Non-transparent Bridging with IDT 89HPES32NT24G2 PCI Express® NTB Switch

Application Note AN-724

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 89HPES32NT24G2 device (referred to hereafter as PES32NT24G2).

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.

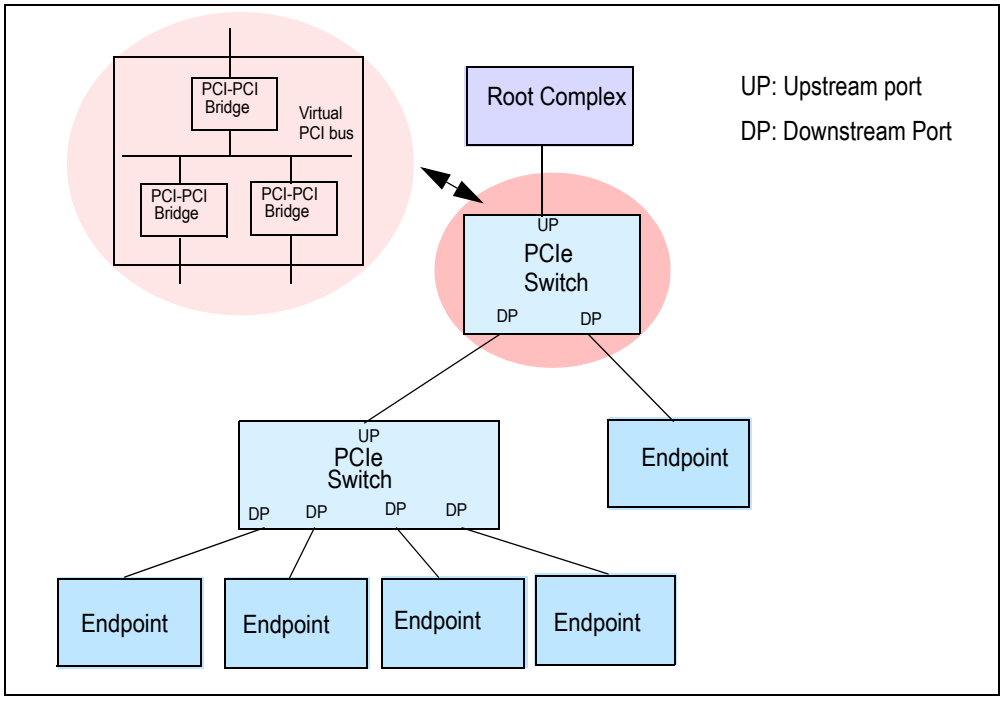

Figure 1 PCIe Tree Topology

Notes

In multi-Root 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 the PES32NT24G2

An architectural block diagram of NTB in the PES32NT24G2 is shown in Figure 2 . NTB is implemented as an NT function. An NT function may co-exist with an upstream switch port or exist as an NT function port (NTB port). Each NT function appears as a PCIe endpoint. There is a virtual NT interconnect within the switch to allow communication among the NT functions. The NT interconnect is invisible to the PCIe hierarchy.

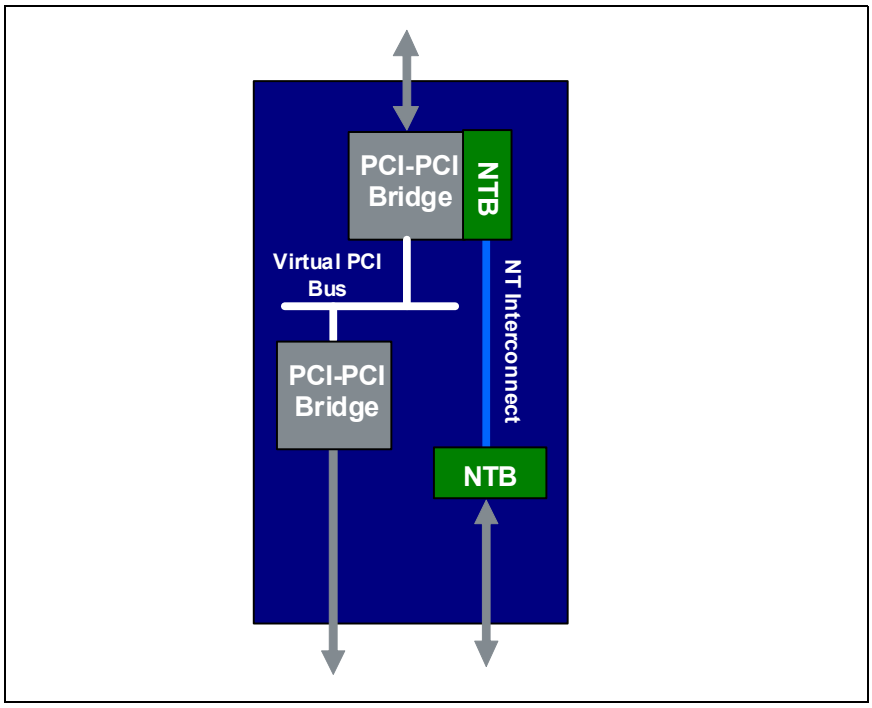

Figure 2 Architecture of NTB in PES32NT24G2

The PES32NT24G2 supports up to 8 NT functions. Figure 3 shows an example of a possible NTB configuration which has 4 partitions and 4 NT functions. The 4 NT functions are all connected via a virtual NT Interconnect to allow inter-domain communication. Two of the partitions represent two independent three-port transparent PCIe switches. The upstream port of each partition is configured to operate as an upstream port with NT function. Two other partitions represent two NT endpoints and are configured to operate as NTB ports. Another possible configuration is shown in Figure 4. In this configuration, 8 partitions are created and the NT functions are configured to operate as NTB ports.

Notes

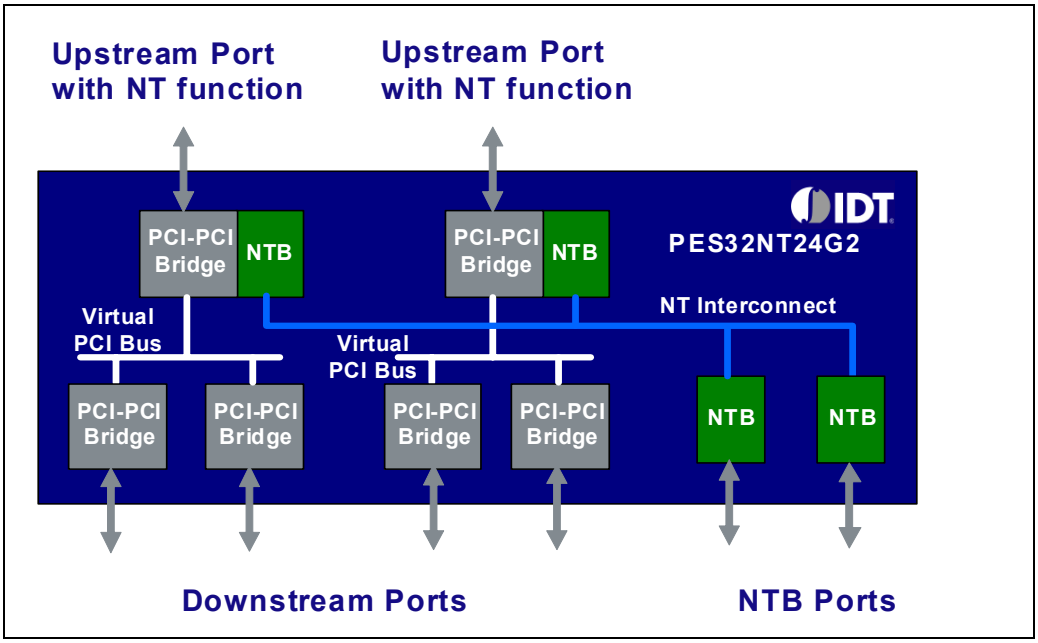

Figure 3 Possible Configurations of NT Functions

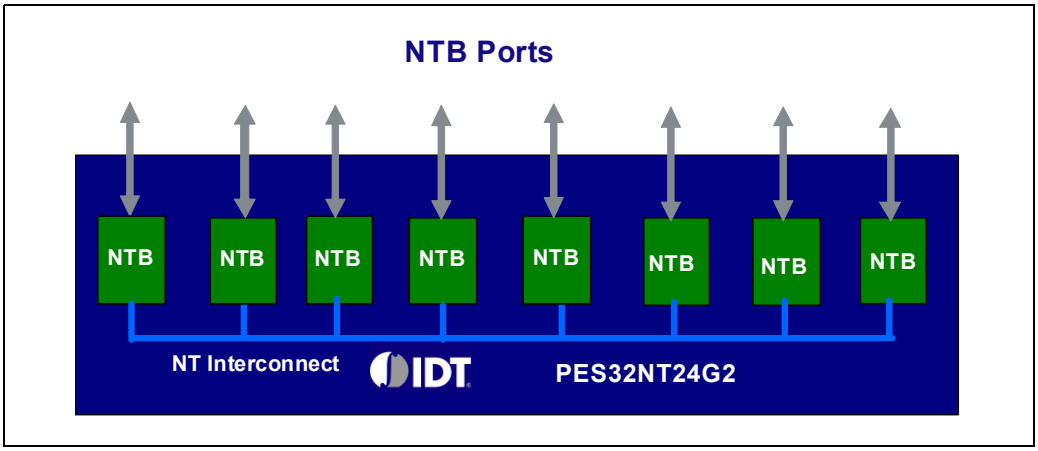

Figure 4 NTB on All Ports

System Architecture

This section provides a few examples of a system topology using NTB to connect multiple Root Complexes. The usage model of these topologies is to allow multiple Root Complexes to exchange data using NTB with a PCIe interface. There are other usage models for 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 PES32NT24G2 is shown in Figure 5. In this topology, the second root complex (CPU 2) is connected to the NTB port of the PES32NT24G2. Two address domains are created: CPU 1 address domain and CPU 2 address domain. CPU 1 and 2 can exchange data via the NTB port. In fact, any endpoint in the CPU 1 address domain may exchange data with CPU 2 and vice versa.

