Enabling Multi-Host System Designs with PCI Express Technology

Although not part of the PCI Express spec, nontransparent bridging is completely consistent with it and can support multiprocessor systems with only minimal software initialization efforts.

by John Gudmundson
PLX Technology

For PCI Express technology to realize its full potential as *the* interconnect solution for chip-to-chip and backplane switching fabric applications, it needs to support multiprocessor-based designs. Since PCI Express protocols were designed from the beginning to be fully software-compatible with their predecessors, PCI and PCI-X, they were not originally designed for such multi-hosted systems. Just as these previous technologies have utilized industry-standard nontransparent bridging (NTB) for such designs, so has PCI Express (PCIe) technology. By utilizing NTB, PCIe applications can now include switched fabrics in multi-hosted topologies including intelligent adapter cards, dual-host systems with redundancy, and even blade server systems.

In legacy PCI-X systems, it was originally assumed there would be only one microprocessor in the topology. Upon power-up this processor attempts to discover all devices present, determine their memory requirements and map these into system memory. A series of control and status registers (CSRs) are used by this discovery and configuration software. For transparent bridges, the CSR with a "Type 1" header informs the processor to keep enumerating beyond this bridge as additional devices lie downstream. These bridges with Type 1 headers include CSR registers for primary, secondary and subordinate bus numbers, which, when programmed by the host, define the CSR addresses of all downstream devices.

Endpoint devices have a "Type 0" header in their CSRs to inform the processor that no additional devices lie downstream. These CSRs include base address registers (BARs) used to request memory and I/O apertures from the host. Both header types include a class code that indicates the type (1 or 0), as well as subclass, device ID and vendor ID fields. Through this header format and addressing rules, the host searches through the entire topology until it reaches all endpoints. Until then, it reads all the class codes of all the discovered devices and assigns bus numbers to all bridges. At the end of this discovery, the system knows which devices are present along with their memory and I/O space requirements. If a subsequent processor is added to the system, both processors will attempt to enumerate and memory map the entire system. This will result in memory mapping conflicts, multiple processor attempts to service the same system requests and system failure.

Nontransparent Bridging Isolates Processor Domains

PCI-X and now PCIe devices have subsequently solved this multiprocessor issue by adding NTB. In addition to the electrical isolation of transparent bridges, NTB adds logical isolation by providing processor domain partitioning and address translation between the memory-mapped spaces of these domains. With NTB, devices on either side of the bridge are not visible from the other side. However, a path is provided between the two buses for data to be transferred between the processor domains. With NTB applied to ports in bridges or switches, these ports will expose a Type 0 header on both sides of the port and appear as an endpoint to

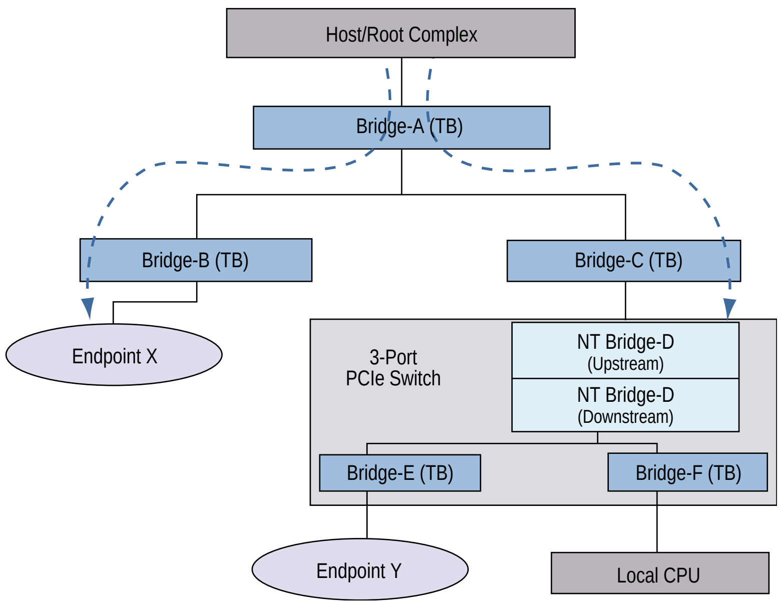

Figure 1 Processor domain partitioning with nontransparent bridging.

discovery software, even though additional elements are present on the other side. Processors on their respective side of the bridge or switch only enumerate until a Type 0 header is found.

