1-800-633-1440
www.mindshare.com
training@mindshare.com

Overview of Changes to PCI Express 2.1 and 3.0

By Mike Jackson, Senior Staff Architect, MindShare, Inc.

The PCISIG has released the 2.1 specification whereas the 3.0 specification release has been delayed to a later date. As one might expect, achieving better performance was a driving goal of the 3.0 revision, but why is that speed needed? One reason is to support accelerator devices that are designed to handle unique algorithms at high speed for specific applications. Graphics processors and other hardware accelerators needed a higher-bandwidth and better-coupled link to the system processor cores and that's part of the motivation for the changes with Gen3.

Several other changes are being added to address previous shortcomings in the design or add new functionality for which the need has become evident since the last revision. These changes have been finalized, but rather than wait for release of a 3.0 specification revision, the PCISIG decided to incorporate these changes into a 2.1 revision specification. With that in mind, we'll list the major changes here, starting with those changes associated with the 2.1 specification revision and then those related to the 3.0 specification revision. This paper is not intended to be a comprehensive list of all the changes, and won't cover much in the way of implementation details, but does still serve as a good introduction.

Overview of Changes to PCI Express 2.1 Above and Beyond the PCI Express 2.0 Specification

1. **Internal Error Reporting** – Goal: make internal errors visible to software. Two new internal errors are defined: corrected or uncorrectable. These are errors that aren't associated with link activity or packets, and their actual definition will be implementation specific.
   Corrected internal errors have been masked or worked around by the component without loss of information or improper operation. An example would be a buffer that suffers an error that is corrected by ECC.
   Uncorrectable internal errors result in improper operation and an example might be a data error that cannot be corrected by ECC. The only recovery for this is a reset or hardware replacement.
2. **Multicast** – Goal: add broadcast capability for memory areas. A new capability register set that defines a base address and index to create a series of memory areas that are each associated with a different Multicast Group (collection of ports and endpoints). Every root port or switch port or endpoint function that supports this would need to implement the new registers. A multicast hit to a root or switch port is forwarded to all the ports that match the group, except the ingress port (to avoid loops). If no downstream ports match and the upstream port is enabled, the multicast can go upstream. If there are no matches at all, the TLP is silently dropped.
   An endpoint that doesn't support multicast can still receive them if the downstream port above it uses an Overlay register to map a multicast address onto an ordinary address the endpoint is already using. This can also go the other way; a switch upstream port can overlay a multicast memory onto an ordinary host memory range (though the spec says this would not usually be done). For the overlap case, the upper address bits of the multicast request are replaced with the appropriate normal address. This will likely cause an ECRC violation if it's being used, so dropping the ECRC or regenerating it are optionally supported to handle this case.
   Congestion Avoidance – Since Multicast can add more traffic to the network there is a potential for congestion to spread. To avoid this, components that serve as MC targets should be designed to consume MC TLPs "at wire speed." Similarly, MC sources should include a rate limiting mechanism. The system designer must choose the right mechanisms and components to serve the application.
3. **Atomic Operations** – Goal: Support SMP-type operations across a PCIe network to allow for things like offloading tasks between CPU cores and accelerators like a GPU. The spec says this enables advanced synchronization mechanisms that are particularly useful with multiple producers or consumers that need to be synchronized in a non-blocking fashion. Three new atomic non-posted requests were added, plus the corresponding completion (the address must be naturally aligned with the operand size or the TLP is malformed):
   a. Fetch and Add – uses one operand as the "add" value. Reads the target location, adds the operand, and then writes the result back to the original location.

b. Unconditional Swap – uses one operand as the “swap” value. Reads the target location and then writes the swap value to it.

c. Compare and Swap – uses 2 operands: first data is compare value, second is swap value. Reads the target location, checks it against the compare value and, if equal, writes the swap value to the target location.

d. AtomicOp Completion – new completion to give the results of an atomic request and indicate that the atomicity of the transaction has been maintained.

Since AtomicOps are not locked they don’t have the performance downsides of the PCI locked protocol. Compared to locked cycles, they provide “lower latency, higher scalability, advanced synchronization algorithms, and dramatically lower impact on other PCIe traffic.” The lock mechanism can still be used across a bridge to PCI or PCI-X to achieve the desired operation.

AtomicOps can go from device to device, device to host, or host to device. Each completer indicates whether it supports this capability and guarantees atomic access if it does. The ability to route AtomicOps is also indicated in the registers for a given port.

