Use of Non-transparent Bridging with IDT PCI Express® PCIe Gen1 NTB Switches

Application Note AN-510

Notes

By Kwok Kong

Overview

This application note describes how a Non-transparent Bridge (NTB) may be used to connect two Root Complexes. It describes the architecture of NTB, the enumeration process, transaction routing including address and ID translations, interprocessor communication mechanism, error handling, and initialization procedures. All descriptions and examples are based on the IDT 89HPES24NT3 device (referred to hereafter as PES24NT3).

Introduction

There are three basic types of devices in a native PCI Express (PCIe®) system; Root Complexes, PCIe switches, and Endpoints. There is only a single Root Complex in a PCIe tree. A Root Complex is a single processor sub-system which includes a single PCIe port, one or more CPUs with associated RAM and memory controller, and other inter-connect and/or bridging functions.

PCI Express routes are based on memory address or ID, depending on the transaction type. Thus, every device (or function within a device) must be uniquely identified within a PCI Express tree. This requires a process called enumeration.

During system initialization, the Root Complex performs enumeration to determine the various buses that exist and the devices that reside on each bus, as well as the required address space for the device's registers and memory. The Root Complex allocates bus numbers to all the PCIe Buses and configures the bus numbers to be used by the PCIe switches. A PCIe switch behaves as if it were multiple PCI-PCI Bridges (see inset in Figure 1). The Root Complex allocates and configures the Memory and I/O address space for each PCIe Switch and Endpoint device. A PCIe tree topology is shown in Figure 1.

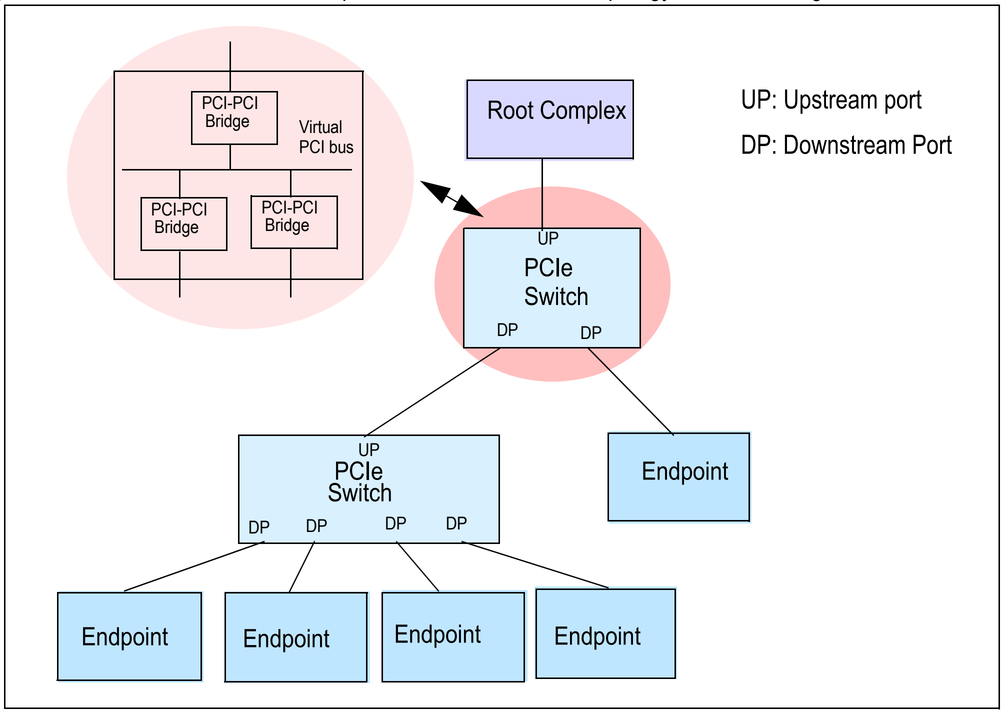

Figure 1 PCIe Tree Topology

Notes

In multi-Root Complex systems, more than one processor sub-system exists within a PCI Express tree. For example, a second Root Complex may be added to the system via the Downstream Port (DP) of a PCIe switch, possibly to act as a warm stand-by to the primary Root Complex. However, an issue arises when the second Root Complex also attempts the enumeration process. Assuming that the link trains (i.e. using crosslink training), it sends out Configuration Read Messages to discover other PCIe devices on the system. Configuration transactions can only move from Upstream to Downstream. A PCIe switch does not forward or respond to Configuration Messages that are received on its Downstream Port (DP); it ignores and silently drops all the configuration messages. Thus, the second Root Complex is isolated from the rest of the PCIe tree and will not detect any PCIe devices in the system. So, simply adding processors to a Downstream Port of a PCIe switch will not provide a multi-Root Complex solution.

One method of supporting multiple Root Complexes in a PCIe system is to use an NTB to isolate the address domains of each of the Root Complexes. An NTB allows two Root Complexes or PCIe trees to be interconnected with one or more shared address windows between them.

NTB in PES24NT3

A functional block diagram of the PES24NT3 device is shown in Figure 2. There are three ports in the device: the upstream port where the Root Complex is connected, the downstream port where a PCIe Endpoint or Switch may be connected, and the Non-Transparent port where a second Root Complex may be connected. The PES24NT3 may be logically viewed as consisting of three PCI-PCI transparent bridges, one per port, and an internal virtual PCI Bus. Beneath the transparent bridge associated with the non-transparent port are two endpoints interconnected by non-transparent bridge functionality. When viewed externally, the non-transparent port appears as an endpoint device. When viewed internally, the NTB beneath the PCI-PCI bridge appears as an endpoint device.

The endpoint and non-transparent bridge functionality closest to the PCI-PCI bridge is referred to as the internal endpoint (IE). The endpoint and non-transparent bridge associated with the external physical link is referred to as the external endpoint (EE). After configuration of the non-transparent bridge complex, the non-transparent bridge appears as a PCI-PCI bridge followed by an endpoint on the internal side and an endpoint on the external side.

Notes

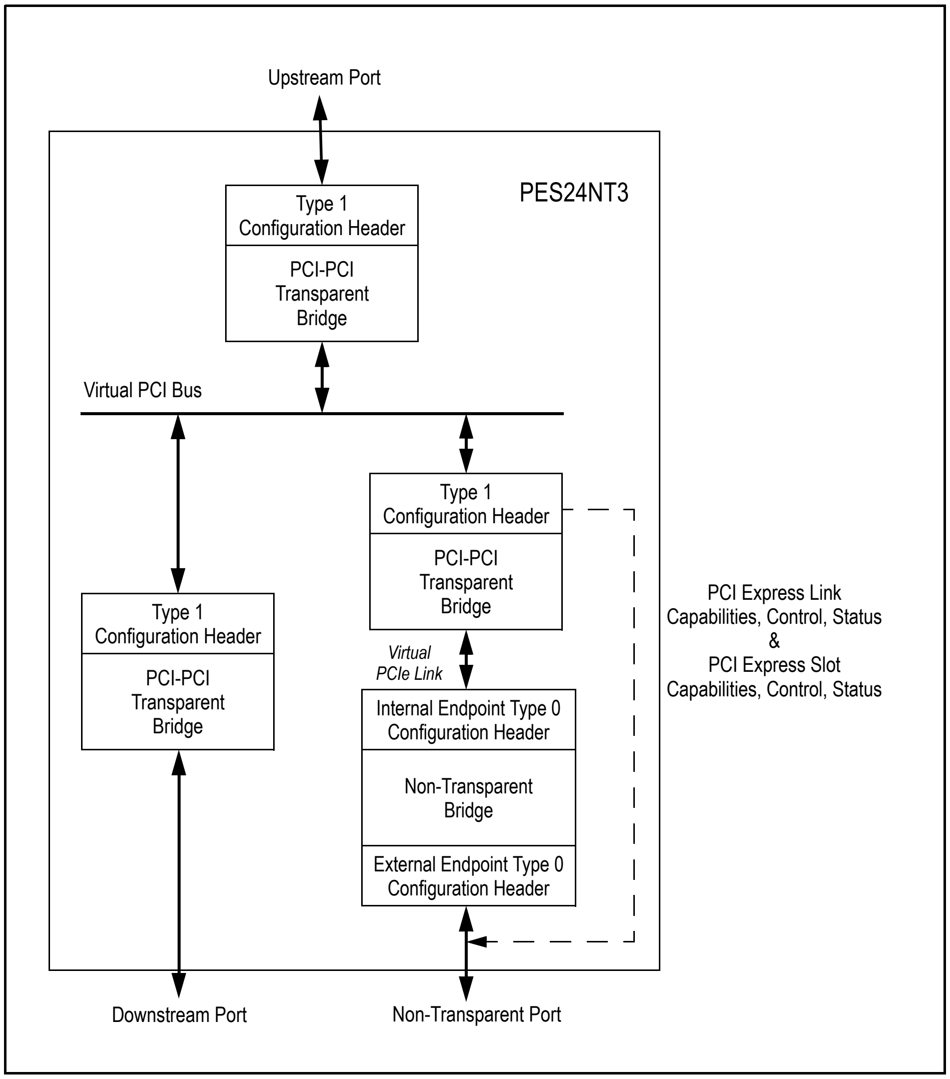

Figure 2 PES24NT3 Functional Block Diagram

System Architecture

This section provides two examples of a system topology using NTB to connect two root complexes. The usage model of these two topologies is to allow two root complexes to exchange data using NTB with a PCIe interface. There are other usage models of NTB, such as a redundant Root Complex, but these are not covered in this application note.