Figure 1 illustrates this isolation. Here, a system host will enumerate through Bridge A and Bridge B (both transparent) on the left branch of the diagram until it reaches the endpoint X. On the right side branch, the host will stop enumeration at Bridge D. Similarly, the local CPU will enumerate through Bridge E and F (both virtual bridges within the switch) and discover and determine the requirements of endpoint Y, but will not attempt to discover elements beyond Bridge D. After this initialization and configuration is completed, all devices have been discovered and the memory, I/O and other requirements are known and addresses assigned for each processor domain. Yet, with two memory

Nontransparent Bridging Requires Minimal Software

Configuring nontransparent bridging (NTB) PCI Express devices requires only a few additional steps beyond those for transparent bridges. NTB devices have two separate configuration register sets—one each for upstream and downstream sides—therefore, they need to be individually programmed. To enable this, control bits are used to block access to these interfaces. A primary bus access bit must be set from the local side processor or EEPROM to allow access to this port. Similarly, a secondary bus access bit must also be set prior to access from the secondary side and can either have a default value or be programmed by EEPROM. Only after the registers are programmed will these control bits be enabled to allow access. These registers should not be subsequently changed prior to any reset.

There are four groups of registers for NTB that require programming. One set includes the primary interface's standard control and status registers (CSRs), which are composed of the first 64 bytes of the configuration header and exclude the address base address registers (BARs). This set is common to both interfaces. The values are normally assigned from EEPROM and include vendor and device ID and device class. A second set includes the PCIe interface configuration that describes the PCIe attributes including maximum payload size and link widths; these values are also common to both interfaces. The third set includes the address BAR configuration registers including those for setup and translation. Communication CSRs, including doorbell and scratchpad, are the fourth set. They are used to set up and define the behavior of the signaling and messaging interfaces used by the host and local processors to communicate with each other. Doorbell registers are used to pass interrupts between the processors and scratchpad registers can be used to pass control and status information.

On power-on-reset, all internal registers except "sticky bits" are set to their default values; this includes setting the primary bus access and secondary bus access bits to zero to disallow PCIe port access from both sides of the NTB device. The CSR contents, including most of the standard CSR space of the primary interface and all base aspects of the device, are then loaded from serial EEPROM and the secondary bus access is set to 1, allowing access to the CSRs from the downstream side. Next, the remaining primary side registers, including setup and translation registers, are configured by the local processor and the primary bus access bit is enabled to allow upstream access to these registers. Here, the secondary side processor creates and configures the resource sizes for the primary side BARs and the forwarding translations. If all resources on the local side are "static," it is possible the values could be loaded from EEPROM and prevent added software intervention on the local side. But processor-based programming gives more flexibility and limits the size requirements of the EEPROM. With these steps completed the PCIe host can run its standard discovery and enumeration software just as with pure transparent systems.

spaces, translations of addresses and device ID fields (for ID-routed packets) are needed to enable transactions to cross from one memory space to the other.

Address Translation

Communication between processor domains requires memory mapping between each domain. This mapping is accomplished through translation registers within the BARs. Each NTB port has two sets of BARs, one each for the primary side (upstream port) and secondary side (downstream port). BARs are used to define address translating windows into the memory space on the other side of the bridge and allow the forwarding of transactions. These NTB-based BARs expand upon the basic capabilities of transparent bridge BARs because each BAR has a setup register, which defines the size and type of the window, and an address translation register. While transparent bridges forward all CSRs based on bus numbers, NTB devices only accept CSR transactions addressed to them. Two methods used in NTB include direct address translation and look-up table-based translation.

In direct address translation the addresses of all upstream and downstream transactions are translated by adding an offset to the BAR in which the transaction landed. Base translation registers within the BARs are used to set up these translations. Figure 2 illustrates this shift from the primary side address map to the secondary address map. The BAR is a 32-bit register and segregated into two sections, a lower 14 bits for window size determination and 18 upper bits for the starting location of the information in each of the memory spaces. BARs may also be combined in pairs to allow 64-bit addressing.

In look-up table-based address translation, the BAR uses a special look-up table for address translation of transactions that fall within its window. This approach provides more flexibility in mapping local addresses to host bus addresses since the location of the index field within the address is programmable to adjust window size. Figure 3 illustrates the use of an index field within the BAR. The index is used to provide the upper bits for the new memory location.

