PCIe x8 IP Core

IP Version: v3.0.0

User Guide

FPGA-IPUG-02243-1.6

December 2025

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Contents 3
Acronyms in This Document 17
1. Introduction 18
1.1. Overview of the IP 18
1.2. Quick Facts 18
1.3. IP Support Summary 19
1.4. Features 20
1.4.1. Hard IP Limitations 21
1.4.2. Soft IP 21
1.5. Licensing and Ordering Information 21
1.6. Hardware Support 21
1.7. Speed Grade Supported 22
1.8. Naming Conventions 22
1.8.1. Nomenclature 22
1.8.2. Signal Names 22
2. Functional Description 23
2.1. PCIe IP Architecture Overview 23
2.2. Clocking 25
2.2.1. Clocking Overview 25
2.3. Reset 27
2.3.1. Reset Overview 27
2.3.2. Clock and Reset Sequence 28
2.4. Protocol Layers 28
2.4.1. ECC and Parity Data Path Protection 29
2.4.2. Error Handling 30
2.4.3. LTSSM State 32
2.5. PHY Equalization (8 GT/s) 36
2.5.1. Equalization Process 36
2.5.2. Equalization Time Limit 36
2.5.3. Equalization Methods 36
2.5.4. Equalization Quality 43
2.6. Multi-Function Support 43
2.7. Power Management 43
2.7.1. Power Management Supported by PCIe IP Core 43
2.7.2. Configuring Core to Support Power Management 44
2.7.3. ASPM L0s 44
2.7.4. ASPM L1s 45
2.8. DMA Support (Hardened) 46
2.8.1. DMA Scatter-Gather and Status Queues 46
2.8.2. DMA Channel Programming and Operation 49
2.8.3. DMA Performance 51
2.8.4. DMA Bypass Interface 52
2.9. DMA Support 52
2.9.1. DMA Overview 52
2.9.2. DMA Descriptor 52
2.9.3. DMA Registers 56
2.9.4. DMA Transaction (AXI-MM) 65
2.10. Non-DMA Support 68
2.10.1. Non-DMA Overview 68
2.10.2. Non-DMA Write 70
2.10.3. Non-DMA Read 70
2.11. Interrupts 70

2.11.1. Generation of the Interrupts 70
2.11.2. Legacy Interrupt 71
2.11.3. MSI Interrupt 71
2.11.4. MSI-X Interrupt 73
2.12. PCIe Endpoint Core Buffers 74
2.12.1. PCI Express Credits 75
2.12.2. Max Payload Size 75
2.13. Hard IP Interface 76
2.13.1. PHY Interface 76
2.13.2. TLP TX/RX Interface 76
2.13.3. LMMI Interface 86
2.14. Soft IP Interface 87
2.14.1. Data Interface Conversion 87
2.14.2. Register Interface Conversion 101
2.15. Multi-Protocol Support 101
3. IP Parameter Description 102
3.1. General 102
3.2. Optional Port 104
3.3. ASPM Capability 104
3.4. DMA/Bridge Mode Support 104
3.5. Flow Control Update 105
3.6. Function 106
3.6.1. Configuration 106
3.6.2. Base Address Register (BAR) [0 to 5] 107
3.6.3. Legacy Interrupt 108
3.6.4. MSI Capability 109
3.6.5. MSI-X Capability 109
3.6.6. Device Serial Number Capability 110
3.6.7. PCI Express Capability 111
3.6.8. Advance Error Reporting Capability 111
3.6.9. ATS Capability 112
3.6.10. Atomic OP Capability 112
3.6.11. Latency Tolerance Reporting Capability 113
3.6.12. Power Budgeting Capability 113
3.6.13. Dynamic Power Allocation Capability 114
4. Signal Description 116
4.1. Clock and Reset Interface 116
4.2. PHY Interface 117
4.3. Transaction Layer Interface 117
4.3.1. TLP Transmit Interface 117
4.3.2. TLP Receive Interface 120
4.4. Lattice Memory Mapped Interface (LMMI) 124
4.5. Power Management Interface 125
4.6. AXI-Lite Configuration Interface 125
4.7. AXI-Stream Data Interface 126
4.7.1. AXI-Stream Transmitter Interface Port Descriptions 126
4.7.2. AXI-Stream Receiver Interface Port Descriptions 127
4.8. AXI Data Interface (Hardened DMA) 128
4.9. AXI Data Interface (DMA) 134
4.10. AXI Data Interface (Bridge Mode) 135
4.11. Function Level Reset (FLR) Interface 137
5. Register Description 139
5.1. Hard IP Core Configuration and Status Registers 139
5.1.1. EP Configuration Settings 140

5.1.2. mgmt_tlb (0x4_2000) 141
5.1.3. mgmt_ptl (0x4_3000) 206
5.1.4. mgmt_ftl (0x4_4000) 218
5.1.5. mgmt_ftl_mf[7:1] (0x4_5000,0x4_6000,0x4_7000,0x4_8000,0x4_9000,0x4_A000,0x4_B000) 243
5.2. PCI Express Configuration Space Registers 249
5.2.1. Type 00 Configuration Header 249
5.2.2. Type 01 Configuration Header 250
5.2.3. Capability and Extended Capability Address Locations 250
5.2.4. Type 00 Configuration Registers 251
5.2.5. PCI Express Capability 252
5.2.6. Power Management Capability 258
5.2.7. MSI-X Capability 259
5.2.8. MSI Capability 260
5.2.9. Advanced Error Reporting Extended Capability 261
5.2.10. ARI Extended Capability 263
5.2.11. Vendor-Specific Extended Capability 264
5.2.12. Secondary PCI Express Extended Capability 265
5.2.13. ATS Extended Capability 266
5.2.14. DSN Extended Capability 266
5.2.15. Resizable BAR Capability 266
5.2.16. Power Budgeting Capability 267
5.2.17. Dynamic Power Allocation Capability 268
5.2.18. L1 PM Substates Extended Capability 268
5.2.19. Latency Tolerance Reporting Capability 269
5.3. DMA Configuration Space Registers 269
6. Example Design 276
6.1. Example Design Supported Configuration 276
6.2. Overview of the Example Design and Features 277
6.3. Example Design Components 278
6.3.1. DMA Design (Hardened DMA) 278
6.3.2. DMA Design (DMA) 281
6.3.3. Non-DMA Design (TLP Interface) 283
6.3.4. Non-DMA Design (Bridge Mode) 287
6.3.5. PDC Settings for Hardware Example Design 288
6.3.6. Running the Hardware Using the Production Driver 288
6.4. Simulating the Example Design 289
6.4.1. Running Functional Simulation 290
6.5. Design Test Case Examples 297
6.5.1. DMA Design 297
6.5.2. Non-DMA Design 298
6.6. Debugging Example Design Issues 299
6.6.1. Signals to Debug 299
6.7. Limitations of the Example Design 301
7. Designing with the IP 302
7.1. Generating and Instantiating the IP 302
7.1.1. Generated Files and File Structure 309
7.1.2. Design Implementation 309
7.1.3. Timing Constraints 309
7.1.4. Multi-Seed Timing Closure 311
8. Debugging 312
8.1. Debug Methods 312
8.1.1. Debug Flow Charts 312
8.1.2. Internal Register Read for Debug 318
8.1.3. PCIe Loopback Test 318

9. Design Consideration 319
9.1. DMA Based Design 319
9.2. Non-DMA Based Design 319
Appendix A. Resource Utilization 320
References 321
Technical Support Assistance 322
Revision History 323

Figures

Figure 2.1. Lattice PCIe x8 IP Core Block Diagram 23
Figure 2.2. Lattice 8-lane SERDES/PCS + PCIe Hard-IP 24
Figure 2.3. Lattice PCIe x8 Core Hard IP 24
Figure 2.4. PCIe IP Clock Domain Block Diagram with External PLL 25
Figure 2.5. PCIe IP Clock Domain Block Diagram for TLP Interface 26
Figure 2.6. Reset Signals in Lattice PCIe IP Core 27
Figure 2.7. Clock and Reset Sequence Diagram 28
Figure 2.8. Source Scatter-Gather Queue Elements 47
Figure 2.9. Destination Scatter-Gather Queue Elements 48
Figure 2.10. Status Queue Elements 49
Figure 2.11. F2H Data Transfer 66
Figure 2.12. H2F Data Transfer 67
Figure 2.13. Non-DMA Application Data Flow – TLP Interface 68
Figure 2.14. Non-DMA Application Data Flow – AXI-Stream Interface 68
Figure 2.15. Non-DMA Application Data Flow – AXI-MM Interface (Bridge Mode) 69
Figure 2.16. Non-DMA Application Data Flow – AXI-Lite Interface (Bridge Mode) 69
Figure 2.17. Non-DMA Write Operation (TLP Data Interface) 70
Figure 2.18. Non-DMA Read Operation (TLP Data Interface) 70
Figure 2.19. MSI Capability Structure Variant 72
Figure 2.20. MSI Capability Structure Variant 72
Figure 2.21. MSI-X Capability Structure Variant 73
Figure 2.22. MSI-X Table Entries 74
Figure 2.23. Pending Bit Array 74
Figure 2.24. TLP Memory Request Header Format for 64-bit Addressing of Memory 76
Figure 2.25. TLP Memory Request Header Format for 32-bit Addressing of Memory 77
Figure 2.26. TLP Memory Read Operation for Link0 (TLP Interface Width of 512 Bits, 256 Bits, and 128 Bits) 79
Figure 2.27. TLP Memory Read Operation for Link0 (TLP Interface Width of 64 Bits) 80
Figure 2.28. Minimum link0_tx_ready_o Timing Diagram 80
Figure 2.29. Wait State of link0_tx_ready_o Timing Diagram 81
Figure 2.30. TLP Packet formation by the Lattice PCIe IP core 82
Figure 2.31 TLP Memory Write Operation for Link0 (TLP Interface Width of 512 Bits, 256 Bits, 128 Bits, and 64 Bits) 83
Figure 2.32. Minimum link0_rx_ready_i Timing Diagram 83
Figure 2.33. Wait State of link0_rx_ready_i Timing Diagram 84
Figure 2.34. LMMI Write Operation 86
Figure 2.35. LMMI Read Operation 87
Figure 2.36. AXI-Stream Data Interface, AXI-Lite Register Interface 87
Figure 2.37. PCIe to AXI-Stream Transaction for x1 88
Figure 2.38. PCIe to AXI-Stream Transaction for x2 88
Figure 2.39. PCIe to AXI-Stream Transaction for x8 88
Figure 2.40. AXI-Stream to PCIe Transaction for x1 89
Figure 2.41. AXI-Stream to PCIe Transaction for x2 89
Figure 2.42. AXI-Stream to PCIe Transaction for x8 89
Figure 2.43. Bridge Mode Enablement (General Tab) 90
Figure 2.44. Bridge Mode Enablement (DMA/Bridge Mode Support Tab) 91
Figure 2.45. User Interrupt Pins Example Waveform 92
Figure 2.46. PCIe Bifurcation 101
Figure 3.1. Attributes in the General Tab 102
Figure 3.2. Attributes in the Optional Port Tab 104
Figure 3.3. Attributes in the ASPM Capability Tab 104
Figure 3.4. Attributes in the DMA/Bridge Mode Support Tab 104
Figure 3.5. Attributes in the Flow Control Update Tab 105
Figure 3.6. Attributes in Function Configuration Tab 106

Figure 3.7. Attributes in BAR Tab 107
Figure 3.8. Attributes in Legacy Interrupt 108
Figure 3.9. Attributes in MSI Capability 109
Figure 3.10. Attributes in MSI-X Capability 109
Figure 3.11. Attributes in Device Serial Number Capability 110
Figure 3.12. Attributes in PCIe Capability 111
Figure 3.13. Attributes in Advance Error Reporting Capability 111
Figure 3.14. Attributes in ATS Capability 112
Figure 3.15. Attributes in Atomic OP Capability 112
Figure 3.16. Attributes in Latency Tolerance Reporting Capability 113
Figure 3.17. Attributes in Power Budgeting Capability 113
Figure 3.18. Attributes in Dynamic Allocation Capability 114
Figure 6.1. PCIe x8 IP Example Design Block Diagram 277
Figure 6.2. Components within the Hardened DMA Design 278
Figure 6.3. IP Configuration for Hardened DMA Example Design – General Tab 279
Figure 6.4. IP Configuration for Hardened DMA Example Design – Link 0: Function 0 Tab (Part 1) 279
Figure 6.5. IP Configuration for Hardened DMA Example Design – Link 0: Function 0 Tab (Part 2) 280
Figure 6.6. File List View of the Created Hardened DMA Example Design 281
Figure 6.7. Components within DMA Example Design 281
Figure 6.8. File List View of the Created DMA Example Design 282
Figure 6.9. Components within Non-DMA Design (TLP Interface) 283
Figure 6.10. Non-DMA Design Data Flow 284
Figure 6.11. Non-DMA Example Design (TLP Mode) Settings (General Tab) 285
Figure 6.12. Non-DMA Example Design (TLP Mode) Settings (Link 0: Function 0 Tab) 285
Figure 6.13. WIDTH Parameter Adjustment in the example_design_top Module According to Bifurcation Selection 286
Figure 6.14. File List View of the Created Non-DMA Example Design 286
Figure 6.15. Components within NON-DMA Design (Bridge Mode) 287
Figure 6.16. File List View of the Created Bridge Mode Example Design 288
Figure 6.17. PCIe x8 IP Example Design Flowchart 289
Figure 6.18. Testbench Files 290
Figure 6.19. Project Naming 291
Figure 6.20. Simulation Top Module 291
Figure 6.21. Simulation Setting 292
Figure 6.22. Expected Log Printing 292
Figure 6.23. Testbench Files 293
Figure 6.24. Project Naming 293
Figure 6.25. Simulation Top Module 294
Figure 6.26. Simulation Setting 294
Figure 6.27. Transcript Log Printing 295
Figure 6.28. System Environment Variable for Windows 296
Figure 6.29. Command of QuestaSim Pro 296
Figure 6.30. Expected Log Printing 296
Figure 6.31. Simulation Waveform 297
Figure 7.1. Module/IP Block Wizard 302
Figure 7.2. IP Configuration for Non-DMA 303
Figure 7.3. IP Configuration for Hardened DMA (General Tab) 303
Figure 7.4. IP Configuration for Hardened DMA (Link 0: Function 0 Tab) Part 1 304
Figure 7.5. IP Configuration for Hardened DMA (Link 0: Function 0 Tab) Part 2 304
Figure 7.6. IP Configuration for Hardened DMA (Link 0: Function 0 Tab) Part 3 305
Figure 7.7. Check Generated Result 305
Figure 7.8. Set IP Top Level Unit 306
Figure 7.9. Select to Add Constraint File 307
Figure 7.10. Select constraint.pdc from the Eval Folder 308
Figure 7.11. Project Compile Done 308

Figure 7.12. Bitstream Not Generated due to the Virtual I/O Setting 309
Figure 7.13. Timing Constraint File (.ldc) for the PCIe x8 IP 310
Figure 7.14. PLL IP Configuration for Input Clock of 100 MHz 310
Figure 7.15. Timing Constraints for PCIe x8 IP Example 311
Figure 7.16. Placement Iteration Setup on Radiant under Strategies Tab 311
Figure 8.1. Hardware Detection Failure Debugging Flow 312
Figure 8.2. Link Training Issue Debugging Flow 313
Figure 8.3. Checkbox to Enable LTSSM Polling 313
Figure 8.4. Data Transfer Issue Debugging Flow 316
Figure 8.5. Debugging the FPGA Configuration Issues Flow 317

Tables

Table 1.1. Summary of the PCIe x8 IP 18
Table 1.2. PCIe x8 IP Support Readiness 19
Table 1.3. Lattice PCIe IP Core Supported Speed Grade 22
Table 2.1. General PCI Express Error List 30
Table 2.2. Physical Layer Error List 30
Table 2.3. Data Link Layer Error List 31
Table 2.4. Transaction Layer Error List 31
Table 2.5. LTSSM State Definition 32
Table 2.6. RX L0s State Description 35
Table 2.7. Preset to Coefficient Conversion 37
Table 2.8. Simulation Data Throughput for FPGA-to-Host (F2H) Transfer 51
Table 2.9. Simulation Data Throughput for Host-to-FPGA (H2F) Transfer 51
Table 2.10. Hardware Data Throughput for FPGA-to-Host (F2H) Transfer 51
Table 2.11. Hardware Data Throughput for Host-to-FPGA (H2F) Transfer 52
Table 2.12. Descriptor Format 53
Table 2.13. DESC_CTRL (0x00) 53
Table 2.14. DMA_LEN (0x04) 53
Table 2.15. NEXT_DESC_ADDR_LO (0x08) 53
Table 2.16. NEXT_DESC_ADDR_HI (0x0C) 54
Table 2.17. SRC_ADDR_LO (0x10) 54
Table 2.18. SRC_ADDR_HI (0x14) 54
Table 2.19. DEST_ADDR_LO (0x18) 54
Table 2.20. DEST_ADDR_HI (0x1C) 54
Table 2.21. First Descriptor Chunk Fetching through MRd TLP 54
Table 2.22. Second Descriptor Chunk Fetching through MRd TLP 55
Table 2.23. Third Descriptor Chunk Fetching through MRd TLP: 56
Table 2.24. Access Types 56
Table 2.25. PCIe DMA Register Group 56
Table 2.26. H2F_DMA_CTRL (0x0000) 57
Table 2.27. H2F_DMA_STS (0x000C) 57
Table 2.28. H2F_DMA_INT_MASK (0x0010) 58
Table 2.29. H2F_CPLT_DESC_COUNT (0x0018) 59
Table 2.30. F2H_DMA_CTRL (0x0100) 59
Table 2.31. F2H_DMA_STS (0x010C) 59
Table 2.32. F2H_DMA_INT_MASK (0x0110) 60
Table 2.33. F2H_CPLT_DESC_COUNT (0x0118) 61
Table 2.34. H2F_DESC_ADDR_LOW (0x0200) 61
Table 2.35. H2F_DESC_ADDR_HIGH (0x0204) 61
Table 2.36. H2F_CONT_REMAIN (0x0208) 61
Table 2.37. F2H_DESC_ADDR_LOW (0x0300) 62
Table 2.38. F2H_DESC_ADDR_HIGH (0x0304) 62
Table 2.39. F2H_CONT_REMAIN (0x0308) 62
Table 2.40. INT_MODE (0x0400) 62
Table 2.41. H2F_MSI_VEC (0x0404) 62
Table 2.42. F2H_MSI_VEC (0x0408) 63
Table 2.43. USR_MSI_VEC_P1 (0x040C) 63
Table 2.44. USR_MSI_VEC_P2 (0x0410) 63
Table 2.45. USR_MSI_VEC_P3 (0x0414) 64
Table 2.46. USR_MSI_VEC_P4 (0x0418) 65
Table 2.47. GENERAL_STS (0x0500) 65
Table 2.48. Register Access for Different Data Interfaces 69
Table 2.49. Base Address to Enable Interrupt 71

Table 2.50. Legacy Interrupt Register 71
Table 2.51. TLP Header Field (Memory Transaction) 77
Table 2.52. Data Byte Order for 32-bit Memory Read TLP 84
Table 2.53. Data Byte Order for 64-bit Memory Read TLP 85
Table 2.54. MSI Advertised Capabilities 92
Table 2.55. MSI-X Table Offsets 92
Table 2.56. MSI-X PBA Offsets 93
Table 2.57. MSI-X Advertised Capabilities 93
Table 2.58. Access Types 94
Table 2.59. USR_MSI_VEC_P1 (0x040C) 94
Table 2.60. USR_MSI_VEC_P2 (0x0410) 95
Table 2.61. USR_MSI_VEC_P3 (0x0414) 96
Table 2.62. USR_MSI_VEC_P4 (0x0418) 98
Table 2.63. USR0_MSIX_TABLE (0x8000) 99
Table 2.64. USR1_MSIX_TABLE (0x8010) 99
Table 2.65. PBA_TABLE (0xC000) 100
Table 3.1. General Tab Attributes Description 102
Table 3.2. Optional Port Attributes 104
Table 3.3. DMA/ Bridge Mode Support Attributes 105
Table 3.4. Flow Control Attributes 106
Table 3.5. Function Configuration Tab Attributes 106
Table 3.6. BAR Tab Attributes 107
Table 3.7. Legacy Interrupt Attribute Descriptions 108
Table 3.8. MSI Capability Attributes 109
Table 3.9. MSI-X Capability Attributes 110
Table 3.10. Device Serial Number Capability Attributes 110
Table 3.11. PCIe Capability Attributes 111
Table 3.12. Advance Error Reporting Capability Attributes 111
Table 3.13. ATS Capability Attribute Description 112
Table 3.14. Atomic OP capability Attributes 112
Table 3.15. Latency Tolerance Reporting Capability Attributes 113
Table 3.16. Power Budgeting Capability Attributes 113
Table 3.17. Dynamic Allocation capability Attributes 114
Table 3.18. Function 1-7 Tab 115
Table 4.1. Clock and Reset Ports 116
Table 4.2. PHY Interface Descriptions 117
Table 4.3. TLP Transmit Interface Ports 117
Table 4.4. TLP Transmit Credit Interface Ports 119
Table 4.5. TLP Receive Interface Ports 120
Table 4.6. TLP Receive Credit Interface Ports 123
Table 4.7. Lattice Memory Mapped Interface Ports 124
Table 4.8. Transaction Pending Ports 125
Table 4.9. AXI-Lite Configuration Interface Ports 125
Table 4.10. AXI-Stream Transmitter Interface Ports 126
Table 4.11. AXI-Stream Receiver Interface Ports 127
Table 4.12. DMA AXI Manager Write Interface 128
Table 4.13. DMA AXI Manager Read Interface 130
Table 4.14. DMA AXI Subordinate Write Interface 132
Table 4.15. DMA AXI Subordinate Read Interface 133
Table 4.16. DMA AXI Interrupt Interface 133
Table 4.17. AXI-MM Manager Interface (DMA) 134
Table 4.18. AXI-MM Manager Write Interface (Bridge Mode) 135
Table 4.19. AXI-Lite Manager Interface (Bridge Mode) 136
Table 4.20. Function Level Reset Interface 137

Table 5.1. Register Access Abbreviations 139
Table 5.2. Hard PCIe Core Register Mapping 139
Table 5.3. CSR Values Recommended for EP Applications 140
Table 5.4. ltssm_simulation Register 0x0 141
Table 5.5. ltssm_cfg_lw_start Register 0x34 141
Table 5.6. ltssm_latch_rx Register 0x38 142
Table 5.7. ltssm_cfg Register 0x3c 142
Table 5.8. ltssm_port_type Register 0x40 144
Table 5.9. ltssm_ds_link Register 0x44 144
Table 5.10. ltssm_detect_quiet Register 0x48 145
Table 5.11. ltssm_rx_det Register 0x4c 145
Table 5.12. ltssm_nfts Register 0x50 145
Table 5.13. ltssm_ds_initial_auto Register 0x54 146
Table 5.14. ltssm_select_deemphasis Register 0x58 146
Table 5.15. ltssm_beacon Register 0x5c 147
Table 5.16. ltssm_mod_cpl Register 0x60 147
Table 5.17. ltssm_rx_elec_idle Register 0x64 147
Table 5.18. ltssm_compliance_toggle Register 0x68 148
Table 5.19. ltssm_prevent_rx_ts_entry_to Register 0x6c 149
Table 5.20. ltssm_link Register 0x80 149
Table 5.21. ltssm_ltssm Register 0x84 150
Table 5.22. ltssm_rx_l0s Register 0x88 152
Table 5.23. l0_to_rec Register 0x8c 153
Table 5.24. ltssm_rx_detect Register 0x90 154
Table 5.25. ltssm_configured Register 0x94 155
Table 5.26. ltssm_direct_to_detect Register 0x98 155
Table 5.27. ltssm_equalization Register 0x9c 155
Table 5.28. ltssm_crosslink Register 0xa0 156
Table 5.29. Physical Layer Tx Underflow Error Status Register – 0xa4 156
Table 5.30. Physical Lane Rx Status Registers 156
Table 5.31. pl_rx0 Register 0xa8 – Lane Rx Status 0 Register 156
Table 5.32. pl_rx1 Register 0xac – Lane Rx Status 1 158
Table 5.33. pl_rx2 Register 0xb0 – Lane Rx Status 2 161
Table 5.34. pl_rx3 Register 0xb4 – Lane Rx Status 3 164
Table 5.35. pl_rx4 Register 0xb8 – Lane Rx Status 4 166
Table 5.36. debugself_crosslink Register 0xc0 169
Table 5.37. debug_rx_det Register 0xc4 169
Table 5.38. debug_force_tx Register 0xc8 170
Table 5.39. debug_direct_scramble_off Register 0xcc 170
Table 5.40. debug_force_scramble_off_fast Register 0xd0 170
Table 5.41. balign Register 0xd4 171
Table 5.42. debug_pipe_rx Register 0xe0 172
Table 5.43. debug_direct_to_loopback Register 0x100 172
Table 5.44. debug_loopback_control Register 0x104 172
Table 5.45. debug_loopback_master_5g Register 0x108 173
Table 5.46. debug_loopback_slave_5g Register 0x10c 174
Table 5.47. debug_loopback_master_8g_deemph Register 0x110 174
Table 5.48. debug_loopback_slave_8g_deemph Register 0x114 175
Table 5.49. debug_direct_to_loopback_status Register 0x118 175
Table 5.50. debug_loopback_err_reset Register 0x11c 176
Table 5.51. debug_loopback_err Register 0x120 176
Table 5.52. phy_control Register 0x140 176
Table 5.53. phy_control_8g Register 0x144 176
Table 5.54. phy_eq_tx_override Register 0x148 F 177

Table 5.55. phy_eq_tx_max Register 0x14c 177
Table 5.56. phy_eq_tx_force Register 0x150 178
Table 5.57. phy_preset_to_coef_conv_control Register 0x15c 178
Table 5.58. phy_preset_conv_tab_pre Register 0x160 179
Table 5.59. phy_preset_conv_tab_post Register 0x170 179
Table 5.60. eq_control Register 0x180 179
Table 5.61. eq_ts_control Register 0x184 180
Table 5.62. eq_reduced_swing Register 0x188 182
Table 5.63. eq_method Register 0x1bc 182
Table 5.64. eq_fmerit_control Register 0x1c0 182
Table 5.65. eq_preset_method_control Register 0x1c4 183
Table 5.66. eq_alg_method_control Register 0x1c8 183
Table 5.67. eq_table_method_control Register 0x1cc 184
Table 5.68. eq_table_method_table Register 0x1d0 184
Table 5.69. eq_updn_control Register 0x240 185
Table 5.70. eq_firmware_control Register 0x280 186
Table 5.71. eq_pre_cursor Register 0x290 187
Table 5.72. eq_post_cursor Register 0x2a0 187
Table 5.73. eq_status Register 0x2c0 188
Table 5.74. eq_status_error Register 0x2c4 189
Table 5.75. eq_status_preset_coef Register 0x2c8 189
Table 5.76. eq_status_feedback_fom Register 0x2d0 189
Table 5.77. eq_status_feedback_dir Register 0x2e0 190
Table 5.78. eq_status_remote_fs Register 0x2e8 190
Table 5.79. eq_status_remote_lf Register 0x2f4 190
Table 5.80. eq_status_remote_precursor Register 0x300 190
Table 5.81. eq_status_remote_postcursor Register 0x30c 191
Table 5.82. pl_rx Register 0x33c 191
Table 5.83. pl_16g Register 0x340 191
Table 5.84. pl_tx_skp Register 0x344 192
Table 5.85. pl_tx_debug Register 0x348 193
Table 5.86. pl_ctrl Register 0x34c 193
Table 5.87. pl_ts_matching Register 0x350 193
Table 5.88. dl_retry_timeout Register 0x380 194
Table 5.89. dl_ack_timeout_div Register 0x384 194
Table 5.90. dl_tx_ctrl Register 0x38c 195
Table 5.91. dl_ctrl Register 0x390 196
Table 5.92. dl_stat Register 0x394 199
Table 5.93. dl_ack_to_nak Register 0x398 202
Table 5.94. dl_inject Register 0x39c 202
Table 5.95. dllp_inject Register 0x3a0 203
Table 5.96. eq_status_table_control Register 0x3d8 204
Table 5.97. eq_status_table_info Register 0x3dc 204
Table 5.98. eq_status_table Register 0x3e0 204
Table 5.99. eq_capture_sel Register 0x3f0 205
Table 5.100. eq_capture Register 0x3f4 205
Table 5.101. phy_eq_tx_force_per_lane Register 0x400 205
Table 5.102. phy_eq_tx_force_per_lane_8g_pre Register 0x404 206
Table 5.103. phy_eq_tx_force_per_lane_8g_post Register 0x410 206
Table 5.104. Simulation Register 0x0 206
Table 5.105. pm_aspm_l0s Register 0x40 206
Table 5.106. pm_aspm_l1 Register 0x50 207
Table 5.107. pm_aspm_l1_min Register 0x54 207
Table 5.108. pm_l1 Register 0x60 207

Table 5.109. pm_l1_min Register 0x64 207
Table 5.110. pm_l1pmss Register 0x68 208
Table 5.111. pm_l2 Register 0x70 208
Table 5.112. pm_pme_to_ack_ep Register 0x80 208
Table 5.113. pm_pme_to_ack_ds Register 0x84 208
Table 5.114. pm_pme Register 0x88 209
Table 5.115. pm_status Register 0x90 209
Table 5.116. vc_rx_c Register 0x108 210
Table 5.117. vc_rx_adv Register 0x10c 210
Table 5.118. vc_rx_control Register 0x110 210
Table 5.119. vc_rx_status Register 0x114 211
Table 5.120. vc_rx_credit_status_cfg Register 0x120 212
Table 5.121. vc_rx_credit_status_p Register 0x124 212
Table 5.122. vc_rx_credit_status_n Register 0x128 212
Table 5.123. vc_rx_credit_status_c Register 0x12c 212
Table 5.124. vc_rx_f_oc_update_timer Register 0x130 213
Table 5.125. vc_rx_p_flow_ctrl Register 0x134 213
Table 5.126. vc_rx_n_flow_ctrl Register 0x138 213
Table 5.127. vc_rx_alloc_size Register 0x140 214
Table 5.128. vc_rx_alloc_p Register 0x144 214
Table 5.129. vc_rx_alloc_n Register 0x148 214
Table 5.130. vc_rx_alloc_c Register 0x14c 215
Table 5.131. vc_rx_alloc_error Register 0x150 215
Table 5.132. vc_tx_np_fifo Register 0x180 216
Table 5.133. vc_tx_status Register 0x184 216
Table 5.134. vc_tx_credit_status_p Register 0x190 216
Table 5.135. vc_tx_credit_status_n Register 0x194 217
Table 5.136. vc_tx_credit_status_c Register 0x198 217
Table 5.137. vc_tx_credit_cleanup Register 0x19c 217
Table 5.138. tlp_tx Register 0x1c4 218
Table 5.139. fc_credit_init Register 0x1c8 218
Table 5.140. simulation Register 0x0 218
Table 5.141. decode Register 0x10 219
Table 5.142. decode_t1 Register 0x14 221
Table 5.143. tlp_processing Register 0x18 222
Table 5.144. Initial Register 0x20 222
Table 5.145. cfg Register 0x30 223
Table 5.146. ds_port Register 0x34 223
Table 5.147. us_port Register 0x38 224
Table 5.148. id1 Register 0x40 224
Table 5.149. id2 Register 0x44 224
Table 5.150. id3 Register 0x48 224
Table 5.151. Cardbus Register 0x4c 225
Table 5.152. Interrupt Register 0x50 225
Table 5.153. bar0 Register 0x60 225
Table 5.154. bar1 Register 0x64 225
Table 5.155. bar2 Register 0x68 226
Table 5.156. bar3 Register 0x6c 226
Table 5.157. bar4 Register 0x70 226
Table 5.158. bar5 Register 0x74 226
Table 5.159. exp_rom Register 0x78 227
Table 5.160. pcie_cap Register 0x80 227
Table 5.161. pcie_dev_cap Register 0x84 228
Table 5.162. pcie_link_cap Register 0x88 229

Table 5.163. pcie_link_stat Register 0x8c 230
Table 5.164. pcie_slot_cap Register 0x90 230
Table 5.165. pcie_dev_cap2 Register 0x98 232
Table 5.166. pcie_link_ctl2 Register 0xa0 233
Table 5.167. pm_cap Register 0xc0 234
Table 5.168. pm Register 0xc4 235
Table 5.169. pm_aux Register 0xc8 235
Table 5.170. ari_cap Register 0xe0 236
Table 5.171. aer_cap Register 0x100 236
Table 5.172. msi_cap Register 0xe8 237
Table 5.173. msix_cap Register 0xf0 238
Table 5.174. msix_table Register 0xf4 238
Table 5.175. msix_pba Register 0xf8 238
Table 5.176. vsec_cap Register 0x110 239
Table 5.177. sris_cap Register 0x120 239
Table 5.178. dsn_cap Register 0x130 239
Table 5.179. dsn_serial Register 0x134 240
Table 5.180. pwr_budget_cap Register 0x150 240
Table 5.181. dpa_cap Register 0x158 240
Table 5.182. dpa_xlcy Register 0x15c 241
Table 5.183. dpa_alloc Register 0x160 241
Table 5.184. ltr_cap Register 0x180 242
Table 5.185. l1pmss_cap Register 0x188 242
Table 5.186. atomic_op_cap Register 0x1cc 243
Table 5.187. Base Address for mgmt_ftl_mf 243
Table 5.188. Function Register 0x8 244
Table 5.189. us_port Register 0x38 244
Table 5.190. id1 Register 0x40 244
Table 5.191. id2 Register 0x44 244
Table 5.192. id3 Register 0x48 245
Table 5.193. Cardbus Register 0x4c 245
Table 5.194. Interrupt Register 0x50 245
Table 5.195. bar0 Register 0x60 245
Table 5.196. bar1 Register 0x64 246
Table 5.197. bar2 Register 0x68 246
Table 5.198. bar3 Register 0x6c 246
Table 5.199. bar4 Register 0x70 246
Table 5.200. bar5 Register 0x74 246
Table 5.201. exp_rom Register 0x78 247
Table 5.202. msi_cap Register 0xe8 247
Table 5.203. msix_cap Register 0xf0 247
Table 5.204. msix_table Register 0xf4 248
Table 5.205. msix_pba Register 0xf8 248
Table 5.206. dsn_cap Register 0x130 248
Table 5.207. dsn_serial Register 0x134 249
Table 5.208. Type 00 Configuration Header 249
Table 5.209. Type 01 Configuration Header 250
Table 5.210. Capability and Extended Capability Items 250
Table 5.211. Type 00 Configuration Registers 251
Table 5.212. PCI Express Capability 252
Table 5.213. Power Management Capability 258
Table 5.214. MSI-X Capability 259
Table 5.215. MSI Capability 260
Table 5.216. Advanced Error Reporting Extended Capability 261

Table 5.217. ARI Extended Capability 263
Table 5.218. Vendor-Specific Extended Capability 264
Table 5.219. Secondary PCI Express Extended Capability 265
Table 5.220. ATS Extended Capability 266
Table 5.221. DSN Extended Capability 266
Table 5.222. Resizable BAR Capability 266
Table 5.223. Power Budgeting Capability 267
Table 5.224. Dynamic Power Allocation (DPA) Capability 268
Table 5.225. L1 PM Substates Extended Capability 268
Table 5.226. Latency Tolerance Reporting (LTR) Capability 269
Table 5.227. DMA Configuration Space Registers 270
Table 6.1. PCIe x8 IP Configuration Supported by the Example Design 276
Table 6.2. Non-DMA Design Status Ports Description 298
Table 6.3. DMA Signals to Debug Description 299
Table 6.4. AXI-Lite Bridge Mode to Debug Description 300
Table 6.5. Non-DMA Signals to Debug Description 301
Table 7.1. Generated File List 309
Table 8.1. PCIe LTSSM State and Sub-State Definition 314
Table A.1. Lattice PCIe IP Core Resource Utilization 320

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
ASPM	Active State Power Management
AXI	Advanced Extensible Interface
AXI-Lite	Advanced Extensible Interface Lite
AXI-MM	Advanced Extensible Interface Memory Mapped
BAR	Base Address Register
CSR	Configuration and Status Register
DLLP	Data Link Layer Packet
DMA	Direct Memory Access
ECC	Error Correction Coding
EP	Endpoint
FIFO	First In First Out
LMMI	Lattice Memory Mapped Interface
LTSSM	Link Training and Status State Machine
MSI	Message Signaled Interrupt
RTL	Register Transfer Language
PCI	Peripheral Component Interconnect
PCIE	Peripheral Component Interconnect Express
PCS	Physical Coding Sublayer
PLL	Phase-Locked Loop
PM	Power Management
PMA	Physical Medium Attachment
RAM	Random Access Memory
RC	Root Complex
RP	Root Port
TLP	Transaction Layer Packet

1. Introduction

1.1. Overview of the IP

PCI Express® is a high performance, fully scalable, and well-defined standard for a wide variety of computing and communications platforms. As a packet-based serial technology, the PCI Express standard greatly reduces the number of required pins and simplifies board routing and manufacturing. PCI Express is a point-to-point technology, as opposed to the multi-drop bus in PCI. Each PCI Express device has the advantage of full duplex communication with its link partner to greatly increase overall system bandwidth.

The Lattice PCIe x8 IP Core provides a flexible, high-performance, easy-to-use Transaction Layer Interface to the PCI Express Bus. The Lattice PCIe x8 IP Core implementation is a hardened IP with soft logic provided for interface conversion options. The hardened IP is an integration of PHY and Link Layer blocks.

The Lattice PCIe x8 IP Core is supported in the Lattice Avant™-AT-G and Lattice Avant-AT-X FPGA device family and is available in the Lattice Radiant™ software.

1.2. Quick Facts

Table 1.1. Summary of the PCIe x8 IP

IP Requirements	Supported Devices	Avant-AT-G, Avant-AT-X, Certus™-N2 (except LN2-CT-20ES)
	IP Changes	Refer to the PCIe x8 IP Release Notes (FPGA-RN-02061).
Resource Utilization	Supported User Interface	LMMI, AXI-Lite, TLP, AXI-Stream, AXI-MM
Design Tool Support	Lattice Implementation	IP Core v3.0.0 – Lattice Radiant Software 2025.2 or later
	Synthesis	Synopsys® Synplify Pro® for Lattice
	Simulation	QuestaSim Pro, QuestaSim Lattice Edition

Notes:

1. In some instances, the IP may be updated without changes to the user guide. This user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.
2. Lattice Implementation indicates the IP version release coinciding with the software version release. Check the software for IP version compatibility with earlier or later software versions.

1.3. IP Support Summary

Table 1.2. PCIe x8 IP Support Readiness

Device Family	IP	User Interface	Gen Speed	Link Width	Data Rate (GT/s)	Radiant Timing Model	Hardware Validated
Avant-AT-G Avant-AT-X	PCIe EP (Non-DMA)	TLP	Gen 4 [1]	x8, x4, x2, x1	128, 64, 32, 16	Preliminary	No
			Gen 3	x8, x4, x2, x1	64, 32, 16, 8	Preliminary	No
			Gen 2	x8, x4, x2, x1	40, 20, 10, 5	Preliminary	No
			Gen 1	x8, x4, x2, x1	20, 10, 5, 2.5	Preliminary	No
		AXI-Stream	Gen 4[1]	x8, x4, x2, x1	128, 64, 32, 16	Preliminary	No
			Gen 3	x8, x4, x2, x1	64, 32, 16, 8	Preliminary	No
			Gen 2	x8, x4, x2, x1	40, 20, 10, 5	Preliminary	No
			Gen 1	x8, x4, x2, x1	20, 10, 5, 2.5	Preliminary	No
		AXI-MM/ AXI-Lite	Gen 2	x1	5	Preliminary	No
			Gen 1	x1	2.5	Preliminary	No
	PCIe DMA (Hardened)	AXI-MM	Gen 4[1]	x8	128	Preliminary	No
			Gen 3	x8	64	Preliminary	No
			Gen 2	x8	40	Preliminary	No
			Gen 1	x8	20	Preliminary	No
	PCIe DMA	AXI-MM	Gen 3	x4, x2, x1	32, 16, 8	Preliminary	No
			Gen 2	x4, x2, x1	20, 10, 5	Preliminary	No
			Gen 1	x4, x2, x1	10, 5, 2.5	Preliminary	No
Certus-N2	PCIe EP (Non-DMA)	TLP	Gen 4	x4, x2, x1	64, 32, 16	Preliminary	No
			Gen 3	x4, x2, x1	32, 16, 8	Preliminary	No
			Gen 2	x4, x2, x1	20, 10, 5	Preliminary	No
			Gen 1	x4, x2, x1	10, 5, 2.5	Preliminary	No
		AXI-Stream	Gen 4	x4, x2, x1	64, 32, 16	Preliminary	No
			Gen 3	x4, x2, x1	32, 16, 8	Preliminary	No
			Gen 2	x4, x2, x1	20, 10, 5	Preliminary	No
			Gen 1	x4, x2, x1	10, 5, 2.5	Preliminary	No
		AXI-MM/AXI-Lite	Gen 2	x1	5	Preliminary	No
			Gen 1	x1	2.5	Preliminary	No
	PCIe DMA	AXI-MM	Gen 3	x4, x2, x1	32, 16, 8	Preliminary	No
			Gen 2	x4, x2, x1	20, 10, 5	Preliminary	No
			Gen 1	x4, x2, x1	10, 5, 2.5	Preliminary	No

Note:

1. Lattice Avant-AT-X features 25G SERDES supporting up to PCIe Gen4.

1.4. Features

The Hard IP PHY key features include:

- Aggregation and bifurcation up to x4 lanes and x8 lanes PHY configuration
- 8b/10b encoding at 2.5 GT/s and 5 GT/s, and 128b/130b encoding at 8 GT/s and 16 GT/s
- Adaptive and configurable RX Continuous Time Linear Equalizer (CTLE) and Decision Feedback Equalizer (DFE)
- Adaptive and programmable TX equalization
- PCIe L1-substate power managements and Separate RefClk Independent SSC Architecture (SRIS)
- Extensive PMA debug capability via read/write and read-only registers in PCS
- Register-based control of all PCS-to-PMA signals
- A wide range of reference clock frequencies with optional fractional frequency correction capability
- A wide range of divided clock frequencies for external-to-PHY usage with optional spread-spectrum clock (SSC) capability
- Built-in, on-chip SSC generation and full configuration from –5000 to +5000 ppm
- Test support features such as near-end loopback, PLL bypass modes, and others.
- Protocol-compatible features such as L0s, squelch, power modes, and others.

The Hard IP Link Layer key features include:

- PCI Express Base Specification Revision 4.0 compliant including compliance with earlier PCI Express Specifications.
- Backward compatible with PCI Express 3.x, 2.x, 1.x
- Comprehensive application support:
- Endpoint
- Root Port (future release)
- Multi-Function support per link
  - PCIe x8 capable Core supports 1-8 PF
- SRIOV support per link (future release for non-DMA)
  - PCIe x8 capable Core supports up to 32 (PF+VF) functions
- Support for Autonomous and Software-Controlled Equalization
- Support for Figure of Merit and Up/Down PIPE PHY Equalization
- Flexible Equalization methods (Algorithm, Preset, User-Table, Adaptive-Table, Firmware-controlled)
- ECC RAM and Parity Data Path Protection
- Core Data Width
  - 64 bits for x1 lane
  - 128 bits for x2 lanes
  - 256 bits for x4 lanes
  - 512 bits for x8 lanes
- Complete error-handling support
- AER, ECRC generation/checking, recovery from Parity and ECC errors
- Supports detection of numerous optional errors, embedded simulation error checks/assertions
- Simulation and hardware error injection features enable error testing
- Flexible core options allow for design complexity/feature tradeoffs
- Configurable Receive, Transmit, and Replay Buffer sizes
- Supports Polarity Inversion, Up/Down-configure, Autonomous Link Width/Speed changes
- Power Management
  - Supports L1, ASPM L0s, and ASPM L1
  - L1 PM Substates with CLKREQ
  - Power Budgeting
  - Dynamic Power Allocation
  - Latency Tolerance Reporting
- Implements Type 0 Configuration Registers in Endpoint Mode
- Implements Type 1 Configuration Registers in Root Port Mode (future release)
  - Complete Root Port Configuration Register implementation (future release)
- Decodes received packets to provide key routing (BAR hits, Tag, and others) information

- Implements all aspects of the required PCIe Configuration Space
- Optionally consumes PCI Express Message TLPs or leaves them in band
- Interfaces have consistent timing and function over all modes of operation
- Implements all three PCI Express Layers (Transaction, Data Link, and Physical)
- Hardened high-performance multi-channel scatter-gather DMA controller
  - Support up to 64 DMA channels
  - Support for PCIe multi-function and SR-IOV capability
  - 64-bit address support

1.4.1. Hard IP Limitations

The following are the limitations of the Hard IP:

- DMA Core always operates with PCIe x8 resource (Compatible with any lower lane width).

1.4.2. Soft IP

- Non-DMA
  - TLP Mode
  - AXI-Stream Data Interface
  - AXI-MM Data Interface (Bridge Mode) [1]
  - AXI-Lite Data Interface (Bridge Mode) [1]
- DMA
  - AXI-MM Data Interface[2]
- LMMI Register Interfaces
- AXI-Lite Register Interface

Notes:

1. Only X1 is supported.
2. Only up to Gen3 X4 is supported.

1.5. Licensing and Ordering Information

The PCIe x8 IP is available with the Lattice Radiant Subscription software. To purchase the Lattice Radiant Subscription license, contact Lattice Sales or go to the Lattice Online Store.

1.6. Hardware Support

Refer to the Example Design section for more information on the boards used.

1.7. Speed Grade Supported

The Lattice PCIe IP core supported speed grade is provided in this section. Different configurations may be supported using different speed grades due to fabric performance requirements.

- 3 – fastest speed grade

Table 1.3. Lattice PCIe IP Core Supported Speed Grade

PCIe Core Config	Device Family	Speed Grade
Gen3x8	Avant-AT-G	1/2/3
Gen4x8	Avant-AT-X	3
Gen3x4	Certus-N2	1/2/3
Gen4x4	Certus-N2	3

1.8. Naming Conventions

1.8.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL.

1.8.2. Signal Names

- _*n* are active low (asserted when value is logic 0)
- _*i* are input signals
- _*o* are output signals

2. Functional Description

2.1. PCIe IP Architecture Overview

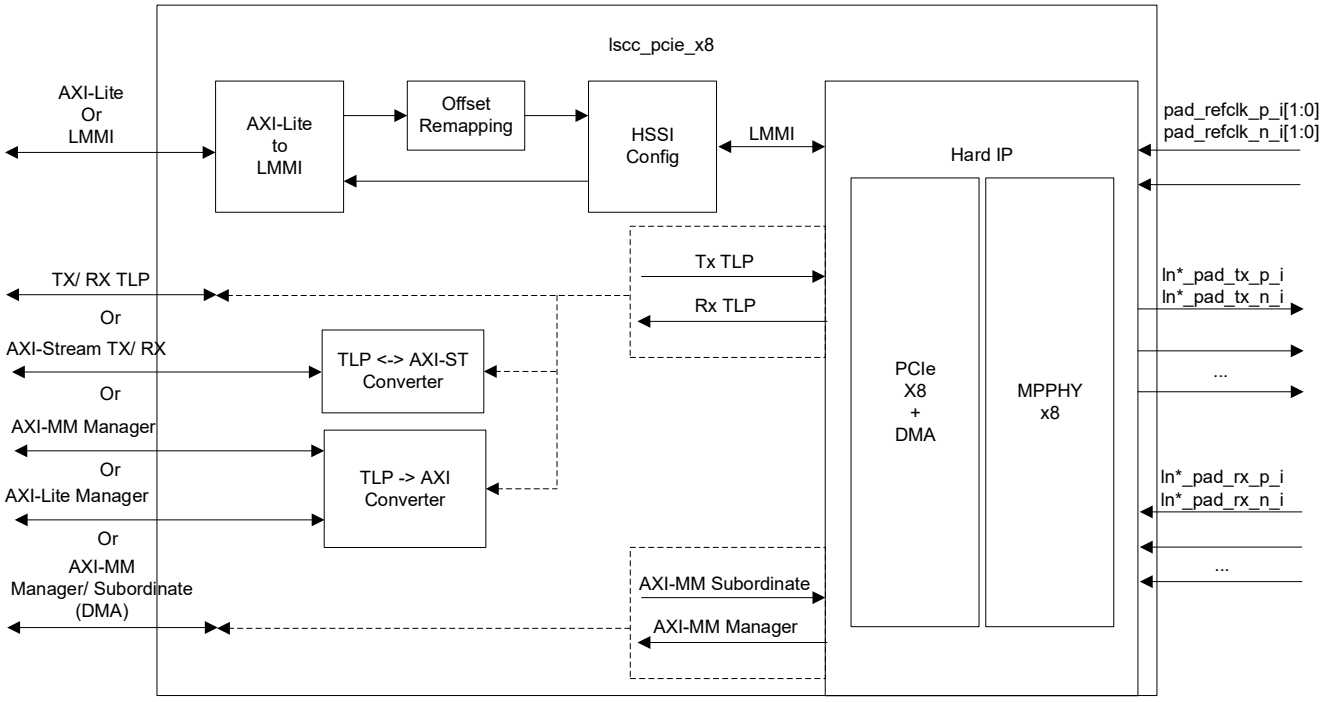


Figure 2.1. Lattice PCIe x8 IP Core Block Diagram

The Lattice PCIe x8 Core implements all three layers defined by the PCI Express specification: Physical, Data Link, and Transaction, up to eight lanes.

The Lattice PCIe x8 Core Hard IP implementation integrates Gen4 capable PCIe Core with DMA controller. A block diagram with an 8-lane instance is shown in Figure 2.2.

For convenience, the Link Layer Cores are referred to as Link 0 – PCIe x8 Core.

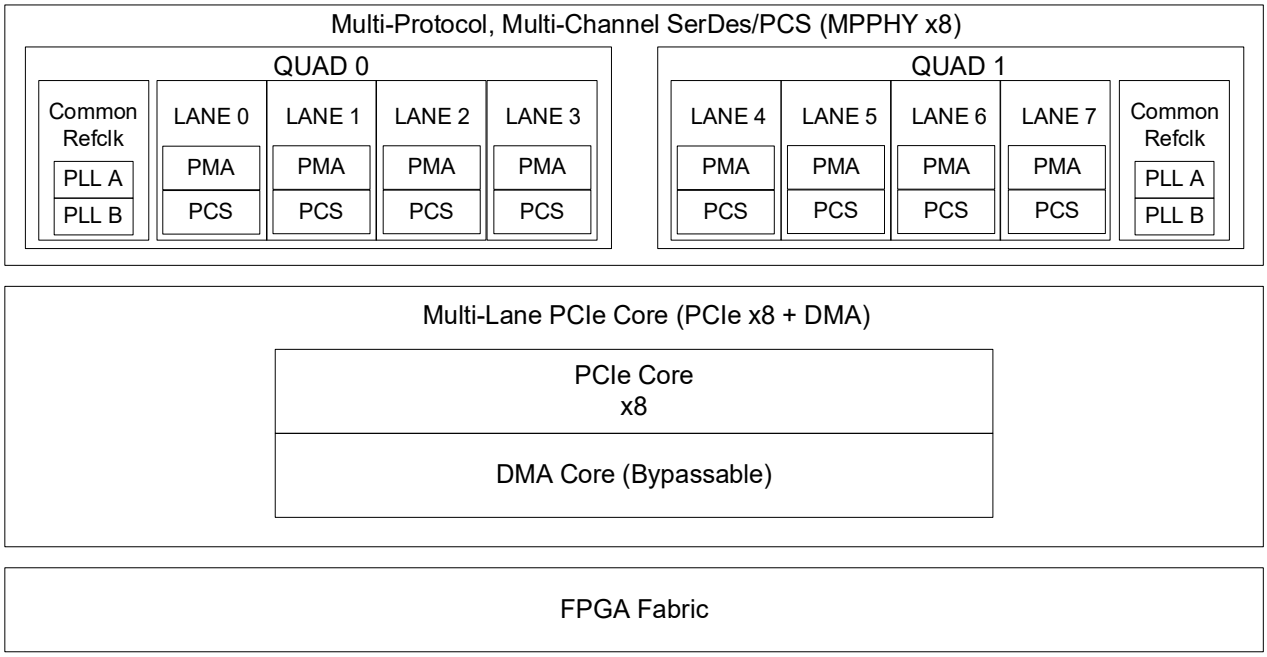


Figure 2.2. Lattice 8-lane SERDES/PCS + PCIe Hard-IP

The Lattice PCIe x8 Core Hard IP has the following interfaces as shown in Figure 2.3. The details of each interface are discussed in the following sections.

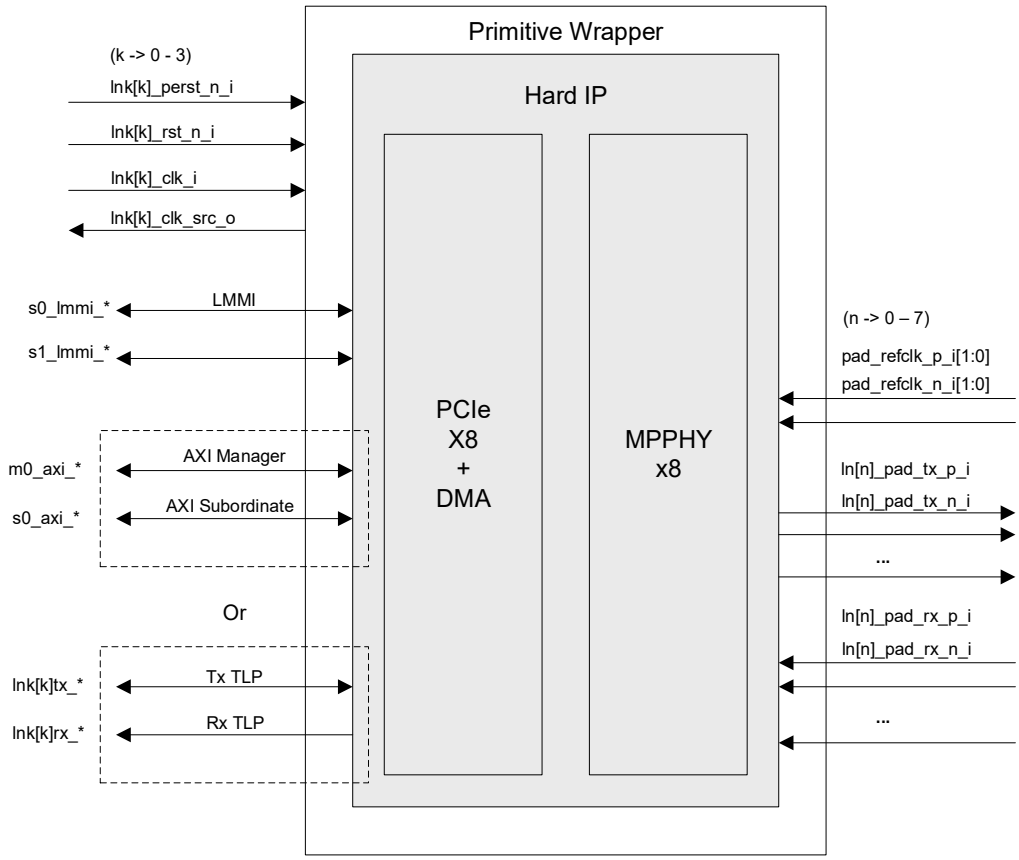


Figure 2.3. Lattice PCIe x8 Core Hard IP

- Clock and Reset Interface
  - The user domain interface can be clocked using one of several PHY clock outputs (sys_clk_i = link0_clk_src_o in future release) or by user-generated clock. The fundamental reset (link0_perst_n_i) resets the core (PHY and Link layer blocks) except for the core Configuration Registers. Another reset (link0_rst_usr_n_i) is provided to glue logic reset the logic between user domain and the PCIe Core.
- PHY Interface
  - High Speed Serial Interface supports maximum rate of 16GT/s
  - Up to eight lanes (grouped into Two Quads)
- TLP Receive Interface
  - Receive TLPs from the PCIe link partner
  - High bandwidth interface
  - 512b data width (64b per active lane)
- TLP Transmit Interface
  - Transmit TLPs to the PCIe link partner high bandwidth interface
    512b data width (64b per active lane)
- Power Management Interface (future release)
  - Ports for implementing power management capabilities (future release)
- AXI Interface
  - Available if Hard DMA Core is enabled
  - 512b data width each (Manager and Subordinate Ports)
- AXI-Lite or LMMI (Mutually exclusive) – Configuration and Status Register (CSR) Interface
  - This interface is used to write and read the core configuration and status registers. A typical customer application requires changing only a small number of the default values such as Vendor ID, Device ID, and BAR configuration.
  - 32b data width (AXI-Lite) or 16b data width (LMMI)

2.2. Clocking

2.2.1. Clocking Overview

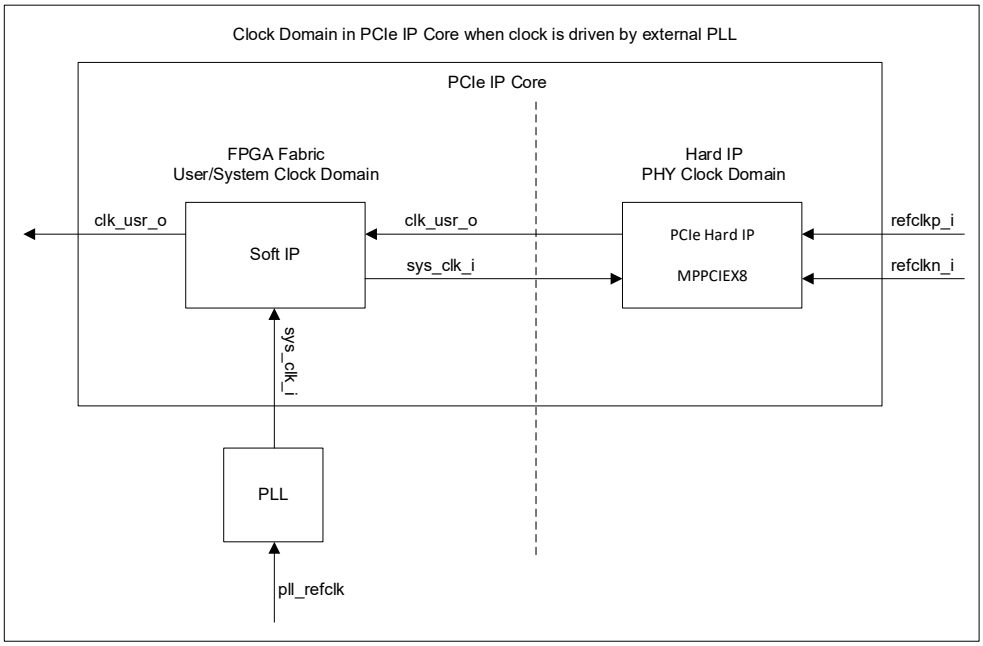


Figure 2.4. PCIe IP Clock Domain Block Diagram with External PLL

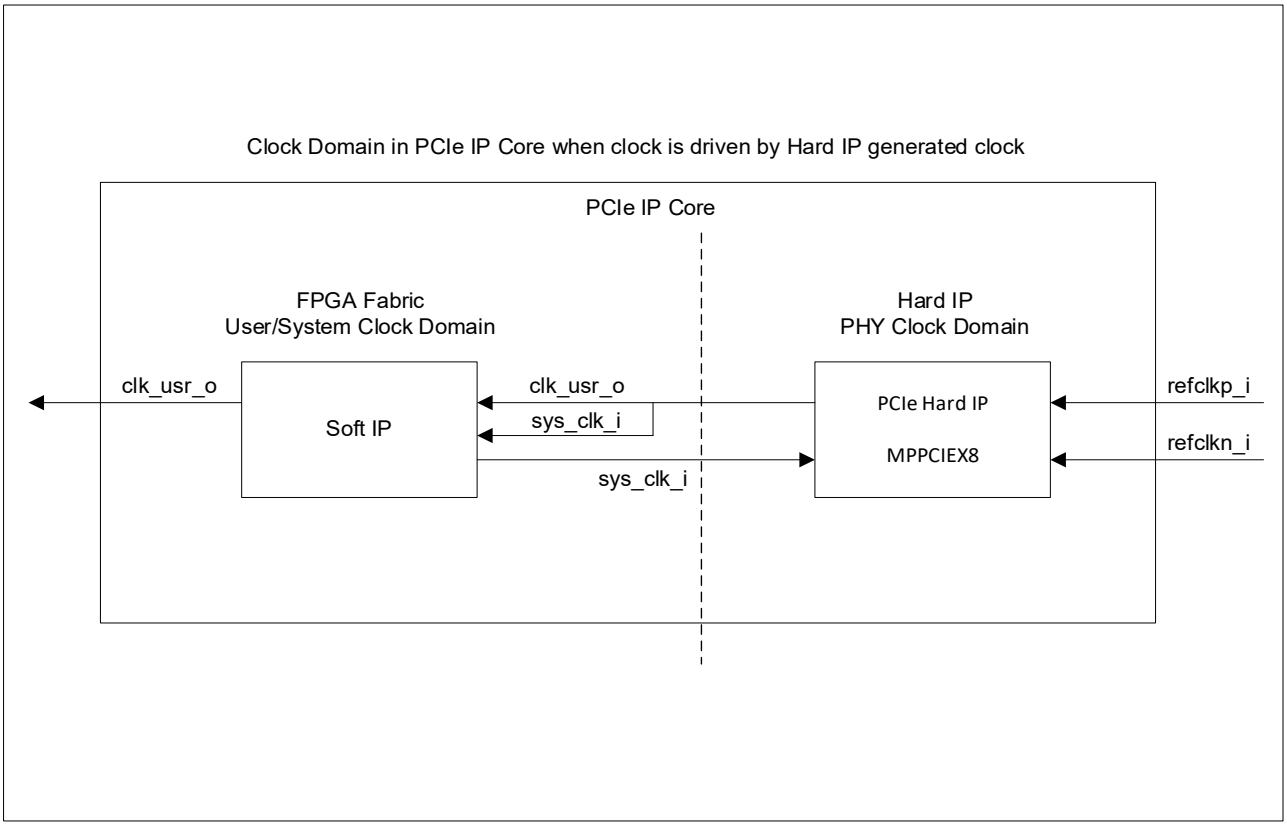


Figure 2.5. PCIe IP Clock Domain Block Diagram for TLP Interface

The PCIe x8 IP includes the following clock domains. The *sys_clk_i* signal is generated during the PLL IP instantiation. For the TLP interface, you can connect *clk_usr_o* back to the *sys_clk_i*, as shown in Figure 2.5.

Note: There is a known limitation that the *link0_clk_usr_o* is incorrectly constrained by the Radiant software. It is recommended not to use *link0_clk_usr_o* until future updates.

- refclkp_i/refclkn_i are differential PHY reference clocks.
  - You can set the reference clock frequency in the PCIe IP Core.
  - There are two options available: 100 MHz or 125 MHz.
- sys_clk_i is the user clock domain input clock.
  - This clock is generated by the system PLL and shared with the Link layer blocks.
  - For the TLP interface variants, you can choose to connect *clk_usr_o* back to the *sys_clk_i* as shown in Figure 2.5.
- clk_usr_o is the User Clock Domain Output Clock.
  - This is the pclk output that comes from the PHY of the PCIe IP core.
  - By default, *clk_usr_o* uses the divided clock from PHY.

2.3. Reset

2.3.1. Reset Overview

There are two fundamental reset events that can occur in PCI Express:

- Cold Reset – This is a fundamental reset applied during power cycling. The signal link [LINK]_perst_n_i is asserted.
- Warm Reset – This is a fundamental reset triggered by hardware without the removal and re-application of power. The perst_n_i signal is asserted.

The fundamental reset *link [LINK]_perst_n_i* resets the core (Link Layer and PHY Layer blocks).

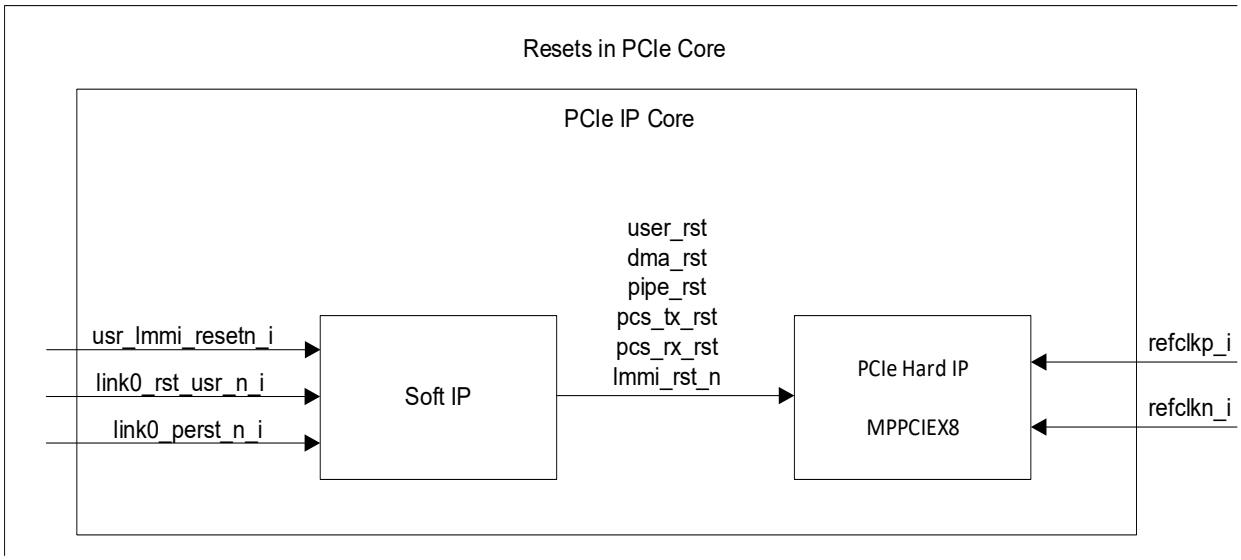


Figure 2.6. Reset Signals in Lattice PCIe IP Core

2.3.2. Clock and Reset Sequence

The PCIe IP clock and reset operation is shown in Figure 2.7.

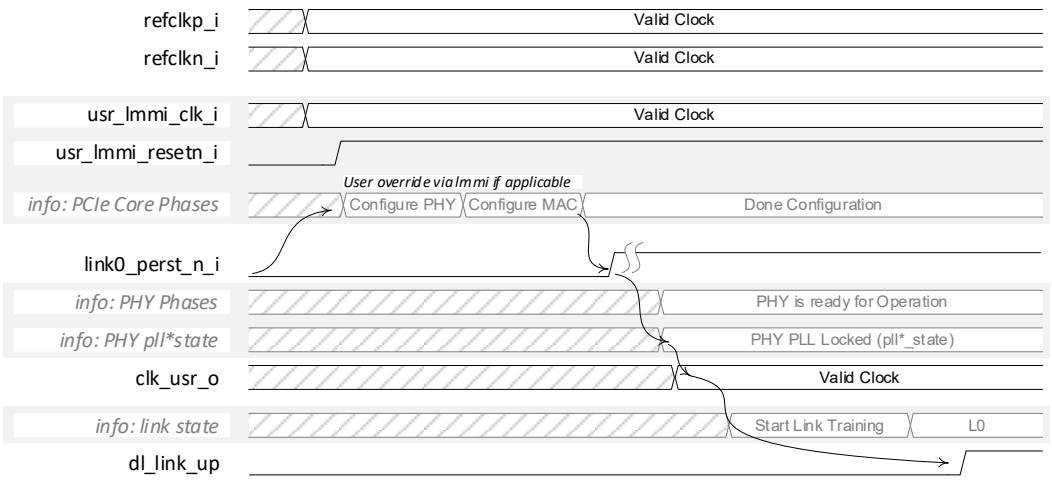


Figure 2.7. Clock and Reset Sequence Diagram

The Lattice PCIe x8 IP Core configuration register implementation has default values that are appropriate for most applications. The PCIe IP core register configuration can be optionally modified via either the LMMI or AXI-Lite interface. When the LMMI or AXI-Lite interface is used to configure the PHY layer registers, the configuration should be done before the deassertion of the *link0_perst_n_i* signal. The PHY Layer is released from reset and is ready for operation once it can generate the PIPE clock output (such as the link0_clk_usr_o signal). You can release the reset to application logic after the dl_link_up assertion.

When usr_lmmi_resetn_i is asserted, it resets both the soft logic operating in the LMMI clock domain and the register configuration interface of the PCIe Hard IP core. Users must also assert link0_perst_n_i to ensure a complete reset of the PCIe core. For further details, refer to the reset flow waveform in Figure 2.7.

2.4. Protocol Layers

There are three major classes of packets in PCIe devices: Transaction Layer Packets (TLP), Data Link Layer Packets (DLLP), and Ordered Sets (OS). The function of the Protocol Layer is to generate and process these packets.

- Transaction Layer
  The Transaction Layer manages the TLPs to communicate request and completion data with other PCIe devices. The TLP packets are assembled at the *transmit side* of the link and disassembled at the *receive side* of the link. The TLP communicates through different formats either in I/O request format or in the memory request format.
- Data Link Layer
  The Data Link Layer transfers data from the Transaction Layer to the Physical Layer. It plays an important role in assuring good reception of the TLP packets. The DLLPs are used to convey information about the link initialization, power management, flow control, and TLP acknowledgements.

- Physical Layer
  The Physical Layer converts the packets from the Data Link Layer into serialized bit streams and transfers it to the external physical link. The receive logic de-serializes the bits, reassembles the packets, and forwards it to the Data Link Layer. It conveys the communication between the Data Link Layer and the external physical link. The Physical layer is divided into the Logical sub-block and the Electrical sub-block. The Logical sub-block frames and deframes the packets and implements the LTSSM state machine. The scrambling, descrambling, and 8B/10B encoding and decoding of data are done in the logical sub-block. The Electrical sub-block provides the physical interface to the Link and contains the differential transmitters and receivers. Ordered sets are exchanged during link training and link initialization.

2.4.1. ECC and Parity Data Path Protection

The Lattice PCIe x8 IP Core protects the TLP data path with Error Correction Coding (ECC) and Parity Protection. This is implemented in the Hard IP block.

ECC is used to protect TLP data in the following data path RAMs:

- Replay Buffer
- Receive Buffer
- Transmit Buffer

The ECC implementation enables correction for 1-bit errors and detection for 2-bit errors. The 8-bit of ECC information is included in the RAMs for each 64 (or fraction thereof) data bits.

Even (XOR) Parity (*parity[i] = ^(data[((i+1) × 8)-1:(i × 8)]*) is used to protect the data path. Parity provides detection for 1-bit errors (and other odd-bit errors). To enable continuous parity protection coverage, parity is passed through RAMs that are also protected by ECC.

The core includes the ability to enable/disable the reporting and handling of ECC/Parity errors. Correctable errors (ECC 1-bit errors) are fixed when correction is enabled. Uncorrectable ECC/Parity errors in the transmit data path result in the associated TLP being discarded or nullified when error handling is enabled. While error handling can be disabled, this is not recommended as passing a known TLP with bad contents can result in a more serious error condition than discarding the TLP.

2.4.1.1. Receive Data Path

For the receive data path, parity is generated for received TLPs prior to the removal and validation of the Link CRC (LCRC). Parity protection is thus overlapped with LCRC protection.

Received TLP parity is passed with the associated received TLP (header and payload) bytes through the Receive Buffer and onto the user Transaction Layer Receive interface. It is expected that parity is checked and errors are handled by the ultimate TLP consumer. Since TLP can have parity errors on any byte (toward the end of a longer TLP for instance), it is generally not possible to avoid processing the error TLP as the earlier portion of the TLP may already have been processed by the time that the error is detected.

Applications that do not want to process TLPs with errors need to store and forward the TLP for processing only after inspecting the parity of all data bytes. If the core Transaction Layer detects a parity error while it is consuming a received TLP (Type 0 Configuration Read/Write, Malformed TLP, and Message), the error is reported as Uncorrectable Error (in AER capability) and the core discards the TLP without processing it.

2.4.1.2. Transmit Data Path

For the transmit data path, parity is generated by the TLP source. For user TLPs (for example those transmitted on the core's Transaction Layer Transmit Interface), the parity is provided along with associated TLP (header and payload) bytes. The provided parity is kept with the associated data as it traverses the core. The parity is checked and discarded just after the TLP PCIe LCRC is generated.

Parity protection is thus overlapped with LCRC protection, including the associated PCIe replay mechanism. If the core detects a parity or uncorrectable ECC error during transmission of a TLP, the error is reported and the associated TLP is nullified (discarded) and not retransmitted. This is a serious error that must be handled by the software. The TLP is discarded to not propagate the error and risk potentially worse consequences in other components that receives TLPs with known bit errors.

2.4.1.3. Uncorrectable Error Recovery

PCI Express includes the ability to nullify or cancel a TLP transmission immediately after it is completed by inverting the LCRC and using End Bad (EDB) end framing instead of the normal TLP end framing. TLP can be nullified to reduce propagation, potentially multiplying the effects of the error. Nullified TLPs are not regenerated by the original TLP source as it is difficult for software to construct the missing TLP. As a result, there is a fatal system error condition regardless of whether the error TLP is nullified or not. When TLP is nullified due to errors, the core attempts to keep the transmit stream active so that the software can be notified of the error using the standard in-band mechanisms (for example, transmission of ERR_NFAT or ERR_FAT message).

TLPs are allocated a sequence number during transmission and the PCIe receiver only accepts TLPs in sequential order. When a TLP is nullified due to an uncorrectable error, the missing sequence number must be recovered before the link can continue to transmit TLPs.

TLPs are allocated Virtual Channel Flow Control Credits when they are transmitted by the Transaction Layer. The PCI Express device receiving the TLP over the PCI Express link frees the associated credits by sending Flow Control Update DLLPs. TLPs, which are nullified due to uncorrectable ECC and Parity errors, are allocated credits by the Transaction Layer, which is never freed since the TLP is nullified and not received by the Receiver. Nullified TLPs are discarded by the Receiver without affecting Flow Control Credits or Sequence Number.

Whenever a transmitted TLP is nullified due to an uncorrectable error, this causes the PCI Express link to be unable to process further TLPs. The sequence number and flow control credits that are allocated to the nullified TLP must be reclaimed before the link is repaired. The Lattice PCIe x8 IP Core contains logic to correct the link when TLPs are nullified due to uncorrectable errors.

Whenever an uncorrectable ECC of Parity error is detected, it is recommended for you to reset the link through the software to reset the link although the link is corrected for further transmission.

2.4.2. Error Handling

The Lattice PCIe x8 IP Core detects and implements the appropriate response to most error conditions without user intervention. You generally only need to detect and report errors that the core does not have enough information to detect.

2.4.2.1. PCIe-Defined Error Types

The following defines the error types in the PCIe. The *Type* column in Table 2.1 to Table 2.4 defines the PCI Express defined error severity:

- COR – Correctable
- NFAT – Uncorrectable – Non-Fatal
- FAT – Uncorrectable – Fatal

Table 2.1. General PCI Express Error List

Error	Type
Corrected Internal Error	COR
Uncorrectable Internal Error	FAT
Header Log Overflow	COR

Table 2.2. Physical Layer Error List

Error	Type
Receiver Error	COR

Table 2.3. Data Link Layer Error List

Error	Type
Bad TLP	COR
Bad DLLP	COR
Replay Timeout	COR
REPLAY_NUM Rollover	COR
Data Link Layer Protocol Error	FAT
Surprise Down	FAT

Table 2.4. Transaction Layer Error List

Error	Type
Poisoned TLP Received	NFAT
ECRC Check Failed	NFAT
Unsupported Request	NFAT
Completion Timeout	NFAT
Completer Abort	NFAT
Unexpected Completion	NFAT
ACS Violation	NFAT
MC Blocked TLP	NFAT
AtomicOp Egress Blocked	NFAT
Receiver Overflow	FAT
Flow Control Protocol Error	FAT
Malformed TLP	FAT

2.4.2.2. User Error Reporting

The User Hardware design must be able to detect and report the following errors.

- Uncorrectable Internal Error
  - Signals if AER Version 0x2 is enabled in the core and user hardware is detected and unable to correct an application-specific error that is not reported through another error mechanism.
  - If AER is supported by the core, the header of the first TLP associated with the error may optionally be logged.
- Poisoned TLP Received with Advisory Non-Fatal Severity
  - Signals if the core's default poison handling is disabled (ignore_poison == 1) and you receive a poisoned TLP that is considered as Advisory Non-Fatal severity. If the data payload of a poisoned packet is used or the poison can be recovered from the software or other mechanism, the poison should be treated as Advisory Non-Fatal since a non-fatal error often causes a system operation to crash.
  - If AER is supported by the core and the core is operating in Endpoint mode, an ERR_COR message is requested and transmitted if enabled.
  - If AER is supported by the core, the header of the poisoned packet must be logged.
- Poisoned TLP with Non-Fatal Severity
  - Signals if the core's default poison handling is disabled (ignore_poison == 1) and you receive a poisoned TLP that is considered as Non-Fatal severity. Handling poison as Non-Fatal severity should be avoided when possible as this is often fatal to the system operation.
  - If AER is supported by the core, the header of the poisoned packet must be logged.

- Unsupported Request
  - A Type0 Vendor-defined message that is received but not supported by user logic is an Unsupported Request. This is uncommon since only devices designed to receive Type0 Vendor-defined messages should receive these. However, compliance tests may require this error to be handled; hence, it is recommended to implement this check. Receiving a message with Message Code == 0x7E should cause Unsupported Request to be reported, unless the user design is designed to receive these messages.
  - Completions that are received with a Reserved Completion status must be handled as if the Completion status is an Unsupported Request.
- Completion Timeout
  - If you initiate a non-posted request (all reads, I/O Write, and Configuration Write), you are required to implement a completion timeout timer that fires if completions to a non-posted request are not received in the allotted time. This error check needs to be implemented by the user design that includes initiating non-posted requests.
- Completion Abort
  - Signals if permanently unable to process a request due to a device-specific error condition. Generally, this error is only signaled if you choose to implement a restricted programming model (that requires the software to always perform DWORD size transactions and not support burst transactions). This is not recommended unless the only software that can access the user design is your own software, which is designed to conform with the restricted programming model.
  - If AER is supported by the core, the header of the aborted request must be logged.
- Unexpected Completion
  - You must signal if a completion is received but the tag does not match any outstanding requests.
  - If the core is enabled for Target_Only mode indicating that the user design does not initiate non-posted requests, the core considers all completions as Unexpected Completions, discards them, and generates the appropriate response. In this case, you do not handle this error.
  - If AER is supported by the core, then the header of the completion must be logged.

As a minimum, it is recommended to report the following errors:

- Completion Timeout if user logic initiates non-posted requests (for example, DMA read requests)
- Unsupported Requests for the cases described above
- Unexpected Completion of the case described above
- Poison, when the core's default poison handling is disabled (ignore_poison == 1)

2.4.3. LTSSM State

2.4.3.1. Main LTSSM

The Lattice PCIe x8 IP Core follows the PCI Express specification for the Link Training and Status State Machine. However, to help hit higher frequencies, the LTSSM is split into one Major State LTSSM state machine and several separate LTSSM sub-state machines, with one sub-state state machine for each major state.

The Lattice PCIe x8 IP Core implements additional LTSSM sub-states that are necessary to meet PCIe specification LTSSM operation but are not given an explicit sub-state in the PCIe specification. Table 2.5 lists each state.

Table 2.5. LTSSM State Definition

LTSSM Major State	LTSSM Sub-state	Description
0 – Detect	0 – DETECT_INACTIVE	The sub-state is *DETECT_INACTIVE* whenever the LTSSM major state is not *Detect*.
	1 – DETECT_QUIET	Detect.Quiet
	2 – DETECT_SPD_CHG0	Detect.Quiet – Sub-state to change speed change back to 2.5G if needed. Request PHY speed change.
	3 – DETECT_SPD_CHG1	Detect.Quiet – Sub-state to change speed change back to 2.5G if needed. Wait for speed change to complete.
	4 – DETECT_ACTIVE0	Detect.Active – First Rx Detection.
	5 – DETECT_ACTIVE1	Detect.Active – Wait 12 ms between Rx Detection attempts.
	6 – DETECT_ACTIVE2	Detect.Active – Second Rx Detection (if needed).

LTSSM Major State	LTSSM Sub-state	Description
	7 – DETECT_P1_TO_P0	Detect.Active – Change PHY power state from P1 to P0 (inactive to active) if needed (that is on Detect – Polling transition).
	8 – DETECT_P0_TO_P1_0	Change PHY power state from P0 to P1 (active to inactive) – Transmit Electrical Idle Ordered Sets to notify the link partner that the link is idle.
	9 – DETECT_P0_TO_P1_1	Change PHY power state from P0 to P1. Wait for TX Electrical Idle Ordered Set transit request made in DETECT_P0_TO_P1_0 to get transmitted at the output of the core.
	10 – DETECT_P0_TO_P1_2	Change PHY power state from P0 to P1. Wait for PHY to reach P1 state before continuing.
1 – Polling	0 – POLLING_INACTIVE	The sub-state is *POLLING_INACTIVE* whenever the LTSSM Major State is not Polling.
	1 – POLLING_ACTIVE_ENTRY	Polling.Active – Entry to *Polling.Active* State exists since in some cases, the LTSSM must exit Polling without Tx of TS OS.
	2 – POLLING_ACTIVE	Polling.Active
	3 – POLLING_CFG	Polling.Configuration
	4 – POLLING_COMP	Polling.Compliance – Transmitting compliance pattern.
	5 – POLLING_COMP_ENTRY	Polling.Compliance entry state – Directs a speed change through POLLING_COMP_EIOS, POLLING_COMP_EIOS_ACK, and POLLING_COMP_IDLE when necessary, before going to POLLING_COMP.
	6 – POLLING_COMP_EIOS	Polling.Compliance – Transmit Electrical Idle Ordered Sets to notify the link partner that the link is idle.
	7 – POLLING_COMP_EIOS_ACK	Polling.Compliance – Wait for the Electrical Idle Ordered Sets transmitted in POLLING_COMP_EIOS to exit the core.
	8 – POLLING_COMP_IDLE	Polling.Compliance – Perform speed change now that link is idle.
2 – Configuration	0 – CONFIGURATION_INACTIVE	The sub-state is *CONFIGURATION_INACTIVE* whenever the LTSSM Major State is not Configuration.
	1 – CONFIGURATION_US_LW_START	Acting as Upstream Port – Configuration.Linkwidth.Start
	2 – CONFIGURATION_US_LW_ACCEPT	Acting as Upstream Port – Configuration.Linkwidth.Accept
	3 – CONFIGURATION_US_LN_WAIT	Acting as Upstream Port – Configuration.Lanenum.Wait
	4 – CONFIGURATION_US_LN_ACCEPT	Acting as Upstream Port – Configuration.Lanenum.Accept
	5 – CONFIGURATION_DS_LW_START	Acting as Downstream Port – Configuration.Linkwidth.Start
	6 – CONFIGURATION_DS_LW_ACCEPT	Acting as Downstream Port – Configuration.Linkwidth.Accept
	7 – CONFIGURATION_DS_LN_WAIT	Acting as Downstream Port – Configuration.Lanenum.Wait
	8 – CONFIGURATION_DS_LN_ACCEPT	Acting as Downstream Port – Configuration.Lanenum.Accept
	9 – CONFIGURATION_COMPLETE	Configuration.Complete
	10 – CONFIGURATION_IDLE	Configuration.Idle

LTSSM Major State	LTSSM Sub-state	Description
3 – L0	0 – L0_INACTIVE	The sub-state is *L0_INACTIVE* whenever the LTSSM Major State is not L0.
	1 – L0_L0	L0 – Link is in L0.
	2 – L0_TX_EL_IDLE	Tx_L0s.Entry, L1.Entry, or L2.Entry – Transmit Electrical Idle Ordered Sets to notify the link partner that the link is idle (that is for preparing to enter low power states such as Tx_L0s, L1, and L2).
	3 – L0_TX_IDLE_MIN	Tx_L0s.Entry, L1.Entry, or L2.Entry – Guarantee the minimum Tx Elec Idle time when entering electrical idle and also require Rx EIOS to have been received when necessary.
4 – Recovery	0 – RECOVERY_INACTIVE	The sub-state is *RECOVERY_INACTIVE* whenever the LTSSM Major state is not *Recovery*.
	1 – RECOVERY_RCVR_LOCK	Recovery.RcvrLock
	2 – RECOVERY_RCVR_CFG	Recovery.RcvrCfg
	3 – RECOVERY_IDLE	Recovery.Idle
	4 – RECOVERY_SPEED0	Recovery.Speed – Transmit Electrical Idle Ordered Sets to notify the link partner that the link is idle.
	5 – RECOVERY_SPEED1	Recovery.Speed – Determine to which speed to change.
	6 – RECOVERY_SPEED2	Recovery.Speed – Wait for remote device to enter electrical idle and wait for the required minimum time.
	7 – RECOVERY_SPEED3	Recovery.Speed – Request PHY change speed and wait for PHY to finish changing speed.
	8 – RECOVERY_EQ_PH0	Recovery.Equalization – Phase 0
	9 – RECOVERY_EQ_PH1	Recovery.Equalization – Phase 1
	10 – RECOVERY_EQ_PH2	Recovery.Equalization – Phase 2
	11 – RECOVERY_EQ_PH3	Recovery.Equalization – Phase 3
5 – Disable	0 – DISABLE_INACTIVE	The sub-state is *DISABLE_INACTIVE* whenever the LTSSM Major state is not *Disable*.
	1 – DISABLE0	Disable – Transmit 16 to 32 TS1 Ordered Sets with Disable Link bit asserted.
	2 – DISABLE1	Disable – Transition to Electrical Idle.
	3 – DISABLE2	Disable – Wait to receive an Electrical Idle Ordered Set and min time of TX_IDLE_MIN afterwards.
	4 – DISABLE3	Disable – Wait until a Disable exit condition occurs.
6 – Loopback	0 – LOOPBACK_INACTIVE	The sub-state is *LOOPBACK_INACTIVE* whenever the LTSSM Major state is not *Loopback*.
	1 – LOOPBACK_ENTRY	Loopback.Entry – Loopback entry state – Loopback Leader may be required to Tx Loopback TS OS before continuing or speed may need to be changed before beginning loopback.
	2 – LOOPBACK_ENTRY_EXIT	Loopback.Entry – Prepare to enter Loopback.Active
	3 – LOOPBACK_EIOS	Loopback.Entry – Transmit Electrical Idle Ordered Sets to notify the link partner that the link is idle. (to change speed).
	4 – LOOPBACK_EIOS_ACK	Loopback.Entry – Wait for the Electrical Idle Ordered Sets transmitted in LOOPBACK_EIOS to exit the core.
	5 – LOOPBACK_IDLE	Loopback.Entry – Stay in Electrical Idle for required minimum time.
	6 – LOOPBACK_ACTIVE	Loopback.Active
	7 – LOOPBACK_EXIT0	Loopback.Exit – Tx Electrical Idle
	8 – LOOPBACK_EXIT1	Loopback.Exit – Stay in Electrical Idle for required minimum time.

LTSSM Major State	LTSSM Sub-state	Description
7 – Hot Reset	0 – HOT_RESET_INACTIVE	The sub-state is *HOT_RESET_INACTIVE* whenever the LTSSM Major state is not *Hot Reset*.
	1 – HOT_RESET_HOT_RESET	Hot Reset – as Follower
	2 – HOT_RESET_ LEADER _UP	Hot Reset – as Leader with Link Up
	3 – HOT_RESET_ LEADER _DOWN	Hot Reset – as Leader with Link Down
8 – TX L0s	0 – TX_L0S_INACTIVE	The sub-state is *TX_L0S_INACTIVE* whenever the LTSSM Major state is not *TX L0s*.
	1 – TX_L0S_IDLE	Tx_L0s.Idle – Idle
	2 – TX_L0S_TO_L0	Tx_L0s.Idle – Exiting TX L0s; wait for PHY to indicate exit from L0s complete
	3 – TX_L0S_FTS0	Tx_L0s.FTS – Transmit requested NFTS.
	4 – TX_L0S_FTS1	Tx_L0s.FTS – Transmit additional FTS required by Cfg Register Extended Sync.
9 – L1	0 – L1_INACTIVE	The sub-state is *L1_INACTIVE* whenever the LTSSM Major state is not *L1*.
	1 – L1_IDLE	L1.Idle
	2 – L1_SUBSTATE	L1.1 or L1.2 depending upon higher level Power Management State Machine control.
	3 – L1_TO_L0	L1.Idle – Exiting L1; wait for PHY to indicate exit from L1 complete.
10 – L2	0 – L2_INACTIVE	The sub-state is *L2_INACTIVE* whenever the LTSSM Major State is not *L2*.
	1 – L2_IDLE	L2.Idle – Idle
	2 – L2_TX_WAKE0	L2.TransmitWake – Transmit a Beacon until remote device exits electrical idle.
	3 – L2_TX_WAKE1	L2.TransmitWake – Assert Tx Electrical Idle before changing power state to P1.
	4 – L2_EXIT	L2.Idle – L2 exit; wait until PHY finishes power change out of L2.
	5 – L2_SPEED	L2.Idle – Change speed if required before going to L2.

2.4.3.2. RX L0s State Machine

The Rx_L0s State Machine follows the L0s state of the receiver. The Rx_L0s State Machine operates independently of the main LTSSM, which controls the state of the transmitter.

Table 2.6. RX L0s State Description

LTSSM Sub-state	Description
0 – RX_L0S_L0	The sub-state is “RX_L0S_L0” whenever the receiver is in L0 (that is not en route to or in Rx L0s).
1 – RX_L0S_ENTRY	Rx_L0s.Entry
2 – RX_L0S_IDLE	Rx_L0s.Idle
3 – RX_L0S_FTS	Rx_L0s.FTS
4 – RX_L0S_REC	Rx_L0s.FTS – Wait until LTSSM Major State == Recovery due to Rx L0s exit error

2.5. PHY Equalization (8 GT/s)

Operating at 8 GT/s data rate requires an equalization process to be completed before data can be reliably transferred. The Lattice PCIe x8 IP Core supports both the autonomous equalization and software-controlled equalization methods.

When equalization is initiated either through the autonomous or software mechanism, the core Link Training and Status State Machine (LTSSM) enters recovery to perform equalization. Equalization is done for both directions of the link. The Equalization process consists of requesting several sets of transmit coefficients for the remote device to use, evaluating the quality of each set of coefficients, and then choosing the best coefficient set for operation.

The Lattice PCIe x8 IP Core supports PHY using the Figure of Merit and Up/Down Equalization feedback methods established by the PIPE Specification, as well as an option for direct firmware control/status. The Lattice PCIe x8 IP Core implements several equalization algorithms that enable users to select the method that works best for the project.

2.5.1. Equalization Process

Equalization is done for both directions of the data flow:

- Remote Transmitter to Local Receiver
  - Equalization is controlled by the local device using the mechanisms described in this section.
- Local Transmitter to Remote Receiver
  - Equalization is controlled by the remote device with the assistance of the local core LTSSM. While the Local Transmitter to Remote Receiver link is undergoing equalization, the *pipe_tx_deemph* port is periodically updated by the core based upon received requests from the remote device. Other than the PHY updating its transmitter to the new *pipe_tx_deemph* settings, no external action is required.

2.5.2. Equalization Time Limit

The equalization state machine follows the PCI Express timeout of 24 ms. You must ensure that the selected method is able to complete equalization within 24 ms, or the process is aborted and considered unsuccessful.

100 µs (*0.1 ms*) should be reserved for each equalization attempt for the LTSSM to communicate the new settings to the remote device and to receive acknowledgement of those settings.

If a PHY takes 2 ms to evaluate each setting and 0.1 ms is reserved per setting for LTSSM, only 11 settings may be evaluated during equalization (*2.1 ms × 11 == 23.1 ms*). Since a final evaluation (with the best of the trial settings) is typically required to finalize the process, a maximum of 10 different trial settings may be evaluated in this case.

The PHY vendor must specify the required time to complete an equalization evaluation. If the PHY vendor does not specify a limit, the worst case of 2 ms must be assumed and a maximum of 10 different settings may be attempted. The smaller the amount of time the PHY takes to evaluate a setting, the more settings can be attempted.

2.5.3. Equalization Methods

The Lattice PCIe x8 IP Core implements a flexible Equalization process that enables users to exercise control over the choice of equalization coefficients requested of the remote device.

Each method requires expressing Pre-Cursor and Post-Cursor Coefficients. Coefficients are expressed as a positive integer ratio *c: c[5:0]/64*. For example, c[5:0] == 8 yields a ratio 8/64 or 0.125.

The remote PCI Express device advertises its PHY's Full Scale (FS) and Low Frequency (LF) values. The FS and LF values and the desired coefficients are used to compute the required coefficient format that is expressed to the remote device. The Lattice PCIe x8 IP Core does not violate the following PCI Express Coefficient rules. If violated, it limits the coefficient to its maximum allowed value:

- $|C\text{-}1| \leq \text{Floor}(FS/4)$
- $C0 - |C\text{-}1| - |C+1| \geq LF$
- $|C\text{-}1| + C0 + |C+1| = FS$

The Core applies the three rules above in the order illustrated. The Pre-Cursor and Post-Cursor coefficient absolute values are used in computations. The Pre-Cursor (C-1) is limited to Floor (FS/4). Then the Post-Cursor (C+1) is limited to ((FS-LF)/2) – C-1 (solving the latter two equations for C+1). Then the Cursor (C0) is computed to be FS – C-1 – C+1. The Lattice PCIe x8 IP Core supports both Figure of Merit and Up/Down PIPE PHY Equalization Methods.

The available Equalization algorithms are described in the following sections.

2.5.3.1. Figure of Merit – Preset Method

TX Preset (as defined by PCIe Specification) is sweep from Presets from Preset[0] to Preset [preset_method_control_addr_limit] and Figure of Merit for each preset is determined. The PCIe Specifications defines presets 0x0 through 0xA. Although, 0xA preset is intended as a diagnostic preset and generally should not be included.

```
for (i = 0; i ≤ eq_preset_method_control_addr_limit; i = i + 1) {
    pre_preset_coef = pre-cursor coef for Preset[i] // See Preset to Coefficient Conversion
    post_preset_coef = post-cursor coef for Preset[i] // See Preset to Coefficient Conversion
    pre = (pre_preset_coef * RemoteFS) / 64
    post = (post_preset_coef * RemoteFS) / 64
    // Core requests link partner PHY Tx use {post, pre} coefficients and then core performs a
Rx EQ Evaluation
    try {post, pre}
}
```

After all (*eq_preset_method_control_addr_limit+1*) trials are completed, an additional Rx EQ evaluation is executed using the post and pre coefficients from the trial which achieved the highest Figure of Merit. This final evaluation is skipped if the current coefficients being used are the best coefficients. If the final Figure of Merit meets or exceeds the threshold configured in *eq_fmerit_control_req_feedback*, the Rx Equalization is considered successful; otherwise, it is unsuccessful.

The preset method is optionally configured to communicate the desired preset for the remote device to select appropriate coefficients and matching presets or by calculating the coefficients equivalent to the preset number and communicating the coefficients to the remote device.

When the core is configured to communicate with the preset through the coefficients rather than preset number, the core uses the coefficients from the pre-cursor coefficient and post-cursor coefficient columns in Table 2.7 to perform the calculation. The coefficient target, expressed as a real number in parenthesis, is provided along with the rounded to 1/64 coefficient value that is used. The same conversion is also used in the other Figure of Merit methods when users need to convert from a preset number into the equivalent coefficients.

Table 2.7. Preset to Coefficient Conversion

Preset Encoding	De-Emphasis dB	Pre-Shoot dB	Pre-Cursor Coefficient	Post-Cursor Coefficient
0	-6	0	0 (0)	16 (-0.25)
1	-3.5	0	0 (0)	11 (-0.167)
2	-4.5	0	0 (0)	13 (-0.2)
3	-2.5	0	0 (0)	8 (-0.125)
4	0	0	0 (0)	0 (0)
5	0	2	6 (-0.1)	0 (0)
6	0	2.5	8 (-0.125)	0 (0)
7	-6	3.5	6 (-0.1)	13 (-0.2)
8	-3.5	3.5	8 (-0.125)	8 (-0.125)
9	0	3.5	11 (-0.167)	0 (0)
A	Special case: Pre-Cursor == 0, Post-Cursor == (FS-LF)/2			

The preset method works well with PHY, which take the maximum of 2 ms to evaluate equalization settings since only the 10 presets from Preset[0] through Preset[9] are evaluated. The available channels in the PCIe Specification is expected to correspond to one of the preset method that works well with PHY, which takes the maximum of 2 ms to evaluate equalization settings since only the 10 Presets from Preset[0] through Preset[9] are evaluated. The preset method is defined by the PCIe Specification to take the maximum of 2 ms to evaluate equalization settings since only the 10 Presets from Preset[0] through Preset[9] would be evaluated. Also, all available channels in the PCIe Specification are expected to correspond to one of the PCIe preset. Therefore, this method is recommended if users are unsure which method to use to evaluate Preset[0] to Preset[9].

2.5.3.2. Figure of Merit – Algorithm Method

This method sweeps through the possible coefficient values. The Algorithm Method enables complete coefficient range coverage at the expense of longer run time. Coefficient coverage ranges from coarse to fine depending upon how many iterations can be tried before the 24 ms Equalization time out.

```
pre_cursor_limit = eq_alg_method_control_pre_cursor_limit
post_cursor_limit = eq_alg_method_control_post_cursor_limit
pre_cursor_step_size = eq_alg_method_control_pre_cursor_step_size
post_cursor_step_size = eq_alg_method_control_post_cursor_step_size
for (pre_coef = 0; pre_coef ≤ pre_cursor_limit; pre_coef = pre_coef + pre_cursor_step_size) {
    for (post_coef = 0; post_coef ≤ post_cursor_limit; post_coef = post_coef +
post_cursor_step_size) {
        pre = (pre_coef × RemoteFS) / 64
        post = (post_coef × RemoteFS) / 64
        // Core requests link partner PHY Tx use {post, pre} coefficients and then core
performs a Rx EQ Evaluation
        try {post, pre}
    }
}
```

After all trials are completed, one additional Rx EQ Evaluation is executed using the {post, pre} coefficients from the trial which achieved the highest Figure of Merit. If the final Figure of Merit meets or exceeds the threshold configured in eq_fmerit_control_req_feedback, the Rx Equalization is considered successful otherwise unsuccessful. This final evaluation is skipped if the current coefficients being used are the best coefficients.

Notes:

- Pre-Cursor coefficients (pre) from 0 to 16 (0 to 0.25) are possible.
- Post-Cursor coefficients (post) from 0 to 32 (0 to 0.5) are possible.

Stepping through all 17 (0-16) Pre-Cursor values and all 33 (0-32) Post-Cursor values takes 561 iterations. Step size is increased to walk through the values more quickly (and coarsely). Limits are lowered to exclude larger values that are less likely to produce the desired results.

Example:

- Steps of 4 for Pre-Cursor and 8 for Post Cursor with Limits == 16 and 32 respectively requires 25 iterations.
- Steps of 8 for Pre-Cursor and 16 for Post Cursor with Limits == 16 and 32 respectively requires 9 iterations.

Caution: Be careful when assigning eq_table_method_control_addr_limit not to exceed the Equalization time limit. Refer to Equalization Time Limit section for more details.

The Algorithm Method works best with PHY, which takes significantly less than the maximum of 2 ms to evaluate equalization settings so that the fine step sizes can be used.

2.5.3.3. Figure of Merit – Table Method

This method selects and sweeps the preset or coefficients configured in the table.

```
for (i = 0; i ≤ eq_table_method_control_addr_limit; i = i + 1) {
    // Read the current table entry
    table_pre         = eq_table_method_table_array[(i×16)+5:(i×16)+0]
    table_post        = eq_table_method_table_array[(i×16)+11:(i×16)+6]
```

```
    table_interpret = eq_table_method_table_array[(i×16)+13:(i×16)+12]
    table_best      = eq_table_method_table_array[(i×16)+14]              // unused in
this method
    // Interpret the table entry
    if (table_interpret == 0) { // Current table entry specifies a preset
        pre_preset_coef = pre-cursor coef for Preset[table_pre[3:0]] // See Preset Method for
coefficients used
        post_preset_coef = post-cursor coef for Preset[table_pre[3:0]] // See Preset Method for
coefficients used
        pre = (pre_preset_coef × RemoteFS) / 64
        post = (post_preset_coef × RemoteFS) / 64
    } else { // Current table entry specifies coefficients
        pre = (table_pre[5:0] × RemoteFS) / 64
        post = (table_post[5:0] × RemoteFS) / 64
    }
    // Core requests link partner PHY Tx use {post, pre} coefficients and then core performs a
Rx EQ Evaluation
    try {post, pre}
}
```

After all (eq_table_method_control_addr_limit+1) trials are completed, one additional Rx EQ Evaluation is executed using the {post, pre} coefficients from the trial which achieved the highest Figure of Merit. This final evaluation is skipped if the current coefficients being used are the best coefficients. If the final Figure of Merit meets or exceeds, the threshold configured in *eq_fmerit_control_req_feedback* then Rx Equalization is considered successful otherwise unsuccessful.

Notes:

- Pre-Cursor coefficients (pre) from 0 to 16 (0 to 0.25) are possible.
- Post-Cursor coefficients (post) from 0 to 32 (0 to 0.5) are possible.

In this method, you specify up to 24 Preset/coefficients to try and may select the preset/coefficients that are most likely to work for the given PHY.

The Table Method works well for users that know the range of coefficients, which typically works well for the PHY since the table values can be concentrated on coefficient ranges that are more likely to work well. The Table Method also provides a lot of flexibility and can be configured easily.

Caution: Be careful when assigning *eq_table_method_control_addr_limit* not to exceed the Equalization time limit. Refer to Equalization Time Limit section for more details.

2.5.3.4. Figure of Merit – Adaptive Table Method

This method sweeps through the user-provided table with adaptive coefficient selection. The Adaptive Table Method is similar to the Table Method and uses the same user-configured table. Although, the Adaptive Table Method may be configured to select latter coefficients dynamically from the results of earlier equalization evaluations.

```
// Initialize the current Up/Down best coefficient pair {best_post, best_pre} to {0,0}
best_pre = 0; best_post = 0; best_fom = 0;
// Initialize the current Relative best coefficient pair {rel_best_post, rel_best_pre} to {0,0}
rel_best_pre = 0; rel_best_post = 0;
for (i = 0; i ≤ eq_table_method_control_addr_limit; i = i + 1) {
    // Read the current table entry
    table_pre         = eq_table_method_table_array[(i×16)+5:(i×16)+0];
    table_post        = eq_table_method_table_array[(i×16)+11:(i×16)+6];
    table_interpret = eq_table_method_table_array[(i×16)+13:(i×16)+12];
    table_best        = eq_table_method_table_array[(i×16)+14];
    // Interpret the current table entry
    if (table_interpret == 0) { // Current table entry specifies a preset
```

```
        pre_preset_coef = pre-cursor coef for Preset[table_pre[3:0]]; // See Preset Method for
coefficients used
        post_preset_coef = post-cursor coef for Preset[table_pre[3:0]]; // See Preset Method
for coefficients used
        pre = (pre_preset_coef × RemoteFS) / 64;
        post = (post_preset_coef × RemoteFS) / 64;
    } else if (table_interpret == 1) { // Current table entry specifies coefficients
        pre = (table_pre[5:0] × RemoteFS) / 64;
        post = (table_post[5:0] × RemoteFS) / 64;
     } else if (table_interpret == 2) { // Table entry contains relative coefficient offsets
        pre = table_pre[5] ? (rel_best_pre - table_pre[1:0]) : (rel_best_pre + table_pre[1:0]);
        post = table_post[5] ? (rel_best_post - table_post[1:0]) : (rel_best_post +
table_post[1:0]);
    } else { // Use the prior evaluation's PHY Up/Down Feedback instead of table_pre,
table_post
        if (prior trial used (table_interpret == 3)) { // Use prior trial feedback when
continuing (table_interpret==3)
            case (Prior trial's predir)
                10        : pre = best_pre – 1;
                01        : pre = best_pre + 1;
                default   : pre = best_pre;
            endcase
            case (Prior trial's postdir)
                10        : post = best_post – 1;
                01        : post = best_post + 1;
                default   : post = best_post;
            endcase
        } else { // Begin (table_interpret == 3) trials from the current best coefficient pair
            pre  = best_pre; post = best_post;
        }
    }
    // Core requests link partner PHY Tx use {post, pre} coefficients and then core performs a
Rx EQ Evaluation
    {fom, postdir, predir} = try {post, pre} // fom == Figure of Merit feedback; post/predir ==
directional feedback
    // Better result or, when using Up/Down feedback, the most recent trial is always
considered the best
    if ((fom ≥ best_fom) | (table_interpret == 3)) {
        best_fom = fom
        best_pre = pre
        best_post = post
    }
    if (table_best == 1) {
        rel_best_pre = best_pre
        rel_best_post = best_post
    }
    if ((table_interpret == 3) and ({postdir, predir} == {0, 0}) and core configured to end on
hold) // {Hold, Hold}
        exit for i loop
}
```

After all (*eq_table_method_control_addr_limit+1*) trials are completed, one additional Rx EQ evaluation is executed using the {post, pre} coefficients from the trial, which achieved the highest Figure of Merit. If the final Figure of Merit meets or exceeds, the threshold configured in *eq_fmerit_control_req_feedback* then Rx Equalization is considered successful otherwise unsuccessful. This final evaluation is skipped if the algorithm ends using table_interpret == 3 (Up/Down Feedback) or the best coefficients are already in use.

Notes:

- Pre-Cursor values (pre) from 0 to 16 (0 to 0.25) are possible.
- Post-Cursor values (post) from 0 to 32 (0 to 0.5) are possible.

The Adaptive Table includes 24 table entries.

While performing the algorithm, coefficient subtractions are limited to 0, coefficient additions are limited to 63 and coefficients are limited (as in all cases), when necessary, to comply with the PCIe Specification coefficient rules.

The Adaptive Table Method can perform a coarse search to quickly identify the region with the best Figure of Merit and to select the best specific coefficients in that region with a fine-grained search. The Adaptive Table Method works well for a broad range of conditions. Although, it works best when 10 to 20 iterations are possible so that it has time to select the best coefficients.

The Adaptive Table Method is very flexible and supports many different use models. The following use models are suggested:

- Coarse Figure of Merit search followed by fine PHY-directed Up/Down search.
- Fill the first N table entries with presets or coefficients that are widely dispersed setting interpret == 0 (absolute Preset) or 1 (absolute coefficients) as desired. Choosing Presets/coefficients that coarsely cover the coefficient space is recommended. Set best == 1 on all N table entries. This results in the following Up/Down feedback evaluations starting from the preset/coefficients that received the highest Figure of Merit in the first N trials.
- Fill the next M table entries with pre == 0 (unused), post == 0 (unused), interpret == 3 (use the prior evaluation's PHY Up/Down Feedback), and best=0. This results in M trials where the PHY controls the next coefficients through its Up/Down/Hold feedback.
- The algorithm finishes the M trials and then exits Equalization. However, an early exit occurs if eq_table_method_control_end_on_hold==1 and all lanes provided a {Hold, Hold} response.
- Coarse Figure of Merit search followed by fine Figure of Merit search.
- Fill the first N table entries with presets or coefficients that are widely dispersed, setting interpret == 0 (absolute Preset) or 1 (absolute coefficients) as desired. Choosing Presets/coefficients that coarsely cover the coefficient space is recommended. Set best == 1 on all N table entries. This results in the following relative offset evaluations starting from the Preset/coefficients that received the highest Figure of Merit in the first N trials.
- Repeat the following sequence as many times as desired and the Equalization time limit allows.
- Fill the next four table entries with relative offsets moving up, down, right, and left by 1 and setting best==1 on only the fourth table entry:
  - pre == 01 (+1), post == 00 (+0), interpret == 2 (relative), best=0
  - pre == 20 (-1), post == 00 (+0), interpret == 2 (relative), best=0
  - pre == 00 (+0), post == 01 (+1), interpret == 2 (relative), best=0
  - pre == 00 (+0), post == 20 (-1), interpret == 2 (relative), best=1
- After each four-table entry iteration, the coefficient may move up to 1 pre or 1 post from the prior best coefficient pair and the new best coefficient pair is selected as the new relative starting location for subsequent iterations. After three iterations of four table entries each, it is possible for the algorithm to move pre or post up to +/- 3 from the starting best coefficient pair established by the first N trials.
- This algorithm can be modified to move in initially larger relative steps (for example, +2 instead of +1), to use different combinations of relative step directions, or to use more offsets in each trial. Additionally, other offset shapes can be utilized. For example, using a combination of X shape ({+1,+1}, {+1,-1}, {-1,-1}, {-1,+1} relative movements and + shape ({+1,0}, {-1, 0}, {0,+1}, {0,-1} relative movements enables the coefficient space to be covered more or less quickly and more or less completely.