An example of a system that uses a single PES24NT3 device is shown in Figure 3. In this topology, the second root complex (Root Complex 2) is connected to the NTB port of the PES24NT3 device. Two address domains are created: Root Complex 1 address domain and Root Complex 2 address domain. Root Complex 1 and 2 can exchange data via the NTB port. In fact, any endpoint in the Root Complex 1 address domain may exchange data with Root Complex 2 and vice versa.

Notes

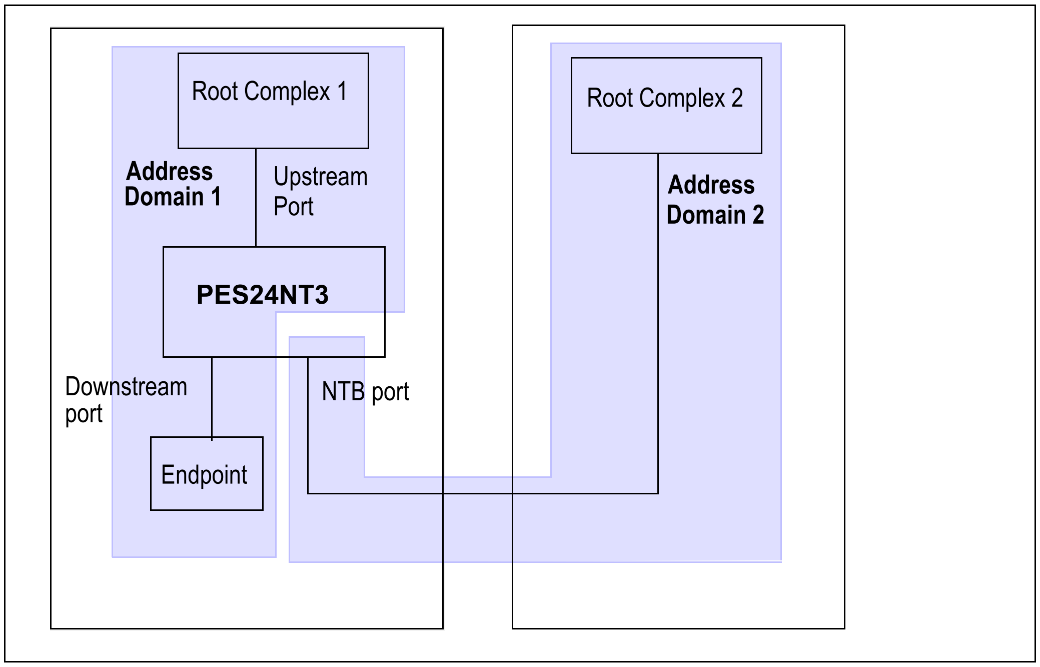

Figure 3 System Topology with a Single PES24NT3 Device

An example of a system that uses two PES24NT3 devices is shown in Figure 4. In this topology, the Root Complex 1 sub-system is identical to the Root Complex 2 sub-system. The two Root Complexes are connected to each other using two NTB ports. Three address domains are created: Root Complex 1 address domain, Root Complex 2 address domain, and the NTB port address domain. This system works very similar to the system shown in Figure 3. Root Complexes 1 and 2 can exchange data via the NTB ports and any endpoint in the Root Complex 1 address domain may exchange data with Root Complex 2 and vice versa.

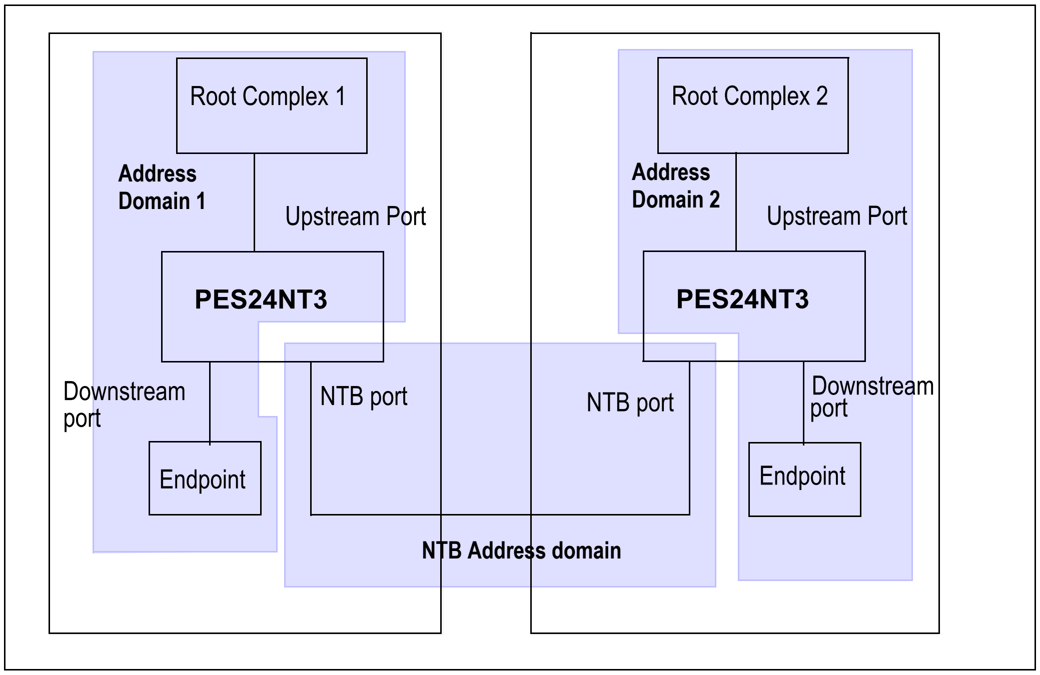

Figure 4 System Topology with Dual PES24NT3 Devices

Notes

Transaction Routing

PCIe defines three transaction routing mechanisms:

- Address routing with 32-bit or 64-bit format
- ID-based routing using bus, device, and function numbers
- Implicit routing using messages

There are four transaction types defined by the PCIe standard: Memory Read/Write, I/O Read/Write, Configuration Read/Write, and Message. Memory Read, Memory Write, I/O Read, I/O Write transactions are address-routed and are forwarded by the PES24NT3 device across the NTB. Completions are ID-routed and are forwarded by the PES24NT3 device across the NTB.

Configuration transactions and messages are not forwarded across the NTB.If a message is received by an internal or external endpoint, then it is silently discarded. If a Configuration transaction is received by an internal or external endpoint, it is assumed that the configuration transaction is destined for the endpoint which receives it.

Address Translation

The Internal and External Endpoints contain five base address registers (BARs) in their Type 0 header. BAR0 through BAR3 may each be used to map 32-bit memory or I/O windows between the internal and external sides of the non-transparent bridge. Odd and even numbered BARs may be paired to form 64-bit address windows.

The internal and external endpoints may each be configured to support four 32-bit address windows, two 64-bit windows, or two 32-bit windows and one 64-bit window. Each BAR has a corresponding setup register and translated base register in the internal and external endpoint Configuration Capability structure. The setup register contains fields that configure the corresponding BAR, such as the type of BAR (Memory or I/O) and the address window size. The translated base register contains the base address of the transactions forwarded through the non-transparent bridge using the corresponding BAR. The base address of a BAR corresponds to those address bits which are examined to determine if an address falls into a region mapped to a BAR.