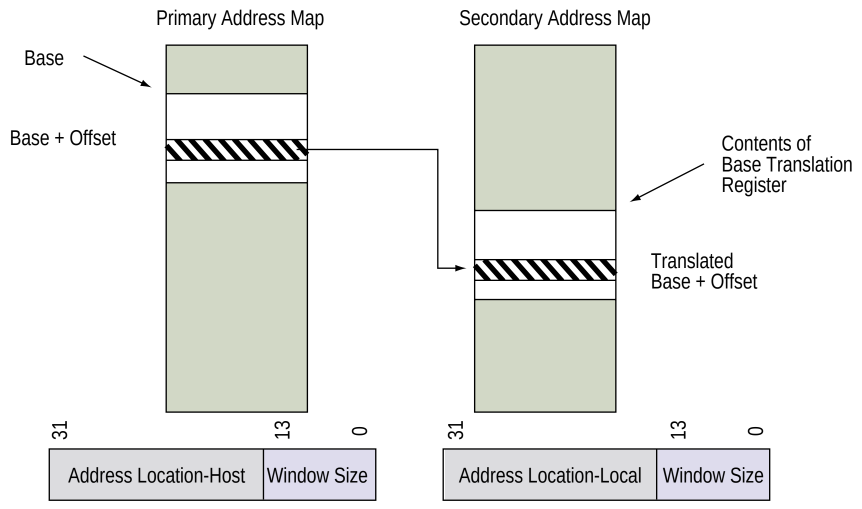

Figure 2 Direct address translation with base translation registers.

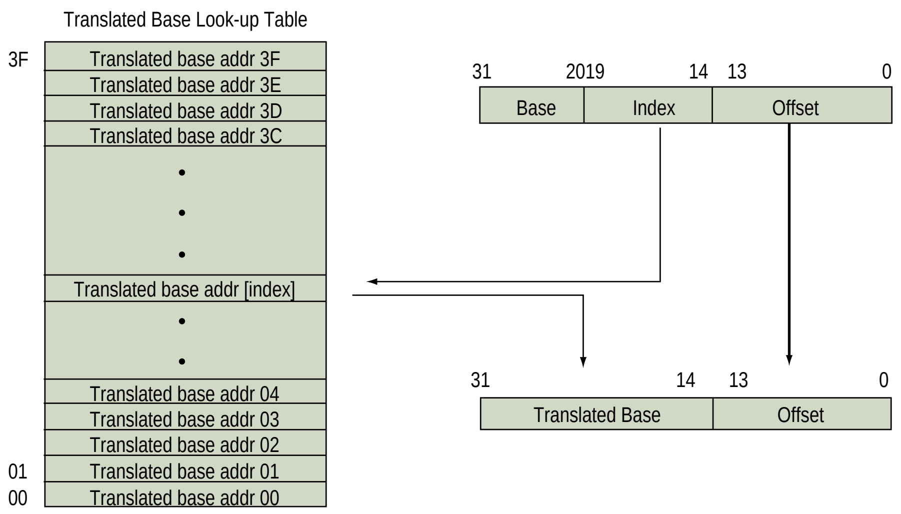

Figure 3 Look-up table-based address translation.

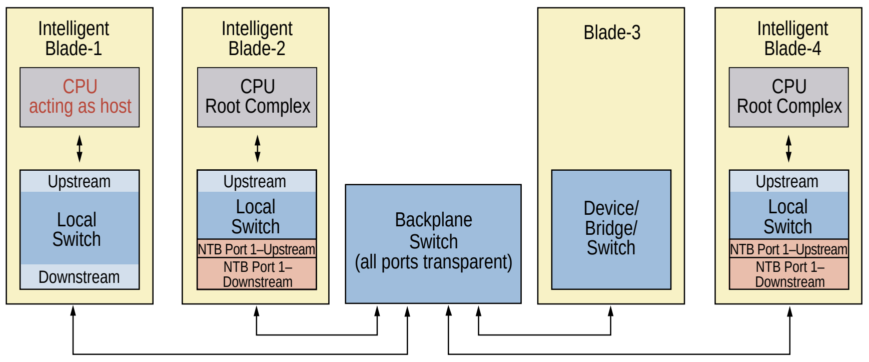

Figure 4 Four-blade system enabled with nontransparent-enabled bridge and switches.

