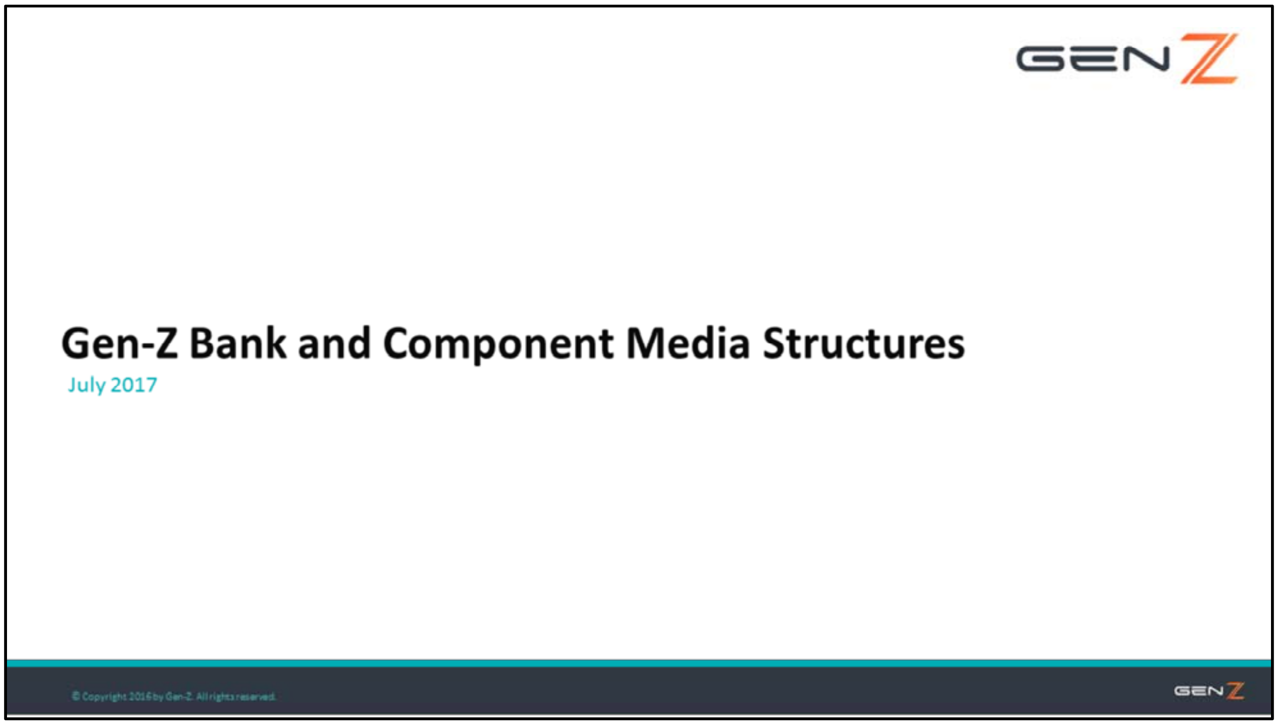
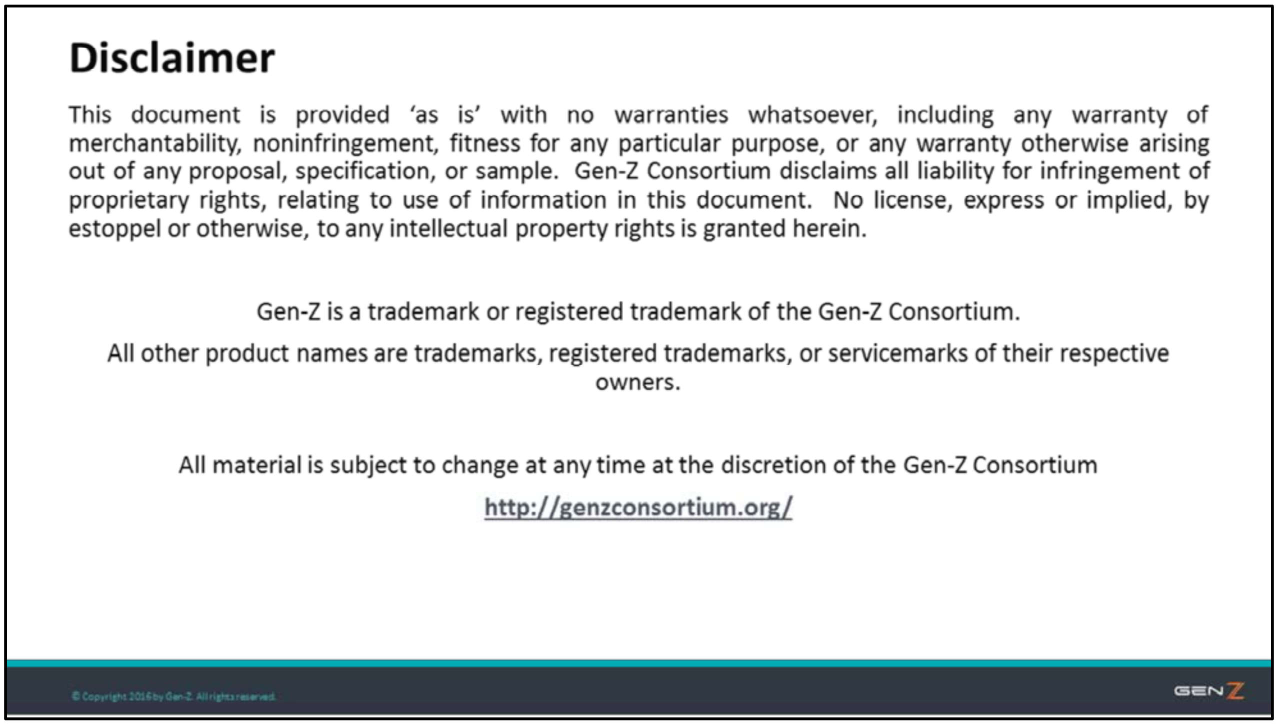