The address translation logic is shown in Figure 5. This example shows how the address is translated when a packet is forwarded from the internal endpoint to the external endpoint. The address translation works exactly the same when a packet is forwarded from the external endpoint to the internal endpoint. When a packet is received by the internal endpoint, the address field is extracted from the PCIe transaction layer packet. The address and type are compared against BAR0 through BAR3. If the address falls within the window size of one of the BARs, the base address of the original address is replaced with the content of the corresponding Translated Base Address Register before the packet is forwarded. If the address does not find a match in BAR0 to BAR3, the packet is dropped.

Notes

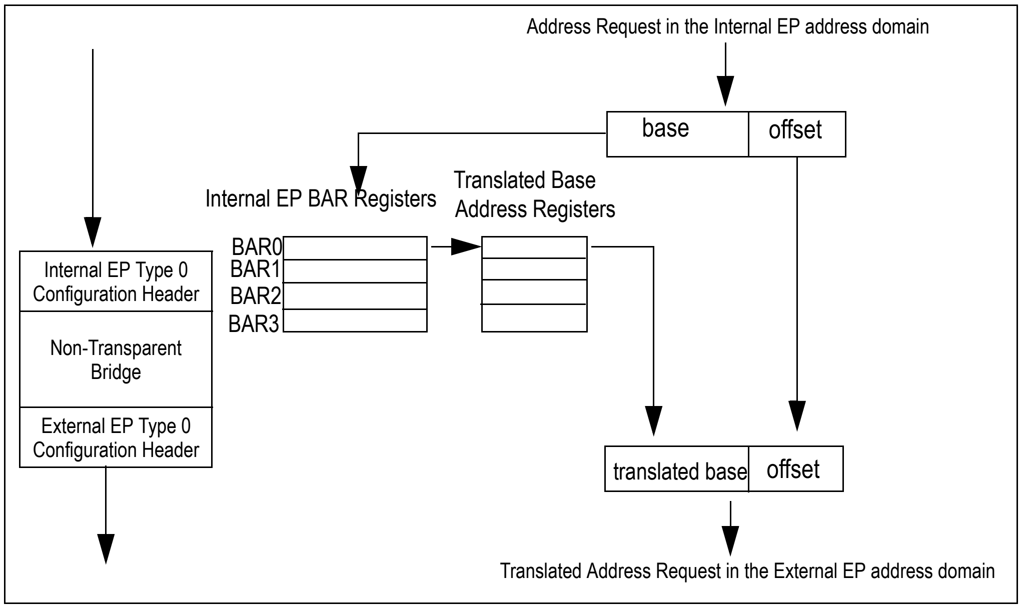
Address Request in the Internal EP address domain
base
offset
Internal EP BAR Registers
Translated Base Address Registers
BAR0
BAR1
BAR2
BAR3
Internal EP Type 0 Configuration Header
Non-Transparent Bridge
External EP Type 0 Configuration Header
translated base
offset
Translated Address Request in the External EP address domain

Figure 5 Address Translation Logic

The address translation works similarly when two non-transparent ports are connected back to back as shown in Figure 6. In this example, the non-transparent bridge ports of two PES24NT3 devices are connected. When a packet is received on the internal endpoint, the address and type are matched against the Internal endpoint BAR0 to BAR3. When the address falls within the address window of a BAR, the base address is replaced with the content of the corresponding Translated Base Address Register before the packet is forwarded to the external endpoint.

When the external endpoint of the other PES24NT3 receives the packet with the translated address, the address and type are matched against the external endpoint BAR0 to BAR3. When the address falls within the address window of a BAR, the base address is replaced with the content of the corresponding Translated Base Address Register before the packet is forwarded to the internal endpoint.

Notes

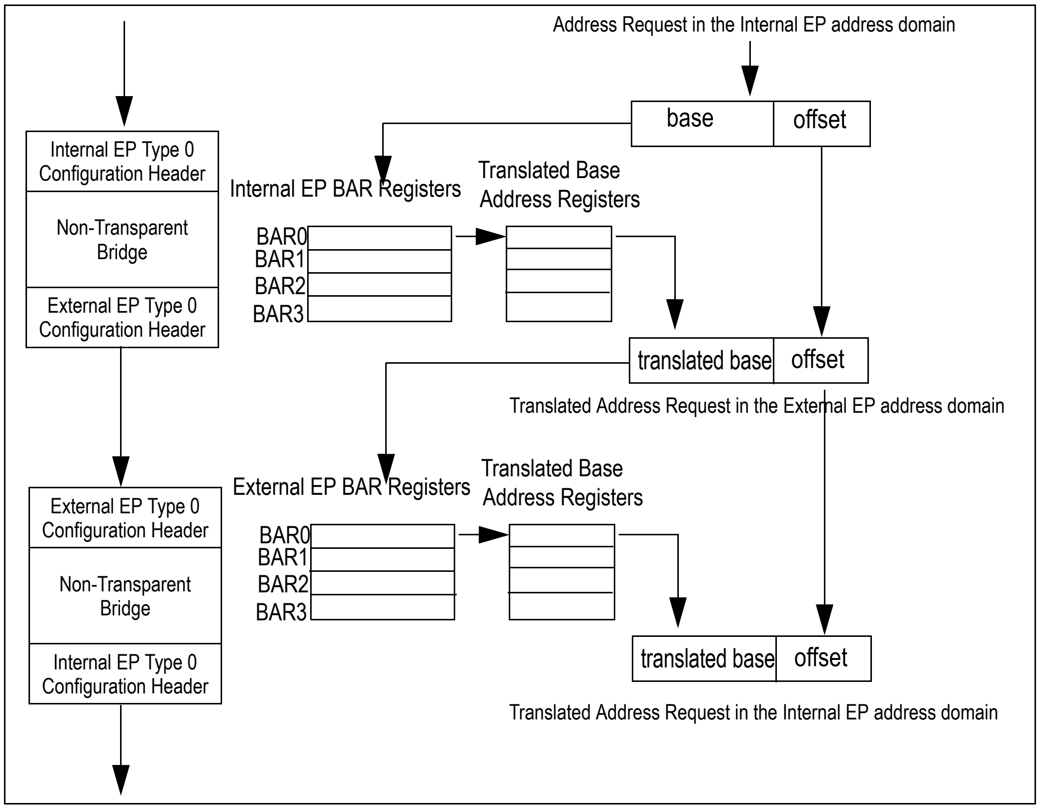
Address Request in the Internal EP address domain
base
offset
Internal EP Type 0 Configuration Header
Non-Transparent Bridge
External EP Type 0 Configuration Header
Internal EP BAR Registers
Translated Base Address Registers
BAR0
BAR1
BAR2
BAR3
translated base
offset
Translated Address Request in the External EP address domain
External EP Type 0 Configuration Header
Non-Transparent Bridge
Internal EP Type 0 Configuration Header
External EP BAR Registers
Translated Base Address Registers
BAR0
BAR1
BAR2
BAR3
translated base
offset
Translated Address Request in the Internal EP address domain

Figure 6 Address Translation Tables for back to back NTB ports

ID Routing and Translation

Associated with the internal and external endpoints are 16-entry mapping tables. The mapping table contains the bus, device, and function (BDF) numbers of the initiators on that side of the endpoint whose transactions may be forwarded to the opposite side of the endpoint, i.e., from the internal endpoint to the external endpoint or from the external endpoint to the internal endpoint. The format of the mapping table is shown in Figure 7. The B, D, F fields contain the bus, device, and function numbers of the initiator and the V field indicates if the entry is Valid or not. The mapping table is filled in during system initialization. If all the transactions are initiated by the Root Complex, only one entry needs to be entered into the mapping table. The entry contains the B,D,F value of the Root Complex. The BDF of the Root Complex is system-dependent.

	31 30 29 28 27 26 25 24	23 22 21 20 19	18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1	0
Mapping Table Entry Address 0	B	D	F	Reserved	V
Mapping Table Entry Address 1	B	D	F	Reserved	V
Mapping Table Entry Address 2	B	D	F	Reserved	V
Mapping Table Entry Address 3	B	D	F	Reserved	V
	⋮	⋮	⋮	⋮	
Mapping Table Entry Address 14	B	D	F	Reserved	V
Mapping Table Entry Address 15	B	D	F	Reserved	V

Figure 7 Endpoint Mapping Table Format

Notes

The requester ID of a posted or non-posted address-routed request packet has to be translated before the packet can be forwarded from one endpoint to the other endpoint. Both posted and non-posted address-routed transactions are handled in exactly the same manner. There is a completion associated with a non-posted address-routed request that needs to crossover the non-transparent bridge from the side opposite to that of the original initiator. Completion transactions are routed by ID, i.e., bus, device, and function numbers (BDF).

