# NVM Express® Support for CXL®

**Sponsored by NVM Express organization, the owner of NVMe® Specifications**

# Speakers

Bill Martin

Jason Molgaard

SAMSUNG

SOLIDIGM™

# Agenda

- Combining CXL® and NVMe® Technologies
- Computational Storage a Use for combining CXL and NVMe Technologies
- Computational Storage Use Cases

# Combining NVMe® Technology and CXL® Protocol

# Why Combine CXL® and NVMe® Technologies?

- NVMe devices are providing host accessible memory in the form of Subsystem Local Memory (SLM)
  - Accessing this memory via a memory protocol is more efficient
  - Allows cache coherency of that memory
  - Allows peer-to-peer communication using a memory model
- Computational Storage Use
  - Computational Storage Drives have more host accessible memory than a traditional Storage Device
  - Benefits from peer-to-peer communication (more on this later)

# Benefits of CXL® Load/Store Access

- What does CXL bring to the table that benefits NVMe® technology?
  - Allows coherent memory between a host and one or more devices with SLM
  - Low latency, fine granularity path to access SLM
  - CXL.mem allows direct Load/Store access to SLM
  - Allows Peer-to-Peer communication using CXL.mem
- How is this different from CMB/PMR?
  - CXL allows both coherency with host memory and MMIO space – CMB/PMR only allows host Load/Store access over PCIe® architecture using uncached MMIO space
  - CXL provides coherency for device access to host memory
  - CXL protocol is more efficient than PCIe memory access protocol
    - CXL enables lower latency and higher throughput
    - CXL protocol has less strict ordering rules than PCIe memory access protocol

# Benefits of Coherency

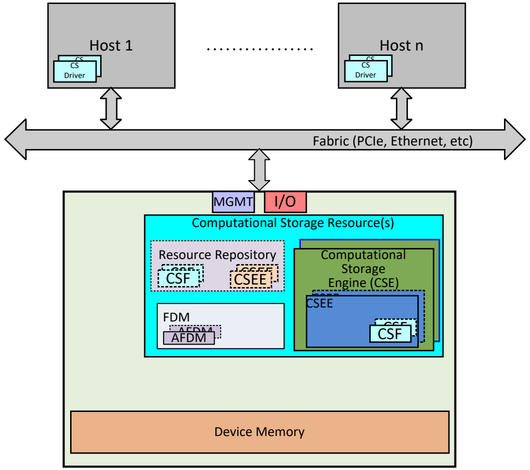
- All devices perceive the same view of memory
  - Memory viewed between devices is consistent
- All devices perceive the same view of shared data
  - Data is up-to-date
- Devices and hosts can push data to each other or pull data from each other
  - This includes device-to-device communication
- Avoids or reduces copies that can grow stale

