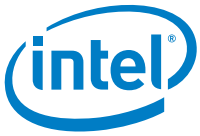


See AC specifications in Section 11.4.2.1.

2.1.13 Miscellaneous

Reserved	Pin Name	Ball #	Type	Name and Function
	LAN_PWR_GOOD	A14	In Pu	LAN Power Good. A 3.3V input signal. A transition from low to high initializes the 82599 into operation. If not used (POR_BYPASS = 0b), an internal Power-on-Reset (POR) circuit triggers the 82599 power-up.
	POR_BYPASS	D19	In Pu	Bypass indication as to whether or not to use the internal POR or the LAN_PWR_GOOD pin. When set to 1b, the 82599 disables the internal POR circuit and uses the LAN_PWR_GOOD pin as a POR indication.
	RSVDAC6_VCC	AC6		This input requires an external pull up. See the product's Schematic Checklist for detail.
	OSC_SEL	AA9	T/s Pu	Defines the input clock connected to the REFCLKIN_p/ REFCLKIN_n pins: 0b XTAL Clock (valid only for 25 MHz) 1b OSC Clock This pin is a strapping option latched at LAN_PWR_GOOD.
	AUX_PWR	AB9	T/s	Auxiliary Power Available. When set, indicates that auxiliary power is available and the 82599 should support $D3_{COLD}$ power state if enabled to do so. This pin is latched at the rising edge of LAN_PWR_GOOD.
	MAIN_PWR_OK	AC9	In	Main Power OK. Indicates that platform main power is up. Must be connected externally.
	LAN1_DIS_N	AD20	T/s Pu	This pin is a strapping pin latched at the rising edge of LAN_PWR_GOOD. If this pin is not connected or driven high during initialization, LAN 1 is enabled. If this pin is driven low during initialization, LAN 1 port is disabled.
	LAN0_DIS_N	AD21	T/s Pu	This pin is a strapping option pin latched at the rising edge of LAN_PWR_GOOD. If this pin is not connected or driven high during initialization, LAN 0 is enabled. If this pin is driven low during initialization, LAN 0 port is disabled. When LAN 0 port is disabled MNG is not functional and it must not be enabled in the EEPROM Control Word 1.

2.1.14 JTAG

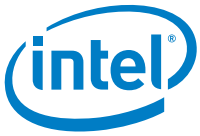
See AC specifications in Section 11.4.2.6.

Reserved	Pin Name	Ball #	Type	Name and Function
	JTCK	B16	In	JTAG Clock Input.
	JTDI	A13	In Pu	JTAG Data Input.
	JTDO	B15	O/d	JTAG Data Output.
	JTMS	B13	In Pu	JTAG TMS Input.
	JRST_N	B14	In Pu	JTAG Reset Input. Active low reset for the JTAG port.

2.1.15 Power Supplies

See AC specifications in Section 11.3.1.

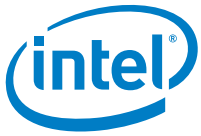
Reserved	Pin Name	Ball #	Type	Name and Function
	VCC1P2		1.2V	Power supply.
		W14, W11, W9, V14, V11, V9, U16, U14, U11, U9, T16, T14, T11, R16, R14, R13, R12, R11, P14, P11, N14, N11, M14, M11, L14, L11, K16, K14, K13, K12, K11, J16, J14, J11, H16, H14, H11, H9, G16, G14, G11, G9, F16, F14, F11, F9, U18, T18, R18, P18, P16, N18, N16, M18, M16, L18, L16, K18, J18, H18, K9, K7, J9, T9, R9, R7, M9, M7, L9, L7, P9, P7, N9, N7		
	VCC3P3		3.3V	Power supply.
		AD19, AD15, AD10, AD6, A19, A15, A10, A6, E7, Y7, L5, P5		
	VSS		0V	Ground
		AD16, AD9, W18, W17, W15, W13, W12, W10, W8, W7, V17, V15, V13, V12, V10, V8, U15, U13, U12, U10, T15, T13, T12, T10, R15, R10, P15, P13, P12, P10, N15, N13, N12, N10, M15, M13, M12, M10, L15, L13, L12, L10, K15, K10, J15, J13, J12, J10, H15, H13, H12, H10, G17, G15, G13, G12, G10, G8, F18, F17, F15, F13, F12, F10, F8, F7, A16, A9, K8, K6, J8, J5, J4, H8, H6, G7, G6, G5, G4, F6, E6, E5, E4, D6, C6, C5, C4, B5, B3, A5, A3, AD5, AD3, AC5, AC3, AB6, AB5, AB4, AA6, Y6, Y5, Y4, W6, V7, V6, V5, V4, U8, U6, T8, T5, T4, R8, R6, M8, M6, M5, M4, M3, L8, L6, L3, K5, K4, K3, K2, K1, J3, H3, H2, H1, G3, F3, F2, F1, E3, D3, D2, D1, C3, B2, B1, A2, A1, AD2, AD1, AC2, AC1, AB3, AA3, AA2, AA1, Y3, W3, W2, W1, V3, U3, U2, U1, T3, R5, R4, R3, R2, R1, P8, P6, P3, N8, N6, N3, AD24, AD23, AD22, AC24, AC23, AC22, AB22, AB21, AB20, AA24, AA23, AA22, Y22, Y21, Y20, Y19, W24, W23, W22, W19, V22, V21, V20, V19, V18, U24, U23, U22, U19, U17, T22, T21, T20, T19, T17, R24, R23, R22, R19, R17, P22, P21, P20, P19, P17, N23, N22, N19, N17, M23, M22, M19, M17, L22, L21, L20, L19, L17, K24, K23, K22, K19, K17, J22, J21, J20, J19, J17, H24, H23, H22, H19, H17, G22, G21, G20, G19, G18, F24, F23, F22, F19, E22, E21, E20, E19, D24, D23, D22, C22, C21, C20, B24, B23, B22, A24, A23, A22		



2.1.16 Pull-Ups

Reserved	Pin Name	Reserved	Internal Pull Up at Power Up		Internal Pull Up at Nominal Active State	
			PUP	Comment	PUP	Comment
	EE_DI		N		N	
	EE_DO		Y		Y	
	EE_SK		N		N	
	EE_CS_N		N		N	
	FLSH_SI		Y		N	
	FLSH_SO		Y		Y	
	FLSH_SCK		Y		N	
	FLSH_CE_N		Y		N	
	SMBCLK		N		N	
	SMBD		N		N	
	SMBALRT_N		N		N	
	SCL0/SCL1		N		N	
	SDA0/SDA1		N		N	
	NCSI_CLK_IN		N		N	
	NCSI_CRS_DV		N		N	
	NCSI_RXD_0		N		N	
	NCSI_RXD_1		N		N	
	NCSI_TX_EN		N		N	
	NCSI_TXD_0		N		N	
	NCSI_TXD_1		N		N	
	MDIO0		N		N	
	MDC0		N		N	
	MDIO1		N		N	
	MDC1		N		N	

