Scaling Hafnium for Advanced DMA Architectures

Qualcomm Core Platform Team

Jack Suann

Agenda

- Requirements and Goals
- Proposals
- Questions

Requirements and Goals

- Support the use of scattered buffers for DMA.
  - Normal world OSes may provide highly fragmented buffers to the secure world.
  - DMA use-cases require these buffers to be virtually contiguous.
- Minimize performance overheads for DMA accesses.
  - Allow SPs to directly map memory to DMA devices and avoid the use of bounce buffers.
- Provide flexibility for SP use-cases.
  - Allow SPs to choose the most suitable DMA isolation model and address space configuration.
- Add support for all DMA isolation models in the FF-A specification.
  - Hafnium currently supports static DMA isolation – extend with dynamic and other (SEPID) models.
    - Static: boot time mappings defined via SP manifest, uses separate stage-2.
    - Dynamic: regions mapped and unmapped at runtime, stage-2 usually shared with SP.
    - Other: uses FF-A memory management to target SEPIDs with separate stage-2.

Overview of Proposals

SMMUv3 Driver Enhancements

- Shared stage-2 between SPs and device streams
- Two-level stream table
- Context descriptor support
- CMDQ invalidation
- Event handling

SEPID & Proxy SP Support

- Parsing of SEPIDs from SP manifest
- Device stream and SEPID bookkeeping
- Dynamic mapping support for IOMMU page tables
- SEPID support in FF-A memory management

SMMU Virtualization

- Hafnium managed stage-1 for SEPIDs
- SP controlled stage-1 for shared stage-2
- SP interfaces for stage-1 management:
  - HF paravirtual
  - Virtio-IOMMU
  - FF-A based

SMMUv3 Driver Enhancements

Shared stage-2 between SPs and device streams

- By default, device streams always have separate stage-2 tables to their owner SP.
  - Memory regions must be associated with device streams for mappings to be created (per static DMA isolation).
  - However, for some use-cases it may be preferred for the SP's stage-2 mappings to always be shared with these streams.
    - Alternative is to leave streams in bypass mode!
  - Sharing of the SP's stage-2 is also required for the dynamic DMA isolation model (with nested translation).
- Challenges:
  - Does FF-A specification allow for shared stage-2 without any further isolation between device & SP?
  - How to determine whether a stream should use a shared or separate stage-2?
    - Via a new property for device regions in the SP manifest?
- Status:
  - Prototype of changes complete.
- Impact:
  - Limited impact on SMMUv3 driver – just need to program STEs with SP's stage-2 rather than separate IOMMU PT.
  - Additional work required for parsing new properties in the SP manifest.

Two-level stream table support

- Add support for two-level stream tables in SMMUv3 driver.
  - Hafnium currently supports linear tables, but this does not scale for larger Stream ID sizes.
- Challenges:
  - SMMUv3 driver currently sets all streams to bypass mode if not configured otherwise. This requires pre-allocating all second level tables; when Stream ID bits are large this can exhaust the Hafnium heap.
  - When not pre-allocating levels, Hafnium needs to allocate second levels on demand (i.e., as streams are configured).
- Status:
  - Prototype of changes complete. Second level tables allocated as necessary.
  - Setting bypass as default translation is now a build time config option.
- Impact:
  - Some refactoring of SMMUv3 driver required, but no external impact.
  - No breakage of existing linear table support; linear tables still used up to a certain Stream ID bit threshold.

Context descriptor support

- Add support for context descriptors to enable stage-1 translation in the SMMU.
  - This enables nested translation and allows for fine-grained management of peripheral mappings.
  - Allows for Substream IDs, which enables separate stage-1 address spaces for each use-case.
  - Can be extended to support stage-1 only translation, however this is currently out of scope for our changes.
- Design:
  - There are two variants of stage-1 support to consider here: SP controlled stage-1 and Hafnium managed stage-1.
    - SP controlled stage-1 enables shared virtual addressing and fine-grained control of page tables.
    - Hafnium managed stage-1 allows for the SP to use virtual interface with map & unmap APIs, simplifying DMA management.
  - For SP controlled stage-1, the SMMUv3 driver just needs to program the STE to point to the given context descriptor(s).
  - For Hafnium managed stage-1, the SMMUv3 driver must allocate and program the context descriptor itself.
    - For Substream ID support, Hafnium must also be able to allocate and manage CD tables.
- Impact:
  - Lower impact for SP controlled stage-1 – only STE configuration affected, plus additional CFGI / TLBI command support.
  - Higher impact for Hafnium managed stage-1 – requires CD allocation and configuration.
    - Additional effort required for Substream ID support – requires CD table management.