When an address routed request packet (posted or non-posted) is received, the bus, device and function numbers are used to look up the mapping table. If there is a match, then the packet is forwarded to the other side of the non-transparent bridge and the requester ID is translated. The bus number in the requester ID in the forwarded packet is equal to the bus number of the non-transparent bridge endpoint on the opposite side of the non-transparent bridge. The device and function number of the requester ID is equal to the matching mapping entry table address. If there is no match in the mapping table, the packet is treated as an Unsupported Request and the error is reported in the non-transparent bridge status register.

When a completion packet is received by a non-transparent bridge endpoint, it is routed by the ID back to the opposite side of the non-transparent bridge. A mapping table entry address is formed using the device and function number of the requester ID field in the completion packet. The mapping table entry address is used to access the mapping table corresponding to the opposite side of the non-transparent bridge from which the completion was received. If the mapping table entry is valid, the completion is routed to the other side of the non-transparent bridge. The bus, device, and function numbers of the requester ID of the forwarded packet are replaced with the corresponding values in the associated mapping table entry. The completer ID of the forwarded packet is replaced with the bus, device, and function numbers of the non-transparent bridge endpoint on which the completion is transmitted. If the mapping table entry is invalid (V bit is 0) or if the mapping table entry address does not point to a valid entry (i.e. bigger than 15), the packet is dropped and an error is reported in the non-transparent bridge status register.

An example of an ID translation in shown in Figure 8. In this example, the Root Complex of the internal endpoint makes a non-posted address-routed request to the Root Complex of the external endpoint. The bus, device, and function (BDF) numbers of the internal EP is 5,0,0. The BDF of the external EP is 6,0,0. During system initialization, the internal EP mapping table is filled with a single entry at an address of 0 with a BDF value of 1,1,0 (Root Complex). It should be noted that the BDF numbers of the Root Complex, internal and external EPs are examples only. The actual BDF numbers are assigned by the operation system dynamically at run time and the assignment is system-dependent. Actual BDF numbers must be used in the implementation.

When a request packet is received by the internal EP, the requester ID (BDF = 1,1,0 for the Root Complex) is used to look up the Internal EP Mapping Table. A match is found and the matching entry has an address of 0. The requester ID is then replaced with the bus number of the External EP (6) and the device and function numbers are replaced with the matching Internal EP Mapping Table Address (0). The forwarded packet has a requester ID of 6,0,0.

A corresponding completion packet is received by the External EP some time later. The requester ID and completer ID of the completion packet are 6,0,0 (Requester ID of the corresponding request packet) and 2,2,0 (Root Complex of the External EP) respectively. The device and function numbers of the Requester ID (0) are used as an address to the Internal EP Mapping Table. The BDF of the Mapping Table entry at address 0 (1, 1, 0) and the BDF of the Internal EP (5,0,0) are used to replace the Requester and Completer ID in the completion packet respectively.

Notes

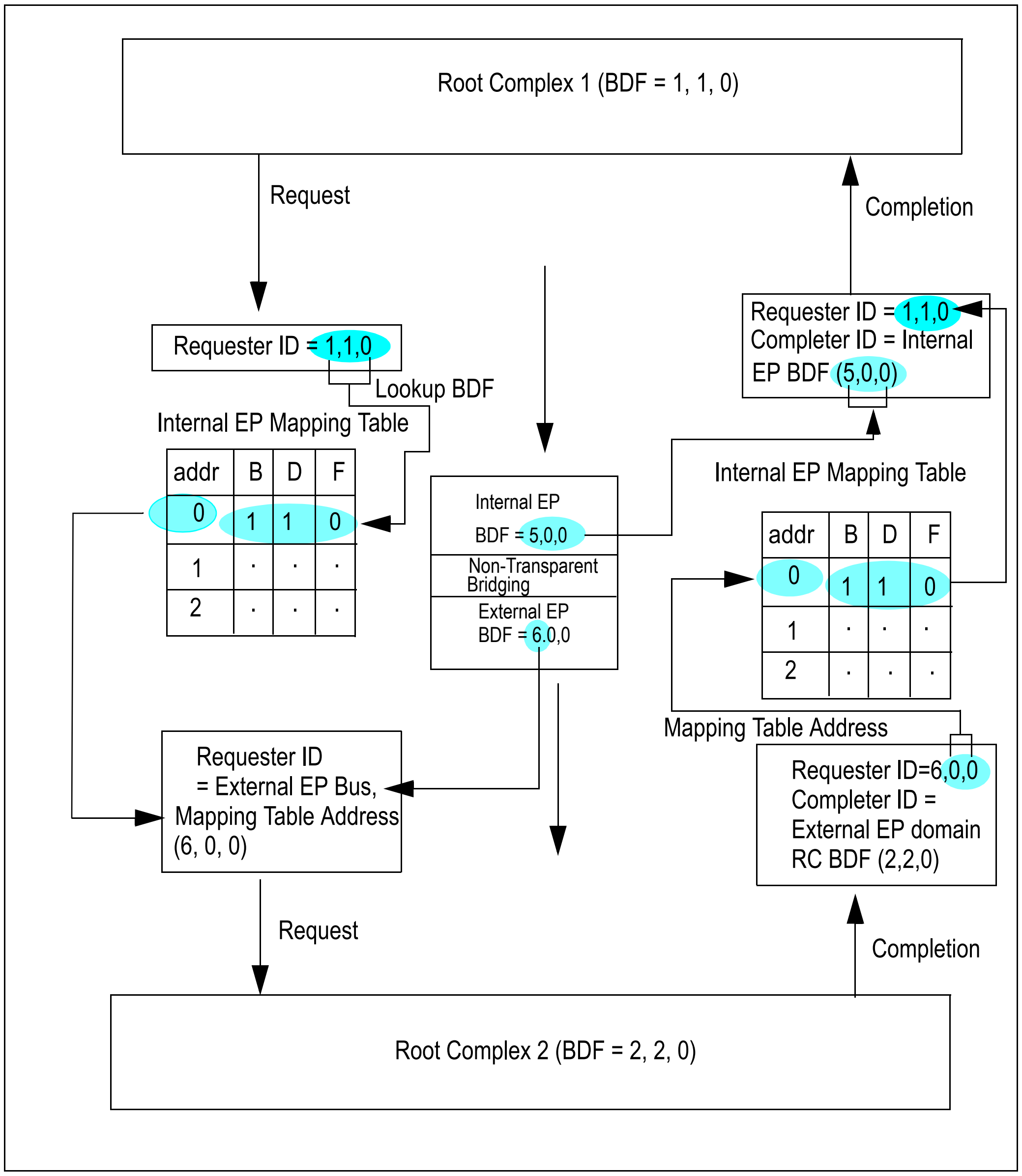
Root Complex 1 (BDF = 1, 1, 0)
Request
Completion
Requester ID = 1,1,0
Lookup BDF
Internal EP Mapping Table
addr
B
D
F
0
1
1
0
1
2
Internal EP
BDF = 5,0,0
Non-Transparent Bridging
External EP
BDF = 6,0,0
Requester ID = 1,1,0
Completer ID = Internal EP BDF (5,0,0)
Internal EP Mapping Table
Mapping Table Address
Requester ID = External EP Bus, Mapping Table Address (6, 0, 0)
Requester ID=6,0,0
Completer ID = External EP domain RC BDF (2,2,0)
Request
Completion
Root Complex 2 (BDF = 2, 2, 0)

Figure 8 ID Translation with NTB

The ID translation works similarly when two non-transparent ports are connected back-to-back as shown in Figure 9. In this example, the non-transparent bridge ports of two PES24NT3 devices are connected. The Root Complex of the Internal EP1, shown at the top of Figure 9, makes a non-posted address-routed request to the Root Complex of the other Internal EP2, shown at the bottom of the figure. A completion response is returned to the requester going through the two PES24NT3 devices some time later.

During system initialization, the Internal and External EP1 are assigned the BDF values of 5,0,0 and 6,0,0 respectively. The Internal and External EP2 are assigned the BDF values of 4,0,0 and 6,0,0 respectively. There is only a single entry in each of the mapping tables, assuming all the requests are from Root Complex to Root Complex. The entry in the Internal EP1 Mapping Table is the BDF values of the Root Complex which is 1,1,0. The entry in the External EP2 Mapping Table is the BDF value of the External EP1 which is 6,0,0.

Notes

When a request packet is received by the Internal EP1, the requester ID (BDF = 1,1,0 for the Root Complex) is used to look up the Internal EP1 Mapping Table. A match is found and the matching entry has an address of 0. The requester ID is then replaced with the bus number of the External EP1 (6) and the device and function numbers are replaced with the matching Internal EP1 Mapping Table Address (0). The forwarded packet has a requester ID of 6,0,0.

