IOV-Enabling Protocol Changes

Steve Glaser

Fellow

AMD

Disclaimer

- NOTE: The information in this presentation refers to a specification still in the development process. This presentation reflects the current thinking of the workgroups, but all material is subject to change before the specification is finally released.

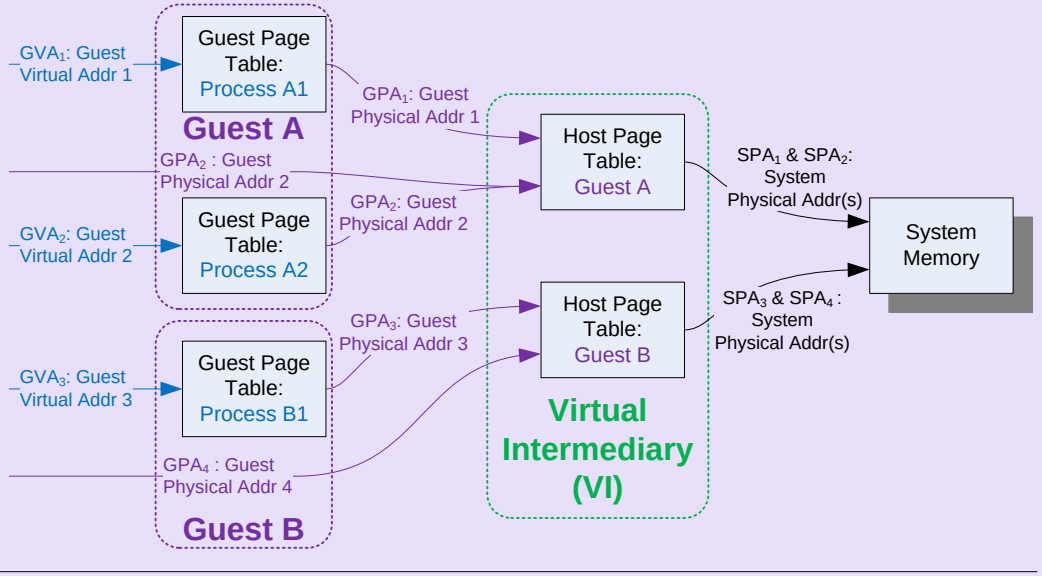
IOV Background

- Address Translation Services (ATS) supports:
  - ✓ Direct assignment of a Function to a Guest OS running on a Virtual Intermediary (Hypervisor)
- Page Request Interface (PRI) supports:
  - ✓ Functions that can raise a Page Fault
- Single Root-I/O Virtualization (SR-IOV) supports:
  - ✓ Light Weight Functions (Virtual Functions)
  - ✓ Large numbers of Functions (multiple Bus Numbers)

PASID Overview

- Support Direct Assignment of I/O to a User Process running on a Guest OS running on a Virtual Intermediary
  - ✓ Untranslated Memory Requests
  - ✓ Translation Requests
  - ✓ Translation Invalidations
  - ✓ Page Requests
- Support Execute Permission
- Support Privileged Mode

PASID Address Mapping



Address Translation Cache (either in TA *or* in Function's ATC)

$GVA_1/A1 \rightarrow\rightarrow SPA_1$
$GVA_2/A2 \rightarrow\rightarrow SPA_2$
$GPA_2 \rightarrow SPA_2$
$GVA_3/B1 \rightarrow\rightarrow SPA_3$
$GPA_4 \rightarrow SPA_4$

Cache Entry Type	*Meaning*
GVA/PASID →→ SPA	*TA / ATC Entry with PASID*
GPA → SPA	*TA / ATC Entry without PASID*

- 3 User Processes
- 2 Guests
- Cache Example:
  - ✓ 3 GVA Entries
    - GVA_1 / A1 →→ SPA_1
    - GVA_2 / A2 →→ SPA_2
    - GVA_3 / B1 →→ SPA_3
    - Intermediate GPA not cached
  - ✓ 2 GPA Entries
    - $GPA_2 \rightarrow SPA_2$
    - $GPA_4 \rightarrow SPA_4$
- Directly Assigned to User Process
  - ✓ GVA_1 / GVA_2 / GVA_3
- Directly Assigned to Guest
  - ✓ GPA_2 / GPA_4
- … → SPA_2 cached twice
  - ✓ GVA_2 / A2 →→ SPA_2
  - ✓ $GPA_2 \rightarrow SPA_2$

