IDT's PCIe® Gen2 Switch Family Non-Transparent Operations

Application Note
AN-707

Notes

Overview

The IDT family of Gen2 PCI Express® switches are designed for high-performance applications, supporting multiple simultaneous peer-to-peer traffic flows. Target applications include multi-host or intelligent I/O based systems where inter-domain communication is required, such as servers, storage, communications, and embedded systems.

This application note discusses the non-transparent bridging (NTB) feature of these switches.

Switch Partitioning

The logical view of a PCI Express switch is shown in Figure 1. A PCI Express switch contains one upstream port and one or more downstream ports. Each port is associated with a PCI-to-PCI (P2P) bridge. All PCI-to-PCI bridges associated with a PCI Express switch are interconnected by a virtual PCI bus.

The primary side of the upstream port's PCI-to-PCI bridge is associated with the external link, while the secondary side connects to the virtual PCI bus. The primary side of a downstream port's PCI-to-PCI bridge is connected to the virtual PCI bus, while the secondary side is associated with the external link.

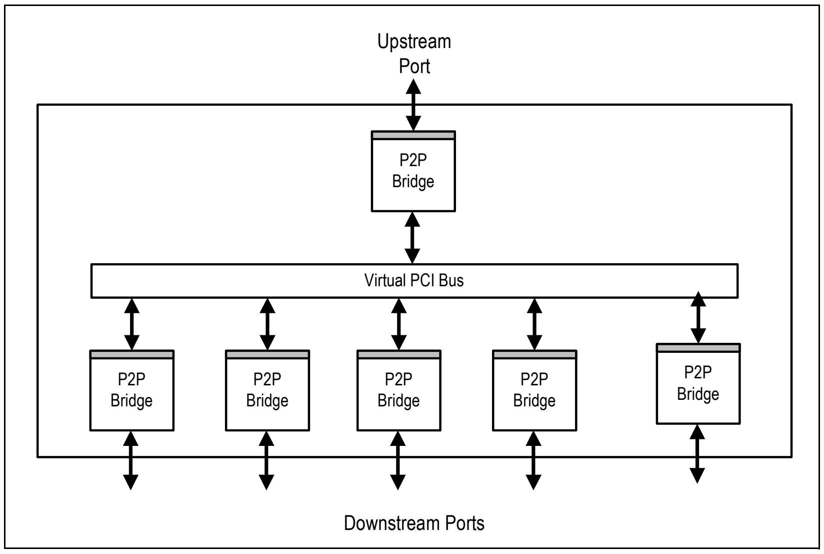


Figure 1 Transparent PCI Express Switch

The switch is a *partitionable* PCI Express switch. This means that in addition to operating as a standard PCI Express switch, the switch ports may be partitioned into groups that logically operate as completely independent PCI Express switches. Figure 2 illustrates a three partition configuration.

Notes

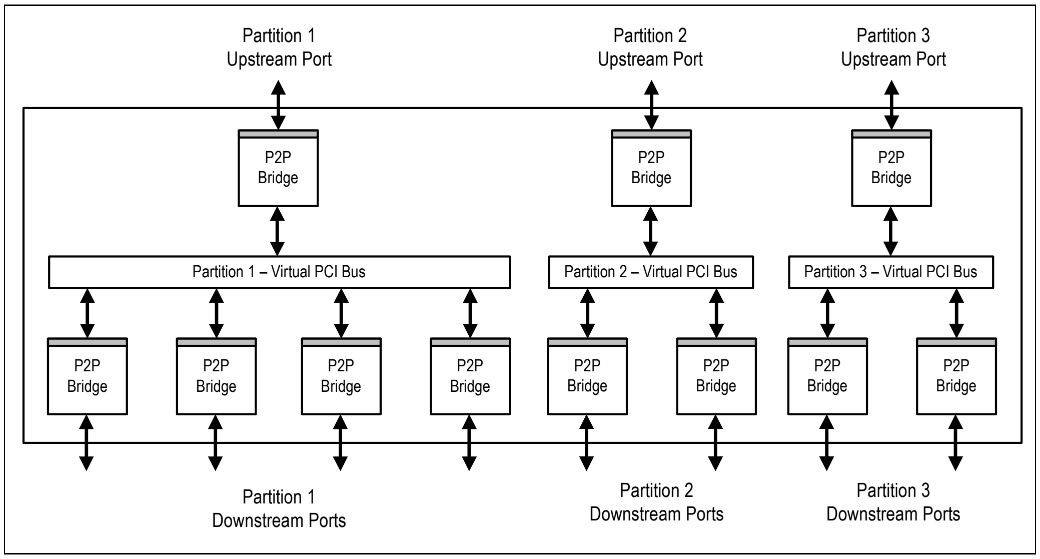


Figure 2 Example of Switch Partitioning

Each partition operates logically as a completely independent PCI Express switch that implements the behavior and capabilities outlined in the PCI Express Base specification required of a switch.

The switch supports boot-time (i.e., Fundamental Reset) and runtime configuration of ports into partitions. Boot time configuration may be performed via serial EEPROM, external SMBus master, or software executing on a root port (e.g., BIOS, OS, driver or hypervisor). Runtime reconfiguration allows the number of active partitions in the device and assignment of ports to partitions to be modified while the system is active. Runtime reconfiguration does not affect partitions whose configuration is not modified.

The PCI Express architectural model is one in which a root, typically the main CPU, is responsible for configuring a tree of endpoints (i.e., a hierarchy of virtual PCI buses). Once configured, any endpoint or root may initiate transactions. The root and endpoints share a common address space with routing configured in PCI-PCI bridges.

A limitation of the PCI Express architectural model is that it allows only a single root and that the root and all of the endpoints must share a common address space. This limitation may be overcome through the use of a *non-transparent bridge (NTB)*. A non-transparent bridge allows two roots or PCI Express trees (i.e., hierarchies) to be interconnected with one or more shared address windows between them.

The switch supports eight non-transparent functions (a.k.a., NT functions or NT endpoints). Each NT function appears as a PCI Express endpoint in the PCI Express hierarchy. The NT function is located in a partition's upstream port. In each partition, the operating mode of the switch's upstream port determines if this port contains an NT function. A port configured to operate in one of the following modes contains an NT function:

– NT function
– NT with DMA function
– Upstream switch port with NT function
– Upstream switch port with NT and DMA functions

Non-Transparent Operation

The PCI architecture defines a hierarchy of buses interconnected by PCI-to-PCI bridges. This hierarchy forms a tree and is referred to as a PCI domain.

– A PCI domain consists of a single memory address space, I/O address space, and ID address space.
– The PCI ID consists of a bus, device and function number that uniquely defines an element in the domain.

Notes

Although PCI Express switches support direct transfers between ports, the logical view seen by software remains that of a hierarchy of buses as defined by the PCI architecture and illustrated in Figure 1. The portion of a PCI domain emanating from a PCI Express root port is referred to as the PCI Express domain.

In many applications a need exists to interconnect two independent PCI domains. A Non-Transparent Bridge (NTB) enables this inter-domain communications. The architecture of an NTB is illustrated in Figure 3.

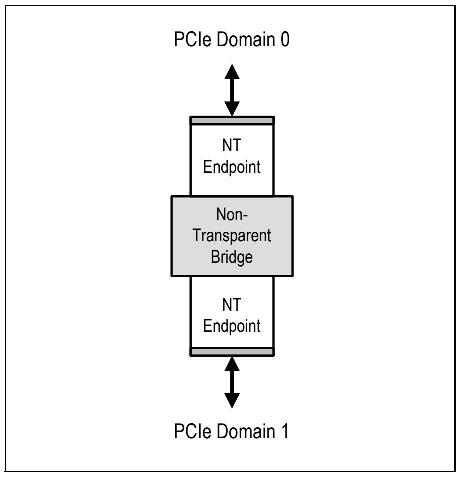


Figure 3 Non-Transparent Bridge

An NTB consists of two PCI functions each defined by a Type 0 PCI header that are interconnected by a bridging function. The two Type 0 PCI functions are referred two as Non-Transparent (NT) endpoints. Each function advertises one or more memory windows using PCI Base Address Registers (BARs). Software executing on each hierarchy allocates PCI memory space to the BAR.

Memory operations that target a memory window defined by an NT endpoint are routed within the domain to that endpoint. When the non-transparent bridge receives a memory operation that targets a BAR used for mapping through the bridge, it translates the address of the transaction to a new address in the opposite domain and forwards the transaction to the other domain. Completions are handled in a similar manner.

As shown in Figure 4, the switch allows two or more non-transparent endpoints to directly communicate over a *non-transparent interconnect*. This extension of non-transparent bridging from two-ports to multiple ports parallels the evolution of two-port PCI bridges to multi-port PCI Express switches.

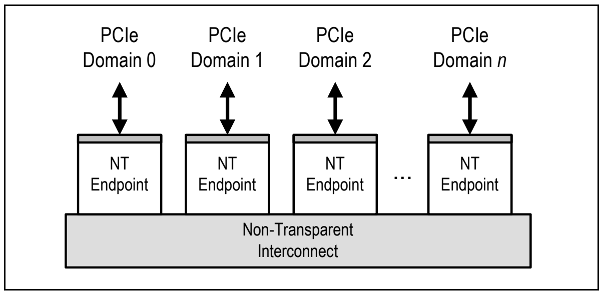


Figure 4 Generalized Multi-Port Non-Transparent Interconnect

There are numerous approaches for integrating a non-transparent bridge into a PCI Express switch. Figure 5 illustrates three approaches.

Figure 5(a) shows an architecture in which a non-transparent bridge is integrated below the PCI-to-PCI bridge associated with a downstream port. This architecture is used in IDT Gen 1 switches. A disadvantage of this approach is that it leads to complex implementations when extended to direct non-transparent switching.

Figure 5(b) illustrates an architecture in which a non-transparent bridge is integrated directly onto the virtual PCI bus. The advantage of this approach is that it is simple to implement since the PCI-to-PCI bridge associated with a downstream port may be replaced (or reconfigured) with a non-transparent bridge. The issue with this approach is that it violates the fundamental requirement outlined in the PCI Express base specification that endpoints (represented by type 0h headers) must not appear to configuration software on a switch's internal bus as peers of the virtual PCI-to-PCI bridges representing the switch downstream ports.

Figure 5(c) exhibits the architecture used in the switch. In this architecture, the upstream port is transformed into a multi-function device with function 0 representing the PCI-to-PCI bridge associated with the upstream port, and function 1 representing the NT endpoint.

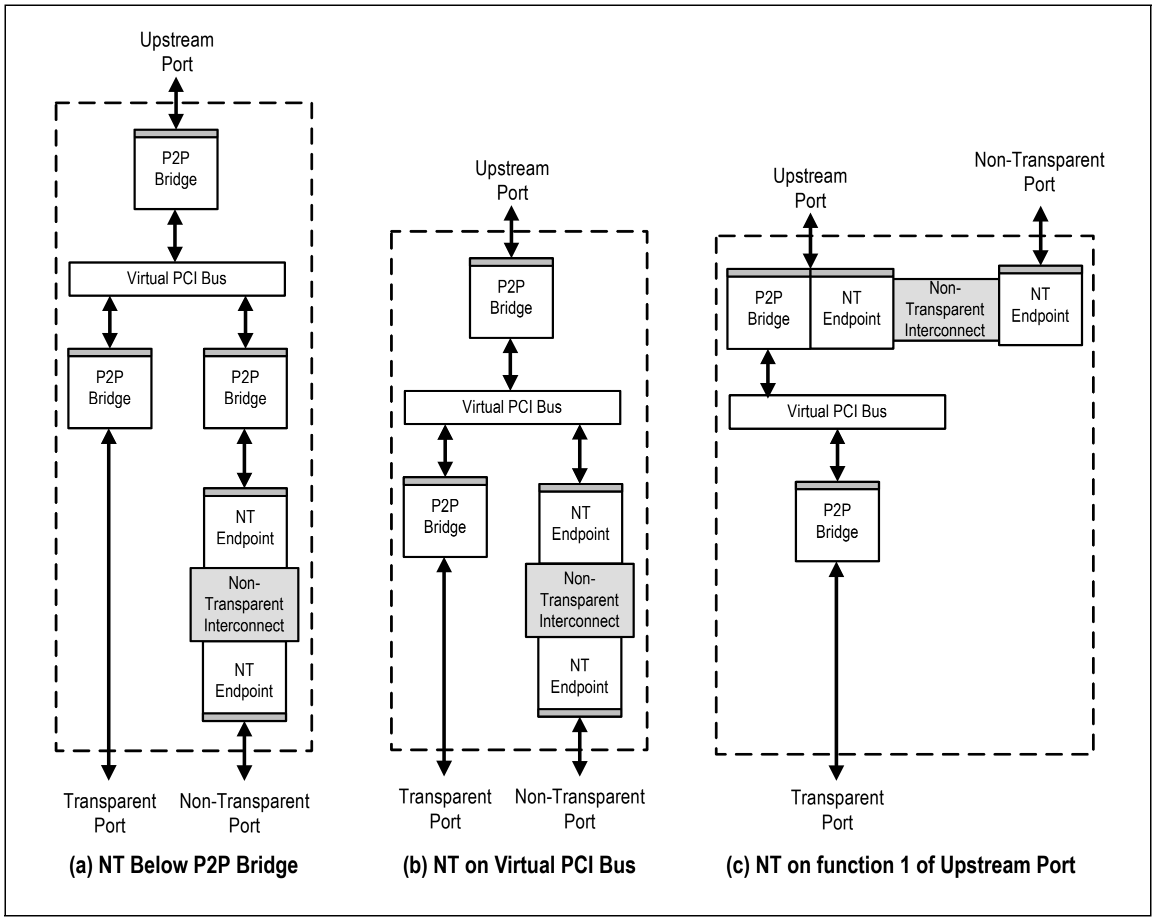


Figure 5 Architectural Approaches for Integrating Non-Transparency into a PCI Express Switch

Notes

A switch port may be configured to operate in one of seven major modes.

– Upstream switch port (i.e., upstream PCI-to-PCI bridge)
– Downstream switch port (i.e., downstream PCI-to-PCI bridge)
– Upstream switch port with DMA function
– Upstream switch port with NT function
– Upstream switch port with NT and DMA functions
– NT function
– NT with DMA function

Figure 6 illustrates a basic non-transparent switch configuration. In this configuration, the ports are split into two partitions. Each partition represents a three-port transparent PCI Express switch. The upstream port of each partition is configured to operate as an *upstream switch port with NT endpoint*.

This configuration allows direct partition to partition communications without consuming external switch ports or links.

The NT endpoints in Figure 6 communicate using the non-transparent interconnect. This allows PCI Express functions in either domain to communicate using the address windows presented by the NT endpoint BARs.

Functions may be connected to the upstream port (e.g., the root) or to a downstream switch port. Upstream port TLPs flow directly to the corresponding NT endpoint. Downstream switch port TLPs flow through the corresponding three-port transparent switch and then back to the NT endpoint via the upstream port.

TLPs flowing from the secondary side of an upstream port's PCI-to-PCI bridge, through the bridge, to the NT endpoint stay entirely within the switch and are not transmitted on the upstream port's link.

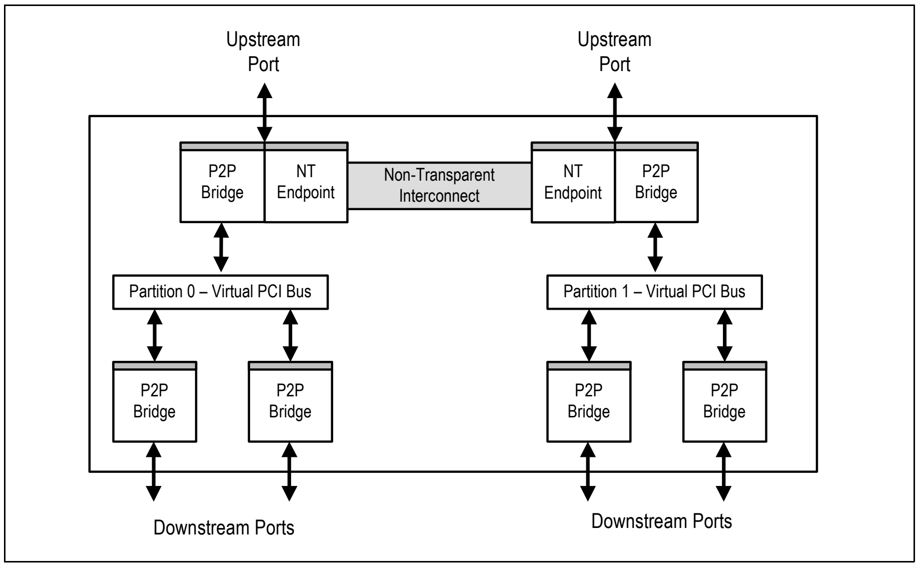


Figure 6 Non-Transparent Switch with Non-Transparency Between Partitions

Figure 7 illustrates a basic non-transparent configuration with NT ports for the switch. In this configuration, the ports are split into four partitions. The first partition, partition 0, represents a three-port transparent PCI Express switch. The remaining three partitions consist of the three NT endpoints[1]. Requesters in any of the NT domains may communicate using the address windows presented by the NT endpoint BARs.