# Computational Storage

## A Use for CXL® and NVMe® Technologies Together

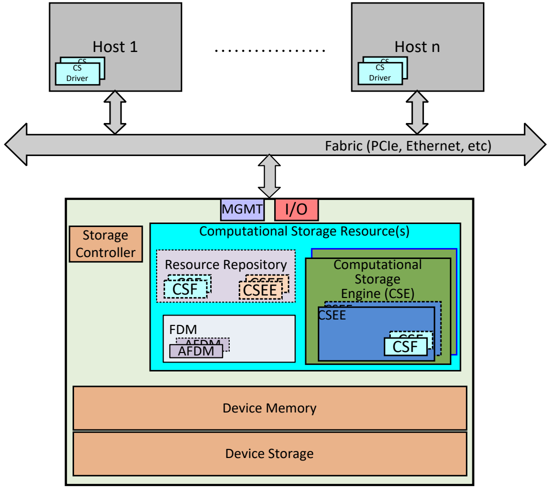
# Computational Storage Architecture

## Computational Storage Processor



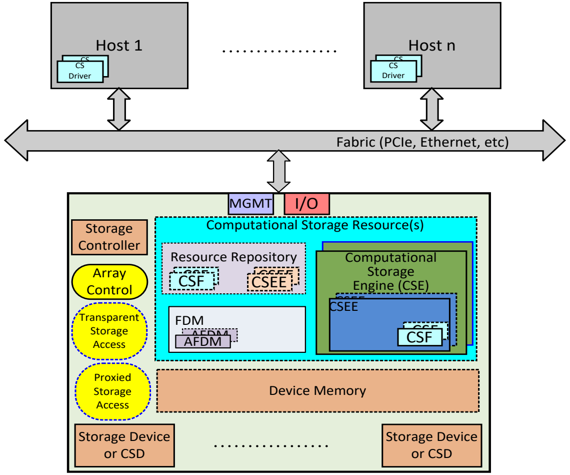
Computational Storage Processor (CSP)

## Computational Storage Drive



Computational Storage Drive (CSD)

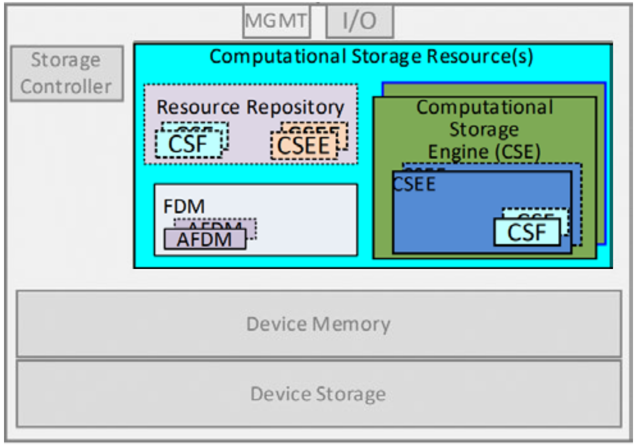
## Computational Storage Array



Computational Storage Array (CSA)

CSx = Computational Storage **Device** – CSP or CSD or CSA

# A Deeper Dive of the CSx Resources



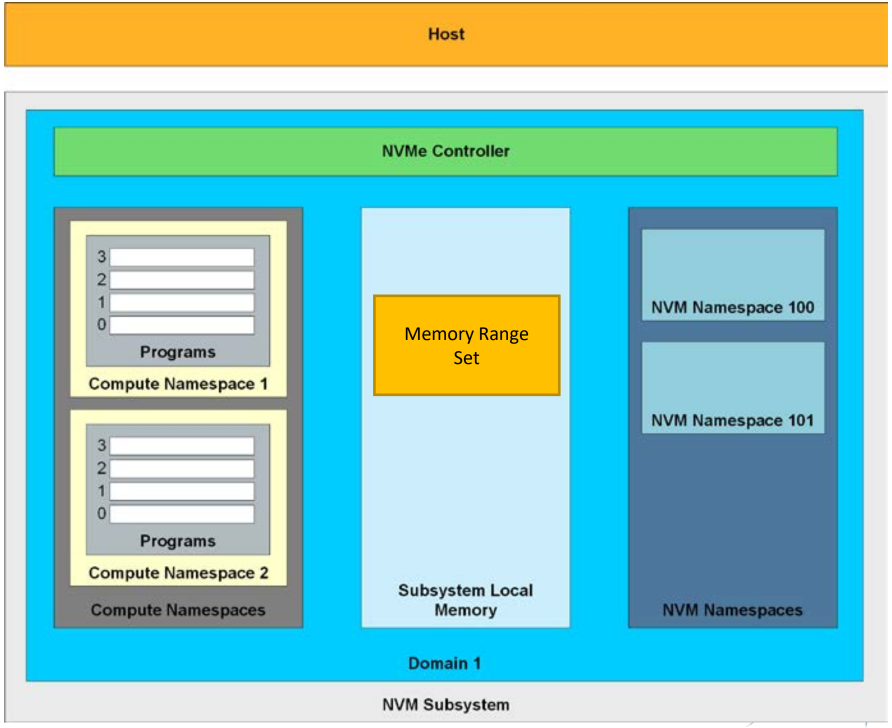
Computational Storage Drive (CSD)