PASID is TWO ECNs

- Process Address Space ID ECN
  (PCI Express Base 3.0)
  - ✓ PASID TLP Prefix
  - ✓ Usage on Untranslated Memory Requests
  - ✓ PASID Capability
- PASID Translation ECN
  (Address Translation Services 1.1)
  - ✓ Usage on ATS Requests
  - ✓ Usage on ATS Invalidation Requests
  - ✓ Usage on PRI Requests
  - ✓ Usage on PRG Responses
  - ✓ ATS Invalidation rules

PASID TLP Prefix

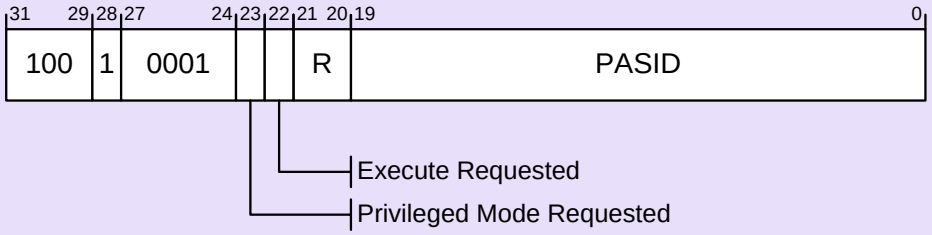
- Permitted on:
  - ✓ Untranslated Memory Request
    - Including Untranslated AtomicOP Requests
  - ✓ ATS Translation Request
  - ✓ ATS Invalidation Request
  - ✓ Page Request Interface Request (PRI)
  - ✓ Page Request Group Response (PRG)

Base ECN

ATS ECN

- PASID does not require ATS support
  - ✓ Functions can use only Untranslated Memory Requests
  - ✓ Without ATS support, pages must be pinned
- PASID does not require PRI support
  - ✓ PRI permits paging and late pinning
    - User Process/Guest OS level
      or
    - Guest OS/Hypervisor level
  - ✓ Without PRI support, pages must be pinned

PASID TLP Prefix



- **Process Address Space ID (PASID)**
  - ✓ Indicates Process Address Space of the request or response
  - ✓ Requests to Host:
    - Use PASID to determine SPA, if PASID not valid, treat as access to unmapped memory
  - ✓ Requests to Function:
    - On ATS Invalidate, return success if PASID not in translation cache
    - On PRG Response, optionally use PASID to match response to PRI Request
  - ✓ Effective PASID width is smaller of TA and ATC supported widths
- **Execute Requested**
  - ✓ If Set, Function wants to know whether memory contains executable code
  - ✓ If Clear, Function will not execute code in this memory and doesn't care about execute permission
  - ✓ Reserved on Requests to Function
    - ATS Invalidation, PRG Response
  - ✓ Reserved on Untranslated Memory Write
- **Privileged Mode Requested**
  - ✓ If Set, Function wants to know permissions available to privileged mode
  - ✓ If Clear, Function wants to know permissions available to non-privileged mode
  - ✓ Reserved on Requests to Function
    - ATS Invalidation, PRG Response

Privileged Mode

- Functions can operate in Privileged and Non-Privileged modes
- Each mode is distinct
  - ✓ Read, Write and Execute permissions
  - ✓ No Write (NW) request bit
  - ✓ Modes are decoupled in ATC
  - ✓ Modes may be coupled in TA
    - Based on system architecture
      - For example, to model x86 CR0.WP behavior

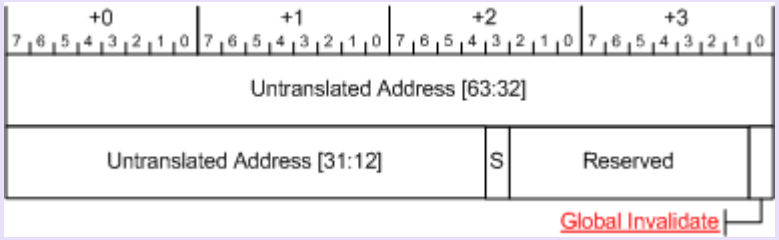
Execute

- "Full" Execute Permission is highly nuanced
  - ✓ Track different permissions for each Function & each Processor
- Hard to implement
  - ✓ Lots of bits (hardware cost)
  - ✓ Intrusive (software cost)
  - ✓ PTE changes unpopular (political cost)
  - ✓ Hard to share Host and I/O Page Tables (additional constraints)
- Various potential TA implementations:
  - ✓ **Punt:** Always grant or deny for a Function
  - ✓ **Shadow Page Tables:** Don't share, each Function gets its own
  - ✓ **Shadow Execute Table:** Share main page table, non-shared Execute Permission table
  - ✓ *Want to permit a variety of schemes*
- Execute Requested means that the TA only has to look when the Function wants an answer
  - ✓ Skip Shadow Execute Table lookup for common cases
  - ✓ If Execute Requested is 1b, translation may grant Execute Permission
  - ✓ If Execute Requested is 0b, translation may not grant Execute Permission
  - ✓ Execute Requested may not affect other bits in translation (addr, size, N, U, R, W)