Notes

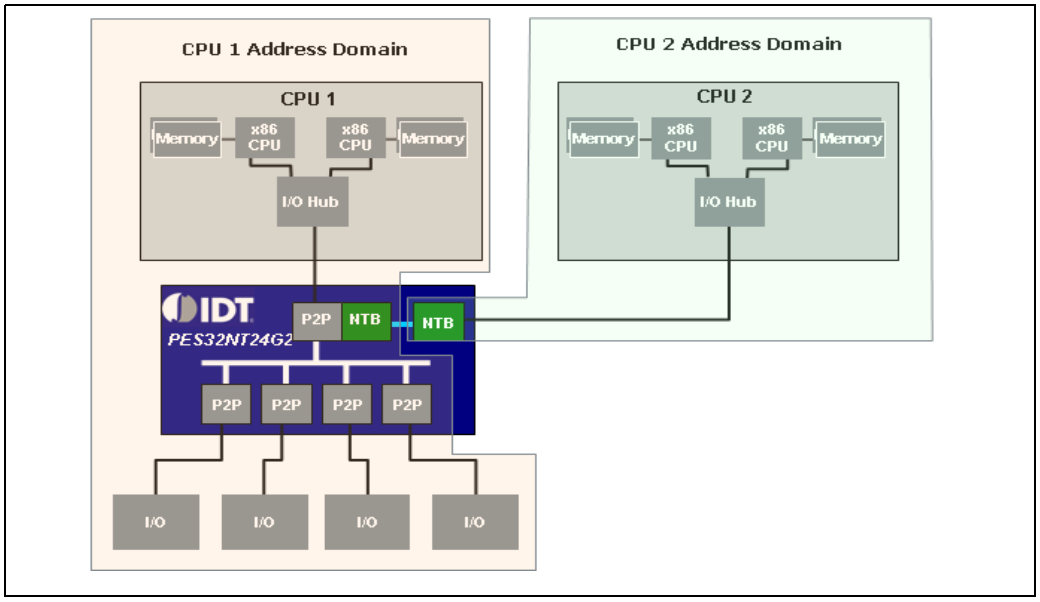

Figure 5 System Topology with Two Root Complexes

A similar example of a system that has 3 NT functions configured is shown in Figure 5. In this topology, the third root complex (CPU 3) is connected to the another NTB port of the PES32NT24G2. A total of three address domains are created: CPU 1, CPU 2, and CPU 3. CPU 1, CPU 2, and CPU 3 can exchange data via the NT functions. The PES32NT24G2 supports up to 8 NT functions; thus, this topology can be extended to connect up to 8 Root Complexes using a single PES32NT24G2.

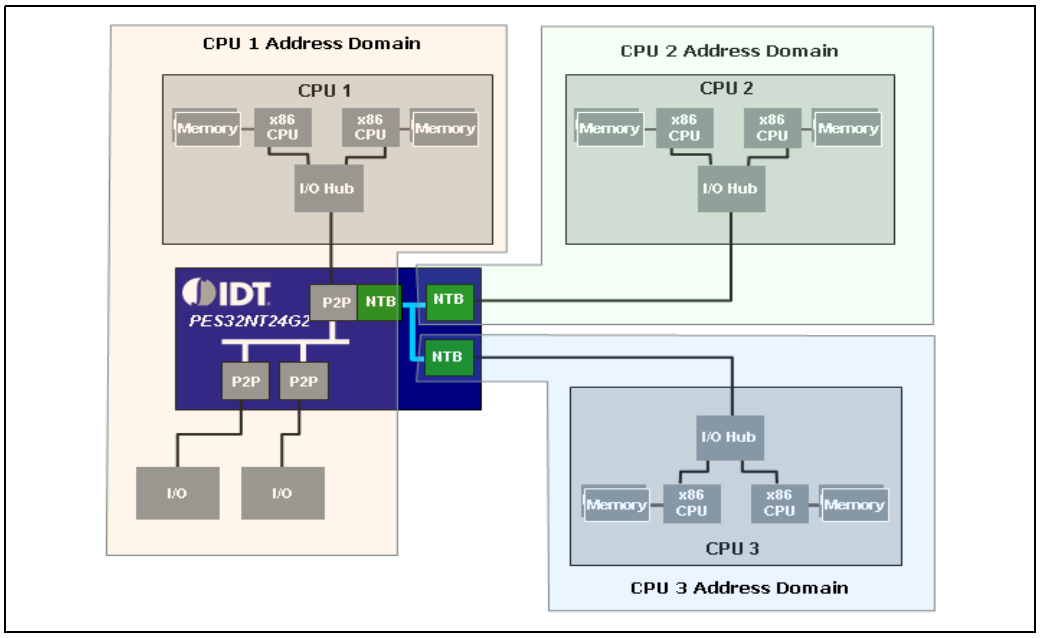

Figure 6 System Topology with Three Root Complexes

An example of a system that uses two PES32NT24G2s is shown in Figure 7. In this topology, the Root Complex CPU 1 sub-system is identical to the Root Complex CPU 2 sub-system. The two Root Complexes are connected to each other using two NTB ports. Three address domains are created: CPU 1 address

Notes

domain, CPU 2 address domain, and the NTB port address domain. This system works in a similar manner to the system shown in Figure 5. CPU 1 and 2 can exchange data via the NTB ports, and any endpoint in the CPU 1 address domain may exchange data with CPU 2 and vice versa.

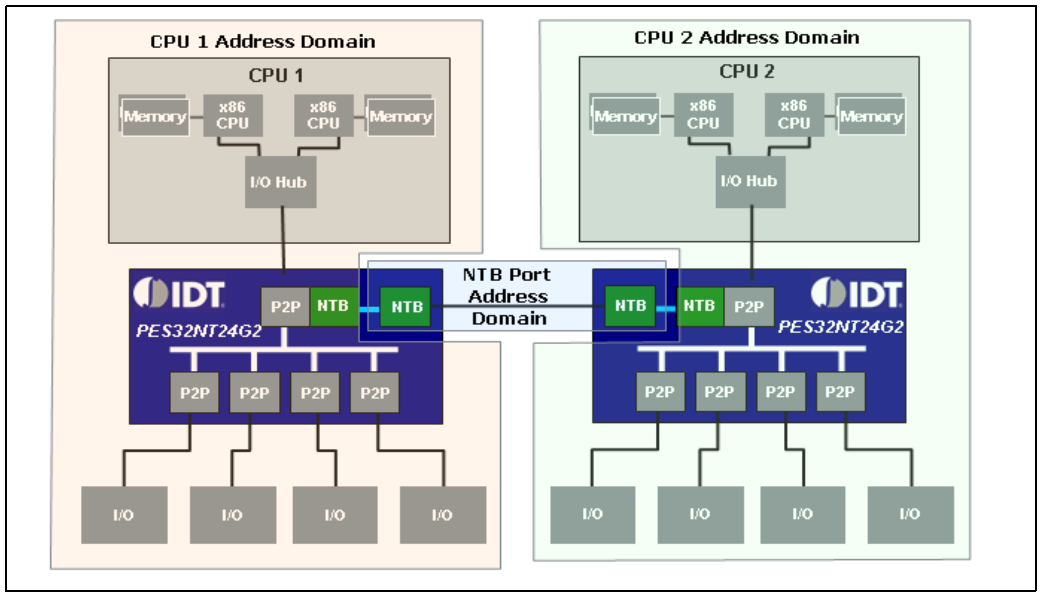

Figure 7 System Topology with NTB to NTB Connection

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 PES32NT24G2 across the NTB. Completions are ID-routed and are forwarded by the PES32NT24G2 across the NTB.

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

Address Translation

Each NT function contains six base address registers (BARs) in its Type 0 header. BARs can be mapped into 32-bit memory address space. Odd and even numbered BARs may be paired to form 64-bit address windows. BAR0 can be configured to enable memory-mapped access to the configuration registers in the NTB endpoint.

An NT function supports Direct Address Translation and Lookup Table Address Translation. All BARs may be configured to support Direct Address Translation. BAR2 and BAR4 may be configured to support Lookup Table Translation. A memory address may be translated from 32-bit to 64-bit or from 64-bit to 32-bit.

Each BAR has a corresponding setup register (BARSETUP) and translated base register (BARLTBASE and BARUTBASE) in the NTB endpoint configuration capability structure. There is a lookup table that may be used by BAR2 and BAR4 for Lookup Table Address Translation. The setup register contains fields that configure the corresponding BAR, such as the type of BAR (Memory or I/O), the address window size, Direct or Lookup Table Address Translation, and the destination partition after translation. The format of the setup register is shown in Figure 8. The translated base register contains the base address of the transac-

Notes

tions forwarded through the NT Interconnect 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.

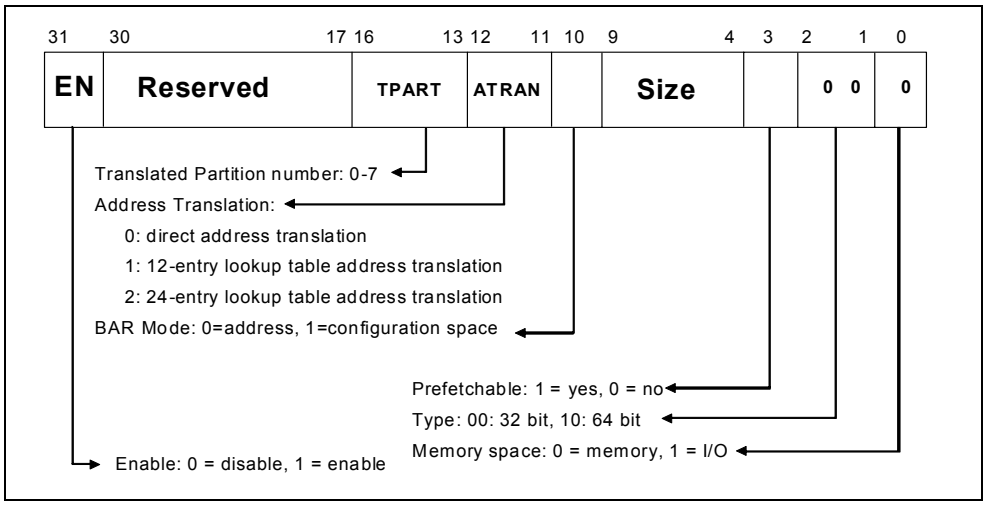

Figure 8 BARSETUP Register Format

The Direct Address Translation logic is shown in Figure 9. It shows how the address is translated when a packet is forwarded from one NTB endpoint to another NTB endpoint. When a Transaction Layer Packet (TLP) is received by an NTB endpoint, the address field is extracted from the PCIe transaction layer packet. The address and type are compared against BAR0 through BAR6. 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. The original address offset is added to the translated address before the packet is forwarded to the destination NTB endpoint (or partition). If the address does not find a match in BAR0 to BAR6, the packet is dropped.

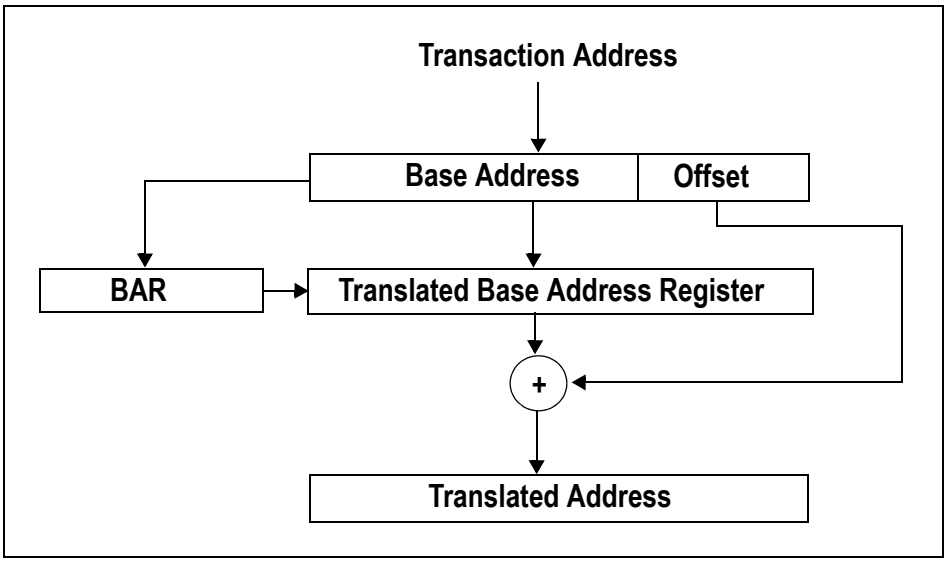

Figure 9 Direct Address Translation

The Lookup Table Address Translation logic is shown in Figure 10. BAR 2 or 4 may be configured to support Lookup Table Address Translation. BAR 2 may be configured to support either a 12-entry or 24-entry lookup table. BAR4 only supports a 12-entry lookup table. When both BAR 2 and 4 are configured to

Notes

support Lookup Table Address Translation, then BAR 2 only support a 12-entry lookup table. Each entry in the look table contains the translated address and the destination partition number where the packet is to be forwarded to.