The Adaptive Table Method is flexible enough to reproduce the other Equalization Methods. However, the other methods have been kept due to the simplicity to which they can be enabled and configured.

Caution: Be careful when assigning *eq_table_method_control_addr_limit* not to exceed the Equalization time limit. Refer to Equalization Time Limit section for more details.

2.5.3.5. Up/Down – Up/Down Convergence Method

The Up/Down convergence method supports PHYs that provide Up/Down feedback response. The Up/Down Convergence Method is described as follows:

- Request the remote device to use an initial set of coefficients. These coefficients can be programmed uniquely for each lane or alternatively, the first coefficients can be provided as a preset.
- After the remote device begins transmitting with the new coefficients, the PHY is told to evaluate the receive waveform (RxEval). The PHY then provides the up/down response for the pre-cursor and post-cursor coefficients.
- The state machine adjusts the pre-cursor and post-cursor coefficients based on the feedback from the PHY. It starts with larger adjustments (initialized with eq_updn_pre_step and eq_updn_post_step). Each time the feedback changes up/dn direction, the step size is halved until it is equal to 1. When the step size is 1 and a direction change occurs in the feedback, then the coefficient is considered converged.
- If any coefficient changes value, the alternate coefficient is allowed to continue to change, even if previously converged since the two coefficients work together.
- If a hold result is received from the PHY, then the coefficient is considered converged. In some PHYs, two hold results in a row must be seen to be considered a hold since only one coefficient is evaluated with each request (the other having a null hold status). In this case, the configuration bit eq_updn_numhold must be set to 1 so the state machine knows to look for two holds.
- Once both coefficients on all active lanes reach convergence, the equalization is complete.

2.5.3.6. Up/Down – Firmware Controlled Method

The Firmware Controlled Method is described as follows:

- The Lattice PCIe x8 IP Core is configured for Firmware to receive an interrupt at the beginning of Rx Equalization (Phase 2 entry for US port and Phase 3 entry for DS port). When an Interrupt is received by Firmware, the Firmware begins a time-critical processing loop to perform Rx Equalization in a minimum amount of time as there is a PCIe Specification-mandated 24 ms time limit to complete Rx Equalization.
- Firmware receives interrupt and begins Rx Equalization evaluation.

```
// Firmware keeps performing Equalization trials until Rx Equalization is complete
done = 0
while (done == 0) {
    Firmware calculates a coefficient pair to try
    Firmware writes the trial coefficients into the core’s registers and sets advance=1 to
begin an evaluation
    Firmware enters a polling loop to wait for the core to complete the requested
evaluation {
       Core negotiates with the link partner to change to the desired coefficients
       Core performs an Rx Equalization evaluation
       Core updates Equalization status registers with the results of the evaluation
       Core sets Equalization status register complete == 1 to inform firmware that the
evaluation is done
       Firmware reads complete == 1 while polling and exits the polling loop
    }
    Firmware reads Equalization status registers to obtain Rx Equalization status
    if Firmware is satisfied with the Equalization status {
        Firmware sets the core’s complete==1 register to exit Rx Equalization
        done = 1
    }
}
```

- Firmware may configure the core to output interrupts instead of Firmware polling for Equalization completion. Firmware may configure the core receives an interrupt each time that the core is ready to receive a new set of coefficients. Whichever method is selected, Equalization must complete in < 24 ms to avoid the PCIe Specification-required LTSSM timeout, so it is critical that the Firmware algorithm be designed to be able to complete within the time period. Refer to Equalization Time Limit section for more details.

2.5.4. Equalization Quality

The Figure of Merit Equalization processes rely on the PHY to perform Receiver Equalization using the current remote transmitter settings and produce a quality result (pipe_eq_rx_eval_feedback_fom) that can be used to gauge the quality of the link. Quality is defined by the PHY. For the Figure of Merit method, the core contains a field in Management Interface to determine what PHY quality level corresponds to the necessary 10-12 Bit Error Rate (BER). For the Figure of Merit method, higher quality numbers represent better link quality (lower BER).

When using the Up/Down Convergence method, the acceptable link quality is assumed when the PHY converges on a set of coefficients or exits its algorithm due to reaching the programmed maximum iteration count.

2.6. Multi-Function Support

The Lattice PCIe x8 IP Core supports 1 to 4 functions. The Multi-Function support can only be enabled for endpoints (functions implementing Type 0 Configuration Space). See the function register 0x8 for the register configuration.

When Multiple Function support is present, each function is assigned a static function number, starting at function number 0 and incrementing upwards. For ports that communicate function-specific information, port[0] applies to Function[0], port[1] applies to Function[1]. If a function is disabled, it does not affect the function number of the other enabled functions. Function [0] is always present and cannot be disabled.

2.7. Power Management

2.7.1. Power Management Supported by PCIe IP Core

The Lattice PCIe x8 IP Core supports L0, ASPM L0s, ASPM L1, L1 PM Substates, L1, and L3 link states. L0 (fully-operational state) and L3 (off) support is always enabled. The remaining link states may be enabled/disabled through the Core Configuration ports. If ASPM L0s, ASPM L1, and L1 PM Substates, or L1 support is enabled, then the user design must configure the power management capabilities of the core and for some link states, take additional action when link states are entered or exited. This section describes the recommended actions user logic should take to control and react to power management states ASPM L0s, ASPM L1, and L1.

The PCI Express Specification defines the following link states:

- L0 – Active
  - Powered
  - Clock and PLLs active; core clock active
  - All PCI Express Transactions and operations are enabled
- ASPM L0s – Low resume latency, energy saving *standby* state
  - Powered
  - Clock and PLLs active; core clock active
  - PHY transmitter in electrical idle
  - Remote PHY receiver must re-establish symbol lock during L0s exit

- When L0s is enabled by power management software, the core autonomously enters L0s when the transmit side of the link is idle and exits L0s when there is pending information to transmit. The link management DLLPs are required to be transmitted periodically so when a link is otherwise idle, it still enters and exits L0s with regularity to transmit link management DLLPs.

- ASPM L1 – Low resume latency, energy saving *standby* state
  - Powered
  - Clock and PLLs active; core clock active
  - Significant portion of PHY powered down
  - PHY transmitter in electrical idle
  - PHY receiver in electrical idle
  - Deeper power savings but longer resume time than ASPM L0s
  - Remote and local PHY must re-establish symbol lock during L1 exit
  - When ASPM L1 is enabled by power management software, the core autonomously negotiates L1 entry with the link partner after an extended period of link inactivity. The link autonomously returns to L0 when either device in the link has TLPs to transmit.
- L1 – Higher latency, lower power *standby* state
  - Powered
  - Clock and PLLs active; core clock active
  - Significant portion of PHY powered down
  - PHY transmitter in electrical idle
  - PHY receiver in electrical idle
  - Remote and local PHY must re-establish symbol lock during L1 exit
  - The L1 state is entered both under control of power management software
  - L3 – Off
  - Main power off; auxiliary power off
  - In this state, all power is removed and the core, PHY, and user logic are all non-operational
  - All state information is lost

a.

2.7.2. Configuring Core to Support Power Management

The Lattice PCIe IP x8 Core allows user logic to implement a wide variety of power management functionality. The design's power management capabilities are primarily advertised and controlled using core configuration ports.

2.7.3. ASPM L0s

The Lattice PCIe x8 IP Core supports Active State Power Management (ASPM) L0s. When L0s support is enabled, ASPM L0s TX Entry Time, the desired amount of time for TLP and DLLP transmissions to be idle before L0s TX is entered, is determined by mgmt_ptl_pm_aspm_l0s_entry_time. The Number of NFTS sets required by the local PHY to recover symbol lock when exiting L0s is determined by mgmt_tlb_ltssm_nfts_nfts. NFTS Timeout Extend, mgmt_tlb_ltssm_nfts_to_extend (see Table 5.12), controls how long the core waits after the expected L0s exit time before directing the link to Recovery to recover from a failed L0s exit. Due to high latencies between a PHY's Rx Electrical Idle output and the associated Rx Data it is normally necessary to choose a relatively high NFTS and NFTS Timeout Extend. See mgmt_tlb_ltssm_nfts_to_extend description (Table 5.12) for details.

- Configuration Register Fields
  - The PCI Express Device Capabilities configuration register has the following L0s fields:
    - Bits [8:6] – Endpoint L0s Acceptable Latency – From PCI Express Base Specification, Rev 2.1 section 7.8.3 – Acceptable total latency that an Endpoint can withstand due to the transition from L0s state to the L0 state. It is essentially an indirect measure of the Endpoint's internal buffering. Power management software uses the reported L0s Acceptable Latency number to compare against the L0s exit latencies reported by all components comprising the data path from this Endpoint to the Root Complex Root Port to determine whether ASPM L0s entry can be used with no loss of performance. Note that the amount of buffering does not refer to the Lattice PCIe x8 IP Core buffering, but rather to user application buffering. You must set this field in accordance with how long a delay is acceptable for the application.

- 000 – Maximum of 64 ns
- 001 – Maximum of 128 ns
- 010 – Maximum of 256 ns
- 011 – Maximum of 512 ns
- 100 – Maximum of 1 µs
- 101 – Maximum of 2 µs
- 110 – Maximum of 4 µs
- 111 – No limit
- Non-Endpoints must hard wire this field to 000

- The PCI Express Link Capabilities configuration register has the following L0s fields:
  - Bits[14:12] – L0s Exit Latency – Length of time required to complete transition from L0s to L0:
    - 000 – Less than 64 ns
    - 001 – 64 ns to less than 128 ns
    - 010 – 128 ns to less than 256 ns
    - 011 – 256 ns to less than 512 ns
    - 100 – 512 ns to less than 1 µs
    - 101 – 1 µs to less than 2 µs
    - 110 – 2 µs-4 µs
    - 111 – More than 4 µs
    - Exit latencies may be significantly increased if the PCI Express reference clocks used by the two devices in the link are common or separate.
  - Bits[11:10] – Active State Power Management (ASPM) Support must be set to 01 or 11 if L0s support is enabled or 00 otherwise.

2.7.4. ASPM L1s

The Lattice PCIe x8 IP Core supports both software controller L1 entry (through the Power State Configuration Register) and hardware autonomous L1 entry (Active State Power Management (ASPM) L1).

- Software-controlled L1 flow for Upstream Ports (Endpoint) is as follows:
  - Software initiates changing a link to L1 by writing the core's Power Management Capability: Power State Configuration Register to a value other than 00 == D0. Note that the component's Device driver participates in this process and must ensure that all traffic is idle before permitting the system to power down to L1.
  - When the core detects a change of Power State to a non-D0 value, the core's power management state machine, which is responsible for the higher-level power management protocol, follows the following sequence:
    - Block further TLP transmissions
    - Wait for all in process TLPs to complete transmission
    - Wait for the Replay Buffer to empty (all transmitted TLPs acknowledged)
    - Core transmits PM_ENTER_L1 DLLPs until receiving a PM_REQ_ACK DLLP from remote device
    - Core directs LTSSM state machine to L1
    - When a TLP is pending or the LTTSM state machine indicates L1 state has been exited due to link partner activity, the core returns to L0.
- Configuration Register Fields:
  - The PCI Express Device Capabilities configuration register has the following L1 fields:
    - Bits [11:9] – Endpoint L1 Acceptable Latency – From PCI Express Base Specification, Rev 2.1 section 7.8.3 – This field indicates the acceptable latency that an Endpoint can withstand due to the transition from L1 state to the L0 state. It is essentially an indirect measure of the Endpoint's internal buffering. Power management software uses the reported L1 Acceptable Latency number to compare against the L1 Exit Latencies reported (see below) by all components comprising the data path from this Endpoint to the Root Complex Root Port to determine whether ASPM L1 entry can be used with no loss of performance. Note that the amount of buffering does not refer to Lattice PCIe x8 IP Core buffering, but rather to user application buffering. You must set this field in accordance with how long a delay is acceptable for the application.
      - 000 – Maximum of 1 µs
      - 001 – Maximum of 2 µs

    - 010 – Maximum of 4 µs
    - 011 – Maximum of 8 µs
    - 100 – Maximum of 16 µs
    - 101 – Maximum of 32 µs
    - 110 – Maximum of 64 µs
    - 111 – No limit
    - Non-Endpoints must hard wire this field to 000.
  - PCI Express Link Capabilities configuration register has the following L1 fields:
    - Bits[17:15] – L1 Exit Latency – Length of time required to complete transition from L1 to L0:
      - 000 – Less than 1 µs
      - 001 – 1 µs to less than 2 µs
      - 010 – 2 µs to less than 4 µs
      - 011 – 4 µs to less than 8 µs
      - 100 – 8 µs to less than 16 µs
      - 101 – 16 µs to less than 32 µs
      - 110 – 32 µs-64 µs
      - 111 – More than 64 µs
      - Exit latencies may be significantly increased if the PCI Express reference clocks used by the two devices in the link are common or separate.
    - Bits[11:10] – Active State Power Management (ASPM) Support must be set to *10* or *11* if L1 support is enabled or *00* otherwise.
- Hardware-autonomous L1 (ASPM L1) entry is initiated only by Upstream Ports (Endpoint). The core ASPM L1 functionality must be enabled and advertised in PCIe Link Capabilities and software must enable ASPM L1 support for the hardware-autonomous L1 to be negotiated. When ASPM L1 support is present and enabled for an Upstream Port, the core requests the link to be directed to L1 using the ASPM L1 protocol, when the link is idle. The link idle refers to the no TLPs or ACK/NAL DLLPs being transmitted.

2.8. DMA Support (Hardened)

The Lattice PCIe x8 IP Core has a DMA Bridge Core that implements high-performance DMA and bridging between PCI Express and AXI. The following are the key features of the DMA Bridge Core.

- AXI Manager Interface
  - High-performance AXI Manager to complete PCIe to AXI bridge and DMA transactions.
  - 64-bit address support.
  - Includes support for up to 8 PCIe to AXI address translations.
- AXI Subordinate Interface
  - High-performance AXI Subordinate to complete AXI to PCIe bridge transactions.
  - 64-bit address support.
  - Includes support for up to 8 AXI to PCIe address translations.
- High-performance Multi-Channel DMA
  - Up to 64 DMA channels support
  - Each DMA channel supports 4 directions of DMA data transfer.
    - PCIe to PCIe, PCIe to AXI, AXI to PCIe, and AXI to AXI

Note: For bridge core support, user needs to ensure that an additional BAR (BAR1 – BAR5) is enabled.

2.8.1. DMA Scatter-Gather and Status Queues

The DMA channels can perform Scatter-Gather separately for the DMA Source and DMA Destination. A Source Scatter Gather and a Destination Scatter Gather Queue is managed by software and the DMA Channel to describe the desired DMA operations. The DMA Queue Registers describe the Starting Queue Address Offsets, Queue Size, and communicate flow control ownership between software and the DMA Channel hardware.

The software provides Source and Destination Queue elements to the DMA Channel to enable the DMA Channel to perform a DMA operation. When the DMA Channel is enabled, the available Source and Destination Queue elements are fetched into a small internal buffer, on need, to complete the DMA operations.

Queue elements are a fixed size. The address of a particular element in the queue is the queue starting address plus the element index multiply the element size.

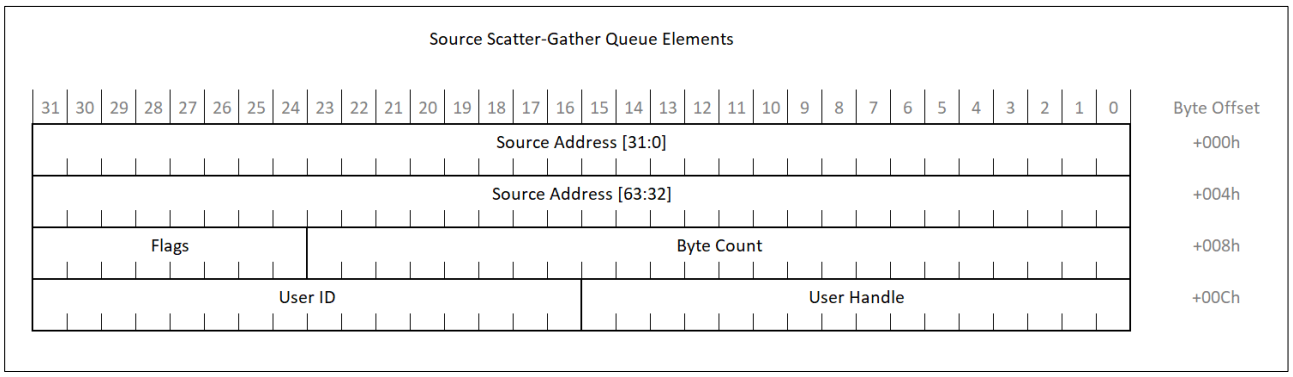


Figure 2.8. Source Scatter-Gather Queue Elements

The Source Scatter-Gather Queue elements are 128-bits.

- [63:0] – Source Address [63:0]
- [87:64] – Byte Count [23:0]
  - 0 – 2^24 bytes
- [95:88] – Flags [7:0]
  - [7:4] – DMA Data Read Attributes
    - If source is AXI [7:4] is used for m0_axi_arcache_o [3:0]
    - If source is PCIe [6:4] is used for PCIe Attributes [2:0]
  - [3] – Reserved.
  - [2] – Interrupt
    - 1 – Generate an interrupt event when the Status Queue is written with DMA Completion Status; only valid when EOP is 1; the interrupt will be generated on PCIe, AXI, or both per the DMA Channel's interrupt registers.
    - 0 – Do not generate an interrupt
  - [1] – EOP
    - 1 – End of packet; a status Queue Element containing DMA packet completion status is written whenever an End of Packet is transferred to the DMA Destination
    - 0 – Not the end of a packet; packets may span multiple Descriptors.
  - [0] – Location
    - 1 – DMA data source is AXI
    - 0 – DMA data source is PCIe

- [111:96] – User Handle [15:0]
  - User Handle is copied from Source Scatter-Gather Queue element when EOP is 1 into the corresponding Source DMA Completion Statue Queue elements' User Handle field.
- [127:112] – User ID [15:0]
  - User ID is copied from Source Scatter-Gather Queue element when EOP is 1 into the corresponding Source DMA Completion Statue Queue elements' User ID field.

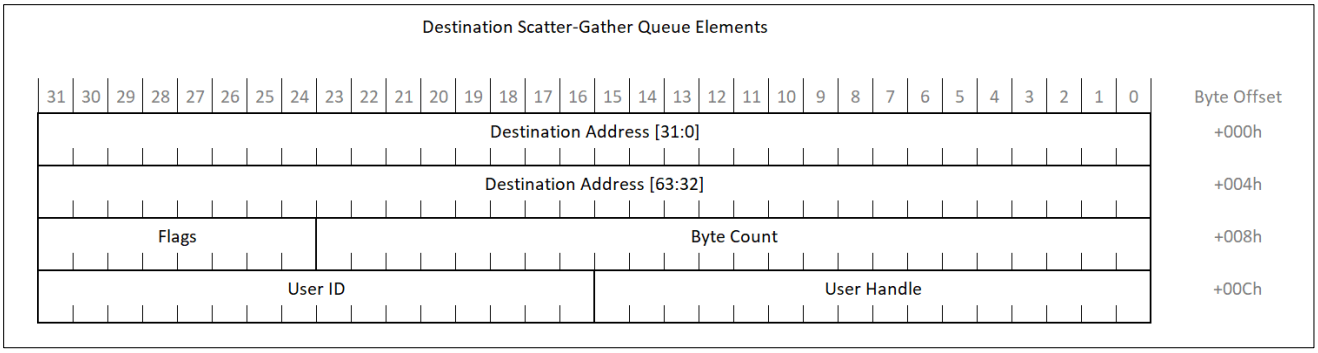


Figure 2.9. Destination Scatter-Gather Queue Elements

The Destination Scatter-Gather Queue elements are 128-bits.

- [63:0] – Destination Address [63:0]
- [87:64] – Byte Count [23:0]
  - 0 – 2^24 bytes
- [95:88] – Flags [7:0]
  - [7:4] – DMA Data Write Attributes
    - If source is AXI [7:4] is used for m0_axi_arcache_o [3:0]
    - If source is PCIe [6:4] is used for PCIe Attributes [2:0]
  - [3:2] – Reserved.
  - [1] – Enable One Packet Per Destination Scatter-Gather Queue
    - 1 – Skip to next Destination Scatter-Gather element on EOP
      - Whenever an end of packet is transferred (as determined by the Source Scatter-Gather element EOP field) the remaining portion of the current Destination Scatter-Gather element (if any) is not used. The next packet starts being written into the beginning of the memory described by the following Destination Scatter-Gather element.
    - 0 – Pack Packets Back-to-Back in Destination Scatter-Gather Element
      - When a packet ends without consuming the entire Destination Scatter-Gather element, the next packet begins will be written into the remaining portion of the current Destination Scatter-Gather element location at the byte following the end of the prior packet.
  - [0] – Location
    - 1 – DMA data source is AXI
    - 0 – DMA data source is PCIe

- [111:96] – User Handle [15:0]
  - User Handle is copied from the final destination Scatter-Gather element used for a packet transfer into the corresponding Destination DMA Completion Status Queue element User Handle field.
- [127:112] – Reserved [15:0]

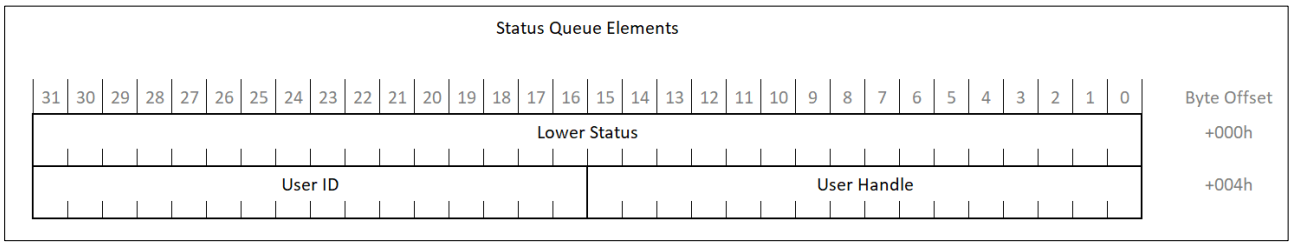


Figure 2.10. Status Queue Elements

The Statue Queue elements are either 32-bit or 64-bit as determined by software DMA Channel register programming.

- [63:48] – User ID [15:0]
  - User ID is copied from Source Scatter-Gather element with EOP equals to 1 into the corresponding Source and Destination DMA Completion Status Queue elements' User ID fields.
  - User ID use is application specific. The DMA Channel copies but does not itself use this information.
- [47:32] – User Handle [15:0]
  - DMA Source Completion Status Queue – User Handle is copied from Source Scatter-Gather element with EOP equals to 1.
  - DMA Destination Completion Status Queue – User Handle is copied from the final Destination Scatter-Gather element used for the packet.
  - User Handle use is application specific. The DMA Channel copies but does not itself use this information.0 – 2^24 bytes
- [31] – Upper Status is Non-Zero
  - For 64-bit status elements, Upper Status Is Non Zero equals to 1 when Status Queue Element bits [63:32] is non-zero. In some cases, CPUs are only capable of 32-bit atomic transactions. For such CPUs, the software reads status bits [31:0]. If bit [31] equals to 1, then software knows that Status Queue Element [63:32] is non-zero and reads status bits [63:32] until a non-0 value is returned. Software must clear all status Queue elements before giving them to the DMA Channel, so that the upper status bits are zero until updated by a DMA Completion.
  - For 32-bit status elements, this field is unused and always reads 0.
- [30:4] – Completed Byte Count [26:0]
  - The possible range is 0 to (2^27)-1 bytes (0 to 128MByte-1). If a DMA exceeds (2^27)-1 bytes, Completed Byte Count limits at its maximum value. If software needs to transfer larger packets than 128MB-1 and keep an accurate byte count, then software must split such transfers into smaller DMA transfers that are < 128 MB.
- [3] – Internal Error Detected during DMA Operation
- [2] – Destination Error Detected during DMA Operation
- [1] – Source Error Detected during DMA Operation
- [0] – Completed

2.8.2. DMA Channel Programming and Operation

Each DMA Channel implements an independent register set for controlling that DMA channel's DMA operations. Registers are provided for Scatter-Gather and Completion Status Queue Management and DMA Control and Status. DMA registers are implemented in structures that can be implemented mostly in RAM to enable large numbers of DMA Channels to be efficiently supported.

DMA registers are described in the *DMA_CHANNEL* register section. When Multi-Function and/or SR-IOV capable PCI Express Cores are used, DMA Channels are allocated in equal portions to each function.

DMA Queues are required to have contiguous memory and address aligned to 64 bytes to enable efficient read/write of multiple queue elements in one read/write operation. This enables higher DMA bandwidth for smaller DMA transfer sizes since the overhead of queue management operations can be amortized over multiple small DMA transfers.

Source and Destination Scatter Gather Queues describe the fragmentation of the Source and Destination memory respectively. The queues are independent. The DMA Channel merges the information from the queues to perform DMA operations considering each Queue's fragmentation. This relieves software from this computation-intensive task and the necessity to pass the fragmentation information between the PCIe and AXI domains.

Prior to enabling the DMA Channel, the Source Scatter-Gather Queue, Destination Scatter-Gather Queue, DMA Source Completion Status Queue, and DMA Destination Completion Status Queue must be initialized. The following DMA Channel Enable sequence is recommended. The process can be completed in any order desired provided DMA Enable is written to 1 last:

- Verify that the DMA Channel is Idle.
  - Read DMA Running and verify it reads 0x0.
- Initialize queue base address and attributes.
  - Write SRC_Q_PTR_LO and SRC_Q_PTR_HI with the base address of the queue.
  - Write DST_Q_PTR_LO and DST_Q_PTR_HI with the base address of the queue.
  - Write STAS_Q_PTR_LO and STAS_Q_PTR_HI with the base address of the queue.
  - Write STAD_Q_PTR_LO and STAD_Q_PTR_HI with the base address of the queue.
- Initialize queue size.
  - Write SRC_Q_SIZE to the size of the queue.
  - Write DST_Q_SIZE to the size of the queue.
  - Write STAS_Q_SIZE to the size of the queue.
  - Write STAD_Q_SIZE to the size of the queue.
- Initialize queue Next pointers to the beginning of the queue.
  - Write SRC_Q_NEXT = 0x0
  - Write DST_Q_NEXT = 0x0
  - Write STAS_Q_NEXT = 0x0
  - Write STAD_Q_NEXT = 0x0
- Initialize Scatter Gather Queues to the Empty Condition (no DMA operations to execute)
  - Write SRC_Q_LIMIT = 0x0
  - Write DST_Q_LIMIT = 0x0
- Initialize Status Queues to the Fully Available Condition (all Status Queue elements except one, which is needed to preserve software flow control, are available)
  - STAS_Q_LIMIT set to (STAS_Q_SIZE-1)
  - STAD_Q_LIMIT set to (STAD_Q_SIZE-1)
- Initialize all STAS and STAD Queue Elements to 0x0.
  - DMA Completion Status Queue elements must be initialized to 0x0 so that software can tell when Status elements complete. Status elements return a non-0 value when complete.
- Optionally Initialize all SRC and DST Scatter Gather Queue Elements.
  - Source and Destination SGL Elements may be initialized or not at software discretion. Source and Destination SGL Elements will not be fetched until they have been filled in with DMA transaction instructions and the associated queue's LIMIT pointer advanced to give these elements to the DMA Channel to execute.
  - Write DMA Enable = 1 to enable the DMA Channel.

DMA transactions are given to the DMA Channel to execute by writing Source SGL and Destination SGL into the SRC/DST SGL Queues and incrementing the SRC/DST_Q_LIMIT registers.

When the DMA Channel completes a DMA operation (completes the transfer of a Source SGL that had EOP ==1), the STAS/STAD Completion Status Queues are updated with the DMA Completion Status and then a DMA Interrupt is generated (as conditioned by the DMA Channel Interrupt Coalesce Count and other registers).

Queue Management registers *_PTR_LO, *_PTR_HI, *_SIZE, and *_NEXT must not be written when the DMA Channel is enabled. The only queue registers that it is permissible to modify while the DMA Channel is enabled is the SRC/DST/STAS/STAD_Q_LIMIT registers which are incremented to provide additional elements for the DMA Channel to execute.

The minimum Queue Size is 2. This is because software must always retain ownership of one Q element for Flow Control. The DMA Channel does not execute/use the Queue element pointed to by the Queue LIMIT pointer.
The minimum Queue Size must be large enough to hold at least 1 full DMA transaction of maximum size. DMA Completion Status Queues are only written when a Source SGL Element is completed that had its EOP flag == 1 (end of a DMA transaction). If the Queue is too small to be able to place all the SGL for a single DMA transaction into the Queue, then the SGL with EOP == 1 will not be able to be added to the Queue and the DMA operation will not be able to complete. S/W will not be able to free Queue elements because no SGL with EOP can complete (because none is in the Queue) and no new SGL can be given to the DMA Channel unless Queue elements are freed.

A Queue Size of N has N Q elements: [0], [1], …, [N-1]. For example, a Q Size of 2 has [0] and [1] elements. The Queue wraps at the N-1 element. For example, for a Queue Size of 2 the following wrap occurs: [0], [1], [0], …The DMA Queues are intended to be setup once and re-used for multiple DMA operations. The DMA Queues are designed to enable highly overlapped transactions. Software can be setting up new DMA operations in the Queue while the DMA Channel is executing operations that software placed in the queue earlier.

The DMA Queues can also be set up for each DMA transaction if desired, although this method has lower performance, due to the need to reconfigure the queues between DMA transactions. DMA Queues may only be reconfigured when the DMA Channel is disabled. So, the process would be wait for all outstanding DMA transactions for the Channel to complete, disable the DMA Channel, reconfigure the Queues, and Re-enable the DMA Channel.

Since the DMA operations to perform are specified by the Source and Destination Scatter Gather Elements that are pointed to by the Queues, rather than being in the Queues, the Queues do not contain specific DMA transfer information. The same Queues are then readily re-used for different DMA transactions.

When a DMA transaction completes (software reads the current DMA Completion Status Queue element and reads Status == Complete), software processes the resulting DMA data and recycles the Source/Destination SGL Queue elements associated with the transfer as well as the associated DMA Source/Destination Completion Status Queue elements. Status Queue elements must be written to 0x0 when recycled since software uses a read of non-0 for a Status Queue element as indication that a DMA transfer completed. Recycled Queue elements are re-used when the queue LIMIT pointer wraps back to the position of the recycled elements.

2.8.3. DMA Performance

Table 2.8 to Table 2.11 show the performance of the PCIe Hardened DMA. Performance is measured from the start of DMA operation to until the MSI is received by the host. The values below are based on simulation using multiple descriptors of the same size.

Bigger descriptors can achieve better efficiency as there are less descriptors and data transfer overhead.

For simulation-based performance data harvesting, the non-posted round trip latency is modeled as 1.5 µs.

2.8.3.1. FPGA-to-Host (F2H) Simulation Transfer

Table 2.8. Simulation Data Throughput for FPGA-to-Host (F2H) Transfer

Descriptor Size	Efficiency	Throughput
256 kB	76.35%	12.033 GB/s

2.8.3.2. Host-to-FPGA (H2F) Simulation Transfer

Table 2.9. Simulation Data Throughput for Host-to-FPGA (H2F) Transfer

Descriptor Size	Efficiency	Throughput
256 kB	26.34%	4.152GB/s

2.8.3.3. FPGA-to-Host (F2H) Hardware Transfer

Table 2.10. Hardware Data Throughput for FPGA-to-Host (F2H) Transfer

Descriptor Size	Efficiency	Throughput
128 kB	48.41%	7.627 GB/s

2.8.3.4. Host-to-FPGA (H2F) Simulation Transfer

Table 2.11. Hardware Data Throughput for Host-to-FPGA (H2F) Transfer

Descriptor Size	Efficiency	Throughput
128 kB	32.28%	5.085 GB/s

Note: The throughput is limited because the hardened DMA only utilizes 16 PCIe tag for memory read. This is a hard IP limitation.

2.8.4. DMA Bypass Interface

The DMA Bypass mode allows the received MWr and MRd TLP to be converted to the AXI-MM Interface. DMA Bypass cycles can be routed through any BARs of the PCIe memory, except for BAR 0.

For read access, Read Data Channel from AXI-MM is converted to CplD TLP by the IP and transmitted to the host.

2.9. DMA Support

The Direct Memory Access (DMA) support is an option provided by soft IP to enable a more efficient data transfer when the device acts as initiator

2.9.1. DMA Overview

The Direct Memory Access is an efficient way of transferring data. In this, a DMA engine handles the data transaction process on behalf of the processor. Once the processor (PCIe Root port) forms descriptors in host memory and programs DMA registers through memory write, the DMA engine handles the bus protocol and address sequencing on its own.

After the IP has its registers written with the total number of descriptor and the address of the first descriptor, it fetches the descriptors from host memory through the Memory Read TLP. When Completion(s) is received, the IP starts the transaction based on descriptor data. Once a transaction with Interrupt bit is set in its descriptor is completed, the DMA IP transmits MSI as an interrupt to the host. The IP supports data transfer for both Host-to-FPGA (H2F) and FPGA-to-Host (F2H). Each direction has a dedicated set of registers. Refer to DMA Overview for DMA registers.

2.9.2. DMA Descriptor

The descriptors are packets of data which contain information such as source address, destination address, length of DMA transfer, and other attributes such as the number of contiguous descriptors and interrupt. The descriptor data is stored in the host memory and to be fetched by the IP through Memory Read. The start address of the descriptor queue in the host memory and the total contiguous descriptor are given from *H2F Descriptor Fetching (0x0200)* and *F2H Descriptor Fetching (0x0300)* registers. Based on the start address of descriptor queue, the IP does the bulk fetching from the host memory.

Table 2.12. Descriptor Format

Offset	Fields				
0x00	RSVD[17:0]	CONT_DESC[5:0]	RSVD[5:0]	INT	EOP
0x04	RSVD[7:0]	LENGTH[23:0]			
0x08	NEXT_DESC_ADDR_LO[31:0]				
0x0C	NEXT_DESC_ADDR_HI[31:0]				
0x10	SRC_ADDR_LO[31:0]				
0x14	SRC_ADDR_HI[31:0]				
0x18	DEST_ADDR_LO[31:0]				
0x1C	DEST_ADDR_HI[31:0]				

Table 2.13. DESC_CTRL (0x00)

Field	Name	Width	Description
31:14	RSVD	18	Reserved
13:8	CONT_DESC	6	The number of contiguous Descriptor from the Descriptor address in NEXT_DESC_ADDR_LO and NEXT_DESC_ADDR_HI. All 0s mean 64 contiguous descriptors. This field is only valid when EOP at the last descriptor of a descriptor chunk is 0. In all other cases, this field is ignored.
2:7	RSVD	6	Reserved
1	INT	1	Interrupt trigger. Once the data transfer described by this descriptor is done, the interrupt is triggered by DMA engine to the Host. Interrupt type and vector mapping are configurable in DMA register.
0	EOP	1	Stop fetching the next descriptor. This bit can be 1 only at the last descriptor of a Descriptor Chunk. This field is ignored by the IP if it is not the last descriptor of a Descriptor Chunk.

Table 2.14. DMA_LEN (0x04)

Field	Name	Width	Description
31:24	RSVD	8	Reserved
23:0	LENGTH	24	DMA transfer length in Byte. 23'd1: 1 Byte transfer 23'd2: 2 Byte transfer and so on. All 0 means 8 Mega Byte transfer.

Table 2.15. NEXT_DESC_ADDR_LO (0x08)

Field	Name	Width	Description
31:0	NEXT_DESC_ADDR_LO	32	Lower 32 bit of the next Descriptor Address. This field is only valid when EOP at the last descriptor of a descriptor chunk is 0. In all other cases, this field is ignored.

Table 2.16. NEXT_DESC_ADDR_HI (0x0C)

Field	Name	Width	Description
31:0	NEXT_DESC_ADDR_HI	32	Upper 32 bit of the next Descriptor Address. This field is only valid when EOP at the last descriptor of a descriptor chunk is 0. In all other cases, this field is ignored.

Table 2.17. SRC_ADDR_LO (0x10)

Field	Name	Width	Description
31:0	SRC_ADDR_LO	32	Lower 32 bit of Source Address

Table 2.18. SRC_ADDR_HI (0x14)

Field	Name	Width	Description
31:0	SRC_ADDR_HI	32	Upper 32 bit of Source Address

Table 2.19. DEST_ADDR_LO (0x18)

Field	Name	Width	Description
31:0	DEST_ADDR_LO	32	Lower 32 bit of Destination Address

Table 2.20. DEST_ADDR_HI (0x1C)

Field	Name	Width	Description
31:0	DEST_ADDR_HI	32	Upper 32 bit of Destination Address

2.9.2.1. Descriptor Rules

The following shows the descriptor rules:

- NEXT_DESC_ADDR must be 8DW-aligned (bit[4:0] = 5'b00000) so that descriptors can end at RCB boundary.
- SRC_ADDR[63:0] and DEST_ADDR[63:0] must be 8DW-aligned (bit[4:0] = 5'b00000). There is no address translation for source address and destination address to the address in AXI-MM interface. The driver must have awareness of the exact physical addresses.
- EOP bit is only observed by the IP at the last descriptor of a descriptor chunk.

Note: Fail to comply to the descriptor rules may result in undefined behaviours.

2.9.2.2. Descriptor Example

In this example, the first descriptor chunk (starting address and number of contiguous descriptors are configured DMA register) has three contiguous descriptors.

The second descriptor chunk (starting address and number of contiguous descriptors from the last descriptor of the first chunk) has two contiguous descriptors.

The third descriptor chunk (starting address and number of contiguous descriptors from the last descriptor of the second chunk) has only one descriptor.

Table 2.21. First Descriptor Chunk Fetching through MRd TLP

Offset	Fields				
0x00	RSVD[17:0]	CONT_DESC[5:0] (Don't care)	RSVD[5:0]	INT	EOP (Don't care)
0x04	RSVD[7:0]	LENGTH[23:0]			
0x08	NEXT_DESC_ADDR_LO[31:0] (Don't care)				
0x0C	NEXT_DESC_ADDR_HI[31:0]				

Offset	Fields				
	(Don’t care)				
0x10	SRC_ADDR_LO[31:0]				
0x14	SRC_ADDR_HI[31:0]				
0x18	DEST_ADDR_LO[31:0]				
0x1C	DEST_ADDR_HI[31:0]				
0x20	RSVD[17:0]	CONT_DESC[5:0] (Don’t care)	RSVD[5:0]	INT	EOP (Don’t care)
0x24	RSVD[7:0]	LENGTH[23:0]			
0x28	NEXT_DESC_ADDR_LO[31:0] (Don’t care)				
0x2C	NEXT_DESC_ADDR_HI[31:0] (Don’t care)				
0x30	SRC_ADDR_LO[31:0]				
0x34	SRC_ADDR_HI[31:0]				
0x38	DEST_ADDR_LO[31:0]				
0x3C	DEST_ADDR_HI[31:0]				
0x40	RSVD[17:0]	CONT_DESC[5:0] = 2	RSVD[5:0]	INT	EOP= 0
0x44	RSVD[7:0]	LENGTH[23:0]			
0x48	NEXT_DESC_ADDR_LO[31:0] = ‘hA0				
0x4C	NEXT_DESC_ADDR_HI[31:0] = ‘h0				
0x50	SRC_ADDR_LO[31:0]				
0x54	SRC_ADDR_HI[31:0]				
0x58	DEST_ADDR_LO[31:0]				
0x5C	DEST_ADDR_HI[31:0]				

Table 2.22. Second Descriptor Chunk Fetching through MRd TLP

Offset	Fields				
0xA0	RSVD[17:0]	CONT_DESC[5:0] (Don’t care)	RSVD[5:0]	INT	EOP (Don’t care)
0xA4	RSVD[7:0]	LENGTH[23:0]			
0xA8	NEXT_DESC_ADDR_LO[31:0] (Don’t care)				
0xAC	NEXT_DESC_ADDR_HI[31:0] (Don’t care)				
0xB0	SRC_ADDR_LO[31:0]				
0xB4	SRC_ADDR_HI[31:0]				
0xB8	DEST_ADDR_LO[31:0]				
0xBC	DEST_ADDR_HI[31:0]				
0xC0	RSVD[17:0]	CONT_DESC[5:0] = 1	RSVD[5:0]	INT	EOP= 0
0xC4	RSVD[7:0]	LENGTH[23:0]			
0xC8	NEXT_DESC_ADDR_LO[31:0] = ‘h1B0				
0xCC	NEXT_DESC_ADDR_HI[31:0] = ‘h0				
0xD0	SRC_ADDR_LO[31:0]				
0xD4	SRC_ADDR_HI[31:0]				

Offset	Fields
0xD8	DEST_ADDR_LO[31:0]
0xDC	DEST_ADDR_HI[31:0]

Table 2.23. Third Descriptor Chunk Fetching through MRd TLP:

Offset	Fields				
0x1B0	RSVD[17:0]	CONT_DESC[5:0] (Don't care)	RSVD[5:0]	INT	EOP= 1
0x1B4	RSVD[7:0]	LENGTH[23:0]			
0x1B8	NEXT_DESC_ADDR_LO[31:0] (Don't care)				
0x1BC	NEXT_DESC_ADDR_HI[31:0] (Don't care)				
0x1C0	SRC_ADDR_LO[31:0]				
0x1C4	SRC_ADDR_HI[31:0]				
0x1C8	DEST_ADDR_LO[31:0]				
0x1CC	DEST_ADDR_HI[31:0]				

2.9.3. DMA Registers

PCIe DMA registers are accessible by the Host when received MWr or MRd TLP has BAR 0 hit. Register read size is limited to maximum 1 DW per MRd TLP.

The Access Types of each register are defined in Table 2.24.

Table 2.24. Access Types

Access Type	Behavior on Read Access	Behavior on Write Access
RO	Returns register value	Ignores write access
WO	Returns 0	Updates register value
RW	Returns register value	Updates register value
RW1C	Returns register value	Writing 1'b1 on register bit clears the bit to 1'b0. Writing 1'b0 on register bit is ignored.
RW1S	Returns register value	Writing 1'b1 on register bit sets the bit to 1'b1. Writing 1'b0 on register bit is ignored.
RC	Returns register value Clear the register to 0 after read.	Ignores write access
RSVD	Returns 0	Ignores write access

Table 2.25. PCIe DMA Register Group

Register Base Offset	Register Group Name
0x0000	H2F DMA Control and Status
0x0100	F2H DMA Control and Status
0x0200	H2F Descriptor Fetching
0x0300	F2H Descriptor Fetching
0x0400	Interrupt Control and Status
Others	RSVD

2.9.3.1. H2F DMA Control and Status (0x0000)

Table 2.26. H2F_DMA_CTRL (0x0000)

Field	Name	Access	Width	Default	Description
31:1	RSVD	RO	31	0	Reserved
0	REQUEST	RW1S	1	0	Request to start DMA operation. Once this bit is 1, writing a 0 to clear it does not take effect. Once the field in “H2F Descriptor Fetching” is consumed by DMA Engine to trigger descriptor fetching, this bit is cleared to 0 by HW.

Table 2.27. H2F_DMA_STS (0x000C)

Field	Name	Access	Width	Default	Description
31:14	RSVD	RO	18	0	Reserved
13	RSVD	RO	1	0	Reserved
12	RSVD	RO	1	0	Reserved
11	DMA_LEN_ERR	RC	1	0	DMA Length Error 1: DMA Length is not DW-aligned. 0: No error
10	H2F_DESTADDR_ERR	RC	1	0	H2F Destination Address Error 1: H2F Destination Address is not 8DW-aligned. 0: No error
9	H2F_SRCADDR_ERR	RC	1	0	H2F Source Address Error 1: H2F Source Address is not 8DW-aligned. 0: No error
8	DESC_ADDR_ERR	RC	1	0	Descriptor Address Error 1: Desc Address is not 8DW-aligned. 0: No error
7	AXI_WRITE_ERR	RC	1	0	AXI Write Error 1: AXI Write Response is not OKAY (2’b00). 0: No error
6	H2F_CPLTO_ERR	RC	1	0	H2F Completion Timeout Error 1: Completion timeout at H2F DMA transfer. 0: No error
5	H2F_CPL_ERR	RC	1	0	H2F Completion Error 1: Completion Status is not Successful Completion. 0: No error
4	DESC_CPLTO_ERR	RC	1	0	Descriptor Completion Timeout Error 1: Completion timeout at H2F Descriptor fetching. 0: No error
3	DESC_CPL_ERR	RC	1	0	Descriptor Completion Error 1: Completion Status is not Successful Completion. 0: No error
2	DMA_INT_DONE	RC	1	0	DMA Interrupt Done 1: DMA transfer is done for descriptor with INT bit = 1. 0: No DMA Interrupt Done event.
1	DMA_EOP_DONE	RC	1	0	DMA EOP Done 1: DMA transfer is done for descriptor with EOP bit = 1. 0: No DMA EOP Done event.
0	BUSY	RO	1	0	DMA Engine busy 1: DMA is busy. 0: DMA in IDLE state, no operation pending.

Table 2.28. H2F_DMA_INT_MASK (0x0010)

Field	Name	Access	Width	Default	Description
31:12	RSVD	RO	20	0	Reserved
11	DMA_LEN_ERR_INTMASK	RW	1	0	DMA Length Error Interrupt Masking 1: Mask off interrupt generation caused by DMA_LEN_ERR. 0: No interrupt masking for DMA_LEN_ERR.
10	H2F_DESTADDR_ERR_INTMASK	RW	1	0	H2F Destination Address Error Interrupt Masking 1: Mask off interrupt generation caused by H2F_DESTADDR_ERR. 0: No interrupt masking for H2F_DESTADDR_ERR.
9	H2F_SRCADDR_ERR_INTMASK	RW	1	0	H2F Source Address Error Interrupt Masking 1: Mask off interrupt generation caused by H2F_SRCADDR_ERR. 0: No interrupt masking for H2F_SRCADDR_ERR.
8	DESC_ADDR_ERR_INTMASK	RW	1	0	Descriptor Address Error Interrupt Masking 1: Mask off interrupt generation caused by DESC_ADDR_ERR. 0: No interrupt masking for DESC_ADDR_ERR.
7	AXI_WRITE_ERR_INTMASK	RW	1	0	AXI Write Error Interrupt Masking 1: Mask off interrupt generation caused by AXI_WRITE_ERR. 0: No interrupt masking for AXI_WRITE_ERR.
6	H2F_CPLTO_ERR_INTMASK	RW	1	0	H2F Completion Timeout Error Interrupt Masking 1: Mask off interrupt generation caused by H2F_CPLTO_ERR. 0: No interrupt masking for H2F_CPLTO_ERR.
5	H2F_CPL_ERR_INTMASK	RW	1	0	H2F Completion Error Interrupt Masking 1: Mask off interrupt generation caused by H2F_CPL_ERR. 0: No interrupt masking for H2F_CPL_ERR.
4	DESC_CPLTO_ERR_INTMASK	RW	1	0	Descriptor Completion Timeout Error Interrupt Masking 1: Mask off interrupt generation caused by DESC_CPLTO_ERR. 0: No interrupt masking for DESC_CPLTO_ERR.
3	DESC_CPL_ERR_INTMASK	RW	1	0	Descriptor Completion Error Interrupt Masking 1: Mask off interrupt generation caused by DESC_CPL_ERR. 0: No interrupt masking for DESC_CPL_ERR.
2	DMA_INT_DONE_INTMASK	RW	1	0	DMA Interrupt Done Interrupt Masking 1: Mask off interrupt generation caused by DMA_INT_DONE. 0: No interrupt masking for DMA_INT_DONE.

Field	Name	Access	Width	Default	Description
1	DMA_EOP_DONE_INTMASK	RW	1	0	DMA EOP Done Interrupt Masking 1: Mask off interrupt generation caused by DMA_EOP_DONE. 0: No interrupt masking for DMA_EOP_DONE.
0	RSVD	RO	20	0	Reserved

Table 2.29. H2F_CPLT_DESC_COUNT (0x0018)

Field	Name	Access	Width	Default	Description
31:0	CPLT_DESC_CNT	RO	32	0	The number of competed descriptors since the last rising edge of REQUEST bit. Reset to 0 at the subsequent rising edge of REQUEST bit.

2.9.3.2. F2H DMA Control and Status (0x0100)

Table 2.30. F2H_DMA_CTRL (0x0100)

Field	Name	Access	Width	Default	Description
31:1	RSVD	RO	31	0	Reserved
0	REQUEST	RW1S	1	0	Request to start DMA operation. 1: Request to start DMA operation Once the field in “F2H Descriptor Fetching” is consumed by DMA Engine to trigger descriptor fetching, this bit is cleared to 0 by HW.

Table 2.31. F2H_DMA_STS (0x010C)

Field	Name	Access	Width	Default	Description
31:14	RSVD	RO	18	0	Reserved
13	AXIST_DATA_L_ERR	RC	1	0	AXI-Stream Data Long Error 1: The received AXI-Stream data is longer than Descriptor length. 0: No error
12	AXIST_DATA_S_ERR	RC	1	0	AXI-Stream Data Short Error 1: The received AXI-Stream data is shorter than Descriptor length. 0: No error
11	DMA_LEN_ERR	RC	1	0	DMA Length Error 1: DMA Length is not DW-aligned. 0: No error
10	F2H_DESTADDR_ERR	RC	1	0	F2H Destination Address Error 1: F2H Destination Address is not 8DW-aligned. 0: No error
9	F2H_SRCADDR_ERR	RC	1	0	F2H Source Address Error 1: F2H Source Address is not 8DW-aligned. 0: No error
8	DESC_ADDR_ERR	RC	1	0	Descriptor Address Error 1: Desc Address is not 8DW-aligned. 0: No error
7	AXI_READ_ERR	RC	1	0	AXI Read Error 1: AXI Read Response is not OKAY (2’b00). 0: No error
6:5	RSVD	RO	2	0	Reserved

Field	Name	Access	Width	Default	Description
4	DESC_CPLTO_ERR	RC	1	0	Descriptor Completion Timeout Error 1: Completion timeout at F2H Descriptor fetching. 0: No error
3	DESC_CPL_ERR	RC	1	0	Descriptor Completion Error 1: Completion Status is not Successful Completion. 0: No error
2	DMA_INT_DONE	RC	1	0	DMA Interrupt Done 1: DMA transfer is done for descriptor with INT bit = 1. 0: No DMA Interrupt Done event.
1	DMA_EOP_DONE	RC	1	0	DMA EOP Done 1: DMA transfer is done for descriptor with EOP bit = 1. 0: No DMA EOP Done event.
0	BUSY	RO	1	0	DMA Engine busy 1: DMA is busy. 0: DMA in IDLE state, no operation pending.

Table 2.32. F2H_DMA_INT_MASK (0x0110)

Field	Name	Access	Width	Default	Description
31:14	RSVD	RO	19	0	Reserved
13	AXIST_DATA_L_ERR_INTMASK	RW	1	0	AXI-Stream Data Long Error Interrupt Masking 1: Mask off interrupt generation caused by AXIST_DATA_L_ERR. 0: No interrupt masking for AXIST_DATA_L_ERR.
12	AXIST_DATA_S_ERR_INTMASK	RW	1	0	AXI-Stream Data Short Error Interrupt Masking 1: Mask off interrupt generation caused by AXIST_DATA_S_ERR. 0: No interrupt masking for AXIST_DATA_S_ERR.
11	DMA_LEN_ERR_INTMASK	RW	1	0	DMA Length Error Interrupt Masking 1: Mask off interrupt generation caused by DMA_LEN_ERR. 0: No interrupt masking for DMA_LEN_ERR.
10	F2H_DESTADDR_ERR_INTMASK	RW	1	0	F2H Destination Address Error Interrupt Masking 1: Mask off interrupt generation caused by F2H_DESTADDR_ERR. 0: No interrupt masking for F2H_DESTADDR_ERR.
9	F2H_SRCADDR_ERR_INTMASK	RW	1	0	F2H Source Address Error Interrupt Masking 1: Mask off interrupt generation caused by F2H_SRCADDR_ERR. 0: No interrupt masking for F2H_SRCADDR_ERR.
8	DESC_ADDR_ERR_INTMASK	RW	1	0	Descriptor Address Error Interrupt Masking 1: Mask off interrupt generation caused by DESC_ADDR_ERR. 0: No interrupt masking for DESC_ADDR_ERR.
7	AXI_READ_ERR_INTMASK	RW	1	0	AXI Read Error Interrupt Masking 1: Mask off interrupt generation caused by AXI_READ_ERR. 0: No interrupt masking for AXI_READ_ERR.

Field	Name	Access	Width	Default	Description
6:5	RSVD	RO	2	0	Reserved
4	DESC_CPLTO_ERR_INTMASK	RW	1	0	Descriptor Completion Timeout Error Interrupt Masking 1: Mask off interrupt generation caused by DESC_CPLTO_ERR. 0: No interrupt masking for DESC_CPLTO_ERR.
3	DESC_CPL_ERR_INTMASK	RW	1	0	Descriptor Completion Error Interrupt Masking 1: Mask off interrupt generation caused by DESC_CPL_ERR. 0: No interrupt masking for DESC_CPL_ERR.
2	DMA_INT_DONE_INTMASK	RW	1	0	DMA Interrupt Done Interrupt Masking 1: Mask off interrupt generation caused by DMA_INT_DONE. 0: No interrupt masking for DMA_INT_DONE.
1	DMA_EOP_DONE_INTMASK	RW	1	0	DMA EOP Done Interrupt Masking 1: Mask off interrupt generation caused by DMA_EOP_DONE. 0: No interrupt masking for DMA_EOP_DONE.
0	RSVD	RO	20	0	Reserved

Table 2.33. F2H_CPLT_DESC_COUNT (0x0118)

Field	Name	Access	Width	Default	Description
31:0	CPLT_DESC_CNT	RO	32	0	The number of competed descriptors since the last rising edge of REQUEST bit. Reset to 0 at the subsequent rising edge of REQUEST bit.

2.9.3.3. H2F Descriptor Fetching (0x0200)

Table 2.34. H2F_DESC_ADDR_LOW (0x0200)

Field	Name	Access	Width	Default	Description
31:0	DESC_ADDR_LOW	RW	32	0	Lower 32-bit address of descriptor

Table 2.35. H2F_DESC_ADDR_HIGH (0x0204)

Field	Name	Access	Width	Default	Description
31:0	DESC_ADDR_HIGH	RW	32	0	Upper 32-bit address of descriptor

Table 2.36. H2F_CONT_REMAIN (0x0208)

Field	Name	Access	Width	Default	Description
31:6	RSVD	RO	26	0	Reserved
5:0	CONT_DESC	RW	6	0	The number of contiguous Descriptor from the Descriptor address in 0x00 and 0x04. All 0s mean 64 contiguous descriptors.

2.9.3.4. F2H Descriptor Fetching (0x0300)

Table 2.37. F2H_DESC_ADDR_LOW (0x0300)

Field	Name	Access	Width	Default	Description
31:0	DESC_ADDR_LOW	RW	32	0	Lower 32-bit address of descriptor

Table 2.38. F2H_DESC_ADDR_HIGH (0x0304)

Field	Name	Access	Width	Default	Description
31:0	DESC_ADDR_HIGH	RW	32	0	Upper 32-bit address of descriptor

Table 2.39. F2H_CONT_REMAIN (0x0308)

Field	Name	Access	Width	Default	Description
31:6	RSVD	RO	26	0	Reserved
5:0	CONT_DESC	RW	6	0	The number of contiguous Descriptor from the Descriptor address in 0x00 and 0x04. All 0s mean 64 contiguous descriptors.

2.9.3.5. Interrupt Control and Status (0x0400)

Table 2.40. INT_MODE (0x0400)

Field	Name	Access	Width	Default	Description
31:2	RSVD	RO	30	0	Reserved
1:0	INT_MODE_ENABLE	RO	2	0	Interrupt Mode Enable. 2'b00: Wire interrupt 2'b01: INTx 2'b10: MSI 2'b11: MSI-X Others: Reserved In the current release, only MSI is supported.

Table 2.41. H2F_MSI_VEC (0x0404)

Field	Name	Access	Width	Default	Description
31:5	RSVD	RO	27	0	Reserved
4:0	CHAN0_MSI_INT_VEC	RW	5	0	Channel 0 H2F MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.

Table 2.42. F2H_MSI_VEC (0x0408)

Field	Name	Access	Width	Default	Description
31:5	RSVD	RO	27	0	Reserved
4:0	CHAN0_F2H_MSI_VEC	RW	5	0	Channel 0 F2H MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.

The number of enabled user interrupt signals reflects the number of USR_MSI_VEC bits available for configuration in the USR_MSI_VEC_P1 to USR_MSI_VEC_P4 registers

Table 2.43. USR_MSI_VEC_P1 (0x040C)

Field	Name	Access	Width	Default	Description
31:29	RSVD	RO	3	0	Reserved
28:24	USR3_MSI_VEC	RW	5	0	User 3 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.
23:21	RSVD	RO	3	0	Reserved
20:16	USR2_MSI_VEC	RW	5	0	User 2 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.
15:13	RSVD	RO	3	0	Reserved
12:8	USR1_MSI_VEC	RW	5	0	User 1 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.
7:5	RSVD	RO	3	0	Reserved
4:0	USR0_MSI_VEC	RW	5	0	User 0 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.

Table 2.44. USR_MSI_VEC_P2 (0x0410)

Field	Name	Access	Width	Default	Description
31:29	RSVD	RO	3	0	Reserved
28:24	USR7_MSI_VEC	RW	5	0	User 7 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.
23:21	RSVD	RO	3	0	Reserved
20:16	USR6_MSI_VEC	RW	5	0	User 6 MSI Vector

Field	Name	Access	Width	Default	Description
					When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.
15:13	RSVD	RO	3	0	Reserved
12:8	USR5_MSI_VEC	RW	5	0	User 5 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.
7:5	RSVD	RO	3	0	Reserved
4:0	USR4_MSI_VEC	RW	5	0	User 4 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.

Table 2.45. USR_MSI_VEC_P3 (0x0414)

Field	Name	Access	Width	Default	Description
31:29	RSVD	RO	3	0	Reserved
28:24	USR11_MSI_VEC	RW	5	0	User 11 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.
23:21	RSVD	RO	3	0	Reserved
20:16	USR10_MSI_VEC	RW	5	0	User 10 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.
15:13	RSVD	RO	3	0	Reserved
12:8	USR9_MSI_VEC	RW	5	0	User 9 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.
7:5	RSVD	RO	3	0	Reserved
4:0	USR8_MSI_VEC	RW	5	0	User 8 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.

Table 2.46. USR_MSI_VEC_P4 (0x0418)

Field	Name	Access	Width	Default	Description
31:29	RSVD	RO	3	0	Reserved
28:24	USR15_MSI_VEC	RW	5	0	User 15 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.
23:21	RSVD	RO	3	0	Reserved
20:16	USR4_MSI_VEC	RW	5	0	User 14 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.
15:13	RSVD	RO	3	0	Reserved
12:8	USR13_MSI_VEC	RW	5	0	User 13 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.
7:5	RSVD	RO	3	0	Reserved
4:0	USR12_MSI_VEC	RW	5	0	User 12 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.

2.9.3.6. General Status (0x0500)

Table 2.47. GENERAL_STS (0x0500)

Field	Name	Access	Width	Default	Description
31:6	RSVD	RO	26	0	Reserved
5:3	DMA_SUPPORT	RO	3	0	DMA Support 3'b000: Support F2H and H2F 3'b001: Support F2H Only 3'b010: Support H2F Only 3'b011: Do not support F2H and H2F Others: Reserved
2:0	DMA_TYPE	RO	3	0	DMA Type 3'b000: AXI-MM DMA 3'b001: AXI-ST DMA Others: Reserved

2.9.4. DMA Transaction (AXI-MM)

The data transfer with the DMA support is illustrated in the following figures. Additional registers required by DMA are implemented as well as status registers and interrupt signals, which are discussed in the subsections below.

2.9.4.1. FPGA-to-Host (F2H) Transaction

In F2H transaction, the core reads the data from memory through AXI-MM Address Read and Read Data Channels and one or more Memory Write TLPs are generated and transmitted to the host through PCIe link until the transfer is completed.

Figure 2.11 shows the block diagram that shows an overall F2H Data Transfer.

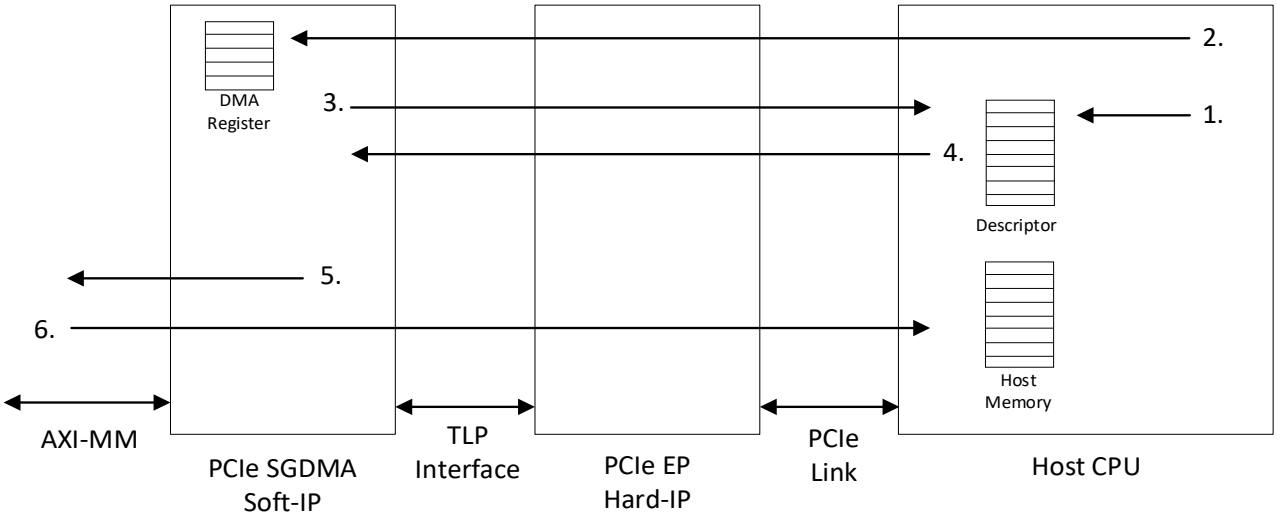


Figure 2.11. F2H Data Transfer

The numbers below are the sequence of F2H flow which corresponds to the numbers in Figure 2.11.

1. Driver forms Descriptors in the Host Memory. The format of Descriptor is available in the DMA Descriptor section.
2. Driver programs DMA registers through Memory Write TLP. It programs F2H Descriptor Fetching (0x0300) field followed by the Request bit in F2H_DMA_CTRL (0x0100) to kick start F2H data transfer.
3. When the Request bit is set, DMA Engine forms Memory Read TLP and transmits it to the Host targeting the address in F2H_DMA_CTRL registers for Descriptor fetching. Block Descriptor fetching can happen if CONT_DESC register is not 0. If CONT_DESC register shows the contiguous Descriptor is beyond MRRS or crossing 4 kB boundary, the descriptor fetching is split into multiple Memory Read TLPs. Host returns Descriptor to the DMA Engine through Completion with Data TLP. When the last received Descriptor has EOP bit = 0, Descriptor fetching through Memory Read moves on to the next Descriptor address.
4. DMA Engine decodes the received Completion with Data TLP to obtain Source Address, Destination Address, and Length information. It triggers memory read through AXI-MM Read Channel to the Source Address. The size of the read data does not cross the 4K boundary. Therefore, the memory read may be split to several AXI-MM Read.
5. FPGA application returns the data corresponding to the AXI-MM Read through AXI-MM Read Data Channel. Upon receiving the Read data, DMA Engine forms Memory Write TLP and transmits it to the Host targeting Destination Address. The IP guarantees the transmitted Memory Write TLP does not cross 4K boundary.

2.9.4.2. Host-to-FPGA (H2F) Transaction

In H2F transaction, the core transmits Memory Read TLP to the host. Incoming completions are matched with the read request entries and transferred to the specified destination through AXI4-MM Address Write and Write Data channels.

Figure 2.12 shows an overall H2F Data Transfer.

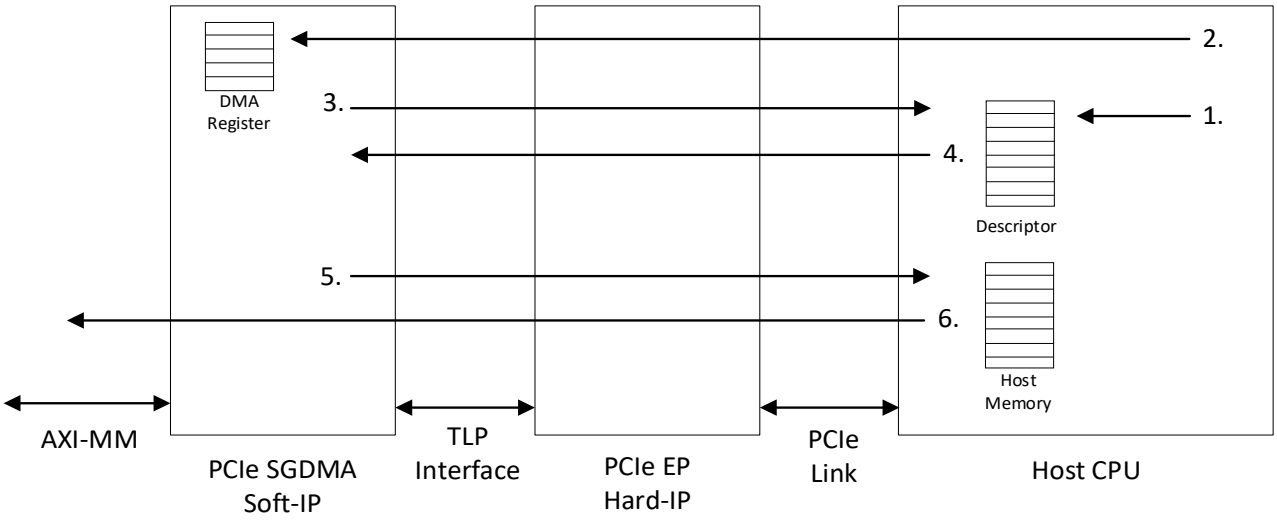


Figure 2.12. H2F Data Transfer

The numbers below are the sequence of H2F flow which is corresponding to the numbers in Figure 2.12.

1. Driver forms Descriptors in the Host Memory. The format of Descriptor is available in the DMA Descriptor section.
2. Driver programs DMA registers through Memory Write TLP. It programs H2F Descriptor Fetching (0x020) field followed by the Request bit in H2F_DMA_CTRL (0x0000) to kick start H2F data transfer.
3. When the Request bit is set, DMA Engine forms Memory Read TLP and transmits it to the Host targeting the address in H2F_DMA_CTRL registers for Descriptor fetching. Block Descriptor fetching can happen if CONT_DESC register is not 0. If CONT_DESC register shows contiguous Descriptor is beyond MRRS or crossing 4 kB boundary, the descriptor fetching is split into multiple Memory Read TLPs. Host returns Descriptor to the DMA Engine through Completion with Data TLP.. When the last received Descriptor has EOP bit = 0, the descriptor fetching through Memory Read moves on to the next Descriptor address.
4. DMA Engine decodes the received Completion with Data TLP to obtain Source Address, Destination Address, and Length information. It generates Memory Read TLP and transmits it to the Host targeting the Source Address. If the Length exceeds MRRS or crossing 4 kB boundary, the memory read will be split into several Memory Read TLPs.
5. Host returns read data as Completion with Data TLP. It may split the read data into several Completion with Data TLPs depending on MPS and RCB. Upon receiving the TLPs, the DMA Engine writes it to the Destination Address through AXI-MM's Write Address Channel and Write Data Channel DMA Interrupt. The IP supports wire interrupts, INTx, MSI, and MSI-X. The selection between the interrupt type is controlled by Radiant user interface. In this release, only MSI is supported.

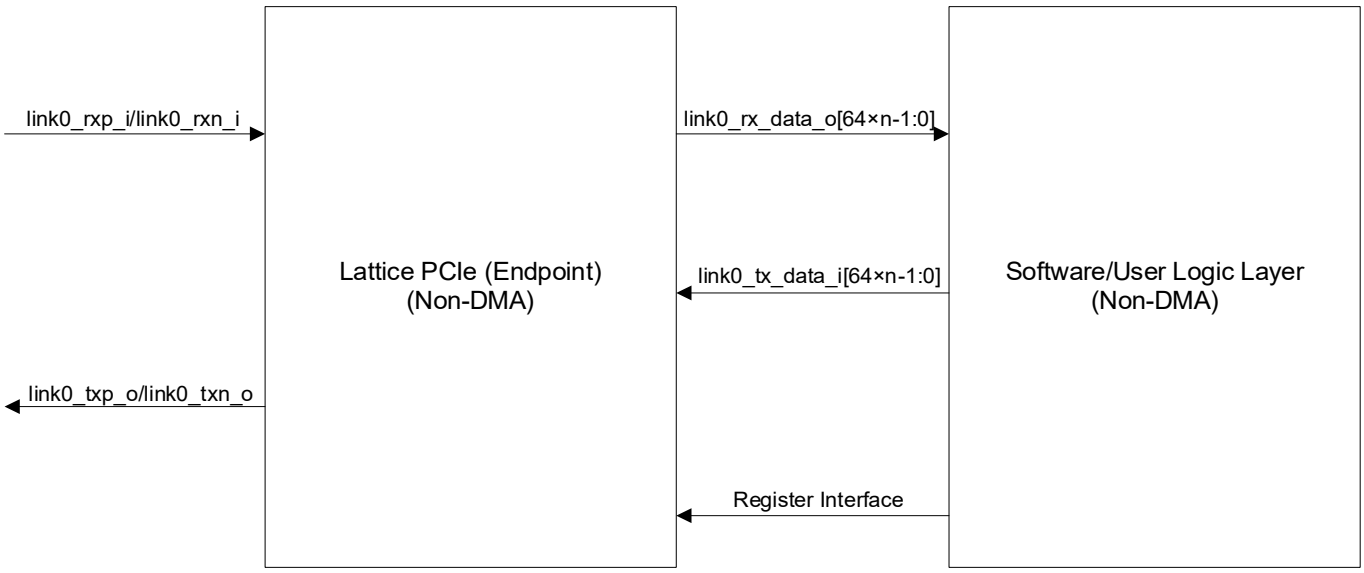
Interrupt can be triggered when a DMA data transfer is completed, or an error occurs.

For DMA data transfer, interrupt is triggered when the last byte of data is transferred corresponding to a descriptor chunk (EOP = 1) or any descriptor with INT bit set to 1 (refer to DMA Descriptor). Interrupt can also be triggered by erroneous cases (refer to DMA Registers).

Note: The IP requires WLAST-to-BVALID latency to be within 40 clock cycles to prevent unexpected behaviour in terms of Memory Read TLPs' Tag number.

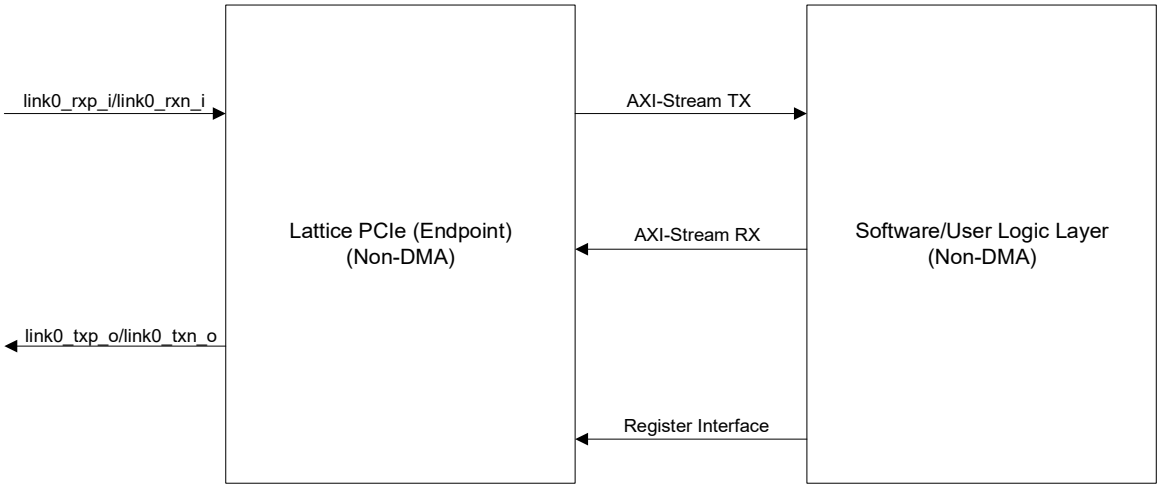
2.10. Non-DMA Support

2.10.1. Non-DMA Overview



Note: n = number of lanes

Figure 2.13. Non-DMA Application Data Flow – TLP Interface



Note: n = number of lanes

Figure 2.14. Non-DMA Application Data Flow – AXI-Stream Interface

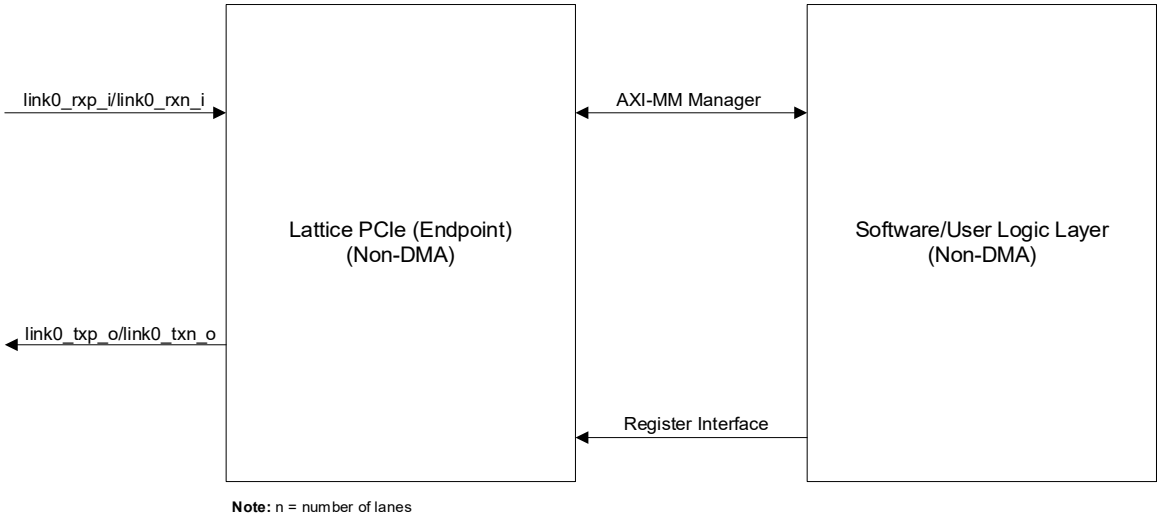


Figure 2.15. Non-DMA Application Data Flow – AXI-MM Interface (Bridge Mode)

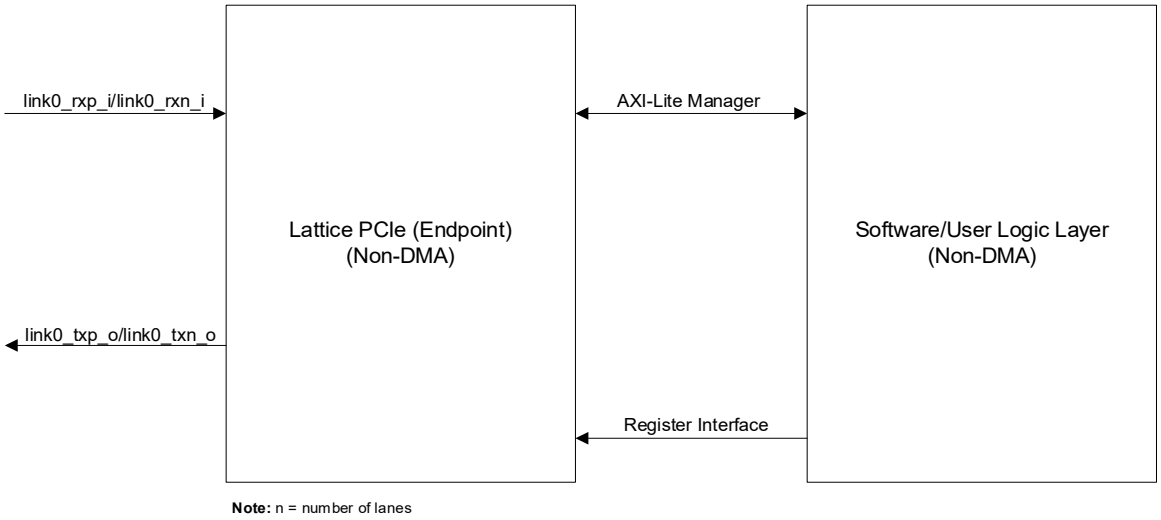


Figure 2.16. Non-DMA Application Data Flow – AXI-Lite Interface (Bridge Mode)

For the non-DMA design, the PCIe EP receives the data through the *link0_rxp_i/link0_rxn_i* serial lines from the Root Complex. The PCIe EP converts the serial data in the form of TLP packets. The TLP packets are sent to the non-DMA application layer through the *link0_rx_data_o* signal. The TLP header info is decoded and the operation is decided whether the data is written or read. For the write operation, the data is written to the RAM present in the application layer. For the read operation, the data is read from the RAM and sends the encoded data to the PCIe EP in the form of TLP packets through the *lnk0_tx_data_i* signal.

The register interface is enhanced as per the data interface selected in the user interface.

Table 2.48. Register Access for Different Data Interfaces

Data Interface	Register Interface
TLP	LMMI
AXI_Stream	AXI-Lite
AXI-MM	LMMI/AXI-Lite
AXI-Lite	LMMI/AXI-Lite

2.10.2. Non-DMA Write

The PCIe EP sends the header data to the non-DMA application layer through the *link0_rx_data_o* signal. The application layer initially verifies the operation by decoding the header information. Once the write operation is detected, the user data is received along with the valid signal from the PCIe IP. The valid data is stored in the RAM present in the application layer.

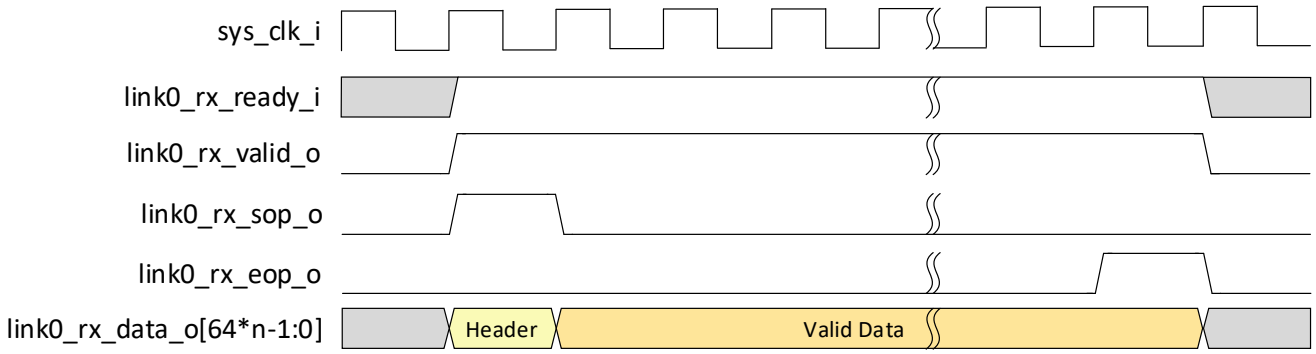


Figure 2.17. Non-DMA Write Operation (TLP Data Interface)

2.10.3. Non-DMA Read

The PCIe EP sends the header data to Non-DMA Application layer through the *link0_rx_data_o* signal. The application layer initially verifies the operation by decoding the header information. Once the operation is detected as read, the application layer waits for the ready signal sent by the PCIe EP. Based on the ready signal and header address, the user data along with the valid signal is send to PCIe EP by the RAM present in the application layer through the *link0_tx_data_i* signal.

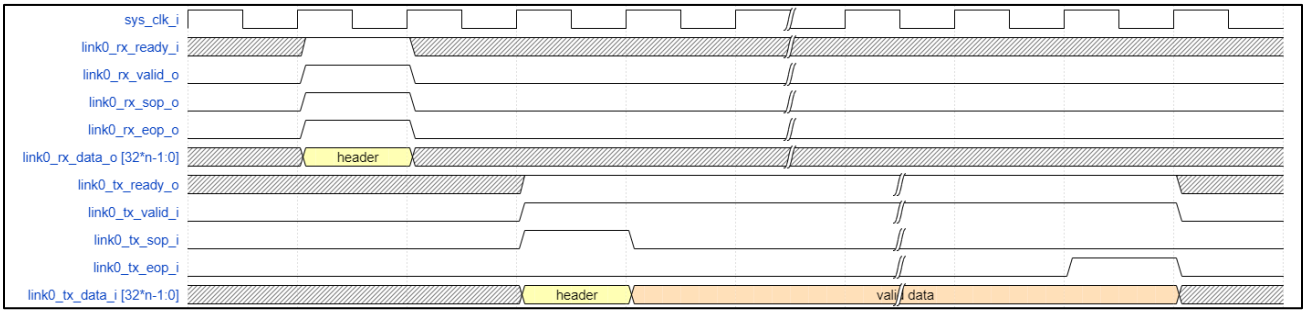


Figure 2.18. Non-DMA Read Operation (TLP Data Interface)

For more details on the TLP write and read data transaction, refer to the TLP TX/RX Interface section.

2.11. Interrupts

2.11.1. Generation of the Interrupts

The Lattice PCIe Core IP supports the Legacy Interrupts, Messaged Signaled Interrupts (MSI), and MSI-X interrupts.

For each function in the PCIe IP core, the system software configures the function to use MSI-X, MSI, or Legacy Interrupt mode as part of the PCI enumeration process.

The Legacy Interrupt is supported by the PCIe Core to support the backward compatibility by enabling the INTx pins.

To minimize the pin count, the function can generate the inband interrupt message packet to indicate the assertion and de-assertion of an interrupt pin. These are the MSI and MSI-X interrupts. This interrupt mechanism is used to conserve the pins because it does not use separate wires for interrupts.

In this mechanism a single Dword provides the information about the interrupt messages MSI-X/MSI interrupts are signaled using MSI-X/MSI Message TLPs, which you can generate and transmit in the Transmit Interface.

The MSI Interrupt is a posted memory write, which is distinguished from the other memory writes by the addresses they target, which are typically reserved by the system for interrupt delivery. The MSI Capability structure is stored in the Configuration Space and is programmed using Configuration Space accesses.

The MSI-X interrupt is the extended version for the MSI interrupts, supporting a greater number of MSI Vectors and the MSI-X capability structure points to an MSI-X table structure and an MSI-X Pending Bit Array (PBA) structure, which are stored in memory.

Enabling and Disabling of interrupts can be done through PCIe IP Core user interface or through Hard IP core configuration status registers.

Table 2.49 describes the register bits to enable and disable each of the interrupts.

Table 2.49. Base Address to Enable Interrupt

Base Address	Offset Address	Register Bits	Description
0x4_4000 (Function 0) 0x4_5000 (Function 1) 0x4_6000 (Function 2) 0x4_7000 (Function 3) 0x4_8000 (Function 4) 0x4_9000 (Function 5) 0x4_A000 (Function 6) 0x4_B000 (Function 7)	0x50	[0]	Support for Legacy Interrupts • 0 – Enable • 1 – Disable
	0xE8	[0]	Support for MSI Interrupts • 0 – Enable • 1 – Disable
	0xF0	[0]	Support for MSI-X Interrupts • 0 – Enable • 1 – Disable

2.11.2. Legacy Interrupt

When the legacy interrupts are enabled, the PCIe IP core emulates the INTx Interrupts using virtual wire. The term INTx refers to the four legacy interrupts: INTA, INTB, INTC, and INTD.

The legacy interrupts (INTA, INTB, INTC, and INTD) can differ across functions.

The selection among the four interrupts can be done through the PCIe IP core user interface or through register interface as described in Table 2.50.

Table 2.50. Legacy Interrupt Register

Base Address	Offset Address	Register Bits	Description
0x4_4000 (Function 0) 0x4_5000 (Function 1) 0x4_6000 (Function 2) 0x4_7000 (Function 3) 0x4_8000 (Function 4) 0x4_9000 (Function 5) 0x4_A000 (Function 6) 0x4_B000 (Function 7)	0x50	[9:8]	Selects which Legacy Interrupt to be used: • 0 – INTA • 1 – INTB • 2 – INTC • 3 – INTD

You must form Legacy Interrupt TLP Message and drive to the PCIe Core through the TX TLP interface.

2.11.3. MSI Interrupt

The Lattice PCIe IP core supports 32 MSI interrupts with a feature of enabling and disabling the vector masking. The MSI request can be either 32-bit addressable Memory Write TLP or a 64-bit addressable Memory Write TLP. There are other two registers called as Mask Bits Register and Pending Bits Register. Since there is a support for 32 interrupts, the mask bit and pending register are 32-bit length, each bit represents the masking or pending status for each interrupt. The MSI-X capability structure values are programmed through the PCI Express configuration space register.

The address is taken from the Message Address and Message Upper Address fields of the MSI Capability Structure, while the payload is taken from the Message Data field.

The type of MSI TLP sent (32-bit addressable or 64-bit addressable) depends on the value of the Upper Address field in the MSI capability structure. By default, the MSI messages are sent as 32-bit addressable Memory Write TLPs. MSI messages use 64-bit addressable Memory Write TLPs only if the system software programs a non-zero value into the Upper Address register.

The message control register in the MSI capability Structure, disables and enables the various support in the MSI Interrupt.

Figure 2.19 and Figure 2.20 show the MSI Capability Structure variant.

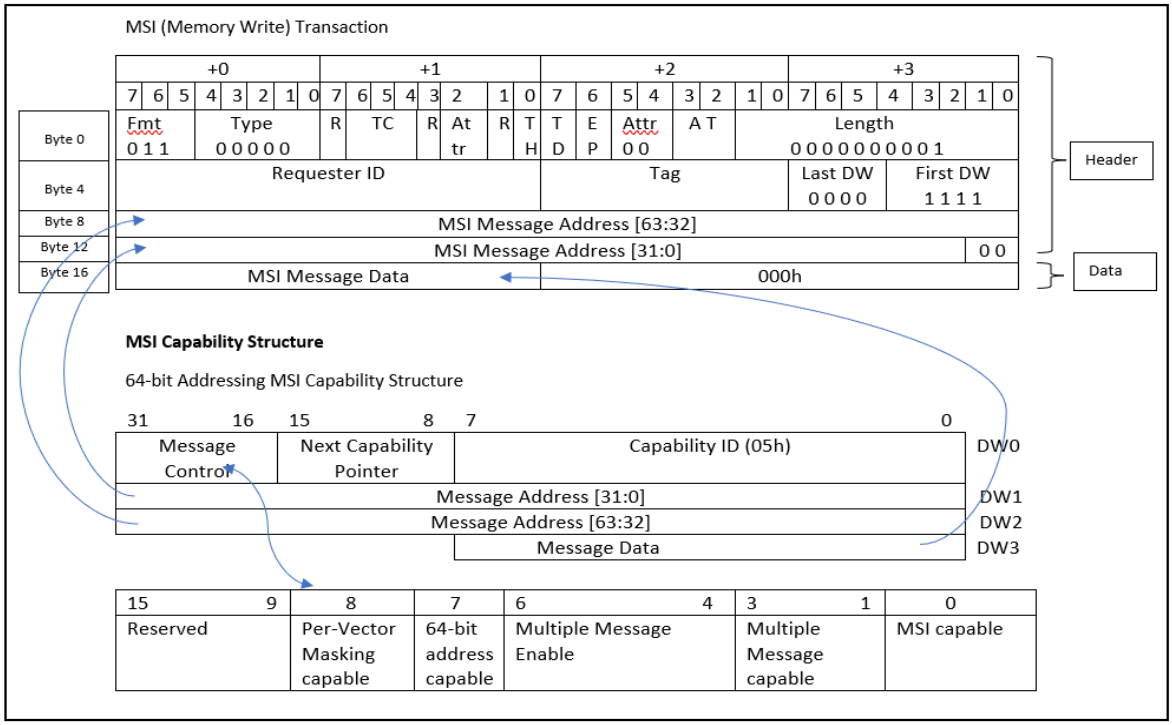


Figure 2.19. MSI Capability Structure Variant

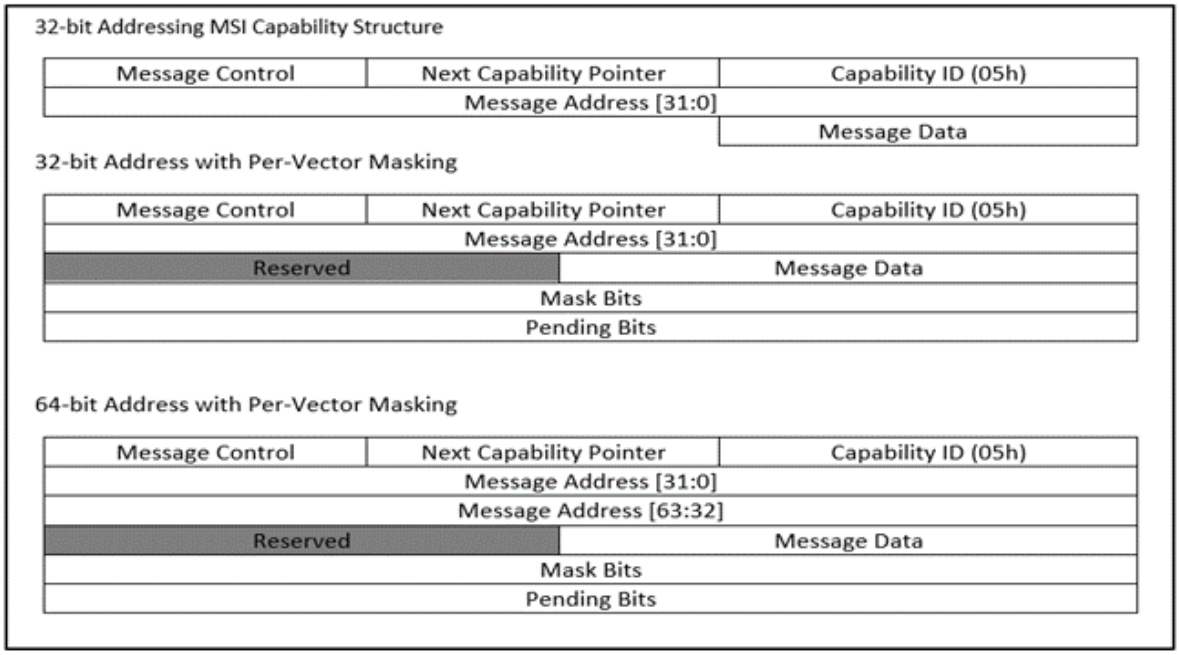


Figure 2.20. MSI Capability Structure Variant

2.11.3.1. MSI Pending Register

The MSI Pending register is used to report MSI Interrupts that are appended in the user design. MSI Pending is a PCIe Configuration Register in the MSI Capability Structure that software uses to obtain status on pending MSI Interrupt vectors.

The MSI Pending register must be written whenever a MSI Interrupt Vector's pending status changes. A 1 must be written to the associated interrupt vector bit when an interrupt becomes pending and a 0 must be written to indicate that the interrupt is no longer pending.

The MSI Pending register must be updated whenever the status of your pending MSI interrupts changes. If MSI interrupts are not used, writing to the MSI Pending Register is not needed.

2.11.4. MSI-X Interrupt

2.11.4.1. MSI-X Capability Structure variant

MSI-X allows the support of large number of vectors with independent message data and address for each vector compared to the MSI Interrupts. It can support up-to 2048 vectors per function. The MSI-X Capability Structure points to an MSI-X table structure and an MSI-X Pending Bit Array (PBA) structure, which are stored in memory. In MSI-X interrupt the vector information is present in the memory location pointed by the Table Base address Indicator Register (BIR).

Figure 2.21 shows the MSI-X capability structure. The MSI-X interrupt configuration is done by the PCIe Configuration Space Registers.

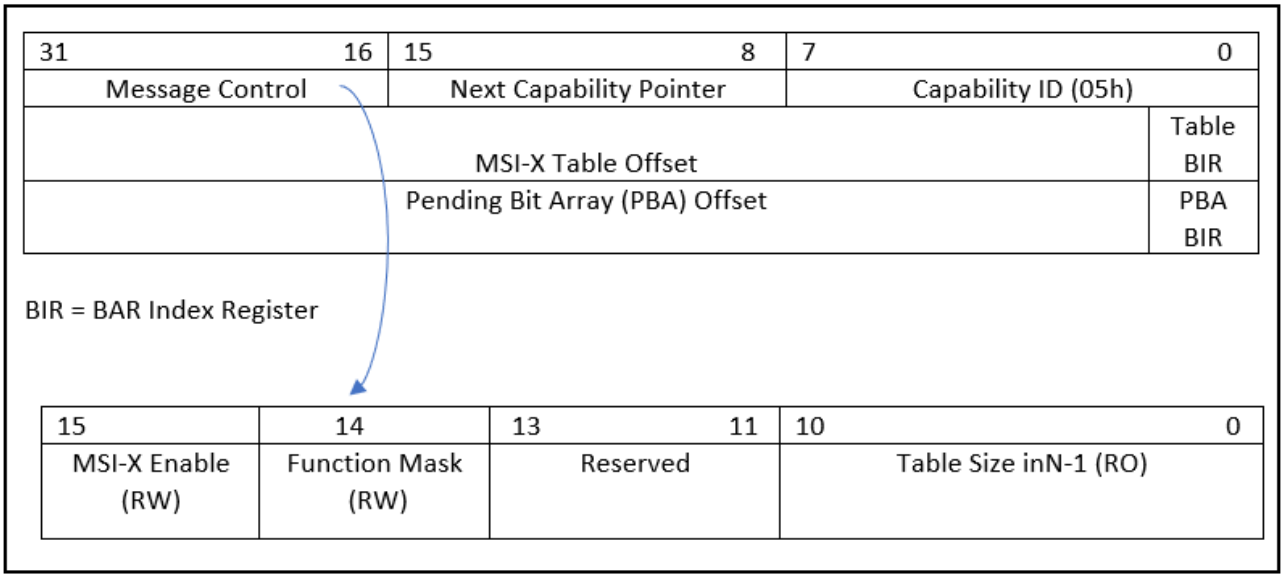


Figure 2.21. MSI-X Capability Structure Variant

The description of each bit in the Message controlled are explained in the section of the PCIe Configuration Space Register configuration for MSI-X Capability Structure. The MSI-X Capability Structure variant contains information about the MSI-X Table and the PBA structure, information such as the pointers to the bases of the MSI-X Table and the PBA structure. The Table BIR in the MSI-X Capability Structure includes information about the BAR location that contains the MSI-X table.

2.11.4.2. MSI-X Table

The MSI-X table is an array of vectors and addresses. The MSI-X Table contains four Dwords. Each entry in the MSI-X table represents one vector. The DW0 and DW1 supply a unique 64-bit address for that vector and DW2 is the 32-bit data pattern for it. The DW3 is the mask bit for the vector and contains only 1 bit at present.

DW3	DW2	DW1	DW0	
Vector Control	Message Data	Upper Address	Lower Address	Entry 0
Vector Control	Message Data	Upper Address	Lower Address	Entry 1
Vector Control	Message Data	Upper Address	Lower Address	Entry 2
….	….	….	….	
….	….	….	….	
Vector Control	Message Data	Upper Address	Lower Address	Entry N-1

Figure 2.22. MSI-X Table Entries

2.11.4.3. Pending Bit Array

The Pending Bit Array (PBA) is located within the memory address. This can use the same MSI-X Table BIR value (that is the same BAR or a different BAR). The PBA can use either qword (64-bit) or Dword (32-bit) accesses. The PBA table contains the pending bit information for each interrupt used. Same as MSI interrupts, if the event that the interrupt triggers and if its mask bit is set, the MSI-X transaction is not sent and the corresponding pending bit is set. If the interrupt vector is unmasked and if the pending bit is still set, that interrupt is generated.

DW1	DW0	
Pending Bits 0 - 63		QW 0
Pending Bits 64 - 128		QW 1
Pending Bits 128 - 191		QW 2
……		
Pending Bits		QW (N-1)/64

Figure 2.23. Pending Bit Array

2.11.4.4. MSI-X Interrupts Operation

- When the MSI-X interrupts are supported, you need to mention the size and location of the MSI-X Table and Pending Bit Array (PBA) through the PCIe CSR and the MSI-X table and the PBA structure must be implemented at the application layer.
- When the MSI-X Interrupts are generated, it uses the contents of the MSI-X Table (Address and Data) and generate a Memory Write through the TLP interface.
- The Host reads the message control register to determine the MSI-X Table size. The maximum entry in the table is 2048 entries. The BAR mentioned in the table BIR can access the MSI-X table.
- The host sets up the MSI-X table by programming the address, data, and the mask bits for each entry in the table.
- When the application generates the interrupt, it reads the MSI-X table information and generates a MWR TLP data and the corresponding bits in the PBA is set.
- The generated TLP is sent to the corresponding address along with the data.
- When the MSI-X interrupt is sent, the application can clear the associated PBA bits.

2.12. PCIe Endpoint Core Buffers

The Lattice PCIe x8 IP Core contains three large RAM buffers:

- Transmit Buffer for transmitting TLPs.
- Receive Buffer for receiving TLPs.
- Replay Buffer for holding TLPs that were transmitted until positive acknowledgement of receipt is received.

The size of the Transmit Buffer, Receive Buffer, and Replay Buffer and the size of the corresponding buffers in the remote PCI Express Device have a fundamental impact on the throughput performance of the PCI Express link.

To achieve the highest throughputs, the buffers for both devices in the PCI Express link must be large enough that they can still accept more data while the oldest data begins to be freed from the buffer. If a buffer is too small, then the link stalls until the buffer has enough space to continue. The buffers must be large enough to overcome the expected latencies or the throughput is affected.

2.12.1. PCI Express Credits

The Flow Control DLLPs communicates the available buffer space in units of Header and Data Credits as defined in the PCI Express Specification. The amount of space required by a Header is 12-20 bytes or (3-5 DWORDs with 1 DWORD == 4 bytes). Each Header Credit represents the capability to store a maximum size packet header, which includes all the transaction control information (address, byte enables, and requester ID) and an optional End to End CRC (ECRC). Each Data Credit represents 16 bytes (4 DWORDs) of data payload. A transaction cannot be transmitted unless there is at least 1 header credit and enough data credits for the packet payload available in the remote device's Receive Buffer.

Credits are further divided into three categories for each of the main types of traffic:

- Posted (memory write requests and messages)
- Non-Posted (all reads, Configuration and I/O writes)
- Completion (responses to Non-Posted Requests) credit categories.

Each type of traffic must obey the PCI Express transaction ordering rules and is stored in its own buffer area. The Credit categories are annotated as:

- PH – Posted Request Header Credits
- PD – Posted Request Data Payload Credits
- NH – Non-Posted Request Header Credits
- ND – Non-Posted Request Data Payload Credits
- CH – Completion Header Credits
- CD – Completion Data Payload Credits

The PCI Express is inherently high-latency due to the serial nature of the protocol (clock rate matching and lane-lane de-skewing) and due to the latency induced by requiring packets to be fully received and robustly checked for errors before forwarding them for higher-level processing.

To achieve the best throughputs, both the Lattice PCIe x8 IP Core and the remote PCI Express device must be designed with a suitable number of credits and the capability to overlap transactions to bury the transaction latency.

The Lattice PCIe x8 IP Core Transmit, Receive, and Replay buffers are delivered with sufficient size to overcome the latencies of typical open system components.

2.12.2. Max Payload Size

The maximum payload size of any given packet is limited by the Max Payload Size field of the Device Control Configuration Register. The PCI Express Specification defines 128, 256, 512, 1024, 2048, and 4096-byte payload sizes. The maximum payload size that a device can support is limited by the size of its posted and completion TLP buffers. The Transmit Buffer and Receive Buffer Posted and Completion TLP storage and the Replay Buffer TLP storage needs to be able to hold at least four Max Payload Size TLPs to be reasonably efficient. Each device advertises the maximum payload size that it can support, and the OS/BIOS configures the devices in a link to use the lowest common maximum payload size. Thus, it is not advantageous to support a greater maximum payload size than the devices with which one is communicating.

The higher the TLP payload size, the lower the TLP header and framing overhead is compared to the data. Above 512-byte Max Payload Size the incremental throughput benefit of higher payload sizes is small and the design area and latency for using these larger payloads is expensive. Thus, it is generally recommended to design for ≤ 512 Max Payload Size.

The Lattice PCIe x8 IP Core supports up to 512 Bytes Max Payload Size and the internal buffers can hold about 3x of the max payload size. However, given that typical PCIe devices currently available to communicate with support 256-byte maximum payloads, supporting greater than this amount is not likely to result in better performance and consumes more memory/logic resources.

2.13. Hard IP Interface

2.13.1. PHY Interface

The Link Layer is used in conjunction with a PCI Express PHY to implement a complete Lattice PCIe x8 IP Core PCI Express implementation. The PHY implements the high-speed serial and analog functions required to support PCI Express while the Link Layer implements most of the digital logic as well as the higher levels of the PCI Express protocol.

The PIPE PHY Interface that connects the Link Layer and PHY is not shown since the interface is only internal and is not visible to you.

The physical interface includes the differential receive and transmit signals along with the differential reference clock to the PCIe.

2.13.2. TLP TX/RX Interface

The Lattice PCIe core implements a complete PCI Express implementation including Physical, Data Link, and Transaction Layer functionality.

The PCI Express TLPs is transmitted on the PCI Express link through the transmit interface and received from the PCI Express link through the receive interface.

The PCIe core uses the Transaction Layer Interface as data interface to transmit/receive the packet in the form of TLP header and payload (if applicable). A minimum of 3DW TLP header is sent through TLP interface. Typically, the TLP interface is sized with a data width of 64-bits per lane. Consequently, the effective interface width scales with the number of lanes: 128-bits for x2 lane configuration, 256-bits for x4 lane configuration, and 512-bits for x8 lane configuration.

To transmit/receive a single TLP packet without payload, x1 lane configuration takes a duration of 2 clock cycles whereas x2, x4, and x8 lane configurations take a duration of 1 clock cycle respectively.

All TLPs on the Transaction Layer Receive and Transmit Interfaces, which are processed through link0_rx_data_o/link0_tx_data_i port(s) must be transmitted in the TLP format. The link0_rx_sel_o and link0_rx_cmd_data_o ports provide useful decoded information about the TLP through receive interface to enable you to determine the destination of the packet (for example, BAR and tag), whether it is a write or a read without having to read and parse the TLP. This allows you to optimize the code to reduce latency and relieves necessity for you to decode the TLP header to determine the packet's destination.

2.13.2.1. TLP Header Description

The Lattice PCIe uses 3DW or 4DW header for memory transactions to transfer the data in the form of TLP packets. The description of each field is described as shown in Figure 2.24 and Figure 2.25.

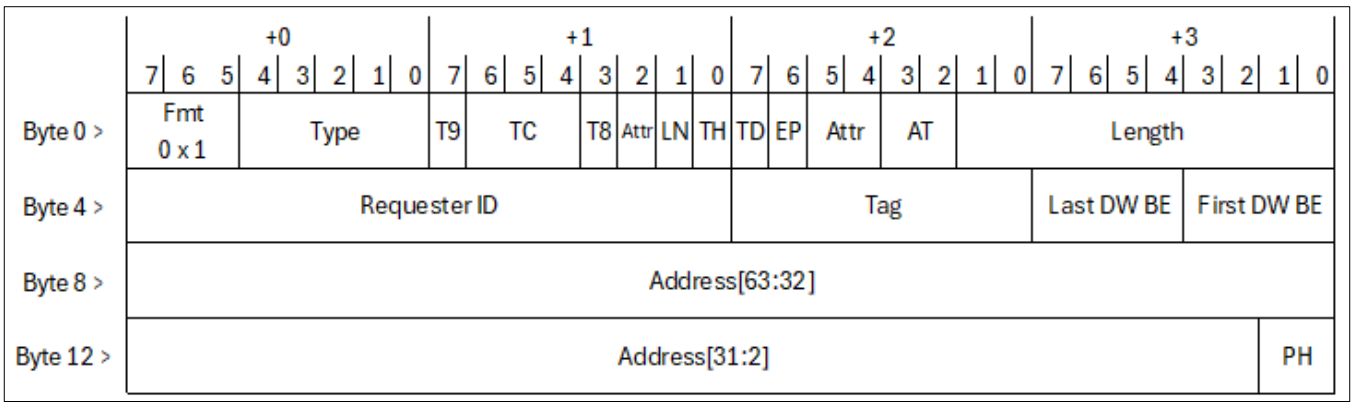


Figure 2.24. TLP Memory Request Header Format for 64-bit Addressing of Memory

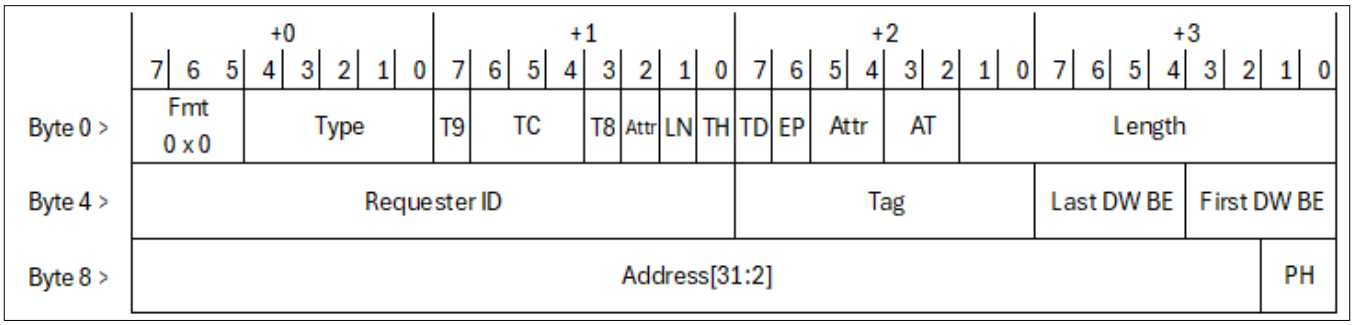


Figure 2.25. TLP Memory Request Header Format for 32-bit Addressing of Memory

Table 2.51 lists the description of each field. Fields not listed on the table are considered not applicable. The Transmit Interface is expected to drive these fields to 0, while the Receive Interface must disregard the values.