Global Pages

- X86 Page Table Entries have a Global Page bit (G)
  - ✓ If 1b, translation applies across all address spaces
  - ✓ Permits better cache utilization, higher performance
  - ✓ Single x86 TLB flush affects Global translation in all address spaces
- Similar mechanism exists in other architectures
- PASID Translation ECN:
  - ✓ Add Global bit to Translation Completion Data Entry
    - Indicates mapping applies to all PASIDs
    - ATC can ignore Global bit
      - Performance issue, not a correctness issue
  - ✓ Add Global Invalidate bit to Invalidation Request
    - If 1b, invalidates global and non-global mappings
    - If 0b, invalidates only non-global mappings
    - Global Invalidate Supported bit added to ATC Capability

ATS Invalidation Request: Global Invalidate

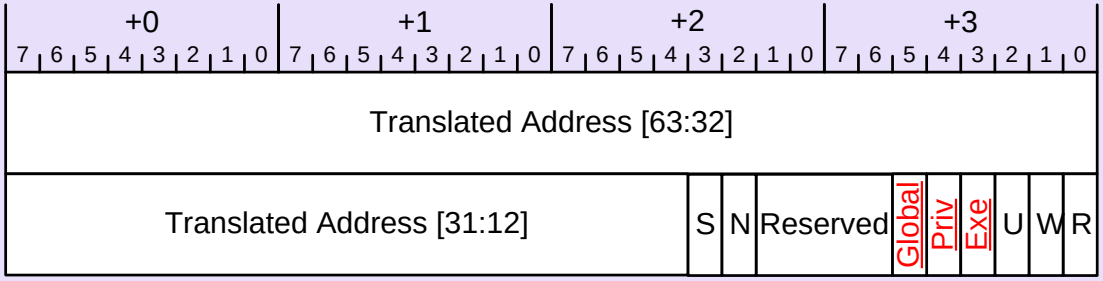


- New Global Invalidate bit in message body
- If 1b, Invalidation Request affects all PASID values
  - ✓ PASID value in PASID TLP Prefix is ignored
  - ✓ Bit Reserved if no PASID TLP Prefix present
  - ✓ Bit ignored if Global Invalidate Supported bit is 0b

PASID Invalidation Rules

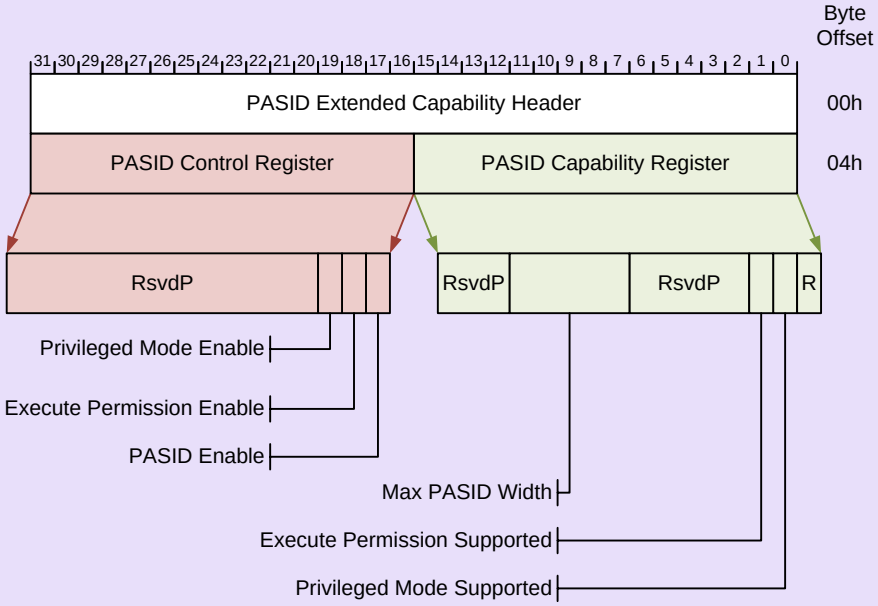
- Invalidation Requests with a PASID TLP Prefix:
  - ✓ Return success if PASID Enable is 0b
  - ✓ If Global Invalidate Supported:
    - If Global Invalidate 1b, invalidate in all PASIDs
    - If Global Invalidate 0b, invalidate in specific PASID
      - Global Mappings may not be invalidated
  - ✓ If Global Invalidate not Supported:
    - Invalidate in specific PASID
  - ✓ No effect on ATC entries that have no PASID
- Invalidation Requests with no PASID TLP Prefix:
  - ✓ Invalidate all ATC entries that have a PASID
  - ✓ Invalidate requested ATC entries that have no PASID
- When stopping a PASID
  - ✓ Implicitly Invalidate non-Global ATC entries for that PASID