Reserved	Pin Name	Reserved	Internal Pull Up at Power Up		Internal Pull Up at Nominal Active State	
			PUP	Comment	PUP	Comment
	SDP0_0 SDP0_1 SDP0_2 SDP0_3 SDP0_4 SDP0_5 SDP0_6 SDP0_7		Y		Y	
	SDP1_0 SDP1_1 SDP1_2 SDP1_3 SDP1_4 SDP1_5 SDP1_6 SDP1_7		Y		Y	
	LED0_0 LED0_1 LED0_2 LED0_3		N		N	
	LED1_0 LED1_1 LED1_2 LED1_3		N		N	
	LAN_PWR_GOOD		Y		Y	
	AUX_PWR		N		N	
	LAN0_DIS_N		Y		Y	
	LAN1_DIS_N		Y		Y	
	MAIN_PWR_OK		N		N	
	JTCK		N		N	
	JTDI		N		N	
	JTDO		N		N	
	JTMS		N		N	
	JRST_N		Y		Y	
	PE_RST_N		N		N	
	PE_WAKE_N		N		N	



Reserved	Pin Name	Reserved	Internal Pull Up at Power Up		Internal Pull Up at Nominal Active State	
			PUP	Comment	PUP	Comment
	OSC_SEL		Y		Y	
	POR_BYPASS		Y		N	

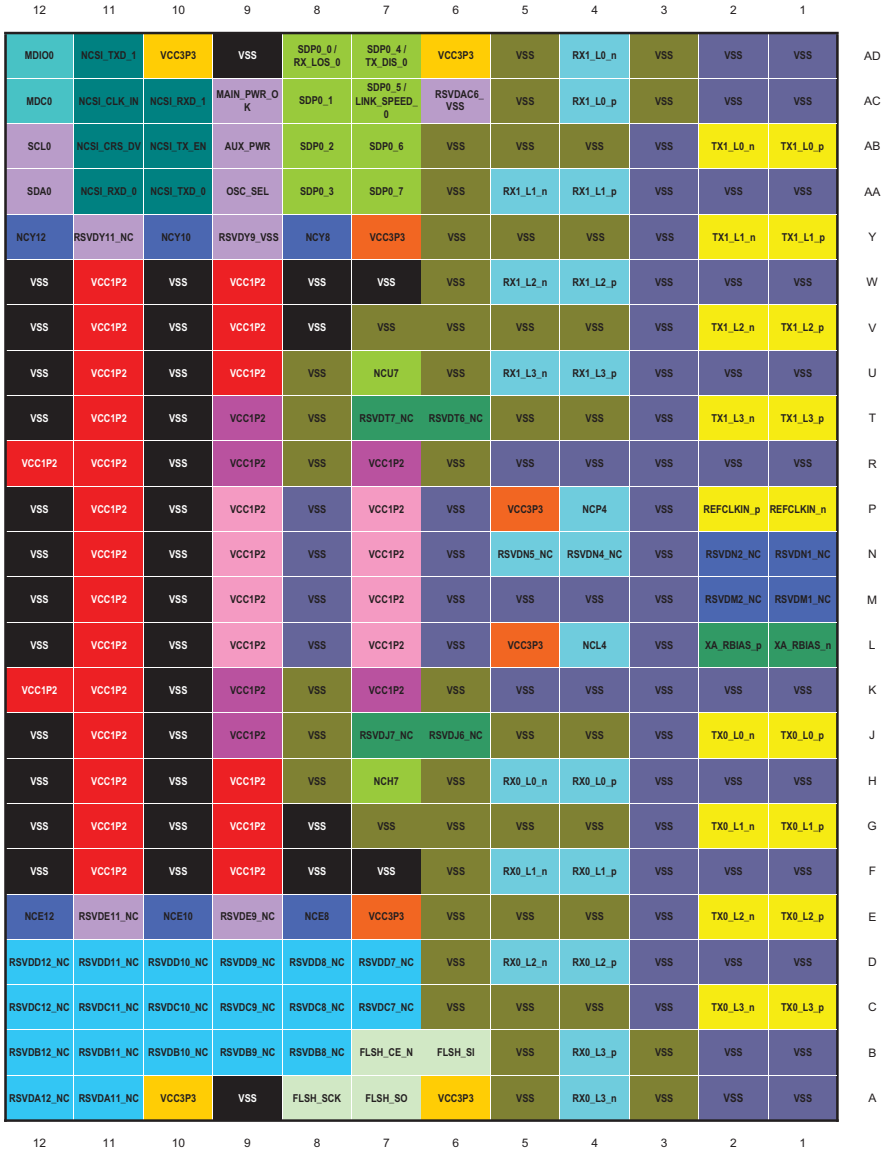
Note: Refer to the reference schematics for implementation details.

2.2 Ball Out — Top Level

Top view, through package.

