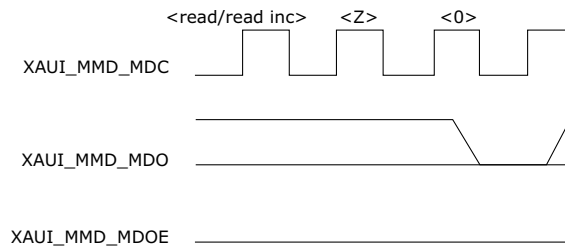
**Figure 49 • XGMII Interface Receive Timing Diagram**



### 3.2.6.2.3 MMD Read Timing

The following figure shows the timing diagram for an MDIO register read. The XAUI\_MMD\_MDO signal is controlled by XAUI\_MMD\_MDOE.

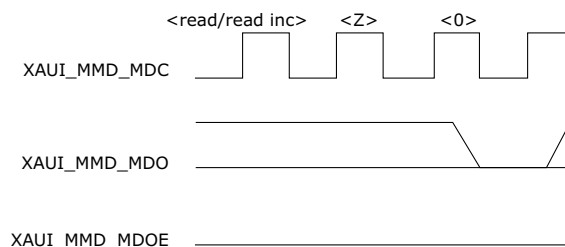
**Figure 50 • MDIO Interface Read Timing Diagram**



### 3.2.6.2.4 MMD Write Timing

The following figure shows the timing diagram for MDIO registers. XAUI\_MMD\_MDOE must not be asserted during a write operation. Refer to the IEEE 802.3ae specification, clause 45, for a complete definition.

**Figure 51 • MDIO Interface Read Timing Diagram**



### 3.2.6.3 XAUI Mode Loopback Test Operation

The XAUI extender block can be placed in loopback mode for testing purposes. It can also be placed in multiple loopback operations.

#### 3.2.6.3.1 XAUI – Near End Loopback Test

Bit 14 of Reg00 can be used to enable the loopback. When loopback mode is enabled, the transmit output is shunted back into the receive input. For loopback mode to work appropriately, the transmit clock is also shunted back into the receive clock inputs. The loopback test data must be fed from the XGMII interface available to fabric.

#### 3.2.6.4 XAUI – Far End Loopback Test

In the XAUI far end loopback test, the transmit interface of the XAUI extender block is connected to the SerDes block interface. In this case, the SerDes block is put in loopback mode, where serial data from transmit side is fed to the serial receive interface. XAUI far-end loopback verifies the transmit, receives the paths of the XAUI extender block, and then tests the validity of the PMA data path. The XAUI\_LOOPBACK\_IN signal is used for this mode.

### 3.2.6.5 SerDes Block System Register Configurations for XAUI Mode

The RTG4 SerDes block subsystem has three regions of configuration and the following two status registers:

- SerDes Block System Register
- Bridge Register Space

These registers are accessed by the 32-bit APB bus. Refer to the [SerDes Block System Registers](#), page 136 for details. In XAUI mode, the PCIe core registers are not used, only subsets of the SerDes

block system registers and SerDes block register are used. These registers can be updated through the 32-bit APB interface after power-up.

### 3.2.6.6 Using MMD Status Registers

The two MMD status registers are XS status 1 (Reg01) and XS status 2 (Reg07). When the reset signal on the XAUI block is de-asserted, the initial state of these status registers indicates a fault condition. The initial fault condition must be ignored, and a read operation has to be performed on these registers to clear the initial false fault-status. When a real fault condition happens (for example, when a link is down), the fault register properly indicates any fault, as expected.

#### 3.2.6.6.1 MDIO Register Map

The following table lists the MDIO registers.

**Table 134 • MDIO Registers**

Register Name	Register Address	Read/ Write	Device Address	Description
Reg00	0x0000	R/W	04h/05h	XS control 1 register
Reg01	0x0001	R/O	04h/05h	XS status 1 register
Reg02	0x 0002	R/O	04h/05h	XS device identifier register low
Reg03	0x0003	R/O	04h/05h	XS device identifier register high
Reg04	0x0004	R/O	04h/05h	XS speed ability register
Reg05	0x0005	R/O	04h/05h	XS devices in package register low2
Reg06	0x0006	R/O	04h/05h	XS devices in package register high
–	0x0007	N/A	04h/05h	Reserved
Reg07	0x0008	R/O	04h/05h	XS status 2
–	0x0009 to 0x000d	N/A	04h/05h	Reserved
Reg08	0x000e	R/O	04h/05h	XS package identifier register low
Reg09	0x000f	R/O	04h/05h	XS package identifier register high
–	0x0010 to 0x0017	N/A	04h/05h	Reserved
Reg10	0x0018	R/O	04h/05h	10G XGXS lane status register
Reg 11	0x0019	R/W	04h/05h	10G XGXS test control register
–	0x001a to 0x7fff	N/A	04h/05h	Reserved
Reg12	0x8000	R/W	04h/05h	Vendor-specific reset Lo 1
Reg13	0x8001	R/W	04h/05h	Vendor-specific reset Lo 2
Reg14	0x8002	R/W	04h/05h	Vendor-specific reset Hi 1
Reg15	0x8002	R/W	04h/05h	Vendor-specific reset Hi 1
–	0x8004 to 0xffff	N/A	04h/05h	Reserved

**Table 135 • Reg00 XS Control 1 Register**

Bit Number	Name	Reset Value	Description
15	Reset	0x0	The XAUI extender block is reset when this bit is set to 1. It returns to 0 when the reset is complete (self-clearing). 1: Block reset 0: Normal operation
14	Loopback	0x0	The XAUI extender block loops the transmit signal back into the receiver. 0: Disable loopback 1: Enable loopback
13	Speed selection	0x1	This bit is for speed selection and is set to 1'b1 for compatibility with clause 22. 0: Unspecified 1: 10 Gbps and above Any write to this bit is ignored.
12	Reserved	0x0	Reserved
11	Low power mode	0x0	When set to 1, the SerDes block is placed in a low power mode. Set to 0 to return to normal operation. 0: Normal operation 1: Low power mode
[10:7]	Reserved	0x0	Reserved
6	Speed selection	0x1	This bit is set to 1'b1 for compatibility with clause 22. 0: Unspecified 1: 10 Gbps and above
[5:2]	Speed selection	0x0	The speed of the PMA/PMD may be selected using bits 5 through 2. 1 x x x: Reserved x 1 x x: Reserved x x 1 x: Reserved 0 0 0 1: Reserved 0 0 0 0: 10 Gbps Any write to this bit is ignored.
[1:0]	Reserved	0x0	Reserved

**Table 136 • Reg01 XS Status 1 Register**

Bit Number	Name	Reset Value	Description
2	PHY/DTE transmit/receive link status	0x0	When read as a one, the receive link is up. 0: Link down 1: Link up The receive link status bit is implemented with latching low behavior.
1	Low power ability	0x1	When read as a one, it indicates that the low power feature is supported. 0: Low power not supported 1: Low power is supported
0	Reserved	0x0	Reserved

**Table 137 • Reg02 XS Device Identifier Low Register**

Bit Number	Name	Reset Value	Description
[15:0]	Organizationally unique identifier (OUI)	0x0	Reg02 and Reg03 provide a 32-bit value, which may constitute a unique identifier for a particular type of SerDes. The identifier is composed of the 3rd through 24th bits of the OUI assigned to the device manufacturer by the IEEE, a 6-bit model number, and a 4-bit revision number. Reg02 sets bits [3:18] of the OUI. Bit 3 of the OUI is located in bit 15 of the unique identifier of the register, and bit 18 of the OUI is located in bit 0 of the register.

**Table 138 • Reg03 the XS Device Identifier High Register**

Bit Number	Name	Reset Value	Description
[15:10]	OUI	0x0	Bits [19:24] of the OUI. Bit 19 of the OUI is located in bit 15 of the register, and bit 24 of the OUI is located in bit 10 of the register.
[9:4]	Manufacturer model number	0x0	Bits [5:0] of the manufacturer model number. Bit 5 of the model number is located in bit 9 of the register, and bit 0 of the model number is located in bit 4 of the register.
[3:0]	Revision number	0x0	Bits [3:0] of the manufacturer model number. Bit 3 of the revision number is located in bit 3 of the register, and bit 0 of the revision number is located in bit 0 of the register.

**Table 139 • Reg04 XS Speed Ability Register**

Bit Number	Name	Reset Value	Description
[15:1]	Reserved	0x0	Reserved
0	10g capable	0x1	0: Not capable of 10g 1: 10g capable

**Table 140 • Reg05 XS Devices in Package Low Register**

Bit Number	Name	Reset Value	Description
[15:6]	Reserved	0x0	Reserved
5	DTE XS present	0x1	0: DTE XS not present in the package 1: DTE XS present in the package
4	PHY XS present	0x0	0: PHY XS not present in the package 1: PHY XS present in the package
3	PCS present	0x0	0: PCS not present in the package 1: PCS present in the package
2	WIS present	0x0	0: WIS not present in the package 1: WIS present in the package
1	PMD/PMA present	0x0	0: PMD/PMA not present in the package 1: PMD/PMA present in the package
0	Clause 22 register present	0x0	0: Clause 22 registers are not present in the package 1: Clause 22 registers are present in the package

**Table 141 • Reg06 XS Devices in Package High Register**

Bit Number	Name	Reset Value	Description
15	Vendor-specific device2 present	0x0	0: Vendor-specific device 2 not present 1: Vendor-specific device 2 present
14	Vendor-specific device1 present	0x0	0: Vendor-specific device 1 not present 1: Vendor-specific device 1 present
[13:0]	Reserved	0x0	Reserved

**Table 142 • Reg07 XS Status 2 Register**

Bit Number	Name	Reset Value	Description
[15:14]	Device present	0x0	10: Device responding at this address. 11: No device responding at this address. 01: No device responding at this address. 00: No device responding at this address.
[13:12]	Reserved	0x0	Reserved
11	Transmit fault	0x0	0: No transmit fault 1: Transmit fault Latched high, clear on read
10	Receive fault	0x0	0: No receive fault 1: Receive fault Latched high, clear on read
[9:0]	Reserved	0x0	Reserved

**Note:** Transmit and Receive fault status bit roles are swapped when the MDIO device ID is set to 5'h04 (PHY equipment). The descriptions in preceding table assume that the MDIO device ID is set for DTE (data terminal equipment).

**Table 143 • Reg08 XS Package ID Low Register**

Bit Number	Name	Reset Value	Description
[15:0]	OUI	0x0	Reg 08 and Reg 09 provide a 32-bit value, which may constitute a unique identifier for a particular type of package that the SerDes is instantiated within. The identifier is composed of the 3rd through 24th bits of the OUI assigned to the package manufacturer by the IEEE, plus a 6-bit model number, and a 4-bit revision number. Reg08 sets bits [3:18] of the OUI. Bit 3 of the OUI is located in bit 15 unique identifier of the register, and bit 18 of the OUI is located in bit 0 of the register.

**Table 144 • Reg09 XS Package ID High Register**

Bit Number	Name	Reset Value	Description
[15:10]	OUI	0x0	Bits [19:24] of the OUI. Bit 19 of the OUI is located in bit 15 of the register, and bit 24 of the OUI is located in bit 10 of the register.
[9:4]	Manufacturer model number	0x0	Bits [5:0] of the manufacturer model number. Bit 5 of the model number is located in bit 9 of the register, and bit 0 of the model number is located in bit 4 of the register.
[3:0]	Revision number	0x0	Bits [3:0] of the manufacturer model number. Bit 3 of the revision number is located in bit 3 of the register, and bit 0 of the revision number is located in bit 0 of the register.

**Table 145 • Reg10 XGXS Lane Status register**

Bit Number	Name	Reset Value	Description
[15:13]	Reserved		Reserved
12	PHY/DTE XGXS lane alignment status	0x0	0: Lanes not aligned 1: Lanes aligned
11	Pattern testing ability	0x1	0: (PHY/DTE)XS is unable to generate test patterns 1: (PHY/DTE)XS is able to generate test patterns
10	PHY XGXS loopback ability	0x1	0: PHY XGXS does not have the ability to perform a loopback 1: PHY XGXS has the ability to perform a loopback
[9:4]	Reserved		Reserved
3	Lane3 synchronized	0x0	When read as a one, this register indicates that the receive Lane3 is synchronized. 0: Lane3 is not synchronized 1: Lane3 is synchronized
2	Lane2 synchronized	0x0	When read as a one, this register indicates that the receive Lane2 is synchronized. 0: Lane2 is not synchronized 1: Lane2 is synchronized
1	Lane1 synchronized	0x0	When read as a one, this register indicates that the receive Lane1 is synchronized. 0: Lane1 is not synchronized 1: Lane1 is synchronized
0	Lane0 synchronized	0x0	When read as a one, this register indicates that the receive Lane1 is synchronized. 0: Lane0 is not synchronized 1: Lane0 is synchronized

**Table 146 • Reg11 the XGXS Test Control Register**

Bit Number	Name	Reset Value	Description
[15:3]	Reserved		Reserved
2	Transmit test pattern enabled	0x0	When this bit is set to a one, pattern testing is enabled on the transmit path. 0: Transmit/receive test pattern disabled 1: Transmit/receive test pattern enabled
[1:0]	Test pattern select	0x0	The test pattern is used when enabled pattern testing is selected using these bits: 00: High frequency test pattern 01: Low frequency test pattern 10: Mixed frequency test pattern 11: Reserved

**Table 147 • Reg12 Vendor-Specific Reset Low 1 Register**

Bit Number	Name	Reset Value	Description
[15:0]	Vendor-specific reset Lo 1	0x0000	General purpose registers that are connected to the output port. XAUI_VNDRRESLO[15:0]. Typically used for external device control.

**Table 148 • Reg13 Vendor-Specific Reset Low 2 Register**

Bit Number	Name	Reset Value	Description
[15:0]	Vendor-specific reset Lo 2	0x0000	General purpose registers that are connected to the output port. XAUI_VNDRRESLO[31:16]. Typically used for external device control.

**Table 149 • Reg14 Vendor-Specific Reset High 1 Register**

Bit Number	Name	Reset Value	Description
[15:0]	Vendor-specific reset Hi 1	0xFFFF	General purpose registers that are connected to the output port. XAUI_VNDRRESLI[15:0]. Typically used for external device

**Table 150 • Reg15 Vendor-Specific Reset High 2 Register**

Bit Number	Name	Reset Value	Description
[15:0]	Vendor-specific reset Hi 2	0xFFFF	General purpose registers that are connected to the output port. XAUI_VNDRRESLI[31:16]. Typically used for external device control.

### 3.2.7 Generating XAUI Block Using Libero SoC

This section provides an overview of how to configure a SerDes block in XAUI mode, and instructions for using XAUI IP in an RTG4 device. RTG4 supports XAUI in any available SerDes Block.

The following sections describe how to instantiate XAUI in a design:

- Using High-Speed Serial Configurator for XAUI Mode
- Simulating the SerDes Block with XAUI Mode

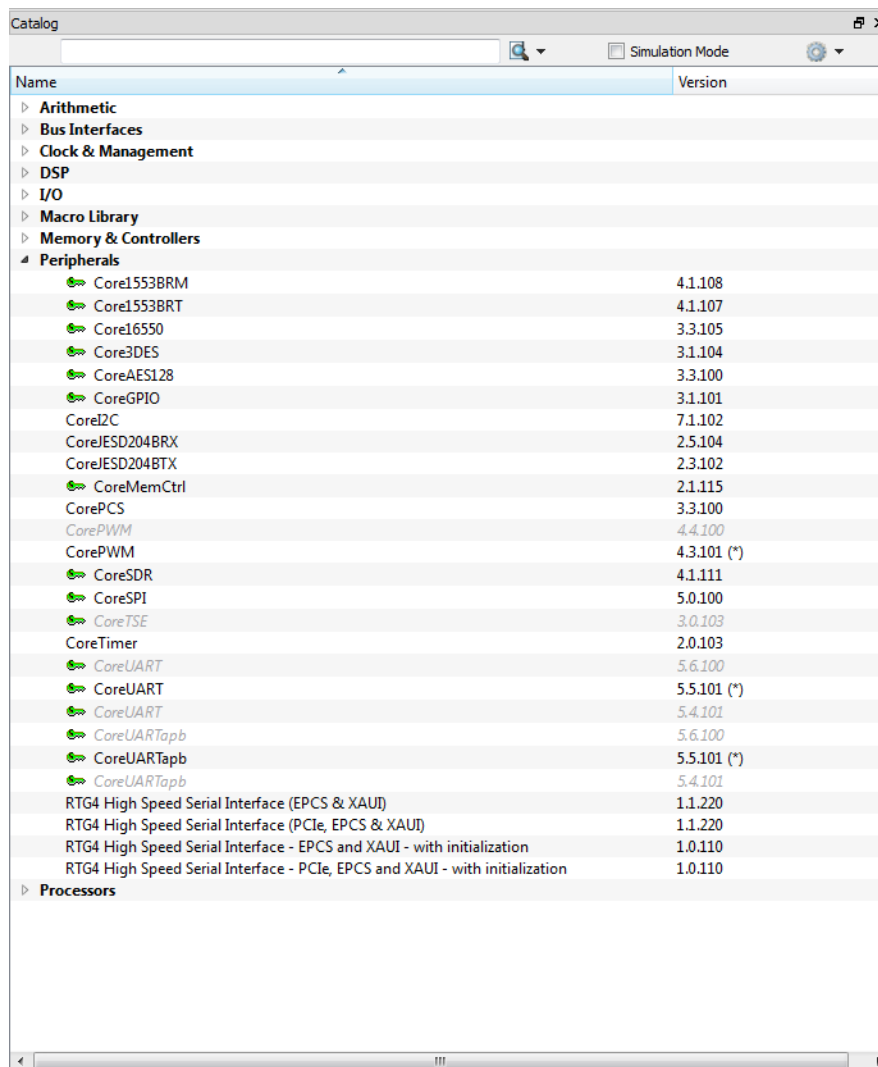


### 3.2.8 Using High-Speed Serial Configurator for XAUI Mode

The high-speed serial interface configurator in the Libero SoC can be used to configure the SerDes block in XAUI mode.

The high-speed serial interface configurator (SerDes block configurator) in Libero allows configuration of the SerDes block in XAUI mode. Following is a brief description of the various protocol configuration options. High-speed serial interface blocks depending on its size. Refer to the [DS0131: RTG4 FPGA Datasheet](#). The Libero SoC software catalog includes four SerDes cores in the Peripherals subsection as shown in the following figure.

**Figure 52 • Libero SoC Catalog**



Refer to the following software configuration documents for specific details of using XAUI implementations.

- RTG4 High Speed Serial Interface (PCIe, EPCS and XAUI) Configuration User Guide (UG0591)
- RTG4 High Speed Serial Interface (PCIe, EPCS and XAUI) with Initialization User Guide(UG0620)
- RTG4 High Speed Serial Interface (EPCS and XAUI) Configuration User Guide (UG0592)
- RTG4 High Speed Serial Interface (EPCS and XAUI) with Initialization User Guide(UG0619)

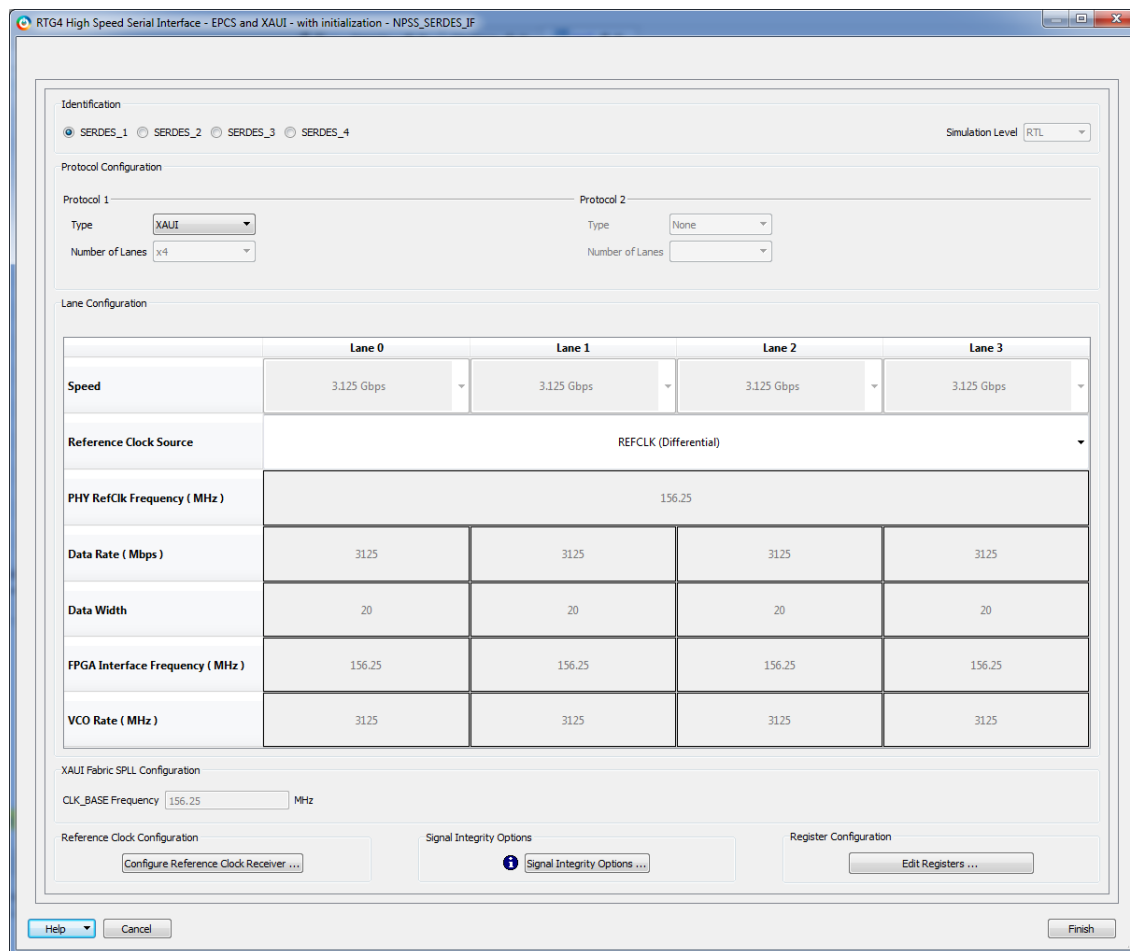
### 3.2.8.1 Protocol Selection

The following settings are used for protocol selection.

- Protocol 1: Select XAUI from the drop-down menu.
- Number of Lanes: Fixed number of lanes used is x4 (configures all four lanes) for XAUI.

**Note:** All four lanes are selected by default in the XAUI protocol type.

**Figure 53 • XAUI Configurator**



RTG4 High Speed Serial Interface - EPCS and XAUI - with initialization - NPSS\_SERDES\_IF

Identification: ☒ SERDES\_1 ☐ SERDES\_2 ☐ SERDES\_3 ☐ SERDES\_4 Simulation Level: RTL

Protocol Configuration

Protocol 1: Type: XAUI Number of Lanes: x4

Protocol 2: Type: None Number of Lanes:

Lane Configuration

	Lane 0	Lane 1	Lane 2	Lane 3
Speed	3.125 Gbps	3.125 Gbps	3.125 Gbps	3.125 Gbps
Reference Clock Source	REFCLK (Differential)			
PHY RefClk Frequency ( MHz )	156.25			
Data Rate ( Mbps )	3125	3125	3125	3125
Data Width	20	20	20	20
FPGA Interface Frequency ( MHz )	156.25	156.25	156.25	156.25
VCO Rate ( MHz )	3125	3125	3125	3125

XAUI Fabric SPLL Configuration

CLK\_BASE Frequency: 156.25 MHz

Reference Clock Configuration: [Configure Reference Clock Receiver ...](#)

Signal Integrity Options: [Signal Integrity Options ...](#)

Register Configuration: [Edit Registers ...](#)

Help Cancel Finish

The following table lists the protocol combinations that are feasible within a single High-speed serial interface block.

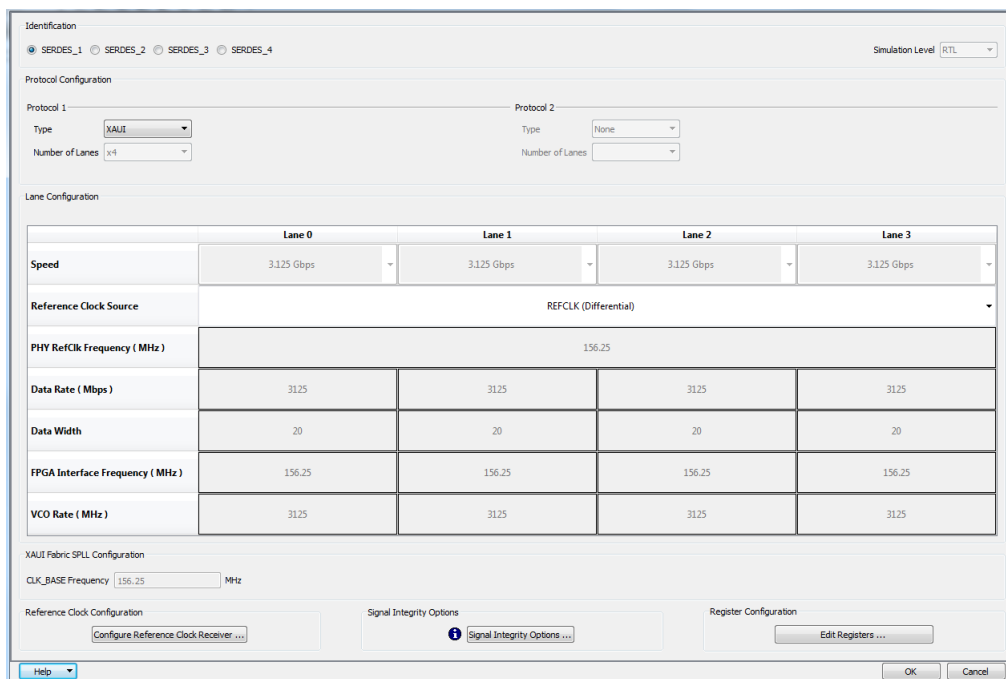
**Table 151 • XAUI Protocol**

Protocol	Lane Width	Lane Assignment	Description	Speed
Protocol 1	X4	Lane 0, Lane 1, Lane 2, and Lane 3		3.125 Gbps

### 3.2.8.2 Lane Configuration

The following figure shows the lane configuration of the XAUI.

**Figure 54 • RTG4 High Speed Serial Interface Configurator—Lane Configuration**



Identification: SERDES\_1 (selected), SERDES\_2, SERDES\_3, SERDES\_4. Simulation Level: RTL.

Protocol Configuration:

Protocol 1: Type: XAUI, Number of Lanes: x1.

Protocol 2: Type: None, Number of Lanes: .

Lane Configuration:

	Lane 0	Lane 1	Lane 2	Lane 3
Speed	3.125 Gbps	3.125 Gbps	3.125 Gbps	3.125 Gbps
Reference Clock Source	REFCLK (Differential)			
PHY RefClk Frequency ( MHz )	156.25			
Data Rate ( Mbps )	31.25	31.25	31.25	31.25
Data Width	20	20	20	20
FPGA Interface Frequency ( MHz )	156.25	156.25	156.25	156.25
VCO Rate ( MHz )	31.25	31.25	31.25	31.25

XAUI Fabric PLL Configuration: CLK\_BASE Frequency: 156.25 MHz.

Reference Clock Configuration: Configure Reference Clock Receiver...

Signal Integrity Options: Signal Integrity Options...

Register Configuration: Edit Registers...

Buttons: Help, OK, Cancel.

- **Speed** - 3.125 Gbps
- **Reference Clock Source** - Clock sources can be differential or single-ended. Select one of the following options for Protocol 1:
  - REFCLK (Differential)
  - REFCLK0 (Voltage Referenced)
  - REFCLK1 (Voltage Referenced)
  - REFCLK0 (Single Ended)
  - REFCLK1 (Single Ended)

**Note:** REFCLK0 and REFCLK1 cannot be used simultaneously. User must select either REFCLK0 or REFCLK1 as reference clock source for XAUI protocol.

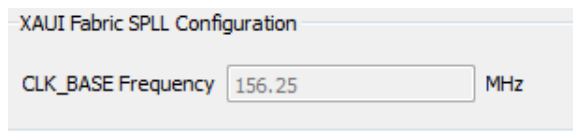
- **PHY RefClk Frequency (MHz)** - The frequency is fixed at 156.25 MHz.
- **Data Rate (Mbps)** - Read-only fixed value. Data rates are computed based on the PHY RefClk frequency.
- **Data Width** - Read-only fixed value.
- **FPGA Interface Frequency (MHz)** - Read-only fixed value.
- **VCO Rate (MHz)** - Read-only fixed value.

### 3.2.9 XAUI SPLL Configuration

These settings are used for SPLL that synchronizes data between the fabric interface to the SerDes block. CLK\_BASE is an SPLL clock setting.

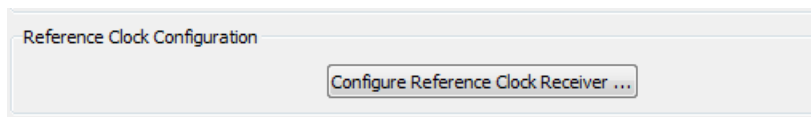
The SPLL is a PLL embedded in the SerDes block to manage the clock phase used for transfers across the SerDes block to FPGA fabric interface. The CLK\_BASE Frequency is read-only and fixed at 156.25 MHz and is related to the APB\_S\_PCLK.

**Figure 55 • SPLL Setting for XAUI**

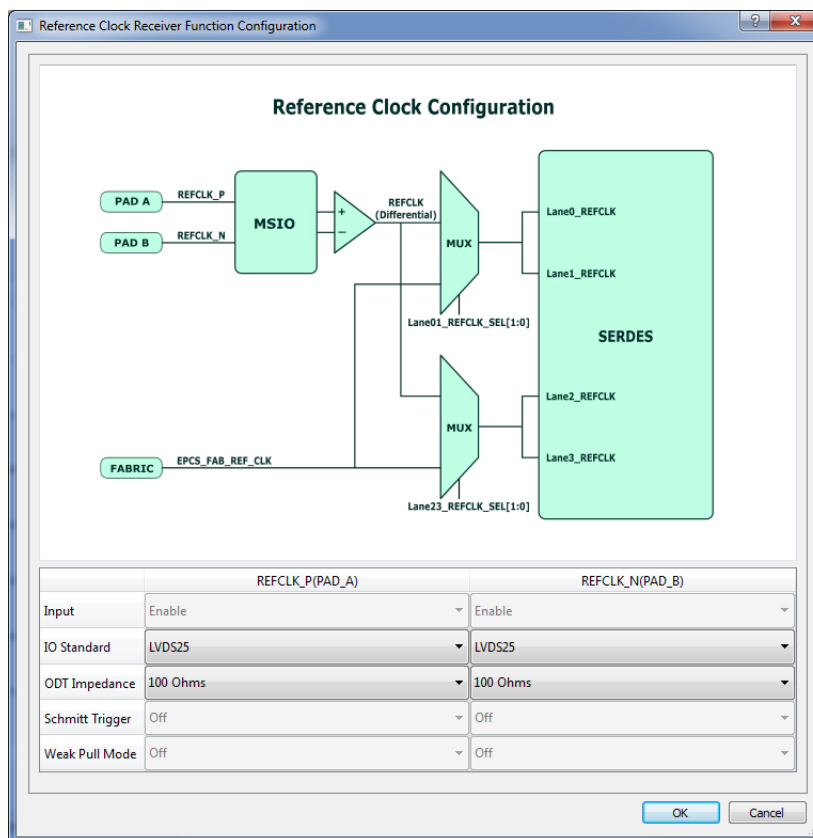


- CLK\_BASE frequency must be set to the same frequency of the APB interface as the operating frequency.
- Protocol 1 PHY Reference Clock: Select the inputs for the PHY reference clock selection. Only REFCLK (Differential) is specified for XAUI applications.