Translation Completion Data Entry



- Execute Permitted (Exe)
  - ✓ If 1b, Function is permitted to “execute code” on pages
    - Only 1b if Execute Requested was also 1b
  - ✓ If 0b and Execute Requested was 1b, Function is not permitted to “execute code” on pages
  - ✓ Definition of “execute code” is out of scope
- Privileged Mode (Priv)
  - ✓ If 1b, permission fields (R, W, Exe) refer to “Privileged context”
    - Only 1b if Privileged Mode Requested was also 1b
  - ✓ If 0b, permission fields (R, W, Exe) refer to “Non-Privileged context”
    - Can be 0b even if Privileged Mode Requested was 1b (e.g. no TA support)
  - ✓ Definition of “Privileged/Non-Privileged context” is out of scope
- Global Page (Global)
  - ✓ If 1b, translation applies to all PASIDs
  - ✓ If 0b, translation applies to PASID from translation request
  - ✓ ATC can ignore this bit

PASID Extended Capability



PASID Control Register

- Privileged Mode Enable
  - ✓ Permitted to Set Privileged Mode Requested
- Execute Enable
  - ✓ Permitted to Set Execute Requested
- PASID Enable
  - ✓ Permitted to send PASID TLP Prefix and cache PASID translations
  - ✓ Clearing PASID Enable invalidates all PASID translations

PASID Capability Register

- Privileged Mode Supported
  - ✓ Supports setting Privileged Mode Requested
- Execute Permission Supported
  - ✓ Supports setting Execute Requested
- Max PASID Width
  - ✓ $0 \le$ Max PASID Width ≤ 20
  - ✓ PASID values $[0 .. 2^{\text{Max PASID Width}} - 1]$

PASID Mapping to Processes

- PASID values are Per-Function
  - ✓ A Process ID maps to {PASID, Function}
  - ✓ PASID values in different Functions are unrelated
  - ✓ A process that is directly assigned to two Functions can have different PASID values in each Function
- Function Context → PASID is Device Specific
  - ✓ Map is dynamic
    - At the Function as well as the TA
  - ✓ Maps are independent
    - e.g. Context A doesn't affect Context B
  - ✓ Function must support stopping use of a PASID
    - e.g. for Process Migration

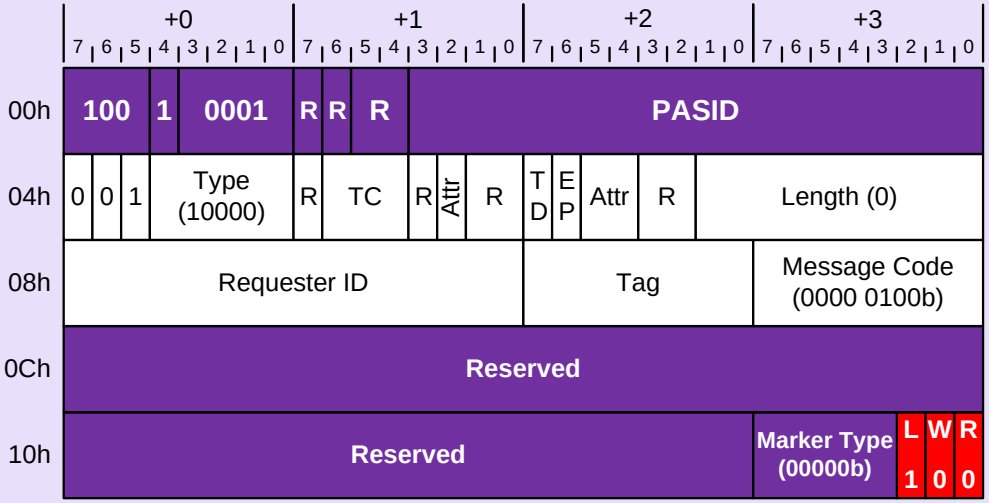
Stopping a PASID at Device

- Base rules:
  1. Stop queuing new Requests for this PASID
  2. Complete all Non-Posted Requests issued for this PASID.
  3. Flush to the host all Posted Requests addressing host memory in all TCs that were used by the PASID. The mechanism used for this is device specific
  4. Optionally flush all Peer-to-Peer Posted Requests to their destination(s). The mechanism used for this is device specific.
- ATS rules:
  5. Stop queuing new Translation Requests and Page Requests for PASID
  6. Finish transmitting multi-page Page Request Messages for PASID
  7. Wait for Translation Requests to complete