4. **Resizable BAR Capability** – Goal: allow the system to select how much system resource is allocated to a device. This new optional set of registers allows functions to communicate their resources size options to system software, which can then select the optimal size and communicate that back to the function. Ideally, the software would use the largest setting reported, since that would give the best performance, but it might choose a smaller size to accommodate constrained resources. Currently, sizes from 1MB to 512GB are possible. If these registers are implemented, there is one capability register to report the possible sizes, and one control register to select the desired size for each BAR. Note that devices might report a smaller size by default to help them be compatible in many systems, but using the smaller size would also reduce its performance.

5. **Dynamic Power Allocation** – Goal: provide more software-controlled power states to improve power management (PM). Some endpoint devices don’t have a device-specific driver to manage their power efficiently, and DPA provides a means to fill that gap. DPA only applies when the device is in the D0 state, and it defines up to 32 substates. Substate0 (default) defines the max power, and successive sub-states have a power allocation equal to or lower than the previous one. Software is permitted to change the sub-states in any order.
The Substate Control Enabled bit can be used to disable this capability. Any time the device is changing between substates, it must always report the highest power requirement of the two until the transition has been completed, and the time needed to make the change is implementation specific.
To allow software to set up PM policies, functions define two transition latency values and every substate associates its max transition time (longest time it takes to enter that substate from any other substate) with one of those.

6. **ID-based Ordering** – Goal: improve performance by avoiding stalls caused by ordering rules. For example, posted writes are never normally allowed to pass each other in a queue, but if they are requested by different functions, we can have some confidence that the requests are not dependent on each other. The previously reserved Attribute bit [2] is now combined with the RO bit to indicate ID ordering with or without relaxed ordering. This only has meaning for memory requests, and is reserved for Configuration or IO requests. Completers are not required to copy this bit into a completion, and only use the bit if their enable bit is set for this operation.

7. **Latency Tolerance Reporting** – Goal: improve system allocation of time and resources based on performance requirements of an endpoint. This allows a device to send a message reporting its tolerance of memory read/write latencies, allowing PM policies for central resources to be designed with that in mind. The message must use TC0 and has no data but reports two values in the header, No-snoop latency and Snoop latency, each of which has 10 bits plus a 3-bit multiplier field, allowing a range from 1ns to 32ms. Each of the two fields also has a “requirement” bit to indicate whether this timing is required. Setting the value and scale to zero indicates the device will be affected by any delay at all and so the best possible service is requested. It’s recommended that a device send this message each time the service requirements change, making sure to allow enough time for the system to respond to it. The RC is strongly encouraged,

but not required, to meet the worst case timing. Software must not enable LTR in an endpoint unless the root and all intermediate switches also indicate support. Switches collect LTR messages from downstream and send a "conglomerated" message upstream (if enabled on the upstream port), adding their own latency to reduce the value reported upstream by as much as 20%.

8. **Alternative Routing-ID Interpretation** – Goal: support a much larger number of functions inside devices. For requesters and completers, this means treating the device number value as though it was really just an extension of the function field to give an 8-bit value for the function number. Since the device number is no longer included, it's always assumed to be 0.
   The spec also defines a new set of optional registers that can be used to assign a function group number to each function. Within an ARI device, several functions can be associated with a single group number, and that can serve as the basis for arbitration or access control.

9. **Extended Tag Enable Default** – Goal: allow vendors to define the default in the way that will be best for them. The default used to be 0b, meaning this was disabled, but now the default is implementation specific.

10. **TLP Processing Hints** – Goal: improve memory latency by associating a packet with a given processor cache (perhaps this is a little like processor affinity). This technique adds an optional header field to allow packets to include information regarding the communication model between Endpoints and the RC.
    A new TH bit in the header indicates that hints are present. The hint is contained in the last 2 bits of the address field that used to be reserved.
    a. The 2 hint bits indicate frequent read/write access to data:
        i. 00 – Bidirectional: by both host and device
        ii. 01 – Requester: by device
        iii. 10 – Target (Completer): by host
        iv. 11 – Target with priority: by host with high temporal locality (using the local cache as much as possible)
    b. Steering Tag (ST) bits are system-specific values that indicate a processing resource is being explicitly targeted by a Requester.
        i. For posted writes, the 8-bit tag field is repurposed for the ST[7:0] field. This may affect completer processing of the request.
        ii. For memory reads, the byte enable fields are repurposed to carry the ST[7:0] field. Now the BEs are implicitly 4 bytes on a single-dword read, and all 8 bytes on a read longer than 1 dword.
        iii. Alternatively, the ST bits can be place in an ST Table in the TPH capability structure or combined with the MSI-X table. The location is discoverable by software and each entry is two bytes. How the device associates a given request with an ST Table entry is design specific. To avoid confusion, it's recommended that software either disable TPH or quiesce the function while making ST Table updates.