When the request packet is received by the External EP2, the requester ID of 6,0,0 is used to look up the External EP2 Mapping Table. A match is found and the matching entry has an address of 0. The requester ID is then replaced with the bus number of the Internal EP2 (4) and the device and function numbers are replaced with the matching External EP2 Mapping Table Address (0). The forwarded packet has the requester ID of 4,0,0.

A corresponding completion packet is received by the Internal EP2 some time later. The requester ID and completer ID of the completion packet are 4,0,0 (Requester ID of the corresponding request packet) and 2,2,0 (Root Complex) respectively. The device and function numbers of the Requester ID (0) is used as an address to access the External EP2 Mapping Table. The BDF of the Mapping Table entry at address 0 (6, 0, 0) and the BDF of the External EP2 (6,0,0) are used to replace the Requester and Completer ID respectively in the completion packet.

When the completion packet is received by the External EP1, the device and function numbers of the Requester ID (0) are used as an address to access the Internal EP1 Mapping Table. The BDF of the Mapping Table entry at address 0 (1, 1, 0) and the BDF of the Internal EP1 (5,0,0) are used to replace the Requester and Completer ID respectively in the completion packet.

Notes

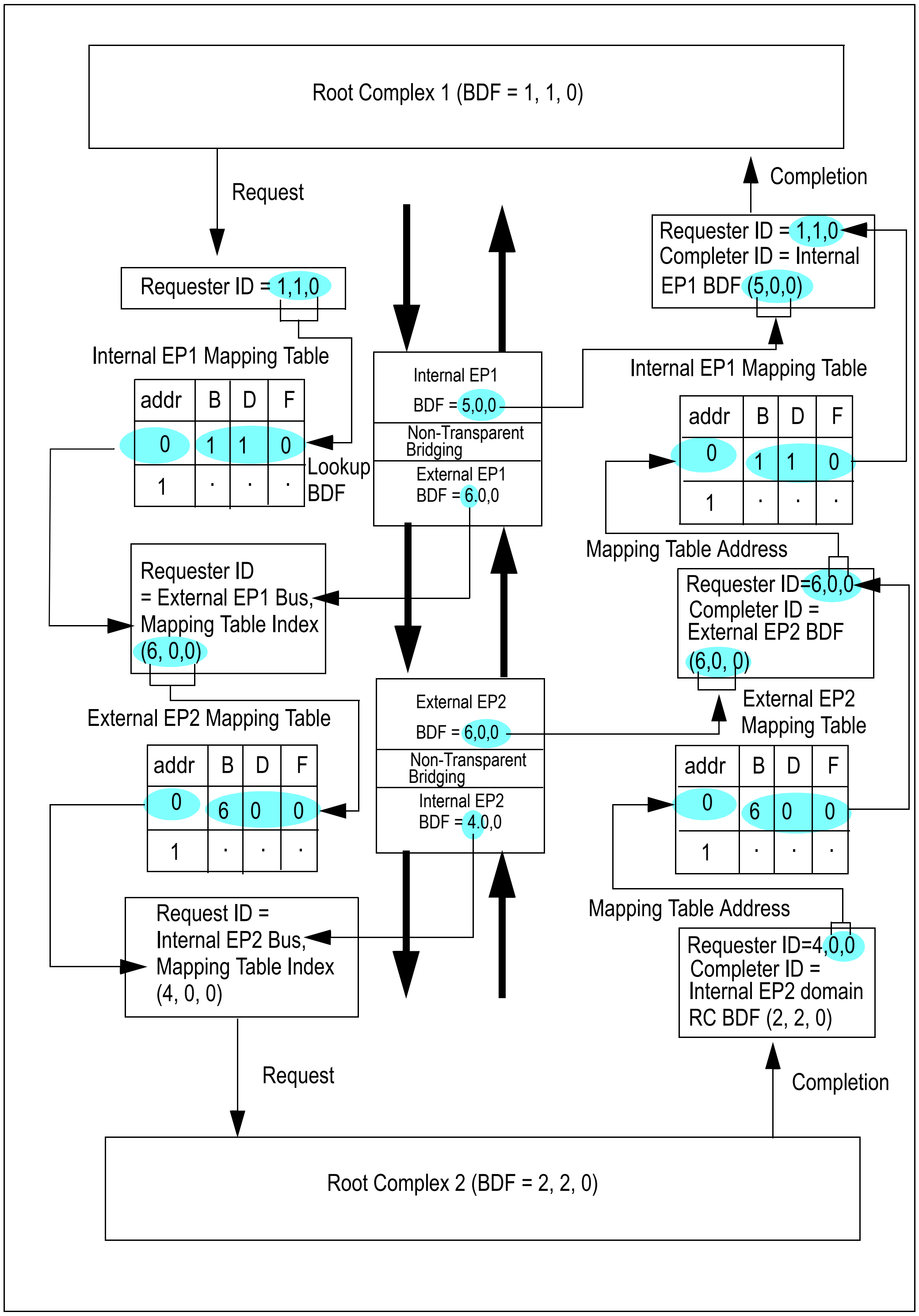

Figure 9 IDT Translation for Back to Back NTB Ports

Notes

Interprocessor Communications

The Non-transparent Bridge Communication Capability has a number of components to aid in communication between processors on opposite sides of the NTB. These are graphically illustrated in Figure 10.

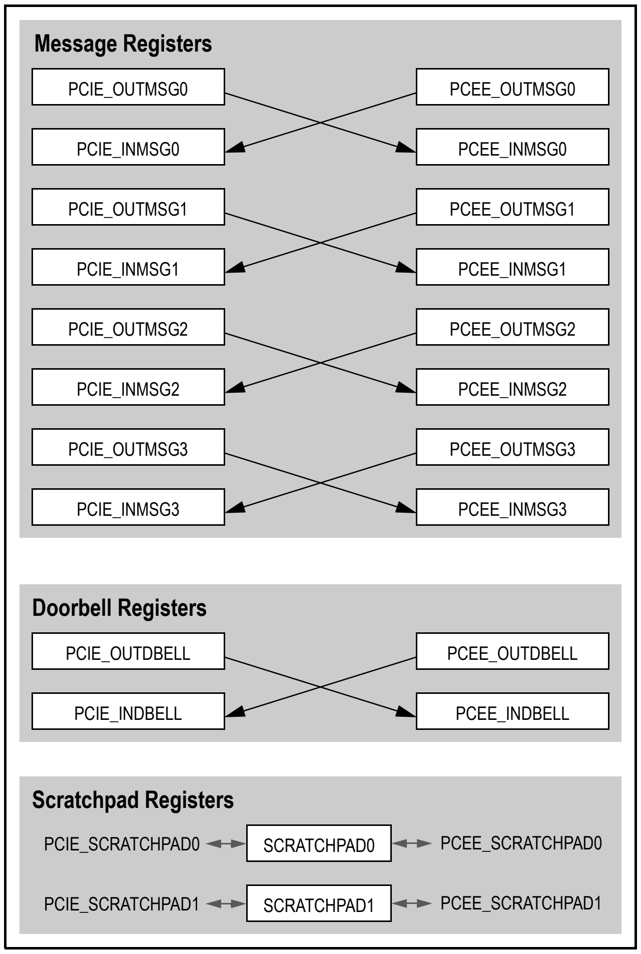

Figure 10 Interprocessor Communication Facilities

Message Registers

The internal and external endpoints each have four Inbound Message (INMSG[0..3]) and Outbound Message (OUTMSG[0..3]) registers. OUTMSG registers can be read or written while INMSG registers are read-only. When an OUTMSG register is written, the corresponding INMSG register on the opposite side of the NTB takes on that written value, and the corresponding Inbound Message (INMSGx) bit is set in the Interrupt Status (INTSTS) register on the opposite side of the bridge.

This mechanism may be used to pass 32-bit quantities with interrupt notification. For example, when a processor on the internal side writes to PCIE_OUTMSG0, then PCEE_INMSG0 takes on the value written and the INMSG0 bit is set in the PCEE_INTSTS register. The setting of the INMSG0 bit may be used to generate an interrupt to the root complex on the external side.