In a Lookup Table Address Translation, the address field is logically divided into the base address, index and offset. The base address is to be used to compare the BAR registers. The index is used to specify one of the entries in the lookup table, and offset is the offset address within a memory window. When a packet is received by an NTB endpoint, the address field is extracted from the PCIe transaction layer packet. The base address and type are compared against BAR0 through BAR6. If the address falls within the window size of either BAR 2 or BAR 4, the base address and the index of the original address is replaced with the content of the corresponding lookup table entry (which contains the translated base address) as pointed to by the index. The original address offset is added to the translated address before the packet is forwarded to the destination NTB endpoint (or partition) as specified in the lookup table entry.

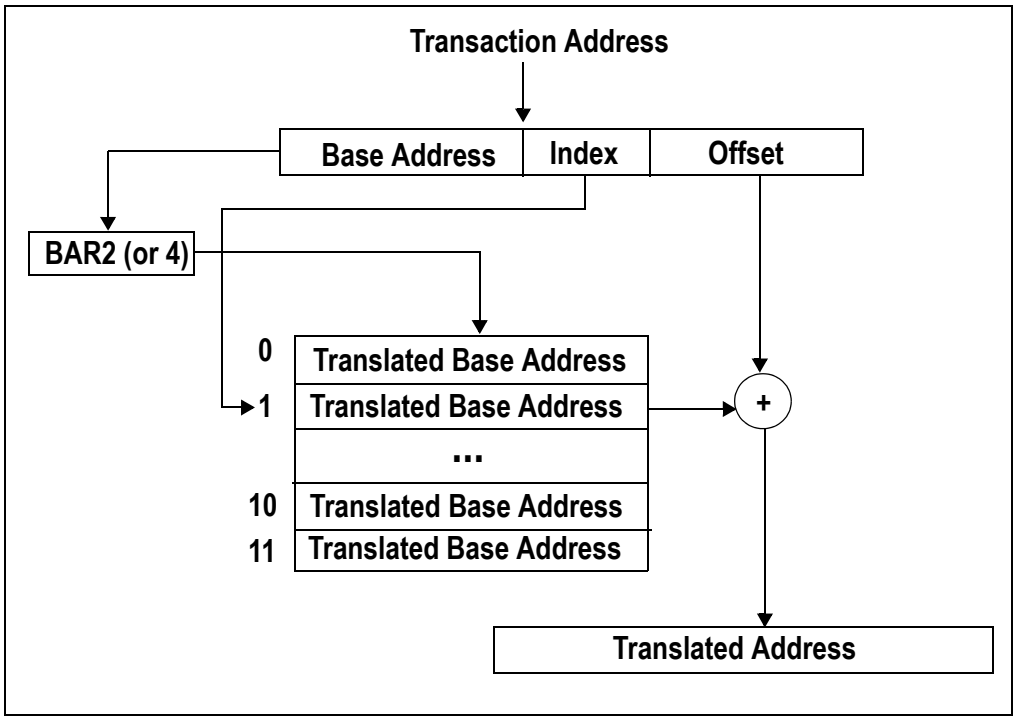

Figure 10 Lookup Table Translation

Address Translation Examples

A topology example of how address translation works is shown in Figure 11. In this example, three NT functions are enabled. The Root Complex (RC) connects to NTB 0 (partition 0), EP1 connects to the NTB 1 (partition 1) port, and EP2 connects to the NTB 2 (partition 2) port. Each Root Complex, EP1, and EP2 creates an outbound address window to access the memory address space of each other. Each one of them creates two inbound address windows to allow the others to access its local memory. It has a dedicated inbound address window for each of the other two NTB endpoints.

Notes

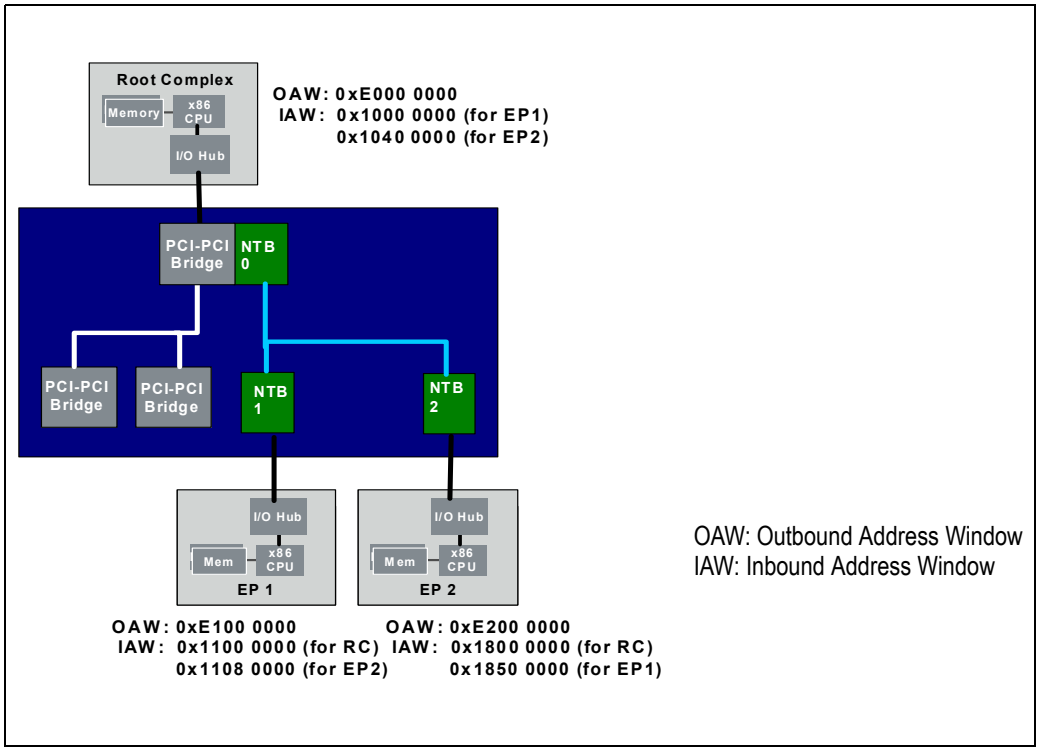

Figure 11 Address Translation Example Topology

The memory map of Address Windows is shown in Figure 12. In the example, EP1 is configured to use Direct Address Translation. The Root Complex and EP2 are configured to use Lookup Table Address Translation. Each RC, EP1, and EP2 allocates two local memory regions as the Inbound Address Windows. The size of each local memory window is 1 MByte. Each of the 1 MByte memory is to be mapped to the Outbound Address Window of the other two NTB endpoints. In this example, the Root Complex's local address window at 0x1000 0000 is mapped to one of the EP1's outbound address window and the local address window at 0x1040 0000 is mapped to one of the EP2's outbound address window. The RC and EP2 use Lookup Table Address Translation, one outbound address window is needed for each one of them. The outbound address window is then sub-divided into two memory windows to map to the other two NTB endpoints. EP2 uses direct address translation, and two outbound address windows are created in order to map the windows to RC and EP2's address space.

Notes

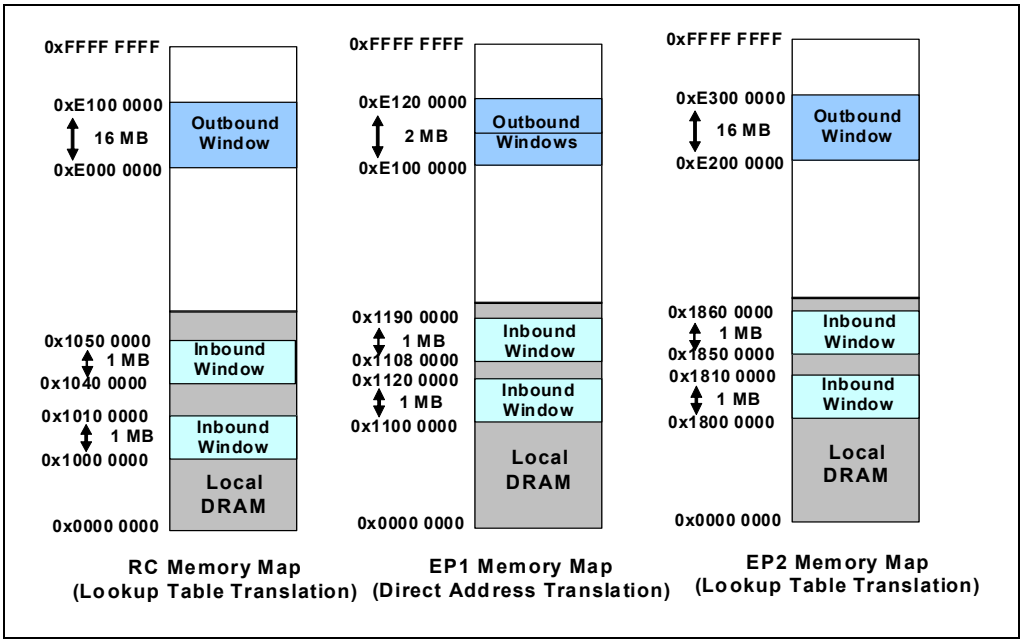

Figure 12 Memory Maps

The lookup table configuration for NTB0 is shown in Figure 13. BAR 2 is configured to use Lookup Table Address Translation with 12-entry. The size of BAR2 is 16 MBytes such that each entry represents 1 MByte of memory size. Only two entries are enabled in the lookup table. Entry 0 is configured to translate the address to 0x1100 0000 and the packet is to be forwarded to NTB 1 (partition 1) after address translation. Entry 1 is configured to translate the address to 0x1800 0000 and the packet is to be forwarded to NTB 2 after address translation.