	24	23	22	21	20	19	18	17	16	15	14	13
AD	VSS	VSS	VSS	LAN0_DIS_N	LAN1_DIS_N	VCC3P3	PE_RST_N	SCL1	VSS	VCC3P3	LED0_0	LED1_0
AC	VSS	VSS	VSS	PER_0_n	PER_0_p	SMBCLK	SDA1	MDIO1	SDP1_0 / RX_LOS_1	SDP1_5 / LINK_SPEED_1	LED0_1	LED1_1
AB	PE_CLK_n	PE_CLK_p	VSS	VSS	VSS	SMBD	MDC1	SDP1_2	SDP1_1	SDP1_6	LED0_2	LED1_2
AA	VSS	VSS	VSS	PER_1_n	PER_1_p	SMBALRT_N	PE_WAKE_N	SDP1_3	SDP1_4 / TX_DIS_1	SDP1_7	LED0_3	LED1_3
Y	PET_0_n	PET_0_p	VSS	VSS	VSS	VSS	RSVDY18_NC	RSVDY17_NC	NCY16	RSVDY15_NC	NCY14	RSVDY13_NC
W	VSS	VSS	VSS	RSVDW21_NC	RSVDW20_NC	VSS	VSS	VSS	RSVDW16_VSS	VSS	VCC1P2	VSS
V	PET_1_n	PET_1_p	VSS	VSS	VSS	VSS	VSS	VSS	RSVDV16_VSS	VSS	VCC1P2	VSS
U	VSS	VSS	VSS	PER_2_n	PER_2_p	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS
T	PET_2_n	PET_2_p	VSS	VSS	VSS	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS
R	VSS	VSS	VSS	PER_3_n	PER_3_p	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VCC1P2
P	PET_3_n	PET_3_p	VSS	VSS	VSS	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS
N	PE_RSENSE	VSS	VSS	RSVDN21_NC	RSVDN20_NC	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS
M	PE_RBIAS	VSS	VSS	RSVDM21_NC	RSVDM20_NC	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS
L	RSVDL24_NC	RSVDL23_NC	VSS	VSS	VSS	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS
K	VSS	VSS	VSS	PER_4_n	PER_4_p	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VCC1P2
J	PET_4_n	PET_4_p	VSS	VSS	VSS	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS
H	VSS	VSS	VSS	PER_5_n	PER_5_p	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS
G	PET_5_n	PET_5_p	VSS	VSS	VSS	VSS	VSS	VSS	VCC1P2	VSS	VCC1P2	VSS
F	VSS	VSS	VSS	RSVDF21_VSS	NCF20	VSS	VSS	VSS	VCC1P2	VSS	VCC1P2	VSS
E	PET_6_n	PET_6_p	VSS	VSS	VSS	VSS	NCE18	RSVDE17_VSS	NCE16	RSVDE15_NC	NCE14	RSVDE13_NC
D	VSS	VSS	VSS	PER_6_n	PER_6_p	POR_BYPASS	RSVDD18_NC	RSVDD17_NC	RSVDD16_NC	RSVDD15_NC	RSVDD14_NC	RSVDD13_NC
C	PET_7_n	PET_7_p	VSS	VSS	VSS	EE_CS_N	RSVDC18_NC	RSVDC17_NC	RSVDC16_NC	RSVDC15_NC	RSVDC14_NC	RSVDC13_NC
B	VSS	VSS	VSS	PER_7_n	PER_7_p	EE_SK	EE_DI	RSVDB17_NC	JTCK	JTDO	JRST_N	JTMS
A	VSS	VSS	VSS	RSVDA21_NC	RSVDA20_NC	VCC3P3	EE_DO	RSVDA17_NC	VSS	VCC3P3	LAN_PWR_GOOD	JTDI
	24	23	22	21	20	19	18	17	16	15	14	13

Figure 2-1 Package Layout - Left View



12	11	10	9	8	7	6	5	4	3	2	1	
MDIO0	NCSI_TXD_1	VCC3P3	VSS	SDP0_0 / RX_LOS_0	SDP0_4 / TX_DIS_0	VCC3P3	VSS	RX1_L0_n	VSS	VSS	VSS	AD
MDC0	NCSI_CLK_IN	NCSI_RXD_1	MAIN_PWR_OK	SDP0_1	SDP0_5 / LINK_SPEED_0	RSVDAC6_VSS	VSS	RX1_L0_p	VSS	VSS	VSS	AC
SCL0	NCSI_CRS_DV	NCSI_TX_EN	AUX_PWR	SDP0_2	SDP0_6	VSS	VSS	VSS	VSS	TX1_L0_n	TX1_L0_p	AB
SDA0	NCSI_RXD_0	NCSI_TXD_0	OSC_SEL	SDP0_3	SDP0_7	VSS	RX1_L1_n	RX1_L1_p	VSS	VSS	VSS	AA
NCY12	RSVDY11_NC	NCY10	RSVDY9_VSS	NCY8	VCC3P3	VSS	VSS	VSS	VSS	TX1_L1_n	TX1_L1_p	Y
VSS	VCC1P2	VSS	VCC1P2	VSS	VSS	VSS	RX1_L2_n	RX1_L2_p	VSS	VSS	VSS	W
VSS	VCC1P2	VSS	VCC1P2	VSS	VSS	VSS	VSS	VSS	VSS	TX1_L2_n	TX1_L2_p	V
VSS	VCC1P2	VSS	VCC1P2	VSS	NCU7	VSS	RX1_L3_n	RX1_L3_p	VSS	VSS	VSS	U
VSS	VCC1P2	VSS	VCC1P2	VSS	RSVDT7_NC	RSVDT6_NC	VSS	VSS	VSS	TX1_L3_n	TX1_L3_p	T
VCC1P2	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VSS	VSS	VSS	VSS	VSS	R
VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC3P3	NCP4	VSS	REFCLKIN_p	REFCLKIN_n	P
VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	RSVDN5_NC	RSVDN4_NC	VSS	RSVDN2_NC	RSVDN1_NC	N
VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VSS	VSS	VSS	RSVDM2_NC	RSVDM1_NC	M
VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC3P3	NCL4	VSS	XA_RBIAS_p	XA_RBIAS_n	L
VCC1P2	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VSS	VSS	VSS	VSS	VSS	K
VSS	VCC1P2	VSS	VCC1P2	VSS	RSVDJ7_NC	RSVDJ6_NC	VSS	VSS	VSS	TX0_L0_n	TX0_L0_p	J
VSS	VCC1P2	VSS	VCC1P2	VSS	NCH7	VSS	RX0_L0_n	RX0_L0_p	VSS	VSS	VSS	H
VSS	VCC1P2	VSS	VCC1P2	VSS	VSS	VSS	VSS	VSS	VSS	TX0_L1_n	TX0_L1_p	G
VSS	VCC1P2	VSS	VCC1P2	VSS	VSS	VSS	RX0_L1_n	RX0_L1_p	VSS	VSS	VSS	F
NCE12	RSVDE11_NC	NCE10	RSVDE9_NC	NCE8	VCC3P3	VSS	VSS	VSS	VSS	TX0_L2_n	TX0_L2_p	E
RSVDD12_NC	RSVDD11_NC	RSVDD10_NC	RSVDD9_NC	RSVDD8_NC	RSVDD7_NC	VSS	RX0_L2_n	RX0_L2_p	VSS	VSS	VSS	D
RSVDC12_NC	RSVDC11_NC	RSVDC10_NC	RSVDC9_NC	RSVDC8_NC	RSVDC7_NC	VSS	VSS	VSS	VSS	TX0_L3_n	TX0_L3_p	C
RSVDB12_NC	RSVDB11_NC	RSVDB10_NC	RSVDB9_NC	RSVDB8_NC	FLSH_CE_N	FLSH_SI	VSS	RX0_L3_p	VSS	VSS	VSS	B
RSVDA12_NC	RSVDA11_NC	VCC3P3	VSS	FLSH_SCK	FLSH_SO	VCC3P3	VSS	RX0_L3_n	VSS	VSS	VSS	A
12	11	10	9	8	7	6	5	4	3	2	1	

Figure 2-2 Package Layout - Right View

NOTE: ***This page intentionally left blank.***

3.0 Interconnects

3.1 PCI-Express* (PCIe*)

3.1.1 Overview

PCIe is an I/O architecture that enables cost competitive solutions as well as provide industry leading price/performance and feature richness. It is an industry-driven specification.