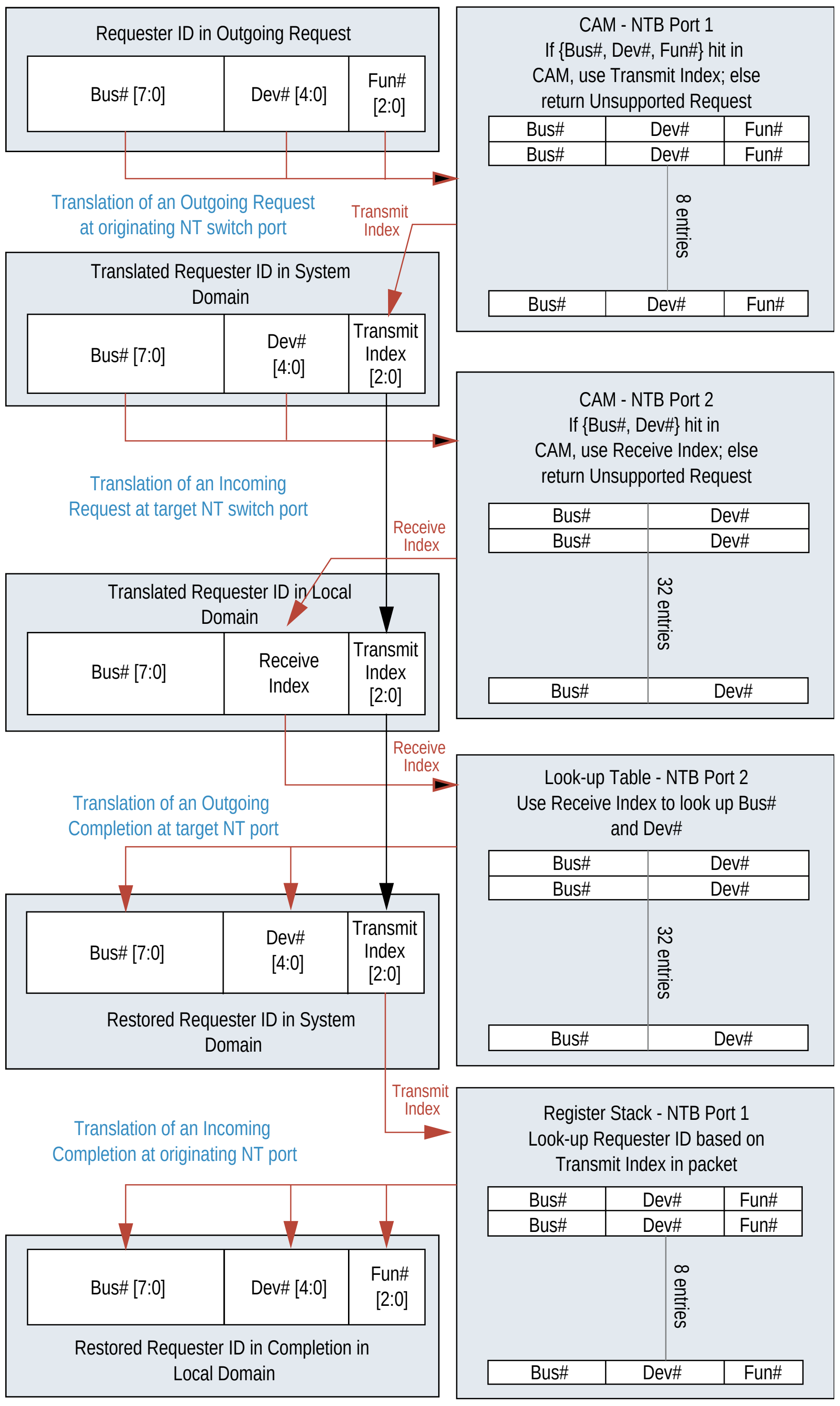
Requester ID in Outgoing Request
Bus# [7:0]
Dev# [4:0]
Fun# [2:0]
CAM - NTB Port 1
If {Bus#, Dev#, Fun#} hit in CAM, use Transmit Index; else return Unsupported Request
Bus# Dev# Fun#
8 entries
Translation of an Outgoing Request at originating NT switch port
Transmit Index
Translated Requester ID in System Domain
Bus# [7:0]
Dev# [4:0]
Transmit Index [2:0]
CAM - NTB Port 2
If {Bus#, Dev#} hit in CAM, use Receive Index; else return Unsupported Request
Bus# Dev#
32 entries
Translation of an Incoming Request at target NT switch port
Receive Index
Translated Requester ID in Local Domain
Bus# [7:0]
Receive Index
Transmit Index [2:0]
Look-up Table - NTB Port 2
Use Receive Index to look up Bus# and Dev#
Translation of an Outgoing Completion at target NT port
Restored Requester ID in System Domain
Register Stack - NTB Port 1
Look-up Requester ID based on Transmit Index in packet
Translation of an Incoming Completion at originating NT port
Restored Requester ID in Completion in Local Domain

Figure 5 The four steps of routing devices through requester ID: A) Translation of an outgoing request across the NTB Port 1; B) Translation of an incoming request through NTB Port 2; C) Inverse translations of the outgoing completions through NTB Port 2; and D) Translation of the incoming completion through NTB Port 1.