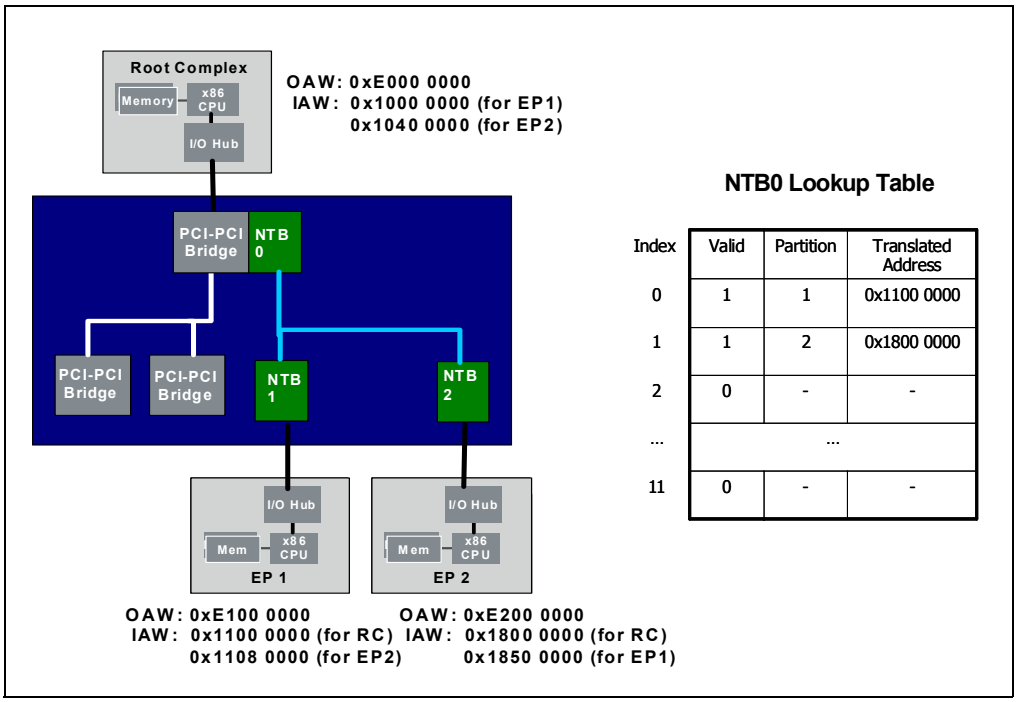

Figure 13 NTB0 Address Translation Configuration

Notes

The NTB0 address translation logic is shown in Figure 14. When a TLP is sent by the Root Complex to EP1, the address must be in the range of 0xE000 0000 to 0xE00F FFFF. The Base address is extracted from the TLP and compared to BAR 0 to 5. The base address (0xE0000 0000) matches BAR2 and the index (0) is used to access the lookup table. The TLP is forwarded to NTB 1 (partition 1) and the address is translated to have the base address of 0x1100 0000. The offset is then added to the translated address before the TLP is forwarded to EP1. When a TLP is sent by the Root Complex to EP2, the address must be in the range of 0xE010 0000 to 0xE01F FFFF. This has an index of 1 and the TLP is forwarded to NTB 2 (partition 2) and the address is translated to have a base address of 0x1800 0000. The offset is then added to the translated address before the TLP is forwarded to EP2.

EP2 is configured to use Lookup Table Address Translation and the address translation works identically with the Root Complex address translation.

.

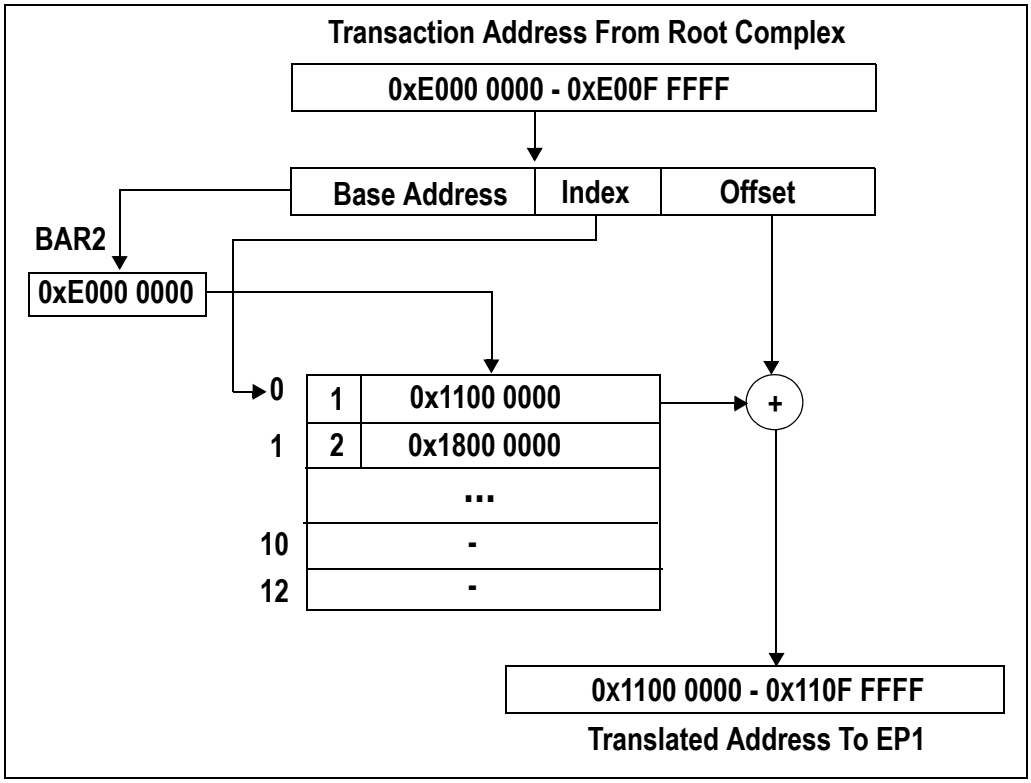

Figure 14 NTB0 Address Translation Logic Example

NTB1 is configured to use direct address translation. A BAR is required to map the outbound address windows to another partition (i.e. NTB). It opens an address window to NTB 0 and a second address window to NTB 2. The address translation configuration is shown in Figure 15. BAR 1 is configured to have a size of 1 Mbyte, destination partition 0 (NTB 0), and the translated base address of 0x1000 0000. This outbound address window is used to access the local memory of the Root Complex. BAR 2 is configured to have a size of 1 Mbyte, destination partition 2 (NTB 2), and the translated base address of 0x1850 0000. This outbound address window is used to access the local memory of EP2.

Notes

BAR Number	BAR Address	BARSETUP			BARBASE (Translated Base Address)
		Size	Address Translation	Partition	
1	0xE100 0000	20 (1MB)	0 (Direct)	0	0x1000 0000
2	0xE110 0000	20 (1MB)	0 (Direct)	2	0x1850 0000

Figure 15 NTB1 Address Translation Configuration

The address translation logic for NTB 1 is shown in Figure 16. When a TLP is sent by EP1 to EP2, the address must be in the range of 0xE110 0000 to 0xE11F FFFF. The base address is extracted from the TLP and compared to BAR 0 to 5. The base address (0xE110 0000) matches BAR2. The TLP is forwarded to NTB 2 (partition 2) and the address is translated to have the base address of 0x1850 0000. The offset is then added to the translated address before the TLP is forwarded to EP2. When a TLP is sent by EP1 to the Root Complex, the address must be in the range of 0xE100 0000 to 0xE10F FFFF. The TLP is forwarded to NTB 0 (partition 0) and the address is translated to have a base address of 0x1000 0000. The offset is then added to the translated address before the TLP is forwarded to the Root Complex.

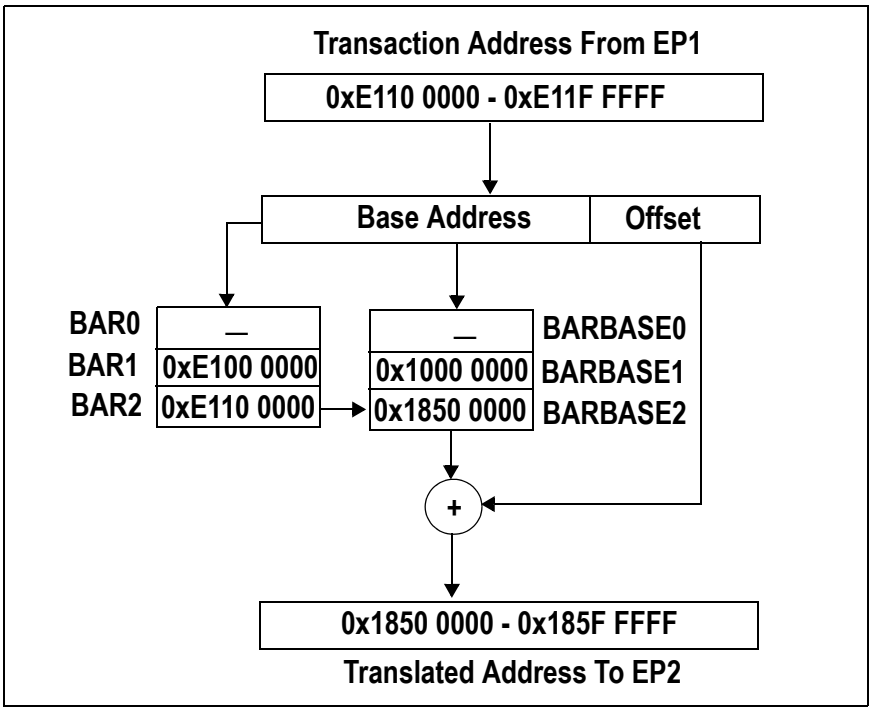

Figure 16 NTB1 Address Translation Logic Example

A topology example that has two PES32NT24G2s connected back to back is shown in Figure 17. In this example, two NT functions are enabled in each PES32NT24G2. RC1 creates an outbound address window (OAW) to access the memory address space of RC2. RC1 allocates 1 MByte of local memory to create an inbound address window (IAW) that is to be mapped to the OAW of RC2. RC2 has created a similar OAW and IAW for the same purpose.

Notes

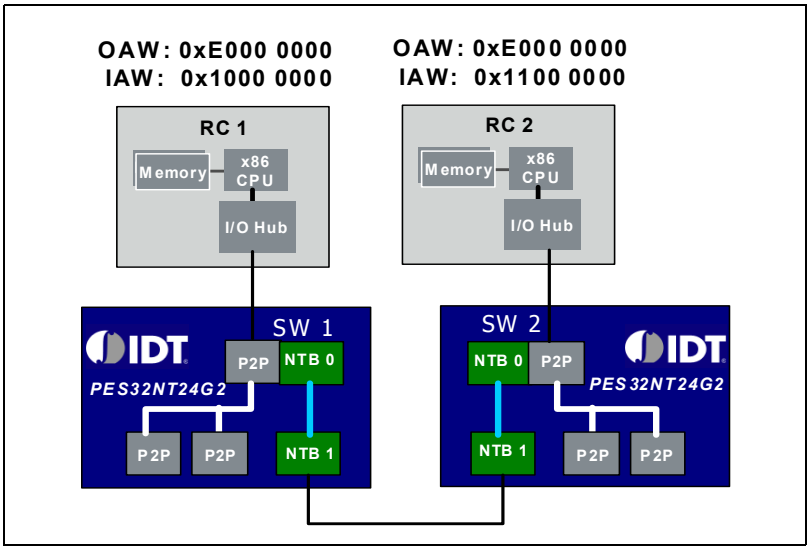

Figure 17 Topology Example with two PES32NT24G2s

There is no Root Complex in the NTB 1 domain. RC1 uses the punch-through feature (see section Punch-Through on page 1-22 for a definition) to configure SW2 NTB1 during initialization. RC1 sends configuration request to configure the BARs of SW2 NTB1. RC2 uses the punch-through feature to configure SW1 NTB1 during initialization. The BAR configuration for SW1 and SW2 is shown in Figure 18 after initial initialization. RC1 configures NTB1 BAR0 of SW2 to have a value of 0x0000 0000 such that RC1 can access all the registers of SW2 NTB1 using memory read and write operation instead of using punch-through to send configuration requests. It also configures SW2 NTB1 BAR2 to have a value of 0x0200 0000 to open an inbound address window into RC2's local memory. RC2 does the same configuration to SW1 NTB1.

