PCI-Express

Transaction Ordering and Address Translation for Peer to Peer

Haggai Eran, May 2017
PCI-Express peer to peer

Direct communication between devices

Raises some complex issues:

- How to order transactions correctly?
- How to isolate between transactions of different applications / virtual machines?
Ordering
PCI-Express Data Link Layer

- Data Link Layer adds LCRC, sequence number to each packet for reliability.
- Data Link Layer Packets (DLLPs) provide Ack/Nak to TLPs.
- Each TLP is kept in buffer until Acked, and retransmitted upon getting a Nak.

CRC - Cyclic Redundancy Check - an error correcting code
Replay mechanism

- Similar to TCP
  - Sequence numbers
  - CRC - Checksum
- TLP Rx:
  - Check CRC
  - Check sequence
  - Send ack/nak
- DLLP Rx:
  - Retransmit or
  - Release buffer

(PCI Express Technology 3.0, Mike Jackson and Ravi Budruk)
Flow Control

- For transmission to work, receiver needs to have buffer space available
- The alternative --- retransmit when there is no space --- wastes time and energy
- Credit-based mechanism
  - The receiver reports amount of buffers to the sender
  - Credit types:
    - Headers / Data
    - Writes / Read requests / Completions
  - Credit updates: DLLPs with current allocated credit counts
- When there are not enough credits TLPs are blocked!
Transaction Ordering

Why these rules?

- Legacy compatibility (PCI bus)
- Deterministic completion
  - Prevent deadlocks
- Preserve the programmer’s intentions
  - Producer consumer example ahead
- Maximize performance
Quality of Service

PCI-express provides different traffic classes (TCs) mapped to virtual channels (VCs).

Intended to prioritize traffic.

A virtual channel has its own:

- Buffers
- Flow control credits

No ordering between TCs.

Controlled by the device driver
Producer/Consumer example

**NIC:**
- write data to GPU
- write flag to memory

**GPU:**
- do { read flag from memory
  } while (!flag);
- access received data
Ordering correctness

Some rules to preserve correctness

● Writes cannot pass writes
  ○ NIC data has to pass before the flag
● Read completions cannot pass writes
  ○ Switch has to pass NIC data writes to GPU before the flag read completion from memory
Why not strong ordering?

- For example:
  - Devices A and B send multiple read requests to each other
  - At some point, buffers are filled and the read requests are blocked
  - As a result of the reads, both devices send completions back
  - If completions are ordered after the read requests, they are blocked as well
  - Deadlock!
- Completions must be allowed to pass earlier blocked reads
Ordering and performance

To improve link utilization, reduce stalls, allow some reordering. (Similar to weak memory consistency)

Reads are allowed to pass other reads and completions.

Optional extensions:

- Relaxed ordering: a bit on TLP to waver some ordering rules
- ID based ordering: do not require ordering between streams of different requestor ID
  - Different device functions
  - Can break peer to peer!
### Simplified* Ordering Rules Table

The table shows whether transaction in the rows can pass the transaction in the column.

*Real table in the specs (Ignores IDO, RO)

<table>
<thead>
<tr>
<th>Row pass Column?</th>
<th>Writes</th>
<th>Reads</th>
<th>Completions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Writes</td>
<td>No</td>
<td>Must</td>
<td>Allowed</td>
</tr>
<tr>
<td>Read</td>
<td>No</td>
<td>Allowed</td>
<td>Allowed</td>
</tr>
<tr>
<td>Completions</td>
<td>No</td>
<td>Must</td>
<td>Allowed</td>
</tr>
</tbody>
</table>
### Simplified* Ordering Rules Table

The table shows whether transaction in the rows can pass the transaction in the column.