**Figure 56 • Reference Clock Configuration**



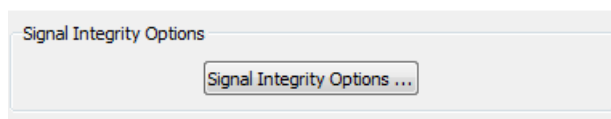
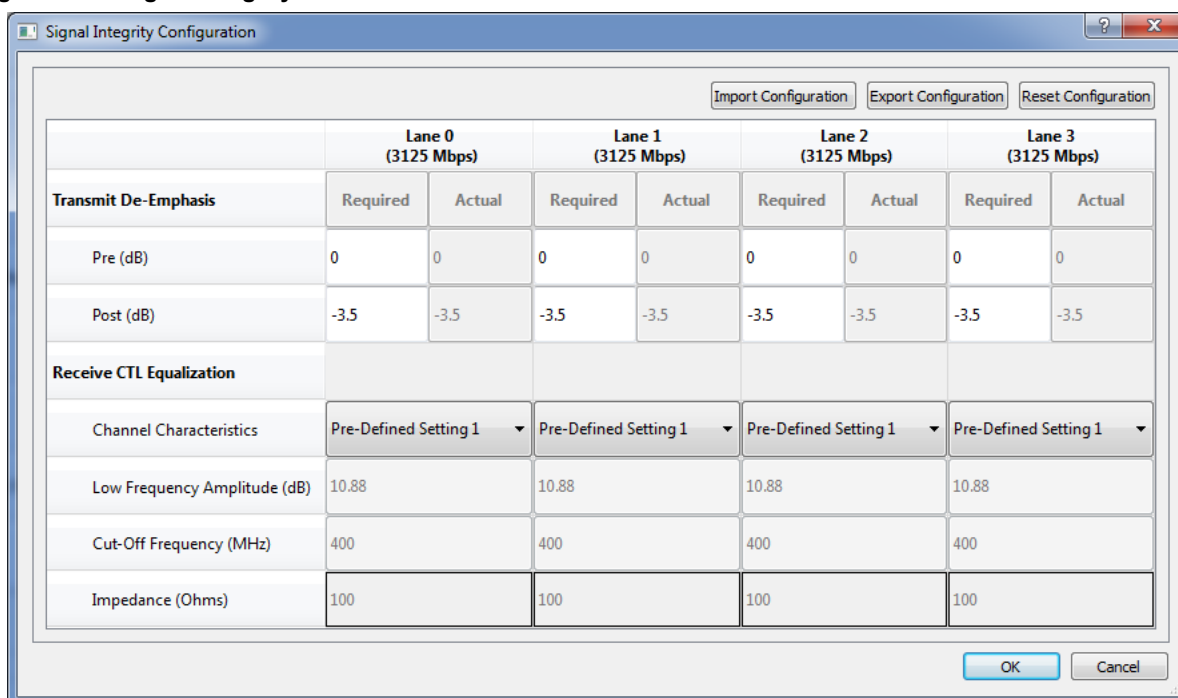
**Figure 57 • Reference Clock Configurator**



**Table 152 • Reference Clock Receiver Function Configuration**

	Options	Default	Details
Input	Enable and disable	Enable	
I/O Standard	LVDS33, LVPECL33, LVDS25, RSDS, and MiniLVDS.	LVDS33	
ODT Impedance	Unterminated and 100 $\Omega$ .	Unterminated	
Schmitt Trigger	On and Off	Off	
Weak Pull Mode	On and Off	Off	

In addition to the protocol settings, various options for signal integrity are available through the **Signal Integrity Options** portion of the XAUI configurator shown in the following figure. Users can adjust the receiver and transmit options to match their system as shown in the following figure.

**Figure 58 • Signal Integrity Options****Figure 59 • Signal Integrity Parameters**

### 3.2.10 Transmit De-Emphasis

Enter any value between 0.0 dB and 36.1 dB in the **Required** edit box for both Pre-transmit and Post-transmit stage. See the following table for all actual values supported. The value entered in the Required box matches to the closest valid actual value. The configurator sets appropriate values for LANE<n>\_TX\_PRE\_RATIO and LANE<n>\_TX\_PST\_RATIO registers based on the actual value.

LANE<n>\_TX\_PRE\_RATIO and LANE<n>\_TX\_PST\_RATIO registers are set based on the actual value. The default value for **Required** is 0 dB for Pre-Transmit and 3.5 dB for Post-Transmit. Lane<n>\_TX\_AMP\_RATIO registers are set to 0x80 for all lanes.

**Note:** LANE<n> denotes the lane number where <n> can be 0, 1, 2 or 3. For example, if you enter 2.4 dB in the **Required** box, 2.5 dB (the closest match) is displayed in the Actual box and the registers are set as follows:

- LANE<n>\_TX\_PRE\_RATIO registers are set to 0x10.
- LANE<n>\_TX\_PST\_RATIO registers are set to 0x10.
- LANE<n>\_TX\_AMP\_RATIO registers are set to 0x80 for all lanes.

### 3.2.11 Receive CTL Equalization

There are 14 predefined settings available for user selection, in addition to the CTLE disabled option. Each predefined setting has the corresponding cut-off frequency and low frequency amplitude values preset and displayed when the predefined setting is selected. The impedance value for all predefined settings is 100  $\Omega$ . Depending on the predefined setting, lane registers LANE<n>\_RE\_AMP\_RATIO and LANE<n>\_RE\_CUT\_RATIO are set to the values listed in the following table.

**Table 153 • CTL Equalization Predefined Settings and Register Values**

Pre-defined Setting #	Cut-Off Frequency (MHz)	Low Frequency Amplitude (dB)	LANE<n>_RE_AMP_RATIO (HEX)	LANE<n>_RE_CUT_RATIO (HEX)
CTLE Disabled	N/A	N/A	0x00	0x00
1	400	10.88	0x77	0x20
2	500	13.06	0x5B	0x2A
3	600	13.98	0x51	0x2F
4	700	12.04	0x4A	0x32
5	800	13.38	0x3E	0x3C
6	900	13.98	0x39	0x40
7	1000	12.57	0x70	0x4F
8	1100	11.6	0x7D	0x58
9	1200	10.88	0x37	0xFE
10	1300	9.95	0x22	0xFC
11	1400	8.52	0x52	0xFE
12	1500	7.47	0x5B	0xFE
13	1600	7.04	0x73	0xFE
14	1700	6.66	0x78	0xFE

**Note:** LANE<n> denotes the lane number where <n> can be 0, 1, 2, or 3.

### 3.2.12 Simulating the SerDes Block with XAUI Mode

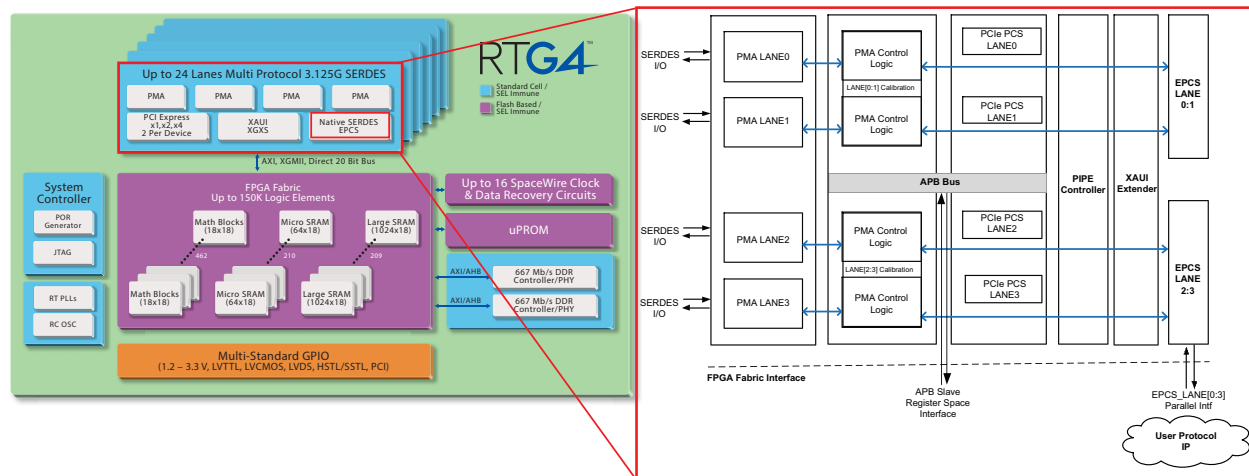
When configured in XAUI mode, the SerDes block only allows the user to run simulation using the APB3 interface. The user can read and write to the SerDes block and SerDes registers using the simulation library. Refer to the [SERDES Block BFM Simulation Guide](#) for details on using RTL mode for simulation. BFM mode for simulation is not supported for XAUI.

## 3.3 EPCS

### 3.3.1 Introduction

The SerDes block integrates the functionality of supporting multiple high-speed serial protocols such as PCIe 2.0, XAUI, and EPCS interfaces, as shown in the following figure. The SerDes block can be configured in various modes, including EPCS mode. In EPCS mode, Lane0 and Lane1 (L01) EPCS interface and Lane2 and Lane3 (L23) EPCS interface are exposed to the fabric and configure the SerDes in PMA only mode. The PCIe and XAUI PCS logic in SerDes is bypassed, however, the PCS logic can be implemented in the FPGA fabric and the EPCS interface signals of the SerDes block can be connected. This allows any user-defined high-speed serial protocol to be implemented in the RTG4 FPGA device.

**Figure 60 • RTG4 SerDes Block Diagram**



### 3.3.2 Features

Following are the main features of the EPCS interface in the RTG4 devices:

- Up to a 20-bit Rx/Tx EPCS Interface to the FPGA fabric.
- Allows the FPGA fabric to directly access the PMA block, bypassing the PCIe PCS block in SerDes and allows implementing any serial protocol for up to four lanes using the PCS logic in the fabric.
- Allows the FPGA fabric to access the SerDes register through the APB interface and allows programming various PMA settings, including programming of the SerDes Tx PLL and Rx PLL settings.

### 3.3.3 Device Support

The following table lists the total number of SerDes blocks available in each RTG4 device that can be configured to support the EPCS interface.

**Table 154 • Available SerDes Blocks in RTG4 Devices**

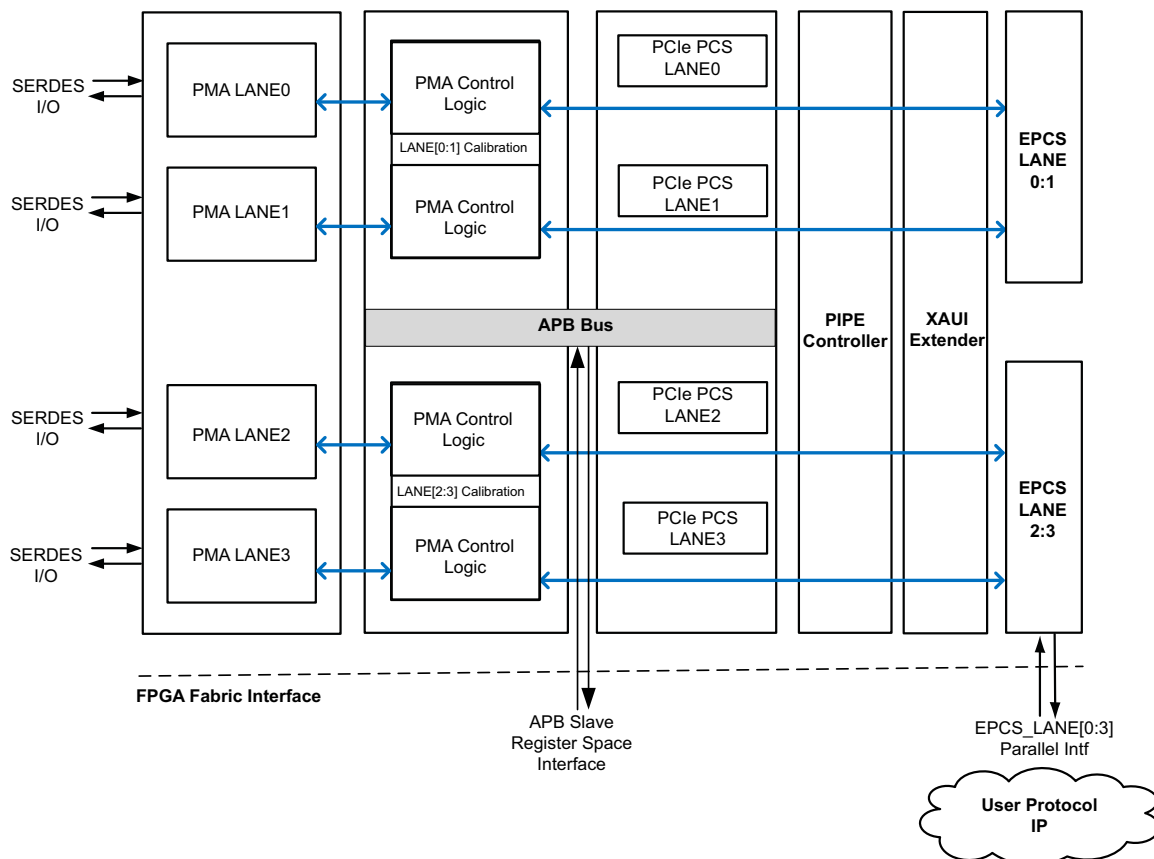
	RT4G150
SerDes block quads available	6
Number of SerDes Lanes available	24

**Note:** Each SerDes block supports 4-EPCS Lanes.

### 3.3.4 RTG4 EPCS Interface

All SerDes blocks can be configured in EPCS mode. This allows the FPGA fabric to directly access the SerDes block. The PCS logic of SerDes is bypassed in this mode to allow user-defined protocol to be supported from the FPGA fabric. The following figure shows an application example using the EPCS interface.

**Figure 61 • Application using SerDes Block EPCS Interface**



**Table 155 • EPCS Interface Usage in Single-Protocol and Multi-Protocol Mode**

Mode	Protocol	Description
Single-protocol	EPCS Protocol	EPCS interface can be configured as x4 or x2 or x1 lane. Configured to use maximum of 4 lanes. In EPCS mode, the user-defined serial protocol implemented within the FPGA fabric is connected through the EPCS interface.
Multi-protocol <sup>1</sup>	PCIe protocol and EPCS protocol	The SerDes block can operate using dedicated Lane2 and Lane3 in EPCS mode, while Lane0 and Lane1 are dedicated to the PCIe protocol link implementation. Any user-defined/other serial protocols connected to the EPCS interface use lane2 and lane3 for this purpose.
Multi-EPCS	EPCS Protocol	Configure multiple independent EPCS protocols across Lane[0:1] and Lane[2:3]. SerDes block allows up to four Lanes to be used independently for the customized EPCS protocols such as SGMII or JESD204b.

1. PCIeSS Type SerDes block only.



**Table 156 • EPCS Interface and SerDes Lane Mapping in Single-Protocol Mode**

Single Protocol EPCS Mode	Lane0	Lane1	Lane2	Lane3
	EPCS			
	EPCS	EPCS		
				EPCS
			EPCS	EPCS
	EPCS	EPCS	EPCS	EPCS

### 3.3.5 SerDes Block Architecture in EPCS Mode

This section provides an overview of the SerDes block in EPCS mode. It includes the following modes:

- SerDes block in EPCS mode
- SerDes block fabric interface in EPCS mode

#### 3.3.5.1 SerDes Block in EPCS Mode

Figure 61, page 109 shows the SerDes block internal architecture during EPCS Single-protocol mode. EPCS mode facilitates the use of four lanes of the SerDes which are exposed to the FPGA fabric with up to 20-bit EPCS interface per lane. EPCS mode gives full control over the PLL configurations in the SerDes using the APB interface to generate the required serial link frequencies. The EPCS interface is suitable for running any protocol, including Ethernet MAC and PCS in the FPGA fabric. The PMA block is used to implement any standard (SRIO, JESD204, etc.) or user-defined serial protocol.

#### 3.3.5.2 SerDes Block Fabric Interface in EPCS Mode

The SerDes block in EPCS mode interfaces with the fabric and differential I/O pads. The following sections list the fabric interfaces:

- EPCS interface signals
- APB interface signals

**Table 157 • SerDes Block – EPCS Interface**

Port	Type	Description
EPCS_0_RESET_N EPCS_1_RESET_N EPCS_2_RESET_N EPCS_3_RESET_N	Input	Active low PHY RESET. These inputs asynchronously reset the associated SerDes logic for each lane. The resets are logically ordered with the power up signal. Toggling EPCS_x_RESET_N restarts calibration.
EPCS_0_READY EPCS_1_READY EPCS_2_READY EPCS_3_READY	Output	PHY ready: This signal is asserted when the PHY has completed the calibration sequence for each specific lane. This calibration can fail if device is not properly powered. The EPCS_READY output to the fabric does not assert until the TX_FWF has completed its startup. If the FWF over or under runs, the EPCS_READY is deasserted. This signal can be used to release the reset for the external PCS and controller, start data transmission data to the PMA, or any other purpose. The EPCS_x_READY output rises synchronously with an internal clock domain inside the SerDes block, which is asynchronous to the user logic domain. The user must use a back-to-back F*F in the fabric to resynchronize this output in the fabric.
EPCS_0_PWRDN EPCS_1_PWRDN EPCS_2_PWRDN EPCS_3_PWRDN	Input	PHY power-down: This active high signal is used to put the PMA in power-down state where RX CDR PLL is bypassed and other low power features are applied to the PMA. When exiting power-down, no calibration is required and the link is operational much faster than when using the EPCS_x_TX_OOB or EPCS_x_RESETN signals. This signal is asynchronous as they are latched by synchronizer inside the SerDes block having no relationship to clocks to the fabric.

**Table 157 • SerDes Block – EPCSInterface**

Port	Type	Description
EPCS_0_TX_OOB EPCS_1_TX_OOB EPCS_2_TX_OOB EPCS_3_TX_OOB	Input	PHY transmit out-of-band (OOB): This signal is used to load electrical idle 3 in the TX driver of the PMA block. It can be used for serial advanced technology attachment (SATA) as part of the sequencing for short OOB signal transmission. These signals are active high. Minimum transfer burst size is 23 symbols. The EPCS_x_TX_OOB is synchronously clocked with the falling edge of the EPCS_x_TXFWF_WCLK.
EPCS_0_TX_VAL EPCS_1_TX_VAL EPCS_2_TX_VAL EPCS_3_TX_VAL	Input	PHY transmit valid: This signal is used to transmit valid data. If de-asserted, the PMA block is put in electrical idle 1. It can be used for protocols requiring electrical idle (SATA) and must also be de-asserted when EPCS_x_READY is not asserted. This signal must be generated one clock cycle earlier than the corresponding EPCS_TXDATA signals. The EPCS_x_TX_VAL is synchronously clocked with the falling edge of the EPCS_x_TXFWF_WCLK.
EPCS_0_TX_DATA[19:0] EPCS_1_TX_DATA[19:0] EPCS_2_TX_DATA[19:0] EPCS_3_TX_DATA[19:0]	Input	PHY transmit data: This signal is used to transmit data. This signal is always 20 bits per lane, but the SerDes block only uses the number of bits selected in the high-speed serial interfaces configurator. The EPCS_x_TX_DATA bus is synchronously clocked with the falling edge of the EPCS_x_TXFWF_WCLK.
EPCS_0_TX_CLK EPCS_1_TX_CLK EPCS_2_TX_CLK EPCS_3_TX_CLK	Output	PHY transmit clock: This clock signal is generated by the TX PLL in the PMA block and must be used by the external PCS logic to provide data on EPCS_x_TX_DATA.
EPCS_0_TX_RESET_N EPCS_1_TX_RESET_N EPCS_2_TX_RESET_N EPCS_3_TX_RESET_N	Output	PHY clean active low reset on the TX clock. The EPCS_x_TX_RESET_N output rises synchronously with an internal clock domain inside the SerDes block, which is asynchronous to the user logic domain. The user must use a back-to-back F*F in the fabric to resynchronize this output in the fabric.
EPCS_0_RX_CLK EPCS_1_RX_CLK EPCS_2_RX_CLK EPCS_3_RX_CLK	Output	PHY receive clock: This clock signal is generated by the RX PLL in the PMA macro and must be used by the external PCS logic to provide data on EPCS_x_RX_DATA.
EPCS_0_RX_RESET_N EPCS_1_RX_RESET_N EPCS_2_RX_RESET_N EPCS_3_RX_RESET_N	Output	PHY active low reset on EPCS_x_RX_CLK: The EPCS_x_RX_RESET_N output rises synchronously with an internal clock domain inside the SerDes block, which is asynchronous to the user logic domain. The user must use a back-to-back flip-flop in the fabric to resynchronize this output in the fabric.
EPCS_0_RX_VAL EPCS_1_RX_VAL EPCS_2_RX_VAL EPCS_3_RX_VAL	Output	PHY receive valid: This signal is used to signal receive valid data. It corresponds to the two conditions completed by the PMA control logic: Receiver detects incoming data (not in electrical idle) CDR PLL is locked to the input bit stream in fine grain state. The EPCS_x_RX_VAL is synchronously clocked with the falling edge of the EPCS_x_RXFWF_RCLK.
EPCS_0_RX_IDLE EPCS_1_RX_IDLE EPCS_2_RX_IDLE EPCS_3_RX_IDLE	Output	PHY receive idle: This signal is used to signal an electrical idle condition detected by the PMA control logic. The EPCS_x_RX_IDLE output rises synchronously with an internal clock domain inside the SerDes block, which is asynchronous to the user logic domain. The user must use a back-to-back flip-flop in the fabric to resynchronize this output in the fabric.
EPCS_0_RXDATA[19:0] EPCS_1_RXDATA[19:0] EPCS_2_RXDATA[19:0] EPCS_3_RXDATA[19:0]	Output	PHY receive data: This signal is always 20 bits per lane and the external PCS can use any number of these bits for its application. The SerDes block only uses the number of bits selected in the high-speed serial interfaces configurator. The EPCS_x_RX_DATA bus is synchronously clocked with the rising edge of the EPCS_x_RXFWF_RCLK.

**Table 157 • SerDes Block – EPCSInterface**

Port	Type	Description
EPCS_0_TX_CLK_STABLE EPCS_1_TX_CLK_STABLE EPCS_2_TX_CLK_STABLE EPCS_3_TX_CLK_STABLE	Output	Active high signal to indicate EPCS interface Lane_X clock is stable, this indicates that the TX PLL is locked. The EPCS_x_TX_CLK_STABLE output rises synchronously with an internal clock domain inside the SerDes block, which is asynchronous to the user logic domain. The user must use a back-to-back flip-flop in the fabric to resynchronize this output in the fabric.
EPCS_0_RX_ERR EPCS_1_RX_ERR EPCS_2_RX_ERR EPCS_3_RX_ERR	Input	Active high input to indicate a receiver error is detected when using the external logic. When there are many receive errors such as, invalid 8b/10b code or disparity error, then the asynchronous signal can be used to switch back the CDRPLL to the frequency lock phase. These pins can be hardwired to 0, and rely only on electrical Idle detection to switch the CDR PLL back to frequency lock state. This signal is asynchronous as they are latched by synchronizer inside the SerDes block having no relationship to the fabric clocks. Typically, this signal is tied low in the design.
GLOBAL_0_OUT GLOBAL_1_OUT	Output	In EPCS modes, these ports are optionally configured to send any of the following signals out of SerDes block on to the dedicated global clock structures: <ul style="list-style-type: none"> <li>• EPCS_RXCLK[3:0]</li> <li>• EPCS_TXCLK[3:0]</li> <li>• REFCLK[P:N]</li> <li>• SPLL_CLK</li> <li>• Logic 0 (Drives logic zero onto the associated GLOBAL_[1:0]_OUT)</li> <li>• Logic 1 (Drives logic one onto the associated GLOBAL_[1:0]_OUT)</li> </ul>
EPCS_0_ARXSKIPBIT EPCS_1_ARXSKIPBIT EPCS_2_ARXSKIPBIT EPCS_3_ARXSKIPBIT	Input	EPCS PMA slip bit for lane [0:3]. Asynchronous rising edge active signal, which shifts the CDR high-speed clock to the next high speed bit. This can be used to shift the EPCS_RX_DATA[n:0] to the left by 1 bit until the proper word alignment is found. The functionality of this signal is internally asynchronous (a rising edge on this pin results in a slip), but the user must initiate it using a flip-flop in the fabric. This signal resets the Rx FWF (fly-wheel FIFO) clearing over/under flow flags. The slip causes the CDR to momentarily change one of its clock periods by 1 UI. ARXSKIPBIT (1UI) adjustments done over a short period of time can falsely cause the Lock Detector to see a different frequency coming from the CDR.
EPCS_0_RXFWF_RCLK EPCS_1_RXFWF_RCLK EPCS_2_RXFWF_RCLK EPCS_3_RXFWF_RCLK	Input	EPCS receive interface FIFO read clock for EPCS lane [0:3]. Read clock of the receive Fly-Wheel-FIFO used to decouple the phase of the FPGA clock from the SerDes receiver clock. Use of the Fly-Wheel-FIFO makes FPGA timing closure easier on the EPCS_RX_DATA[n:0] if the same clock is used to launch and capture data into the fabric. For more information, see <a href="#">Flywheel FIFO</a> , page 115.
EPCS_0_TXFWF_WCLK EPCS_1_TXFWF_WCLK EPCS_2_TXFWF_WCLK EPCS_3_TXFWF_WCLK	Input	EPCS transmit interface FIFO write clock for EPCS lane [0:3]: Write clock to the transmit Fly-Wheel-FIFO, used to decouple the phase of the FPGA clock from the SerDes transmit clock. Use of the Fly-Wheel-FIFO makes the FPGA timing closure easier on the EPCS_TX_DATA[n:0] by removing the round trip time of the clock from the SerDes to the fabric clock resource and back to the SerDes to capture the data. For more information, see <a href="#">Flywheel FIFO</a> , page 115.
EPCS_FAB_REF_CLK	Input	Available when fabric is selected as the reference clock source in the configurator.

**Table 158 • EPCS APB Slave Interface**

Port	Type	Description
APB_S_CLK	Input	PCLK for APB slave interface in the SerDes block.
APB_S_PSEL	Input	APB slave select; selects signal for register for reads or writes.
APB_S_PENABLE	Input	APB strobe. This signal indicates the second cycle of an APB transfer.
APB_S_PWRITE	Input	APB write or read. If high, a write occurs when an APB transfer takes place. If low, a read takes place.
APB_S_PADDR[13:0]	Input	APB address bus.
APB_S_PWDATA[31:0]	Input	APB write data.
APB_S_PREADY	Output	APB ready. Used to insert wait states.
APB_S_PRDATA[31:0]	Output	APB read data.
APB_S_PSLVERR	Output	APB Error.

### 3.3.6 Reset and Clocks

This section describes the functional aspects of the reset and clock circuitry inside the SerDes block in EPCS mode.

#### 3.3.6.1 EPCS Mode Clocking

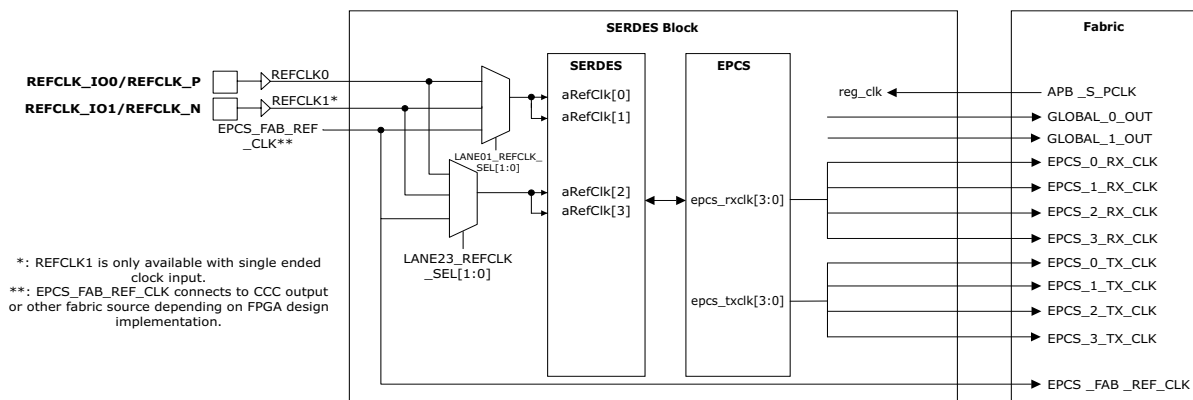
When the SerDes block is configured in EPCS mode, it has multiple clock inputs and outputs. This section describes the EPCS clocking scheme.

##### 3.3.6.1.1 SerDes Block Clock Network in EPCS

Figure 62, page 114 shows the SerDes block clock network in EPCS mode. In EPCS mode, data is exchanged between the fabric and SerDes block. The SerDes PMA has two PLLs (Tx PLL and CDR PLL) that generate the required clock frequency and send 4-Rx and 4-Tx clocks for each lane through the EPCS interface. The user design is required to use these clocks within the FPGA fabric, in custom logic or FPGA IP, to transfer data between the SerDes block and the FPGA fabric. There is an additional clock (asynchronous) dedicated to the APB interface, called APB\_S\_CLK, used for accessing the SerDes block configuration registers. The user has the option to connect several clock outputs from SerDes blocks to two available global clock structures within the FPGA fabric. GLOBAL\_[0:1]\_OUT ports are available per SerDes block and can be programmed to expose the EPCS\_RXCLK[3:0], EPCS\_TXCLK[3:0], REFCLK[0:1], or SPL\_CLK to the global fabric routing.

The RTG4 devices use two vertical stripes of Row Global Buffers (RGBs), 1 per vertical half chip of the device, and 24 chip-level Global Buffers (GBs). The RTG4 clocking architecture is limited to drive no more than 8 global clocks in any half-row. The SerDes blocks span two half-rows – one PCIESS and two NPSS in each half. PCIESS\_SERDES\_PCIE\_0, NPSS\_SERDES\_1, and NPSS\_SERDES\_2 are located in the left vertical half of the device while NPSS\_SERDES\_3, NPSS\_SERDES\_4, and PCIESS\_SERDES\_PCIE\_5 are located on the right vertical half of the device. The user design must consider this limit when connecting the SerDes clocks to globals. Therefore, for multiple SerDes block designs, placement considerations are required to prevent global clock utilization issues. For more information about the RTG4 global network, see [UG0586: RTG4 FPGA Clocking Resources User Guide](#).