[1] A port configured as an NT port logically consists of only an NT endpoint and represents a switch partition.

Notes

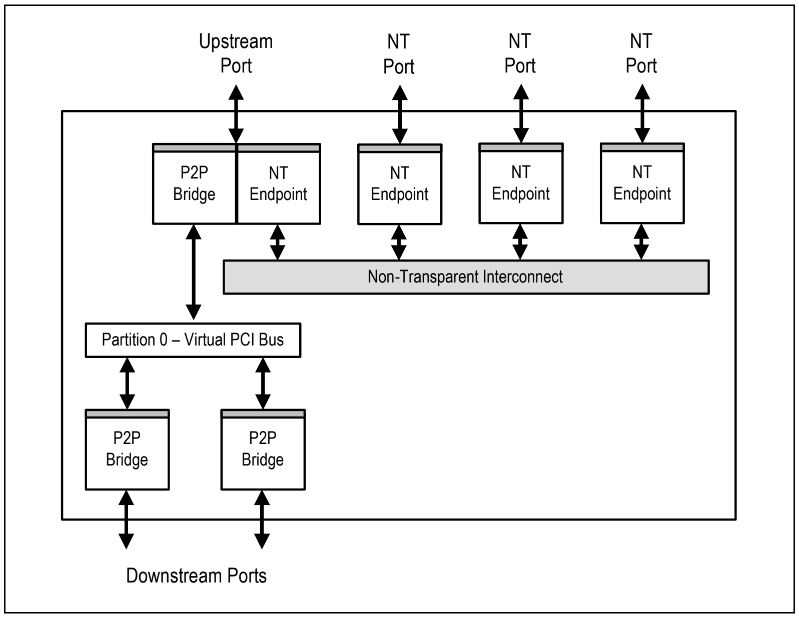


Figure 7 Non-Transparent Switch with Non-Transparent Ports Example # 1

Figure 8 illustrates the configuration in which all ports are configured as NT endpoints. Such a configuration may be useful in bladed systems.

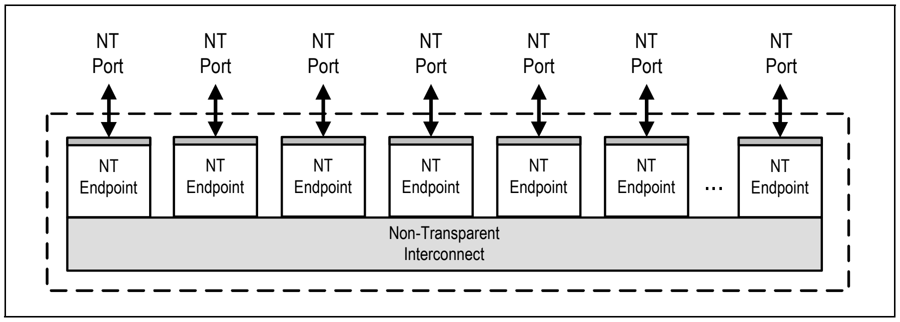


Figure 8 Non-Transparent Switch with Non-Transparent Ports Example # 2

This section outlined several possible switch NTB configurations. The ability to configure ports to operate in a variety of modes together with support for up to 16 partitions provides the switch with the flexibility required for a wide variety of system applications. Figure 9 illustrates a configuration with three transparent switch partitions and four NT port partitions. Non-transparent communication is supported between all partitions except partition zero.

Notes

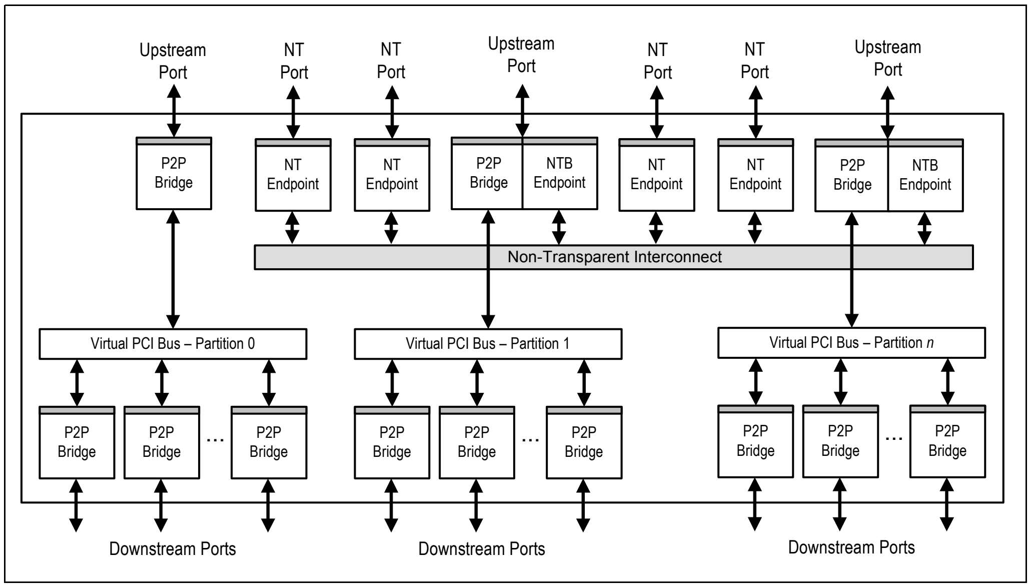


Figure 9 Non-Transparent Switch with Non-Transparent Ports Example # 3

Base Address Registers (BARs)

Each NT-endpoint implements six Base Address Registers (BARs) labeled BAR 0 through BAR 5. Table 1 summarizes supported BAR configurations. All BARs may be configured to create 32-bit memory[1] windows between the PCI Express domain and the non-transparent interconnect.

- All BARs support direct address translation
- BAR 2 (or BAR 2/3 in 64-bit mode) supports direct address translation or lookup address translation
- BAR 4 (or BAR 4/5 in 64-bit mode) supports direct address translation or lookup address translation

Even and odd BARs may be paired to form 64-bit prefetchable memory space. The 4 KB configuration space associated with the NT endpoint may be mapped into 32-bit memory using BAR 0. BAR 0 and BAR 1 may be paired to map the 4 KB configuration space associated with the NT endpoint into 64-bit memory.

BAR	Description
BAR 0	32-bit BAR that maps 4 KB NT-endpoint configuration registers Lower half of 64-bit BAR that maps 4 KB NT-endpoint configuration registers 32-bit BAR with direct address translation Lower half of 64-bit BAR with direct address translation
BAR 1	Upper half of 64-bit BAR that maps 4 KB NT-endpoint configuration registers 32-bit BAR with direct address translation Upper half of 64-bit BAR with direct address translation
BAR 2	32-bit BAR with direct or lookup table address translation Lower half of 64-bit BAR with direct or lookup address translation

Table 1 NT Endpoint BARs (Page 1 of 2)

[1] The NT function's BARs do not support I/O space.

Notes

BAR	Description
BAR 3	32-bit BAR with direct address translation Upper half of 64-bit BAR with direct or lookup table address translation
BAR 4	32-bit BAR with direct or lookup table address translation Lower half of 64-bit BAR with direct or lookup table address translation
BAR 5	32-bit BAR with direct address translation Upper half of 64-bit BAR with direct or lookup table address translation

Table 1 NT Endpoint BARs (Page 2 of 2)

Each BAR has a corresponding setup register. For example, BAR 0 (BAR0) has an associated BAR Setup 0 (BARSETUP0) register. BAR setup registers allow a BAR to be enabled or disabled, control the operating mode of the BAR as well as advertised capabilities (e.g., size of the BAR window), and if applicable, control the address translation mechanism.

When an even BAR is configured for 64-bit operation, the odd BAR takes on the function of the upper 32-bits of the BADDR field of the even BAR. When an even and odd BAR are merged for 64-bit operation, the fields of the odd BAR Setup register remain read-write but have no functional effect on the operation of the device.

BAR Limit

Base Address Registers are used by a function to specify the amount of memory space required by the function. Software configures read-write BAR register bits with the base address of the allocated memory range.

Since BARs specify the size of an aperture with read-only bits in the BADDR field, the PCI architecture only allows apertures to be requested that are a power of two in size. In many applications, it is desirable to allocate smaller apertures.

Associated with each BAR is a BAR Limit Address (BARLIMITx) register. The limit address specified by this register allows arbitrary control of the aperture size associated with a BAR. Using this capability, the *effective aperture size* may be set arbitrarily to any value, in 1 KB multiples, up to the power of two aperture size requested by the BAR.

– The lower 10-bits of the BARLIMITx register are reserved and assumed to be all ones by the hardware. Thus, the BAR limit address may be anywhere in the range from 0x3FF (i.e., 1KB) to the highest possible memory address.
– Setting the address limit of a BAR to a value less than the BAR base address effectively disables the BAR. The effective aperture size in this case is zero.
– Setting the address limit of a BAR to a value between the BAR base address and the BAR base address plus the power of two aperture size requested by the BAR sets the effective aperture to be from the BAR base address up to and including the BAR limit address.
– Setting the address limit of a BAR to a value greater than the BAR base address plus the power of two aperture size requested by the BAR disables the limit capability. The aperture and effective aperture in this case are both equal to the power of two size requested by the BAR.
– The default value of the BARLIMITx register causes the BAR limit address to point to the highest possible address, in effect disabling the effect of the BARLIMIT register.
– The BARLIMIT0 register is ignored when BAR 0 is mapped to the NT endpoint's configuration space.

When an even BAR is configured for 64-bit operation, the odd BAR takes on the function of the upper 32-bits of the BADDR field of the even BAR. In this mode, the odd BARLIMITx register acts as the upper 32-bits of the LADDR field associated with the even BAR. Figure 10 graphically illustrates the operation of the BAR limit address.

Notes

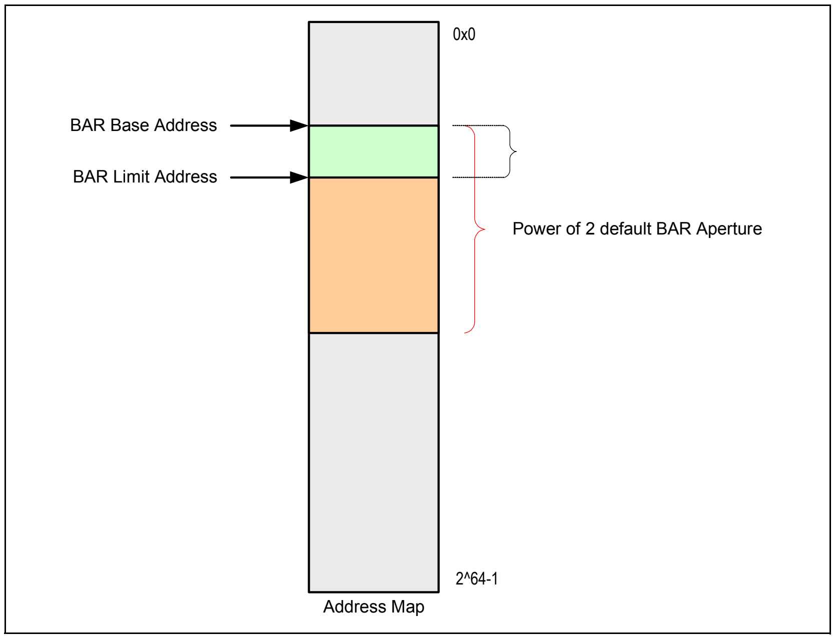


Figure 10 BAR Limit Operation

A received TLP whose address does not fall within the effective BAR aperture of a BAR is not processed by the BAR. A received TLP whose address falls within a BAR aperture, but outside the *effective BAR aperture*, is treated as an unsupported request by the NT function. The behavior for TLPs whose address falls within the effective address of multiple BARs is undefined.

Mapping NT Configuration Space to BAR 0

As mentioned above, the 4 KB configuration space associated with the NT endpoint may be mapped into 32-bit memory using BAR 0. BAR 0 and BAR 1 may be paired to map the 4 KB configuration space associated with the NT endpoint into 64-bit memory.

– Mapping NT configuration space to BAR 0 allows these configuration space registers to be accessed via memory read or writes.
– Mapping NT configuration space to BAR 0 requires that the MODE field be set appropriately in the BARSETUP0 register. When NT configuration space is mapped to BAR 0, the size of the BAR aperture is automatically set to 4 KB and the BARLIMIT0 register is ignored.

When the NT function's configuration space is mapped to BAR 0, it is recommended that this configuration space be placed in non-prefetchable memory space, as some registers may generate side-effect actions when accessed. In addition, memory read or write requests to BAR 0 must specify a length of 1 DWord. Violating this last requirement produces undefined results.

TLP Translation

Direct Address Translation

All BARs may be configured to support direct address translation. Figure 11 illustrates the address translation process for a BAR configured as a memory address window with direct address translation.

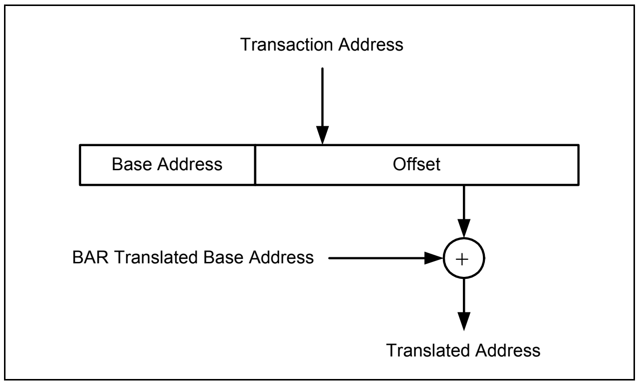


Figure 11 Direct Address Translation

The address of a TLP that falls within the effective BAR aperture of a BAR may be partitioned into a base address and an offset.

- The base address is equal to the value programmed in BAR BADDR field bits that are read/write.
- The offset address corresponds to address bits that are not part of the base address.

Associated with each BAR is a translated base address.

- The translated base address is a 64-bit quantity. The upper 32-bits are set to zero when the translated base address is less than or equal to 0xFFFF_FFFF (i.e., lower 4 GB).
- The translated base address is always DWord aligned. Therefore, the bottom two bits are always zero.

The translated address for the TLP is equal to the sum of the TLP offset address with the corresponding translated base address field.

- Following formation of the translated address, the TLP header size is adjusted accordingly.
  - If the upper 32-bits of the 64-bit address are all zero, then a 3 DWord header is used.
  - If the upper 32-bits of the 64-bit address are non-zero, then a 4 DWord header is used.

The partition of the translated TLP is specified by the Translated Partition (TPART) field in the corresponding BARSETUPx register.

- If the partition associated with the translated TLP is invalid (e.g., there is no NT endpoint associated with the destination partition, the destination partition is not in the active state, or the destination partition is the same as the original partition), then the TLP is treated as an unsupported request by the NT endpoint that received the request.

The behavior of a TLP whose translated address falls within the BAR aperture(s) of the NT function in the destination (i.e., translated) partition is undefined.

Lookup Table Address Translation

BARs two and four may be configured to support lookup table address translation.

- BAR two may be configured to support either a 16-entry or 32-entry lookup table.
- BAR four only supports 16-entry lookup table address translation. Configuring BAR four for 32-entry lookup table address translation produces undefined results.
- If both BARs two and four are configured for lookup table address translation, then BAR two only supports a 16-entry lookup table. Configuring BAR two for 32-entry lookup table address translation while BAR four is configured for 16-entry lookup table address translation produces undefined results.

Figure 12 illustrates the address translation process for a BAR configured as a memory address window with lookup table address translation.