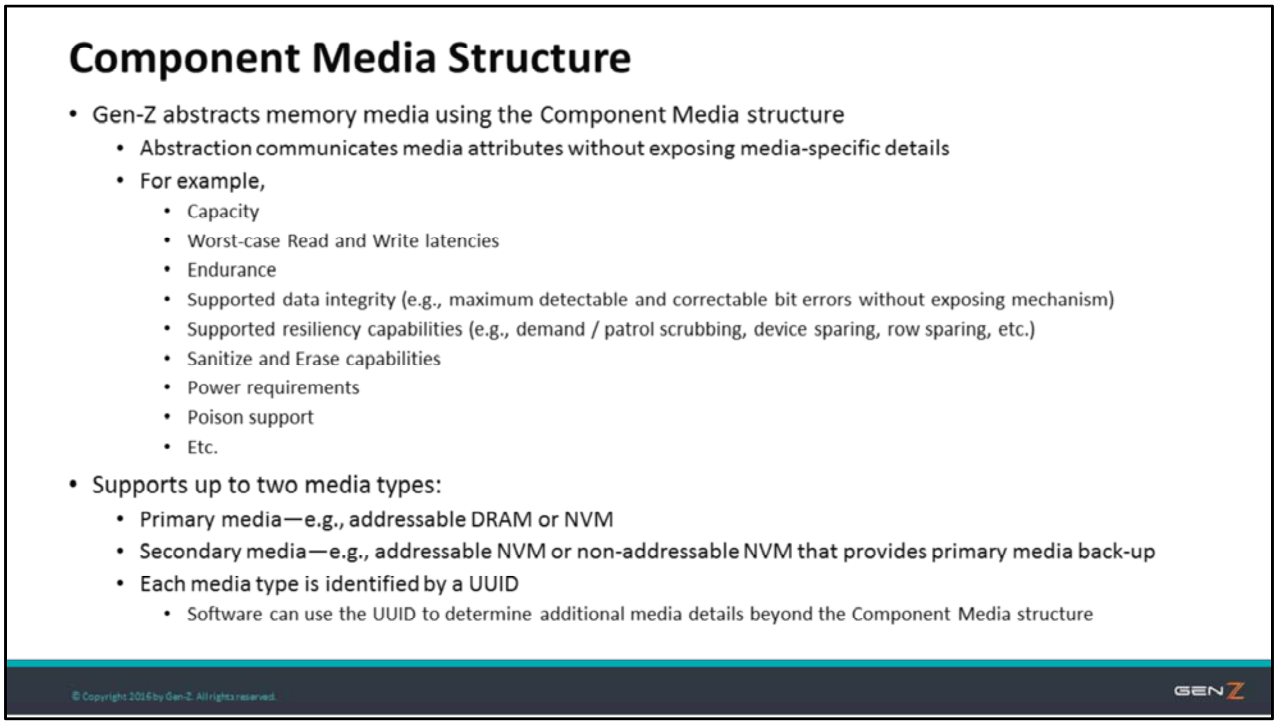
Gen-Z Bank and Component Media Structures