Doorbell Registers

The internal and external endpoints each have an Inbound Doorbell (INDBELL) and Outbound Doorbell (OUTDBELL) register. The OUTDBELL register may be read and written while INDBELL register is read and cleared. The doorbell registers provide 32 doorbells in each direction through the non-transparent bridge. When a bit is set in the OUTDBELL register, the corresponding bit is set in the INDBELL register on the opposite side of the NTB. When any bit in the INDBELL register is set, the INDBELL bit is set in the INTSTS register on that side of the non-transparent bridge and may be used to generate an interrupt.

Notes

Scratchpad Registers

There are two scratchpad (SCRATCHPAD[0..1]) registers shared between the internal and external sides of the non-transparent bridge. Writing to a scratchpad register immediately modifies its value on both sides of the non-transparent bridge. There are no interrupts or other notification associated with scratchpad register modifications.

Configuration

All normal PCIe endpoint configurations must be performed before the NTB forwards a transaction. For example, the BAR registers and the Mapping Tables of the internal and external endpoints must be configured before memory transactions are routed through the non-transparent bridge.

Configuration Space

Associated with the internal and external endpoints is a 4KB PCIe configuration space containing a Type 0 header. Refer to the IDT PES24NT3 user manual for details on the organization of these configuration spaces. The configuration spaces are symmetric, meaning that the same fields are located in the same location on both the internal and external endpoints (EPs).

The internal root complex configures internal endpoint configuration space registers by performing configuration read and write operations. The external root complex configures external endpoint configuration space registers by performing configuration read and write operations.

Registers associated with the internal or external endpoints occupy the bottom 2KB of their configuration space. The upper 2KB of each endpoint's configuration space contains the configuration space of the endpoint associated with the opposite side of the NTB. Thus, an internal root may configure any external endpoint register simply by adding a 2KB offset and referencing the internal endpoint's configuration space. The external root has a similar capability. For system security reasons, access to the configuration by endpoints on the opposite side of the NTB using configuration transactions may be disabled.

The cross-coupling of NTB endpoint configuration spaces is graphically illustrated in Figure 11. BAR4 of each NTB endpoint permits the entire 4KB configuration space to be memory mapped into PCIe space, allowing any master to access configuration registers. The organization of this 4KB memory is the same as that for the corresponding configuration space.

Notes

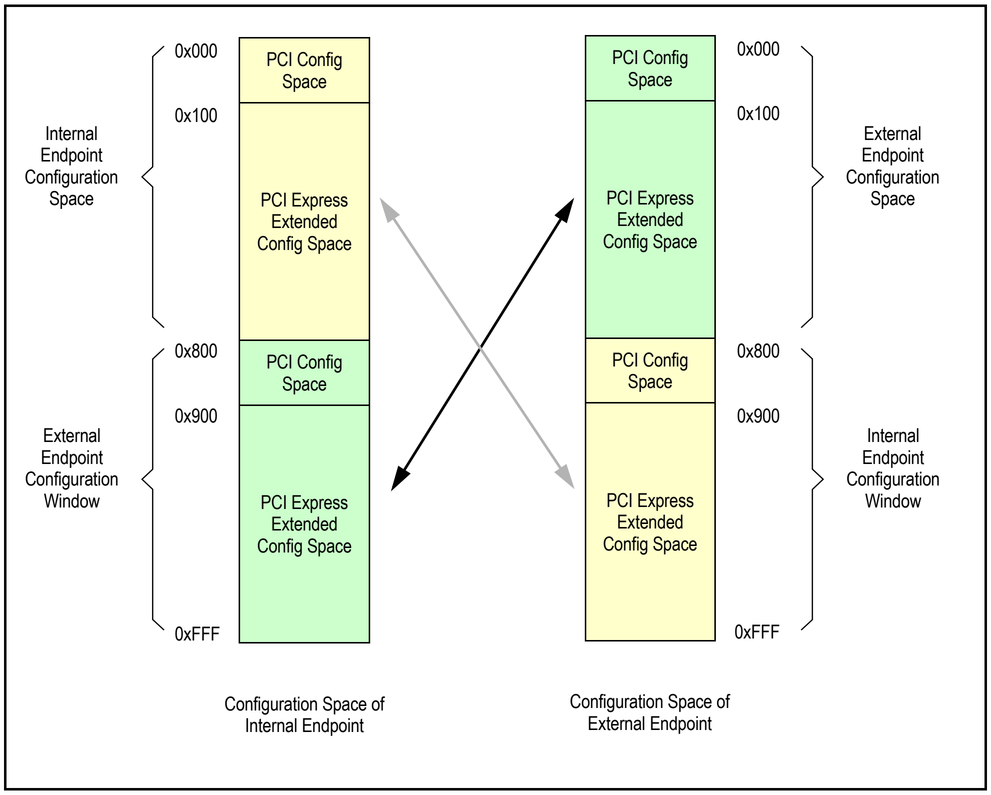
Internal Endpoint Configuration Space
External Endpoint Configuration Window
0x000
0x100
0x800
0x900
0xFFF
PCI Config Space
PCI Express Extended Config Space
PCI Config Space
PCI Express Extended Config Space
Configuration Space of Internal Endpoint
PCI Config Space
PCI Express Extended Config Space
PCI Config Space
PCI Express Extended Config Space
0x000
0x100
0x800
0x900
0xFFF
External Endpoint Configuration Space
Internal Endpoint Configuration Window
Configuration Space of External Endpoint

Figure 11 Non-Transparent Bridge Configuration Window

Opposite Side Configuration Requests

The internal endpoint has the capability to generate configuration transactions on the external side of the NTB. This mechanism, referred to as punch through, is provided to facilitate systems in which a root complex does not exist on the external side of the NTB. An example of this would be two NTB ports connected back-to-back.

Error Detection and Handling

End-to-End CRC (ECRC) is supported for transactions that are forwarded through the NTB. A new ECRC must be computed since the packet is modified. When a packet is forwarded through the NTB with the ECRC included in the packet, ECRC is checked at the same time a new ECRC is computed for the modified packet. If there is no ECRC error in the original packet, the ECRC in the original packet is replaced with the ECRC that was computed in the modified packet.

If an ECRC error is detected, the new ECRC computed in the modified packet is first inverted and then replaces the original ECRC in the new packet. The error is logged in the non-transparent bridge status registers associated with the endpoint on which the packet was received. An ERR_NONFATAL error signaling message is generated to the root complex on which the packet was received if non-fatal error reporting is enabled.

Physical and data link layers are associated with external endpoints. Error messages generated as a result of detected errors are sent to the external root complex (i.e. the root complex on the external side of the external endpoint) and not to the internal root complex (i.e. the root complex on the internal side of the internal endpoint)

The transaction layer-related errors are handled as follows:

- Internal EP related errors are reported to the internal root complex
- External EP related errors are reported to the external root complex

Notes

Initialization

For memory transactions to be routed between internal and external endpoints, the following configuration must be performed on both the internal and external endpoints:

- The Memory Access Enable (MAE) bit must be set to enable an endpoint to forward memory transactions.
- The Bus Master Enable (BME) bit must be set to enable an endpoint to generate transactions.
- BARSETUP[0-3] registers must be configured to enable the BAR, set the address space size, select non-prefetchable, use either 32-bit or 64-bit addressing, and to set the address to be memory space.
- BARTBASE[0-3] registers must be set to configure the translated base address.
- BAR[0-3] registers must be set to configure the address windows.
- Endpoint Mapping Table must be initialized to specify the ID translation.

Configuration registers may be loaded with just EEPROM during power-on reset or a combination of EEPROM loading and software programming. If the target system is a closed system and the memory map is fixed, then EEPROM load during power on reset is the simplest method to initialize the PES24NT3 device. However, certain operating systems, such as Linux, allocate PCI address space and local memory dynamically. There is also a mapping between virtual and physical memory. The memory map is not fixed without a major change to the operating system. To initialize the PES24NT3, a combination of EEPROM loading and software programming may be a better solution.

At a minimum, the BARSETUP[0-3] registers must be loaded via EEPROM during power-on reset and the rest of the registers may be programmed via software for the initialization. The actual number of registers that may be loaded via EEPROM is system-dependent.

Example

Two examples are described here to show how a PES24NT3 device may be initialized using a combination of EEPROM load and software programming. In both examples, the configuration is assumed to be:

- 32-bit address is used.
- Single address window of 1MByte
- All requests across the NTB are memory requests and are made between the root complexes
- There is a system-dependent out-of-band communication channel between the two root complexes for a root complex to check if the other root complex is operating.