PCIe defines a basic set of requirements that addresses the majority of the targeted application classes. Higher-end applications' requirements (Enterprise class servers and high-end communication platforms) are addressed by a set of advanced extensions that complement the baseline requirements.

To guarantee headroom for future applications, PCIe provides a software-managed mechanism for introducing new, enhanced capabilities.

Figure 3-1 shows the PCIe architecture.

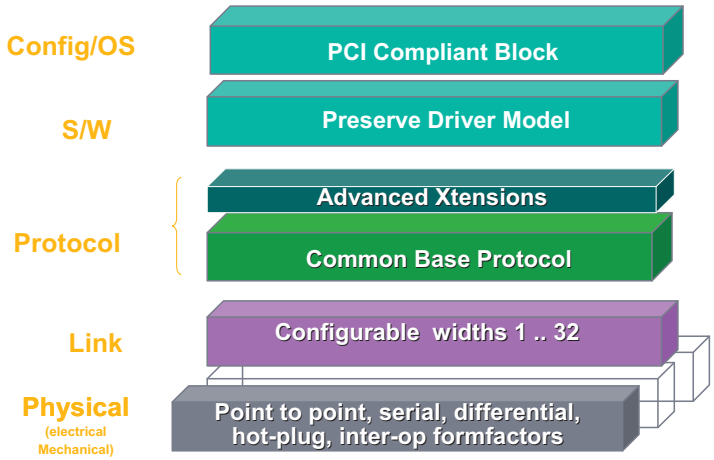


Figure 3-1 PCIe Stack Structure

The PCIe physical layer consists of a differential transmit pair and a differential receive pair. Full-duplex data on these two point-to-point connections is self-clocked such that no dedicated clock signals are required. The bandwidth of this interface increases in direct proportion with frequency increases.

The packet is the fundamental unit of information exchange and the protocol includes a message space to replace a variety of side-band signals found on previous interconnects. This movement of hard-wired signals from the physical layer to messages within the transaction layer enables easy and linear physical layer width expansion for increased bandwidth.

The common base protocol uses split transactions along with several mechanisms to eliminate wait states and to optimize the re-ordering of transactions to further improve system performance.

3.1.1.1 Architecture, Transaction and Link Layer Properties

- Split transaction, packet-based protocol
- Common flat address space for load/store access (for example, PCI addressing model)
  - 32-bit memory address space to enable a compact packet header (must be used to access addresses below 4 GB)
  - 64-bit memory address space using an extended packet header
- Transaction layer mechanisms:
  - PCI-X style relaxed ordering
- Credit-based flow control
- Packet sizes/formats:
  - Maximum packet size: 512 bytes
  - Maximum read request size: 2 KB
- Reset/initialization:
  - Frequency/width/profile negotiation performed by hardware
- Data integrity support
  - Using CRC-32 for Transaction layer Packets (TLP)
- Link Layer Retry (LLR) for recovery following error detection
  - Using CRC-16 for Link Layer (LL) messages
- No retry following error detection
  - 8b/10b encoding with running disparity
- Software configuration mechanism:
  - Uses PCI configuration and bus enumeration model
  - PCIe-specific configuration registers mapped via PCI extended capability mechanism

- Baseline messaging:
  - In-band messaging of formerly side-band legacy signals (interrupts, etc.)
  - System-level power management supported via messages
- Power management:
  - Full support for PCIm
  - Wake capability from D3cold state
  - Compliant with ACPI, PCIm software model
  - Active state power management
- Support for PCIe V2.0 (2.5GT/s or 5GT/s)
  - Support for completion time out control
  - Support for additional registers in the PCIe capability structure

3.1.1.2 Physical Interface Properties

- Point-to-point interconnect
  - Full-duplex; no arbitration
- Signaling technology:
  - Low Voltage Differential (LVD)
  - Embedded clock signaling using 8b/10b encoding scheme
- Serial frequency of operation: PCIe V2.0 (2.5GT/s or 5GT/s).
- Interface width of 1, 2, 4, or 8 PCIe lanes.
- DFT and DFM support for high-volume manufacturing

3.1.1.3 Advanced Extensions

PCIe defines a set of optional features to enhance platform capabilities for specific usage modes. The 82599 supports the following optional features:

- Advanced Error Reporting (AER) — Messaging support to communicate multiple types/severity of errors
- Device Serial Number — Allows exposure of a unique serial number for each device
- Alternative RID Interpretation (ARI) — allows support of more than eight functions per device
- Single Root I/O Virtualization (SR-IOV) — allows exposure of virtual functions controlling a subset of the resources to Virtual Machines (VMs)

3.1.2 General Functionality

3.1.2.1 Native/Legacy

All 82599 PCI functions are native PCIe functions.

3.1.2.2 Locked Transactions

The 82599 does not support locked requests as a target or a master.

3.1.3 Host Interface

PCIe device numbers identify logical devices within the physical device (the 82599 is a physical device). The 82599 implements a single logical device with two separate PCI Functions: LAN 0 and LAN 1. The device number is captured from each type 0 configuration write transaction.

Each of the PCIe functions interfaces with the PCIe unit through one or more clients. A client ID identifies the client and is included in the *Tag* field of the PCIe packet header. Completions always carry the tag value included in the request to enable routing of the completion to the appropriate client.

3.1.3.1 TAG ID Allocation

Tag IDs are allocated differently for read and write as detailed in the following sections.

3.1.3.1.1 TAG ID Allocation for Read Transactions

Table 3-1 lists the Tag ID allocation for read accesses. The Tag ID is used by hardware in order to be able to forward the read data to the required internal client.