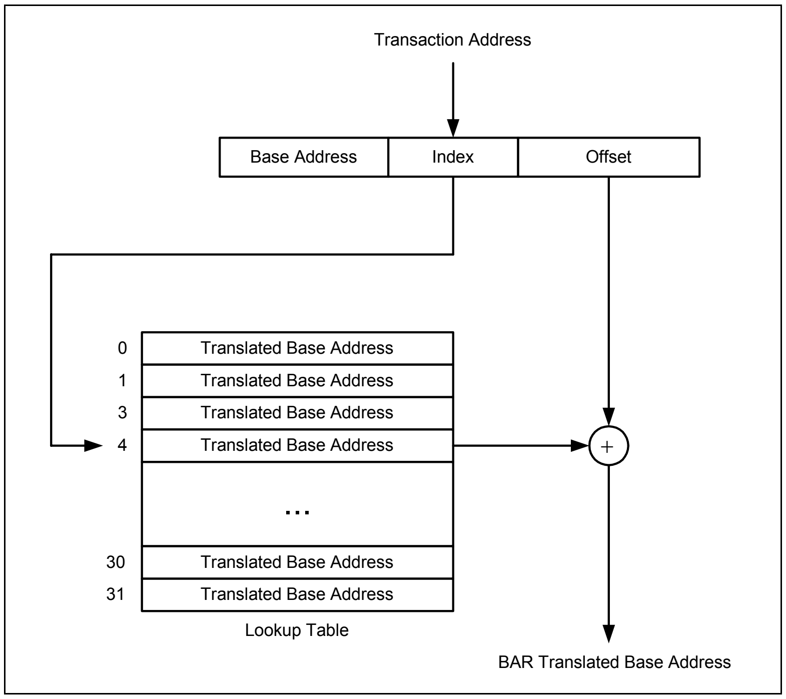


Figure 12 Lookup Table Translation

The address of a TLP that falls within the effective BAR aperture of a BAR may be partitioned into a base address, index, and offset.

- The base address is equal to the value programmed in the BAR BADDR field bits that are read/write.
- If the BAR is configured for a 16-entry lookup table, then the index corresponds to the next 4-bits of the address. If the BAR is configured for a 32-entry lookup table, then the index corresponds to the next 5-address bits.
- The offset address corresponds to address bits that are not part of the base address or index.

When the BAR is configured to operate as an address window with lookup table address translation, valid values for the SIZE field in the corresponding BARSETUPx register are 14 through 37 (values greater than 32 require a 64-bit BAR). Setting the SIZE field outside this range produces undefined results. Associated with a BAR configured to use lookup table address translation is a 16 or 32-entry lookup table. The format of a table entry is shown in Figure 13.

- The translated base address field plays the same role as the translated base address in direct address translation.
- The partition field specifies the destination partition associated with the translated TLP.
- The valid field indicates if the table entry is valid.

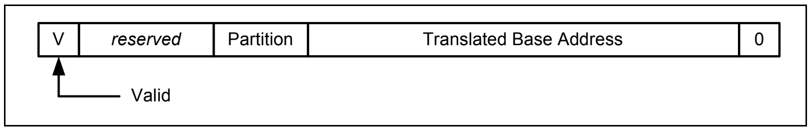


Figure 13 Lookup Table Entry Format

Notes

When a TLP is processed by a BAR that is configured for lookup table address translation, the index portion of the TLP address is used as an index into the lookup table. If the table entry pointed to by the index is not valid, then the TLP is treated as an unsupported request. If the table entry pointed to by the index is valid, then the translated address for the TLP is formed by adding the offset field of the TLP address to the translated base address associated with the table entry.

- The translated base address is a 64-bit quantity allowing the translated window to be located anywhere within the address space of the destination PCI Express domain.
- The size of a PCI Express TLP header varies depending on the size of the address. A 32-bit address has a three DWord header while a 64-bit address has a four DWord header. Following formation of the translated address, the TLP header size is adjusted accordingly.

The destination partition associated with the translated TLP is specified by the partition field in the lookup table entry. If the partition associated with the translated TLP is invalid (e.g., there is no NT endpoint associated with the destination partition, the destination partition is not in the active state, or the destination partition is the same as the original partition), then the TLP is treated as an unsupported request by the NT endpoint that received the request. The behavior of a TLP whose translated address falls within the BAR aperture(s) of the NT function in the destination partition is undefined.

Table 2 and Table 4 summarize the parameters associated with a 16-entry and 32-entry lookup table. Page size refers to the size of the address space translated by each table entry.

BARSETUPx SIZE Field	Aperture Size	Page Size	Base Address (bits)[1]	Index (bits)	Offset (bits)
14	16 KB	1 KB	[63:14]	[13:10]	[9:0]
15	32 KB	2 KB	[63:15]	[14:11]	[10:0]
16	64 KB	4 KB	[63:16]	[15:12]	[11:0]
17	128 KB	8 KB	[63:17]	[16:13]	[12:0]
18	256 KB	16 KB	[63:18]	[17:14]	[13:0]
19	512 KB	32 KB	[63:19]	[18:15]	[14:0]
20	1 MB	64 KB	[63:20]	[19:16]	[15:0]
21	2 MB	128 KB	[63:21]	[20:17]	[16:0]
22	4 MB	256 KB	[63:22]	[21:18]	[17:0]
23	8 MB	512 KB	[63:23]	[22:19]	[18:0]
24	16 MB	1 MB	[63:24]	[23:20]	[19:0]
25	32 MB	2 MB	[63:25]	[24:21]	[20:0]
26	64 MB	4 MB	[63:26]	[25:22]	[21:0]
27	128 MB	8 MB	[63:27]	[26:23]	[22:0]
28	256 MB	16 MB	[63:28]	[27:24]	[23:0]
29	512 MB	32 MB	[63:29]	[28:25]	[24:0]
30	1 GB	64 MB	[63:30]	[29:26]	[25:0]
31	2 GB	128 MB	[63:31]	[30:27]	[26:0]
32	4 GB	256 MB	[63:32]	[31:28]	[27:0]
33	8 GB	512 MB	[63:33]	[32:29]	[28:0]
34	16 GB	1 GB	[63:34]	[33:30]	[29:0]

Table 2 16-Entry Lookup Table Parameters (Page 1 of 2)

Notes

BARSETUPx SIZE Field	Aperture Size	Page Size	Base Address (bits)[1]	Index (bits)	Offset (bits)
35	32 GB	2 GB	[63:35]	[34:31]	[30:0]
36	64 GB	4 GB	[63:36]	[35:32]	[31:0]
37	128 GB	8 GB	[63:37]	[36:33]	[32:0]

Table 2 16-Entry Lookup Table Parameters (Page 2 of 2)

1. Assumes 64-bit TLP address. If only 32-bits are used then bits [31:x] are used.

BARSETUPx SIZE Field	Aperture Size	Page Size	Base Address (bits)[1]	Index (bits)	Offset (bits)
14	16 KB	512 B	[63:14]	[13:9]	[8:0]
15	32 KB	1 KB	[63:15]	[14:10]	[9:0]
16	64 KB	2 KB	[63:16]	[15:11]	[10:0]
17	128 KB	4 KB	[63:17]	[16:12]	[11:0]
18	256 KB	8 KB	[63:18]	[17:13]	[12:0]
19	512 KB	16 KB	[63:19]	[18:14]	[13:0]
20	1 MB	32 KB	[63:20]	[19:15]	[14:0]
21	2 MB	64 KB	[63:21]	[20:16]	[15:0]
22	4 MB	128 KB	[63:22]	[21:17]	[16:0]
23	8 MB	256 KB	[63:23]	[22:18]	[17:0]
24	16 MB	512 KB	[63:24]	[23:19]	[18:0]
25	32 MB	1 MB	[63:25]	[24:20]	[19:0]
26	64 MB	2 MB	[63:26]	[25:21]	[20:0]
27	128 MB	4 MB	[63:27]	[26:22]	[21:0]
28	256 MB	8 MB	[63:28]	[27:23]	[22:0]
29	512 MB	16 MB	[63:29]	[28:24]	[23:0]
30	1 GB	32 MB	[63:30]	[29:25]	[24:0]
31	2 GB	64 MB	[63:31]	[30:26]	[25:0]
32	4 GB	128 MB	[63:32]	[31:27]	[26:0]
33	8 GB	256 MB	[63:33]	[32:28]	[27:0]
34	16 GB	512 MB	[63:34]	[33:29]	[28:0]
35	32 GB	1 GB	[63:35]	[34:30]	[29:0]
36	64 GB	2 GB	[63:36]	[35:31]	[30:0]
37	128 GB	4 GB	[63:37]	[36:32]	[31:0]

Table 3 32-Entry Lookup Table Parameters

1. Assumes 64-bit TLP address. If only 32-bits are used then bits [31:x] are used.

The lookup table for both BARs is configured using the Lookup Table Offset (LUTOFFSET), Lookup Table Lower Data (LUTLDATA), Lookup Table Middle Data (LUTMDATA), Lookup Table Upper Data (LUTUDATA) registers.

Notes

Fields associated with lookup entries are modified by accessing the LUTLDATA, LUTLMDATA and LUTUDATA registers. A read from one of these registers returns the field values of the lookup table entry pointed to by the LUTOFFSET register. Similarly, a write updates the fields of the lookup entry pointed to by the LUTOFFSET register.

The BAR field in the LUTOFFSET register selects the lookup table associated with the corresponding BAR while the INDEX field in the LUTOFFSET field selects the lookup table entry. The state of the lookup table is preserved across all resets except a switch fundamental reset. Following a switch fundamental reset, the state of all lookup table fields except the Valid (V) field is undefined. Following a switch fundamental reset, the Valid field is cleared in all entries.

ID Translation

PCI Express TLPs may be categorized into request TLPs and completion TLPs.

– A request TLP is a packet used to initiate a transaction.

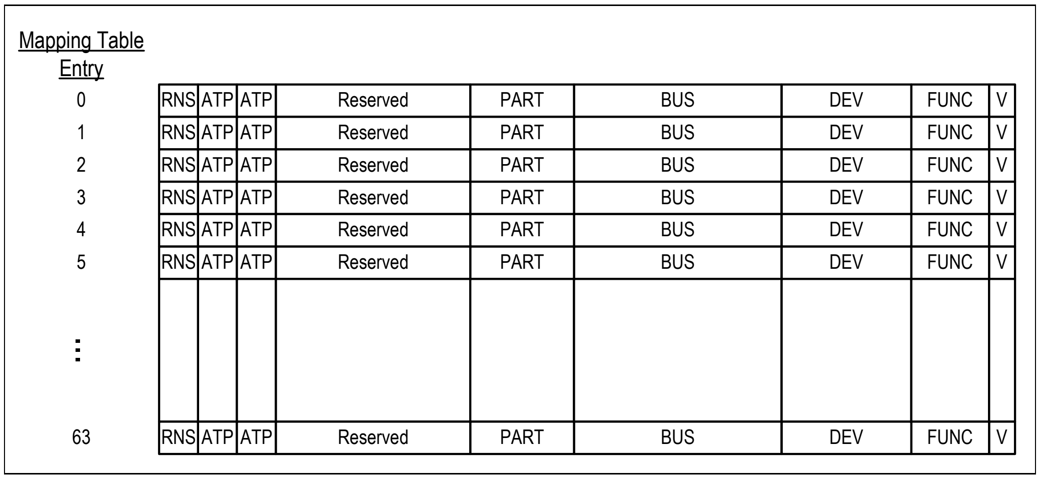
– A completion TLP is a packet used to terminate, or partially terminate a transaction sequence.

Request TLPs contain a requester ID field that defines the unique PCI Express identifier associated with the requester that generated the request TLP. Completion TLPs contain both a requester ID field and a completer ID field. The completer ID field defines the unique PCI Express identifier associated with the completer that generated the completion TLP.

A PCI Express identifier consists of a 16-bit quantity that is unique for each function in a PCI Express hierarchy. The 16-bit quantity may be interpreted as an 8-bit bus number, 5 bit device number, and 3 bit function number. The Alternate Routing-ID Interpretation ECN specifies a secondary interpretation.

NT Mapping Table

Associated with the switch is a 64-entry Non-Transparent (NT) mapping table. The NT mapping table is used to perform ID translation and ID based protection. The NT mapping table is a global table shared by all ports configured for NT operation. The format of the NT mapping table is shown in Figure 14 and the fields are described in Table 4.



Mapping Table Entry									
0	RNS	ATP	ATP	Reserved	PART	BUS	DEV	FUNC	V
1	RNS	ATP	ATP	Reserved	PART	BUS	DEV	FUNC	V
2	RNS	ATP	ATP	Reserved	PART	BUS	DEV	FUNC	V
3	RNS	ATP	ATP	Reserved	PART	BUS	DEV	FUNC	V
4	RNS	ATP	ATP	Reserved	PART	BUS	DEV	FUNC	V
5	RNS	ATP	ATP	Reserved	PART	BUS	DEV	FUNC	V
⋮									
63	RNS	ATP	ATP	Reserved	PART	BUS	DEV	FUNC	V

Figure 14 NT Mapping Table

Notes

Bit Field	Field Name	Description
0	V	**Valid**. This bit is set if the mapping table is valid.
3:1	FUNC	**Function**. This field contains the mapping table entry PCI Express function number.
8:4	DEV	**Device**. This field contains the mapping table entry PCI Express device number.
16:9	BUS	**Bus**. This field contains the mapping table entry PCI Express bus number.
19:17	PART	**Partition**. This field contains the mapping table entry partition number.
29	ATP	**Address Type Processing.** This field specifies the processing of the address type (AT) field on request TLPs. Refer to Section .
30	CNS	**Completion No Snoop Processing**. This field specifies the no snoop processing on completion TLPs. Refer to Section .
31	RNS	**Request No Snoop Processing**. This field specifies the no snoop processing on request TLPs. Refer to Section .

Table 4 NT Mapping Table Field Description

The state of the NT mapping table is preserved across all resets except a switch fundamental reset. Following a switch fundamental reset, the state of all NT mapping table fields except the Valid (V) field is undefined. Following a switch fundamental reset, the Valid field is cleared in all entries. The mapping tables may be initialized by using the NT Mapping Table Address (NTMTBLADDR) and NT Mapping Table Data (NTMTBLDATA) registers.

To access a mapping table entry, the NT Mapping Table Address (ADDR) field in the port's NTMTBLADDR register is initialized with the partition NT mapping table entry to be accessed. Reading from the NT Mapping Table Data (NTMTBLDATA) register returns the value of the fields of the corresponding partition NT mapping table entry pointed to by the ADDR field in the NTMTBLADDR register. Writing to the NTMTBLDATA register causes the fields in the corresponding partition NT mapping table entry pointed to by the ADDR field in the NTMTBLADDR register to be updated with the value written. The NTMTBLDATA register must be accessed using DWORD operations. The behavior for all other access sizes is undefined.

Causing an NT mapping table protection violation results in the NT Mapping Table Access Error (ERR) bit to be set in the NT Mapping Table Status (NTMTBLSTS) register. Since the NT mapping table is global and shared by all partitions, the switch supports NT mapping table protection and virtualization.

Located in the Switch Control and Status register space is an NT Mapping Table Protection (NTMTBLPROTx) register for each partition. NT Mapping Table Base (TBLBASE) and NT Mapping Table Limit (TBLLIMIT) fields in the NTMTBLPROTx register control how partition NT mapping table accesses are translated into physical NT mapping table accesses. The translation process is shown in Figure 15.

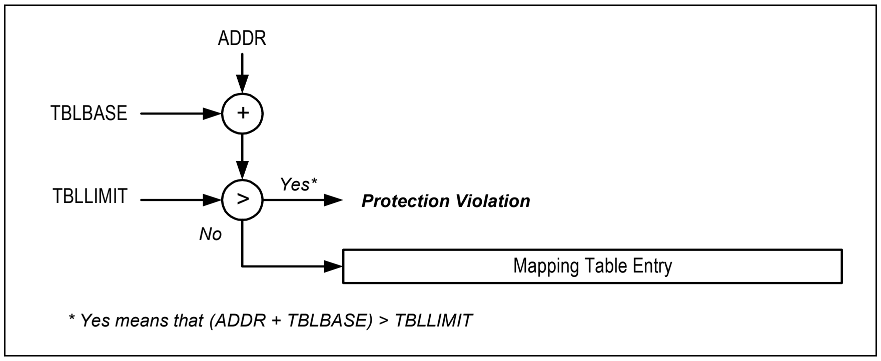


Figure 15 NT Table Partitioning

The physical NT mapping table entry accessed is equal to the sum of the partition NT mapping table entry, specified by the ADDR field in the NTMTBLADDR register, with the TBLBASE field in the NTMTBLPROT register associated with the partition. If the resulting physical NT mapping table entry address is less than or equal to the value in the TBLLIMIT field in the NTMTBLPROT field associated with the partition, then the read or write access is performed on the physical NT mapping table entry. If the value of the resulting NT mapping table entry address is greater than the NTBLLIMIT field, then a protection violation is signaled.

Located in each NTMTBLPROT register is a Partition Blocking Vector (PARTBLOCK). Associated with each partition in the device is a corresponding bit in the PARTBLOCK vector. When a write to the NTMTBLDATA register is performed with a PART field value whose corresponding bit is set in the PARTBLOCK vector, then a protection violation is signaled.

