



PCI-SIG ENGINEERING CHANGE NOTICE

TITLE:	MMCONFIG ECN for PCI Express
DATE:	February 17, 2004
AFFECTED DOCUMENTS:	PCI Express Base Specification 1.0a PCI Express Root Complex Topology Discovery ECN
SPONSOR:	Dong Wei & Joe Cowan; Hewlett-Packard Company

Part I

1. Summary of the Functional Changes

Make the Enhanced Configuration Access Mechanism required for PC-compatible platforms, but optional for platforms based on other processor/system architectures where firmware abstractions are provided for the configuration space access (e.g., DIG64 compliant systems).

Make the number of bits used for bus numbers (when mapping from memory address to Configuration Space) flexible. This removes the implied 256MB alignment and makes a system only needing to allocate address space needed for the number of busses it supports. This makes the currently defined enhance configuration mechanism more flexible/friendly as far as implementation is concerned.

Added the requirement and an implementation note (leveraged from PCI-X 2.0a) that the system hardware must provide a method for the system software to guarantee that a write transaction using the enhanced configuration access mechanism is completed by the completer before system software execution continues.

2. Benefits as a Result of the Changes

This change makes the approach of enhanced configuration access mechanism defined in PCI Express 1.0a consistent with that of the original (CF8/CFC) configuration access mechanism defined in PCI 2.3. In PCI 2.3, the CF8/CFC mechanism is required only on PC-compatible systems. Other systems, particularly those that do not rely on shrink-wrapped operating systems or systems that require a standard firmware-based SAL (e.g., DIG64-compliant systems) are not burdened with unnecessary hardware-level standards.

This change also provides the flexibility of the alignment of the Enhanced Configuration Address Space for the small PC-compatible systems.

This change makes it possible for the system software to guarantee the write transaction completion for the enhanced configuration accesses.

3. Assessment of the Impact

This is a minor addition to the existing revisions of the Specifications, comparable to the conventional configuration access mechanism statement in PCI 2.3.

There is no impact to systems that currently conform to the PCI Express specification.

It is also allowed for DIG64 compliant systems to implement the mechanism as describe in 7.2.2, it is up to the designers to choose, but the specification does not need to require such implementation because the OS access the enhanced configuration space through SAL calls.

4. Analysis of the Hardware Implications

There is no impact to existing hardware unless the system hardware does not provide a method for the system software to guarantee that a write transaction using the enhanced configuration access mechanism is completed by the completer before system software execution continues.

This allows non PC-compatible systems to be able to more freely leverage the existing chipsets or interoperate with existing chipsets.

The alignment flexibility provided by this change also provides small systems better flexibility in their design to make more memory available to the customers

5. Analysis of the Software Implications

A device driver should use the API provided by the operating system to access the Configuration Space of its device and not directly by way of the hardware mechanism, so there is no impact to the device drivers.

Operating system software on PC-compatible systems can still use direct hardware mechanism to access the Extended Configuration Space as is. Operating system software that utilizes the Root Complex Topology Discovery mechanism may be incompatible with new hardware that's configured to support less than eight bus number bits.

Operating system software on DIG64 compliant systems have been using firmware abstractions to access the conventional configuration space and will continue to do so for the Extended Configuration Space, so there is no need for a specified hardware access mechanism.

The alignment flexibility provided is fully covered by the ACPI interfaces defined in the PCI Firmware Specification 3.0 for the PC-compatible systems.

Part II

Detailed Description of the changes: PCI Express Base Specification 1.0a

Change Section 1.3, page 30 as follows:

1.3 PCI Express Fabric Topology

A fabric is composed of point-to-point Links that interconnect a set of components – an example fabric topology is shown in Figure 1 2. This figure illustrates a single fabric instance referred to as a hierarchy – composed of a Root Complex (RC), multiple Endpoints (I/O devices), a Switch, and a PCI Express-PCI Bridge, all interconnected via PCI Express Links. ~~Each of the components of the topology are mapped in a single flat address space and can be accessed using PCI-like load/store accesses transaction semantics.~~

Change Section 7.2.2, page 314 as follows:

7.2.2. PCI Express Enhanced Configuration Mechanism

For systems that are PC-compatible, or that do not implement a processor-architecture-specific firmware interface standard that allows access to the Configuration Space, the enhanced configuration access mechanism is required as defined in this section.