Table 3-1 TAG ID Allocation Table for Read Transactions

TAG ID	Description	TAG ID	Description
0x0	Data Request 0x0	0x10	Tx Descriptor 0
0x1	Data Request 0x1	0x11	Tx Descriptor 1
0x2	Data Request 0x2	0x12	Tx Descriptor 2
0x3	Data Request 0x3	0x13	Tx Descriptor 3
0x4	Data Request 0x4	0x14	Tx Descriptor 4
0x5	Data Request 0x5	0x15	Tx Descriptor 5
0x6	Data Request 0x6	0x16	Tx Descriptor 6
0x7	Data Request 0x7	0x17	Tx Descriptor 7

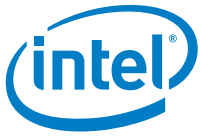


Table 3-1 TAG ID Allocation Table for Read Transactions (Continued)

TAG ID	Description	TAG ID	Description
0x8	Data Request 0x8	0x18	Rx Descriptor 0
0x9	Data Request 0x9	0x19	Rx Descriptor 1
0xA	Data Request 0xA	0x1A	Rx Descriptor 2
0xB	Data Request 0xB	0x1B	Rx Descriptor 3
0xC	Data Request 0xC	0x1C	Rx Descriptor 4
0xD	Data Request 0xD	0x1D	Rx Descriptor 5
0xE	Data Request 0xE	0x1E	Rx Descriptor 6
0xF	Data Request 0xF	0x1F	Rx Descriptor 7

3.1.3.1.2 TAG ID Allocation for Write Transactions

Request tag allocation depends on these system parameters:

- DCA supported or not supported in the system (DCA_CTRL.DCA_DIS)
- DCA enabled or disabled (DCA_TXCTRL.TX Descriptor DCA EN, DCA_RXCTRL.RX Descriptor DCA EN, DCA_RXCTRL.RX Header DCA EN, DCA_RXCTRL.Rx Payload DCA EN)
- System type: Legacy DCA versus DCA 1.0 (DCA_CTRL.DCA_MODE)
- CPU ID (DCA_RXCTRL.CPUID or DCA_TXCTRL.CPUID)

Case 1 — DCA Disabled in the System:

The following table lists the write requests tags:

Tag ID	Description
2	Write-back descriptor Tx /write-back head.
4	Write-back descriptor Rx.
6	Write data.
30	MSI and MSI-X.

Case 2 — DCA Enabled in the System, but Disabled for the Request:

- Legacy DCA platforms — If DCA is disabled for the request, the tags allocation is identical to the case where DCA is disabled in the system (refer to the previous table).
- DCA 1.0 platforms — All write requests have the tag of 0x00.

Case 3 — DCA Enabled in the System, DCA Enabled for the Request:

- Legacy DCA Platforms: the request tag is constructed as follows:
  - Bit[0] — DCA Enable = 1b
  - Bits[3:1] — The *CPU ID* field taken from the CPUID[2:0] bits of the DCA_RXCTRL or DCA_TXCTRL registers

— Bits[7:4] — Reserved

- DCA 1.0 Platforms: the request tag (all eight bits) is taken from the *CPU ID* field of the DCA_RXCTRL or DCA_TXCTRL registers

3.1.3.2 Completion Timeout Mechanism

In any split transaction protocol, there is a risk associated with the failure of a requester to receive an expected completion. To enable requesters to attempt recovery from this situation in a standard manner, the completion timeout mechanism is defined.

The completion timeout mechanism is activated for each request that requires one or more completions when the request is transmitted. The 82599 provides a programmable range for the completion timeout, as well as the ability to disable the completion timeout altogether. The completion timeout is programmed through an extension of the PCIe capability structure.

The 82599's reaction to a completion timeout is listed in Table 3-8.

The 82599 controls the following aspects of completion timeout:

- Disabling or enabling completion timeout
- Disabling or enabling resending a request on completion timeout
- A programmable range of timeout values
- Programming the behavior of completion timeout is listed in Table 3-2. Note that system software can configure a completion timeout independently per each LAN function.

Table 3-2 Completion Timeout Programming

Capability	Programming Capability
Completion Timeout Enabling	Controlled through PCI configuration. Visible through a read-only CSR bit.
Resend Request Enable	Loaded from the EEPROM into a read-only CSR bit.
Completion Timeout Period	Controlled through PCI configuration.

Completion Timeout Enable — Programmed through the PCI configuration space. The default is: Completion Timeout Enabled.

Resend Request Enable — The *Completion Timeout Resend* EEPROM bit (loaded to the *Completion_Timeout_Resend* bit in the PCIe Control Register (GCR) enables resending the request (applies only when completion timeout is enabled). The default is to resend a request that timed out.

3.1.3.2.1 Completion Timeout Period

Programmed through the PCI configuration. Visible through bits 3:0 in the Device Capabilities 2 Register (0xC4; RO) register (see Section 9.3.10.10). The 82599 supports all four ranges defined by PCIe V2.0 (2.5GT/s or 5GT/s):

- 50 µs to 10 ms
- 10 ms to 250 ms
- 250 ms to 4 s
- 4 s to 64 s

System software programs a range (one of nine possible ranges that sub-divide the four previous ranges) into the PCI configuration register. The supported sub-ranges are:

- 50 µs to 50 ms (default).
- 50 µs to 100 µs
- 1 ms to 10 ms
- 16 ms to 55 ms
- 65 ms to 210 ms
- 260 ms to 900 ms
- 1 s to 3.5 s
- 4 s to 13 s
- 17 s to 64s

A memory read request for which there are multiple completions are considered completed only when all completions have been received by the requester. If some, but not all, requested data is returned before the completion timeout timer expires, the requestor is permitted to keep or to discard the data that was returned prior to timer expiration.

3.1.4 Transaction Layer

The upper layer of the PCIe architecture is the transaction layer. The transaction layer connects to 82599's core using an implementation-specific protocol. Through this core-to-transaction-layer protocol, the application-specific parts of the 82599 interact with the PCIe subsystem and transmits and receives requests to or from the remote PCIe agent, respectively.