A protection violation during an NTMTBLDATA read operation results in a value of zero being returned in all fields and setting of the NT Mapping Table Access Error (ERR) bit in the NT Mapping Table Status (NTMTBLSTS) register. A protection violation during an NTMTBLDATA write operation causes the write operation to be ignored (i.e., no table entry or register field is actually updated) and setting of the NT Mapping Table Access Error (ERR) bit in the NT Mapping Table Status (NTMTBLSTS) register.

Following a switch fundamental reset, NT mapping table protection and virtualization is disabled.

– All partitions may access all 64 physical NT mapping table entries.
– The virtual NT mapping table address of all partitions is equal to the physical NT mapping table address.
– A partition may update any NT mapping table entry with any PART field value.

Request ID Translation

Request TLPs contain a requester ID field that defines the unique PCI Express identifier associated with the requester that generated the request TLP. When a request TLP is received by an NT endpoint that is to be routed on the NT interconnect (i.e., the TLP hits an NT endpoint BAR aperture and has a valid translation), a requester ID lookup and translation are performed.

The lookup is performed by matching the 16-bit requester ID in the request TLP along with the partition associated with the NT endpoint to entries in the NT mapping table. If a lookup match is not found, then the TLP is treated as an unsupported request. Otherwise, the TLP is processed normally as described below:

– 16-bit requester ID is compared to the 16-bit value in each NT mapping table consisting of the BUS, DEV, and FUNC fields regardless of the requester ID interpretation. A requester ID match occurs when the 16-bit value in the requester ID field of the TLP matches an NT mapping table entry.
– The partition associated with a request TLP is the partition ID associated with the NT endpoint which received the request TLP. A partition match occurs when the partition associated with a request TLP matches the PART field of an NT mapping table entry.
– A lookup match occurs for a request TLP when a NT mapping table entry exists that is valid (i.e., the V bit is set) and has both a requester ID match as well as a partition match.
– The behavior of a request TLP with multiple lookup matches is undefined. Multiple lookup matches are the result of an invalid configuration.

PCI Express allows a function to expand the number of supported outstanding requests requiring completions beyond 256 through the use of phantom function numbers. When phantom function numbers are enabled, the Tag field in the TLP header may be logically expanded by using unimplemented function numbers. These unimplemented function numbers are referred to as phantom function numbers. A requester that uses phantom function numbers when communicating with the NT endpoint requires a unique NT mapping table entry for each phantom function number.

The requester ID field associated with a request TLP that has a lookup match is translated as shown in Figure 16.

– The bus field is replaced by the captured bus number of the NT endpoint associated with the partition of the translated TLP.
– Bit 4 of the device field is set to one.
– The lower four bits of the device field and the function field are replaced with the lookup table match entry.

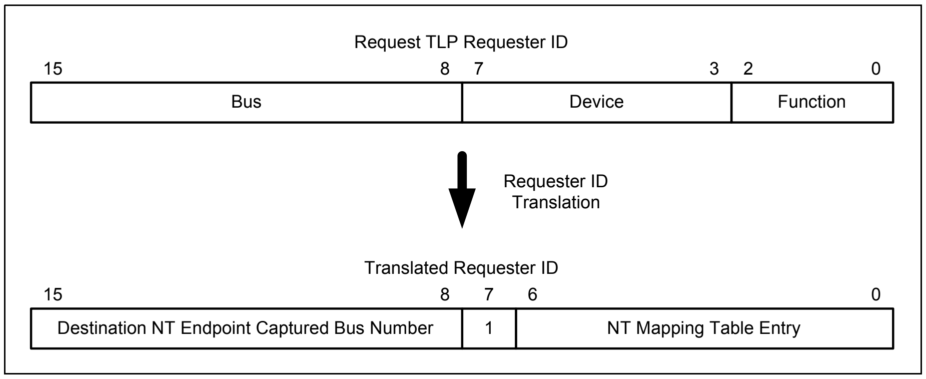


Figure 16 Request TLP Requester ID Translation

One function of request TLP ID translation process is that it allows a corresponding reverse translation to occur for completions. This reverse translation is described in section Completion ID Translation on page 18. Posted requests (e.g., memory writes) have no corresponding completions. Therefore, the primary role of the NT mapping table lookup for these TLPs is to provide a form of protection (i.e., only authorized requesters are allowed to issue TLPs that map onto the NT interconnect). When the ID Protection Check Disable (IDPROTDIS) bit in the Endpoint Control (NTCTL) register is set, the NT table lookup for posted requests is skipped and all posted requests are allowed to map onto the NT interconnect regardless of requester ID.

The requester ID field associated with a posted request TLP is translated as shown in Figure 17 when the IDPROTDIS bit is set in the NTCTL register.

– The bus field is replaced by the captured bus number of the NT endpoint associated with the partition of the translated TLP.
– The device and function fields are replaced by the value 0x3. This corresponds to device 0, function 3.

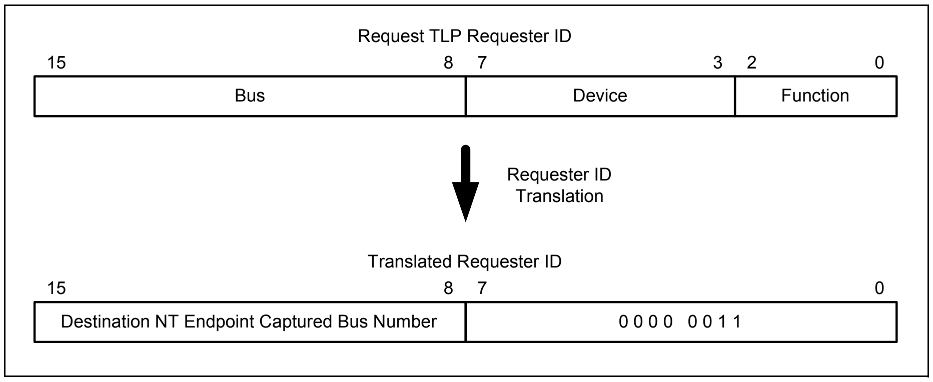


Figure 17 Request TLP Requester ID Translation with ID Protection Check Disabled

If the Bus Master Enable (BME) bit is cleared in the PCI Command (PCICMD) register of the NT endpoint associated with the translated TLP (i.e., in the destination partition), then the request is treated as an unsupported request by the NT endpoint that received the request (i.e., in the source partition).

Notes

Completion ID Translation

Completion TLPs contain both a requester ID field and a completer ID field. The completer ID field defines the unique PCI Express identifier associated with the completer that generated the completion TLP.

When a completion TLP is received and claimed[1] by an NT endpoint the following processing is performed:

– If the TLP's requester ID function field matches that of the NT endpoint (i.e., either zero or one depending on the port operating mode), and a punch-through configuration is in progress, the TLP is accepted by the NT endpoint and processed as described in section Punch-Through Configuration Requests on page 22.

– If the TLP's requester ID function field matches that of the NT endpoint and a punch-through configuration is not in progress, then the TLP is handled as an unexpected completion by the NT function.
  - Refer to section Error Detection and Handling by the NT Function on page 27.

– Otherwise, the 8-bit value consisting of the device and function fields are extracted from the requester ID field to form the NT lookup table entry index, and the processing described below is performed.

The NT mapping table entry index extracted from the TLP is used as an index into the NT lookup table. The requester ID of the translated completion TLP is formed as follows:

– The requester ID bus field is replaced by the NT mapping table bus field.

– The requester ID device field is replaced by the NT mapping table device field.

– The requester ID function field is replaced by the NT mapping table function field.

The completer ID of the translated completion TLP is equal to the bus, device, and function of the NT endpoint associated with the partition of the translated completion TLP (i.e., the NT function that emits the TLP).

Note that the setting of the Bus Master Enable (BME) bit in the PCICMD register of the NT endpoint in the destination partition has no effect on translation of completion TLPs.

TLP Attribute Processing

No Snoop Processing

The No Snoop attribute in the header of request TLPs indicates whether hardware enforced cache coherence is expected. Some platforms lack the ability to control the no snoop attribute for generated requests. Therefore, the switch provides the ability to modify the No Snoop attribute for TLPs flowing through the NT interconnect.

When an NT table lookup is performed for request TLPs (described in Section), the Request No Snoop Processing (RNS) field in the matching NT mapping table entry is examined. If the RNS bit is set, then the No Snoop attribute in the translated TLP is inverted. If the RNS bit is cleared, then the No Snoop attribute in the translated TLP is equal to that of the received request TLP (i.e., the No Snoop attribute is not modified).

If the Completion No Snoop Processing (CNS) field in the NT mapping entry corresponding to the extracted NT mapping table index (see Section) is set, then the No Snoop attribute in the translated TLP is inverted. If the CNS bit is cleared, then the No Snoop attribute in the translated TLP is equal to that of the received completion TLP (i.e., the No Snoop attribute is not modified).

Address Type Processing

As described in the Address Translation Services specification [2] and the PCI Express specification [1], the Address Type (AT) field in the header of a memory read or memory write TLP indicates the type of address in the TLP (i.e., untranslated, translation request, translated).

[1] A completion TLP is claimed by the NT endpoint when the TLP's requester ID matches the NT function's bus/device/function assignment within the PCI Express hierarchy or matches a valid entry in the NT function's mapping table.

Notes

The NT endpoint does not support Address Translation Services (ATS) as defined by the PCI-SIG, but it has the ability to modify the AT field for TLPs that cross the NTB. This allows the NTB to receive TLPs with translated addresses (i.e., AT field set to 'translated') in a source partition and emit them as TLPs with translated or untranslated addresses in the destination partition, or vice-versa.

Address type processing is only applied to memory read or write TLPs whose AT field is set to 'translated' or 'untranslated'. Address type processing is not applied to TLPs whose AT field is set to 'translation request'.

When an NT table lookup is performed for a request TLP (described in section Request ID Translation on page 16), the Address Type Processing (ATP) field in the matching NT mapping table entry is examined. If the ATP field is set to 0x1, then the AT field is set to 'translated' in the TLP emitted by the NT endpoint in the destination partition. Otherwise, the AT field is set to 'untranslated' in the TLP emitted by the NT endpoint in the destination partition.

Note that completion TLPs always have the AT attribute set to zero and are not subject to address type field modification.

NT Multicast

The NT function supports non-transparent (NT) multicast, which allows a TLP received by the NT function to be transmitted by zero or more ports of the switch, across partitions. A discussion of this function is beyond the scope of this application note. Contact IDT at ssdhelp@idt.com for additional information.

Inter-Domain Communications

The NT inter-domain communications capability structure provides facilities for supporting communications between processors in different PCI Express domains. The NT inter-domain communications capability provides the following facilities:

– Doorbell registers
– Message registers

Doorbell Registers

Doorbells facilitate event signalling between partitions. Associated with each NT endpoint are 32 outbound and inbound doorbell registers.

An outbound doorbell request from an NT endpoint is initiated by writing a one to the corresponding bit in the Outbound Doorbell Set (OUTDBELLSET) register.

– An outbound doorbell request from an NT endpoint is terminated by writing a one to the corresponding bit in the Outbound Doorbell Clear (OUTDBELLCLR) register.
– Writing a zero to a bit position in either the OUTDBELLSET or OUTSBELLDBELLCLR register has no effect on the register contents or corresponding doorbell request.

An inbound doorbell request to the NT endpoint results in the setting of the corresponding bit in the Inbound Doorbell Status (INDBELLSTS) register.

– The setting of a bit in the INDBELLSTS register may be used to generate an NT endpoint interrupt.
– Individual bits in the INDBELLSTS register may be masked from generating an interrupt by setting the corresponding bit in the Inbound Doorbell Mask (INDBELLMSK) register.

The logical operation of doorbells is illustrated in Figure 18.

– For each of the 32 outbound doorbell request, the requests from all partitions are logically OR-ed together to form a global doorbell request. This global doorbell request is then used to initiate inbound doorbell requests to each of the partitions.
    • The global doorbell request status may be determined by reading the Global Doorbell Status (GDBELLSTS) register.

Notes

- A bit in the GDBSELLSTS register is set if there exists an unmasked outbound doorbell request in any partition that corresponds to the bit.

– An outbound doorbell may initiate inbound doorbell requests in one or more partitions. All inbound doorbell requests share the same index. In other words, writing a one to bit position 8 in the OUTDBELLSET register may initiate an inbound doorbell request in multiple partitions, but each inbound doorbell request will be associated with the same bit position (i.e., position 8) as that of the outbound request.

– Associated with each outbound doorbell is a Global Outbound Doorbell Mask (GODBELLMSK[31:0]) register that contains a bit corresponding to each partition. When a bit in this register is set, outbound doorbell requests from the corresponding partition are masked. For example, setting bit 7 in the GOBDBELLMSK4 register masks doorbell 4 requests from partition 7.

- When a doorbell request from a partition is masked, the state of the doorbell in the corresponding partition plays no role in determining the state of the global doorbell status.

– A global doorbell request results in the initiation of corresponding inbound doorbell requests to all unmasked partitions.

- A global doorbell request may be masked to a partition by setting the corresponding partition bit in the Global Inbound Doorbell Mask (GIDBELLMSK[31:0]) register.
- When an inbound doorbell request is masked to a partition, the state of the global doorbell status plays no role in determining the state of the corresponding inbound doorbell request in that partition.

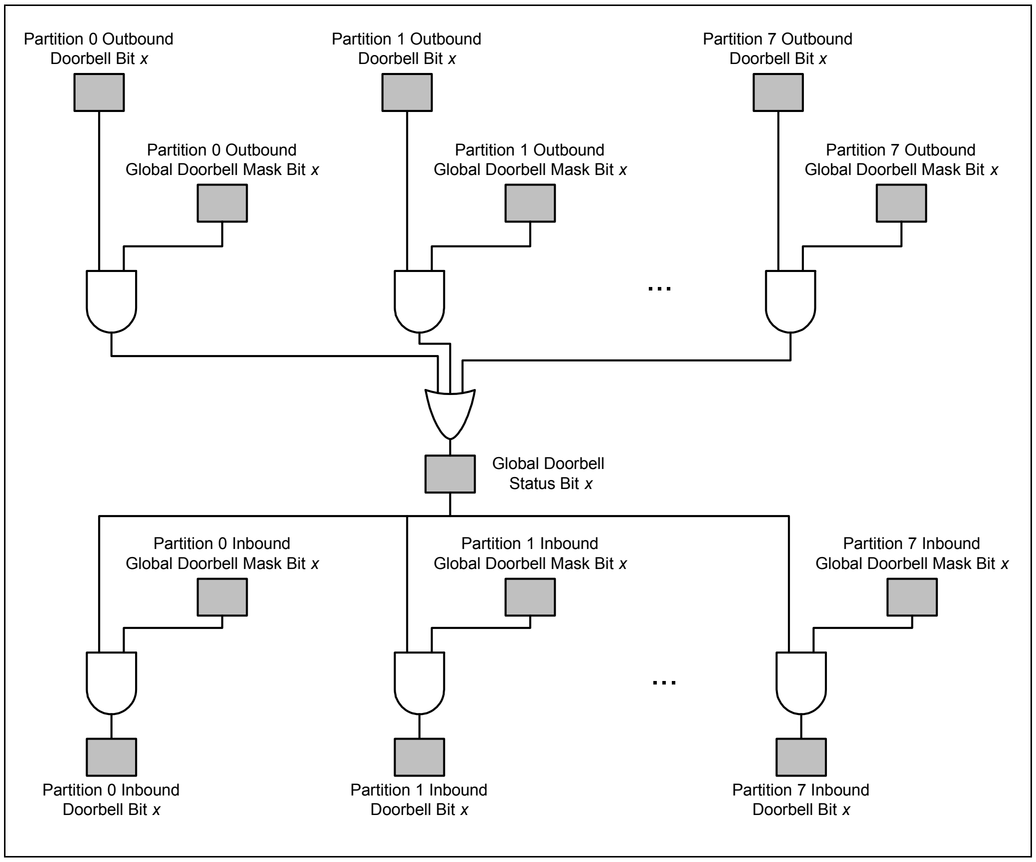

Partition 0 Outbound Doorbell Bit *x*
Partition 1 Outbound Doorbell Bit *x*
Partition 7 Outbound Doorbell Bit *x*
Partition 0 Outbound Global Doorbell Mask Bit *x*
Partition 1 Outbound Global Doorbell Mask Bit *x*
Partition 7 Outbound Global Doorbell Mask Bit *x*
...
Global Doorbell Status Bit *x*
Partition 0 Inbound Global Doorbell Mask Bit *x*
Partition 1 Inbound Global Doorbell Mask Bit *x*
Partition 7 Inbound Global Doorbell Mask Bit *x*
...
Partition 0 Inbound Doorbell Bit *x*
Partition 1 Inbound Doorbell Bit *x*
Partition 7 Inbound Doorbell Bit *x*

Figure 18 Logical Representation of Doorbell Operation

Doorbells are “level sensitive.” This means that a global doorbell request remains set as long as there exists an unmasked outbound request from any partition. Bits in the Inbound Doorbell Status (INDBELLSTS) register may be used to generate NT endpoint interrupts. A bit in the INDBELLSTS register may be masked from generating an interrupt by setting the corresponding bit in the Inbound Doorbell Mask (INDBELLMSK) register.