For systems that implement a processor-architecture-specific firmware interface standard that allows access to the Configuration Space, the operating system uses the standard firmware interface, and the hardware access mechanism defined in this section is not required. For example, for systems that are compliant with *Developer's Interface Guide for 64-bit Intel Architecture-based Servers (DIG64)*, Version 2.1 (DIG64 2.1)¹, the operating system uses the SAL firmware service to access the Configuration Space.

In all systems, device drivers are encouraged to use the application programming interface (API) provided by the operating system to access the Configuration Space of its device and not directly use the hardware mechanism.

The enhanced PCI Express configuration access mechanism utilizes a flat memory-mapped address space to access device configuration registers. In this case, the memory address determines the configuration register accessed and the memory data updates (for a write) or returns the contents of (for a read) the addressed register. The mapping from memory address A[27:0]space to PCI Express configuration space address is defined in Table 7-1. ~~The base address A[63:28] is allocated in an implementation-specific manner and reported by the system firmware to the operating system.~~

The size and base address for the range of memory addresses mapped to the Configuration Space are determined by the design of the host bridge and the firmware. They are reported by the firmware to the operating system in an implementation-specific manner. The size of the range is determined by the number of bits that the host bridge maps to the Bus Number field in the configuration address. In Table 7-1, this number of

¹ *Developer's Interface Guide for 64-bit Intel Architecture-based Servers (DIG64)*, Version 2.1, January 2002, www.dig64.org

bits is represented as n , where $1 \leq n \leq 8$. A host bridge that maps n memory address bits to the Bus Number field supports bus numbers from 0 to $2^n - 1$, inclusive, and the base address of the range is aligned to a $2^{(n+20)}$ byte memory address boundary. Any bits in the Bus Number field that are not mapped from Memory Address bits must be set to zero.

For example, if a system maps three Memory Address bits to the Bus Number field, the following are all true:

- $n = 3$.
- Address bits A[63:23] are used for the base address, which is aligned to a 2^{23} byte (8 MB) boundary.
- Address bits A[22:20] are mapped to bits [2:0] in the Bus Number field.
- Bits [7:3] in the Bus Number field are set to zero.
- The system is capable of addressing bus numbers between 0 and 7, inclusive.

A minimum of one Memory Address bit ($n=1$) must be mapped to the Bus Number field. Systems are encouraged to map additional Memory Address bits to the Bus Number field as needed to support a larger number of buses. Systems that support more than 4 GB of memory addresses are encouraged to map eight bits of Memory Address ($n=8$) to the Bus Number field. Note that in systems that include multiple host bridges with different ranges of bus numbers assigned to each host bridge, the highest bus number for the system is limited by the number of bits mapped by the host bridge to which the highest bus number is assigned. In such a system, the highest bus number assigned to a particular host bridge would be greater, in most cases, than the number of buses assigned to that host bridge. In other words, for each host bridge, the number of bits mapped to the Bus Number field, n , must be large enough that the highest bus number assigned to each particular bridge must be less than or equal to $2^n - 1$ for that bridge.

In some processor architectures, it is possible to generate memory space requests that cannot be expressed in a single Configuration Request, for example due to crossing a DW aligned boundary, or because a locked access is used. A Root Complex implementation is not required to support the translation to Configuration Requests of such memory space requests.

Table 7-1: Enhanced Configuration Address Mapping

Memory Address ²	PCI Express Configuration Space
A[27(20+n-1):20]	Bus Number ($1 \leq n \leq 8$)
A[19:15]	Device Number
A[14:12]	Function Number
A[11:8]	Extended Register Number
A[7:2]	Register Number
A[1:0]	Along with size of the access, used to generate Byte Enables

² This address refers to the byte-level address from a software point of view.