July 2017

This presentation covers Gen-Z Bank and Component Media structures. Gen-Z Bank structures apply only to components that support P2P-Core OpClass. The Component Media structure is used by any component that supports addressable media.



Disclaimer



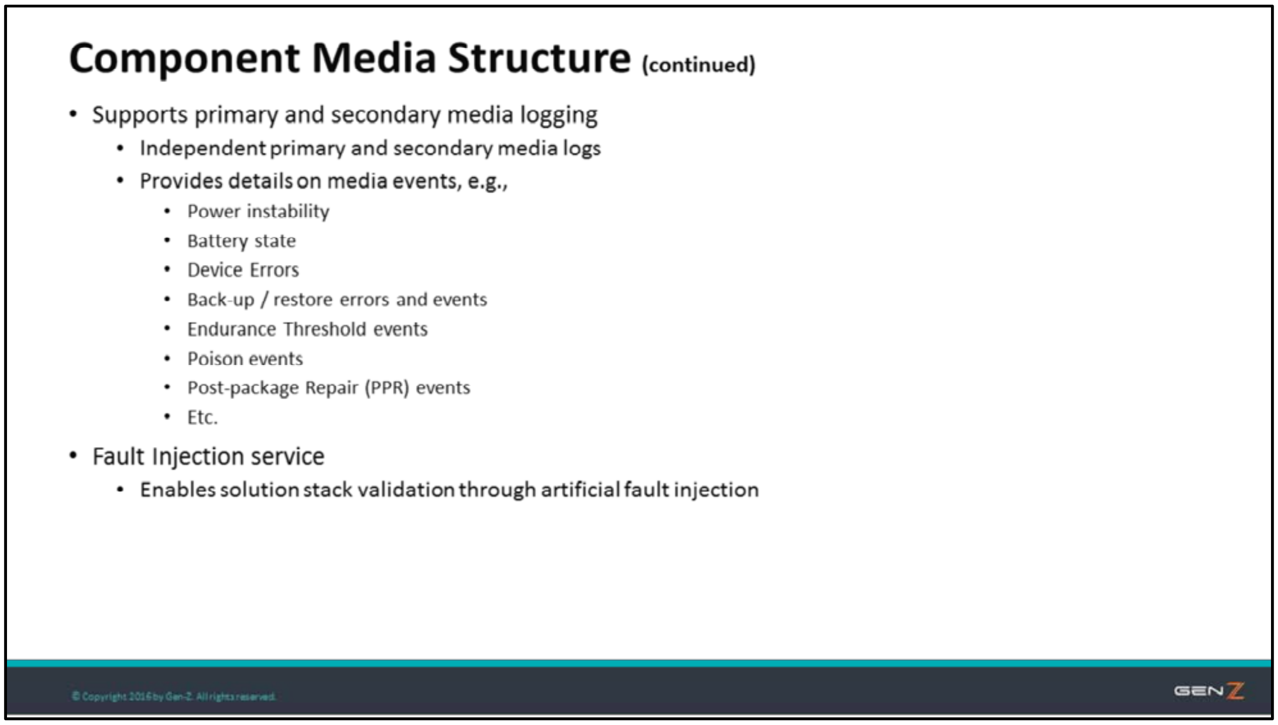
Component Media Structure

- Gen-Z abstracts memory media using the Component Media structure
  - Abstraction communicates media attributes without exposing media-specific details
  - For example,
    - Capacity
    - Worst-case Read and Write latencies
    - Endurance
    - Supported data integrity (e.g., maximum detectable and correctable bit errors without exposing mechanism)
    - Supported resiliency capabilities (e.g., demand / patrol scrubbing, device sparing, row sparing, etc.)
    - Sanitize and Erase capabilities
    - Power requirements
    - Poison support
    - Etc.
- Supports up to two media types:
  - Primary media—e.g., addressable DRAM or NVM
  - Secondary media—e.g., addressable NVM or non-addressable NVM that provides primary media back-up
  - Each media type is identified by a UUID