Notes

Message Registers

Message registers enable 32-bit values to be passed between partitions with interrupt notification. Each NT endpoint supports four Inbound Message (INMSG[3:0]) registers and four Outbound Message (OUTMSG[3:0]) registers. The logical operation of doorbells is illustrated in Figure 19.

– Associated with each outbound message register in a partition is a Switch Partition Message Control (SWPxMSGCTL[3:0]) register.
  - The register SWPxMSGCTLy corresponds to outbound message register y in switch partition x.
– When an outbound message register is written, the value written to the register is transferred to the inbound message register specified by the Register Select (REG) field of the SWPxMSGCTLy register in the partition specified by the Partition (PART) field of the SWPxMSGCTLy register. Thus, fields in the SWPxMSGCTLy register specify the routing of an outbound message register in one partition to an inbound message register in typically a different partition.

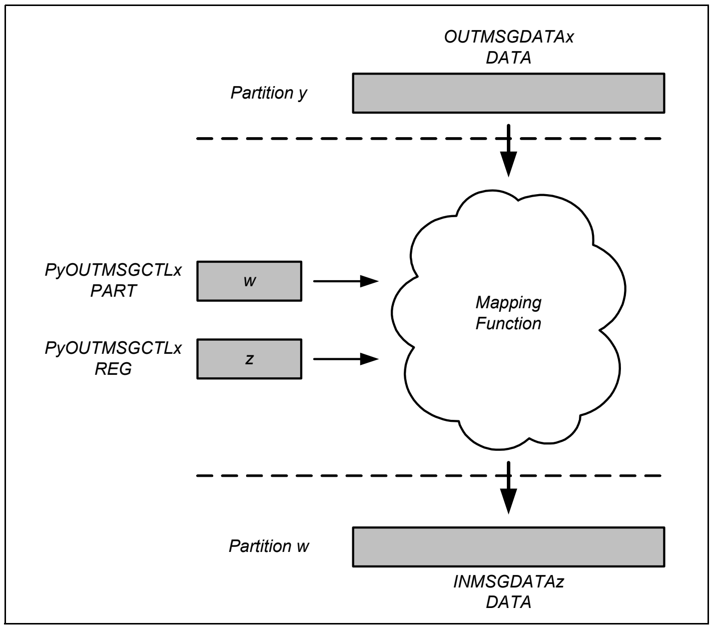


Figure 19 Logical Representation of Message Register Operation

Since the mapping of outbound message registers to inbound message registers need not be one-to-one, it is possible to map multiple outbound message registers, from typically different partitions, to a single inbound message register. In such a configuration, it is necessary to deal with possible contention to the inbound message register.

– When an outbound message register is written, the written value is transferred to the inbound message register specified by the corresponding SWPxMSGCTLy register. The transferred value may be accepted or rejected by the inbound message register.
– A transferred value is accepted if the message register is empty. When a transferred value is accepted, the Inbound Message (INMSG) field in the corresponding Inbound Message (INMSGx) register is updated with the transferred value, the Inbound Message Source Partition (SRC) field in the Inbound Message Source (INMSGSRC) register is updated with the partition number from which the message arrived, and the corresponding Inbound Message Status (INMSGSTSx) bit is set in the Message Status (MSGSTS) register. Once a transferred value is accepted, the inbound message register becomes full and remains full until the corresponding INTMSGSTSx bit is cleared.
– A transferred value is rejected if the message register is full. When a transfer value is rejected, the Outbound Message Status (OUTMSGSTSx) bit is set in the Message Status (MSGSTS) register that corresponds to the outbound message register that was written. This bit may be used to determine a transfer failed and needs to be retried.

Notes

Bits in the Message Status (MSGSTS) register may be used to generate NT interrupts. A bit in the MSGSTS register may be masked from generating an interrupt by setting the corresponding bit in the Message Status Mask (MSGSTSMSK) register.

Punch-Through Configuration Requests

The NT endpoint has the capability to generate PCI Express configuration transactions on the upstream link. This mechanism, referred to as *punch-through*, is provided to facilitate configuration of systems in which a root complex is not present in the PCI Express hierarchy associated with the NT endpoint. In essence, the NT endpoint may be crosslinked to another endpoint or to a switch device, and issue configuration requests to configure these devices.

Punch-through requests are always emitted on the NT function's link. In port operating modes with multiple functions (e.g., upstream switch port with NT function), it is not allowed for punch through requests issued by the NT function to hit the primary/secondary/subordinate window of the PCI-to-PCI bridge function or the bus/device/function ID associated with other functions in the port. Breaking this rule produces undefined results.

To generate a punch-through configuration transaction on the NT endpoint's link, the following sequence should be executed. Note that the registers that control punch-through requests are located in the configuration space of the NT function. These registers may be programmed via another port (i.e., using the global address space indirection registers) or via the SMBus interface.

1. Check if the punch-through configuration interface is busy by examining the Busy (BUSY) bit in the Punch-Through Configuration Status (PTCSTS) register (located in the configuration space of the NT function) and wait until the interface is not busy.
2. Configure the operation (e.g., read or write) in the Punch-Through Configuration Control (PTCCTL) register.
3. Write to the Punch-Through Configuration Data (PTCDATA) register to initiate the configuration read or write operation as selected by the OP field in the PTCCTL register.
   – This step causes the NT endpoint to emit a PCI Express configuration request TLP. The requester ID in the configuration request TLP is as follows.
     • The bus field is replaced by the captured bus number of the NT endpoint associated in the target partition.
     • The device and function fields are replaced by the value 0x4. This corresponds to device 0, function 4.
     • The tag field is set to 0x0.
   – In addition, the BUSY bit in the PTCSTS register is set to indicate a punch-through configuration transaction is in progress.
4. Wait for the operation to complete by polling the status of the Done (DONE) bit in the PTCSTS register.
   – The Done bit is set when the NT function receives a completion[1] whose destination ID matches the NT function's requester ID (see the requester ID description above).
5. Check the transaction completion status in the Status (STATUS) field of the PTCSTS register. If the initiated transaction was a read and it successfully completed, then the read result may be read from the PTCDATA register.
   – The STATUS field in the PTCSTS register reflects the status of the received completion (e.g., successful completion, unsupported request, completer abort, etc.)

It is possible for a completion to not be received in response to a punch-through configuration transaction. A punch-through operation may be aborted by writing a one to the DONE bit in the PTCSTS register. This will cause subsequent completions to be discarded until a new punch-through configuration transaction is generated. This mechanism should only be used when it is certain that a completion is lost and will never arrive. It is up to the user to make this determination.

[1]. The NT function assumes that the received completion is a completion with data (CplD) TLP and does not check for any violations in the format of the TLP.

Notes

Re-programming the Bus Number of the NT Function

In some systems, it may be desirable to use a PCI Express switch to interconnect several intelligent devices without the presence of a PCI Express root (i.e., the switch can be configured via SMBus or EEPROM). One of the challenges in building this type of system is the assignment of PCI Express requester IDs (i.e., bus, device, function) to each of the intelligent devices. Such assignment is a pre-requisite in order for ID-routed TLPs (i.e., completions) to be correctly routed by the PCI Express switch.

Normally, devices with a PCI Express port capture the bus number associated with the port on reception of type 0 configuration write requests that target the port. In system scenarios where there is no root complex in the PCI Express hierarchy, the devices will not receive type 0 configuration write requests. As a result, the default bus number (i.e., bus number 0) will be used by the devices, and ID-routing across the hierarchy won't be possible.

The switch contains a feature that allows software to explicitly configure the bus number associated with a switch port that has an NT function. The programming is done by writing to the Bus (BUS) field in the TLCNTCFG register located in the port's configuration space. Programming of the port's bus number is only allowed when the port operates in NT function mode or NT with DMA function mode.

Figure 20 shows a system scenario where a PCI Express switch is used to connect several intelligent devices. This system does not have a root complex, and communication among the intelligent devices is desired. Each intelligent device uses a switch NT port to connect to the PCI Express switch. Prior to initiating communication, the CPU located in the intelligent device programs the switch NT port that faces the rootless PCI Express hierarchy with an appropriate requester ID (i.e., by writing to the BUS field in the NT port's TLCNTCFG register; this register can be accessed by the processor on the intelligent device using the switch's global address space access registers).

Once the requester IDs for each intelligent device are programmed with unique values, traffic across the rootless PCI Express switch routes normally.

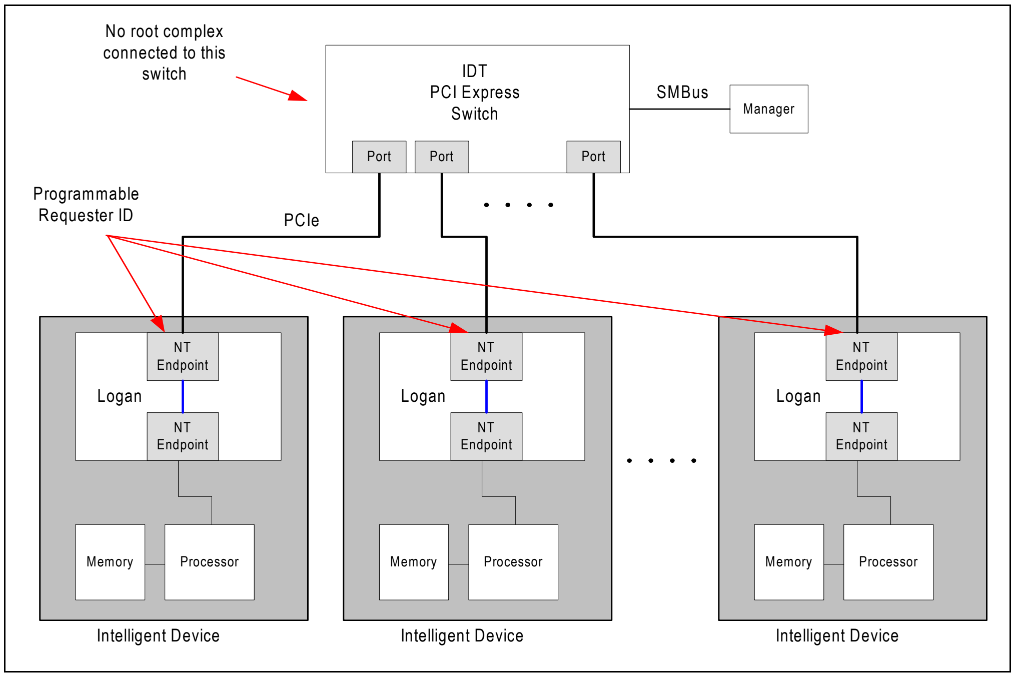


Figure 20 Example of a rootless PCI Express hierarchy with Bus Number Reprogramming

Notes

Interrupts

The NT endpoint has six sources of interrupts.

- Message status
- Doorbell status
- Switch signals to the partition with which the NT endpoint is associated
- Switch events
- Failover change initiated by the failover capability associated with the partition
- Failover change completed by the failover capability associated with the partition
- A temperature sensor alarm

The interrupt sources each have a corresponding status bin in the NT Endpoint Interrupt Status (NTINTSTS) register.

- When an interrupt source requests service, the corresponding bit in the NTINTSTS register is set.
- An interrupt source may be masked from generating an interrupt by setting the corresponding mask bit in the NT Endpoint Interrupt Mask (NTINTMSK) register. By default, all interrupt sources are masked.

When an unmasked interrupt condition occurs, an MSI or interrupt message is generated by the NT endpoint as described in Table 5. The removal of the interrupt condition occurs when unmasked status bits causing the interrupt are masked or cleared.

- When an NT endpoint is configured to generate INTx messages, the INTx used (i.e., INTA, INTB, etc.) depends on the programming of the Interrupt Pin (INTRPIN) register.

An MSI may be transmitted to a device (i.e., link partner) associated with any switch port of the partition, except for the cases below.

- An MSI must not target the NT or DMA functions (if present) in a switch partition. An MSI generated with an address that maps to the NT or DMA function within the switch partition produces undefined results.

Unmasked Interrupt	EN bit in MSICAP Register	INTXD bit in PCICMD Register	Action
Asserted	1	X	MSI message generated
	0	0	Assert_INTx message request generated
	0	1	None
Negated	1	X	None
	0	0	Deassert_INTx message request generated
	0	1	None

Table 5 NT Endpoint Interrupts

Virtual Channel Support

The NT function contains a VC Capability Structure that provides architected port arbitration and TC/VC mapping for VC0. For port operating modes in which the NT function is function 0 of the port, the VC Capability Structure in this function provides architected port arbitration and TC/VC mapping for all functions of the port. For port operating modes in which the NT function is not function 0 of the port, the registers in the NT function's VC Capability Structure are 'reserved'[1] and must not be programmed. In these modes, the VC Capability Structure in function 0 of the port provides architected port arbitration and TC/VC mapping for all functions of the port, including the NT function.

[1] Reading from a reserved address returns and undefined value. Writes to a reserved address complete successfully but produce undefined behavior on the register.

Notes

Maximum Payload Size

The switch requires that the Maximum Payload Size (MPS) field in the PCI Express Device Control (PCIEDCTL) register be set identically in all functions (i.e., PCI-to-PCI bridge, NT, and DMA) of a partition. In addition, when inter-partition transfers are possible between two or more partitions (i.e., across the NT interconnect), all switch functions in these partitions must have the same MPS setting. Violating this rule produces undefined results.

Note that a port with a maximum link width of x1 supports a Maximum Payload Size (MPS) of up to 1 KB. Ports with a maximum link width of x2, x4, or x8 support an MPS of up to 2 KB. The MPAYLOAD field in the PCI Express Device Capabilities (PCIEDCAP) register is automatically set by the hardware based on the port's maximum link width to reflect this.

Bus Locking

The NT function does not support bus locking. Memory read request-locked TLPs received by an NT function are treated as unsupported requests and an unsupported request completion with no data (CplLk) is returned. The operation of a switch partition is undefined when bus locking is performed in a partition that contains an NT function in its upstream port.

ECRC Support

End-to-End CRC (ECRC) is supported for transactions that are forwarded through the NT interconnect. Since the TLP contents (i.e., header) are modified for TLPs flowing between NT endpoints, a new ECRC must be computed. When a TLP is forwarded on to the NT interconnect by an NT endpoint, the NT endpoint computes the ECRC for the new translated TLP in parallel with checking the ECRC, if it exists, of the received TLP. The existence of an ECRC in the received TLP is indicated by the TD bit in the TLP header.

The NT function only checks and logs ECRC errors when the ECRC Check Enable (ECRCCE) bit is set in the function's AER Control (AERCTL) register, and the TLP with ECRC is received from the upstream port's link.

- ECRC error checking and logging is not performed by the NT function when it does not receive the TLP from the link.
  - In this case, the ECRC error checking and logging is done by the port that received the TLP from the link (e.g., downstream port).

If ECRC checking applicable as described above and an ECRC error is detected, then an ECRC error is reported by the NT endpoint that received the TLP. See Section , *Error Detection and Handling by the NT Function* for details.

- If ECRC checking is enabled in an NT endpoint, then ECRC is checked in all TLPs received by the NT endpoint that contain an ECRC. The reception of a TLP without ECRC is not considered an error (i.e., the TLP is processed normally).

ECRC generation is enabled in the NT endpoint when the ECRC Generation Enable (ECRCGE) bit is set in the function's AER Control (AERCTL) register.

If ECRC generation is enabled in the NT endpoint associated with the destination partition of the translated TLP, then the translated TLP contains an ECRC and the TD bit in the translated TLP header is set.

- If ECRC checking is not enabled in the NT endpoint that received the TLP, or if the received TLP does not contain an ECRC, or if ECRC checking is enabled and the ECRC computed by the NT endpoint is correct, then the ECRC associated with the translated TLP is that computed by the NT endpoint.
- If ECRC checking is enabled in the NT endpoint and the received TLP contains an ECRC error, then the ECRC associated with the translated TLP is equal to the ECRC computed by the NT endpoint with even bits inverted.