SW 1	
BAR	Address
NTB0 BAR2	0xE000 0000
NTB1 BAR0	0x0000 0000
NTB1 BAR2	0x0200 0000

SW 2	
BAR	Address
NTB0 BAR2	0xE000 0000
NTB1 BAR0	0x0000 0000
NTB1 BAR2	0x0200 0000

Figure 18 SW1 and SW2 BAR Configuration

The lookup table configuration is shown in Figure 19. SW1 NTB0 BAR2 is configured to have two entries. The entry at index 0 contains the translated address to access the registers of SW2 NTB1. The entry at index 1 contains the translated address to access the local memory of RC2. SW1 NTB1 BAR2 contains a single entry to allow RC2 to access RC1's local memory. SW2 is configured similarly.

Notes

SW1 NTB0 BAR2 Lookup Table

Index	Valid	Partition	Translated Address
0	1	1	0x0000 0000
1	1	1	0x0200 0000
...	...		
11	0	-	-

SW2 NTB0 BAR2 Lookup Table

Index	Valid	Partition	Translated Address
0	1	1	0x0000 0000
1	1	1	0x0200 0000
...	...		
11	0	-	-

SW1 NTB1 BAR2 Lookup Table

Index	Valid	Partition	Translated Address
0	1	0	0x1000 0000
...	...		
11	0	-	-

SW2 NTB1 BAR2 Lookup Table

Index	Valid	Partition	Translated Address
0	1	0	0x1100 0000
...	...		
11	0	-	-

Figure 19 SW1 and SW2 Lookup Table Configuration

The SW1 and SW2 address translation logic is shown in Figure 20. When a TLP is sent by RC1 to RC2, the address must be in the range of 0xE010 0000 to 0xE01F FFFF. The base address is extracted from the TLP and compared to BAR 0 to 5. The base address (0xE000 0000) matches BAR2. The address has an index of 1 in the lookup table. The TLP is forwarded to NTB 1 (partition 1) and the address is translated to have the base address of 0x0200 0000. The offset is then added to the translated address before the TLP is forwarded. When SW2 receives the TLP, the base address is extracted and compared to BAR 0 to 5. The base address (0x0200 0000) matches BAR2. The address has an index of 0 in the lookup table. The TLP is forwarded to RC2 and the address is translated to have the base address of 0x1100 0000. The offset is then added to the translated address before the TLP is forwarded.

RC1 can access the registers of SW2 NTB1 using the address between 0xE000 0000 and 0xE000 0FFF. The base address (0xE000 0000) matches BAR2. The address has an index of 0 in the lookup table. The TLP is forwarded to NTB1 and the address is translated to have the base address of 0x0000 0000. The offset is then added to the translated address before the TLP is forwarded. When SW2 receives the TLP, the base address is compared to BAR 0 to 5. The base address (0x0000 0000) matches BAR0 and the TLP is used to access the registers of SW2 NTB1.

RC2 works in a similar way to access the local memory of RC1 and the registers of SW1 NTB1.

Notes

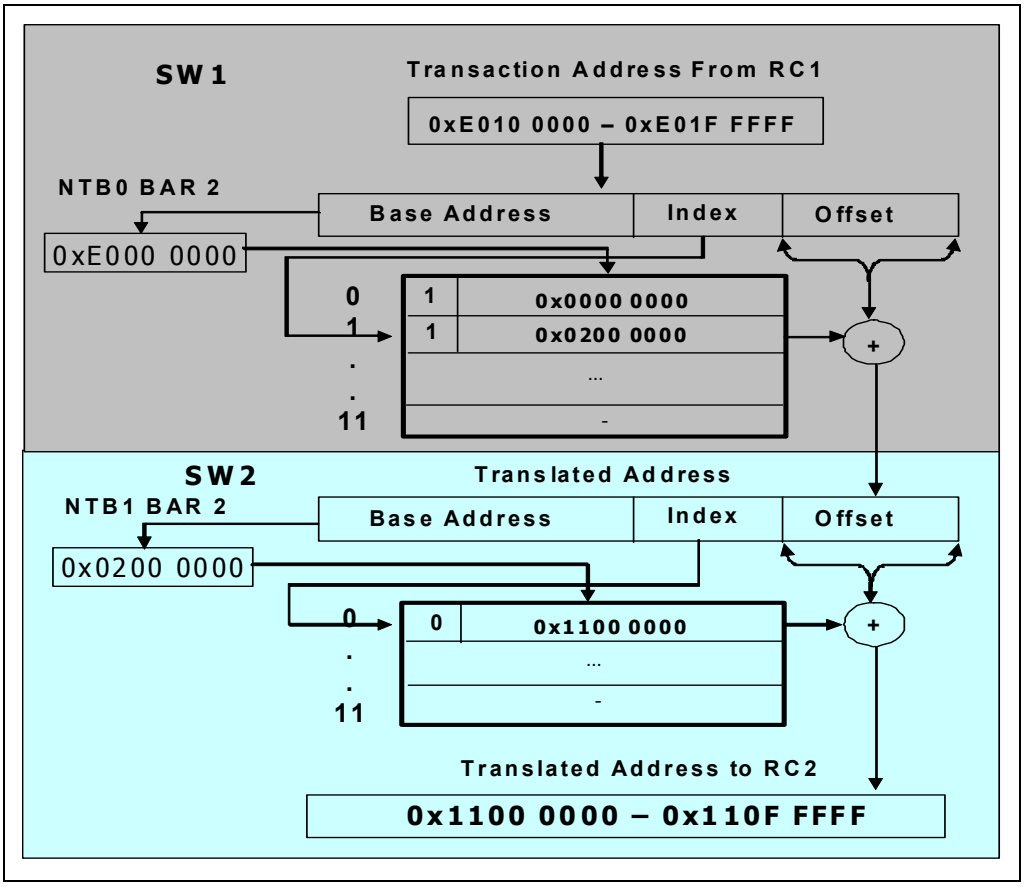

Figure 20 SW1 and SW2 Address Translation Logic

ID Routing and Translation

Associated with the PES32NT24G2 is a 64-entry NT mapping table. The NT mapping table is used to perform ID translation. The ID is the bus, device, and function number of a PCIe function. The NT mapping table is a global table shared by all NT functions. The format of the NT mapping table is shown in Figure 21 and the description of each field in the table is shown in Table 1. For the purpose of ID translation, only the Valid, Function, Device, Bus and Partition are described here. Please refer to the PES32NT24G2 user manual for details of the other fields.

Notes

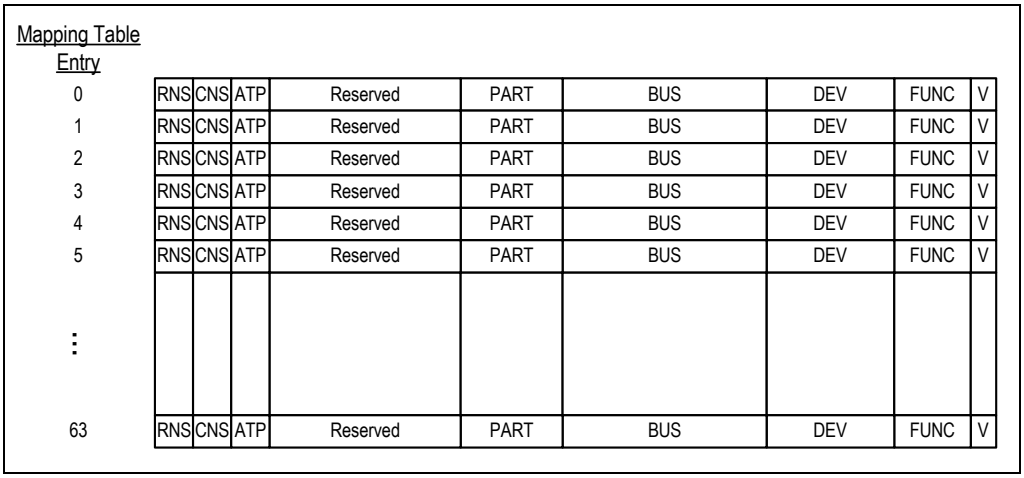

Figure 21 NT Mapping Table Format

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.
30	CNS	**Completion No Snoop Processing**. This field specifies the no snoop processing on completion TLPs.
31	RNS	**Request No Snoop Processing**. This field specifies the no snoop processing on request TLPs.

Table 1 NT Mapping Table Field Description

The NT mapping table contains the partition number, bus, device, and function (BDF) numbers of the initiators on that side of the endpoint whose transactions may be forwarded to the NT Interconnect to another partition. The V field indicates if the entry is Valid or not. The mapping table is filled in during system initialization. There is a single NT mapping table and is shared by all ports configured for NT operation.

Request ID Translation

When a request TLP is received by an NT endpoint that is to be routed on the NT interconnect, a requester ID lookup and translation operation is 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. The partition number and the requester ID are compared to each of the valid entries in the NT mapping table. When there is a match, the requester ID is translated as shown in Figure 22. The bus field is replaced by the captured bus number of the NT endpoint associated with the partition of the trans-

Notes

lated TLP. Bit 4 of the device field is set to one and Bit 3 of the device field is set to zero. The lower three bits of the device field and all three bits of the function field are replaced with the mapping table match entry number.

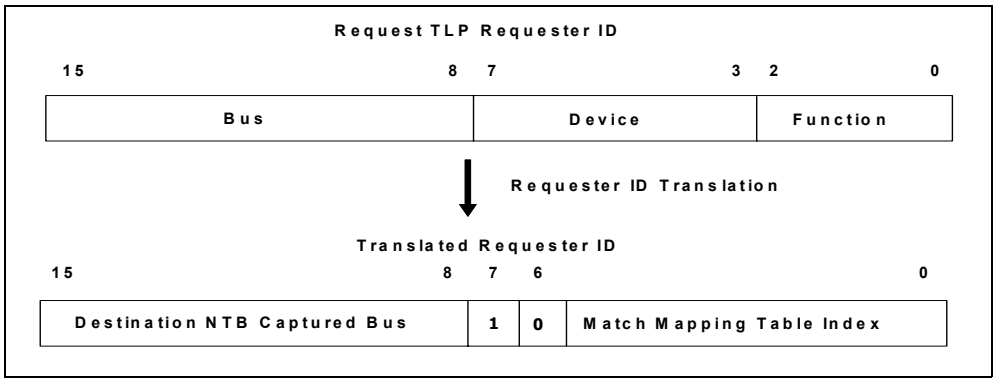

Figure 22 Request TLP Requester ID Translation

Completion ID Translation

When a completion TLP is received by an NT endpoint, the request ID lookup and completion ID translation is performed. The Requester ID must be in the valid range (i.e. Bit 7 and 6 of the Request ID must be "10") before a ID lookup is performed.