*R: Real table in the specs (Ignores IDO, RO)

**Posted**: writes and messages

**Non-posted**: read requests, configuration read and writes

<table>
<thead>
<tr>
<th>Row pass Column?</th>
<th>Posted</th>
<th>Non-posted</th>
<th>Completions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Posted</strong></td>
<td>No</td>
<td>Must</td>
<td>Allowed</td>
</tr>
<tr>
<td><strong>Non-posted</strong></td>
<td>No</td>
<td>Allowed</td>
<td>Allowed</td>
</tr>
<tr>
<td><strong>Completions</strong></td>
<td>No</td>
<td>Must</td>
<td>Allowed</td>
</tr>
</tbody>
</table>
Address Translation
User-space I/O

- Operating Systems allow multiple applications to share a single device
- All applications go through a common driver
- Improve performance by allowing direct device access
- But:
  - Applications use virtual memory addresses
  - Memory uses physical addresses
  - Need to protection between different processes
- Also applies to virtual machines
Addresses Taxonomy

On the CPU

● Process uses a (guest) *virtual* address
● OS translates into a (guest) *physical* address
● (Optional) Hypervisor translates into a host physical address

For devices:

● GPU application uses a (guest) *virtual* address
● GPU’s MMU translates into a *bus* address (guest physical)
● (Optional) IOMMU translates into *machine* address

Note: Latest specs allow IOMMU to handle multiple processes
IOMMU role

- Protect against accesses on behalf of other processes
- Translate addresses to have a consistent memory map
- Commonly used today for virtual machines
- Perhaps can also replace the GPU or NIC MMU units in the future.

(Intel® Virtualization Technology for Directed I/O Architecture Specification)

(Wikipedia)
Access Control Services (ACS)

- Switches do not know the translation
  - They could forward incorrectly!
- PCI-Express access control services (ACS) allow fine grained control over forwarding, such as:
  - Block peer to peer transactions
  - Redirect to IOMMU

Note: not every system has a switch
Address Translation Services (ATS)

- IOMMU translation can increase latency
  - IOTLB misses require page table walks
- Address Translation Cache (ATC)
  - Similar to per-core TLB
  - Custom eviction policy
    - Device has better information about its access patterns
- Device sends translation request, receives translation completion
- Device then uses translated addresses
- When CPU changes translation, send invalidation request and wait for invalidation completion
  - Like TLB software coherency

Table 2-5: Address Type (AT) Field Encodings

<table>
<thead>
<tr>
<th>AT Coding (b)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Default/Untranslated</td>
</tr>
<tr>
<td>01</td>
<td>Translation Request</td>
</tr>
<tr>
<td>10</td>
<td>Translated</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
## Peer to peer and address translation

<table>
<thead>
<tr>
<th></th>
<th>Performance</th>
<th>Protection</th>
<th>Trust in the device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable address translation</td>
<td>✔</td>
<td>✗</td>
<td>High</td>
</tr>
<tr>
<td>Redirect to IOMMU for untranslated and block translated</td>
<td>✗</td>
<td>✔</td>
<td>Little</td>
</tr>
<tr>
<td>Direct translated P2P</td>
<td>✔</td>
<td>✔</td>
<td>Medium (just ATC)</td>
</tr>
</tbody>
</table>
Single-root I/O Virtualization (SR-IOV)

Assigning a complete device to a VM prevents sharing.

Hardware virtualization: The device presents itself as multiple functions.

Virtual functions (VFs)

- Lightweight PCI-express functions
- Controlled by their physical function (PF)
  - One BAR on the physical functions configures MMIO for all VFs
  - Minimal VF configuration space
- VFs can be enabled/disabled after enumeration

VFs transactions can use a different IOMMU translation table
Summary

PCI-Express ordering

- Flow control
- Transaction ordering
- Deadlock avoidance

Address translation

- IOMMU
- Access Control
- Address Translation Services

Peer to peer presents some interesting challenges

For more information:

- *PCI Express Base Specifications* ([CD-ROM in the library](#))
- *PCI Express Technology 3.0*, Mike Jackson and Ravi Budruk.