If ECRC generation is enabled in an NT endpoint, then all TLPs originated by that endpoint contain an ECRC.

Notes

Access Control Services (ACS)

The NT function supports the following ACS checks[1]:

- ACS Peer-to-Peer[2] Request Redirect
- ACS Peer-to-Peer Completion Redirect
- ACS Direct Translated Peer-to-Peer

ACS is programmed via the ACS Capability Structure in the NT function's configuration space.

- The NT function supports ACS checks when the port operates in the following port operating modes:
  - Upstream switch port with NT function
  - Upstream switch port with NT and DMA function
- In these modes, the ACS Capability Structure is linked into the NT function's configuration space.
- The NT function does not support ACS checks when the port operates in any other mode.

The NT function applies the above ACS checks for TLPs it emits (i.e., TLPs received on another partition that have undergone NT address translation).

- ACS checks are not applied to punch-through configuration requests issued by the NT function.

Table 6 lists ACS checking and handling performed by the NT function. Note that there are none of the ACS checks results in an ACS violation error.

ACS Check	PCI Express Specification[1] Section	Error Reporting Condition	Action Taken
ACS Peer-to-Peer (P2P) Request Redirect	6.12.1.1	N/A (not an ACS violation)	Offending request is redirected upstream towards root complex.
ACS P2P Completion Redirect			Offending completion is redirected upstream towards root complex.
ACS Direct Translated P2P			Offending TLP is subject to ACS P2P Request Redirect rules.

Table 6 ACS Checks Performed by the NT Function in a Port Operating in Multi-function Mode

1. Refer to [1]

When an ACS check causes a TLP to be re-directed, the re-direction is implemented such that TLPs emitted by the NT function that are ACS re-directed follow the ordering rules. Note that ACS Direct Translated Peer-to-Peer requires that the NT function perform a check on the Address Type (AT) field in request TLPs it emits. Prior to performing this ACS check, the AT field in the emitted TLPs is subject to the processing described in section Address Type Processing on page 18.

- If the NT function clears the AT field in a TLP it emits (i.e., the TLP is marked as untranslated), the ACS Direct Translated P2P check is reduced to an ACS P2P Request Redirect check.

ACS checks are only applicable to certain TLP types. Table 7 list the ACS checks supported by the NT function and the TLP types on which they are applied.

1. The switch does not support ACS Peer-to-Peer Egress Control among the functions of a multi-function upstream port.
2. In a multi-function upstream port, 'peer-to-peer' implies traffic exchanged among the port functions (e.g., from the port's NT function to the port's PCI-to-PCI bridge function).

Notes

ACS Check	Applicable to the following TLP Type(s)
ACS Peer-to-Peer (P2P) Request Re-direct	Peer-to-Peer Request TLPs
ACS P2P Completion Re-direct	Peer-to-Peer Completion TLPs
ACS Direct Translated P2P	Peer-to-Peer Memory Request TLPs

Table 7 TLP Types Affected by ACS Checks

As an example of an ACS check performed by the NT function, consider the case where software enables ACS Peer-to-Peer Request Redirect in the NT function. This commands the NT function to re-direct upstream (i.e., transmit on the upstream link) all requests that it issues which would have otherwise been logically routed via the upstream port's PCI-to-PCI bridge function. Figure Figure 21 shows an example of a ACS Peer-to-Peer Request Redirect. The green lines mark the requests intended route, and the orange lines the request's re-directed route due to ACS.

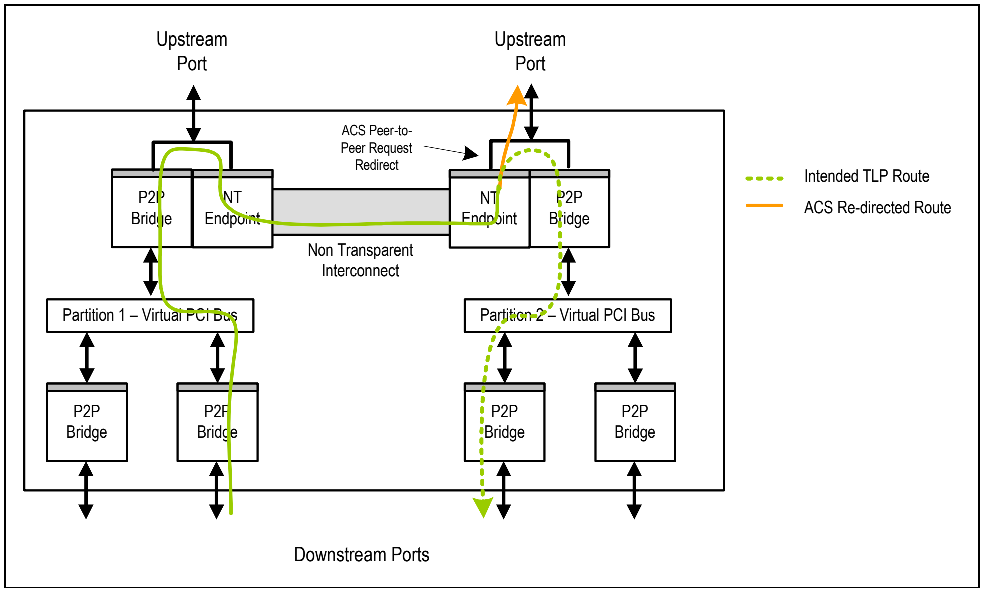


Figure 21 Example of ACS Peer-to-Peer Request Redirect Applied by the NT Function

When multiple ACS checks are enabled, they are prioritized. Refer to [1] for further information on ACS.

Error Detection and Handling by the NT Function

This section describes error conditions associated with non-transparent switch operation. This includes physical, data-link, and transaction layer errors detected by the switch ports, as well as application layer errors associated with the non-transparent-bridge (NTB) functionality.

– Internal switch errors (i.e., parity errors, switch time-out, and internal memory errors) are associated with the switch core and not with a specific port function.

The errors described here apply to ports that operate in a mode that includes the NT function (e.g., NT endpoint mode, upstream switch port with NT endpoint mode, etc.) This section focuses specifically on errors related to the NT function. Errors that affect all functions of the port (i.e., non function-specific errors) are noted where appropriate.

Error detection and handling in the switch follows the requirements in the PCI Express specification [1].

Notes

The error checking and handling described here is performed by each switch NT function. In cases where the error condition propagates among multiple NT functions (e.g., a poisoned TLP received by an NT function, passed across the NT interconnect, and emitted by the NT function in another partition), each NT function performs error checking and handling independently. Refer to section NTB Inter-Partition Error Propagation on page 36 for further details and examples on this.

The errors described below are associated with specific actions to log and report the error. The terms 'uncorrectable error processing' and 'correctable error processing' refer to the processing described in Section 6.2.5 of [1].

Errors which are not function-specific are logged in the corresponding status & logging registers of <u>all functions in the port</u>. Errors that are function-specific are logged in the status & logging register of the affected function only. Signaling of non function-specific errors follows the rules in Section 6.2.4 of [1].

Some of the errors described below are marked as function-specific when the "function claims the TLP". A function claims a TLP in the following cases:

- NT Endpoint function:
  - Address Routed TLPs: The TLP address falls within the address space range(s) programmed in the function's base address registers (BARs).
  - ID Routed TLPs: The TLP destination ID matches a <u>valid</u> entry in the NT function's mapping table or the NT function's bus/device/function assignment within the PCI Express hierarchy.
  - Implicit Route TLPs: Always.
- PCI-to-PCI Bridge function
- DMA function

For a port in Upstream Switch Port with NT Endpoint mode, the TLP is claimed when either one of its two functions claims the TLP.

Physical Layer Errors

All physical layer errors are non-function specific. These errors are described in the error handling section for the PCI-to-PCI bridge function.

Data Link Layer Errors

All data link layer errors are non-function specific. These errors are described in the error handling section for the PCI-to-PCI bridge function.

Transaction Layer Errors

Table 8 lists non-ACS error checks performed by the transaction layer and the action taken when an error is detected. ACS error checks & handling are discussed in section Transaction Layer Error Pollution on page 34.

Per [1], transaction layer errors are ignored in cases where the error is associated with a received packet for which the physical or data-link layers report an error. This prevents error pollution across the stack layers. Within the transaction layer, there are error pollution rules that resolve the cases where two or more errors are detected simultaneously. Refer to Section for details on transaction layer error pollution.

Notes

Error Condition	PCI Express Specification[1] Section	Function-Specific Error	Role Based (Advisory) Error Reporting Condition	Action Taken
Poisoned TLP received	2.7.2.2, 6.2.3.2.4.3	Yes	Advisory when the corresponding error is configured as non-fatal in the AERUESV register	Detected Parity Error bit (PCISTS.DPE) is set. If the poisoned TLP is a completion, the Master Data Parity Error Detected bit (PCISTS.MDPED) is set (if PCICMD.PERRE is set). If the TLP is received from the NT port's link (i.e., the port is the ultimate receiver): Non-advisory case: uncorrectable error processing. Advisory case: correctable error processing. Affected packet is forwarded across the NTB to the destination partition's NT Endpoint (unless the TLP does not hit an NT mapping window, in which case it is dropped). If the packet is forwarded across the NTB, the following are the actions taken by the NT function in destination partition: Master Data Parity Error Detected (PCISTS.MDPED) is set if the poisoned TLP is a write request.
ECRC check failure[2]	2.7.1	No	N/A (always non-advisory)	Affected packet's ECRC is modified by inverting even bits in the ECRC and packet is forwarded across the NT bridge (unless the TLP does not hit an NT mapping window, in which case it is dropped). If the TLP is received from the NT port's link (i.e., the port is the ultimate receiver): Uncorrectable error processing.
Unsupported request	See Table 9	Yes if a function in the port claims the TLP. Else No.	Advisory when the corresponding error is configured as non-fatal in the AERUESV register and the request is non-posted	Non-advisory case: uncorrectable error processing. Advisory case: correctable error processing. For Non-Posted unsupported requests, the function that claims the TLP generates a completion with UR status. If the request is not claimed, then function 0 generates the completion with UR status.

Table 8 Transaction Layer Errors Associated with the NT Function (Page 1 of 2)

Notes

Error Condition	PCI Express Specification[1] Section	Function-Specific Error	Role Based (Advisory) Error Reporting Condition	Action Taken
Completion time-out	2.8	N/A	N/A (always non-advisory)	Not applicable. The NT function does not check for completion tim-eout as it does not track requests issued across the NT bridge.
Completer abort	2.3.1	N/A	N/A (always non-advisory)	Not applicable. The NT function does not issue completions with 'Completer Abort' status except for ACS violations. For this exception case, the error is considered an ACS error and is not logged as a completer abort error.
Unexpected completion received	See Table 10	Yes if a function in the port claims the TLP. Else No.	Advisory when the corresponding error is configured as non-fatal in the AERUESV register	Non-advisory case: uncorrectable error processing. Advisory case: correctable error processing. The unexpected completion is dropped.
Completion with UR status received[3]	6.2.3.2.5	N/A	N/A (always non-advisory)	Reception of a completions with UR status is handled as any other received completion. In addition, the Received Master Abort Status (RMAS) bit in the PCISTS register is set.
Completion with CA status received[4]	6.2.3.2.5	N/A	N/A (always non-advisory)	Reception of a completions with CA status is handled as any other received completion. In addition, the Received Target Abort Status (RTAS) bit in the PCISTS register is set.
Receiver overflow	2.6.1.2	No	N/A (always non-advisory)	Uncorrectable error processing. TLP is nullified.
Flow control protocol error	2.6.1	No	N/A (always non-advisory)	Not applicable. The switch does not check for any flow control protocol errors.
Malformed TLP	See section TLP Malformation Checks on page 32	No	N/A (always non-advisory)	Uncorrectable error processing. TLP is nullified.
Internal Error	Internal Error Reporting ECN	Yes	N/A (always non-advisory)	

Table 8 Transaction Layer Errors Associated with the NT Function (Page 2 of 2)

1. Refer to [1].
2. Refer to section ECRC Support on page 25.
3. If the completion is unexpected, then it is handled as an unexpected completion received error.
4. If the completion is unexpected, then it is handled as an unexpected completion received error.

Notes

Unsupported Requests

Table 9 lists the conditions for which the NT function handles requests as unsupported requests (UR).

Conditions Handled as UR	Description	PCI Express Specification Section
Effective BAR Aperture check	Refer to section BAR Limit on page 8.	n/a
Lookup Table Address Translation error: Table entry invalid or partition entry invalid.	Refer to section Lookup Table Address Translation on page 10.	n/a
Destination Partition errors: Destination partition not active, destination partition does not have an NT endpoint, destination partition is same as source partition, or BME bit cleared in NT function of destination partition when translating a request TLP.	Refer to section Lookup Table Address Translation on page 10, and section Request ID Translation on page 16	n/a
Requester ID miss in NT Mapping Table	Refer to section Request ID Translation on page 16	n/a
NT function in D3Hot state		5.3.1.4.1
NT function in destination partition in D3Hot state		n/a
Type 1 Configuration Requests	Reception of a Type 1 configuration request at the NT function.	7.3.3
Vendor Defined Type 0 message reception[1]	Vendor Defined Type 0 message which targets the NT function.	2.2.8.6
Messages with invalid message code	Reception of a message TLP with invalid message code that targets the NT function.	2.3.1
Poisoned IO request, memory write request, or message with data targeting the NT function	Reception of a poisoned IO request, memory write request, or message with data (except Vendor Defined messages) that targets a switch port's NT function.	2.7.2.2
Reception of MRdLk Request	Reception of an MRdLk request by the NT function	6.5.7

Table 9 Conditions Handled as Unsupported Requests (UR) by the NT Function

1. NOTE: Vendor Defined Type 1 messages which target the PCI-to-PCI bridge function are silently discarded.

Notes

Unexpected Completions

Table 10 lists the conditions for which the NT function handles completions as unexpected completions.

Conditions Handled as UC	Description	PCI Express Specification Section
Non function-specific unexpected completion	Port receives a completion TLP that is not claimed by any function of the port. This is a non function-specific error and is therefore logged in all functions of the port.	6.2.4
NT function unexpected completion	Port receives a completion TLP whose requester ID matches the NT function's bus/dev/function ID (i.e., zero or one depending on the port operating mode) and a punch-through operation is not in progress[1]. This is a function-specific error associated with the NT function. Note that this does not refer to a completion TLP whose requester ID matches a valid entry in the NT function's mapping table. Such TLPs are handled as described in Section .	2.3.2

Table 10 Conditions Handled as Unexpected Completion (UC) by the NT Function

[1] If a punch-through configuration transaction is in progress, then the completion is accepted by the NT function and processed as described in section Punch-Through Configuration Requests on page 22.

TLP Malformation Checks

TLP malformation checks are done by the port and are non function-specific. Table 11 lists the TLP malformation checks performed by a switch port on reception of TLPs. These checks are performed whenever the port receives the packet from the link.

TLP Type	Error Check
All	TLP must have a valid FMT/TYPE combination Data payload length <= Max_Payload_Size (i.e., MPS field in PCIEDCTL register)
All TLPs with data (i.e., FMT[1]=1)	LENGTH field must match actual payload data
All TLPs with ECRC (i.e., TD=1)	Actual TLP length must match calculated length (HEADER + PAYLOAD + ECRC)

I/O read or write request	LENGTH = 1 (doubleword)
	TC = 0
	ATTR = 0
	Last DWord BE[3:0] = 0b0000

Configuration read or write request	LENGTH = 1 (doubleword)
	TC = 0
	ATTR = 0
	Last DWord BE[3:0] = 0b0000

Table 11 Ingress TLP Formation Checks associated with the PCI-to-PCI Bridge Function (Page 1 of 2)

Notes

TLP Type	Error Check
Message Requests interrupt message Power management message Error signalling message Unlock message Set power limit message	TC = 0
TLPs with Route to Root Complex routing.	May only be received on downstream switch ports
TLPs with Broadcast from Root Complex routing.	May only be received on upstream ports
TLPs with Gathered and Routed to Root Complex routing	May only be received by the downstream switch ports Must be a PME_TO_ACK message (all other TLP types with this routing are illegal)
Interrupt messages (INTx)	May only be received by the downstream switch ports
All	TLP traffic class (TC) must be mapped to VC0. TC to VC mapping is controlled by the TC/VC Map (TCVCMAP) field in the VC Capability Structure associated with function 0 of the ingress port.

Table 11 Ingress TLP Formation Checks associated with the PCI-to-PCI Bridge Function (Page 2 of 2)

Table 12 lists the TLP formation error checks performed whenever a port transmits a packet.