The lower 8-bit value of the Requester ID that consists of the device and function fields is extracted from the requester ID field. The top two bits are not used and the lower six bits form an NT Mapping table entry index. If the NT mapping table entry index points to a valid entry in the NT mapping table, the completion TLP is forwarded to the partition as specified in the NT mapping table. Before the TLP is forwarded, both the requester ID and the completer ID are translated as follows:

- The requester ID Bus, Device, and Function fields are replaced by the NT mapping table Bus, Device, Function fields.
- 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).

Requester ID Capture Register

In order to program the NT mapping table, the requester IDs of initiators in the PCI Express hierarchy must be known. Typically, at least the ID of the Root Complex is added to the NT mapping table. The ID of the Root Complex is system dependent.

To facilitate programming of the NT mapping table, the Requester ID Capture register (REQIDCAP) may be used to discover the ID of an initiator. When an initiator issues a configuration read request to the REQIDCAP register, a completion is generated that contains the ID of the initiator that issued the configuration read request.

Requester ID Translation Example

An example of an ID translation in shown in Figure 23. In this example, there is a single PES32NT24G2 with 3 NT functions enabled. The IDs of the Root Complex, EP1 and EP2 have the same value of 0.1.0. The IDs of NTB0, NTB1 and NTB2 are 1.0.1, 1.0.0 and 2.0.0 respectively. There are three initiators or requesters in the system. The IDs of the Root Complex, EP1 and EP2 are added to entry 0, 1, and 2 of the NT mapping table respectively.

Notes

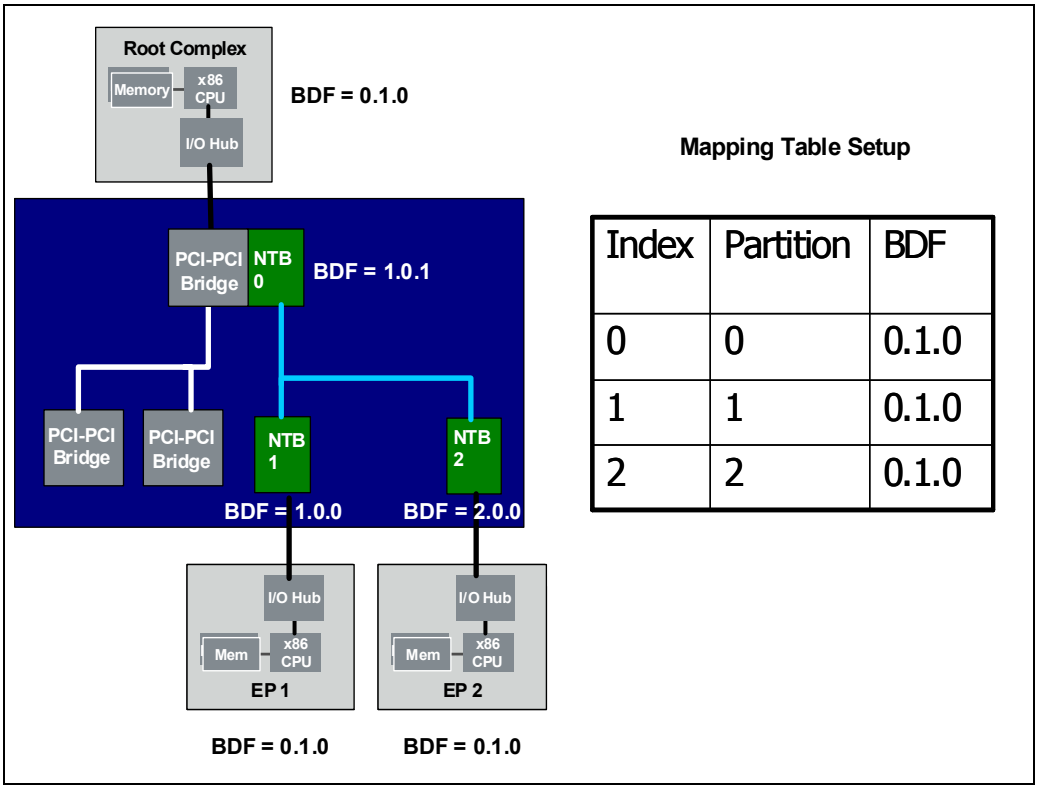

Figure 23 Requester ID Translation Example Topology

An example of the requester ID translation logic is shown in Figure 24. Root Complex performs a memory read request to EP1's local memory. When the request TLP is received by NTB 0, the TLP is routed by address to the destination NTB 1 as described in Address Translation. The address is translated before the TLP is forwarded. In addition to the address translation, the Requester ID has to be translated as well. The partition (0) and requester ID (BDF = 0.1.0) of the Root Complex are used to look up in the NT mapping table. A match is found and the matching entry has an index of 0. The requester ID is then replaced with the bus number of NTB 1 (1) and part of the device and function numbers are replaced with the matching NT mapping table index of 0. The forwarded packet has a requester ID of 1.16.0.

A corresponding completion packet is received by NTB 1 some time later. The requester ID and completer ID of the completion packet are 1.16.0 (Requester ID of the corresponding request packet) and 0.1.0 (EP1) respectively. Part of the device and function numbers of the Requester ID (0) are used as an index to the NT mapping Table. The BDF of the NT mapping table entry at index 0 (0.1.0) and the BDF of the NTB 0 (1.0.1) are used to replace the Requester and Completer ID in the completion packet respectively. The translated packet is forwarded to partition 0 as indicated in entry 0 of the NT mapping table.

Notes

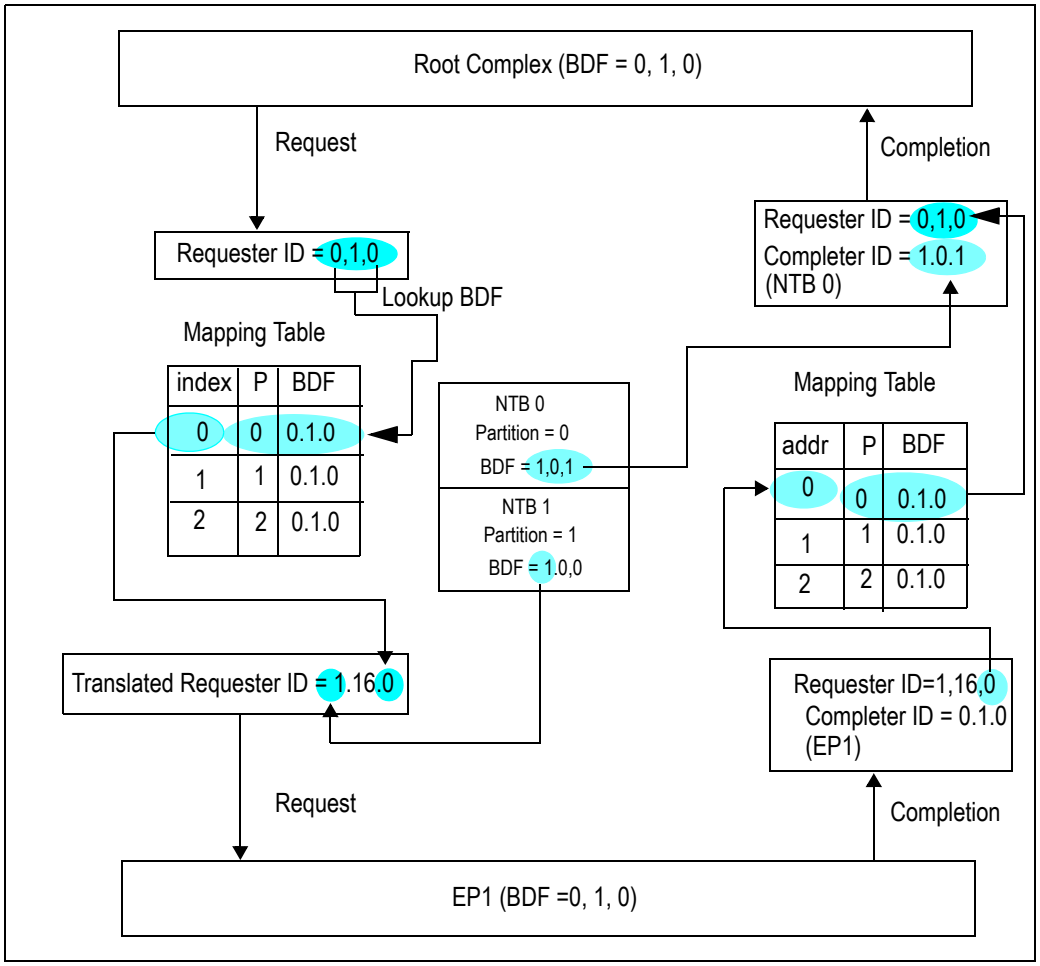

Figure 24 Requester ID Translation Logic

The ID translation works similarly when two NT ports are connected back-to-back as shown in Figure 25. During initialization, RC1 configures the BDF of SW2 NTB1 to be 0.16.0 using punch-through. RC2 configures the BDF of SW1 NTB1 to be 0.16.0 using punch-through. The SW1 and SW2 NT mapping tables are initialized as shown in Figure 25.

Notes

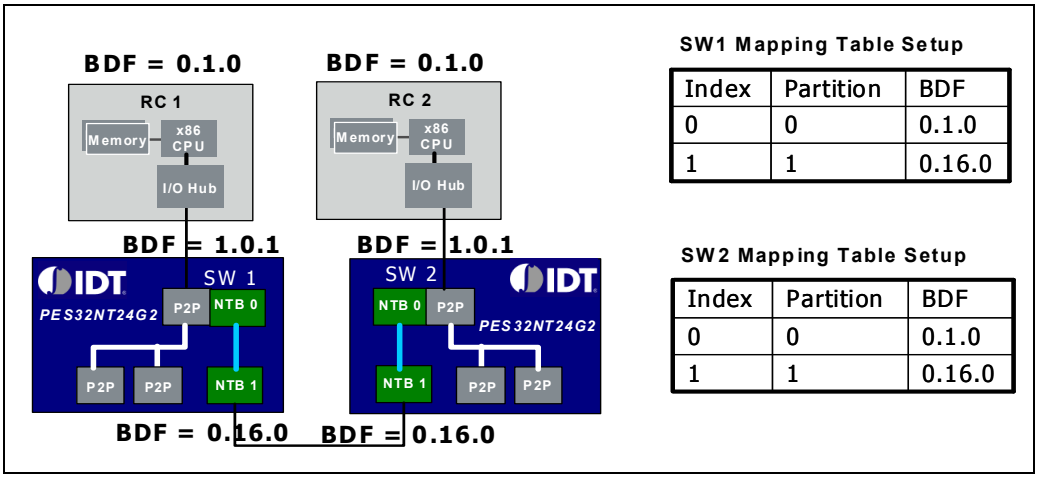

Figure 25 Requester ID Translation Example with Back to Back NTB ports

An example of the requester ID translation logic with back-to-back NTB ports is shown in Figure 24. RC1 performs a memory read request to RC2's local memory. When the request TLP is received by SW1 NTB0, the TLP is routed by address to the destination SW1 NTB1 as described in Address Translation. The address is translated before the TLP is forwarded. In addition to the address translation, the Requester ID has to be translated as well. The partition (0) and requester ID (BDF = 0.1.0) of RC1 are used to look up in the SW1 NT mapping table. A match is found and the matching entry has an index of 0. The requester ID is then replaced with the bus number of NTB1 (0), and part of the device and function numbers are replaced with the matching NT mapping table index of 0. The forwarded packet has a requester ID of 0.16.0.