CMDQ invalidation

- Hafnium currently supports broadcast of CPU TLBI instructions for SMMU invalidations.
  - This works well when page tables are shared with the CPU but may be expensive when only used by the SMMU.
    - Using DVM for SEPIDs requires temporarily switching VMID in VTTBR_EL2, which isn't currently supported in Hafnium.
  - TLB invalidation via CMDQ is generally preferred for SMMU-only page tables.
    - It may also be necessary for SMMU implementations where broadcast isn't supported or is prohibitively expensive.
- Challenges:
  - How best to configure page tables to use CMDQ invalidation?
    - We may need other configuration options too: use of outer shareable TLBIs for DVM.
  - Could many TLBIs cause the CMDQ to fill up? How best to handle this?
- Status:
  - Some prototyping of changes done: TLBI followed by CMD_SYNC.
- Impact:
  - Requires support for additional TLBI commands, and hooks in MM code to trigger this invalidation.
  - Additional work required if CMDQ is at risk of filling up from many TLBI commands.

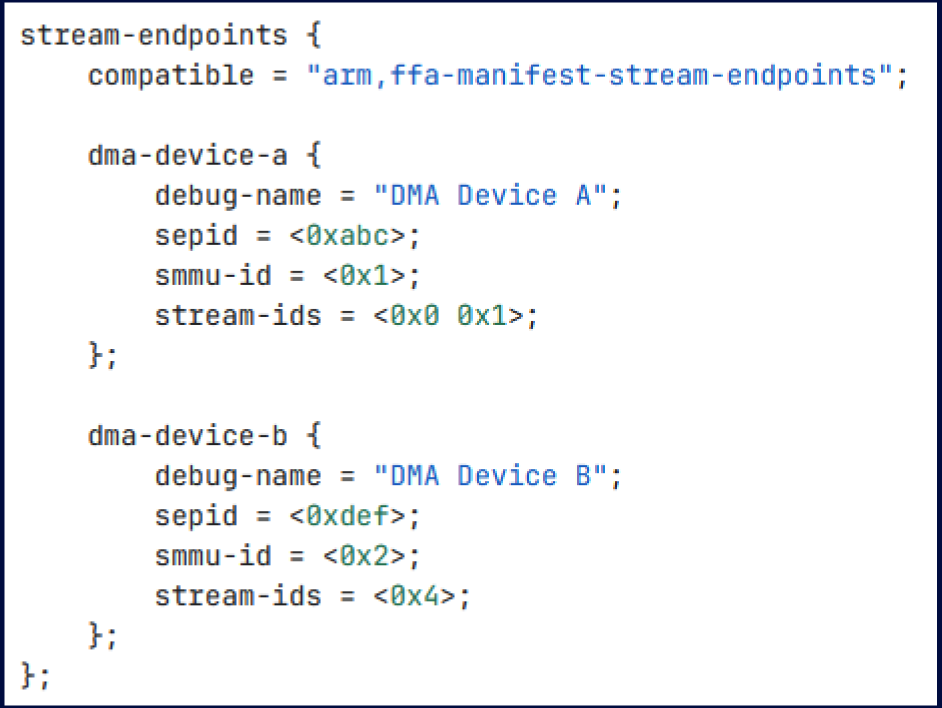
Event handling

- Hafnium allocates the event queue as part of SMMUv3 initialization, but no handling of events or errors is currently performed.
  - It would be useful to log such events for debugging of SMMU faults and other errors.
  - We should also check and record any global errors that may occur (via GERROR configuration).
  - Forwarding of events to SPs may also be useful for certain DMA use-cases.
    - SP may want event information to decide what action to take for a faulting DMA transaction.
    - SP may also want to be informed of issues related to context descriptors that it controls.
- Challenges:
  - How to configure the handling required for each type of event? (logging, event forwarding, panic, etc.)
- Impact:
  - Requires handling of event interrupts and consumption of event queue & GERROR register.
  - Additional effort required for forwarding events to SPs – need to convert to format expected by SP interface.