TLP Type	Error Check
All	TLP traffic class (TC) must be mapped to VC0. TC to VC mapping is controlled by the TC/VC Map (TCVCMAP) field in the egress port's VC Resource 0 Control (VCR0CTL) register of the PCI-to-PCI bridge function.

Table 12 Egress Malformed TLP Error Checks

TLP Header Logging

TLP header logging is subject to the rules outlined in section 6.2.5 of [1], as well as the Internal Error Reporting ECN [3]. The following non function-specific errors require that the offending TLP's header be logged in the AER capability structure of all function's in the port.

– Reception of a TLP with ECRC error on the port's link.
– Reception of a request that is unsupported on the port's link, when no function in the port claims the TLP.
– Reception of an unexpected completion on the port's link, when no function in the port claims the TLP.
– Reception of a malformed TLP on the port's link.

The following function-specific errors require that the offending TLP's header be logged in the NT function's AER capability structure.

– Reception of a request that is unsupported and is claimed by the NT function.
– Reception of an unexpected completion that is claimed by the NT function.
– Reception of a poisoned TLP on the upstream port's link that is claimed by the NT function.
  • When the TLP is not received on the link, header logging is not performed.

Transaction Layer Error Pollution

Per section 6.2.3.2.3 of [1], transaction layer errors may be prioritized to prevent error pollution in AER. Error pollution rules only apply to errors associated with the reception of a TLP. Errors not associated with the reception of a TLP are logged for each occurrence of the error.

In addition, the Detected Parity Error bit (DPE) in the PCISTS and SECSTS registers is not subject to error pollution rules and is therefore set when the PCI-to-PCI bridge receives a poisoned TLP on its primary or secondary side respectively, even if error pollution rules indicate that the poisoned TLP received error is superseded by a higher priority error.

Table 13 shows the prioritization of transaction layer errors used by switch ports. Except for internal errors, all the errors listed in the table are associated with the reception of a TLP[1]. Errors not applicable to the switch (e.g., completion timeout and completer-abort) are not shown. Higher priority errors have precedence over lower priority errors. Errors with the same priority are mutually exclusive (the errors can't occur simultaneously).

Error	Associated with Packet Reception	Priority
Internal Error	Depends on error	8 (highest)
Receiver Overflow	Yes	7
ECRC Check failure	Yes	6
Malformed TLP received	Yes	5
ACS Violation	Yes	4
Multicast Blocked TLP	Yes	3
Unsupported Request	Yes	2
Unexpected Completion received	Yes	
Poisoned TLP received	Yes	1 (lowest)

Table 13 Prioritization of Transaction Layer Errors

The prioritization of errors shown in Table 13 determines the error that is logged and reported when multiple errors are detected simultaneously for the received TLP. Higher priority errors inhibit the logging and reporting of lower priority errors in AER.

Still, higher priority errors do not inhibit the checking and TLP handling action of lower priority errors, unless the higher priority error results in the TLP being consumed, dropped, or nullified by the detecting function.

Figure 22 shows the decision diagram for error checking and logging on a received TLP taking into account the error pollution rules and priorities.

[1]. Internal errors may be associated with a packet (i.e., double-bit memory error when extracting a TLP from the IFB or EFB) or may not associated with a packet (i.e., switch time-out).

Notes

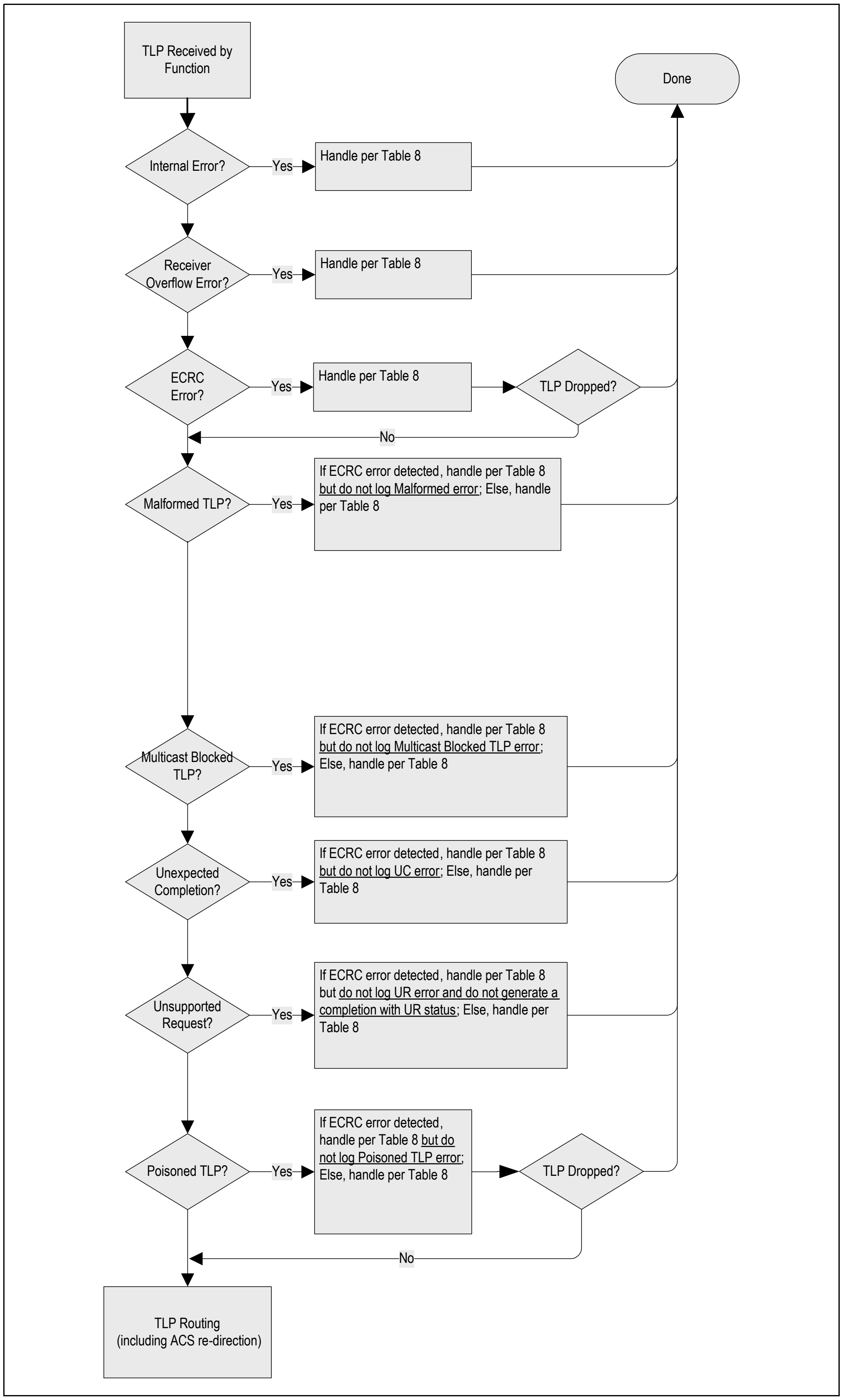

TLP Received by Function
Done
Internal Error?
Yes
Handle per Table 8
Receiver Overflow Error?
Yes
Handle per Table 8
ECRC Error?
Yes
Handle per Table 8
TLP Dropped?
No
Malformed TLP?
Yes
If ECRC error detected, handle per Table 8 but do not log Malformed error; Else, handle per Table 8
Multicast Blocked TLP?
Yes
If ECRC error detected, handle per Table 8 but do not log Multicast Blocked TLP error; Else, handle per Table 8
Unexpected Completion?
Yes
If ECRC error detected, handle per Table 8 but do not log UC error; Else, handle per Table 8
Unsupported Request?
Yes
If ECRC error detected, handle per Table 8 but do not log UR error and do not generate a completion with UR status; Else, handle per Table 8
Poisoned TLP?
Yes
If ECRC error detected, handle per Table 8 but do not log Poisoned TLP error; Else, handle per Table 8
TLP Dropped?
No
TLP Routing (including ACS re-direction)

Figure 22 Error Checking and Logging on a Received TLP

Notes

NTB Inter-Partition Error Propagation

This section describes the switch's handling of error conditions that propagate across the device during inter-partition transfers. This section builds upon the error handling concepts described above for the NT function.

Figure 23 shows a basic configuration of the switch with two partitions connected by an non-transparent bridge (NTB). Note that this is only a sample configuration chosen to illustrate the concepts described in this section. These concepts are applicable to all other possible multi-partition configurations allowed by the switch.

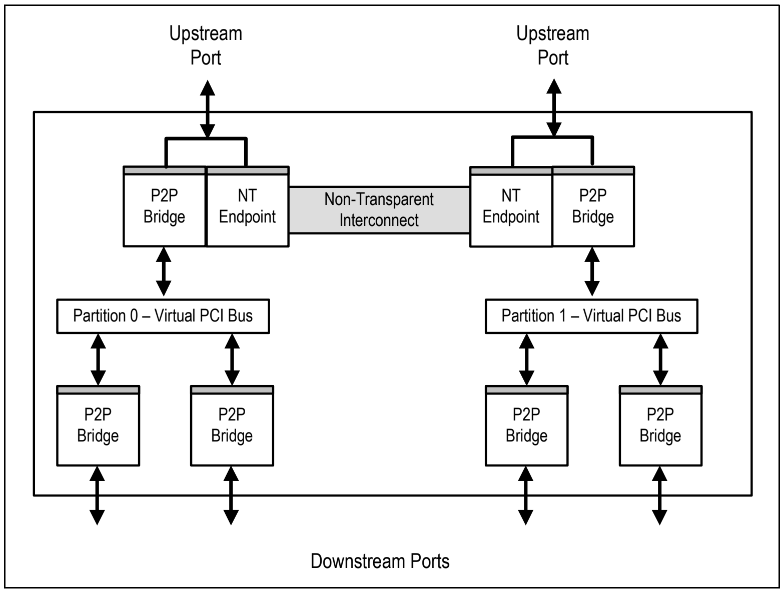


Figure 23 Basic Non-Transparent Switch Configuration

In the sample configuration shown in the figure, each partition is composed of three PCI-to-PCI (P2P) bridge functions (i.e., a 3-port transparent switch). The NTB is associated with an NT function in each partition, and the upstream port of each partition is configured in Upstream Switch Port with NT Endpoint mode.

Note that both functions of the upstream port (i.e., PCI-to-PCI bridge function and NT function) connect to the PCI Express link, as well as to each other. Each of the logical functions shown in the figure has independent error detection and handling capabilities. The NT functions performs error handling as described in the sections above.

When a TLP is routed across the switch (within a partition or across partitions via the NTB), each function that receives the TLP checks for errors. Thus, it is possible that more than one function detect and report an error associated with the TLP.

This section focuses on errors related to TLP routes that cross the NTB (i.e., inter-partition routes). Routes that do not cross the NTB (i.e., intra-partition routes) are not discussed here as these are well understood from the error handling description in the PCI Express specification [1].

The following sections describe error propagation for the different types of errors detected by the switch ports.

Physical Layer Errors

Physical layer errors are only detected by the port that receives the packet from the link. No other ports or functions in the logical path of the TLP check for physical layer errors.

Data Link Layer Errors

Data link layer errors are detected by the ingress port that receives the packet from the link, or by the egress port (if any) that transmits the packet. No other ports or functions in the logical path of the TLP check for data link layer errors.

Transaction Layer Errors

Transaction Layer errors associated with the reception of a TLP are checked by each switch function that receives the TLP. In some cases, the first function in the TLP's logical path that detects an error will nullify or drop the TLP. As a result, no other functions in the TLP's logical path will detect the error condition.

In other cases, the first function in the TLP's logical path that detects an error will log the error appropriately and allow the TLP to continue in its path. As a result, other functions in the TLP's logical path may check, detect, and log errors associated with the TLP. In this case, it is possible that multiple functions detect and report errors associated with the TLP.

For inter-partition transfers, functions in both partitions may detect and report errors associated with the TLP. A function that detects an error and is configured to signal the error by issuing an error message (i.e., Fatal, Non-Fatal, or Correctable message), does so by issuing the message towards the upstream port's link associated with the function's partition.

Additionally, for certain types of errors (e.g., unsupported request on a non-posted request), it is possible that the function that detects the error generate a completion TLP destined to the original sender of the TLP (i.e., the requester). In the case where the request TLP was received by the switch on one partition, crossed the NTB, and the error was found in a second partition, the completion TLP generated by the function that detects the error in the second partition logically crosses the NTB and is sent towards the requester in the first partition.

The following sections describe error propagation for each type of transaction layer error.

Receiver Overflow Error

Receiver overflow errors are only detected by the port that receives the TLP from the link. The TLP is nullified and no other ports or functions in the logical path of the TLP will detect this type of error.

ECRC Error

ECRC errors are only detected and logged in AER by the function(s) in the port that received the TLP from the link. No other functions in the logical path of the TLP log the ECRC error. An NT function that receives a TLP with ECRC error handles it as described in section ECRC Support on page 25.

Malformed TLP Error

In the switch, malformed TLP errors are checked at the ingress port that receives the packet from the link, or at the egress port (if any) that transmits the packet. Malformed TLPs are nullified by the function that detects the error and thus no other functions in the logical path of the TLP will detect this type of error.

Note that since TLP malformation checks are only performed by the ingress port that receives the packet from the link and by the egress port that transmits the packet to the link, intermediate functions in the TLPs logical path do not detect formation errors. To prevent device malfunction, the user must guarantee that:

– The TC/VC mappings of all functions in the switch are configured such that incoming TLPs are always mapped to VC 0.

– The Maximum Payload Size (MPS) setting of all functions in the switch is identical.

Unsupported Request (UR) Error

Each switch function checks for UR errors upon reception of a request TLP (i.e., posted or non-posted). If the function determines the request is unsupported, the TLP is consumed by this function and is handled as a UR error. Else, the TLP is consumed and processed normally by the function, or forwarded across the function (e.g., PCI-to-PCI bridge), depending on its final destination. For non-posted requests, a function that treats the request as a UR must generate a completion TLP and send it to the original requester.

Notes

In the switch, it is possible that a non-posted TLP that is routed across partitions via the NTB (i.e., from a source partition to a destination partition) be UR-ed by a switch function in the destination partition. In this case, the completion TLP generated by the function that detects the error is logically routed back within the switch towards the initiator of the request. Thus, the completion TLP generated by the function in the destination partition logically crosses the NTB and is routed towards the request initiator in the source partition.

Figure 24 shows an example of a non-posted request TLP received by an switch port on a first partition that is transferred across the NTB to a second partition. The request is URed by the PCI-to-PCI bridge function in the switch upstream port of the second partition.

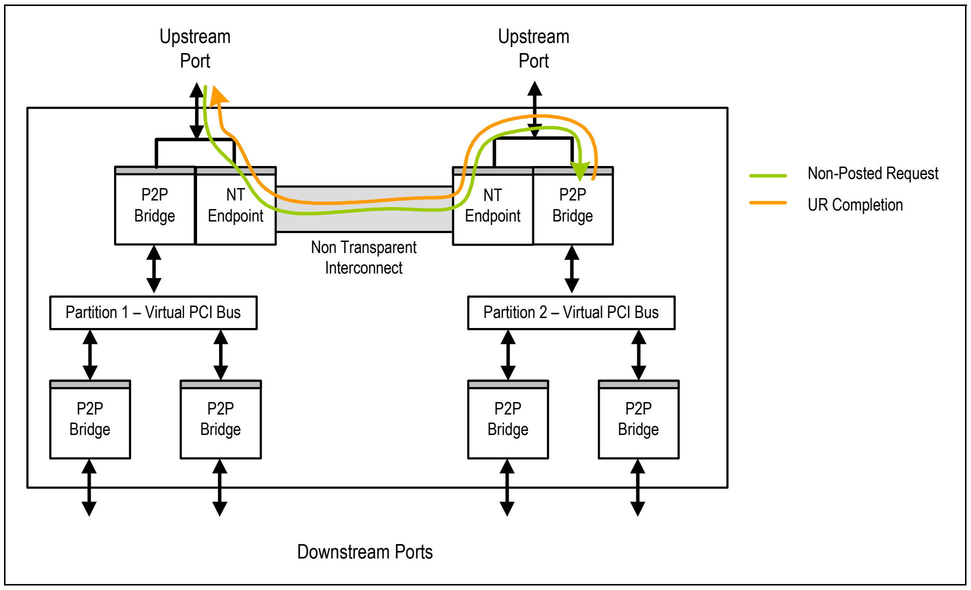


Figure 24 Unsupported Request Example #1

When the PCI-to-PCI bridge function in Partition 2 detects the UR error, it logs it as such and generates a completion TLP destined towards the NT Endpoint function associated with Partition 2. The completion TLP is then transferred across the NTB and transmitted by the NT Endpoint in Partition 1 towards the initiator of the request.[1]

Note that each function that receives the TLP (i.e., NT function in partition 1 and PCI-to-PCI bridge function in partition 2) checks for UR errors. Also, notice that the request TLP logically stops at the PCI-to-PCI bridge function in partition 2 since this function detects the UR error. For this example, error logging would occur as shown in Table 14.