Table 2.51. TLP Header Field (Memory Transaction)

Field Name (with Size)	Header Byte/Bit	Function
Fmt [1:0] (Format)	Byte 0 Bits 6:5	Packet Formats: 00b = Memory Read 10b = Memory Write
Type [4:0]	Byte 0 Bits 4:0	TLP packet Type field: 00000b = Memory Read or Write
TC [2:0] (Traffic Class)	Byte 1 Bits 6:4	These bits encode the traffic class to be applied to a Request and to any associated completion. 000b = Traffic Class 0 (Default)
TD (TLP Digest)	Byte 2 Bit 7	If 1, the optional TLP Digest field is included with this TLP.
EP (Poisoned Data)	Byte 2 Bit 6	If 1, the data accompanying this packet is considered to have an error although the transaction is allowed to complete normally.
Attr [1:0] (Attribute)	Byte 2 Bits 5:4	Bit 5 = Relaxed ordering. When set to 1, PCI-X relaxed ordering is enabled for this TLP. Otherwise, strict PCI ordering is used. Bit 4 = No Snoop. If 1, system hardware is not required to cause processor cache snoop for coherency for this TLP. Otherwise, cache snooping is required.
Length [9:0]	Byte 2 Bits 1:0 Byte 3 Bits 7:0	TLP data payload transfer size, in DW. Maximum size is 1024 DW (4 kB).
Requester ID [15:0]	Byte 4 Bit 7:0 Byte 5 Bit 7:0	Identifies a requester's return address for a completion: Byte 4, 7:0 = Bus Number Byte 5, 7:3 = Device Number Byte 5, 2:0 = Function Number
{T9, T8, Tag [7:0]}	Byte 1 Bit 7 Byte 1 Bit 3 Byte 6 Bit 7:0	These identify each outstanding request issued by the Requester. By default, only bit 4:0 is used, allowing up to 32 requests to be in progress at a time. If the Extended Tag bit in the Control Register is set, then all 8 bits may be used (256 tags). T9 and T8 are included for PCIe Gen4 10 bits tag capable device.
Last DW BE [3:0] (Last DW Byte Enables)	Byte 7 Bit 7:4	These qualify bytes within the last DW of data transferred.
First DW BE [3:0] (First DW Byte Enables)	Byte 7 Bit 3:0	These qualify bytes within the first DW of the data payload.

Field Name (with Size)	Header Byte/Bit	Function
Address [31:2]	For 32 bits addressing format Byte 8 Bit 7:0 Byte 9 Bit 7:0 Byte 10 Bit 7:0Byte 11 Bit 7:2 For 64 bits addressing format Byte 12 Bit 7:0 Byte 13 Bit 7:0 Byte 14 Bit 7:0 Byte 15 Bit 7:2	The 32 bits start address for the memory transfer are used. The lower two bits of the address are reserved, forcing a DW-aligned start address.
Address [63:32]	For 64 bits addressing format Byte 8 Bit 7:0 Byte 9 Bit 7:0 Byte 10 Bit 7:0 Byte 11 Bit 7:0	Applicable to 64 bits addressing only. The upper 32 bits start address for the memory transfer are used.

2.13.2.2. TLP Transmit Interface

The Transmit Interface is the mechanism with which you transmit PCI Express TLPs over the PCI Express bus. You send a complete TLP comprised of 3DW or 4DW packet header, data payload, and optionally a TLP Digest. The core Data Link Layer adds the necessary framing (STP/END/EDB), sequence number, Link CRC (LCRC), and optionally computes and appends the ECRC (TLP Digest) when ECRC is not already present in the TLP.

You transmit TLPs as completion packets in response to non-posted transaction packets sent by the Lattice PCIe IP core. If the remote device does not have sufficient space in the receive buffer for transmit TLPs, the Lattice PCIe IP core pauses the TLP transmission until space becomes available.

The Transmit Interface includes the option to nullify TLPs (instruct the Receiver to discard the TLP) to support you to cancel TLP transmissions when errors are detected after the TLP transmission has started. Nullified TLPs that target internal core resources (Root Port Configuration Registers and Power Management Messages) are discarded without affecting the internal core resources. Nullified TLPs that do not hit internal resources are discarded.

Transmit Credit Interface

The Transmit Credit Interface provides the means for flow control of non-posted transmit transactions between you and the core transmit buffer. This is important for allowing Posted and Completion TLPs to continue to make progress when non-posted TLPs are blocked (which can be necessary in some cases to avoid potential deadlock conditions). The amount of non-posted TLP storage in the core transmit buffer is communicated on the transmit credit Interface. You are expected to use this interface to limit simultaneously outstanding TLP transmission of non-posted TLPs, to the amount of non-posted TLPs that the core can absorb into the non-posted transmit buffer.

When the core Transaction Layer for Link[i] is ready to accept TLP transmissions, the core asserts link0_tx_credit_init_o == 1 for one clock cycle and indicates the non-posted TLP Header storage capacity (NH) of the transmit buffer on link0_tx_credit_nh_o[11:0] on the same cycle. You are expected to keep and initialize the non-posted TLP Header capacity (NH) available transmit credit counters on link0_tx_credit_init_o==1.

When a non-posted TLP is pending for transmission, you must check the currently available NH credit count for the associated link and hold the transmission until enough NH credits are available to transmit the TLP. Once the TLP is committed for transmit, the amount of NH credits required by the TLP are decremented from the NH credit count. The core forwards transmitted TLPs from the transmit buffer and thus makes room for new TLPs, the core asserts link0_tx_credit_return_o==1 for one clock cycle and places the number of NH credits being returned on link0_tx_credit_nh_o[11:0].

In this manner, you can manage sending only enough non-posted TLPs that the core can hold in its Transmit Buffer. This allows you to know when non-posted TLPs are blocked and thus sends posted and/or completed TLPs instead. This is important for avoiding deadlocks and keeps non-posted TLP blockage from reducing posted and completion throughput. When the core receives more non-posted TLPs than the core can store in its non-posted TLP transmit storage, the core pauses the TLP transmission rather than allow an overflow to occur. Thus, if you do not wish to use the Transmit Credit

Interface, you may ignore this interface provided you are willing to permit blocked non-posted TLPs from also blocking following posted and completion TLPs.

Note that core/link partner transmit TLP flow control is not managed through this interface, the core manages transmit flow control between the core and the PCIe link partner Receive Buffer without user intervention.

Transmit Interface Example Transactions

As mentioned, the TLP data interface option is made available when non-DMA support is enabled through PCIe user interface. The following are examples of the memory read transactions that you need to send in completion to the read requests.

In case of memory read transactions, the Lattice PCIe IP core sends the header packet, which contains information about the address and size of data that you need to send in completion to the received packet. You need to send the completion packet with the header followed by the data when the link0_tx_ready_o signal from the PCIe is high as the data you sent is validated only when PCIe is ready. The TLP interface width is sized based on the number of lanes used, the packet header and data are transmitted accordingly as shown in the below figures for eight lanes (512 bits), four lanes (256 bits), two lanes (128 bits), and one lane (64 bits) respectively.

In the timing diagrams below, the Memory Read transaction features a 3DW header without a payload or ECRC. It is assumed that the completion TLP packet to be transmitted exceeds the width of the TLP interface, requiring multiple cycles to complete the transaction.

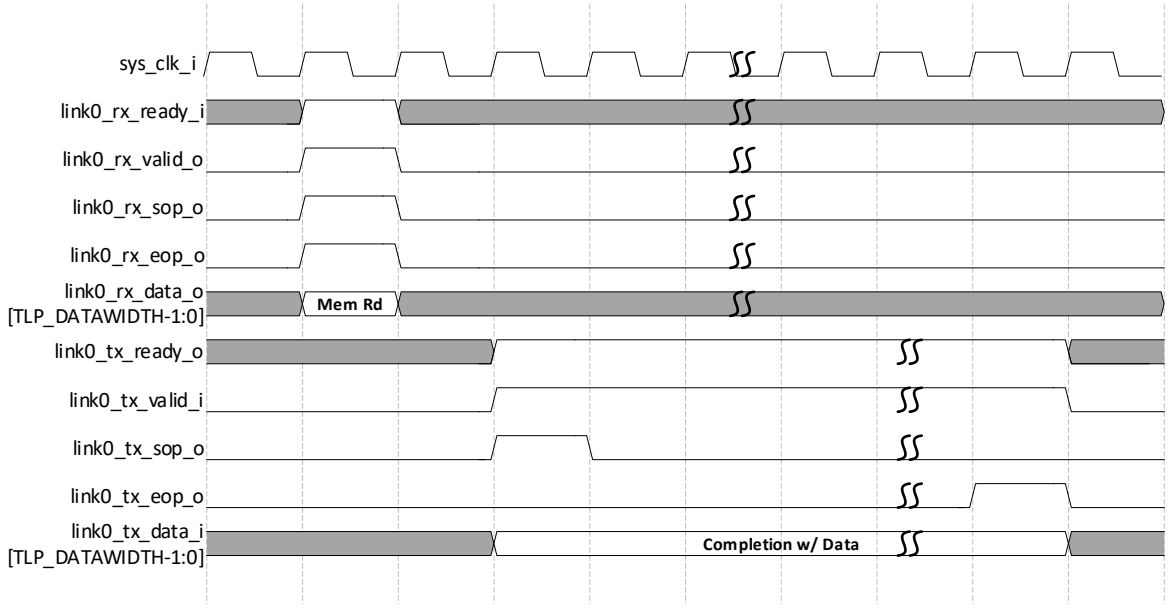


Figure 2.26. TLP Memory Read Operation for Link0 (TLP Interface Width of 512 Bits, 256 Bits, and 128 Bits)

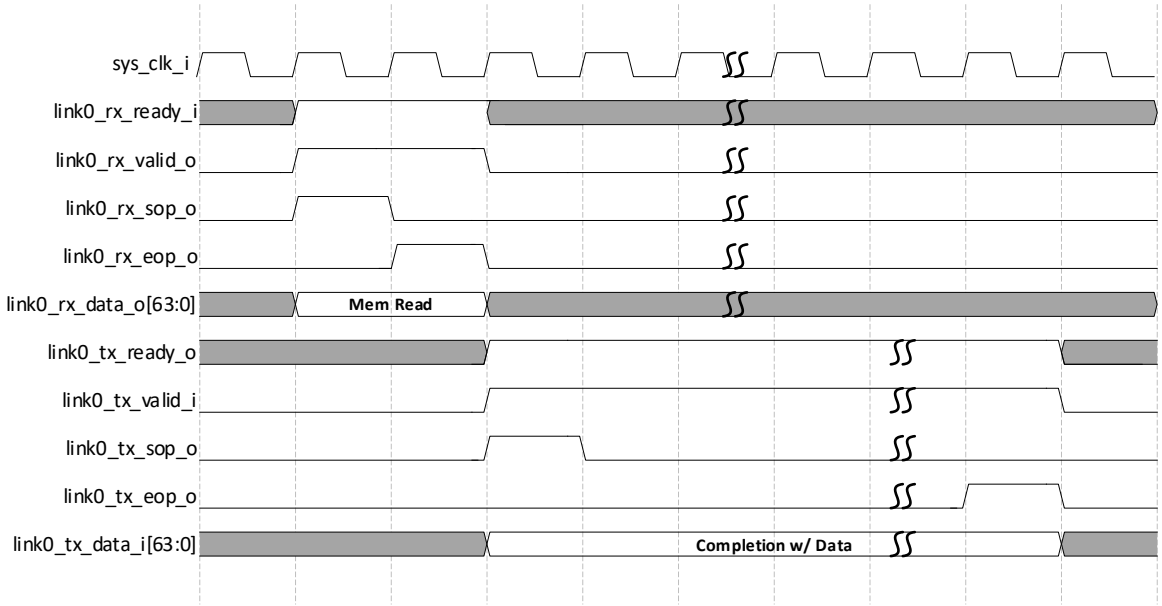


Figure 2.27. TLP Memory Read Operation for Link0 (TLP Interface Width of 64 Bits)

Figure 2.28 and Figure 2.29 show the TLP transaction according to the tx_ready_o behavior based on the minimum timing of link0_tx_ready_o:

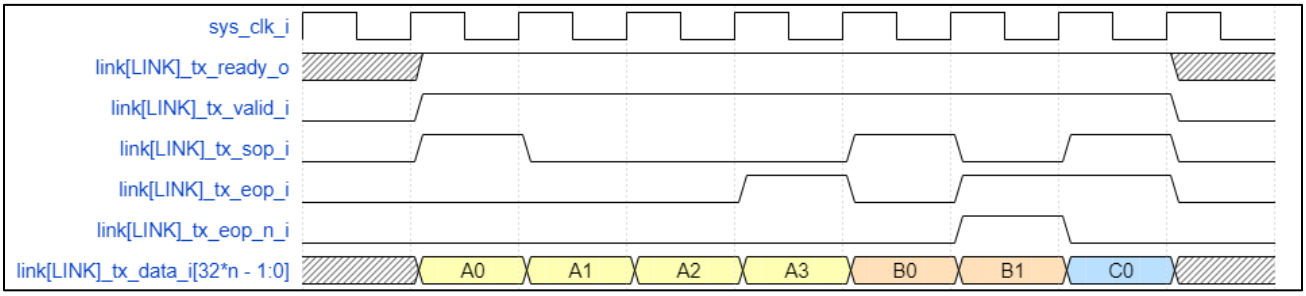


Figure 2.28. Minimum link0_tx_ready_o Timing Diagram

Transaction A begins on cycle 2 with the assertion of link0_tx_sop_i and ends on cycle 5 with the assertion of link0_tx_eop_i==link0_tx_valid_i==link0_tx_ready_o==1. The packet transfers with minimum timing with link0_tx_valid_i==link0_tx_ready_o==1 on cycles 2-5.

Transaction B begins immediately after Transaction A on cycle 6 with the assertion of link0_tx_sop_i and ends on cycle 7 with the assertion of link0_tx_eop_i==link0_tx_valid_i==link0_tx_ready_o==1. The packet transfers with minimum timing with link0_tx_valid_i==link0_tx_ready_o==1 on cycles 6 to 7. Transaction B is nullified (dropped) during packet forwarding on cycle 7 because of the following conditions: link0_tx_eop_n_i==1 happens when link0_tx_eop_i==1.

Transaction C begins immediately after Transaction B on cycle 8 with the assertion of link0_tx_sop_i and ends on the same cycle with the assertion of link0_tx_eop_i==link0_tx_valid_i==link0_tx_ready_o==1. The packet transfers with minimum timing with link0_tx_valid_i==link0_tx_ready_o==1 on cycle 8 considering the wait state timing of link0_tx_ready_o.

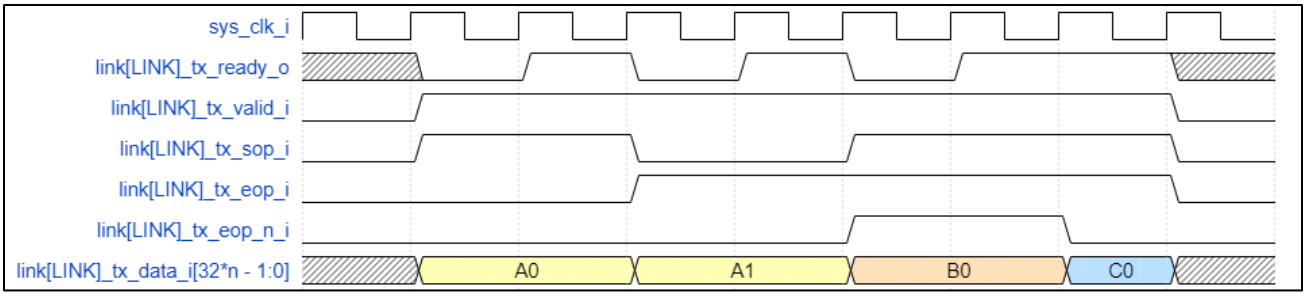


Figure 2.29. Wait State of link0_tx_ready_o Timing Diagram

Transaction A begins on cycle 2 with the assertion of link0_tx_sop_i and ends on cycle 5 with the assertion of link0_tx_eop_i==link0_tx_valid_i==link0_tx_ready_o==1. The packet transfers only on cycles 3 and 5 when link0_tx_valid_i==link0_tx_ready_o==1.

Transaction B begins immediately after Transaction A on cycle 6 with the assertion of link0_tx_sop_i and ends on cycle 7 with the assertion of link0_tx_eop_i==link0_tx_valid_i==link0_tx_ready_o==1. The packet transfers only on cyle 7 when link0_tx_valid_i==link0_tx_ready_o==1. Transaction B is nullified (dropped) during packet forwarding on cycle 7 because of the following conditions: link0_tx_eop_n_i==1 happens when link0_tx_eop_i==1.

Transaction C begins immediately after Transaction B on cycle 8 with the assertion of link0_tx_sop_i and ends on the same cycle with the assertion of link0_tx_eop_i==link0_tx_valid_i==link0_tx_ready_o==1. The packet transfers with minimum timing (no wait states) with link0_tx_valid_i==link0_tx_ready_o==1 on cycle 8.

Transmit Interface Considerations

The following considerations are provided to simplify logic using the Transmit Interface and to address common problems, which must be avoided:

- For each TLP that you transmit, the core adds a minimum of 2-bytes of STP/END/EDB framing, a 2-byte Sequence Number, and a 4-byte Link CRC for a total of 8 bytes (64-bits). These additional 8 bytes, which the core transmits but do not appear on link [LINK]_tx_data_i, allows you the flexibility of not using every clock cycle on the Transmit Interface. This flexibility can be useful to simplify user logic and improve design timing closure.
- Completions, which are transmitted in response to a previously received non-posted request, must reflect the Traffic Class, Requester ID, Tag, and Attributes of the original request. While most of these are obvious, it may not be obvious to reflect the attributes, and this is known to cause problems on some systems.
- When the link trains at less than full width or speed, link [LINK]_tx_ready_o is gaped in relation to the number of lanes being used and the number of lanes available in the core. You must remember to include a simulation case which forces the link into lower than full-width and/or speed to test that the logic properly handles the gaping of link [LINK]_tx_ready_o and the corresponding lower data transfer rate in this case.
- While TLPs are transmitted over PCI Express, these are placed into a replay buffer in case the TLPs need to be replayed due to transmission errors. The core negotiates the replay process in conjunction with the remote PCI Express Device and does not require any user intervention. You can monitor the frequency of replays, if desired, by monitoring the appropriate error status registers.
- The Lattice PCIe core interface is designed to support high throughput applications. Small interruptions in transmissions occurs, however, as the core periodically needs to transmit link management DLLPs and SKP Ordered Sets and may also need to transmit error messages, configuration write/read completions, and interrupt TLPs.
- The Lattice PCIe core handles all Data Link Layer functionality for you and handles most of the error cases for you as well. To accomplish these functions, the core occasionally delays your Local Transmit Interface requests while it completes its own TLP transmissions for these purposes. All the core's TLP transmissions are short, so it delays your request for only a few clock cycles. The core transmits DLLPs used for link maintenance, TLP messages to communicate errors, interrupt TLPs, and completions to notify the system of malformed or un-routable TLP requests.

If the user TLP transmit requests are delayed for extended periods of time, this may be due to insufficient link partner receive buffer space or local replay buffer space or due to the link having to wake from a lower power state or recover from an error before transmission can occur.

2.13.2.3. TLP Receive Interface

The Receive Interface is the mechanism with which receives the PCI Express TLPs from the PCI Express link partner. You receive complete Transaction Layer Packets (TLPs) comprised of a three or four DWORD TLP header, data payload (if present), and TLP Digest (ECRC, if present).

The TLPs, which were received without errors and were not nullified, are presented on the receive interface. Therefore, the user logic only needs to handle valid received TLPs.

The PCIe core forwards the TLPs only after considering the following checks:

- The core checks received TLPs for transmission errors (Sequence Number or LCRC error) and negotiates replay of TLPs with the link partner as required.
- The core discards TLPs which are nullified by the link partner during transmission (TLP is received without transmission errors and with EDB instead of END framing).
- The core checks received TLPs which were received without transmission errors and without being nullified for Malformed TLP due to length and content errors.
  - If the core determines that a received TLP is malformed due to length (TLP length calculated from the received TLP Header Format and Type, Length, and TLP Digest does not match the received TLP length), the core discards the TLP and report the error.
  - If the TLP fails to hit an enabled resource or is malformed due to its content (invalid Traffic Class, invalid Format and Type, and invalid Byte Enables), the core discards the TLP and reports the error.

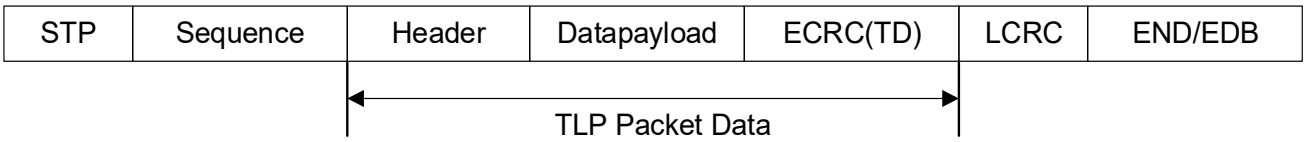


Figure 2.30. TLP Packet formation by the Lattice PCIe IP core

If the TLP passes all the above checks, it is considered a valid TLP and is forwarded to the receive interface for the user's logic to consume. The core strips the Physical Layer framing (STP/END/EDB) and Data Link Layer Sequence Number and Link CRC (LCRC) before presenting the TLPs to you on the Receive Interface. The core does not strip the received TLP ECRC (if present) as some user designs require forwarding the ECRC either to transmit the TLP out another PCIe port. The ECRC value is also checked at a later point in the user's data path to continue the ECRC error detection protection for a larger portion of the receive data path. If an ECRC is present in the TLP, the core checks the validity of the received ECRC and reports detected ECRC errors on the receive interface.

The core also decodes received TLPs against its Configuration Registers and provides the transaction decode information on the Receive Interface such that the TLP can be directed to the appropriate destination without the need for you to parse the TLP until its destination. For example, if the received TLP is an I/O or Memory write or read request, the Base Address Register (BAR) resource that is hit is indicated and if the TLP is a completion, the TLP's tag field is provided. The core also provides additional useful transaction attributes.

Receive Credit Interface

The Receive Credit Interface provides the means for flow control of non-posted receive transactions between the core receive buffer and user receive TLP logic. This is important for allowing Posted and Completion TLPs to continue to make progress when non-posted TLPs are blocked (which is necessary in some cases to avoid potential deadlock conditions). The amount of non-posted TLP storage in the user design is communicated on the Receive Credit Interface. The core uses this interface to limit the simultaneously outstanding receive non-posted TLPs to the amount of non-posted TLPs that the user design advertises that it can absorb into the non-posted receive buffer.

When you are ready to accept non-posted TLP reception, assert the link0_rx_credit_init_i == 1(where LINK=0 or 1) for one clock cycle and the non-posted TLP header storage capacity of the user design is indicated through link0_rx_credit_nh_i[11:0] on the same clock cycle. Holding off credit initialization for an extended period can cause received non-posted TLP transactions to timeout in the source component which may cause serious errors.

The core limits simultaneous outstanding non-posted receive TLPs on the receive interface to ensure no more than the initialized NH credits are simultaneously outstanding to user receive TLP logic.

Once the received non-posted TLPs are processed/forwarded such that more room is available to receive new non-posted TLPs, assert the link0_rx_credit_return_i==1 for one clock cycle and place the number of NH credits being returned on link0_rx_credit_nh_i[11:0]. In this manner, you can limit the outstanding core receive TLPs to the user design. This permits the core to know when non-posted TLPs are blocked and thus send posted and/or completion TLPs to the user design instead. This is important for avoiding deadlocks and keeps non-posted TLP blockage from reducing posted and completion throughput.

Note that the link partner/core receive TLP flow control is not managed through this interface; the core manages receive buffer flow control between itself and the PCIe link partner transmit gating function without user intervention.

Receive Interface Example Transactions

The Lattice PCIe core sends the data in the form of TLP packets when non-DMA option is enabled. The receive interface presents the TLP data through link0_rx_data_o signal. The data is validated only when link0_rx_valid_o signal is high and you are ready (for example, link0_rx_ready_i must be high to access the data sent by the core). As the Lattice PCIe core transmits TLP packet, which consists of 3DW or 4DW header along with data (in TLP frames), the last DW of TLP packet is sent as trash value(X) to ensure the complete TLP is transmitted.

The timing diagrams below show the receive interface behavior when the PCIe core receives a Memory Write TLP. The Memory Write TLP consists of 3DW header and payload, without ECRC. It is assumed that the TLP packet received exceeds the width of the TLP interface, requiring multiple cycles to complete the transaction.

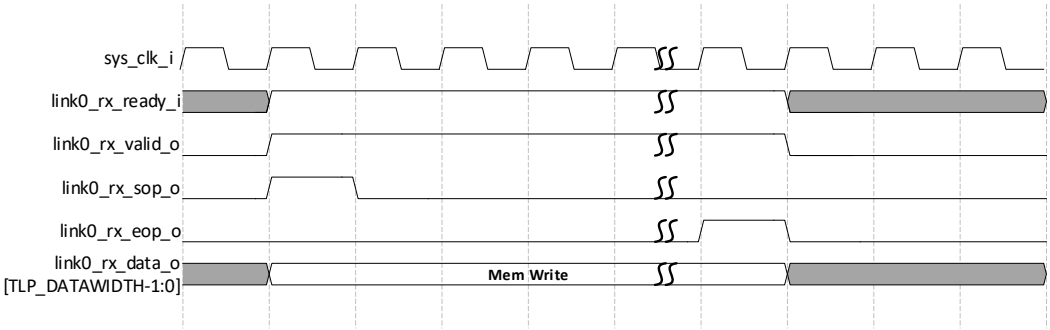


Figure 2.31 TLP Memory Write Operation for Link0 (TLP Interface Width of 512 Bits, 256 Bits, 128 Bits, and 64 Bits)

Figure 2.32 and Figure 2.33 show the TLP transaction according to the rx_ready_i behaviour based on the minimum timing of link0_rx_ready_i.

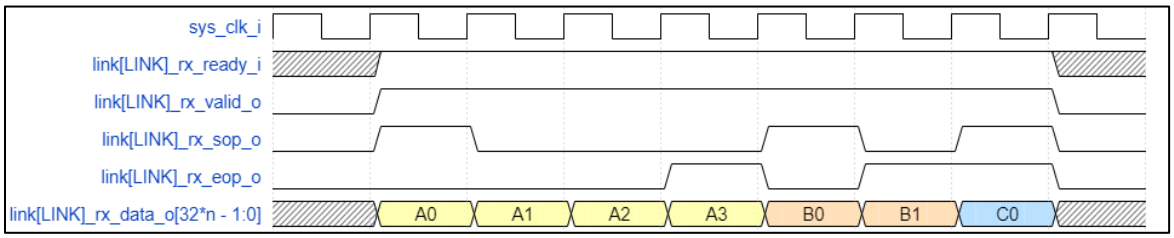


Figure 2.32. Minimum link0_rx_ready_i Timing Diagram

Transaction A begins on cycle 2 with the assertion of link0_rx_sop_o==link0_rx_valid_o==1 and ends on cycle 5 with the assertion of link0_rx_eop_o==link0_rx_valid_o==link0_rx_ready_i==1. The packet transfers with minimum timing since link0_rx_valid_o == link0_rx_ready_i == 1 on cycles 2 to 5. Data transfers on cycles 2 to 5.

Transaction B begins immediately after Transaction A on cycle 6 with the assertion of link0_rx_sop_o==link0_rx_valid_o==1 and ends on cycle 7 with the assertion of link0_rx_eop_o==link0_rx_valid_o==link0_rx_ready_i==1. Data transfers to cycles 6 to 7.

Transaction C begins immediately after Transaction B on cycle 8 with the assertion of link0_rx_sop_o==link0_rx_valid_o==1 and ends on the same cycle since link0_rx_eop_o==link0_rx_ready_i==1 is also asserted. Data transfers on cycle 8 considering the wait state timing of link0_rx_ready_i:

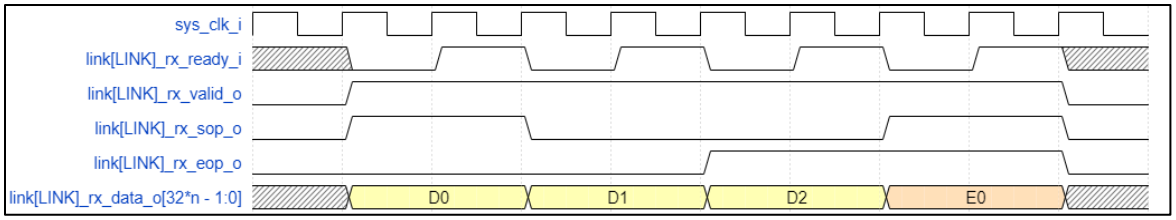


Figure 2.33. Wait State of link0_rx_ready_i Timing Diagram

Transaction D begins on cycle 2 with the assertion of link0_rx_sop_o==link0_rx_valid_o==1 and ends on cycle 7 with the assertion of link0_rx_eop_o==link0_rx_valid_o==link0_rx_ready_i==1. The packet transfer wait states due to link0_rx_ready_i==0 on cycles 2, 4, and 6. Data transfers on cycles 3, 5, and 7.

Transaction E begins on cycle 8 with the assertion of link0_rx_sop_o==link0_rx_valid_o==1 but is wait stated due to link0_rx_ready_i==1 on cycle 8. On cycle 9 the transaction completes with link0_rx_sop_o==link0_rx_eop_o==link0_rx_valid_o==link0_rx_ready_i==1.

Receive Interface Considerations

The following considerations are provided to simplify logic using the receive interface and to address common problems, which must be avoided:

- For each TLP that you receive, the core strips a minimum 2-bytes of STP/END/EDB framing, a 2-bytes of Sequence Number, and a 4-bytes of Link CRC, for a total of 8 bytes (64-bits). These additional 8 bytes, which the core receives but which do not appear on link0_rx_data_o, allows you the flexibility of not using every clock cycle on the receive interface. This flexibility can be useful to simplify user logic and improve design timing closure.
- TLPs that appear on the receive interface have passed the Physical Layer and Link Layer error detection and correction logic and can be assumed to be free of transmission errors. When the core receives a TLP with a STP/END/EDB framing, Sequence Number, or Link CRC error, the core coordinates re-transmission of the TLP with the remote PCI Express device and only forwards packets that pass transmission error checks onto to the receive interface.
- TLPs that are received from PCI Express are decoded for validity against the core's configuration registers and are only forwarded to the receive interface if they hit an enabled resource. Therefore, you only need to handle valid TLPs which target the user resources. TLPs, which do not hit user resources, are terminated by the core and the appropriate error message and response is handled by the core on the user's behalf.
- The Lattice PCIe core handles all Data Link Layer functionality for you and handles most of the Transaction Layer error cases as well. The core consumes Configuration Transactions, Messages, and TLPs which do not map to user resources and transmits the appropriate response. TLPs which are handled by the core do not appear on the Receive Interface.
- User logic that manages read requests (for DMA) and assigns a tag to each read request that is transmitted. The core provides the tag of each received completion on link0_rx_cmd_data_o to allow user logic to route completions from different sources to the destination without having to parse the TLP for tag information. The core does not track the outstanding tags that are in use by the user. If a completion is received with a tag that does not correspond to an outstanding user read request, then you must report the error.

Data Byte Order

The core transmits the TLP data in the following byte order:

- link0_tx_data_i[7:0], link0_tx_data_i[15:8], link0_tx_data_i[23:16],…

The core receives the TLP data in the following byte order:

- link0_rx_data_o[7:0], link0_rx_data_o[15:8], link0_rx_data_o[23:16],…

For example, in the case of 64 bits TLP interface width, you transmit, or the core receives a 32-bit and 64-bit Memory Read Transaction Layer Packet in the following byte order as shown in Table 2.52 and Table 2.53 respectively.

Table 2.52. Data Byte Order for 32-bit Memory Read TLP

link0_tx_data_i/link0_rx_data_o	First and Second Data Word	Third Data Word
[7:0]	{R, Fmt[1:0], Type[4:0]}	Addr[31:24]
[15:8]	{T9, TC[2:0], T8, R[2:0]}	Addr[23:16]
[23:16]	{TD, EP, Attr[1:0], Length[9:8]}	Addr[15:8]

link0_tx_data_i/link0_rx_data_o	First and Second Data Word	Third Data Word
[31:24]	Length[7:0]	{Addr[7:2], R[1:0]}
[39:32]	RequesterID[15:8]	8'h0
[47:40]	RequesterID[7:0]	8'h0
[55:48]	Tag[7:0]	8'h0
[63:56]	{LastDWBE[3:0], 1stDWBE[3:0]}	8'h0

Table 2.53. Data Byte Order for 64-bit Memory Read TLP

link0_tx_data_i/link0_rx_data_o	First and Second Data Word	Third and Fourth Data Word
[7:0]	{R, Fmt[1:0], Type[4:0]}	Addr[63:56]
[15:8]	{T9, TC[2:0], T8, R[2:0]}	Addr[55:48]
[23:16]	{TD, EP, Attr[1:0], Length[9:8]}	Addr[47:40]
[31:24]	Length[7:0]	Addr[39:32]
[39:32]	RequesterID[15:8]	Addr[31:24]
[47:40]	RequesterID[7:0]	Addr[23:16]
[55:48]	Tag[7:0]	Addr[15:8]
[63:56]	{LastDWBE[3:0], 1stDWBE[3:0]}	{Addr[7:2], R[1:0]}

2.13.2.4. Transaction Layer Interface Error Detection and Correction

The Lattice PCIe IP Core has built in error detection and correction mechanisms for both Transaction Layer Packets (TLPs) which are transferred between PCI Express and Transaction Layer Interface and Data Link Layer Packets (DLLPs) which are used by the core internally for link management.

The Lattice PCIe IP core adds the required Physical Layer framing (STP/END/EDB) and Data Link Layer error detection and correction information (Sequence Number/Link CRC) to the TLP packets transmitted on the Transmit Interface. Likewise, when TLP packets are received from PCI Express, the core validates that the packet is received correctly by checking the Physical Layer framing (STP/END/EDB) and Data Link Layer error detection and correction information (Sequence Number/Link CRC). Packets that are forwarded to you on the receive interface are sent after stripping the Physical Layer framing (STP/END/EDB) and Data Link Layer error detection and correction information (Sequence Number/Link CRC).

If transmission errors are detected in packet transmission or reception, the core coordinates with the remote PCI Express device to retry the transaction and recover from the error. This process occurs without any user intervention. The Lattice PCIe core logs both corrected and uncorrected errors. This error status information is made available through the status registers and is accessed by system software through the Configuration Registers. The core generates and transmits error Message TLPs to the remote PCI Express device in response to different types of errors detected.

ECRC (TLP Digest) generation and checking is a core option. When ECRC generation support is enabled by the software (AER Capability: ECRC Generation Enable == 1), the core generates and adds ECRC to all transmitted TLPs (except those that already contain an ECRC with TD bit set to 1). When ECRC checking support is enabled by software (AER Capability: ECRC Check Enable == 1), the ECRC fields present in received TLPs are checked for validity and any errors are noted on the Receive Interface and are reported in the AER Capability. The core does not modify the ECRC or TD (TLP Digest == ECRC indicator) fields on received TLPs and passes these fields onto the receive interface as received.

The Lattice PCIe core also handles TLPs that are Type 0 Configuration transaction requests, messages requests for link management, TLPs that don't hit an enabled resource and any requests that the core determines are malformed.

If the core found TLP having transmission errors, then that TLP is consumed by the core (and not forwarded) and then transmits any required completion packet(s), generates required error messages and logs any required errors.

The core has been designed in such a way that it is feasible to you to only consume and generate the TLPs and can make use of these TLPs for transferring data and control information between your application and the remote PCI Express devices.

2.13.3. LMMI Interface

When you select the TLP as data interface option in the PCIe IP user interface, the IP by default configures LMMI as register interface. The Core Configuration and Status Registers (CSR) are made accessible to the user design through the Lattice Memory Mapped interface (LMMI).

An example of the register configuration through the LMMI is shown below in the LMMI write and read timing diagrams.

The data transaction, through the LMMI, only starts when both usr_lmmi_request_i and usr_lmmi_ready_o are asserted high. Consecutive request must be done with at least one clock period wait cycle (for example, usr_lmmi_request_i should de-assert first after a successful transaction before making another request).

When both usr_lmmi_request_i and usr_lmmi_ready_o are asserted high, usr_lmmi_wr_rdn_i, and usr_lmmi_offset_i must be valid and describe the transaction to execute; if the transaction is a write as indicated by usr_lmmi_wr_rdn_i being asserted to high, usr_lmmi_wdata_i must also be valid.

Notes:

- Only one request should be active at a given time.
- Once usr_lmmi_resetn_i de-asserted, you must wait 10 or more usr_lmmi_clk_i clocks before accessing LMMI interface. This is to allow reset to be properly propagated through the PCIe controller.

2.13.3.1. LMMI Write Operations

The data is written to PCIe registers only when PCIe IP gets a request from you by the asserting usr_lmmi_request_i signal to high, when the PCIe IP indicates that it is ready by asserting usr_lmmi_ready_o to high and when a write transaction is being indicated from your end by driving usr_lmmi_wr_rdn_i to high. When all three of these signals are asserted, a successful write transaction will take place.

For example, you need to write 0X01 data into 0X0A register and then 0X02 data into 0X0B register. The 0X01 data is written into 0X0A register in one transaction only as ready signal is high when request is asserted. But to write 0X02 data into 0X0B register took two transactions because ready signal is low when request is asserted in first transaction. Therefore, the data is written to the register in the second transaction only when ready signal is high as shown in Figure 2.34.

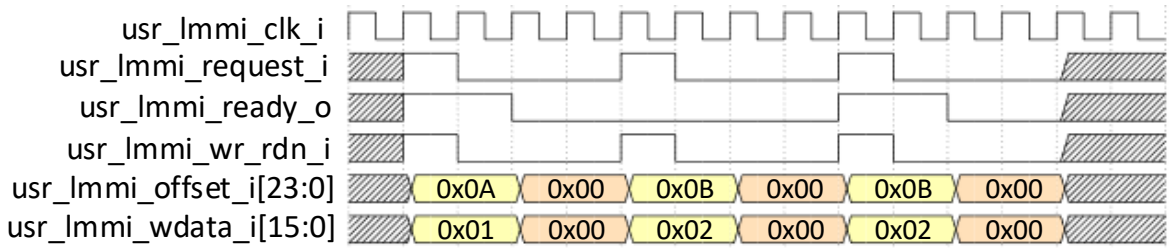


Figure 2.34. LMMI Write Operation

2.13.3.2. LMMI Read Operation

The data is read from PCIe registers only when PCIe IP gets a request from the user. For example, usr_lmmi_request_i must be asserted high to indicate a new request is being made and usr_lmmi_wr_rdn_i signal must be asserted low to indicate a read transaction. A valid offset value through usr_lmmi_offset_i needs to be supplied from your end during the request.

You can read the data from PCIe core registers only when the ready and read valid signals are received from PCIe IP. For example, the usr_lmmi_ready_o signal and the usr_lmmi_rdata_valid_o signal must be asserted high together for a valid read transaction to take place.

For example, you want to read the data [0X0000001d00000000] from lane 0 PMA Status register offset 0x7F. The transaction follows the steps as shown in Figure 2.35.

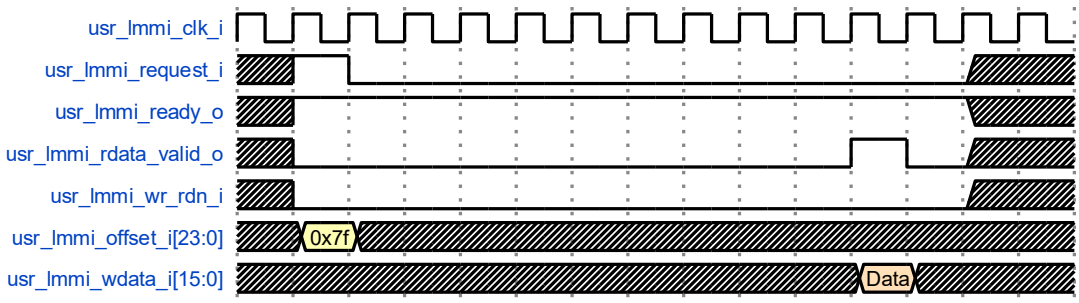


Figure 2.35. LMMI Read Operation

The following are the registers to be configured through the LMMI:

- Simulation Registers
  - Register address – 0x4_2000
    This register is used to reduce the ltssm ts_1 and timeouts to fasten the simulation when asserted as *1*.
  - Register address – 0x4_3000
    This register is used to reduce the Power Management State Machine timeouts to fasten the simulation when asserted as *1*.
  - Register address – 0x4_4000
    This register is used to reduce the timeouts to fasten the simulation when asserted as *1*.

2.14. Soft IP Interface

2.14.1. Data Interface Conversion

2.14.1.1. AXI-Stream Interface

This interface is available if the data interface type selected in the IP generation user interface is *AXI4_STREAM*. The data width of the transaction varies based on the lane support.

- Lane x8: 512 data width
- Lane x4: 256 data width
- Lane x2: 128 data width
- Lane x1: 64 data width

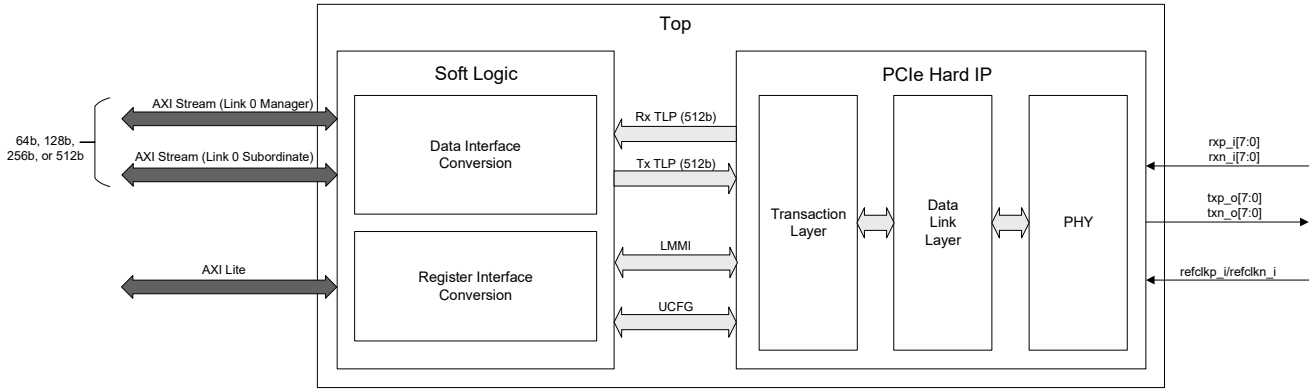


Figure 2.36. AXI-Stream Data Interface, AXI-Lite Register Interface

PCIe to AXI-Stream Transfer

For the PCIe to AXI-Stream transfer, the PCIe sends the data to the user application. The transaction has the header of size three double-word. After the header transaction, the actual data is transferred as shown in Figure 2.37 to Figure 2.39.

Note: The *DATA* in the transactions below are random data sent by the PCIe IP to compensate for the data width.

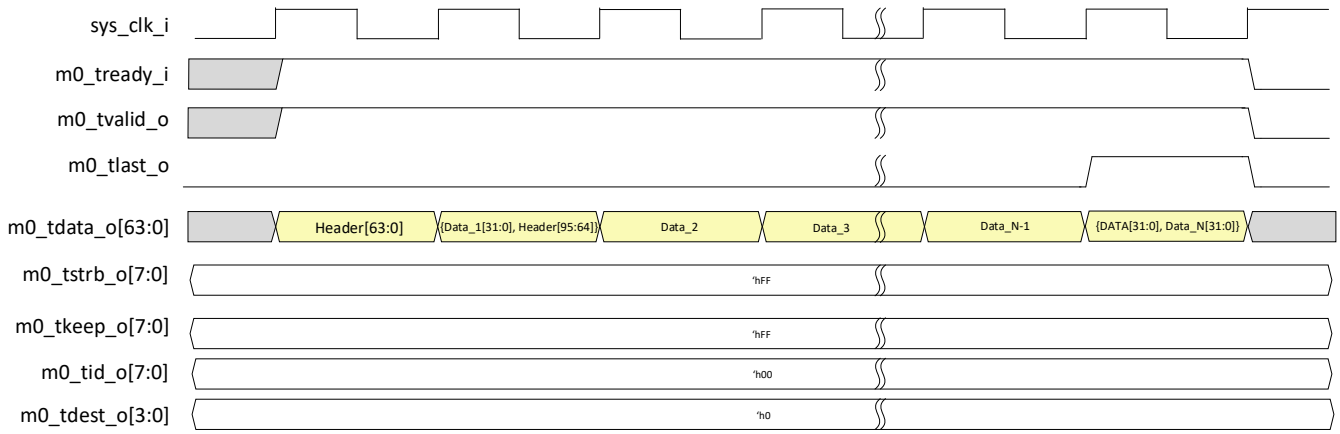


Figure 2.37. PCIe to AXI-Stream Transaction for x1

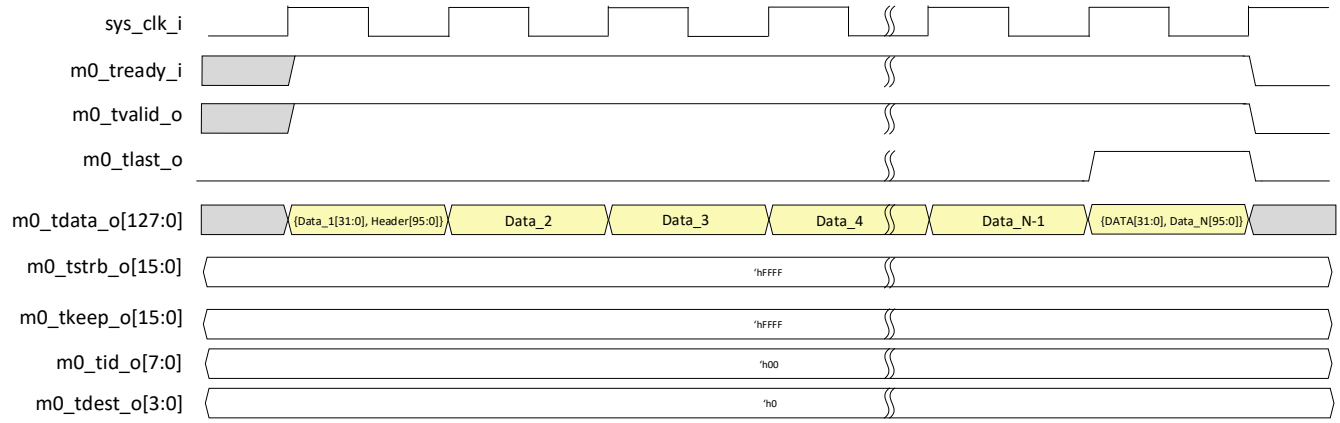


Figure 2.38. PCIe to AXI-Stream Transaction for x2

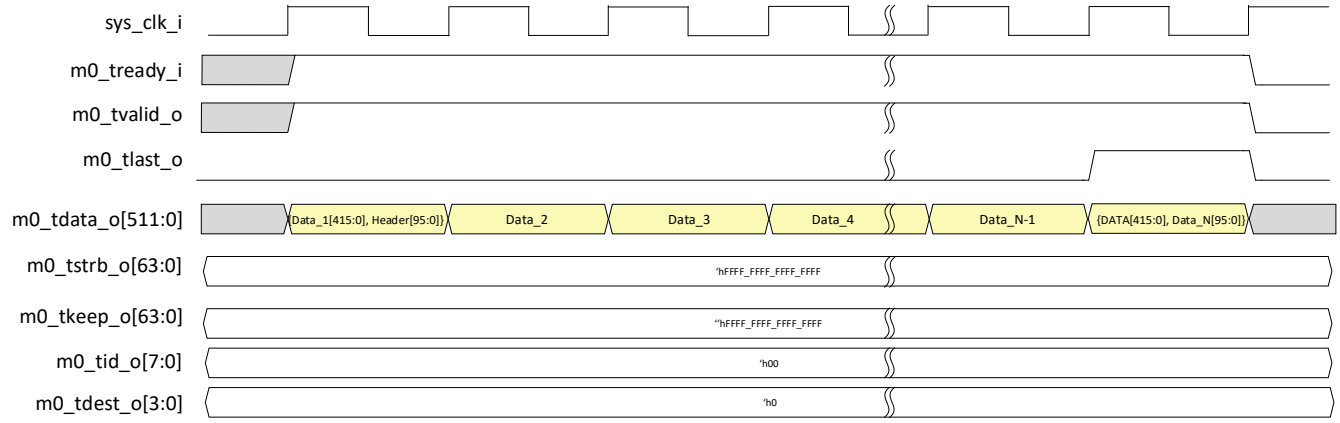


Figure 2.39. PCIe to AXI-Stream Transaction for x8

AXI-Stream to PCIe Transaction

For the AXI-Stream to PCIe transaction, the user application sends the data to the PCIe Endpoint IP. Similar to the PCIe to AXI-Stream transfer, there is a header data of size three double word, which is transferred first followed by the actual data as shown in Figure 2.40 to Figure 2.42.

Note: The *DATA* in the transactions below are random data sent by the PCIe IP to compensate the data width.

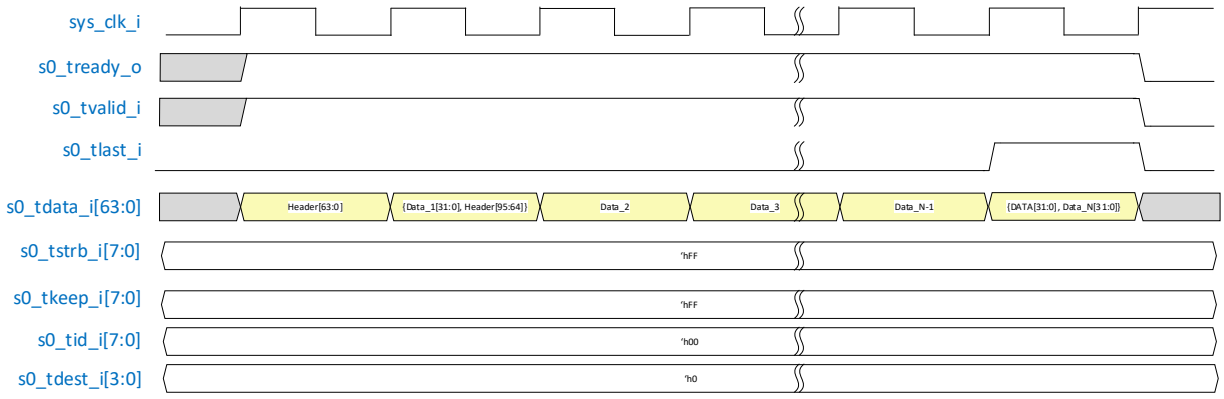


Figure 2.40. AXI-Stream to PCIe Transaction for x1

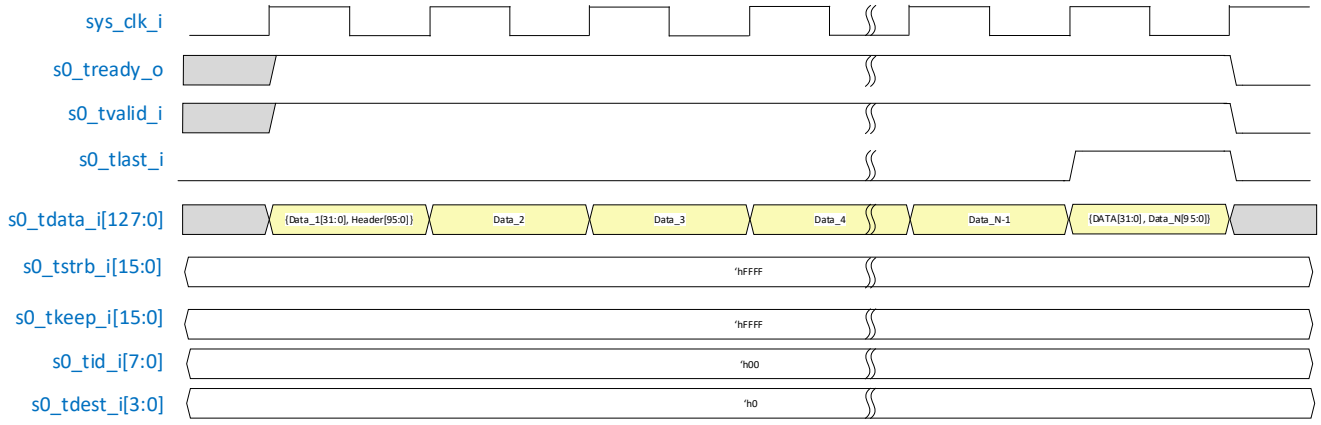


Figure 2.41. AXI-Stream to PCIe Transaction for x2

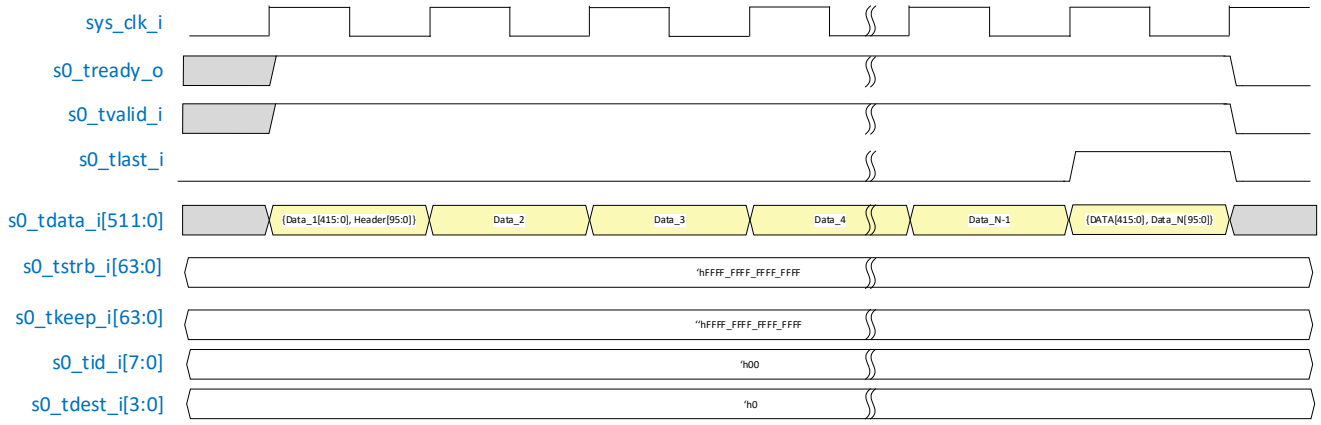


Figure 2.42. AXI-Stream to PCIe Transaction for x8

2.14.1.2. Bridge Mode

Bridge Mode is a non-DMA mode which allows the received MWr and MRd TLP to be converted to AXI-MM or AXI-Lite Manager Interface. Subordinate interface is not supported.

In the Radiant user interface, you can configure Bridge Mode interface type (AXI-MM or AXI-L), BAR number that is associated to DMA Bypass, and BAR size.

When a received MWr/MRd TLP targets Bridge Mode BAR, the IP converts the TLP to AXI-MM or AXI-Lite Manager Interface. The BAR value is masked off to 0 when presented at AXI Read/Write Address.

For MRd TLP, the read data at AXI-MM/AXI-Lite Read Data Channel is converted to CplD TLP and be transmitted to PCIe link partner.

The following are the limitations of Bridge Mode:

- Only 1-DW MWr/MRd TLP is supported. If AXI-MM interface is selected, it supports only 32-bit data width without burst mode (AWLEN and ARLEN are always 0).
- Only DW-aligned address is supported. The 4-bit LSB of read/write address must be 0x0, 0x4, 0x8, or 0xC.
- Only 32-bit addressing BAR is supported.

In addition, when Bridge Mode is selected by the Radiant user interface, user interrupt pins can be enabled. Refer to the User Interrupts section for more details.

Figure 2.43 and Figure 2.44 show the Bridge Mode Radiant user interface settings.

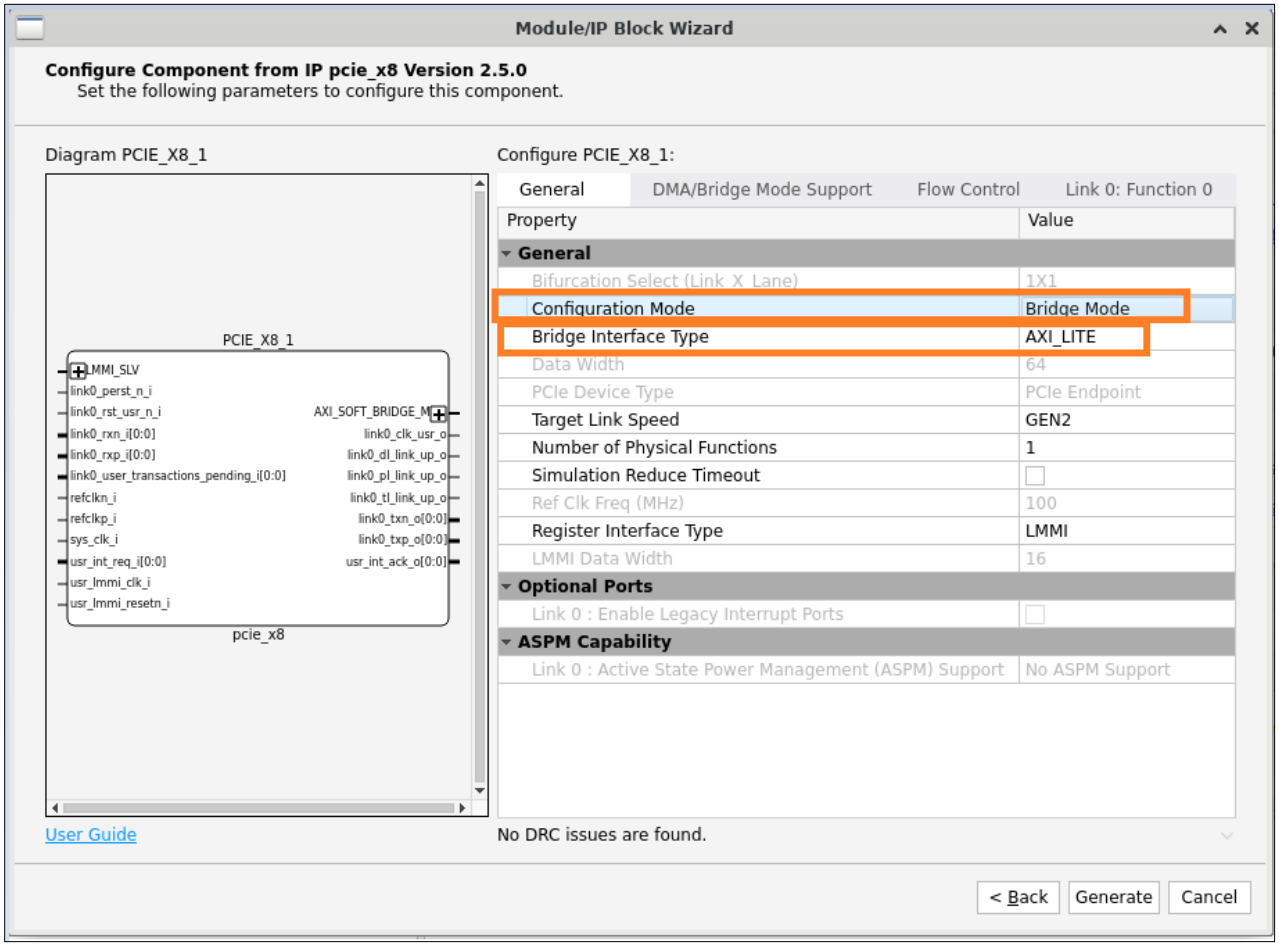


Figure 2.43. Bridge Mode Enablement (General Tab)

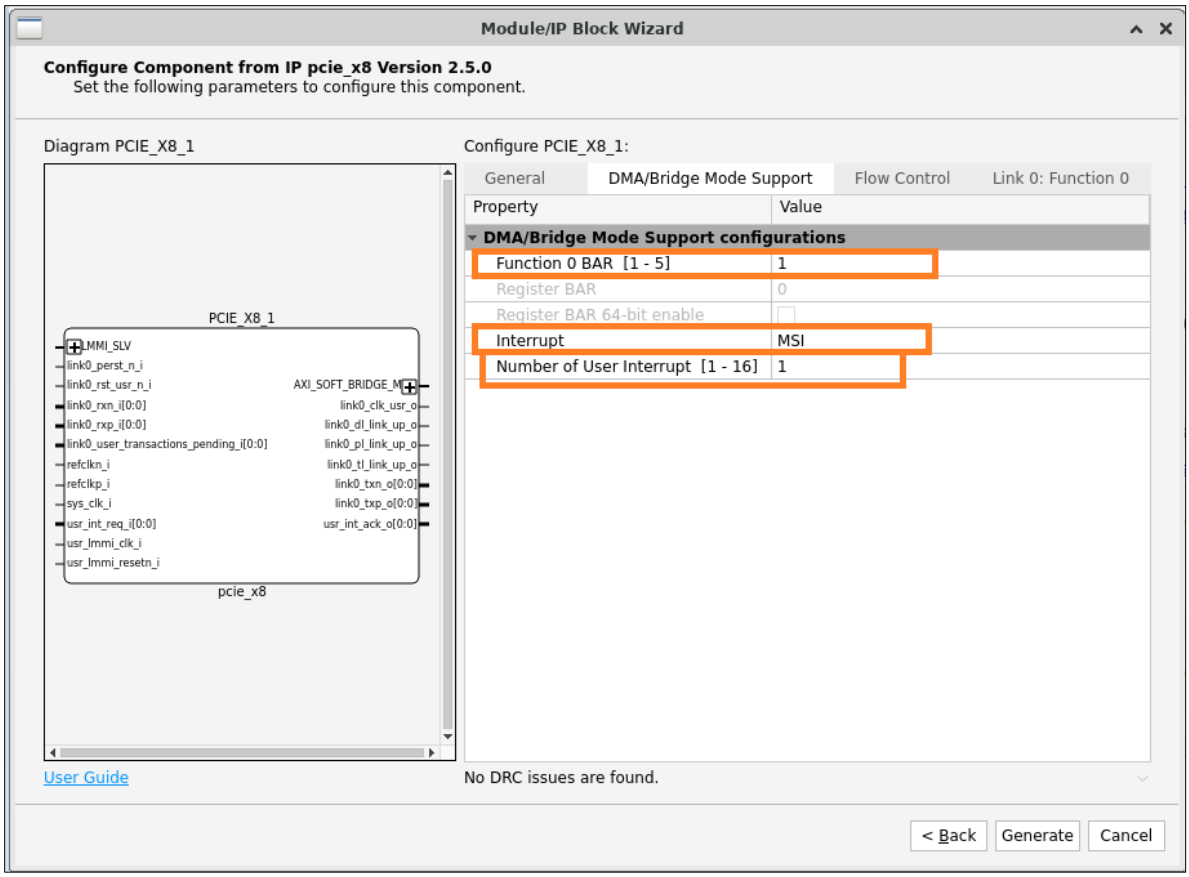


Figure 2.44. Bridge Mode Enablement (DMA/Bridge Mode Support Tab)

In the Radiant user interface, the Bridge Mode is enabled when *Bridge Mode* is selected in *Configuration Mode* drop-down menu in *General* tab (see Figure 2.43). Bridge Mode interface type (AXI-MM or AXI-Lite) is configured by *Bridge Interface Type* in *General* tab.

The Bridge Mode BAR number is configured by *Function 0 BAR* in the *DMA/Bridge Mode Support* tab (see Figure 2.43). The available options are BAR 1 to BAR5. BAR0 is reserved for Bridge Mode registers (refer to the Bridge Mode Register section). The Bridge Mode BAR size is configured in *Link 0: Function 0* tab. DMA interrupt can be MSI or MSI-X, configurable in *DMA/ Bridge Mode Support* tab.

With MSI, the IP supports up to 16 user interrupts. With MSI-X, the IP supports up to 64 user interrupts. The total number of user interrupts is configured by *Number of User Interrupt* in the *DMA/Bridge Mode Support* tab (see Figure 2.44). Refer to the User Interrupts section for more details.

User Interrupts

The PCIe IP supports up to 16 user interrupts and 64 user interrupts for MSI and MSI-X, respectively. The number of user interrupts is configured by the Radiant user interface.

Each user interrupt has a pair of request and acknowledgement pins at the IP interface, such as usr_intr_req_i and usr_intr_ack_o, respectively. When user logic asserts any usr_intr_req_i, The PCIe IP transmits MSI/MSI-X TLP to PCIe link partner. If more than one usr_intr_req_i are asserted, an arbiter in the IP arbitrates these requests with round-robin arbitration scheme. The interrupt vector (MSI vector) associated with a user interrupt is configured through the USR_INT_VEC_P* registers. Refer to the Bridge Mode Register for more details.

The following are the requirements of usr_intr_req_i[NUM-1:0] and usr_intr_ack_o[NUM-1:0]:

1. Each user interrupt has their corresponding usr_intr_req_i[] and usr_intr_ack_o[] pins at the IP interface. The bit number of these signals refers to the user number.
2. User application logic must assert usr_intr_req_i[] when it requires PCIe IP to send interrupt (MSI or MSI-X) to the host.

3. usr_intr_req_i[] and usr_intr_ack_o[] must comply to full handshake relationship.
4. usr_intr_req_i[] can only assert when usr_intr_ack_o[] is 0.
5. usr_intr_req_i[] can only de-assert when usr_intr_ack_o[] is 1.
6. usr_intr_ack_o[] = 1 means the corresponding user request (through usr_intr_req_i[] assertion) is translated to MSI/MSI-X transmission.

The violation of the rule between usr_intr_req_i[] and usr_intr_ack_o[] may cause undefined behavior. Figure 2.45 shows the example waveform:

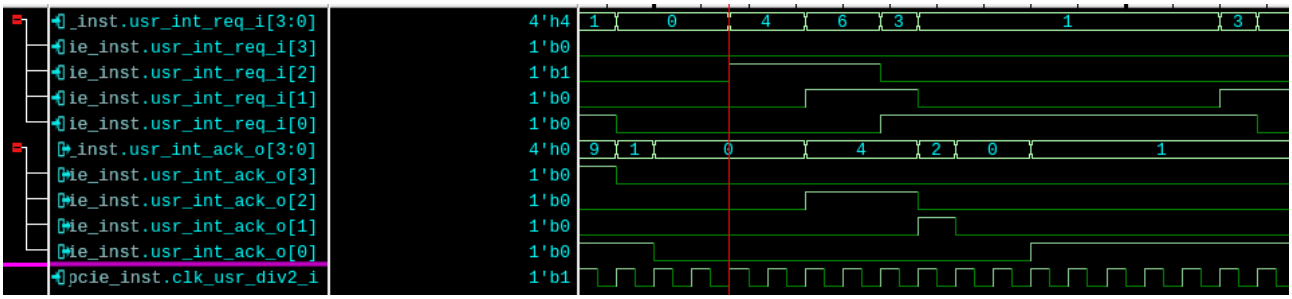


Figure 2.45. User Interrupt Pins Example Waveform

The requests from user logic are translated to MSI or MSI-X TLP. Refer to MSI Bridge Mode and MSI-X Bridge Mode sections for details.

MSI Bridge Mode

The Message Signaled Interrupts (MSI) is supported by PCIe IP as an approach to interrupt Host when user logic asserts specific pins to the IP (see Bridge Mode Register section).

A full 32 vectors are advertised, but only up to 16 interrupt requesters (the number of requesters is configurable) are supported. You can configure user-interrupt-pin-to-MSI-vector mapping through writing to USR_INT_VEC_P* registers. Refer to Bridge Mode Register section for detail. Per-Vector Masking and Extended Message Data are not supported.

MSI Advertised Capabilities

This section specifies MSI-related capabilities and the advertised values. These values are not configurable.

Table 2.54. MSI Advertised Capabilities

Registers	Advertised Value	Remark
Multiple Message Capable	3'b101	32 vectors are advertised despite there are only 16 user interrupt pins.
64-bit address capable	1'b1	64-bit addressing is supported.
Per-Vector Masking Capable	1'b0	Per-Vector Masking is not supported.
Extended Message Data Capable	1'b0	Extended Message Data is not supported.

MSI-X Bridge Mode

Similar to MSI, the Message Signaled Interrupts – Extended (MSI-X) is supported by PCIe IP as an approach to interrupt Host when user logic asserts designated pins to the IP (see User Interrupts section).

Up to 64 interrupt requesters (the number of requesters is configurable) are supported. Each requester has a corresponding MSI-X table entry specified by PCIe specification, which means a total of 64 table entries are supported by the IP.

The MSI-X Table entry associated with user interrupts is as below:

Table 2.55. MSI-X Table Offsets

Offset	MSI-X Table Entries
0x8000	User0 Msg Address
0x8004	User0 Msg Upper Address

Offset	MSI-X Table Entries
0x8008	User0 Msg Data
0x800C	User0 Vector Control
0x8010	User1 Msg Address
0x8014	User1 Msg Upper Address
0x8018	User1 Msg Data
0x801C	User1 Vector Control
…	…
0x83F0	User63 Msg Address
0x83F4	User63 Msg Upper Address
0x83F8	User63 Msg Data
0x83FC	User63 Vector Control

Table 2.56 shows the PBA (Pending Bit Array) table entry associated with user interrupts.

Table 2.56. MSI-X PBA Offsets

Offset	PBA Table Entries
0xC000	user_int_pb[63:0]

The MSI-X table and PBA table are residing in IP registers at BAR0 (memory space). The offsets in the table above refer to BAR0 offset.

Per PCIe specification requirement, MSI-X must support Function Masking and Per-Vector Masking (PVM).

When the Function Mask bit in the PCIe Capability register is set to 1, if the IP is to send the MSI-X TLP (due to user interrupt request), this TLP is not sent and instead the IP asserts the corresponding pending bit at offset 0xC000.

Once the Function Mask bit is cleared to 0, the IP screens through the pending bits, for the bit(s) that is 1, and its corresponding per-vector mask bit is 0, the IP generates and transmits MSI-X TLP to the host. Afterwards, the IP clears the corresponding Pending Bit(s) to 0. It is possible to have multiple MSI-X TLPs transmitted by the IP after Function Mask bit is cleared by the SW.

When per-vector Mask bit register in Vector Control register is set to 1, if the IP was to send an MSI-X TLP (due to user interrupts) of this vector, this TLP is not sent, and instead the IP asserts the corresponding Pending Bit at offset 0xC000.

Once the mask bit is cleared to 0 (and Function Mask bit is also 0), the IP refers to the corresponding vector's pending bits. If the pending bit is 1, the IP generates and transmits MSI-X TLP to the host. Afterwards, the IP clears the corresponding pending bit to 0.

Steering Tag (optional per PCIe specification) is not supported, therefore all other bits in the vector control register (other than bit 0) are reserved.

MSI-X Advertised Capabilities

This section specifies MSI-X-related capabilities and the advertised values. These values are not configurable.

Table 2.57. MSI-X Advertised Capabilities

Registers	Advertised Value	Remark
Table Size	10'h0 – 10'h63	The advertised number of table entries depends on the number of user interrupts.
Table BIR	3'b000	BAR 0
Table Offset	29'b1000_0000_0000_0000	MSI-X Table is from BAR 0 Offset 0x8000
PBA BIR	3'b000	BAR 0
PBA Offset	29'b1100_0000_0000_0000	PBA Table is from BAR 0 Offset 0xC000

Bridge Mode Register

The Bridge Mode registers are accessible by the Host when the received MWr or MRd TLP targets BAR 0. The register access size is limited to maximum 1 DW per TLP.

The Access Types of each register are defined in Table 2.58.

Table 2.58. Access Types

Access Type	Behavior on Read Access	Behavior on Write Access
RO	Returns register value	Ignores write access
WO	Returns 0	Updates register value
RW	Returns register value	Updates register value
RW1C	Returns register value	Writing 1'b1 on register bit clears the bit to 1'b0. Writing 1'b0 on register bit is ignored.
WHC	Returns register value	Only Write to 1'b1 when the register is 1'b0 takes effect.
RC	Returns register value Clear the register to 0 after read.	Ignores write access
RSVD	Returns 0	Ignores write access

Table 2.59. USR_MSI_VEC_P1 (0x040C)

Field	Name	Access	Width	Default	Description
31:29	RSVD	RO	3	0	Reserved
28:24	USR3_MSI_VEC	RW	5	0	User 3 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 4.
23:21	RSVD	RO	3	0	Reserved
20:16	USR2_MSI_VEC	RW	5	0	User 2 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 3.
15:13	RSVD	RO	3	0	Reserved
12:8	USR1_MSI_VEC	RW	5	0	User 1 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 2.
7:5	RSVD	RO	3	0	Reserved
4:0	USR0_MSI_VEC	RW	5	0	User 0 MSI Vector When MSI is selected:

Field	Name	Access	Width	Default	Description
					5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.
31:29	RSVD	RO	3	0	Reserved
28:24	USR3_MSI_VEC	RW	5	0	User 3 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 4.
23:21	RSVD	RO	3	0	Reserved
20:16	USR2_MSI_VEC	RW	5	0	User 2 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 3.
15:13	RSVD	RO	3	0	Reserved
12:8	USR1_MSI_VEC	RW	5	0	User 1 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 2.
7:5	RSVD	RO	3	0	Reserved
4:0	USR0_MSI_VEC	RW	5	0	User 0 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on.

Table 2.60. USR_MSI_VEC_P2 (0x0410)

Field	Name	Access	Width	Default	Description
31:29	RSVD	RO	3	0	Reserved
28:24	USR7_MSI_VEC	RW	5	0	User 7 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 8.

Field	Name	Access	Width	Default	Description
23:21	RSVD	RO	3	0	Reserved
20:16	USR6_MSI_VEC	RW	5	0	User 6 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 7.
15:13	RSVD	RO	3	0	Reserved
12:8	USR5_MSI_VEC	RW	5	0	User 5 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 6.
7:5	RSVD	RO	3	0	Reserved
4:0	USR4_MSI_VEC	RW	5	0	User 4 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 5.

Table 2.61. USR_MSI_VEC_P3 (0x0414)

Field	Name	Access	Width	Default	Description
31:29	RSVD	RO	3	0	Reserved
28:24	USR11_MSI_VEC	RW	5	0	User 11 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 12.
23:21	RSVD	RO	3	0	Reserved
20:16	USR10_MSI_VEC	RW	5	0	User 10 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 11.

Field	Name	Access	Width	Default	Description
15:13	RSVD	RO	3	0	Reserved
12:8	USR9_MSI_VEC	RW	5	0	User 9 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 10.
7:5	RSVD	RO	3	0	Reserved
4:0	USR8_MSI_VEC	RW	5	0	User 8 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 9.
31:29	RSVD	RO	3	0	Reserved
28:24	USR11_MSI_VEC	RW	5	0	User 11 MSI Vector When MSI is selected: 5'd0: MSIVector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 12.
23:21	RSVD	RO	3	0	Reserved
20:16	USR10_MSI_VEC	RW	5	0	User 10 MSI Vector When MSI is selected: 5'd0: MSIVector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 11.
15:13	RSVD	RO	3	0	Reserved
12:8	USR9_MSI_VEC	RW	5	0	User 9 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 10.
7:5	RSVD	RO	3	0	Reserved
4:0	USR8_MSI_VEC	RW	5	0	User 8 MSI Vector When MSI is selected: 5'd0: MSI Vector 0

Field	Name	Access	Width	Default	Description
					5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 9.

Table 2.62. USR_MSI_VEC_P4 (0x0418)

Field	Name	Access	Width	Default	Description
31:29	RSVD	RO	3	0	Reserved
28:24	USR15_MSI_VEC	RW	5	0	User 15 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 16.
23:21	RSVD	RO	3	0	Reserved
20:16	USR4_MSI_VEC	RW	5	0	User 14 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 15.
15:13	RSVD	RO	3	0	Reserved
12:8	USR13_MSI_VEC	RW	5	0	User 13 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 14.
7:5	RSVD	RO	3	0	Reserved
4:0	USR12_MSI_VEC	RW	5	0	User 12 MSI Vector When MSI is selected: 5'd0: MSI Vector 0 5'd1: MSI Vector 1 And so on. This register is RO if the number of user interrupt is set to less than 13.

Table 2.63. USR0_MSIX_TABLE (0x8000)

Field	Name	Access	Width	Default	Description
127:97	RSVD	RW	31	0	Reserved. By default, the value of these bits must be 0. RO when MSI-X is disabled or when User Interrupt is disabled.
96	USR0_MASK_BIT	RW	1	1	User 0 Mask Bit When this bit is Set, the Function is prohibited from sending a message using this MSI-X Table entry for user interrupt 0. However, any other MSI-X Table entries programmed with the same vector will still be capable of sending an equivalent message unless they are also masked. Default value of this bit is 1b (entry is masked) RO when MSI-X is disabled or when User Interrupt is disabled.
95:64	USR0_MSG_DATA	RW	32	0	User 0 Message Data Message Data of MSI-X caused by User Interrupt 0. RO when MSI-X is disabled or when User Interrupt is disabled.
63:32	USR0_MSG_UPPER_ADDR	RW	32	0	User 0 Message Upper Address Upper 32-bit address of MSI-X caused by User Interrupt 0. If this field is zero, 32-bit address messages are used. If this field is non-zero, 64-bit address messages are used. RO when MSI-X is disabled or when User Interrupt is disabled.
31:0	USR0_MSG_ADDR	RW	32	0	User 0 Message Address Lower 32-bit address of MSI-X caused by User Interrupt 0. For proper DWORD alignment, software must always write zeroes to LSB two bits; otherwise the result is undefined. RO when MSI-X is disabled or when User Interrupt is disabled.

Table 2.64. USR1_MSIX_TABLE (0x8010)

Field	Name	Access	Width	Default	Description
127:97	RSVD	RW	31	0	Reserved. By default, the value of these bits must be 0. RO when MSI-X is disabled or when User Interrupt is disabled.
96	USR1_MASK_BIT	RW	1	1	User 1 Mask Bit

Field	Name	Access	Width	Default	Description
					When this bit is Set, the Function is prohibited from sending a message using this MSI-X Table entry for user interrupt 1. However, any other MSI-X Table entries programmed with the same vector is still capable of sending an equivalent message unless they are also masked. Default value of this bit is 1b (entry is masked) RO when MSI-X is disabled or when User Interrupt is disabled.
95:64	USR1_MSG_DATA	RW	32	0	User 1 Message Data Message Data of MSI-X caused by User Interrupt 1. RO when MSI-X is disabled or when User Interrupt is disabled.
63:32	USR1_MSG_UPPER_ADDR	RW	32	0	User 1 Message Upper Address Upper 32-bit address of MSI-X caused by User Interrupt 1. If this field is zero, 32-bit address messages are used. If this field is non-zero, 64-bit address messages are used. RO when MSI-X is disabled or when User Interrupt is disabled.
31:0	USR1_MSG_ADDR	RW	32	0	User 1 Message Address Lower 32-bit address of MSI-X caused by User Interrupt 1. For proper DWORD alignment, software must always write zeroes to LSB two bits; otherwise the result is undefined. RO when MSI-X is disabled or when User Interrupt is disabled.

All Other User Interrupt MSI-X Table (0x8020 to 0x83FF)

For User Interrupt 2 to 63, the register definition is similar to USR0_MSIX_TABLE and USR1_MSIX_TABLE, with only user interrupt number differences. Offset wise, they are packed in incremental order after USR1_MSIX_TABLE, as each of them are 4DW size, same with USR0_MSIX_TABLE and USR1_MSIX_TABLE.

Table 2.65. PBA_TABLE (0xC000)

Field	Name	Access	Width	Default	Description
63:0	USR_INT_PB	RO	64	0	User Interrupt Pending Bit Each bit is associated to User Interrupt Pending Bits, where LSB refer to User Interrupt 0, in incremental order, up to User Interrupt 31. For each Pending Bit that is Set, the Function has a pending message for the associated MSI-X Table entry, which is suppressed by Function Mask bit and/or the corresponding Mask bit in Vector Control field in MSI-X Table.

2.14.2. Register Interface Conversion

2.14.2.1. AXI-Lite Interface

This interface is available if the register interface type selected in the IP generation user interface is *AXI-Lite.* For more information on the AXI-Lite signal description, refer to the AXI-Lite Configuration Interface section. The address of the registers to be accessed is available in the Register Description section.

The Core Configuration and Status Registers (CSR) are made accessible to the user design through the AXI-Lite. The registers that are configured through AXI-Lite are as follows:

- Simulation registers:
  - Register address – 0X4_2000
    This register is used to reduce the ltssm ts_1 and timeouts to fasten the simulation when asserted as *1*.
  - Register address – 0X4_3000
    This register is used to reduce the Power Management State Machine timeouts to fasten the simulation when asserted as *1*.
  - Register address – 0X4_4000
    This register is used to reduce the timeouts to fasten the simulation when asserted as *1*.

2.15. Multi-Protocol Support

The Lattice Avant-AT-G supports multi-protocol starting Radiant 2024.1. Quad merging is handled by the Lattice Radiant software.

PCIe Bifurcation								
	PHY Lanes							
Mode	0	1	2	3	4	5	6	7
DMA+PCIe Mode	x8							
PCIe Mode 1x8	x8							
PCIe Mode 1x4	x4				any protocol			
PCIe Mode 1x2	x2		NA	NA	any protocol			
PCIe Mode 1x1	x1	NA	NA	NA	any protocol			
	Quad0				Quad1			
PCIe Mode 1x2 + PIPE Direct (2x1/1x2)	x2		(PIPE x1/x2)					
PCIe Mode 1x1 + PIPE Direct (2x1/1x2	x1	NA	(PIPE x1/x2)					

Figure 2.46. PCIe Bifurcation

3. IP Parameter Description

The PCIe Endpoint Core attributes are configurable through the IP Catalog's Module/IP wizard of the Lattice Radiant Software. Refer to Table 3.1 for the description of each attribute.

3.1. General

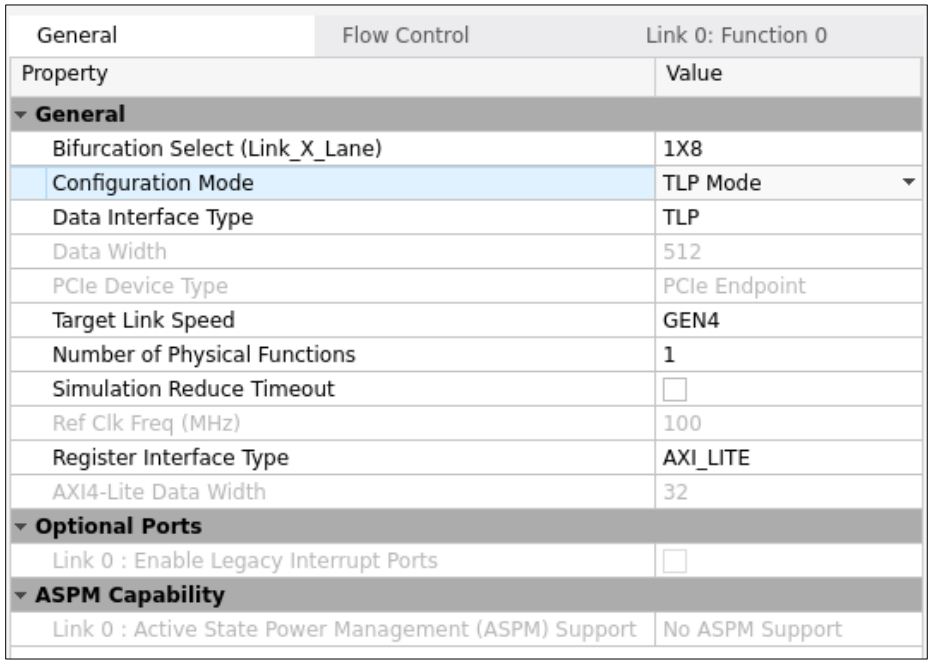


Figure 3.1. Attributes in the General Tab

Table 3.1. General Tab Attributes Description

Attribute	Selectable Values	Description	Parameter
Bifurcation Select	1×1 1×2 1×4 1×8	Configures the number of Links and Lanes.	PCIE_BIFUR_SEL 2=1×1, 1=1×2, 5=1×4, 6=1×8
Configuration Mode	"TLP Mode" "DMA Only Mode" "Bridge Mode" "DMA with Bridge Mode" "Harden DMA Mode"	To select configuration mode. Available modes are TLP Mode, DMA Only, Bridge Mode, DMA with Bridge Mode, and Harden DMA Mode.	—
Data Interface Type	"TLP" "AXI_STREAM" "AXI_MM"	Available option per mode: • "TLP Mode": "TLP" and "AXI_STREAM" • "DMA Only Mode": "AXI_MM". • "DMA with Bridge Mode": "AXI_MM". • "Harden DMA Mode": "AXI_MM".	—
Bridge Interface Type	"AXI_MM" "AXI_LITE"	Only selectable in Bridge Mode and DMA with Bridge Mode.	—

Attribute	Selectable Values	Description	Parameter
		It configures Bridge Mode interface type.	
Data Width	64, 128, 256, 512	Display only	—
PCIe Device Type	PCIe Endpoint	Display only. PCIe IP core supports only PCIe Endpoints	LINK0_DEVICE_TYPE = “PCIe Endpoint”
Target Link Speed	Gen1 Gen2 Gen3 Gen4	• Initial value of Target Link Speed Configuration Register. • Determines the maximum initial link speed which can be reached during initial training. • Must be set to the lesser of the maximum speed supported by the core and the maximum speed at which you desire the core to operate.	LINK0_FTL_INITIAL_TARGET_LINK_SPEEED = {0,1,2,3}
Number of Physical Functions	1–8	Set the number of enabled functions.	LINK0_NUM_FUNCTIONS = {1..8}
Simulation Reduce Timeout	Checked Unchecked	• Must be checked for simulation run. • Otherwise, leave it unchecked.	—
Ref Clk Freq (MHz)	100	• Display only. Ref Clk from PCIe slot	—
Register Interface Type	AXI_Lite, LMMI	• Available if default interface is not selected and mapping of register to subordinate data interface is not selected. • The selected interface replaces the native Lattice Memory Mapped Interface (LMMI) of the hard IP by adding a soft logic bridge.	USR_CFG_IF_TYPE = {"LMMI", " AXI4_LITE"}
AXI4-Lite Data Width	32	• Available if Register Interface Type is AXI_LITE • Display only. AXI4-Lite data width	—
LMMI Data Width	16	• Available if Register Interface Type is LMMI • Display only. Lattice Memory Mapped Interface (LMMI) data width	—

3.2. Optional Port

Figure 3.2. Attributes in the Optional Port Tab

Table 3.2. Optional Port Attributes

Link [k] (k == 0 - 1) Optional Ports			
Attribute	**Selectable Values**	**Description**	**Parameter**
Link [k] Enable Legacy Interrupt Ports	Checked Unchecked	• Available if legacy interrupt is enabled. • Set to add the Legacy interrupt ports.	LINK[k]_MAIN_CTRL_4_EN_PORT_MGMT_INTERRUPT_LEG = {0,1}

3.3. ASPM Capability

Figure 3.3. Attributes in the ASPM Capability Tab

Note: ASPM will be available in future release.

3.4. DMA/Bridge Mode Support

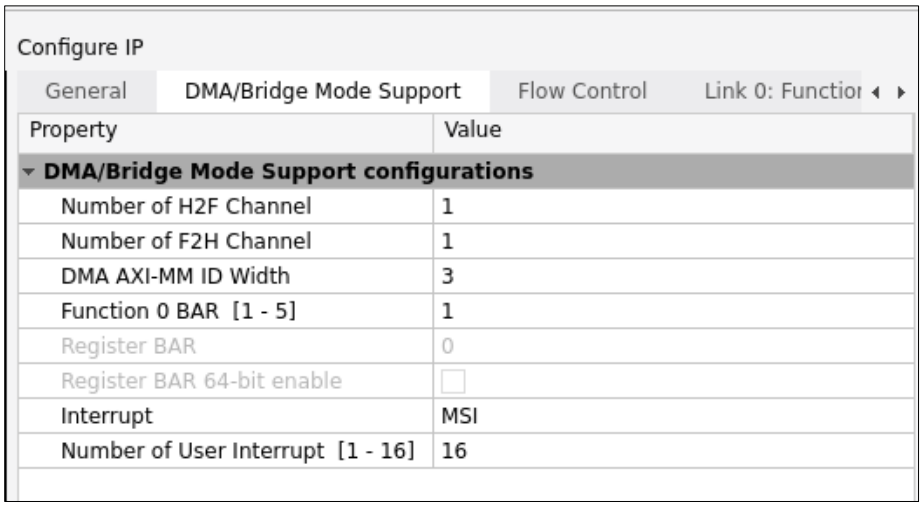


Figure 3.4. Attributes in the DMA/Bridge Mode Support Tab

Table 3.3. DMA/ Bridge Mode Support Attributes

Attribute	Selectable Values	Description	Parameter
Number of H2F Channel	0–1	• Number of H2F channel. • Maximum 1 channel is supported in the current release.	NUM_H2F_CHAN = 1
Number of F2H Channel	0–1	• Number of F2H channel. • Maximum 1 channel is supported in the current release.	NUM_F2H_CHAN = 1
DMA AXI-MM ID Width	Integer	• Data width for AXI-MM interface's AWID, BID, ARID, and RID. • Should use a value not greater than 8.	DMA_AXI_ID_WIDTH = 3
Function 0 BAR	1–5	• PCIe Endpoint BAR that is allocated for Bridge Mode.	
Register BAR	0	• BAR mapping for DMA/ Bridge register. • Only 0 is supported in the current release.	—
Register BAR 64-bit enable	Checked, Unchecked	• To select if DMA/ Bridge register BAR is 32 bits or 64 bits. • Only unchecked (32-bit) is supported in the current release.	—
Interrupt	MSI, MSI-X	• DMA/Bridge Interrupt mode. • With DMA: Only MSI is supported in the current release. • Bridge only: MSI or MSI-X.	—
Number of User Interrupt	MSI: 1–16 MSI-X: 1–64	Refer to User Interrupts section for more details.	—

3.5. Flow Control Update

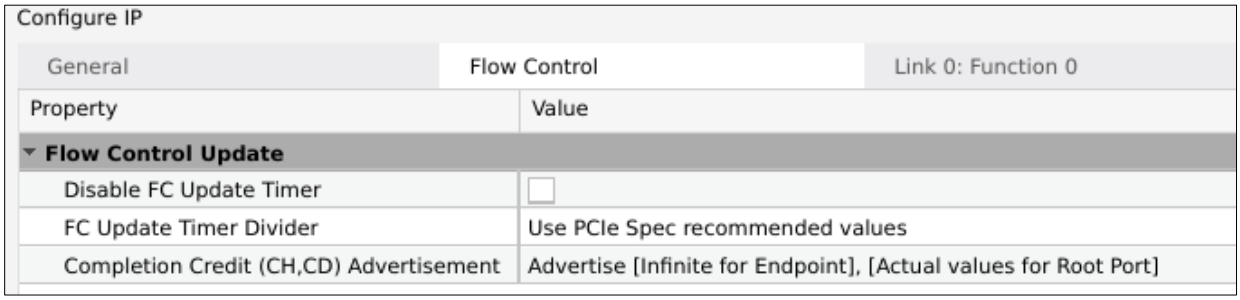


Figure 3.5. Attributes in the Flow Control Update Tab

Table 3.4. Flow Control Attributes

Link [k] (k == 0 - 1) Flow Control Update			
Attribute	**Selectable Values**	**Description**	**Parameter**
Link [k] Disable FC Update Timer	Checked Unchecked	• Set to disable FC Update Timer (that is, schedule a FC Update on Every Consumed RX TLP • Otherwise, schedule FC Updates in accordance with PCIe Specification recommended values)	LINK[k]_PTL_RX_CTRL_FC_UPDATE_TIMER_DISABLE = {0,1}
Link [k] FC Update Timer Divider	Use PCIe Spec recommended values, Divide by 2, Divide by 4, Divide by 8	Select the FC Update frequency of the Receive Buffer when FC update timer is enabled.	LINK[k]_PTL_RX_CTRL_FC_UPDATE_TIMER_DIV = {0,1,2,3}
Link [k] Completion Credit Advertisement	Advertise Infinite for Endpoint and Actual for Root Port, Advertise Actual, Advertise Infinite	Select the completion credit advertisement behavior.	LINK[k]_PTL_RX_CTRL_ADV_CH_CD_SEL = {0,1,2}

3.6. Function

3.6.1. Configuration

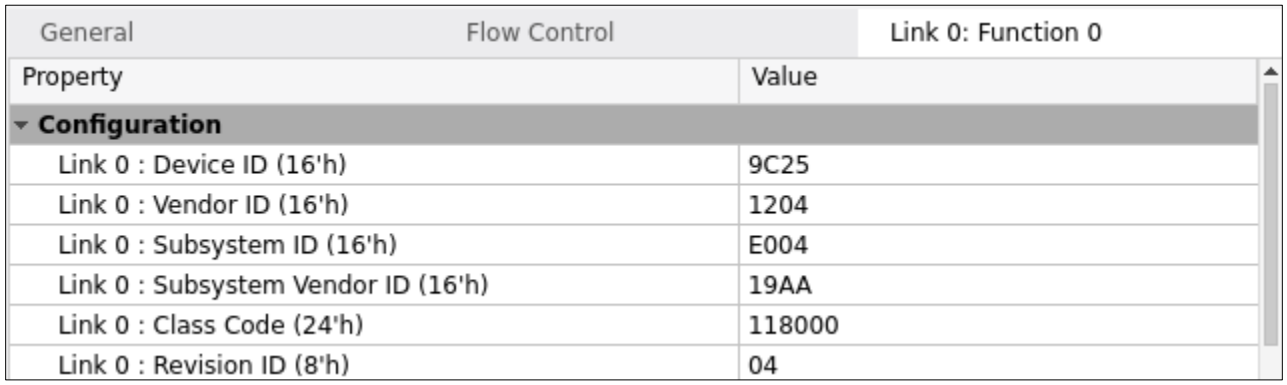


Figure 3.6. Attributes in Function Configuration Tab

Table 3.5. Function Configuration Tab Attributes

Configuration			
Attribute	**Selectable Values**	**Description**	**Parameter**
Link [k] Device ID	(Hex) 0000 – FFFF	Value returned when the Device ID Configuration Register is read.	LINK[k]_FTL_ID1_DEVICE_ID = {16'h0000 – 16'hFFFF}
Link [k] Vendor ID	(Hex) 0000 – FFFF	Value returned when the Vendor ID Configuration Register is read.	LINK[k]_FTL_ID1_VENDOR_ID = {16'h0000 – 16'hFFFF}
Link [k] Subsystem ID	(Hex) 0000 – FFFF	Value returned when the Subsystem ID Configuration Register is read.	LINK[k]_FTL_ID2_SUBSYSTEM_ID = {16'h0000 – 16'hFFFF}

Configuration			
Attribute	**Selectable Values**	**Description**	**Parameter**
Link [k] Subsystem Vendor ID	(Hex) 0000 – FFFF	Value returned when the Subsystem Vendor ID Configuration Register is read.	LINK[k]_FTL_ID2_SUBSYSTEM_VENDOR_ID = {16'h0000 – 16'hFFFF}
Link [k] Class Code	(Hex) 00000 – FFFFF	Value returned when the Class Code Configuration Register is read.	LINK[k]_FTL_ID3_CLASS_CODE = {16'h0000 – 16'hFFFF}
Link [k] Revision ID	(Hex) 00 – FF	Value returned when the Revision ID Configuration Register is read.	LINK[k]_FTL_ID3_REVISION_ID = {8'h00 – 8'hFF}

3.6.2. Base Address Register (BAR) [0 to 5]

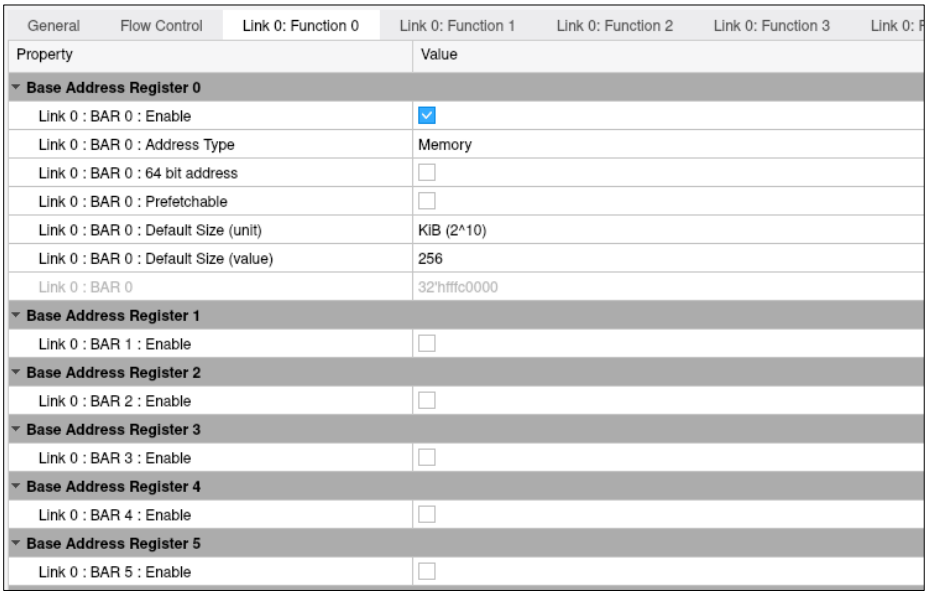


Figure 3.7. Attributes in BAR Tab

Table 3.6. BAR Tab Attributes

Link [k] (k == 0 - 1) Base Address Register n (n == 0 - 5)			
Attribute	**Selectable Values**	**Description**	**Parameter**
Link [k] BAR n – Enable	Checked Unchecked	Set to enable the BAR.	—
Link [k] BAR n – Address Type	Memory, I/O	Select if the BAR is for Memory or I/O space.	—
Link [k] BAR n – 64-bit Address	Checked Unchecked	• Applicable for memory space only. • Set to use 64-bit address. Note that BAR n and BAR n+1 are used for the 64-bit address.	—
Link [k] BAR n – Prefetchable	Checked Unchecked	• Applicable for memory space only. • Set to identify the memory address as prefetchable.	—
Link [k] BAR n – Default Size (unit)	Bytes, kB (210), MB (220), GB (230),	Select the size of Memory space.[1]	—

Link [k] (k == 0 - 1) Base Address Register n (n == 0 - 5)			
Attribute	**Selectable Values**	**Description**	**Parameter**
	TB (240), PB (250), EB (260),		
Link [k] BAR n – Default Size (value)	(Power of 2) 32 bits Memory Space: 16 bytes – 2 GB 64 bits Memory Space: 64 bits: 4 GB – 8 EB 32 bits I/O Space: 2 Bytes – 256 Bytes	Select the size of Memory or I/O space.[1]	—
Link [k] BAR n	32 bits: FFFF_FFF0 - 1000_0000 64 bits: FFFF_FFFF_0000_0000 - 1000_0000_0000_0000	Display Only	Function 0: LINK[k]_FTL_BAR0_CFG … … LINK[k]_FTL_BAR5_CFG Function m: LINK[k]_FTL_MF1_BAR0_CFG … … LINK[k]_FTL_MF[m]_BAR[n]_CFG

3.6.3. Legacy Interrupt

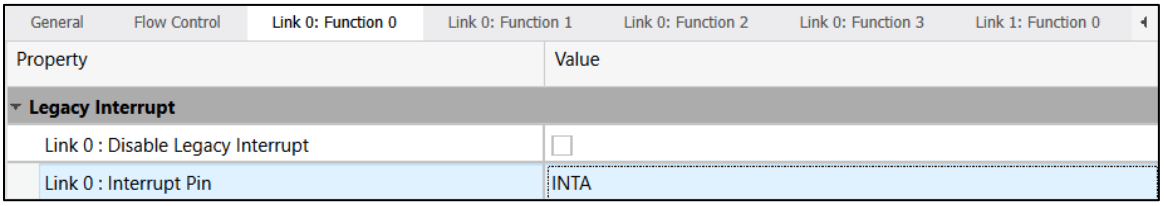


Figure 3.8. Attributes in Legacy Interrupt

Table 3.7. Legacy Interrupt Attribute Descriptions

Link [k] (k == 0 - 1) Legacy Interrupt			
Attributes	**Value**	**Description**	**Parameters**
Link [k] Disable Legacy Interrupt	Checked Unchecked	• RTL always supports legacy interrupt. • The current attribute is only used for port activation.	LINK[k]_FTL_INTERRUPT_DISABLE = {0,1}
Link [k] Interrupt Pin	INT A, INT B, INT C, INT D	Select which legacy interrupt pin is used.	LINK[k]_FTL_INTERRUPT_PIN = {0,1,2,3}

3.6.4. MSI Capability

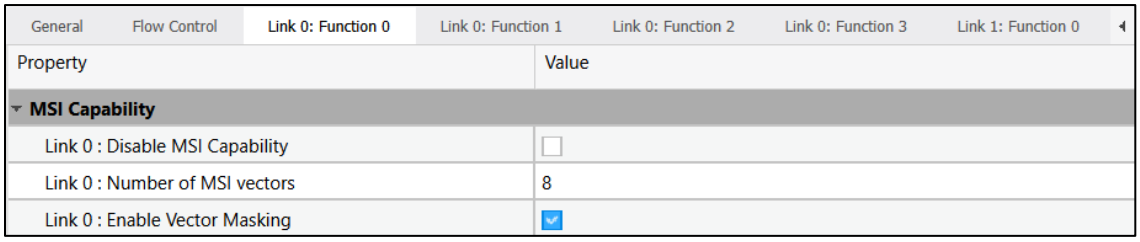


Figure 3.9. Attributes in MSI Capability

Table 3.8. MSI Capability Attributes

Link [k] (k == 0 - 1) MSI Capability			
Attributes	**Value**	**Description**	**Parameters**
Link [k] Disable MSI Capability	Checked Unchecked	Set to disable the MSI Capability.	LINK[k]_FTL_MSI_CAP_DISABLE = {0,1}
Link [k] Number of MSI vectors	1 – 32	Set the number of requested MSI vectors.	LINK[k]_FTL_MSI_CAP_MULT_MESSAGE_CAPABLE = {0,1,2,3,4,5}
Link [k] Enable Vector Masking	Checked Unchecked	Set to enable vector masking capability.	LINK[k]_FTL_MSI_CAP_VEC_MASK_CAPABLE = {0,1}

3.6.5. MSI-X Capability

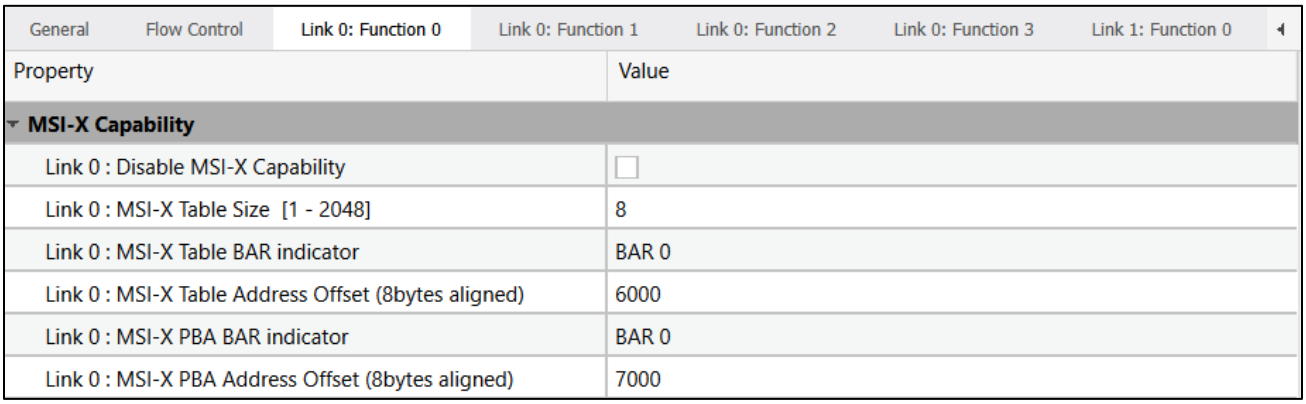


Figure 3.10. Attributes in MSI-X Capability

Table 3.9. MSI-X Capability Attributes

Link [k] (k == 0 - 1) MSI-X Capability			
Attributes	**Value**	**Description**	**Parameters**
Link [k] Disable MSI-X Capability	Checked Unchecked	Set to disable the MSI-X Capability.	LINK[k]_FTL_MSIX_CAP_DISABLE = {0,1}
Link [k] MSI-X Table Size	1–2048	Set the number of requested MSI-X vectors.	LINK[k]_FTL_MSIX_CAP_TABLE_SIZE = {0 – 2047}
Link [k] MSI-X Table BAR indicator	BAR 0, BAR 1, BAR 2, BAR 3, BAR 4, BAR 5	• Select which Base Address register. • Located beginning at 10h in Configuration Space, is used to map the MSI-X Table into Memory Space.	LINK[k]_FTL_MSIX_TABLE_BIR = {0,1,2,3,4,5}
Link [k] MSI-X Table Address Offset	(Hex, 8 bytes aligned) 0000_0000 – FFFF_FFF8	Set the byte address offset (8 bytes aligned), within the BAR selected by MSI-X Table BAR indicator, at which the MSI-X table begins.	LINK[k]_FTL_MSIX_TABLE_OFFSET = {29'h00000000 – 29'h1FFFFFFF}
Link [k] MSI-X PBA BAR indicator	BAR 0, BAR 1, BAR 2, BAR 3, BAR 4, BAR 5	Select which Base Address register, located beginning at 10h in Configuration Space, is used to map the MSI-X PBA into Memory Space.	LINK[k]_FTL_MSIX_PBA_BIR = {0,1,2,3,4,5}
Link [k] MSI-X PBA Address Offset	(Hex, 8 bytes aligned) 0000_0000 – FFFF_FFF8	Set the byte address offset (8 bytes aligned), within the BAR selected by MSI-X PBA BAR indicator, at which the MSI-X PBA begins.	LINK[k]_FTL_MSIX_PBA_OFFSET = {29'h00000000 – 29'h1FFFFFFF}

3.6.6. Device Serial Number Capability

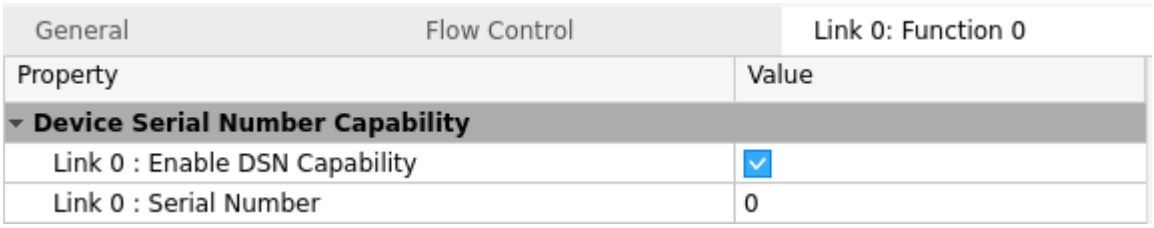


Figure 3.11. Attributes in Device Serial Number Capability

Table 3.10. Device Serial Number Capability Attributes

Link [k] (k == 0 - 1) Device Serial Number Capability			
Attributes	**Value**	**Description**	**Parameters**
Link [k] Enable DSN Capability	Checked Unchecked	Set to enable the Device Serial Number capability.	LINK[k]_FTL_DSN_CAP_ENABLE = {0,1}
Link [k] Serial Number	(Hex) 0000_0000_0000_0000 – FFFF_FFFF_FFFF_FFFF	Set the device serial number.	LINK[k]_FTL_DSN_SERIAL_NUMBER

3.6.7. PCI Express Capability

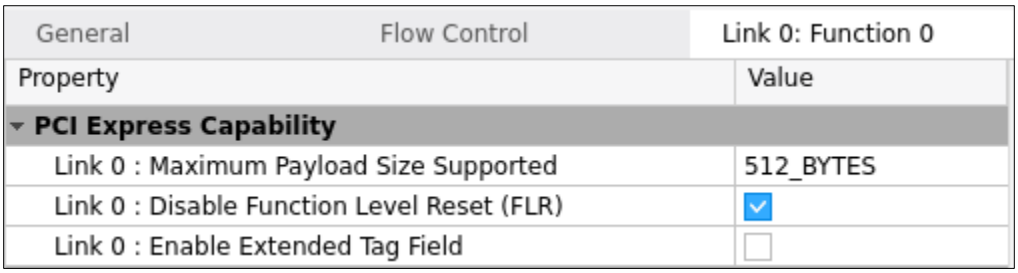


Figure 3.12. Attributes in PCIe Capability

Table 3.11. PCIe Capability Attributes

Link [k] (k == 0 - 1) PCIe Device Capability			
Attributes	**Value**	**Description**	**Parameters**
Link [k] Maximum Payload Size Supported	128 Bytes, 256 Bytes, 512 Bytes	Select the maximum payload size supported.	LINK[k]_FTL_PCIE_DEV_CAP_MAX_PAYLOAD_SIZE_SUPPORTED = {0,1,2}
Link [k] Disable Function Level Reset (FLR)	Checked Unchecked	Set to disable Function Level Reset capability.	LINK[k]_FTL_PCIE_DEV_CAP_DISABLE_FLR_CAPABILITY = {0,1}
Link [k] Enable Extended Tag Field	Checked Unchecked	Set to enable Extended Tag Field (8-bit tag field).	LINK[k]_FTL_PCIE_DEV_CAP_EXTENDED_TAG_FIELD_SUPPORTED = {0,1}

3.6.8. Advance Error Reporting Capability

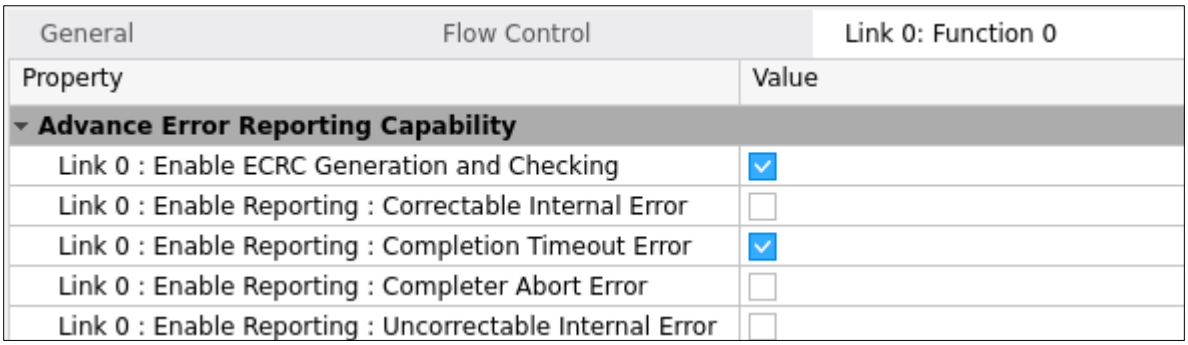


Figure 3.13. Attributes in Advance Error Reporting Capability

Table 3.12. Advance Error Reporting Capability Attributes

Link [k] (k == 0 - 1) Advance Error Reporting Capability			
Attributes	**Value**	**Description**	**Parameters**
Link [k] Enable ECRC Generation and Checking	Checked Unchecked	Set to enable ECRC generation and checking.	LINK[k]_FTL_AER_CAP_ECRC_GEN_CHK_CAPABLE = {0,1}
Link [k] Enable Reporting: Correctable Internal Error	Checked Unchecked	Set to enable reporting of correctable internal error.	LINK[k]_FTL_AER_CAP_EN_CORR_INTERNAL_ERROR = {0,1}
Link [k] Enable Reporting: Completion Timeout Error	Checked Unchecked	Set to enable reporting of completion timeout error.	LINK[k]_FTL_AER_CAP_EN_COMPLETION_TIMEOUT = {0,1}

Link [k] (k == 0 - 1) Advance Error Reporting Capability			
Attributes	**Value**	**Description**	**Parameters**
Link [k] Enable Reporting: Completer Abort Error	Checked Unchecked	Set to enable reporting of completer abort error.	LINK[k]_FTL_AER_CAP_EN_COMPLETER_ABORT = {0,1}
Link [k] Enable Reporting: Uncorrectable Internal Error	Checked Unchecked	Set to enable reporting of uncorrectable internal error.	LINK[k]_FTL_AER_CAP_EN_UCORR_INTERNAL_ERROR = {0,1}

3.6.9. ATS Capability

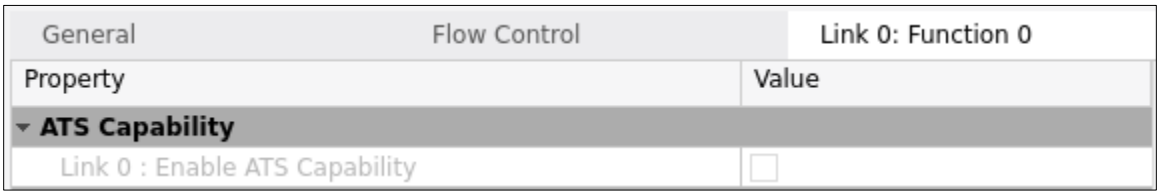


Figure 3.14. Attributes in ATS Capability

Table 3.13. ATS Capability Attribute Description

Link [k] (k == 0 - 1) ATS Capability			
Attributes	**Value**	**Description**	**Parameters**
Link [k] Enable ATS Capability	Checked Unchecked	Set to enable the ATS Capability.	LINK[k]_FTL_ATS_CAP_ENABLE = {0,1}

3.6.10. Atomic OP Capability

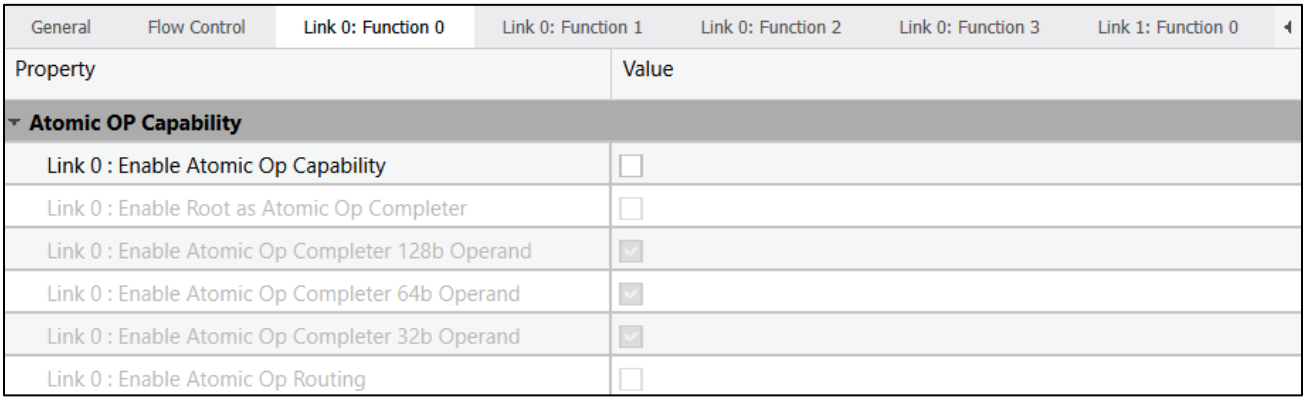


Figure 3.15. Attributes in Atomic OP Capability

Table 3.14. Atomic OP capability Attributes

Link [k] (k == 0 - 1) Atomic OP Capability			
Attributes	**Value**	**Description**	**Parameters**
Link [k] Enable Atomic Op Capability	Checked Unchecked	Set to enable Atomic Operations Capability.	LINK[k]_FTL_ATOMIC_OP_CAP_ENABLE = {0,1}
Link [k] Enable Root as Atomic Op Completer	Checked Unchecked	Set to enable Root as Atomic OP Completer.	LINK[k]_ FTL_ATOMIC_OP_CAP_RP_COMPLETER_ENABLE = {0,1}

Link [k] (k == 0 - 1) Atomic OP Capability			
Link [k] Enable Atomic Op Completer 128b Operand	Checked Unchecked	Set to support Atomic Op 128b operand.	LINK[k]_FTL_ATOMIC_OP_CAP_COMPLETER_128_SUPPORTED = {0,1}
Link [k] Enable Atomic Op Completer 64b Operand	Checked Unchecked	Set to support Atomic Op 64b operand	LINK[k]_FTL_ATOMIC_OP_CAP_COMPLETER_64_SUPPORTED = {0,1}
Link [k] Enable Atomic Op Completer 32b Operand	Checked Unchecked	Set to support Atomic Op 32b operand.	LINK[k]_FTL_ATOMIC_OP_CAP_COMPLETER_32_SUPPORTED = {0,1}
Link [k] Enable Atomic Op Completer Routing	Checked Unchecked	Set to support Atomic Op routing.	LINK[k]_ FTL_ATOMIC_OP_CAP_ROUTING_SUPPORTED = {0,1}

3.6.11. Latency Tolerance Reporting Capability

Figure 3.16. Attributes in Latency Tolerance Reporting Capability

Table 3.15. Latency Tolerance Reporting Capability Attributes

Link [k] (k == 0 - 1) Latency Tolerance Reporting Capability			
Attributes	**Value**	**Description**	**Parameters**
Link [k] Enable LTR Capability	Checked Unchecked	Set to enable the Latency Tolerance Reporting capability.	LINK[k]_FTL_LTR_CAP_ENABLE = {0,1}

3.6.12. Power Budgeting Capability

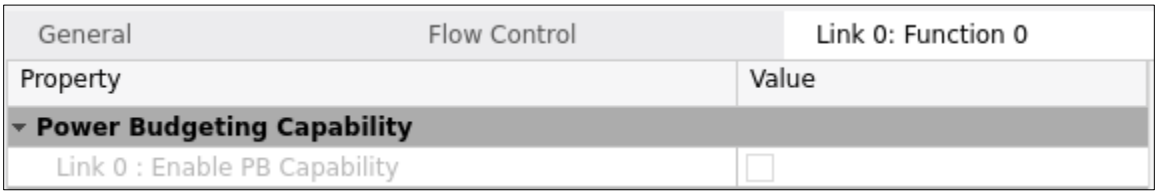


Figure 3.17. Attributes in Power Budgeting Capability

Table 3.16. Power Budgeting Capability Attributes

Link [k] (k == 0 - 1) Power Budgeting Capability			
Attributes	**Value**	**Description**	**Parameters**
Link [k] Enable PB Capability	Checked Unchecked	Set to enable the Power Budgeting capability.	LINK[k]_FTL_PWR_BUDGET_CAP_ENABLE = {0,1}

3.6.13. Dynamic Power Allocation Capability

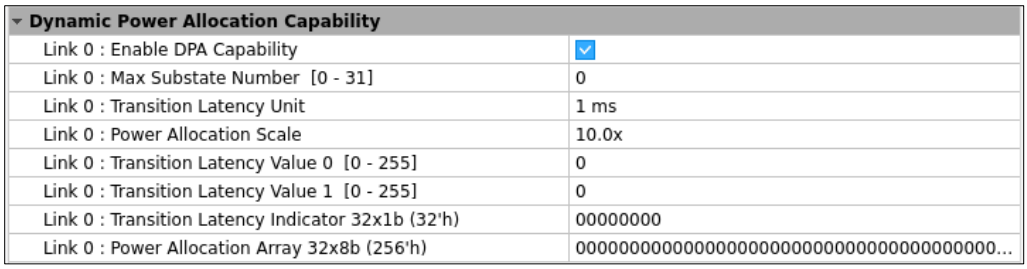


Figure 3.18. Attributes in Dynamic Allocation Capability

Table 3.17. Dynamic Allocation capability Attributes

Link [k] (k == 0 - 1) Dynamic Power Allocation Capability			
Attributes	**Value**	**Description**	**Parameters**
Link [k] Enable DPA Capability	Checked Unchecked	Set to enable the Dynamic Power Allocation capability.	LINK[k]_FTL_DPA_CAP_ENABLE = {0,1}
Link [k] Max Substate Number	0 – 31	• Specifies the maximum substate number. • Substates from [substate_max:0] are supported. • For example, substate_max==0 indicates support for 1 substate.	LINK[k]_FTL_DPA_CAP_SUBSTATE_MAX = {0 - 31}
Link [k] Transition Latency Unit	1 ms, 10 ms, 100 ms	Specifies Transition Latency Unit.	LINK[k]_FTL_DPA_CAP_TLUNIT = {0 - 2}
Link [k] Power Allocation Scale	10.0x, 1.0x, 0.1x, 0.01x	Specifies Power Allocation Scale.	LINK[k]_FTL_DPA_CAP_PAS = {0 - 3}
Link [k] Transition Latency Value 0	0 – 255	Specifies Transition Latency Value 0.	LINK[k]_FTL_DPA_CAP_XLCY0 = {0 - 3}
Link [k] Transition Latency Value 1	0 – 255	Specifies Transition Latency Value 1.	LINK[k]_FTL_DPA_CAP_XLCY1 = {0 - 3}
Link [k] Transition Latency Indicator 32x1b	(Hex) 00000000 – FFFFFFFF	• Specifies which Transition Latency Value applies to each substate. • Each bit corresponds to a substate.	LINK[k]_ FTL_DPA_XLCY_INDICATOR = {32'h00000000 – 32'hFFFFFFFF}
Link [k] Power Allocation Array 32x8b	(Hex) {32{00}} – {32{FF}}	• Substate Power Allocation Array. • Each entry is 8b value.	LINK[k]_ FTL_DPA_ALLOC_ARRAY = { {32{8'h00}} – {32{8'hFF}}}

Table 3.18. Function 1-7 Tab

Link [k] (k == 0 - 1) Function n (n == 1 – 7)			
Configuration			
Disable Function	Checked Unchecked	Available if the number of physical functions enabled is set to greater than 1. Set to disable the function. Parameter: LINK[k]_FTL_MF1_FUNCTION_DISABLE = {0,1} LINK[k]_FTL_MF2_FUNCTION_DISABLE = {0,1} LINK[k]_FTL_MF3_FUNCTION_DISABLE = {0,1}	Parameter: LINK[k]_FTL_MF1_FUNCTION_DISABLE = {0,1} LINK[k]_FTL_MF2_FUNCTION_DISABLE = {0,1} LINK[k]_FTL_MF3_FUNCTION_DISABLE = {0,1}
Device ID	Refer to Function section.		—
Vendor ID			
Subsystem ID			
Subsystem Vendor ID			—
Class Code			—
Revision ID			—
Base Address Register (see the Lattice PCIe x8 Core Configuration user interface in Function section)			—
Legacy Interrupt (see the Lattice PCIe x8 Core Configuration user interface in Function section)			—
MSI Capability (see the Lattice PCIe x8 Core Configuration user interface in Function section)			—
MSI-X Capability (see the Lattice PCIe x8 Core Configuration user interface in Function section)			—
Device Serial Number Capability (see the Lattice PCIe x8 Core Configuration user interface in Function section)			—

4. Signal Description

The Lattice PCIe x8 IP Core Ports for Link 0 is defined in the following sub-sections.