An EPCS transmit and receive interface FIFO is in the data path permitting relaxation of the clock phase relationship between the fabric logic and EPCS block. Without the use of interface FIFOs, the recovered clocks provided directly from the PMA block and EPCS data must have their timing relationships tightly controlled to ensure correct RX data capture in the FPGA fabric. The EPCS fabric interface Tx/Rx FIFO permits the fabric clock to have a slight phase offset, which enables TX\_CLK and RX\_CLK sharing across lanes for a multi-lane link. This is done by exposing the FIFO WRITE and READ (EPCS\_x\_TXFWF\_WCLK and EPCS\_x\_RXFWF\_RCLK) clocks. See [Flywheel FIFO](#), page 115 for more information about the EPCS interface FIFO.

**Figure 62 • SerDes Block Clocking in EPCS Mode**

**Note:** GLOBAL\_0\_OUT and GLOBAL\_1\_OUT can be optionally connected to any of the following SerDes clock outputs; EPCS\_RXCLK[3:0], EPCS\_TXCLK[3:0], REFCLK[0:1], SPILL\_CLK. This option directly drives the selected SerDes clock onto a global clock resource.

The following table lists the various clocks in EPCS mode.

**Table 159 • Clock Signals in EPCS Mode**

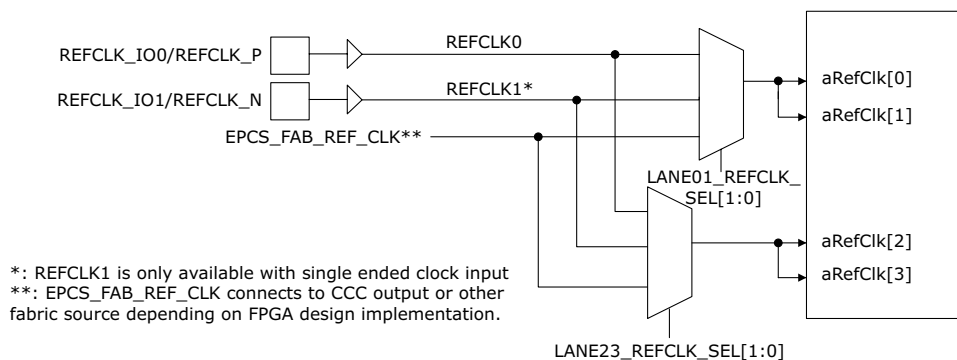
Clock Signal	Direction	Description
aREFCLK	Input	Reference clock for SerDes (Internal signals to TxPLL and RxPLL from REFCLK).
EPCS_x_TX_CLK	Output	EPCS interface LaneX (X = 0, 1, 2, 3) TxClock.
EPCS_x_RX_CLK	Output	EPCS interface LaneX (X = 0, 1, 2, 3) RxClock.
APB_S_CLK	Input	PCLK for APB interface.
REFCLK_P REFCLK_N	IN PAD	Differential reference clock input I/O to REFCLK_0 port (REFCLK_1 is not available with differential clock input).
REFCLK_IO0	IN PAD	Single-ended reference clock input to either Lanes[0:1] or Lanes[2:3].
REFCLK_IO1	IN PAD	Single-ended reference clock input to either Lanes[0:1] or Lanes[2:3].
EPCS_FAB_REF_CLK	Input	Fabric reference clock for SerDes PMA. Can optionally be programmed by Libero to be sent into the fabric.
EPCS_x_TXFWF_WCLK	Input	Fly-wheel FIFO write clock
EPCS_x_RXFWF_RCLK	Input	Fly-wheel FIFO read clock
GLOBAL_0_OUT/ GLOBAL_1_OUT	Output	Global clock resource optionally connected by Libero to any of the dedicated SerDes clock pins.

### 3.3.6.1.2 SerDes Reference Clock Selection

The PMA in the SerDes block needs a reference clock on each of its lanes for Tx and Rx clock generation through the PLLs. It has three options for the reference clock. Two REFCLK I/O pad options or from the FPGA fabric. Lane0 and lane1 share the same reference clock; lane2 and lane3 share the same reference clock, or alternatively the same clock can be shared among all four lanes. The reference clock pads can be configured as one differential reference clock input per quad or as two single-ended reference clock inputs per quad. See Figure 63, page 115.

The reference clock selection in the SerDes block configurator is available in Libero. See [Reference Clock Configuration](#), page 125 for information about Libero REFCLK configuration.

**Figure 63 • SerDes Reference Clock Options for EPCS Modes**



### 3.3.6.2 Flywheel FIFO

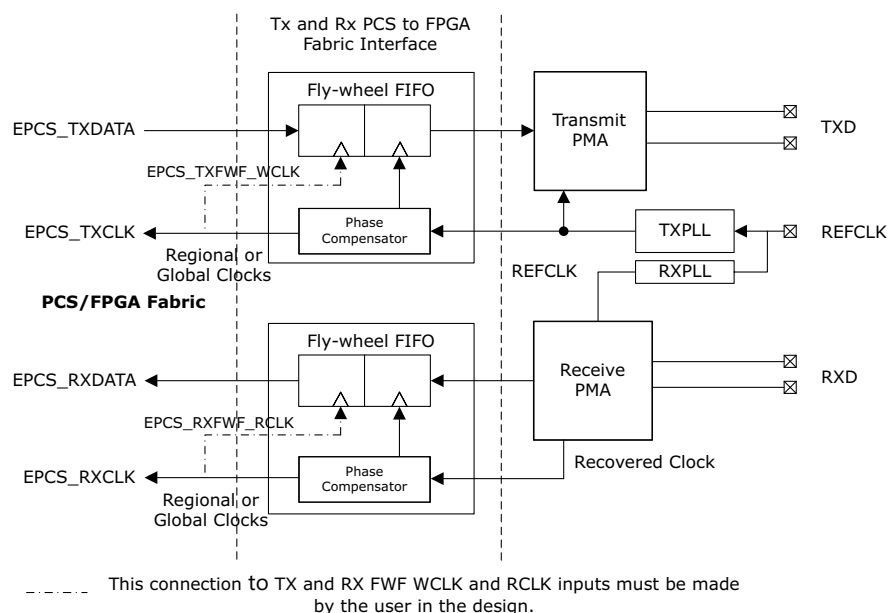
As shown in [Figure 64](#), page 115, the SerDes PMA to FPGA fabric/PCS data path includes a Flywheel FIFO (FWF) interface, which is used to transfer data between clock domains. The FWF is always present in EPCS mode and cannot be bypassed. The FWF is designed to handle phase (not frequency) crossing as data moves from the SerDes PMA to the fabric-based protocol logic (fabric PCS). In EPCS mode, the use of the FWF requires the user to connect the FWF input ports of the SerDes block to the appropriate clock resources.

The FWF is a simple form of FIFO where its write and read clocks are known to be at the same nominal frequency with some allowed phase difference and jitter that is compensated within the block.

The addition of the FWF block in the data path handles the phase crossing of every PMA lane, ensuring that timing is met across this interface. The received data, along with the recovered clock, is passed to the FWF, which synchronizes data and clock for consumption in the FPGA fabric.

In the transmit direction, data from the fabric or PCS is passed through FWF along with a clock from the fabric to the PMA ensuring synchronous clock and data relationships passing to the PMA interface.

**Figure 64 • Flywheel FIFO**





### 3.3.6.2.1 Using Flywheel FIFO with Direct Global Connections

As shown in Figure 62, page 114 in EPCS modes, the SerDes clocks can be optionally configured to be sent out of the SerDes block onto the dedicated global clock tree. There are two global clocks (GLOBAL\_0\_OUT, GLOBAL\_1\_OUT) available per quad (4-lanes). The EPCS\_RXCLK and EPCS\_TXCLK outputs from SerDes can route to a global in the fabric using these dedicated global clocks. The Libero configurator can be used to select the SerDes global outputs via hardwired connections into the FPGA fabric by connecting the selected SerDes clock to the GLOBAL\_[1:0]\_OUT. In cases where both GLOBAL\_OUT connections are used, the user can manually instantiate RCLKINTs/CLKINTs, however, this requires careful design planning to ensure the number of globals required in the top-row of RGBs does not exceed the eight available per half-row. Furthermore, when manually instantiating RCLKINT/CLKINT macros to promote the recovered EPCS RXCLK, ensure the output of the RCLKINT/CLKINT is sent to both the fabric RX data capture logic and the FWF RCLK input to ensure the skew between the user's PCS logic and FWF fabric interface is minimized.

Typically, the RX data should connect to a register immediately after passing into the FPGA fabric. The EPCS\_RXCLK[3:0] and EPCS\_TXCLK[3:0] are typically used for the read and write side of the RX and TX Flywheel FIFOs, respectively. Phase differences due to clock insertion delay can be compensated by FWF as data passes from PMA to fabric logic (PCS). These clocks are applied to the fabric global network if SerDes global outputs are used. These can be used with the Flywheel FIFO when a quad with multiple lanes is used. However, since only two global clock resources are available, this requires careful planning and clock sharing of the design when four lanes used. If not, the design could unnecessarily consume all the RGB resources in the top-row of the fabric, which drive the SerDes FWF WCLK and RCLK clock inputs. In other words, any global clock inputs to the SerDes quads must be driven by RGBs in the top-row of the fabric.

### 3.3.6.2.2 FWF Write and Read Common Clock Connection for x4 EPCS Link

#### Design Use Case 1

Design cases can use the recovered clock (EPCS\_x\_RX\_CLK) of one lane as the FWF read clk (EPCS\_x\_RXFWF\_RCLK) for up to x4 lanes in a SerDes block. For sharing the recovered clock, the sending end of the link to the RTG4 device must use a common transmit clock for all its lanes. This common transmit clock of the sending device can have ppm offset relative to the RTG4 device's reference clock as per the RTG4 datasheet. Each lane's TXPLL must lock to the same reference clock frequency and all of lane's FWF write clock (EPCS\_x\_TXFWF\_WCLK) inputs to the SerDes block can be driven by the common EPCS\_x\_TX\_CLK through a single global output of the SerDes quad. EPCS\_x\_TX\_CLK can have different short-term period jitter characteristics since the FWF compensates for the slight differences between EPCS\_x\_TX\_CLK in each lane as the data passes from the fabric clock domain to TX clock in the SerDes PMA. The fabric logic for the transmitter protocol IP must also use the same EPCS\_x\_TX\_CLK to keep the fabric side of the TX FWF and the fabric data path aligned.

#### Design Use Case 2

This design case uses a local reference clock as the source for the FWF read clocks (EPCS\_x\_RXFWF\_RCLK). The local reference clock can only be used if it is frequency locked to the sending end of the link. If there is any ppm difference, then this use-case will not work. All four EPCS\_x\_RXFWF\_RCLK inputs to the SerDes block can be driven by a common EPCS\_x\_RX\_CLK through a single global output of the SerDes quad, if a common clocking topology (with 0ppm offset) is used between both sides of the SerDes link. Each lane's RXPLL must recover the same clock frequency. However, the recovered clocks EPCS\_x\_RX\_CLK in each lane can have different phase relationships since the FWF compensates for the phase differences between EPCS\_x\_RX\_CLK in each lane as the RX data passes from the SerDes PMA domain to the fabric receiver logic for the serial protocol. The fabric logic for the receiver protocol IP must also use the common EPCS\_x\_RX\_CLK to keep the fabric side of the RX FWF and the fabric data path aligned.

If the common lane is reset, it affects the clock to all four TX/RX FWF and the serial protocol IP in the fabric. If there is a requirement for a clock to be available regardless of whether the SerDes is in reset then another alternative is to broadcast the REFCLK\_P/N out of the SerDes global output and use that for all four lanes TX/RX FWF and the fabric logic for the protocol IP.

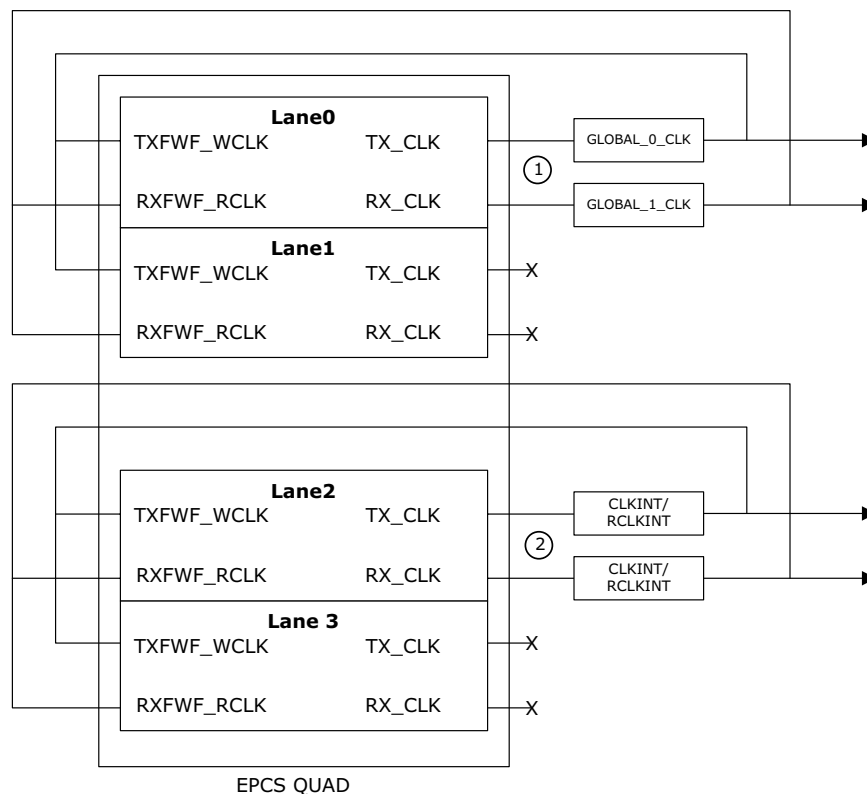
The following applies to both use-cases:

- The data transferred on each lane cannot differ substantially in the run length.
- Each lane recovers a clock with unique frequency variations which results in EPCS\_x\_RX\_VAL going to 0 on the shared lanes initially. ARXSKIPBIT pulses must be applied from the FPGA logic to correct for the initial instability. The ARXSKIPBIT resets the over/under run condition of the FWF.

### 3.3.6.2.3 How to Save Globals for EPCS X2 Links

For two independent, full-duplex X2 links within a quad—Lane 0 EPCS\_0\_TX\_CLK/EPCS\_0\_RX\_CLK can be connected to FWF WCLK/RCLK for lane 0/1 and Lane 2 EPCS\_2\_TX\_CLK/EPCS\_2\_RX\_CLK can be connected to FWF WCLK/RCLK for lanes 2/3. Four fabric global buffers will be required. Two direct global out ports and two CLKINT/RCLKINT accessed through local routing.

**Figure 65 • EPCS X2 Links**



Notes:

- 1 – Dedicated connection directly to global clock spine.
- 2 – Clock connection is indirect from SerDes pin to CLKINT/RCLKINT

### 3.3.6.2.4 Multi-lane Common Clock Synchronization Considerations

Designs using the before-mentioned common RX\_CLK schemes must implement a periodic reset circuit to ensure proper startup.

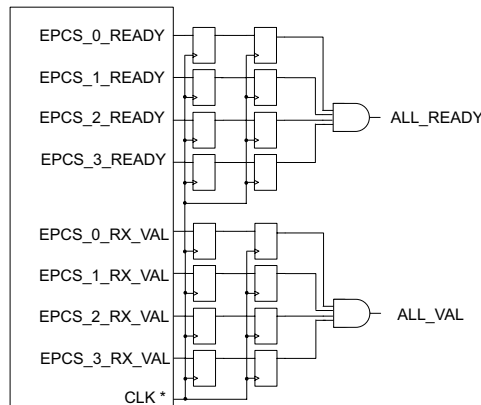
The key to sharing Rx FWF read clocks is resetting the FWFs once all CDRs reach steady-state and the over/under run conditions of the RxFWF can be negated by a circuit (see Figure 66, page 118) that does the following.

1. When ALL\_READY= 0, restart this flow.
2. Apply 1 to EPCS\_{0,1,2,3}\_ARXSKIPBIT pins all at the same time. Wait 24 clocks.
3. Apply 0 to EPCS\_{0,1,2,3}\_ARXSKIPBIT pins all at the same time. Wait 24 clocks.
4. If ALL\_VAL = 1, then go to step (5); else wait for about 1 us and then go to step (2).



5. Fabric PCS process may begin. If the FPGA based PCS detects a loss of sync where symbols on any lane continuously make no sense (disparity and/or code-violations), then go back to step (1).

**Figure 66 • CDR Steady State Considerations Diagram**



### 3.3.6.2.5 General Guidelines for SerDes Clock Connections

Due to SerDes quad placement at the north side of the die, global clock inputs from the fabric to SerDes block must come from top-most row-global buffer (RGBs).

There are eight half-row globals. Each half of RTG4 die is connected to three SerDes quads. These three SerDes quads can only receive eight global clock inputs from the fabric.

Careful planning is required to ensure there are sufficient fabric globals available to route fabric clocks in to FWF. It is also recommended to spread the SerDes usage across both halves of the die if possible rather than trying to force all the SerDes in the design onto one half of the die.

While using full-duplex x4 EPCS link, EPCS\_x\_TXFWF\_WCLK and EPCS\_x\_RXFWF\_RCLK can be connected using two global output clocks (GLOBAL\_0\_OUT, GLOBAL\_1\_OUT).

If the design uses three separate x4 EPCS links on the same half of the die, it is possible to have enough row global clocks for all three quads if the guidelines described in [FWF Write and Read Common Clock Connection for x4 EPCS Link](#), page 116 are followed. If independent clock for TX and RX FWF in each lane is used, then it would require eight global inputs for just a single quad. Performing this for more than one quad results in Libero placer failure.

### 3.3.6.2.6 EPCS Startup and Operating Behavior (20-bit Width)

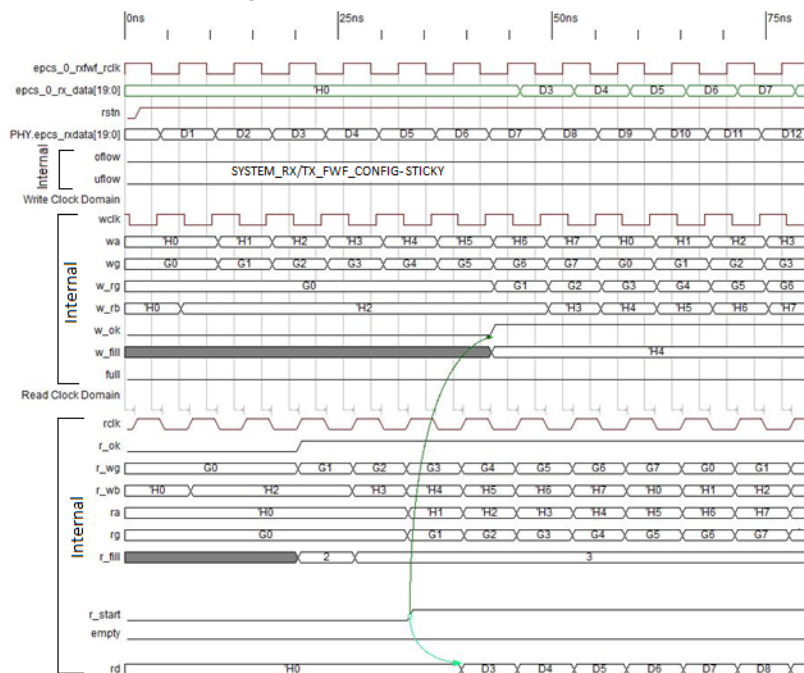
Data from the fabric enters each register file on the falling edge of `txfwf_i_wclk_N` (which is an inversion of `epcs_x_txfwf_wclk`). The data consists of the 20-bit data plus a valid marker and an out-of-band control. This data is launched on the rising edge of `epcs_x_txfwf_wclk` by fabric logic. The use of the falling edge in the ASIC avoids a hold-time problem caused by the fabric-to-ASIC clock-injection delay. The out-of-band control does not strictly need to be synchronous with the transit data and valid control. Options are provided for the out-of-band control and valid to be used independently of the FIFO data controlled by Libero to set the configuration registers (`TXFWF_TXOOB_USE_FIFO/TXVAL_USE_FIFO`).

The embedded FIFO controller (`fwf_ctrl`) is internally transparent to the user. It advances the write address on each clock cycle of `txfwf_i_wclk_N`, such that data is written to the register file in a repetitive cycle of addresses, (0,1,2,...5,6,7,0,1,2,...).

The `fwf_ctrl` similarly advances the read address on each clock cycle of the PHY clock, `EPCS_TXCLK_N`. Upon reset, the `fwf_ctrl` institutes a delay in the startup of the read address cycle count. This startup delay produces a multi-cycle delay of data across the register file. When this offset is setup, it persists unless the clocks from the PHY are not the same frequency as those provided by the fabric. Startup and normal behavior of the `txfwf` are shown in the following figure. The reset given to the `txfwf` is held asserted until after `epcs_ready` from the phy is observed to ensure that the clocks are stable. The `EPCS_READY` output to the fabric does not assert until the `txfwf` has completed its startup. If `EPCS_READY` from the phy deasserts, the `epcs_ready` output to the fabric deasserts immediately. The

EPCS\_READY output to the fabric is from a register on the read side of the txfwf; the fabric is responsible to synchronization to its own clock domain if required.

**Figure 67 • EPCS Startup and Operating Behavior (20-bit Width)**



Startup is caused by the asynchronous reset event,  $\text{rstn}=0$  (EPCS\_x\_RESET\_N). This causes both the write and read address to become 0. The write side begins its cyclic behavior of the write address (wa) as soon as the reset is released. The read side contains a form of reset synchronization causing it to defer normal activity until  $\text{r\_ok}=1$ , which takes three clock cycles after release of the reset. The read domain gets a double-clocked snapshot of a gray-encoded write address upon which it evaluates the fill-level of the FIFO. When  $\text{r\_ok}=1$  and the fill-level of the FIFO equals or exceeds three locations, the read domain logic begins the cyclic incrementing of the read address and asserts  $\text{r\_start}$ .

For the TxFWF, the  $r\_start=0$  condition prevents the passing of FIFO output data to the PHY (light green arrow shown in the preceding figure). The  $r\_start$  rising edge causes the write clock domain to flag  $w\_ok$ . The  $w\_ok=1$  state is used to enable the overflow detection logic. Prior to  $r\_ok$  and  $w\_ok$ , the FIFO cannot issue an underflow nor an overflow error. When  $r\_start=0$ , the FIFO controller hold its valid output low, which is used to force the read data to all zeros.

After  $r\_start=1$ , the FIFO fill level will typically be three or four locations. The phase relationship between the read and write clocks is unknown, and if they are nearly aligned, it is possible for the read side estimation of the fill level to fall to two locations during normal operation (if write clock jitters from just before the read clock to just after). When  $r\_start=1$ , if the read-side fill-level becomes fewer than two (with low-water mark  $LWM=0$ ) or one (with  $LWM=1$ ) locations, the FIFO declares an underflow ( $uflow=1$ ) and remains in this state until the FIFO is reset. This also causes the  $epcs\_ready$  output to the fabric to de-assert.

The write domain does a double-clock capture of the gray-encoded read domain address and evaluates the fill-level of the FIFO. After  $r\_start=1$ , the FIFO fill level will typically be three or four locations. The phase relationship between the read and write clocks is unknown, and if they are nearly aligned, it is possible for the write side estimation of the fill level to rise to five locations during normal operation (if write clock jitters from just after the read clock to just before). When  $w\_ok=1$  and the write side fill-level exceeds six (with  $LWM=0$ ) or seven (with  $LWM=1$ ) locations, the FIFO declares an overflow ( $oflow=1$ ) and remains in this condition until the FIFO is reset. This also causes the  $epcs\_ready$  output to the fabric to de-assert.

Variables referenced in the preceding timing diagram that are not explained are:

- $wg$  = write address in gray code form; this is a write clock domain register
- $w\_rg$  = write domain copy of the gray coded read address
- $w\_rb$  = write domain approximation of the read binary address
- $w\_fill$  = write domain approximation of the fill-level
- $r\_wg$  = read domain copy of the gray coded write address
- $r\_wb$  = read domain approximation of the write binary address
- $rg$  = read address in gray code form; this is a read clock domain register
- $r\_fill$  = read domain approximation of the fill-level

SYSTEM\_TX/RX\_FWF\_CONFIG controls are provided by Libero to correctly setup the FWF, for binding register files to the FIFO controllers, and for optionally allowing the valid and/or out of band signals to bypass the FIFO. It includes Tx and Rx underflow and overflow status registers which are sticky. When set, the overflow and underflow status do not clear until the FIFO is reset.

The Transmit and Receive Flywheel FIFO require 0ppm frequency difference between write and read clocks. Any deviation in frequency between the write and read sides of the FIFO quickly results in an underflow or an overflow because the FIFO depth are very shallow. The overflow and underflow detection logic assume similar clock frequencies on the read and write domains. The clock domain crossing logic may fail when the read and write domain frequencies are extremely dissimilar (> 100,000 ppm apart). With extremely dissimilar clocks applied to the read and write domains, the outcomes include:

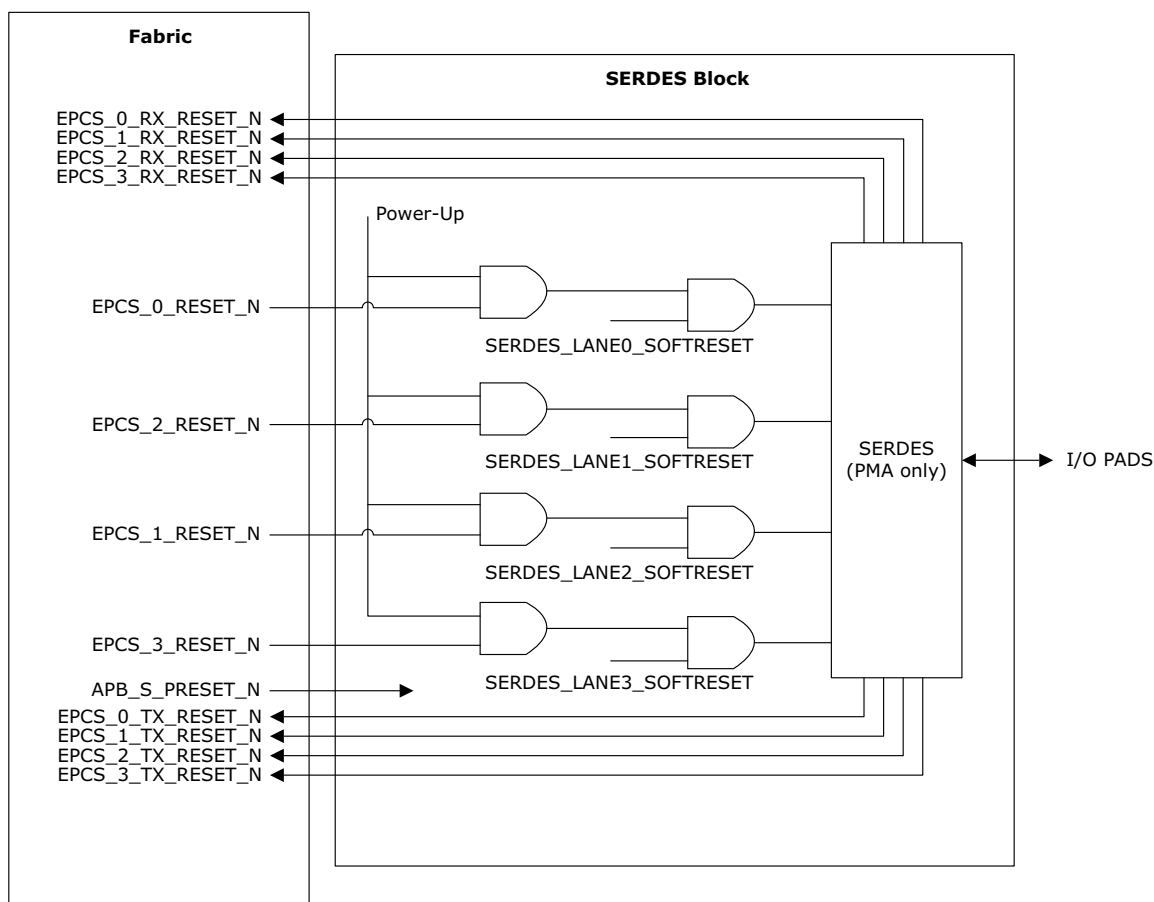
- flywheel may not start-up after reset is de-asserted;
- overflow and/or underflow may never be detected;
- overflow may appear to be detected as underflow, or vice-versa.

### 3.3.6.3 EPCS Mode Reset Network

The following figure shows the reset network for the EPCS interface x4 lanes implementation. The resets for all four lanes (EPCS\_0\_RESET\_N, EPCS\_1\_RESET\_N, EPCS\_2\_RESET\_N, and EPCS\_3\_RESET\_N) are gated with the power valid signal from RTG4 program control and again with SERDES\_LANE<sub>x</sub>\_SOFTRESET ( $x = 0, 1, 2, 3$ ).

The SERDES\_LANE<sub>x</sub>\_SOFTRESET signals are controlled by the [SerDes Block System Registers](#), page 136, SERDES\_Block\_SOFT\_RESET, (active low reset signal for each lane), which can be programmed using the APB interface. On the output side, the SerDes block in EPCS mode generates 4 sets of reset signals, and one for each lane. [Table 160](#), page 122 describes the reset signals and recommended connections.

**Figure 68 • SerDes Block Reset Signals in EPCS Mode**



**Table 160 • SerDes Block Reset Signals in EPCS Mode**

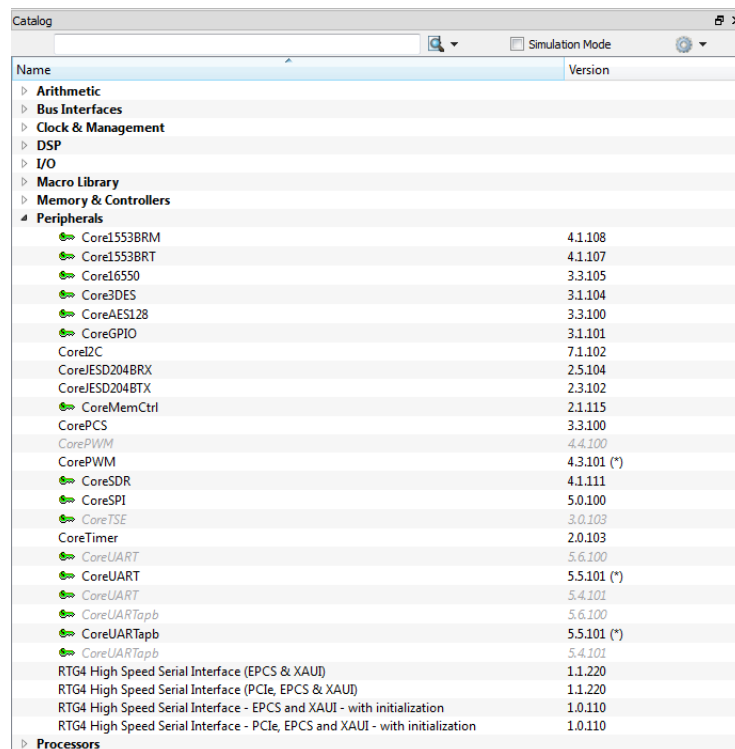
Port	Type	Description
EPCS_x_RESET_N	Input	EPCS interface LaneX (X = 0, 1, 2, 3) reset of the PMA. Asserting this reset low will also cause the PMA to re-calibrate. EPCS_x_RESET_N should be asserted for a minimum of 16 reference clock cycles for a reset pulse-width. The pulse-width on reset is adequately specified to overcome the glitch-filtering implemented in the asynchronous reset tree.
APB_S_PRESET_N	Input	APB asynchronous reset to all APB registers
EPCS_x_Rx_RESET_N	Output	EPCS interface LaneX (X = 0, 1, 2, 3) reset output, de-asserted on EPCS_x_RX_CLK
EPCS_x_Tx_RESET_N	Output	EPCS interface LaneX (X = 0, 1, 2, 3) reset output, de-asserted on EPCS_x_TX_CLK

### 3.3.7 Generating EPCS Block Using Libero SoC

This section describes how to configure the SerDes block in EPCS mode and instructions for using the EPCS interface.