3.1.4.1 Transaction Types Accepted by the 82599

Table 3-3 Transaction Types Accepted by the Transaction Layer

Transaction Type	FC Type	Tx Layer Reaction	Hardware Should Keep Data From Original Packet	For Client
Configuration Read Request	NPH	CPLH + CPLD	Requester ID, TAG, attribute	Configuration space
Configuration Write Request	NPH + NPD	CPLH	Requester ID, TAG, attribute	Configuration space
Memory Read Request	NPH	CPLH + CPLD	Requester ID, TAG, attribute	CSR space
Memory Write Request	PH + PD	-	-	CSR space
IO Read Request	NPH	CPLH + CPLD	Requester ID, TAG, attribute	CSR space
IO Write Request	NPH + NPD	CPLH	Requester ID, TAG, attribute	CSR space
Read Completions	CPLH + CPLD	-	-	DMA
Message	PH	-	-	Message unit/INT/ PM/ error unit

Flow Control Types Legend:

CPLD — Completion Data Payload

CPLH — Completion Headers

NPD — Non-Posted Request Data Payload

NPH — Non-Posted Request Headers

PD — Posted Request Data Payload

PH — Posted Request Headers

3.1.4.2 Transaction Types Initiated by the 82599

Table 3-4 Transaction Types Initiated by the Transaction Layer

Transaction type	Payload Size	FC Type	From Client
Configuration Read Request Completion	Dword	CPLH + CPLD	Configuration space
Configuration Write Request Completion	-	CPLH	Configuration space
IO Read Request Completion	Dword	CPLH + CPLD	CSR
IO Write Request Completion	-	CPLH	CSR
Read Request Completion	Dword/Qword	CPLH + CPLD	CSR
Memory Read Request	-	NPH	DMA
Memory Write Request	<= MAX_PAYLOAD_SIZE	PH + PD	DMA
Message	-	PH	Message unit/INT/PM/ error unit

Note: MAX_PAYLOAD_SIZE is loaded from the EEPROM (up to 512 bytes). Effective MAX_PAYLOAD_SIZE is defined for each PCI function according to the configuration space register for that function.

3.1.4.2.1 Data Alignment

Note: Requests must never specify an address/length combination that causes a memory space access to cross a 4 KB boundary.

The 82599 breaks requests into 4 KB-aligned requests (if needed). This does not pose any requirement on software. However, if software allocates a buffer across a 4 KB boundary, hardware issues multiple requests for the buffer. Software should consider aligning buffers to a 4 KB boundary in cases where it improves performance.

The general rules for packet alignment are as follows. Note that these apply to all the 82599 requests (read/write):

- The length of a single request does not exceed the PCIe limit of MAX_PAYLOAD_SIZE for write and MAX_READ_REQ for read.
- The length of a single request does not exceed the 82599 internal limitations.
- A single request does not span across different memory pages as noted by the 4 KB boundary alignment previously mentioned.

If a request can be sent as a single PCIe packet and still meet the general rules for packet alignment, then it is not broken at the cache line boundary but rather sent as a single packet (motivation is that the chipset can break the request along cache line boundaries, but the 82599 should still benefit from better PCIe use). However, if any of the three general rules require that the request is broken into two or more packets, then the request is broken at the cache line boundary.

3.1.4.2.2 Multiple Tx Data Read Requests (MULR)

The 82599 supports 16 multiple pipelined requests for transmit data. In general, requests can belong to the same packet or to consecutive packets. However, the following restrictions apply:

- All requests for a packet must be issued before a request is issued for a consecutive packet.
- Read requests can be issued from any of the supported queues, as long as the previous restriction is met. Pipelined requests can belong to the same queue or to separate queues. However, as previously noted, all requests for a certain packet are issued (from the same queue) before a request is issued for a different packet (potentially from a different queue).
- The PCIe specification does not insure that completions for separate requests return in-order. Read completions for concurrent requests are not required to return in the order issued. The 82599 handles completions that arrive in any order. Once all completions arrive for a given request, it can issue the next pending read data request.
- The 82599 incorporates a reorder buffer to support re-ordering of completions for all issued requests. Each request/completion can be up to 512 bytes long. The maximum size of a read request is defined as the minimum {2 KB bytes, Max_Read_Request_Size}.
- In addition to the transmit data requests, the 82599 can issue eight pipelined read requests for Tx descriptors and eight pipelined read requests for Rx descriptors. The requests for Tx data, Tx descriptors, and Rx descriptors are independently issued.

3.1.4.3 Messages

3.1.4.3.1 Received Messages

Message packets are special packets that carry a message code. The upstream device transmits special messages to the 82599 by using this mechanism. The transaction layer decodes the message code and responds to the message accordingly.

Table 3-5 Supported Message in the 82599 - as a Receiver

Message Code [7:0]	Routing r2r1r0	Message	82599 Later Response
0x14	100b	PM_Active_State_NAK	Internal Signal Set
0x19	011b	PME_Turn_Off	Internal Signal Set
0x50	100b	Slot power limit support (has one Dword data)	Silently Drop
0x7E	010b, 011b,100b	Vendor_defined type 0 No data	Unsupported Request
0x7E	010b,011b,100b	Vendor_defined type 0 data	Unsupported Request

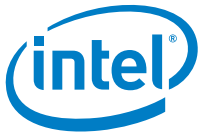


Table 3-5 Supported Message in the 82599 - as a Receiver (Continued)

Message Code [7:0]	Routing r2r1r0	Message	82599 Later Response
0x7F	010b,011b,100b	Vendor_defined type 1 no data	Silently Drop
0x7F	010b, 011b,100b	Vendor_defined type 1 data	Silently Drop
0x00	011b	Unlock	Silently Drop

3.1.4.3.2 Transmitted Messages

The transaction layer is also responsible for transmitting specific messages to report internal/external events (such as interrupts and PMEs).

Table 3-6 Supported Message in the 82599 - as a Transmitter

Message code [7:0]	Routing r2r1r0	Message
0x20	100b	Assert INT A
0x21	100b	Assert INT B
0x22	100b	Assert INT C
0x23	100b	Assert INT D
0x24	100b	DE- Assert INT A
0x25	100b	DE- Assert INT B
0x26	100b	DE- Assert INT C
0x27	100b	DE- Assert INT D
0x30	000b	ERR_COR
0x31	000b	ERR_NONFATAL
0x33	000b	ERR_FATAL
0x18	000b	PM_PME
0x1B	101b	PME_TO_Ack

3.1.4.4 Ordering Rules

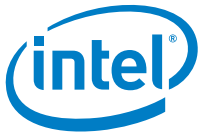
The 82599 meets the PCIe ordering rules by following the PCI simple device model:

1. Deadlock Avoidance – The 82599 meets the PCIe ordering rules that prevent deadlocks:
   a. Posted writes overtake stalled read requests. This applies to both target and master directions. For example, if master read requests are stalled due to lack of credits, master posted writes are allowed to proceed. On the target side, it is acceptable to timeout on stalled read requests in order to allow later posted writes to proceed.
   b. Target posted writes overtake stalled target configuration writes.
   c. Completions overtake stalled read requests. This applies to both target and master directions. For example, if master read requests are stalled due to lack of credits, completions generated by the 82599 are allowed to proceed.
2. Descriptor/Data Ordering — The 82599 insures that a Rx descriptor is written back on PCIe only after the data that the descriptor relates to is written to the PCIe link.
3. MSI and MSI-X Ordering Rules – System software can change the MSI or MSI-X tables during run-time. Software expects that interrupt messages issued after the table has been updated are using the updated contents of the tables.
   a. Since software doesn't know when the tables are actually updated in the 82599, a common scheme is to issue a read request to the MSI or MSI-X table (a PCI configuration read for MSI and a memory read for MSI-X). Software expects that any message issued following the completion of the read request, is using the updated contents of the tables.
   b. Once an MSI or MSI-X message is issued using the updated contents of the interrupt tables, any consecutive MSI or MSI-X message does not use the contents of the tables prior to the change.
4. The 82599 meets the rules relating to independence between target and master accesses:
   a. The acceptance of a target posted request does not depend upon the transmission of any TLP.
   b. The acceptance of a target Non-posted Request does not depend upon the transmission of a non-posted request.
   c. Accepting a completion does not depend upon the transmission of any TLP.

3.1.4.4.1 Out of Order Completion Handling

In a split transaction protocol, when using multiple read requests in a multi-processor environment, there is a risk that completions for separate requests arrive from the host memory out of order and interleaved. In this case, the 82599 sorts the completions and transfers them to the network in the correct order.

Note: Completions for separate read requests are not guaranteed to return in order. Completions for the same read request are guaranteed to return in address order.



3.1.4.5 Transaction Definition and Attributes

3.1.4.5.1 Max Payload Size

The 82599's policy for determining Max Payload Size (MPS) is as follows:

1. Master requests initiated by the 82599 (including completions) limit Max Payload Size to the value defined for the function issuing the request.
2. Target write accesses to the 82599 are accepted only with a size of one Dword or two Dwords. Write accesses in the range of three Dwords (MPS) are flagged as unreliable. Write accesses above MPS are flagged as malformed.

3.1.4.5.2 Traffic Class (TC) and Virtual Channels (VC)

The 82599 only supports TC = 0 and VC = 0 (default).

3.1.4.5.3 Relaxed Ordering

The 82599 takes advantage of the relaxed ordering rules in PCIe. By setting the relaxed ordering bit in the packet header, the 82599 enables the system to optimize performance in the following cases:

1. Relaxed ordering for descriptor and data reads — When the 82599 masters a read transaction, its split completion has no ordering relationship with the writes from the CPUs (same direction). It should be allowed to bypass the writes from the CPUs.
2. Relaxed ordering for receiving data writes — When the 82599 masters receive data writes, it also enables them to bypass each other in the path to system memory because software does not process this data until their associated descriptor writes are done.
3. The 82599 cannot relax ordering for descriptor writes or an MSI write.

Relaxed ordering is enabled globally in the 82599 by clearing the CTRL_EXT.RO_DIS bit and further enabled per queue in the DCA_RXCTRL[n] registers.

3.1.4.6 Flow Control

3.1.4.6.1 Flow Control Rules

The 82599 only implements the default Virtual Channel (VC0). A single set of credits is maintained for VC0.

Table 3-7 Flow Control Credits Allocation

Credit Type	Operations	Number of Credits (dual port)
Posted Request Header (PH)	Target write Message (one unit)	**16** credit units to support tail write at wire speed.
Posted Request Data (PD)	Target Write (Length/16 bytes = one) Message (one unit)	max{MAX_PAYLOAD_SIZE/16, 32}.
Non-Posted Request Header (NPH)	Target read (one unit) Configuration read (one unit) Configuration write (one unit)	**Four** units (to enable concurrent target accesses to both LAN ports).
Non-Posted Request Data (NPD)	Configuration write (one unit)	**Four** units.
Completion Header (CPLH)	Read completion (n/a)	Infinite (accepted immediately).
Completion Data (CPLD)	Read completion (n/a)	Infinite (accepted immediately).

Rules for FC updates:

- The 82599 maintains two credits for NPD at any given time. It increments the credit by one after the credit is consumed, and sends an UpdateFC packet as soon as possible. UpdateFC packets are scheduled immediately after a resource is available.
- The 82599 provides **16** credits for PH (such as for concurrent target writes) and four credits for NPH (such as for four concurrent target reads). UpdateFC packets are scheduled immediately after a resource is available.
- The 82599 follows the PCIe recommendations for frequency of UpdateFC FCPs.

3.1.4.6.2 Upstream Flow Control Tracking

The 82599 issues a master transaction only when the required flow control credits are available. Credits are tracked for posted, non-posted, and completions (the later to operate against a switch).

3.1.4.6.3 Flow Control Update Frequency

In all cases, Update Flow Control Packets (FCPs) are scheduled immediately after a resource is available.

When the link is in the L0 or L0s link state, Update FCPs for each enabled type of non-infinite flow control credit must be scheduled for transmission at least once every 30 µs (-0% /+50%), except when the Extended Sync bit of the Control Link register is set, in which case the limit is 120 µs (-0% /+50%).

3.1.4.6.4 Flow Control Timeout Mechanism

The 82599 implements the optional flow control update timeout mechanism.

The mechanism is active when the link is in L0 or L0s link state. It uses a timer with a limit of 200 µs (-0% /+50%), where the timer is reset by the receipt of any Init or Update FCP. Alternately, the timer can be reset by the receipt of any DLLP.

Upon timer expiration, the mechanism instructs the PHY to retrain the link (via the LTSSM recovery state).

3.1.5 Link Layer

3.1.5.1 ACK/NAK Scheme

The 82599 supports two alternative schemes for ACK/NAK rate:

- ACK/NAK is scheduled for transmission following any TLP.
- ACK/NAK is scheduled for transmission according to timeouts specified in the PCIe specification.

The *PCIe Error Recovery* bit (loaded from the EEPROM) determines which of the two schemes is used.

3.1.5.2 Supported DLLPs

The following DLLPs are supported by the 82599 as a receiver:

- ACK
- NAK
- PM_Request_Ack
- InitFC1-P
- InitFC1-NP
- InitFC1-Cpl
- InitFC2-P
- InitFC2-NP
- InitFC2-Cpl
- UpdateFC-P
- UpdateFC-NP
- UpdateFC-Cpl

The following DLLPs are supported by the 82599 as a transmitter:

- ACK
- NAK

- PM_Enter_L1
- PM_Enter_L23
- InitFC1-P
- InitFC1-NP
- InitFC1-Cpl
- InitFC2-P
- InitFC2-NP
- InitFC2-Cpl
- UpdateFC-P
- UpdateFC-NP

Note: UpdateFC-Cpl is not sent because of the infinite FC-Cpl allocation.