When SW2 NTB1 receives the request TLP, the TLP is routed by address to the destination SW1 NTB0 towards RC2. The address and Requester ID are translated. The partition (1) and requester ID (BDF = 0.16.0) are used to look up in the SW2 NT mapping table. A match is found and the matching entry has an index of 1. The requester ID is then replaced with the bus number of NTB0 (1), and part of the device and function numbers are replaced with the matching NT mapping table index of 1. The forwarded packet has a requester ID of 1.16.1.

A corresponding completion packet is received by SW2 NTB0 some time later. The requester ID and completer ID of the completion packet are 1.16.1 (Requester ID of the corresponding request packet) and 0.1.0 (RC1) respectively. Part of the device and function numbers of the Requester ID (1) are used as an index to the SW2 NT mapping table. The BDF of the NT mapping table entry at index 1 (0.16.0) and the BDF of the SW2 NTB1 (0.16.0) are used to replace the Requester and Completer ID in the completion packet respectively. The translated packet is forwarded to partition 1 as indicated in entry 1 of SW2 NT mapping table.

The translated completion packet is received by SW1 NTB1. Both the requester ID and completer ID of the completion packet are 0.16.0. Part of the device and function numbers of the Requester ID (0) are used as an index to the SW1 NT mapping Table. The BDF of the NT mapping table entry at index 0 (0.1.0) and the BDF of SW1 NTB0 (1.0.1) are used to replace the Requester and Completer ID in the completion packet respectively. The translated packet is forwarded to partition 0 towards RC1 as indicated in entry 0 of the NT mapping table.

Notes

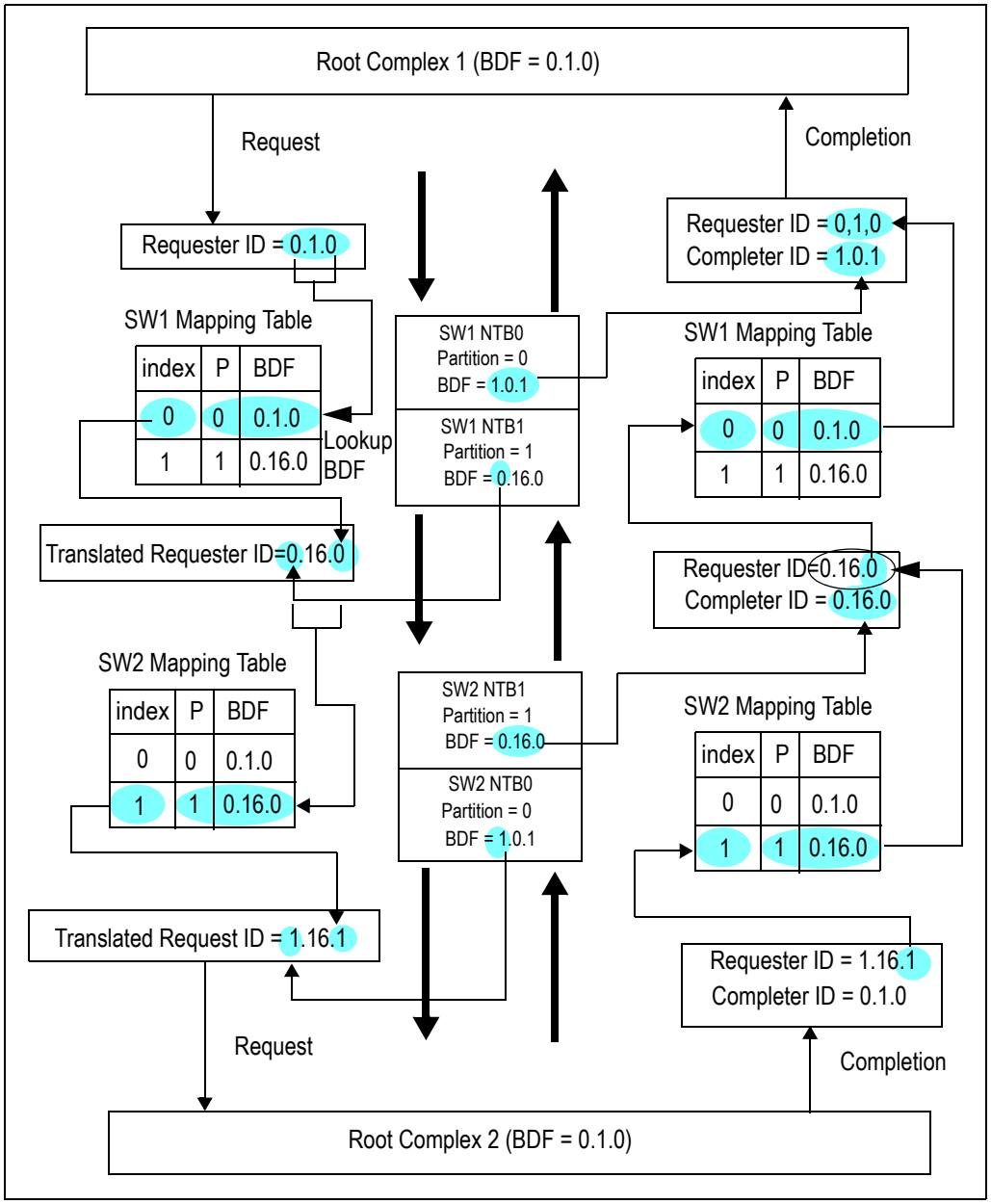

Figure 26 Request ID Translation Logic for Back to Back NTB Ports

Notes

Interprocessor Communications

The NT inter-domain communication capability provides Message and Doorbell Registers to support communication between processors in different PCIe domains (partitions).

Message Registers

NTB messaging allows 32-bit values to be passed between PCIe domains with interrupt notification. Each NT function has the capability to send or receive messages to/from any NT function in another PCIe domain. Reception of a message causes the NT function receiving the message to generate an interrupt towards the root-complex in its PCIe domain.

Each NT function has 4 outbound message registers and 4 inbound message registers. The PES32NT24G2 supports flexible mapping between outbound and inbound message registers, such that any outbound message register in a switch partition's NT function can be mapped to any inbound message in another partition's NT function. Each outbound message register can only be mapped to a single inbound message register. It is possible that multiple outbound message registers, from typically different NT functions, map to the same inbound message register in another NT function

When a switch partition receives an inbound message, the NT function associated with that partition generates an interrupt. The generation of the interrupt can be masked independently on each NT function, on a per-message basis. After the inbound message is read, the inbound message is cleared to allow reception of more messages.

An outbound message is only delivered when it's mapped inbound message register is empty. Otherwise, the delivery is deemed unsuccessful and an interrupt is generated by the NT function which contains the outbound message. The switch never queues outbound messages.

Doorbell Registers

NTB doorbells facilitate signaling of events between PCIe domains. For example, after completing a data transfer across the NTB, an agent in a PCIe domain may use a doorbell to notify the target agent in another PCIe domain that the transfer is finished.

Each NT function has the capability to send doorbells to any other NT function, across switch partitions. Reception of a doorbell causes the NT function receiving the doorbell to generate an interrupt towards the root-complex in its PCIe domain.

Each NT function has a 32-bit outbound doorbell register and a 32-bit inbound doorbell register. Each bit in these registers is associated with a doorbell (i.e., bit 0 is associated with doorbell 0, bit 1 is associated with doorbell 1, etc.). The PES32NT24G2 supports flexible mapping of outbound-to-inbound doorbells between its NT functions. Specifically, for each outbound doorbell, it is possible to indicate which switch partitions receive the corresponding inbound doorbell. An outbound doorbell may be received by zero, one, or several partitions simultaneously, as long as each receiving partition has an NT function in its upstream port.

When a switch partition receives an inbound doorbell, the NT function associated with that partition generates an interrupt. The generation of the interrupt can be masked independently on each NT function, on a per-doorbell basis.

Doorbells are edge-triggered and never queued by the switch. Reception of an inbound doorbell results in interrupt generation only when a prior inbound doorbell (if any) has been serviced.

Configuration

All normal PCIe endpoint configurations must be performed before the NTB forwards a transaction. For example, the BAR registers, address translation, and the NT mapping table must be configured before memory transactions are routed through the NT interconnect.

Notes

Configuration Space

Associated with each of the NTB functions is a 4KB PCIe configuration space containing a Type 0 header. Refer to the PES32NT24G2 user manual for details on the organization of these configuration spaces.

The root complex configures the NT function which is in the same partition. It accesses the configuration space registers by performing configuration read and write operations. BAR0 of each NT function 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.

Punch-Through

The NT function has the capability to generate PCIe 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 PCIe hierarchy associated with the NT endpoint. In essence, the NT endpoint may be crosslinked to another endpoint or to a switch device and may issue configuration requests to configure these devices. Punch-through requests are always emitted on the NT function's link.

Error Detection and Handling

End-to-End CRC (ECRC) is supported for transactions that are forwarded through the NT interconnect. A new ECRC must be computed for each packet each time the packet is modified. When a packet is forwarded through the NT interconnect 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 NT function 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 the link of the NT function. Error messages generated as a result of detected errors are sent to the root complex of that particular NT function.

Initialization

For memory transactions to be routed across the NT interconnect, the following configuration must be performed on all the NT functions:

- The Memory Access Enable (MAE) bit must be set to enable an NT function to forward memory transactions.
- The Bus Master Enable (BME) bit must be set to enable an NT function to generate transactions.
- The Completion Enable (CPEN) bit must be set to send completion TLPs that have crossed the NT interconnect.
- BARSETUP[0-5] 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[0-5] registers must be set to configure the translated base address.
- BAR[0-5] registers must be set to configure the address windows.
- Address translation lookup table must be configured if Lookup Table Address Translation is used. The registers LUTOFFSET, LUTLDATA, LUTMDATA, and LUTUDATA are used to configure the lookup table.
- NT mapping table must be initialized to specify the ID translation.