Function	Error Logging
NT Endpoint (Partition 2)	Refer to row corresponding to 'Completion with UR status received' in Table 8
NT Endpoint (Partition 1)	No error logged.

Table 14 Error Logging at Each Function for UR Example # 1

[1] Note that the completion is guaranteed to cross the NTB since it corresponds to a request that had previously crossed the NTB. That is, the completion's destination ID will hit a valid entry in the NT mapping table.

Notes

Figure 25 shows another example of an unsupported request condition for an inter-partition transfer. In this case, a non-posted request TLP received by the downstream switch port in a first partition is logically routed towards the NT endpoint in the same partition. The request TLP is then transferred across the NTB to a second partition. The request is then URed by the PCI-to-PCI bridge function in the switch downstream switch port in the second partition.

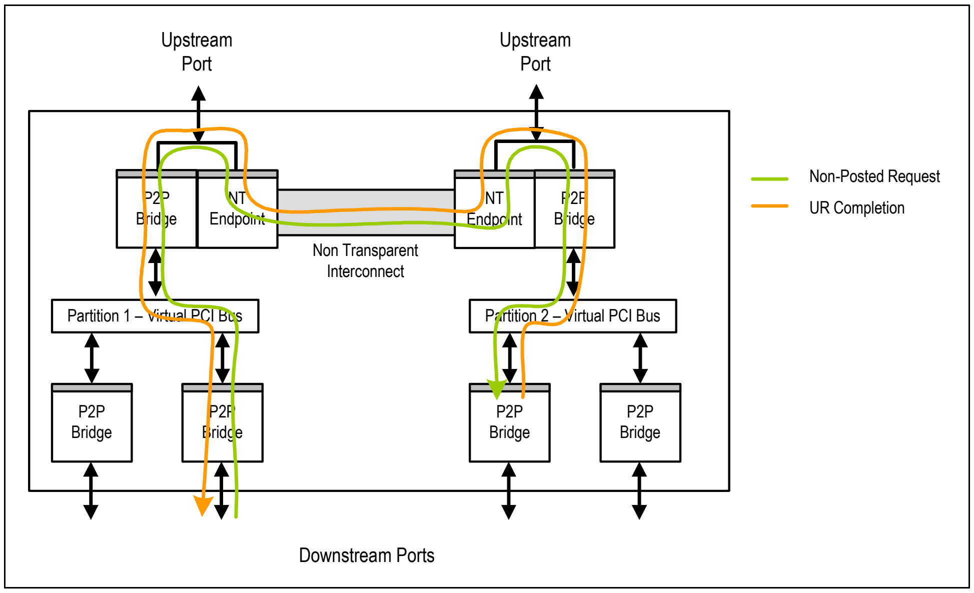


Figure 25 Unsupported Request Example # 2

When the downstream switch port's PCI-to-PCI bridge function in Partition 2 detects the UR error, it generates a completion TLP destined towards the NT Endpoint function associated with Partition 2. The completion TLP is then transferred across the NTB and sent by the NT Endpoint in Partition 1 towards the initiator of the request. Again, note that each function that receives the TLP (i.e., NT function in partition 1, PCI-to-PCI bridge function of the upstream port in partition 2, and PCI-to-PCI bridge function of the downstream switch port in partition 2) checks for UR errors. Also, notice that the request TLP logically stops at the PCI-to-PCI bridge function of the downstream switch port in partition 2 since this function detects the UR error.

For this example, error logging occurs as shown in Table 15.

Function	Error Logging
NT Endpoint (Partition 2)	Refer to row corresponding to 'Completion with UR status received' in Table 8
NT Endpoint (Partition 1)	No error logged.
Upstream PCI-to-PCI Bridge (Partition 1)	No error logged.
Downstream PCI-to-PCI Bridge (Partition 1)	No error logged.

Table 15 Error Logging at Each Function for UR Example # 2

Unexpected Completion Received Error

Unexpected completion TLPs are dropped by the function that detects the error. Therefore, no other functions in the logical path of the TLP will detect this type of error.

Notes

Poisoned TLP Received Error

Poisoned TLP errors may propagate across multiple switch ports. The following rules apply:

- All functions in the logical path of the poisoned TLP log the error in the PCI legacy error registers.
  - The PCI-to-PCI bridge function logs the error in the PCI Status (PCISTS) or Secondary Status (SECSTS) registers as appropriate.
  - The NT and DMA functions log the error in the PCISTS register.
- The function that claims the TLP in the port that receives the poisoned TLP from the link logs the error in its AER Capability Structure and handles it appropriately.
  - An NT function that receives a poisoned TLP handles it as described in Table 8.

Figure 26 shows an example of a poisoned TLP propagating across the switch during an inter-partition transfer. As the TLP is logically routed from the downstream switch port in a first partition, across the NTB, to the downstream switch port in a second partition, each function in the TLP's path logs the poisoned TLP error per the rules described above.

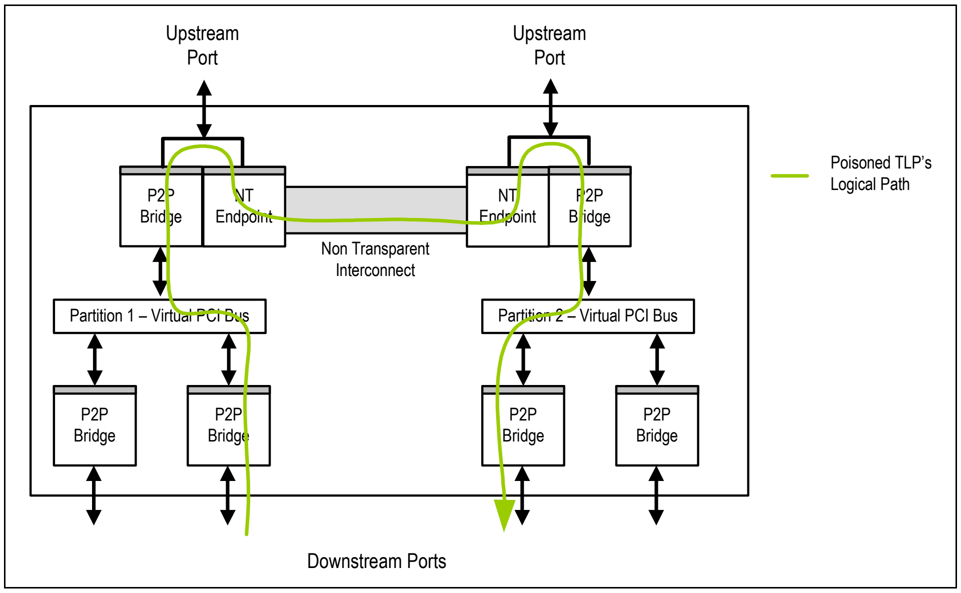


Figure 26 Poisoned TLP Error Propagation Example

Table 16 shows the error logging for each function in the TLP's logical path.

Notes

Function	Error Logging
Downstream PCI-to-PCI Bridge (Partition 1)	Refer to [1]. Note that this port is considered the ultimate receiver in partition 1, as it is the port that receives the TLP on the link and the TLP targets the NT function in partition 1's upstream port.
Upstream PCI-to-PCI Bridge (Partition 1)	Refer to [1]. Note that the bridge only logs parity error reception in the SECSTS register as it did not receive the TLP directly from the link.
NT Endpoint (Partition 1)	Refer to row corresponding to 'Poisoned TLP received' in Table 8. Note that the NT function only logs parity error reception in the PCISTS register as it did not receive the TLP directly from the link.
NT Endpoint (Partition 2)	Refer to row corresponding to 'Poisoned TLP received' in Table 8. In the table, this NT function is considered the "NT endpoint in the destination partition".
Upstream PCI-to-PCI Bridge (Partition 2)	Refer to [1]. Note that the bridge only logs parity error reception in the PCISTS register as it did not receive the TLP directly from the link.
Downstream PCI-to-PCI Bridge (Partition 2)	Refer to [1]. Note that the bridge only logs parity error reception in the PCISTS register as it did not receive the TLP directly from the link.

Table 16 Error Logging at Each Function for Poisoned TLP Example

ACS Errors

ACS checks may cause the offending TLP to be dropped or re-directed towards the root-complex by the detecting function. In cases where the TLP is dropped, no other functions in the TLP's logical path detect the ACS error. In cases where the TLP is re-directed towards the root-complex, other functions in the TLP's logical path may perform ACS checks on the TLP.

ACS re-direction may occur at three logical points in a TLPs path:

– Downstream switch port: A TLP received on a downstream switch port that is destined towards another downstream switch port in the same partition is re-directed towards the root-complex.
– Upstream port's PCI-to-PCI Bridge function: A TLP received on the upstream PCI-to-PCI bridge function's secondary side that is destined towards the NT function in the same port is re-directed towards the root-complex.
– Upstream port's NT function: A TLP transmitted by the upstream NT function's that is destined towards the PCI-to-PCI bridge function in the same port is re-directed towards the root-complex.

Refer section Transaction Layer Error Pollution on page 34 for further details on ACS re-direction.

Combined Transaction Layer Errors

As a TLP is logically routed across the switch functions, it is possible for functions to detect different types of transaction layer errors for the TLP. For example, on reception of a TLP, a PCI-to-PCI bridge function may log a poisoned TLP error and forward the TLP to the next function in its logical route. This next function may detect an unsupported request (UR) error on the TLP and handle it accordingly[1].

Notes

In general, when a first function receives a TLP, detects an error, and forwards the TLP to the next function on the route, the next function in the TLP's path may not detect an error, may detect the same error, or may detect a higher priority error.

Figure 27 shows an example of a poisoned non-posted TLP received on a downstream switch port of a first partition. As the TLP propagates within the first partition, across the NTB, and towards a second partition, the functions in the logical path of the TLP log the poisoned TLP error reception appropriately. As the TLP reaches a downstream switch port in the second partition, the port's PCI-to-PCI bridge function detects that the non-posted TLP's request is unsupported (e.g., the downstream switch port's link is down). As a result, the downstream switch port handles the TLP as an supported request error and generates a completion TLP with UR status.

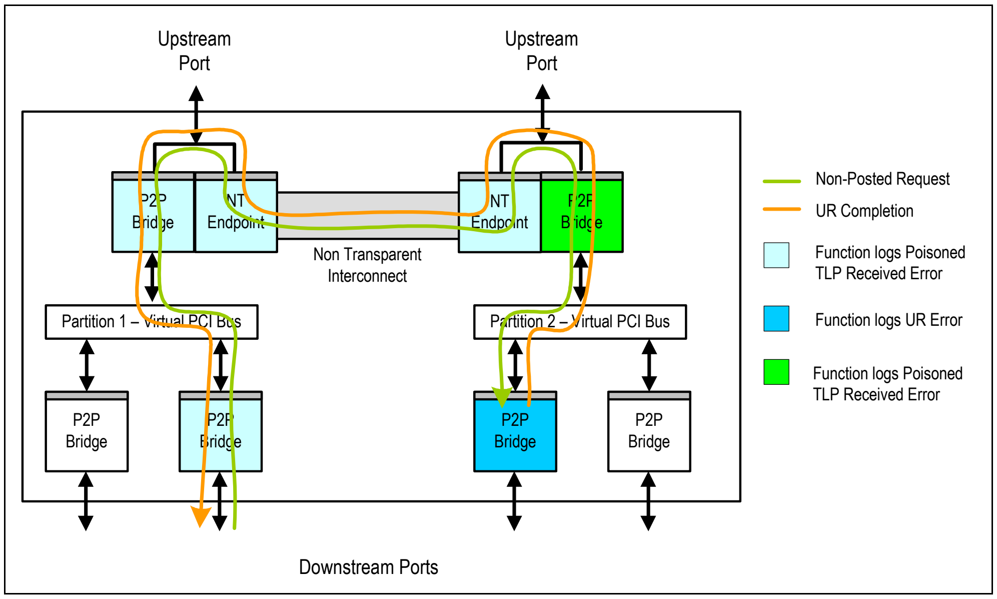


Figure 27 Example of Combined Transaction Layer Error Handling

Table 17 shows the error logging for each function in the TLP's logical path.

Function	Error Logging
Downstream PCI-to-PCI Bridge (Partition 1)	Refer to [1]. Note that this port is considered the ultimate receiver in partition 1, as it is the port that receives the TLP on the link and the TLP targets the NT function in partition 1's upstream port.
Upstream PCI-to-PCI Bridge (Partition 1)	Refer to [1]. Note that the bridge only logs parity error reception in the SECSTS register as it did not receive the TLP directly from the link.
NT Endpoint (Partition 1)	Refer to row corresponding to 'Poisoned TLP received' in Table 8. Note that the NT function only logs parity error reception in the PCISTS register as it did not receive the TLP directly from the link.

Table 17 Error Logging at Each Function for Poisoned TLP Example (Page 1 of 2)

[1] Note that UR errors have higher precedence than poisoned TLP errors per the error pollution rules described in Section .

Function	Error Logging
NT Endpoint (Partition 2)	Refer to row corresponding to 'Poisoned TLP received' in Table 8. In the table, this NT function is considered the "NT endpoint in the destination partition". Additionally, refer to row corresponding to 'Completion with UR status received' in Table 8
Upstream PCI-to-PCI Bridge (Partition 2)	Refer to [1]. Note that the bridge only logs parity error reception in the PCISTS register as it did not receive the TLP directly from the link.
Downstream PCI-to-PCI Bridge (Partition 2)	Refer to [1].

Table 17 Error Logging at Each Function for Poisoned TLP Example (Page 2 of 2)

Note that the NT function in partition 2 logs the reception of the poisoned TLP reception, as well as the reception of a completion TLP with UR status (i.e., the completion TLP logically generated by the downstream PCI-to-PCI bridge function in partition 2).

Error Emulation Control in the NT Function

The switch provides the capability to emulate error occurrence in the AER uncorrectable and correctable error status registers. Associated with the NT function are two error emulation registers. The NT function Uncorrectable Error Emulation (NTUEEM) and NT function Correctable Error Emulation (NTCEEM) registers allow emulation of errors in the NT function.

When a bit in these registers is set, it causes the corresponding bit in the AER uncorrectable or correctable status register to be set. Please refer to the definition of the error emulation registers for further details. In addition, the First Error Pointer (FEPTR) field in the AER Control (AERCTL) is set appropriately by the hardware such that it points to the first error bit set in the AERUES register. Refer to the description of the FEPTR field.

Finally, note that the AER Header Log registers (AERHL[1:4]DW) have RWL type, such that they may be modified when the REGUNLOCK field in the Switch Control (SWCTL) register is set.

Non-Transparent Operation Restrictions

The following lists usage restrictions associated with non-transparent operation.

– TLPs received and translated by the NT function must <u>not</u> map into an NT BAR aperture in the destination partition.
– TLPs received and translated by the NT function must <u>not</u> map into the BAR 0 aperture of the DMA function in the destination partition.[1]
– TLPs received and translated by the NT function must <u>not</u> map into a multicast BAR aperture in the destination partition.
– The PCI Express specification mandates that a requester not issue memory requests whose address/length combination crosses 4 KB boundaries. To honor this requirement, the translated base address(es) in the NT function should be programmed such that the twelve lower bits are set to 0x0.

[1]. Note that DMA BAR 0 maps the configuration registers of the DMA to PCI memory space. Still, this restriction does not imply that it is not possible to configure the DMA function using requests issued by an agent in another partition. Such an operation is still possible and is accomplished by accessing the DMA configuration registers via the switch's global address space.

Notes

References

1. PCI Express Base Specification Revision 2.0., December 20, 2006, PCI-SIG.
2. Address Translation Services Specification, March 8, 2007, PCI-SIG.
3. Internal Error Reporting Engineering Change Notice to 1 above, April 24, 2008, PCI-SIG.

Revision History

December 10, 2008: Initial publication.

May 12, 2009: Added the following new sections: Re-programming the Bus Number of the NT Function, Max Payload Size, and Non-Transparent Operation Restrictions. Revised the following sections: Punch-Through Configuration Requests, Lookup Table Address Translation (note that the number of entries in the Lookup Table was reduced from 64 to 32), and Virtual Channel Support. Replaced old heading "TLP Translation" with new heading "Base Address Registers (BARs)" and updated that entire section. Moved heading "TLP Translation" to page 9.