SEPID & Proxy SP Support

Parsing of SEPIDs from SP manifest

- Add support for parsing of SEPID information from SP manifest.
  - As defined by FF-A specification, this should be supplied as tuples of (Name, SEPID, SMMU ID, Stream ID).
- Challenges:
  - How best to describe these tuples in device tree format?
  - Any other properties that may be useful?
    - Stream configuration? (nested or stage-2 only)
    - Substream configuration? (S1CDMax, valid SSIDs)
- Status:
  - Prototype of changes complete.
- Impact:
  - Requires additional parsing of nodes in DT manifest.



```
stream-endpoints {
    compatible = "arm,ffa-manifest-stream-endpoints";

    dma-device-a {
        debug-name = "DMA Device A";
        sepid = <0xabc>;
        smmu-id = <0x1>;
        stream-ids = <0x0 0x1>;
    };

    dma-device-b {
        debug-name = "DMA Device B";
        sepid = <0xdef>;
        smmu-id = <0x2>;
        stream-ids = <0x4>;
    };
};
```

Device stream and SEPID bookkeeping

- Hafnium's handling of device streams is focused on static DMA isolation.
  - This needs to be extended to support dynamic DMA isolation and SEPIDs.
  - The existing bookkeeping also has some limitations.
    - One issue is that multiple SPs can claim the same stream; the last loaded will overwrite previous STE configuration.
  - Hafnium should ensure streams assigned to device and memory regions are consistent with SEPIDs.
- Challenges:
  - How best to keep track of which streams have already been claimed by a SP?
  - How to ensure SEPID configuration is consistent with device and memory regions of SPs?
- Status:
  - Separation of device state from VM structure has been prototyped.
    - Global device state makes it easier to track device ownership and stream configuration.
- Impact:
  - Variable – depends on how much refactoring required to existing bookkeeping.
    - Global device state is a somewhat substantial change – how sophisticated do we need to be?

Dynamic mapping support for IOMMU page tables

- Refactor IOMMU-specific MM APIs to support all standard mapping operations.
  - Hafnium currently supports identity map operations for static mappings; for dynamic mappings we also need to be able to unmap memory and defrag the page tables.
    - Support for non-identity mapping operations is also required.
  - IOMMU page tables should also use the relevant TLBI operations (i.e. CMDQ invalidation) as necessary.
- Challenges:
  - These APIs currently use a DMA device ID for selecting the relevant page table.
    - How do we translate from a SEPID to this internal device ID?
- Status:
  - Prototype of changes complete.
- Impact:
  - Requires updates to MM code to support additional IOMMU PT APIs.
  - Additional effort required for translating SEPID to the relevant IOMMU page table.

SEPID support in FF-A memory management

- Allow SPs to target SEPIDs when performing FF-A memory management operations.
  - When retrieving or relinquishing on behalf of a SEPID, the respective IOMMU page table should be updated rather than the tables of the caller SP.
- Challenges:
  - How to determine if a SEPID is being targeted in an FF-A memory management operation?
    - Is it safe to assume that if the endpoint ID does not match the SP’s ID, we have a SEPID?
- Status:
  - Prototype of changes complete.
- Impact:
  - Requires updates to FF-A memory management to support handling of SEPIDs, and to operate on their page tables (rather than the calling SP’s stage-2).

SMMU Virtualization

Virtualizing the SMMU

- SPs need an interface allowing them to manage the VA space of device streams.
  - This is necessary to support dynamic and other DMA isolation models.
  - The SPMC owns the SMMU, so we must provide a virtualized interface to enable this.
  - This interface needs to support both Hafnium managed and SP controlled stage-1 address spaces.
- Why not support full virtualization of the SMMUv3?
  - High complexity and effort, certain operations may have high overhead.
    - Need to support virtual stream table, CMDQ, EVENTQ, etc.
  - Requires SMMUv3 driver in SPs, doesn't work for other IOMMU architectures.
- Advantages of a paravirtual interface:
  - Lightweight, just need to support interfaces necessary for address space management.
  - Can be used across different architectures; SP needs little or no awareness of underlying architecture.
    - For SP controlled stage-1, SPs need to be able to configure context descriptors.
  - May be able to leverage existing drivers and frameworks in SPs.