#### 3.3.7.1 Using High-Speed Serial Interfaces Configurator in EPCS Mode

The high-speed serial interfaces configurator (SerDes block configurator) in the Libero SoC software allows configuring the SerDes block with EPCS mode in single-protocol mode, multi-protocol mode or multi-EPCS modes. All SerDes cores within the RTG4 can support EPCS. High-speed serial interface blocks depending on its size. Refer to the RTG4 Datasheet. The Libero SoC software catalog includes four SerDes cores in the Peripherals subsection as shown in the following figure.

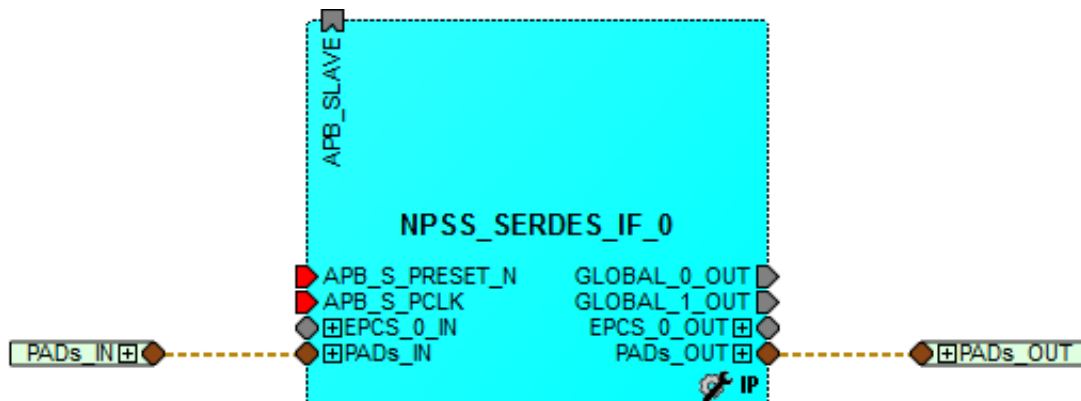
**Figure 69 • Libero SoC Catalog**

Refer to the following software configuration documents for specific details of using XAUI implementations.

- RTG4 High Speed Serial Interface (PCIe, EPCS and XAUI) Configuration User Guide (UG0591)
- RTG4 High Speed Serial Interface (PCIe, EPCS and XAUI) with Initialization User Guide (UG0620)
- RTG4 High Speed Serial Interface (EPCS and XAUI) Configuration User Guide (UG0592)
- RTG4 High Speed Serial Interface (EPCS and XAUI) with Initialization User Guide (UG0619)

For all data rates, there is a CUSTOM SPEED option within the SerDes block configurator. This feature allows designers to craft customized SerDes implementations that requires a high-level of knowledge to program and setup the SerDes block.

**Figure 70 • EPCS Block Instantiation**

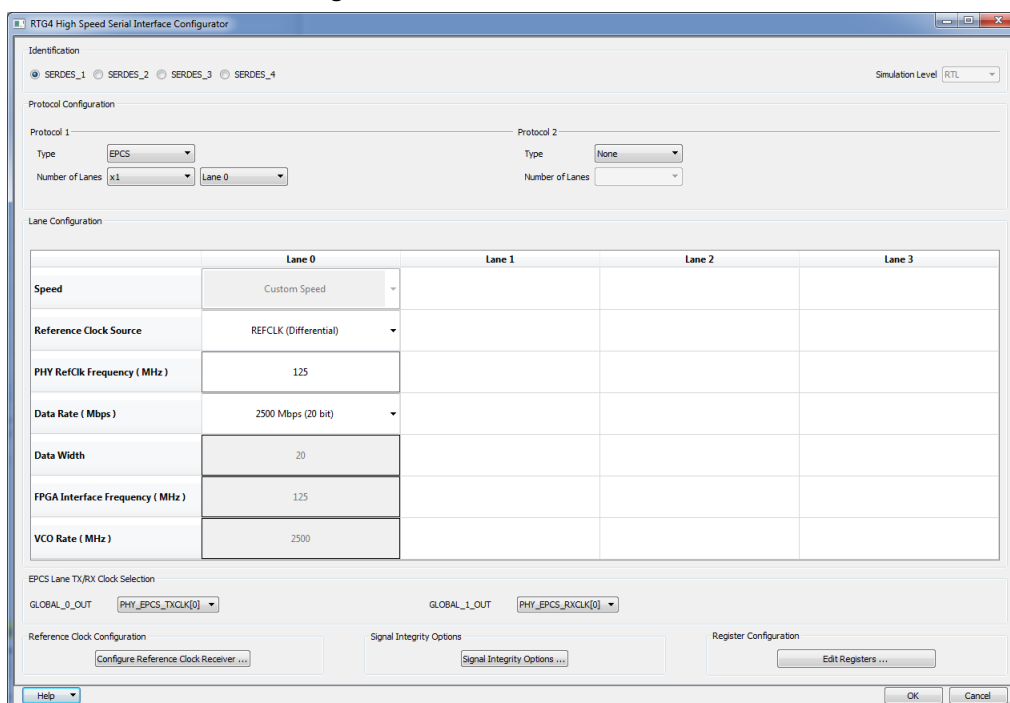


### 3.3.7.1.1 Protocol Selection

The following settings are used for protocol selection:

- Protocol 1 or Protocol 2 Type: Select protocol settings, select EPCS from the drop-down based on single-protocol or multi-protocol mode.
- Configure Protocol 1 before configuring Protocol 2. In Protocol 2, EPCS is available only when XAUI is not selected for Protocol 1. Protocol 2 is disabled when XAUI is selected in Protocol 1.

**Figure 71 • EPCS SerDes Block Configurator**



The screenshot shows the RTG4 High Speed Serial Interface Configurator window. The 'Identification' tab is selected, showing 'SERDES\_1' as the selected device. The 'Protocol Configuration' section shows 'Protocol 1' with 'Type' set to 'EPCS' and 'Number of Lanes' set to 'x1'. 'Protocol 2' is set to 'None'. The 'Lane Configuration' table shows settings for Lane 0:

	Lane 0	Lane 1	Lane 2	Lane 3
Speed	Custom Speed			
Reference Clock Source	REFCLK (Differential)			
PHY RefClk Frequency (MHz)	125			
Data Rate (Mbps)	2500 Mbps (20 bit)			
Data Width	20			
FPGA Interface Frequency (MHz)	125			
VCO Rate (MHz)	2500			

Below the table, the 'EPCS Lane TX/RX Clock Selection' section shows 'GLOBAL\_0\_OUT' set to 'PHY\_EPCS\_TXCLK[0]' and 'GLOBAL\_1\_OUT' set to 'PHY\_EPCS\_RXCLK[0]'. The 'Reference Clock Configuration' section has a 'Configure Reference Clock Receiver' button. The 'Signal Integrity Options' section has a 'Signal Integrity Options' button. The 'Register Configuration' section has an 'Edit Registers' button. The bottom of the window has 'Help', 'OK', and 'Cancel' buttons.

The following table lists the protocol combinations that are feasible within a single High-speed serial interface block.

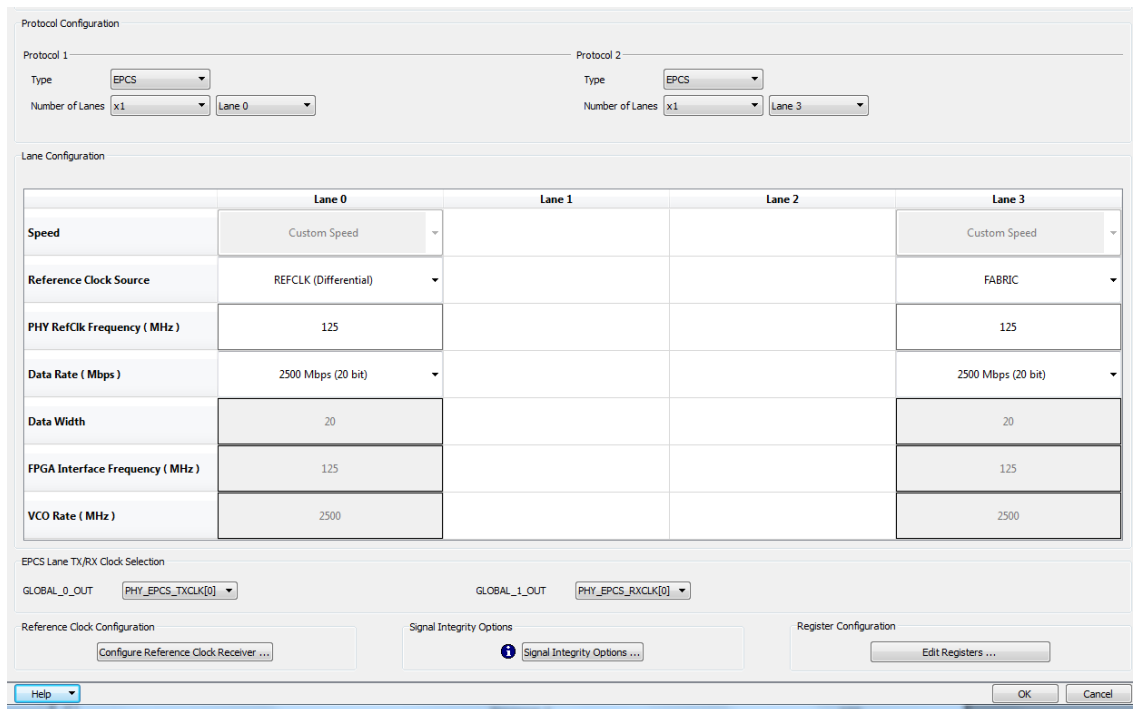
**Table 161 • XAUI Protocol**

Protocol	Lane Width	Lane Assignment	Description	Speed
Protocol 1	X1	Lane 0, Lane 1, Lane 2, and Lane 3	Lane 2 or 3 can be selected when Protocol 2 is not used.	3.125 Gbps
	X2	Lane 0, and Lane 1		
	X4	Lane 0, Lane 1, Lane 2, and Lane 3		
Protocol 2	X1	Lane 2 and Lane 3	Not available when Protocol 1 is XAUI	
	X2	Lane 2 and Lane 3		

### 3.3.8 Lane Configuration

The following figure shows the lane configuration of the EPCS.

**Figure 72 • RTG4 High Speed Serial Interface Configurator—Lane Configuration**



- **Speed** - Custom speed
- **Reference Clock Source** - Clock sources can be differential or single-ended. Select one of the following options for Protocol 1:
  - REFCLK (Differential)
  - REFCLK0 (Voltage Referenced)
  - REFCLK1 (Voltage Referenced)
  - REFCLK0 (Single Ended)
  - REFCLK1 (Single Ended)
  - FABRIC

**Note:** Lane 0 and Lane 1 share the same reference clock and Lane 2 and Lane 3 share the same reference clock. The selected reference clock is available as REFCLK0\_OUT or REFCLK1\_OUT and can be used as clock source for logic in the fabric.

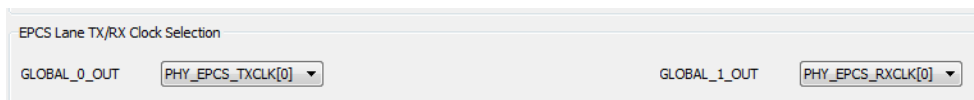
- **PHY RefClk Frequency (MHz)** - enter values between 100 MHz to 160 MHz.
- **Data Rate (Mbps)** - select the data rate from the drop-down menu. Data Rates are computed based on the PHY RefClk Frequency.

- **Data Width** - the data width varies with Data Rate (Mbps).
- **FPGA Interface Frequency (MHz)** - the displayed value is computed and updated based on the PHY RefClk Frequency and Data Rate.
- **VCO Rate (MHz)** - the displayed value is computed and updated based on the PHY RefClk Frequency and Data Rate.

### 3.3.9 EPCS Lane TX/RX Clock Selection

The EPCS Lane TX/RX Clock Selection option is only relevant when the Protocol selected is EPCS. This option allows to select the clocks to use for the transmit/receive fabric interface and the logic in the fabric.

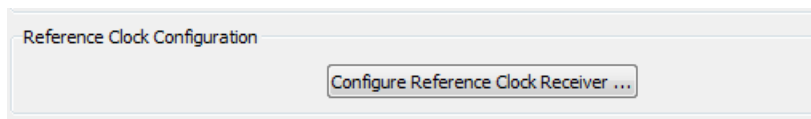
**Figure 73 • EPCS Lane TX/RX Clock Selection**



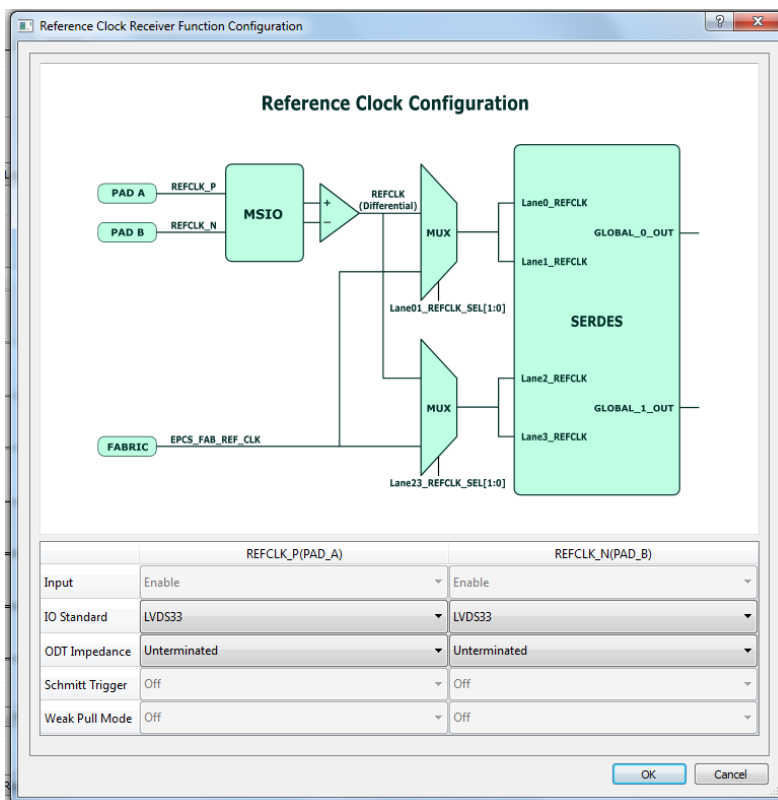
### 3.3.10 Reference Clock Configuration

Reference clock configuration allows to configure the multi-standard user I/O (MSIO) reference clock. Click **Configure Reference Clock Receiver...** to open the Reference Clock Receiver Function Configuration window.

**Figure 74 • Reference Clock Configuration**



**Figure 75 • Reference Clock Receiver Function Configuration**



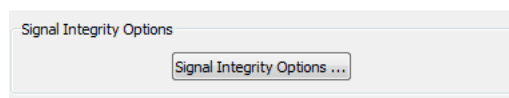
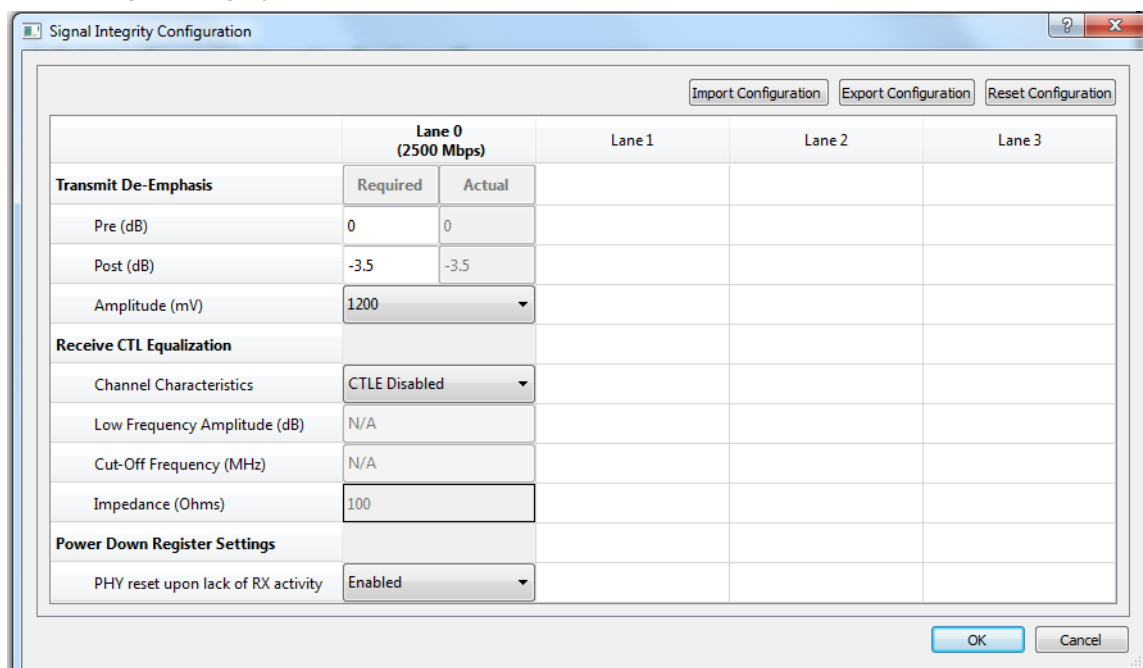


**Table 162 • Reference Clock Receiver Function Configuration**

	Options	Default	Details
Input	Enable and disable	Enable	
I/O Standard	LVDS33, LVPECL33, LVDS25, RSDS, and MiniLVDS.	LVDS33	
ODT Impedance	Unterminated and 100 $\Omega$ .	Unterminated	
Schmitt Trigger	On and Off	Off	
Weak Pull Mode	On and Off	Off	

### 3.3.11 Signal Integrity Options

Click **Signal Integrity Options** to open the signal integrity configuration dialog box.

**Figure 76 • Signal Integrity Options****Figure 77 • Signal Integrity Parameters**

The values entered are used to set register values related to signal integrity. Lanes which are not used are grayed-out in the dialog box.

### 3.3.12 Transmit De-Emphasis

Enter any value between 0.0 dB and 36.1 dB in the **Required** edit box for both Pre-transmit and Post-transmit stage. See the following table for all actual values supported. The value entered in the Required box matches to the closest valid actual value. The configurator sets appropriate values for LANE<n>\_TX\_PRE\_RATIO and LANE<n>\_TX\_PST\_RATIO registers based on the actual value.

LANE<n>\_TX\_PRE\_RATIO and LANE<n>\_TX\_PST\_RATIO registers are set based on the actual value. The default value for **Required** is 0 dB for Pre-Transmit and 3.5 dB for Post-Transmit. Lane<n>\_TX\_AMP\_RATIO registers are set to 0x80 for all lanes.

**Note:** LANE<n> denotes the lane number where <n> can be 0, 1, 2 or 3. For example, if you enter 2.4 dB in the **Required** box, 2.5 dB (the closest match) is displayed in the Actual box and the registers are set as follows:

- LANE<n>\_TX\_PRE\_RATIO registers are set to 0x10.
- LANE<n>\_TX\_PST\_RATIO registers are set to 0x10.
- LANE<n>\_TX\_AMP\_RATIO registers are set to 0x80 for all lanes.

### 3.3.13 Receive CTL Equalization

There are 14 predefined settings available for user selection, in addition to the CTLE disabled option. Each predefined setting has the corresponding cut-off frequency and low frequency amplitude values preset and displayed when the predefined setting is selected. The impedance value for all predefined settings is 100  $\Omega$ . Depending on the predefined setting, lane registers LANE<n>\_RE\_AMP\_RATIO and LANE<n>\_RE\_CUT\_RATIO are set to the values listed in the following table.

**Table 163 • CTL Equalization Predefined Settings and Register Values**

Pre-defined Setting #	Cut-Off Frequency (MHz)	Low Frequency Amplitude (dB)	LANE<n>_RE_AMP_RATIO (HEX)	LANE<n>_RE_CUT_RATIO (HEX)
CTLE Disabled	N/A	N/A	0x00	0x00
1	400	10.88	0x77	0x20
2	500	13.06	0x5B	0x2A
3	600	13.98	0x51	0x2F
4	700	12.04	0x4A	0x32
5	800	13.38	0x3E	0x3C
6	900	13.98	0x39	0x40
7	1000	12.57	0x70	0x4F
8	1100	11.6	0x7D	0x58
9	1200	10.88	0x37	0xFE
10	1300	9.95	0x22	0xFC
11	1400	8.52	0x52	0xFE
12	1500	7.47	0x5B	0xFE
13	1600	7.04	0x73	0xFE
14	1700	6.66	0x78	0xFE

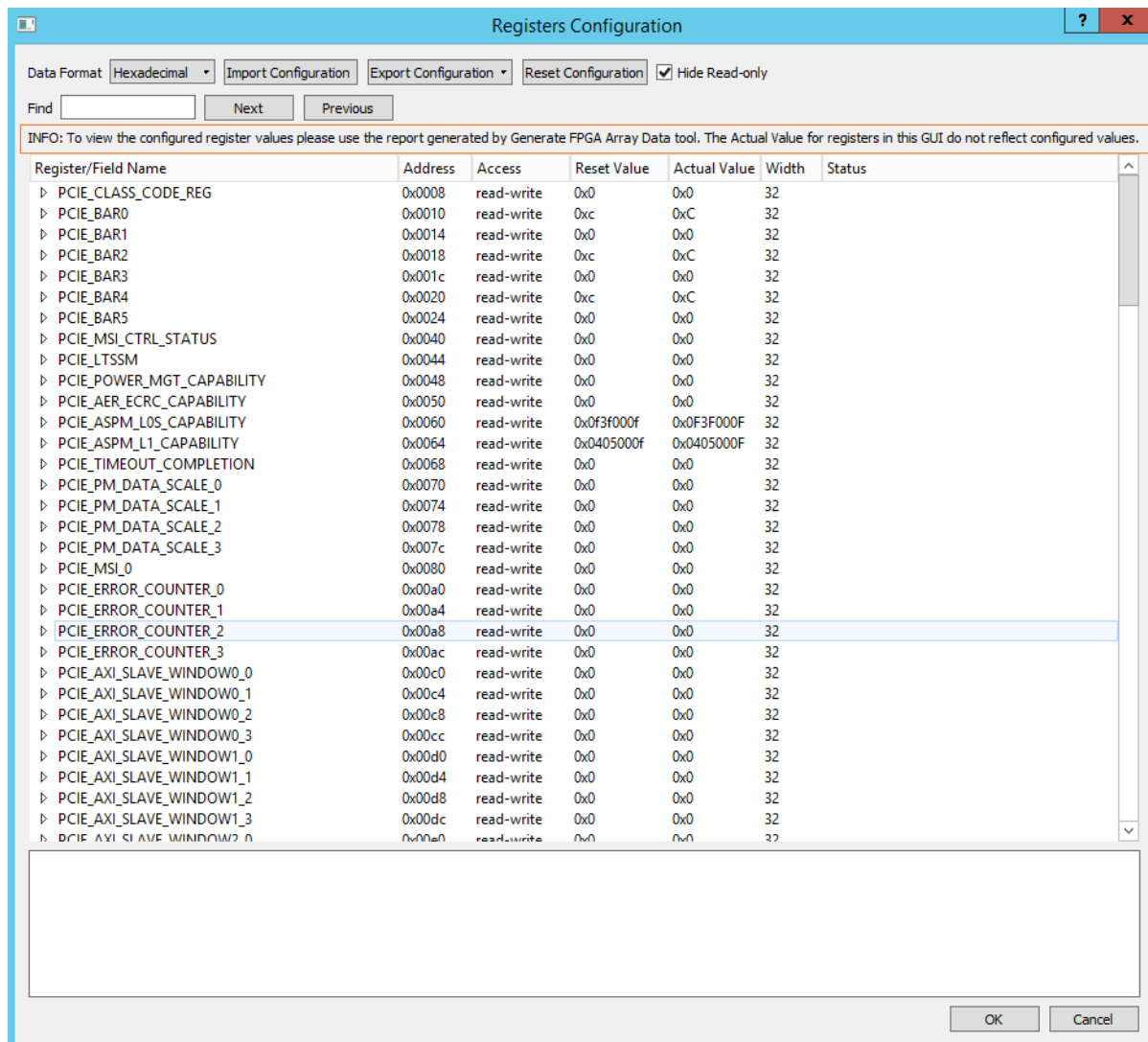
**Note:** LANE<n> denotes the lane number where <n> can be 0, 1, 2, or 3.

### 3.3.14 SerDes Edit Registers

The SerDes configurator includes an Edit Registers GUI. Users can set the SerDes control registers to non-standard values. As shown in the following figure, the "Edit Registers" GUI does not reflect the actual settings. The user needs to have a detailed understanding about register map before using the Edit Register tool. Users should rely on the pre-defined register values that are generated by Libero SoC and should only alter the values when needed.

Libero version 12.2 release and beyond provides a report that is created when 'Generate FPGA Array Data' is performed. The report can be used to view the Libero configured register values including those SerDes registers which have been manually edited in the "Edit Registers" GUI. The report contains detail about all the SerDes, DDR, and CCC blocks in the user design.

**Figure 78 • SerDes Edit Register GUI**



### 3.3.14.1 Data Rate below 1 Gbps

The native speed of the RTG4 SerDes is between 1 through 3.125 Gbps. Using oversampling, each data bit is sampled in multiple clock cycles before it is transmitted. For example, to transmit a 400 Mbps data rate over a 1.2 Gbps serial link, each bit can be sampled three times and spread over three clock cycles for both transmitting and receiving data, called 3x oversampling. Using this technique, lower data rates can be transmitted, while the SerDes PLL continues to run within its valid operating range (1 Gbps minimum).

### 3.3.14.2 Simulating the SerDes Block in EPCS Mode

The SerDes block, when configured in EPCS mode requires the RTL simulation model which is selectable in the high-speed serial interfaces configurator. The RTL model requires external testbenches to stimulate the EPCS design. Refer to the *SERDESIF BFM Simulation Guide* for details.

### 3.3.14.3 Creating an Application in EPCS Mode

An RTG4 EPCS protocol design can be completed using the SerDes block configurator. Libero promotes the SerDes I/Os to top level and exposes the EPCS parallel interface for each lane into the FPGA fabric; the SerDes block exposes the APB interface to the FPGA fabric. Fabric logic or FPGA IP is required to connect to the EPCS interface.

## 3.4 Register Initialization

The registers contained in the SerDes block require initialization when the device powers-up using data stored in non-volatile storage in the device. This is required for PCIe, XAUI, and EPCS configurations. There are two types of initialization that are used to set the value in the registers; flash bits and APB accessed registers.

#### Flash Bits (Flash)

There are several flash bits that are associated with each SerDes block. These flash bits contain the settings for registers that need to be initialized quickly when the device powers up such as PLL and clock configurations, PCIe configuration space, and resets. The flash bits are set by the Libero configuration GUI based on the user selections, programmed into the device, and are statically set at device power-up.

#### APB

The SerDes block supports an APB slave interface connected to the fabric interface. When the Libero software assembles the SerDes block supporting modules, the APB needs to be connected to the module's APB master port. After the device powers up the supporting modules initialize the necessary registers in the SerDes block that use the APB interface.

It is possible that the APB initialization overwrites the registers that have been initialized by the flash bits, since APB is written after the flash bit value has been loaded.

#### Fixed

These registers are read-only registers and their values are fixed based on the implementation of the device.

### 3.4.1 SerDes Initialization

The RTG4 SerDes block requires the user to include a SerDes Initialization solution for the purpose of programming the SerDes configuration register values. One solution is to build the configuration and initialization circuitry in SmartDesign for the SerDes block. This initialization circuitry consists of:

- CoreABC soft IP core
- CoreAPB3 bus soft IP Core

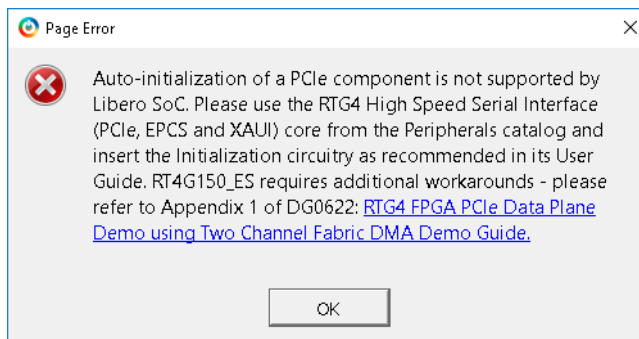
The RTG4 also has SerDes block cores with built-in initialization. These cores include initialization state machine. Upon power-up or device reset, the SerDes block is initialized with the user configurations.

To build the initialization circuitry, see *RTG4 High Speed Serial Interface (PCIe, EPCS and XAUI) Configuration User Guide*. Automatic initialization can be done for XAUI and EPCS modes by using the HSS Interface Core with Initialization. For PCIe, the standard HSS Interface core without initialization must be selected and the user can add the PCIe initialization circuitry. To add this PCIe initialization

circuitry, see *RTG4 High Speed Serial Interface (PCIe, EPCS and XAUI) Configuration User Guide*. Even though the standard HSS Interface core without initialization is used for PCIe applications, it still generates the required register initialization code to use with the CoreABC instance in the PCIe initialization subsystem. For more information, see *RTG4 High Speed Serial Interface (PCIe, EPCS and XAUI) Configuration User Guide*. This initialization program for CoreABC is located in the Libero project's component/work/<HSS\_component\_name>/<HSS\_component\_name\_#> sub-folder.

**Note:** If the user attempts to generate the HSS Interface core with automatic initialization for a PCIe application, Libero generates a pop-up window, as shown in the following figure, re-directing the user to the standard HSS Interface core. For example of the initialization subsystem, see *DG0622: RTG4 FPGA PCIe Data Plane Demo using Two Channel Fabric DMA Demo Guide*. DG0622 also contains a section called "Appendix 3 - Known Issues" describing the implementation of PCIe reset logic to support a hot reset that correctly resets and re-initializes the SerDes block and AXI interface.