    - Software can use the UUID to determine additional media details beyond the Component Media structure

The Component Media structure can be used by any component that contains addressable memory media (volatile or non-volatile) or storage media. Its primary purpose is to abstract the media to enable access by any Requester without having to be media-specific aware. For example, a Requester such as a processor that supports the Component Media structure could interoperate with multiple media types—DRAM, MRAM, PCM, STTRAM, etc. using the same Gen-Z operations to access the media. Further, the abstraction enables a media controller to provide numerous services, e.g., wear leveling, resiliency (row sparing, device sparing, etc.), demand and patrol scrubbing, etc. without involving the memory controller. This can improve performance and reduce power consumption by eliminating unnecessary data movement.

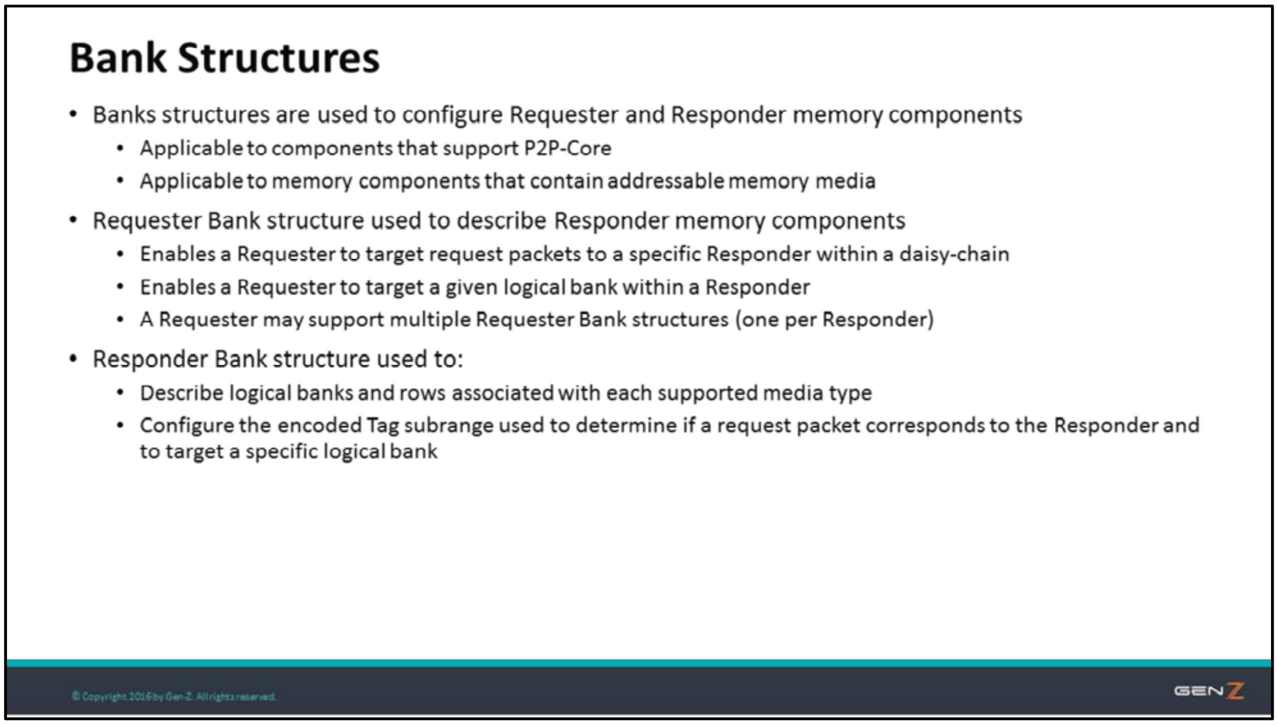
The Component Media structure communicates multiple media attributes including capacity, latency, endurance, etc. It also communicates additional media services and capabilities, e.g., scrubbing, device and row sparing, sanitize and erase, etc. capabilities.