4.1. Clock and Reset Interface

Table 4.1. Clock and Reset Ports

Port	Type	Description
link0_perst_n_i	Input	• PCI Express Fundamental Reset • Active-low asynchronous assert, synchronous de-assert reset to the Link Layer, PHY, and Soft Logic blocks. On link0_perst_n_i and link0_rst_usr_n_i deassertion, the core starts in the Detect.Quiet Link Training and Status State Machine (LTSSM) state with the Physical Layer down and Data Link Layer down. • Normally, the PHY registers are automatically configured based on the user interface settings you selected. However, it is still possible for the user to modify and configure the registers through the selected register interface. link0_perst_n_i must remain asserted while the PHY registers are being configured.
link0_rst_usr_n_i	Input	This interface is non-operational. You need to tie this port to 1'b1.

sys_clk_i	Input	• User Clock Domain Input Clock
		• It is recommended to use the following minimum clock frequency to achieve the maximum throughput with respect to link data rate:
		  • Gen 4 (16.0G) – 250 MHz
		  • Gen 3 (8.0G) – 125 MHz
		  • Gen 2 (5.0G) – 62.5 MHz
		  • Gen 1 (2.5G) – 31.25 MHz
		• All ports of the corresponding PCIE link are synchronized to this clock.

link0_clk_usr_o	Output	• Optional Clock source.
		• This clock is a divided clock from PIPE clock. The clock frequency is as follows for different link data rate:
		  • Gen 4 (16.0G) – 250 MHz
		  • Gen 3 (8.0G) – 125 MHz
		  • Gen 2 (5.0G) – 125 MHz
		  • Gen 1 (2.5G) - 62.5 MHz

link0_pl_link_up_o	Output	• Physical Layer Link Up Status – (1) Up and (0) Down • link0_pl_link_up_o is used as an active-low, synchronous reset for the core's Data Link Layer. • You are not expected to use this port except for status since user design does not interface directly with the Data Link Layer.
link0_dl_link_up_o	Output	• Data Link Layer Link Up Status – (1) Up and (0) Down • link0_dl_link_up_o is used as an active-low, synchronous reset for the Transaction Layer and also indicates when TLPs can be successfully transmitted across the link. • For Endpoint-only applications, users must use link0_dl_link_up_o as a synchronous reset for their RTL interfacing to the core's Transaction Layer interfaces.
link0_tl_link_up_o	Input	• Transaction Layer Link Up Status – (1) Up and (0) Down • link0_tl_up_o is an active-low, synchronous reset to the core's upper transaction layer. • A Downstream Port's (Root Port) Configuration Registers, and the RTL which enables them to be accessed from the User Transmit and Receive Interfaces, are reset by link0_tl_up_o so that the PCIe Configuration Registers may be

Port	Type	Description
		read even when the link is down. • Downstream Port applications must use link0_tl_up_o as a synchronous reset for their RTL interfacing to the core's Transaction Layer interfaces. When link0_tl_up_o == 1 the core's PCIe Configuration Registers may be read. When link0_dl_link_up_o == 1, accessing the PCIe devices behind the DS Port may be attempted. When link0_dl_link_up_o == 0 accesses to the PCIe devices behind the DS Port is not possible; if accesses are attempted, they fail with error status.

4.2. PHY Interface

The Link Layer is used in conjunction with a third-party PCI Express PHY to implement a complete Lattice PCIe x8 Core PCI Express implementation. The PHY implements the high-speed serial and analog functions required to support PCI Express while the Link Layer implements most of the digital logic as well as the higher levels of the PCI Express protocol.

The PIPE PHY Interface that connects the Link Layer and PHY is not shown here since the interface is only internal and is not visible to users.

Table 4.2. PHY Interface Descriptions

Port	Type	Description

link0_rxp_i	Input	• Differential Receive Serial signal, Rx+.
		• When PCIE Core is only enabled, the PCIE link can be configured as:
		  • x8 lanes – (Lane 0 – 7) or
		  • x4 lanes – (Lane 0 – 3) or
		  • x2 lanes – (Lane 0 – 1) or
		  • x1 lane – (Lane 0)

link0_rxn_i	Input	Differential Receive Serial signal, Rx- See PCIE Hard IP Core Lane assignment above.
link0_txp_o	Output	Differential Transmit Serial signal, Tx+ See PCIE Hard IP Core Lane assignment above.
link0_txn_o	Output	Differential Transmit Serial signal, Tx- See PCIE Hard IP Core Lane assignment above.
refclkp_i	Input	Differential Reference Clock, CLK+ (100 MHz)
refclkn_i	Input	Differential Reference Clock, CLK- (100 MHz)

Note:

1. LANE – range (0, 7)

4.3. Transaction Layer Interface

4.3.1. TLP Transmit Interface

Refer the TLP Transmit Interface section for more information and timing diagrams.

4.3.1.1. TLP Transmit Interface Port Description

Table 4.3. TLP Transmit Interface Ports

Port	Clock Domain	Direction	Description
link0_tx_valid_i	sys_clk_i	Input	Source valid. (1==Valid, 0==Not valid)
link0_tx_ready_o	sys_clk_i	Output	• Destination ready. (1==Ready, 0==Not ready) • A transfer occurs on the transmit interface only when link0_tx_valid_i==link0_tx_ready_o==1.

Port	Clock Domain	Direction	Description
link0_tx_sop_i	sys_clk_i	Input	• Start of packet indicator. • Set == 1 coincident with the first link0_tx_data_i word in each TLP.
link0_tx_eop_i	sys_clk_i	Input	• End of packet indicator. • Set == 1 coincident with the last link0_tx_data_i word in each TLP.
link0_tx_eop_n_i	sys_clk_i	Input	• Nullify packet indicator. • Set == 1 coincident with link0_tx_eop_i == 1 to instruct the core to nullify the current TLP (invert LCRC and use EDB framing) instead of transmitting the TLP normally.

link0_tx_data_i [NUM_LANES × 64-1:0]	sys_clk_i	Input	• TLP data to transfer
			• link0_tx_data_i must be valid from the assertion of link0_tx_sop_i until the TLP is fully consumed with the assertion of link0_tx_eop_i.
			• TLP data must comprise a complete Transaction Layer Packet (TLP) as defined by the PCI Express Specification including the entire 3 or 4 DWORD TLP header, data payload (if present), and optionally a TLP Digest (ECRC). The core adds the necessary STP/END/EDB framing, Sequence Number, LCRC, and add ECRC (if enabled to do so and ECRC is not already present in the transmission) as part of its Data Link Layer functionality.
			• Transmitted TLPs are required to be formulated correctly by PCIe Specification including filling in the Requester/Completer ID, Attributes, Traffic Class, and so on. For Multi-Function, the core uses the Requestor/Completer ID in transmitted TLPs to determine which function's Configuration Registers must be applied to determine the validity of the transmitted TLP.
			• Data width depends on the number of lanes configured for a particular link.
			  • x8 – 512b
			  • x4 – 256b
			  • x2 – 128b
			  • x1 – 64b
			Note: There is a known hardware limitation for the December 2023 release. As a workaround for this issue, application logic must have at least 20 clocks gap from tx_eop_i of a TLP to the tx_sop_i of the next TLP. In addition to performance impact, the problem that may be observed is completion timeout for continuous Memory Read TLP from Host to the FPGA.
			The hardware limitation is resolved in 2024.1.

link0_tx_datap_i [NUM_LANES × 8-1:0]	sys_clk_i	Input	• Parity of associated link0_tx_data_i:
			  • link0_tx_datap_i [i] == ^(link0_tx_data_i[((i+1) ×8)-1:(i×8)])
			  • link0_tx_datap_i must be valid for all bytes in link0_tx_data_i that contain a portion of a TLP (header, payload, and TLP digest (if present)).
			• Parity width changes as data width, 1 parity bit per 8 bits of data.

Note:

1. LINK – values (0); NUM_LANES – range (1,8)

4.3.1.2. TLP Transmit Credit Interface Port Description

Table 4.4. TLP Transmit Credit Interface Ports

Port	Clock Domain	Direction	Description
link0_tx_credit_init_o	sys_clk_i	Output	When the core Transaction Layer for Link[i] is ready to accept TLP transmissions, the core asserts link0_tx_credit_init_o == 1 for one clock cycle and on the same cycle indicates the non-posted TLP Header strorage capacity of the Transmit Buffer on link0_tx_credit_nh_o[11:0]. Users are expected to keep and initialize their NH available transmit credit counters on link0_tx_credit_init_o==1. When a non-posted TLP is pending for transmission within user logic, user logic should check the currently available NH credit count for the associated link and hold the transmission until enough NH credits are available to transmit the TLP. Once the TLP has been committed for transmit, the amount of NH credits required by that TLP are decremented from the NH credit count. As the core forwards transmitted TLPs from the Transmit Buffer and thus makes room for new TLPs, the core asserts link0_tx_credit_return_o==1 for one clock cycle and places the number of NH credits being returned on link0_tx_credit_nh_o[11:0]. In this manner, the user can manage sending only enough non-posted TLPs that the core can hold in its Transmit Buffer. This permits the user to know when non-posted TLPs are blocked and thus send posted and/or completion TLPs instead. This is important for avoiding deadlocks and keeps non-posted TLP blockage from reducing posted and completion throughput. Should the core receive more non-posted TLPs than the core can store in its non-posted TLP transmit storage, the core pauses TLP transmission rather than allow an overflow to occur. Thus, users that do not wish to use the Transmit Credit Interface may ignore this interface provided they are willing to permit blocked non-posted TLPs from also blocking following posted and completion TLPs.
link0_tx_credit_return_o	sys_clk_i	Output	As the core forwards transmitted TLPs from the Transmit Buffer and thus makes room for new TLPs, the core asserts link0_tx_credit_return_o==1 for one clock cycle and places the number of NH credits being returned on link0_tx_credit_nh_o[11:0].
link0_tx_credit_nh_o[11:0]	sys_clk_i	Output	Number of NH credits to return == link0_tx_credit_nh_o[11:0].

4.3.2. TLP Receive Interface

Refer to the TLP Receive Interface section for more information and timing diagrams.

4.3.2.1. TLP Receive Interface Port Descriptions

Table 4.5. TLP Receive Interface Ports

Port	Clock Domain	Direction	Description
link0_rx_valid_o	sys_clk_i	Output	• link0_rx_valid_o == 1 when the other link0_rx_* output ports are valid. Otherwise, value is 0. A data transfer occurs when link0_rx_valid_o == 1 and link0_rx_ready_i == 1.
link0_rx_ready_i	sys_clk_i	Input	• Set link0_rx_ready_i == 1 whenever the user logic is ready to accept received TLP data. A data transfers occur when link0_rx_valid_o == 1 and link0_rx_ready_i == 1. • For maximum throughput, user's design must consume TLPs at the rate that they are presented (link0_rx_ready_i == 1 on all clock cycles).

link0_rx_sel_o[1:0]	sys_clk_i	Output	• Receive TLP type indicator:
			  • 0 == Posted Request (write request)
			  • 1 == Non-Posted Request (request requiring a completion)
			  • 2 == Completion (completion to a previous request)
			  • 3 == Reserved
			• link0_rx_sel_o is useful for steering the TLP to the appropriate processing logic. For example, Posted Requests should be directed to receive write logic while Non-Posted Requests should be directed to receive read logic. Completions should be directed back to the original read request source using the TLP Tag information.
			• link0_rx_sel_o is valid for the entire TLP (from link0_rx_sop_o == 1 through link0_rx_eop_o == 1)

Port	Clock Domain	Direction	Description

link0_rx_cmd_data_o[12:0]	sys_clk_i	Output	• Received TLP Type Indicator
			• link0_rx_cmd_data_o provides information about the received TLP to facilitate user TLP processing. This port has a different meaning in Root Port Modes of operation and Endpoint operation.
			• The Lattice PCIe x8 Core decodes received TLPs to determine their destination. The core passes this information to the Transaction Layer Receive Interface by asserting the appropriate bits in this field.
			• Individual bits of link0_rx_cmd_data_o[12:0] carry the following information:
			  • Bits[12:10] – Traffic Class[2:0] of the TLP
			  • Bit[9] – Completion/Base Address Region indicator
			  • (1) indicates the TLP is a Completion or Message routed by ID
			  • (0) indicates the TLP is a read or write request or a Message routed by address that hit an enabled Base Address Region.
			  • Bits[8:0] are decoded differently based on the value of Bit[9]
			  • Bits[8:0] – Completion or Message routed by ID (Bit[9] == 1)
			  • Bits[8] – Reserved
			  • Bits[7:0] – Tag[7:0] – The Requestor Tag contained in the TLP. Tag[7:0] is used to associate the completion with its original source request. This field is reserved if the TLP is a message rather than a completion.
			  • Bits[8:0] – Read or write request or a Message routed by address that hit an enabled Base Address Region (Bit[9] == 0)
			  • Bit[8] – When (1), the packet is a *write* transaction; when (0), the packet is a *read* transaction.
			  • Bit[7] – When (1), the packet requires one or more Completion transactions as a response; (0) otherwise.
			  • Bit[6] – (1) the TLP hit the Expansion ROM else (0)
			  • Bit[5] – (1) the TLP hit Base Address Region 5 else (0)
			  • Bit[4] – (1) the TLP hit Base Address Region 4 else (0)
			  • Bit[3] – (1) the TLP hit Base Address Region 3 else (0)
			  • Bit[2] – (1) the TLP hit Base Address Region 2 else (0)
			  • Bit[1] – (1) the TLP hit Base Address Region 1 else (0)
			  • Bit[0] – (1) the TLP hit Base Address Region 0 else (0)
			• link0_rx_cmd_data_o is valid for the entire packet (from link0_rx_sop_o == 1 through link0_rx_eop_o == 1).

link0_rx_f_o[7:0]	sys_clk_i	Output	• Function Hit by the Received TLP • link0_rx_f_o indicates which PCIe Function received the TLP. • link0_rx_f_o == 0 indicates Function #0. link0_rx_f_o == 1 indicates Function #1 and so on. **Note:** This signal is only functional in 2024.2 or later. Prior to 2024.2 release, the application logic must decode the address/ BDF of the received TLP and compare against the BAR in PCIe Configuration Registers to know which Physical Function is hit.
link0_rx_sop_o	sys_clk_i	Output	• Start of TLP indicator • link0_rx_sop_o == 1 coincident with the first link0_rx_data_o word in each TLP. Otherwise, value is 0.

Port	Clock Domain	Direction	Description
link0_rx_eop_o	sys_clk_i	Output	• End of TLP indicator. • link0_rx_eop_o == 1 coincident with the last link0_rx_data_o word in each TLP. Otherwise, value is 0.
link0_rx_err_ecrc_o	sys_clk_i	Output	• Received TLP ECRC error indicator • link0_rx_err_ecrc_o == 1 from link0_rx_sop_o to link0_rx_eop_o inclusive for received TLPs which contain a detected ECRC error. Otherwise, value is 0. • link0_rx_err_ecrc_o only reports ECRC errors when ECRC checking is enabled. ECRC checking is enabled by software through the AER Capability. • TLPs with ECRC errors are presented on the Receive Interface in the same format that they are received including the TLP Digest (ECRC). • ECRC errors are serious, uncorrectable errors. The user design must decide how to handle/recover from the error including whether to use the TLP with the error. ECRC errors need for higher level software to correct/handle the error. PCIe does not have a standard mechanism for retransmitting TLPs end to end as it does for a given PCIe link (through the LCRC/Sequence Number and Replay mechanisms).

link0_rx_data_o [NUM_LANES×64-1:0]	sys_clk_i	Output	• Received TLP Data
			• Received TLP data comprises a complete Transaction Layer Packet (TLP) as defined by the PCI Express Specification including the entire TLP header, data payload (if present), and TLP Digest (ECRC, if present). The core strips the packet's STP/END/EDB framing, Sequence Number, and Link CRC (LCRC) prior to the TLP appearing on this interface. The core checks TLP ECRC, when present and when checking is enabled, and can be optionally enabled to remove the ECRC from the TLP.
			• Data width depends on the number of lanes configured for a particular link.
			  • x8 – 512b
			  • x4 – 256b
			  • x2 – 128b
			  • x1 – 64b

link0_rx_datap_o [NUM_LANES×8-1:0]	sys_clk_i	Output	• Received TLP Data Parity
			• Even parity of associated link0_rx_data_o:
			  • link0_rx_datap_o[i] == ^(link0_rx_data_o[((i+1) ×8)-1:(i×8)])
			  • link0_rx_datap_o is valid for all bytes in link0_rx_data_o that contain a portion of a TLP (header, payload (if present), and TLP digest (ECRC, if present). link0_rx_data_p_o is not valid for any trailing, unused bytes in the final link0_rx_data_o word in a TLP.
			• Parity width changes as data width, 1 parity bit per 8 bits of data.

Note:

1. LINK – values (0); NUM_LANES – range (1,8)

4.3.2.2. TLP Receive Credit Interface Port Description

Table 4.6. TLP Receive Credit Interface Ports

Port	Clock Domain	Direction	Description
link0_rx_credit_init_i	sys_clk_i	Input	When the user transaction layer logic is ready to accept non-posted TLP reception, assert the link0_rx_credit_init_i == 1 for one clock cycle and on the same cycle indicates the non-posted TLP header storage capacity of the user design in link0_rx_credit_nh_i[11:0]. You must initialize link0_rx_credit_init_i shortly (within 10s of clocks) after u_tl_link_up for Root Port and shortly after u_dl_link_up for Endpoint. Holding off credit initialization for an extended period can cause received non-posted TLP transactions to timeout in the source component which may be serious errors. The core limits simultaneous outstanding non-posted receive TLPs on the receive interface to ensure no more than the initialized NH credits are simultaneously outstanding to user receive TLP logic. Once the received non-posted TLPs are processed/forwarded such that more room is available to receive new non-posted TLPs, assert link0_rx_credit_return_i==1 for one clock cycle and places the number of NH credits being returned on link0_rx_credit_nh_i[11:0]. In this manner, you can limit the outstanding core receive TLPs to the user design. This permits the core to know when non-posted TLPs are blocked and thus send posted and/or completion TLPs to the user design instead. This is important for avoiding deadlocks and keeps non-posted TLP blockage from reducing posted and completion throughput. If you do not wish to implement flow control of NH credits through this interface, link0_rx_credit_init==1 and link0_rx_credit_nh_inf_i is set to 1 to advertise infinite NH credits. The NH credit flow control is not implemented for links that advertised infinite NH credits.
link0_rx_credit_return_i	sys_clk_i	Input	Once the received non-posted TLPs are processed/forwarded such that more room is available to receive new non-posted TLPs, assert link0_rx_credit_return_i==1 for one clock cycle and places the number of NH credits being returned on link0_rx_credit_nh_i[11:0].
link0_rx_credit_nh_i[11:0]	sys_clk_i	Input	Number of NH credits to return == link0_rx_credit_nh_i[11:0].
link0_rx_credit_nh_inf_i	sys_clk_i	Input	Infinite NH Credits link0_rx_credit_nh_inf_i: 1==Do not limit TLP reception due to NH credits 0==Limit simultaneously outstanding NH credits to the value of link0_rx_credit_nh_i[11:0] when link0_rx_credit_init was 1.

4.4. Lattice Memory Mapped Interface (LMMI)

The Lattice PCIe x8 IP Core implements a bus for configuring core options and obtaining core status. The Core Configuration and Status Registers (CSR) are made accessible to the user design through the Lattice Memory Mapped interface (LMMI).

Due to some known issue, you are advised to use AXI-Lite interface for PCIe x8 IP Core configuration registers access.

Table 4.7. Lattice Memory Mapped Interface Ports

Port	Clock Domain	Direction	Description
usr_lmmi_clk_i	usr_lmmi_clk_i	Input	LMMI Clock (100 MHz max).
usr_lmmi_resetn_i	usr_lmmi_clk_i	Input	Active low, asynchronous assert, synchronous de-assert reset **Note:** Once usr_lmmi_resetn_i de-asserted, you must wait for 10 or more usr_lmmi_clk_i clocks before accessing LMMI interface. This is to allow reset to be properly propagated through the PCIe controller.

usr_lmmi_offset_i [23:0]	usr_lmmi_clk_i	Input	• Register address. See Register Description for more details on address mapping.
			• usr_lmmi_offset_i [23:22] – Not used. Drive 0
			• usr_lmmi_offset_i[21:20] – Quad identifier
			• 2'b00 – Quad 0
			• 2'b01 – Quad 1
			• 2'b10 – Quad 2 (Future release)
			• 2'b11 – Quad 3 (Future release)
			• usr_lmmi_offset_i [19] – Not used. Drive 0
			• usr_lmmi_offset_i [18:16] – Core Register Block Identifier
			• PCIE Configuration Space Registers:
			  • 3'b000 – 3'b001 – Hard PCIE x8 Core PF and VF registers
			• PCIE Core Registers:
			  • 3'b100 – Hard PCIE x8 Core registers
			• MPP Registers:
			  • 3'b101 – MPPHY, External MPLLA, External MPLLB, Aggregate Control, CMU, and HRC registers
			• PMA Registers:
			  • 3'b110 – 3'b111 – PMA Lane, Raw PCS Lane, Raw PCS Always-on Logic, PMA Extra Space, Raw PCS Extra Space, Raw PCS Always-on Logic Extra Space, PMA ROM and PMA RAM Memory access registers
			• For word-aligned register space:
			  • usr_lmmi_offset_i [15:1] – Register offset
			  • usr_lmmi_offset_i [0] – reserved (tie to 0).

usr_lmmi_request_i	usr_lmmi_clk_i	Input	Start Transaction (1==Active; 0==Otherwise) A transaction is started when usr_lmmi_request_i==usr_lmmi_ready_o==1. When usr_lmmi_request_i==usr_lmmi_ready_o==1, usr_lmmi_wr_rdn_i, and usr_lmmi_offset_i must be valid and describe the transaction to execute; if the transaction is a write as indicated by usr_lmmi_wr_rdn_i==1, usr_lmmi_wdata_i must also be valid.
usr_lmmi_wr_rdn_i	usr_lmmi_clk_i	Input	Direction (1==Write, 0==Read)
usr_lmmi_wdata_i [15:0]	usr_lmmi_clk_i	Input	Write data
usr_lmmi_rdata_o [15:0]	usr_lmmi_clk_i	Output	Read data
usr_lmmi_ready_o	usr_lmmi_clk_i	Output	Target is ready to start a new transaction. (1==Ready; 0==Not ready)
usr_lmmi_rdata_valid_o	usr_lmmi_clk_i	Output	Read transaction is complete and usr_lmmi_rdata_o contains valid data (1==Valid; 0==Otherwise).

4.5. Power Management Interface

The Lattice PCIe x8 IP Core supports optional capabilities such as Dynamic Power Allocation, Latency Tolerance Reporting, and Power Budgeting.

Note: This is available in future release.

Table 4.8. Transaction Pending Ports

Port	Clock Domain	Direction	Description
link[LINK]_user_transactions_pending_i [NUM_FUNCTIONS-1:0]	sys_clk_i	Input	• Set to 1 when you have an outstanding (not yet completed) non-posted requests else set to 0. • The value of this port is reflected in the PCIe Configuration Register: PCIe Status – Transactions Pending.

4.6. AXI-Lite Configuration Interface

The Lattice PCIe x8 Core provides an option to use AXI-Lite interface as an alternative to LMMI for register access.

The clock supplied to soft IP's user configuration path must be synchronous to LMMI clock in Hard-IP. If AXI-Lite requires a separate clock domain, the clock crossing from AXI-Lite clock to LMMI clock must be done externally to the soft IP.

Table 4.9. AXI-Lite Configuration Interface Ports

Port	Clock Domain	Type	Function
usr_lmmi_clk_i	usr_lmmi_clk_i	Input	Clock for AXI-Lite register interface logic (100 MHz max).
usr_lmmi_resetn_i	usr_lmmi_clk_i	Input	Reset for AXI-Lite register interface logic. Active low, asynchronous assert, synchronous de-assert reset **Note:** Once usr_lmmi_resetn_i de-asserted, you must wait for 10 or more usr_lmmi_clk_i clocks before accessing AXI-Lite interface. This is to allow reset to be properly propagated through the PCIe controller.
Write Address Channel			
usr0_axil_awvalid_i	usr_lmmi_clk_i	Input	Write address valid
usr0_axil_awaddr_i [23:0]	usr_lmmi_clk_i	Input	Write address
usr0_axil_awready_o	usr_lmmi_clk_i	Output	Write address ready
Write Data Channel			
usr0_axil_wvalid_i	usr_lmmi_clk_i	Input	Write valid
usr0_axil_wdata_i [31:0]	usr_lmmi_clk_i	Input	Write data
usr0_axil_wstrb_i [3:0]	usr_lmmi_clk_i	Input	Write strobes
usr0_axil_wready_o	usr_lmmi_clk_i	Output	Write ready
Write Response Channel			
usr0_axil_bready_i	usr_lmmi_clk_i	Input	Response ready
usr0_axil_bvalid_o	usr_lmmi_clk_i	Output	Write response valid
usr0_axil_bresp_o[1:0]	usr_lmmi_clk_i	Output	Write response
Read Address Channel			
usr0_axil_arvalid_i	usr_lmmi_clk_i	Input	Read address valid
usr0_axil_araddr_i [23:0]	usr_lmmi_clk_i	Input	Read address
usr0_axil_arready_o	usr_lmmi_clk_i	Output	Read address ready
Read Data Channel			
usr0_axil_rready_i	usr_lmmi_clk_i	Input	Read ready

Port	Clock Domain	Type	Function
usr0_axil_rvalid_o	usr_lmmi_clk_i	Output	Read valid
usr0_axil_rdata_o [31:0]	usr_lmmi_clk_i	Output	Read data
usr0_axil_rresp_o [1:0]	usr_lmmi_clk_i	Output	Read response

4.7. AXI-Stream Data Interface

This interface is available when Data Interface Type is configured to AXI_STREAM in the *Data Interface Type* when *Configuration Mode* is *TLP Mode*.

4.7.1. AXI-Stream Transmitter Interface Port Descriptions

Table 4.10. AXI-Stream Transmitter Interface Ports

Port	Clock Domain	Direction	Description
m0_tready_i	sys_clk_i	Input	Destination ready. 1==Ready, 0==Not ready. A transfer occurs when m_tvalid_o==m_tready_i==1.
m0_tvalid_o	sys_clk_i	Output	Source valid 1==Valid 0==Not valid.
m0_tdata_o [NUM_LANES*64-1:0]	sys_clk_i	Output	For Link 0, data width depends on bifurcation option selected. • 1×8 – 512b • 1×4 – 256b • 1×2 – 128b • 1×1 – 64b

m0_tstrb_o [NUM_LANES*8-1:0]	sys_clk_i	Output	• Byte qualifier that indicates whether the content of the associated byte of m_tdata_o is processed as a data byte or a position byte.
			• For Link 0, tstrb width depends on bifurcation option selected.
			  • 1×8 – 512b. The value is 64'hFFFFFFFFFFFFFFFF.
			  • 1×4 – 256b. The value is 32'hFFFFFFFF.
			  • 1×2– 128b. The value is 16'hFFFF.
			  • 1×1– 64b. The value is 8'hFF.

m0_tkeep_o [NUM_LANES*8-1:0]	sys_clk_i	Output	• Byte qualifier that indicates whether the content of the associated byte of m_tdata_o is processed as part of the data stream.
			• Associated bytes that have the m_tkeep_o byte qualifier deasserted are null bytes and can be removed from the data stream.
			• For Link 0, tkeep width depends on bifurcation option selected.
			  • 1×8 – 512b. The value is 64'hFFFFFFFFFFFFFFFF.
			  • 1×4 – 256b. The value is 32'hFFFFFFFF.
			  • 1×2 – 128b. The value is 16'hFFFF.
			  • 1×1 – 64b. The value is 8'hFF.

m0_tlast_o	sys_clk_i	Output	End of TLP indicator. m_tlast_o == 1 coincident with the last m_tdata_o word in each TLP. Otherwise, 0.

Port	Clock Domain	Direction	Description
m0_tid_o [7:0]	sys_clk_i	Output	Data stream identifier that indicates different streams of data. m0_tid_o[2:0] has the BAR number information when rx_cmd_data[9] = 0. Otherwise, 0 when rx_cmd_data[9] = 1(completion). • m0_tid_o[3] = link0_rx_err_par • m0_tid_o[6:4] = link0_rx_cmd_data[12:10] • m0_tid_o[7] = link0_rx_err_ecrc

m0_tdest_o [9:0]	sys_clk_i	Output	• m_tdest_o provides routing information for the data stream.
			• Bits [9:2] – Function Hit by the Received TLP
			• Bits [1:0] – Receive TLP type indicator:
			  • 0 == Posted Request (write request)
			  • 1 == Non-Posted Request (request requiring a completion)
			  • 2 == Completion (completion to a previous request)
			Note: Bits [9:0] are only functional in 2024.1 release or later. Prior to 2024.1 release, the application logic must decode the address/BDF of the received TLP and compare against the BAR in PCIe Configuration Registers to know which Physical Function is hit.

Note:

1. NUM_LANES – range (1 ,8)

4.7.2. AXI-Stream Receiver Interface Port Descriptions

Table 4.11. AXI-Stream Receiver Interface Ports

Port	Clock Domain	Direction	Description
s0_tvalid_i	sys_clk_i	Input	Source valid 1==Valid 0==Not valid.

s0_tdata_i [NUM_LANES*64-1:0]	sys_clk_i	Input	• TLP data to transfer
			• For Link 0, data width depends on bifurcation option selected.
			  • 1×8 – 512b
			  • 1×4 – 256b
			  • 1×2 –128b
			  • 1×1 – 64b
			Note: There is a known hardware limitation in the December 2023 release. As workaround for this issue, application logic must have at least 20 clocks gap from tx_eop_i of a TLP to the tx_sop_i of the next TLP. In addition to performance impact, another problem to look out for is the completion timeout for continuous Memory Read TLP from Host to the FPGA. This hardware limitation is resolved in the 2024.1 release.

s0_tstrb_i [NUM_LANES*8-1:0]	sys_clk_i	Input	• Byte qualifier that indicates whether the content of the associated byte of m_tdata_o is processed as a data byte or a position byte.
			• For Link 0, tstrb width depends on the bifurcation option selected.
			  • 1×8 – 512b. The value is 64'hFFFFFFFFFFFFFFFF.
			  • 1×4 – 256b. The value is 32'hFFFFFFFF.
			  • 1×2 – 128b. The value is 16'hFFFF.
			  • 1×1 – 64b. The value is 8'hFF.

Port	Clock Domain	Direction	Description

s0_tkeep_i [NUM_LANES*8-1:0]	sys_clk_i	Input	• Byte qualifier that indicates whether the content of the associated byte of m_tdata_o is processed as part of the data stream.
			• Associated bytes that have the m_tkeep_o byte qualifier deasserted are null bytes and can be removed from the data stream.
			• For Link 0, tkeep width depends on the bifurcation option selected.
			  • 1×8 – 512b. The value is 64'hFFFFFFFFFFFFFFFF.
			  • 1×4 – 256b. The value is 32'hFFFFFFFF.
			  • 1×2 – 128b. The value is 16'hFFFF.
			  • 1×1 – 64b. The value is 8'hFF.

s0_tlast_i	sys_clk_i	Input	End of packet indicator. Set == 1 coincident with the last s[LINK]_tdata_i word in each TLP.
s0_tid_i [7:0]	sys_clk_i	Input	Unused. Set to 8'h0.
s0_tdest_i [9:0]	sys_clk_i	Input	Unused. Set to 10'h0.
s0_tready_o	sys_clk_i	Output	Destination ready. 1==Ready 0==Not ready A transfer occurs when s_tvalid_i==s_tready_o==1.

Note:

1. NUM_LANES – range (1, 8)

4.8. AXI Data Interface (Hardened DMA)

This interface is available when *Configuration Mode* is in *Hardened DMA Mode*.

Table 4.12. DMA AXI Manager Write Interface

Port	Clock Domain	Direction	Description
m0_axi_awid_o [3:0]	sys_clk_i	Output	This signal is the identification tag for the write address group of signals.
m0_axi_awaddr_o [63:0]	sys_clk_i	Output	The write address gives the address of the first transfer in a write burst transaction.
m0_axi_awlen_o [3:0]	sys_clk_i	Output	The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
m0_axi_awsize_o [2:0]	sys_clk_i	Output	This signal indicates the size of each transfer in the burst.
m0_axi_awprot_o [2:0]	sys_clk_i	Output	This signal is used only for Security. Only m0_axi_awprot [1] is used; [2],[0] are unused and set to 0. • m0_axi_awprot [1] = 0: Secure • m0_axi_awprot [1] = 1: Non-Secure
m0_axi_awcache_o [3:0]	sys_clk_i	Output	This signal indicates how transactions are required to progress through a system.
m0_axi_awvalid_o	sys_clk_i	Output	This signal indicates that the channel is signaling valid write address and control information.
m0_axi_awready_i	sys_clk_i	Input	This signal indicates that the subordinate is ready to accept an address and associated control signals.

Port	Clock Domain	Direction	Description

m0_axi_awuser_o [47:0]	sys_clk_i	Output	Provides additional useful transaction information which may be optionally used:
			• m0_axi_awuser_o [47:32]
			  • Provides the PCI Express Requester ID {Bus[7:0], Device[4:0], Function[2:0]} (or {Bus[7:0], Function[7:0]} for ARI capable sources) received with transactions that originated from PCI Express.
			• m0_axi_awuser_o [31:24]
			  • Provides the function number targeted by the transaction. For transaction originating from PCIe, this is the function that has claimed the transaction. For DMA transactions, this is the function that contains the registers for the DMA channel.
			• m0_axi_awuser_o [23]
			  • For DMA sources (m0_axi_awuser_o [11] == 1), it indicates when the transaction is the end of a DMA packet. This information is typically needed for FIFO DMA applications. This bit is always 1 for non-DMA sources. FIFO DMA applications need to know when the packet ends because the final packet word cannot be written to a DMA write FIFO or popped from a DMA read FIFO until the full word has been provided/consumed or the end of a packet has occurred indicating no more data will be coming/requested.
			• m0_axi_awuser_o [22] – Reserved.
			• m0_axi_awuser_o [21:12]
			  • Provides the exact byte count being requested. The requested byte count is otherwise only known to the resolution m0_axi_awsize_o of or by inspecting the byte enables.
			• m0_axi_awuser_o [11:0]
			  • Encodes the transaction source. FIFO DMA applications, which may associate each hardware FIFO with a DMA Channel, can use this information to determine which FIFO is being addressed. m0_axi_awuser_o [11:0] encoding is described in the text following this table.

m0_axi_wid_o [3:0]	sys_clk_i	Output	This signal is the ID tag of the write data transfer. Supported only in AXI3.
m0_axi_wdata_o [511:0]	sys_clk_i	Output	Write data.
m0_axi_wstrb_o [63:0]	sys_clk_i	Output	This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
m0_axi_wlast_o	sys_clk_i	Output	This signal indicates the last transfer in a write burst.
m0_axi_wvalid_o	sys_clk_i	Output	This signal indicates that valid write data and strobes are available.
m0_axi_wready_i	sys_clk_i	Input	This signal indicates that the subordinate can accept the write data.

Port	Clock Domain	Direction	Description
m0_axi_bid_i [3:0]	sys_clk_i	Input	This signal is the ID tag of the write response.
m0_axi_bresp_i [1:0]	sys_clk_i	Input	This signal indicates the status of the write transaction.
m0_axi_bvalid_i	sys_clk_i	Input	This signal indicates that the channel is signaling a valid write response.
m0_axi_bready_o	sys_clk_i	Output	This signal indicates that the manager can accept a write response.

Table 4.13. DMA AXI Manager Read Interface

Port	Clock Domain	Direction	Description
m0_axi_arid_o [3:0]	sys_clk_i	Output	This signal is the identification tag for the read address group of signals.
m0_axi_araddr_o [63:0]	sys_clk_i	Output	The read address gives the address of the first transfer in a read burst transaction.
m0_axi_arlen_o [3:0]	sys_clk_i	Output	This signal indicates the exact number of transfers in a burst.
m0_axi_arsize_o [2:0]	sys_clk_i	Output	This signal indicates the size of each transfer in the burst.
m0_axi_arprot_o [2:0]	sys_clk_i	Output	This signal is used only for Security. Only m0_axi_arprot [1] is used; [2],[0] are unused and set to 0. • m0_axi_arprot [1] = 0: Secure • m0_axi_arprot [1] = 1: Non-Secure
m0_axi_arcache_o [3:0]	sys_clk_i	Output	This signal indicates how transactions are required to progress through a system.
m0_axi_arvalid_o	sys_clk_i	Output	This signal indicates that the channel is signaling valid read address and control information.
m0_axi_arready_i	sys_clk_i	Input	This signal indicates that the subordinate is ready to accept an address and associated control signals.

Port	Clock Domain	Direction	Description

m0_axi_aruser_o [47:0]	sys_clk_i	Input	Provides additional useful transaction information which may be optionally used:
			• m0_axi_awuser_o [47:32]
			  • Provides the PCI Express Requester ID {Bus[7:0], Device[4:0], Function[2:0]} (or {Bus[7:0], Function[7:0]} for ARI capable sources) received with transactions that originated from PCI Express.
			• m0_axi_awuser_o [31:24]
			  • Provides the function number targeted by the transaction. For transaction originating from PCIe, this is the function that has claimed the transaction. For DMA transactions, this is the function that contains the registers for the DMA channel.
			• m0_axi_awuser_o [23]
			  • For DMA sources (m0_axi_awuser_o [11] == 1), it indicates when the transaction is the end of a DMA packet. This information is typically needed for FIFO DMA applications. This bit is always 1 for non-DMA sources. FIFO DMA applications need to know when the packet ends because the final packet word cannot be written to a DMA write FIFO or popped from a DMA read FIFO until the full word has been provided/consumed or the end of a packet has occurred indicating no more data will be coming/requested.
			• m0_axi_awuser_o [22] – Reserved.
			• m0_axi_awuser_o [21:12]
			  • The requested byte count is otherwise only known to the resolution of m0_axi_arsize_o. FIFO DMA applications, which generally cannot over-read, typically need this information.
			• m0_axi_awuser_o [11:0]
			  • Encodes the transaction source. FIFO DMA applications, which may associate each hardware FIFO with a DMA Channel, can use this information to determine which FIFO is being addressed. m0_axi_awuser_o [11:0] encoding is described in the text following this table.

m0_axi_rid_i [3:0]	sys_clk_i	Input	This signal is the identification tag for the read data group of signals generated by the subordinate.
m0_axi_rdata_i [511:0]	sys_clk_i	Input	Read data.
m0_axi_rresp_i [1:0]	sys_clk_i	Input	This signal indicates the status of the read transfer.
m0_axi_rlast_i	sys_clk_i	Input	This signal indicates the last transfer in a read burst.
m0_axi_rvalid_i	sys_clk_i	Input	This signal indicates that the channel is signaling the required read data.
m0_axi_rready_o	sys_clk_i	Output	This signal indicates that the manager can accept the read data and response information.

Table 4.14. DMA AXI Subordinate Write Interface

Port	Clock Domain	Direction	Description
s0_axi_awid_i [15:0]	sys_clk_i	Input	This signal is the identification tag for the write address group of signals.
s0_axi_awaddr_i [63:0]	sys_clk_i	Input	The write address gives the address of the first transfer in a write burst transaction.
s0_axi_awlen_i [3:0]	sys_clk_i	Input	The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. Max of 16 beat bursts.
s0_axi_awsize_i [2:0]	sys_clk_i	Input	This signal indicates the size of each transfer in the burst.
s0_axi_awprot_i [2:0]	sys_clk_i	Input	This signal is used only for Security. Only s0_axi_awprot [1] is used; [2],[0] are unused and set to 0. • s0_axi_awprot [1] = 0: Secure • s0_axi_awprot [1] = 1: Non-Secure
s0_axi_awcache_i [3:0]	sys_clk_i	Input	This signal indicates how transactions are required to progress through a system.
s0_axi_awvalid_i	sys_clk_i	Input	This signal indicates that the channel is signaling valid write address and control information.
s0_axi_awready_o	sys_clk_i	Output	This signal indicates that the subordinate is ready to accept an address and associated control signals.
s0_axi_awuser_i [7:0]	sys_clk_i	Input	Use to specify the function number to be used if this write request is destined for PCIe and the PCIe core supports multiple functions.
s0_axi_wid_i [3:0]	sys_clk_i	Input	This signal is the ID tag of the write data transfer. Supported only in AXI3. Must tie to ‘b0 in AXI4.
s0_axi_wdata_i [511:0]	sys_clk_i	Input	Write data.
s0_axi_wstrb_i [63:0]	sys_clk_i	Input	This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
s0_axi_wlast_i	sys_clk_i	Input	This signal indicates the last transfer in a write burst.
s0_axi_wvalid_i	sys_clk_i	Input	This signal indicates that valid write data and strobes are available.
s0_axi_wready_o	sys_clk_i	Output	This signal indicates that the subordinate can accept the write data.
s0_axi_bid_o [3:0]	sys_clk_i	Output	This signal is the ID tag of the write response.
s0_axi_bresp_o [1:0]	sys_clk_i	Output	This signal indicates the status of the write transaction.
s0_axi_bvalid_o	sys_clk_i	Output	This signal indicates that the channel is signaling a valid write response.
s0_axi_bready_i	sys_clk_i	Input	This signal indicates that the manager can accept a write response.

Table 4.15. DMA AXI Subordinate Read Interface

Port	Clock Domain	Direction	Description
s0_axi_arid_i [15:0]	sys_clk_i	Output	This signal is the identification tag for the read address group of signals.
s0_axi_araddr_i [63:0]	sys_clk_i	Output	The read address gives the address of the first transfer in a read burst transaction.
s0_axi_arlen_i [3:0]	sys_clk_i	Output	This signal indicates the exact number of transfers in a burst.
s0_axi_arsize_i [2:0]	sys_clk_i	Output	This signal indicates the size of each transfer in the burst.
s0_axi_arprot_i [2:0]	sys_clk_i	Output	This signal is used only for Security. Only m0_axi_arprot [1] is used; [2],[0] are unused and set to 0. • s0_axi_arprot [1] = 0: Secure • s0_axi_arprot [1] = 1: Non-Secure
s0_axi_arcache_i [3:0]	sys_clk_i	Output	This signal indicates how transactions are required to progress through a system.
s0_axi_arvalid_i	sys_clk_i	Output	This signal indicates that the channel is signaling valid read address and control information.
s0_axi_arready_o	sys_clk_i	Input	This signal indicates that the subordinate is ready to accept an address and associated control signals.
s0_axi_aruser_i [7:0]	sys_clk_i	Input	Use to specify the function number to be used if this write request is destined for PCIe and the PCIe core supports multiple functions.
s0_axi_rid_i [3:0]	sys_clk_i	Input	This signal is the identification tag for the read data group of signals generated by the subordinate.
s0_axi_rdata_i [511:0]	sys_clk_i	Input	Read data.
s0_axi_rresp_i [1:0]	sys_clk_i	Input	This signal indicates the status of the read transfer.
s0_axi_rlast_i	sys_clk_i	Input	This signal indicates the last transfer in a read burst.
s0_axi_rvalid_i	sys_clk_i	Input	This signal indicates that the channel is signaling the required read data.
s0_axi_rready_o	sys_clk_i	Output	This signal indicates that the manager can accept the read data and response information.

Table 4.16. DMA AXI Interrupt Interface

Port	Clock Domain	Direction	Description
s_int_tx_i[63:0]	sys_clk_i	Input	This signal is used to cause interrupts to be generated on PCIe. It must be driven depending upon the value of s_int_tx_edge_level_n_o: 1 – Edge Interrupt mode PCIe Core is configured for MSI-X or MSI interrupts Set s_int_tx_i[i] == 1 for one clock to request an interrupt on interrupt vector[i] 0 – Level Interrupt mode PCIe Core is configured for Legacy interrupts Set s_int_tx_i[i] == 1 for as long as interrupt vector[i] continues to be pending s_int_tx[i] must be 0 except when an interrupt is desired to be generated.
s_int_tx_edge_level_n_o[63:0]	sys_clk_i	Output	s_int_tx_edge_level_n_o[i] indicates how interrupts on s_int_tx_i[i] must be signaled. For cores supporting multi-function or SR-IOV, different interrupt vectors are associated with each function and each function has independent MSI-X and MSI enables.

4.9. AXI Data Interface (DMA)

Table 4.17. AXI-MM Manager Interface (DMA)

Port	Clock Domain	Direction	Description
m0_dma_axi_awid_o [2:0]	sys_clk_i	Output	This signal is the identification tag for the write address group of signals.
m0_dma_axi_awaddr_o [63:0]	sys_clk_i	Output	The write address in a write transaction.
m0_dma_axi_awlen_o [7:0]	sys_clk_i	Output	The burst length gives the exact number of transfers in a burst.
m0_dma_axi_awsize_o [2:0]	sys_clk_i	Output	This signal indicates the size of each transfer.
m0_dma_axi_awburst_o [1:0]	sys_clk_i	Output	Burst mode. Always 2'b01.
m0_dma_axi_awlock_o	sys_clk_i	Output	This signal is unused and always 0.
m0_dma_axi_awprot_o [2:0]	sys_clk_i	Output	This signal is unused and always 0.
m0_dma_axi_awcache_o [3:0]	sys_clk_i	Output	This signal is unused and always 0.
m0_dma_axi_awvalid_o	sys_clk_i	Output	This signal indicates that the channel is signaling valid write address and control information.
m0_dma_axi_awready_i	sys_clk_i	Input	This signal indicates that the subordinate is ready to accept an address and associated control signals.
m0_dma_axi_wdata_o [255:0]	sys_clk_i	Output	Write data.
m0_dma_axi_wstrb_o [31:0]	sys_clk_i	Output	This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
m0_dma_axi_wlast_o	sys_clk_i	Output	This signal indicates the last transfer in a write burst.
m0_dma_axi_wvalid_o	sys_clk_i	Output	This signal indicates that valid write data and strobes are available.
m0_dma_axi_wready_i	sys_clk_i	Input	This signal indicates that the subordinate can accept the write data.
m0_dma_axi_bid_i [3:0]	sys_clk_i	Input	This signal is the ID tag of the write response.
m0_dma_axi_bresp_i [1:0]	sys_clk_i	Input	This signal indicates the status of the write transaction.
m0_dma_axi_bvalid_i	sys_clk_i	Input	This signal indicates that the channel is signaling a valid write response.
m0_dma_axi_bready_o	sys_clk_i	Output	This signal indicates that the manager can accept a write response.
m0_dma_axi_arid_o [2:0]	sys_clk_i	Output	This signal is the identification tag for the read address group of signals.
m0_dma_axi_araddr_o [63:0]	sys_clk_i	Output	The read address gives the address of the first transfer in a read burst transaction.
m0_dma_axi_arlen_o [7:0]	sys_clk_i	Output	This signal indicates the exact number of transfers in a burst.
m0_dma_axi_arsize_o [2:0]	sys_clk_i	Output	This signal indicates the size of each transfer.
m0_dma_axi_arburst_o [1:0]	sys_clk_i	Output	Burst mode. Always 2'b01.
m0_dma_axi_arprot_o [2:0]	sys_clk_i	Output	This signal is unused and always 0.
m0_dma_axi_arlock_o	sys_clk_i	Output	This signal is unused and always 0.
m0_dma_axi_arcache_o [3:0]	sys_clk_i	Output	This signal is unused and always 0.
m0_dma_axi_arvalid_o	sys_clk_i	Output	This signal indicates that the channel is signaling valid read address and control information.
m0_dma_axi_arready_i	sys_clk_i	Input	This signal indicates that the subordinate is ready to accept an address and associated control signals.
m0_dma_axi_arqos_o [3:0]	sys_clk_i	Output	This signal is unused and always 0.
m0_dma_axi_aruser_o [7:0]	sys_clk_i	Output	This signal is unused and always 0.
m0_dma_axi_rid_i [2:0]	sys_clk_i	Input	This signal is the identification tag for the read data group of signals generated by the subordinate.
m0_dma_axi_rdata_i [255:0]	sys_clk_i	Input	Read data.
m0_dma_axi_rresp_i [1:0]	sys_clk_i	Input	This signal indicates the status of the read transfer.

Port	Clock Domain	Direction	Description
m0_dma_axi_rlast_i	sys_clk_i	Input	This signal indicates the last transfer in a read burst.
m0_dma_axi_rvalid_i	sys_clk_i	Input	This signal indicates that the channel is signaling the required read data.
m0_dma_axi_rready_o	sys_clk_i	Output	This signal indicates that the manager can accept the read data and response information.

4.10. AXI Data Interface (Bridge Mode)

Table 4.18. AXI-MM Manager Write Interface (Bridge Mode)

Port	Clock Domain	Direction	Description
m0_aximm_awid_o [7:0]	sys_clk_i	Output	This signal is the identification tag for the write address group of signals.
m0_aximm_awaddr_o [63:0]	sys_clk_i	Output	The write address in a write transaction.
m0_aximm_awlen_o [7:0]	sys_clk_i	Output	Burst mode is not supported. Always 8'h00.
m0_aximm_awsize_o [2:0]	sys_clk_i	Output	This signal indicates the size of each transfer.
m0_aximm_awburst_o [1:0]	sys_clk_i	Output	Burst mode is not supported. Always 2'b00.
m0_aximm_awlock_o	sys_clk_i	Output	This signal is unused and always 0.
m0_aximm_awprot_o [2:0]	sys_clk_i	Output	This signal is unused and always 0.
m0_aximm_awcache_o [3:0]	sys_clk_i	Output	This signal is unused and always 0.
m0_aximm_awvalid_o	sys_clk_i	Output	This signal indicates that the channel is signaling valid write address and control information.
m0_aximm_awready_i	sys_clk_i	Input	This signal indicates that the subordinate is ready to accept an address and associated control signals.
m0_aximm_wdata_o [31:0]	sys_clk_i	Output	Write data.
m0_aximm_wstrb_o [3:0]	sys_clk_i	Output	This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
m0_aximm_wlast_o	sys_clk_i	Output	This signal indicates the last transfer in a write burst.
m0_aximm_wvalid_o	sys_clk_i	Output	This signal indicates that valid write data and strobes are available.
m0_aximm_wready_i	sys_clk_i	Input	This signal indicates that the subordinate can accept the write data.
m0_aximm_bid_i [7:0]	sys_clk_i	Input	This signal is the ID tag of the write response.
m0_aximm_bresp_i [1:0]	sys_clk_i	Input	This signal indicates the status of the write transaction.
m0_aximm_bvalid_i	sys_clk_i	Input	This signal indicates that the channel is signaling a valid write response.
m0_aximm_bready_o	sys_clk_i	Output	This signal indicates that the manager can accept a write response.
m0_aximm_arid_o [7:0]	sys_clk_i	Output	This signal is the identification tag for the read address group of signals.
m0_aximm_araddr_o [63:0]	sys_clk_i	Output	The read address gives the address of the first transfer in a read burst transaction.
m0_aximm_arlen_o [7:0]	sys_clk_i	Output	This signal indicates the exact number of transfers in a burst.
m0_aximm_arsize_o [2:0]	sys_clk_i	Output	This signal indicates the size of each transfer.
m0_aximm_arburst_o [1:0]	sys_clk_i	Output	Burst mode is not supported. Always 2'b00.
m0_aximm_arprot_o [2:0]	sys_clk_i	Output	This signal is unused and always 0.
m0_aximm_arlock_o	sys_clk_i	Output	This signal is unused and always 0.
m0_aximm_arcache_o [3:0]	sys_clk_i	Output	This signal is unused and always 0.

Port	Clock Domain	Direction	Description
m0_aximm_arvalid_o	sys_clk_i	Output	This signal indicates that the channel is signaling valid read address and control information.
m0_aximm_arready_i	sys_clk_i	Input	This signal indicates that the subordinate is ready to accept an address and associated control signals.
m0_aximm_arqos_o [3:0]	sys_clk_i	Output	This signal is unused and always 0.
m0_aximm_aruser_o [7:0]	sys_clk_i	Output	This signal is unused and always 0.
m0_aximm_rid_i [7:0]	sys_clk_i	Input	This signal is the identification tag for the read data group of signals generated by the subordinate.
m0_aximm_rdata_i [31:0]	sys_clk_i	Input	Read data.
m0_aximm_rresp_i [1:0]	sys_clk_i	Input	This signal indicates the status of the read transfer.
m0_aximm_rlast_i	sys_clk_i	Input	This signal indicates the last transfer in a read burst.
m0_aximm_rvalid_i	sys_clk_i	Input	This signal indicates that the channel is signaling the required read data.
m0_aximm_rready_o	sys_clk_i	Output	This signal indicates that the manager can accept the read data and response information.

Table 4.19. AXI-Lite Manager Interface (Bridge Mode)

Port	Clock Domain	Direction	Description
m0_axil_awaddr_o [63:0]	sys_clk_i	Output	The write address in a write transaction.
m0_axil_awvalid_o	sys_clk_i	Output	This signal indicates that the channel is signaling valid write address and control information.
m0_axil_awready_i	sys_clk_i	Input	This signal indicates that the subordinate is ready to accept an address and associated control signals.
m0_axi_wdata_o [31:0]	sys_clk_i	Output	Write data.
m0_axil_wstrb_o [3:0]	sys_clk_i	Output	This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
m0_axil_wvalid_o	sys_clk_i	Output	This signal indicates that valid write data and strobes are available.
m0_axil_wready_i	sys_clk_i	Input	This signal indicates that the subordinate can accept the write data.
m0_axil_bresp_i [1:0]	sys_clk_i	Input	This signal indicates the status of the write transaction.
m0_axil_bvalid_i	sys_clk_i	Input	This signal indicates that the channel is signaling a valid write response.
m0_axil_bready_o	sys_clk_i	Output	This signal indicates that the manager can accept a write response.
m0_axil_araddr_o [63:0]	sys_clk_i	Output	The read address gives the address of the first transfer in a read burst transaction.
m0_axil_arvalid_o	sys_clk_i	Output	This signal indicates that the channel is signaling valid read address and control information.
m0_axil_arready_i	sys_clk_i	Input	This signal indicates that the subordinate is ready to accept an address and associated control signals.
m0_axil_rdata_i [31:0]	sys_clk_i	Input	Read data.
m0_axil_rresp_i [1:0]	sys_clk_i	Input	This signal indicates the status of the read transfer.
m0_axil_rvalid_i	sys_clk_i	Input	This signal indicates that the channel is signaling the required read data.
m0_axil_rready_o	sys_clk_i	Output	This signal indicates that the manager can accept the read data and response information.

4.11. Function Level Reset (FLR) Interface

This interface is available when the option Link 0: Disable Function Level Reset (FLR) is unticked.

Table 4.20. Function Level Reset Interface

Port	Clock Domain	Direction	Description
link0_flr_ack_i[LINK0_FLR_NUM-1:0]	sys_clk_i	Input	Per function FLR acknowledge. For hard DMA variant, LINK0_FLR_NUM is set to 32 to support the increased number of Virtual Functions (VFs) required by SR-IOV. In other variants where only Physical Functions (PFs) are supported, LINK0_FLR_NUM is equivalent to LINK0_NUM_FUNCTIONS. Function Index Mapping (0-32) **Indices 0–7**: Map directly to *physical functions 0–7*. The index for respective virtual functions can be derived using the following equation: **FLR index for VF** = (7 + (*pf_num* * 3) + *vf_num* where *pf_num* is the physical function's number and *vf_num* is the *vf_num* is the virtual function's number. **Example:** For physical function 1, virtual function 3, the FLR index for that VF is 7 + (1*3) + 3 = 13. For physical function 5, virtual function 2, the FLR index for that VF is 7 + (5*3) + 2 = 24. In response to link0_flr_o[i] == 1, set link0_flr_ack_i[i] == 1 for one clock to indicate that you completed processing an active FLR for function[i] and is ready to exit FLR for the function. If the user has no special requirements for FLR, link0_flr_ack_i, it may be tied to all 1s to permit the core to immediately exit FLR whenever it is entered through link0_flr_o[i] == 1. FLR is only enabled for Endpoints. Per PCIe spec, SRIOV capable Endpoints may not disable FLR support.
link0_flr_o[LINK0_FLR_NUM-1:0]	sys_clk_i	Output	Per function FLR indicator. For hard DMA variant, LINK0_FLR_NUM is set to 32 to support the increased number of Virtual Functions (VFs) required by SR-IOV. In other variants where only Physical Functions (PFs) are supported, LINK0_FLR_NUM is equivalent to LINK0_NUM_FUNCTIONS. Function Index Mapping (0-32) **Indices 0–7**: Map directly to *physical functions 0–7.* The index for respective virtual functions can be derived using the following equation: **FLR index for VF** = (7 + (*pf_num* * 3) + *vf_num* where *pf_num* is the physical function's number and *vf_num* is the *vf_num* is the virtual function's number. **Example:** For physical function 1, virtual function 3, the FLR index for that VF is 7 + (1*3) + 3 = 13. For physical function 5, virtual function 2, the FLR index for that VF is 7 + (5*3) + 2 = 24.

Port	Clock Domain	Direction	Description
			link0_flr_o[i] == 1 indicates FLR is active for function[i]. indicates FLR is active for function[i]. link0_flr_o[i] == 0 indicates FLR is not active for function[i]. FLR is a function-specific soft reset that occurs when software writes the FLR register in a function's configuration space to 1. When FLR is active, the function's Configuration Space registers are reset to their default values (except Sticky registers as specified by PCIe spec.). A function's FLR Configuration Space register remains set until flr_ack[i] for the associated function[i] is set to 1 for one clock to indicate that you completed resetting the application logic associated with that function.

5. Register Description

Table 5.1. Register Access Abbreviations

Abbreviation	Meaning
RW	Read and Write access
RO	Read only
WO	Write only
RW1C	Read write 1 to clear

5.1. Hard IP Core Configuration and Status Registers

The Lattice PCIe x8 IP Core configuration registers have default values that are appropriate for most applications. Customers typically only want to change a small number of values such as Vendor/Device ID and BAR configuration. Such changes can be made through LMMI writes prior to core reset release or through the IP generation user interface. The registers defined in the sections below are the same set for all links. The registers are configured through LMMI or AXI-Lite interface.

Table 5.2 lists the offset address for the Hard IP Core Registers to access through AXI-Lite interface.

Due to a known issue, you are advised to use AXI-Lite interface for PCIe x8 IP Core configuration registers access.

Table 5.2. Hard PCIe Core Register Mapping

Hard IP Core	Register Block	Start Byte Offset	End Byte Offset
QUAD 0			
PCIe x8	**PCIe Configuration Space Registers**		
	Physical Function 0	0x0_0000	0x0_0FFF
	Physical Function 1	0x0_1000	0x0_1FFF
	Physical Function 2	0x0_2000	0x0_2FFF
	Physical Function 3	0x0_3000	0x0_3FFF
	Physical Function 4	0x0_4000	0x0_4FFF
	Physical Function 5	0x0_5000	0x0_5FFF
	Physical Function 6	0x0_6000	0x0_6FFF
	Physical Function 7	0x0_7000	0x0_7FFF
	PF0 Virtual Function 0	0x0_8000	0x0_8FFF
	PF0 Virtual Function 1	0x0_9000	0x0_9FFF
	PF0 Virtual Function 2	0x0_A000	0x0_AFFF
	PF1 Virtual Function 0	0x0_B000	0x0_BFFF
	PF1 Virtual Function 1	0x0_C000	0x0_CFFF
	PF1 Virtual Function 2	0x0_D000	0x0_DFFF
	PF2 Virtual Function 0	0x0_E000	0x0_EFFF
	PF2 Virtual Function 1	0x0_F000	0x0_FFFF
	PF2 Virtual Function 2	0x1_0000	0x1_0FFF
	PF3 Virtual Function 0	0x1_1000	0x1_1FFF
	PF3 Virtual Function 1	0x1_2000	0x1_2FFF
	PF3 Virtual Function 2	0x1_3000	0x1_3FFF
	PF4 Virtual Function 0	0x1_4000	0x1_4FFF
	PF4 Virtual Function 1	0x1_5000	0x1_5FFF
	PF4 Virtual Function 2	0x1_6000	0x1_6FFF

Hard IP Core	Register Block	Start Byte Offset	End Byte Offset
	PF5 Virtual Function 0	0x1_7000	0x1_7FFF
	PF5 Virtual Function 1	0x1_8000	0x1_8FFF
	PF5 Virtual Function 2	0x1_9000	0x1_9FFF
	PF6 Virtual Function 0	0x1_A000	0x1_AFFF
	PF6 Virtual Function 1	0x1_B000	0x1_BFFF
	PF6 Virtual Function 2	0x1_C000	0x1_CFFF
	PF7 Virtual Function 0	0x1_D000	0x1_DFFF
	PF7 Virtual Function 1	0x1_E000	0x1_EFFF
	PF7 Virtual Function 2	0x1_F000	0x1_FFFF
	PCIe x8 Core Registers		
	mgmt_tlb	0x4_2000	0x4_2717
	mgmt_ptl	0x4_3000	0x4_32E3
	mgmt_ftl	0x4_4000	0x4_4337
	mgmt_ftl_mf1	0x4_5000	0x4_5337
	mgmt_ftl_mf2	0x4_6000	0x4_6337
	mgmt_ftl_mf3	0x4_7000	0x4_7337
	mgmt_ftl_mf4	0x4_8000	0x4_8337
	mgmt_ftl_mf5	0x4_9000	0x4_9337
	mgmt_ftl_mf6	0x4_A000	0x4_A337
	mgmt_ftl_mf7	0x4_B000	0x4_B337
Quad 0 MPPHY	**MPPHY 0 Registers**		
	MPPHY Lane 0 – Lane 3	0x05_0000	0x05_B1FF
	Quad 0 PMA, Common	0x06_0000	0x06_FFFF
	Quad 0 PHY ROM	0x07_0000	0x07_3FFF
	Quad 0 PHY RAM	0x07_4000	0x07_FFFF
QUAD 1			
Quad 1 MPPHY	**MPPHY 1 Registers**		
	MPPHY Lane 4 – Lane 7	0x15_0000	0x15_B1FF
	Quad 1 PMA, Common	0x16_0000	0x16_FFFF
	Quad 1 PHY ROM	0x17_0000	0x17_3FFF
	Quad 1 PHY RAM	0x17_4000	0x17_FFFF

5.1.1. EP Configuration Settings

The Lattice PCIe x8 IP Core supports Endpoint (EP) operation. The current mode of operation is determined by the core CSR. The following table illustrates the CSR values that are recommended for EP and RP applications.

Table 5.3. CSR Values Recommended for EP Applications

Register Field	Offset	EndPoint
mgmt_tlb_ltssm_port_type_ds_us_n	0x4_2040	1'b0
mgmt_ftl_cfg_type1_type0_n	0x4_4030	1'b0
mgmt_ftl_decode_ignore_poison	0x4_4010	1'b0
mgmt_ftl_decode_t1_rx_bypass_msg_dec	0x4_4014	1'b0
mgmt_ftl_pcie_cap_slot_implemented	0x4_4080	1'b0
mgmt_ftl_pcie_cap_device_port_type	0x4_4080	4'h0
mgmt_ftl_id3_class_code	0x4_4048	User Application Specific

Register Field	Offset	EndPoint
mgmt_ftl_ari_cap_disable	0x4_40E0	1'b0
mgmt_ftl_msi_cap_disable	0x4_40E8	1'b0
mgmt_ftl_msi_cap_mult_message_capable	0x4_40E8	User Application Specific
mgmt_ftl_msix_cap_table_size	0x4_40F0	User Application Specific
mgmt_ftl_msix_cap_disable	0x4_40F0	1'b0 (Enabled)
mgmt_ftl_aer_cap_en_surprise_down_error	0x4_4100	1'b0

5.1.2. mgmt_tlb (0x4_2000)

The following are the register sets with the 0x4_2000 base address.

Note: Registers in the range not described in this section are considered reserved and must not be accessed. The access to this range results in unexpected behavior.

5.1.2.1. LTSSM Register Set

ltssm_simulation Register 0x0

This register set is used for LTSSM simulation speed reduction.

Table 5.4. ltssm_simulation Register 0x0

Field	Name	Access	Width	Reset	Description
[31:2]	reserved	RO	30	0x0	—
[1]	reduce_ts1	RW	1	0x0	Reduce the minimum number of TS1 transmitted in Polling.Active from 1024 to 16 to shorten simulation time. 0 – Disable 1 – Enable
[0]	reduce_timeouts	RW	1	0x0	Reduce LTSSM timeouts to shorten simulation time. When enabled, 1 ms-> 20 µs, 2 ms->40 µs, 12 ms->60 µs, 24 ms->80 µs, 32 ms->100 µs, and 48 ms->160 µs. 0 – Disable 1 – Enable

ltssm_cfg_lw_start Register 0x34

This register set is used for LTSSM CFG.LWSTART configuration.

Table 5.5. ltssm_cfg_lw_start Register 0x34

Field	Name	Access	Width	Reset	Description
[31:2]	reserved	RO	30	0x0	—
[1:0]	min_time	RW	2	0x0	Minimum time spent in Cfg.LW.Start before exit is permitted. 0 – 4 µs 1 – 16 µs 2 – 64 µs

ltssm_latch_rx Register 0x38

This register set is used for LTSSM latch RX configuration.

Table 5.6. ltssm_latch_rx Register 0x38

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	30	0x0	—
[0]	link_lane	RW	1	0x1	Enable latching each lane's received link and lane numbers and state exit condition during LTSSM Configuration link width negotiation. 0 – Disable. The lane is included in the link if it is receiving the state exit criteria on the clock cycle that the link width and state exit transition is occurring. A received Physical Layer error occurring close to the clock cycle that the link width is being determined results in a reduction of link width even if the lane had previously recorded valid state exit criteria. 1 – Enable. The lane is included in the link if it met the state exit criteria at any time during the state. This is the recommended setting since received Physical Layer errors are less likely to result in reduced link width.

ltssm_cfg Register 0x3c

This register set is used for LTSSM configuration.

Table 5.7. ltssm_cfg Register 0x3c

Field	Name	Access	Width	Reset	Description
[31:28]	lw_start_updn_end_delay	RW	4	0x9	LTSSM CFG_[US/DS]_LW_START normal CFG_[US/DS]_LW_START TS1 transmissions and parsing of received TS OS begins (lw_start_updn_end_delay × 64) symbols after the bp_ltssm_cfg_lw_start_updn 1 to 0 transition occurs at the end of PHY adaptation. This delay is intended to flush any corrupted PHY rx data due to the PHY adaptation through the Link Layer Core before the Core begins paying attention to received data again.
[27:24]	lw_start_updn_start_delay	RW	4	0x8	LTSSM CFG_[US/DS]_LW_START bp_ltssm_cfg_lw_start_updn==1 assertion is delayed by (lw_start_updn_start_delay × 64) symbols from CFG_[US/DS]_LW_START state entry. The start delay is intended to avoid the PHY beginning adaptation, and thus corrupting the input data, before the link partner data stream has ended. When the Core reaches CFG_[US/DS]_LW_START before the link partner, the link partner may still be in Recovery.Idle with an active data stream. The start delay must be long enough to delay PHY adaptation until the receive data stream has ended or else SKP Data Parity Errors and Receiver Errors can be detected and recorded by the Core due to the PHY corrupting the receive data stream due to adaptation.
[23:12]	lw_start_updn_count	RW	12	0xfa	LTSSM CFG_[US/DS]_LW_START bp_ltssm_cfg_lw_start_updn==1 duration is set to (lw_start_updn_count × 1024) ns. 0==Disabled.

Field	Name	Access	Width	Reset	Description
[11:8]	lw_start_updn_rate_en	RW	4	0xf	LTSSM CFG_[US/DS]_LW_START bp_ltssm_cfg_lw_start_updn==1 rate enable/disable. Controls for which speeds the bp_ltssm_cfg_lw_start_updn feature is supported. One bit is provided to enable/disable each speed supported {16G, 8G, 5G, 2.5G}. Bit positions for speeds that are not supported by a given core delivery must be set to 0. 0 – Disable feature when at the associated link speed. 1 – Enable feature when at the associated link speed.
[7:6]	reserved	RO	2	0x0	—
[5]	lw_start_updn_eie_en	RW	1	0x0	*lw_start_updn_eie_en* LTSSM CFG_[US/DS]_LW_START bp_ltssm_cfg_lw_start_updn==1 EIE Tx OS enable. 0 – Disabled 1 – Enabled
[4]	lw_start_updn_en_dir_ds	RW	1	0x0	LTSSM CFG_[US/DS]_LW_START bp_ltssm_cfg_lw_start_updn==1 directed down-configure enable. 0 – Do not assert bp_ltssm_cfg_lw_start_updn==1 when the CFG_[US/DS]_LW_START entry is due to locally directed downconfigure. 1 – Assert bp_ltssm_cfg_lw_start_updn==1 when the CFG_[US/DS]_LW_START entry is due to locally directed down-configure.
[3:2]	reserved	RO	2	0x0	—
[1]	lw_start_updn_timer_en	RW	1	0x0	LTSSM CFG_[US/DS]_LW_START bp_ltssm_cfg_lw_start_updn Timer Enable. Register lw_start_updn_timer_en can be set to stay in adaptation for a fixed period instead of relying on the PHY to have a port bp_ltssm_cfg_lw_start_updn_ack that is asserted at the end of adaptation. Only one of lw_start_updn_timer_en and lw_start_updn_ack_en can be set to 1. 0 – Disabled. 1 – Deassert bp_ltssm_cfg_lw_start_updn after (lw_start_updn_count × 1024) ns has expired.

Field	Name	Access	Width	Reset	Description
[0]	lw_start_updn_ack_en	RW	1	0x0	LTSSM Configuration Link Width Start bp_ltssm_cfg_lw_start_updn Ack Enable 0 – Disabled. Output port bp_ltssm_cfg_lw_start_updn is held == 0 and input port bp_ltssm_cfg_lw_start_updn_ack is ignored. When CFG_[DS/US]_LW_START is entered from Recovery, the transition from CFG_[DS/US]_LW_START to CFG_[DS/US]_LW_ACCEPT occurs after a minimum of 4 µs. 1 – Enabled. If also enabled, through mgmt_tlb_ltssm_cfg_lw_start_updn_rate_en, at the current link speed, output port bp_ltssm_cfg_lw_start_updn is set upon CFG_[DS/US]_LW_START entry from Recovery and input port bp_ltssm_cfg_lw_start_updn_ack is used. The transition from CFG_[DS/US]_LW_START to CFG_[DS/US]_LW_ACCEPT occurs only after the PHY has asserted bp_ltssm_cfg_lw_start_updn_ack == 1 and additionally a minimum of 4 µs has elapsed. bp_ltssm_cfg_lw_start_updn_ack must not be withheld so long that the state timeout of 24 ms expires or the link exits to detect, and the link goes down, which is a serious error.

ltssm_port_type Register 0x40

This register set is used for the LTSSM port type configuration.

Table 5.8. ltssm_port_type Register 0x40

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	ds_us_n	RW	1	0x0	Determines the PCI Express port type which affects many aspects of LTSSM training. 0 – Upstream Port 1 – Downstream Port

ltssm_ds_link Register 0x44

This register set is used for the LTSSM downstream link configuration.

Table 5.9. ltssm_ds_link Register 0x44

Field	Name	Access	Width	Reset	Description
[31:5]	reserved	RO	25	0x0	—
[4:0]	number	RW	5	0x0	For downstream ports only, unique Link Number assigned to the link and used in TS sets during LTSSM Configuration

ltssm_detect_quiet Register 0x48

This register set is used for the LTSSM Detect.Quiet configuration.

Table 5.10. ltssm_detect_quiet Register 0x48

Field	Name	Access	Width	Reset	Description
[31:5]	reserved	RO	25	0x0	—
[4:0]	number	RW	5	0x0	For downstream ports only, unique Link Number assigned to the link and used in TS sets during LTSSM Configuration

ltssm_rx_det Register 0x4c

This register set is used for the LTSSM receiver detection configuration.

Table 5.11. ltssm_rx_det Register 0x4c

Field	Name	Access	Width	Reset	Description
[31]	override	RW	1	0x0	Lane receiver detection mask enable. 0 – Disable 1 – Enable
[30:16]	reserved	RO	15	0x0	—
[15:0]	mask	RW	16	0x0	*mask* Lane receiver detection mask. When override==1, mask determines which lanes attempt receiver detection. For each lane[i]: 0 – Skip receiver detection and exclude the lane from the link. 1 – Perform receiver detection and use result to determine whether to include/exclude the lane from the link.

ltssm_nfts Register 0x50

This register set is used for the LTSSM NFTS configuration.

Table 5.12. ltssm_nfts Register 0x50

Field	Name	Access	Width	Reset	Description
[31:16]	reserved	RO	16	0x0	—
[15:8]	to_extend	RW	8	0x7f	Number of FTS set transfer times to wait in addition to the time required to transmit the requested NFTS sets before timing out to Recovery on Rx_L0s exit.
[7:0]	nfts	RW	8	0xff	Number of FTS sets to request link partner transmit when exiting L0s. NFTS value transmitted in TS1 and TS2 Ordered Sets during training.

ltssm_ds_initial_auto Register 0x54

This register set is used for the LTSSM initial link speed configuration.

Table 5.13. ltssm_ds_initial_auto Register 0x54

Field	Name	Access	Width	Reset	Description
[31]	rate_enable	RW	1	0x0	*rate_enable* Determines whether link speed up is requested by the core after the first entry to L0 following state Detect. If neither port directs the link at a higher speed, the link remains at 2.5G unless software initiates a speed change. It is recommended to set rate_enable=1 and rate==maximum supported speed. 0 – Let the link partner or software initiate initial speed changes. 1 – Make 1 attempt to direct the link to the maximum speed specified by rate. The speed achieved is the maximum speed, less than or equal to rate, that both the core and link partner support.
[30:2]	reserved	RO	29	0x0	Number of FTS set transfer times to wait in addition to the time required to transmit the requested NFTS sets before timing out to Recovery on Rx_L0s exit.
[1:0]	rate	RW	2	0x0	*rate* When rate_enable==1, indicates the maximum rate that is attempted to negotiate on the initial link training from Detect. Only speeds supported by the core can be indicated. 0 – 2.5G 1 – 5G 2 – 8G[1] 3 – 16G[1]

Note:

1. Available in 2024.1 release.

ltssm_select_deemphasis Register 0x58

This register set is used for the LTSSM 2.5/5G deemphasis configuration.

Table 5.14. ltssm_select_deemphasis Register 0x58

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	6db_3_5db_n	RW	1	0x1	For 5G capable cores only: For upstream ports only, sets the default deemphasis for 5G operation during LTSSM State Detect. 0 – -3.5dB 1 – -6dB

ltssm_beacon Register 0x5c

This register set is used for the LTSSM Beacon configuration.

Table 5.15. ltssm_beacon Register 0x5c

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	l2_d3hot_enable	RW	1	0x0	*l2_d3hot_enable* L2 wake Beacon transmission control. 0 – Disabled. The customer design must wake the link through WAKE# pin assertion. Set to 0 when using PHY which do not support Beacon transmission. Set to 0 if the core is not clocked (some PHY remove the core's clock in L2 while others supply a keep alive clock) or powered (some applications remove core power in L2 to maximize power savings) in L2, as the core is unable to initiate Beacon generation in these cases. 1 – Transmit beacon when directed to wake the link from L2.

ltssm_mod_cpl Register 0x60

This register set is used for the LTSSM Modified Compliance configuration.

Table 5.16. ltssm_mod_cpl Register 0x60

Field	Name	Access	Width	Reset	Description
[31:2]	reserved	RO	31	0x0	—
[1]	one_eieos	RW	1	0x1	When entering Modified Compliance Pattern determines the number of EIEOS blocks to send. 0 – Send 8 EIEOS blocks to ensure receiver lock 1 – Send 1 EIEOS (as per Spec)
[0]	exit_direct_to_detect	RW	1	0x0	*exit_direct_to_detect* When transmitting Modified Compliance Pattern and cfg_enter_compliance == 0, determines which of the two PCIe Specification optional behaviors is selected. 0 – Do not exit to Detect for this reason. 1 – Exit to Detect.

ltssm_rx_elec_idle Register 0x64

This register set is used for the LTSSM Rx Electrical Idle configuration.

Table 5.17. ltssm_rx_elec_idle Register 0x64

Field	Name	Access	Width	Reset	Description
[31]	rec_spd_infer_rcvr_lock	RW	1	0x0	Recovery Speed successful and unsuccessful inference expand to Recovery.RcvrLock enable. 0 – Do not include time spent in Recovery.RcvrLock when calculating successful and unsuccessful speed change electrical idle inference in Recovery.Speed. 1 – Include time spent in Recovery.RcvrLock when calculating successful and unsuccessful speed change electrical idle inference in Recovery.Speed.

Field	Name	Access	Width	Reset	Description
[30]	rec_pd_infer_rcvr_cfg	RW	1	0x0	Recovery Speed successful and unsuccessful inference expand to Recovery.RcvrCfg enable. 0 – Do not include time spent in Recovery.RcvrCfg when calculating successful and unsuccessful speed change electrical idle inference in Recovery.Speed. 1 – Include time spent in Recovery.RcvrCfg when calculating successful and unsuccessful speed change electrical idle inference in Recovery.Speed.
[29]	rec_spd_infer_eq_ph0123	RW	1	0x0	Recovery Speed successful and unsuccessful inference expand to Recovery.EqPhase0123 enable. 0 – Do not include time spent in Recovery.EqPhase0123 when calculating successful and unsuccessful speed change electrical idle inference in Recovery.Speed. 1 – Include time spent in Recovery.EqPhase0123 when calculating successful and unsuccessful speed change electrical idle inference in Recovery.Speed.
[28:4]	reserved	RO	25	0x0	—
[3:0]	filter	RW	4	0x1	After entering an LTSSM state that monitors, pipe_rx_elec_idle for exit, ignore pipe_rx_elec_idle for 128 × filter) nanoseconds to enable tolerance for pipe_rx_elec_idle not latency matched with the associaetd pipe_rx_data.

ltssm_compliance_toggle Register 0x68

This register set is used for the LTSSM Compliance Toggle configuration.

Table 5.18. ltssm_compliance_toggle Register 0x68

Field	Name	Access	Width	Reset	Description
[31:4]	reserved	RO	28	0x0	—
[3:2]	max_speed	RW	2	0x3	Maximum speed of compliance patterns that must be generated. 0 – 2.5G 1 – 5G 2 – 8G 3 – 16G
[1:0]	min_speed	RW	2	0x0	Minimum speed of compliance patterns that must be generated. 0 – 2.5G 1 – 5G 2 – 8G 3 – 16G

ltssm_prevent_rx_ts_entry_to Register 0x6c

This register set is used for the LTSSM State Rx TS Transition Prevention configuration.

Table 5.19. ltssm_prevent_rx_ts_entry_to Register 0x6c

Field	Name	Access	Width	Reset	Description
[31:4]	reserved	RO	28	0x0	—
[3]	compliance	RW	1	0x0	LTSSM to Polling.Compliance Rx TS state transition disable. 0 – Enabled 1 – Disabled
[2]	loopback	RW	1	0x0	LTSSM to Loopback Follower Rx TS state transition disable. 0 – Enabled 1 – Disabled
[1]	hot_reset	RW	1	0x0	LTSSM to Hot Reset Rx TS state transition disable. 0 – Enabled 1 – Disabled
[0]	disable	RW	1	0x0	LTSSM to Disable Rx TS state transition disable. 0 – Enabled 1 – Disabled

ltssm_link Register 0x80

This register is used for the Current Link Status configuration.

Table 5.20. ltssm_link Register 0x80

Field	Name	Access	Width	Reset	Description
[31]	dl_link_up	RO	1	0x0	Data Link Layer link up status. 0 – Down 1 – Up
[30]	pl_link_up	RO	1	0x0	Physical Layer link up status. 0 – Down 1 – Up
[29:20]	Reserved	RO	10	0x0	—
[19:16]	lane_rev_status	RO	4	0x0	Indicates the current lane reversal status: lane_rev_status[0], 1 == Full Reverse is in effect else 0 lane_rev_status[1], 1 == x2 Reverse is in effect (≥ 4 lane only) else 0 lane_rev_status[2], 1 == x4 Reverse is in effect (≥ 8 lane only) else 0 lane_rev_status[3], 1 == x8 Reverse is in effect (≥ 16 lane only) else 0
[15]	idle_infer_rec_rcvr_cfg	RW1C	1	0x0	Electrical Idle inference status in Recovery.RcvrCfg. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[14]	idle_infer_loopback_slave	RW1C	1	0x0	Electrical Idle inference status in Loopback.Active as a Loopback Follower. 0 – Otherwise 1 – Event occurred. Write 1 to clear
[13]	idle_infer_rec_speed2_success	RW1C	1	0x0	Electrical Idle inference status in Recovery.Speed on a successful speed negotiation. 0 – Otherwise 1 – Event occurred. Write 1 to clear.

Field	Name	Access	Width	Reset	Description
[12]	idle_infer_rec_speed2_unsuccess	RW1C	1	0x0	Electrical Idle inference status in Recovery.Speed on an unsuccessful speed negotiation. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[11]	idle_infer_l0_to_rec_rcvr_lock	RW1C	1	0x0	Electrical Idle inference status in L0 – event causes entry into Recovery.RcvrLock. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[10:9]	Reserved	RO	2	0x0	—
[8]	speed_change_fail	RW1C	1	0x0	Speed Change Failure error indicator. 0 – Otherwise 1 – Speed change failure occurred. Write 1 to clear.
[7:2]	Reserved	RO	6	0x0	—
[1:0]	speed	RO	2	0x0	Current LTSSM Link Speed. Only link speeds supported by the core is be indicated. 0 – 2.5G 1 – 5G 2 – 8G 3 – 16G

ltssm_ltssm Register 0x84

This register set is used for LTSSM State Machine State configuration.

Table 5.21. ltssm_ltssm Register 0x84

Field	Name	Access	Width	Reset	Description
[31:20]	Reserved	RO	12	0x0	—
[19:16]	sub_state	RO	4	0x1	Current LTSSM Minor State. Encoding varies depending upon the Current LTSSM Major State. 0 – DETECT_INACTIVE 1 – DETECT_QUIET 2 – DETECT_SPD_CHG0 3 – DETECT_SPD_CHG1 4 – DETECT_ACTIVE0 5 – DETECT_ACTIVE1 6 – DETECT_ACTIVE2 7 – DETECT_P1_TO_P0 8 – DETECT_P0_TO_P1_0 9 – DETECT_P0_TO_P1_1 10 – DETECT_P0_TO_P1_2 0 – POLLING_INACTIVE 1 – POLLING_ACTIVE_ENTRY 2 – POLLING_ACTIVE 3 – POLLING_CFG 4 – POLLING_COMP 5 – POLLING_COMP_ENTRY 6 – POLLING_COMP_EIOS 7 – POLLING_COMP_EIOS_ACK 8 – POLLING_COMP_IDLE 0 – CONFIGURATION_INACTIVE 1 – CONFIGURATION_US_LW_START 2 – CONFIGURATION_US_LW_ACCEPT

Field	Name	Access	Width	Reset	Description
					3 – CONFIGURATION_US_LN_WAIT 4 - CONFIGURATION_US_LN_ACCEPT 5 – CONFIGURATION_DS_LW_START 6 – CONFIGURATION_DS_LW_ACCEPT 7 – CONFIGURATION_DS_LN_WAIT 8 – CONFIGURATION_DS_LN_ACCEPT 9 – CONFIGURATION_COMPLETE 10 – CONFIGURATION_IDLE 0 – L0_INACTIVE 1 – L0_L0 2 – L0_TX_EL_IDLE 3 – L0_TX_IDLE_MIN 0 – RECOVERY_INACTIVE 1 – RECOVERY_RCVR_LOCK 2 – RECOVERY_RCVR_CFG 3 – RECOVERY_IDLE 4 – RECOVERY_SPEED0 5 – RECOVERY_SPEED1 6 – RECOVERY_SPEED2 7 – RECOVERY_SPEED3 8 – RECOVERY_EQ_PH0 9 – RECOVERY_EQ_PH1 10 – RECOVERY_EQ_PH2 11 – RECOVERY_EQ_PH3 0 – DISABLED_INACTIVE 1 – DISABLED_0 2 – DISABLED_1 3 – DISABLED_2 4 – DISABLED_3 0 – LOOPBACK_INACTIVE 1 – LOOPBACK_ENTRY 2 – LOOPBACK_ENTRY_EXIT 3 – LOOPBACK_EIOS 4 – LOOPBACK_EIOS_ACK 5 – LOOPBACK_IDLE 6 – LOOPBACK_ACTIVE 7 – LOOPBACK_EXIT0 8 – LOOPBACK_EXIT1 0 – HOT_RESET_INACTIVE 1 – HOT_RESET_HOT_RESET 2 – HOT_RESET_ LEADER _UP 3 – HOT_RESET_ LEADER _DOWN 0 – TX_L0S_INACTIVE 1 – TX_L0S_IDLE 2 – TX_L0S_TO_L0 3 – TX_L0S_FTS0 4 – TX_L0S_FTS1 0 – L1_INACTIVE 1 – L1_IDLE 2 – L1_SUBSTATE 3 – L1_TO_L0 0 – L2_INACTIVE 1 – L2_IDLE 2 – L2_TX_WAKE0

Field	Name	Access	Width	Reset	Description
					3 – L2_TX_WAKE1 4 – L2_EXIT 5 – L2_SPEED
[15:4]	Reserved	RO	12	0x0	—
[3:0]	state	RO	4	0x0	Current LTSSM Major State 0 – DETECT 1 – POLLING 2 – CONFIGURATION 3 – L0 4 – RECOVERY 5 – DISABLED 6 – LOOPBACK 7 – HOT_RESET 8 – TX_L0S 9 – L1 10 – L2

ltssm_rx_l0s Register 0x88

This register set is used for the Rx L0s State Machine State configuration.

Table 5.22. ltssm_rx_l0s Register 0x88

Field	Name	Access	Width	Reset	Description
[31:3]	Reserved	RO	29	0x0	—
[2:0]	state	RO	3	0x0	Current LTSSM RX L0s State. 0 – RX_L0S_L0 1 – RX_L0S_ENTRY 2 – RX_L0S_IDLE 3 – RX_L0S_FTS 4 – RX_L0S_REC

l0_to_rec Register 0x8c

This register set is used to report different events causing L0 state to Recovery state transition.

Table 5.23. l0_to_rec Register 0x8c

Field	Name	Access	Width	Reset	Description
[31:15]	reserved	RO	17	0x0	—
[14]	direct_to_detect_fast	RW1C	1	0x0	Recovery is entered from L0 due to assertion of mgmt_tlb_ltssm_direct_to_detect_fast. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[13]	direct_to_recovery_ch_bond	RW1C	1	0x0	Recovery is entered from L0 due to more lane skew than the Channel Bond circuit can tolerate or is due to channel bond failing to occur within the expected timeout period. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[12]	direct_to_loopback_entry	RW1C	1	0x0	Recovery is entered from L0 due to being directed into Leader Loopback. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[11]	directed_speed_change	RW1C	1	0x0	Recovery is entered from L0 due to being directed to make a speed change. This includes the initial hardware-initiated speed change(s) which are made when first exiting Detect.Quiet to L0. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[10]	l0_to_rec_rcvr_lock_rx_ts12	RW1C	1	0x0	Recovery is entered from L0 due to receiving TS1 or TS2 ordered sets. The link partner is directing Recovery entry. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[9]	l0_to_rec_rcvr_lock_rx_8g_eie	RW1C	1	0x0	Recovery is entered from L0 due to receiving EIE ordered sets at ≥ 8G. The link partner is directing Recovery entry. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[8]	l0_to_rec_rcvr_lock_rx_infer	RW1C	1	0x0	Recovery is entered from L0 due to inferring Electrical Idle due to no SKP ordered set received in 128 µs. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[7]	direct_to_recovery_phy	RW1C	1	0x0	Recovery is entered from L0 due to receiving a burst of ~1024 clock cycles of data containing PHY errors at 2.5G or 5G. This normally occurs only when the PHY has lost lock on one or more lanes. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[6]	direct_to_recovery_frame	RW1C	1	0x0	Recovery is entered from L0 due to receiving one or more framing errors at ≥ 8G. This occurs due to Rx bit errors which are expected every few minutes at PCIe Specified BER of 10^-12. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[5]	direct_to_recovery_replay	RW1C	1	0x0	Recovery is entered from L0 due to the original and three replay TLP transmissions failing to receive ACK DLLP acknowledgment. 0 – Otherwise 1 – Event occurred. Write 1 to clear.

Field	Name	Access	Width	Reset	Description
[4]	direct_to_hot_reset	RW1C	1	0x0	Recovery is entered from L0 due to being directed into Hot Reset (Secondary Bus Reset Register). 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[3]	direct_to_disable	RW1C	1	0x0	Recovery is entered from L0 due to being directed into Disable (Link Disable Register). 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[2]	rx_l0s_direct_to_recovery	RW1C	1	0x0	Recovery is entered from L0 due to failing to receive the complete Rx_L0S FTS exit sequence within the PCIe Specification allowed timeout period. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[1]	autonomous_width_change	RW1C	1	0x0	Recovery is entered from L0 due to directed autonomous width change. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[0]	directed_retrain_link	RW1C	1	0x0	Recovery is entered from L0 due to directed retrain link (Retrain Link Register). 0 – Otherwise 1 – Event occurred. Write 1 to clear.

ltssm_rx_detect Register 0x90

This register set is used for the Receiver detection status.

Table 5.24. ltssm_rx_detect Register 0x90

Field	Name	Access	Width	Reset	Description
[31:16]	reserved	RO	16	0x0	—
[15:0]	lanes	RO	16	0x0	Per lane receiver detection status. For each lane: 0 – Unconnected 1 – Present

ltssm_configured Register 0x94

This register set is used for the Configured link status.

Table 5.25. ltssm_configured Register 0x94

Field	Name	Access	Width	Reset	Description
[31:25]	reserved	RO	7	0x0	—
[24:16]	link_num	RO	9	0x1ff	Link Number configured during LTSSM Training. link_num == 0x1FF on fundamental reset, changes to 0x1F7 (KPAD) when entering CFG_US_LW_START or CFG_DS_LW_START (the start of LTSSM Configuration), and then changes to the negotiated Link Number determined during LTSSM Configuration when the LTSSM changes from CFG_COMPLETE to CFG_IDLE. This field is provided for diagnostics.
[15:0]	lanes	RO	16	0x0	Per lane configured link status. Each lane status resets to 0. After Receiver Detection results are available, each lane status is updated to show which lanes detected receivers. After a link has been formed, each lane status is updated to show which lanes are part of the configured link. For each lane: 0 – Lane did not configure into the link. 1 – Lane configured into the link.

ltssm_direct_to_detect Register 0x98

This register set is used for the Rec Rcvr Lock to Detect controls.

Table 5.26. ltssm_direct_to_detect Register 0x98

Field	Name	Access	Width	Reset	Description
[31:16]	reserved	RO	16	0x0	—
[15]	fast	RW	9	0x0	A rising edge on this signal instructs the state machine to proceed from L0 or Recovery to Detect as quickly as possible.
[14:8]	Reserved	RO	7	0x0	—
[7:0]	timer	RW	8	0x0	This value determines the timeout delay for the state machine to proceed from Recovery Rcvr Lock to Detect when no TS sets are received. A value of 0 disables this timeout.

ltssm_equalization Register 0x9c

This register set is used for the for ≥ 8G capable cores only: LTSSM equalization status.

Table 5.27. ltssm_equalization Register 0x9c

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	fail	RO	1	0x0	Equalization Failure error indicator. 0 – Otherwise 1 – Equalization failure.

ltssm_crosslink Register 0xa0

This register set is used for the TSSM crosslink status.

Table 5.28. ltssm_crosslink Register 0xa0

Field	Name	Access	Width	Reset	Description
[31:2]	reserved	RO	30	0x0	—
[1]	ds_us_n	RO	1	0x0	Crosslink port type. When active==1, indicates which personality the port assumed during crosslink negotiation. 0 – Upstream 1 – Downstream
[0]	active	RO	1	0x0	Crosslink active indicator. 0 – Otherwise 1 – Link is operating in a crosslink configuration.

5.1.2.2. Physical Layer Status Register Set

Physical Layer Tx Underflow Error Status Register – 0xa4

Table 5.29. Physical Layer Tx Underflow Error Status Register – 0xa4

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	err_tx_pipe_underflow	RW1C	1	0x0	0 – Otherwise 1 – Physical Layer Tx data needed to be forwarded to the lanes for transmission and some, but not all lanes, were ready to accept data causing some lanes to under low. This bit stays asserted once set. Write 1 to clear.

Table 5.30 illustrates the Physical Lane RX Status Register set with its offset and register address.