Requester ID Translation

Although memory and I/O read and write transactions are routed by addresses, configuration reads and writes (both types 0 and 1), completions and some messages require device ID routing and NTB devices also allow for this. The NTB device must either take ownership of non-posted requests passing through it or translate their requester IDs in a way that ensures their completions can be routed back to their requesters.

Routing devices through requester ID translation is the preferred method because it avoids complexity and having to save state information for every outstanding transaction. In order to route request/completions where two NTB devices are encountered between source and destination, four translation steps are needed. Figure 4 shows an example of a blade server system where three of the blades comprise processor-based systems with NTB deployed as shown. Request transactions are sent from one blade behind an NTB device through the fabric to a second intelligent blade behind a second NTB device and the corresponding completion is routed back to the requestor blade. The four steps include: translation of an outgoing request across the NTB Port 1; translation of an incoming request through NTB Port 2; inverse translations of the outgoing completions through NTB Port 2; and translation of the incoming completion through NTB Port 1.

Figure 5 illustrates the four translation steps starting with an outgoing request through an NTB device, which requires an 8-entry content addressable memory (CAM). Only the 3-bit function value will vary because the bus and device of the node directly behind the NTB device are known. The CAM entries support all outgoing requests and any number of outstanding requests made by a single node. The CAM is configured by EEPROM or local firmware before it is possible to send requests to the system domain. When the outgoing request arrives at the first NTB port, the packet requester ID is associated into this CAM. If the ID falls within the CAM, a corresponding transmit index replaces the function number field of the requester ID; otherwise, an unsupported request completion is returned. The contents of the bus and device number fields are replaced with bus and device fields last written to the NTB port's registers.

At the second NTB device the incoming request is translated using a 32-entry CAM. If the incoming bus/device number falls within the CAM entries, a 5-bit receive index is substituted in the device field; otherwise an unsupported request completion is returned. The bus number is replaced with that second NTB device's upstream bus number and the packet forwarded to the destination node. The function number field retains the transmit index value. Only 32 entries are required as the NTB device bus number is known and only the 5-bit device field may vary, though this limits the number of requesting devices elsewhere in the system doing peer-to-peer transfers with devices behind the NTB device to 32.

The corresponding outgoing completion for the original request is translated at the second NTB device through a look-up table (LUT) such that the NTB port appears to be the source of the completion. At this LUT, the receive index is used to look up the new bus and device field numbers that replaced the previous bus number and receive index fields. The transmit index remains the same. The completion packet is sent out of the NTB device. As this completion packet is received at the incoming port of the first NTB device, it is again translated; here, the transmit index originally assigned by this NTB device is used to look up the original bus and device number fields as well as the function number. The completion packet is then forwarded to the requester node.

Specification Compatibility

Since NTB technology is not part of the PCIe specification, steps have been taken to ensure compatibility. Each NTB port acts as both an endpoint during configuration and discovery, as well as a switch/bridge for other types of transactions. Yet, the PCIe specification provides different and potentially conflicting capabilities for the two types of devices. For instance, endpoints may not be located on the virtual PCI bus of a switch; but this restriction does not apply to NTB devices because PCIe switch/bridges only use the same type of CSR header format as an endpoint and do not actually function as endpoints.

Endpoints must provide infinite completion credits while switches must flow control completions. Yet, since NTB devices are not true endpoints, they provide flow control for their link mates. Endpoints are allowed 32 outstanding transactions by default while a switch has no such limit to the number it forwards. But as mentioned, NTB devices do not take ownership of non-posted requests and hence no limit is placed on the number of outstanding transactions.

The PCIe specification requires all devices to capture both bus and device numbers on every configuration write request and use these values in request and completion IDs. But as noted earlier, with NTB, where a requester ID of requests forwarded from the system host into the switching fabric occurs, a CAM index value (transmit index), not the mandated device number, is used. Nevertheless, functionality is still provided for and the PCIe specification can still be met if a switch is limited to one NTB port. As the only NTB port, it is the only port capable of sending such a requester ID and only the bus number is needed to route the completion back to the NTB port.

PLX Technology
Sunnyvale, CA.
(408) 774-9060.
[www.plxtech.com].

PLX
TECHNOLOGY®
THE I/O INTERCONNECT SOLUTION