A component can support multiple media types. Each Component Media structure specifies up to two media types—a primary media and a secondary media. If a component supports more than two media types, then it provisions additional Component Media structures. Each media type is identified by a UUID. Not only does a UUID enable software to identify the media type, the UUID can identify numerous media attributes, e.g., media-specific attributes not abstracted by the Component Media structure such as a specific generation of a given media type.


Component Media Structure (continued)

- Supports primary and secondary media logging
  - Independent primary and secondary media logs
  - Provides details on media events, e.g.,
    - Power instability
    - Battery state
    - Device Errors
    - Back-up / restore errors and events
    - Endurance Threshold events
    - Poison events
    - Post-package Repair (PPR) events
    - Etc.
- Fault Injection service
  - Enables solution stack validation through artificial fault injection

In addition to media abstraction, the Component Media structure is used to manage media logs and to provide a fault injection service to ensure correct solution stack operation (the ability to test all fault conditions using standard controls is extremely important to customers not just for quality assurance, but also to help debug problems in the field).

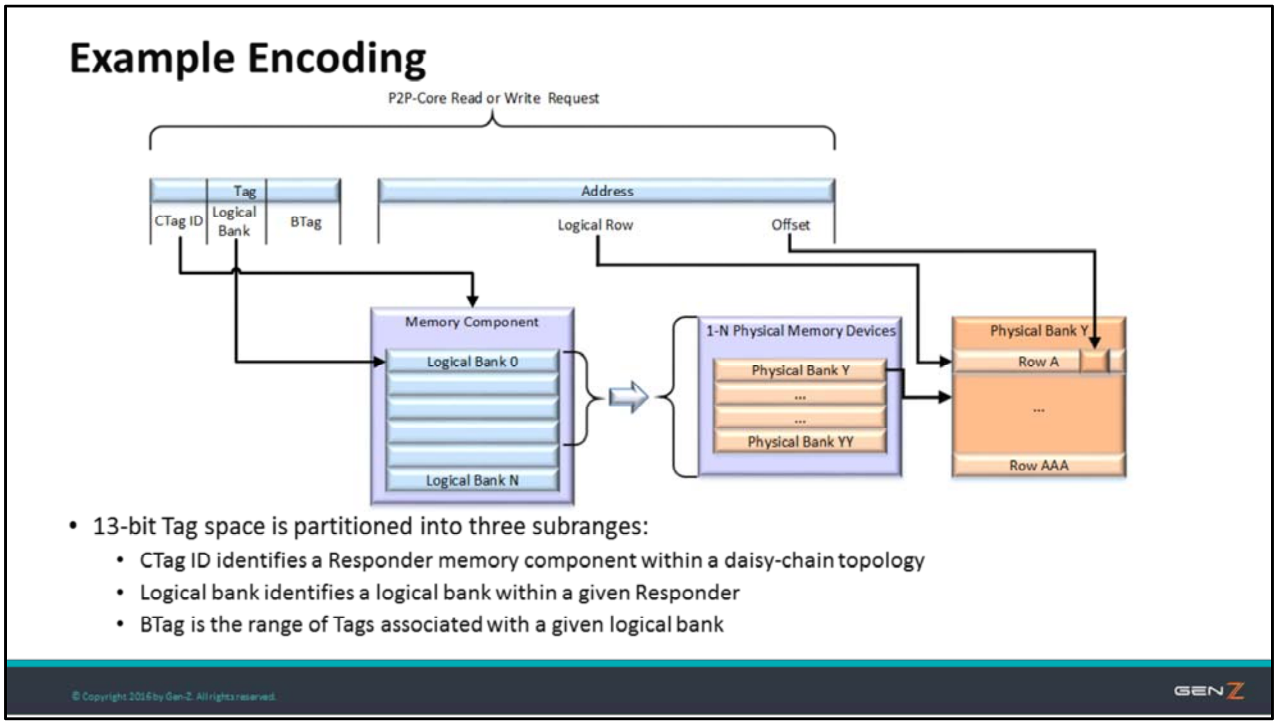


Bank Structures

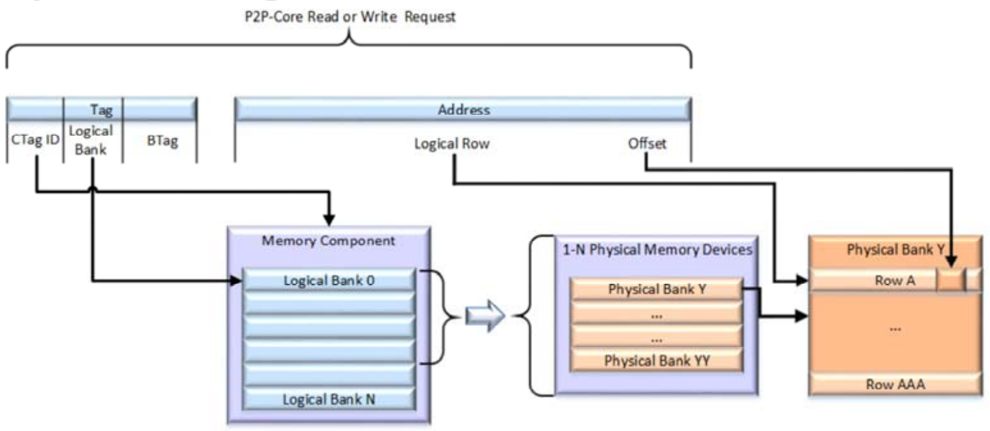
- Banks structures are used to configure Requester and Responder memory components
  - Applicable to components that support P2P-Core
  - Applicable to memory components that contain addressable memory media
- Requester Bank structure used to describe Responder memory components
  - Enables a Requester to target request packets to a specific Responder within a daisy-chain
  - Enables a Requester to target a given logical bank within a Responder
  - A Requester may support multiple Requester Bank structures (one per Responder)
- Responder Bank structure used to:
  - Describe logical banks and rows associated with each supported media type
  - Configure the encoded Tag subrange used to determine if a request packet corresponds to the Responder and to target a specific logical bank

Bank structures are applicable only to components that support the P2P-Core OpClass. There are two bank structure types—the Requester Bank structure and the Responder Bank Structure.

- Requester Bank structures are supported by Requesters. Each structure is used to access a specific Responder and configure logical bank information to ensure efficient access with minimal latency and overhead.
- Responder Bank structures are supported by Responders. Each structure describes the Responder-specific capabilities—number of logical banks, number of logical rows per bank, bank size, and to configure the bit masks used to interpret the encoded Tag field within request packets.