The message and door bell registers are used for communicating between the two Root Complexes. The four message registers can be grouped together as a single message unit with a maximum length of 16 bytes (4 x 32-bits) and sent as an interprocessor communication (IPC) Protocol Data Unit (PDU) from one root complex to the other root complex. The message interrupt is unused and disabled. The Door Bell register is used to notify the other root complex that a PDU is ready to be processed. When the IPC PDU has been read from the message registers, the Door Bell register is again used to acknowledge that the IPC PDU has been read and a new IPC PDU may be written again.

The procedure to send an IPC PDU from RC1 to RC2 is as follows:

1. RC1 writes the IPC PDU to the message register 0 - 3.
2. RC1 writes a value of 1 (bit 0) to the Door Bell register to interrupt RC2.
3. RC2 reads the Door Bell register. Bit 0 is set, indicating that an IPC PDU is ready to be read.
4. RC2 reads the IPC PDU from the message register 0 - 3 and queues it for later processing.
5. RC2 writes a value of 2 (bit 1) to the Door Bell register to interrupt RC1.
6. RC1 reads the Door Bell register. Bit 1 is set, indicating that the IPC PDU has been read from the message register. A new IPC PDU may be written if there is any messages pending.
7. RC2 may send an IPC PDU back to RC1 in response to the IPC PDU that it received from RC1 earlier.

Notes

Single PES24NT3 Device

The topology of the first example, which has only a single PES24NT3, is shown in Figure 12. RC1 is connected to the Upstream port of a PES24NT3 device and hence is the Root Complex of the internal endpoint (IE). RC2 is connected to the NTB port of the PES24NT3 device and hence is the Root Complex of the external endpoint (EE). There are two address domains in this topology. The RC1 address domain and the RC2 address domain.

BARSETUP0 register of the IE and EE are initialized during power-on reset using EEPROM load and hence only BAR0 is used. It is set to have an address space size of 1MByte, is non-prefetchable, uses 32-bit addressing, and is a memory space.

During system initialization, the "normal"[1] PCI initialization and enumeration procedure is followed. All the PCI-PCI bridges are initialized with the proper configuration for both Address and ID routing. When RC1 detects the internal endpoint (IE), it checks BAR registers 0-4. Detecting that BAR0 is asking for 1MByte of memory, RC1 allocates 1 Mbyte of PCI address and writes the base address to BAR0. Based on the vendor ID and device ID of the internal endpoint, RC1 loads and passes control to the internal endpoint device driver. The internal endpoint device driver then sets the Memory Access Enable bit and Bus Master Enable bit on the internal endpoint. An entry is also added to the internal endpoint mapping table. The entry has a bus, device, and function numbers of 0,0,0 (BDF of RC1) and it is at index 0. The Internal Endpoint device driver waits for RC2 to be ready. At this point, no transaction is forwarded between the internal and external endpoints.

RC2 also follows the "normal"[1] PCI initialization and enumeration procedure after power-on. When RC2 detects the external endpoint (EE), it checks BAR registers 0-4. Detecting that BAR0 is asking for 1MByte of memory, RC2 allocates 1MByte of PCI address and writes the base address to BAR0. Based on the vendor ID and device ID of the external endpoint, RC2 loads and passes control to the external endpoint device driver. The external endpoint device driver then sets the Memory Access Enable bit and Bus Master Enable bit on the external endpoint. An entry is also added to the external endpoint mapping table. The entry has a bus, device, and function numbers of 0,0,0 (BDF of RC2) and it is at index 0. RC2 is now operating.

When RC1 detects that RC2 is operating using the system-dependent out-of-band communication channel, RC1 sends an IPC PDU to RC2 requesting RC2 to allocate 1MByte of memory to be used as the translated address base. When RC2 receives the memory allocation IPC PDU, it allocates 1MByte of memory from the system memory. The memory must be on a 1 Mbyte alignment. The operating system may not support this alignment requirement for memory allocation. RC2 may have to allocate 2MBytes of memory and adjust the base address to the nearest 1MByte boundary. RC2 then returns the base address of this 1 Mbyte of memory to RC1 using the message registers.

Upon receiving the base address from RC2, RC1 configures the BAR0 Translated Base Address of the Internal Endpoint with this base address. RC1 is ready to send an address-routed request to RC2.

RC2 detects that RC1 is operating using the system dependent out of band communication channel. RC2 sends an IPC PDU to RC1 requesting RC1 to allocate 1MByte of memory to be used as the translated address base. When RC1 receives the memory allocation IPC PDU, it allocates 1MByte of memory from system memory. The memory must be on a 1 Mbyte alignment. The operating system may not support this alignment requirement for memory allocation. RC1 may have to allocate 2MBytes of memory and adjust the base address to the nearest 1MByte boundary. RC1 then returns the base address of this 1 Mbyte of memory to RC2 using the message registers.

Upon receiving the base address from RC1, RC2 configures the BAR0 Translated Base Address of the External Endpoint with this base address. RC2 is ready to send an address-routed request to RC1.

At this point, the initialization process is completed. RC1 and RC2 are ready to send address-routed requests to each other across the NTB port.

[1]. Normal Linux or Windows PCI enumeration procedure.

Notes

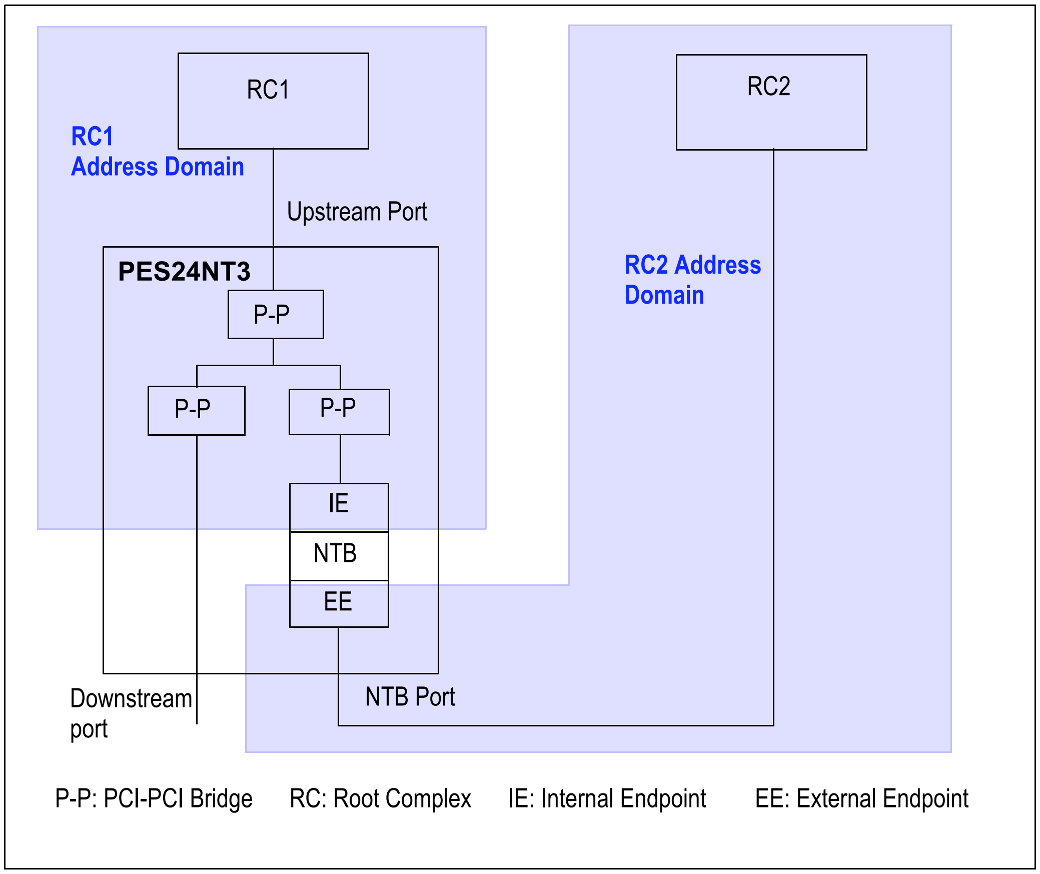

Figure 12 Single NTB Topology

Two PES24NT3 Devices

The topology of the second example, which has two PES24NT3 devices, is shown in Figure 13. The NTB ports are connected to each other. The PES24NT3 device supports the crosslink feature and hence the two NTB ports will link train. This creates three address domains: the RC1 address domain which includes IE1, the RC2 address domain which includes IE2, and the NTB address domain which is between (and includes) EE1 and EE2. There is no root complex on the NTB address domain. All error messages and interrupts that generated by EE1 are dropped by EE2 and vice versa. The internal endpoint of a PES24NT3 device supports the punch through feature to generate configuration transactions on the external side of the NTB port. This allows RC1 to send configuration transactions to initialize and configure EE2. EE1 may also be configured by RC2 using this method.