**Figure 79 • PCIe Error Message**

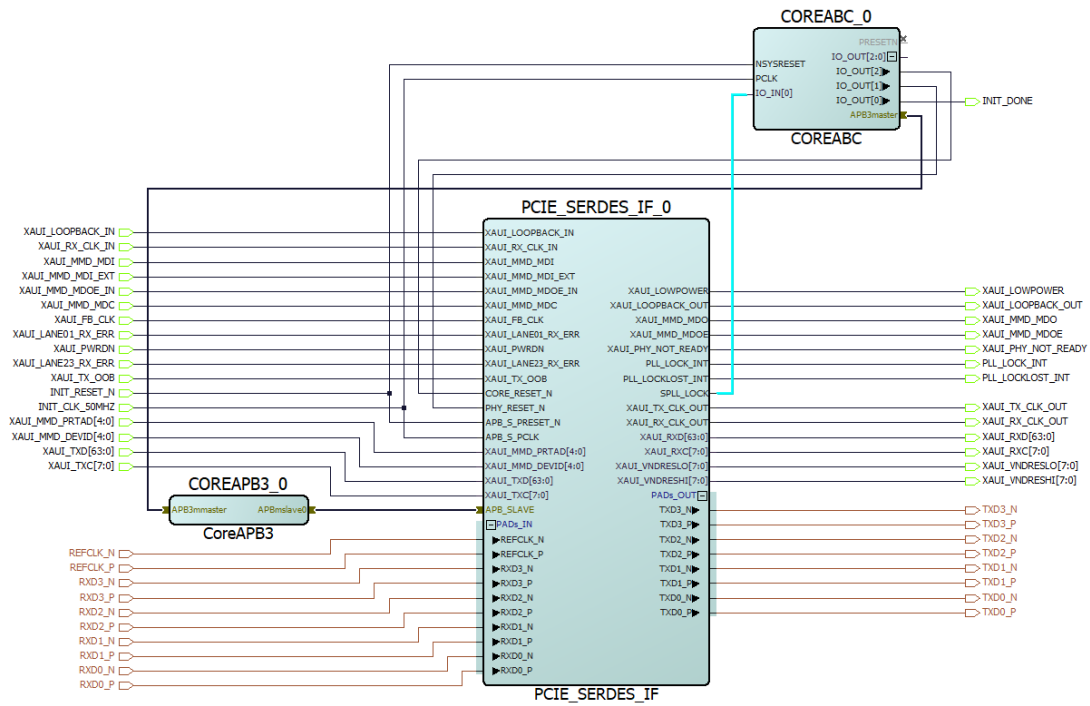


### 3.4.2 Calibration Initialization

Libero SoC v12.4 includes enhanced CoreABC initialization that adds SPLN calibration to mitigate the lock stability dependence on junction temperature, as described in *CN19009*, *CN19009A*, and *CN19009B*. For XAUI applications, the HSS interface core with automatic initialization can be used. In this case, the user can upgrade to the latest version of the HSS interface core with automatic initialization and regenerate the core to automatically add SPLN calibration to the core. For PCIe applications, the updated CoreABC initialization program must be manually pasted into the CoreABC instance used in the user-constructed PCIe initialization circuitry described in the preceding paragraphs.

CoreABC initialization code controls the appropriate sequence to ensure the proper calibration of the SPLN at initialization time. The CoreABC does the following:

- Performs APB writes to set the SPLN VCO into high frequency range.
- Issues a SPLN reset which allows the VCO to calibrate a high gain setting.
- Performs APB writes to restore user desired frequency.
- Waits for SPLN to Lock via a CoreABC input connected to the SerDes block SPLN Lock output. See *Figure 80*, page 131.
- Resumes initialization of the SerDes registers per the user settings

**Figure 80 • SPLL Lock Connection**

The delays built into the updated code requires the frequency of APB clock for CoreABC and the SerDes configuration interface to 50 MHz or slower. A slower clock increases the time it takes to complete the SPLL calibration. The system requirements for SerDes initialization time should be considered if using a slower APB clock. The following figure shows an example of CoreABC program code that is required to match APB clock that governs the speed of initialization.

**Note:** For robust operation, the user should synchronize the FCCC lock signal using the INIT\_CLK\_50MHz before driving INIT\_RESET\_N.

**Figure 81 • CoreABC Code**

```

Configuring SERDES_IF_INIT_C0_COREABC_0 (COREABC 3.6.100)
Parameters Program Analysis
// -----
// CoreABC SERDES Initialization Sequence
// -----

// Allow time for the APB interface to be ready after reset
NOP
NOP
NOP

// SYSTEM_SERDES_APB_SOFT_RSTN_SEL
APBWRIT DAT 0 0x206c 0x0

// De-assert PHY_RESET_N
IOVRT DAT 0x2

// actual_ref_div = 4
// actual_fb_div = 4
// actual_vco_div = 4
// actual_vco_freq = 625
// highvco_ref_div = 4
// highvco_fb_div = 3
// highvco_vco_div = 8
// highvco_vco_freq= 937.5

// PLL_CONFIG_LOW for high vco configuration
APBWRIT DAT 0 0x2000 0x30083

// Assert SPLL POWERDOWN (PLL_CONFIG_HIGH:PLL_PD=1)
APBWRIT DAT 0 0x2004 0xd02e3

// SPLL PLL_PD pulse time
// Using 2us settling time @ 50MHz
// - #cycles = 2*1000/20=100
// - loop value = #cycles / 6 (2 instructions)
LOADZ 0x10
$WaitpdSettling
DECZ
JUMP IFNOT ZZERO $WaitpdSettling

// De-assert SPLL POWERDOWN (PLL_CONFIG_HIGH:PLL_PD=0)
APBWRIT DAT 0 0x2004 0xc02e3

// SPLL high vco settling time
// Using 150us settling time @ 50MHz
// - #cycles = 150*1000/20=7500
// - loop value = #cycles / 6 (2 instructions)
LOADZ 0x4e2
$WaitHighVCOSettling
DECZ
JUMP IFNOT ZZERO $WaitHighVCOSettling

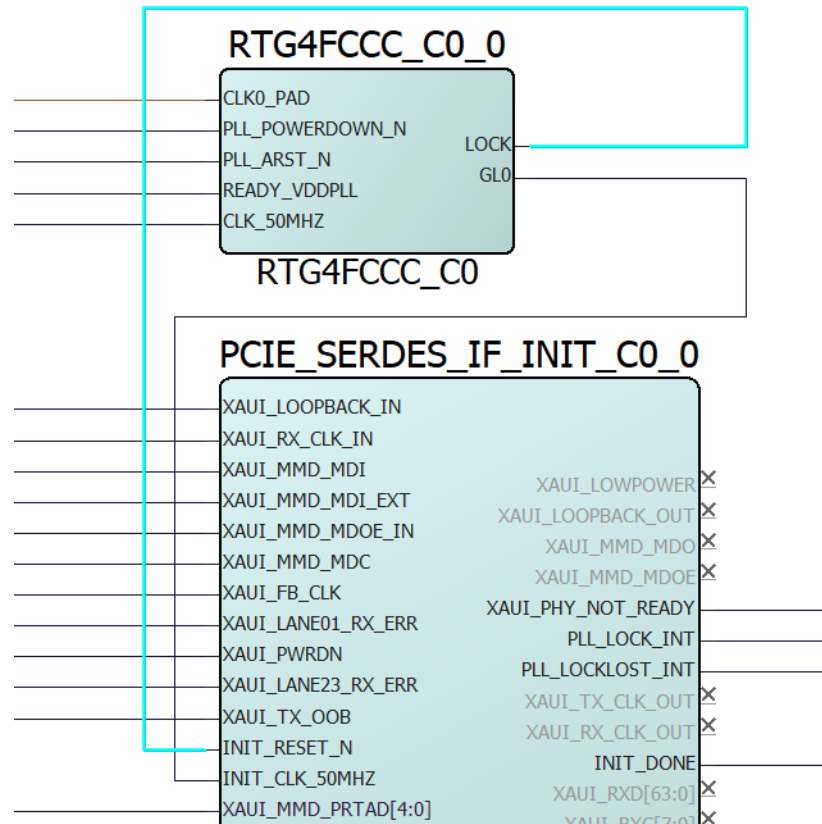
// PLL_CONFIG_LOW for actual configuration
APBWRIT DAT 0 0x2000 0x200c3

// SPLL actual vco loss lock after reconfiguration
// Using 5us settling time @ 50MHz
// - #cycles = 5*1000/20=250
// - loop value = #cycles / 6 (2 instructions)
LOADZ 0x29
$WaitActualVCOLockLoss
DECZ

```

If the reference clock (CLK\_BASE) to the SPLL is sourced from an RTG4 FCCC with Enhanced PLL Calibration, then the user must stagger the release of the SerDes Initialization INIT\_RESET\_N until the fabric PLL lock has been acquired. This will prevent the SPLL calibration and SerDes initialization from proceeding before the SPLL reference clock is stable.

**Figure 82 • FCCC Lock**



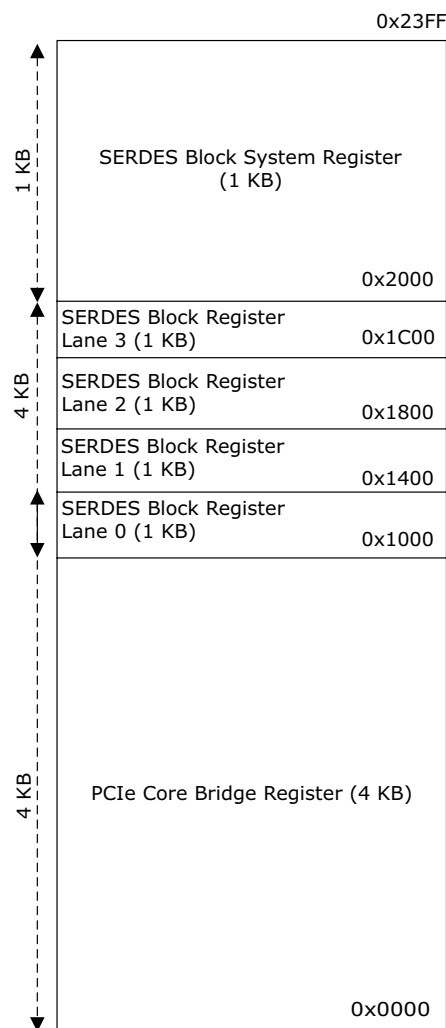


## 4 SerDes Block Register Access Map

### 4.1 Configuration of SerDes Block

The SerDes block contains a large number of internal registers required to properly configure the SerDes block module. The register settings provide initial programming at power-up and most portions of the SerDes block can be dynamically reconfigured while in operation. A SerDes block APB configuration interface accessed through the FPGA fabric provides the resources to allow the programming capabilities. The RTG4 uses a FPGA module to initialize peripherals and access the system controller. An initialization module provides connectivity to the APB bus matrix, allowing similar initialization of SerDes block using this FPGA IP module in the fabric. Initialization module supports the SerDes block peripheral using the Libero SoC software to provide and program the customized features.

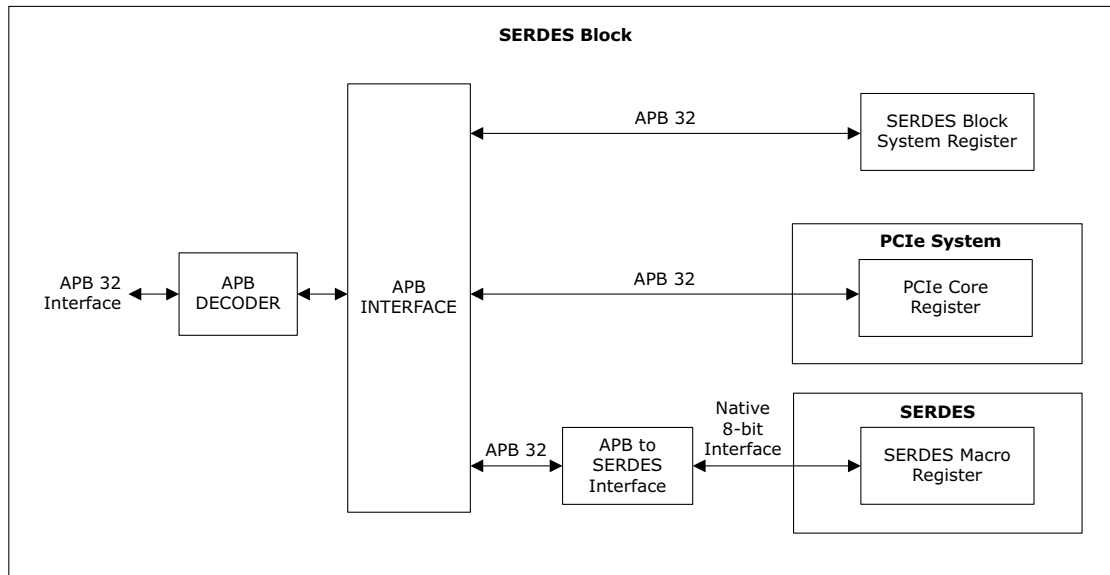
**Figure 83 • SerDes Block Memory Map**



The SerDes block has three sets of configuration and status registers. Configuration of the top-level functionality of the PCIe core, XAUI block, and SerDes block is also done through these registers. The three regions of configuration and status registers are described in [SerDes Block System Register](#), page 135.

The following figure shows the APB implementation of three sets of configuration and status registers. The APB decoder is an interface for the FPGA fabric to access these registers.

**Figure 84 • Address Decoder Logic Block Diagram**



### 4.1.1 SerDes Block System Register

The SerDes block system register controls the SerDes block module for single protocol or multi-protocol support implementation. It occupies 1 KB of the configuration memory map. The physical offset location of the SerDes block system register is 0x2000-0x23FF from the SerDes block subsystem memory map. These registers can be accessed through the 32-bit APB interface and the default values of these registers can be configured using the Libero SoC software. These flash bits have the settings for registers that require to be initialized quickly when the device powers up such as PLL and clock configurations, PCIe configuration space, and resets. The flash bits are set by the Libero configuration GUI based on the selections programmed into the device, and are statically set at device power-up. However, if required, the SerDes block system registers can be updated through the 32-bit APB interface.

### 4.1.2 Single-Event Upset Immunity

SerDes block has system registers and PCIe Core Bridge registers used for controlling the portions implemented with single-event upset (SEU) protection. These registers are protected from corrupted configuration causing a device malfunction. There are some registers, however, that have not been SEU protected within the SerDes block. The SerDes block registers within the PMA are not SEU protected. These registers are defined in the PMA macro and are susceptible to soft error hits.

**Table 164 • SerDes Block System Registers<sup>1</sup>**

Register Name	Address Offset	Register Type	Description
SYSTEM_SER_PLL_CONFIG_LOW (0x2000)	0x00	R/W	Sets SerDes PLL (SPLL) configuration bits (LSBs), see <a href="#">Table 165</a> , page 138.
SYSTEM_SER_PLL_CONFIG_HIGH (0x2004)	0x04	R/W	Sets SPLL configuration bits (MSBs), see <a href="#">Table 166</a> , page 138.
SYSTEM_SER_SOFT_RESET (0x2008)	0x08	R/W	PCIe controller, XAUI, and SerDes lanes soft RESET, see <a href="#">Table 167</a> , page 140.
SYSTEM_SER_INTERRUPT_ENABLE (0x200C)	0x0C	R/W	SPLL lock interrupt enable, see <a href="#">Table 168</a> , page 140.
SYSTEM_CONFIG_AXI_AHB_BRIDGE (0x2010)	0x10	R/W	Defines whether AXI3/AHB master interface is implemented on the master interface to fabric, see <a href="#">Table 169</a> , page 140.
SYSTEM_CONFIG_ECC_INTR_ENABLE (0x2014)	0x14	R/W	Sets ECC and ECC interrupt enable for PCIe memories, see <a href="#">Table 170</a> , page 141.
SYSTEM_CONFIG_PCIE_PM (0x2020)	0x 20	R/W	Used to inform the configuration space, the slot power, PHY reference clock, power mode and soon, see <a href="#">Table 171</a> , page 141.
SYSTEM_CONFIG_PHY_MODE_0 (0x2024)	0x24	R/W	Selects the protocol default settings of the PHY, see <a href="#">Table 172</a> , page 142.
SYSTEM_CONFIG_PHY_MODE_1 (0x2028)	0x 28	R/W	Selects PCS mode, link to lane settings, see <a href="#">Table 173</a> , page 142.
SYSTEM_CONFIG_PHY_MODE_2 (0x202C)	0x2C	R/W	Sets the equalization calibration performed by the PMA control logic of the lane or use the calibration result of adjacent lane, see <a href="#">Table 174</a> , page 143.
SYSTEM_CONFIG_PCIE_0 (0x2030)	0x30	R/W	Defines PCIe vendor ID and device ID for PCIe identification registers, see <a href="#">Table 175</a> , page 143.
SYSTEM_CONFIG_PCIE_1 (0x2034)	0x34	R/W	Defines PCIe subsystem vendor ID and subsystem device ID for PCIe identification registers, see <a href="#">Table 176</a> , page 143.
SYSTEM_CONFIG_PCIE_2 (0x2038)	0x38	R/W	Defines PCIe subsystem revision ID and classcode, see <a href="#">Table 177</a> , page 143.
SYSTEM_CONFIG_PCIE_3 (0x203C)	0x3C	R/W	Sets PCIe link speed, see <a href="#">Table 178</a> , page 143.
SYSTEM_CONFIG_BAR_SIZE_0_1 (0x2040)	0x40	R/W	Sets BAR0 and BAR1 of PCIe core registermap, see <a href="#">Table 179</a> , page 144.
SYSTEM_CONFIG_BAR_SIZE_2_3 (0x2044)	0x44	R/W	Sets BAR2 and BAR3 of PCIe core registermap, see <a href="#">Table 180</a> , page 144.
SYSTEM_CONFIG_BAR_SIZE_4_5 (0x2048)	0x48	R/W	Sets BAR4 and BAR5 of PCIe core registermap, see <a href="#">Table 181</a> , page 145.
SYSTEM_SER_CLK_STATUS (0x204C)	0x4C	R/O	This register describes SPLL lock information, see <a href="#">Table 182</a> , page 146.
SYSTEM_SER_INTERRUPT (0x2058)	0x58	R/O	SPLL/FPLL lock interrupt, see <a href="#">Table 183</a> , page 146.
SYSTEM_SERDES_INTR_STATUS Register (0x205C)	0x5C	R/O	SECEDED interrupt status for PCIe memories, see <a href="#">Table 184</a> , page 146.

**Table 164 • SerDes Block System Registers<sup>1</sup> (continued)**

Register Name	Address Offset	Register Type	Description
SYSTEM_REFCLK_SEL (0x2064)	0x64	R/W	Selects reference clock for the four lanes of PMA, see Table 185, page 146.
SYSTEM_PCLK_SEL Register (0x2068)	0x68	R/W	Selects PCIe core clock, see Table 186, page 147.
SERDES_APB_SOFT_RSTN_SEL (0x206C)	0x6C	R/W	Selects EPCS reset register from fabric, see Table 187, page 147.
SYSTEM_SERDES_TEST_OUT (0x2074)	0x74	R/O	Status Test output of PCIe PHY, see Table 188, page 147.
SYSTEM_FABRIC_CLK_ENA (0x207C)	0x7C	R/W	Select which clocks used for GB_CLK outputs of SERDES, see Table 189, page 148.
SYSTEM_CONF_AXI_MSTR_WND W_0 (0x2084)	0x84	R/W	PCIe AXI3-master window0 configuration register – 0, see Table 190, page 148.
SYSTEM_CONF_AXI_MSTR_WND W_1 (0x2088)	0x88	R/W	PCIe AXI3-master window0 configuration register – 2, see Table 191, page 148.
SYSTEM_CONF_AXI_MSTR_WND W_2 (0x208C)	0x8C	R/W	PCIe AXI3-master window0 configuration register – 2, see Table 192, page 149.
SYSTEM_CONF_AXI_MSTR_WND W_3 (0x2090)	0x90	R/W	PCIe AXI3-master window0 configuration register – 3, see Table 193, page 149.
SYSTEM_CONF_AXI_SLV_WNDW _0 (0x2094)	0x94	R/W	PCIe AXI3-slave window0 configuration register –0, see Table 194, page 149.
SYSTEM_CONF_AXI_SLV_WNDW _1 (0x2098)	0x98	R/W	PCIe AXI3-slave window0 configuration register – 1, see Table 195, page 149.
SYSTEM_CONF_AXI_SLV_WNDW _2 (0x209C)	0x9C	R/W	PCIe AXI3-slave window0 configuration register –2, see Table 196, page 149.
SYSTEM_CONF_AXI_SLV_WNDW _3 (0x20A0)	0xA0	R/W	PCIe AXI3-slave window0 configuration register –4, see Table 197, page 149.
SYSTEM_DESKEW_CONFIG (0x20A4)	0xA4	R/W	PLL REF clock DESKEW register, see Table 198, page 149.
SYSTEM_IDDQ(0x20b0)	0xb0	R/W	Puts the PMA in the IDDQ mode. IDDQ mode minimizes power consumption when the PMA is not used. Bit0: Lane 0, Bit1: Lane 1, Bit2: Lane2, Bit 3: Lane 3.
SYSTEM_ADVCONFIG (0x20b4)	0xb4	R/W	Advanced mode setting for PCI controller, see Table 199, page 150.
SYSTEM_ECC_ERR_INJECT (0x20bC)	0xbC	W1P	Writing "1" to these registers will pulse and inject an ECC error.
SYSTEM_REFCLK_MSIO_CONFIG (0x20c4)	0xc4	R/W	Configure multi-standard Reference clock, see Table 201, page 151.
SYSTEM_ENHANCEMENT (0x20c8)	0xc8	R/W	Enhancement setting. This is used in conjunction with the RX_SLIP word alignment operation, see Table 202, page 152.
SYSTEM_TXFWF_CONFIG (0x20cc)	0xcc	R/W	TX Fly-Wheel FIFO Configuration, see Table 203, page 152.
SYSTEM_RXFWF_CONFIG (0x20d0)	0xd0	R/W	RX Fly-Wheel FIFO Configuration, see Table 204, page 153.

**Table 164 • SerDes Block System Registers<sup>1</sup> (continued)**

Register Name	Address Offset	Register Type	Description
SYSTEM_FWF_STATUS(0x20d4)	0xd4	SW1C	Tx and Rx Fly-wheel FIFO overflow and underflow flags.

1. Refer to the individual register description for the reset value. Reset values are related to default hardware values (that is, un-programmed device). Libero autonomously writes registers defaults based on user inputs at design creation.
- R/W: Read and write allowed.
  - R/O: 0 Read only.
  - SW1C: Write 1 clears value
  - Undefined register addresses are Reserved.

Reset Values in the following table are either the un-programmed default value of the device or the reset value as defined by Libero for initialization.

**Table 165 • SYSTEM\_SER\_PLL\_CONFIG\_LOW (0x2000)**

Bit Number	Name	Reset Value	Description
18:16	PLL_OUTPUT_DIVISOR	0x1	These bits set SPLL output divider value: 000: ÷1 001: ÷2 010: ÷4 011: ÷8 100: ÷16 101: ÷32
13:6	PLL_FEEDBACK_DIVISOR	0x2	These bits set SPLL feedback divider value (SSE =0) (binary value + 1) 0000000000: ÷1 0000000001: ÷2 0000000010: ÷3 ... 1111111111: ÷1,025
5:0	PLL_REF_DIVISOR	0x2	These bits set SPLL reference divider value (binary value + 1): 000000: ÷1 000001: ÷2 000010: ÷3 ... 111111: ÷65 Both REFCK and post-divide REFCK must be within the range specified in the PLL datasheet.

**Table 166 • SYSTEM\_SER\_PLL\_CONFIG\_HIGH (0x2004)**

Bit Number	Name	Reset Value	Description
16	PLL_PD	0x0	A power-down (PD) register is provided for the lowest quiescent current. When PD is asserted, the PLL powers down and outputs are low. PD has precedence over all other functions.

**Table 166 • SYSTEM\_SER\_PLL\_CONFIG\_HIGH (0x2004) (continued)**

Bit Number	Name	Reset Value	Description
15	PLL_FSE	0x0	<p>This register chooses between internal and external input paths:            0: Feedback (FB) pin input            1: RESERVED</p> <p>FB must be tied off (high or low) and not left floating when FSE is high. FB must connect directly or through the clock tree to PLLOUT when FSE is low. SSE is ineffective when FSE = 0. If the FACC source multiplexer is configured to select a clock other than the PLL output clock, then the fddr_pll_fse register must be set to 1, when the PLL is powered up.</p>
12	PLL_BYPASS	0x1	<p>A bypass register is provided which powers down the PLL core and also bypasses it as the PLLOUT tracks REFCK. Bypass has precedence over reset. Microsemi recommends that either bypass or reset are asserted until all configuration controls are set in the desired working value; the power supply and reference clocks are stable within operating range, and the feedback path is functional. Either bypass or reset is used for power- down IDDQ testing.</p>
11	PLL_RESET	0x1	PLL reset (asserted high).
10:7	PLL_LOCKCNT	0xF	<p>These bits contain lock counter value (<math>2^{\text{binary value} + 5}</math>):            0000: 32            0001: 64            ...            1111: 1048576</p> <p>The above mentioned lock counter values represent the number of reference cycles which occur before the lock is asserted or detected.</p>
6:4	PLL_LOCKWIN	0x0	<p>These bits contain phase error window for lock assertion as a fraction of divided reference period:            000: 500ppm            001: 1000ppm            010: 1500ppm            011: 2000ppm            100: 3000ppm            101: 4000ppm            110: 6000ppm            111: 16000ppm</p> <p>Values are at typical process, voltage, and temperature (PVT) only and are not PVT compensated.</p>
3:0	PLL_FILTER_RANGE	0x4	<p>These bits contain PLL filter range (frequency range after PLL reference input dividers to the phase-frequencydetector):            000: BYPASS            001: 10–16.8 MHz            010: 16.8–26.8 MHz            011: 26.8–43 MHz            100: 43–69 MHz            101: 69–110 MHz            110: 110–175 MHz            111: 75–200 MHz</p>

**Note:** All the registers are 32-bit. Bits not shown in the table are reserved.

**Table 167 • SYSTEM\_SER\_SOFT\_RESET (0x2008)**

Bit Number	Name	Reset Value	Description
5	SERDES_LANE3_SOFTRESET	0x1	SerDes lane3 soft reset. When in EPCS mode, resets lane 3. When in PCI mode, resets lane 3 and other lanes in the same link. Active low.
4	SERDES_LANE2_SOFTRESET	0x1	SerDes lane2 soft reset. When in EPCS mode, resets lane 2. When in PCI mode, resets lane 2 and other lanes in the same link. Active low.
3	SERDES_LANE1_SOFTRESET	0x1	SerDes lane1 soft reset. When in EPCS mode, resets lane 1. When in PCI mode, resets lane 1 and other lanes in the same link. Active low.
2	SERDES_LANE0_SOFTRESET	0x1	SerDes lane0 soft reset. When in EPCS mode, resets lane 0. When in PCI mode, resets lane 0 and other lanes in the same link. Active low.
1	XAUI_CTLR_SOFTRESET	0x1	XAUI controller soft reset, active low.
0	PCIE_CTLR_SOFTRESET	0x1	PCIE controller soft reset, active low.

**Note:** All the registers are 32-bit. Bits not shown in the table are reserved.

**Note:** Toggling any of the preceding SOFTRESETS restarts PMA calibration.

**Table 168 • SYSTEM\_SER\_INTERRUPT\_ENABLE (0x200C)**

Bit Number	Name	Reset Value	Description
3	FPLL_LOCKLOST_INT_ENABLE	0x0	This bit sets FPLL lock lost interrupt output enable.
2	FPLL_LOCK_INT_ENABLE	0x0	This bit sets FPLL lock interrupt output enable.
1	SPLL_LOCKLOST_INT_ENABLE	0x0	This bit sets SerDes PLL lock lost interrupt output enable.
0	SPLL_LOCK_INT_ENABLE	0x0	This bit sets SerDes PLL lock interrupt output enable.

**Table 169 • SYSTEM\_CONFIG\_AXI\_AHB\_BRIDGE (0x2010)**

Bit Number	Name	Reset Value	Description
1	CFGR_AXI_AHB_MASTER	0x1	Defines whether AXI3/AHB slave interface is implemented on the master interface to fabric. 0: AHB, 32-bit AHB slave implemented in fabric 1: AXI3, 64-bit AXI3 slave implemented in fabric
0	CFGR_AXI_AHB_SLAVE	0x1	Defines whether AXI3/AHB master interface is implemented on the slave interface to fabric. 0: AHB, 32-bit AHB master implemented in fabric 1: AXI3, 64-bit AXI3 master implemented in fabric

**Table 170 • SYSTEM\_CONFIG\_ECC\_INTR\_ENABLE (0x2014)**

Bit Number	Name	Reset Value	Description
7:4	CFGR_PCIE_ECC_INTR_EN	0x7	ECC interrupt enable for PCIe Memories Bit-0: Replay RAM - 1'b1 = ECC interrupt enabled, 1'b0 = disabled Bit-1: Receive RAM - 1'b1 = ECC interrupt enabled, 1'b0 = disabled Bit-2: Transmit RAM - 1'b1 = ECC interrupt enabled, 1'b0 = disabled Bit-3: 0: Generate interrupts for single and two bit errors 1: Generate interrupts only for 2-bit errors
3:0	CFGR_PCIE_ECC_EN	0x7	ECC enable for PCIe Memories, 1'b1 = enabled, 1'b0 = disabled Bit-0: Replay RAM ECC enable Bit-1: Receive RAM ECC enable Bit-2: Transmit RAM ECC enable

**Table 171 • SYSTEM\_CONFIG\_PCIE\_PM (0x2020)**

Bit Number	Name	Reset Value	Description
3	CFGR_TX_SWING	0x0	Transmit swing: This register is per-link, which is generated by each link PCIe. The PCS logic performs the internal mapping of link to lanes.  <b>Note:</b> This is only for PCIe Gen2 controller, not for PCIe GEN1 controller.
2	CFGR_L2_P2_ENABLE	0x0	L2/P2 enable. 1'b1: Enable L2/P2 (Default-L2P2-Enabled) 1'b0: Disable L2/P2 If L2/P2 is enabled, cfgr_pm_auxpwr must also be enabled.
1	CFGR_PM_AUX_PWR	0x0	Slot auxiliary power: This register specifies whether the device uses the slot, auxiliary power source. This is used only if the core supports D3 cold. 1'b1: Auxiliary power source available. Default L2P2-Enabled. 1'b0: Auxiliary power source unavailable.
0	CFGR_SLOT_CONFIG	0x0	Slot clock configuration: This register is used only to inform the configuration space, if the reference clock of the PHY is same as that of the slot. 0: Independent clock 1: Slot clock This is synchronous to CLK.