Table 5.30. Physical Lane Rx Status Registers

Register Name	Offset Address	Description
pl_rx0 Register	0xa8	Lane Rx Status 0 register – TS2 and TS1 OS detection [0 to 15 bits]
pl_rx1 Register	0xac	Lane Rx Status 1 – Inverted TS2 and TS1 OS detection [0 to 15 bits]
pl_rx2 Register	0xb0	Lane Rx Status 2 – FTS and SKP OS detection
pl_rx3 Register	0xb4	Lane Rx Status 3 – EIOS detection and EIE detection
pl_rx4 Register	0xb8	Lane Rx Status 4 – Data Block is received and SDS ordered set detection

pl_rx0 Register 0xa8 – Lane Rx Status 0 Register

Table 5.31. pl_rx0 Register 0xa8 – Lane Rx Status 0 Register

Field	Name	Access	Width	Reset	Description
[31]	ts2_detect15	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[30]	ts2_detect14	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[29]	ts2_detect13	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear

Field	Name	Access	Width	Reset	Description
[28]	ts2_detect12	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[27]	ts2_detect11	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[26]	ts2_detect10	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[25]	ts2_detect9	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[24]	ts2_detect8	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[23]	ts2_detect7	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[22]	ts2_detect6	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[21]	ts2_detect5	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[20]	ts2_detect4	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[19]	ts2_detect3	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[18]	ts2_detect2	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[17]	ts2_detect1	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[16]	ts2_detect0	RW1C	1	0x0	ts2_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[15]	ts1_detect15	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[14]	ts1_detect14	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[13]	ts1_detect13	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[12]	ts1_detect12	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear

Field	Name	Access	Width	Reset	Description
[11]	ts1_detect11	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[10]	ts1_detect10	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[9]	ts1_detect9	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[8]	ts1_detect8	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[7]	ts1_detect7	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[6]	ts1_detect6	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[5]	ts1_detect5	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[4]	ts1_detect4	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[3]	ts1_detect3	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[2]	ts1_detect2	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[1]	ts1_detect1	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear
[0]	ts1_detect0	RW1C	1	0x0	ts1_detect[i] is set to 1 when a TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear

pl_rx1 Register 0xac – Lane Rx Status 1

Table 5.32. pl_rx1 Register 0xac – Lane Rx Status 1

Field	Name	Access	Width	Reset	Description
[31]	ts2i_detect15	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[30]	ts2i_detect14	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.

Field	Name	Access	Width	Reset	Description
[29]	ts2i_detect13	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[28]	ts2i_detect12	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[27]	ts2i_detect11	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[26]	ts2i_detect10	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[25]	ts2i_detect9	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[24]	ts2i_detect8	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[23]	ts2i_detect7	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[22]	ts2i_detect6	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[21]	ts2i_detect5	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[20]	ts2i_detect4	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[19]	ts2i_detect3	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[18]	ts2i_detect2	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[17]	ts2i_detect1	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.

Field	Name	Access	Width	Reset	Description
[16]	ts2i_detect0	RW1C	1	0x0	ts2i_detect[i] is set to: 1 when an inverted TS2 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[15]	ts1i_detect15	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[14]	ts1i_detect14	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[13]	ts1i_detect13	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[12]	ts1i_detect12	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[11]	ts1i_detect11	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[10]	ts1i_detect10	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[9]	ts1i_detect9	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[8]	ts1i_detect8	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[7]	ts1i_detect7	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[6]	ts1i_detect6	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[5]	ts1i_detect5	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[4]	ts1i_detect4	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.

Field	Name	Access	Width	Reset	Description
[3]	ts1i_detect3	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[2]	ts1i_detect2	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[1]	ts1i_detect1	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[0]	ts1i_detect0	RW1C	1	0x0	ts1i_detect[i] is set to: 1 when an inverted TS1 ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.

pl_rx2 Register 0xb0 – Lane Rx Status 2

Table 5.33. pl_rx2 Register 0xb0 – Lane Rx Status 2

Field	Name	Access	Width	Reset	Description
[31]	fts_detect15	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[30]	fts_detect14	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[29]	fts_detect13	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[28]	fts_detect12	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[27]	fts_detect11	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[26]	fts_detect10	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[25]	fts_detect9	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.

Field	Name	Access	Width	Reset	Description
[24]	fts_detect8	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[23]	fts_detect7	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[22]	fts_detect6	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[21]	fts_detect5	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[20]	fts_detect4	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[19]	fts_detect3	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[18]	fts_detect2	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[17]	fts_detect1	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[16]	fts_detect0	RW1C	1	0x0	fts_detect[i] is set to: 1 when the FTS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[15]	skp_detect15	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[14]	skp_detect14	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[13]	skp_detect13	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.

Field	Name	Access	Width	Reset	Description
[12]	skp_detect12	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[11]	skp_detect11	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[10]	skp_detect10	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[9]	skp_detect9	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[8]	skp_detect8	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[7]	skp_detect7	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[6]	skp_detect6	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[5]	skp_detect5	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[4]	skp_detect4	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[3]	skp_detect3	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[2]	skp_detect2	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[1]	skp_detect1	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[0]	skp_detect0	RW1C	1	0x0	skp_detect[i] is set to: 1 when a SKP ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.

pl_rx3 Register 0xb4 – Lane Rx Status 3

Table 5.34. pl_rx3 Register 0xb4 – Lane Rx Status 3

Field	Name	Access	Width	Reset	Description
[31]	eie_detect15	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[30]	eie_detect14	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[29]	eie_detect13	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[28]	eie_detect12	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[27]	eie_detect11	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[26]	eie_detect10	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[25]	eie_detect9	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[24]	eie_detect8	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[23]	eie_detect7	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[22]	eie_detect6	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[21]	eie_detect5	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[20]	eie_detect4	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.

Field	Name	Access	Width	Reset	Description
[19]	eie_detect3	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[18]	eie_detect2	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[17]	eie_detect1	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[16]	eie_detect0	RW1C	1	0x0	eie_detect[i] is set to: 1 when an EIE ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[15]	eios_detect15	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[14]	eios_detect14	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[13]	eios_detect13	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[12]	eios_detect12	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[11]	eios_detect11	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[10]	eios_detect10	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[9]	eios_detect9	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[8]	eios_detect8	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[7]	eios_detect7	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.

Field	Name	Access	Width	Reset	Description
[6]	eios_detect6	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[5]	eios_detect5	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[4]	eios_detect4	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[3]	eios_detect3	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[2]	eios_detect2	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[1]	eios_detect1	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[0]	eios_detect0	RW1C	1	0x0	eios_detect[i] is set to: 1 when an EIOS ordered set is received on Lane[i]. 0 otherwise. This bit stays asserted once set. Write 1 to clear.

pl_rx4 Register 0xb8 – Lane Rx Status 4

Table 5.35. pl_rx4 Register 0xb8 – Lane Rx Status 4

Field	Name	Access	Width	Reset	Description
[31]	data_detect15	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[30]	data_detect14	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[29]	data_detect13	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[28]	data_detect12	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.

Field	Name	Access	Width	Reset	Description
[27]	data_detect11	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[26]	data_detect10	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[25]	data_detect9	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[24]	data_detect8	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[23]	data_detect7	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[22]	data_detect6	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[21]	data_detect5	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[20]	data_detect4	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[19]	data_detect3	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[18]	data_detect2	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[17]	data_detect1	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[16]	data_detect0	RW1C	1	0x0	data_detect[i] is set to: 1 when a Data Block is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[15]	sds_detect15	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.

Field	Name	Access	Width	Reset	Description
[14]	sds_detect14	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[13]	sds_detect13	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[12]	sds_detect12	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[11]	sds_detect11	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[10]	sds_detect10	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[9]	sds_detect9	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[8]	sds_detect8	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[7]	sds_detect7	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[6]	sds_detect6	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[5]	sds_detect5	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[4]	sds_detect4	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[3]	sds_detect3	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[2]	sds_detect2	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.

Field	Name	Access	Width	Reset	Description
[1]	sds_detect1	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.
[0]	sds_detect0	RW1C	1	0x0	sds_detect[i] is set to: 1 when a SDS ordered set is received on Lane[i] at 8G. 0 otherwise. This bit stays asserted once set. Write 1 to clear.

5.1.2.3. Debug Register Set

debugself_crosslink Register 0xc0

This register set is used for debug to allow Rx detection when Tx is externally looped back to Rx.

Table 5.36. debugself_crosslink Register 0xc0

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	enable	RW	1	0x0	0 – Otherwise 1 – For debug use only, configure LTSSM so that it links with itself when core Tx is externally looped back to core Rx.

debug_rx_det Register 0xc4

This register set is used for the LTSSM receiver detection bypass configuration.

Table 5.37. debug_rx_det Register 0xc4

Field	Name	Access	Width	Reset	Description
[31:17]	reserved	RO	15	0x0	—
[16]	inhibit	RW	1	0x0	Link receiver detection inhibit. 0 – Perform receiver detection and use result to determine whether to include/exclude lanes from the link. 1 – Skip receiver detection and assume receivers are not present on all lanes.
[15:1]	reserved	RO	15	0x0	—
[0]	bypass	RW	1	0x0	Link receiver detection bypass. If both bypass and inhibit are asserted, bypass takes precedence. 0 – Perform receiver detection and use result to determine whether to include/exclude lanes from the link. 1 – Skip receiver detection and assume receivers are present on all lanes.

debug_force_tx Register 0xc8

This register set is used for debug using TX PIPE signals.

Table 5.38. debug_force_tx Register 0xc8

Field	Name	Access	Width	Reset	Description
[31:10]	reserved	RO	22	0x0	—
[9]	deemph_5g_enable	RW	1	0x0	For 5G capable cores only: Force pipe_tx_deemph at 5G enable. 0 – Disable. 1 – Enable. Force phy_tx_deemph at 5G speed to the value specified by deemph_5g_6db_3_5db_n. The force is applied at 5G speed except during Polling.Compliance, where for compatibility with PCI SIG Workshop Electrical Testing, the force is not applied.
[8]	deemph_5g_3_5db_6db_n	RW	1	0x0	For 5G capable cores only: Force pipe_tx_deemph at 5G value. 0 – -6dB 1 – -3.5dB
[7:4]	reserved	RO	4	0x0	—
[3]	margin_enable	RW	1	0x0	Force pipe_tx_margin enable. 0 – Drive pipe_tx_margin per PCIe Specification. 1 – Drive pipe_tx_margin to value.
[2:0]	margin_value	RW	3	0x0	Force pipe_tx_margin Value.

debug_direct_scramble_off Register 0xcc

This register set is used for scrambling disable control for 2.5G and 5G data rate.

Table 5.39. debug_direct_scramble_off Register 0xcc

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	enable	RW	1	0x0	LTSSM direct scrambling disabled at 2.5G and 5G. 0 – Otherwise 1 – Direct to disable scrambling at 2.5G and 5G during Configuration.Complete.

debug_force_scramble_off_fast Register 0xd0

This register set is used for scrambling disable control for 8G data rate.

Table 5.40. debug_force_scramble_off_fast Register 0xd0

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	enable	RW	1	0x0	LTSSM force scrambling disabled at ≥ 8G. 0 – Otherwise 1 – Disable scrambling at ≥ 8G. Only works for simulation when link partner is also disabling scrambling. Cannot be set for hardware because disabling scrambling at ≥ 8G is not permitted per PCIe Specification.

balign Register 0xd4

This register set is used for pipe_block_align_control generation options for 8G data rate. It is not recommended to change the default values of this register.

Table 5.41. balign Register 0xd4

Field	Name	Access	Width	Reset	Description
[31]	state_data_n	RW	1	0x0	When generating pipe_block_align_control (which may be used by some PHY to aid in acquiring ≥ 8G block alignment), select between the LTSSM State Algorithm and the Rx Data Observation Algorithm. 0 – Use Rx Data Observation Algorithm 1 – Use the LTSSM State Algorithm
[30:6]	reserved	RO	25	0x0	—
[5]	exclude_loopback_master	RW	1	0x0	When generating pipe_block_align_control through the LTSSM State algorithm, exclude/include the Loopback Leader state in driving pipe_block_align_control to 0. 0 – Include 1 – Exclude
[4]	exclude_cfg_complete	RW	1	0x0	When generating pipe_block_align_control through the LTSSM State algorithm, exclude/include the CFG_COMPLETE LTSSM state in driving pipe_block_align_control to 0. pipe_block_align_control is only driven to 0 during CFG_COMPLETE after receiving the required Rx exit criteria to state CFG_IDLE. Due to this requirement, the core may need to stay in CFG_COMPLETE for a while after receiving the Rx exit criteria to meet the required Tx exit criteria. 0 – Include 1 – Exclude
[3]	exclude_cfg_idle	RW	1	0x0	When generating pipe_block_align_control through the LTSSM State algorithm, exclude/include the CFG_IDLE LTSSM state in driving pipe_block_align_control to 0. 0 – Include 1 – Exclude
[2]	exclude_rec_rcvr_cfg	RW	1	0x0	When generating pipe_block_align_control through the LTSSM State algorithm, exclude/include the REC_RCVR_CFG LTSSM state in driving pipe_block_align_control to 0. pipe_block_align_control is only driven to 0 during REC_RCVR_CFG after receiving the required Rx exit criteria to state REC_IDLE. Due to the PCIe Specification, the core may need to stay in REC_RCVR_CFG for a while after receiving the Rx exit criteria to also meet the required Tx exit criteria. 0 – Include 1 – Exclude
[1]	exclude_rec_idle	RW	1	0x0	When generating pipe_block_align_control through the LTSSM State algorithm, exclude/include the REC_IDLE LTSSM state in driving pipe_block_align_control to 0. 0 – Includee 1 – Exclude
[0]	exclude_l0	RW	1	0x0	When generating pipe_block_align_control through the LTSSM State algorithm, exclude/include the L0 and TX_L0s LTSSM states in driving pipe_block_align_control to 0. 0 – Include 1 – Exclude

debug_pipe_rx Register 0xe0

This register set is used for the PIPE Interface Debug status.

Table 5.42. debug_pipe_rx Register 0xe0

Field	Name	Access	Width	Reset	Description
[31:16]	polarity	RO	16	0x0	PHY PIPE Interface pipe_rx_polarity current value. For each lane: 0 – Otherwise 1 – PHY lane has been instructed to invert its receiver polarity to compensate for serial rx_p and rx_n being swapped.
[15:0]	valid	RO	16	0x0	PHY PIPE Interface pipe_rx_valid current value. For each lane: 0 – Otherwise 1 – PHY lane is locked to data stream

debug_direct_to_loopback Register 0x100

This register set is used to enable LTSSM Leader loopback.

Table 5.43. debug_direct_to_loopback Register 0x100

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	enable	RW	1	0x0	LTSSM leader loopback enable. 0 – Otherwise. 1 – Direct LTSSM to Loopback.Leader. Before this field is set to 1, all relevant regsiters containing Leader Loopback control options must be set to the desired values. When mgmt_tlb_debug_direct_to_loopback == 1 no Leader Loopback control options may be changed.

debug_loopback_control Register 0x104

This register set is used enable different control features related to loopback for 2.5G and 5G data rates.

Table 5.44. debug_loopback_control Register 0x104

Field	Name	Access	Width	Reset	Description
[31:28]	inject_err_lane_select	RW	4	0x0	Lane selection to inject error in Loopback. Only lanes configured by the core may be programmed. 0 = Lane 0. 15 = Lane 15.
[27]	inject_rx_2bit_data_err	RW	1	0x0	When enabled during loopback, the rising edge of inject_rx_2bit_data_err bit injects a back-to-back error on the received loopback data. This simulates the PHY losing lock and increments the counter by one, and the Loopback Leader restarts the loopback pattern so that the PHY can recover symbol lock.
[26]	inject_rx_1bit_data_err	RW	1	0x0	When enabled during loopback, the rising edge of inject_rx_1bit_data_err bit injects a single clk error on the received loopback data. This causes the error count to increment by 1 for each received data byte.
[25]	inject_rx_valid_err	RW	1	0x0	When enabled during loopback, the rising edge of inject_rx_valid_err bit injects a single clk error on the received PIPE PHY interface phy_rx_valid signal. This simulates the PHY losing lock during Loopback Leader operation which causes the error count to increment by one, and the Loopback Leader restarts the loopback pattern so that the PHY can recover symbol lock.

Field	Name	Access	Width	Reset	Description
[24]	inject_rx_skp_err	RW	1	0x0	When enabled during loopback, the rising edge of inject_rx_skp_err bit injects a single clk error on the next received SKP Ordered Set. When a SKP Ordered Set is corrupted, the lane's RX descrambling LFSR goes out of sync with the transmitter lane's scrambling LFSR causing all the subsequent data checks to fail. This simulates the PHY losing lock and increments the counter by one, and the Loopback Leader restarts the loopback pattern so that the PHY can recover symbol lock.
[23:19]	reserved	RO	5	0x0	—
[18:16]	pattern	RW	3	0x0	Loopback data pattern. 0 – Unscrambled PRBS31 Polynomial Pattern using Galois implementation with non-inverted output. The polynomial representation is G(x) = X31 + X28 +1.
[15:9]	reserved	RO	7	0x0	—
[8]	tx_comp_receive	RW	1	0x0	Loopback compliance receive behavior. 0 – Loopback Leader does not assert Compliance Receive (recommended default) 1 – Loopback Leader asserts Compliance Receive in TS sets transmitted during Loopback Entry
[7:2]	reserved	RO	6	0x0	—
[1:0]	speed	RW	2	0x0	Desired speed in loopback. Only speeds supported by the core may be programmed. A speed change is only implemented if Loopback is entered from Configuration; if entered from Recovery, the speed is not changed. 0 – 2.5G 1 – 5G 2 – 8G 3 – 16G

debug_loopback_master_5g Register 0x108

This register set is used to select Deemphasis values by loopback Leader for 2.5G and 5G data rates.

Table 5.45. debug_loopback_master_5g Register 0x108

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	deemph	RW	1	0x0	Select Deemphasis value used by Loopback Leader when Loopback.Active occurs at 5G data rate. 0 – -6.0dB 1 – -3.5dB

debug_loopback_slave_5g Register 0x10c

This register set is used to select Deemphasis value transmitted in TS sets during loopback for 2.5G and 5G data rates.

Table 5.46. debug_loopback_slave_5g Register 0x10c

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	deemph	RW	1	0x0	Select Deemphasis value transmitted in TS sets for the Follower to use when Loopback.Active occurs at 5G data rate. 0 – -6.0dB 1 – -3.5dB

debug_loopback_master_8g_deemph Register 0x110

This register set is used to select TX Preset related coefficients for 8G data rate for loopback Leader.

Table 5.47. debug_loopback_master_8g_deemph Register 0x110

Field	Name	Access	Width	Reset	Description
[31]	coef_en	RW	1	0x0	Leader coefficient enable. 0 – Otherwise 1 – Direct local transmitter to use coef == mgmt_tlb_debug_loopback_master_8g_deemph_coef when acting as a Loopback Leader in Loopback.Active.
[30:26]	reserved	RO	5	0x0	—
[25:8]	coef	RW	18	0x0	Coefficients to use when mgmt_tlb_debug_loopback_master_8g_deemph_coef_en==1. The coefficients must be a valid set of coefficients, considering the leader's FS and LF values and PCI Express coefficient rules, or the results are undefined. Coefficient mapping: [17:12]==Post-Cursor [11:6]==Cursor [5:0]==Pre-Cursor
[7]	preset_en	RW	1	0x0	Leader preset enable. 0 – Otherwise 1 – Direct local transmitter to use preset == mgmt_tlb_debug_loopback_master_8g_deemph_preset when acting as a Loopback Leader in Loopback.Active
[6:4]	reserved	RO	3	0x0	—
[3:0]	preset	RW	4	0x0	Preset to use when mgmt_tlb_debug_loopback_master_8g_deemph_preset_en==1. Must be a valid preset in range 0x0 to 0xa or the result is undefined.

debug_loopback_slave_8g_deemph Register 0x114

This register set is used to select TX Preset related coefficients for 8G data rate for loopback Follower.

Table 5.48. debug_loopback_slave_8g_deemph Register 0x114

Field	Name	Access	Width	Reset	Description
[31]	coef_en	RW	1	0x0	Follower coefficient enable. 0 – Otherwise 1 – Direct follower's transmitter to use coef == mgmt_tlb_debug_loopback_slave_8g_deemph_coef through TS sets transmitted while directing slave to Loopback.
[30:26]	reserved	RO	5	0x0	—
[25:8]	coef	RW	18	0x0	Coefficients to use when mgmt_tlb_debug_loopback_slave_8g_deemph_coef_en==1. The coefficients must be a valid set of coefficients, considering the follower's FS and LF values and PCI Express coefficient rules, or the follower rejects them. Coefficient mapping: [17:12]==Post-Cursor [11:6]==Cursor [5:0]==Pre-Cursor.
[7]	preset_en	RW	1	0x0	Follower preset enable. 0 – Otherwise 1 – Direct follower's transmitter to use preset == mgmt_tlb_debug_loopback_slave_8g_deemph_preset through TS sets transmitted while directing follower to Loopback.
[6:4]	hint	RW	3	0x0	Follower Rx Hint transmitted in EQ TS1 sets when mgmt_tlb_debug_loopback_slave_8g_deemph_preset_en == 1. PCIe Specification does not indicate what to transmit for RxHint when requesting a Preset through 2.5/5G EQTS1 sets, the follower likely ignores whatever is transmitted in this field.
[3:0]	preset	RW	4	0x0	Follower preset enable. 0 – Otherwise 1 – Direct follower's transmitter to use preset == mgmt_tlb_debug_loopback_slave_8g_deemph_preset through TS sets transmitted while directing follower to Loopback.

debug_direct_to_loopback_status Register 0x118

This register set is used for the Leader loopback status.

Table 5.49. debug_direct_to_loopback_status Register 0x118

Field	Name	Access	Width	Reset	Description
[31:16]	sync	RO	16	0x0	Loopback per lane sync to data pattern indicator. For each lane: 0 – Not locked to loopback pattern. 1 – Locked to loopback pattern.
[15:1]	reserved	RO	15	0x0	—
[0]	cfg_entry	RO	1	0x0	Loopback entered from Configuration or Recovery indicator. 0 – Loopback entry is from Recovery. 1 – Loopback entry is from Configuration.

debug_loopback_err_reset Register 0x11c

This register set is used for the Leader loopback error reset.

Table 5.50. debug_loopback_err_reset Register 0x11c

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	enable	RW	1	0x0	Loopback error counter reset. 0 – Leader Loopback error count increments as errors are detected during Leader Loopback – saturating at maximum value. 1 – Reset the leader loopback error count on all lanes to 0x0. The reset stays in force for as long as mgmt_tlb_debug_loopback_err_reset_enable remains at 1.

debug_loopback_err Register 0x120

This register set is used for the Leader loopback error count.

Table 5.51. debug_loopback_err Register 0x120

Field	Name	Access	Width	Reset	Description
[255:0]	count	RO	256	0x0	Loopback per lane error count – 16 bits per lane. Errors are counted only after the lane is locked to the loopback pattern.

5.1.2.4. Physical control Register Set

phy_control Register 0x140

This register set is used for LTSSM PIPE Interface configuration for 2.5G and 5G data rates.

Table 5.52. phy_control Register 0x140

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	pipe_tx_swing	RW	1	0x0	Directly controls the value of pipe_tx_swing which sets PHY 2.5G/5G Transmitter Amplitude. 0 – Full Swing Full Swing is required for most applications. 1 – Reduced Swing Reduced Swing is useful to support low power form factors which encourage or require reduced transmitter amplitudes.

phy_control_8g Register 0x144

This register set is used for LTSSM 8G PIPE Interface configuration for 8G data rate.

Table 5.53. phy_control_8g Register 0x144

Field	Name	Access	Width	Reset	Description
[31:2]	reserved	RO	30	0x0	—
[1]	no_tx_idle_delay	RW	1	0x0	Controls the deassertion of pipe_tx_elec_idle to the PHY when operating with 128b130b encoding. 0 – Deassert pipe_tx_elec_idle at the next data_valid gap. 1 – Deassert pipe_tx_elec_idle at the next ordered set boundary.

Field	Name	Access	Width	Reset	Description
[0]	double_tx_data_valid	RW	1	0x0	Controls the number of consecutive 8G pipe_tx_data_valid deassertions used when compensating for 128b130b encoding differences. 0 – Deassert pipe_tx_data_valid for 1 clock every 64 clocks – the required value for the majority of PHY. 1 – Deassert pipe_tx_data_valid for 2 back-back clocks every 128 clocks, simplifies connecting 8G PHY which have double the per lane width of the controller.

phy_eq_tx_override Register 0x148 F

This register set is used for local PHY transmitter FS/LF override for 8G data rate.

Table 5.54. phy_eq_tx_override Register 0x148 F

Field	Name	Access	Width	Reset	Description
[31:30]	reserved	RO	2	0x0	—
[29:24]	fs	RW	6	0x30	Local PHY Transmitter: Full Scale Value. When 177gmt._tlb_phy_eq_tx_override_enable == 1, 177gmt._tlb_phy_eq_tx_override_fs is used for the local PHY Full Scale (FS) value, otherwise PIPE PHY interface port pipe_local_fs is used.
[23:22]	reserved	RO	2	0x0	—
[21:16]	lf	RW	6	0x8	Local PHY Transmitter: Low Frequency Value. When 177gmt._tlb_phy_eq_tx_override_enable == 1, 177gmt._tlb_phy_eq_tx_override_lf is used for the local PHY Low Frequency (LF) value, otherwise PIPE PHY interface port pipe_local_fs is used.
[15:1]	reserved	RO	15	0x0	—
[0]	enable	RW	1	0x0	Controls whether 177gmt._tlb_phy_eq_tx_override_fs and 177gmt._tlb_phy_eq_tx_override_lf or pipe_local_fs and pipe_local_lf are used to determine the FS and LF values of the local PHY. 0 – Use pipe_local_fs and pipe_local_lf. 1 – Use 177gmt._tlb_phy_eq_tx_override_fs and 177gmt._tlb_phy_eq_tx_override_lf.

phy_eq_tx_max Register 0x14c

This register set is used to specify local PHY maximum allowed coefficient values for 8G data rate.

Table 5.55. phy_eq_tx_max Register 0x14c

Field	Name	Access	Width	Reset	Description
[31:30]	reserved	RO	2	0x0	—
[29:24]	pre	RW	6	0x0	Local PHY transmitter maximum pre-cursor[5:0] coefficient value. If a coefficient request exceeds mgmt_tlb_phy_eq_tx_max_pre, the coefficient is limited to mgmt_tlb_phy_eq_tx_max_pre before being passed to the PHY.
[23:22]	reserved	RO	2	0x0	—

Field	Name	Access	Width	Reset	Description
[21:16]	post	RW	6	0x0	Local PHY transmitter maximum post-cursor [5:0] coefficient value. If a coefficient request exceeds mgmt_tlb_phy_eq_tx_max_post, then the coefficient is limited to mgmt_tlb_phy_eq_tx_max_post before being passed to the PHY.
[15:0]	reserved	RO	16	0x0	—

phy_eq_tx_force Register 0x150

This register set is used to force PHY TX Deemphasis configuration for 8G data rate. This register forces the local PHY TX Deemphasis instead of allowing the link partner to determine the TX Deemphasis during Equalization.

Table 5.56. phy_eq_tx_force Register 0x150

Field	Name	Access	Width	Reset	Description
[31:26]	reserved	RO	6	0x0	—
[25:8]	coef	RW	18	0x0	Coefficients to use when coef_enable==1
[7:4]	preset	RW	4	0x0	Preset to use when preset_enable==1
[3:2]	reserved	RO	2	0x0	—
[1]	coef_enable	RW	1	0x0	Force local PHY pipe_tx_deemph Tx De-Emphasis port. 0 – Determine local PHY ≥ 8G pipe_tx_deemph per PCIe Specification. 1 – Force all local PHY lanes' pipe_tx_deemph output port at ≥ 8G to the coefficients specified by coef. This setting is not PCIe compliant and is intended for debug only.
[0]	preset_enable	RW	1	0x0	Force remote PHY transmitter Tx De-Emphasis to the specified Preset during Equalization Phase 2/3. 0 – Normal operation. 1 – For Figure of Merit Equalization, override the standard 8G Equalization coefficient selection methods and force the core to use the Preset Equalization method with only one Preset == preset. Two total Rx Eq Evaluations are performed, 1 Trial + 1 Final, both using Preset == preset. For Up/Down Equalization, force Equalization to start requesting the link partner change to Preset = preset.

phy_preset_to_coef_conv_control Register 0x15c

This register set is used for local PHY TX preset to coefficient conversion configuration for 8G data rate.

Table 5.57. phy_preset_to_coef_conv_control Register 0x15c

Field	Name	Access	Width	Reset	Description
[31:2]	reserved	RO	30	0x0	—
[1:0]	conv_method	RW	2	0x0	Local PHY Transmitter Preset Conversion Method. 0 – Compute using the coefficients from PCIe Specification. PCIe Preset Table 4.3.5.2.2. Tx Equalization Presets. 1 – Lookup table specified by mgmt_tlb_phy_preset_conv_tab_post and mgmt_tlb_phy_preset_conv_tab_pre. 2 – Lookup table obtained from the PIPE PHY using the pipe_local_get_* PIPE interface ports.

phy_preset_conv_tab_pre Register 0x160

This register set is used for pre-cursor coefficients configuration for 8G data rate.

Table 5.58. phy_preset_conv_tab_pre Register 0x160

Field	Name	Access	Width	Reset	Description
[127:66]	reserved	RO	62	0x0	—
[65:0]	coef	RW	66	0xc20c2040000000	For ≥ 8G capable cores only: Pre-cursor table – 6-bit pre-cursor coefficients for each of the 11 possible preset to coefficient conversions packed back-back. A particular preset pre-cursor entry[i] is accessed as [(i*6)+5:(i*6)].

phy_preset_conv_tab_post Register 0x170

This register set is used for post-cursor coefficients configuration for 8G data rate.

Table 5.59. phy_preset_conv_tab_post Register 0x170

Field	Name	Access	Width	Reset	Description
[127:66]	reserved	RO	62	0x0	—
[65:0]	coef	RW	66	0xc20c2040000000	For ≥ 8G capable cores only: Post-cursor table – 6-bit post-cursor coefficients for each of the 11 possible preset to coefficient conversions packed back-back. A particular preset post-cursor entry[i] is accessed as [(i*6)+5:(i*6)].

5.1.2.5. Equalization Configuration Register Set

eq_control Register 0x180

This register set is used for upstream and downstream port preset configuration for 8G data rate.

Table 5.60. eq_control Register 0x180

Field	Name	Access	Width	Reset	Description
[31]	reserved	RO	1	0x0	—
[30:28]	us_port_rx_preset_hint	RW	3	0x2	Rx Preset Hint value that is requested for the Upstream Port to use during the initial stage of Equalization (value transmitted by Downstream Ports in EQ TS2 Ordered Sets) Upstream Ports (Endpoints) do not use this field.
[27:24]	us_port_tx_preset	RW	4	0x4	Tx Preset value that is requested for the Upstream Port to use during the initial stage of Equalization (value transmitted by Downstream Ports in EQ TS2 Ordered Sets) Upstream Ports (Endpoints) do not use this field.
[23]	reserved	RO	1	0x0	—
[22:20]	ds_port_rx_preset_hint	RW	3	0x1	Downstream Port Rx Preset Hint used during initial stage of Equalization. Upstream Ports (Endpoints) uses the value requested by the link partner. If no value is provided by the link partner, Upstream Ports (Endpoints) use this value.

Field	Name	Access	Width	Reset	Description
[19:16]	ds_port_tx_preset	RW	4	0x3	Downstream Port Tx Preset used during initial stage of Equalization. Upstream Ports (Endpoints) uses the value requested by the link partner. If the value requested by the link partner is illegal, or no value is requested by the link partner, Upstream Ports (Endpoints) use this value.
[15:2]	reserved	RO	14	0x0	—
[1]	reset_eieos_interval_count	RW	1	0x0	Reset_EIEOS_Interval_Count :- value transmitted for Reset EIEOS Interval Count in TS1/2 Ordered Set transmissions during appropriate Recovery.Equalization states.
[0]	downstream_eq_skip_phase_2_3	RW	1	0x0	Downstream_Eq_Skip_Phase_2_3 may be set in simulation for Root Port cores to speed simulation since the time-consuming portions of Equalization (Phase 2 and Phase 3) are skipped. 0 – Normal operation (perform all 4 equalization phases) 1 – Skip Equalization Phase 2 and Phase 3 (it is known that full equalization is unnecessary)

eq_ts_control Register 0x184

This register set is used for the Equalization reduced swing configuration for 8G data rate.

Table 5.61. eq_ts_control Register 0x184

Field	Name	Access	Width	Reset	Description
[31:24]	reserved	RO	8	0x0	—
[23:16]	rx_eq_resp_wait	RW	8	0x2	This register determines the number of microseconds the core waits after making an equalization remote PHY Tx coefficient change request before timing out and giving up on getting a TS response from the link partner for that request. This timeout is only used if the link partner fails to acknowledge the request within the timeout window. A timeout may occur, for example, if the local PHY receiver is unable to recover the receive data stream after the link partner changes to the new remote Tx coefficients that were requested. The timeout value is set to (1.024 × rx_eq_resp_wait) µs. 0 is a special value that selects (1.024 × 8) µs. PHY use EIEOS Rx reception at ≥ 8G and COM symbol reception at ≤ 5G to lock to the data stream. For the PHY to have a good opportunity to recover the data stream, the timeout value chosen must be large enough to include at least several of the Rx OS that the PHY needs to lock to the data stream. At ≥ 8G, 1 EIE OS is received every 32 TS OS so 1 EIE OS is received every 33 OS with OS size == 16 symbols. At ≤ 5G a COM symbol is received at the beginning of every OS. rx_eq_resp_wait==8 (8.192 µs) is recommended for most PHY. At 16G, 8.192 µs allows for ~31 EIE OS (8.192 µs / 264 ns) to be received before the timeout occurs. At 8G, 8.192 µs allows for ~15.5 EIE OS (8.192 µs / 528 ns per 1EIE+32TS) to be received before the timeout occurs. At 5G, 8.192 µs allows for ~256 COM symbols (8.192 µs / 32 ns per TS OS) to be received before the timeout occurs. At 2.5G, 8.192 µs allows for ~128 COM symbols (8.192 µs / 64 ns per TS OS) to be received before the timeout occurs.

Field	Name	Access	Width	Reset	Description
[15:8]	ts1_ack_delay	RW	8	0x1f	Defines how long the upstream port (Phase 2) or downstream port (Phase 3) waits after requesting new coefficients/presets before looking for incoming EQ TS1 sets from the remote link partner. This delay by specification should be set to the round-trip delay to the remote link partner (including logic delays in the requesting port) + 500 ns. The delay value used = (eq_ts1_ack_delay [7:0] × 16) + 500 ns. However, 0 is a special value that selects 4.596 microseconds.
[7:6]	request_eq_max_count	RW	2	0x2	Maximum times Request Equalization bit is set in Recovery.RcvrCfg. When set to 2'b00, selects infinite times.
[5]	tx_eq_eval_cnt_sel	RW	1	0x0	Determines the number of clock cycles to wait after the first lane receives an Equalization Tx De-emphasis change request from the link partner until all lanes transmit a change response. The wait time must include the worst-case lane skew that can exist on the lanes as well as time to complete the coefficient computations for the new request. 1 == Wait 127 clocks (conservative). 0 == Wait 8 clocks plus 64 symbols which is: 72 clocks for 8-bit per lane PHY, 40 clocks for 16-bit per lane PHY, or 24 clocks for 32-bit per lane PHY.
[4]	skip_final_coef_check	RW	1	0x0	When set to 1, the Upstream Port skips the check in in the Recovery.RcvrLock state after RX Equalization, which compares the TS sets coefficient/preset data with the last requested coefficients/preset from Phase 2 of Equalization. This exception is required for some non-compliant Downstream Port devices.
[3]	ts1_ack_block_use_preset	RW	1	0x1	When set to 1, the *use preset* bit is always forced to 0 in all EQ TS1 ordered sets transmitted in RX Equalization.
[2]	ts1_ack_mask_use_preset	RW	1	0x1	When set to 1, ignores the state of the use preset bit in incoming EQ TS1 sets when in Phase 2 (upstream port) or Phase 3 (downstream port). When set to 0, the use_preset bit is checked to make sure it matches the setting used in the EQ TS1 sets sent to request remote transmitter settings. In either setting, the preset value or coefficient values are compared as required by the PCIe 3.0 Specification. The value of this bit MUST be 1 for proper PCIe operation.
[1]	early_rx_eval	RW	1	0x0	When set to 1, the local PHY is told to evaluate the RX serial signals (using RxEval) BEFORE checking incoming TS1 sets for acknowledgment after each new request. When 0, the local PHY is told to evaluate the RX serial signals (using RxEval) AFTER checking incoming TS1 sets for acknowledgment after each new request. The 0 state is the default recommended state.
[0]	no_remote_change	RW	1	0x0	When set to 1, the equalization algorithm monitors the advertised coefficients from the Link Partner before equalization starts, and requests the same coefficients when performing equalization, so that the link partner does not change the TX coefficients during equalization. This is primarily for debugging purposes.

eq_reduced_swing Register 0x188

This register set is used for the Equalization reduced swing configuration for 8G data rate.

Table 5.62. eq_reduced_swing Register 0x188

Field	Name	Access	Width	Reset	Description
[31:19]	reserved	RO	13	0x0	—
[18:8]	preset_reject	RW	11	0x0	Specifies which presets (if any) are rejected if requested by the remote link partner during 8G RX Equalization. A request to use preset[i] is rejected if mgmt_tlb_eq_reduced_swing_en==1 and mgmt_tlb_eq_reduced_swing_preset_reject[i]==1 and is otherwise accepted. Per PCIe Specification, all preset requests with a valid preset value (0x0 to 0xA) must be accepted with Full Swing, but selected presets may be rejected when implementing reduced swing.
[7:1]	reserved	RO	7	0x0	—
[0]	en	RW	1	0x0	Reduced swing support enable. When mgmt_tlb_eq_reduced_swing_en==1, a request to transmit with preset[i] is rejected as illegal if mgmt_tlb_eq_reduced_swing_preset_reject[i] == 1. Only affects the acceptance/rejection of preset requests. It is also necessary to use mgmt_tlb_phy_control_pipe_tx_swing to configure the PHY transmitter for reduced swing.

eq_method Register 0x1bc

This register set is used to select Equalization method for 8G data rate.

Table 5.63. eq_method Register 0x1bc

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	select_dir_fom_n	RW	1	0x0	The core implements two primary methods: Figure of Merit and Up/Down. This register chooses which major algorithm is used. Each major algorithm has its own CSR registers for control/status Primary Equalization Method Selection. 0 – Use Figure of Merit Equalization Methods 1 – Use Up/Down Equalization Methods

eq_fmerit_control Register 0x1c0

This register set is used for Equalization Figure of Merit Method configuration for 8G data rate.

Table 5.64. eq_fmerit_control Register 0x1c0

Field	Name	Access	Width	Reset	Description
[31:24]	req_feedback	RW	8	0x80	When phy_eq_rx_eval_f_merit ≥ eq_req_feedback, the link is BER 10^12 or better.
[23:2]	reserved	RO	22	0x0	—

Field	Name	Access	Width	Reset	Description
[1:0]	method	RW	2	0x3	Equalization Method Selection. 0 – Step through PCIe-defined Tx Presets 1 – Evenly step through the coefficients range 2 – Step through the user-provided coefficient table 3 – Step through the user-provided coefficient table with adaptive coefficient selection

eq_preset_method_control Register 0x1c4

This register set is used for Equalization Figure of Merit Preset Method configuration for 8G data rate.

Table 5.65. eq_preset_method_control Register 0x1c4

Field	Name	Access	Width	Reset	Description
[31:12]	reserved	RO	20	0x0	—
[11:8]	addr_limit	RW	4	0x4	Last preset to use. The Preset Algorithm steps through PCIe-Specification-defined Tx Equalization Presets from 0 to eq_preset_addr_limit; eq_preset_addr_limit has a maximum value of 9 (step through presets 0-9).
[7:1]	reserved	RO	7	0x0	—
[0]	use_coef	RW	1	0x0	Controls whether the presets are communicated to the remote device using (1) the associated coefficient values or (0) the Preset value. 1 – The LTSSM requests coefficient values that are equivalent to the desired presets in the C-1, C0, and C+1 fields of TS1 Ordered Sets. 0 – LTSSM requests the desired presets using the Use Preset and Tx Preset fields of TS1 Ordered Sets.

eq_alg_method_control Register 0x1c8

This register set is used for Equalization Figure of Merit Algorithm Method configuration for 8G data rate.

Table 5.66. eq_alg_method_control Register 0x1c8

Field	Name	Access	Width	Reset	Description
[31:30]	reserved	RO	2	0x0	—
[29:24]	post_cursor_step_size	RW	6	0x8	Step size to use when walking through Post-Cursor coefficient values.
[23:22]	reserved	RO	2	0x0	—
[21:16]	pre_cursor_step_size	RW	6	0x4	Step size to use when walking through Pre-Cursor coefficient values.
[15:14]	reserved	RO	2	0x0	—
[13:8]	post_cursor_limit	RW	6	0x20	Upper bound on the Post-Cursor coefficient values to try. Permissible values are 0-32 (0 to 0.5).
[7:6]	reserved	RO	2	0x0	—
[5:0]	pre_cursor_limit	RW	6	0x10	Upper bound on the Pre-cursor coefficient values to try. Permissible values are 0-16 (0 to 0.25).

eq_table_method_control Register 0x1cc

This register set is used for Equalization Figure of Merit Table Method and Adaptive Table Method configuration for 8G data rate.

Table 5.67. eq_table_method_control Register 0x1cc

Field	Name	Access	Width	Reset	Description
[31:9]	reserved	RO	23	0x0	—
[8]	end_on_hold	RW	1	0x1	When the Adaptive Table Method is selected, determines whether to exit early if a Prior PHY Up/Down Feedback response == {HOLD, HOLD} is received on all lanes while processing a Table Entry with interpret==3. 0 – NoExitOnHold 1 – YesExitOnHold
[7:5]	reserved	RO	3	0x0	—
[4:0]	addr_limit	RW	5	0x11	Last table entry to use for the Table Method and Adaptive Table Method. The Table method walks through table entries from i == 0 to eq_table_addr_limit selecting eq_table_method_pre_cursor[(i*6)+5:(i*6)] as the pre-cursor coefficient and eq_table_method_post_cursor[(i*6)+5:(i*6)] as the post-cursor coefficient for each Equalization evaluation trial.

eq_table_method_table Register 0x1d0

This register set is used to set the table array for the Equalization Figure of Merit Table Method and Adaptive Table Method for 8G data rate.

Table 5.68. eq_table_method_table Register 0x1d0

Field	Name	Access	Width	Reset	Description
[383:0]	array	RW	384	0x300030003000300030003000 300030003000300030003000400 07000800090006000500004	For ≥ 8G capable cores only. The array consists of 16-bit table entries, packed back-back, for each of the 24 implemented table entries. Table entries are used to configure the Rx Equalization algorithms Table Method and Adaptive Table Method.

Each 16-bit table entry [i] is in the following format:

- array[(i*16)+15] – reserved
- array[(i*16)+14] – best
- array[(i*16)+13:(i*16)+12] – Interpret[1:0]
- array[(i*16)+11:(i*16)+ 6] – post[5:0]
- array[(i*16)+ 5:(i*16)+ 0] – pre [5:0]

When the Table Method is selected:

- When interpret[1:0] == 00, use the coefficients corresponding to Preset[pre[3:0]].
- When interpret[1:0] == 01, pre[5:0] is the desired pre-cursor coefficient and post[5:0]is the desired post-cursor coefficient. best is unused

When the Adaptive Table Method is selected:

- When interpret[1:0] == 00, use the coefficients corresponding to Preset[pre[3:0]].
- When interpret[1:0] == 01, pre[5:0] is the desired pre-cursor coefficient and post[5:0] is the desired post-cursor coefficient.
- When interpret[1:0] == 10, pre[5:0] is the relative pre-cursor offset from the current relative best coefficient pair {rel_best_post, rel_best_pre} and post[5:0] is the relative post-cursor offset from the current relative best coefficient pair {rel_best_post, rel_best_pre}.

- When interpret[1:0] == 11, apply the prior PHY Up/Down Feedback result to the current best coefficient pair {best_post, best_pre} and make the result the new best coefficient pair. best==1 instructs the core to set {rel_best_post, rel_best_pre} to the coefficient pair that returned the highest Figure of Merit from among all the coeficient pairs tried since the beginning of Rx Equalization (that is, include the currently executing table entry through table entry 0 inclusive).

eq_updn_control Register 0x240

This register set is used for the Equalization Up/Down Convergence Method configuration for 8G data rate.

Table 5.69. eq_updn_control Register 0x240

Field	Name	Access	Width	Reset	Description
[31:26]	iteration_max	RW	6	0x0	The iteration count at which Up/Dn Equalization should assume convergence if not yet converged. A 0 value indicates no limit.
[25]	start_remote_adv	RW	1	0x0	If (1), then the initial pre-cursor and post-cursor coefficients requested from the remote link partner are initialized by the coefficients advertised by the link partner in received TS sets. This option is available to simplify the selection of the initial coefficient values by using the initial coefficients decided determined to be an optimal starting case for the link partner.
[24]	fail_limit_err	RW	1	0x0	If (1), then if an *up* response is received while a coefficient is at its maximum legal value, this is considered an error which causes RX Equalization to fail. If (0), it does not cause RX Equalization to fail, but the coefficient value is limited by the maximum value. This is also true for cases where a *down* response is received for a coefficient set to the minimum value (0).
[23:18]	reserved	RO	6	0x0	—
[17:16]	post_step	RW	2	0x2	Initial step size for changing the post-cursor coefficient values. 0 – Step Size 1 1 – Step Size 2 2 – Step Size 4 3 – Step Size 8
[15:10]	reserved	RO	6	0x0	—
[9:8]	pre_step	RW	2	0x1	This field defines the initial step size for changing the pre-cursor coefficient values based on the PHY up/dn response. 0 – Step Size 1 1 – Step Size 2 2 – Step Size 4 3 – Step Size 8
[7:3]	reserved	RO	5	0x0	—
[2]	use_coef	RW	1	0x0	Controls whether the presets are communicated to the remote device using (1) the associated coefficient values or (0) the Preset value. When (1) the LTSSM requests coefficient values that are equivalent to the desired presets in the C-1, C0, and C+1 fields of TS1 Ordered Sets. When (0) the LTSSM requests the desired presets using the Use Preset and Tx Preset fields of TS1 Ordered Sets.
[1]	start_preset	RW	1	0x1	If set to (1) than the initial pre-cursor and post-cursor coefficients requested from the remote link partner are controlled as a preset value, which is loaded into eq_pre_cursor_laneX[3:0]. This option is available to simplify the selection of the initial coefficient values by using the preset settings specified in the PCIe Specification.

Field	Name	Access	Width	Reset	Description
[0]	numhold	RW	1	0x0	If set to (1), then two consecutive hold status responses are needed for each coefficient to indicate that the coefficient is at an optimal setting. This is required for PHYs which update only one coefficient on each response and set to the other coefficient response to hold. If set to (0), then a single hold response is sufficient to consider a coefficient at an optimal setting.

eq_firmware_control Register 0x280

This register set is used for Equalization Up/Down Firmware Controlled Method configuration for 8G data rate.

Table 5.70. eq_firmware_control Register 0x280

Field	Name	Access	Width	Reset	Description
[31:7]	reserved	RO	25	0x0	—
[6]	tx_quiesce	RW	1	0x0	Transmit Quiesce Guarantee Control Value of Quiesce Guarantee in transmitted TS2 sets in Recovery.RcvrCfg when in the DL_ACTIVE Data Link Layer state (which is L0 and having exchanged Flow Control DLLPs successfully). Quiesce Guarantee is set to 1 in Detect.Quiet and is set to tx_quiesce when in DL_ACTIVE. Quiesce Guarantee holds the last value to which it is set. When the Upstream Port is requesting Equalization be re-run, the Upstream Port is permitted, but not required, to set Quiesce Guarantee == 1 to inform the Downstream Port that an equalization process initiated within 1 ms does not cause any side-effects to its operation.
[5]	tx_req_eq	RW	1	0x0	Transmit Link Equalization Request Control. Set to 1 to set Link Equalization Request == 1 in transmitted TS2 sets in Recovery.RcvrCfg. May be set to 1 by an Upstream Port to inform the Downstream Port link partner (which is the only port that is able to re-run Equalization) that the Upstream Port wants to re-run Equalization. This is a hint to the Downstream Port and the Downstream Port may or may not re-run Equalization.
[4]	int_clr	RW	1	0x0	Writing a 1 to this register clears any pending Equalization interrupts in eq_status – rx_quiesce_hold eq_status – rx_req_eq_hold eq_status – int
[3]	int_en	RW	1	0x0	Local Rx Equalization Interrupt Enable. Enables interrupts to be generated for Rx Equalization events. An interrupt is generated on Entry to Local Rx Equalization Phase 2 (US Port) or Phase 3 (DS Port) and after the PHY Response is received for each Local Rx Equalization trial.
[2]	complete	RW	1	0x0	Firmware Controlled Equalization – Complete. Set to 1 to instruct the Local Rx Firmware Controlled Equalization algorithm to consider Rx Equalization complete. Complete is set to 1, instead of setting advance to 1, when no new equalization step is needed, and Equalization can be considered complete. 0 – NotComplete 1 – Complete

Field	Name	Access	Width	Reset	Description
[1]	advance	RW	1	0x0	Firmware Controlled Equalization – Advance. Set to 1 to instruct the Local Rx Firmware Controlled Equalization algorithm to try the new coefficients specified in the eq_pre_cursor and eq_post_cursor registers. When the core completes a trial, the core stores the resulting PHY Figure of Merit and Directional feedback and waits until new coefficients are loaded by Firmware and Firmware sets the advance register to start another trial or else Firmware sets the complete register to end Equalization. 0 – DoNotAdvance 1 – Advance
[0]	ext_control	RW	1	0x0	Specifies whether the local PHY RX Equalization is under Firmware or local hardware control. This option is only supported when the Equalization Up/Down method is selected (eq_method:select_dir_fom_n == 1). 0 – HardwareControl 1 – FirmwareControl

eq_pre_cursor Register 0x290

This register set is used to set pre-cursor coefficients for the Equalization Up/Down Firmware Controlled Method for 8G data rate.

Table 5.71. eq_pre_cursor Register 0x290

Field	Name	Access	Width	Reset	Description
[95:0]	coef	RW	96	0x0	For ≥ 8G capable cores only, which are delivered with PHY implementing Up/Down Equalization Feedback -Pre-cursor table – 6-bit pre-cursor coefficients for each of the 16 possible lanes packed back-back. A particular lane entry[i] is accessed as [(i*6)+5:(i*6)].

eq_post_cursor Register 0x2a0

This register set is used to set post-cursor coefficients for the Equalization Up/Down Firmware Controlled Method for 8G data rate.

Table 5.72. eq_post_cursor Register 0x2a0

Field	Name	Access	Width	Reset	Description
[95:0]	coef	RW	96	0x0	For ≥ 8G capable cores only, which are delivered with PHY implementing Up/Down Equalization Feedback – Post-cursor table – 6-bit post-cursor coefficients for each of the 16 possible lanes packed back-back. A particular lane entry[i] is accessed as [(i*6)+5:(i*6)].

eq_status Register 0x2c0

This register set is used to read the status registers for the Equalization Up/Down Firmware Controlled Method for 8G data rate.

Table 5.73. eq_status Register 0x2c0

Field	Name	Access	Width	Reset	Description
[31:7]	reserved	RO	25	0x0	—
[6]	rx_quiesce_hold	RO	1	0x0	When the Request Equalization bit is set on ingress TS2 sets at 8G speed, mgmt_tlb_eq_status_rx_quiesce_hold is set to the value of the Quiesce Guarantee bit in those same TS2 sets. The Quiesce Guarantee bit indicates that it is safe for the remote link partner's application to re-run RX Equalization, which can take tens of milliseconds. mgmt_tlb_eq_status_rx_quiesce_hold is cleared when mgmt_tlb_eq_firmware_control_int_clr==1. Refer to the PCI Express 3.0 Spec, Section 4.2.3 for details on how to use this information.
[5]	rx_req_eq_hold	RO	1	0x0	When the Request Equalization bit is set on ingress TS2 sets at 8G speed, mgmt_tlb_eq_status_rx_req_eq_hold is set to 1. This indicates that the remote link partner is requesting RX Equalization be re-run. mgmt_tlb_eq_status_rx_req_eq_hold is cleared when mgmt_tlb_eq_firmware_control_int_clr==1. See the PCI Express 3.0 Spec, Section 4.2.3 for details on how to use this information.
[4]	int_edge	RO	1	0x0	Pulsed interrupt signal. Indicates an interrupt is signaled with a 1 clock wide pulse. Used for state machine logic, not applicable for CSR register reads or polling.
[3]	err	RO	1	0x0	Indicates that an error occurred during RX Equalization.
[2]	exit	RO	1	0x1	Indicates that RX Equalization has not started or has completed.
[1]	ready	RO	1	0x0	Indicates the RX Equalization process requires a response from the external control process.
[0]	int	RO	1	0x0	Rx Equalization Interrupt Status. int==1 indicates an interrupt is active. int is set to 1 when mgmt_tlb_eq_status_ready is set to 1 indicating that the Firmware Controlled Equalization algorithm is ready for new coefficients to try. int is also set to 1 when mgmt_tlb_eq_status_rx_req_eq_hold is set to 1 indicating that the link partner set Link Equalization Request==1 in its transmitted TS2 OS in Recovery.RcvrCfg to request that Equalization be re-run. int=1 is cleared to 0 when eq_firmware_control:advance is set to 1 indicating that new coefficients have been provided or when eq_firmware_control:complete is set to 1 indicating that Equalization is complete, or when eq_firmware_control:int_clr is written to 1.

eq_status_error Register 0x2c4

This register set is used to read the error status registers for the Equalization Up/Down Firmware Controlled Method for 8G data rate.

Table 5.74. eq_status_error Register 0x2c4

Field	Name	Access	Width	Reset	Description
[31:16]	lane_error	read-only	16	0x0	Per-lane equalization error status. Errors can be caused either by the coefficients being rejected by the remote link partner (which is only permitted when certain presets are used in reduced-swing mode) or if a response is not detected by the remote link partner (which might be caused by extremely low link signal quality). For each lane: 0 – No Error 1 – Error
[15:0]	lane_active	read-only	16	0x0	Per-lane active indicator. Only lanes that are active (trained to be part of the link) can be expected to participate in equalization. For each lane: 0 – Inactive 1 – Active

eq_status_preset_coef Register 0x2c8

This register set is used to read the preset coefficient registers for the Equalization Up/Down Firmware Controlled Method for 8G data rate.

Table 5.75. eq_status_preset_coef Register 0x2c8

Field	Name	Access	Width	Reset	Description
[31:16]	lane_match	RO	16	0x0	Per-lane Coefficients or Preset Match Last Request. Indicates that in TS sets received in Recovery.RcvrLock, coefficients or preset fields matched those of the last coefficient or preset request. 0 – No Error 1 – Error
[15:0]	lane_match_valid	RO	16	0x0	Per-lane Remote PHY Match Field Valid. Indicates that the last time through Recovery.RcvrLock, RX Equalization phases 2 and 3 had completed, and this lane received 8 consecutive 8G EC00 TS Sets. 0 – Inactive 1 – Active

eq_status_feedback_fom Register 0x2d0

This register set is used to read the per lane FOM from local PHY for the Equalization Figure of Merit Method for 8G data rate.

Table 5.76. eq_status_feedback_fom Register 0x2d0

Field	Name	Access	Width	Reset	Description
[127:0]	value	RO	128	0x0	Per-lane Figure of Merit Equalization feedback received from the local PHY – 8-bits per lane. value[(i*8)+7:(i*8)] is the measure of Equalization quality for Lane[i] with higher values indicating better BER.

eq_status_feedback_dir Register 0x2e0

This register set is used to read the per lane pre-cursor and post-cursor values from local PHY for the Equalization Up/Down Feedback Method for 8G data rate.

Table 5.77. eq_status_feedback_dir Register 0x2e0

Field	Name	Access	Width	Reset	Description
[63:0]	value	RO	164	0x0	Per-lane Up/Down Equalization feedback received from the local PHY – 4-bits per lane. value[(i*4)+3:(i*4)+2] is the post-cursor feedback and [(i*4)+1:(i*4)+0] is the pre-cursor feedback for Lane[i] with feedback encoded as:- 00==NoChange/Hold, 01==Increment, 10==Decrement, and 11==Reserved.

eq_status_remote_fs Register 0x2e8

This register set is used to read the per lane FS value of remote link partner during RX Equalization for 8G data rate.

Table 5.78. eq_status_remote_fs Register 0x2e8

Field	Name	Access	Width	Reset	Description
[95:0]	value	RO	96	0x0	Per-lane FS value advertised by the remote link partner in TS Sets received in Recovery.Equalization.Phase1. The FS value for lane[i] is value[(i*6)+5:(i*6)].

eq_status_remote_lf Register 0x2f4

This register set is used to read the per lane LF value of remote link partner during RX Equalization for 8G data rate.

Table 5.79. eq_status_remote_lf Register 0x2f4

Field	Name	Access	Width	Reset	Description
[95:0]	value	RO	96	0x0	Per-lane LF value advertised by the remote link partner in TS Sets received in Recovery.Equalization.Phase1. The LF value for lane[i] is value[(i*6)+5:(i*6)].

eq_status_remote_precursor Register 0x300

This register set is used to read the per lane PHY pre-cursor coefficient of remote link partner during RX Equalization for 8G data rate.

Table 5.80. eq_status_remote_precursor Register 0x300

Field	Name	Access	Width	Reset	Description
[95:0]	value	RO	96	0x0	Per-lane Remote PHY Pre-Cursor Coefficient. Indicates the remote device pre-cursor coefficient advertised in 8G TS Sets received in the last time through Recovery.RcvrLock. The precursor for lane[i] is value[(i*6)+5:(i*6)].

eq_status_remote_postcursor Register 0x30c

This register set is used to read the per lane PHY post-cursor coefficient of remote link partner during RX Equalization for 8G data rate.

Table 5.81. eq_status_remote_postcursor Register 0x30c

Field	Name	Access	Width	Reset	Description
[95:0]	value	RO	96	0x0	Per-lane Remote PHY Post-Cursor Coefficient. Indicates the remote device post-cursor coefficient advertised in 8G TS Sets received in the last time through Recovery.RcvrLock. The postcursor for lane[i] is value[(i*6)+5:(i*6)].

5.1.2.6. Physical Layer Register Set

pl_rx Register 0x33c

This register set is used for the Physical Layer Per Lane Receiver Control.

Table 5.82. pl_rx Register 0x33c

Field	Name	Access	Width	Reset	Description
[31:17]	reserved	RO	15	0x0	—
[16]	inject_data_error_en	RW	1	0x0	pipe_rx_data error injection for test purposes. inject_data_error_mask[NUM_LANES-1:0] controls which lane(s) is used for error injection. For example, to enable a bit error on lane 2, first write 0x04 to inject_data_error_mask, next write a 1 to inject_data_error_en, then finally write 0 to inject_data_error_en. This is a test only feature which is not used for normal operation. inject_data_error_en must be set to 0 whenever error injections are not desired. The write to this field must occur after writes to the other fields in this register have established the desired parameters of the injection.
[15:0]	inject_data_error_mask	RW	16	0x0	Lane select for data error injection. Bit 0 corresponds to lane 0, Bit 1 corresponds to lane 1. Setting the bit for a corresponding lane to 1 result in an error being injected on that lane when inject_data_error_en changes from 0 to 1. Always setup inject_data_error_mask before setting inject_data_error_en to 1. This field may not be changed while inject_data_error_en==1.

pl_16g Register 0x340

This register set is used for the Physical Layer 16G Optional Behavior Enable.

Table 5.83. pl_16g Register 0x340

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	enable_16g_eie_same_as_8g	RW	1	0x0	Determines whether 16G EIEOS use Draft 0.7 and higher OS definition or Draft 0.5 and lower OS definition (which is the same as 8G). 0 – Use PCIe 4.0 Draft 0.7 and higher 16G EIEOS definition – 0xFFFF0000FFFF0000FFFF0000FFFF0000 1 – Use PCIe 4.0 Draft 0.5 and lower 16G EIEOS definition – 0xFF00FF00FF00FF00FF00FF00FF00FF00

pl_tx_skp Register 0x344

This register set is used for the Physical Layer Transmit SKP Period Control.

Table 5.84. pl_tx_skp Register 0x344

Field	Name	Access	Width	Reset	Description
[31:30]	reserved	RO	2	0x0	—
[29:24]	period_sris_128b130b	RW	6	0x0	The transmit SKP period used when operating at ≥ 8G with the SRIS capability enabled and configured for SRIS = period_sris_128b130b Blocks. PCIe Specification is < 38 blocks. 0 is a special case that selects 36 Blocks. This register must be configured for a PCIe Specification compliant value.
[23:16]	period_srns_128b130b	RW	8	0x0	The transmit SKP period used when operating at ≥ 8G with the SRIS capability disabled or with SRIS enabled but configured for SRNS = 256 + period_srns_128b130b Blocks. PCIe Specification is 370-375 blocks. 0 is a special case that selects 116 == 372 Blocks. This register must be configured for a PCIe Specification compliant value.
[15:8]	period_sris_8b10b	RW	8	0x0	The transmit SKP period used when operating at ≤ 5G with the SRIS capability enabled and configured for SRIS = period_sris_8b10b Symbol Times. PCIe Specification is < 154 Symbol Times. 0 is a special case that selects 146 Symbol Times. This register must be configured for a PCIe Specification compliant value. The number of symbol times selected must be a multiple of the PHY per lane symbol data width or the lower bits are truncated. For example: For 16-bit per lane PHY, period_sris_8b10b[0] is always treated as 0. For 32-bit per lane PHY, period_sris_8b10b[1:0] are always treated as 00. For 64-bit per lane PHY, period_sris_8b10b[2:0] are always treated as 000.
[7:0]	period_srns_8b10b	RW	8	0x0	The transmit SKP period used when operating at ≤ 5G with the SRIS capability disabled or with SRIS enabled but configured for SRNS = (256 + period_srns_8b10b) * 4 Symbol Times. PCIe Specification is 1180-1538 Symbol Times. 0 is a special case that selects 44 == 1200 Symbol Times. This register must be configured for a PCIe Specification compliant value. The number of symbol times selected must be a multiple of the PHY per lane symbol data width or the lower bits are truncated. For example: For 64-bit per lane PHY period_srns_8b10b[0] is always treated as 0. For 32, 16, and 8-bit per lane PHY, all of period_srns_8b10b is relevant.

pl_tx_debug Register 0x348

This register set is used for the Physical Layer Debug Control.

Table 5.85. pl_tx_debug Register 0x348

Field	Name	Access	Width	Reset	Description
[31:3]	reserved	RO	29	0x0	—
[2]	inject_margin_crc_error	RW	1	0x0	Setting this to 1 injects errors into the margin crc value of the control skp ordered set on all lanes
[1]	inject_margin_parity_error	RW	1	0x0	Setting this to 1 injects errors into the margin parity bit of the control skp ordered set on all lanes
[0]	inject_data_parity_error	RW	1	0x0	Setting this to 1 injects errors into the data parity bit of the control skp ordered set on all lanes

pl_ctrl Register 0x34c

This register set is used for the Physical Layer Control.

Table 5.86. pl_ctrl Register 0x34c

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	8b10b_err_rec_entry_sel	RW	1	0x0	Selects the Physical Layer error threshold required to be received in L0, when operating with 8b10b encoding (2.5G/5G speed), before the link is directed to Recovery. 0 – When in L0 and using 8b10b encoding, direct the link to Recovery after receiving a single Physical Layer Error. This is the more conservative setting but has the disadvantage of causing Recovery entry on all L0 Physical Layer errors – even those errors that the link would be able to recover from on its own without having to go through Recovery. 1 – When in L0 and using 8b10b encoding, direct the link to Recovery only after receiving a massive burst of errors (which typically is an indication that there is a persistent problem, such as PHY loss of lock, for which Recovery entry is required to fix). The core implements an error counter. For each enabled PHY Rx clock cycle, the error counter is incremented when a clock cycle contains a Physical Layer error, and the error counter is decremented when a clock cycle contains no Physical Layer errors. If the counter reaches 1023, the link is directed to Recovery.

pl_ts_matching Register 0x350

This register set is used for the Physical Layer TS Match Control.

Table 5.87. pl_ts_matching Register 0x350

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	legacy_mode	RW	1	0x1	Setting this to 1 compares all symbols when matching TS sets (legacy behavior). Setting this to 0 compares only the symbols required to meet specifications.

5.1.2.7. Data Link Layer Control Register Set

dl_retry_timeout Register 0x380

This register set is used for the Replay Timeout Control.

Table 5.88. dl_retry_timeout Register 0x380

Field	Name	Access	Width	Reset	Description
[31:24]	pcie4_symt_sync	RW	8	0x0	Replay Timeout Timer PCIe 4.0 Simplified REPLAY_TIMER Limits Extended Sync==1 Value. {pcie4_symt_sync, 10'h0} = Symbol times to use for Replay Timer when pcie4_enable==1 and Extended Synch==1. 0 is a special case selecting 8'd80.
[23:16]	pcie4_symt_sync_n	RW	8	0x0	Replay Timeout Timer PCIe 4.0 Simplified REPLAY_TIMER Limits Extended Sync==0 Value. {pcie4_symt_sync_n, 10'h0} = Symbol times to use for Replay Timer when pcie4_enable==1 and Extended Synch==0. 0 is a special case selecting 8'd24.
[15]	pcie4_enable	RW	1	0x1	Replay Timeout Timer PCIe 4.0 Simplified REPLAY_TIMER Limits Enable. 0 – Use PCIe 3.0 Specification and prior REPLAY_TIMER Limits from UNADJUSTED REPLAY_TIMER LIMITS tables in the PCIe Specification. 1 – Use PCIe 4.0 Specification Simplified REPLAY_TIMER Limits.
[14:1]	l0s_adj	RW	14	0x180	Replay Timeout L0s Adjustment. The number of symbol times to add to the recommended PCIe Replay Timer timeout period to compensate for the remote link having to exit L0s before it can send an ACK/NAK DLLP. l0s_adj is added to the Replay Timer timeout period after the optional doubling controlled by mult_enable is applied. Not applicable when pcie4_enable==1.
[0]	mult_enable	RW	1	0x0	Replay Timeout Timer Multiplier Enable. Not applicable when pcie4_enable==1. 0 – The Replay Timer timeout period implemented is the recommended (-0%) value in the PCIe Specification UNADJUSTED REPLAY_TIMER LIMITS FOR 2.5/5.0/8.0 GT/S MODE OPERATION BY LINK WIDTH AND MAX_PAYLOAD_SIZE tables. 1 – The Replay Timer timeout period implemented is 2 times the recommended (-0%) value in the PCIe Specification UNADJUSTED REPLAY_TIMER LIMITS FOR 2.5/5.0/8.0 GT/S MODE OPERATION BY LINK WIDTH AND MAX_PAYLOAD_SIZE tables.

dl_ack_timeout_div Register 0x384

This register set is used for the ACK Timer Control.

Table 5.89. dl_ack_timeout_div Register 0x384

Field	Name	Access	Width	Reset	Descriptions
[31:1]	reserved	RO	31	0x0	—
[0]	enable	RW	1	0x0	ACK Timer Control. 0 – Ack according to specifications. 1 – Ack twice as often as recommended by specifications.

dl_tx_ctrl Register 0x38c

This register set is used for the Data Link Layer TX Control.

Table 5.90. dl_tx_ctrl Register 0x38c

Field	Name	Access	Width	Reset	Descriptions
[31]	stp_override_en	RW	1	0x0	8G STP Symbol Debug Length Override Enable. Enables the injection of 8G STP symbol length errors (for debug only). On a rising edge of stp_override_en, a request to inject an 8G STP length error is stored until it can be acted upon. The next TLP with a computed 8G STP length == stp_override_len has its STP token length field replaced by stp_override_new_len instead of using the computed value. Per PCIe Specification, the 8G STP token length must consider the full framed TLP length including 8G STP Token, TLP Header, TLP Payload, ECRC (if present), and LCRC, but must not include the EDB symbol (if the TLP is being nullified). These fields enable you to transmit an 8G TLP of incorrect length by placing the EOP at the end of the TLP data to transmit and then writing these fields to substitute the desired STP length matching the TLP being transmitted for the actual length of the TLP that would be computed from the TLP header. At 2.5 and 5G, these fields are unused as the full framed TLP length is not included in the STP token at these speeds and a malformed length TLP can be transmitted just by placing EOP at the incorrect location. 0 – Disabled 1 – Enabled
[30:27]	reserved	RO	4	0x0	—
[26:16]	stp_override_new_len	RW	11	0x0	8G STP Symbol Debug Length Replacement Value.
[15:11]	reserved	RO	5	0x0	—
[10:0]	stp_override_len	RW	11	0x0	8G STP Symbol Debug Length Replacement Match.

dl_ctrl Register 0x390

This register set is used for the Data Link Layer Control.

Table 5.91. dl_ctrl Register 0x390

Field	Name	Access	Width	Reset	Descriptions
[31:26]	reserved	RO	6	0x0	—
[25]	tx_pfx_par_inject_en	RW	1	0x0	Transmit Data Link Layer Prefix Parity Error Injection Enable. 0 – Do not inject error. 1 – On the rising edge, a single prefix parity error injection, applied prior to assigning the TLP sequence number, is scheduled and injected at the next opportunity (TLP transmit). Tx prefix parity error Handling and Reporting are governed by tx_par2_handle_disable and tx_par2_report_disable.
[24]	rx_early_forward_disable	RW	1	0x0	Receive Data Link Layer down-trained early forwarding disable. 0 – When down-trained, forward Rx Data Link Layer data for processing whenever a TLP/DLLP end occurs without a following TLP/DLLP start the same clock cycle. This setting results in lower Rx TLP/DLLP latency. 1 – When down-trained, always aggregate Rx Data Link Layer data to full width before forwarding the data. For example, a x16 core operating at x1 receives and aggregate 16 clock cycles of 1 lane data before outputting one clock cycle of 16 lane data for further processing.
[23]	reserved	RO	1	0x0	—
[22]	tx_gap_inject_en	RW	1	0x0	Transmit Data Link Layer TX Valid Gap Injection Enable. 0 – Do not inject gap. 1 – On the rising edge, a single clock bp_tx_valid gap is scheduled and is injected at the next opportunity (within a TLP). This gap in the bp_tx_valid can cause a data underflow at the Physical Layer.
[21]	rx_malf_inject_en	RW	1	0x0	Receive Data Link Layer Malformed Length TLP Injection Enable. 0 – Do not inject error. 1 – On the rising edge, a single malformed TLP error injection is scheduled and is injected at the next opportunity (Tx TLP EOP). The TLP is malformed by deleting its end of the TLP indicator causing the TLP to end at the incorrect location.
[20]	rx_lcrc_inject_en	RW	1	0x0	Receive Data Link Layer LCRC Injection Enable. 0 – Do not inject error. 1 – On the rising edge, a single LCRC TLP error injection is scheduled and injected at the next opportunity (Tx TLP EOP). The LCRC is corrupted by inverting LCRC bit 0 of the received TLP.
[19]	reserved	RO	1	0x0	—

Field	Name	Access	Width	Reset	Descriptions
[18]	rx_dl_active_disable	RW	1	0x0	Control the use of DL_Active to block reception of TLPs. 0 – Block reception of TLPs when dl_active is low. 1 – Do not block TLP reception based on dl_active.
[17]	rx_inhibit_tlp	RW	1	0x0	Receive Data Link Layer TLP Rx Inhibit Enable. 0 – Process received TLPs per PCIe Specification Required setting for compliant PCIe operation. 1 – For test purposes only, discard and do not accept received TLPs. Received TLPs are processed as if the Sequence Number is one greater than received. This prevents the TLP with the current expected Sequence Number and all following TLPs from being received. This causes the link partner to do TLP replays as received TLPs are incorrect due to perceived Sequence Number errors.
[16]	rx_inhibit_ack_nak	RW	1	0x0	Receive Data Link Layer ACK/NAK Inhibit Enable. 0 – Process received ACK and NAK DLLPs per PCIe Specification. Required setting for compliant PCIe operation. 1 – For test purposes only, discard and do not process received ACK and NAK DLLPs. This causes the core to do TLP replays because TLP acknowledgements do not received.
[15]	reserved	RO	1	0x0	—
[14]	tx_par2_report_disable	RW	1	0x0	Transmit Data Link Layer Parity 2 Error Reporting Disable. 0 – Enable reporting. 1 – Disable reporting of Data Link Layer transmit parity errors.
[13]	tx_par2_handle_disable	RW	1	0x0	Transmit Data Link Layer Parity 2 Error Handling Disable. 0 – Enable handling. TLPs with errors are nullified and not retransmitted. 1 – Disable handling of Data Link Layer transmit parity errors. When error handling is disabled, TLPs with parity errors continue to be transmitted.
[12]	tx_par2_inject_en	RW	1	0x0	Transmit Data Link Layer Parity 2 Error Injection Enable. 0 – Do not inject error. 1 – On the rising edge, a single parity error injection, applied after assigning the TLP sequence number, is scheduled and is injected at the next opportunity (TLP transmit).
[11:9]	reserved	RO	3	0x0	—

Field	Name	Access	Width	Reset	Descriptions
[8]	tx_par1_inject_en	RW	1	0x0	Transmit Data Link Layer Parity 1 Error Injection Enable. 0 – Do not inject error. 1 – On the rising edge, a single parity error injection, applied prior to assigning the TLP sequence number, is scheduled and is injected at the next opportunity (TLP transmit).
[7]	reserved	RO	1	0x0	—
[6]	tx_replay_ecc2_report_disable	RW	1	0x0	Transmit Replay Buffer ECC 2-bit Error Reporting Disable. 0 – Enable reporting. 1 – Disable reporting of ECC 2-bit errors.
[5]	tx_replay_ecc2_handle_disable	RW	1	0x0	Transmit Replay Buffer ECC 2-bit Error Handling Disable. 0 – Enable handling. TLPs with errors are nullified and not retransmitted. 1 – Disable handling of ECC 2-bit errors. When error handling is disabled, TLPs with ECC 2-bit errors continue to be transmitted.
[4]	tx_replay_ecc2_inject_en	RW	1	0x0	Transmit Replay Buffer ECC 2-bit Error Injection Enable. 0 – Do not inject error. 1 – On the rising edge, a single ECC 2-bit error injection is scheduled and is injected at the next opportunity (Replay Buffer RAM write). The error is only seen if a Replay occurs of the TLP receiving the error injection.
[3]	reserved	RO	1	0x0	—
[2]	tx_replay_ecc1_report_disable	RW	1	0x0	Transmit Replay Buffer ECC 1-bit Error Reporting Disable. 0 – Enable reporting. 1 – Disable reporting of ECC 1-bit errors.
[1]	tx_replay_ecc1_handle_disable	RW	1	0x0	Transmit Replay Buffer ECC 1-bit Error Handling Disable. 0 – Enable correction. 1 – Disable correction of ECC 1-bit errors. When error correction is disabled, ECC 1-bit errors are treated the same as uncorrectable ECC 2-bit errors.
[0]	tx_replay_ecc1_inject_en	RW	1	0x0	Transmit Replay Buffer ECC 1-bit Error Injection Enable. 0 – Do not inject error. 1 – On the rising edge, a single ECC 1-bit error injection is scheduled and is injected at the next opportunity (Replay Buffer RAM write). The error is only seen if a Replay occurs of the TLP receiving the error injection.

dl_stat Register 0x394

This register set is used for the Data Link Layer Status.

Table 5.92. dl_stat Register 0x394

Field	Name	Access	Width	Reset	Description
[31]	info_bad_tlp_null_err	RW, W1C	1	0x0	Nullified TLP Received Status. This is not a reported error but is useful information to store for debug. This is a sub-class of err_aer_bad_tlp that provides more information as to why the TLP is bad. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[30]	info_bad_tlp_phy_err	RW, W1C	1	0x0	TLP PHY Error Status. This is not a reported error but is useful information to store for debug. This is a sub-class of err_aer_bad_tlp that provides more information as to why the TLP is bad. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[29]	info_bad_tlp_malf_err	RW, W1C	1	0x0	Malformed TLP Status. This is not a reported error but is useful information to store for debug. This is a sub-class of err_aer_bad_tlp that provides more information as to why the TLP is bad. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[28]	info_bad_tlp_ecrc_err	RW, W1C	1	0x0	TLP ECRC Mismatch Status. This is not a reported error but is useful information to store for debug. This is a sub-class of err_aer_bad_tlp that provides more information as to why the TLP is bad. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[27]	info_schedule_dupl_ack	RW, W1C	1	0x0	Duplicate TLP Received Status. This is not a reported error but is useful information to store for debug. Duplicate TLPs are received during TLP Replay. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[26]	info_bad_tlp_seq_err	RW, W1C	1	0x0	TLP Sequence Number Mismatch Status. This is not a reported error but is useful information to store for debug. This is a sub-class of err_aer_bad_tlp that provides more information as to why the TLP is bad. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[25]	info_bad_tlp_crc_err	RW, W1C	1	0x0	TLP LCRC Mismatch Status. This is not a reported error but is useful information to store for debug. This is a sub-class of err_aer_bad_tlp that provides more information as to why the TLP is bad. 0 – Otherwise 1 – Event occurred. Write 1 to clear.

Field	Name	Access	Width	Reset	Description
[24]	info_nak_received	RW, W1C	1	0x0	NAK Received Status. This is not a reported error but is useful information to store for debug. Receiving a NAK indicates that the link partner requested a Replay. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[23]	info_deskew_overflow_error	RW, W1C	1	0x0	Rx Deskew FIFO Overflow Error Status. The lane-lane skew of Rx data on one or more lanes are latent from the other lanes that the deskew range of the Rx Deskew FIFO is exceeded. This is a correctable error since the core drives the link to Recovery to fix this issue. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[22]	info_tx_data_underflow	RW, W1C	1	0x0	Physical Layer TLP Transmit Underflow Error Status. A TLP is transmitted by the physical layer and more data is needed to continue the transmission, but no data is provided. This error is normally caused by the Transaction Layer failing to provide TLP data at ≥ PCIe Line Rate. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[21]	info_replay_started	RW, W1C	1	0x0	A Replay is started. This is not a reported error but is useful information to store for debug. Indicates a Replay occurred on local TX interface due to either Ack Timeout or Nack Reception. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[20]	reserved	RO	1	0x0	—
[19]	err_aer_tx_par2	RW, W1C	1	0x0	Transmit Data Link Layer Parity 2 Error Status. Indicates that a Data Link Layer transmit parity error is detected after sequence number application. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[18]	reserved	RO	1	0x0	—
[17]	err_aer_tx_replay_ecc2	RW, W1C	1	0x0	Transmit Replay Buffer ECC 2-bit Error Status. Indicates that an uncorrectable ECC error occurred during Replay. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[16]	err_aer_tx_replay_ecc1	RW, W1C	1	0x0	Transmit Replay Buffer ECC 1-bit Error Status. Indicates that a correctable ECC error occurred during Replay. 0 – Otherwise 1 – Event occurred. Write 1 to clear.
[15:8]	reserved	RO	8	0x0	—
[7]	reserved	RO	1	0x0	—

Field	Name	Access	Width	Reset	Description
[6]	err_aer_surprise_down	RW, W1C	1	0x0	Surprise Down Error Status. 0 – Otherwise 1 – Error occurred. Errors of this type must be logged in the Transaction Layer AER Capability. Write 1 to clear.
[5]	err_aer_dl_protocol_error	RW, W1C	1	0x0	DL Protocol Error Status. 0 – Otherwise 1 – Error occurred. Errors of this type must be logged in the Transaction Layer AER Capability. Write 1 to clear.
[4]	err_aer_replay_timer_timeout	RW, W1C	1	0x0	Replay Timer Timeout Error Status. 0 – Otherwise 1 – Error occurred. Errors of this type must be logged in the Transaction Layer AER Capability. Write 1 to clear.
[3]	err_aer_replay_num_rollover	RW, W1C	1	0x0	Replay Num Rollover Error Status. 0 – Otherwise 1 – Error occurred. Errors of this type must be logged in the Transaction Layer AER Capability. Write 1 to clear.
[2]	err_aer_bad_dllp	RW, W1C	1	0x0	Bad DLLP Error Status. 0 – Otherwise 1 – Error occurred. Errors of this type must be logged in the Transaction Layer AER Capability. Write 1 to clear.
[1]	err_aer_bad_tlp	RW, W1C	1	0x0	Bad TLP Error Status. 0 – Otherwise 1 – Error occurred. Errors of this type must be logged in the Transaction Layer AER Capability. Write 1 to clear.
[0]	err_aer_receiver_error	RW, W1C	1	0x0	Receiver Error Status. 0 – Otherwise 1 – Error occurred. Errors of this type must be logged in the Transaction Layer AER Capability. Write 1 to clear.

dl_ack_to_nak Register 0x398

This register set is used for the ACK-to-NAK error injection controls.

Table 5.93. dl_ack_to_nak Register 0x398

Field	Name	Access	Width	Reset	Description
[31]	enable	RW	1	0x0	Enable ACK-to-NAK injection. The write to this field must occur after writes to the other fields in this register have established the desired parameters of the injection. 0 – Do nothing 1 – Enable injection and load parameters into the ACK-to-NAK injector.
[30:24]	reserved	RO	7	0x0	—
[23:16]	count	RW	8	0x0	Number of times to replace the ACK with a NAK.
[15:12]	reserved	RO	4	0x0	—
[11:0]	seq_num	RW	12	0x0	Sequence Number of ACK to be changed to a NAK.

dl_inject Register 0x39c

This register set is used for the DLLP CRC/TLP ECRC error injection controls.

Table 5.94. dl_inject Register 0x39c

Field	Name	Access	Width	Reset	Description
[31]	dllp_crc_err_enable	RW	1	0x0	Enable DLLP CRC error injection. The write to this field must occur after writes to the other fields in this register have established the desired parameters of the injection. 0 – Disable the DLLP CRC Error injector. 1 – Enable DLLP CRC Error injector.
[30:28]	reserved	RO	3	0x0	—
[27:16]	dllp_crc_err_rate	RW	12	0x0	Rate at which DLLP CRC errors are to be injected. A value of 0 injects a single DLLP CRC error. A non-zero value injects errors at intervals of Rate*256*clk_period. This field may not be changed while dllp_crc_err_enable==1.
[15:13]	reserved	RO	3	0x0	—
[12]	dllp_inject_enable	RW	1	0x0	Inject a DLLP (transmit) using the data in the dllp_inject_data register. A single DLLP is injected after each rising edge of this signal.
[11:9]	reserved	RO	3	0x0	—
[8]	tlp_seq_err_enable	RW	1	0x0	Modify the sequence number in the next transmitted TLP to an invalid value (bad sequence number error). A single TLP is altered after each rising edge of this signal.
[7:4]	reserved	RO	4	0x0	—
[3]	tlp_lcrc_err_enable	RW	1	0x0	Enable TLP LCRC error injection. The write to this field must occur after writes to the other fields in this register have established the desired parameters of the injection. 0 – Disable the TLP LCRC Error injector 1 – Enable TLP LCRC Error injector

Field	Name	Access	Width	Reset	Description
[2:0]	tlp_lcrc_err_rate	RW	3	0x0	Rate at which TLP LCRC errors are to be injected. A value of 0 injects LCRC errors into all TLPs. A non-zero value injects an error into a TLP and then pass Rate TLPs without error and then repeat. This field may not be changed while tlp_lcrc_err_enable==1.

dllp_inject Register 0x3a0

This register set is used for the DLLP Injector Data.

Table 5.95. dllp_inject Register 0x3a0

Field	Name	Access	Width	Reset	Description
[31:0]	data	RW	32	0x0	Data to include in injected DLLP. This field may not be changed while dl_inject_dllp_inject_enable==1.

5.1.2.8. LTSSM Equalization Status Control

eq_status_table_control Register 0x3d8

This register set is used for LTSSM Equalization Status Control for 8G data rate.

The LTSSM Equalization Status is a debug feature that captures the status of the most recent Equalization execution from a single selected lane and allows software to read the status.
To use this feature:

1. Select a lane to monitor by writing lane_select.
2. Cause Equalization to be executed; Equalization can be executed by writing the following values in sequence to the Downstream Port PCIe Cfg Registers (see PCIe Specification for details):
   a. Perform Equalization = 1
   b. Target Link Speed = Desired Link Speed (1 = 2.5G, 2 = 5G, 3 = 8G[1], 4 = 16G[1])
   c. Retrain Link = 1 (this must be written last)
3. Wait for Equalization to complete by reading eq_status_table_info_done until it indicates done.
4. The speed at which EQ is executed may be read through eq_status_table_info_16g_speed.
5. For (i=0; i<25; i=i+1).

```
begin
Write eq_status_table_control_step_select = i
Read eq_status_table_data == Equalization Status of iteration[i]
End
```

Note:

1. Available in 2024.1 release.

Table 5.96. eq_status_table_control Register 0x3d8

Field	Name	Access	Width	Reset	Description
[31:13]	reserved	RO	19	0x0	—
[12:8]	step_select	RW	5	0x0	Used to select which step (which EQ trial iteration) of the most recently completed EQ process is accessed at the eq_status_table_data. After Equalization has completed, step_select is intended to be written from 0 to 24 to read the Equalization status for all trials.
[7:4]	reserved	RO	4	0x0	—
[3:0]	lane_select	RW	4	0x0	Used to select which lane's data is collected during the next Equalization. This value of this register is captured at the start of Equalization and Equalization results are then saved for the selected lane.

eq_status_table_info Register 0x3dc

This register set is used to read whether the RX Equalization status registers are valid or not at 8G data rate.

Table 5.97. eq_status_table_info Register 0x3dc

Field	Name	Access	Width	Reset	Description
[31:2]	reserved	RO	30	0x0	—
[1]	16g_speed	RO	1	0x0	Indicates speed at which Equalization is executed. This register is reset only by fundamental reset so that the status survives link down and other soft reset conditions. 0 – Equalization run at 8G speed 1 – Equalization run at 16G speed
[0]	done	RO	1	0x0	Indicates that Equalization has been completed, and the table results are valid. This register is reset only by fundamental reset so that the status survives link down and other soft reset conditions.

eq_status_table Register 0x3e0

This register set is used to read RX Equalization status registers for 8G data rate.

Table 5.98. eq_status_table Register 0x3e0

Field	Name	Access	Width	Reset	Description

[31:0]	data	RO	32	0x0	• Equalization Status for the selected lane and step (iteration) recorded as follows:
					  • [1:0] – EQ Pre-Cursor Up/Down Feedback[1:0] (00=Hold,01=Inc,10=Dec,11=Rsvd)
					  • [3:2] – EQ Post-Cursor Up/Down Feedback[1:0] (00=Hold,01=Inc,10=Dec,11=Rsvd)
					  • [11: 4] – EQ Figure of Merit Feedback[7:0]
					  • [17:12] – Remote PHY Tx Post Cursor[5:0]
					  • [23:18] – Remote PHY Tx Pre Cursor[5:0]
					  • [24] – Error Status: 1==Error, 0==No Error
					  • [25] – Active Status: 1==Lane is part of the link, 0==Lane is not part of the link
					  • [31:16] – Reserved
					• Equalization Status is reset to 0x0 for all iterations when Equalization begins so iterations that were not executed reads 0x0.
					• Equalization Status is re-captured on every Equalization, so Equalization Status may only be read between Equalization attempts or it is unknown from which Equalization run the status is captured.
					• Captured Equalization Status is reset only by fundamental reset so that the status survives link down and other soft reset conditions.

eq_capture_sel Register 0x3f0

This register set is used for selection of lane, TX or RX for RX Equalization results for 8G data rate.

Table 5.99. eq_capture_sel Register 0x3f0

Field	Name	Access	Width	Reset	Description
[31:6]	reserved	RO	26	0x0	—
[5]	dir	RW	1	0x0	Used to select TX (dir == 1) or RX (dir == 0) for equalization results read-back.
[4]	speed	RW	1	0x0	Used to select 16G (speed == 1) or 8G (speed == 0) for equalization results read-back.
[3:0]	lane	RW	4	0x0	Used to select the lane number (0 to 15) for equalization results read-back.

eq_capture Register 0x3f4

This register set is used to read out RX Equalization results for 8G data rate.

Table 5.100. eq_capture Register 0x3f4

Field	Name	Access	Width	Reset	Description
[31:22]	reserved	RO	10	0x0	—

[21:0]	result	RO	22	0x0	• Equalization results from LTSSM recorded as follows:
					  • [5: 0] – EQ Post-Cursor[5:0]
					  • [11: 6] – EQ Cursor[5:0]
					  • [17:12] – EQ Pre-Cursor[5:0]
					  • [21:18] – EQ Preset
					  Note: The Pre-Cursor, Cursor, and Post-Cursor values is 0x0 if Equalization Presets are used. Similarly, Preset is 0x0 if Equalization Coefficients are used.
					• Equalization results are set to 0x0 for all lanes, speed, and directions by a fundamental reset.
					• The values are updated at each step of the equalization procedure and then held until the next equalization or fundamental reset.

phy_eq_tx_force_per_lane Register 0x400

This register set is used to enable per-Lane Hardcoded Preset/Coefficient configuration for 8G data rate.

Table 5.101. phy_eq_tx_force_per_lane Register 0x400

Field	Name	Access	Width	Reset	Description
[31:4]	reserved	RO	28	0x0	—
[3]	16g_coef_enable	RW	1	0x0	Used to enable per-lane hardcoded coefficients at 16G speed.
[2]	16g_preset_enable	RW	1	0x0	Used to enable per-lane hardcoded presets at 16G speed
[1]	8g_coef_enable	RW	1	0x0	Used to enable per-lane hardcoded coefficients at 8G speed.
[0]	8g_preset_enable	RW	1	0x0	Used to enable per-lane hardcoded presets at 8G speed

phy_eq_tx_force_per_lane_8g_pre Register 0x404

This register set is used for per-Lane Hardcoded Preset/Pre-cursor values configuration for 8G data rate.

Table 5.102. phy_eq_tx_force_per_lane_8g_pre Register 0x404

Field	Name	Access	Width	Reset	Description
[95:0]	value	RW	96	0x0	Per-lane preset/pre-cursor values to use in hardcoded per-lane mode at 8G speed.

phy_eq_tx_force_per_lane_8g_post Register 0x410

This register set is used for per-Lane Hardcoded Preset/Post-cursor values configuration for 8G data rate.

Table 5.103. phy_eq_tx_force_per_lane_8g_post Register 0x410

Field	Name	Access	Width	Reset	Description
[95:0]	value	RW	96	0x0	Per-lane post-cursor values to use in hardcoded per-lane mode at 8G speed.

5.1.3. mgmt_ptl (0x4_3000)

The following are the register sets with the 0x3000 base address.

5.1.3.1. Simulation Register

Simulation Register 0x0

This register set is used for the Partial Transaction Layer simulation speed reduction.

Table 5.104. Simulation Register 0x0

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	pm_reduce_timeouts	RW	1	0x0	Reduce Power Management State Machine timeouts from their value in ms to their value in µs to shorten simulation time. 0 – Disable 1 – Enable

5.1.3.2. Power Management State Machine Register Set

pm_aspm_l0s Register 0x40

This register set is used for the Power Management State Machine ASPM L0s entry control.

Table 5.105. pm_aspm_l0s Register 0x40

Field	Name	Access	Width	Reset	Description
[31:16]	entry_time	RW	16	0x0	ASPM L0s TX Entry Time in µs. 0 is a special case == 6.9 µs.
[15:1]	reserved	RO	15	0x0	—
[0]	enable	RW	1	0x1	Enable Power Management State Machine to enter ASPM L0s. 0 – Disable 1 – Enable

pm_aspm_l1 Register 0x50

This register set is used for the Power Management State Machine ASPM L1 entry control.

Table 5.106. pm_aspm_l1 Register 0x50

Field	Name	Access	Width	Reset	Description
[31:16]	entry_time	RW	16	0x0	ASPM L1 TX Entry Time in ms. 0 is a special case == 1000 µs
[15:1]	reserved	RO	15	0x0	—
[0]	enable	RW	1	0x1	Enable Power Management State Machine to enter ASPM L1. 0 – Disable 1 – Enable

pm_aspm_l1_min Register 0x54

This register set is used for the Power Management State Machine ASPM L1 re-entry control.

Table 5.107. pm_aspm_l1_min Register 0x54

Field	Name	Access	Width	Reset	Description
[31:30]	reserved	RO	2	0x0	—
[29:16]	reentry_time	RW	14	0x0	When reentry_disable==0, specifies the minimum time between ASPM L1 requests in ns. 0 is a special case == 9500 ns (PCIe Specification value).
[15:1]	reserved	RO	15	0x0	—
[0]	reentry_disable	RW	1	0x0	Disable enforcing a minimum time between ASPM L1 requests. 0 – Enable 1 – Disable

pm_l1 Register 0x60

This register set is used for the Power Management State Machine L1 entry control.

Table 5.108. pm_l1 Register 0x60

Field	Name	Access	Width	Reset	Description
[31:16]	us_port_ps_entry_time	RW	16	0x0	Upstream Ports only: Number of µs to wait for the transmission of the completion to the PowerState Cfg Write that initiated L1 entry, before beginning to block TLPs and enter L1. 0 is a special case == 4 µs.
[15:1]	reserved	RO	15	0x0	—
[0]	enable	RW	1	0x1	Enable Power Management State Machine to enter L1. 0 – Disable 1 – Enable

pm_l1_min Register 0x64

This register set is used for the Power Management State Machine L1 re-entry control.

Table 5.109. pm_l1_min Register 0x64

Field	Name	Access	Width	Reset	Description
[31:24]	reserved	RO	8	0x0	—
[23:16]	ps_reentry_time	RW	8	0x0	Minimum number of µs to wait following an L1 exit when Power State != D0, before re-entering L1 due to Power State != D0. A wait time is needed to give the transaction layer time to process a Power State Cfg Write to D0 that caused L1 exit. 0 is a special case == 50 µs.
[15:0]	reserved	RO	16	0x0	—

pm_l1pmss Register 0x68

This register set is used for the Power Management State Machine L1PMSS control.

Table 5.110. pm_l1pmss Register 0x68

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	ds_drive_clkreq	RW	1	0x0	Enable driveing the clkreq_n signal when operating as a downstream port. 0 – Disable 1 – Enable

pm_l2 Register 0x70

This register set is used for the Power Management State Machine L2 entry control.

Table 5.111. pm_l2 Register 0x70

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	enable	RW	1	0x1	Enable Power Management State Machine to enter L2. 0 – Disable 1 – Enable

pm_pme_to_ack_ep Register 0x80

This register set is used for the Power Management State Machine Endpoint PME_TO_Ack control.

Table 5.112. pm_pme_to_ack_ep Register 0x80

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	user_auto_n	RW	1	0x0	For Endpoints only: PME_TO_Ack message transmission scheduling method. Endpoints are required to respond to a PME_Turn_Off message with a PME_TO_Ack message when they are ready to allow power down. 0 – Schedule PME_TO_Ack message automatically on reception of PME_Turn_Off message. 1 – Schedule PME_TO_Ack message under user control through the pm_l2_enter_ack rising edge.

pm_pme_to_ack_ds Register 0x84

This register set is used for the Power Management State Machine Downstream Port PME_TO_Ack control.

Table 5.113. pm_pme_to_ack_ds Register 0x84

Field	Name	Access	Width	Reset	Description
[31:8]	reserved	RO	24	0x0	—
[7:0]	timeout_threshold	RW	8	0x0	For Root Port only: ms to wait for a transmitted PME_Turn_Off Message to be acknowledged by receipt of PME_TO_Ack message before continuing with L2/L3 entry. 0xFF is a special case that disables the timeout mechanism. 0x00 is a special case == 10 ms.

pm_pme Register 0x88

This register set is used for the Power Management State Machine PM_PME control.

Table 5.114. pm_pme Register 0x88

Field	Name	Access	Width	Reset	Description
[31:12]	reserved	RO	20	0x0	—
[11:0]	timeout_threshold	RW	12	0x0	ms to wait for a transmitted PM_PME Message to be acknowledged, by clearing of the PME_Status register, before reissuing the PM_PME message. 0xFFF is a special case that disables the timeout mechanism. 0x000 is a special case == 100 ms.

pm_status Register 0x90

This register set is used for the Power Management State Machine Status.

Table 5.115. pm_status Register 0x90

Field	Name	Access	Width	Reset	Description
[31:5]	reserved	RO	27	0x0	—
[4:0]	state	RO	5	0x0	Power Management State Machine State. 0 – IDLE 1 – L1_WAIT_IDLE 2 – L1_WAIT_REPLAY 3 – L1_READY 4 – L1_STOP_DLLP 5 – L1 6 – L1_1 7 – L1_2_ENTRY 8 – L1_2_IDLE 9 – L1_2_EXIT 10 – L1_EXIT 11 – L2_WAIT_IDLE 12 – L2_WAIT_REPLAY 13 – L23_READY 14 – L2_STOP_DLLP 15 – L2 16 – L0S

5.1.3.3. Receive Buffer

vc_rx_c Register 0x108

This register set is used for the Receive Buffer completion handling configuration.

Table 5.116. vc_rx_c Register 0x108

Field	Name	Access	Width	Reset	Description
[31:2]	reserved	RO	30	0x0	—
[1]	force_ro	RW	1	0x0	Force completion relaxed ordering (RO==1) behavior for all completion TLPs, even those with RO==0. Note that setting this register to 1 is not PCIe Specification compliant but this may be fine for some designs since it is acceptable in many designs for C without RO==1 to pass prior P. 0 – Disable. Received completions are handled using the received RO attribute. 1 – Enable. All received completions are handled as if the RO attribute is 1.
[0]	priority	RW	1	0x0	Completion priority enable. 0 – Disable. Arbitration between Posted, Non-Posted, and Completion TLPs is round robin. 1 – Enable. While arbitrating between putting pending received Posted, Non-Posted, and Completion TLPs on the user received TLP interface, completions are given highest priority. Posted and non-posted requests transact only when a completion is not pending or as needed to prevent starving.

vc_rx_adv Register 0x10c

This register set is used for the Receive Buffer completion credit advertisement configuration.

Table 5.117. vc_rx_adv Register 0x10c

Field	Name	Access	Width	Reset	Description
[31:2]	reserved	RO	30	0x0	—
[1:0]	ch_cd_sel	RW	2	0x0	PCIe Specification requires CH and CD credit advertisements to be infinite for Endpoints and the finite (actual credit values) for Root Port. ch_cd_sel may be configured to over-ride the default PCIe Specification expected behavior. 0 – Implement CH, CD credit advertisements per port type: Endpoints == infinite, Root Port == actual. 1 – Advertise actual CH, CD credits. 2 – Advertise Infinite CH, CD credits

vc_rx_control Register 0x110

This register set is used for the Receive Buffer Parity/ECC control.

Table 5.118. vc_rx_control Register 0x110

Field	Name	Access	Width	Reset	Description
[31:20]	reserved	RO	12	0x0	—
[19]	reserved	RO	1	0x0	—
[18]	par1_report_disable	RW	1	0x0	Receive Buffer Prefix Parity Error and Parity 1 Error Reporting Disable. 0 – Enable reporting. 1 – Disable reporting of detected parity errors.
[17]	pfx_par_inject_en	RW	1	0x0	Receive Buffer Prefix Parity Error Injection Enable. 0 – Do not inject error. 1 – On the rising edge, a single prefix parity error injection is scheduled and is injected at the next opportunity (TLP receipt).

Field	Name	Access	Width	Reset	Description
[16]	par1_inject_en	RW	1	0x0	Receive Buffer Parity 1 Error Injection Enable. 0 – Do not inject error. 1 – On the rising edge, a single parity error injection is scheduled and is injected at the next opportunity (TLP receipt).
[15:13]	reserved	RO	3	0x0	—
[12]	ecc2_report_disable	RW	1	0x0	Receive Buffer ECC 2-bit Error Reporting Disable. 0 – Enable reporting. 1 – Disable reporting of ECC 2-bit errors.
[11]	ecc2_handle_disable	RW	1	0x0	Receive Buffer ECC 2-bit Error Handling Disable. 0 – Enable handling. 1 – Disable handling of ECC 2-bit errors.
[10:9]	reserved	RO	2	0x0	—
[8]	ecc2_inject_en	RW	1	0x0	Receive Buffer ECC 2-bit Error Injection Enable. 1 – On the rising edge, a single ECC 2-bit error injection is scheduled and is injected at the next opportunity (Receive Buffer RAM read).
[7:5]	reserved	RO	3	0x0	—
[4]	ecc1_report_disable	RW	1	0x0	Receive Buffer ECC 1-bit Error Reporting Disable. 0 – Enable reporting. 1 – Disable reporting of ECC 1-bit errors.
[3]	ecc1_handle_disable	RW	1	0x0	Receive Buffer ECC 1-bit Error Handling Disable. 0 – Enable correction. 1 – Disable correction of ECC 1-bit errors. When error correction is disabled, ECC 1-bit errors are treated the same as uncorrectable ECC 2-bit errors.
[2:1]	reserved	RO	2	0x0	—
[0]	ecc1_inject_en	RW	1	0x0	Receive Buffer ECC 1-bit Error Injection Enable. 1 – On the rising edge, a single ECC 1-bit error injection is scheduled and is injected at the next opportunity (Receive Buffer RAM read).

vc_rx_status Register 0x114

This register is used fort the Receive Buffer Parity/ECC status.

Table 5.119. vc_rx_status Register 0x114

Field	Name	Access	Width	Reset	Description
[31:4]	reserved	RO	28	0x0	—
[3]	err_pfx_par	RW, wr:oneToClear	1	0x0	Receive Buffer Prefix Parity Error Detection Status. 0 – Otherwise 1 – Error occurred.
[2]	err_par1	RW, wr:oneToClear	1	0x0	Receive Buffer Parity 1 Error Detection Status. 0 – Otherwise 1 – Error occurred.
[1]	err_ecc2	RW, wr:oneToClear	1	0x0	Receive Buffer ECC 2-bit Error Detection Status. 0 – Otherwise 1 – Error occurred.
[0]	err_ecc1	RW, wr:oneToClear	1	0x0	Receive Buffer ECC 1-bit Error Detection Status. 0 – Otherwise 1 – Error occurred.

vc_rx_credit_status_cfg Register 0x120

This register set is used for the Receive Buffer credit status configuration.

Table 5.120. vc_rx_credit_status_cfg Register 0x120

Field	Name	Access	Width	Reset	Description
[31:2]	reserved	RO	30	0x0	—
[1:0]	sel	RW	2	0x0	Receive Buffer credit status configuration selection. Configures which credit values are displayed in the remaining vc_rx_credit_status registers. 0 – Display static initial credit advertisements. 1 – Display dynamic current credits available. 2 – Display static number of credits implemented by the receiver buffer. This may be larger than the initial credit advertisement values when number of credits implemented exceeds the PCIe Specification advertised maximum values of 127 H and 2047 D. 3 – Reserved.

vc_rx_credit_status_p Register 0x124

This register set is used for the Receive Buffer Posted credit status.

Table 5.121. vc_rx_credit_status_p Register 0x124

Field	Name	Access	Width	Reset	Description
[31:16]	d	RO	16	0x0	Receive Buffer Posted Data credit status. Content dependent on the value of vc_rx_credit_status_cfg_sel.
[15:0]	h	RO	16	0x0	Receive Buffer Posted Header credit status. Content dependent on the value of vc_rx_credit_status_cfg_sel.

vc_rx_credit_status_n Register 0x128

This register set is used for the Receive Buffer Non-Posted credit status.

Table 5.122. vc_rx_credit_status_n Register 0x128

Field	Name	Access	Width	Reset	Description
[31:16]	d	RO	16	0x0	Receive Buffer Non-Posted Data credit status. Content dependent on the value of vc_rx_credit_status_cfg_sel.
[15:0]	h	RO	16	0x0	Receive Buffer Non-Posted Header credit status. Content dependent on the value of vc_rx_credit_status_cfg_sel.

vc_rx_credit_status_c Register 0x12c

This register set is used for the Receive Buffer Completion credit status.

Table 5.123. vc_rx_credit_status_c Register 0x12c

Field	Name	Access	Width	Reset	Description
[31:16]	d	RO	16	0x0	Receive Buffer Completion Data credit status. Content dependent on the value of vc_rx_credit_status_cfg_sel.
[15:0]	h	RO	16	0x0	Receive Buffer Completion Header credit status. Content dependent on the value of vc_rx_credit_status_cfg_sel.

vc_rx_f_oc_update_timer Register 0x130

This register set is used for the Receive Buffer FC Update Timer Control.

Table 5.124. vc_rx_f_oc_update_timer Register 0x130

Field	Name	Access	Width	Reset	Description
[31:3]	reserved	RO	29	0x0	—
[2:1]	div	RW	2	0x0	Divider Control to Reduce Period of FC Updates for Highly Latent Systems 0 – No adjustment 1 – Divide the PCIe Spec Guideline Value by 2 2 – Divide the PCIe Spec Guideline Value by 4 3 – Divide the PCIe Spec Guideline Value by 8
[0]	disable	RW	1	0x0	Disable Control for the FC Update Timer 0 – Enable the FC Update Timer. 1 – Disable FC Update Timer, sending FC Updates on Every Consumed RX Packet

vc_rx_p_flow_ctrl Register 0x134

Receive Buffer Posted TLP Flow Control.

Table 5.125. vc_rx_p_flow_ctrl Register 0x134

Field	Name	Access	Width	Reset	Description
[31:26]	reserved	RO	6	0x0	—
[25:16]	thresh	RW	10	0x10	Receive Buffer Posted TLP Flow Control Threshold Enable. Threshold to use when thresh_en == 1.
[15:9]	reserved	RO	7	0x0	—
[8]	thresh_en	RW	1	0x0	Receive Buffer Posted TLP Flow Control Threshold. 0 – Use threshold provided on vc_rx_p_thresh/ptl_rx_p_thresh input port. 1 – Use threshold provided by the thresh register
[7:1]	reserved	RO	7	0x0	—
[0]	disable	RW	1	0x0	Receive Buffer Posted TLP Flow Control Disable. 0 – Enable Posted TLP Flow Control. 1 – Disable Posted TLP Flow Control

vc_rx_n_flow_ctrl Register 0x138

This register set is used for the Receive Buffer Non-Posted TLP Flow Control.

Table 5.126. vc_rx_n_flow_ctrl Register 0x138

Field	Name	Access	Width	Reset	Description
[31:26]	reserved	RO	6	0x0	—
[25:16]	thresh	RW	10	0x10	Receive Buffer Non-Posted TLP Flow Control Threshold. 0 – Use threshold provided on vc_rx_np_thresh/ptl_rx_np_thresh input port. 1 – Use threshold provided by the thresh register.
[15:9]	reserved	RO	7	0x0	—
[8]	thresh_en	RW	1	0x0	Receive Buffer Non-Posted TLP Flow Control Disable. 0 – Enable Posted TLP Flow Control. 1 – Disable Posted TLP Flow Control.
[7:1]	reserved	RO	7	0x0	—
[0]	disable	RW	1	0x0	—

vc_rx_alloc_size Register 0x140

This register set is used for the Receive Buffer Size Information.

Table 5.127. vc_rx_alloc_size Register 0x140

Field	Name	Access	Width	Reset	Description
[31:24]	c_hdr	RO	8	0x0	Number of Receive Buffer Data storage bytes required for each unit of alloc_ch.
[23:16]	pn_hdr	RO	8	0x0	Number of Receive Buffer Data storage bytes required for each unit of alloc_ph and alloc_nh.
[15:8]	storage_data	RO	8	0x0	Receive Buffer Data Storage size in bytes == 2^storage_data. The storage space required for alloc_ph, alloc_nh, alloc_ch, alloc_pd, alloc_nd, and alloc_cd must be ≤ (2^storage_data) bytes. The space requried for each unit of alloc_ph, alloc_nh, and alloc_cd is indicated in the fields pn_hdr and c_hdr. 16 bytes of space is required for each unit of alloc_pd, alloc_nd, and alloc_cd. Credit allocations must fit within the Receive Buffer Data Storage, or the Receive Buffer is unable to function correctly, which leads to serious errors.
[7:0]	storage_hdr	RO	8	0x0	Receive Buffer Header Storage size in number of TLPs == 2^storage_hdr. alloc_ph + alloc_nh + alloc_ch must be ≤ 2^storage_hdr. Credit allocations must fit within the Receive Buffer Header Storage, or the Receive Buffer is unable to function correctly, which leads to serious errors.

vc_rx_alloc_p Register 0x144

This register set is used for the Receive Buffer Posted TLP Credit Allocation.