- **CSR -** Computational Storage Resources are the resources available in a CSx necessary for that CSx to store and execute a CSF
- **CSF -** A Computational Storage Function is a set of specific operations that may be configured and executed by a CSE in a CSEE
- **CSE -** Computational Storage Engine is a CSR that is able to be programmed to provide one or more specific operation(s)
- **CSEE -** A Computational Storage Engine Environment is an operating environment space for the CSE
- **FDM -** Function Data Memory is device memory that is available for CSFs to use for data that is used or generated as part of the operation of the CSF
- **AFDM -** Allocated Function Data Memory is a portion of FDM that is allocated for one or more specific instances of a CSF operation
- **Resource Repository –** Resources that are available but not activated

# NVMe® Computational Storage Basics

- Computational Programs command set introduced Compute Namespace
- Subsystem Local Memory (SLM) command set introduced Memory Namespace
- Compute Namespace can access SLM Namespace using a Memory Range Set
- CSE = Compute Engine
- CSF = Program
- Function Data Memory (FDM) = SLM
- Allocated FDM = Memory Range Set
- Device Storage = NVM Namespace

®



# NVMe® TP4184
# Host Addressable SLM

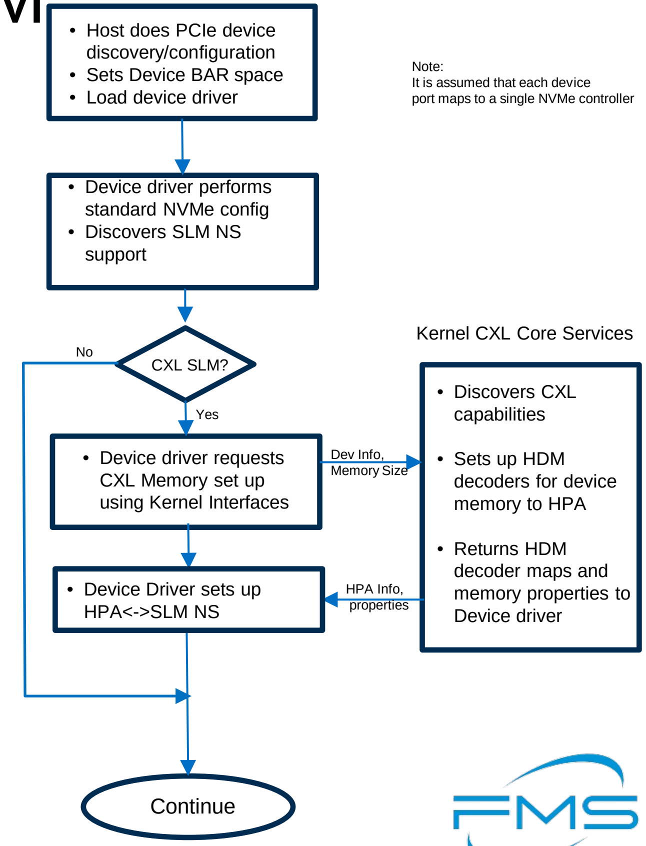
# Host Addressable SLM – NVMe® TP4184

- NVMe TP4184 is in the Architecture Definition phase
- SLM is addressed at a Host Physical Address (HPA)
  - PCIe® BAR; or
  - CXL®
- SLM memory can have a virtual mapping for host applications
  - Host application does not have to switch contexts for SLM memory access
- SLM memory is accessible by the host and the device
- SLM can be read/written with:
  - Host Load/Store commands
  - CXL.mem commands
- Compute is triggered with Computational Programs commands
  - Utilizes the host addressable SLM
- Allows P2P data movement based upon HPA
- SLM is still accessible using the SLM Command Set Memory Read and Memory Write commands