**Note:** All registers are 32-bit. Bits not shown in the table are reserved.



**Table 172 • SYSTEM\_CONFIG\_PHY\_MODE\_0(0x2024)**

Bit Number	Name	Reset Value	Description
15:0	CONFIG_PHY_MODE	0x0	<p>Selects the protocol default settings of the PHY, which sets the reset value of the registers space for each lane. For instance, the following mapping is associated to a four lane PHY:</p> <p>phy_mode[3:0]: Mode associated to lane0            phy_mode[7:4]: Mode associated to lane1            phy_mode[11:8]: Mode associated to lane2            phy_mode[15:12]: Mode associated to lane3</p> <p>PHY_MODE settings:            4'b0000 - PCIE mode            4'b0001 - XAUI mode            4'b0010 - Reserved            4'b0011 - Reserved            4'b0100 - Reserved            4'b0101 - EPCS (custom) mode            4'b1111 - SerDes PHY lane is off</p>

**Table 173 • SYSTEM\_CONFIG\_PHY\_MODE\_1(0x2028)**

Bit Number	Name	Reset Value	Description
11:8	CONFIG_REG_LANE_SEL	0xF	<p>Lane select: This register defines lanes that are accessed and must be one-hot encoded for read transaction. For write transaction, one or several lanes can be written at the same time when several bits are asserted. Each bit represents a SerDes Lane. The default allows all lanes to be selected to be written. A 0b deselects the registers for a specific lane to not be written.</p>
7:4	CONFIG_LINKK2LANE	0xF	<p>This register is used in PCIe mode in order to select the association of lane to link and must be one-hot encoded (each lane can be associated only to one link).            For example, a four lane PHY, which can be configured in 1 or 2 link might have pipe_lk2ln[3:0]—lane associated to link0 and pipe_lk2ln[7:4]—lane associated to link1</p> <p><b>Note:</b> This signal must be static at power-up or stable before reset de-assertion.</p>
3:0	CONFIG_EPCS_SEL	0x0	<p>For each lane, one bit of this signal defines whether the external PCS interface or the PCIe PCS is enabled:            0b: PCIe mode            1b: External PCS mode</p> <p>For instance, the mapping associated to a four lane PHY is:            epcs_sel[0]: External PCS selection associated to lane0            epcs_sel[1]: External PCS selection associated to lane1            epcs_sel[2]: External PCS selection associated to lane2            epcs_sel[3]: External PCS selection associated to lane3</p>

**Table 174 • SYSTEM\_CONFIG\_PHY\_MODE\_2 (0x202C)**

Bit Number	Name	Reset Value	Description
7:0	CONFIG_REXT_SEL	0x0	For each lane, 2 bits of this signal select whether the Tx, Rx, and Rx equalization calibration are performed by the PMA control logic of the lane or use the calibration result of adjacent lane (upper or lower lanes): 00b: perform calibration using the lane calibration algorithm, which requires the Rext resistor to be present on board 01b: use calibration result of the lower lane 10b: use calibration result of the upper lane 11b: reserved

**Note:** All registers are 32-bit. Bits not shown in the table are reserved.

**Table 175 • SYSTEM\_CONFIG\_PCIE\_0 (0x2030)**

Bit Number	Name	Reset Value	Description
31:16	PCIE_DEVICE_ID	0x0	Specifies hardwired settings for PCIe identification registers: Defines PCIe device ID.
15:0	PCIE_VENDOR_ID	0x0	Specifies hardwired settings for PCIe identification registers: Defines PCIe vendor ID.

**Table 176 • SYSTEM\_CONFIG\_PCIE\_1 (0x2034)**

Bit Number	Name	Reset Value	Description
31:16	PCIE_SUB_DEVICE_ID	0x0	Specifies hardwired settings for PCIe identification registers: Defines PCIe subsystem device ID.
15:0	PCIE_SUB_VENDOR_ID	0x0	Specifies hardwired settings for PCIe identification registers: Defines PCIe subsystem vendor ID.

**Table 177 • SYSTEM\_CONFIG\_PCIE\_2 (0x2038)**

Bit Number	Name	Reset Value	Description
31:16	PCIE_CLASS_CODE	0x0	Specifies hardwired settings for PCIe identification registers: Defines PCIe class code.
15:0	PCIE_REV_ID	0x0	Specifies hardwired settings for PCIe identification registers: Defines PCIe revision ID.

**Note:** All registers are 32-bit. Bits not shown in the table are reserved.

**Table 178 • SYSTEM\_CONFIG\_PCIE\_3 (0x203C)**

Bit Number	Name	Reset Value	Description
5:2	K_BRIDGE_SPEC_REV	0x0	Reserved
1	K_BRIDGE_EMPH	0x0	Reserved
0	K_BRIDGE_SPEED	0x0	Reserved

**Table 179 • SYSTEM\_CONFIG\_BAR\_SIZE\_0\_1(0x2040)**

Bit Number	Name	Reset Value	Description
17:13	CONFIG_BAR_SIZE_1	0x0	These bits set the size of the BAR1 memory. For example, 32-bit BAR: CONFIG_BAR_SIZE_1 - 5'd22 translates to BAR0 - (2 MB) "1111_1111_1110_0000_0000_0000_CONFIG_BAR_CONTROL_1"
12:9	CONFIG_BAR_CONTROL_1	0x0	LSB bits of BAR1 register in PCIe core register map Bit0: Memory/IO type indicator Bit[2:1]: Size of memory, 00-32-bit memory, 10 - 64-bit memory Bit3: Prefetchable/non-prefetchable memory
8:4	CONFIG_BAR_SIZE_0	0x0	These bits set the size of the BAR0 memory. For example, 32-bit BAR: CONFIG_BAR_SIZE_0 - 5'd21 translates to BAR0 - (1 MB) "1111_1111_1111_0000_0000_0000_CONFIG_BAR_CONTROL_0"
3:0	CONFIG_BAR_CONTROL_0	0x0	LSB bits of BAR 0 register in PCIe core register map Bit0: Memory/IO type indicator Bit[2:1]: Size of memory, 00-32-bit memory, 10 - 64-bit memory Bit3: Prefetchable/non-prefetchable memory

**Note:** All registers are 32-bit. Bits not shown in the table are reserved.

**Table 180 • SYSTEM\_CONFIG\_BAR\_SIZE\_2\_3(0x2044)**

Bit Number	Name	Reset Value	Description
17:13	CONFIG_BAR_SIZE_3	0x0	These bits set the size of the BAR3 memory. For example, 32-bit BAR: CONFIG_BAR_SIZE_3 - 5'd24 translates to BAR3 - (8 MB) "1111_1111_1000_0000_0000_0000_CONFIG_BAR_CONTROL_3"
12:9	CONFIG_BAR_CONTROL_3	0x0	[3:0] LSB bits of BAR 3 register in PCIe core register map Bit0: Memory/IO type indicator Bit[2:1]: Size of memory, 00-32-bit memory, 10-64-bit memory Bit3: Prefetchable/non-prefetchable memory
8:4	CONFIG_BAR_SIZE_2	0x0	These bits set the size of the BAR2 memory. For example, 32-bit BAR: CONFIG_BAR_SIZE_2 - 5'd23 translates to BAR0 - (4 MB) "1111_1111_1100_0000_0000_0000_CONFIG_BAR_CONTROL_2"
3:0	CONFIG_BAR_CONTROL_2	0x0	[3:0] LSB bits of BAR 2 register in PCIe core register map Bit0: Memory/IO type indicator Bit[2:1]: Size of memory, 00-32-bit memory, 10 - 64-bit memory Bit3: Prefetchable/non-prefetchable memory

**Table 181 • SYSTEM\_CONFIG\_BAR\_SIZE\_4\_5(0x2048)**

Bit Number	Name	Reset Value	Description
17:13	CONFIG_BAR_SIZE_5	0x0	These bits set the size of the BAR5 memory. For example, 32-bit BAR: CONFIG_BAR_SIZE_5 - 5'd26 translates to BAR5 - (32 MB) "1111_1110_0000_0000_0000_0000_CONFIG_BAR_CONTROL_5"
12:9	CONFIG_BAR_CONTROL_5	0x0	[3:0] LSB bits of BAR 5 register in PCIe core register map Bit0: Memory/IO type indicator Bit[2:1]: Size of memory, 00 - 32-bit memory, 10 - 64-bit memory Bit3: Prefetchable/non-prefetchable memory
8:4	CONFIG_BAR_SIZE_4	0x0	These bits set the size of the BAR4 memory. For example, 32-bit BAR: CONFIG_BAR_SIZE_4 - 5'd25 translates to BAR4 - (16 MB) "1111_1111_0000_0000_0000_0000_CONFIG_BAR_CONTROL_4"
3:0	CONFIG_BAR_CONTROL_4	0x0	[3:0] LSB bits of BAR 4 register in PCIe core register map Bit0: Memory/IO type indicator Bit[2:1]: Size of memory, 00-32 bit memory, 10 - 64-bit memory Bit3: Prefetchable/non-prefetchable memory

**Note:** All registers are 32-bit. Bits not shown in the table are reserved.

**Note:** CONFIG\_BAR\_CONTROL[3:0] sets the BAR type.

Bit0: Memory Space 0=Memory. 1 = IO

Bits 2: 1= Memory Size, 00=32 bit memory

Bit 3: 1 = Prefetchable, 0=Non-prefetchable memory.

**Note:** CONFIG\_BAR\_SIZE[4:0] sets the BAR size

(Decimal Values)

0: BAR DISSABLED

13: 4KB (min)

14 8KB

15: 16KB

16: 32KB

17: 64KB

18: 128KB

19: 256KB

20: 512KB

21: 1MB

22: 2MB

23: 4MB

24: 8MB

25: 16MB

26: 32MB

27: 64MB

28: 128MB

29: 256MB

30: 512MB

3:1 1GB  
 12: 2GB (max)  
 1-11: ILLEGAL

**Table 182 • SYSTEM\_SER\_CLK\_STATUS (0x204C)**

Bit Number	Name	Reset Value	Description
1	FAB_PLL_LOCK	RO	Fabric PLL lock information, CLK BASE, 1:LOCKED
0	PLL_LOCK	RO	SPLL lock information, 1:LOCKED

**Table 183 • SYSTEM\_SER\_INTERRUPT (0x2058)**

Bit Number	Name	Reset Value	Description
1	PLL_LOCK_INT	SW1C	SPLL/FPLL lock interrupt output
0	PLL_LOCKLOST_INT	SW1C	SPLL/FPLL lock lost interrupt output

**Table 184 • SYSTEM\_SERDES\_INTR\_STATUS Register (0x205C)**

Bit Number	Name	Reset Value	Description
2:0	SERDES_INTR_STATUS	SW1C	ECC interrupt status for PCIe memories Bit 0: Transmit RAM multiple bit error Bit 1: Transmit RAM single bit error Bit 2: Receive RAM multiple bit error Bit 3: Receive RAM single bit error Bit 4: Replay RAM multiple bit error Bit 5: Replay RAM single bit error Write '1' to clear

**Table 185 • SYSTEM\_REFCLK\_SEL (0x2064)**

Bit Number	Name	Reset Value	Description
3:2	LANE23_REFCLK_SEL	0x0	Reference clock selection for Lane 2 and Lane 3 of PMA: 2'b00: Selects refclk_io0 2'b01: Selects refclk_io1 2'b10: Selects ccc_ref_clk 2'b11: Selects fab_ref_clk
1:0	LANE01_REFCLK_SEL	0x0	Reference clock selection for Lane 0 and Lane 1 of PMA: 2'b00: Selects refclk_io0 2'b01: Selects refclk_io1 2'b10: Selects ccc_ref_clk 2'b11: Selects fab_ref_clk

**Table 186 • SYSTEM\_PCLK\_SEL Register(0x2068)**

Bit Number	Name	Reset Value	Description
5:4	PIPE_PCLKIN_LANE23_SEL	0x0	PIPE clock input selection for lane2 and lane3, can be selected from one of pipeclk_out[3:0]: 00: Selects pipeclk_out[0] clock as pipeclk_in for lane2 and lane3. 01: Selects pipeclk_out[1] clock as pipeclk_in for lane2 and lane3. 10: Selects pipeclk_out[2] clock as pipeclk_in for lane2 and lane3. 11: Selects pipeclk_out[3] clock as pipeclk_in for lane2 and lane3.
3:2	PIPE_PCLKIN_LANE01_SEL	0x0	PIPE clock input selection for lane0 and lane1, can be selected from one of pipeclk_out[3:0]: 00: Selects pipeclk_out[0] clock as pipeclk_in for lane0 and lane1. 01: Selects pipeclk_out[1] clock as pipeclk_in for lane0 and lane1. 10: Selects pipeclk_out[2] clock as pipeclk_in for lane0 and lane1. 11: Selects pipeclk_out[3] clock as pipeclk_in for lane0 and lane1.
1:0	PCIE_CORECLK_SEL	0x0	PCIe core clock selection. PCIe core clock can be selected from one of pipeclk_out[3:0]: 00: Selects pipeclk_out[0] clock as PCIe core clock. 01: Selects pipeclk_out[1] clock as PCIe core clock. 10: Selects pipeclk_out[2] clock as PCIe core clock. 11: Selects pipeclk_out[3] clock as PCIe core clock.

**Note:** All registers are 32-bit. Bits not shown in the table are reserved.

**Table 187 • SYSTEM\_EPCS\_RSTN\_SEL(0x206C)**

Bit Number	Name	Reset Value	Description
3:0	FABRIC_EPCS_RSTN_SEL	0x0	EPCS reset signal selection from the fabric. When '1', the APB reset is used for PHY Reset. When '0', the APB reset is not used for PHY Reset. In either case, Power-on Reset and SERDES_LANE*_REG_SOFTRESET_N resets the PHY registers.

**Table 188 • SYSTEM\_SERDES\_TEST\_OUT(0x2074)**

Bit Number	Name	Reset Value	Description
31:0	SERDES_TEST_OUT	0x0	Status TESTOUT output of PCIe PHY. SERDES_TEST_OUT[31:24] - Debug signal for lane3 SERDES_TEST_OUT[23:16] - Debug signal for lane2 SERDES_TEST_OUT[15:8] - Debug signal for lane1 SERDES_TEST_OUT[7:0] - Debug signal for lane0 Bit[0]: Tx PLL reset - Active low Bit[1]: Rx PLL reset - Active low Bit[2]: Activity detected Bit[3]: CDR PLL locked on data Bit[4]: Tx PLL locked Bit[5]: Rx PLL locked Bit[7:6]: reserved

**Note:** All registers are 32-bit. Bits not shown in the table are reserved.

**Table 189 • SYSTEM\_FABRIC\_CLK\_ENA (0x207C)**

Bit Number	Name	Reset Value	Description
3:0	GB_CLK_0_SEL	0x1	Select clocks used for GB_CLK outputs of serial subsystem 4'h0-4'h3 : phy epcs_txclk[0:3] 4'h4-4'h7 : phy epcs_rxclk[0:3] 4'h8 : refclk0 4'h9 : refclk1 4'hA : spll_o_clk 4'hB-4'hD : reserved 4'hE : 1'b0 4'hF : 1'b1
7:4	GB_CLK_1_SEL	0x0	Select clocks used for GB_CLK_1 outputs of serial subsystem 4'h0-4'h3 : phy epcs_txclk[0:3] 4'h4-4'h7 : phy epcs_rxclk[0:3] 4'h8 : refclk0 4'h9 : refclk1 4'hA : spll_o_clk 4'hB-4'hD : reserved 4'hE : 1'b0 4'hF : 1'b1
8	EPCS_0_RX_CLK_ENA	0x0	Enable selected clock to be sent to fabric
9	EPCS_1_RX_CLK_ENA	0x0	Enable selected clock to be sent to fabric
10	EPCS_2_RX_CLK_ENA	0x0	Enable selected clock to be sent to fabric
11	EPCS_3_RX_CLK_ENA	0x0	Enable selected clock to be sent to fabric
12	EPCS_0_TX_CLK_ENA	0x0	Enable selected clock to be sent to fabric
13	EPCS_1_TX_CLK_ENA	0x0	Enable selected clock to be sent to fabric
14	EPCS_2_TX_CLK_ENA	0x0	Enable selected clock to be sent to fabric
15	EPCS_3_TX_CLK_ENA	0x0	Enable selected clock to be sent to fabric

**Table 190 • SYSTEM\_CONF\_AXI\_MSTR\_WNDW\_0(0x2084)**

Bit Number	Name	Reset Value	Description
31:0	CONF_AXI_MSTR_WNDW_0	0x0	PCIe AXI3-master Window0 configuration register – 0

**Note:** All registers are 32-bit. Bits not shown in the table are reserved.

**Table 191 • SYSTEM\_CONF\_AXI\_MSTR\_WNDW\_1 (0x2088)**

Bit Number	Name	Reset Value	Description
31:0	CONF_AXI_MSTR_WNDW_1	0x0	PCIe AXI3-master Window0 configuration register – 1

**Note:** All registers are 32-bit. Bits not shown in the table are reserved.

**Table 192 • SYSTEM\_CONF\_AXI\_MSTR\_WNDW\_2(0x208C)**

Bit Number	Name	Reset Value	Description
31:0	CONF_AXI_MSTR_WNDW_2	0x0	PCIe AXI3-master Window0 configuration register – 2

**Note:** All registers are 32-bit. Bits not shown in the table are reserved.

**Table 193 • SYSTEM\_CONF\_AXI\_MSTR\_WNDW\_3 (0x2090)**

Bit Number	Name	Reset Value	Description
31:0	CONF_AXI_MSTR_WNDW_3	0x0	PCIe AXI3-master Window0 configuration register – 3

**Note:** All registers are 32-bit. Bits not shown in the table are reserved.

**Table 194 • SYSTEM\_CONF\_AXI\_SLV\_WNDW\_0(0x2094)**

Bit Number	Name	Reset Value	Description
31:0	CONF_AXI_SLV_WNDW_0	0x0	PCIe AXI3-slave Window0 configuration register – 0

**Note:** All registers are 32-bit. Bits not shown in the table are reserved.

**Table 195 • SYSTEM\_CONF\_AXI\_SLV\_WNDW\_1(0x2098)**

Bit Number	Name	Reset Value	Description0
31:0	CONF_AXI_SLV_WNDW_1	0x0	PCIe AXI3-slave Window0 configuration register – 1

**Note:** All registers are 32-bit. Bits not shown in the table are reserved.

**Table 196 • SYSTEM\_CONF\_AXI\_SLV\_WNDW\_2(0x209C)**

Bit Number	Name	Reset Value	Description
31:0	CONF_AXI_SLV_WNDW_2	0x0	PCIe AXI3-slave Window0 configuration register – 2

**Note:** All registers are 32-bit. Bits not shown in the table are reserved.

**Table 197 • SYSTEM\_CONF\_AXI\_SLV\_WNDW\_3 (0x20A0)**

Bit Number	Name	Reset Value	Description
31:0	CONF_AXI_SLV_WNDW_3	0x0	PCIe AXI3-slave Window0 configuration register – 3

**Note:** All registers are 32-bit. Bits not shown in the table are reserved.

**Table 198 • SYSTEM\_DESKEW\_CONFIG(0x20A4)**

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



**Table 198 • SYSTEM\_DESKEW\_CONFIG(0x20A4)**

3:2	DESKEW_PLL_FDB_CLK	0x0	These bits set the PLL FEEDBACK clock DESKEW register. Delay addition of cells in the path of FEEDBACK clock to PLL. 00: Bypass delay cells 01: Add 1-cells 10: Add 2-cells 11: Add 3-cells
1:0	DESKEW_PLL_REF_CLK	0x0	These bits set the PLL REF clock DESKEW register. Delay addition of cells in the path of REFERENCE clock to PLL. 00: Bypass delay cells 01: Add 1-cells 10: Add 2-cells 11: Add 3-cells

**Table 199 • SYSTEM\_ADVCONFIG (0x20b4)**

Bit Number	Name	Reset Value	Description
0	K_BRIDGE_MODE	0x0	1'b0: Device is native endpoint, 1'b1: Device is a Root-port.
1	K_BRIDGE_ADDR_DEC	0x0	Advanced mode setting - PCIe controller 0 address decoding type, 1'b0: address decoding is performed by the BAR (RP/EP) 1'b1: address decoding is performed by Windows (RP only). This bit must be set to 0 for endpoint.
2	K_INFER_ELEC_IDLE	0x0	Enables the inferred electrical idle function in the PCIe core (k_glb[44])
3	DISABLE_PCIE_RESET	0x0	When '1' disables the ability for the internal DLUP/HOTRST and L2P2 reset generator to perform an automatic reset sequence of the main PCIe core logic.
4	DISABLE_PIPE_RESET	0x0	When '1' disables the ability for the internal PERST monitor and L2P2 reset generator to perform an automatic reset sequence of the PIPE logic and PMA block.
5	ENABLE_PERSTN_SUPPORT	0x0	When '1' enables the ability for the internal PERST monitor to enter reset when PERSTN is asserted. When '0' PERSTN is only used to detect the exit from L2P2 and initiate the reset sequence.
6	PCIE_CONFIG_NOSTALL	0x0	PCIe enabled: When 1, if PCIe configuration space is read before the PMA clock (PCLK) is stable, an APB SLVERR is generated, otherwise the APB cycle is stalled until the PCLK is stable. PCIe disabled: When '1', an APB PSLVERR is generated if the PCIe configuration space is accessed, otherwise the APB cycle is allowed to complete with simple assertion of PREADY.

**Table 200 • SYSTEM\_ECC\_ERR\_INJECT (0x20bc)**

Bit Number	Name	Reset Value	Description
7:0	CFGR_PCIE_ECC_ERROR_INJECT	0x0	<p>When one or more of the following bits are set, an ECC error is injected in the PCIe memories output when the next read occurs. This should then cause an ECC error to be detected.</p> <p>Bit 0: Transmit RAM multiple bit error</p> <p>Bit 1: Transmit RAM single bit error</p> <p>Bit 2: Receive RAM multiple bit error</p> <p>Bit 3: Receive RAM single bit error</p> <p>Bit 4: Replay RAM multiple bit error</p> <p>Bit 5: Replay RAM single bit error</p> <p>Single and double bit errors should not be injected at the same time on the same memory.</p>

**Table 201 • SYSTEM\_REFCLK\_MSIO\_CONFIG (0x20c4)**

Bit Number	Name	Reset Value	Description
0	REFCLK_MSIO_DISN	0x0	I/O N disable input 1 = disabled 0 = enabled
1	REFCLK_MSIO_DISP	0x0	I/O P disable input 1 = disabled 0 = enabled
4:2	REFCLK_MSIO_VSN	0x0	I/O N input decode bits 000 = ratio PCI, LVTTTL, or LVCMOS 010 = reference SSTL or HSTL 110 = differential LVDS, RSDS, or LVPECL
7:5	REFCLK_MSIO_VSP	0x0	I/O P input decode bits 000 = ratio PCI, LVTTTL, or LVCMOS 010 = reference SSTL or HSTL 110 = differential LVDS, RSDS, or LVPECL
10:8	REFCLK_MSIO_IMPN	0x0	I/O N impedance termination settings 000 = unterminated 001 = 75 $\Omega$ to vddi/2, single-ended 150up/150dn 010 = 150 $\Omega$ to vddi/2, single-ended 300up/300dn 011 = 50 $\Omega$ to vddi/2, single-ended 100up/100dn 100 = 100 $\Omega$ padp to padndifferential
13:11	REFCLK_MSIO_IMPP	0x0	I/O P impedance termination settings 000 = unterminated 001 = 75 $\Omega$ to vddi/2, single-ended 150up/150dn 010 = 150 $\Omega$ to vddi/2, single-ended 300up/300dn 011 = 50 $\Omega$ to vddi/2, single-ended 100up/100dn 100 = 100 $\Omega$ padp to padndifferential
14	REFCLK_MSIO_SCHMITT_SELN	0x0	I/O N 0 = schmitt trigger receiver, hysteresis off. 1 = single ended receiver, hysteresis on.
15	REFCLK_MSIO_SCHMITT_SELN	0x0	I/O P 0 = schmitt trigger receiver, hysteresis off. 1 = single ended receiver, hysteresis on.

**Table 201 • SYSTEM\_REFCLK\_MSIO\_CONFIG (0x20c4) (continued)**

Bit Number	Name	Reset Value	Description
18:16	REFCLK_MSIO_RT_ENN	0x0	I/O N receiver RT ctrl
21:19	REFCLK_MSIO_RT_ENP	0x0	I/O P receiver RT ctrl
22	REFCLK_MSIO_WPDN	0x0	I/O N weak pull down. 0 = off 1 = on
23	REFCLK_MSIO_WPDP	0x0	I/O P weak pull down. 0 = off 1 = on
24	REFCLK_MSIO_WPUN	0x0	I/O N weak pull up. 0 = off 1 = on
25	REFCLK_MSIO_WPUP	0x0	I/O P weak pull up. 0 = off 1 = on

**Table 202 • SYSTEM\_ENHANCEMENT (0x20c8)**

Bit Number	Name	Reset Value	Description
9:0	RSRV4	0x0	Reserved to maintain bit positions in register
10	EPCS_RXSKIP_ENABLE	0x0	When 1, enables the EPCS RXSKIP inputs to the PMA cores. Functions for XAUI and EPCS modes. The hardware default for this register is 0x0. However, for XAUI and EPCS modes, this register is set to 0x1 by the Libero software.

**Table 203 • SYSTEM\_TXFWF\_CONFIG (0x20cc)**

Bit Number	Name	Reset Value	Description
0	TXFWF_WMARK	0x0	1 = underflow is detected if wptr/rptr gap reaches 0 locations, overflow at 8 0 = underflow detected at 1 location, overflow at 7. Recommended setting is 0
2:1	TXFWF_FIFO_LANE_CTRL	0x0	00 = each lane controller controls its respective fifo storage 01 = controller 0 controls all 4 fifos 10 = controller 0 controls fifo 0 and fifo 1, while controller 2 controls fifo 2 and fifo 3 11 = reserved
3	TXFWF_TXOOB_USE_FIFO	0x0	1 = txoob signal goes through fwf, 0 = txoob signal goes around fwf
4	TXFWF_TXVAL_USE_FIFO	0x0	1 = txval signal goes through fwf, 0 = txval signal goes around fwf

**Table 204 • SYSTEM\_RXFWF\_CONFIG(0x20d0)**

Bit Number	Name	Reset Value	Description
0	RXFWF_WMARK	0x0	1 = underflow is detected if wptr/rptr gap reaches 0 locations, overflow at 8 0 = underflow detected at 1 location, overflow at 7. Recommended setting is 0
2:1	Reserved	0x0	Reserved
3	RXFWF_CLK_COMP	0x0	1 = use xaui xfifo in clock compensation mode 0 = use rxfwf/xfifo in flywheel mode

**Table 205 • SYSTEM\_FWF\_STATUS(0x20d4)**

Bit Number	Name	Reset Value	Description
15:12	RXFWF_UFLOW	0x0	RXFWF underflow lane[3:0]
11:8	RXFWF_OFLOW	0x0	RXFWF overflow lane[3:0]
7:4	TXFWF_UFLOW	0x0	TXFWF underflow lane[3:0]
3:0	TXFWF_OFLOW	0x0	TXFWF overflow lane[3:0]

### 4.1.3 SerDes Block Register

The SerDes block register map contains control and status information for the SerDes block and lanes. Each block uses 256 register bytes. However, these 256 bytes are mapped to 1 KB to generate a 32-bit APB output. The APB-to-SerDes programming interface bridge is implemented to convert the 32-bit APB bus system transactions to appropriate 8 bits. Since the RTG4 devices map the 4 SerDes lanes to 1 KB blocks, the overall register map size is 4 KB. The physical offset location of the SerDes block registers from the SerDes block system memory map is as follows:

- 0x1000-0x13FF - 1 KB - SerDes programming interface (Lane0)
- 0x1400-0x17FF - 1 KB - SerDes programming interface (Lane1)
- 0x1800-0x1BFF - 1 KB - SerDes programming interface (Lane2)
- 0x1C00-0x1FFF - 1 KB - SerDes programming interface (Lane3)

The SerDes block system register memory map occupies 1 KB of configuration memory map. Physical offset location of the SerDes block system registers is 0x2000 - 0x23FF from the SerDes block memory map. Table 164, page 136 describes the SerDes block system registers.

The 1 Kbyte register space can be divided between the protocol-specific read/write register and generic purpose register.

- **Configuration PHY registers (offset 0x000 to 0x03C):** These 16 registers are protocol-specific, with a reset value depending on the selected protocol, according to CONFIG\_PHY\_MODE register settings. For example, PLL\_F\_PCLK\_RATIO register may have different reset values for PCIe and XAUI mode. PCIe Gen1 features are configured using these 16 8-bit registers.
- Reserved registers (offset 0x040 to 0x0BC)
- **SerDes electrical parameter registers (offset 0x0C0 to 0x18C):** These 48 registers are internally reported values of parameters programmed inside the SerDes block. The register descriptions are reserved for factory testing only.
- **SerDes testing registers (offset 0x190 to 0x1FC):** These registers are used for testing the SerDes block. The register descriptions are reserved for factory testing only.
- **SerDes recompute register (offset 0x200):** This register is a command register that requires PMA control logic to recompute the SerDes parameter based on the new programmed set of register values.

- **SerDes PRBS error counter registers (offset 0x204 to 0x400):** There are 14 registers that are used for bit error rate testing. These registers are for lane0, lane1, lane2, or lane3 and the only difference between lane0, lane1, lane2, and lane3 is the base address specifying which lane it is for. The rest of the register spaces are unused.

The following table lists the SerDes block register mappings including reset values. Unused lanes default to 0x00 values.