Table 5.128. vc_rx_alloc_p Register 0x144

Field	Name	Access	Width	Reset	Description
[31:16]	d	RW	16	0x440	Number of PCI Express PD credits to allocate in the Receive Buffer. The minimum value is (Max Payload Size Supported in Bytes/16). Number of PCI Express CD credits to allocate in the Receive Buffer. The minimum value is (Max Payload Size Supported in Bytes/16). If more than 2047 credits are allocated, the PCIe Core reserves the requested amount of space in the Receive Buffer, but it advertises the maximum allowed PCIe Specification value of 2047.
[15:12]	reserved	RO	4	0x0	—
[11:0]	h	RW	12	0x20	Number of PCI Express PH credits to allocate in the Receive Buffer. The minimum value is 1. If more than 127 credits are allocated, the PCIe Core reserves the requested amount of space in the Receive Buffer but it advertises the maximum allowed PCIe Specification value of 127.

vc_rx_alloc_n Register 0x148

This register set is used for the Receive Buffer Non-Posted TLP Credit Allocation.

Table 5.129. vc_rx_alloc_n Register 0x148

Field	Name	Access	Width	Reset	Description
[31:16]	d	RW	16	0x40	Number of PCI Express ND credits to allocate in the Receive Buffer. The minimum value is 2.
[15:12]	reserved	RO	4	0x0	—
[11:0]	h	RW	12	0x20	Number of PCI Express NH credits to allocate in the Receive Buffer. The minimum value is 1. If more than 127 credits are allocated, the PCIe Core reserves the requested amount of space in the Receive Buffer but it advertises the maximum allowed PCIe Specification value of 127.

vc_rx_alloc_c Register 0x14c

This register set is used for the Receive Buffer Completion TLP Credit Allocation.

Table 5.130. vc_rx_alloc_c Register 0x14c

Field	Name	Access	Width	Reset	Description
[31:16]	d	RW	16	0x700	Number of PCI Express CD credits to allocate in the Receive Buffer. The minimum value is (Max Payload Size Supported in Bytes/16). If more than 2047 credits are allocated, the PCIe Core reserves the requested amount of space in the Receive Buffer, but it advertises the maximum allowed PCIe Specification value of 2047 (or infinite credits when operating as an Endpoint as required by PCIe Specification.).
[15:12]	reserved	RO	4	0x0	—
[11:0]	h	RW	12	0x1c0	Number of PCI Express CH credits to allocate in the Receive Buffer. The minimum value is 1. If more than 127 credits are allocated, the PCIe Core reserves the requested amount of space in the Receive Buffer but it advertises the maximum allowed PCIe Specification value of 127.

vc_rx_alloc_error Register 0x150

This register set is used for the Receive Buffer Allocation Error Status.

Table 5.131. vc_rx_alloc_error Register 0x150

Field	Name	Access	Width	Reset	Description
[31:4]	reserved	RO	28	0x0	—
[3]	d_sum	RO	1	0x0	Receive Buffer Data Storage Allocation Error. The storage space required for alloc_ph, alloc_nh, alloc_ch, alloc_pd, alloc_nd, and alloc_cd must be ≤ (2^storage_data) bytes. The space requried for each unit of alloc_ph, alloc_nh, and alloc_cd is indicated in the fields pn_hdr and c_hdr. 16 bytes of space is required for each unit of alloc_pd, alloc_nd, and alloc_cd. Credit allocations must fit within the Receive Buffer Data Storage, or the Receive Buffer is unable to function correctly, which leads to serious errors. 0 – No Error 1 – Error
[2]	h_sum	RO	1	0x0	Receive Buffer Header Storage Allocation Error. alloc_ph + alloc_nh + alloc_ch must be ≤ 2^storage_hdr. Credit allocations must fit within the Receive Buffer Header Storage, or the Receive Buffer is unable to function correctly, which leads to serious errors. 0 – No Error 1 – Error
[1]	min_d	RO	1	0x0	Receive Buffer Data Credit Allocation Minimum Error. The alloc_pd and alloc_cd minimum value is Max Payload Size Supported. alloc_nd minimum value is 2. alloc_pd, alloc_nd, and alloc_cd must be a multiple of 2 for cores with DATA_WIDTH==256. 0 – No Error 1 – Error

Field	Name	Access	Width	Reset	Description
[0]	min_h	RO	1	0x0	Receive Buffer Header Credit Allocation Minimum Error. alloc_ph, alloc_nh, and alloc_ch must be > 0. 0 – No Error 1 – Error

vc_tx_np_fifo Register 0x180

This register set is used for the Transmit buffer non-posted TLP FIFO configuration.

Table 5.132. vc_tx_np_fifo Register 0x180

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	disable	RW	1	0x0	Transmit buffer non-posted TLP FIFO disable. The transmit non-posted TLP FIFO is present to allow Posted and Completion TLPs to continue to make progress, to avoid deadlock conditions, when Non-Posted TLP transmission in blocked by available credits in the link partner receive buffer. 0 – Enable non-posted TLP FIFO. PCIe Specification required value. 1 – For debug only: Disable non-posted TLP FIFO in which case non-posted TLPs are carried in the same data path as posted and completion TLPs.

vc_tx_status Register 0x184

This register set is used for the Transmit buffer status.

Table 5.133. vc_tx_status Register 0x184

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	remote_credit_block	RO	1	0x0	Set to one whenever a transmit TLP transmission is blocked by insufficient credits in the remote device's receive buffer.

vc_tx_credit_status_p Register 0x190

This register set is used for the TLP transmit Posted credits currently available in link partner receive buffer.

Table 5.134. vc_tx_credit_status_p Register 0x190

Field	Name	Access	Width	Reset	Description
[31:28]	reserved	RO	4	0x0	—
[27:16]	d	RO	12	0x0	Link partner current Receive Buffer Posted Data credits available.
[15:8]	reserved	RO	8	0x0	—
[7:0]	h	RO	8	0x0	Link partner current Receive Buffer Posted Header credits available.

vc_tx_credit_status_n Register 0x194

This register set is used for the TLP transmit Non-Posted credits currently available in link partner receive buffer.

Table 5.135. vc_tx_credit_status_n Register 0x194

Field	Name	Access	Width	Reset	Description
[31:28]	reserved	RO	4	0x0	—
[27:16]	d	RO	12	0x0	Link partner current Receive Buffer Non-Posted Data credits available.
[15:8]	reserved	RO	8	0x0	—
[7:0]	h	RO	8	0x0	Link partner current Receive Buffer Non-Posted Header credits available.

vc_tx_credit_status_c Register 0x198

This register set is used for the TLP transmit Completion credits currently available in link partner receive buffer.

Table 5.136. vc_tx_credit_status_c Register 0x198

Field	Name	Access	Width	Reset	Description
[31:28]	reserved	RO	4	0x0	—
[27:16]	d	RO	12	0x0	Link partner current Receive Buffer Completion Data credits available.
[15:8]	reserved	RO	8	0x0	—
[7:0]	h	RO	8	0x0	Link partner current Receive Buffer Completion Header credits available.

vc_tx_credit_cleanup Register 0x19c

This register set is used for the TLP transmit error credit cleanup control.

Table 5.137. vc_tx_credit_cleanup Register 0x19c

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	method	RW	1	0x0	TLP Transmit Credit Cleanup Method. 0 – Use the headers of the cleaned-up TLPs to recover the credits. The credits in TLPs with corrupted headers are not recovered. 1 – Use a credit lookup table based on the ID assigned to the TLP. This table is implemented in pcie_user_if.

5.1.3.4. TLP Transmit Control

tlp_tx Register 0x1c4

This register set is used to enable TD bit.

Table 5.138. tlp_tx Register 0x1c4

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	td1_means_add_has_n	RW	1	0x0	TLP Transmit TD==1 Header Field Interpretation. 0 – When a TLP is transmitted with TLP header bit TD==1, this means that the TLP already contains an ECRC. The core transmits the TLP with the TLP's existing ECRC and does not attempt to generate/append a new ECRC. 1 – Not supported for Full Transaction Layer cores like the CrossLink™-NX cores. td1_means_add_has_n must be set to 0.

5.1.3.5. FC Credit Init Control

fc_credit_init Register 0x1c8

This register set is used to force the core to reperform FC credit initialization.

Table 5.139. fc_credit_init Register 0x1c8

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	redo	RW	1	0x0	Force the core to redo FC Credit Initialization without taking the link down. This is only possible if both ends of the link are instructed to redo the initialization.

5.1.4. mgmt_ftl (0x4_4000)

5.1.4.1. Simulation Register

simulation Register 0x0

This register set is used for simulation only such as Full Transaction Layer simulation speed reduction.

Table 5.140. simulation Register 0x0

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	reduce_timeouts	RW	1	0x0	Reduce timeouts to shorten simulation time. When enabled ms timeouts are shortened to the value in µs. 0 – Disable 1 – Enable

5.1.4.2. Transaction Layer Decode Configuration Register

decode Register 0x10

This register set is used for the Transaction Layer Decode configuration.

Table 5.141. decode Register 0x10

Field	Name	Access	Width	Reset	Description
[31:24]	reserved	RO	8	0x0	—
[23:18]	reserved	RO	6	0x0	—
[17]	tx_bypass_decode_en	RW	1	0x0	Bypass the TLP decode block in the Transmit path. 0 – Decode_in_path module is enabled. 1 – Decode_in_path module is bypassed.
[16]	rx_bypass_decode_en	RW	1	0x0	Bypass the TLP decode block in the Receive path. 0 – Decode_in_path module is enabled. 1 – Decode_in_path module is bypassed.
[15:11]	reserved	RO	5	0x0	—
[10]	tx_convert_ur_to_ca	RW	1	0x0	When decoding TX packets convert Unsupported Request (UR) packets to Completer Abort (CA). 0 – Normal Operation. 1 – Convert UR to CA.
[9]	rx_convert_ur_to_ca	RW	1	0x0	When decoding RX packets convert Unsupported Request (UR) packets to Completer Abort (CA). 0 – Normal Operation. 1 – Convert UR to CA.
[8]	t0_rx_bypass_msg_dec	RW	1	0x0	When implementing Type 0 Configuration Space (Endpoint) – Bypass RX Message TLP Decode Enable. 0 – Normal operation. The core claims and does not forward Message TLPs to the TLP Receive Interface. 1 – All valid Msg TLPs received on PCIe (except Routed by ID and Routed by Address which are routed according to the routing type) are forwarded to the TLP Receive Interface.
[7:3]	reserved	RO	5	0x0	—
[2]	vendor0_ur	RW	1	0x1	Vendor Type 0 Messages received from PCIe are reported as UR. 0 – Do not report received Vendor Type 0 Messages as Unsupported Request (UR). 1 – Report received Vendor Type 0 Messages as Unsupported Request (UR).
[1]	target_only	RW	1	0x0	Target Only. Enable for user designs that implement purely target-only functionality. When enabled all received completions are considered Unexpected Completions and are not forwarded to the TLP Receive Interface. 0 – Disable 1 – Enable

Field	Name	Access	Width	Reset	Description
[0]	ignore_poison	RW	1	0x1	Ignore Poison – Set to 1 to have the core ignore the EP poison indicator for received TLPs with data payload that do not terminate in the core. When set to 1, the core passes all poisoned TLPs to you the same way it would pass the TLP if the TLP is not poisoned. Note that the Ignore Poison control is forced to 1 by the core when the core is configured as a Root-Port. Note that the following TLP types ignore the setting of this bit. Poisoned Configuration Type 0 writes is terminated in the core in all cases, independent of the Ignore Poison bit setting. A completion with UR status is generated and the appropriate error message, ERR COR or ERR FAT, is generated if not masked. Note that Poisoned Configuration Type 0 reads are always treated as if they were not poisoned. The read completes with successful completion status and an optional Advisory Non-Fatal Error status is set provided the severity level is set to NON-FATAL. Poisoned packets without data payload is passed to you in all cases since EP should not be set on packets without data payload and these packets should generally be handled as if they were not poisoned or alternatively handled as Advisory Non-Fatal Errors by user logic. Poisoned Vendor-defined Type 1 messages with data payload are always passed to you and, if ignore poison is 0, additionally an Advisory Non-Fatal Error status is set provided the severity level is set to NON-FATAL. When Ignore Poison is set to 0, the core handles the remaining poisoned TLPs with data payload as follows. Poisoned Write request and poisoned read completions with data TLPs are consumed by the core and handled as TLP Poisoned errors that generate the appropriate poison, ERR NON-FATAL or ERR FATAL, depending upon the error severity register error message. Poisoned Message with data payload (other than vendor-defined type 1) are consumed by the core and handled as TLP Poisoned errors that generate the appropriate poison, ERR NON-FATAL or ERR FATAL depending upon the error severity register, error message. The recommended default for target-only endpoints is to set Ignore Poison == 0 and to have user logic ignore the EP header bit on TLPs that it receives. In this case poisoned TLPs with data payload (other than config 0 writes and vendor-defined type 1 messages) generates a NON-FATAL error message and is discarded by the core. Poisoned TLPs without data payload (for which EP does not apply) is processed as if they were not poisoned. 0 – Disable 1 – Enable

decode_t1 Register 0x14

This register set is used for the Type 1 Configuration Space Transaction Layer Decode configuration.

Table 5.142. decode_t1 Register 0x14

Field	Name	Access	Width	Reset	Description
[31:24]	reserved	RO	8	0x0	—
[23:16]	reserved	RO	8	0x0	—
[15:11]	reserved	RO	5	0x0	—
[10]	bypass_addr_dec	RW	1	0x0	When implementing Type 1 Configuration Space (Root Port): Bypass TLP Address Decode Enable. bypass_addr_dec controls the decoding of received Memory and I/O Request TLPs on both the Primary and Secondary sides of the core. 0 – Normal PCI Express compliant address validity checks are performed. Transactions must target a valid region to be forwarded. Transactions with an invalid address cause an error response to be generated. Recommended value unless the customer application requires a value of 1 to be used.) 1 – The address validity checks for I/O, Memory, and Prefetchable Memory regions are bypassed. Memory Requests, I/O Requests, and Messages Routed by Address is considered valid regardless of address. bypass_addr_dec does not affect the other validity checks (Memory and I/O regions enabled, in a Power State where transactions can be accepted, link is up, and not poisoned). Transactions, which fail these other validity checks, still causes an error response to be generated. When bypass_addr_dec == 1, Memory Requests which appear on the Receive Interface indicates a hit to the Prefetchable Memory Window and I/O Requests which appear on the Receive Interface indicates a hit to the I/O Window.
[9]	tx_bypass_msg_dec	RW	1	0x0	When implementing Type 1 Configuration Space (Root Port): Bypass TX Message TLP Decode Enable. 0 – Normal operation. Claim and do not forward Local Routing messages and Error Messages that are transmitted without error propagation being enabled. Other message types are left in the stream. 1 – All valid Message TLPs transmitted towards PCIe (except Routed by ID and Routed by Address, which are routed according to the routing type) are forwarded to PCIe.
[8]	rx_bypass_msg_dec	RW	1	0x0	When implementing Type 1 Configuration Space (Root Port): Bypass RX Message TLP Decode Enable. 0 – Normal operation. Claim and do not forward Local Routing messages and Error Messages that are transmitted without error propagation being enabled. Other message types are left in the stream. 1 – All valid Message TLPs received from PCIe (except Routed by ID and Routed by Address which are routed according to the routing type) are forwarded to the TLP Receive Interface.
[7:0]	reserved	RO	8	0x0	—

5.1.4.3. Transaction Layer TLP Processing Configuration Register

tlp_processing Register 0x18

This register set is used for the Transaction Layer TLP Processing configuration.

Table 5.143. tlp_processing Register 0x18

Field	Name	Access	Width	Reset	Description
[31:24]	reserved	RO	8	0x0	—
[23:16]	reserved	RO	8	0x0	—
[15:8]	reserved	RO	8	0x0	—
[7:2]	reserved	RO	6	0x0	—
[1]	ignore_ecrc	RW	1	0x0	Ignore ECRC Error Enable. When enabled ECRC errors are ignored for TLPs passed to you in the TLP Receive Interface. 0 – Disable 1 – Enable
[0]	crs_enable	RW	1	0x0	Configuration Request Retry Status Enable. 0 – Disable. Type 0 Configuration Writes and Reads are performed normally. 1 – Enable. Type 0 Configuration Writes and Reads return Configuration Request Retry Status.

5.1.4.4. Initial Register

Initial Register 0x20

This register set is used for the initial speed and width configuration.

Table 5.144. Initial Register 0x20

Field	Name	Access	Width	Reset	Description
[31:19]	reserved	RO	13	0x0	—
[18:16]	max_link_width	RW	3	0x5	Max Link Width Override. This setting, if different from zero, overrides the value of Maximum Link Width in the PCIe Link Capabilities register. 0 – Maximum core lane width 1 – 1 lane 2 – 2 lanes 3 – 4 lanes 4 – 8 lanes 5 – 16 lanes 6 – Reserved6 7 – Reserved7
[15:2]	reserved	RO	14	0x0	—

Field	Name	Access	Width	Reset	Description
[1:0]	target_link_speed	RW	2	0x3	Initial value of Target Link Speed Configuration Register. Determines the maximum initial link speed which can be reached during initial training. Must be set to the lesser of the maximum speed supported by the core and the maximum speed at which you desired the core to operate. 0 – 2.5G 1 – 5.0G 2 – 8.0G 3 – 16.0G

5.1.4.5. Configuration Register type

cfg Register 0x30

This register set is used for the Configuration Register type.

Table 5.145. cfg Register 0x30

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	type1_type0_n	RW	1	0x0	Determines the type of Configuration Registers implemented by the core. 0 – Type 0 – Endpoint 1 – Type 1 – Root

5.1.4.6. Downstream Port Configuration

ds_port Register 0x34

This register set is used for the Downstream Port configuration.

Table 5.146. ds_port Register 0x34

Field	Name	Access	Width	Reset	Description
[31:17]	reserved	RO	15	0x0	—
[16]	rcb	RW	1	0x0	Read Completion Boundary (RCB). RCB value advertised when the core is operating as a Root Port.
[15:0]	id	RW	16	0x0	Root Port ID. This 16-bit field is used to define the ID used for PCIe Requester ID and Completer ID when the core is operating as a Root Port.

5.1.4.7. Upstream Port Configuration

us_port Register 0x38

This register set is used for the Upstream Port configuration.

Table 5.147. us_port Register 0x38

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	adv_target_link_speed	RW	1	0x0	For an upstream port, advertise the link speeds specified by the target_link_speed field rather than the maximum supported speed.

5.1.4.8. Device ID Configuration

id1 Register 0x40

This register set is used for the ID1 configuration.

Table 5.148. id1 Register 0x40

Field	Name	Access	Width	Reset	Description
[31:16]	device_id	RW	16	0xe004	Value returned when the Device ID Configuration Register is read.
[15:0]	vendor_id	RW	16	0x19aa	Value returned when the Vendor ID Configuration Register is read.

id2 Register 0x44

This register set is used for the ID2 configuration.

Table 5.149. id2 Register 0x44

Field	Name	Access	Width	Reset	Description
[31:16]	subsystem_id	RW	16	0xe004	Value returned when the Subsystem ID Configuration Register is read.
[15:0]	subsystem_vendor_id	RW	16	0x19aa	Value returned when the Subsystem Vendor ID Configuration Register is read.

id3 Register 0x48

This register set is used for the ID3 configuration.

Table 5.150. id3 Register 0x48

Field	Name	Access	Width	Reset	Description
[31:8]	class_code	RW	24	0x118000	Value returned when the Class Code Configuration Register is read. Must be set to the correct value for the type of device being implemented; see PCI Local Bus Specification Revision 2.3 Appendix D for details on setting Class Code.
[7:0]	revision_id	RW	8	0x4	Value returned when the Revision ID Configuration Register is read.

5.1.4.9. Cardbus Configuration

Cardbus Register 0x4c

This register set is used for the Cardbus configuration.

Table 5.151. Cardbus Register 0x4c

Field	Name	Access	Width	Reset	Description
[31:0]	cis_pointer	RW	32	0x0	Value returned when the Cardbus CIS Pointer Configuration Register is read. Set to 0x00000000 unless a Cardbus CIS structure is implemented in memory (which is rare), in which case set to the address of the CIS Structure.

5.1.4.10. Interrupt Configuration

interrupt Register 0x50

This register set is used for the Interrupt configuration.

Table 5.152. Interrupt Register 0x50

Field	Name	Access	Width	Reset	Description
[31:10]	reserved	RO	22	0x0	—
[9:8]	pin	RW	2	0x0	Selects which legacy interrupt is used. 0 – INTA 1 – INTB 2 – INTC 3 – INTD
[7:1]	reserved	RO	7	0x0	—
[0]	disable	RW	1	0x0	Disable support for interrupts. 0 – Enable 1 – Disable

5.1.4.11. BAR Configuration

bar0 Register 0x60

This register set is used for the BAR0 configuration.

Table 5.153. bar0 Register 0x60

Field	Name	Access	Width	Reset	Description
[31:0]	cfg	RW	32	0xffff000c	Configuration of BAR0 (Cfg address 0x10). Use to define a 32-bit Memory or I/O region or combine with an adjacent BAR to define a 64-bit Memory region.

bar1 Register 0x64

This register set is used for the BAR1 configuration.

Table 5.154. bar1 Register 0x64

Field	Name	Access	Width	Reset	Description
[31:0]	cfg	RW	32	0xffffffff	Configuration of bar1 (Cfg address 0x14). Use to define a 32-bit Memory or I/O region or combine with an adjacent BAR to define a 64-bit Memory region.

bar2 Register 0x68

This register set is used for the BAR2 configuration.

Table 5.155. bar2 Register 0x68

Field	Name	Access	Width	Reset	Description
[31:0]	cfg	RW	32	0xfffffe00c	Configuration of bar2 (Cfg address 0x18). Use to define a 32-bit Memory or I/O region or combine with an adjacent BAR to define a 64-bit Memory region.

bar3 Register 0x6c

This register set is used for the BAR3 configuration.

Table 5.156. bar3 Register 0x6c

Field	Name	Access	Width	Reset	Description
[31:0]	cfg	RW	32	0xffffffff	Configuration of bar3 (Cfg address 0x1C). Use to define a 32-bit Memory or I/O region or combine with an adjacent BAR to define a 64-bit Memory region.

bar4 Register 0x70

This register set is used for the BAR4 configuration.

Table 5.157. bar4 Register 0x70

Field	Name	Access	Width	Reset	Description
[31:0]	cfg	RW	32	0xfffffe00c	Configuration of bar4 (Cfg address 0x20). Use to define a 32-bit Memory or I/O region or combine with an adjacent BAR to define a 64-bit Memory region.

bar5 Register 0x74

This register set is used for the BAR5 configuration.

Table 5.158. bar5 Register 0x74

Field	Name	Access	Width	Reset	Description
[31:0]	cfg	RW	32	0xffffffff	Configuration of bar5 (Cfg address 0x24). Use to define a 32-bit Memory or I/O region or combine with an adjacent BAR to define a 64-bit Memory region.

5.1.4.12. Expansion ROM Configuration

exp_rom Register 0x78

This register set is used for the Expansion ROM configuration.

Table 5.159. exp_rom Register 0x78

Field	Name	Access	Width	Reset	Description
[31:0]	cfg	RW	32	0x0	Configuration of exp_rom. Use to define a 32-bit Memory Expansion ROM region. If an Expansion ROM region is defined, the region must map to PCIe-compliant Expansion ROM code, or the device may fail to boot.

5.1.4.13. PCI Express Configuration

pcie_cap Register 0x80

This register set is used for the PCI Express Capabilities configuration.

Table 5.160. pcie_cap Register 0x80

Field	Name	Access	Width	Reset	Description
[31:14]	reserved	RO	18	0x0	—
[13:9]	interrupt_message_number	RW	5	0x0	MSI/MSI-X vector is used for the interrupt message generated in association with any of the status bits of the PCI Express Capability structure.
[8]	slot_implemented	RW	1	0x0	Indicates that the Link associated with this Port is connected to a slot. This field is valid for Downstream Ports only.
[7:4]	device_port_type	RW	4	0x0	Indicates the specific type of this PCI Express Function. 0 – PCI Express Endpoint 1 – Legacy PCI Express Endpoint 2 – Reserved 3 – Reserved 4 – Root Port of PCI Express Root Complex 5 – Upstream Port of PCI Express Switch 6 – Downstream Port of PCI Express Switch 7 – PCI Express to PCI/PCI-X Bridge 8 – PCI/PCI-X to PCI Express Bridge 9 – Root Complex Integrated Endpoint 10 – Root Complex Event Collector 11 – Reserved 12 – Reserved 13 – Reserved 14 – Reserved 15 – Reserved
[3:0]	capability_version	RW	4	0x2	Indicates PCI-SIG defined PCI Express Capability structure version number. Must be set to 0x2.

pcie_dev_cap Register 0x84

This PCI Express Device Capabilities configuration.

Table 5.161. pcie_dev_cap Register 0x84

Field	Name	Access	Width	Reset	Description
[31:29]	reserved	RO	3	0x0	—
[28]	disable_flr_capability	RW	1	0x0	Function Level Reset Capability 0 – Enable 1 – Disable
[27:26]	reserved	RO	2	0x0	—
[25:18]	reserved	RO	8	0x0	—
[17:16]	reserved	RO	2	0x0	—
[15]	reserved	RO	1	0x0	—
[14:13]	reserved	RO	2	0x0	—
[12]	extended_tag_field_en_default	RW	1	0x1	Extended Tag Field Enable Default Value. PCIe Specification allows the Extended Tag Field Enable register to reset to either 1 or 0. This register determines the reset value. 0 – 5-bit Tag field enabled on reset 1 – 8-bit Tag field enabled on reset
[11:9]	endpoint_l1_acceptable_latency	RW	3	0x0	Endpoint L1 Acceptable Latency 0 – Maximum of 1 µs. Must be 0 when not an Endpoint. 1 – Maximum of 2 µs 2 – Maximum of 4 µs 3 – Maximum of 8 µs 4 – Maximum of 16 µs 5 – Maximum of 32 µs 6 – Maximum of 64 µs 7 – No limit
[8:6]	endpoint_l0s_acceptable_latency	RW	3	0x0	Endpoint L0s Acceptable Latency 0 – Maximum of 64 ns. Must be 0 when not an Endpoint. 1 – Maximum of 128 ns 2 – Maximum of 256 ns 3 – Maximum of 512 ns 4 – Maximum of 1 µs 5 – Maximum of 2 µs 6 – Maximum of 4 µs 7 – No limit
[5]	extended_tag_field_supported	RW	1	0x1	Extended Tag Field Supported 0 – 5-bit Tag field supported 1 – 8-bit Tag field supported
[4:3]	phantom_functions_supported	RW	2	0x0	Phantom Functions Supported 0 – No Function Number bits are used for Phantom Functions 1 – The most significant bit of the Function number in Requester ID is used for Phantom Functions 2 – The two most significant bits of Function Number in Requester ID are used for Phantom Functions 3 – All 3 bits of Function Number in Requester ID used for Phantom Functions.

Field	Name	Access	Width	Reset	Description
[2:0]	max_payload_size_supported	RW	3	0x2	Max Payload Size Supported 0 – 128 Bytes 1 – 256 Bytes 2 – 512 Bytes 3 – 1024 Bytes 4 – 2048 Bytes 5 – 4096 Bytes 6 – Reserved 7 – Reserved

pcie_link_cap Register 0x88

This register set is used for the PCI Express Link Capabilities configuration.

Table 5.162. pcie_link_cap Register 0x88

Field	Name	Access	Width	Reset	Description
[31:24]	port_number	RW	8	0x0	Indicates the PCI Express Port number for the PCI Express Link.
[23:18]	reserved	RO	6	0x0	—
[17:15]	l1_exit_latency	RW	3	0x7	L1 Exit Latency. The value reported indicates the length of time this Port requires to complete transition from ASPM L1 to L0. 0 – Less than 1 µs 1 – 1 µs to less than 2 µs 2 – 2 µs to less than 4 µs 3 – 4 µs to less than 8 µs 4 – 8 µs to less than 16 µs 5 – 16 µs to less than 32 µs 6 – 32 µs to 64 µs 7 – More than 64 µs
[14:12]	l0s_exit_latency	RW	3	0x7	L0s Exit Latency. The value reported indicates the length of time this Port requires to complete transition from ASPM L0s to L0. 0 – Less than 64 ns 1 – 64 ns to less than 128 ns 2 – 128 ns to less than 256 ns 3 – 256 ns to less than 512 ns 4 – 512 ns to less than 1 µs 5 – 1 µs to less than 2 µs 6 – 2 µs to 4 µs 7 – More than 4 µs
[11:10]	aspm_support	RW	2	0x3	Active State Power Management (ASPM) Support 0 – No ASPM Support 1 – L0s Supported 2 – L1 Supported 3 – L0s and L1 Supported
[9:0]	reserved	RO	10	0x0	—

pcie_link_stat Register 0x8c

This register set is used for the PCI Express Link Status configuration.

Table 5.163. pcie_link_stat Register 0x8c

Field	Name	Access	Width	Reset	Description
[31:13]	reserved	RO	19	0x0	—
[12]	slot_clock_configuration	RW	1	0x1	Indicates whether the component uses the physical reference clock that the platform provides on the connector. 0 – Using independent reference clock. 1 – Using reference clock provided by slot.
[11:0]	reserved	RO	12	0x0	—

pcie_slot_cap Register 0x90

This register set is used for the PCI Express Slot Capabilities configuration.

Table 5.164. pcie_slot_cap Register 0x90

Field	Name	Access	Width	Reset	Description
[31:19]	physical_slot_number	RW	13	0x1	Indicates whether the physical slot number attached to this Port. This field must be hardware initialized to a value that assigns a slot number that is unique within the chassis, regardless of the form factor associated with the slot. This field must be initialized to zero for Ports connected to devices that are either integrated on the system board or integrated within the same silicon as the Root Port.
[18]	no_command_completed_support	RW	1	0x0	Indicates whether the slot generates software notification when an issued command is completed by the Hot-Plug Controller. This bit is only permitted to be 1 if the hot-plug capable Port can accept writes to all fields of the Slot Control register without delay between successive writes. 0 – Software notification provided. 1 – Software notification not provided.
[17]	em_interlock_present	RW	1	0x0	Indicates whether an Electromechanical Interlock is implemented on the chassis for this slot. 0 – Not Supported 1 – Supported
[16:15]	slot_power_limit_scale	RW	2	0x0	Slot Power Limit Scale. In combination with the Slot Power Limit Value, it specifies the upper limit on power supplied by the slot or by other means to the adapter. Refer to PCIe Specification section for details.

Field	Name	Access	Width	Reset	Description
[14:7]	slot_power_limit_value	RW	8	0xa	Slot Power Limit Value. In combination with the Slot Power Limit Scale, it specifies the upper limit on power supplied by the slot or by other means to the adapter. Refer to PCIe Specification section for details.
[6]	hot_plug_capable	RW	1	0x0	Indicates whether this slot can support hot-plug operations. 0 – Not Supported 1 – Supported
[5]	hot_plug_surprise	RW	1	0x0	Indicates whether an adapter present in this slot might be removed from the system without any prior notification. This is a form factor specific capability. This bit is an indication to the operating system to allow for such removal without impacting continued software operation. 0 – Hot Plug Surprise not possible 1 – Hot Plug Surprise possible
[4]	power_indicator_present	RW	1	0x0	Indicates whether a Power Indicator is electrically controlled by the chassis for this slot. 0 – Not Supported 1 – Supported
[3]	attention_indicator_present	RW	1	0x0	Indicates whether an Attention Indicator is electrically controlled by the chassis. 0 – Not Supported 1 – Supported
[2]	mrl_sensor_present	RW	1	0x0	Indicates whether a MRL Sensor is implemented on the chassis for this slot. 0 – Not Supported 1 – Supported
[1]	power_controller_present	RW	1	0x0	Indicates whether a software programmable Power Controller is implemented for this slot/adapter. 0 – Not Supported 1 – Supported
[0]	attention_button_present	RW	1	0x0	Indicates whether an Attention Button for this slot is electrically controlled by the chassis. 0 – Not Supported 1 – Supported

pcie_dev_cap2 Register 0x98

This register set is used for the PCI Express Device Capabilities 2 configuration.

Table 5.165. pcie_dev_cap2 Register 0x98

Field	Name	Access	Width	Reset	Description
[31:22]	reserved	RO	10	0x0	—
[21]	end_end_prefixes_supported	RW	1	0x0	End-End TLP Prefix Supported 0 – Not Supported 1 – Supported
[20:19]	reserved	RO	2	0x0	—
[18]	obff_supported	RW	1	0x0	OBFF Supported 0 – OBFF Not Supported 1 – OBFF supported using Message signaling only
[17:16]	reserved	RO	2	0x0	—
[15:8]	reserved	RO	8	0x0	—
[7:5]	reserved	RO	3	0x0	—
[4]	cpl_timeout_disable_supported	RW	1	0x1	Completion Timeout Disable Supported. Completion timeout is not implemented by the core, so the advertised value must match the capabilities of the connected design which is implementing completion timeouts. 0 – Not Supported 1 – Supported

Field	Name	Access	Width	Reset	Description
[3:0]	cpl_timeout_ranges_supported	RW	4	0x0	Completion Timeout Ranges Supported advertised value. Completion timeout is not implemented by the core, so the advertised value must match the capabilities of the connected design which is implementing completion timeouts. 0 – Completion Timeout programming not supported. Timeout value in the range 50 µs to 50 ms is used. 1 – Range A (50 µs to 10 ms) 2 – Range B (10 ms to 250 ms) 3 – Range A (50 µs to 10 ms) and B (10 ms to 250 ms) 4 – Range B (10 ms to 250 ms) and C (250 ms to 4 s) 5 – Range A (50 µs to 10 ms) and B (10 ms to 250 ms) and C (250 ms to 4 s) 6 – Range B (10 ms to 250 ms) and C (250 ms to 4 s) and D (4 s to 64 s) 7 – Range A (50 µs to 10 ms) and B (10 ms to 250 ms) and C (250 ms to 4 s) and D (4 s to 64 s) 8 – Reserved 9 – Reserved 10 – Reserved 11 – Reserved 12 – Reserved 13 – Reserved 14 – Reserved 15 – Reserved

pcie_link_ctl2 Register 0xa0

This register set is used for the PCI Express Link Control 2 configuration.

Table 5.166. pcie_link_ctl2 Register 0xa0

Field	Name	Access	Width	Reset	Description
[31:7]	reserved	RO	25	0x0	—
[6]	selectable_deemphasis	RW	1	0x0	Selectable Deemphasis setting for Root Port only: When the Link is operating at 5.0 GT/s speed, this bit is used to control the transmit deemphasis of the link. 0 – -6 dB 1 – -3.5 dB
[5:0]	reserved	RO	6	0x0	—

5.1.4.14. Power Management Configuration

pm_cap Register 0xc0

This register set is used for the Power Management Capabilities configuration.

Table 5.167. pm_cap Register 0xc0

Field	Name	Access	Width	Reset	Description
[31:16]	reserved	RO	16	0x0	—
[15:11]	pme_support	RW	5	0x1f	PME Support. Indicates the power states from which the function may generate a PME. For each power state {D3Cold, D3hot, D2, D1, D0}: 0 – PME# not supported 1 – PME# supported
[10]	d2_support	RW	1	0x1	D2 Power Management State support. 0 – Not supported 1 – Supported
[9]	d1_support	RW	1	0x1	D1 Power Management State support. 0 – Not supported 1 – Supported
[8:6]	aux_current	RW	3	0x0	Aux Current. Reports on the 3.3 Vaux auxiliary current requirements for the PCI function. See PCIe Specification for details. 0 – Self-powered 1 – 55 mA 2 – 100 mA 3 – 160 mA 4 – 220 mA 5 – 270 mA 6 – 320 mA 7 – 375 mA
[5]	dsi	RW	1	0x0	Device Specific Initialization. Indicates whether a special initialization of this function is required (beyond the standard PCI configuration header) before the generic class device driver can use it. 0 – No Device Specific Initialization necessary. 1 – Function requires a device specific initialization sequence following transition to the D0 uninitialized state.
[4]	reserved	RO	1	0x0	—
[3]	pme_clock	RW	1	0x0	PME Clock. Does not apply to PCI Express and must be 0.
[2:0]	version	RW	3	0x3	PCI Power Management Interface Specification Version. Must be set to 0x3 to indicate revision 1.2 of the PCI Power Management Interface Specification.

pm Register 0xc4

This register set is used for the Power Management Control/Status configuration.

Table 5.168. pm Register 0xc4

Field	Name	Access	Width	Reset	Description
[31:24]	data	RW	8	0x0	—
[23]	pmcsr_bus_p_c_en	RW	1	0x0	—
[22]	pmcsr_b2_b3_support	RW	1	0x0	—
[21:16]	reserved	RO	6	0x0	—
[15]	reserved	RO	1	0x0	—
[14:13]	cstat_data_scale	RW	2	0x0	0 – Unknown scale 1 – power = data × 0.1 Watts 2 – power = data × 0.01 Watts 3 – power = data × 0.001 Watts
[12:9]	cstat_data_select	RW	4	0x0	0 – D0 Power Consumed 1 – D1 Power Consumed 2 – D2 Power Consumed 3 – D3 Power Consumed 4 – D0 Power Dissipated 5 – D1 Power Dissipated 6 – D2 Power Dissipated 7 – D3 Power Dissipated 8 – Common logic power consumption. For multifunction devices, reported in Function 0 only. 9 – Reserved 10 – Reserved 11 – Reserved 12 – Reserved 13 – Reserved 14 – Reserved 15 – Reserved
[8:0]	reserved	RO	9	0x0	—

pm_aux Register 0xc8

This register set is used for the Power Management Auxiliary Power configuration.

Table 5.169. pm_aux Register 0xc8

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—

[0]	power_required	RW	1	0x0	• Identifies whether the design requires auxiliary power.
					  • 0 – Aux Power is not required.
					  • 1 – Aux Power is required. If Aux Power is required, PME is advertised supported from D3 Cold, or advertised.
					• aux_current != 0, then the value of Aux Power PM Enable is sticky and preserved through conventional reset when
					• Aux Power is provided.

5.1.4.15. ARI Capability Configuration

ari_cap Register 0xe0

This register set is used for the ARI Capability configuration.

Table 5.170. ari_cap Register 0xe0

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	disable	RW	1	0x0	ARI Capability Disable When disabled, the ARI Capability does not appear in PCIe Configuration Space. This must be enabled when SR-IOV is enabled and must be disabled for downstream ports, Root Complex Integrated Endpoints, and Root Complex Event Collectors.

aer_cap Register 0x100

This register set is used for the AER Capability configuration.

Table 5.171. aer_cap Register 0x100

Field	Name	Access	Width	Reset	Description
[31]	en_tlp_prefix_blocked	RW	1	0x0	Enable TLP Prefix Blocked error reporting. 0 – Disable 1 – Enable
[30]	en_atomicop_egress_blocked	RW	1	0x0	Enable AtomicOp Egress Blocked error reporting. 0 – Disable 1 – Enable
[29]	en_mc_blocked_tlp	RW	1	0x0	Enable MC Blocked TLP error reporting. Not supported by core, so must be 0. 0 – Disable 1 – Enable
[28]	en_ucorr_internal_error	RW	1	0x0	Enable Uncorrectable Internal Error. 0 – Disable 1 – Enable
[27]	en_acs_violation	RW	1	0x0	Enable ACS Violation error reporting. Not supported by core, so must be 0. 0 – Disable 1 – Enable

Field	Name	Access	Width	Reset	Description
[26]	en_receiver_overflow	RW	1	0x0	Enable Receiver Overflow error reporting. Not supported by core, so must be 0. 0 – Disable 1 – Enable
[25]	en_completer_abort	RW	1	0x0	Enable Completer Abort error reporting. 0 – Disable 1 – Enable
[24]	en_completion_timeout	RW	1	0x1	Enable Completion Timeout error reporting. 0 – Disable 1 – Enable
[23]	en_surprise_down_error	RW	1	0x0	Enable Surprise Down Error reporting. 0 – Disable 1 – Enable
[22]	en_corr_internal_error	RW	1	0x0	Enable Correctable Internal Error reporting. 0 – Disable 1 – Enable
[21:16]	reserved	RO	6	0x0	—
[15:2]	reserved	RO	14	0x0	—
[1]	ecrc_gen_chk_capable	RW	1	0x1	ECRC Generation/Checking Capable. 0 – Not supported 1 – Supported
[0]	version	RW	1	0x0	AER Capability Version. 0 – Version 0x1 1 – Version 0x2

5.1.4.16. MSI Capability Configuration

msi_cap Register 0xe8

This register set is used for the MSI Capability configuration.

Table 5.172. msi_cap Register 0xe8

Field	Name	Access	Width	Reset	Description
[31:8]	reserved	RO	24	0x0	—
[7]	reserved	RO	1	0x0	—
[6:4]	mult_message_capable	RW	3	0x5	Number of requested MSI vectors. 0 – 1 1 – 2 2 – 4 3 – 8 4 – 16 5 – 32 6 – Reserved 7 – Reserved
[3:2]	reserved	RO	2	0x0	—
[1]	vec_mask_capable	RW	1	0x1	MSI Capability Per Vector Mask Capable. 0 – Disable 1 – Enable
[0]	disable	RW	1	0x0	MSI Capability Disable. When disabled, the MSI Capability does not appear in PCIe Configuration Space. 0 – Enable 1 – Disable

msix_cap Register 0xf0

This register set is used for the MSI-X Capability configuration.

Table 5.173. msix_cap Register 0xf0

Field	Name	Access	Width	Reset	Description
[31:27]	reserved	RO	5	0x0	—
[26:16]	table_size	RW	11	0x1f	Number of requested MSI-X vectors == (table_size+1).
[15:1]	reserved	RO	15	0x0	—
[0]	disable	RW	1	0x0	MSI-X Capability Disable. When disabled, the MSI-X Capability does not appear in PCIe Configuration Space. 0 – Enable 1 – Disable

msix_table Register 0xf4

This register set is used for the MSI-X Capability – MSI-X Table configuration.

Table 5.174. msix_table Register 0xf4

Field	Name	Access	Width	Reset	Description
[31:3]	offset	RW	29	0xc00	{offset, 3'b000} == byte address offset, within the BAR selected by bir, at which the MSI-X Table begins.
[2:0]	bir	RW	3	0x0	Indicates which Base Address register, located beginning at 10h in Configuration Space, is used to map the MSI-X Table into Memory Space. 0 – 0x10 (BAR0) 1 – 0x14 (BAR1) 2 – 0x18 (BAR2) 3 – 0x1C (BAR3) 4 – 0x20 (BAR4) 5 – 0x24 (BAR5) 6 – Reserved 7 – Reserved

msix_pba Register 0xf8

This register set is used for the MSI-X Capability – MSI-X PBA configuration.

Table 5.175. msix_pba Register 0xf8

Field	Name	Access	Width	Reset	Description
[31:3]	offset	RW	29	0xe00	{offset, 3'b000} == byte address offset, within the BAR selected by bir, at which the MSI-X PBA begins
[2:0]	bir	RW	3	0x0	Indicates which Base Address register, located beginning at 10h in Configuration Space, is used to map the MSI-X PBA into Memory Space. 0 – 0x10 (BAR0) 1 – 0x14 (BAR1) 2 – 0x18 (BAR2) 3 – 0x1C (BAR3) 4 – 0x20 (BAR4) 5 – 0x24 (BAR5) 6 – Reserved 7 – Reserved

5.1.4.17. Vendor-Specific Capability Configuration

vsec_cap Register 0x110

This register set is used for the Vendor-Specific Capability configuration.

Table 5.176. vsec_cap Register 0x110

Field	Name	Access	Width	Reset	Description
[31:16]	id	RW	16	0x1	Vendor-Specific Capability ID.
[15:1]	reserved	RO	15	0x0	—
[0]	enable	RW	1	0x1	Vendor-Specific Capability Enable. When disabled, the VSEC Capability does not appear in PCIe Configuration Space. 0 – Disable 1 – Enable

5.1.4.18. SRIS Capability Configuration

sris_cap Register 0x120

This register set is used for the SRIS Capability configuration.

Table 5.177. sris_cap Register 0x120

Field	Name	Access	Width	Reset	Description
[31:16]	reserved	RO	16	0x0	—
[15:12]	low_skp_generation_speeds	RW	4	0x0	SRIS Lower SKP OS Generation Supported Speeds Vector advertisement
[11:8]	low_skp_reception_speeds	RW	4	0x0	SRIS Lower SKP OS Reception Supported Speeds Vector advertisement
[7:1]	reserved	RO	7	0x0	—
[0]	enable	RW	1	0x0	SRIS Capability Enable. When disabled, the SRIS Capability does not appear in PCIe Configuration Space. 0 – Disable 1 – Enable

5.1.4.19. Device Serial Number

dsn_cap Register 0x130

This register set is used for the DSN capable cores only such as Device Serial Number Capability configuration.

Table 5.178. dsn_cap Register 0x130

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	enable	RW	1	0x0	Device Serial Number Capability Enable. When disabled, the Device Serial Number Capability does not appear in PCIe Configuration Space. 0 – Disable 1 – Enable

dsn_serial Register 0x134

This register set is used for the Device Serial Number Capability – Serial Number.

Table 5.179. dsn_serial Register 0x134

Field	Name	Access	Width	Reset	Description
[63:0]	number	RW	64	0x0	Device Serial Number.

5.1.4.20. Power Budgeting Capability Configuration

pwr_budget_cap Register 0x150

This register set is used for the Power Budgeting Capability configuration.

Table 5.180. pwr_budget_cap Register 0x150

Field	Name	Access	Width	Reset	Description
[31:2]	reserved	RO	30	0x0	—
[1]	sys_alloc	RW	1	0x0	Power Budgeting System Allocated. 0 – Power Budget should use Power Budgeting Capability Values 1 – Power Budget is System Allocated
[0]	enable	RW	1	0x0	Power Budgeting Capability Enable. When disabled, the Power Budgeting Capability does not appear in PCIe. Configuration Space. 0 – Disable 1 – Enable

5.1.4.21. Dynamic Power Allocation Configuration

dpa_cap Register 0x158

This register set is used for the Dynamic Power Allocation Capability configuration.

Table 5.181. dpa_cap Register 0x158

Field	Name	Access	Width	Reset	Description
[31:24]	xlcy1	RW	8	0x0	Transition Latency Value 1. When the Transition Latency Indicator for a substate is 1, this value is multiplied by the Transition Latency Unit to determine the maximum Transition Latency for the substate.
[23:16]	xlcy0	RW	8	0x0	Transition Latency Value 0. When the Transition Latency Indicator for a substate is 0, this value is multiplied by the Transition Latency Unit to determine the maximum Transition Latency for the substate.
[15:14]	reserved	RO	2	0x0	—
[13:12]	pas	RW	2	0x0	Power Allocation Scale. The value of the substate Power Allocation Register is multiplied by the decoded value of this field to determine the power allocation of the substate. 0 – 10x 1 – 1x 2 – 0.1x 3 – 0.01x
[11:10]	reserved	RO	2	0x0	—

Field	Name	Access	Width	Reset	Description
[9:8]	tlunit	RW	2	0x0	Transition Latency Unit. The substate Transition Latency Value is multiplied by the decoded Transition Latency Unit to Determine the maximum Transition Latency for the substate. 0 – 1 ms 1 – 10 ms 2 – 100 ms 3 – Reserved
[7:3]	substate_max	RW	5	0x0	Substate_Max. Specifies the maximum substate number. Substates from [substate_max:0] are supported. For example, substate_max==0 indicates support for 1 substate.
[2:1]	reserved	RO	2	0x0	—
[0]	enable	RW	1	0x0	Dynamic Power Allocation (DPA) Capability Enable. When disabled, the Dynamic Power Allocation Capability does not appear in PCIe Configuration Space. 0 – Disable 1 – Enable

dpa_xlcy Register 0x15c

This register set is used for the Dynamic Power Allocation – Transition Latency.

Table 5.182. dpa_xlcy Register 0x15c

Field	Name	Access	Width	Reset	Description
[31:0]	indicator	RW	32	0x0	Transition Latency Indicator. Indicates which Transition Latency Value applies to each substate. For each substate[i], indicator[i] indicates which Transition Latency Value applies: 0 – Use Transition Latency Value 0 1 – Use Transition Latency Value 1

dpa_alloc Register 0x160

This register set is used for the Dynamic Power Allocation Capability – Dynamic Power Allocation Array.

Table 5.183. dpa_alloc Register 0x160

Field	Name	Access	Width	Reset	Description
[255:0]	array	RW	256	0x0	Substate Power Allocation Array. For each substate[i], multiply array[(i*8)+7i*8)] times the Power Allocation Scale to determine the power allocation in Watts for the associated substate.

5.1.4.22. Latency Tolerance Reporting Capability Configuration

ltr_cap Register 0x180

This register set is used for the Latency Tolerance Reporting Capability configuration.

Table 5.184. ltr_cap Register 0x180

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	enable	RW	1	0x0	Latency Tolerance Reporting Capability Enable. When disabled, the Latency Tolerance Reporting Capability does not appear in PCIe Configuration Space. 0 – Disable 1 – Enable

5.1.4.23. L1 PM Substates Capability Configuration

l1pmss_cap Register 0x188

This register set is used for the L1 PM Substates Capability configuration.

Table 5.185. l1pmss_cap Register 0x188

Field	Name	Access	Width	Reset	Description
[31:24]	cm_restore_time	RW	8	0x0	Default Common Mode Restore Time. Default time, in microseconds, used by the Downstream Port for timing the re establishment of common mode. See the L1 PM Substates ECN for further details.
[23:16]	port_cm_restore_time	RW	8	0x0	Port Common Mode Restore Time. Time, in microseconds, required for this port to re-establish common mode. See the L1 PM Substates ECN for further details
[15:11]	port_tpower_on_value	RW	5	0x0	Port TPOWER_ON Value. Required for ports supporting PCI-PM L1.2 or ASPM L1.2. The value of TPOWER_ON is calculated by multiplying the value in this field by the decoded TPOWER_ON Scale field.
[10]	reserved	RO	1	0x0	—
[9:8]	port_tpower_on_scale	RW	2	0x0	Port TPOWER_ON Scale. Required for ports supporting PCI-PM L1.2 or ASPM L1.2. 0 – 2 µs 1 – 10 µs 2 – 100 µs 3 – Reserved
[7]	pcipm_l1_1_supported	RW	1	0x1	PCI-PM L1.1 Substate Supported. Must be set to 1 for all ports supporting L1 PM Substates. 0 – Not supported 1 – Supported
[6]	pcipm_l1_2_supported	RW	1	0x1	PCI-PM L1.2 Substate Supported. 0 – Not supported 1 – Supported
[5]	aspm_l1_1_supported	RW	1	0x1	ASPM L1.1 Substate Supported. 0 – Not supported 1 – Supported
[4]	aspm_l1_2_supported	RW	1	0x1	ASPM L1.2 Substate Supported. 0 – Not supported 1 – Supported
[3]	l1pm_supported	RW	1	0x1	L1 PM Substates Supported. 0 – Not supported 1 – Supported

Field	Name	Access	Width	Reset	Description
[2:1]	reserved	RO	2	0x0	—
[0]	enable	RW	1	0x0	L1 PM Substates Capability Enable. When disabled, the L1 PM Substates Capability does not appear in PCIe. Configuration Space. 0 – Disable 1 – Enable

5.1.4.24. Atomic Op Capability Configuration

atomic_op_cap Register 0x1cc

This register set is used for the Atomic Op Capability configuration.

Table 5.186. atomic_op_cap Register 0x1cc

Field	Name	Access	Width	Reset	Description
[31:6]	reserved	RO	26	0x0	—
[5]	rp_completer_enable	RW	1	0x0	Enable Root Port to be an Atomic Op Completer which means that the Root Port completes rather than forwards Atomic Op TLPs. 0 – Disable 1 – Enable
[4]	completer_128_supported	RW	1	0x0	Atomic Op Completer 128-bit Operand Support. 0 – Not Supported 1 – Supported
[3]	completer_64_supported	RW	1	0x0	Atomic Op Completer 64-bit Operand Support. 0 – Not Supported 1 – Supported
[2]	completer_32_supported	RW	1	0x0	Atomic Op Completer 32-bit Operand Support. 0 – Not Supported 1 – Supported
[1]	routing_supported	RW	1	0x0	Atomic Op Routing Supported. 0 – Not Supported 1 – Supported
[0]	enable	RW	1	0x0	Atomic Op Capability Enable. When disabled, the Atomic Op Capability does not appear in PCIe Configuration Space. 0 – Disable 1 – Enable

5.1.5. mgmt_ftl_mf[7:1] (0x4_5000,0x4_6000,0x4_7000,0x4_8000,0x4_9000,0x4_A000,0x4_B000)

The base address for mgmt._ftl_mf is shown in Table 5.187.

Table 5.187. Base Address for mgmt_ftl_mf

Port	Base Address
mgmt_ftl_mf1_BASE	0x4_5000
mgmt_ftl_mf2_BASE	0x4_6000
mgmt_ftl_mf3_BASE	0x4_7000
mgmt_ftl_mf4_BASE	0x4_8000
mgmt_ftl_mf5_BASE	0x4_9000
mgmt_ftl_mf6_BASE	0x4_A000

Port	Base Address
mgmt_ftl_mf7_BASE	0x4_B000

5.1.5.1. Function Register 0x8

This register set is used for the Function disable for Functions[3:1]. Function[0] may not be disabled.

Table 5.188. Function Register 0x8

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	disable	RW	1	0x0	Function disable for Functions[7:1]. Function[0] may not be disabled. 0 – Enable 1 – Disable

5.1.5.2. us_port Register 0x38

This register set is used for the Upstream Port configuration.

Table 5.189. us_port Register 0x38

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	adv_target_link_speed	RW	1	0x0	For an upstream port, advertise the link speeds specified by the target_link_speed field rather than the maximum supported speed.

5.1.5.3. id1 Register 0x40

This register set is used for the ID1 configuration.

Table 5.190. id1 Register 0x40

Field	Name	Access	Width	Reset	Description
[31:16]	device_id	RW	16	0xe004	Value returned when the Device ID Configuration Register is read
[15:0]	vendor_id	RW	16	0x19aa	Value returned when the Vendor ID Configuration Register is read

5.1.5.4. id2 Register 0x44

This register set is used for the ID2 configuration.

Table 5.191. id2 Register 0x44

Field	Name	Access	Width	Reset	Description
[31:16]	subsystem_id	RW	16	0xe004	Value returned when the Subsystem ID Configuration Register is read.
[15:0]	subsystem_vendor_id	RW	16	0x19aa	Value returned when the Subsystem Vendor ID Configuration Register is read.

5.1.5.5. id3 Register 0x48

This register set is used for the ID3 configuration.

Table 5.192. id3 Register 0x48

Field	Name	Access	Width	Reset	Description
[31:8]	class_code	RW	24	0x118000	Value returned when the Class Code Configuration Register is read. Must be set to the correct value for the type of device being implemented. Refer to PCI Local Bus Specification Revision 2.3 Appendix D for details on setting Class Code.
[7:0]	revision_id	RW	8	0x4	Value returned when the Revision ID Configuration Register is read.

5.1.5.6. Cardbus Register 0x4c

This register set is used for the Cardbus configuration.

Table 5.193. Cardbus Register 0x4c

Field	Name	Access	Width	Reset	Description
[31:0]	cis_pointer	RW	32	0x0	Value returned when the Cardbus CIS Pointer Configuration Register is read. Set to 0x00000000 unless a Cardbus CIS structure is implemented in memory (which is rare), in which case set to the address of the CIS Structure.

5.1.5.7. Interrupt Register 0x50

This register set is used for the Interrupt configuration.

Table 5.194. Interrupt Register 0x50

Field	Name	Access	Width	Reset	Description
[31:10]	reserved	RO	22	0x0	—
[9:8]	pin	RW	2	0x0	Selects which legacy interrupt is used. 0 – INTA 1 – INTB 2 – INTC 3 – INTD
[7:1]	reserved	RO	7	0x0	—
[0]	disable	RW	1	0x0	Disable support for interrupts. 0 – Enable 1 – Disable

5.1.5.8. bar0 Register 0x60

This register set is used for the BAR0 configuration.

Table 5.195. bar0 Register 0x60

Field	Name	Access	Width	Reset	Description
[31:0]	cfg	RW	32	0xffff000c	Configuration of BAR0 (Cfg address 0x10). Use to define a 32-bit Memory or I/O region or combine with an adjacent BAR to define a 64-bit Memory region.

5.1.5.9. bar1 Register 0x64

This register set is used for the BAR1 configuration.

Table 5.196. bar1 Register 0x64

Field	Name	Access	Width	Reset	Description
[31:0]	cfg	RW	32	0xffffffff	Configuration of bar1 (Cfg address 0x14). Use to define a 32-bit Memory or I/O region or combine with an adjacent BAR to define a 64-bit Memory region.

5.1.5.10. bar2 Register 0x68

This register set is used for the BAR2 configuration.

Table 5.197. bar2 Register 0x68

Field	Name	Access	Width	Reset	Description
[31:0]	cfg	RW	32	0xffffe00c	Configuration of bar2 (Cfg address 0x18). Use to define a 32-bit Memory or I/O region or combine with an adjacent BAR to define a 64-bit Memory region.

5.1.5.11. bar3 Register 0x6c

This register set is used for the BAR3 configuration.

Table 5.198. bar3 Register 0x6c

Field	Name	Access	Width	Reset	Description
[31:0]	cfg	RW	32	0xffffffff	Configuration of bar3 (Cfg address 0x1C). Use to define a 32-bit Memory or I/O region or combine with an adjacent BAR to define a 64-bit Memory region.

5.1.5.12. bar4 Register 0x70

This register set is used for the BAR4 configuration.

Table 5.199. bar4 Register 0x70

Field	Name	Access	Width	Reset	Description
[31:0]	cfg	RW	32	0xffffe00c	Configuration of bar4 (Cfg address 0x20). Use to define a 32-bit Memory or I/O region or combine with an adjacent BAR to define a 64-bit Memory region.

5.1.5.13. bar5 Register 0x74

This register set is used for the BAR5 configuration.

Table 5.200. bar5 Register 0x74

Field	Name	Access	Width	Reset	Description
[31:0]	cfg	RW	32	0xffffffff	Configuration of bar5 (Cfg address 0x24). Use to define a 32-bit Memory or I/O region or combine with an adjacent BAR to define a 64-bit Memory region.

5.1.5.14. exp_rom Register 0x78

This register set is used for the Expansion ROM configuration.

Table 5.201. exp_rom Register 0x78

Field	Name	Access	Width	Reset	Description
[31:0]	cfg	RW	32	0x0	Configuration of exp_rom. Use to define a 32-bit Memory Expansion ROM region. If an Expansion ROM region is defined, the region must map to PCIe-compliant Expansion ROM code or the device may fail to boot.

5.1.5.15. msi_cap Register 0xe8

This register set is used for the MSI Capability configuration.

Table 5.202. msi_cap Register 0xe8

Field	Name	Access	Width	Reset	Description
[31:8]	reserved	RO	24	0x0	—
[7]	reserved	RO	1	0x0	—
[6:4]	mult_message_capable	RW	3	0x5	Number of requested MSI vectors. 0 – 1 1 – 2 2 – 4 3 – 8 4 – 16 5 – 32 6 – Reserved 7 – Reserved
[3:2]	reserved	RO	2	0x0	—
[1]	vec_mask_capable	RW	1	0x1	MSI Capability Per Vector Mask Capable. 0 – Disable 1 – Enable
[0]	disable	RW	1	0x0	MSI Capability Disable. When disabled, the MSI Capability does not appear in PCIe Configuration Space. 0 – Enable 1 – Disable

5.1.5.16. msix_cap Register 0xf0

This register set is used for the MSI-X Capability configuration.

Table 5.203. msix_cap Register 0xf0

Field	Name	Access	Width	Reset	Description
[31:27]	reserved	RO	5	0x0	—
[26:16]	table_size	RW	11	0x1f	Number of requested MSI-X vectors == (table_size+1)
[15:1]	reserved	RO	15	0x0	—
[0]	disable	RW	1	0x0	MSI-X Capability Disable. When disabled, the MSI-X Capability does not appear in PCIe Configuration Space. 0 – Enable 1 – Disable

5.1.5.17. msix_table Register 0xf4

This register set is used for the MSI-X Capability – MSI-X Table configuration.

Table 5.204. msix_table Register 0xf4

Field	Name	Access	Width	Reset	Description
[31:3]	offset	RW	29	0xc00	{offset, 3'b000} == byte address offset, within the BAR selected by bir, at which the MSI-X Table begins
[2:0]	bir	RW	3	0x0	Indicates which Base Address register, located beginning at 10h in Configuration Space, is used to map the MSI-X Table into Memory Space. 0 – 0x10 (BAR0) 1 – 0x14 (BAR1) 2 – 0x18 (BAR2) 3 – 0x1C (BAR3) 4 – 0x20 (BAR4) 5 – 0x24 (BAR5) 6 – Reserved 7 – Reserved

5.1.5.18. msix_pba Register 0xf8

This register set is used for the MSI-X Capability – MSI-X PBA configuration.

Table 5.205. msix_pba Register 0xf8

Field	Name	Access	Width	Reset	Description
[31:3]	offset	RW	29	0xe00	{offset, 3'b000} == byte address offset, within the BAR selected by bir, at which the MSI-X PBA begins.
[2:0]	bir	RW	3	0x0	Indicates which Base Address register, located beginning at 10h in Configuration Space, is used to map the MSI-X PBA into Memory Space. 0 – 0x10 (BAR0) 1 – 0x14 (BAR1) 2 – 0x18 (BAR2) 3 – 0x1C (BAR3) 4 – 0x20 (BAR4) 5 – 0x24 (BAR5) 6 – Reserved 7 – Reserved

5.1.5.19. dsn_cap Register 0x130

This register set is used for the DSN capable cores only such as Device Serial Number Capability configuration.

Table 5.206. dsn_cap Register 0x130

Field	Name	Access	Width	Reset	Description
[31:1]	reserved	RO	31	0x0	—
[0]	enable	RW	1	0x0	Device Serial Number Capability Enable. When disabled, the Device Serial Number Capability does not appear in PCIe Configuration Space. 0 – Disable 1 – Enable

5.1.5.20. dsn_serial Register 0x134

This register set is used for the Device Serial Number Capability – Serial Number.

Table 5.207. dsn_serial Register 0x134

Field	Name	Access	Width	Reset	Description
[63:0]	number	RW	64	0x0	Device Serial Number

5.2. PCI Express Configuration Space Registers

The Lattice PCIe x8 IP Core implements Header Type 00 and Header Type 01 Configuration Registers, including Capability and Extended Capability Items, as detailed in the PCI Express Base Specification, Rev 3.0, PCI Local Interface Specification Revision 3.0, and PCI Bus Power Management Interface Specification Revision 1.2.

Type 00 and Type 01 Configuration Registers implement the first 64 bytes of Configuration Space differently:

- Type 00 – Implemented by Endpoints; refer to Table 5.208.
- Type 01 – Implemented by Root Ports; refer to Table 5.209.

Capability and Extended Capability Items are located at the same addresses regardless of which the header type is implemented, see Table 5.210 for details.

Table 5.208, Table 5.209, and Table 5.210 illustrate the core's PCIe Configuration Register map.

The core's Configuration Registers are highly configurable. In dual-mode (Root-Port/Endpoint) applications, the registers configure themselves according to the mode of operation, changing between Type 00 and Type 01 for instance when changing between an Endpoint and Root Port design.

The Configuration Registers provide the ability for standard PCI/PCIe BIOS/OS software to discover the device, determine its capabilities, and configure the core's features. Since there are a tremendous variety of applications, the core's Configuration Registers are highly configurable.

5.2.1. Type 00 Configuration Header

Table 5.208. Type 00 Configuration Header

Addr	Byte3	Byte2	Byte1	Byte0
00	Device ID		Vendor ID	
04	Status		Command	
08	Class Code			Revision ID
0C	BIST	Header Type	Latency Timer	Cache Line Size
10	Base Address Register 0			
14	Base Address Register 1			
18	Base Address Register 2			
1C	Base Address Register 3			
20	Base Address Register 4			
24	Base Address Register 5			
28	Cardbus CIS Pointer			
2C	Subsystem ID		Subsystem Vendor ID	
30	Expansion ROM Base Address			
34	Reserved			Capabilities Pointer
38	Reserved			
3C	Max Latency	Min Grant	Interrupt Pin	Interrupt Line

5.2.2. Type 01 Configuration Header

Table 5.209. Type 01 Configuration Header

Addr	Byte3	Byte2	Byte1	Byte0
00	Device ID		Vendor ID	
04	Status		Command	
08	Class Code			Revision ID
0C	BIST	Header Type	Primary Latency Timer	Cache Line Size
10	Base Address Register 0			
14	Base Address Register 1			
18	Secondary Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number
1C	Secondary Status		I/O Limit	I/O Base
20	Memory Limit		Memory Base	
24	Prefetchable Memory Limit		Prefetchable Memory Base	
28	Prefetchable Base Upper 32 Bits			
2C	Prefetchable Limit Upper 32 Bits			
30	I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits	
34	Reserved			Capability Pointer
38	Expansion ROM Base Address			
3C	Bridge Control		Interrupt Pin	Interrupt Line

5.2.3. Capability and Extended Capability Address Locations

Table 5.210. Capability and Extended Capability Items

Addr	Byte3	Byte2	Byte1	Byte0
7B-40	PCI Express Capability			
7F-7C	Reserved			
87-80	Power Management Capability			
8F-88	Reserved			
9B-90	MSI-X Capability			
9F-9C	Reserved			
B7-A0	MSI Capability			
FF-B8	Reserved			
147-100	Advanced Error Reporting Capability			
14F-148	ARI Capability			
17F-150	Vendor-Specific Extended Capability			
1AB-180	Secondary PCI Express Extended Capability			
1FF-1AC	Reserved			
207-200	Reserved			
20F-208	Reserved			
21B-210	DSN Capability			
26B-240	Reserved			
2BF-280	Reserved			
38F-2C0	Reserved			
39F-390	Power Budgeting Capability			
3CF-3A0	Dynamic Power Allocation (DPA) Capability			
3DF-3D0	L1 PM Substates Extended Capability			
3E7-3E0	Latency Tolerance Reporting (LTR) Capability			
FFF-3E8	Reserved			

5.2.4. Type 00 Configuration Registers

Table 5.211. Type 00 Configuration Registers

Addr	Config Register	Register Description
01–00	Vendor ID	Read Only: This field identifies the manufacturer of the device.
03–02	Device ID	Read Only: This field identifies the device.
05–04	Command Register	Command Register Bits: Bits 10, 8, 6, and 2..0 are Read/Write. Bits[15:11] = 00000. Not implemented. Bit[10] – Interrupt Disable – If set, interrupts are disabled and cannot be generated; if clear interrupts are enabled Bit[9] = 0. Not implemented. Bit[8] – SERR Enable – When set enables the reporting of fatal and non-fatal errors detected by the device to the root complex (not supported). Bit[7] = 0. Not implemented. Bit[6] – Parity Error Enable – Affects the mapping of PCI Express errors to legacy PCI errors. See *PCI Express Base Specification Rev1.1*, Section 6.2 for details. Bit[5] = 0. Not implemented. Bit[4] = 0. Not implemented. Bit[3] = 0. Not implemented. Bit[2] – Bus Leader Enable – Memory and I/O Requests can only be generated on the Transaction Layer Interface if this bit is set. Bit[1] – Memory Space Enable – If set, the core decodes the packets to determine memory BAR hits; if clear, memory BARs are disabled. Bit[0] – I/O Space Enable – If set, the core decodes the packets to determine I/O BAR hits; if clear, I/O BARs are disabled.
07–06	Status Register	Status Register Bits: Bits 15...11 and 8 are Read/Write. Writing a 1 to a bit location clears that bit. Writing a 0 to a bit location has no affect. Bit[15] – Set by a device whenever it receives a Poisoned TLP. Bit[14] – Set when a device sends an ERR_FATAL or ERR_NONFATAL Message and the SERR Enable bit in the Command Register is set. Bit[13] – Set when a requestor receives a completion with Unrecognized Request Completion Status Bit[12] – Set when a requestor receives a completion with Completer Abort Completion Status Bit[11] – Set when a device completes a request using Completer Abort Completion Status Bits[10:9] = 00. Not implemented. Bit[8] – Leader Data Parity Error – This bit is set by a Requestor if its Parity Error Enable bit is set and either a Completion is received that is marked poisoned or the requestor poisons a write request. Bits[7:5] = 000. Not implemented. Bit[4] = 1 to indicate the presence of a Capabilities List. Bit[3] – Interrupt Status – Reflects the value of mgmt_interrupt. Bits[2:0] = 000. Reserved.
08	Revision ID	Read Only: This register specifies the device specific revision identifier.
0B–09	Class Code	Read Only: The Class Code identifies the generic function of the device.
0C	0x0C: Cache Line Size	Read/Write: Cache Line Size is not used with PCI Express but is still implemented as read/write register for legacy compatibility purposes.
0D	0x0D: Latency Timer	Read Only returning 0x00.
0E	0x0E: Header Type	Read Only: This register reads 0x00 to indicate that the core complies to the standard PCI configuration register mapping and that it is a single function device.
0F	0x0F: BIST	Not implemented. Reads return 0x00.
13–10	Base Address Register 0	Read/Write: Base Address Register0, Base Address Register1, Base Address Register2, Base Address Register3, Base Address Register4, and Base Address Register5 inform system software of the device's resource requirements and are subsequently programmed to allocate memory and I/O resources to the device.

Addr	Config Register	Register Description
17–14	Base Address Register 1	See Base Address Register 0 description.
1B–18	Base Address Register 2	See Base Address Register 0 description.
1F–1C	Base Address Register 3	See Base Address Register 0 description.
23–20	Base Address Register 4	See Base Address Register 0 description.
27–24	Base Address Register 5	See Base Address Register 0 description.
2B–28	Card Bus CIS Pointer	Read Only: Reads return the value of the Cardbus CIS Pointer.
2D–2C	Subsystem Vendor ID	Read Only: Additional vendor information. Reads return the value of the Subsystem Vendor ID.
2F–2E	Subsystem ID	Read Only: Additional device information. Reads return the value of the Subsystem ID.
33–30	Expansion ROM Base Addr. Reg.	Informs system software of the device's Expansion ROM resource requirements and is subsequently programmed to allocate memory resources to the device. Read/Write: Expansion ROM Base Address Register Bits[31:11] – Written to specify where to locate this region in memory space Bits[10:1] = 0..0 Reserved Bit[0] = Set by S/W to enable decoding the Expansion ROM and clear to disable
34	Capabilities Pointer	Read Only: Reads return 0x40 which is the beginning address of the PCI Express Capabilities Item.
37–35	Reserved	Not implemented. Reads return 0x000000.
3B–38	Reserved	Not implemented. Reads return 0x00000000.
3C	Interrupt Line	Legacy interrupt is always ENABLED.
3D	Interrupt Pin	Interrupt support is enabled/disabled by CSR register. When interrupts are enabled, Interrupt Pin returns 0x01 indicating the core implements INTA# and when interrupts are disabled, Interrupt Pin returns 0x00 indicating no interrupts are used.
3E	Minimum Grant	Read Only: Returns 0x00.
3F	Maximum Latency	Read Only: Returns 0x00.

5.2.5. PCI Express Capability

Table 5.212. PCI Express Capability

Addr	Config Register	Register Description
40	PCI Express Capability ID	Read Only = 0x10 (Beginning of PCI Express Capability Item)
41	Next Capability Pointer	Read Only = 0x80 (Pointer to beginning of Power Management Capability)

43-42	PCI Express Capabilities	Read Only
		• Bits[15:14] – Reserved = 00
		• Bits[13:9] – Interrupt Message Number[4:0]; MSI/MSI-X interrupt vector associated with interrupts generated by Configuration Register events (change in link bandwidth and root port error)
		• Bit[8] – Slot Implemented; Downstream Switch/Root Port only
		• Bits[7:4] – Device/Port Type – Must match the core application since the value programmed enables/hides Configuration Registers and functionality that is only applicable to some Device/Port types:
		  • 0000 – PCI Express Endpoint
		  • Required for Endpoint applications

Addr	Config Register	Register Description
		• 0001 – Legacy PCI Express Endpoint • 0100 – Root Port of PCI Express Root Complex (future release) • Required for Root Port applications • 0101 – Upstream Port of PCI Express Switch • Required for Upstream Switch Ports • 0110 – Downstream Port of PCI Express Switch • Required for Downstream Switch Ports • 0111 – PCI Express to PCI/PCI-X Bridge • 1000 – PCI/PCI-X to PCI Express Bridge • 1001 – Root Complex Integrated Endpoint (future release) • 1010 – Root Complex Event Collector (future release) • Bits[3:0] – Capability Version – Must be 0x2 for PCIe 3.0
47-44	Device Capabilities Register	Read Only • Bits[31:29] – Reserved. • Bit[28] – Function Level Reset Capability • 1 – Capability Present • 0 – Capability Not Present • Bits[27:26] – Captured Slot Power Limit Scale • Bits[25:18] – Captured Slot Power Limit Value • Bits[17:16] = 00. Reserved. • Bit[15] = 1. Role-based Error Reporting • Bit[14] = 0 – Reserved • Bit[13] = 0 – Reserved • Bit[12] = 0 – Reserved • Bits[11:9] – Endpoint L1 Acceptable Latency • Bit[8:6] – Endpoint L0s Acceptable Latency • Bit[5] – Extended Tag Field Supported • Bits[4:3] – Phantom Functions Supported • Bits[2:0] – Max Payload Size Supported • 000 – 128 bytes max payload size • 001 – 256 bytes max payload size • 010 – 512 bytes max payload size • 011 – 1024 bytes max payload size • 100 – 2048 bytes max payload size • 101 – 4096 bytes max payload size • 110 – Reserved • 111 – Reserved
49-48	Device Control Register	Read/Write • Bit[15] – Bridge Configuration Retry Enable/Initiate Function Level Reset • Bits[14:12] – Max Read Request Size; the Transmit Interface may not transmit a read request TLP with a length larger than the size indicated by Max Read Request Size: • 000 == 128 bytes • 001 == 256 bytes • 010 == 512 bytes • 011 == 1024 bytes • 100 == 2048 bytes • 101 == 4096 bytes • 110 == Reserved • 111 == Reserved • Bit[11] – Enable No Snoop

Addr	Config Register	Register Description

		• Bit[10] – Aux Power PM Enable
		• Bit[9] – Phantom Functions Enable
		• Bit[8] – Extended Tag Field Enable
		• Bits[7:5] – Max Payload Size; the Transmit Interface may not transmit a TLP with a payload larger than the size indicated by Max Payload Size:
		  • 000 == 128 bytes
		  • 001 == 256 bytes
		  • 010 == 512 bytes
		  • 011 == Reserved
		  • 100 == Reserved
		  • 101 ==Reserved
		  • 110 == Reserved
		  • 111 == Reserved
		• Bit[4] – Enable Relaxed Ordering
		• Bit[3] – Unsupported Request Reporting Enable
		• Bit[2] – Fatal Error Reporting Enable
		• Bit[1] – Non-Fatal Error Reporting Enable
		• Bit[0] – Correctable Error Reporting Enable

4B-4A	Device Status Register	Bits[15:4] are Read Only. Bits[3:0] are cleared by writing a 1 to the corresponding bit location. • Bits[15:6] = 0000000000. Reserved • Bit[5] – Transactions Pending • Bit[4] – AUX Power Detected • Bit[3] – Unsupported Request Detected • Bit[2] – Fatal Error Detected • Bit[1] – Non-Fatal Error Detected • Bit[0] – Correctable Error Detected

4F-4C	Link Capabilities Register	Read Only.
		• Bits[31:24] – Port Number
		• Bits[22] = 1. ASPM Optional Compliance
		• Bit[21] – Link Bandwidth Notification Capability
		  • == 1 when operating as a Downstream Port; else 0
		• Bit[20] – Data Link Layer Active Reporting Capable
		  • == 1 when operating as a Downstream Port; else 0
		• Bit[19] – Surprise Down Error Reporting Capable
		  • == 1 when operating as a Downstream Port; else 0
		• Bit[18] = 0. Clock Power Management
		• Bits[17:15] – L1 Exit Latency
		• Bits[14:12] – L0s Exit Latency
		• Bits[11:10] – Active State Power Management (ASPM) Support
		  • 00 – No ASMP Support
		  • 01 – L0s Supported
		  • 10 – L1 Supported
		  • 11 – L0s and L1 Supported
		• Bits[9:4] – Maximum Link Width
		  • 000001 – x1
		  • 000010 – x2
		  • 000100 – x4
		  • 001000 – x8
		  • 010000 – x16
		• Bits[3:0] – Maximum Link Speed

Addr	Config Register	Register Description
		• 0001 (2.5GT/s) • 0010 (5GT/s) • 0011 (8GT/s) (2024.1) • 0100 (16GT/s) (2024.1)

51-50	Link Control Register	Read/Write
		• Bits[15:12] = 0000. Reserved.
		• Bit[11] – Link Autonomous Bandwidth Interrupt Enable
		• Bit[10] – Link Bandwidth Management Interrupt Enable
		• Bit[9] – Hardware Autonomous Width Disable
		• Bit[8] = 0. Enable Clock Power Management
		• Bit[7] – Extended Sync
		• Bit[6] – Common Clock Configuration
		• Bit[5] – Retrain Link
		• Bit[4] – Link Disable
		• Bit[3] – Read Completion Boundary (RCB)
		  • 0 == 64 bytes
		  • 1 == 128 bytes
		• Bit[2] = 0. Reserved.
		• Bits[1:0] – Active State Power Management (ASPM) Control
		  • 00 – Disabled
		  • 01 – L0s Enabled
		  • 10 – L1 Enabled
		  • 11 – L0s and L1 Enabled

53-52	Link Status Register	Read Only
		• Bit[15] – Link Autonomous Bandwidth Status
		• Bit[14] – Link Bandwidth Management Status
		• Bit[13] – Data Link Layer Active
		• Bit[12] – Slot Clock Configuration
		• Bit[11] – Link Training
		• Bit[10] = 0. Reserved.
		• Bits[9:4] Negotiated Link Width – indicates the number of lanes currently in use
		  • 010000 = x16
		  • 001000 = x8
		  • 000100 = x4
		  • 000010 = x2
		  • 000001 = x1
		• Bits[3:0] Link Speed
		  • 0001 (2.5 GT/s)
		  • 0010 (5.0 GT/s)
		  • 0011 (8.0 GT/s)

57-54	Slot Capabilities Root Port/Switch Only	Normally Read Only; Writable when HW.Init Write Enable == 1 (see Table 5.218). • Bits[31:19] – Physical Slot Number • Bit[18] – No Command Completed Support • Bit[17] – Electromechanical Interlock Present • Bits[16:15] – Slot Power Limit Scale[1:0] • Bits[14:7] – Slot Power Limit Value[7:0] • Bit[6] – Hot-Plug Capable • Bit[5] – Hot-Plug Surprise • Bit[4] – Power Indicator Present • Bit[3] – Attention Indicator Present

Addr	Config Register	Register Description
		• Bit[2] – MRL Sensor Present • Bit[1] – Power Controller Present • Bit[0] – Attention Button Present
59-58	Slot Control Root Port/Switch Only	Read Only • Bits[15:13] = 0. Reserved. • Bit[12] – Data Link Layer State Changed Enable • Bit[11] = 0. Electromechanical Interlock Control • Bit[10] – Power Controller Control • Bit[9:8] – Power Indicator Control • Bit[7:6] – Attention Indicator Control • Bit[5] – Hot-Plug Interrupt Enable • Bit[4] – Command Completed Interrupt Enable • Bit[3] – Presence Detect Changed Enable • Bit[2] – MRL Sensor Changed Enable • Bit[1] – Power Fault Detected Enable • Bit[0] – Attention Button Pressed Enable
5b-5a	Slot Status Root Port/Switch Only	Read Only • Bits[15:9] = 0. Reserved. • Bit[8] – Data Link Layer State Changed • Bit[7] – Electromechanical Interlock Status • Bit[6] – Presence Detect State • Bit[5] – MRL Sensor State • Bit[4] – Command Completed • Bit[3] – Presence Detect Changed • Bit[2] – MRL Sensor Changed • Bit[1] – Power Fault Detected • Bit[0] – Attention Button Pressed
5d-5c	Root Control Root Port Only	Read/Write • Bits[15:5] = 0. Reserved. • Bit[4] = CRS Software Visibility Enable • Bit[3] – PME Interrupt Enable • Bit[2] – System Error on Fatal Error Enable • Bit[1] – System Error on Non-Fatal Error Enable • Bit[0] – System Error on Correctable Error Enable
5f-5e	Root Capabilities Root Port Only	Read Only; Bit[16] – Write 1 to clear. • Bits[15:1] = 0. Reserved • Bit[0] = 1. CRS Software Visibility supported.
63-60	Root Status Root Port Only	Read Only; Bit[16] – Write 1 to clear. • Bits[31:18] = 0. Reserved • Bit[17] – PME Pending • Bit[16] – PME Status • Bits[15:0] – PME Requester ID
67-64	Device Capabilities 2	Read Only • Bits[31:24] = 0. Reserved • Bits[23:22] = 00. Max End-End TLP Prefixes • Bit[21] = 0. End-End TLP Prefix Supported • Bit[20] = 0. Extended Fmt Field Supported • Bit[19:18] = 00. OBFF Supported • Bits[17:14] = 0000. Reserved • Bits[13:12] = 00. TPH Completer Supported • Bit[11] = LTR Mechanism Supported

Addr	Config Register	Register Description
		• Bit[10] = 0. No RO-enabled PR-PR Passing • Bit[9] = 0. 128-bit CAS Completer Supported • Bit[8] = 0. 64-bit AtomicOp Completer Supported • Bit[7] = 0. 32-bit AtomicOp Completer Supported • Bit[6] = 0. AtomicOp Routing Supported • Bit[5] = 0. ARI Forwarding Supported • Bit[4] – Completion Timeout Disable Supported • Bits[3:0] – Completion Timeout Ranges Supported
69-68	Device Control 2	Read/Write • Bit[15] – End-End TLP Prefix Blocking • Bits[14:13] – OBFF Enable; not supported • Bits[12:11] = 00. Reserved. • Bit[10] – LTR Mechanism Enable • Bit[9] – IDO Completion Enable • Bit[8] – IDO Request Enable • Bit[7] – AtomicOp Egress Blocking • Bit[6] – AtomicOp Request Enable • Bit[5] – ARI Forwarding Enable • Bit[4] – Completion Timeout Disable – Set by system software to disable this device from generating completion timeouts. You must disable completion timeout error generation when this bit is set. • Bits[3:0] – Completion Timeout Value – Set by system software to select the completion timeout range which must be used by users which are implementing completion timeouts. See PCI Express Specification Table 7.24 for details.
6B-6A	Device Status 2	Reserved by PCI SIG for future use. Reads return 0x00000000.
6F-6C	Link Capabilities 2	Read Only • Bits[31:23] – Reserved • Bits[22:16] – Lower SKP OS Reception Supported Speeds Vector • Bits[15:9] – Lower SKP OS Generation Supported Speeds Vector • Bit[8] – Crosslink Supported • Bits[7:1] – Supported Link Speeds Vector • Bit[0] = 0. Reserved

71-70	LinkControl 2	Read/Write
		• Bit[15:12] – Compliance Preset/De-emphasis[3:0]
		• Bit[11] – Compliance SOS
		• Bit[10] – Enter Modified Compliance
		• Bits[9:7] – Transmit Margin
		• Bit[6] – Selectable De-emphasis
		• Bit[5] – Hardware Autonomous Speed Disable
		• Bit[4] – Enter Compliance
		• Bits[3:0] – Target Link Speed[3:0]
		  • 0001 (2.5 GT/s)
		  • 0010 (5.0 GT/s)
		  • 0011 (8.0 GT/s)

Addr	Config Register	Register Description

73-72	Link Status 2	Read Only; Bit[5] – write 1 to clear:
		• Bits[15:6] = 0000000000. Reserved.
		• Bit[5] – Link Equalization Reset
		• Bit[4] – Equalization Phase 3 Successful
		• Bit[3] – Equalization Phase 2 Successful
		• Bit[2] – Equalization Phase 1 Successful
		• Bit[1] – Equalization Complete
		• Bit[0] – Current De-emphasis Level
		  • 1==-3.5 dB
		  • 0==-6 dB

77-74	Slot Capabilities 2 Root Port/Switch Only	Reserved by PCI SIG for future use. Reads return 0x00000000.
79-78	Slot Control 2 Root Port/Switch Only	Reserved by PCI SIG for future use. Reads return 0x00000000.
7b-7a	Slot Status 2 Root Port/Switch Only	Reserved by PCI SIG for future use. Reads return 0x00000000.
7F-7C	Reserved	Reads return 0x00000000.

5.2.6. Power Management Capability

Table 5.213. Power Management Capability

Addr	Config Register	Register Description
80	Power Management Capability ID	Read Only = 0x01 (Beginning of Power Management Capability Item)
81	Next Capability Pointer	Read Only. Pointer to next Capability Item in the list.
83-82	Power Management Capabilities	Read Only. • Bits[15:11] – PME Support; recommended default == 0. • Bits[10] – D2 Support (1) Yes (0) No; this bit must be set for the core to allow Power State to be written to D2; recommended default == 0. • Bit[9] – D1 Support (1) Yes (0) No; this bit must be set for the core to allow Power State to be written to D1; recommended default == 0. • Bit[8:6] – Aux Current; recommended default = 0. • Bit[5] – Device Specific Initialization(DSI); recommended default = 0. • Bit[4] – Reserved; set to 0. • Bit[3] – PME Clock; recommended default = 0. • Bits[2:0] – Version; set to 011 (complies with revision 1.2 of the PCI Power Management Interface Specification). Refer Error Handling for additional detail.

Addr	Config Register	Register Description

85-84	Power Management Control/Status	Read/Write.
		• Bit[15] – PME Status; if Power Management Capabilities[15] == 1 indicating that PME is generated from D3cold, then PME_Status is implemented by the core; otherwise PME_Status == 0.
		• Bits[14:13] – Data Scale; recommend == 0 (Data not implemented)
		• Bits[12:9] – Data Select; recommend == 0 (Data not implemented)
		• Bit[8] – PME En – ; if Power Management Capabilities[15:11] == 0 indicating that PME is not generated from any power state then PME_En == 0; is implemented by the core and written by system software to enable PME generation from D3cold; otherwise PME_En == 0.
		• Bits[7:4] – Reserved – set to 0
		• Bit[3] – No Soft Reset – Core sets to 1 since the core is not reset when transitioning from D3hot to D0 purely due to power state changes. This bit is used by system software to know whether the device needs to be reinitialized when transitioning between D3hot and D0.
		• Bit[2] – Reserved; set to 0
		• Bits[1:0] – Power State; software writes this field to transition a device into a different power state; increasing Dx numbers represent increasingly lower power states.
		  • 00 – D0; normal operation
		  • 01 – D1; not allowed to be written unless D1 Support == 1
		  • 10 – D2; not allowed to be written unless D2 Support == 1
		  • 11 – D3hot; “off”
		Refer Error Handling for additional detail.

86	PMCSR PCI to PCI Bridge Support	Read Only. • Bit[7] – Bus Power/Clock Control Enable; set to 0 • Bit[6] – B2/B3 Support for D3bat; set to 0 • Bits[5:0] – Reserved; set to 0
87	Data	Read Only. • Bits[7:0] – Data; recommended default = 0; not implemented
8F-88	Reserved	Reads return 0x00000000.

5.2.7. MSI-X Capability

Table 5.214. MSI-X Capability

Addr	Config Register	Register Description
90	MSI-X Capability ID	Read Only = 0x11 (Beginning of MSI-X Capability Item) MSI-X Support may be enabled/disabled through the CSR registers. If present, its capability is defined as follows otherwise all the following registers reads 0x0.
91	Next Capability Pointer	Read Only. Pointer to next Capability Item in the list.
93-92	Message Control	Only bits[15:14] are Read/Write. • Bit[15] – MSI-X Enable (Read/Write) • Bit[14] – Function Mask (Read/Write) • Bits[13:11] – Reserved – 000 (Read Only) • Bit[10:0] – Table Size[10:0] (Read Only) • The number of MSI-X vectors requested/supported by the user’s design is Table Size + 1.
97-94	Table_Offset, Table_BIR	Read Only. Bits[31:3] – Table_Offset[31:3] • {Table_Offset[31:3], 000} is the offset into the BAR indicated by Table_BIR where the MSI-X Table begins. • Bits[2:0] – Table BIR[2:0]

Addr	Config Register	Register Description

		• Indicates which BAR location contains the MSI-X Table. In the case of a 64-bit BAR Table BIR indicates the BAR that contains the lower 32-bit address:
		  • 000 – BAR0
		  • 001 – BAR1
		  • 010 – BAR2
		  • 011 – BAR3
		  • 100 – BAR4
		  • 101 – BAR5
		  • 110, 111 – Reserved

9B-98	PBA_Offset, PBA_BIR	Read Only. Bits[31:3] – PBA_Offset[31:3] • Same as Table Offset above but indicates the location of the PBA (Pending Bit Array). • Bits[2:0] – PBA BIR[2:0] • Same as Table BIR above, but indicates the location of the PBA.

5.2.8. MSI Capability

Table 5.215. MSI Capability

Addr	Config Register	Register Description
9F-9C	Reserved	Reads return 0x00000000.
A0	Message Capability ID	Read Only = 0x05 (Beginning of Message Capability Item); MSI Support is enabled/disabled by CSR registers. If present, its capability is defined as follows otherwise all the following registers read 0x0.
A1	Next Capability Pointer	Read Only. Pointer to next Capability Item in the list.
A3-A2	Message Control	• Bits[6:4] and Bit[0] are Read/Write; remainder are Read Only. • Bits[15:9] = 0x00. Reserved. • Bit[8] = 0. Note per vector masking capable. • Bit[7] – 64-bit Address Capable = 1 (Capable of generating 64-bit messages). • Bits[6:4] – Multiple Message Enable – system software writes the number of allocated messages; 000==1, 001==2, 010==4, 011==8, 100==16, 101==32, 110 Reserved, 111 Reserved. • Bits[3:1] – Multiple Message Capable – Number of messages requested by the device == 000 (1 Message). • Bit[0] – MSI Enable – System software sets this bit to enable MSI. When set, the core uses the MSI mechanism instead of the legacy interrupt mechanism to forward user interrupts on mgmt_interrupt to PCI Express.
A7-A4	Message Address	Bits[31:2] are Read/Write; Bits[1:0] are Read Only. • Bits[31:2] Message Address[31:2] • Bits[1:0] – Reserved – Message Address[1:0] is always 00.
AB-A8	Message Upper Address	Read/Write • Bits[31:0] Message Address[63:32]; if Message Address[63:32] == 0, then the core uses only Message Address[31:0] and does 32-bit address MSI writes. If Message Address[63:32] != 0, the core uses Message Address[63:0] and does 64-bit address MSI writes.
AD-AC	Message Data	Read/Write • Bits[15:0] Message Data[15:0] – An MSI Message is sent by writing Message Data to Message Address.

5.2.9. Advanced Error Reporting Extended Capability

Table 5.216. Advanced Error Reporting Extended Capability

Addr	Config Register	Register Description
103-100	Advanced Error Reporting Enhanced Capability Header	Beginning of Advanced Error Reporting (AER) Capability; the AER capability is only present if AER support is enabled in the design, however, AER support is a standard core feature that is present unless AER removal has been specifically requested to be excluded at core deliver time (which is unusual). • Bits[15:0] – Read Only = 0x0001 == AER Capability ID • Bits[19:16] – Read Only = 0x01 == AER Capability Version (PCIe 2.0/1.1) • Bits[31:20] – Read Only. Pointer to next Enhanced/Extended Capability Item in the list.
107-104	Uncorrectable Error Status	• Read/Write: Bit set when corresponding error event occurs, and the error is not masked by the Uncorrectable Error Mask register; clear set bits by writing a 1: • Bits[3:0] – Reserved == 0 • Bit[4] – DataLink_Protocol_Error_Status • Bit[5] – Surprise_Down_Error_Status • Bits[11:6] – Reserved == 0 • Bit[12] – Poisoned_TLP_Status • Bit[13] – Flow_Control_Protocol_Error_Status • Bit[14] – Completion_Timeout_Status • Bit[15] – Completer_Abort_Status • Bit[16] – Unexpected_Completion_Status • Bit[17] – Receiver_Overflow_Status • Bit[18] – Malformed_TLP_Status • Bit[19] – ECRC_Error_Status • Bit[20] – Unsupported_Request_Error_Status • Bit[21] – Reserved = 0 • Bit[22] – Uncorrectable Internal Error Status • Bits[31:23] – Reserved == 0
10B-108	Uncorrectable Error Mask	Read/Write: Set corresponding bit to mask (not report) selected error events; clear to unmask (report): • Bits[3:0] – Reserved == 0 • Bit[4] – DataLink_Protocol_Error_Mask • Bit[5] – Surprise_Down_Error_ Mask • Bits[11:6] – Reserved == 0 • Bit[12] – Poisoned_TLP_ Mask • Bit[13] – Flow_Control_Protocol_Error_ Mask • Bit[14] – Completion_Timeout_ Mask • Bit[15] – Completer_Abort_ Mask • Bit[16] – Unexpected_Completion_ Mask • Bit[17] – Receiver_Overflow_ Mask • Bit[18] – Malformed_TLP_ Mask • Bit[19] – ECRC_Error_ Mask • Bit[20] – Unsupported_Request_Error_ Mask • Bit[21] – Reserved = 0 • Bit[22] – Uncorrectable Internal Error Mask • Bits[31:23] – Reserved == 0
10F-10C	Uncorrectable Error Severity	Read/Write: Set corresponding bit to mark selected error events as FATAL errors; clear to mark selected error events as NON-FATAL errors: • Bits[3:0] – Reserved == 0 • Bit[4] – DataLink_Protocol_Error_Severity • Bit[5] – Surprise_Down_Error_Severity • Bits[11:6] – Reserved == 0

Addr	Config Register	Register Description
		• Bit[12] – Poisoned_TLP_Severity • Bit[13] – Flow_Control_Protocol_Error_Severity • Bit[14] – Completion_Timeout_Severity • Bit[15] – Completer_Abort_Severity • Bit[16] – Unexpected_Completion_Severity • Bit[17] – Receiver_Overflow_Severity • Bit[18] – Malformed_TLP_Severity • Bit[19] – ECRC_Error_Severity • Bit[20] – Unsupported_Request_Error_Severity • Bit[21] – Reserved = 0 • Bit[22] – Uncorrectable Internal Error Severity • Bits[31:23] – Reserved == 0
113-110	Correctable Error Status	Read/Write: Bit set when corresponding error event occurs, and the error is not masked by the Correctable Error Mask register; clear set bits by writing a 1: • Bit[0] – Receiver_Error_Status • Bits[5:1] – Reserved == 0 • Bit[6] – Bad_TLP_Status • Bit[7] – Bad_DLLP_Status • Bit[8] – Replay_Num_Rollover_Status • Bits[11:9] – Reserved == 000 • Bit[12] – Replay_Timer_Timeout_Status • Bit[13] – Advisory_Non_Fatal_Error_Status • Bit[14] – Corrected Internal Error Status • Bit[15] – Header Log Overflow Status • Bits[31:16] – Reserved == 0
117-114	Correctable Error Mask	Read/Write: Set corresponding bit to mask (not report) selected error events; clear to unmask (report): • Bit[0] – Receiver_Error_Mask • Bits[5:1] – Reserved == 0 • Bit[6] – Bad_TLP_Mask • Bit[7] – Bad_DLLP_Mask • Bit[8] – Replay_Num_Rollover_Mask • Bits[11:9] – Reserved == 000 • Bit[12] = Replay_Timer_Timeout_Mask • Bit[13] = Advisory_Non_Fatal_Error_Mask • Bit[14] – Corrected Internal Error Mask • Bit[15] – Header Log Overflow Mask • Bits[31:16] – Reserved == 0

11B-118	Advanced Error Capabilities and Control	Read/Write: Misc Capabilities and Control
		• Bits[4:0] = Read Only – First_Error_Pointer[4:0]
		• Bit[5] = Read Only – ECRC_Generation_Capable
		  • 1==Device can generate ECRC; set if the core includes ECRC generation logic (non-standard core option).
		  • 0 == Device is not capable of generating ECRC.
		• Bit[6] = Read/Write – ECRC_Generation_Enable
		  • Software sets to control whether ECRCs are generated and inserted for TLPs transmitted by the core; if ECRC support is not implemented in the core, this bit is Read Only == 0.
		  • 1== Generate and insert ECRC for TLPs transmitted by the core.
		  • 0 == Do not generate and insert ECRC.

Addr	Config Register	Register Description

		• Bit[7] = Read Only – ECRC_Check_Capable
		  • 1==Device is capable of checking ECRC; set if the core includes ECRC generation logic (non-standard core option).
		  • 0 == Device is not capable of checking ECRC.
		• Bit[8] = Read/Write – ECRC_Check_Enable
		  • Software sets to control whether ECRCs are checked for TLPs received by the core; if ECRC support is not implemented in the core, this bit is Read Only == 0.
		  • 1== Check ECRC for all TLPs with ECRC received by the core.
		  • 0 == Do not check ECRC.
		• Bits[31:9] – Reserved = 0

12B-11C	Header Log	Header[127:0] of the TLP associated with the error. TLP format is in same order as illustrated in PCIe Specification: • 0x11F-11C – {Byte0, Byte1, Byte2, Byte3} • 0x123-120 – {Byte4, Byte5, Byte6, Byte7} • 0x127-124 – {Byte8, Byte9, Byte10, Byte11} • 0x12B-0x128 – {Byte12, Byte13, Byte14, Byte15}
137-12C	Reserved	Only implemented by AER Root Ports. Reads return 0x00000000.
147-138	Reserved	TLP Prefix Log Register

5.2.10. ARI Extended Capability

ARI Is located at offset 0x148 unless AER is not present in which case it is moved to 0x100.

Table 5.217. ARI Extended Capability

Addr	Config Register	Register Description
14B-148 or 103-100	ARI Capability Extended Capability Header	Beginning of ARI Extended Capability – Read Only • Bits[31:20] – Pointer to next Enhanced/Extended Capability Item in the list. • Bits[19:16] = 0x1 == Capability Version • Bits[15:0] = 0x000E == Capability ID
14D-14C or 105-104	ARI Capability Register	Read Only • Bits[15:8] – Next Function Number = 0 (not implemented) • Bits[7:2] – Reserved = 0 • Bit[1] – ACS Function Groups Capability = 0 (not implemented) • Bit[0] – MFVC Function Groups Capability = 0 (not implemented)
14F-14E or 107-106	ARI Control Register	Read Only • Bits[15:7] – Reserved • Bit[6:4] – Function Group = 0 (not implemented) • Bits[3:2] – Reserved = 0 • Bit[1] – ACS Function Groups Enable = 0 (not implemented) • Bit[0] – MFVC Function Groups Enable = 0 (not implemented)

5.2.11. Vendor-Specific Extended Capability

Table 5.218. Vendor-Specific Extended Capability

Addr	Config Register	Register Description
153-150	Vendor-Specifc PCI Express Extended Capability Header	Beginning of Vendor-Specific Extended Capability (VSEC) • Bits[31:20] – Read Only. Pointer to next Enhanced/Extended Capability Item in the list. • Bits[19:16] – Read Only = 0x1 == Capability Version • Bits[15:0] – Read Only = 0x000B == Capability ID
157-154	Vendor-Specific Header	Read Only • Bits[31:20] – VSEC Length = 0x24 (36 bytes) • Bits[19:16] – VSEC Rev = 0x1 • Bits[15:0] – VSEC ID

15B-158	HW.Init	Read/Write
		• Bits[31:1] = 0. Reserved
		• Bit[0] – HW.Init Write Enable – Used to allow software to write some Configuration Registers which are type *HW.Init* when they would otherwise not be writable; default value == 0
		  • 1 – HW.Init Write Enabled – Allow specific HW.Init fields to be written by software; only relevant for Configuration Registers in this document which specifically state they are writable when HW.Init Write Enable == 1 (for example, PCI Express Capability: Slot Capabilities).
		  • 0 – HW.Init Write Disabled

15F-15C	Link Power Down Root Port / Downstream Switch Port Only	Read/Write – Used by system software in a Root Port or Downstream Switch Port application to cause a PME_Turn_Off Message to be transmitted on PCI Express to request that the downstream PCI Express heirarchy prepare for Power Down. • Bits[31:3] = 0. Reserved. • Bit[2] – L2 Request Timeout; indicates when an L2 Request completed due to a timeout; L2 Request Timeout is cleared (0) when L2_Request is written to 1 or when 1 is written to this register; L2 Request Timeout is set (1) when a PME_TO_ACK message is not received in response to a transmitted PME_Turn_Off within the expected 100 ms (100 µs for simulation when mgmt_short_sim == 1) timeout window. • Bit[1] – L2 Request Status; indicates when an L2 Request has completed either due to receiving the expected PME_TO_Ack message response or due to timeout; L2 Request Status is cleared (0) when L2_Request is written to 1 or when 1 is written to this register; L2 Request Status is set (1) when a PME_TO_ACK message is received or a timeout occurs • Bit[0] – L2 Request; write to 1 to cause a PME_Turn_Off Message to be transmitted downstream to the PCI Express hierarchy; after all downstream devices have prepared for power-down the core should receive a PME_TO_Ack message in response indicating the downstream PCIe hierarchy is ready for removal of power; L2 Request stays set until a PME_TO_ACK message is received or a timeout occurs.

Addr	Config Register	Register Description
163-160	Autonomous Recovery, Speed, and Width	Read/Write – Used by system software in an US Port to perform autonomous speed change, width change, or entry to recovery. Bits[31:16] – Lane Width Mask. A 1 indicates that the lane can be used. [16] = Lane 0 .. [31] = Lane 15. Bits[15:12] – Reserved Bits[11:8] – Target Speed. (1=2.5G, 2=5G,3=8G, 4=16G). Bits[7:3] – Reserved Bit[2] – Autonomous Entry to Recovery command. When Set to 1, Bits[1] and [0] must both be set to 0. Setting this bit to 1 causes the Link to immediately transition to recovery. Bit[1] – Autonomous Width Change command. When set to 1, the Link transitions to recovery to perform a link width change, using the Lane Width Mask field. This bit is ignored if HW Autonomous Width Disable has been set in the Link Control register. Bit[0] – Autonomous Speed Change command. When set to 1, the Link transitions to Recovery to perform a speed change, using the Target Speed field. This bit is ignored if HW Autonomous Speed Disable has been set in the Link Control 2 register. Speed and width changes can be signaled together. However, entry to recovery must be signaled independently from speed or width changes.
173-164	Reserved	Reserved

5.2.12. Secondary PCI Express Extended Capability

Table 5.219. Secondary PCI Express Extended Capability

Addr	Config Register	Register Description
183-180	Secondary PCI Express Extended Capability Header	Beginning of Secondary PCI Express Extended Capability; this capability is only present if the PCIe 3.0 support is enabled in the design. If the AER capability is not present, this capability is located at offset 0x100 instead. Bits[31:20] – Read Only. Pointer to next Enhanced/Extended Capability Item in the list. • Bits[19:16] – Read Only = 0x1 == Capability Version • Bits[15:0] – Read Only = 0x0019 == Capability ID
187-184	Link Control 3	Read/Write • Bits[31:16] – Reserved = 0 • Bits[15:9] – Enable Lower SKP OS Generation Vector • Bits[8:2] – Reserved = 0 • Bit[1] – Link Equalization Request Interrupt Enable • Bit[0] – Perform Equalization
18B-188	Lane Error Status	Read Only: Indicates lane-specific error status. • Bits[31:NUM_LANES] – Reserved = 0 • Bit[Lane#] – 1 == Error detected on lane[[Lane#]; 0 == no error
1AB-18C	Lane Equalization Control Register	Read Only: Control and status fields for link equalization; 16-bits per lane starting with Lane[0] with higher lane #s at higher addresses. Per lane format is as follows: • Bit[15] – Reserved = 0 • Bits[14:12] – Upstream Port Receiver Preset Hint • Bits[11:8] –Upstream Port Transmitter Preset • Bit[7] – Reserved = 0 • Bits[6:4] – Downstream Port Receiver Preset Hint • Bits[3:0] –Downstream Port Transmitter Preset

5.2.13. ATS Extended Capability

Table 5.220. ATS Extended Capability

Addr	Config Register	Register Description
203-200	ATS Capability Extended Capability Header	Beginning of ATS Extended Capability – Read Only • Bits[31:20] – Pointer to next Enhanced/Extended Capability Item in the list. • Bits[19:16] = 0x1 == Capability Version • Bits[15:0] = 0x000F == Capability ID
205-204	ATS Capability Register	Read Only • Bits[4:0] – Invalidate Queue Depth • Bit[5]Page Aligned Request • Bits[15:6] – Reserved
207-206	ATS Control Register	Read/Write • Bits[4:0] – Smallest Translation Unit • Bits[14:5] – Reserved • Bit[15] – Enable

5.2.14. DSN Extended Capability

Table 5.221. DSN Extended Capability

Addr	Config Register	Register Description
213-210	DSN Capability Extended Capability Header	Beginning of DSN Extended Capability – Read Only • Bits[31:20] – Pointer to next Enhanced/Extended Capability Item in the list. • Bits[19:16] = 0x1 == Capability Version • Bits[15:0] = 0x0003 == Capability ID
21B-214	DSN Serial NUmber	Read Only • Bits[63:0] – DSN Serial Number

5.2.15. Resizable BAR Capability

Table 5.222. Resizable BAR Capability

Addr	Config Register	Register Description
283-280	Resizable BAR Extended Capability Header	Resizable BAR Capability Header – Read Only • Bits[31:20] – Pointer to next Enhanced/Extended Capability Item in the list. • Bits[19:16] = 0x1 == Capability Version • Bits[15:0] = 0x0015 == Capability ID
287-284	Resizable BAR Capability(0)	Read Only • Bits[31:24] – Reserved • Bits[23:4] – When Bit n is Set, The BAR Indicated by the BAR Index in the Control Register operates with BAR sized to 2^(n+16) Bytes. For example, bit[4] = 2^20 Bytes = 1 MB • Bits[3:0] – Reserved

Addr	Config Register	Register Description

28B-288	Resizable Bar Control Register(0)	• Read Only
		  • Bits[31:13] – Reserved
		• Read/Write
		  • Bits[12:8] – BAR Size. Encoded Value for the Size this BAR should use.
		• Read Only
		  • Bits[7:5] – Number of Resizable BARs. Value must be between 1 and 6. These bits are only valid in the Resizable BAR Control Register (0). In Control Registers (1) or higher, these bits are Reserved.
		  • Bits[2:0] – BAR Index for this BAR:
		  • 0 = BAR located at offset 0x10
		  • 1 = BAR located at offset 0x14
		  • 2 = BAR located at offset 0x18
		  • 3 = BAR located at offset 0x1C
		  • 4 = BAR located at offset 0x20
		  • 5 = BAR located at offset 0x24
		  • Other values reserved. For a 64-bit BAR, this index should point to the lower DWORD.

2BF-28C	Resizable BAR Capability and Control Registers (1..6)	See Resizable BAR Capability (0) See Resizable BAR Control Register(0) The number of Implemented BAR Capability and Control Registers depends on the setting of “Number of Resizable BARs” Control Register (0).

5.2.16. Power Budgeting Capability

Table 5.223. Power Budgeting Capability

Addr	Config Register	Register Description
393-390	Power Budgeting Capability Extended Capability Header	Beginning of Power Budgeting Extended Capability – Read Only • Bits[31:20] – Pointer to next Enhanced/Extended Capability Item in the list. • Bits[19:16] = 0x1 == Capability Version • Bits[15:0] = 0x0004 == Capability ID
394	Data Select Register	Read/Write • Bits[7:0] – Data Select Register
397-395	Reserved	Reserved
39B-398	Data Register	Read Only • Bits[31:21] – Reserved • Bits[20:18] – Power Rail (0:12V,1:3.3V,2:1.5/1.8V,7:Thermal) • Bits[17:15] – Type (0:PME Aux,1:Aux,2:Idle,3:Sustained,7:Max) • Bits[14:13] – PM State (0:D0,1:D1,2:D2,3:D3) • Bits[12:10] – PM Sub State (0:Default,others: Device Specific) • Bits[9:8] – Data Scale (0:1x,1:0.1x,2:0.01x,3:0.001x) • Bits[7:0] – Base Power
39C	Capabilities Register	Read Only • Bits[7:1] – Reserved • Bit[0] – System Allocated – Set to 1 to indicate that the Power Budget Should be System Allocated, and the values from the Data Register should NOT be used for System Power Budgeting. Set to 0 to indicate that the values provided in the Data Register should be used for System Power Budgeting.