# Potential Config Flow for CXL® SLM

- NVMe® device with CXL SLM is similar to a CXL Type 2 device
- Plan for a CXL Type 2 device is for the OS  to do standard PCIe® configuration  (e.g. allocating BAR space) and then load driver CXL configuration using device's PCI ID
- NVMe device with CXL SLM will use an enhanced NVMe driver
  - Enhanced for CXL based SLM configuration
  - Device will use existing NVMe Class Code and may use a new Programming Interface (PI) identifier to expose CXL capabilities
- Device driver discovers device capabilities and calls OS CXL core services for CXL Memory set up
  - CXL core services offers kernel interfaces for the driver to set up required CXL capabilities such as HDM decoders and return necessary information (e.g. HPA range)
    - Device CXL memory allocation is controlled by the NVMe driver
    - Linux support for CXL Type 2 devices is not yet available
  - Driver owns runtime management of Device CXL memory

**Per-Port High-Level Config Flow**



- Host does PCIe device discovery/configuration
- Sets Device BAR space
- Load device driver

Note:
It is assumed that each device port maps to a single NVMe controller

- Device driver performs standard NVMe config
- Discovers SLM NS support

CXL SLM?

No

Yes

- Device driver requests CXL Memory set up using Kernel Interfaces

Dev Info, Memory Size

Kernel CXL Core Services

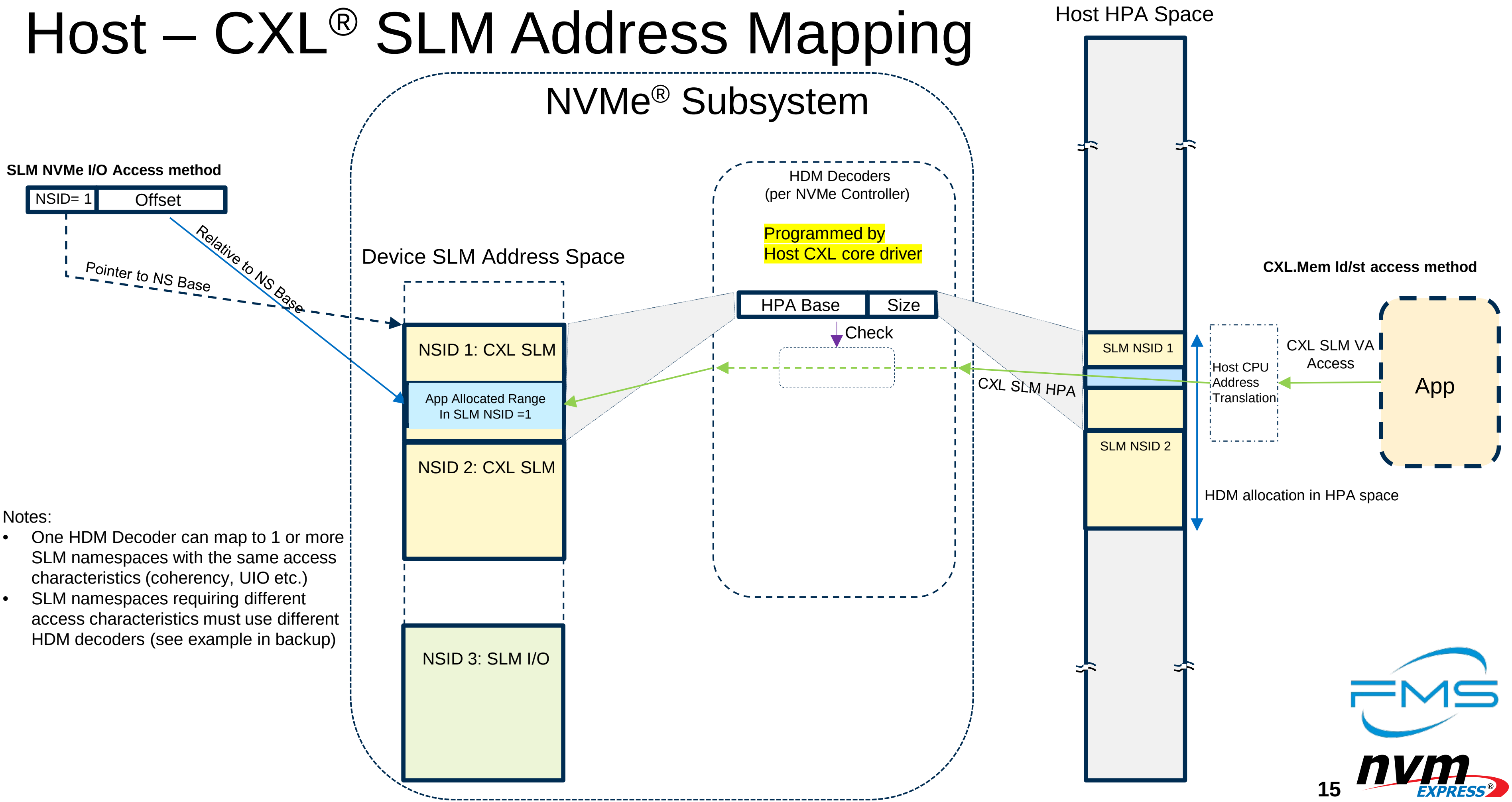
- Discovers CXL capabilities
- Sets up HDM decoders for device memory to HPA
- Returns HDM decoder maps and memory properties to Device driver