The system hardware must provide a method for the system software to guarantee that a write transaction using the enhanced configuration access mechanism is completed by the completer before system software execution continues.



IMPLEMENTATION NOTE

Ordering Considerations for the Enhanced Configuration Access Mechanism

The enhanced configuration access mechanism converts memory transactions from the host CPU into configuration transactions on the PCI Express fabric. This conversion potentially creates ordering problems for the software, because writes to memory addresses are typically posted transactions but writes to Configuration Space are not posted on the PCI Express fabric.

Generally, software does not know when a posted transaction is completed by the completer. In those cases in which the software must know that a posted transaction is completed by the completer, one technique commonly used by the software is to read the location that was just written. For systems that follow the PCI ordering rules throughout, the read transaction will not complete until the posted write is complete. However, since the PCI ordering rules allow non-posted write and read transactions to be reordered with respect to each other, the CPU must wait for a non-posted write to complete on the PCI Express fabric to be guaranteed that the transaction is completed by the completer.

As an example, software may wish to configure a device's Base Address register by writing to the device using the enhanced configuration access mechanism, and then read a location in the memory-mapped range described by this Base Address register. If the software were to issue the memory-mapped read before the enhanced-configuration-access-mechanism write was completed, it would be possible for the memory-mapped read to be re-ordered and arrive at the device before the configuration write, thus causing unpredictable results.

To avoid this problem, processor and host bridge implementations must ensure that a method exists for the software to determine when the write using the enhanced configuration access mechanism is completed by the completer.

This method may simply be that the processor itself recognizes a memory range dedicated for mapping enhanced configuration accesses as unique, and treats accesses to this range in the same manner that it would treat other accesses that generate non-posted writes on the PCI Express fabric, i.e., that the transaction is not posted from the processor's viewpoint. An alternative mechanism is for the host bridge (rather than the processor) to recognize the memory-mapped Configuration Space accesses and not to indicate to the processor that this write has been accepted until the non-posted configuration transaction has completed on the PCI Express fabric. A third alternative would be for the processor and host bridge to post the memory-mapped write to the enhanced configuration access mechanism and for the host bridge to provide a separate register that the software can read to determine when the configuration write has completed on the PCI Express fabric. Other alternatives are also possible.



IMPLEMENTATION NOTE

Generating Configuration Requests

Because Root Complex implementations are not required to support the generation of Configuration Requests from memory space accesses that cross DW boundaries, or that use locked semantics, software should take care not to cause the generation of such requests when using the memory-mapped configuration access mechanism unless it is known that the Root Complex implementation being used will support the translation.

7.2.2.1. Host Bridge Requirements

[For those systems that implement the enhanced configuration access mechanism, the](#)The PCI Express Host Bridge is required to translate the memory-mapped PCI Express configuration space accesses from the host processor to PCI Express configuration transactions. The use of Host Bridge PCI class code is reserved for backwards compatibility; host Bridge configuration space is opaque to standard PCI Express software and may be implemented in an implementation specific manner that is compatible with PCI Host Bridge Type 0 configuration space.

7.2.2.2. PCI Express Device Requirements

Devices must support an additional 4 bits for decoding configuration register access, i.e., they must decode the Extended Register Address[3:0] field of the Configuration Request header.

Detailed Description of the changes: PCI Express Root Complex Topology Discovery ECN

Change Section 7.13.3.2.2, page 10 as follows:

7.13.3.2.2 Link Address for Link Type 1

~~For a link pointing to the configuration space of a Root Complex element (Link Type = 1), Bits 27:12 of the first DWORD specify the bus, device and function number of the configuration space of the target element as shown in the table below; bits 11:0 are reserved and hardwired to 0.~~ For a link pointing to the configuration space of a Root Complex element (Link Type = 1), bits in the first DWORD specify the bus, device, and function number of the target element. As shown in Figure 7-62, bits 2:0 (N) encode the number of bits n associated with the bus number, with $N=000b$ specifying $n=8$ and all other encodings specifying $n=<value\ of\ N>$. Bits 11:3 are reserved and hardwired to 0. Bits 14:12 specify the function number, and bits 19:15 specify the device number. Bits $(19+n):20$ specify the bus number, with $1 \leq n \leq 8$.