5.2.17. Dynamic Power Allocation Capability

Table 5.224. Dynamic Power Allocation (DPA) Capability

Addr	Config Register	Register Description
3A3-3A0	DPA Capability Extended Capability Header	Beginning of DPA Extended Capability – Read Only • Bits[31:20] – Pointer to next Enhanced/Extended Capability Item in the list. • Bits[19:16] = 0x1 == Capability Version • Bits[15:0] = 0x0016 == Capability ID
3A7-3A4	DPA Capability Register	Read Only • Bits[31:24] – Transition Latency Value1 (xlcy1) • Bits[23:16] – Transition Latency Value0 (xlcy0) • Bits[15:14] – Reserved • Bits[13:12] – Power Allocation Scale (PAS) • Bits[11:10] – Reserved • Bits[9:8] – Transition Latency Unit (tlunit) • Bits[7:5] –Reserved • Bits[4:0] – Substate_Max
3AB-3A8	DPA Latency Indicator Register	Read Only • Bits[31:Substate_Max+1] – Reserved • Bits[Substate_Max:0] – Transition Latency Indicator Bits
3AD-3AC	DPA Status Register	Read Only • Bits[15:9] – Reserved Read, Write 1 to Clear • Bits[8] – Substate Control Enabled Read Only • Bits[7:0] – Substate Status
3EF-3AE	DPA Control Register	Read Only • Bits[15:5] – Reserved Read/Write • Bits[4:0] – Substate Control
3CF-3B0	DPA Power Allocation Array	Read Only • Bits[7:0] – Substate Power Allocation Register Address 3B0 is for Substate 0 Address 3B1 is for Substate 1, up to Substate Substate_Max

5.2.18. L1 PM Substates Extended Capability

Table 5.225. L1 PM Substates Extended Capability

Addr	Config Register	Register Description
3D3-3D0	L1 PM Substates Capability Extended Capability Header	Beginning of L1 PM Substates Extended Capability – Read Only • Bits[31:20] – Pointer to next Enhanced/Extended Capability Item in the list. • Bits[19:16] = 0x1 == Capability Version • Bits[15:0] = 0x001E == Capability ID
3D7-3D4	L1 PM Substates Capabilities Register	HwInit • Bits[31:24] – Reserved • Bits[23:19] – Port TPOWER_ON Value • Bits [18] – Reserved • Bits[17:16] – Port TPOWER_ON Scale • Bits[15:8] – Port Common_Mode_Restore_Time (in µs) • Bits[7:5] – Reserved • Bit[4] – L1 PM Substates Supported • Bit[3] – ASPM L1.1 Supported

Addr	Config Register	Register Description
		• Bit[2] – ASPM L1.2 Supported • Bit[1] – PCI-PM L1.1 Supported • Bit[0] – PCI-PM L1.2 Supported
3DB-3D8	L1 PM Substates Control 1 Register	Read/Write • Bits[31:29] – LTR_L1.2_THRESHOLD_Scale • Bits[28:26] – Reserved • Bits[25:16] – LTR_L1.2_THRESHOLD_Value • Bits[15:8] – Common_Mode_Restore_Time • Bits[7:4] – Reserved • Bit[3] – ASPM L1.1 Enable • Bit[2] – ASPM L1.2 Enable • Bit[1] – PCI-PM L1.1 Enable • Bit[0] – PCI-PM L1.2 Enable
3DF-3DC	L1 PM Substates Control 2 Register	Read/Write • Bits[31:8] – Reserved • Bits[7:3] – TPOWER_ON Value • Bit[2] – Reserved • Bits[1:0] – T_{POWER_ON} Scale

5.2.19. Latency Tolerance Reporting Capability

Table 5.226. Latency Tolerance Reporting (LTR) Capability

Addr	Config Register	Register Description
3E3-3E0	LTR Capability Extended Capability Header	Beginning of LTR Extended Capability – Read Only • Bits[31:20] – Pointer to next Enhanced/Extended Capability Item in the list. • Bits[19:16] = 0x1 == Capability Version • Bits[15:0] = 0x0018 == Capability ID
3E5-3E4	Max Snoop Latency Register	Read/Write • Bits[15:13] – Reserved • Bits[12:10] – Max Snoop LatencyScale • Bits[9:0] – Max Snoop LatencyValue
3E7-3E6	Max No-Snoop Latency Register	Read/Write • Bits[15:13] – Reserved • Bits[12:10] – Max No-Snoop LatencyScale • Bits[9:0] – Max No-Snoop LatencyValue

5.3. DMA Configuration Space Registers

Each DMA Channel implements 128 (0x80) bytes of DMA Registers. DMA Channel Registers are accessed through the PCI Express through the PCI Express Base Address Register associated with DMA Channel Registers. When multiple PCIe functions are present, each function is assigned to two DMA Channels and each function's share of DMA Channels is accessed through the function's associated DMA Register BAR. DMA Channel Registers within the same function are packed back-back.

To discover all 64 DMA Channels, 8 physical functions and SR-IOV need to be enabled. This gives a total of 32 functions available to the PCIe Endpoint.

Table 5.227. DMA Configuration Space Registers

Register Offset	Config Register	Register Description

0x0	SRC_Q_PTR_LO	• Bits[31:6] – Queue Base Address [31:6]. Queues are required to be 64-byte aligned.
		• Bits[5:2] – Queue Read Request Attributes. Transaction attributes used by Queue read requests when reading SGL elements. If Queue Location == AXI, read_attr[3:0] is used for m_arcache[3:0]. If Queue Location == PCIe, read_attr[2:0] is used for PCIe Attr[2:0].
		• Bits[1] – Queue Enable
		  • 0 – Disabled. The DMA Channel will not read from the queue.
		  • 1 – Enabled. The DMA Channel will fetch queue elements as needed from queue as long as Q_LIMIT != Q_NEXT
		• Bits[0] – Queue Location
		  • 0 – Queue is in PCIe Memory
		  • 1 – Queue is in AXI Memory

0x4	SRC_Q_PTR_HI	• Bits[31:0] – Queue Base Address [63:32]. Must be set to 0x0 if the Queue is in 32-bit address space.
0x8	SRC_Q_SIZE	• Bits[31:0] – Queue Size. Number of Elements in the Queue. queue_size must be >= 2. A minimum of 2 elements is required to support software/hardware queue flow control ownership. queue_size is used to identify the wrap boundary of the Queue.
0xC	SRC_Q_LIMIT	• Bits[31:0] – Queue Flow Control - Limit Pointer. Index of the first Queue element still “owned” by software. Incremented by software to give the DMA Channel additional elements to execute. DMA Channel hardware will pause and not utilize queue elements when Q_LIMIT is reached until Q_LIMIT is advanced to provide additional elements to execute.

0x10	DST_Q_PTR_LO	• Bits[31:6] – Queue Base Address [31:6]. Queues are required to be 64-byte aligned.
		• Bits[5:2] – Queue Read Request Attributes. Transaction attributes used by Queue read requests when reading SGL elements. If Queue Location == AXI, read_attr[3:0] is used for m_arcache[3:0]. If Queue Location == PCIe, read_attr[2:0] is used for PCIe Attr[2:0].
		• Bits[1] – Queue Enable
		  • 0 – Disabled. The DMA Channel will not read from the queue.
		  • 1 – Enabled. The DMA Channel will fetch queue elements as needed from queue as long as Q_LIMIT != Q_NEXT
		• Bits[0] – Queue Location
		  • 0 – Queue is in PCIe Memory
		  • 1 – Queue is in AXI Memory

0x14	DST_Q_PTR_HI	• Bits[31:0] – Queue Base Address [63:32]. It must be set to 0x0 if the Queue is in 32-bit address space.
0x18	DST_Q_SIZE	• Bits[31:0] – Queue Size. Number of Elements in the Queue. queue_size must be >= 2. A minimum of 2 elements is required to support software/hardware queue flow control ownership. queue_size is used to identify the wrap boundary of the Queue.
0x1C	DST_Q_LIMIT	• Bits[31:0] – Queue Flow Control - Limit Pointer. Index of the first Queue element still “owned” by software. Incremented by software to give the DMA Channel additional elements to execute. DMA Channel hardware will pause and not utilize queue elements when Q_LIMIT is reached until Q_LIMIT is advanced to provide additional elements to execute.

0x20	STAS_Q_PTR_LO	• Bits[31:6] – Queue Base Address [31:6]. Queues are required to be 64-byte aligned.
		• Bits[5:2] – Queue Read Request Attributes. Transaction attributes used by Queue read requests when reading SGL elements. If Queue Location == AXI, read_attr[3:0] is used for m_arcache[3:0]. If Queue Location == PCIe, read_attr[2:0] is used for PCIe Attr[2:0].
		• Bits[1] – Queue Enable
		  • 0 – Disabled. The DMA Channel will not read from the queue.
		  • 1 – Enabled. The DMA Channel will fetch queue elements as needed from queue as long as Q_LIMIT != Q_NEXT

Register Offset	Config Register	Register Description

		• Bits[0] – Queue Location
		  • 0 – Queue is in PCIe Memory
		  • 1 – Queue is in AXI Memory

0x24	STAS_Q_PTR_HI	• Bits[31:0] – Queue Base Address [63:32]. Must be set to 0x0 if the Queue is in 32-bit address space.
0x28	STAS_Q_SIZE	• Bits[31:0] – Queue Size. Number of Elements in the Queue. queue_size must be >= 2. A minimum of 2 elements is required to support software/hardware queue flow control ownership. queue_size is used to identify the wrap boundary of the Queue.
0x2C	STAS_Q_LIMIT	• Bits[31:0] – Queue Flow Control - Limit Pointer. Index of the first Queue element still “owned” by software. Incremented by software to give the DMA Channel additional elements to execute. DMA Channel hardware will pause and not utilize queue elements when Q_LIMIT is reached until Q_LIMIT is advanced to provide additional elements to execute.

0x30	STAD_Q_PTR_LO	• Bits[31:6] – Queue Base Address [31:6]. Queues are required to be 64-byte aligned.
		• Bits[5:2] – Queue Read Request Attributes. Transaction attributes used by Queue read requests when reading SGL elements. If Queue Location == AXI, read_attr[3:0] is used for m_arcache[3:0]. If Queue Location == PCIe, read_attr[2:0] is used for PCIe Attr[2:0].
		• Bits[1] – Queue Enable
		  • 0 – Disabled. The DMA Channel will not read from the queue.
		  • 1 – Enabled. The DMA Channel will fetch queue elements as needed from queue as long as Q_LIMIT != Q_NEXT
		• Bits[0] – Queue Location
		  • 0 – Queue is in PCIe Memory
		  • 1 – Queue is in AXI Memory

0x34	STAD_Q_PTR_HI	• Bits[31:0] – Queue Base Address [63:32]. Must be set to 0x0 if the Queue is located in 32-bit address space.
0x38	STAD_Q_SIZE	• Bits[31:0] – Queue Size. Number of Elements in the Queue. queue_size must be >= 2. A minimum of 2 elements is required to support software/hardware queue flow control ownership. queue_size is used to identify the wrap boundary of the Queue.
0x3C	STAD_Q_LIMIT	• Bits[31:0] – Queue Flow Control - Limit Pointer. Index of the first Queue element still “owned” by software. Incremented by software to give the DMA Channel additional elements to execute. DMA Channel hardware will pause and not utilize queue elements when Q_LIMIT is reached until Q_LIMIT is advanced to provide additional elements to execute.
0x40	SRC_Q_NEXT	• Bits[31:0] – Queue Flow Control - Next Pointer. Index of the next Queue element that will be read by DMA Channel hardware. Incremented by DMA Channel hardware as queue read requests are generated. The number of queue elements available for the DMA Channel == SRC_Q_LIMIT - SRC_Q_NEXT (considering wrapping). SRC_Q_NEXT does not indicate that Queue Elements have been completed, only that the DMA Channel has started processing for the queue elements. Software must write this register to 0x0 to initialize the queue prior to enabling the DMA Channel. Software is prohibited from writing this register while the DMA Channel is enabled. SRC_Q_NEXT is utilized by DMA Channel hardware to track its location in the queue and should not be used by DMA software.
0x44	DST_Q_NEXT	• Bits[31:0] – Queue Flow Control - Next Pointer. Index of the next Queue element that will be read by DMA Channel hardware. Incremented by DMA Channel hardware as queue read requests are generated. The number of queue elements available for the DMA Channel == SRC_Q_LIMIT - SRC_Q_NEXT (considering wrapping). SRC_Q_NEXT does not indicate that Queue Elements have been completed, only that the DMA Channel has started processing for the queue elements. Software must write this register to 0x0 to initialize the queue prior to enabling the DMA Channel. Software is prohibited from writing this register while the DMA Channel is enabled. SRC_Q_NEXT

Register Offset	Config Register	Register Description
		is utilized by DMA Channel hardware to track its location in the queue and should not be used by DMA software.
0x48	STAS_Q_NEXT	• Bits[31:0] – Queue Flow Control - Next Pointer. Index of the next Queue element that will be read by DMA Channel hardware. Incremented by DMA Channel hardware as queue read requests are generated. The number of queue elements available for the DMA Channel == SRC_Q_LIMIT - SRC_Q_NEXT (considering wrapping). SRC_Q_NEXT does not indicate that Queue Elements have been completed, only that the DMA Channel has started processing for the queue elements. Software must write this register to 0x0 to initialize the queue prior to enabling the DMA Channel. Software is prohibited from writing this register while the DMA Channel is enabled. SRC_Q_NEXT is utilized by DMA Channel hardware to track its location in the queue and should not be used by DMA software.
0x4C	STAD_Q_NEXT	• Bits[31:0] – Queue Flow Control - Next Pointer. Index of the next Queue element that will be read by DMA Channel hardware. Incremented by DMA Channel hardware as queue read requests are generated. The number of queue elements available for the DMA Channel == SRC_Q_LIMIT - SRC_Q_NEXT (considering wrapping). SRC_Q_NEXT does not indicate that Queue Elements have been completed, only that the DMA Channel has started processing for the queue elements. Software must write this register to 0x0 to initialize the queue prior to enabling the DMA Channel. Software is prohibited from writing this register while the DMA Channel is enabled. SRC_Q_NEXT is utilized by DMA Channel hardware to track its location in the queue and should not be used by DMA software.
0x50	SCRATCH0	• Bits[31:0] – Scratchpad Register. Intended to enable information to be passed between software. For example, applications with both an AXI CPU and an PCIe CPU may use this register to pass information between CPUs. The DMA Channel implementation does not use or alter this information.
0x54	SCRATCH1	• Bits[31:0] – Scratchpad Register. Intended to enable information to be passed between software. For example, applications with both an AXI CPU and an PCIe CPU may use this register to pass information between CPUs. The DMA Channel implementation does not use or alter this information.
0x58	SCRATCH2	• Bits[31:0] – Scratchpad Register. Intended to enable information to be passed between software. For example, applications with both an AXI CPU and an PCIe CPU may use this register to pass information between CPUs. The DMA Channel implementation does not use or alter this information.
0x5C	SCRATCH3	• Bits[31:0] – Scratchpad Register. Intended to enable information to be passed between software. For example, applications with both an AXI CPU and an PCIe CPU may use this register to pass information between CPUs. The DMA Channel implementation does not use or alter this information.
0x60	PCIE_INTERRUPT_CONTROL	• Bits[31:24] – Reserved. • Bits[23:16] – PCIe DMA SGL Interrupt Coalesce Count. Controls the frequency at which a DMA SGL Interrupt Event, completion of a source SGL Element that had both the "EOP" and "Interrupt" SGL control bits set to 1, causes PCIe Interrupts. An internal DMA Channel-specific PCIe DMA SGL Coalesce Counter is maintained. For each DMA SGL Interrupt Event, if the current PCIe DMA SGL Coalesce Counter is equal to PCIe DMA SGL Interrupt Coalesce Count, then a PCIe interrupt is generated and the PCIe DMA SGL Coalesce Counter is cleared otherwise the PCIe DMA SGL Coalesce Counter is incremented. The PCIe DMA SGL Coalesce Counter is also cleared when DMA Enable == 0. This mechanism allows software to configure the DMA Channel to interrupt PCIe once every 1-256 DMA SGL Interrupt Events. When a non-0 value is programmed into PCIe DMA SGL Interrupt Coalesce Count, software must anticipate that at the end of all DMA packet transfers, interrupts may be pending in the PCIe DMA SGL Coalesce Counter that does not cause an interrupt because no more interrupts arrive for PCIe DMA SGL Coalesce Counter to reach the PCIe DMA SGL Interrupt Coalesce Count threshold. • Bits[15:3] – Reserved.

Register Offset	Config Register	Register Description

		• Bits[2] – PCIe Enable DMA SGL Interrupt Event
		  • 0 – Disabled. DMA SGL Interrupt Events are ignored and will not cause PCI Express interrupts to be generated.
		  • 1 – Enabled. DMA SGL Interrupt Event interrupts are enabled for this DMA Channel.
		• Bits[1] – PCIe Enable DMA Error Interrupt Event.
		  • 0 – Disabled. DMA Error Interrupt Events are ignored and will not cause PCI Express interrupts to be generated.
		  • 1 – Enabled. DMA Error Interrupt Event interrupts are enabled for this DMA Channel.
		• Bits[0] – PCIe Interrupt Enable
		  • 0 – Disable Interrupts. Hold PCIe interrupts which occur for this DMA Channel and assert them when PCIe Interrupt Enable is set back to 1. Pending interrupts are cleared when DMA_Enable == 0.
		  • 1 – Enable Interrupts. Allow PCIe interrupts to be asserted by this DMA Channel.

0x64	PCIE_INTERRUPT_STATUS	• Bits[31:4] – Reserved.
		• Bits[3] – PCIe Software Interrupt Status.
		  • 0 – No software interrupt pending
		  • 1 – A Software Interrupt was generated
		• Bits[2] – PCIe DMA SGL Interrupt Status.
		  • 0 – No DMA interrupt pending
		  • 1 – DMA SGL Interrupt Events caused an interrupt to be generated
		• Bits[1] – PCIe DMA Error Interrupt Status.
		  • 0 – No DMA interrupt pending
		  • 1 – DMA Error Interrupt Events caused an interrupt to be generated
		• Bits[0] – Reserved.

0x68	AXI_INTERRUPT_CONTROL	• Bits[31:24] – Reserved.
		• Bits[23:16] – AXI DMA SGL Interrupt Coalesce Count. Controls the frequency at which a DMA SGL Interrupt Event (completion of a source SGL Element that had both the "EOP" and "Interrupt" SGL control bits set to 1) causes AXI Interrupts. An internal DMA Channel-specific AXI DMA SGL Coalesce Counter is maintained. For each DMA SGL Interrupt Event, if the current AXI DMA SGL Coalesce Counter is equal to AXI DMA SGL Interrupt Coalesce Count, then a AXI interrupt is generated, and the AXI DMA SGL Coalesce Counter is cleared otherwise the AXI DMA SGL Coalesce Counter is incremented. The AXI DMA SGL Coalesce Counter is also cleared when DMA Enable == 0. This mechanism allows software to configure the DMA Channel to interrupt AXI once every 1-256 DMA SGL Interrupt Events. When a non-0 value is programmed into AXI DMA SGL Interrupt Coalesce Count, software must anticipate that at the end of all DMA packet transfers, interrupts may be pending in the AXI DMA SGL Coalesce Counter that will not cause an interrupt because no more interrupts will arrive for AXI DMA SGL Coalesce Counter to reach the AXI DMA SGL Interrupt Coalesce Count threshold.
		• Bits[15:3] – Reserved.
		• Bits[2] – AXI Enable DMA SGL Interrupt Event
		  • 0 – Disabled. DMA SGL Interrupt Events are ignored and will not cause PCI Express interrupts to be generated.
		  • 1 – Enabled. DMA SGL Interrupt Event interrupts are enabled for this DMA Channel.
		• Bits[1] – AXI Enable DMA Error Interrupt Event.
		  • 0 – Disabled. DMA Error Interrupt Events are ignored and will not cause PCI Express interrupts to be generated.

Register Offset	Config Register	Register Description
		• 1 – Enabled. DMA Error Interrupt Event interrupts are enabled for this DMA Channel. • Bits[0] – AXI Interrupt Enable • 0 – Disable Interrupts. Hold PCIe interrupts which occur for this DMA Channel and assert them when PCIe Interrupt Enable is set back to 1. Pending interrupts are cleared when DMA_Enable == 0. • 1 – Enable Interrupts. Allow PCIe interrupts to be asserted by this DMA Channel.
0x6C	AXI_INTERRUPT_STATUS	• Bits[31:4] – Reserved. • Bits[3] – AXI Software Interrupt Status. • 0 – No software interrupt pending • 1 – A Software Interrupt was generated • Bits[2] – AXI DMA SGL Interrupt Status. • 0 – No DMA interrupt pending • 1 – DMA SGL Interrupt Events caused an interrupt to be generated • Bits[1] – AXI DMA Error Interrupt Status. • 0 – No DMA interrupt pending • 1 – DMA Error Interrupt Events caused an interrupt to be generated • Bits[0] – Reserved.
0x70	PCIE_INTERRUPT_ASSERT	• Bits[31:4] – Reserved. • Bits[3] – PCIe Software Interrupt. Write a 1 to this register to generate a PCIe Software Interrupt. An interrupt is generated and propagated through the PCI Express Core to PCIe in the same manner as a DMA Channel interrupt (using the same Interrupt Vector). Interrupts are generated by writes to this register independent of whether the DMA Channel is enabled. • Bits[2:0] – Reserved.
0x74	AXI_INTERRUPT_ASSERT	• Bits[31:4] – Reserved. • Bits[3] – AXI Software Interrupt. Write a 1 to this register to generate a AXI Software Interrupt. An interrupt is generated and propagated through to AXI in the same manner as a DMA Channel interrupt (using the same Interrupt Vector). Interrupts are generated by writes to this register independent of whether the DMA Channel is enabled. • Bits[2:0] – Reserved.
0x78	DMA_CONTROL	• Bits[31:3] – Reserved. • Bits[2] – DMA Channel Completion Status Queue Element Size. If User ID and/or User Handle information is needed for an application, the application must setup the DMA Channel with 64-bit Status Queue Elements. If these features are not needed, then the DMA Channel may be setup with 32-bit Status Queue Elements to reduce the bus utilization required to write status Queue Elements. • 0 – 32-bit - Status Queue elements contain only essential DMA information • 1 – 64-bit - Status Queue elements include additional User ID and/or User Handle information that is useful for some applications • Bits[1] – DMA Channel Reset. Each DMA Channel has a small Source SGL FIFO, Destination SGL FIFO, Source DMA Completion Status FIFO, and Destination DMA Completion Status FIFO to enable overlapping of DMA transactions for higher throughput. When a DMA Channel is disabled, these FIFOs may not empty fully and may need to be flushed before the DMA Channel can be reused for a new operation. • 0 – Normal operation. • 1 – Reset DMA Channel Source SGL FIFO, Destination SGL FIFO, Source DMA Completion Status FIFO, and Destination DMA Completion Status FIFO • Bits[0] – DMA Channel Enable • 0 – Disable DMA Channel. DMA Channel Queues must be configured while the DMA Channel is disabled. After disabling a DMA Channel, software must wait for

Register Offset	Config Register	Register Description
		all outstanding DMA transactions to complete before re-enabling the DMA Channel or software can cause the prior disabled, but not yet quiescent DMA operation, to fail potentially causing a fatal error. DMA Running can be read to determine when a prior DMA operation has completed. • 1 – Enable DMA Channel. DMA Channel fetches Source and Destination Queue Scatter Gather Elements, executes them, and writes DMA Packet Completion status to the Source and Destination DMA Completion Status Queues. The DMA Channel pauses DMA data transactions whenever it does not have at least 1 queue element available for execution in all queues. Queue LIMIT registers are advanced to provide additional elements to execute.

0x7C	DMA_STATUS	• Bits[31:16] – Reserved.
		• Bits[15] – DMA Channel Present. During initialization, the DMA Driver can read this register at all possible DMA Channel Register locations to determine how many DMA Channels are implemented.
		  • 0 – Otherwise
		  • 1 – A DMA Channel Register set is present at this location
		• Bits[14] – Reserved.
		• Bits[13:4] – DMA Channel Number. Unique DMA Channel Number assigned to this DMA Channel. The DMA Channel Number register is for informational purposes and is not needed for DMA operation. DMA Channel number is unique for each DMA Channel even for multi-function and SR-IOV applications.
		• Bits[3:1] – Reserved.
		• Bits[0] – DMA Running. Prior to transitioning DMA Enable from 0 to 1 or modifying the contents of the DMA Channel's Queue Management Registers, software must read DMA Running == 0 to verify that the prior DMA operation completed and that it is safe to re-initialize and re-start the DMA Channel. The DMA Channel Source and Destination Queues can be re-used for new DMA operations (by changing the Queue Scatter-Gather List contents) without having to disable the DMA Channel. In general software should setup the Source SGL, Destination SGL, Source DMA Completion Status, and Destination DMA Completion Status Queues once at driver initialization and keep the DMA Channel Enabled for the entire time the driver is loaded. When there is new work to do, the Driver makes the associated queue elements available to be executed by updating the Queue LIMIT pointers.
		  • 0 – DMA Channel is idle.
		  • 1 – DMA Channel is busy processing.

6. Example Design

The PCIe x8 IP example designs are available for simulation and hardware in this IP version. The steps to run the functional simulation are described in the Running Functional Simulation section. The PCIe x8 IP generates two types of example designs:

- DMA Design
- Non-DMA Design

6.1. Example Design Supported Configuration

Based on the supported configurations listed in Table 6.1, you can configure the desired parameters through the graphical user interface to generate the appropriate IP configuration for the Example Design. For parameters without selectable options stated in Table 6.1, you are expected to configure the parameters accordingly in the graphical user interface.

Table 6.1. PCIe x8 IP Configuration Supported by the Example Design

PCIe x8 IP User Interface Parameter	PCIe x8 IP Configuration Supported in the Example Demo Design	
	DMA Design	Non-DMA Design
Bifurcation Select (Link_X_Lane)	1 × 8 (Hardened DMA) 1 × 1, 1 × 2, 1 × 4 (DMA)	1 × 1, 1 × 2, 1 × 4, 1 × 8
Target Link Speed	Gen 1, Gen 2, Gen 3[2], Gen 4[2] (Hardened DMA) Gen 1, Gen 2, Gen 3 (DMA)	Gen 1, Gen 2, Gen 3[2], Gen 4[2]
Data Interface Type	AXI-MM	TLP AXI-MM (Bridge Mode) AXI-Lite (Bridge Mode)
Register Interface Type	LMMI	LMMI
Link 0 : Number of Physical Function	1 is supported (Function 0)	1 is supported (Function 0)
Flow Control Tab	N/A	Refer to Flow Control Update section for the configuration performed in this tab.
Configuration Device ID and Vendor ID Subsystem ID Subsystem Vendor ID Class Code and Revision ID	Default value shown in the User Interface	Device ID and Vendor ID are configured from the graphical user interface or AXI-Lite and the rest are default
BAR 0 Enable	✓[1]	✓[1]
BAR 1 Enable	✓[1]	✓[1]
BAR 2, BAR 3, BAR 4, BAR 5 Enable	X[1]	X[1]
Disable Legacy Interrupt	X[1]	X[1]
Disable MSI Capability	X[1]	✓[1]
Disable MSI-X capability	✓[1]	✓[1]
Maximum Payload Size Supported	128 bytes, 256 bytes, or 512 bytes	128 bytes, 256 bytes, or 512 bytes
Disable Function Level Reset	✓[1]	✓[1]
Enable Extended Tag Filed	X[1]	✓[1]

PCIe x8 IP User Interface Parameter	PCIe x8 IP Configuration Supported in the Example Demo Design	
	DMA Design	Non-DMA Design
Advance Error Reporting Capability	X[1]	Refer to the Advance Error Reporting Capability section for the configuration done in this tab.

Notes:

1. ✓ refers to a checked option in the PCIe x8 IP example design and X refers to an unchecked option or a non-applicable option in the PCIe x8 IP example design.
2. This link speed is available from 2024.1 release onwards.

6.2. Overview of the Example Design and Features

Figure 6.1 illustrates the PCIe example design architecture. It shows how test cases interact with the PCIe Rootport BFM and the Lattice PCIe endpoint, which connects to user logic blocks with DMA enabled or disabled. Configuration parameters such as PCIe generation, lane width, interface type, and DMA settings are applied through the dut_params.v module generated as part of the PCIe IP.

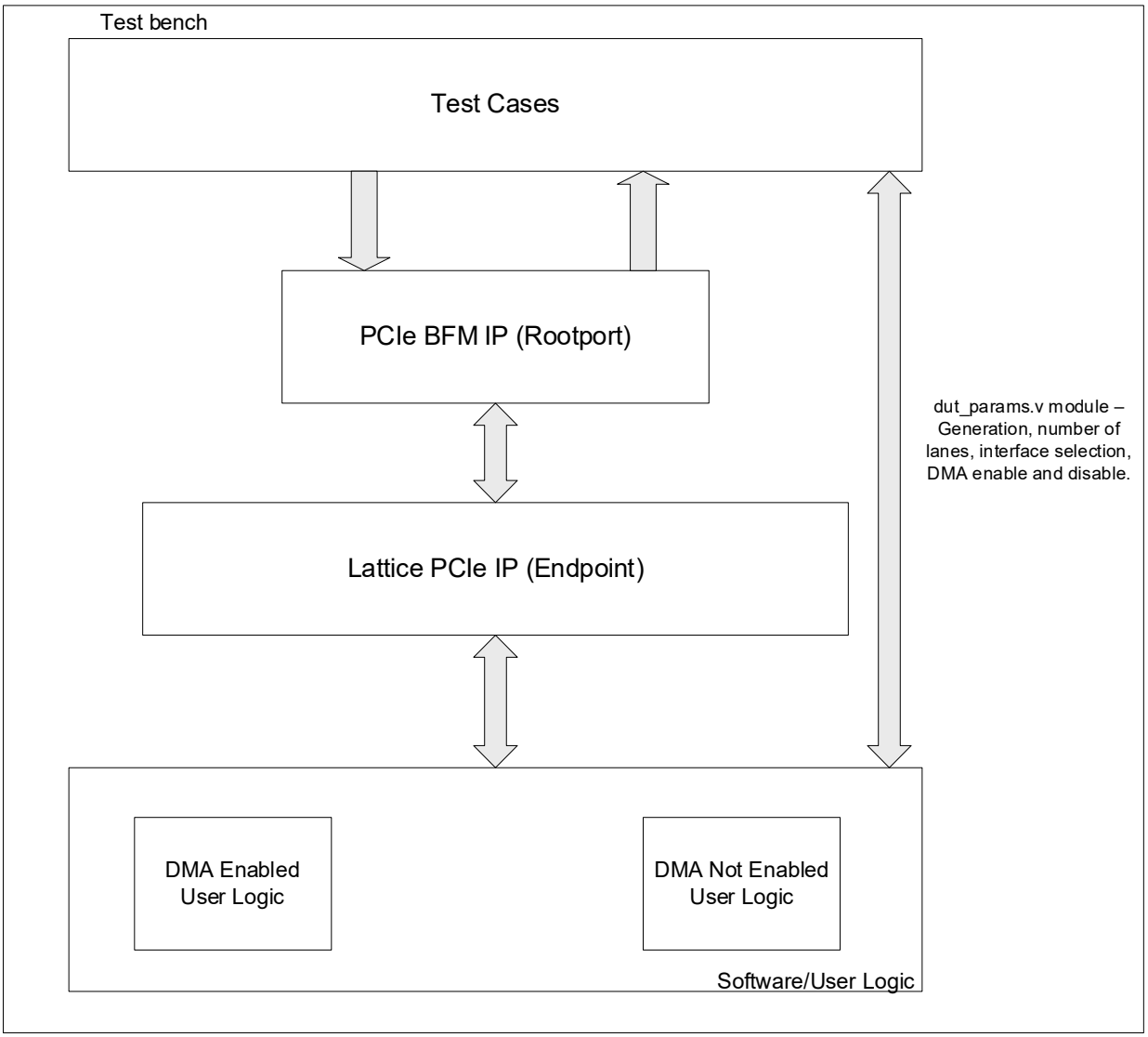


Figure 6.1. PCIe x8 IP Example Design Block Diagram

6.3. Example Design Components

6.3.1. DMA Design (Hardened DMA)

The PCIe x8 implements the hardened DMA example design with the following components:

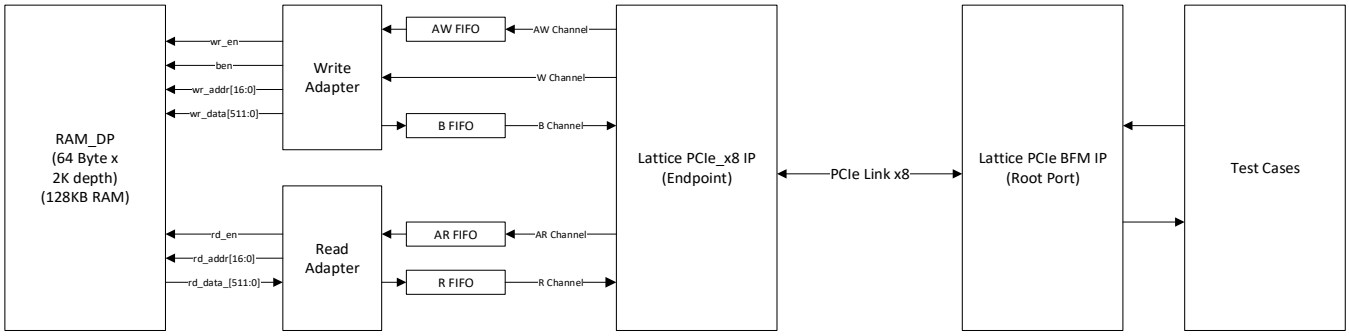


Figure 6.2. Components within the Hardened DMA Design

The DMA Example Design supports both DMA mode and non-DMA mode (DMA Bypass mode). In DMA mode, the DUT fetches data from RAM_DP to Root Port or vice versa depending on DMA Descriptors. In DMA Bypass mode, the read/write operation from Root Port bypasses the DMA engine and is forwarded out to RAM_DP through the same AXI interface. The DMA Bypass mode is accessed through BAR1 while all DMA traffic uses BAR0.

Note: Both DMA and non-DMA traffic in the Example Design targets RAM_DP. This is done by ignoring the upper bits of BAR0 and BAR1 address (bit [63:17]) on the AXI address bus.

6.3.1.1. Generating Hardened DMA Example Design

To successfully generate the DMA example design for hardware using the Lattice Radiant software, perform the following steps:

1. Create a Lattice Radiant software project. Double-click the **PCIE_X8** in the IP Catalog, configure the IP as required for your specific application. Be aware of the supported configurations when determining the configuration of the IP. The limitations are stipulated in the Limitations of the Example Design section.

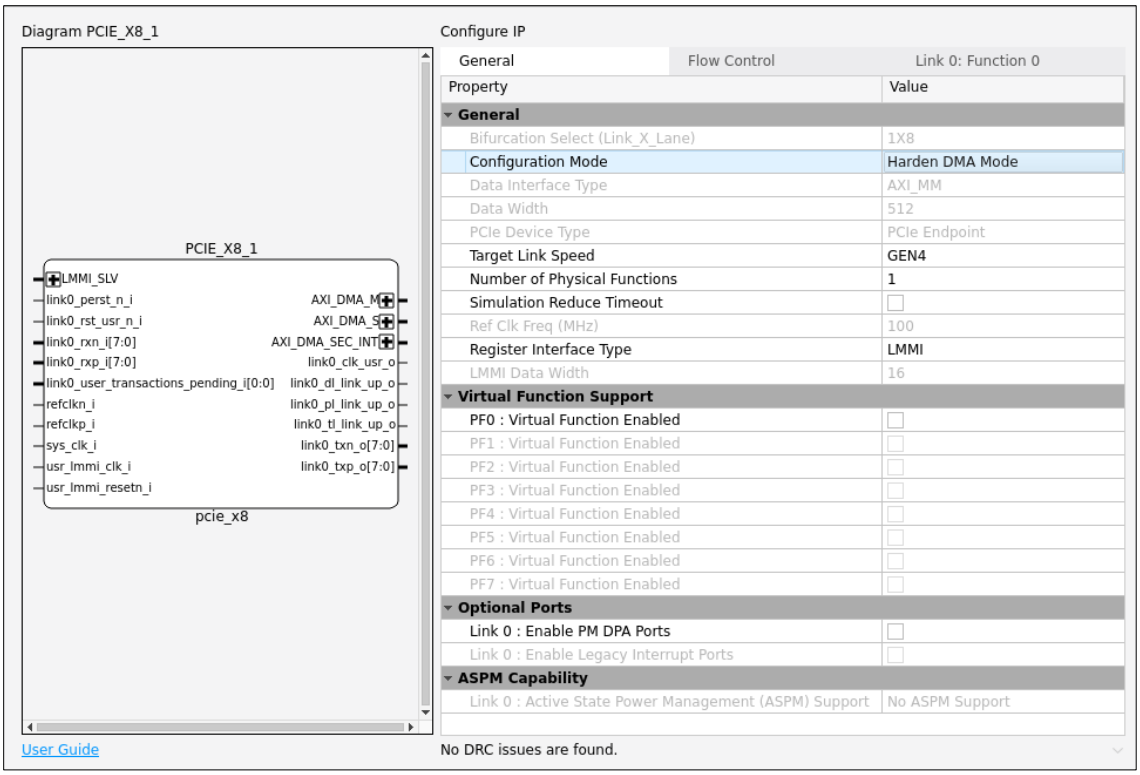


Figure 6.3. IP Configuration for Hardened DMA Example Design – General Tab

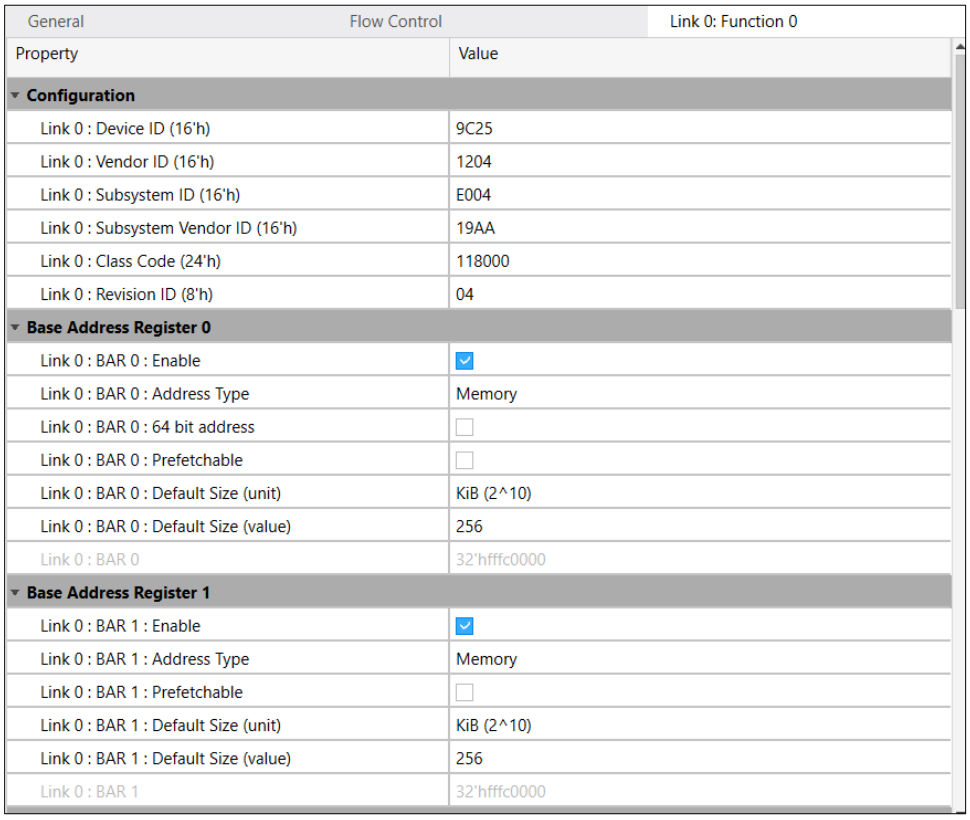
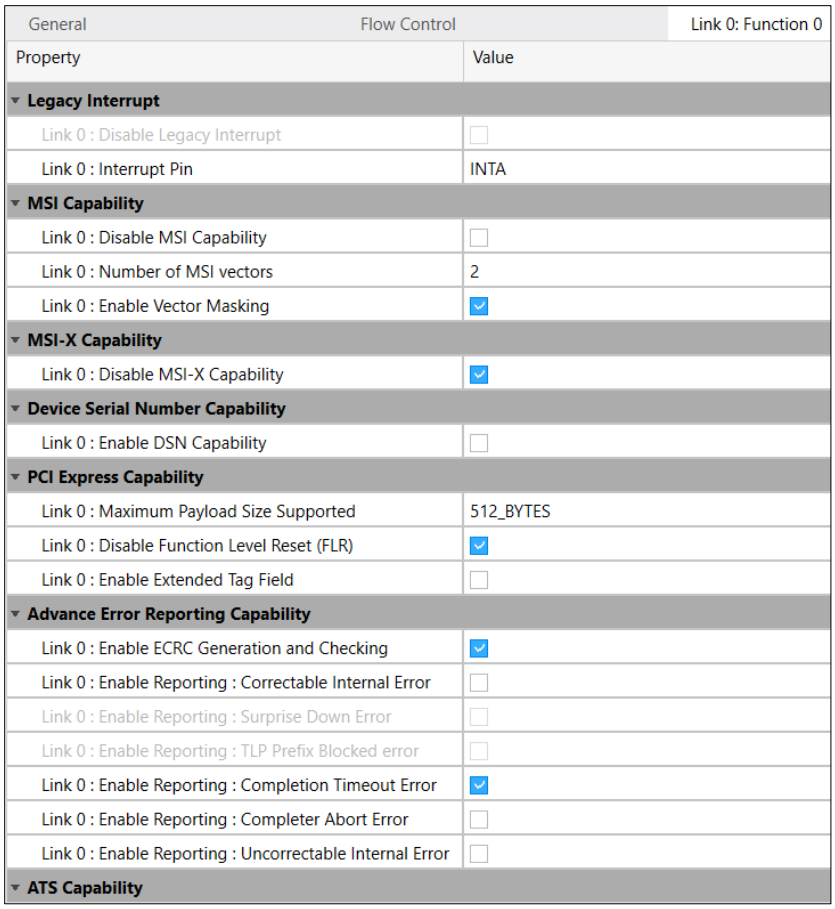


Figure 6.4. IP Configuration for Hardened DMA Example Design – Link 0: Function 0 Tab (Part 1)



General	Flow Control	Link 0: Function 0
Property	Value	
Legacy Interrupt		
Link 0 : Disable Legacy Interrupt	☐	
Link 0 : Interrupt Pin	INTA	
MSI Capability		
Link 0 : Disable MSI Capability	☐	
Link 0 : Number of MSI vectors	2	
Link 0 : Enable Vector Masking	☑	
MSI-X Capability		
Link 0 : Disable MSI-X Capability	☑	
Device Serial Number Capability		
Link 0 : Enable DSN Capability	☐	
PCI Express Capability		
Link 0 : Maximum Payload Size Supported	512_BYTES	
Link 0 : Disable Function Level Reset (FLR)	☑	
Link 0 : Enable Extended Tag Field	☐	
Advance Error Reporting Capability		
Link 0 : Enable ECRC Generation and Checking	☑	
Link 0 : Enable Reporting : Correctable Internal Error	☐	
Link 0 : Enable Reporting : Surprise Down Error	☐	
Link 0 : Enable Reporting : TLP Prefix Blocked error	☐	
Link 0 : Enable Reporting : Completion Timeout Error	☑	
Link 0 : Enable Reporting : Completer Abort Error	☐	
Link 0 : Enable Reporting : Uncorrectable Internal Error	☐	
ATS Capability		

Figure 6.5. IP Configuration for Hardened DMA Example Design – Link 0: Function 0 Tab (Part 2)

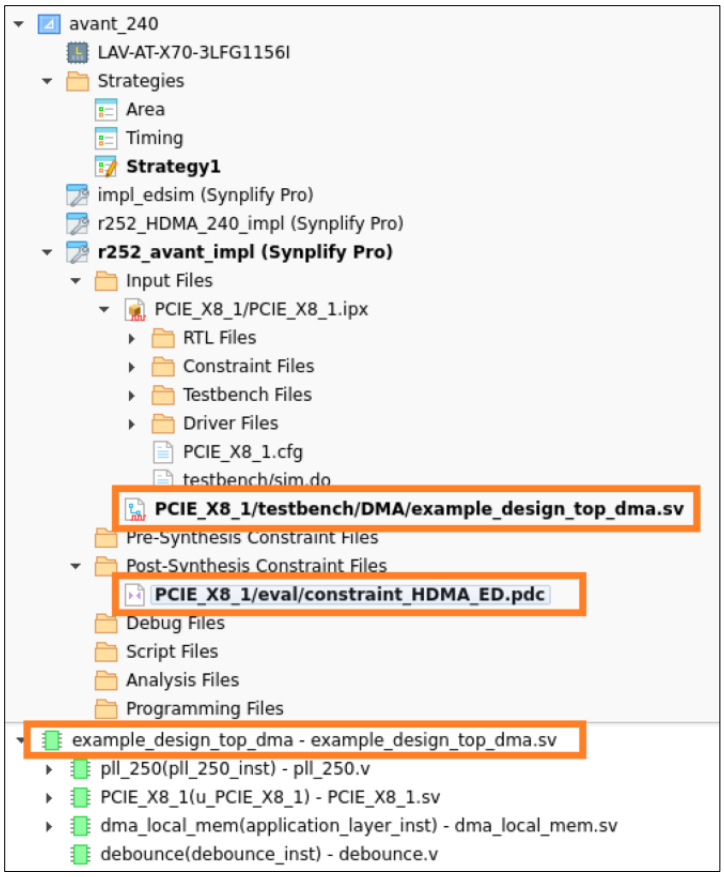


Figure 6.6. File List View of the Created Hardened DMA Example Design

2. Add the DMA files. Right-click on the Input Files and select **Add > Existing Files**. Manually add the DMA Example Design example_design_top_dma.sv file located in the folder of the generated IP under the *<instance name>/testbench/DMA* directory.
3. Set **example_design_top_dma.sv** as the top level of the design.
4. Select the PDC file: **<instance name>/eval/constraint_HDMA_ED.pdc**.

6.3.2. DMA Design (DMA)

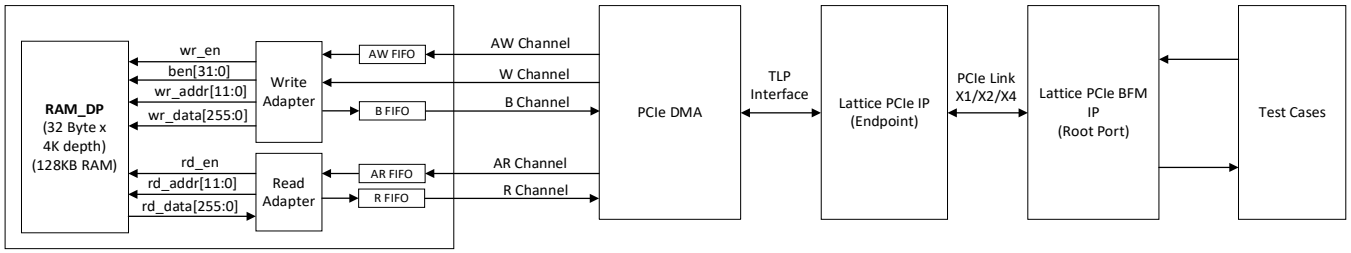


Figure 6.7. Components within DMA Example Design

The PCIe x8 Example Design implements the DMA Design with the following components:

- AR FIFO, B FIFO, AR FIFO, R FIFO – FIFOs that stores information from AXI-MM interface.
- RAM_DP – A True Dual Port RAM that is configured to 128 kB size, with 32 Byte width and 4K depth. It is a storage on FPGA for F2H and H2F data transfer.
- Write Adapter – A component that converts AXI-MM write to RAM_DP write interface.
- Read Adapter – A component that converts AXI-MM read to RAM_DP read interface.

- PCIe DMA – The PCIe IP DMA is used to implement the DMA Operations.

The following shows the DMA Design's data flow:

- Read the configuration of the Lattice PCIe x8 IP DMA
- The BFM waits for the Linkup to occur.
- The BFM setups a single entry of H2F descriptor table with transfer size of 16 kB and with INTR and EOP bits set to 1.
- The BFM programs the PCIe DMA H2F registers to start DMA transfer.
- The BFM waits for MSI interrupt from DUT.
- The BFM setups a single entry of F2H descriptor table with transfer size of 16 kB and with INTR and EOP bits set to 1.
- The BFM programs the PCIe DMA F2H registers to start DMA transfer.
- The BFM waits for MSI interrupt from DUT.
- The BFM does data comparison for F2H and H2H to make sure they are intact.

6.3.2.1. Generating DMA Example Design

To generate the DMA example design:

1. Create a Lattice Radiant software project. Double-click the **PCIE_X8** in the **IP Catalog** and generate the IP by selecting **DMA only Mode** at **Configuration Mode** and **AXI_MM** at **Data Interface Type** drop-down menu.
2. Configure Target Link Speed and Bifurcation Select. Right-click on Input Files and select **Add > Existing Files**.
3. Add **<Component Name>/testbench/example_design_top.sv**.

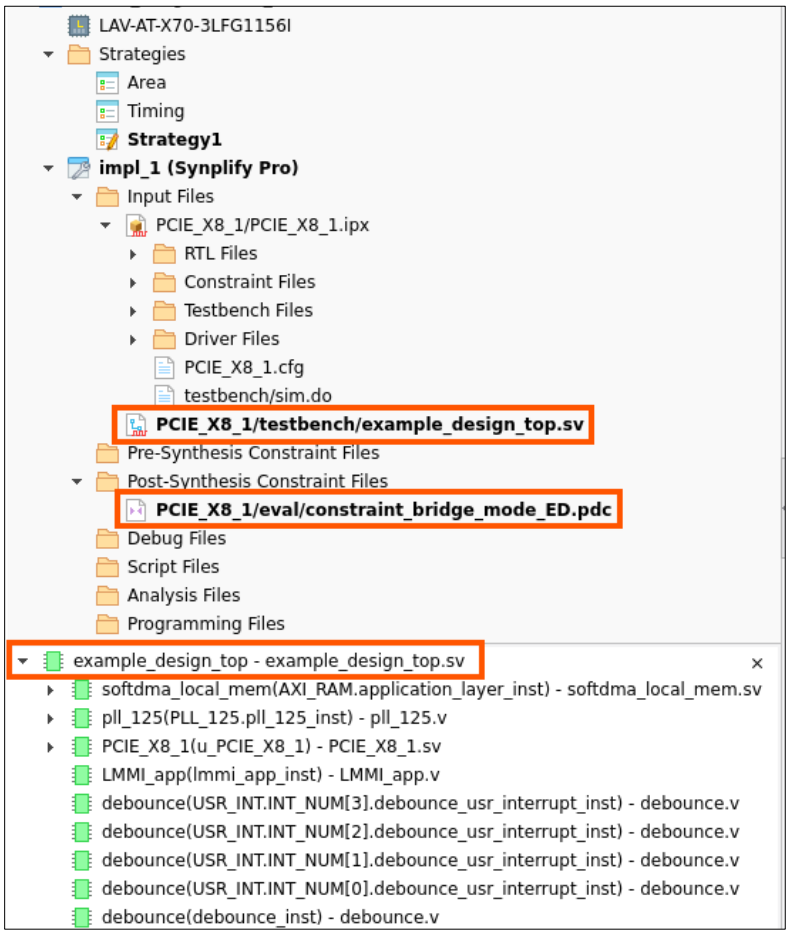


Figure 6.8. File List View of the Created DMA Example Design

4. Right-click on **Post-Synthesis Constraint Files**.
5. Add **<Component Name>/eval/ constraint_bridge_mode_ED.pdc**.
6. Proceed to the Radiant flow if the hierarchical view shows **example_design_top** as the top module.

6.3.3. Non-DMA Design (TLP Interface)

The PCIe x8 Example Design implements the Non-DMA Design (TLP Interface) with the following components:

- PCIe_rx_engine – The received TLPs on the Rx TLP Interface is decoded in this block. For write(posted) operations, the received TLP data is sent to the *pcie_ep_mem* block to store this data into a RAM. For read(non-posted) operations, the received TLP header information is sent to the *pcie_tx_engine* block for the completion TLP.
- PCIe_tx_engine – The transmitted TLPs on Tx TLP Interface managed by this block. This block sends out the completion packets in response to the received non-posted TLP packets to meet the PCIe specification requirements. For example, in case of memory read TLP type packet. The required header information for the completion packet is received from the *pcie_rx_engine*. The data payload is read from the *pcie_ep_mem* block for transmitting along with the completion header.
- PCIe_ep_mem – The *PCIe_ep_mem* module receives the instructions from *rx_engine*, whether the data is written or read.This module consist of a RAM, which is used to store the received data. The design consists of two bars (BAR 0 and BAR 1) enabled in the PCIe endpoint. Based on the address of the BAR, the RAM writes/reads the data from/to the bar specified.

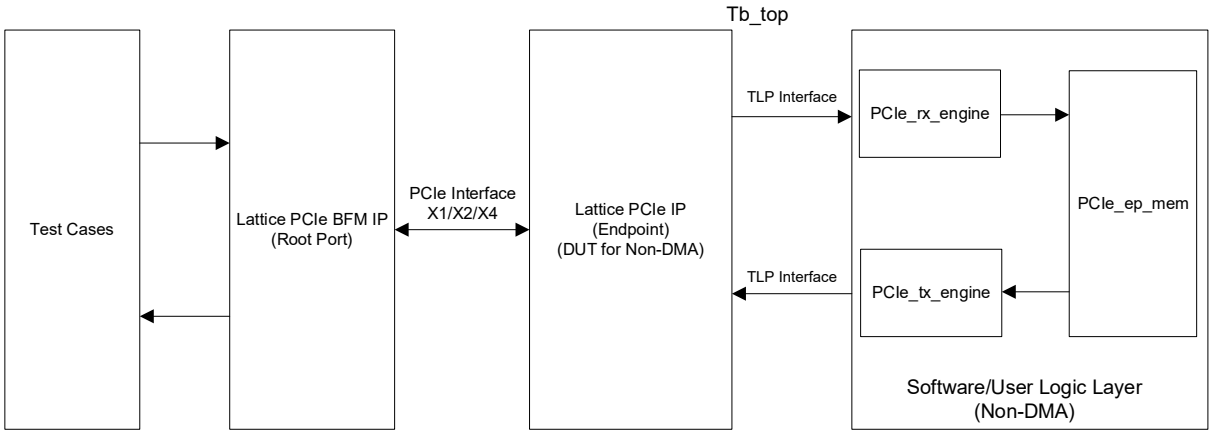


Figure 6.9. Components within Non-DMA Design (TLP Interface)

Additional Lattice IPs are used to enable the components required for the Non-DMA design as specified below:

- Avant OSC Module – Lattice Radiant Software (FPGA-IPUG-02184) – An internal oscillator is implemented to generate a clock to drive the LMMI clock.
- Memory Modules User Guide (FPGA-IPUG-02033) –RAM_DQ is instantiated in the pcie_ep_mem module to store the data and can read or write the data from/to this RAM.

The following shows the Non-DMA data flow:

- Reading the configuration of Lattice PCIe x8 IP
- BFM performs the linkup with PCIe IP.
- Once linkup is done, the BFM sends the header info of the data packet to the PCIe whether to write/read into/from the preferred BAR location of the PCIe.
- PCIe sends the packet information to application layer, which the *pcie_rx_engine* decodes the header data and performs read/write operation accordingly.
- For write operations, the data is written into RAM which stores the data received.
- For read operations, the data is read from RAM and sent to the PCIe along with the header information of the packet.

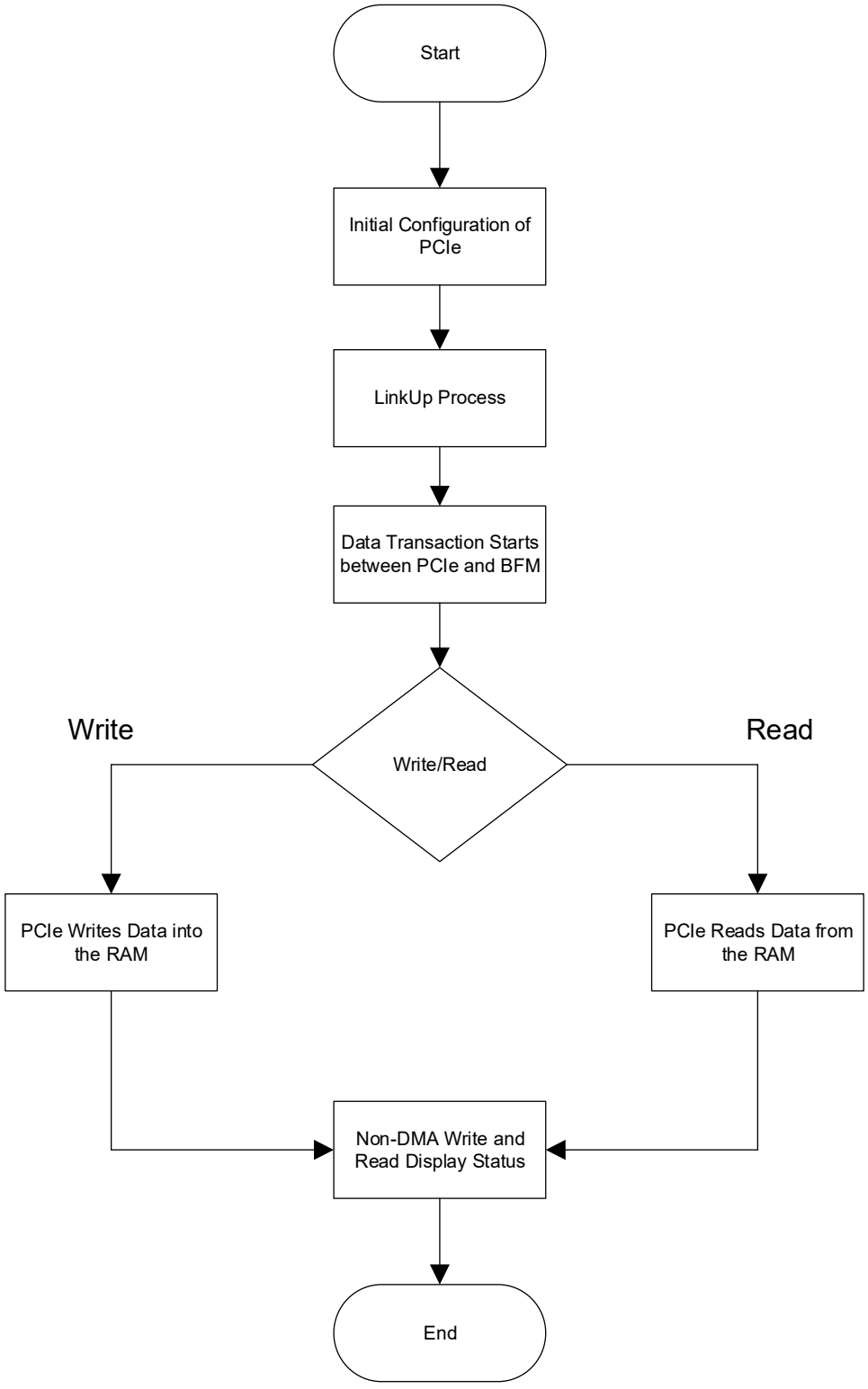


Figure 6.10. Non-DMA Design Data Flow

6.3.3.1. Generating the Non-DMA Example Design

To generate the Non-DMA example design:

1. Create a Lattice Radiant software project. Double-click the **PCIE_X8** in the **IP Catalog** and generate the IP with your desired choice of PCIe generation support, bifurcation and ensure that the **Configuration Mode** being chosen is **TLP Mode**. Some screenshots are provided below to guide you through the IP generation process.

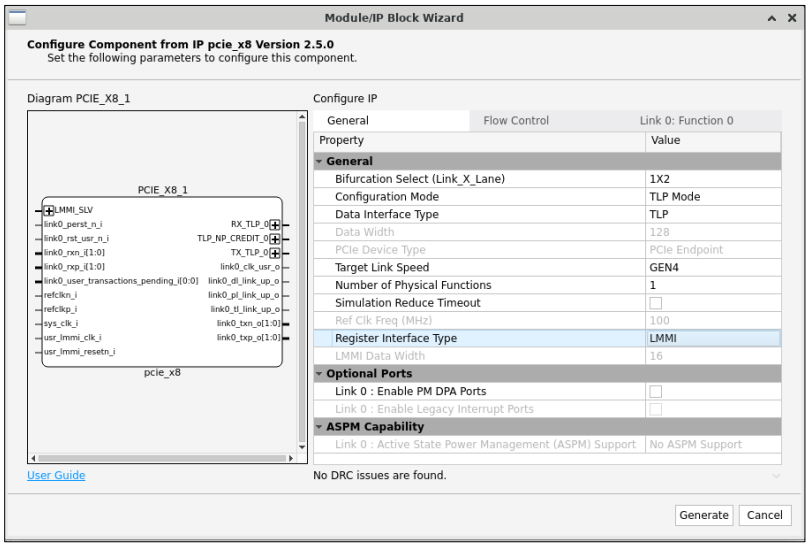


Figure 6.11. Non-DMA Example Design (TLP Mode) Settings (General Tab)

a. BAR 0 must be enabled here and the other BAR's must be disabled.

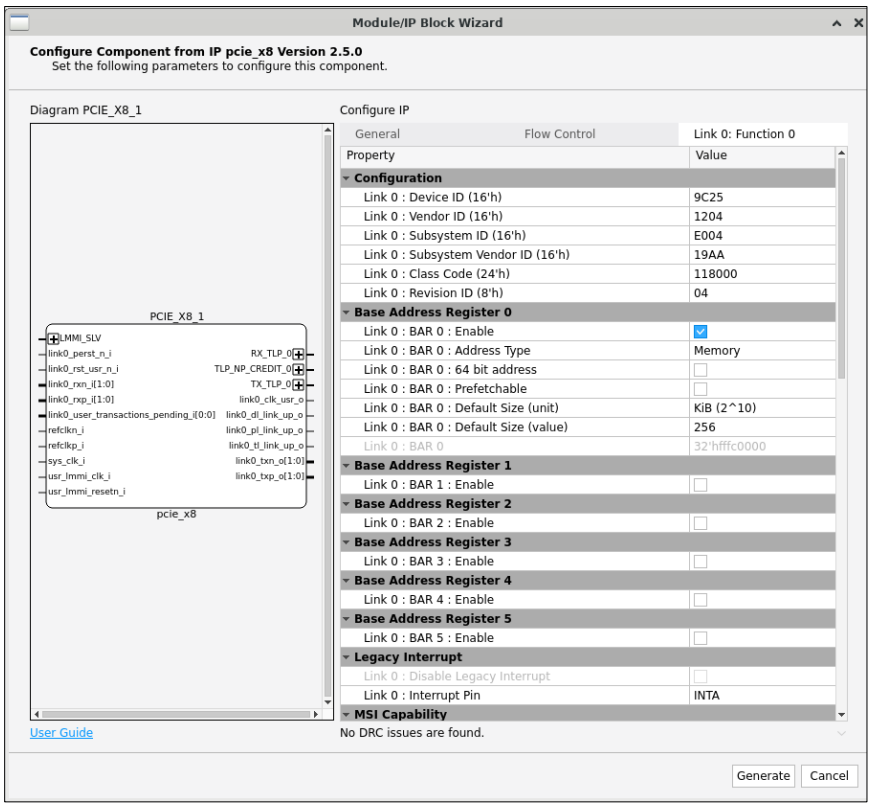


Figure 6.12. Non-DMA Example Design (TLP Mode) Settings (Link 0: Function 0 Tab)

b. The rest of the settings can be left as default.

2. Right-click on **Input Files** and select **Add > Existing Files**.
3. Add **<Component Name>/testbench/NON_DMA/example_design_tlp_top.sv**.
4. Based on the selected bifurcation, adjust the WIDTH parameter accordingly.

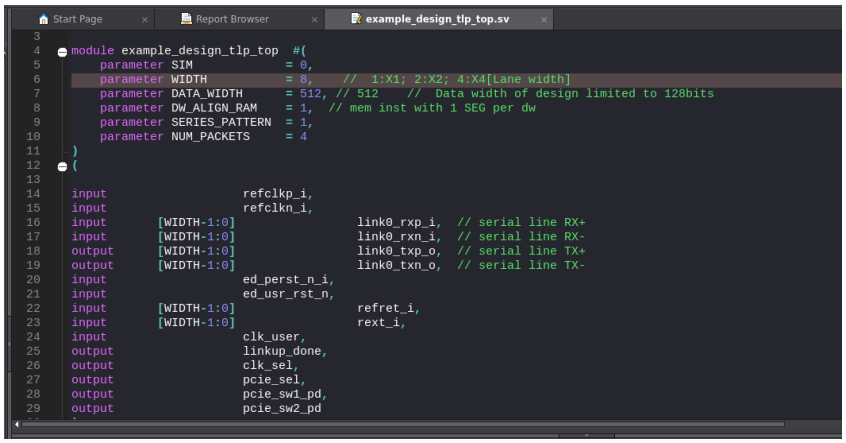


Figure 6.13. WIDTH Parameter Adjustment in the example_design_top Module According to Bifurcation Selection

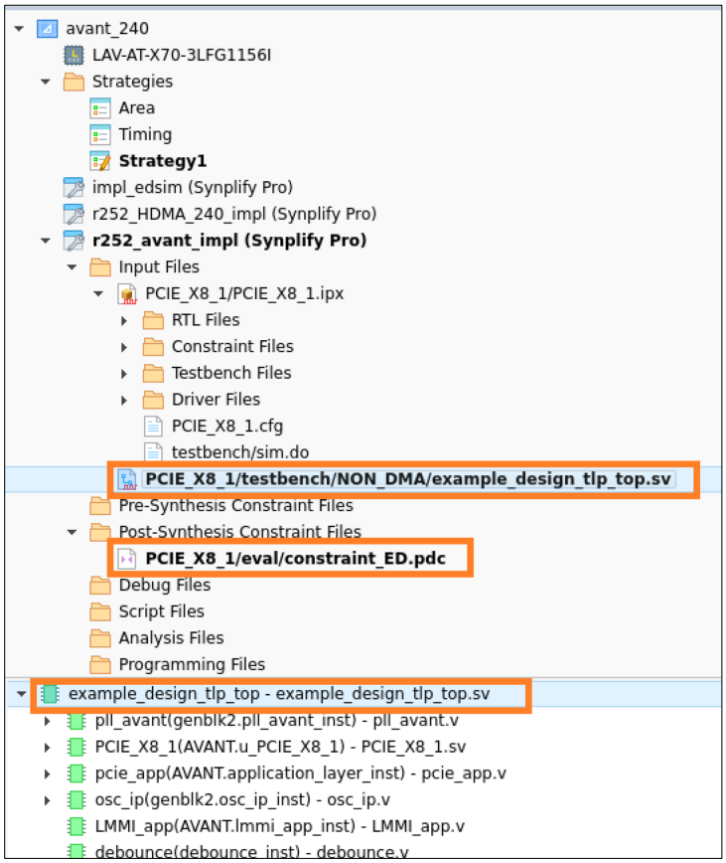


Figure 6.14. File List View of the Created Non-DMA Example Design

7. Right-click on **Post-Synthesis Constraint Files**.
8. Add <Component Name>/eval/constraint_ED.pdc.
9. Proceed to the Radiant flow if the hierarchical view shows **example_design_tlp_top** as the top module.

6.3.4. Non-DMA Design (Bridge Mode)

The PCIe x8 Example Design implements the Non-DMA Design (Bridge Mode) with the following components:

- Write Adapter – Convert AXI Write to the pmi_fifo write interface.
- Read Adapter – Convert AXI Read to the pmi_fifo read interface.
- RAM_DP – pmi_fifo that contains EBR-based RAM.

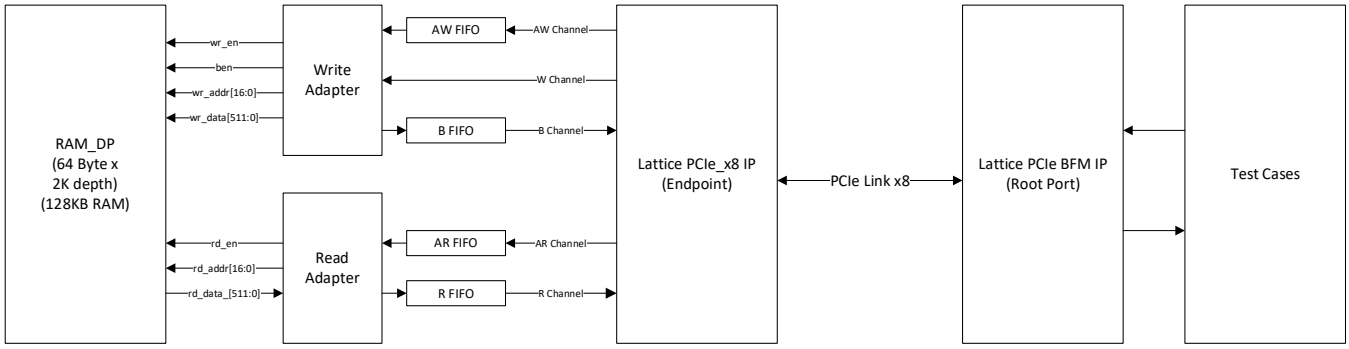


Figure 6.15. Components within NON-DMA Design (Bridge Mode)

The following shows the Non-DMA (Bridge Mode) data flow:

- Reading the configuration of Lattice PCIe x8 IP
- BFM performs the linkup with PCIe IP.
- Once linkup is done, the BFM sends the header info of the data packet to the PCIe whether to write/read into/from the BAR1 location of the PCIe.
- PCIe sends the packet information to application layer via AXI Write or AXI Read Channels, which the *Write Adapter*/Read Adapter decodes AXI Write/Read Channel and performs write/read operation accordingly.
- For write operations, the data is written into RAM which stores the data received.
- For read operations, the data is read from RAM and sent to the IP.

6.3.4.1. Generating the Bridge Mode Example Design

To generate the Bridge Mode example design:

1. Create a Lattice Radiant software project. Double-click the **PCIE_X8** in the **IP Catalog** and generate the IP by selecting **Bridge Mode** in **Configuration Mode** drop-down menu.
2. Right-click on Input Files and select **Add > Existing Files**.
3. Add **<Component Name>/testbench/example_design_top.sv**.

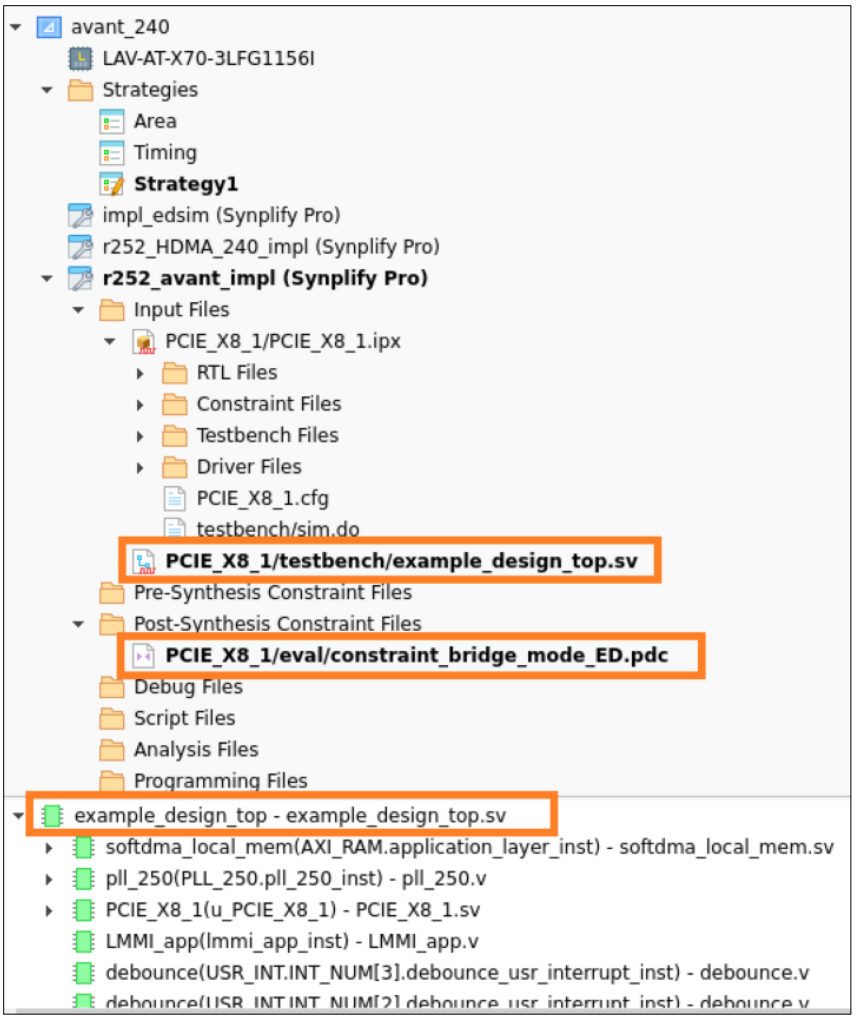


Figure 6.16. File List View of the Created Bridge Mode Example Design

4. Right-click on **Post-Synthesis Constraint Files**.
5. Add **<Component Name>/eval/constraint_bridge_mode_ED.pdc**.
6. Proceed to the Radiant flow if the hierarchical view shows **example_design_top** as the top module.

6.3.5. PDC Settings for Hardware Example Design

When using **SPI Flash programming** on the *Avant-X Versa* board, the following system configuration constraint must be applied:

ldc_set_sysconfig { CONFIGIO_VOLTAGE_BANK2=3.3 CONFIGIO_VOLTAGE_BANK1=1.8 MASTER_SPI_PORT=DISABLE MCCLK_FREQ=57.1 COMPRESS_CONFIG=ON }

For more details, refer to the Avant-G/X Versa Board – User Guide (FPGA-EB-02063).

6.3.6. Running the Hardware Using the Production Driver

After completing the Radiant integration flow and successfully generating the bitstream, proceed to program the bitstream into the Avant Versa board. For the steps to program the bitstream, you can refer to the programming section of Avant-G/X Versa Board – User Guide (FPGA-EB-02063).

1. For the hardened DMA example design, refer to the document linked in Avant-G/X PCIe Host DMA Driver Software User Guide (FPGA-TN-02405) for the steps on how to set up the driver for the DMA version of the IP in order to interact with the IP and perform required transactions.

2. For non-DMA (TLP Interface and Bridge Mode) example designs, refer to the document linked on Lattice Avant and Nexus PCIe Basic Memory-Mapped Host Driver (Non-DMA) User Guide (FPGA-TN-02387) for steps on how to set up the driver for the NON-DMA version of the in order to interact with the IP and perform required transactions.
3. For the DMA example design, refer to Lattice Avant and Nexus PCIe Host DMA Driver Software User Guide (FPGA-TN-02386) for steps on how to set up the driver for the DMA version of the IP to interact with the IP and perform required transactions.

6.4. Simulating the Example Design

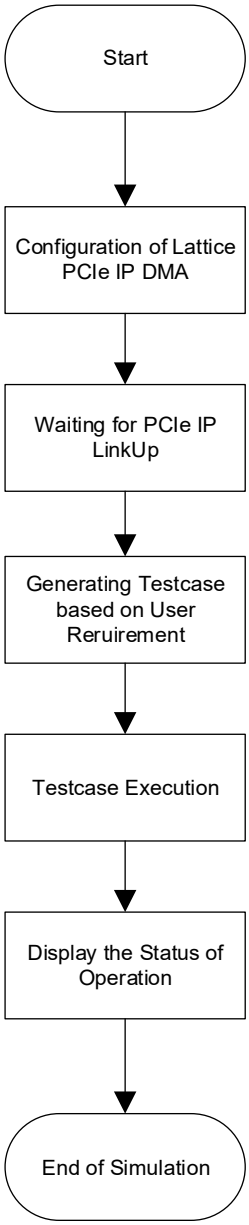


Figure 6.17. PCIe x8 IP Example Design Flowchart

The Example Design can run in simulation as follows:

1. Generate the PCIe x8 IP with the required configuration. Some of the configurations of PCIe can be done through the LMMI. The testbench then waits for the linkup to occur.
2. Enumeration is started and wait for completion.
3. The BFM waits for the PCIe to link up.
4. The BFM starts sending the testcase based on the user requirement.
5. The status of the testcase is displayed as PASS or FAIL.

6.4.1. Running Functional Simulation

Functional Simulation can be performed after the IP is generated through the Example Design testbench. For more details on the Example Design configuration and test cases, refer to the Example Design Supported Configuration section. The limitation of functional simulation is described in the Limitations of the Example Design section.

QuestaSim Lattice Edition simulator is supported in PCIe IP from 2025.2 release.

6.4.1.1. Running the QuestaSim Lattice Edition

To run the QuestaSim Lattice Edition, perform the following:

1. Make sure that the testbench files are generated during PCIe x8 IP generation.

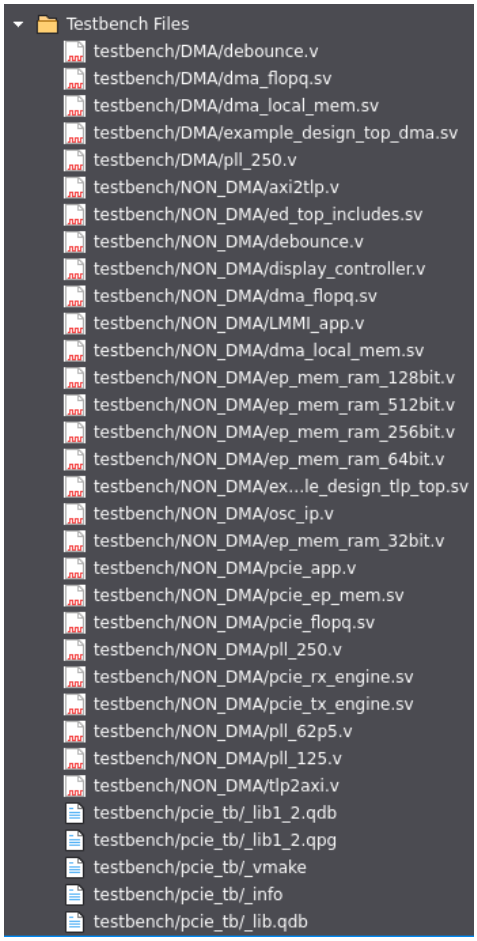


Figure 6.18. Testbench Files

2. Click the icon to initiate the Simulation Wizard and create a new simulation project.
3. Name the project. Click **Next.**

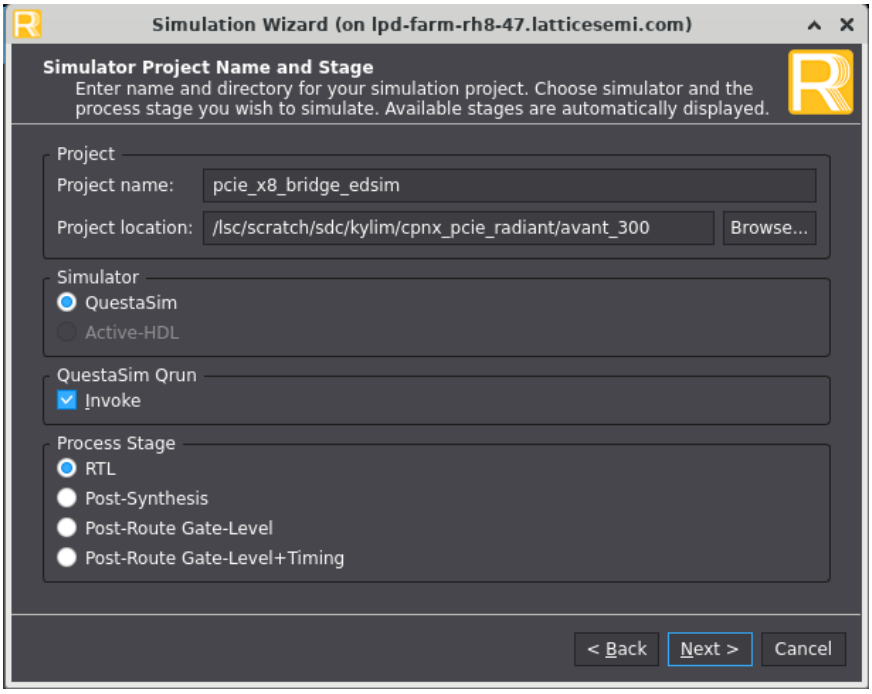


Figure 6.19. Project Naming

4. Select **tb_top** as Simulation Top Module. Click **Next**.

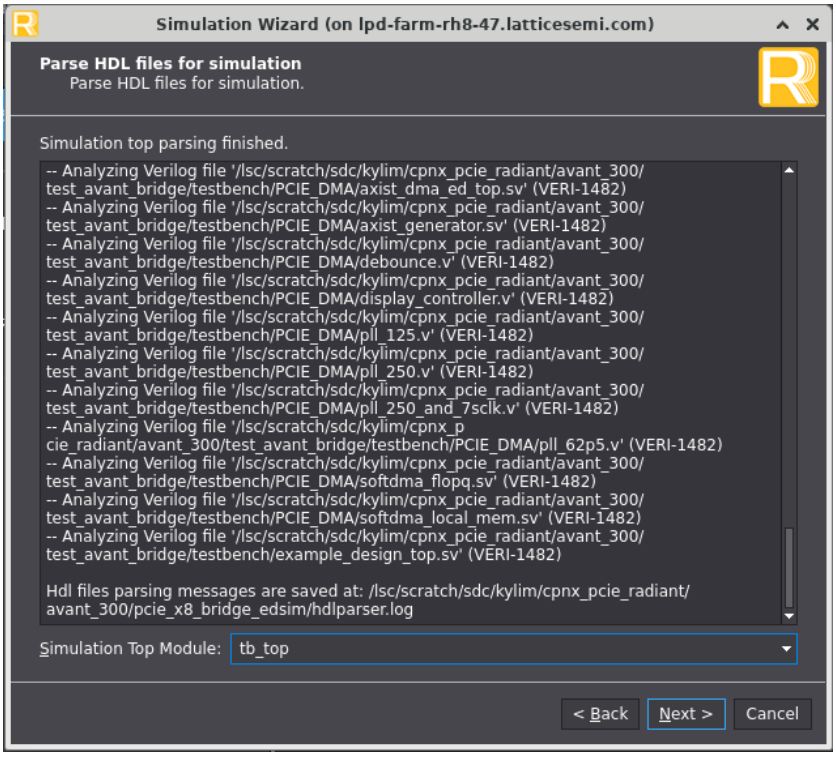


Figure 6.20. Simulation Top Module

5. Use the following simulation settings. Set *Default Run* to **0**. Click **Finish.**

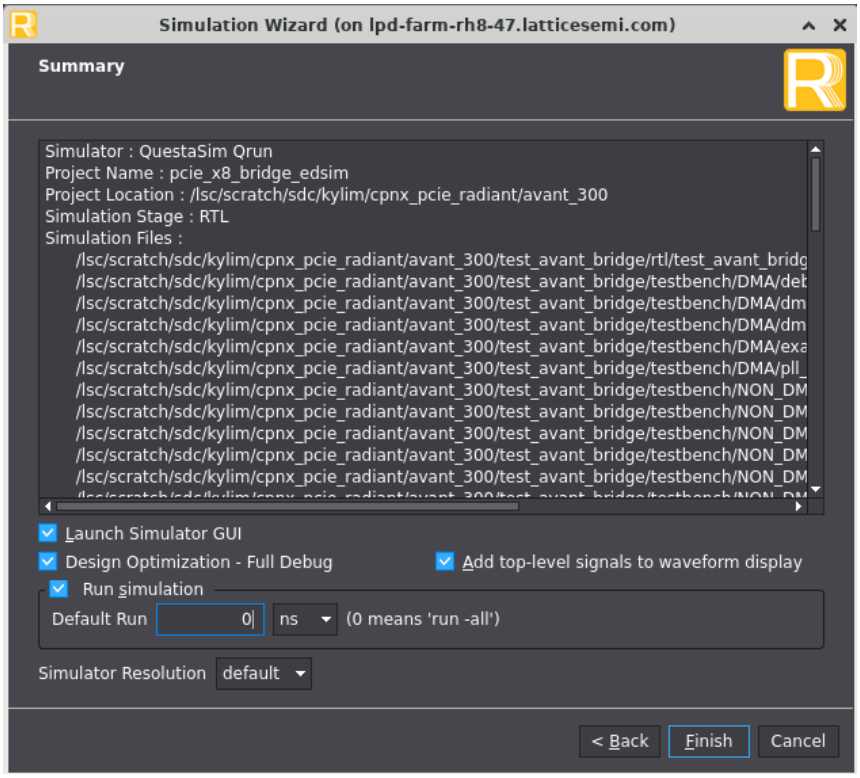


Figure 6.21. Simulation Setting

6. QuestaSim Lattice Edition is launched and once simulation run is completed, *SIMULATION STATUS of ALL TESTS PASS* or *ERRORS DETECTED* is indicated in the transcript window.

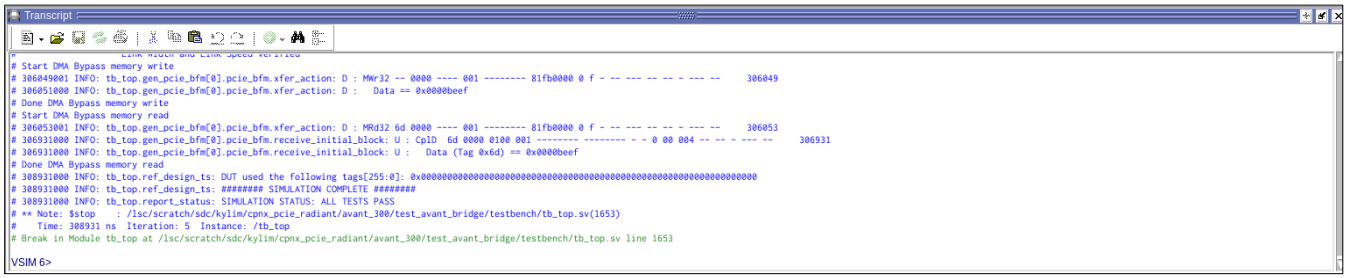


Figure 6.22. Expected Log Printing

6.4.1.2. QuestaSim Pro

To run the QuestaSim Pro, perform the following:

1. Make sure that the testbench files are generated during PCIe x8 IP generation.

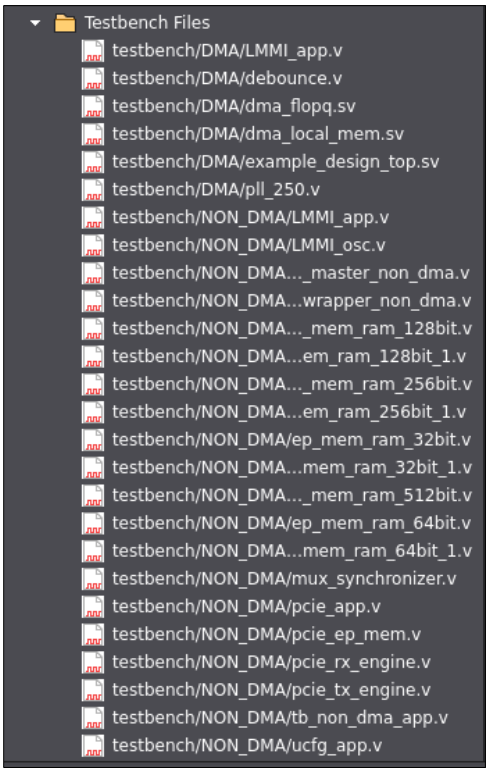


Figure 6.23. Testbench Files

2. Click the  icon to initiate the Simulation Wizard and create a new simulation project.
3. Name the project. Click **Next**. In the *Add and Reorder Source* window, click **Next**.

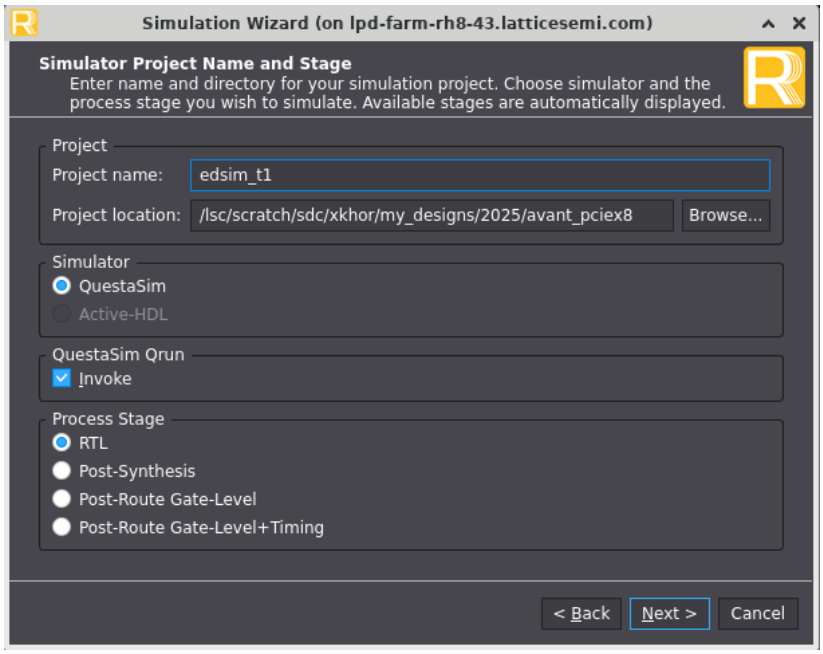


Figure 6.24. Project Naming

4. Select **tb_top** as *Simulation Top Module*.

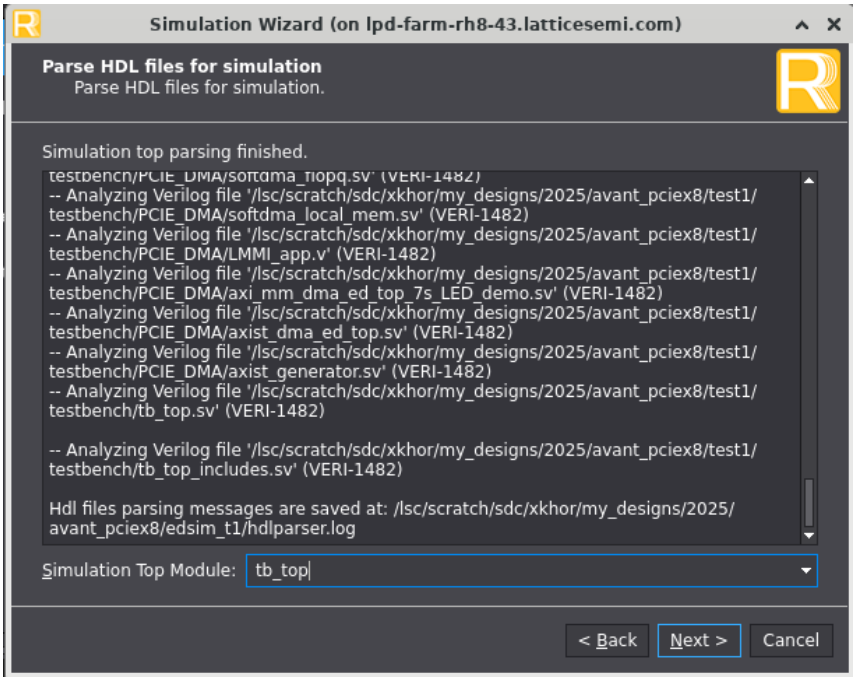


Figure 6.25. Simulation Top Module

5. Use the following simulation settings. Untick *Run simulation* option.

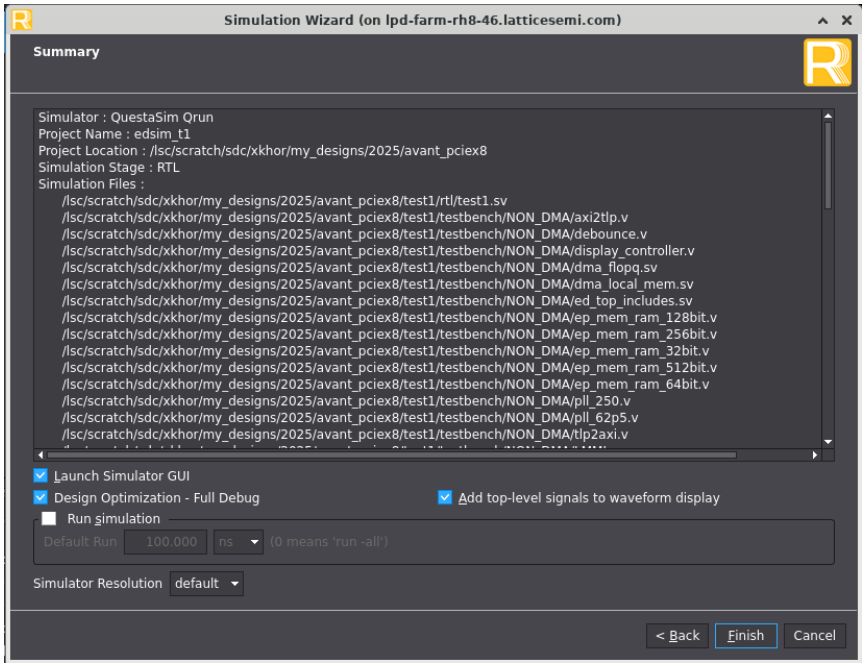


Figure 6.26. Simulation Setting

6. QuestaSim Lattice Edition is launched to perform design compilation. Proceed to close the current QuestaSim window once design compilation is completed.

Figure 6.27. Transcript Log Printing

7. Open the *<project>.f* and then update the *-reflib* to following:

   For Linux:

   ```
   -reflib /home/rel/<Radiant_installation_directory>/cd-
   LIN/diamondng/questasim_tool/cae_library/simulation/libs/pmi_work
   -reflib /home/rel/<Radiant_installation_directory>/cd-
   LIN/diamondng/questasim_tool/cae_library/simulation/libs/ovi_ap6a00c
   ```

   Example for Linux :

   ```
   -reflib /home/rel/ng2025_1p.39/cd-
   LIN/diamondng/questasim_tool/cae_library/simulation/libs/pmi_work
   -reflib /home/rel/ng2025_1p.39/cd-
   LIN/diamondng/questasim_tool/cae_library/simulation/libs/ovi_ap6a00c
   ```

   For Windows:

   ```
   -reflib
   C:/lscc/radiant/<Radiant_installation_directory>/cae_library/simulation/libs/pmi_work
   -reflib
   C:/lscc/radiant/<Radiant_installation_directory>/cae_library/simulation/libs/ovi_ap6a00c
   ```

   Example for Windows :

   ```
   -reflib C:/lscc/radiant/2025.1/cae_library/simulation/libs/pmi_work
   -reflib C:/lscc/radiant/2025.1/cae_library/simulation/libs/ovi_ap6a00c
   ```

8. Update the *<project>.f* file to include the BFM files.

   ```
   "<project_path>/testbench/testbench/pcie_model_x8_4.v"
   "<project_path>/testbench/testbench/pcie_bfm_x8_4.v"
   ```

9. In the *<project>.f* file, append the simulation run command at the end of the line.

   ```
   -do "run -all"
   ```

10. In the *<project>.vdo* file, remove the following line:

    ```
    -do "<project_path>/testbench/sim.do"
    ```

11. Set up the environment variable for FOUNDRY before launching the simulator.

    For Linux:

    a. In the terminal used to launch the simulator, set the environment variable as follows:

    ```
    setenv FOUNDRY <Radiant_installation_directory>/rtf/cae_library
    ```

    Example: setenv FOUNDRY /home/rel/ng2025_1p.39/rtf/cae_library

    For Windows:

    a. Run the Command Prompt as Admin, then set the FOUNDRY as shown in Figure 6.28.

    b. After that, run the QuestaSim Pro.

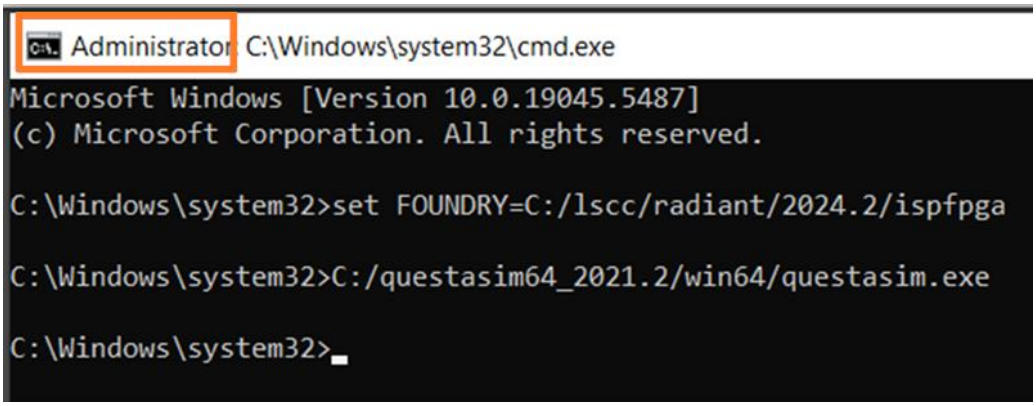


Figure 6.28. System Environment Variable for Windows

12. Launch QuestaSim Pro. Make sure QuestaSim is in the correct project directory and run following qrun command.

Figure 6.29. Command of QuestaSim Pro

13. Once simulation is completed, the log printing below must be observed.

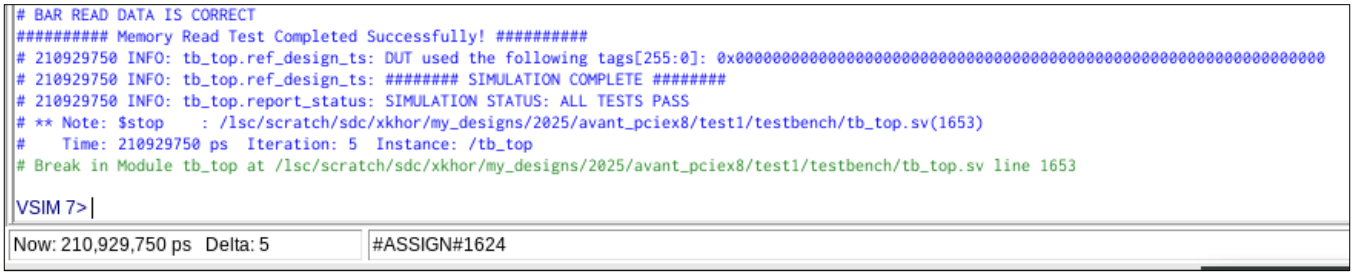


Figure 6.30. Expected Log Printing

14. The simulation run completion is indicated by *SIMULATION STATUS of ALL TESTS PASS* or *ERRORS DETECTED* in the QuestaSim transcript window. Figure 6.31 shows the example of a simulation waveform.

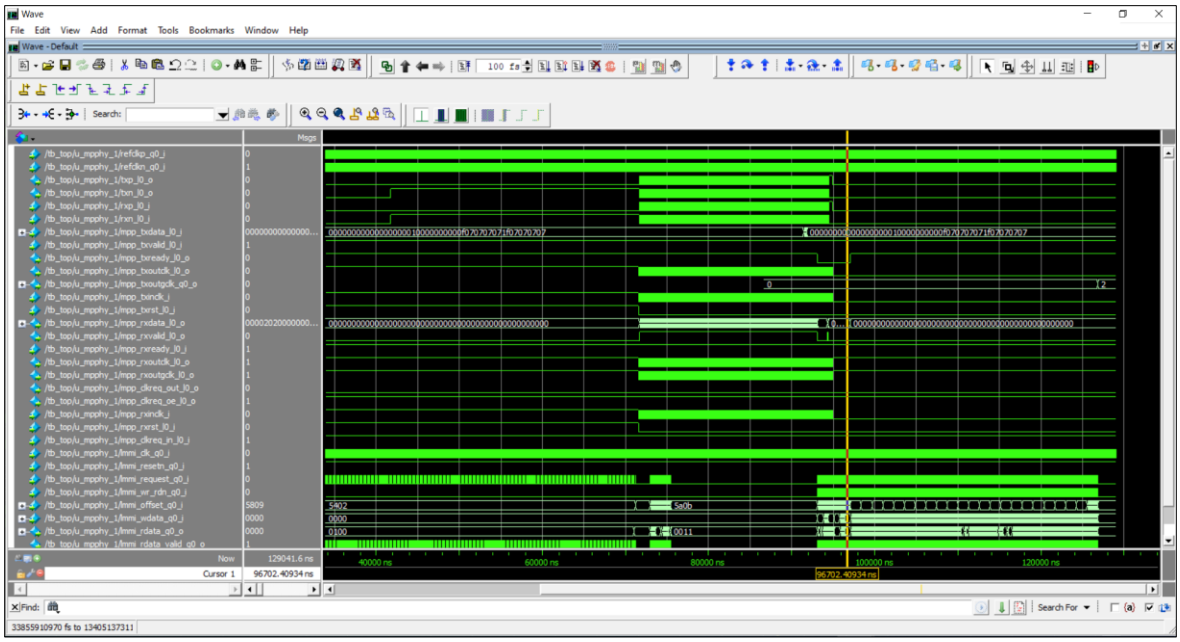


Figure 6.31. Simulation Waveform

6.5. Design Test Case Examples

6.5.1. DMA Design

In the simulation testbench of Hardened DMA, the testcases are implemented using the task included in the ref_design_ts module. In this example design, the host is the Lattice PCIe BFM and the FPGA is the RAM component. In the DMA design, the tasks performed are:

- setup_pcie_axi_hdma_h2f_scatter_gather_queue
  - This task is used to setup the scatter gather queue for the host-to-FPGA (H2F) DMA transaction. For the host-to-FPGA data transfer, the source and destination addresses are configured to the address space of Lattice PCIe BFM and RAM respectively. The DMA data transfer flags for the host-to-FPGA transaction are setup in this task also.
- setup_pcie_axi_hdma_regs
  - This task is used to setup the scatter gather queue registers for the DMA transfer. The source and destination queue pointers, source and destination queue size, source and destination next queue pointer and source and destination queue limit are configured. Finally, the interrupt generation for the respective DMA channel is configured also.
- enable_dma_channel
  - This task is used to enable the DMA channel to start the DMA transfer.
- disable_dma_channel
  - This task is used to disable the DMA channel to stop the DMA transfer.
- setup_pcie_axi_hdma_f2h_scatter_gather_queue
  - This task is used to setup the scatter gather queue for the FPGA-to-host (F2H) DMA transaction. For the FPGA-to-host data transfer, the source and destination addresses are configured to the address space of RAM and Lattice PCIe BFM respectively. The DMA data transfer flags for the FPGA-to-host transaction are setup in this task also.
- reinit_pcie_axi_hdma_regs
  - This task is used to re-initialize the the scatter gather queue registers for the DMA transfer if the scatter gather queue is being setup again. The source and destination next queue pointer and source and destination queue limit are re-configured.
- dma_data_check
  - This task is used to do the DMA data check between the data being transferred through the H2F and F2H DMA transactions.

6.5.2. Non-DMA Design

The testcases are implemented using the task included in the ref_design_ts module.

For the non-DMA design (TLP Interface), the task performed is:

- generic_testcase_for_non_dma
  - This task is used to write data patterns for single transactions and multiple transactions. The task also reads data from PCIe IP and implements a status check on the read data.

Table 6.2 lists the status ports related to the non-DMA design operation.

Table 6.2. Non-DMA Design Status Ports Description

Ports	Description
wr_vector	The array used to generate write data pattern.
mul_wr_vector	The array used to generate write data pattern in case of multiple transactions.
address	The value corresponds to address of the DWORD (32-bit data).
read_data	The read data from the PCIe IP when read test case is implemented.
READ_DATA_CHECK	This signal indicates the status of the read data.

For the Non-DMA (TLP interface) example design, you can change the parameters below in *tb_top* module, under Non-DMA Design parameters section only:

- NON_DMA_TESTCASE_TYPE
  - 3'd0 – SINGLE BAR MEM_WRITE
  - 3'd1 – SINGLE BAR MEM_READ
  - 3'd2 – SINGLE BAR MEM_WRITE AND MEM_READ
  - 3'd3 – TWO BAR CONCURRENT MEM_WRITE AND READ
  - 3'd4 – CONFIGURE WRITE
  - 3'd5 – CONFIGURE READ
  - 3'd6 – Single BAR Multiple Write and Multiple Read transactions to incremental address with incremental data.
  - 3'd7 – Two BAR Multiple Write and Multiple Read transactions to incremental address with incremental data.
  - 3'd8 – Single BAR Multiple Write and Multiple Read transactions to same address with similar data.
  - 3'd9 – Two BAR Multiple Write and Multiple Read transactions to same address
- NON_DMA_SINGLE_BAR_SEL – This is selected only when single BAR operations are performed.
  - 1'b0 – BAR0
  - 1'b1 – BAR1
- NON_DMA_SERIES_PATTERN:
  - 1'b1 – Incremental pattern
  - 1'b0 – Fixed pattern
- NON_DMA_FIXED_DATA – You can give a fixed 32-bit data when fixed pattern is selected.
- NON_DMA_NUM_DWORD – Number of DWORDS (32-bit data) to be written in each packet.
- NON_DMA_NUM_PACKETS – Number of packets used in data transaction.
- NUM_WRITES – Number of write transactions in multiple transactions.
- NUM_READS – Number of read transactions in multiple transactions.

Note: The number of bytes in a packet in a Non-DMA design must be less than the Maximum Payload Size. Otherwise, the maximum payload size is taken as the number of bytes in a packet.

For the non-DMA design (Bridge Mode), the task performed is:

- Trigger Memory Write with 1DW data (32'h0000BEEF).
- Trigger Memory Read to the same address.
- The task expects the read data is 32'h0000BEEF.

The list of ports related to the non-DMA design (Bridge Mode) operation is available in the AXI Data Interface (Bridge Mode) section.

6.6. Debugging Example Design Issues

6.6.1. Signals to Debug

6.6.1.1. Simulation Debug for DMA Design

Table 6.3. DMA Signals to Debug Description

Module Name	Signal Name	Description
tb_top	link0_pl_link_up_o	PCIe IP Physical Layer linkup
tb_top	link0_dl_link_up_o	PCIe IP Data Link Layer linkup
tb_top	link0_tl_link_up_o	PCIe IP Transaction Layer linkup
Host-to-FPGA		
tb_top	m0_dma_axi_awaddr_o	Write address. The write address gives the address of the first transfer in a write burst transaction.
tb_top	m0_dma_axi_awlen_o	Burst length. The burst length gives the exact number of transfers (beat) in a burst. This information determines the number of data transfers associated with the address. 0x00 – 1 beat 0x01 – 2 beats … 0xFF – 256 beats
tb_top	m0_dma_axi_awvalid_o	Write address valid. This signal indicates that the channel is signaling valid write address and control information.
tb_top	m0_dma_axi_awready_i	Write address ready. This signal indicates that the subordinate is ready to accept an Address and associated control signals.
tb_top	m0_dma_axi_wdata_o	Write data.
tb_top	m0_dma_axi_wstrb_o	Write strobes. This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
tb_top	m0_dma_axi_wlast_o	Write last. This signal indicates the last transfer in a write burst.
tb_top	m0_dma_axi_wready_i	Write ready. This signal indicates that the subordinate can accept the write data.
tb_top	m0_dma_axi_bresp_i	Write response. This signal indicates the status of the write transaction.
tb_top	m0_dma_axi_bvalid_i	Write response valid. This signal indicates that the channel is signaling a valid write response.
tb_top	m0_dma_axi_bready_o	Response ready. This signal indicates that the manager can accept a write response.
FPGA-to-Host		
tb_top	m0_dma_axi_araddr_o	Read address. The read address gives the address of the first transfer in a read burst transaction.
tb_top	m0_dma_axi_arlen_o	Burst length. The burst length gives the exact number of transfers (beat) in a burst. This information determines the number of data transfers associated with the address. 0x00 – 1 beat 0x01 – 2 beats … 0xFF – 256 beats
tb_top	m0_dma_axi_arvalid_o	Read address valid. This signal indicates that the channel is signaling valid read address and control information.
tb_top	m0_dma_axi_arready_i	Read address ready. This signal indicates that the subordinate is ready to accept an Address and associated control signals.
tb_top	m0_dma_axi_rdata_i	Read data.