HPA Info, properties

- Device Driver sets up HPA<->SLM NS

Continue

# Host – CXL® SLM Address Mapping



Notes:

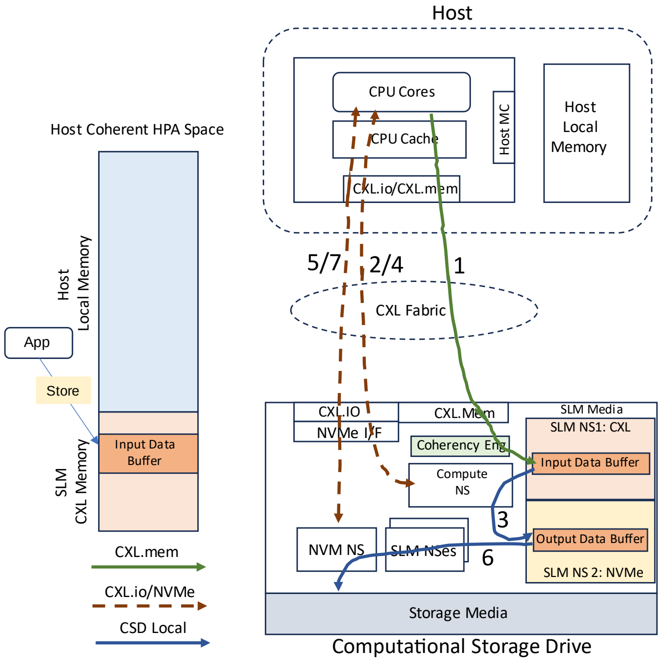
- One HDM Decoder can map to 1 or more SLM namespaces with the same access characteristics (coherency, UIO etc.)
- SLM namespaces requiring different access characteristics must use different HDM decoders (see example in backup)

# Use Cases

# Use Case 1: Data Post-Processing (Before Writing to Storage)

- Value Proposition
  - Avoid copying data using DMA from/to Host Memory
  - Lower latency CXL® based direct Load/Store access, especially for small input data
- Configuration
  - Input Data Buffer is in SLM CXL memory address space
  - Output Data Buffer is in SLM
- Example Use Case