**Table 206 • SerDes Block Lane Registers**

Register Name	Address Offset (Hex)	Reset Value	Type	Description
CR0	0X000	0x80	R/W	Lane control register 0 (Table 207, page 157).
ERRCNT_DEC	0X004	0x20	R/W	Clock count for error counter decrement (Table 208, page 158)
RXIDLE_MAX_ERRCNT_THR	0X008	0x48 or 0xF8	R/W	Error counter threshold – RX0 idle detect maximum latency (Table 209, page 158). Reset value for PCIe mode: 0x48 Reset value for other mode: 0xF8
IMPED_RATIO	0X00C	0x80	R/W	TX impedance ratio (Table 210, page 158).
PLL_F_PCLK_RATIO	0X010	0x24, 0x34, or 0x00	R/W	PLL F settings and PCLK ratio (Table 211, page 159) Reset value for PCIe mode: 0x24: 16-bit pipe interface and 250 MHz PCLK 0x34: 16-bit pipe interface and other PCLK 0x24: 8-bit pipe interface Reset value for other modes: 0x00
PLL_M_N	0X014	0x04, 0x13, or 0x69 <sup>1</sup>	R/W	PLL M and N settings (Table 212, page 159) Reset value for PCIe mode: 0x04 Reset value for XAUI mode: 0x13
CNT250NS_MAX	0X018	0x7C, 0x27, or 0x1F <sup>1</sup>	R/W	250 ns timer base count (Table 213, page 159) Reset value for PCIe mode: 0x7C Reset value for XAUI mode: 0x27
RE_AMP_RATIO	0X01C	0x00	R/W	RX equalization amplitude ratio (Table 214, page 160)
RE_CUT_RATIO	0X020	0x00	R/W	RX equalization cut frequency (Table 215, page 160)
TX_AMP_RATIO	0X024	0x80	R/W	TX amplitude ratio (Gen 1 PCIe and lower data rates) (Table 216, page 160)
TX_PST_RATIO	0X028	0x15 or 0x00	R/W	TX post-cursor ratio (Table 217, page 161) Reset value for PCIe mode: 0x15 Reset value for XAUI mode: 0x15 Reset value for EPCS mode: 0x00
TX_PRE_RATIO	0X02C	0x00	R/W	TX pre-cursor ratio (Table 218, page 161)
ENDCALIB_MAX	0X030	0x10	R/W	End of calibration counter (Table 219, page 161)
CALIB_STABILITY_COUNT	0X034	0x38	R/W	Calibration stability counter (Table 220, page 161)
POWERDOWN	0X038	0x00	R/W	Power-down feature (Table 221, page 162)
RX_OFFSET_COUNT	0X03C	0x70	R/W	RX offset counter (Table 222, page 164)

**Table 206 • SerDes Block Lane Registers (continued)**

Register Name	Address Offset (Hex)	Reset Value	Type	Description
<b>Note:</b> The published register syntax is prefixed by LANEn (where n is 0 : 3). The register bit is appended to the block register name. Example: LANE0_CR0				
Reserved	0X040	–	R/W	Reserved
Reserved	0X044	0x09	R/W	Reserved
Reserved	0X048	0x7C	R/W	Reserved
Reserved	0X050	0x15	R/W	Reserved
Reserved	0X054	0x00	R/W	Reserved
Reserved	0X058	0x20	R/W	Reserved
Reserved	0X05C	0x00	R/W	Reserved
Reserved	0X060	0x80	R/W	Reserved
Reserved	0X064	0x78	R/W	Reserved
Reserved	0X068	0x68	R/W	Reserved
Reserved	0X06C	0x60	R/W	Reserved
Reserved	0X070	0x58	R/W	Reserved
Reserved	0X074	0x50	R/W	Reserved
Reserved	0X078	0x48	R/W	Reserved
Reserved	0X07C	0x40	R/W	Reserved
Reserved	0X080	0x00	R/W	Reserved
Reserved	0X084	0x00	R/W	Reserved
Reserved	0X088	0x00	R/W	Reserved
Reserved	0X08C	0x00	R/W	Reserved
Reserved	0X090	0x15	R/W	Reserved
Reserved	0X094	0x00	R/W	Reserved
Reserved	0X098	0x20	R/W	Reserved
Reserved	0X09C	0x00	R/W	Reserved
Reserved	0X0A0	0x50	R/W	Reserved
Reserved	0X0A4	0x58	R/W	Reserved
Reserved	0X0A8	0x48	R/W	Reserved
Reserved	0X0AC	0x40	R/W	Reserved
Reserved	0X0B0	0x38	R/W	Reserved
Reserved	0X0B4	0x30	R/W	Reserved
Reserved	0X0B8	0x28	R/W	Reserved
Reserved	0X0BC	0x20	R/W	Reserved
<b>Note:</b> Reserved R/W registers are listed with default values for reference only.				
PMA_STATUS	0X0C0	0x80	R/O	PMA status register- correct read back value = 0x80 (Table 223, page 164)
PRBS_CTRL	0X190	0x00	R/W	PRBS control register (Table 224, page 164)

**Table 206 • SerDes Block Lane Registers (continued)**

Register Name	Address Offset (Hex)	Reset Value	Type	Description
PRBS_ERRCNT	0X194	0x00	R/O	PRBS error counter register (Table 225, page 165)
PHY_RESET_OVERRIDE	0X198	0x00	R/W	PHY reset override register (Table 226, page 165)
PHY_POWER_OVERRIDE	0X19C	0x00	R/W	PHY power override register (Table 227, page 165)
CUSTOM_PATTERN_7_0	0X1A0	0x00	R/W	Custom pattern byte 0 (Table 228, page 166)
CUSTOM_PATTERN_15_8	0X1A4	0x00	R/W	Custom pattern byte 1 (Table 229, page 166)
CUSTOM_PATTERN_23_16	0X1A8	0x00	R/W	Custom pattern byte 2 (Table 230, page 166)
CUSTOM_PATTERN_31_24	0X1AC	0x00	R/W	Custom pattern byte 3 (Table 231, page 167)
<b>Note:</b> Registers 49-99 are factory reserved for testing purposes.				
CUSTOM_PATTERN_39_32	0X1B0	0x00	R/W	Custom pattern byte 4 (Table 232, page 167)
CUSTOM_PATTERN_47_40	0X1B4	0x00	R/W	Custom pattern byte 5 (Table 233, page 167)
CUSTOM_PATTERN_55_48	0X1B8	0x00	R/W	Custom pattern byte 6 (Table 234, page 168)
CUSTOM_PATTERN_63_56	0X1BC	0x00	R/W	Custom pattern byte 7 (Table 235, page 168)
CUSTOM_PATTERN_71_64	0X1C0	0x00	R/W	Custom pattern byte 8 (Table 236, page 168)
CUSTOM_PATTERN_79_72	0X1C4	0x00	R/W	Custom pattern byte 9 (Table 237, page 169)
CUSTOM_PATTERN_CTRL	0X1C8	0x00	R/W	Custom pattern control (Table 238, page 169)
Reserved	0X1CC	0x00	R/O	Custom pattern status register (Table 239, page 169)
PCS_LOOPBACK_CTRL	0X1D0	0x00	R/W	PCS loopback control (Table 240, page 170)
GEN1_TX_PLL_CCP	0X1D4	0x06	R/W	Gen1 transmit PLL current charge pump (Table 241, page 170)
GEN1_RX_PLL_CCP	0X1D8	0x66	R/W	Gen1 receive PLL current charge pump (Table 242, page 170)
Reserved	0X1DC	0x00	R/W	Reserved
Reserved	0X1E0	0x00	R/W	Reserved
CDR_PLL_MANUAL_CR	0X1E4	0x00	R/W	CDR PLL manual control (Table 243, page 171)
UPDATE_SETTINGS	0X200	0x00	W/O	Update settings command register (Table 244, page 171)
PRBS_ERR_CYC_FIRST_7_0	0X280	0x00	R/O	PRBS first error cycle counter register bits[7:0] (Table 245, page 171)
PRBS_ERR_CYC_FIRST_15_8	0X284	0x00	R/O	PRBS first error cycle counter register bits[15:8] (Table 246, page 172)
PRBS_ERR_CYC_FIRST_23_16	0X288	0x00	R/O	PRBS first error cycle counter register bits[23:16] (Table 247, page 172)
PRBS_ERR_CYC_FIRST_31_24	0X28C	0x00	R/O	PRBS first error cycle counter register bits[31:24] (Table 248, page 172)
PRBS_ERR_CYC_FIRST_39_32	0X290	0x00	R/O	PRBS first error cycle counter register bits[39:32] (Table 249, page 172)



**Table 206 • SerDes Block Lane Registers (continued)**

Register Name	Address Offset (Hex)	Reset Value	Type	Description
PRBS_ERR_CYC_FIRST_47_40	0X294	0x00	R/O	PRBS first error cycle counter register bits[47:40] (Table 250, page 173)
PRBS_ERR_CYC_FIRST_49_48	0X298	0x00	R/O	PRBS first error cycle counter register bits [49:48] (Table 251, page 173)
PRBS_ERR_CYC_LAST_7_0	0X2A0	0x00	R/O	PRBS last error cycle counter register bits[7:0] (Table 252, page 173)
PRBS_ERR_CYC_LAST_15_8	0X2A4	0x00	R/O	PRBS last error cycle counter register bits[15:8] (Table 253, page 173)
PRBS_ERR_CYC_LAST_23_16	0X2A8	0x00	R/O	PRBS last error cycle counter register bits[23:16] (Table 254, page 174)
PRBS_ERR_CYC_LAST_31_24	0X2AC	0x00	R/O	PRBS last error cycle counter register bits[31:24] (Table 255, page 174)
PRBS_ERR_CYC_LAST_39_32	0X2B0	0x00	R/O	PRBS last error cycle counter register bits[39:32] (Table 256, page 174)
PRBS_ERR_CYC_LAST_47_40	0X2B4	0x00	R/O	PRBS last error cycle counter register bits[47:40] (Table 257, page 175)
PRBS_ERR_CYC_LAST_49_48	0X2B8	0x00	R/O	PRBS last error cycle counter register bits[49:48] (Table 258, page 175)
<b>Note:</b> The published register syntax is prefixed by LANEn (where n is 0 : 3). The register bit is appended to the block register name. For example: LANE0_CR0				

1. This is programmed based on the inputs provided in the Libero configurator.

#### 4.1.4 SerDes Block Register Bit Definitions

The following tables list the bit definitions for the registers of the SerDes block registers. These registers are not SEU protected.

**Table 207 • CR0**

Bit Number	Name	Reset Value	Description
7	AUTO_SHIFT	0x1	Defines whether the electrical idle 1 pattern is automatically shifted in the SerDes block after loading the drive pattern. When set to 1, electrical idle 1 or drive mode can be entered within a single aTXClk clock cycle. When set to 0, 23 clock cycles are required to dynamically switch between electrical idle 1 and drive mode. In general, this bit is always set to 1. Unused lanes are set to 0.
6	FORCE_RX_DETECT	0x0	Forces the result of PCIe receiver detect operation to be always detected. This register can be used on unreliable results of RX detect operations. When set to 1, the result of the PCIe receiver detect operation is always positive and thus makes the PHY, non-compliant to PCIe.
[5:4]	CDR_REFERENCE	0x0	Defines the CDR reference PLL mode. By default, these two bits must be set to 0 when RefClk is used for the CDR reference clock.



**Table 207 • CR0 (continued)**

Bit Number	Name	Reset Value	Description
3	PMA_DRIVEN_MODE	0x0	Places the CDR PLL in PMA driven mode. When set to 0, the PCS driven mode is selected for locking the SerDes CDR circuitry and when set to 1, PMA driven mode is used.
2	CDR_PLL_DELTA	0x0	Defines the frequency comparator threshold value to switch from fine grain locking to frequency lock, which controls the input signal of the PMA block when CDR is configured in PMA driven mode, and the equivalent function when the PMA is configured in PCS driven mode. When set to 0, the RX clock and TX clock must be in a 0.4% difference range; 0.8% when set to 1.
1	SIGNAL_DETECT_THRESHOLD	0x0	Defines the Schmitt trigger signal detection threshold used to detect the electrical idle on RX. When set to 0, threshold is 125 mV ( $\pm 40\%$ ), and when set to 1, threshold is 180 mV ( $\pm 33\%$ ).
0	TX_SELECT_RX_FEEDBACK	0x0	Must be set to 0 when RefClk is used for TX PLL. Set to 1 when the CDR PLL is used as TX PLL reference clock.

**Note:** This register can be reprogrammed when the PHY is reset or when calibration has completed (PMA is ready).

**Table 208 • ERRCNT\_DEC**

Bit Number	Name	Reset Value	Description
[7:0]	ERRCNT_DEC	0x20	In PCS driven mode, the PMA control logic counts the number of errors detected by the PCS logic in order to switch back to frequency lock mode of the CDR PLL. This counter is used to decrement the error counter for every 16*errcnt_dec[7:0] aTXClk clock cycles.

**Note:** This register can be reprogrammed when the PHY is under reset or when calibration has completed (PMA is ready).

**Table 209 • RXIDLE\_MAX\_ERRCNT\_THR**

Bit Number	Name	Reset Value	Description
[7:4]	RXIDLE_MAX	0x4-PCIe 0xF-others	Defines the number of clock cycles required before the activity detected output of the PMA block and reports either electrical idle or valid input data. This register must be set to at least 3 because the activity detected signal is considered as metastable by the PCS logic.
[3:0]	ERRCNT_THR	0x8	In PCS driven mode, the PMA control logic counts the number of errors detected by the PCS logic in order to switch back frequency threshold value after which the CDR PLL switches.

**Note:** This register can be reprogrammed any time during operation.

**Table 210 • IMPED\_RATIO**

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

**Table 210 • IMPED\_RATIO**

[7:0]	IMPED_RATIO	0x80	Fine-tunes the impedance ratio of the PMA block with a nominal value of 100 $\Omega$ , corresponding to a multiplication factor of 1, which is encoded as 8'd128. A 150 $\Omega$ , impedance corresponds to 2/3 ratio, encoded as 8'd85.
-------	-------------	------	--

**Note:** This register can be reprogrammed when the PHY is reset or when calibration has completed (PMA is ready).

**Table 211 • PLL\_F\_PCLK\_RATIO**

Bit Number	Name	Reset Value	Description
[7:6]	Reserved	PCIe mode: 0x0 XAUI mode: 0x0 EPCS mode: 0x0	
[5:4]	DIV_MODE0	PCIe mode: 0x3 XAUI mode: 0x0 EPCS mode: 0x0	Defines the ratio between PCLK and aTXClk. PCLK is used by the PCIe PCS logic as well as by majority of the PMA control logic. It is also useful for other protocols to reduce the amount of logic required in a high aTX0Clk frequency. In non-PCIe mode, this register is only useful if pipe_pclkout is used by any logic. A value of 00 is used for divide-by-1, 10 for divide-by-2 and 11 for divide-by-4.
[3:0]	F	PCIe mode: 0x4 XAUI mode: 0x00: 0x0	Defines the aRXF[3:0] and aTXF[3:0] settings of the PMA block. The same F value is applied to both RX and TX PLL.

**Note:** This register must only be reprogrammed when PHY is under reset or when both RX PLL and TX PLL are under reset.

**Table 212 • PLL\_M\_N**

Bit Number	Name	Reset Value (Libero Programmed)	Description
7	CNT250NS_MAX[8]	PCIe mode XAUI mode EPCS mode	This bit is concatenated to the Reg06 register as MSB to define the 250 ns base time.
[6:5]	M[1:0]	PCIe mode XAUI mode EPCS mode	Defines the TX PLL M values and CDR PLL M value settings of the PMA block. For PCIe, it corresponds to the Gen1 settings. The same M value is applied to both RX and TX PLL.
[4:0]	N[4:0]	PCIe mode XAUI mode EPCS mode	Defines the TX PLL N values and CDR PLL N value settings of the PMA block. For PCIe, it corresponds to the Gen1 settings. The same N value is applied to both RX and TX PLL.

**Note:** This register must only be reprogrammed when PHY is under reset or when both RX PLL and TX PLL are under reset. See Table 206, page 154 for information about Libero-based values.

**Table 213 • CNT250NS\_MAX**

Bit Number	Name	Reset Value (Libero Programmed)	Description
------------	------	---------------------------------	-------------

**Table 213 • CNT250NS\_MAX**

[7:0]	CNT250NS_MAX	PCIe mode XAUI mode EPCS mode	Defines the base count of a 250 ns event based on the aTXClk clock. This counter is used by the CDR PLL in PCS driven mode and also by the PMA control logic for operations such as receiver detect, electrical idle 2 and 3 states. In case of a non-integer value, the base count must be rounded up. This register must be set correctly for all protocols. The CNT250NS_MAX setting determines the time FINE_LOCK qualified before raising the RX_VALID signal.
-------	--------------	-------------------------------------	---

**Note:** This register must only be reprogrammed when the PHY is under reset for proper operation. It impacts the PCS-driven CDR PLL mode as well as calibration and thus has no effect after calibration is completed (PMA is ready) or if the PHY CDR PLL is used in PMA driven mode.

**Table 214 • RE\_AMP\_RATIO**

Bit Number	Name	Reset Value	Description
[7:0]	RE_AMP_RATIO	0x00	Defines the RX equalization amplitude ratio where the maximum value of 8'd128 corresponds to 100%. If RX equalization is not used, this register can be set to zero.

**Note:** This register can be reprogrammed during normal operation but the effect appears only when the parameters for the SerDes receiver are updated at the end of calibration or when UPDATE\_SETTINGS is programmed.

**Table 215 • RE\_CUT\_RATIO**

Bit Number	Name	Reset Value	Description
[7:0]	RE_CUT_RATIO	0x00	Defines the RX equalization. See <a href="#">Table 4</a> , page 12.

**Note:** This register can be reprogrammed during normal operation but the effect appears only when the parameters for the SerDes receiver are updated at the end of calibration or when UPDATE\_SETTINGS is programmed.

**Table 216 • TX\_AMP\_RATIO**

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO	0x80	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are not specified in RTG4 FPGA datasheet and should be validated in application. For PCIe, this register is used for Gen1 speed only.

**Note:** This register can be reprogrammed during normal operation but the effect appears only when the parameters for the SerDes transmitter are updated at the end of calibration, on entry or exit of TX electrical idle I or when UPDATE\_SETTINGS is programmed.

**Table 217 • TX\_PST\_RATIO**

Bit Number	Name	Reset Value	Description
[7:0]	TX_PST_RATIO	0x15 or 0x00	Defines the TX post-cursor ratio for the Gen1 speed used for selecting the de-emphasis of the switching bit versus non-switching bit. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. A value of –3.5 dB corresponds to 8'd21 encoding.

**Note:** This register can be reprogrammed during normal operation but the effect appears only when the parameters for the SerDes transmitter are updated at the end of calibration, on entry or exit of TX Electrical Idle I or when UPDATE\_SETTINGS is programmed.

**Table 218 • TX\_PRE\_RATIO**

Bit Number	Name	Reset Value	Description
[7:0]	TX_PRE_RATIO	0x00	Defines the TX pre-cursor ratio for the Gen1 speed used for selecting the de-emphasis of the switching bit versus non-switching bit. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%.

**Note:** This register can be reprogrammed during normal operation but the effect appears only when the parameters for the SerDes transmitter are updated (at the end of calibration, on entry or exit of TX electrical idle I or when UPDATE\_SETTINGS is programmed).

**Table 219 • ENDCALIB\_MAX**

Bit Number	Name	Reset Value	Description
[7:0]	ENDCALIB_MAX	0x10	Defines the amount of time in microseconds required by the PMA to settle its electrical level after loading electrical idle 1 in the TX driver at the end of calibration. All operations are automatically performed by the PMA control logic but the SerDes transmitter can start driving data on the link immediately after the end of calibration. By default, a 10 $\mu$ s delay between end of calibration and mission mode is set (but any value can work).

**Note:** This register can be reprogrammed when the PHY is under reset or when calibration has completed (PMA is ready).

**Table 220 • CALIB\_STABILITY\_COUNT**

Bit Number	Name	Reset Value	Description
[7:5]	CALIB_SETTLE_MAX	0x1	Defines the amount of time in microseconds required by the PMA to settle its electrical level after loading electrical idle 1 in the TX driver at the end of calibration. Note that the entire operation is automatically performed by the PMA control logic but the SerDes transmitter can start driving data on the link immediately after end of the calibration. By default, a 10 $\mu$ s delay between end of calibration and mission mode is set (but any value can work).

**Table 220 • CALIB\_STABILITY\_COUNT**

[4:0]	CALIB_STABLE_MAX	0x18	Defines the number of clock cycles before which the impedance calibrator results (aZCompOp = 1, impedance calibrator result is greater than nominal; and aZCompOp = 0, impedance calibrator result is less than nominal) can be checked for stability after impedance calibration control values (aZCalib) modification. This is used for TX, RX, and RX equalization calibration.
-------	------------------	------	---

**Note:** This register can be reprogrammed when the PHY is under reset or when calibration has completed (PMA is ready).

**Table 221 • POWERDOWN**

Bit Number	Name	Reset Value	Description
[7:6]	RXIDLE_MSB	0x0	These bits are used as the most significant bits (MSBs) of the activity detector logic to specify that no activity has been detected up to 61 aTXClkp clock cycles. These bits are the two MSBs; the rxidle_max[3:0] field of Reg02 represents the least significant bit (LSB) part.
5	FORCE_SIGNAL	0x0	When this bit is set, the PHY disables the idle detection circuitry and forces signal detection on the receiver. This bit is generally always set to disable (0x0) unless the activity detector logic must be bypassed. In the latter case, the PMA control logic always reports activity detected on the link (when set to 1). This bit can be used, if the activity detector of the SerDes PMA block does not work for the selected protocol (as it is outside range of functionality). The activity detector's output causes the Rx PLL to go to its <b>FREQ_LOCK</b> state under a <b>no signal</b> condition. The Rx PLL will otherwise wander if left in the <b>FINE_LOCK</b> state while no serial data is present. This wander can lead to loss of control of the Rx PLL when the output frequency rises above a critical maximum frequency. If the activity detector is forced to flag the <b>signal present</b> condition ( <b>FORCE_SIGNAL=0x1</b> ), then it is up to the user to identify the <b>no signal</b> condition by other means and assert a reset to the PHY as a corrective measure.
4	FORCE_IDLE	0x0	When this bit is set, the PHY disables the idle detection circuitry and forces electrical Idle detection on the receive side. By default, this bit is cleared and may be set only for specific conditions or during testing such as generating a fake loss of signal to the PCS or MAC layer, forcing a retraining of word aligner or any training state machine. As long as this bit is set, the activity detector logic of the PMA control logic reports that no signal is detected on the receive side. If the CDR PLL PCS driven mode is selected, the CDR PLL is directed in lock to the reference clock state, leading to potential wrong data received by the SerDes (because the CDR PLL is not locked to incoming data).
3	NO_FCMP	0x0	When set, this bit disables the frequency comparator logic of the PCS driven CDR PLL control logic. When not set, the frequency comparator logic is no longer part of the condition for going from fine-grain lock state to frequency acquisition. This mode locks to the refclk.

**Table 221 • POWERDOWN (continued)**

Bit Number	Name	Reset Value	Description
2	PMFF_ALL	0x0	This is used with PCIe only. When set, it disables the function that waits for every active lane to have valid data to transmit before generating a global read enable. This bit is intended to be used in case of any issue with this function. When set, each lane may start transmitting data with one 500 MHz clock uncertainty (corresponding to 5 or 10 bits time, depending on the speed of the link). Even if it violates the protocol requirement, the PCIe standard is strong enough to support the non-compliance.
1	CDR_ERR	0x0	When set, this register internally disables the error counter of the CDR PLL state machine, which switches the CDR PLL back to frequency mode acquisition when the number of errors counted is higher than the predefined error threshold. This bit is intended to disabling this function in case of any issue with the PHY. This is available for all SerDes modes.
0	CDR_P1	0x0	<p>Defines the state of the CDR PLL when PHY is in P1 low power mode.</p> <p>When set to 0, the CDR PLL is put in reset and low power, that enables maximum power savings. When the opposite component sends the TS1 ordered set to drive the link in recovery, only the PIPE_RXELECIDLE signal is deasserted at the PIPE interface and the PHY waits for the controller to change the pipe_powerdown[1:0] signal back to P0 before retraining the CDR PLL (~6 <math>\mu</math>s) and sending received data to the controller.</p> <p>When set to 1, the CDR PLL is kept alive in frequency lock mode in the P1 state, which enables a faster recovery time from the P1 state but consumes more power (all RX logic is kept alive and consumes power in the P1 state). This register must not be set for applications which remove the reference clock in P1 mode (generally associated with the CLKREQ# signal, express card application, and more generally power sensitive applications).</p>

**Note:** This register can be reprogrammed when the PHY is under reset or when calibration has completed (PMA is ready), except for bit 2, which can only be modified under reset condition.

**Table 222 • RX\_OFFSET\_COUNT**

Bit Number	Name	Reset Value	Description
[7:5]	RXOFF_SETTLE_MAX	0x3	Defines the number of clock cycles before which the aRXDNullDat signal can be checked for stability after aRXDNull[3:0] modification. This is also used for aRXD, aRXT, and Schmitt trigger calibration. The value of this register expresses a number of (2*N+1) PCLK clock cycles.
[4:0]	RXOFF_STABLE_MAX	0x10	Defines the number of clock cycles where the aRXDNullDat signal is checked for stability.

**Table 223 • PMA\_STATUS**

Bit Number	Name	Reset Value	Description
[7]	PMA_RDY	0x01	This read-only register indicates that the PMA has completed its internal calibration sequence after power-up and PHY reset de-assertion.
[6:0]			Reserved

**Table 224 • PRBS\_CTRL**

Bit Number	Name	Reset Value	Description
7			Unused
6	PRBS_CHK	0x0	When set, this signal starts the PRBS pattern checker. It can be set at the same time as the PRBS generator, while the PRBS checker logic waits for 256 clock cycles and CDR to be in lock state to enable the PRBS pattern comparison (allows a total latency of 256 cycles to loop back the transmitted data).
[5:4]	Reserved	–	Reserved
[3:2]	PRBS_TYP[1:0]	0x0	Defines the type of PRBS pattern which is applied. PRBS7, when set to 00; PRBS11, when set to 01; PRBS23, when set to 10; and PRBS31, when set to 11.
1	LPBK_EN	0x0	When set, the PMA is put in near-end loopback (serial loopback from TX back to RX). PRBS tests can be done using the near-end loopback of the PMA, some load board, or any far-end loopback implemented in the opposite component. When near-end loopback bit is set, the idle detector always reports valid data, enabling the PCS driven CDR PLL locking logic to lock on the input data.
0	PRBS_GEN	0x0	When set, this signal starts the PRBS pattern transmission.

**Note:** This register can be programmed any time but has functional impact because it can configure the SerDes in loopback or generate the PRBS pattern.

**Table 225 • PRBS\_ERRCNT**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERRCNT[7:0]	0x00	<p>This test reports the number of PRBS errors detected when the PRBS test is applied.</p> <p>This register is automatically cleared when the PRBS_CHK register is cleared (requires testing the value of this register when the test is running).</p> <p>The PRBS error counter saturates at 254 errors, the 255 count value corresponding to an error code where the CDR PLL is not locked to incoming data. When such an error code is detected, the PRBS test must wait for a longer time for the CDR PLL to synchronize on input data, before enabling the PRBS checker or simply timeout, reporting that no data has been received at all.</p> <p>The PRBS error counter logic also counts errors when the PRBS invariant (all zero value) is obtained, considering input data as error data.</p>

**Table 226 • PHY\_RESET\_OVERRIDE**

Bit Number	Name	Reset Value	Description
7	RXHF_CLKDN	0x0	When set, this signal disables the RX PLL VCO settings by applying a static zero to the PMA aRXHfClkDnb signal.
6	TXHF_CLKDN	0x0	When set, this signal disables the TX PLL VCO by applying a static zero to the PMA aTXHfClkDnb.
5	RXPLL_RST	0x0	When set, this signal resets the RX PLL settings by applying a static zero to the PMA aCdrPIIRstb signal.
4	TXPLL_RST	0x0	When set, this signal initializes the TX PLL settings by applying a static zero to the PMA aTXPIIRstb signal.
3	RXPLL_INIT	0x0	When set, this signal resets the RX PLL settings by applying a static zero to the PMA aCdrPIIRstb signal.
2	TXPLL_INIT	0x0	When set, this signal initializes the TX PLL settings by applying a static one to the PMA aTXPIIDivInit signal.
1	RX_HIZ	0x0	When set, this signal forces the RX driver to hiZ, applying a static one to the PMA aForceRXHiZ signal.
0	TX_HIZ	0x0	When set, this signal forces the TX driver to hiZ, applying a static one to the PMA aForceTXHiZ signal.

**Note:** This register can be programmed any time but has functional impact on the SerDes because it can put the PLL under reset or place part of the SerDes in low power mode, bypassing the functional mode.

**Table 227 • PHY\_POWER\_OVERRIDE**

Bit Number	Name	Reset Value	Description
[7:1]	–	–	Reserved
0	RX_PWRDN	0x0	When set, this register forces the RX PMA logic to be in power-down mode.



**Note:** This register can be programmed any time but has functional impact on the SerDes because it can power-down the receiver part of the SerDes, bypassing the functional mode.

**Table 228 • CUSTOM\_PATTERN\_7\_0**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN[7:0]	0x00	Enables bit 7 to bit 0 to program a custom pattern instead of the implemented PRBS generator/checker. The PRBS mode must still be selected to transmit this custom pattern on the transmit line, but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the receiver line. In the latter case, the PMA block function is used to perform a word alignment

**Note:** This register can be programmed any time and has no functional impact as long as the custom pattern generation is not enabled and selected.

**Table 229 • CUSTOM\_PATTERN\_15\_8**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [15:8]	0x00	Enables bit 15 to bit 8 to program a custom pattern instead of the implemented PRBS generator/checker. The PRBS mode must still be selected to transmit this custom pattern on the transmit line, but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the receiver line. In the latter case, the PMA block function is used to perform a word alignmentfunction.

**Note:** This register can be programmed any time and has no functional impact as long as the custom pattern generation is not enabled and selected.

**Table 230 • CUSTOM\_PATTERN\_23\_16**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [23:16]	0x00	Enables bit 23 to bit 16 to program a custom pattern instead of the implemented PRBS generator/checker. PRBS mode must still be selected to transmit this custom pattern on the transmit line but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the receiver line. In the latter case, the PMA block function is used to perform a word alignment function.

**Note:** This register can be programmed any time and has no functional impact as long as the custom pattern generation is not enabled and selected.

**Table 231 • CUSTOM\_PATTERN\_31\_24**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [31:24]	0x00	This register enables bit 31 to bit 24 to program a custom pattern instead of the implemented PRBS generator/checker. PRBS mode must still be selected to transmit this custom pattern on the transmit line, but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the receiver line. In the latter case, the PMA block function is used to perform word alignment function.

**Note:** This register can be programmed any time and has no functional impact as long as the custom pattern generation is not enabled and selected.

**Table 232 • CUSTOM\_PATTERN\_39\_32**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [39:32]	0x00	Enables bit 39 to bit 32 to program a custom pattern instead of the implemented PRBS generator/checker. PRBS mode must still be selected to transmit this custom pattern on the transmit line, but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the receiver line. In the latter case, the PMA block function is used to perform a word alignment function.

**Note:** This register can be programmed any time and has no functional impact as long as the custom pattern generation is not enabled and selected.

**Table 233 • CUSTOM\_PATTERN\_47\_40**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [47:40]	0x00	Enables bit 47 to bit 40 to program a custom pattern instead of the implemented PRBS generator/checker. PRBS mode must still be selected to transmit this custom pattern on the transmit line, but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for purpose of eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the receiver line. In the latter case, the PMA block function is used to perform a word alignment function.

**Note:** This register can be programmed any time and has no functional impact as long as the custom pattern generation is not enabled and selected.

**Table 234 • CUSTOM\_PATTERN\_55\_48**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [55:48]	0x00	Enables bit 55 to bit 48 to program a custom pattern instead of the implemented PRBS generator/checker. The PRBS mode must still be selected to transmit this custom pattern on the transmit line but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the receiver line. In the latter case, the PMA block function is used to perform a word alignment function.

**Note:** This register can be programmed any time and has no functional impact as long as the custom pattern generation is not enabled and selected.

**Table 235 • CUSTOM\_PATTERN\_63\_56**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [63:56]	0x00	Enables bit 63 to bit 56 to program a custom pattern instead of the implemented PRBS generator/checker. The PRBS mode must still be selected to transmit this custom pattern on the transmit line but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the receiver line. In the latter case, the PMA block function is used to perform a word alignment function.

**Note:** This register can be programmed any time and has no functional impact as long as the custom pattern generation is not enabled and selected.

**Table 236 • CUSTOM\_PATTERN\_71\_64**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [71:64]	0x00	Enables bit 71 to bit 64 to program a custom pattern instead of the implemented PRBS generator/checker. The PRBS mode must still be selected to transmit this custom pattern on the transmit line but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the receiver line. In the latter case, the PMA block function is used to perform a word alignment function.