Module Name	Signal Name	Description
tb_top	m0_dma_axi_rresp_i	Read response. This signal indicates the status of the read transfer.
tb_top	m0_dma_axi_rlast_i	Read last. This signal indicates the last transfer in a read burst.
tb_top	m0_dma_axi_rvalid_i	Read valid. This signal indicates that the channel is signaling the required read data.
tb_top	m0_dma_axi_rready_o	Read ready. This signal indicates that the manager can accept the read data and response information.

6.6.1.2. Simulation Debug for Non-DMA (Bridge Mode) Design

Table 6.4. AXI-Lite Bridge Mode to Debug Description

Module Name	Signal Name	Description
tb_top	link0_pl_link_up_o	PCIe IP Physical Layer linkup
tb_top	link0_dl_link_up_o	PCIe IP Data Link Layer linkup
tb_top	link0_tl_link_up_o	PCIe IP Transaction Layer linkup
AXI-Lite		
tb_top	m0_axil_awaddr_o	Write address. The write address gives the address in a write transaction.
tb_top	m0_axil_awvalid_o	Write address valid. This signal indicates that the channel is signaling valid write address.
tb_top	m0_axil_awready_i	Write address ready. This signal indicates that the subordinate is ready to accept an Address.
tb_top	m0_axil_wdata_o	Write data.
tb_top	m0_axil_wstrb_o	Write strobes. This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
tb_top	m0_axil_wready_i	Write ready. This signal indicates that the subordinate can accept the write data.
tb_top	m0_axil_bresp_i	Write response. This signal indicates the status of the write transaction.
tb_top	m0_axil_bvalid_i	Write response valid. This signal indicates that the channel is signaling a valid write response.
tb_top	m0_axil_bready_o	Response ready. This signal indicates that the manager can accept a write response.
tb_top	m0_axil_araddr_o	Read address. The read address gives the address in a read transaction.
tb_top	m0_axil_arvalid_o	Read address valid. This signal indicates that the channel is signaling valid read address.
tb_top	m0_axil_arready_i	Read address ready. This signal indicates that the subordinate is ready to accept an Address.
tb_top	m0_axil_rdata_i	Read data.
tb_top	m0_axil_rresp_i	Read response. This signal indicates the status of the read transfer.
tb_top	m0_axil_rvalid_i	Read valid. This signal indicates that the channel is signaling the required read data.
tb_top	m0_axil_rready_o	Read ready. This signal indicates that the manager can accept the read data and response information.

Module Name	Signal Name	Description
User Interrupt		
tb_top	usr_int_req_i	Request by application logic to trigger interrupt to the Host via the IP.
tb_top	usr_int_ack_o	Acknowledgement by the IP with respect to the request from signal usr_int_req_i.

6.6.1.3. Simulation Debug for Non-DMA (TLP Interface) Design

Table 6.5. Non-DMA Signals to Debug Description

Module Name	Signal Name	Description
tb_top	lmmi_offset	Lower 17-bit address of LMMI interface registers
tb_top	lmmi_wdata	Data written into PCIe IP through LMMI interface
tb_top	lmmi_rdata	Data read from PCIe IP through LMMI interface
tb_top	link0_pl_link_up	PCIe IP physical layer linkup
tb_top	link0_dl_link_up	PCIe IP data layer linkup
tb_top	link0_tl_link_up	PCIe IP transaction layer linkup

The following are the steps to debug the non-DMA design:

- You must check whether the initial configuration is performed properly. You can check the *lmmi_offset*, *lmmi_wdata*, and *lmmi_rdata* signals to verify. Note that in the actual application, the register configuration may not be necessary if the corresponding register is configured through the IP Catalog's Module/IP wizard.
- The linkup signals such as *link0_pl_link_up*, *link0_dl_link_up*, and *link0_tl_link_up* must be asserted.

6.7. Limitations of the Example Design

For non-DMA (TLP) variant, the functional simulation with the example design testbench can only support the following:

- TLP interface for Data, LMMI interface for Config
- Bifurcation 1 × 1, 1 × 2, 1 × 4 and 1 x 8
- MSI-X capability disabled

For Hardened DMA variant, the functional simulation with the example design testbench can only support the following IP configuration:

- AXI-MM interface for Data, LMMI interface for Config
- Bifurcation 1 × 8
- No Virtual Function Supported
- MSI-X capability disabled

7. Designing with the IP

This section provides information on how to generate the IP Core using the Lattice Radiant software and how to run simulation and synthesis. For more details on the Lattice Radiant software, refer to the Lattice Radiant software user guide.

Note: The screenshots provided are for reference only. Details may vary depending on the version of the IP or software being used. If there have been no significant changes to the user interface, a screenshot may reflect an earlier version of the IP.

7.1. Generating and Instantiating the IP

You can use the Lattice Radiant software to generate IP modules and integrate them into the device's architecture. The steps below describe how to generate the PCIe x8 IP in the Lattice Radiant software.

To generate the PCIe x8 IP:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the **IP Catalog** tab, double-click **PCIE_X8** under **IP, Connectivity** category. The **Module/IP Block Wizard** opens as shown in Figure 7.1. Enter values in the **Component name** and the **Create in** fields and click **Next**.

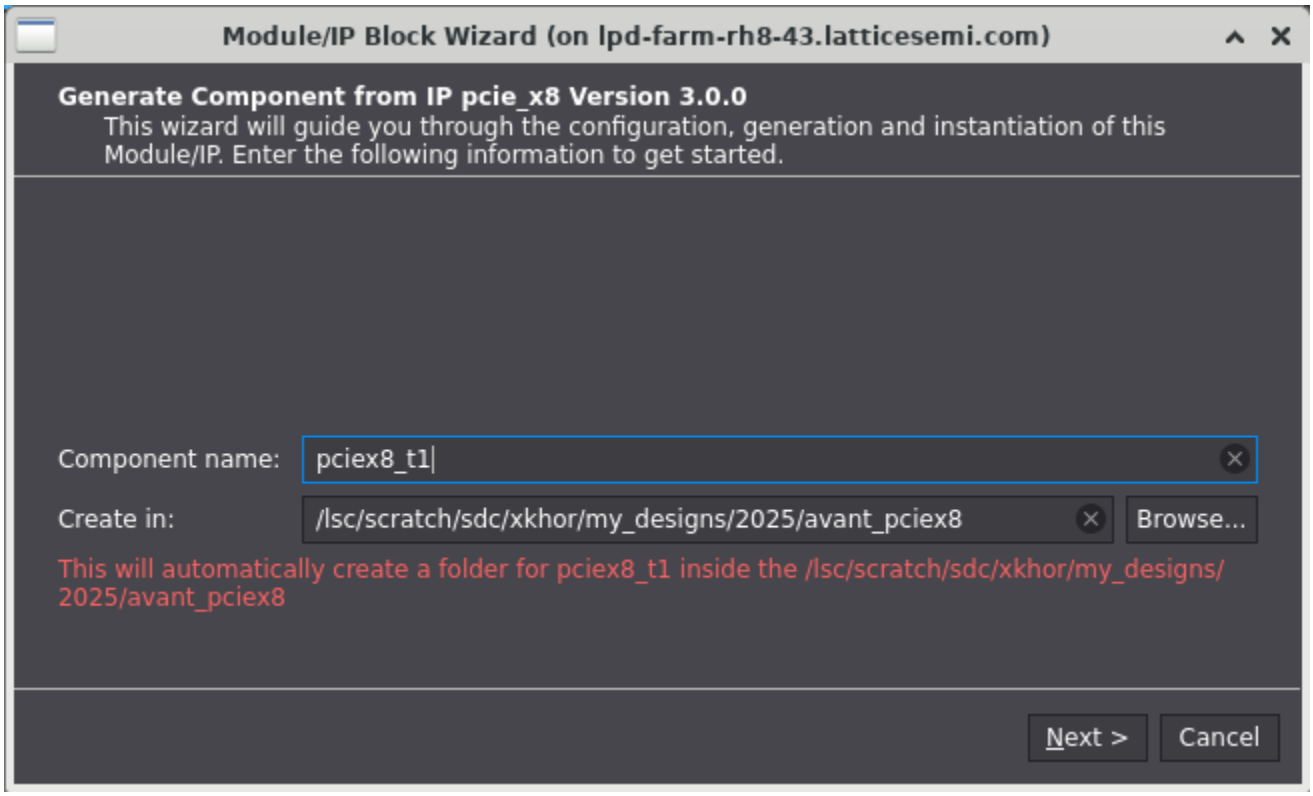


Figure 7.1. Module/IP Block Wizard

3. In the next **Module/IP Block Wizard** window, customize the selected PCIe x8 IP using drop-down lists and check boxes. Figure 7.2 shows an example configuration of the non-DMA PCIe x8 IP. For the hardened DMA example design support, Figure 7.3 to Figure 7.6 show the example configuration of the hardened DMA PCIe x8 IP. For details on the configuration options, refer to the IP Parameter Description section.

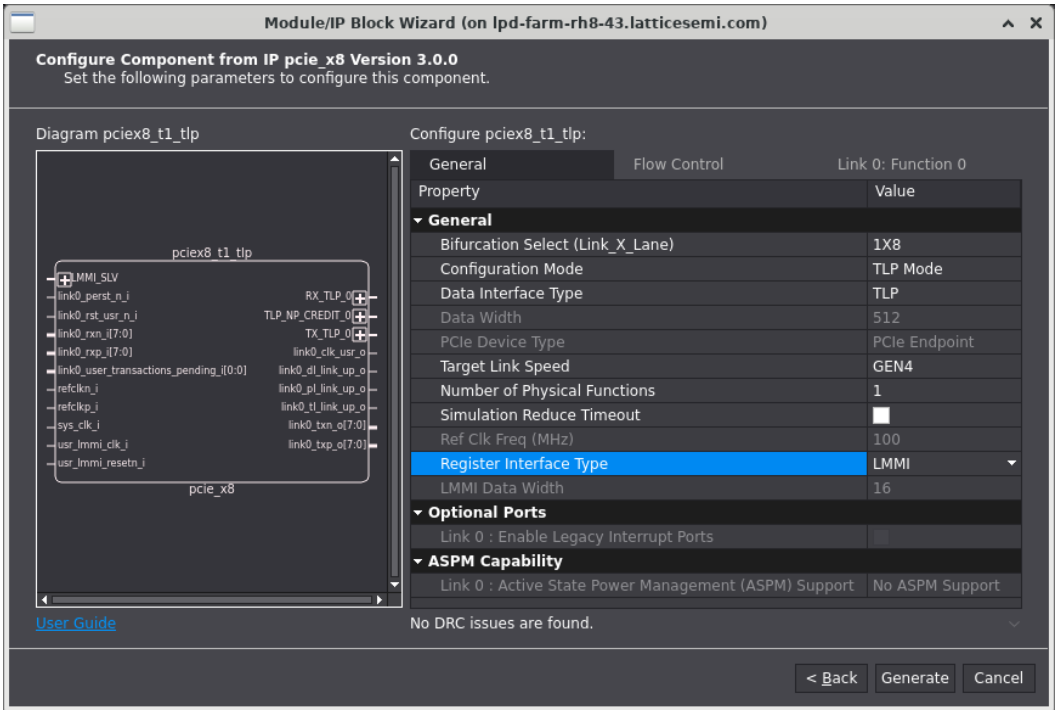


Figure 7.2. IP Configuration for Non-DMA

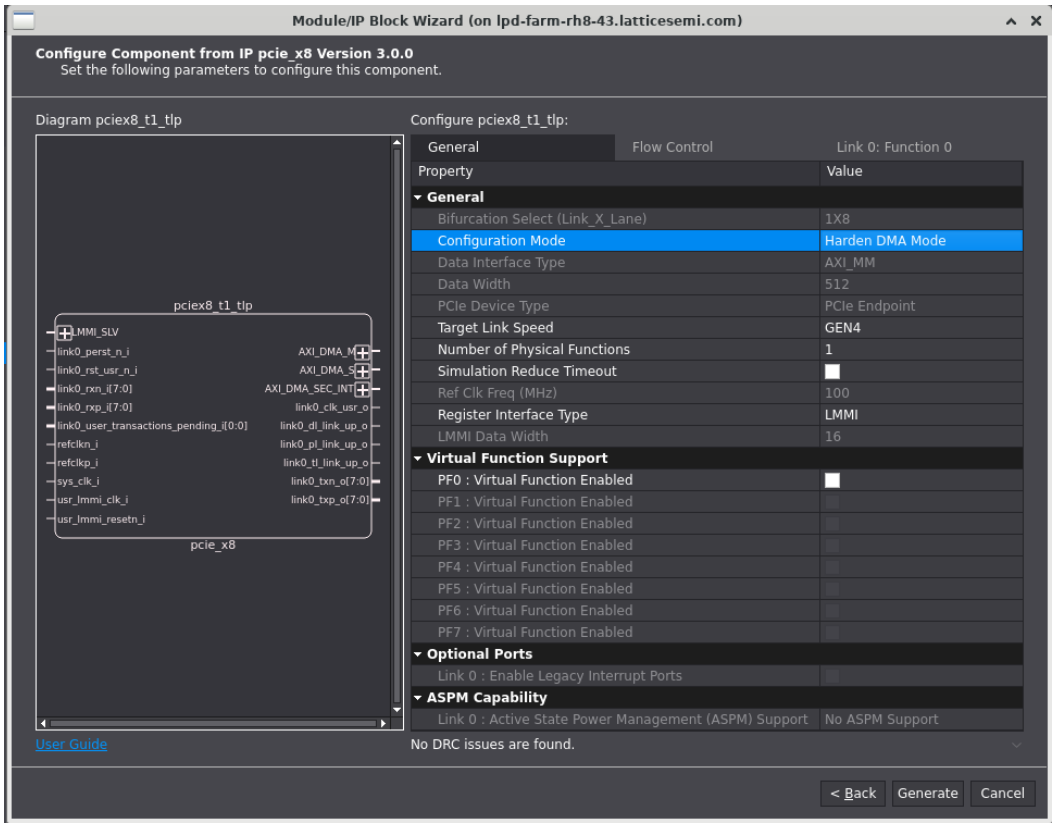


Figure 7.3. IP Configuration for Hardened DMA (General Tab)

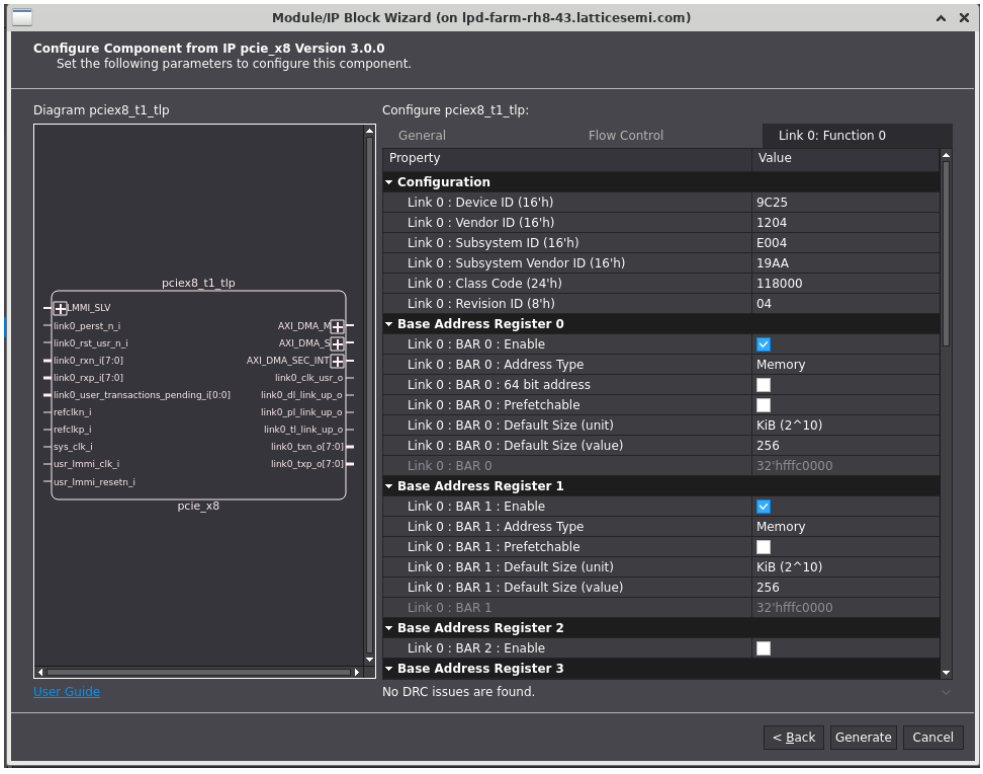


Figure 7.4. IP Configuration for Hardened DMA (Link 0: Function 0 Tab) Part 1

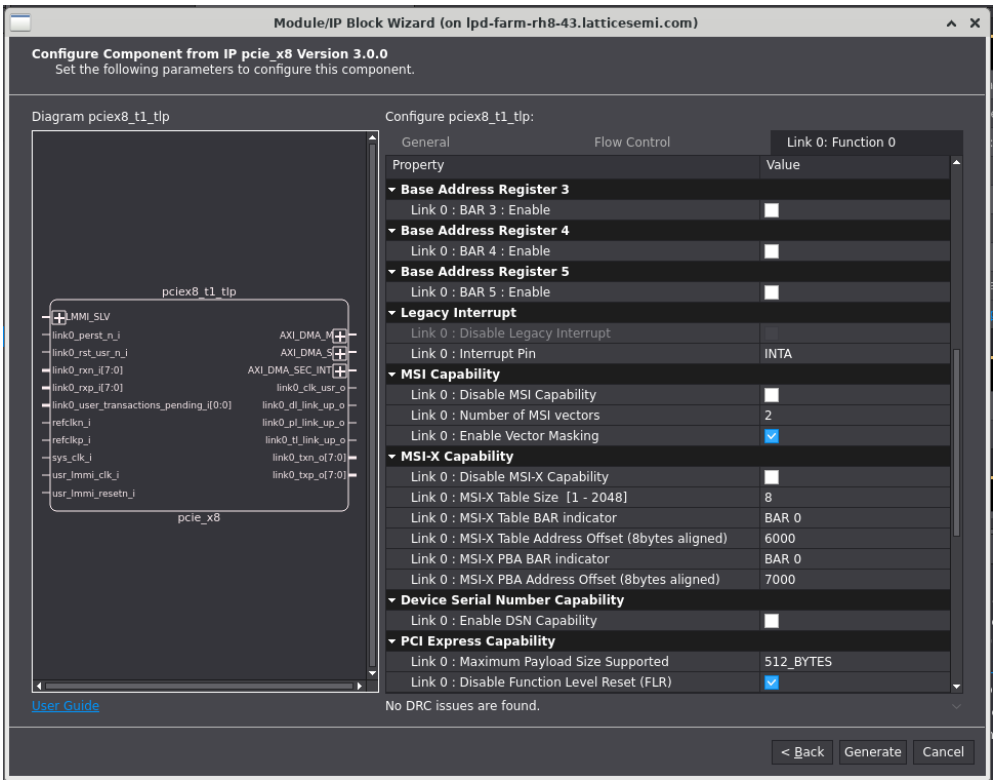


Figure 7.5. IP Configuration for Hardened DMA (Link 0: Function 0 Tab) Part 2

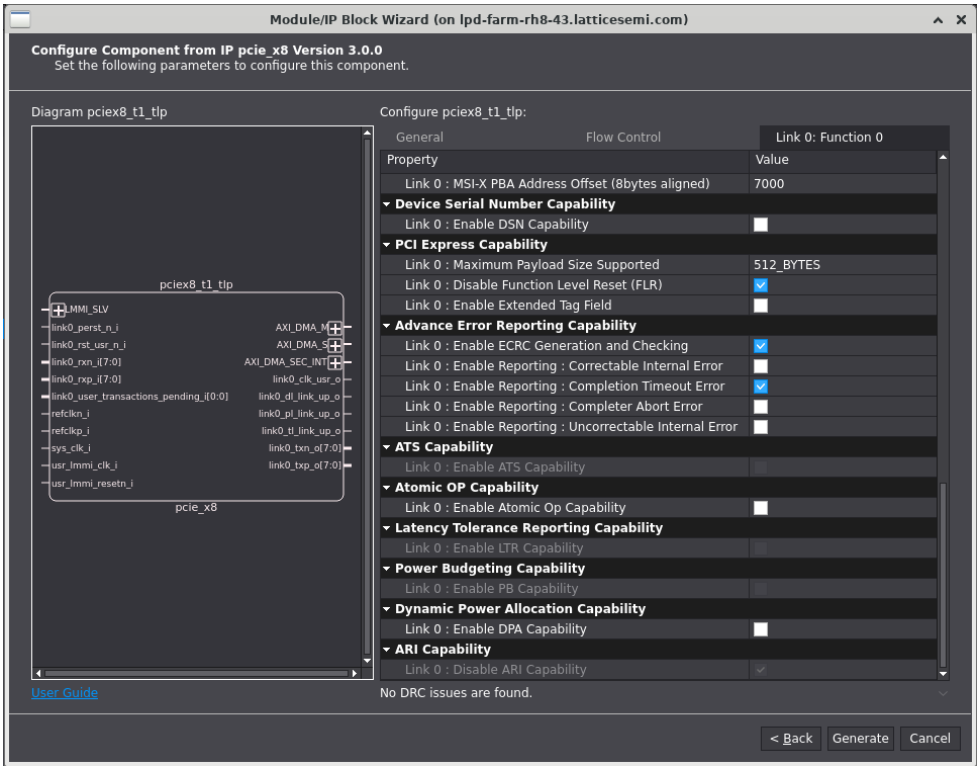


Figure 7.6. IP Configuration for Hardened DMA (Link 0: Function 0 Tab) Part 3

4. Click **Generate**. The **Check Generated Result** dialog box opens, showing design block messages and results as shown in Figure 7.7.

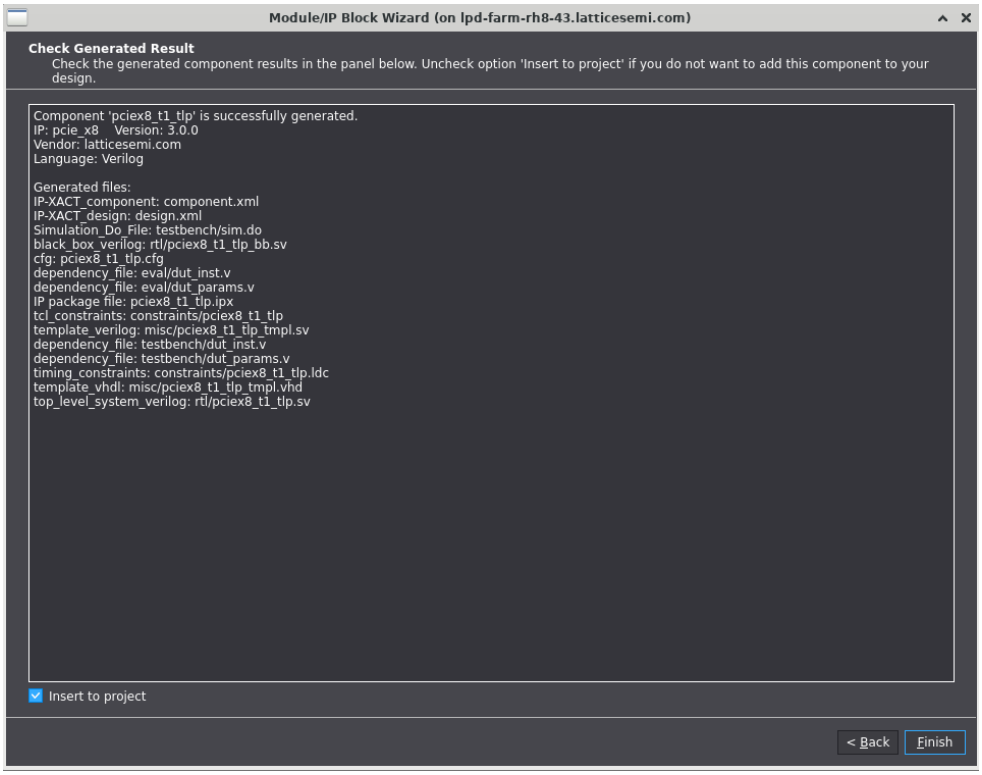


Figure 7.7. Check Generated Result

5. Click **Finish**. All the generated files are placed under the directory paths in the **Create in** and the **Component name** fields shown in Figure 7.1.
6. IP version 2.1.0 or below requires an additional step to set the IP top level. Right-click on the IP name and select *Set as Top-Level Unit* shown in Figure 7.8.

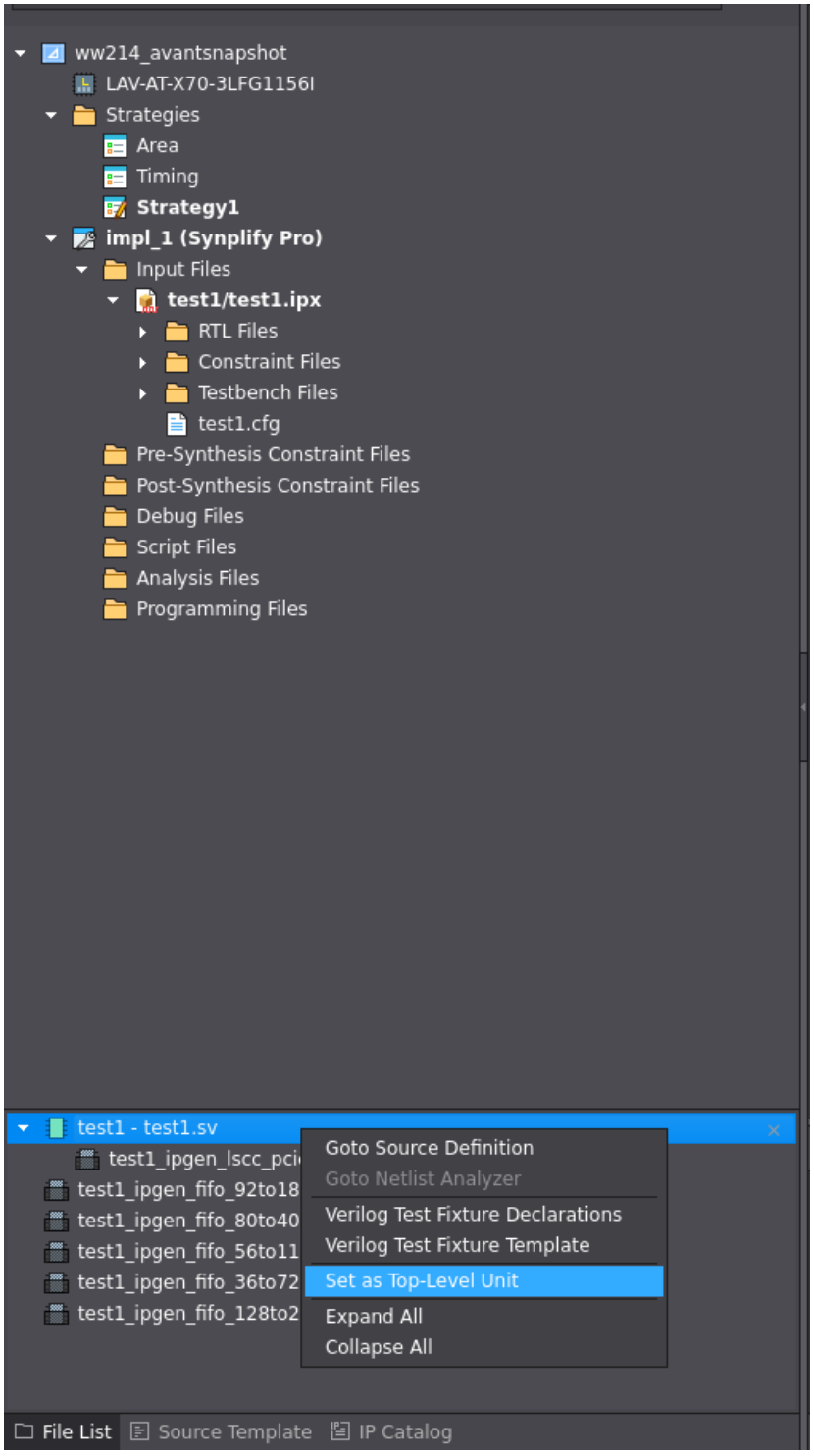


Figure 7.8. Set IP Top Level Unit

7. As this is an IP level compilation, you need to set all *TLP/AXI-MM/AXI-Stream/LMMI/AXI-Lite* signal as Virtual I/O through a constraint file. In the system level design where these I/O are connected properly to the user bus or config bus, these constraints are not required.

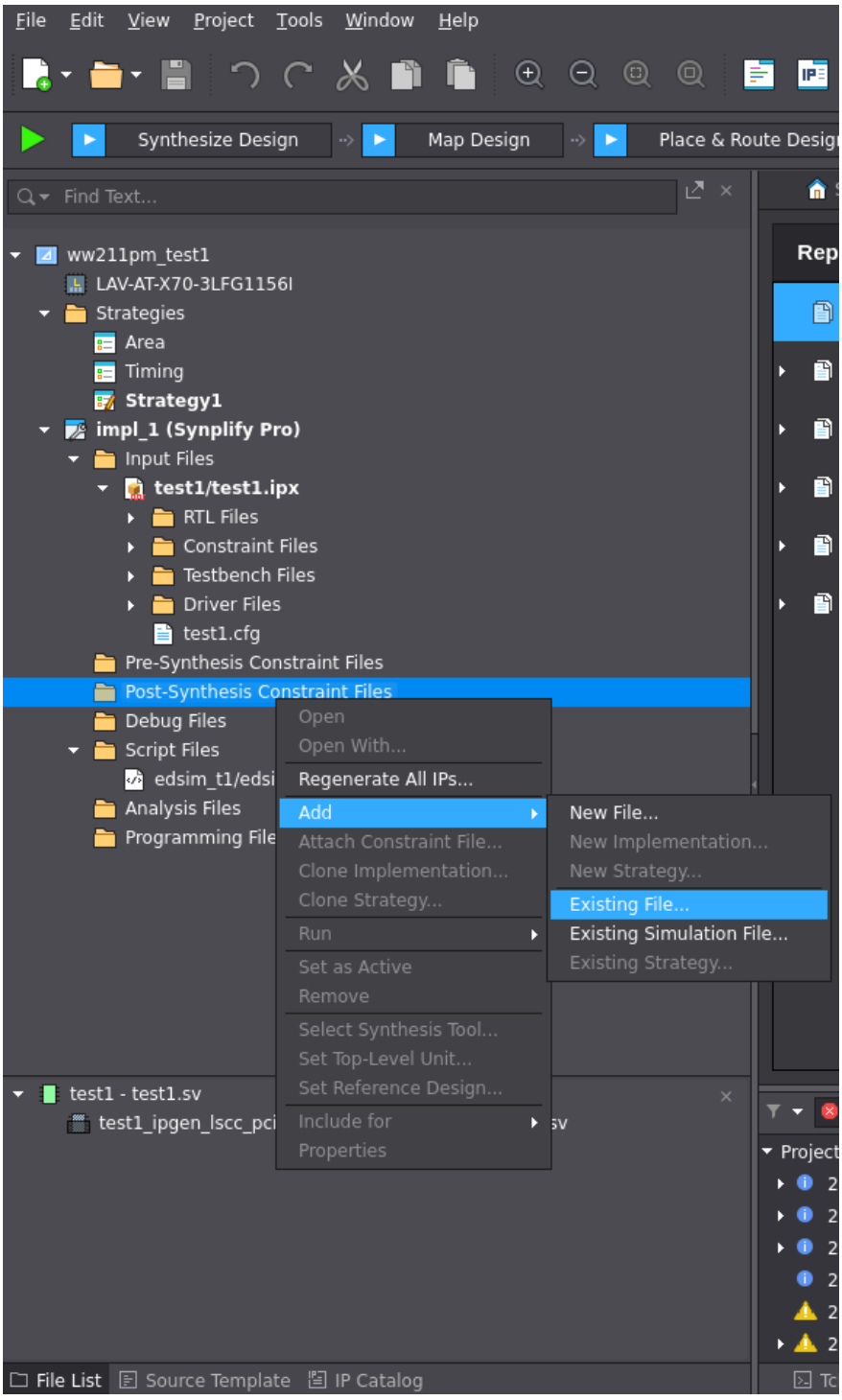


Figure 7.9. Select to Add Constraint File

8. Refer to the settings in Figure 7.10. Browse to *<project_folder>/<project_name>/<ip_name>/eval/* and select **constraint.pdc**.

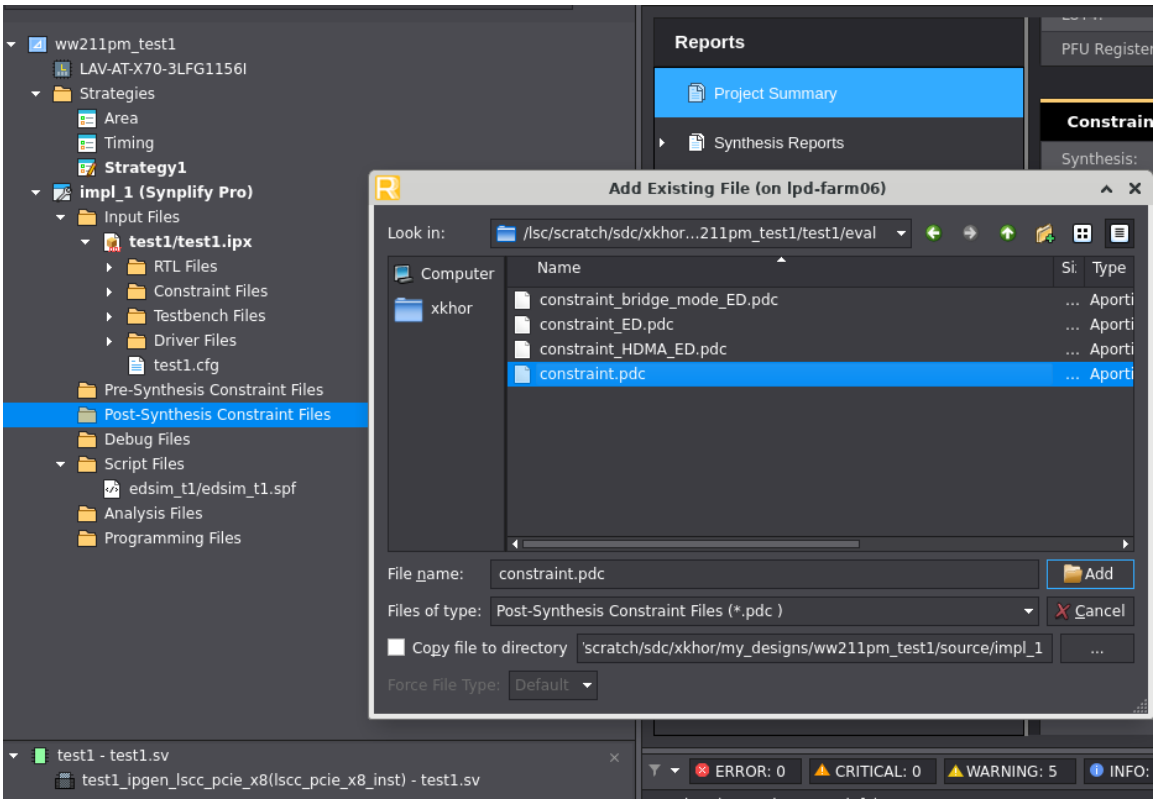


Figure 7.10. Select constraint.pdc from the Eval Folder

9. Click **Run All** to compile the IP. Successful compilation is shown in Figure 7.11.

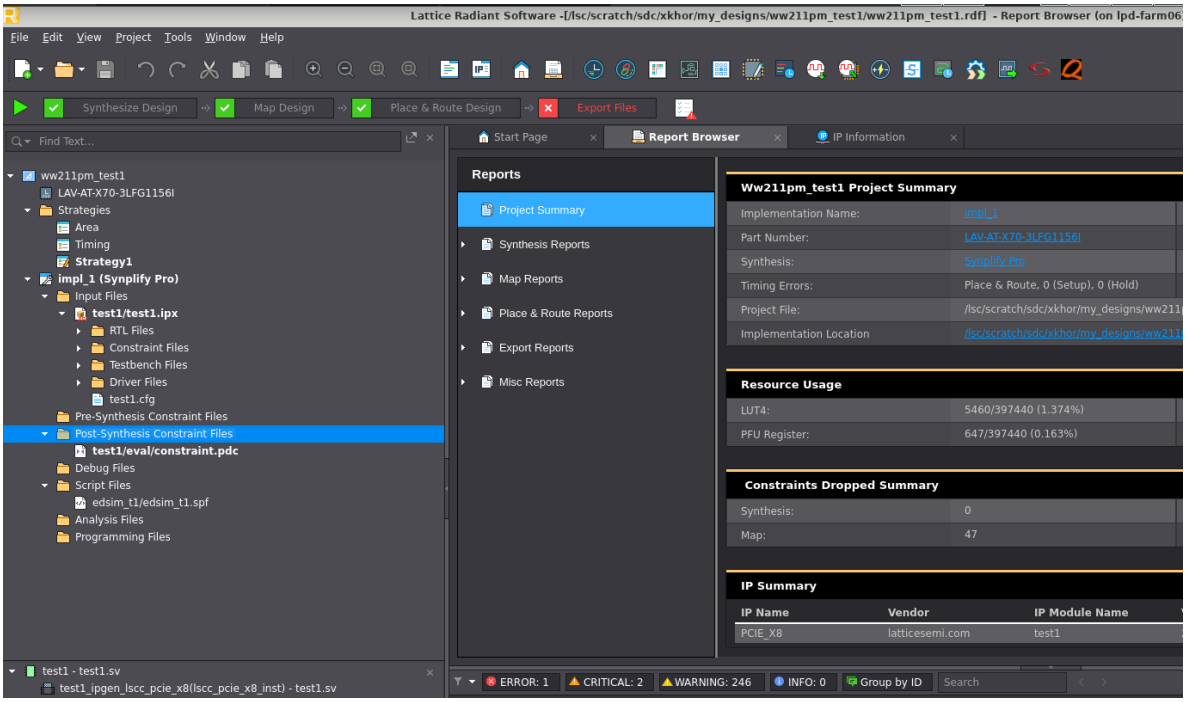


Figure 7.11. Project Compile Done

10. As the IP level compile involves virtual I/O, the bitstream cannot be generated. For bitstream generation, refer to PCIe Reference design on Lattice website.

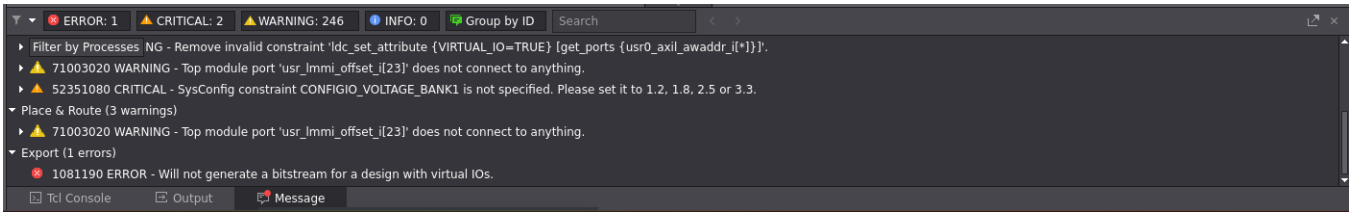


Figure 7.12. Bitstream Not Generated due to the Virtual I/O Setting

7.1.1. Generated Files and File Structure

The generated PCIe x8 module package includes the black box (<Component name>_bb.sv) and instance templates (<Component name>_tmpl.v/vhd) that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file (<Component name>.sv) that can be used as an instantiation template for the module is also provided. You may also use this top-level reference as the starting template for the top-level for the complete design. The generated files are listed in Table 7.1.

Table 7.1. Generated File List

Attribute	Description
<Component name>.ipx	This file contains the information on the files associated to the generated IP.
<Component name>.cfg	This file contains the parameter values used in IP configuration.
component.xml	Contains the ipxact:component information of the IP.
design.xml	Documents the configuration parameters of the IP in IP-XACT 2014 format.
rtl/<Component name>.v	This file provides an example RTL top file that instantiates the module.
rtl/<Component name>_bb.v	This file provides the synthesis black box.
misc/<Component name>_tmpl.v misc/<Component name>_tmpl.vhd	These files provide instance templates for the module.

7.1.2. Design Implementation

Completing the design includes additional steps to specify analog properties, pin assignments, and timing constraints. You can add and edit the constraints using the Device Constraint Editor or by manually creating a PDC File.

7.1.2.1. Device Constraint Editor

Refer to the latest Lattice Radiant User Guide in the Lattice Radiant web page for more information on how to use the device constraint editor.

7.1.2.2. Manual PDC File Creation

To create the manual PDC file, add the .pdc (post synthesis constraint file) file in the Lattice Radiant software and define the I/O pins according to the schematic design for ports defined in your design. You can define different types of constraints such as pins, clocks, and other timing paths. Post-Synthesis constraint files (.pdc) contain both timing and non-timing constraint .pdc source files for storing logical timing/physical constraints.

Refer to the relevant sections in the Lattice Radiant Software User Guide for more information on how to create or edit constraints.

7.1.3. Timing Constraints

The timing constraints are based on the clock frequency used. The timing constraints for the IP are defined in relevant constraint files. The example below shows the IP timing constraints generated for the PCIe x8 IP.

```
create_clock -name {refclkp_i}  -period 10 -waveform {0  5} [get_ports {refclkp_i}]
create_clock -name {refclkn_i}  -period 10 -waveform {0  5} [get_ports {refclkn_i}]
create_clock -name {usr_lmmi_clk_i}    -period  10 -waveform {0  5} [get_ports usr_lmmi_clk_i]

if {$LINK0_FTL_INITIAL_TARGET_LINK_SPEED == "3"} {
    create_clock -name {sys_clk_i}  -period 4 -waveform {0  2} [get_ports sys_clk_i]
} else {
    create_clock -name {sys_clk_i}  -period 8 -waveform {0  4} [get_ports sys_clk_i]
}
```

Figure 7.13. Timing Constraint File (.ldc) for the PCIe x8 IP

- Add the timing constraints shown in Figure 7.15 in the design's .pdc or constraint file. Refer to the latest Lattice Radiant User Guide in the Lattice Radiant web page to learn more about the .pdc file.
- For sys_clk_i, refer to Table 2.1 on selecting the frequencies for Gen1, Gen2, Gen3, or Gen4 data rates. You can use a PLL IP to create these clocks. Refer to the PLL Module IP User Guide (FPGA-IPUG-02063) for instantiation and generation of PLL IP. Figure 7.14 shows the IP configuration for Gen 4 rates if using an input clock of 100 MHz.

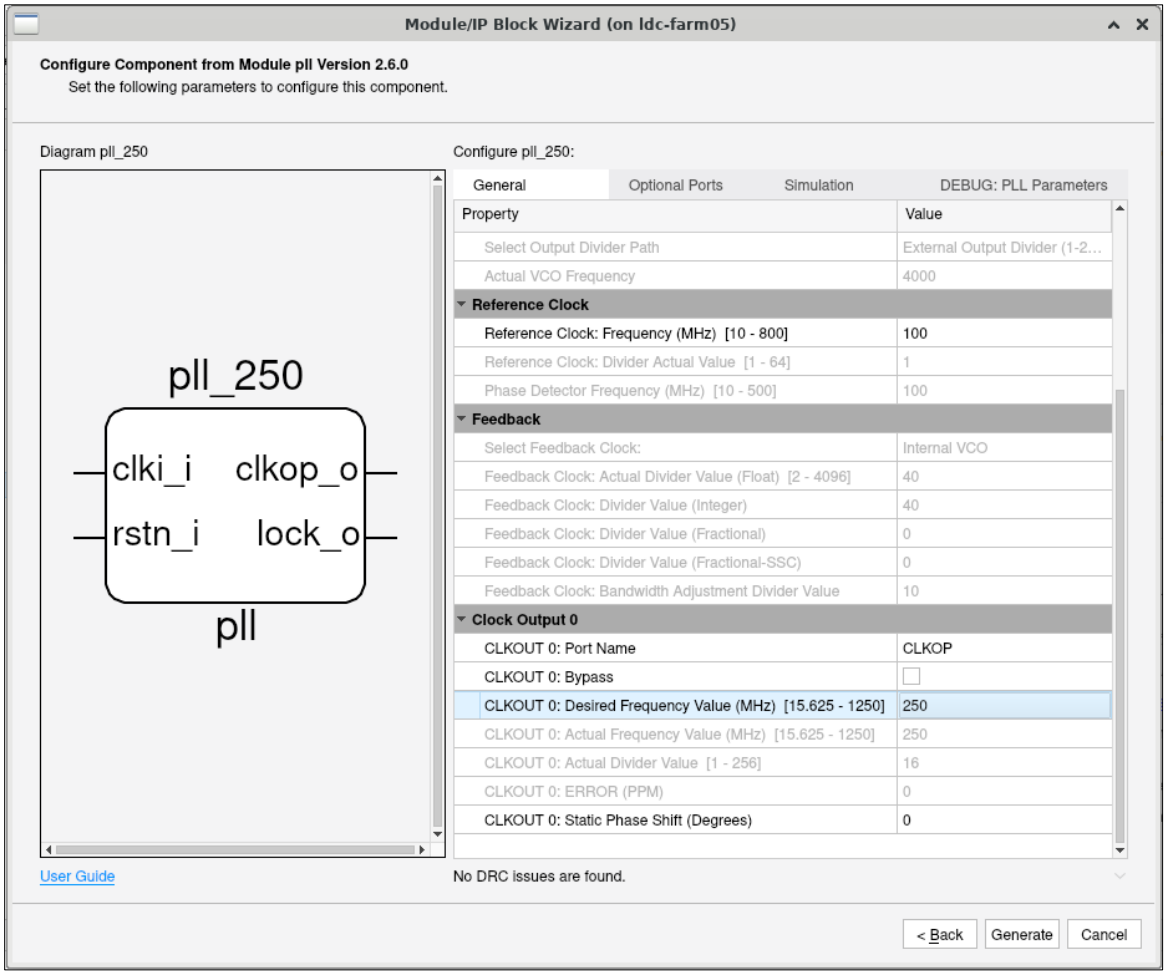


Figure 7.14. PLL IP Configuration for Input Clock of 100 MHz

- Define input reference clock of PLL in the timing constraints using *create_clock -name ….* As shown in Figure 7.15, the input reference clock of 100 MHz is sourced from an I/O pin and named *clk_100* in the design.

```
create_clock -name {clk_100} -period 10 [get_ports clk_100]
```

Figure 7.15. Timing Constraints for PCIe x8 IP Example

7.1.4. Multi-Seed Timing Closure

There is a known issue where the PCIe hardened DMA in Gen4x8 configuration may not be able to close timing on a certain seed. The recommended workaround is to enable multi seed run on Radiant and choose the best/passing seed.

This can be done by changing the *Placement Iterations* to 10 under *Place & Route Design* tab in Radiant settings.

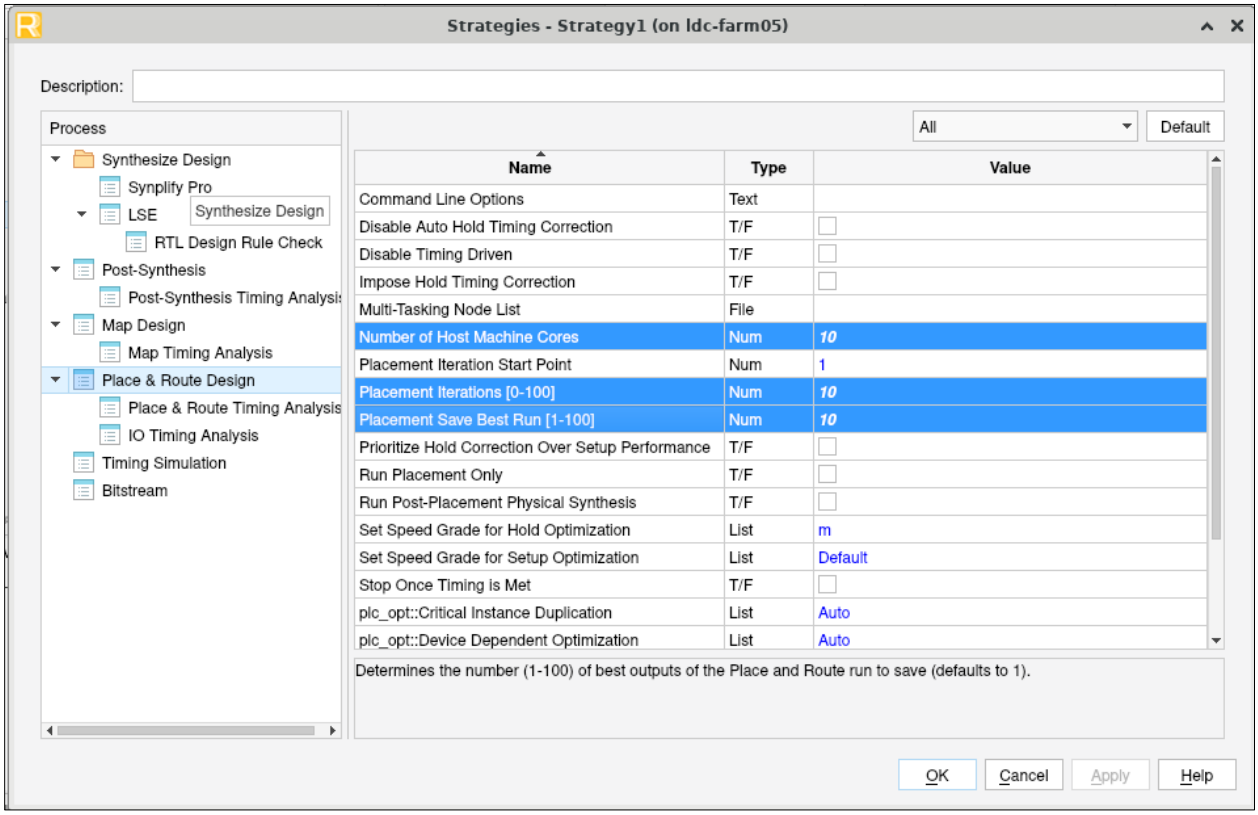


Figure 7.16. Placement Iteration Setup on Radiant under Strategies Tab

8. Debugging

The PCIe protocol involves the interface between a root port and endpoint with both sides being linked up. Hence, PCIe issues can range from device recognition issues, link training issue, flow control errors, enumeration issues, link down due to fatal errors, and others. This section provides the debug flow diagrams for some of the most common issues when using the PCIe x8 IP. Several debug flow charts are introduced with additional information on critical debug registers to refer to and loopback diagnostic features. This section also provides a short description of signals to be used for debugging simulation.

8.1. Debug Methods

8.1.1. Debug Flow Charts

One debugging method is to identify the type of PCIe issue. The following sections show the steps to debug various issues.

8.1.1.1. Hardware Detection Failure

Follow the steps shown in the flow diagram below if the system is not detecting the hardware.

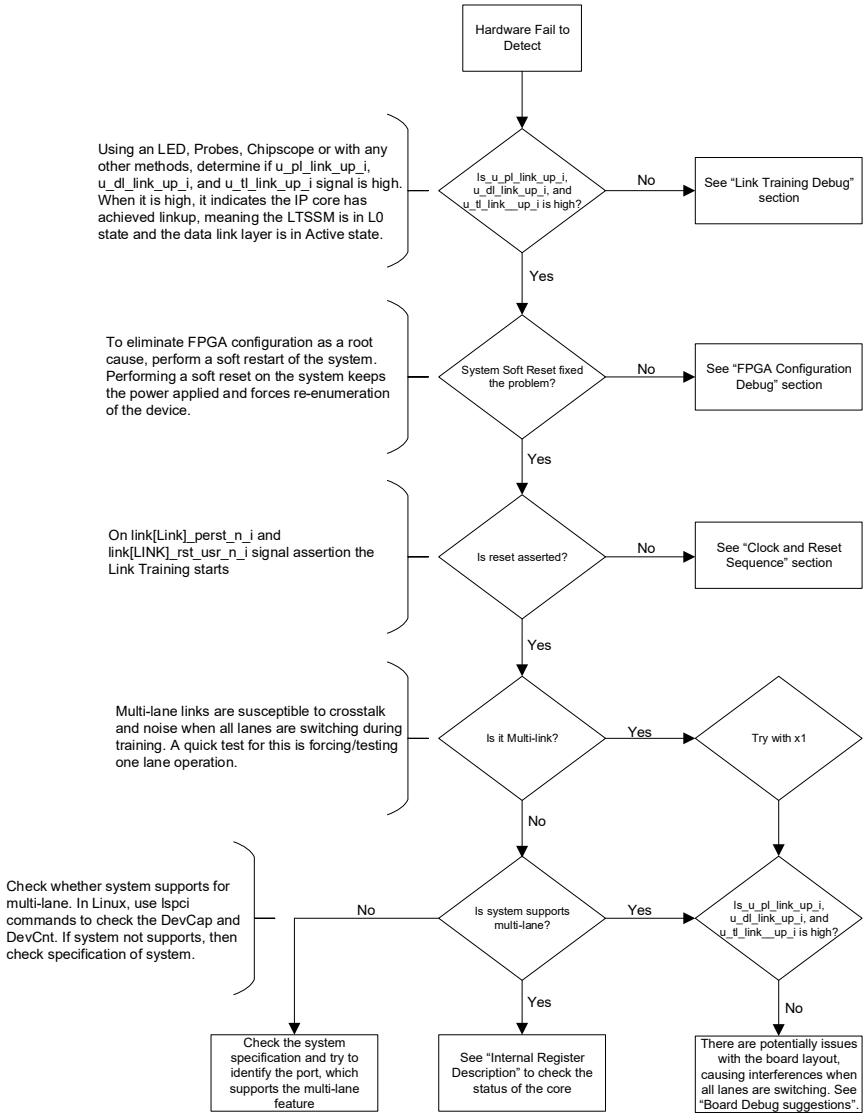


Figure 8.1. Hardware Detection Failure Debugging Flow

8.1.1.2. Link Training Debug

For the link training debug, refer to the flow chart as shown Figure 8.2.

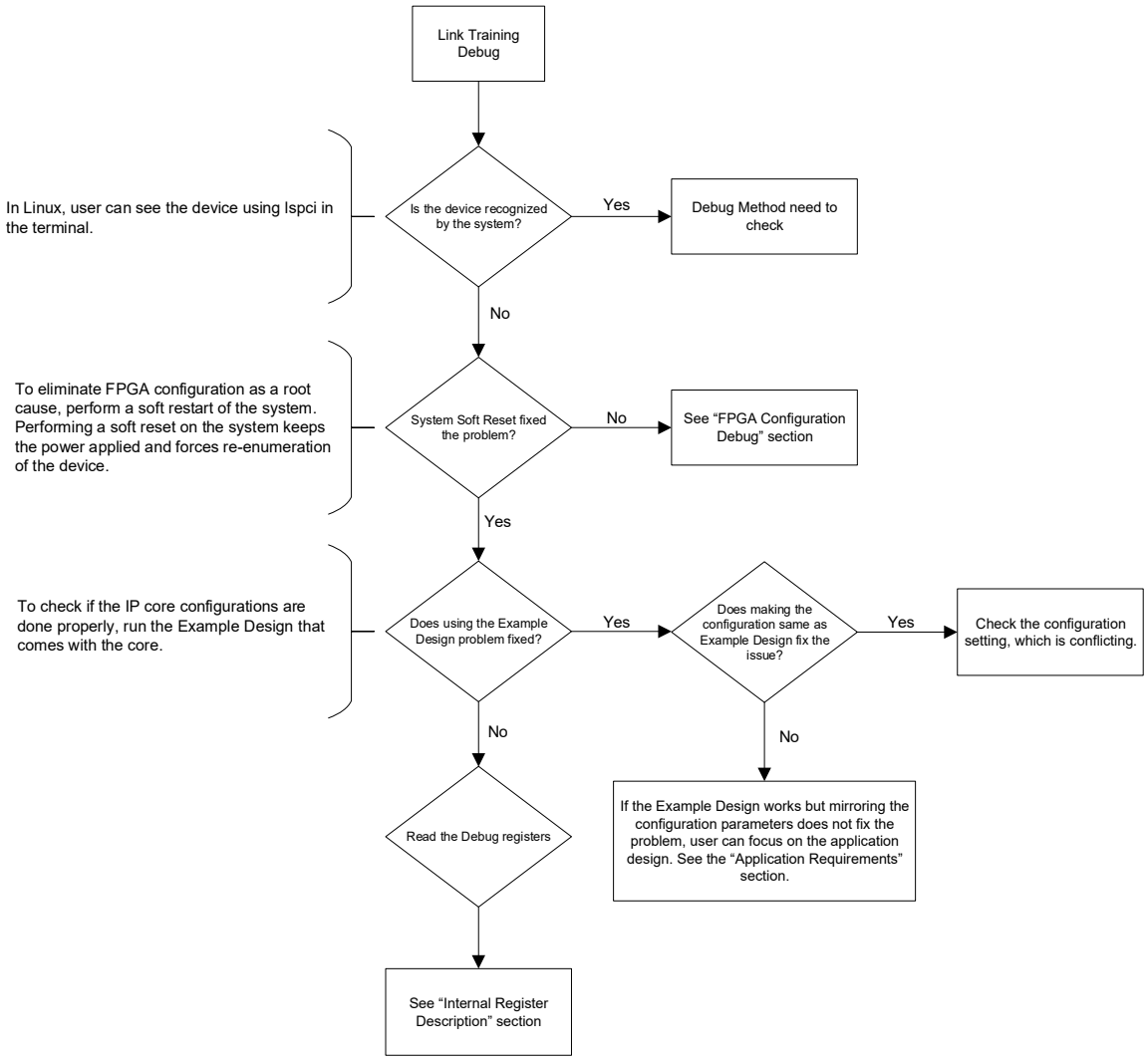


Figure 8.2. Link Training Issue Debugging Flow

You can enable the LTSSM polling feature to assist the debugging. This feature is disabled by default and must only be enabled for debugging and disabled for production. You can drive the ltssm_polling_en signal to 1 to enable. When enabled, the ltssm_state and ltssm_sub_state signals indicate the PCIe LTSSM state and sub-state.

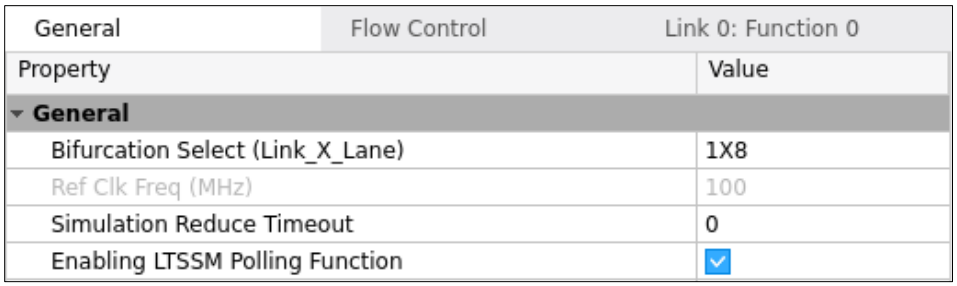


Figure 8.3. Checkbox to Enable LTSSM Polling

Table 8.1. PCIe LTSSM State and Sub-State Definition

State Value	State Definition	Sub-state Value	Sub-state Definition
0	DETECT	0	DETECT_INACTIVE
		1	DETECT_QUIET
		2	DETECT_SPD_CHG0
		3	DETECT_SPD_CHG1
		4	DETECT_ACTIVE0
		5	DETECT_ACTIVE1
		6	DETECT_ACTIVE2
		7	DETECT_P1_TO_P0
		8	DETECT_P0_TO_P1_0
		9	DETECT_P0_TO_P1_1
		10	DETECT_P0_TO_P1_2
1	POLLING	0	POLLING_INACTIVE
		1	POLLING_ACTIVE_ENTRY
		2	POLLING_ACTIVE
		3	POLLING_CFG
		4	POLLING_COMP
		5	POLLING_COMP_ENTRY
		6	POLLING_COMP_EIOS
		7	POLLING_COMP_EIOS_ACK
		8	POLLING_COMP_IDLE
2	CONFIGURATION	0	CONFIGURATION_INACTIVE
		1	CONFIGURATION_US_LW_START
		2	CONFIGURATION_US_LW_ACCEPT
		3	CONFIGURATION_US_LN_WAIT
		4	CONFIGURATION_US_LN_ACCEPT
		5	CONFIGURATION_DS_LW_START
		6	CONFIGURATION_DS_LW_ACCEPT
		7	CONFIGURATION_DS_LN_WAIT
		8	CONFIGURATION_DS_LN_ACCEPT
		9	CONFIGURATION_COMPLETE
		10	CONFIGURATION_IDLE
3	L0	0	L0_INACTIVE
		1	L0_L0
		2	L0_TX_EL_IDLE
		3	L0_TX_IDLE_MIN
4	RECOVERY	0	RECOVERY_INACTIVE
		1	RECOVERY_RCVR_LOCK
		2	RECOVERY_RCVR_CFG
		3	RECOVERY_IDLE
		4	RECOVERY_SPEED0
		5	RECOVERY_SPEED1
		6	RECOVERY_SPEED2
		7	RECOVERY_SPEED3
		8	RECOVERY_EQ_PH0
		9	RECOVERY_EQ_PH1
		10	RECOVERY_EQ_PH2
		11	RECOVERY_EQ_PH3

State Value	State Definition	Sub-state Value	Sub-state Definition
5	DISABLED	0	DISABLED_INACTIVE
		1	DISABLED_0
		2	DISABLED_1
		3	DISABLED_2
		4	DISABLED_3
6	LOOPBACK	0	LOOPBACK_INACTIVE
		1	LOOPBACK_ENTRY
		2	LOOPBACK_ENTRY_EXIT
		3	LOOPBACK_EIOS
		4	LOOPBACK_EIOS_ACK
		5	LOOPBACK_IDLE
		6	LOOPBACK_ACTIVE
		7	LOOPBACK_EXIT0
		8	LOOPBACK_EXIT1
7	HOT_RESET	0	HOT_RESET_INACTIVE
		1	HOT_RESET_HOT_RESET
		2	HOT_RESET_MASTER_UP
		3	HOT_RESET_MASTER_DOWN
8	TX_L0S	0	TX_L0S_INACTIVE
		1	TX_L0S_IDLE
		2	TX_L0S_TO_L0
		3	TX_L0S_FTS0
		4	TX_L0S_FTS1
9	L1	0	L1_INACTIVE
		1	L1_IDLE
		2	L1_SUBSTATE
		3	L1_TO_L0
10	L2	0	L2_INACTIVE
		1	L2_IDLE
		2	L2_TX_WAKE0
		3	L2_TX_WAKE1
		4	L2_EXIT
		5	L2_SPEED

8.1.1.3. Data Transfer Debug

If data transfer fails, refer to the steps shown in Figure 8.4.

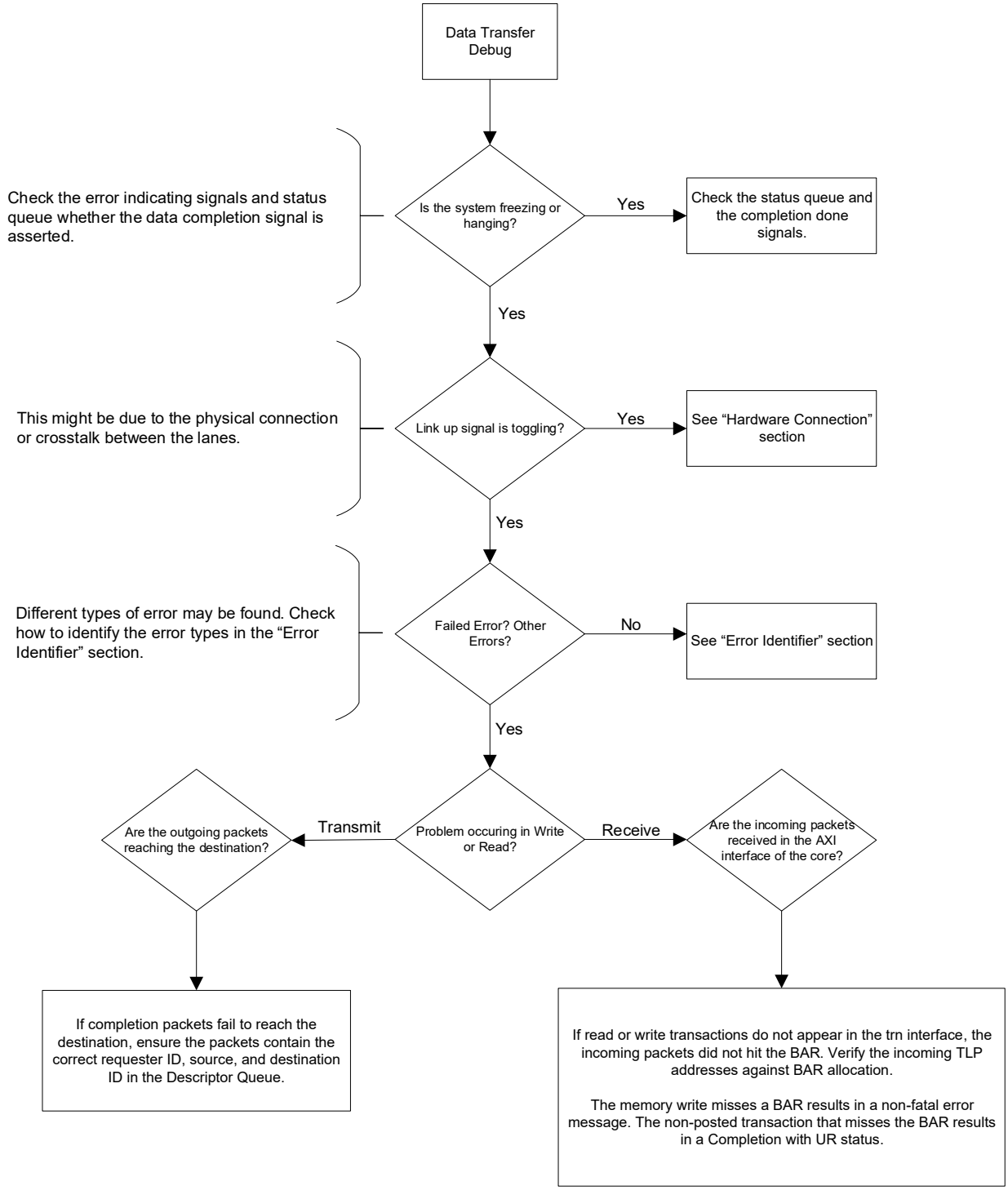


Figure 8.4. Data Transfer Issue Debugging Flow

8.1.1.4. FPGA Configuration Debug

Device initialization and configuration issues can be caused by not having the FPGA configured fast enough to enter link training and be recognized by the system. Prior to becoming operational, the FPGA goes through a sequence of states, including initialization, configuration, and wake-up. After programming the FPGA, a soft reset is required to configure the FPGA from flash. Performing the soft reset operation restarts the enumeration process.

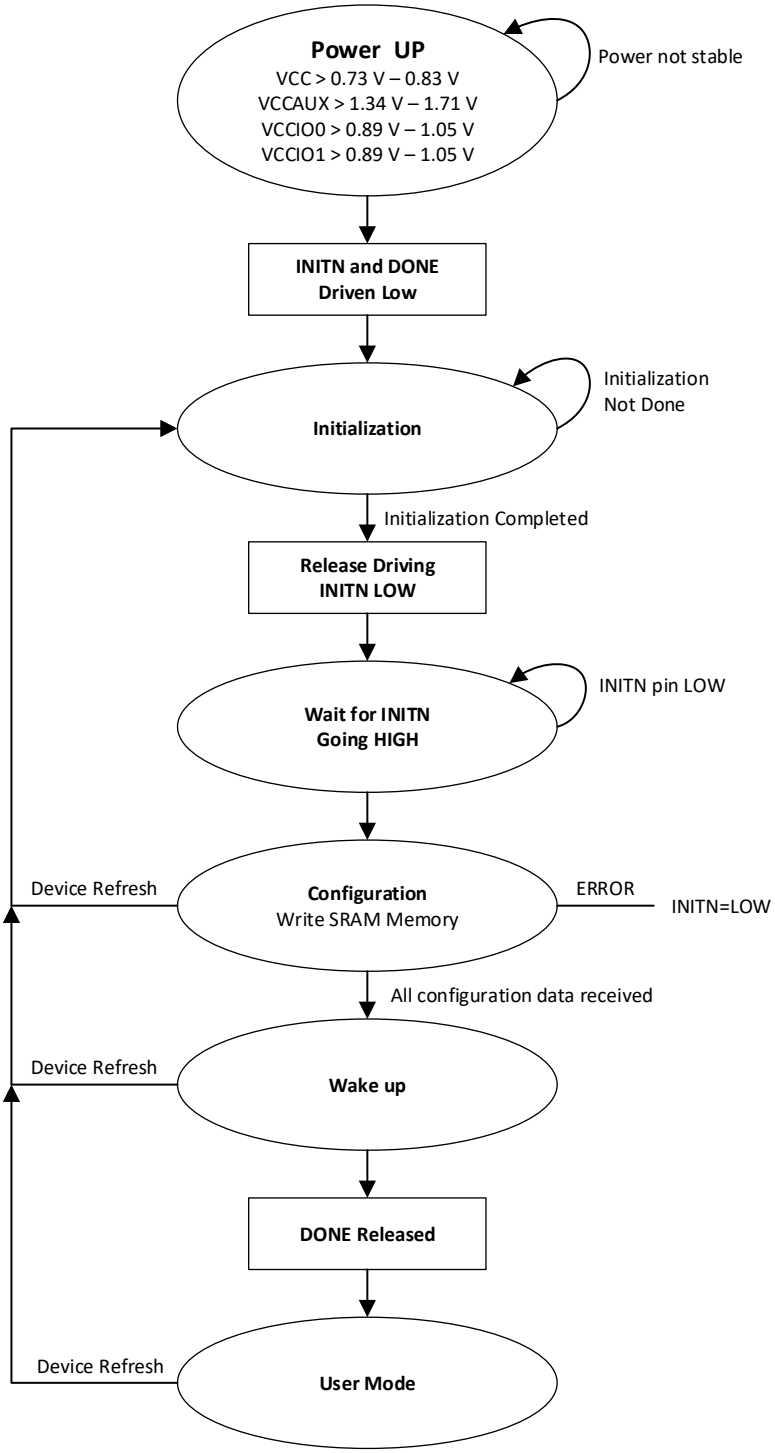


Figure 8.5. Debugging the FPGA Configuration Issues Flow

You can refer to the sysCONFIG User Guide for Nexus Platform (FPGA-TN-02099) document for more information on FPGA configuration.

8.1.2. Internal Register Read for Debug

If the above flowcharts do not capture the issues mentioned, you can read the 0x03114 (*vc_rx_status*) register, which indicates the Receive Buffer Parity/ECC Status where the error detection status can be seen.

The PCIe capability register addresses listed below also provide relevant information for debugging the PCIe issues:

- 0x47-0x44 (Device capability register) address for checking the different supported capabilities of the connected system.
- 0x49-0x48 (Device Control Register) address for checking the supported capabilities of the device.
- 0x4B-0x4A (Device Status Register) address for the device status. You can obtain the error status through this register address.

8.1.3. PCIe Loopback Test

The PCIe loopback test is a diagnostic feature specified by the PCIe Specification that can aid in debugging. The LTSSM Loopback is a state of the Link Training and Status State Machine (LTSSM), which is a mechanism for managing the link state of a serial bus such as PCI Express. In this state, the link partner can test its own transmitter and receiver by sending and receiving data packets without involving the link partner.

The LTSSM Loopback state can be entered from two different states: Configuration or Recovery. The entry into Loopback state is initiated by a Leader Loopback. Before register 0x2100 field is set to 1, all relevant registers containing Leader Loopback control options must be set to the desired values.

When mgmt_tlb_debug_direct_to_loopback = 1, no Leader Loopback control options may be changed. The LTSSM Loopback state has three substates: Entry, Active, and Exit. In Entry substate, both link partners wait for eight EIOS (End of Initialization Ordered Sets) before transitioning to Active substate. In Active substate, both link partners exchange data packets for testing purposes. In Exit substate, both link partners wait for eight EIOS before transitioning to Recovery or Configuration state depending on whether they received an Electrical Idle signal or not.

The LTSSM Loopback is useful for debugging and characterizing the performance of PCI Express links during Link Equalization Training which is a process of optimizing the signal quality between two link partners by adjusting various parameters such as amplitude, de-emphasis, preshoot, and jitter. For more information on the Loopback state, see Table 2.5.

9. Design Consideration

9.1. DMA Based Design

To create a DMA-based design:

1. According to the PCIe IP configuration, select the appropriate clocking architecture. Refer to the Clocking section.
2. Select a designated DMA option in *Configuration Mode* Drop-down menu. Refer to the General section.
3. Initialize the register using LMMI interface/AXI-Lite interface which are configured from the IP wizard. Refer to the LMMI Interface section.
4. Verify DMA transaction for both Host-to-FPGA and FPGA-to-Host Transactions. Refer to the AXI Data Interface (Hardened DMA) section for Hardened DMA and the AXI Data Interface (DMA) section for DMA.

9.2. Non-DMA Based Design

To create a Non-DMA based design:

1. According to the PCIe IP configuration, select the appropriate clocking architecture. Refer to the Clocking section.
2. Select the proper Configuration Mode in IP according to the design requirement. Refer to the General section.
3. Initialize the register using LMMI interface/AXI-Lite interface which are configured from the IP wizard. Refer to the LMMI Interface section.
4. Verify the TLP write and read Transactions. Refer to the Transaction Layer Interface section.
5. Verify the AXI-Stream write and read Transactions. See AXI-Stream Data Interface section.
6. Verify the AXI-MM manager transactions. See Non-DMA AXI Data Interface section.
7. Verify the AXI-Lite manager transactions. See Non-DMA AXI Data Interface section.
8. Select the BAR's with BAR size according to the requirements. See Base Address Register (BAR) [0 to 5] section.

Appendix A. Resource Utilization

The Lattice PCIe IP core utilization report is provided in this section. You can check the resource utilized by the IP core and design top logic based on the available resource in the Avant-AT-G/X FPGA device.

Table A.1 shows a sample resource utilization of the Lattice PCIe x8 IP Core on LAV-AT-G70-1LI with various link widths.

Table A.1. Lattice PCIe IP Core Resource Utilization

PCIe Core Config	Device Family	Map Resource Utilization								
		DMA				Non-DMA				
		LUT4	PFU Register	I/O Buffer	EBR	LUT4	PFU Register	I/O Buffer	EBR	Data Interface Type / Register Interface Type
1×1 EP	Avant-AT-G/X	—	—	—	—	1095	120	10	0	AXI-Stream/ AXI-Lite
1×2 EP	Avant-AT-G/X	—	—	—	—	1606	132	10	0	AXI-Stream/ AXI-Lite
1×4 EP	Avant-AT-G/X	—	—	—	—	2670	156	10	0	AXI-Stream/ AXI-Lite
1×8 EP	Avant-AT-G/X	—	—	—	—	4774	204	10	0	AXI-Stream/ AXI-Lite
1x1 EP	Avant-AT-G/X	—	—	—	—	341	3	43	0	TLP/ LMMI
1x2 EP	Avant-AT-G/X	—	—	—	—	485	3	43	0	TLP/ LMMI
1x4 EP	Avant-AT-G/X	—	—	—	—	773	3	43	0	TLP/ LMMI
1x8 EP	Avant-AT-G/X	—	—	—	—	1350	3	43	0	TLP/ LMMI
1×1 EP	Avant-AT-G/X	—	—	—	—	3469	1997	42	0	AXI-MM/ LMMI
1×1 EP	Avant-AT-G/X	—	—	—	—	3743	2081	42	0	AXI-MM/ AXI-Lite
1x8 EP (Hardened DMA)	Avant-AT-G/X	2555	647	11	0	—	—	—	—	AXI-MM/ LMMI
1x8 EP (Hardened DMA)	Avant-AT-G/X	2754	734	11	0	—	—	—	—	AXI-MM/ AXI-Lite
1x4 EP (DMA)	Avant-AT-G/X	10557	12412	20	29	—	—	—	—	AXI-MM/ LMMI
1x2 EP (DMA)	Avant-AT-G/X	8829	8886	20	25	—	—	—	—	AXI-MM/ LMMI
1x1 EP (DMA)	Avant-AT-G/X	7800	7459	20	23	—	—	—	—	AXI-MM/ LMMI

Note: Resource utilization differs with different configurations of the PCIe IP. The above resource utilization is provided for reference only. You can view the resource utilization under *Report > Map > Map Resource Usage*. To view the resource usage, you must run the Synthesize and Map Design.

References

- PCIe x8 IP Release Notes (FPGA-RN-02061)
- Avant-G/X PCIe Host DMA Driver Software User Guide (FPGA-TN-02405)
- Lattice Avant and Nexus PCIe Host DMA Driver Software User Guide (FPGA-TN-02386)
- Lattice Avant and Nexus PCIe Basic Memory-Mapped Host Driver (Non-DMA) User Guide (FPGA-TN-02387)
- Avant-G web page
- Avant-X web page
- PCI Express Base Specification, Revision 4.0
- PCI Local Interface Specification Revision 3.0
- PCI Bus Power Management Interface Specification Revision 1.2

Other references:

- Lattice Insights for Lattice Semiconductor training courses and learning plans
- Lattice Radiant FPGA design software

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Note: In some instances, the IP may be updated without changes to the user guide. The user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.

Revision 1.6, IP v3.0.0, December 2025

Section	Change Summary
All	• Updated IP version on the cover page. • Made editorial fixes across the document. • Changed harden to *hardened* across the document.
Introduction	• Updated Table 1.1. Summary of the PCIe x8 IP to add Certus-N2 device, update IP and Radiant version, updated QuestaSim information to *QuestaSim Pro, QuestaSim Lattice Edition,* and added table notes regarding IP versions. • Updated Table 1.2. PCIe x8 IP Support Readiness to change PCIe Harden DMA to *PCIe DMA (Hardened)*, added PCIe DMA row, and added Certus-N2 device support. • Updated Soft IP section content. • Updated Licensing and Ordering Information section content including removing Ordering Part Number section. • Updated Table 1.3. Lattice PCIe IP Core Supported Speed Grade to add Certus-N2 device support.
Functional Description	• Updated sub-bullet point for clk_usr_o in Clocking Overview. • Updated Reset section content including updating Figure 2.6. Reset Signals in Lattice PCIe IP Core. • Corrected APSM to *ASPM* references in the Power Management section, including the section names. • Updated section name to DMA Support (Hardened) and section content, including updating table values and adding Table 2.10. Hardware Data Throughput for FPGA-to-Host (F2H) Transfer to Table 2.11. Hardware Data Throughput for Host-to-FPGA (H2F) Transfer. • Added DMA Support section. • Updated Non-DMA Support section content, including updating Figure 2.13. Non-DMA Application Data Flow – TLP Interface, Table 2.48. Register Access for Different Data Interfaces, and Figure 2.17. Non-DMA Write Operation (TLP Data Interface); and removed AXI Bridge Mode section. • Updated Hard IP Interface section content, including updating and adding tables; updated Figure 2.24. TLP Memory Request Header Format for 64-bit Addressing of Memory, added Figure 2.25. TLP Memory Request Header Format for 32-bit Addressing of Memory, Figure 2.26. TLP Memory Read Operation for Link0 (TLP Interface Width of 512 Bits, 256 Bits, and 128 Bits), Figure 2.27. TLP Memory Read Operation for Link0 (TLP Interface Width of 64 Bits), and Figure 2.31 TLP Memory Write Operation for Link0 (TLP Interface Width of 512 Bits, 256 Bits, 128 Bits, and 64 Bits);and removed some tables and figures. • Updated Data Interface Conversion to add Bridge Mode sub-section.

IP Parameter Description	• Updated Figure 3.1. Attributes in the General Tab and Table 3.1. General Tab Attributes Description to add and remove rows and updated the Attribute and Selectable Values columns.
	• Added the following sections:
	  ◦ Optional Port
	  ◦ ASPM Capability
	  ◦ DMA/Bridge Mode Support
	  ◦ ATS Capability
	  ◦ Latency Tolerance Reporting Capability
	  ◦ Power Budgeting Capability
	• Updated table caption to Table 3.18. Function 1-7 Tab and section row to *Link [k] (k == 0 - 1) Function n (n == 1 – 7)*.

Section	Change Summary
Signal Description	• Made some formatting fixes in the tables across this section. • Updated the Description column for link0_rst_usr_n_i, sys_clk_i, and link0_clk_usr_o in Table 4.1. Clock and Reset Ports. • Updated version from 2024 to *2024.1* release in Table 4.3. TLP Transmit Interface Ports. • Updated Table 4.5. TLP Receive Interface Ports to update port to *link0_rx_f_o[7:0]* and software version to *2024.2 release*. • Updated Lattice Memory Mapped Interface (LMMI) section content, including Table 4.7. Lattice Memory Mapped Interface Ports, to add information on a known issue. • Updated rows in Table 4.9. AXI-Lite Configuration Interface Ports. • Updated AXI-Stream Data Interface section description and updated software release from 2024 to *2024.1* in Table 4.10. AXI-Stream Transmitter Interface Ports. • Updated rows, including version from 2024 to *2024.1* release in Table 4.11. AXI-Stream Receiver Interface Ports. • Updated section name to AXI Data Interface (Hardened DMA) and section description to: *This interface is available when Configuration Mode is in Hardened DMA Mode.* • Added AXI Data Interface (DMA) and AXI Data Interface (Bridge Mode) sections. • Removed Non-DMA AXI Data Interface section.
Register Description	• Updated version from 2024 to *2024.1* release across this section. • Updated Hard IP Core Configuration and Status Registers section content to include known issue information and updated values in MPPHY 0 Register and MPPHY 1 Register sections.

Example Design	• Updated Example Design section content.
	  • Updated Example Design Supported Configuration content including Table 6.1. PCIe x8 IP Configuration Supported by the Example Design to add and remove some rows, updated values, and add table note 2.
	  • Updated Overview of the Example Design and Features section content.
	  • Updated Example Design Components section content, including adding Generating Hardened DMA Example Design and DMA Design (DMA) sub-sections, renamed section to Non-DMA Design (TLP Interface) and updated content, including removing figures and adding Generating the Non-DMA Example Design sub-section, added Non-DMA Design (Bridge Mode), PDC Settings for Hardware Example Design, and Running the Hardware Using the Production Driver sections.
	  • Updated Running Functional Simulation section content, including section name, and removed Running Functional Simulation (DMA) section.
	  • Removed Generating the Example Design for Hardware section.
	  • Updated Design Test Case Examples and Limitations of the Example Design section content.
	  • Updated Debugging Example Design Issues section content, including adding Simulation Debug for DMA Design to Simulation Debug for Non-DMA (TLP Interface) Design sub-sections.

Designing with the IP	• Added note for screenshots. • Updated Generating and Instantiating the IP section content, including figures and tables.
Design Consideration	Updated the steps and reference section in DMA Based Design and Non-DMA Based Design.
Appendix A. Resource Utilization	Updated Table A.1. Lattice PCIe IP Core Resource Utilization to update values and add rows for DMA.
References	Added *Lattice Avant and Nexus PCIe Host DMA Driver Software User Guide (FPGA-TN-02386)* document reference.
Revision History	Added note regarding IP version.

Revision 1.5, IP v2.4.0, October 2025

Section	Change Summary
All	• Updated IP version on the cover page. • Made editorial fixes across the document.
Introduction	• Updated IP core and Radiant software version in Table 1.1. Summary of the PCIe x8 IP. • Updated Single Seat Annual column for Avant-X to change PCI-EXP4-AVG-US and PCI-EXP2-AVG-US to PCI-EXP4-AVX-US and PCI-EXP2-AVX-US in Table 1.3. Ordering Part Number.
Functional Description	Updated Clocking Overview section content to add note regarding link0_clk_usr_o limitation.
IP Parameter Description	• Removed Optional Port, Latency Tolerance Reporting Capability, and Power Budgeting Capability sections as these are not supported by the IP. • Updated Figure 3.1. Attributes in the General Tab, Figure 3.3. Attributes in Function Configuration Tab, and Figure 3.10. Attributes in Advance Error Reporting Capability. • Updated Table 3.10. Advance Error Reporting Capability Attributes to remove Link [k] Enable Reporting Surprise Down Error row.
Signal Description	• Updated Table 4.1. Clock and Reset Ports to add link0_clk_usr_o row and remove sys_rst_n_i row. • Added Function Level Reset (FLR) Interface section.
Example Design	• Added note in the Overview of the Example Design and Features section. • Updated Non-DMA Variant to changed reference document name to Lattice Avant and Nexus PCIe Basic Memory-Mapped Host Driver (Non-DMA) User Guide (FPGA-TN-02387).
References	Updated reference document name from CertusPro-NX and Avant-G/X PCIe Basic Memory-Mapped Host Driver (Non-DMA) User Guide (FPGA-TN-02387) to *Lattice Avant and Nexus PCIe Basic Memory-Mapped Host Driver (Non-DMA) User Guide (FPGA-TN-02387)*.

Revision 1.4, IP v2.3.0, June 2025

Section	Change Summary
All	• Updated IP version on the cover page. • Made editorial fixes across the document. • Changed AXI4to *AXI* and AXI-L to *AXI-Lite* instances across the document, including in section names, figures, tables, and figure/table captions.
Acronyms in this Document	Updated from AXI-L to *AXI-Lite* and added AXI-MM acronym.
Introduction	• Removed basic data rate info in the Overview of the IP section. • Updated section name from Supported FPGA Family to Supported Devices; updated the IP core and Radiant version, removed Targeted Devices row, and updated Supported User Interface row including adding AXI-MM support in Table 1.1. Summary of the PCIe x8 IP. • Updated IP Support Summary section to add table caption Table 1.2 PCIe x8 IP Support Readiness and update table content as well, including adding AXI-MM/AXI-Lite rows and table note. • Updated Features section to remove some bullet items and added AXI-MM and AXI-Lite bullet points in the Soft IP section. • Updated Gen4x8 speed grade to 3 only in Table 1.4. Lattice PCIe IP Core Supported Speed Grade.
Functional Description	• Updated PCIe IP Architecture Overview section content including Figure 2.1. Lattice PCIe x8 IP Core Block Diagram, Figure 2.2. Lattice 8-lane SERDES/PCS + PCIe Hard-IP, and • Figure 2.3. Lattice PCIe x8 Core Hard IP. • Updated Clocking section content including Figure 2.4. PCIe IP Clock Domain Block Diagram with External PLL and Figure 2.5. PCIe IP Clock Domain Block Diagram for TLP Interface. • Removed reset usr_cfg_reset_n_i signal info and updated Figure 2.6. Reset Signals in Lattice PCIe IP Core in Reset Overview. • Updated Protocol Layers, and Legacy Interrupt section content. • Updated Non-DMA Support section content, including updating Table 2.10. Register

Section	Change Summary
	Access for Different Data Interfaces to add AXI-MM and AXI-Lite rows, adding new figures such as Figure 2.12. Non-DMA Application Data Flow – AXI-Stream Interface, Figure 2.13. Non-DMA Application Data Flow – AXI-MM Interface (Bridge Mode), Figure 2.14. Non-DMA Application Data Flow – AXI-Lite Interface (Bridge Mode) and added AXI Bridge Mode section. • Changed m0_axi4_arcache_o [3:0] to m0_axi_arcache_o [3:0] in the DMA Scatter-Gather and Status Queues section. • Updated AXI-Stream Interface section to add Lane x8 bullet point, updated section name to AXI-Stream to PCIe Transaction, and updated Figure 2.37. AXI-Stream Data Interface, AXI-Lite Register Interface.

IP Parameter Description	• Updated the following figures:
	  • Figure 3.1. Attributes in the General Tab
	  • Figure 3.4. Attributes in Function Configuration Tab
	  • Figure 3.5. Attributes in BAR Tab
	• Updated section name to Function.
	• Updated Table 3.1. General Tab Attributes Description to update values including adding rows for Number of User Interrupt and PCIe BAR Mapping.
	• Updated Table 3.4. Function Configuration Tab Attributes to remove Disable Function row.
	• Updated Table 3.5. BAR Tab Attributes to remove Local Memory Base Address row.

Signal Description	• Updated Table 4.1. Clock and Reset Ports to remove link0_clk_usr_o row and table notes and reference to table note. • Updated Table 4.7. Lattice Memory Mapped Interface Ports to change value to 100 MHz for usr_lmmi_clk_i, updated port name to usr_lmmi_offset_i [23:0] including Description values, updated port names to usr_lmmi_wdata_i [15:0] and usr_lmmi_rdata_o [15:0], and removed table note. • Added Table 4.8. Transaction Pending Ports. • Updated content of Table 4.9. AXI-Lite Configuration Interface Ports, mostly in Ports and Clock Domain. • Updated AXI-Stream Data Interface section content including Table 4.10. AXI-Stream Transmitter Interface Ports and Table 4.11. AXI-Stream Receiver Interface Ports. • Added Non-DMA AXI Data Interface section. • Updated Hard DMA AXI Data Interface content including section name, Table 4.16. DMA AXI Manager Write Interface and changing Slave instances to Subordinate. • Removed Unused Interface section.

Register Description	• Removed table notes in the following tables:
	  • Table 5.18. ltssm_compliance_toggle Register 0x68
	  • Table 5.20. ltssm_link Register 0x80
	  • Table 5.44. debug_loopback_control Register 0x104

Example Design	• Updated section content including Example Design Supported Configuration and Table 6.1. PCIe x8 IP Configuration Supported by the Example Design. • Updated Overview of the Example Design and Features section to remove footnote. • Updated Example Design Components to add TLP Interface in Non-DMA Design and added Non-DMA Design (Bridge Mode) support, including Figure 6.4. Components within Non-DMA Design (Bridge Mode). • Updated Running Functional Simulation (Non-DMA) section content including Figure 6.7. Testbench Files. • Updated Generating the Example Design for Hardware section content including adding Table 6.2. Top-level Files and PDC Files Corresponding to Data Interface Type Selected for Example Design. • Updated Design Test Case Examples to add TLP Interface and Non-DMA Design (Bridge Mode support in Non-DMA Design.
Designing with the IP	Updated section content including all figures and added Multi-Seed Timing Closure section.

Section	Change Summary
Design Consideration	Updated DMA Based Design section content and added AXI-MM and AXI-Lite steps in Non-DMA Based Design.
Appendix A. Resource Utilization	Updated section content including removing text mentioning the IP and Radiant version, updated Table A.1. Lattice PCIe IP Core Resource Utilization to update values and add rows for 1x1 EP and 1x8 AXI-MM and AXI-Lite.

Revision 1.3, IP v2.2.0, March 2025

Section	Change Summary
All	Updated IP version in the cover page.
Introduction	Updated the IP core and Radiant version in Table 1.1. Summary of the PCIe x8 IP.
Functional Description	Added DMA Performance and DMA Bypass Interface sections.
IP Parameter Description	• Updated Figure 3.1, Figure 3.2. Attributes in the Optional Port Tab, and Figure 3.10. • Updated Table 3.2. Optional Port Attributes to remove rows. • Updated section name to PCI Express Capability.
Example Design	• Updated Table 6.1. PCIe x8 IP Configuration Supported by the Example Design to remove *Use TLP Interface* row and update Non-DMA Design column values for *Bifurcation Selection*, *Data Interface Type*, and *Number of Physical Function*. • Updated Overview of the Example Design and Features description. • Updated Figure 6.2. Components within the Harden DMA Design, DMA Design section content, and OSC document reference under Non-DMA Design section in Example Design Components. • Updated section name to Simulating the Example Design and moved Running Functional Simulation (Non-DMA) and Running Functional Simulation (DMA) under this section (previously from the Designing with the IP section); updated section content as well including adding additional steps and figures. • Added Generating the Example Design for Hardware section for DMA and non-DMA designs. • Updated Limitations of the Example Design section content.
Debugging	Updated Link Training Debug section content including adding Figure 8.2. Link Training Issue Debugging Flow and Table 8.1. PCIe LTSSM State and Sub-State Definition.
References	Added Avant PCIe Driver document references.

Revision 1.2, IP v2.1.0, December 2024

Section	Change Summary
All	Added IP version to the cover page and revision history.
Introduction	• Updated Table 1.1. Summary of the PCIe x8 IP to change to All Avant-AT-G and Avant-AT-X family in Targeted Devices, added IP Changes row, and updated content as well, and updated IP core and Radiant software version. • Updated SRIOV bullet item to add future release for non-DMA text and removed future release info for Hardened high-performance bullet point in Features section. • Added Radiant 2024.1 SP1 information in Soft IP section. • Updated part numbers in Table 1.2. Ordering Part Number. • Renamed IP Validation Summary section to Hardware Support and updated section content. • Added IP Support Summary Hardware Support and Speed Grade Supported section.
Functional Description	• Added a note for bridge core support in the DMA Support section. • Added a note for usr_lmmi_rst_n_i once asserted in the LMMI Interface section.
IP Parameter Description	Updated Figure 3.1. Attributes in the General Tab and Table 3.1. General Tab Attributes Description to add and remove rows and updated Selectable Values and Parameter columns.

Section	Change Summary
Signal Description	• Updated Table 4.7. Lattice Memory Mapped Interface Ports to add note for usr_lmmi_rst_n_i. • Updated Table 4.11. DMA AXI4 Manager Write Interface to update values in Clock Domain column and change m0_axi4 to m0_axi across the table. • Added Table 4.15. DMA AXI4 Interrupt Interface and Unused Interface section.
Example Design	• Updated DMA Design and Non-DMA Design columns in Table 6.1. PCIe x8 IP Configuration Supported by the Example Design. • Updated DMA Design section content including adding Figure 6.2. Components within the Harden DMA Design in Example Design Components. • Updated DMA Design section content in Design Test Case Examples. • Updated Limitations of the Example Design section content including adding non-DMA and Harden DMA info.
Designing with the IP	• Updated Generating and Instantiating the IP section to add non-DMA and Harden DMA info, including updating steps and adding related figures. • Updated section name to Running Functional Simulation (Non-DMA) and section content including updating steps and related figures. • Added Running Functional Simulation (DMA) section.
Appendix A. Resource Utilization	Moved section to Appendix, updated section content including updating Table A.1. Lattice PCIe IP Core Resource Utilization to add 1x8 EP for AXI-MM/LMMI row.
References	Added document reference for PCIe x8 release notes.

Revision 1.1, June 2024

Section	Change Summary
Acronyms in This Document	Rearranged some acronyms in alphabetical order.
Introduction	• Updated IP core version to 1.1.0 in Table 1.1. Summary of the PCIe x8 IP. • Moved Resource Utilization from Appendix A to this section, updated IP core version to 1.1.0, added note that x8 is generated from version 2.0.0, and updated Table 1.4. Lattice PCIe IP Core Resource Utilization to add rows for 1x8 EP AXI4 Stream/AXI-Lite and 1x8 EP TLP/LMMI; updated row values for 1x2 and 1x4 EP TLP/LMMI, including removing DMA EBR values. • Removed future release text in some bullet points and added Future Radiant releases in Radiant 2024.1 bullet point in Features section. • Removed Interrupt ports of Hard DMA Core and Linkup Status ports bullet points and added DMA is supported only in x8 mode bullet point in Hard IP Limitations section.

Section	Change Summary

Functional Description	• Updated the following in PCIe IP Architecture Overview:
	  • Removed soft logic information in the first paragraph.
	  • Adjusted bullet points in this section.
	  • Changed link0_clk_i to *sys_clk_i* in the Clock Interface bullet point.
	  • Removed future release text in *Available if Hard DMA Core is enabled*.
	• Updated APSM L0s section to add link to Table 5.12. ltssm_nfts Register 0x50 for mgmt_tlb_ltssm_nfts_to_extend.
	• Updated entire DMA Support section content, including additional sections and figures.
	• Updated Figure 2.12. Non-DMA Write Operation (TLP Data Interface).
	• Updated the following in Hard IP Interface:
	  • Updated Figure 2.20. TLP Memory Read Operation for Link0 (x4 Lane), Figure 2.21. TLP Memory Read Operation for Link0 (x2 Lane), and Figure 2.22. TLP Memory Read Operation for Link0 (x1 Lane).
	  • Updated the following in LMMI Interface:
	  • Updated text to: *The data transaction, through the LMMI, only starts when both usr_lmmi_request_i and usr_lmmi_ready_o are asserted high* and *When both usr_lmmi_request_i and usr_lmmi_ready_o are asserted high, usr_lmmi_wr_rdn_i, and usr_lmmi_offset_i must be valid and describe the transaction to execute; if the transaction is a write as indicated by usr_lmmi_wr_rdn_i being asserted to high, usr_lmmi_wdata_i must also be valid*.
	  • Updated section information for LMMI Write Operations and LMMI Read Operation, including Figure 2.31. LMMI Write Operation and Figure 2.32. LMMI Read Operation.
	  • Updated Figure 2.34. PCIe to AXI4-Stream Transaction for x1, Figure 2.35. PCIe to AXI4-Stream Transaction for x2, Figure 2.36. PCIe to AXI4-Stream Transaction for x8, Figure 2.37. AXI4-Stream to PCIe Transaction for x1, Figure 2.38. AXI4-Stream to PCIe Transaction for x2, and Figure 2.39. AXI4-Stream to PCIe Transaction for x8.
	• Updated Multi-Protocol Support section content, including adding Figure 2.40.

IP Parameter Description	Removed table note in Table 3.16. Function 1-3 Tab.

Section	Change Summary

Signal Description	• Updated the following in Table 4.1. Clock and Reset Ports:
	  • Updated link0_perst_n_i description field to change link0_rst_n_i to *link0_rst_usr_n_i*.
	  • Updated link0_rst_n_i port and description fields to change link0_rst_n_i to *link0_rst_usr_n_i*.
	  • Changed link0_clk_i port to *sys_clk_i*.
	  • Changed link0_clk_src_o to *link0_clk_usr_o* port and link0_clk_i to *sys_clk_i* in the description field.
	  • Updated port and description fields for link0_pl_up_o, link0_dl_up_o, and link0_tl_up_o to add *link* to the port name.
	  • Removed sys_clk_i row.
	• Updated port and description fields content in Table 4.2. PHY Interface Descriptions, including removing LINK – range [0, 3] in table note.
	• Updated Table 4.4. TLP Transmit Credit Interface Ports to remove table notes and note in link0_tx_credit_nh_o[11:0].
	• Updated Table 4.6. TLP Receive Credit Interface Ports to remove table notes and note in link0_rx_credit_nh_i[11:0] and link0_rx_credit_nh_inf_i.
	• Updated Table 4.7. Lattice Memory Mapped Interface Ports to update port and description fields for usr_lmmi_offset_i.
	• Updated note in Power Management Interface to add information that it is supported in 2024.1 and later.
	• Updated axi4l_awaddr_i and axi4l_araddr_i to change [19:0] to [23:0] in Table 4.8. AXI-L Configuration Interface Ports.
	• Removed Soft DMA Core information in AXI-4 Stream Data Interface.
	• Updated section content and name from DMA Interrupt Interface to DMA Interface, including adding Table 4.11. DMA AXI4 Manager Write Interface to Table 4.14. DMA AXI4 Subordinate Read Interface.

Register Description	• Updated Table 5.2. Hard PCIe Core Register Mapping to update Start Byte Offset column values for Quad 0 MPPHY; updated Start and End Byte Offset columns for Quad 1 MPPHY. • Added DMA Configuration Space Registers section.
Example Design	• Updated Table 6.1. PCIe x8 IP Configuration Supported by the Example Design to DMA Design column values to Not supported in 2024.1 release. • Updated DMA Design Example not supported information to 2024.1 release in Example Design Components and Design Test Case Examples sections. • Updated bullet point in Limitations of the Example Design to include ModelSim 2024.1 not supported information.
Designing with the IP	Updated Design Implementation and Timing Constraints sections to update Lattice Radiant Software user guide version to 2024.1.
Appendix A. Resource Utilization	Updated Table 1.4. Lattice PCIe IP Core Resource Utilization to add rows for 1x8 EP AXI4 Stream/AXI-Lite and 1x8 EP TLP/LMMI; updated row values for 1x2 and 1x4 EP TLP/LMMI.

Revision 1.0, December 2023

Section	Change Summary
All	• Changed document status to Production release. • Changed all lnk signal references to *link* across the document.

Introduction	• Updated Table 1.1. Summary of the PCIe x8 IP to change IP Core and Radiant software version and remove ModelSim information.
	• Updated the following in Features section:
	  • Added information that some features are available in future release.
	  • Updated supported lane configurations to *Radiant 2023.2* and *future release*.
	  • Changed PCIe x4 core support to *1-8 PF*.
	  • Removed Soft IP features.

Section	Change Summary
	• Updated Hard IP Limitations to remove bullet item for burst or consecutive access for PCIe config. • Updated Soft IP to remove Data Interfaces bullet item. • Updated Table 1.2. Ordering Part Number to add OPNs. • Merged Hardware Evaluation content with IP Validation Summary section. • Moved Ordering Part Number under Licensing and Ordering Information section. • Updated Naming Conventions to remove [LINK] index bullet item.
Functional Description	• Updated PCIe IP Architecture Overview to remove Link 1 to 3 bullet items for link layer cores. • Updated Figure 2.2. Lattice 8-lane SERDES/PCS + PCIe Hard-IP to change information that Quad 1 and DMA Core are available in future release. • Updated AXI interface to AXI-S Rx and AXI-S Tx in Figure 2.3. Lattice PCIe x8 Core Hard IP. • Updated Clocking to remove clk_usr_ps90_i information. • Updated Reset to remove c_apb_preset_n_i and change reset signal from usr_lmmi_resetn_i to usr_cfg_reset_n_i. • Updated Table 2.9. Base Address to Enable Interrupt and Table 2.10. Legacy Interrupt Register to change the base address. • Removed Register Address and Lane Register Address in LMMI Read Operation. • Updated AXI-L Interface to remove AHB-Lite and Register Address information and other content related to PCIe AXI-L, including diagrams. • Removed AXI-S Interface, Mapping of the PCIe Config Register to AXI-S Subordinate Interface, and DMA Enabled sections.

IP Parameter Description	• Updated the following in Table 3.1. General Tab Attributes Description:
	  • Added 16.0G selectable values for Link 0 Target Link Speed.
	  • Changed selectable value for Link 0 Number of physical functions to *1-8* and parameter to *LINK0_NUM_FUNCTIONS = {1..8}*.
	  • Added information for Enable DMA Support that attribute is not supported in 2023.2 release.
	  • Removed Merge Register and Subordinate Interface row.
	• Added table note in Table 3.16. Function 1-3 Tab.
	• Removed RX TLP Destination Base Address section.

Signal Description	• Merged content of Clock Interface and Reset Interface, including tables, and moved to Clock and Reset Interface section. Added note in Table 4.1. Clock and Reset Ports that 250 MHz is supported in 2024 release.
	• Updated content of PHY Interface, Power Management Interface, and AXI-L Configuration Interface section.
	• Updated lnk0_tx_data_i and lnk0_tx_data_p_i in Table 4.3. TLP Transmit Interface Ports.
	• Updated content of Table 4.5. TLP Receive Interface Ports and Table 4.7. Lattice Memory Mapped Interface Ports.
	• Updated description of link0_rx_credit_nh_i[11:0] and link0_rx_credit_nh_inf_i and added note 2 in Table 4.6. TLP Receive Credit Interface Ports.
	• Updated AXI-4 Stream Data Interface section and the following items in Table 4.9. AXI-4 Stream Transmitter Interface Ports:
	  • Changed m0_tready_i to *m0_tready_i*.
	  • Changed m0_tvalid_o to *m0_tvalid_o*.
	  • Updated 1x2 and 1x1 values for m0_tstrb_o [NUM_LANES*8-1:0] port.
	  • Updated 1x2 and 1x1 values for m0_tkeep_o [NUM_LANES*8-1:0] port.
	  • Added note for m0_tdest_o[3:0].
	  • Updated port names from m[LINK]_tlast_o and m[LINK]_tid_o [7:0] to *m0_tlast_o* and *m0_tid_o [7:0]*.
	• Updated ports, including updating 1x2 and 1x1 values and added note in s0_tdata_i[NUM_LANES*64-1:0] in Table 4.10. AXI-4 Stream Receiver Interface Ports.

Section	Change Summary
	• Removed Legacy Interrupt Interface section.

Register Description	• Updated content including table name in Table 5.2. Hard PCIe Core Register Mapping.
	• Updated Table 5.3. CSR Values Recommended for EP Applications to add 0x4 in Offset values.
	• Added note and changed section name to mgmt_tlb (0x4_2000) section.
	• Updated Table 5.7. ltssm_cfg Register 0x3c to add that 8G and 16G for [11:8] lw_start_updn_rate_en is available for future release.
	• Added note and reference to table note in the following tables:
	  • Table 5.13. ltssm_ds_initial_auto Register 0x54
	  • Table 5.18. ltssm_compliance_toggle Register 0x68
	  • Table 5.20. ltssm_link Register 0x80Table 5.44. debug_loopback_control Register 0x104
	• Added note and reference to note for Target Link Speed in eq_status_table_control Register 0x3d8.
	• Changed section name to mgmt_ptl (0x4_3000) and mgmt_ftl (0x4_4000).
	• Updated mgmt_ftl_mf[7:1] (0x4_5000,0x4_6000,0x4_7000,0x4_8000,0x4_9000,0x4_A000,0x4_B000) to change section name and update content of Table 5.187. Base Address for mgmt_ftl_mf.
	• Updated Table 5.188. Function Register 0x8 to change the values of the Description column for [31:1] and [0] fields.
	• Updated Table 5.212. PCI Express Capability to add 1001 and 1010 in 43-42 Addr is for future release and added 0100 (16GT/s) value for 4F-4C addr.
	• Updated Table 5.218. Vendor-Specific Extended Capability to add that 8G and 16G in 163-160 addr are for future release.
	• Added ATS Extended Capability and Resizable BAR Capability sections.
	• Removed pcie_11(0x0F000) and Soft IP Configuration, Control, and Status Registers sections.

Example Design	• Updated DMA Design in the section to add that support is available in next release. • Updated DMA Design and Non-DMA Design support and added table note 2 in Table 6.1. PCIe x8 IP Configuration Supported by the Example Design. • Added note and reference to note and changed AXI/AHB-Lite to TLP in Overview of the Example Design and Features. • Updated OSC Module document link and title in Example Design Components. • Updated content in Limitations of the Example Design section.
Designing with the IP	• Updated Figure 7.1. Module/IP Block Wizard, Figure 7.2. IP Configuration, and Figure 7.3. Check Generated Result to change to pcie_x8. • Updated content of Running Functional Simulation section.
Design Consideration	Added this section.
Appendix A. Resource Utilization	Moved Resource Utilization from Introduction to Appendix section and updated Table A.1. Lattice PCIe IP Core Resource Utilization.
References	Updated Avant webpage links.

Revision 0.80, October 2023

Section	Change Summary
All	Initial preliminary release.

www.latticesemi.com