BARSETUP0 registers of the IE and EE are initialized during power-on reset using EEPROM load and hence only BAR0 is used. It is set to have an address space size of 1MByte, is non-prefetchable, uses 32-bit addressing, and is a memory space.

During system initialization, the "normal"[2] PCI initialization and enumeration procedure is followed. All the PCI-PCI bridges are initialized with the proper configuration for both Address and ID routing. When RC1 detects IE1, it checks BAR registers 0-4. Detecting that BAR0 is asking for 1MByte of memory, RC1 allocates 1 Mbyte of PCI address and writes the base address to BAR0. Based on the vendor ID and device ID of IE1, RC1 loads and passes control to the internal endpoint device driver.

To finish the initialization process, the internal endpoint device driver sets the following registers:

1. Set the Memory Access Enable and Bus Master Enable bits on IE1.
2. Add an entry to the IE1 mapping table. The entry has bus, device, and function numbers of 0,0,0 (BDF of RC1) and it is at index 0.

2. Normal Linux or Windows PCI enumeration procedure.

Notes

3. Set the IE1 BAR0 Translated Base Address to a value chosen by the system designer. Select 0 for this example. This value must match the content of BAR0 of EE2.
4. Set BAR0 of EE1 to a value chosen by the system designer. This value must match the value programmed to the IE2 BAR0 Translated Base Address. Select 0 for this example.
5. Set the Memory Access Enable and Bus Master Enable bits on EE1.
6. Add an entry to the EE1 mapping table. The entry has bus, device, and function (BDF) numbers of EE2 and it is at index 0. Select a BDF value of 10,0,0.
7. Wait for RC2 to be ready.

RC2 also follows the "normal"[3] PCI initialization and enumeration procedure after power on. The initialization procedures is exactly the same as RC1. When RC2 detects IE2, it checks the BAR register 0-4. On seeing that BAR0 asking for 1MByte of memory, RC2 allocates 1 Mbyte of PCI address and writes the base address to BAR0. Based on the vendor ID and device ID of IE2, RC2 loads and passes control to the internal endpoint device driver.

To finish the initialization process, the internal endpoint device driver sets the following registers:

1. Set the Memory Access Enable and Bus Master Enable bits on IE2.
2. Add an entry to the IE2 mapping table. The entry has bus, device, and function numbers of 0,0,0 (BDF of RC1) and it is at index 0.
3. Set the IE2 BAR0 Translated Base Address to a value chosen by the system designer. This value must match the content of BAR0 of EE1 which is 0.
4. Set BAR0 of EE2 to a value chosen by the system designer (see step 3 above for the setting of IE1 BAR0). This value must match the value programmed to the IE1 BAR0 Translated Base Address (which in this example is 0).
5. Set the Memory Access Enable and Bus Master Enable bits on EE2.
6. Add an entry to the EE2 mapping table. The entry has bus, device, and function (BDF) numbers of EE1 and it is at index 0. Select a BDF value of 10,0,0 which is the same as EE1.
7. Send a "Ready" message to RC1.

RC2 has to write to the Message and Doorbell Registers of EE1 in order to send IPC PDUs. RC2 use the punch through feature of the NTB to send configuration transactions to EE1. RC2 uses the value of 10,0,0 as the BDF of EE1.

When RC1 detects that RC2 is operating using the system-dependent out-of-band communication channel, RC1 sends an IPC PDU to RC2 requesting RC2 to allocate 1MByte of memory to be used as the translated address base. When RC2 receives the memory allocation IPC PDU, it allocates 1MByte of memory from system memory. The memory must be on a 1 Mbyte alignment. The operating system may not support this alignment requirement for memory allocation. RC2 may have to allocate 2MBytes of memory and adjust the base address to the nearest 1MByte boundary. RC2 then writes the newly allocated memory address to the EE2 BAR0 Translated Base Address. RC2 then returns the status of "Done" to RC1 using the message registers of EE1. RC1 is ready to send address-routed requests to RC2.

When RC2 detects that RC1 is operating using the system-dependent out-of-band communication channel, RC2 sends an IPC PDU to RC1 requesting RC1 to allocate 1MByte of memory to be used as the translated address base. When RC1 receives the memory allocation IPC PDU, it allocates 1MByte of memory from system memory. The memory must be on a 1 Mbyte alignment. RC1 then configures the EE1 BAR0 Translated Base Address with this base address. RC1 then returns the status of "Done" to RC2 using the message registers of EE2. RC2 is ready to send address-routed requests to RC1.

At this point, the initialization process is completed. RC1 and RC2 are ready to send address-routed requests to each other across the two NTB ports.

[3] Normal Linux or Windows PCI enumeration procedure.

Notes

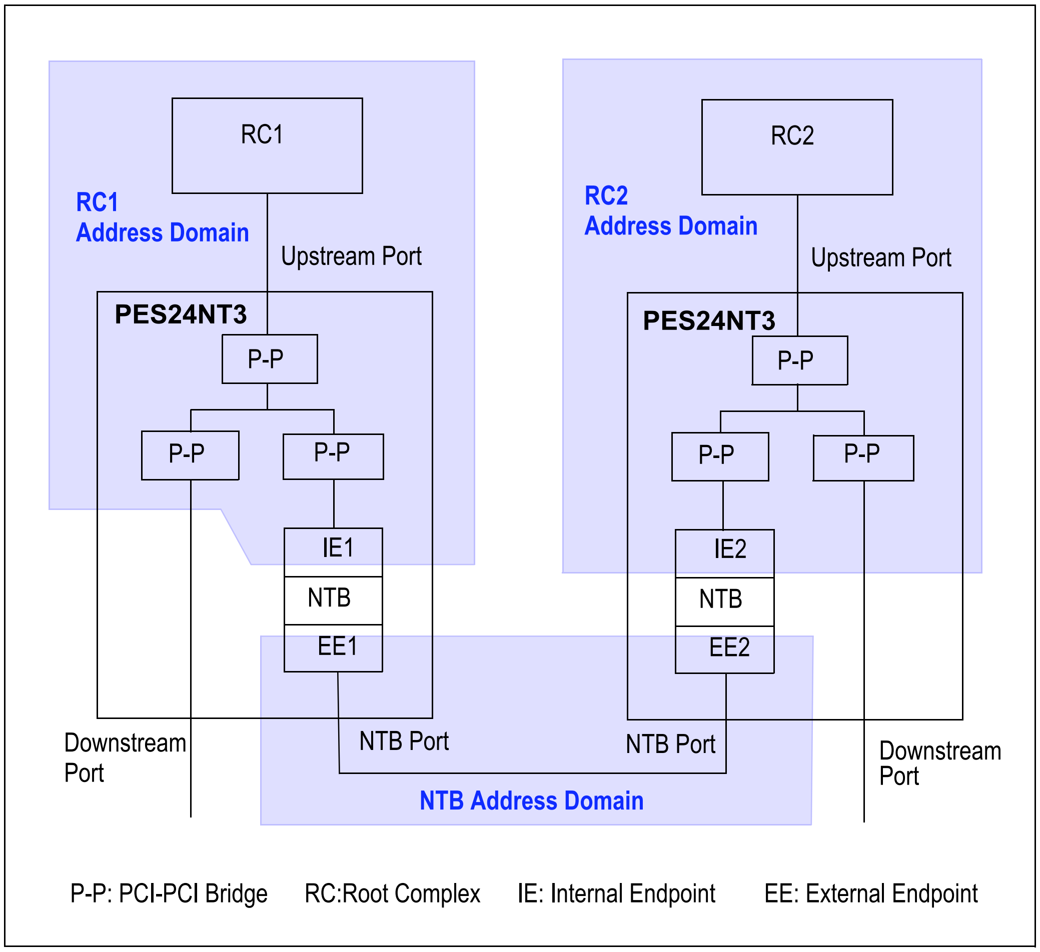

Figure 13 Dual NTB Topology

Summary

Non-transparent bridging may be used to connect two root complexes in a PCIe system. This allows the two root complexes to exchange data with each other. Detailed procedures and a theory of operation were given to show how a PES24NT3 device forwards and translates PCIe transaction packets across an NTB port. As discussed above, the PES24NT3 device provides all the features — such as address translation, ID translation, interprocessor communication features, and error checking and handling — to efficiently support a dual root complex system.

Revision History

January 23, 2007: Initial publication.

September 15, 2009: Changed app note title.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.