iv. ST Modes (only 3 defined):
1. 000 – No ST mode. ST value must be all zeroes.
2. 001 – Interrupt vector mode. Each ST value is selected by an MSI/MSI-X interrupt vector number and the table is required.
3. 010 – Device specific. Recommended to get ST value from a table, but not required.

11. **TLP Prefix** – Goal: provide more packet information for routing or processing hints. The previously-reserved upper Format bit now identifies TLPs that use a prefix. If this prefix bit is set (Format = 100b), the Type field bit 4 indicates whether the prefix has meaning locally or end-to-end. The size of a prefix is always 1DW, but there can be multiple prefixes and a mix of types is permitted.

    A Local TLP Prefix has only one value defined at present: MR-IOV. The MR-IOV spec states that this prefix is used to identify virtual link and virtual hierarchy information to facilitate things like routing and congestion management. When both local and end-to-end prefixes are present, the local one takes precedent.

    An End-to-end Prefix has three options: Extended TPH, and Vendor Defined version 0 and 1. Only 4 end-to-end prefixes are allowed per TLP.

    Receivers look at the first byte of incoming TLPs to see whether a prefix is present and, if so, go through all the prefixes until the header is found (it's required that there be a header; a packet can't consist of just prefixes).

    Flow control must take this into account because one header credit will now be the header, ECRC, and the max number of allowed prefixes. A TLP Prefix log is also added to the AER registers to store a copy of the prefixes if an error occurs, much like the header log that's already there.

12. **System-board eye height** – Goal: decrease minimum receiver voltage from 300 to 250mV to make it easier for systems to meet the requirements in a typical add-in card environment.
13. **Transaction Ordering Table** – Goal: simplify the table by reducing the number of entries, making it easier for new devices to be compliant.

Overview of Changes to PCI Express 3.0 Above and Beyond the PCI Express 2.1 Specification

1. **Higher Speed** – Goal: improve performance. Each successive generation doubles the bit rate of the previous generation, and that holds true for Gen3, too, but there's a significant difference this time. Since the previous speed was 5.0 GT/s, the new speed would normally have been 10.0 GT/s, but the spec writers considered a signal that used a 10GHz clock problematic because of the board design and signal integrity issues that many vendors would face. Constrained to stay under that frequency, they were forced to consider other options. The solution they chose was to move away from the 8b/10b encoding scheme that PCIe and most other serial transports have used because it adds a 20% overhead from the receiver's perspective. Instead, they chose a modified scrambling method that effectively creates a 128/130 encoding method. This gain in efficiency meant that an increase in frequency to only 8.0GHz would be enough to achieve a doubling of the bandwidth and meet this goal

2. **Optimized Buffer Flush/Fill** – Goal: improve system PM by informing endpoints of system power states, allowing them to know the best times for bus mastering or interrupts because the system is already active and thus the PM cost will be low.

    a. The information can be sent using a message, or by sending a pattern on the WAKE pin of the device. Five different transitions between the 3 states can be indicated by patterns on the WAKE signal (technically, there are 6 possible transitions, but the sequence to Idle from either of the other two is the same). Figure 6-19 in the spec shows that both methods can be mixed in the same topology but *using the pin is preferred because it takes less power*, and the links don't have to wake up from a low-power state for the functions to get the information. Consequently, software is strongly encouraged to use the message form only when the pin is not available.

b. How the system state is determined will be design specific, but the spec currently defines only 3 states (if received, a reserved code should be treated as an "active" message). If the system goes to Active or OBFF, it's highly recommended that it not go back to Idle for at least 20us. The states are:
   i. 1111 – System is fully active for all actions, including bus mastering and interrupts. (Active)
   ii. 0001 – Path to system memory is active for device memory read/write bus master activity. (OBFF)
   iii. 0000 – System is in a low-power state. (Idle).

For training on this subject or to purchase a self-paced eLearning module on this subject, please visit: www.mindshare.com or email training@mindshare.com or call 800.633.1440