Bits 31: ~~$(20+n)$~~ ²⁸ of the first DWORD together with the second DWORD optionally identify the target element's hierarchy ~~(for multi-hierarchy systems~~ implementing the PCI Express Enhanced Configuration Access Mechanism) by specifying bits 63: ~~$(20+n)$~~ ²⁸ of the memory-mapped configuration space base address of the PCI Express hierarchy associated with the targeted element; single hierarchy systems that do not implement more than one memory mapped configuration space are allowed to report a value of 0 to indicate default configuration space.

A configuration space base address [63: ~~$(20+n)$~~ ²⁸] equal to zero indicates that the configuration space address defined by bits [~~$(19+n)$~~ ²⁷:12] (bus, device number, and function number) exists in the default configuration space segment; any non-zero value indicates a separate configuration space base address.

Software must not use n outside the context of evaluating the bus number and memory-mapped configuration space base address for this specific target element. In particular, n does not necessarily indicate the maximum bus number supported by the associated configuration space segment.

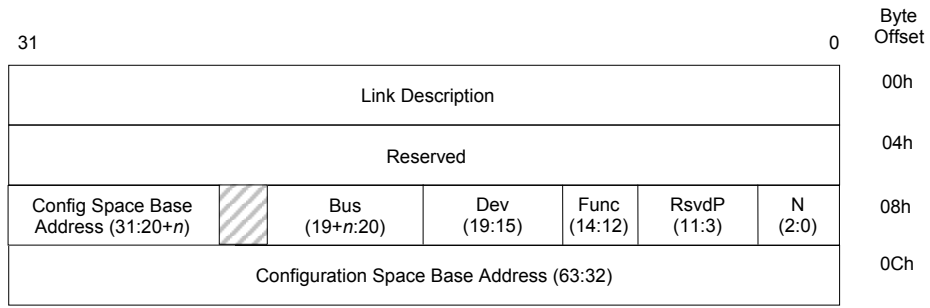
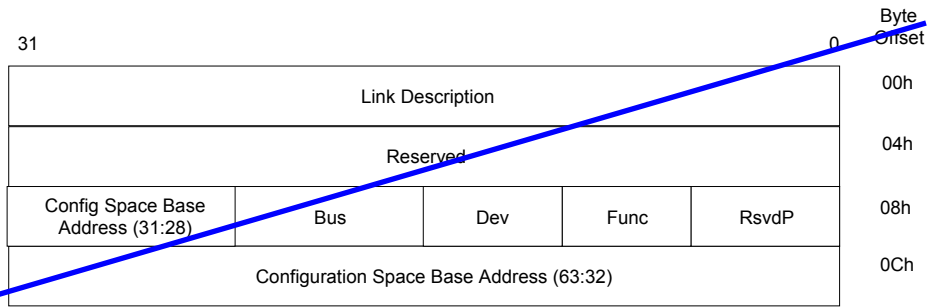


Figure 7-62 Link Address for Link Type 1

Table 7-53 Link Address for Link Type 1

Bit Location	Description	Register Attribute
2:0	N – encoded number of Bus Number bits	Hwlnit
14:12	Function Number	Hwlnit
19:15	Device Number	Hwlnit
(19+n)27:20	Bus Number $(1 \leq n \leq 8)$	Hwlnit
63: (20+n)28	PCI Express Configuration Space Base Address $(1 \leq n \leq 8)$ Note: A Root Complex that does not implement multiple configuration spaces is allowed to report this field as 0.	Hwlnit