  Either:
  8. Wait for PRG Response Messages associated with PASID
  9. Indicate PASID has stopped and no Stop Marker will be generated

  or:
  10. Internally mark all outstanding and queued Page Requests for PASID as stale
      - Ignore resulting PRG Responses except to return Page Request Allocation & PRG Index
  11. Indicate PASID has stopped and Stop Marker will be generated
  12. Send Stop Marker Message when all stale Page Requests transmitted
      - Subsequent Page Requests are for a new use of the PASID value

Stop Marker Message



- PRI Request with PASID TLP Prefix
  - ✓ L, W, R == 1b 0b 0b
  - ✓ TC == 0, Relaxed Ordering == 0b
  - ✓ Doesn't consume Page Request allocation
- PRI Requests that precede Stop Marker are for **OLD PASID**
- PRI Requests that follow Stop Marker are for **NEW PASID**

Stopping a PASID at Host

- Host Software Algorithm:
  1. Wait if too many stop requests outstanding
     - Function says how many it supports
  2. Issue Stop Request
  3. Wait for Stop Request to complete
     - When PRI Requests arrive, generate error PRG Response
  4. Can now reassign PASID to new use
  5. If completion said expect Stop Marker:
     - PRI Requests before Stop Marker are stale, generate error PRG Response
     - PRI Requests after Stop Marker Message are new PASID, process normally

PASID on PRG Response

- PRG Index is 9 bits
  - ✓ Up to 512 outstanding Page Requests
- PRG Index optionally qualified by PASID
  - ✓ New Page Request Status register bit PRG Response PASID Required
  - ✓ If 1b, PRG Responses must have a PASID TLP Prefix
  - ✓ If 0b, PRG Responses may not have a PASID TLP Prefix
  - ✓ Permits more outstanding Page Requests

Use Cases

RDMA NIC Use Case

- Want to assign a Queue Pair (QP) to a Guest OS user process
  - ✓ Support Bits:
    - Execute Permission Supported is 0b
      - NIC does not execute code
    - Privileged Mode Supported is 0b
      - NIC does not have a protected mode (all transactions are Non-Privileged Mode)
    - Max PASID Width based on Max QP Count
      - Actually, max number of QPs per VF
  - ✓ Performance
    - If NIC uses ATS and PRI, no pinning of memory is required
    - No Hypervisor interaction required on QP open/close

GPU Use Case

- Support:
  - ✓ Execute Permission Supported is 1b:
    - GPU can fetch executable code from host memory
    - Execute Permission bit describes access on a page granularity
      - Such pages contain GPU code, not CPU code
    - Execute Permission bit could be different PTE bit(s) from the processor EXE bit(s)
  - ✓ Privileged Mode Support is 1b:
    - GPU can operate in either Privileged or Non-Privileged modes
    - OS Protection model is extended to the GPU
    - E.g. Only Privileged Mode code can access Privileged pages
    - E.g. Only Privileged Mode code can execute certain instructions
- Address Spaces:
  - ✓ If Supervisor and User share a single address space (e.g. Linux)
    - Privileged Mode bit defines protection
  - ✓ If Supervisor and User do not share address spaces (e.g. Solaris)
    - Use two distinct PASIDs
    - TA always grants Privileged access to the “Supervisor” PASID

Process Address Space

∞

	Privileged Mode	*Non-Privileged Mode*	*Global*
OS Per-Process Data	Read, Write	No Access	
OS Shared Data	Read, Write	No Access	Global
OS Code	Read, Execute	No Access	Global
Global Library Code	Read, Execute	Read, Execute	Global
Global Library RO Data	Read Only	Read Only	Global
Global Library RW Data	Read Only	Read, Write	
Coprocessor Code	Read Only	Read, Execute	
User RW Data	Read Only	Read, Write	
User RO Data	Read Only	Read Only	
User Code	Read Only	Read Only	

0

Thank you for attending the
PCI-SIG Developers Conference 2011.

For more information please go to
www.pcisig.com