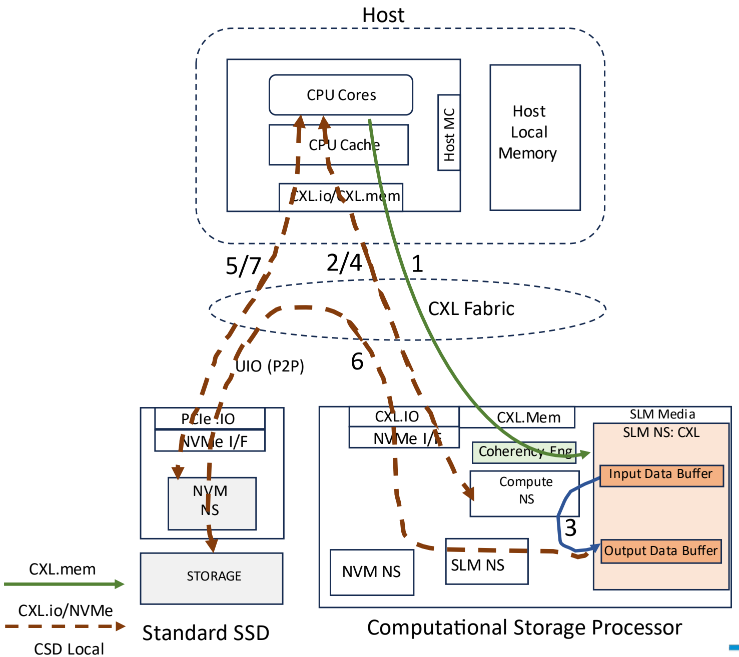
1. Application writes (Store) Input Data Buffer using CXL.mem
   - Some or all data may reside in Host Cache on completion
2. Host issues NVMe® Execute Program command to Compute Namespace
3. Compute Namespace Operates on data in Input Data Buffer and stores results in Output Data Buffer
   - Uses CXL BI Snoop protocol to keep Host Cache coherent with Input Data Buffer
4. CQE is posted for the Compute Namespace
5. Host issues NVMe Copy command to copy data from Output Data Buffer to Storage Media
6. Data is copied to Storage Media from Output Data Buffer
7. CQE is posted for the NVM Namespace



# Use Case 2: Data Post-Processing with a Standard SSD

- Value Proposition
  - Bypass data movement through Host memory
- Configuration
  - Input Data Buffer is in SLM CXL® memory address space
  - Output Data Buffer is in SLM CXL memory address space
- Example Use Case

1. Application writes (Store) Input Data Buffer using CXL.mem
   - Some or all data may reside in Host Cache on completion
2. Host issues NVMe® Execute Program command to Compute Namespace
3. Compute Namespace operates on data in Input Data Buffer and stores results in Output Data Buffer
   - Uses CXL BI Snoop protocol to keep Host Cache coherent with Input Data Buffer and Output Data Buffer
4. CQE is posted for Compute Namespace
5. Host generates IO Write to SSD NVM Namespace
   - Data Pointer points to Output Buffer in SLM (HDM)
6. SSD uses PCIe® UIO for direct P2P from HDM space and writes to storage media
   - Since output buffer is in CXL HDM space, UIO can't use BAR space for P2P
7. CQE is posted for NVM Namespace



# Summary and Next Steps

- CXL® and NVMe® technologies can be used simultaneously
  - CXL brings Load/Store access to NVMe SLM
  - CXL and NVMe work together to support NVMe IO Command Sets including the Computational Storage
- Benefits
  - Coherency between device SLM and host
  - Lower latency for small data transfers between host and SLM
  - Since SLM is addressable via HPA, no need to copy data from host memory to SLM
  - Bypassing the host for peer-to-peer data movement
- Looking Ahead
  - CXL and NVMe Computational Storage are on trajectories that will intersect
  - Enhancing NVMe SLM to support CXL is a step to enable convergence/collaboration

# Questions?

nvm EXPRESS®

*Architected for Performance*