3.1.5.3 Transmit EDB Nullifying (End Bad)

If retrain is necessary, there is a need to guarantee that no abrupt termination of the Tx packet happens. For this reason, early termination of the transmitted packet is possible. This is done by appending the EDB to the packet.

3.1.6 Physical Layer

3.1.6.1 Link Speed

The 82599 supports PCIe V2.0 (2.5GT/s or 5GT/s). The following configuration controls link speed:

- PCIe *Supported Link Speeds* bit — Indicates the link speeds supported by the 82599. Loaded from the PCIe *Link Speed* field in the EEPROM.

EEPROM Word Offset (Starting at Odd Word)	Allow PCIe V2.0(Default)	Force PCIe V2.0 Setting	Description
2*N+1	0x094		MORIA6 register offset (lower word).
2*N+2	0x0000	0x0100	Disabling PCIe V2.0 is controlled by setting bit[8] in this register. When the bit is set the 82599 does not advertise PCIe V2.0 link-speed support.

- PCIe *Current Link Speed* bit — Indicates the negotiated Link speed.
- PCIe *Target Link Speed* bit — used to set the target compliance mode speed when software is using the *Enter Compliance* bit to force a link into compliance mode. The default value is the highest link speed supported defined by the previous *Supported Link Speeds*.

The 82599 does not initiate a hardware autonomous speed change.

The 82599 supports entering compliance mode at the speed indicated in the *Target Link Speed* field in the PCIe Link Control 2 register. Compliance mode functionality is controlled via the PCIe Link Control 2 register.

3.1.6.2 Link Width

- The 82599 supports a maximum link width of x8, x4, x2, or x1 as determined by the PCIe Analog Configuration Module in the EEPROM and can be set as follows. Note that these settings might not be needed during normal operation:

EEPROM Word Offset (Starting at Odd Word)	Enable x8 Setting (Default)	Limit to x4 Setting	Limit to x2 Setting	Limit to x1 Setting	Description
2*N+1	0x094				MORIA6 register offset (lower word).
2*N+2	0x0000	0x00F0	0x00FC	0x00FE	Lanes can be disabled by setting bits[7:0] in this offset. Having bit[X] set causes laneX to be disabled, resulting in narrower link widths (bits per lane).

The maximum link width is loaded into the *Max Link Width* field of the PCIe Capability register (LCAP[11:6]). Hardware default is the x8 link.

During link configuration, the platform and the 82599 negotiate on a common link width. The link width must be one of the supported PCIe link widths (x1, 2x, x4, x8), such that:

- If Maximum Link Width = x8, then the 82599 negotiates to either x8, x4, x2 or x1[1]
- If Maximum Link Width = x4, then the 82599 negotiates to either x4 or x1
- If Maximum Link Width = x1, then the 82599 only negotiates to x1

The 82599 does not initiate a hardware autonomous link width change.

Note: Some PCIe x8 slots are actually configured as x4 slots. These slots have insufficient bandwidth for full 10 GbE line rate with dual port 10 GbE devices. If a solution suffers bandwidth issues when both 10 GbE ports are active, it is recommended to verify that the PCIe slot is indeed a true PCIe x8.

3.1.6.3 Polarity Inversion

If polarity inversion is detected, the receiver must invert the received data.

During the training sequence, the receiver looks at symbols 6-15 of TS1 and TS2 as the indicators of lane polarity inversion (D+ and D- are swapped). If lane polarity inversion occurs, the TS1 symbols 6-15 received are D21.5 as opposed to the expected D10.2. Similarly, if lane polarity inversion occurs, symbols 6-15 of the TS2 ordered set are D26.5 as opposed to the expected 5 D5.2. This provides the clear indication of lane polarity inversion.

1. See restriction in Section 3.1.6.6.

3.1.6.4 L0s Exit Latency

The number of FTS sequences (N_FTS) sent during L0s exit is loaded from the EEPROM into an 8-bit read-only register.

3.1.6.5 Lane-to-Lane De-Skew

A multi-lane link can have many sources of lane-to-lane skew. Although symbols are transmitted simultaneously on all lanes, they cannot be expected to arrive at the receiver without lane-to-lane skew. The lane-to-lane skew can include components, which are less than one bit time, bit time units (400/200 ps for 2.5/5 Gb), or full symbol time units (4/2 ns). This type of skew is caused by the retiming repeaters' insert/delete operations. Receivers use TS1 or TS2 or Skip Ordered Sets (SOS) to perform link de-skew functions.

The 82599 supports de-skew of up to 12 symbols time [48 ns for PCIe v2.0 (2.5GT/s) and 24 ns for PCIe V2.0 (5GT/s)].

3.1.6.6 Lane Reversal

Auto lane reversal is supported by the 82599 at its hardware default setting. The following lane reversal modes are supported:

- Lane configurations x8, x4, x2, and x1
- Lane reversal in x8 and in x4
- Degraded mode (downshift) from x8 to x4 to x2 to x1 and from x4 to x1, with one restriction — if lane reversal is executed in x8, then downshift is only to x1 and not to x4.

Figure 3-2 through Figure 3-5 shows the lane downshift in both regular and reversal connections as well as lane connectivity from a system level perspective.

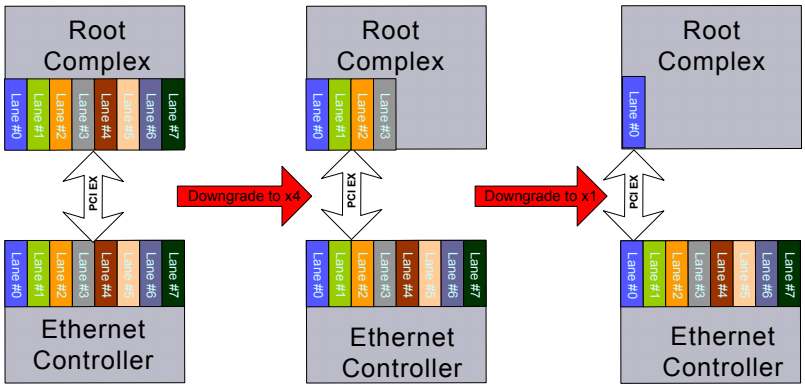


Figure 3-2 Lane Downshift in an x8 Configuration