Notes

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 PES32NT24G2 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 PES32NT24G2, a combination of EEPROM loading and software programming may be a better solution.

At a minimum, the BARSETUP[0-5] 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 PES32NT24G2 may be initialized using a combination of EEPROM load and software programming.

The message and door bell registers are used for communicating between the 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 one root complex (RC1) to the other root complex (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.

Single PES32NT24G2

The topology of the first example, which has only a single PES32NT24G2, is shown in Figure 11. The procedure to initialize the Root Complex and EP1 are described below.

BARSETUP registers are initialized during power-on reset using EEPROM load. The configurations of the BARSETUP registers are shown in Table 2. BARSETUP 0 and 2 registers are initialized by the EEPROM for NTB0. BARSETUP 0 and 1 registers are initialized by the EEPROM for NTB1.

	NTB0	NTB1
BARSETUP0	Enabled and mapped to configuration space, size is 4 KBytes.	Enabled and mapped to configuration space, size is 4 KBytes
BARSETUP1	Disabled	Enabled with 32-bit non-prefetchable memory address space, size is 1 MByte, direct address translation, translated partition is 0,

Table 2 EEPROM Configuration

Notes

	NTB0	NTB1
BARSETUP2	Enabled with 32-bit non-prefetchable memory address space, size is 16 MBytes, 12-entry lookup address translation	Disabled.

Table 2 EEPROM Configuration

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. NTB0 is initialized as follows:

1. The root complex allocates 4 Kbytes of PCI address space and writes the base address to BAR0.
2. The root complex allocates 16 Mbytes of PCI address and writes the base address to BAR2. This address happens to be 0xE000 0000 in this example.
3. Based on the vendor ID and device ID of the NT endpoint, the root complex loads and passes control to the NTB device driver.
4. The NTB device driver sets the Memory Access Enable bit, Bus Master Enable bit, and the Completion Enable bit on NTB0.
5. The NTB device driver discovers the ID of the root complex to be 0.1.0 by reading the Requester ID Capture register.
6. The NTB device driver discovers its port number by reading the PCIe Link Capabilities register. The port number is used as an index to the Switch Port Control register to discover the partition number (0) of which this port is in.
7. The NTB device driver adds the partition number 0 and the ID 0.1.0 to entry 0 of the NT mapping table.
8. The NTB device driver writes a known pattern to EP0 using the outbound message register. It then waits for EP0 to be ready by checking its inbound message register.
9. At this point, no transaction is forwarded to the NT interconnect.

EP1 also follows the "normal"[1] PCI initialization and enumeration procedure after power-on. NTB1 is initialized as follows:

1. EP1 allocates 4 Kbytes of PCI address space and writes the base address to BAR0.
2. EP1 allocates 1 Mbyte of PCI address and writes the base address to BAR1. This address happens to be 0xE100 0000 in this example.
3. Based on the vendor ID and device ID of the NT endpoint, EP1 loads and passes control to the NTB device driver.
4. The NTB device driver sets the Memory Access Enable bit, Bus Master Enable bit, and the Completion Enable bit on NTB1.
5. The NTB device driver discovers the ID of EP1 to be 0.1.0 by reading the Requester ID Capture register.
6. The NTB device driver discovers its port number by reading the PCIe Link Capabilities register. The port number is used as an index to the Switch Port Control register to discover the partition number (1) of which this port is in.
7. The NTB device driver adds the partition number 1 and the ID 0.1.0 to entry 1 of the NT mapping table.
8. The NTB device driver checks its inbound message register to discover that the root complex is ready. It then sends a known pattern to the root complex's inbound message register to indicate that EP1 is ready.

[1] Normal Linux or Windows PCI enumeration procedure.

Notes

When the root complex detects that EP1 is operating using the inbound message register, the root complex sends an IPC PDU to EP1 requesting EP1 to allocate 1MByte of memory to be used as the translated base address. When EP1 receives the memory allocation IPC PDU, it allocates 1MByte of memory from the system memory. EP1 then returns the base address of this 1 Mbyte of memory to the root complex using the message registers.

Upon receiving the base address from EP1, the root complex adds this base address and the partition 1 into entry 0 of the BAR2 address lookup table. The root complex is ready to send an address-routed request to EP1.

When EP1 detects that the root complex is operating, EP1 sends an IPC PDU to the root complex requesting the root complex to allocate 1MByte of memory to be used as the translated base address. When the root complex receives the memory allocation IPC PDU, it allocates 1MByte of memory from the system memory. The root complex then returns the base address of this 1 Mbyte of memory to EP1 using the message registers.

Upon receiving the base address from the root complex, EP1 programs this base address to the BAR1 Lower Translated Base Address register. EP1 is ready to send an address-routed request to the root complex. At this point, the initialization process is completed. The root complex and EP1 are ready to send address-routed requests to each other across the NTB port.

Two PES32NT24G2

The topology of the second example, which has two PES32NT24G2, is shown in Figure 25. The NTB ports are connected to each other back-to-back. The PES32NT24G2 supports the crosslink feature and hence the two NTB ports will link train. This creates three address domains: the RC1 address domain which includes SW1 NTB0 and all the SW1 PCI-PCI Bridges, the RC2 address domain which includes SW2 NTB0 and all the SW2 PCI-PCI bridges, and the NTB address domain which is between (and includes) SW1 NTB1 and SW2 NTB1. There is no root complex on the NTB address domain. All error messages and interrupts that were generated by SW1 NTB1 are dropped by SW2 NTB1 and vice versa. The PES32NT24G2 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 SW2 NTB1. SW1 NTB1 may also be configured by RC2 using this method.

BARSETUP registers are initialized during power-on reset using EEPROM load. The configurations of the BARSETUP registers are shown in Table 3. Both SW1 and SW2 have the same EEPROM configuration. BARSETUP 0 and 2 registers are initialized by the EEPROM for both NTB0 and NTB1.

	NTB0	NTB1
BARSETUP0	Enabled and mapped to configuration space, size is 4 KBytes.	Enabled and mapped to configuration space, size is 4 KBytes
BARSETUP2	Enabled with 32-bit non-prefetchable memory address space, size is 16 MBytes, 12-entry lookup address translation.	Enabled with 32-bit non-prefetchable memory address space, size is 16 MBytes, 12-entry lookup address translation.

Table 3 EEPROM Configuration

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. SW1 NTB0 and NTB1 are configured as follows:

1. RC1 allocates 4 Kbytes of PCI address space and writes the base address to BAR0.
2. The root complex allocates 16 Mbytes of PCI address and writes the base address to BAR2. This address happens to be 0xE000 0000 in this example.

[2] Normal Linux or Windows PCI enumeration procedure.

Notes

3. Based on the vendor ID and device ID of the NT endpoint, RC1 loads and passes control to the NTB device driver.
4. The NTB device driver sets the Memory Access Enable bit, Bus Master Enable bit, and the Completion Enable bit on NTB0.
5. RC1 uses the punch-through feature on NTB1 to generate a configuration read request to get the vendor ID and device ID of the device that connects to NTB1. It discovers that it is connecting to another NTB port. This is a back-to-back NTB connection.
6. The NTB device driver sets the Memory Access Enable bit, Bus Master Enable bit, and the Completion Enable bit on NTB1.
7. The NTB device driver discovers the ID of RC1 to be 0.1.0 by reading the Requester ID Capture register.
8. The NTB device driver discovers its port number by reading the PCIe Link Capabilities register. The port number is used as an index to the Switch Port Control register to discover the partition number (0) of which this port is in.
9. The NTB device driver adds the partition number 0 and the ID 0.1.0 to entry 0 of the NT mapping table.
10. The NTB device driver discovers the port number of NTB1 by reading the PCIe Link Capabilities register. The port number is used as an index to the Switch Port Control register to discover the partition number (1) of which NTB1 is in.
11. The NTB device driver adds the partition number 1 and the ID 0.16.0 to entry 1 of the NT mapping table. NTB device driver always uses the ID 0.16.0 for back-to-back NTB connection. Any ID may be chosen by the software designer.
12. The NTB device driver uses the punch-through feature to send a configuration read request to find out the BAR requirements for SW2 NTB1. SW2 BAR0 requires 4 KBytes of address space and SW2 BAR1 requires 1 MByte of address space.
13. The NTB device driver sets SW2 NTB1 BAR0 to be 0x0000 0000 and SW2 NTB1 BAR2 to be 0x0200 0000. These values are chosen by the NTB device driver in this example and may contain any values chosen by the system designer.
14. The NTB device driver adds partition 1 and the translated address 0x0000 0000 to entry 0 of the address lookup table for SW1 NTB0 BAR2. It also adds partition 1 and the translated address 0x0200 0000 to entry 1 of the address lookup table for SW1 NTB0 BAR2. These translated addresses match the content of SW2 NTB1 BAR0 and BAR2. At this point, RC1 can access all the NTB registers of SW2 NTB1 by doing a memory read/write request to the address 0xE000 0000.
15. The NTB device driver writes a known pattern to the inbound message register of SW2 NTB0 using the SW2 NTB1 outbound message register. It then waits for RC2 to be ready by checking its inbound message register.

SW2 NTB0 and NTB1 are initialized in exactly the same way as SW1 NTB0 and NTB1.

When RC1 detects that RC2 is operating using the SW1 NTB0 inbound message register, RC1 sends an IPC PDU to RC2 requesting RC2 to allocate 1MByte of memory to be used as the translated base address. When RC2 receives the memory allocation IPC PDU, it allocates 1MByte of memory from the system memory. RC2 then returns the base address of this 1 Mbyte of memory to RC1 using the SW1 NTB1 message registers.

Upon receiving the base address from RC2, RC1 adds this base address and partition 0 to entry 0 of the SW2 NTB1 BAR2 address lookup table. RC1 is ready to send an address-routed request to RC2.

When RC2 detects that RC1 is operating using the SW2 NTB0 inbound message register, RC2 sends an IPC PDU to RC1 requesting RC1 to allocate 1MByte of memory to be used as the translated base address. When RC1 receives the memory allocation IPC PDU, it allocates 1MByte of memory from system memory. RC1 then returns the base address of this 1 Mbyte of memory to RC2 using the SW2 NTB1 message registers.

Notes

Upon receiving the base address from RC1, RC2 adds this base address and partition 0 to entry 0 of the SW1 NTB1 BAR2 address lookup table. The 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.

Summary

Non-transparent bridging may be used to connect two or more root complexes in a PCIe system. This allows multiple root complexes to exchange data with each other. Detailed procedures and a theory of operation were given to show how a PES32NT24G2 forwards and translates PCIe transaction packets across an NTB port. As discussed above, the PES32NT24G2 provides all the features — such as address translation, ID translation, interprocessor communication features, and error checking and handling — to efficiently support a multiple root complex system.

References

PCI Express Base Specification Revision 1.1

PCI Express Base Specification Revision 2.0

IDT 89HPES32NT24G2 PCI Express® Switch User Manual

Application Note AN-713, Introduction to the PES32NT24G2 PCI Express® Switch Features, August, 2009.

Application Note AN-722, Inter-Domain Communication in the PES32NT24G2 PCI Express® Switch

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

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/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.