Hafnium managed stage-1 for SEPIDs

- Allow proxy SPs to map and unmap memory in SEPID stage-1.
  - This enables fine-grained access control of SEPIDs for proxy SPs on top of FF-A memory management.
  - Hafnium is responsible for allocating the stage-1 tables and context descriptors.
    - Preferred over SP controlled stage-1 as that requires sharing page table and context descriptor memory with the SEPID.
- Challenges:
  - Stage-1 PTs and context descriptors must be mapped in stage-2. How can this be achieved if Hafnium allocates them?
    - Solution: map Hafnium heap in a reserved region of stage-2 which is inaccessible to stage-1.
    - We can deny map and unmap operations that try to touch this reserved region.
  - Why not use stage-1 only translation to avoid mapping Hafnium heap in stage-2?
    - Requires alternative bookkeeping to track memory ownership state.
    - How to handle FFA_MEM_RELINQUISH when the memory is mapped in stage-1 in this scenario?
- Impact:
  - SMMUv3 driver requires support for nested translations but with Hafnium allocated context descriptors / tables.
    - On top of existing stage-1 support discussed earlier, this may require some tweaks to STE configuration.
  - Requires support in MM code for safely mapping Hafnium heap in stage-2 tables.

SP controlled stage-1 for shared stage-2

- Allow SPs to provide stage-1 tables when it shares stage-2 with its devices.
  - This allows SPs to share stage-1 tables between applications and devices.
- Challenges:
  - How does the SP know to use SMMUv3 context descriptors for providing the stage-1 table?
    - SP interface must support querying underlying hardware to discover page table format.
  - Could the paravirtual interface hide these hardware details instead?
    - SP could provide some memory to use for the context descriptor (or CD tables).
    - How does the SP discover how much memory to provide? How should Hafnium track and manage this memory?
  - Why not support Hafnium managed stage-1 tables for shared stage-2?
    - Mapping the Hafnium heap in stage-2 is unsafe as Hafnium does not control the SP's stage-1.
    - Hafnium would need to selectively map tables / CDs, or use a separate heap dedicated for this purpose.
- Impact:
  - Minimal impact on SMMUv3 driver – most effort is already covered by context descriptor support discussed earlier.
  - Additional effort required to support querying of page table format (depending on interface chosen).

SP interface: Hafnium paravirtual

- Add a custom paravirtual interface for virtual IOMMU management.
  - Following convention of existing paravirtual interfaces: HF_IOMMU_*.
  - Lightweight and can align closely with underlying operations and/or hardware.
- Challenges:
  - What operations should this paravirtual interface support?
    - Hafnium managed S1: create/destroy S1 PT, attach/detach stream, map/unmap
    - SP controlled S1: Attach/detach context descriptor, invalidate
  - How to forward events with this interface?
    - Would a VIRQ be required here?
- Impact:
  - Lower impact to Hafnium codebase – could use minimal wrappers to internal functions.
  - Non-standardized interface; new drivers required in SPs.
  - May be useful as short-term solution – can move to standardized interface later.

SP interface: Virtio-IOMMU

- Add Virtio-IOMMU backend for virtual IOMMU management.
  - Standardized and well understood paravirtual interface.
  - Virtio support could be leveraged for other use-cases in Hafnium.
- Challenges:
  - Which transport to use?
    - MMIO: standardized and well understood transport, requires trap-and-emulate support in Hafnium.
    - Virtio-msg over FF-A: aligned with FF-A but not yet standardized and major changes required for targeting SPMC or an LSP as an endpoint.
  - Requires non-standard extension for SP controlled stage-1 (ATTACH_TABLE).
    - Custom extensions may be required for other functionality (Substream IDs, arch-specific mapping attributes).
  - Integration: Should a generic Virtio framework library be used, or something Hafnium specific?
- Impact:
  - Higher impact on Hafnium codebase – base Virtio device support, transport layer, IOMMU specific handling.
    - May be able to leverage generic framework to reduce effort here.
    - Code added for Virtio could be leveraged for other use-cases (trap-and-emulate, other Virtio devices, etc.).
  - Lower impact on SPs: can leverage existing Virtio drivers / framework.

SP interface: FF-A based

- Define new FF-A based interfaces for virtual IOMMU / DMA management.
  - Can be closely aligned with DMA isolation models defined in FF-A.
- Challenges:
  - Requires discussion and buy-in from FF-A community to become standardized.
  - Would this use new FF-A functions or existing messaging interfaces (i.e. framework messages)?
  - How to forward events with this interface?
    - Via framework notifications?
- Impact:
  - Low to medium impact on Hafnium codebase – can leverage existing FF-A framework.
  - Additional effort required to standardize interface.
  - Medium impact on SPs: need to support new interface but can leverage existing FF-A drivers.

Questions?

Thank you

Follow us on:

For more information, visit us at qualcomm.com & qualcomm.com/blog