**Note:** This register can be programmed any time and has no functional impact as long as the custom pattern generation is not enabled and selected.

**Table 237 • CUSTOM\_PATTERN\_79\_72**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [79:72]	0x00	Enables bit 79 to bit 72 to program a custom pattern instead of the implemented PRBS generator/checker. The PRBS mode must still be selected to transmit this custom pattern on the transmit line but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the receiver line. In the latter case, the PMA block function is used to perform a word alignment function.

**Note:** This register can be programmed any time and has no functional impact as long as the custom pattern generation is not enabled and selected.

**Table 238 • CUSTOM\_PATTERN\_CTRL**

Bit Number	Name	Reset Value	Description
7	Reserved	-	Reserved
6	CUST_AUTO	0x0	When this register is set, the word alignment is performed automatically by a state machine that checks whether the received pattern is word-aligned with the transmitted pattern and automatically uses the PMA CDR PLL skip bit function to find the alignment. Once the word alignment is detected, the custom pattern checker is word-aligned and can be enabled for detecting and counting any error over time.
5	Reserved	0x0	–
4	Reserved	0x0	–
[3:1]	CUST_TYP	0x0	Defines whether the custom pattern generated on the link is generated by the custom pattern register or by one of the hard-coded patterns: 000: Custom pattern register 100: All-zero pattern (0000...00) 101: All-one pattern (1111...11) 110: Alternated pattern (1010...10) 111: Dual alternated pattern (1100...1100)
0	CUST_SEL	0x0	When set, this signal replaces the PRBS data transmitted on the link by the custom pattern. The PRBS_SEL register must also be set for transmitting the custom pattern on the link.

**Note:** This register can be programmed any time but has functional impact on the SerDes because it can directly activate some part of the SerDes (aRXSkipBit), changing the current bitstream reception thus creating alignment errors.

**Table 239 • Reserved**

Bit Number	Name	Reset Value	Description
[7:0]	Reserved	0x0	–

**Table 240 • PCS\_LOOPBACK\_CTRL**

Bit Number	Name	Reset Value	Description
[7:4]	–	–	Unused
3	MESO_SYNC	0x0	This is read-only and reports whether the mesochronous clock alignment state machine has completed its process, and aligns the CDR receive clock to the transmit clock.
2	MESO_LPBK	0x0	When set, this register enables mesochronous loopback mode, which forces PMA received data to be re-transmitted on the PMA TX interface. This mode requires that no PPM exists between RX data and TX data (thus, both sides of the link use the same reference clock) and performs the CDR clock alignment to the transmit clock using the PMA CDR PLL skip bit functionality. This alignment is automatically performed by a state machine when the loopback register is set.
1	Reserved	0x0	–
0	Reserved	0x0	–

**Note:** This register can be used only with XAUI and EPCS protocols.

**Table 241 • GEN1\_TX\_PLL\_CCP**

Bit Number	Name	Reset Value	Description
[7:4]	–	–	Reserved
[2:0]	ATXICP_RATE0[2:0]	0x0	Defines the TX PLL charge pump current when the PMA runs in PCIe Gen1 speed or in any other protocol. This register is R/W in order to enable the default value modification by register programming, which is expected to be performed before reset deassertion.

**Note:** This register can be programmed when the PHY is reset.

**Table 242 • GEN1\_RX\_PLL\_CCP**

Bit Number	Name	Reset Value	Description
7	–	–	Reserved
[6:4]	ARXCDRCP_RATE0[2:0]	0x0	Defines the RX PLL charge pump current when the PMA is frequency locked and runs in PCIe Gen1 speed or in any other protocol. This register is R/W in order to enable the default value modification by register programming, which is expected to be performed before reset deassertion.
3	–	–	Reserved
[2:0]	ARXICP_RATE0[2:0]	0x0	Defines the RX PLL charge pump current when the PMA is CDR locked and runs in PCIe Gen1 speed or in any other protocol. This register is R/W in order to enable the default value modification by register programming, which is expected to be performed before reset deassertion.

**Note:** This register can be programmed when the PHY is reset.

**Table 243 • CDR\_PLL\_MANUAL\_CR**

Bit Number	Name	Reset Value	Description
[7:3]	–	–	Reserved
2	FINE_GRAIN	0x0	In PCS-driven mode when this register is set, it enables forcing the CDR PLL state machine to fine grain state. In this state, the CDR PLL locks on receive data, making RX data and RX CLOCKP valid on the PMA interface.
1	COARSE_GRAIN	0x0	When set, this register enables forcing the CDR PLL state machine to coarse grain state when used in PCS driven mode (see Reg00 bit 3 set to 0). In this state, the CDR PLL performs a coarse grain lock on receive data, enabling adjustment of its clock up to 5000 PPM.
0	FREQ_LOCK	0x0	When set, this register enables forcing the CDR PLL state machine to frequency lock state when used in PCS-driven mode (see Reg00 bit 3 set to 0). In this state, the CDR PLL does not lock on receive data but on the reference clock.

**Table 244 • UPDATE\_SETTINGS**

Bit Number	Name	Reset Value	Description
[7:0]	UPDATE_SETTINGS[7:0]	0x00	A transient register (read always reports 0) where writing a 1 in bit 0 triggers a new computation of PMA settings based on the value written in register space registers. For PCIe, Microsemi recommends not to use this command register when the link does not transit to low power state or change rate.

**Note:** This register can be programmed any time, except during calibration, and triggers the RX/TX shift load logic to load new programmed settings into the SerDes. Thus, it must be written 0x01 only after a coherent set of register updates need to be loaded.

**Table 245 • PRBS\_ERR\_CYC\_FIRST\_7\_0**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_FIRST[7:0]	0x00	PRBS last error cycle counter register bits[7:0]. This read-only register reports on which clock cycle, the error counter has first been incremented after the PRBS error counter is enabled. It is a 50-bit counter, that enables the performance of bit error rate testing (BERT).

**Note:** The first error cycle counter information complementing the total number of errors detected may give information about bursts of errors or distributed errors, but may also require statistics. The test can be rerun several times with different test periods.

**Table 246 • PRBS\_ERR\_CYC\_FIRST\_15\_8**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_FIRST[15:8]	0x00	PRBS last error cycle counter register bits[15:8]. This read-only register reports on which clock cycle, the error counter has first been incremented after the PRBS error counter is enabled. It is a 50-bit counter, that enables the performance of BERT.

**Note:** The first error cycle counter information complementing the total number of errors detected may give information about bursts of errors or distributed errors, but may also require statistics. The test can be rerun several times with different test periods.

**Table 247 • PRBS\_ERR\_CYC\_FIRST\_23\_16**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_FIRST[23:16]	0x00	PRBS last error cycle counter register bits[23:16]. This read-only register reports on which clock cycle, the error counter has first been incremented after the PRBS error counter is enabled. It is a 50-bit counter, that enables performance of BERT.

**Note:** The first error cycle counter information complementing the total number of errors detected may give information about bursts of errors or distributed errors, but may also require statistics. The test can be rerun several times with different test periods.

**Table 248 • PRBS\_ERR\_CYC\_FIRST\_31\_24**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_FIRST[31:24]	0x00	PRBS last error cycle counter register bits[31:24]. This read-only register reports on which clock cycle the error counter has first been incremented after the PRBS error counter is enabled. It is a 50-bit counter, enabling performance of BERT.

**Note:** The first error cycle counter information complementing the total number of errors detected might give information about bursts of errors or distributed errors, but statistics might also be required. The test can be rerun several times with different test periods.

**Table 249 • PRBS\_ERR\_CYC\_FIRST\_39\_32**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_FIRST[39:32]	0x00	PRBS last error cycle counter register bits[39:32]. This read-only register reports on which clock cycle, the error counter has first been incremented after the PRBS error counter is enabled. It is a 50-bit counter, that enables the performance of BERT.

**Note:** The first error cycle counter information complementing the total number errors detected may give information about bursts of errors or distributed errors, but may also require statistics. The test can be rerun several times with different test periods.

**Table 250 • PRBS\_ERR\_CYC\_FIRST\_47\_40**

Bit Number	Name	Reset Value	Description
[7:2]	–	–	Reserved
[1:0]	PRBS_ERR_CYC_FIRST[47:40]	0x0	PRBS last error cycle counter register bits [47:40]. This read-only register reports on which clock cycle, the error counter has first been incremented after the PRBS error counter is enabled. It is a 50-bit counter, that enables the performance of BERT.

**Note:** The first error cycle counter information complementing the total number of errors detected may give information about bursts of errors or distributed errors, but may also require statistics. The test can be rerun several times with different test periods.

**Table 251 • PRBS\_ERR\_CYC\_FIRST\_49\_48**

Bit Number	Name	Reset Value	Description
[7:2]	–	–	Reserved
[1:0]	PRBS_ERR_CYC_FIRST[49:48]	0x0	PRBS last error cycle counter register bits [49:48]. This read-only register reports on which clock cycle, the error counter has first been incremented after the PRBS error counter is enabled. It is a 50-bit counter, that enables the performance of BERT.

**Note:** The first error cycle counter information complementing the total number of errors detected may give information about bursts of errors or distributed errors, but may also require statistics. The test can be rerun several times with different test periods.

**Table 252 • PRBS\_ERR\_CYC\_FIRST\_7\_0**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_LAST[7:0]	0x0	PRBS last error cycle counter register bits [7:0]. This read-only register reports on which clock cycle, the error counter has last been incremented after the PRBS error counter is enabled. It is a 50-bit counter, that enables the performance of BERT.

**Note:** The last error cycle counter information complementing the total number of errors detected may give information about bursts of errors or distributed errors, but may also require statistics. The test can be rerun several times with different test periods.

**Table 253 • PRBS\_ERR\_CYC\_FIRST\_15\_8**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_LAST[15:8]	0x00	PRBS last error cycle counter register bits [15:8]. This read-only register reports on which clock cycle, the error counter has last been incremented after the PRBS error counter is enabled. It is a 50-bit counter, that enables the performance of BERT.



**Note:** The last error cycle counter information complementing the total number of errors detected may give information about bursts of errors or distributed errors, but may also require statistics. The test can be rerun several times with different test periods.

**Table 254 • PRBS\_ERR\_CYC\_FIRST\_23\_16**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_LAST[23:16]	0x00	PRBS last error cycle counter register bits [23:16]. This read-only register reports on which clock cycle, the error counter has last been incremented after the PRBS error counter is enabled. It is a 50-bit counter, that enables the performance of BERT.

**Note:** The last error cycle counter information complementing the total number of errors detected may give information about bursts of errors or distributed errors, but may also require statistics. The test can be rerun several times with different test periods.

**Table 255 • PRBS\_ERR\_CYC\_FIRST\_31\_24**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_LAST[31:24]	0x00	PRBS last error cycle counter register bits [31:24]. This read-only register reports on which clock cycle, the error counter has last been incremented after the PRBS error counter is enabled. It is a 50-bit counter, that enables the performance of BERT.

**Note:** The last error cycle counter information complementing the total number of errors detected may give information about bursts of errors or distributed errors, but may also require statistics. The test can be rerun several times with different test periods.

**Table 256 • PRBS\_ERR\_CYC\_FIRST\_39\_32**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_LAST[39:32]	0x0	PRBS last error cycle counter register bits [39:32]. This read-only register reports on which clock cycle, the error counter has last been incremented after the PRBS error counter is enabled. It is a 50-bit counter, that enables the performance of BERT.

**Note:** The last error cycle counter information complementing the total number of errors detected may give information about bursts of errors or distributed errors, but may also require statistics. The test can be rerun several times with different test periods.

**Table 257 • PRBS\_ERR\_CYC\_FIRST\_47\_40**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_LAST[47:40]	0x0	PRBS last error cycle counter register bits [47:40]. This read-only register reports on which clock cycle, the error counter has last been incremented after the PRBS error counter is enabled. It is a 50-bit counter, that enables the performance of BERT.

**Note:** The last error cycle counter information complementing the total number of errors detected may give information about bursts of errors or distributed errors, but may also require statistics. The test can be rerun several times with different test periods.

**Table 258 • PRBS\_ERR\_CYC\_FIRST\_49\_48**

Bit Number	Name	Reset Value	Description
[7:2]	Reserved	0x0	Reserved
[1:0]	PRBS_ERR_CYC_LAST[49:48]	0x0	PRBS last error cycle counter register bits [49:48]. This read-only register reports on which clock cycle, the error counter has last been incremented after the PRBS error counter is enabled. It is a 50-bit counter that enables the performance of BERT.

**Note:** The last error cycle counter information complementing the total number of errors detected may give information about bursts of errors or distributed errors, but may also require statistics. The test can be rerun several times with different test periods.

## 4.2 SerDes Register Lock Bits Configuration

Register Lock Bits can prevent the SerDes configuration registers from being overwritten by hosts that have access to these registers. The lock bits can be managed using the Configure Register Lock Bits utility in the Libero SoC v12.5 or later design software.

### 4.2.1 Lock Bit File

Click **Generate FPGA Array Data** in the **Design Flow** pane to generate an initial default lock bit file. This file can be modified to lock specific SerDes registers in the bitstream.

The default file located at

`<proj_location>/designer/<root>/<root>_init_config_lock_bits.txt` can be used to make the required changes.

**Note:** Save the file using a different name if you modify the text file to set the lock bits.

### 4.2.2 Lock Bit File Syntax

A valid entry in the lock bit configuration file is defined as a `<lock_parameters> <lock bit value>` pair format.

The lock parameters are structured as follows:

- Lock bits syntax for a register: `<Physical block name>_<register name>_LOCK`
- Lock bits syntax for a specific field: `<Physical block name>_<register name>_<field name>_LOCK`

Example: `PCIE0_SER_PLL_CONFIG_LOW_PLL_REF_DIVISOR_LOCK 0`

Set the lock bit value to 1 to indicate that the register can be written to (unlocked). Set the lock bit value to 0 to indicate that the register cannot be written to (locked).

Lines starting with # or ; are comments. Empty lines are allowed in the lock bit configuration file. The following figure shows the lock bits example format.

**Figure 85 • Lock Bits Example Format**

```

1 # Register Lock Bits Configuration File for SERDES(s) and Fabric DDR
2 # Microsemi Corporation - Microsemi Libero Software Release v12.5 (Version 12.900.10.16)
3 # Date: Thu Oct 29 09:45:41 2020
4
5 # PCIE_SERDES_IF_C0_0/PCIE_SERDES_IF_C0_0/SERDESIF_INST/INST_PCIE_IP
6 PCIE0_SER_PLL_CONFIG_LOW_PLL_REF_DIVISOR_LOCK          0
7 PCIE0_SER_PLL_CONFIG_LOW_PLL_FEEDBACK_DIVISOR_LOCK     1
8 PCIE0_SER_PLL_CONFIG_LOW_PLL_OUTPUT_DIVISOR_LOCK       1
9 PCIE0_SER_PLL_CONFIG_HIGH_PLL_FILTER_RANGE_LOCK        1
10 PCIE0_SER_PLL_CONFIG_HIGH_PLL_LOCKWIN_LOCK            1
11 PCIE0_SER_PLL_CONFIG_HIGH_PLL_LOCKCNT_LOCK            1
12 PCIE0_SER_PLL_CONFIG_HIGH_PLL_RESET_LOCK              1
13 PCIE0_SER_PLL_CONFIG_HIGH_PLL_BYPASS_LOCK             1
14 PCIE0_SER_PLL_CONFIG_HIGH_PLL_FSE_LOCK               1
15 PCIE0_SER_PLL_CONFIG_HIGH_PLL_PD_LOCK                 1

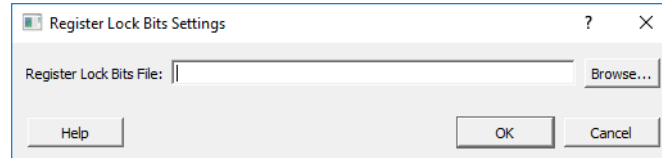
```

## 4.2.3 Locking and Unlocking a Register

Register lock bits are set in a text (\*.txt) file, which is then imported into the RTG4 project. A register can be locked or unlocked by setting the appropriate lock bit value in the lock bit configuration .txt file.

- Perform one or both of the following:
  - Set the lock bit value to 0 for the registers you want to lock.
  - Set the lock bit value to 1 for the registers you want to unlock.
- Save the file and import the file into the project (**Design Flow** window > **Configure Register Lock Bits**).
- From the **Design Flow** window, click **Configure Register Lock Bits** to open the configurator.
- Click **Browse...** to navigate to the text file (\*.txt) that contains the Register Lock Bit settings as shown in the following figure.

**Figure 86 • Register Lock Bit Settings**



- Re-run **Generate FPGA Array Data** and **Generate Bitstream**.

## 4.2.4 Configuration Report with Lock

The design\_init\_config.xml report records the state of the lock bits in the generated data shown in the following figure. This report is generated when the **Generate FPGA Data** is completed.

**Figure 87 • Configuration Report with Lock**

### Configuration Report for SERDES(s), Fabric DDR and Fabric CCC(s)

Microsemi Corporation - Microsemi Libero Software Release v12.5 (Version 12.900.10.16)

#### PCIE\_SERDES\_IF\_C0\_0/PCIE\_SERDES\_IF\_C0\_0/SERDESIF\_INST/INST\_PCIE\_IP

Register	Field	INIT	Value	Lock INIT	Lock Value(*)
SER_PLL_CONFIG_LOW	PLL_REF_DIVISOR	INIT[120:115]	6'h01	INIT[0]	0 ← Locked
SER_PLL_CONFIG_LOW	PLL_FEEDBACK_DIVISOR	INIT[128:121]	8'h02	INIT[1]	1
SER_PLL_CONFIG_LOW	PLL_OUTPUT_DIVISOR	INIT[131:129]	3'h2	INIT[2]	1
SER PLL CONFIG HIGH	PLL FILTER RANGE	INIT[134:132]	3'h4	INIT[3]	1

## 5 Debug

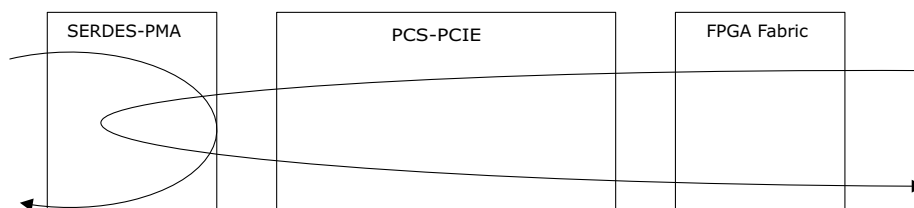
### 5.1 Diagnostic Loopbacks

This chapter covers how to configure the SerDes in loopback to test the signal integrity of the SerDes block. It also details the internal pattern generation and checking capability to assist with system debug.

#### 5.1.1 Serial Loopbacks

Serial loopback modes are specialized configurations of the SerDes data path where the data is folded back to the source. Typically, a specific traffic pattern is transmitted and then compared to check for errors. Loopback test modes fall into two broad categories—Near End and Far End.

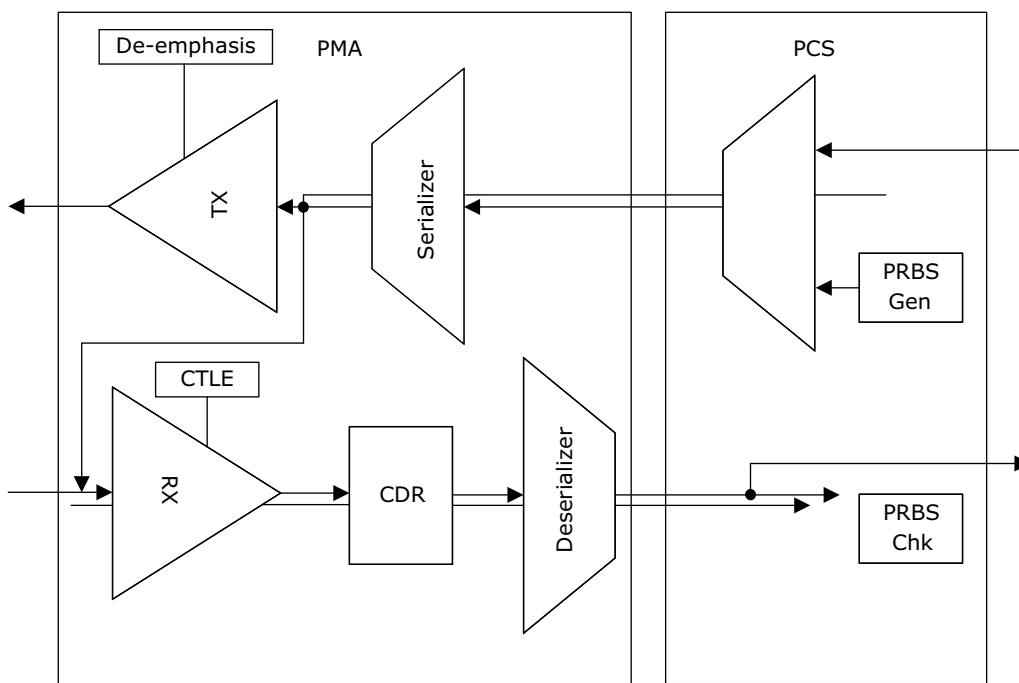
**Figure 88 • Diagnostic Loopbacks**



#### 5.1.2 Near End Serial PMA Loopback

The SerDes block provides support for a serial loopback back onto itself for test purposes. When the LPBK\_EN bit (bit1 of the PRBS\_CTRL register) is set, the serial data is fed back to the CDR block, which extracts the clock and data. Loopback may be operated in full frequency mode (PLLs active) or bypass mode (PLLs bypassed). This mode is useful when used in conjunction with the on-die PRBS data generator and checker. In this scenario, the data can be sent and received without going off-chip.

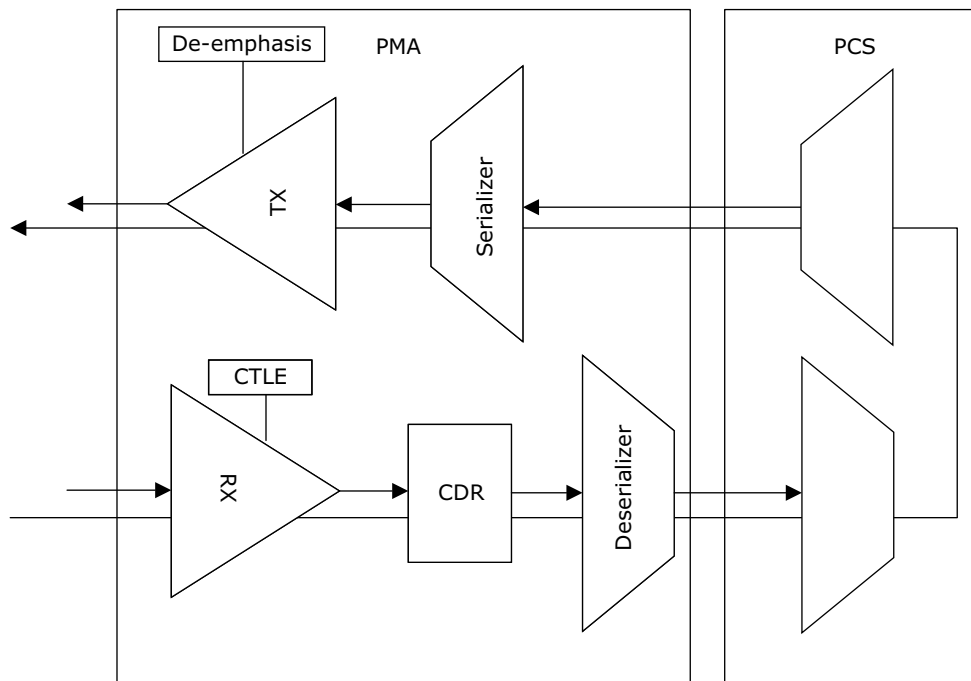
**Figure 89 • Near End Serial PMA Loopback**



### 5.1.3 PCS Far End PMA Receive to Transmit Loopback

The parallel transmit clock recaptures received data in loopback mode (also termed meso\_lpbk, as shown in the following figure) before being sent to the PMA transmitter. The receive is sent back to the transmit and requires no PPM differences between the reference clock used by the transmit and received data. In this case, data is usually sent to the receiver from test equipment such as a Bit error-rate tester (BERT) and brought back out of the device transmit to the BERT to be checked. Typically the tester provides a reference clock to the device. This loopback is available for XAUI and EPCS protocols.

**Figure 90 • Far End PMA RX to TX Loopback**



**Note:** To activate far end PMA receive to transmit loopback program the PCS\_LOOPBACK\_CTRL[2].

**Table 259 • Diagnostic Loopback Support per Protocol**

Loopback	PCIe	XAUI	EPCS
Near End Serial PMA Loopback (Tx to Rx)	Supported	Supported	Supported
PCS Far End PMA RX to Tx Loopback	Not Supported	Supported	Supported

## 5.2 Pseudo-Random Bit Sequences Pattern Generator

Pseudo-random bit sequences (PRBS) are commonly used to test the signal integrity of SerDes. The SerDes block allows pattern generation using the PRBS\_CTRL register. This pattern can be looped back in the PMA and verified as explained in the [Pseudo-Random Bit Sequences Pattern Checker](#), page 179.

The following bits describe the PRBS pattern generation feature:

- PRBS\_GEN: This signal starts the PRBS pattern transmission.
- PRBS\_TYP[1:0]: This signal defines the type of PRBS pattern which is applied. PRBS7 when set to 00b, PRBS11 when set to 01b, PRBS23 when set to 10b, and PRBS31 when set to 11b.

**Table 260 • SerDes Block PRBS Patterns**

Name	Polynomial	Length of Sequence	Descriptions
PRBS-7	$1 + X^6 + X^7$	$2^7 - 1$ bits	Used to test channels which use 8b/10b encoding. Available for PCIe, XAUI, and EPCS protocols. The PRBS-7 polynomial is not necessarily a telecommunications standard but is typically used by test equipment because of its similarity with 8b10b-encoded patterns. PRBS-7 may be bit-wise inverted relative to other devices or visa-versa. Switching the P/N wiring may be needed to properly communicate between devices using the same PRBS-7 polynomial.
PRBS-11	$1 + X^9 + X^{11}$	$2^{11} - 1$ bits	ITU-T recommendation O.150. PRBS-11 is available for EPCS protocols.
PRBS-23	$1 + X^{18} + X^{23}$	$2^{23} - 1$ bits	ITU-T recommendation O.150. PRBS-23 is often used for non-8b/10b encoding scheme. One of the recommended test patterns in the SONET specification. Available for EPCS protocols.
PRBS-31	$1 + X^{28} + X^{31}$	$2^{31} - 1$ bits	ITU-T recommendation O.150. PRBS-31 is often used for non-8b/10b encoding schemes. A recommended PRBS test pattern for 10 GbE. Refer to the IEEE 802.3ae-2002 specification. Available for EPCS protocols.

**Note:** ITU-T Recommendation O.150 provides general requirements for instrumentation for performance measurements on digital transmission equipment.

## 5.3 Pseudo-Random Bit Sequences Pattern Checker

The SerDes block includes a built-in PRBS checker to test the signal integrity of the channel. Using the internal PMA loopback or a complete external path from the transmitter to receiver, this pattern checker allows SerDes to check the four industry-standard PRBS patterns mentioned in [Table 260](#), page 179. The PRBS\_CTRL, page 164 register and PRBS\_ERRCNT, page 165 register allow pattern checking.

- LPBK\_EN: The LPBK\_EN signal, bit 1 of the PRBS\_CTRL register, puts the PMA block in near-end loopback (serial loopback from TX back to RX). PRBS tests can be done using the near-end loopback of the PMA block or using any far-end loopback implemented in the opposite component.
- PRBS\_CHK: The PRBS\_CHK signal, bit 6 of the PRBS\_CTRL register, starts the PRBS pattern checker. Refer to the PRBS\_CTRL, page 164 register for more information.
- PRBS\_ERRCNT: The PRBS\_ERRCNT, page 165 register reports the number of PRBS errors detected when the PRBS test is applied. Refer to PRBS\_ERRCNT, page 165 register for more information.

## 5.4 Custom Pattern Generator

The SerDes block allows generation of a user-defined pattern. There is no pattern checking available for custom patterns, including any of the non-PRBS patterns. The SerDes block allows pattern generation using PRBS related registers. The following bits describe the custom pattern generation feature:

- CUSTOM\_PATTERN[7:0]: The custom pattern registers (register offset 0X190 to 0X1CC) enable programming of a custom pattern. Refer to the custom pattern registers (starting with [CUSTOM\\_PATTERN\\_7\\_0](#), page 166) for more information.
- CUST\_SEL (CUSTOM\_PATTERN\_CTRL[0]): This signal replaces the PRBS data transmitted on the link by the custom pattern. The PRBS\_SEL register must also be set for transmitting the custom pattern on the link.
- CUST\_TYP (CUSTOM\_PATTERN\_CTRL[3:1]): This signal defines whether the custom pattern generated on the link is generated by the custom pattern register or by one of the hard-coded patterns:
  - 00b: Custom pattern register
  - 100b: All-zero pattern (0000...00)
  - 101b: All-one pattern (1111...11)
  - 110b: Alternated pattern (1010...10)
  - 111b: Dual alternated pattern (1100...1100)
- CUST\_SKIP (CUSTOM\_PATTERN\_CTRL[5]): This register is used in RX Word alignment manual mode.
- CUST\_AUTO (CUSTOM\_PATTERN\_CTRL[6]): This allows the word alignment to be performed automatically by a state machine that checks whether the received pattern is word-aligned with the transmitted pattern and to automatically use the PMA CDR PLL skip bit function to find the alignment.
- CUST\_ERROR (CUSTOM\_PATTERN\_CTRL[3:0]): When the custom pattern checker is enabled, this status register reports the number of errors detected by the logic when the custom word aligner is in synchronization. It starts counting only after a first matching pattern has been detected.
- CUST\_SYNC (CUSTOM\_PATTERN\_CTRL[4]): This bit reports that the custom pattern is word-aligned.
- CUST\_STATE (CUSTOM\_PATTERN\_CTRL[7:5]): This register reports the current state of the custom pattern word alignment state machine.

## 5.5 Using SmartDebug Utility for SerDes

The SmartDebug utility included with the Libero design software provides SerDes access that assists FPGA and the board designers to perform SerDes real-time signal integrity testing and tuning in a system including SerDes control and test capabilities to assist with debugging high-speed serial designs with no extra steps.

The SmartDebug JTAG interface extends access to configure, control and observe SerDes operations and is accessible in every SerDes design. Users simply implement their design with the Libero System Builder to incorporate the SerDes block enabling SerDes access from the SmartDebug tool set. This quickly enables designers to explore configuration options without going through FPGA recompilation or making changes to the board. GUI displays real-time system and lane status information. SerDes configurations are supported with TCL scripting allowing access to the entire register map for real-time customized tuning. *TU0530: SmartFusion2 and IGLOO2 SmartDebug Hardware Design Debug Tools Tutorial* demonstrates the tools capabilities. For more information about SmartDebug utility, see *SmartDebug for Software User's Guide*.