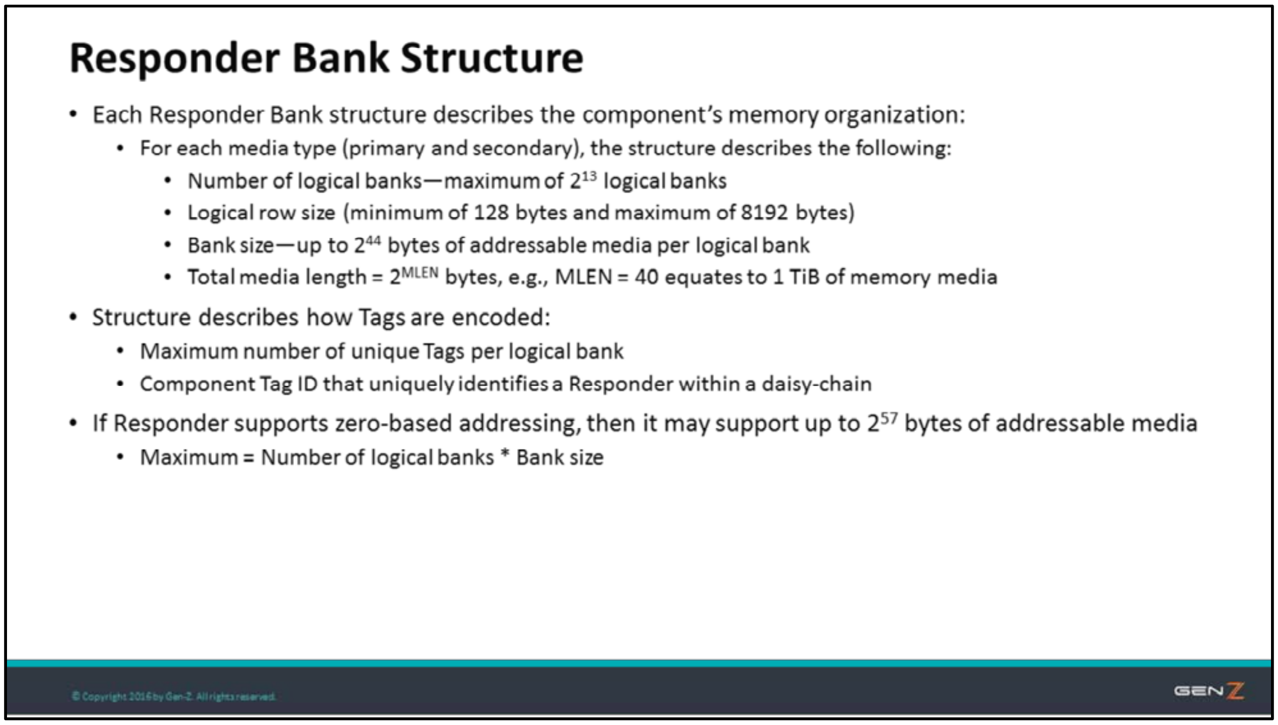
Example Encoding



- 13-bit Tag space is partitioned into three subranges:
  - CTag ID identifies a Responder memory component within a daisy-chain topology
  - Logical bank identifies a logical bank within a given Responder
  - BTag is the range of Tags associated with a given logical bank

The Tag field within a P2P-Core OpClass request packet is encoded. Bit masks are applied to decode the Tag:

- The CTag ID identifies each responder within a daisy-chain. For example, if there were two Responders, then the CTag ID is just a single bit.
- A subset of the bits represent the logical bank. For example, if a component supported 128 logical banks, then 7 bits were used to identify a given logical bank.
- The BTag bits identify a specific request packet within a given logical bank. For example, if 8 bits are used to identify the component and the logical bank, then a Requester can issue up to 256 request packets per logical bank.

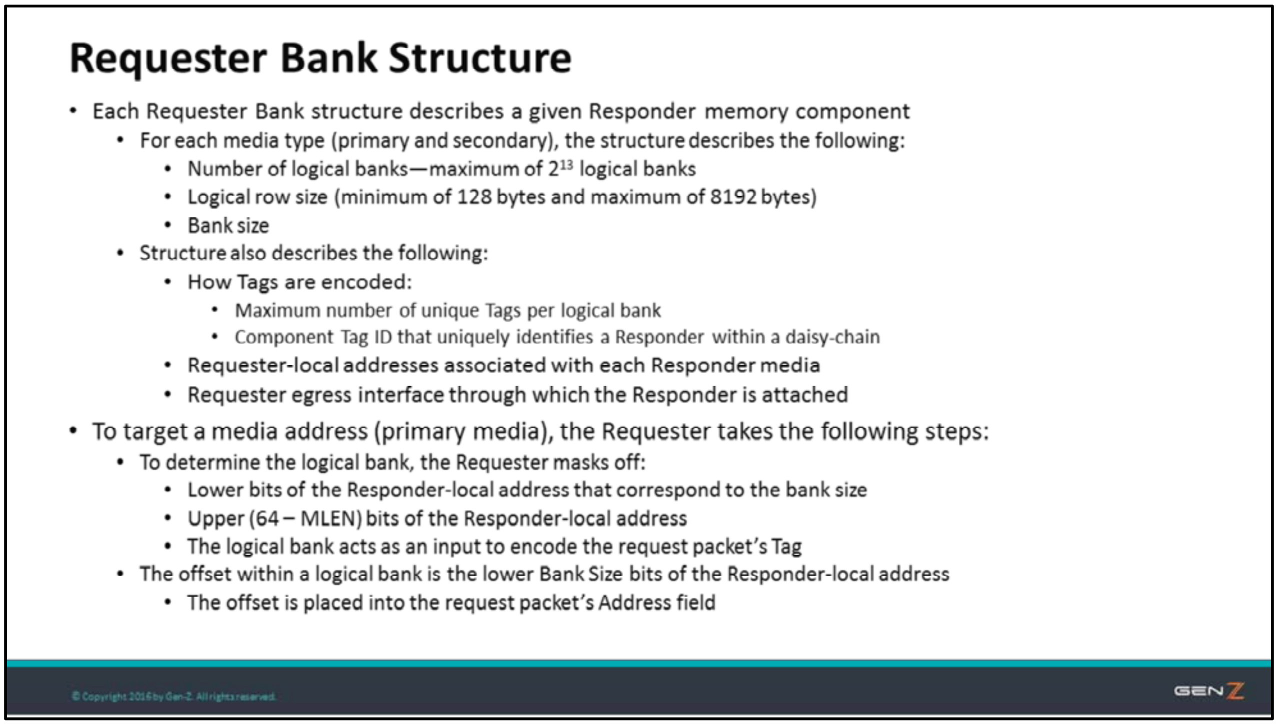


Responder Bank Structure

- Each Responder Bank structure describes the component's memory organization:
  - For each media type (primary and secondary), the structure describes the following:
    - Number of logical banks—maximum of 2^{13} logical banks
    - Logical row size (minimum of 128 bytes and maximum of 8192 bytes)
    - Bank size—up to 2^{44} bytes of addressable media per logical bank
    - Total media length = 2^{MLEN} bytes, e.g., MLEN = 40 equates to 1 TiB of memory media
- Structure describes how Tags are encoded:
  - Maximum number of unique Tags per logical bank
  - Component Tag ID that uniquely identifies a Responder within a daisy-chain
- If Responder supports zero-based addressing, then it may support up to 2^{57} bytes of addressable media
  - Maximum = Number of logical banks * Bank size

The Responder Bank structure describes a P2P-Core component's memory organization, i.e., the number of logical banks, the size of a logical row, the amount of resources provisioned per logical bank, logical row size, etc. These values are per media type. Nearly all of this structure's fields are read-only.

The Responder Bank structure is configured to indicate the number of bits used to identify a component and the CTag ID value associated with the specific Responder.

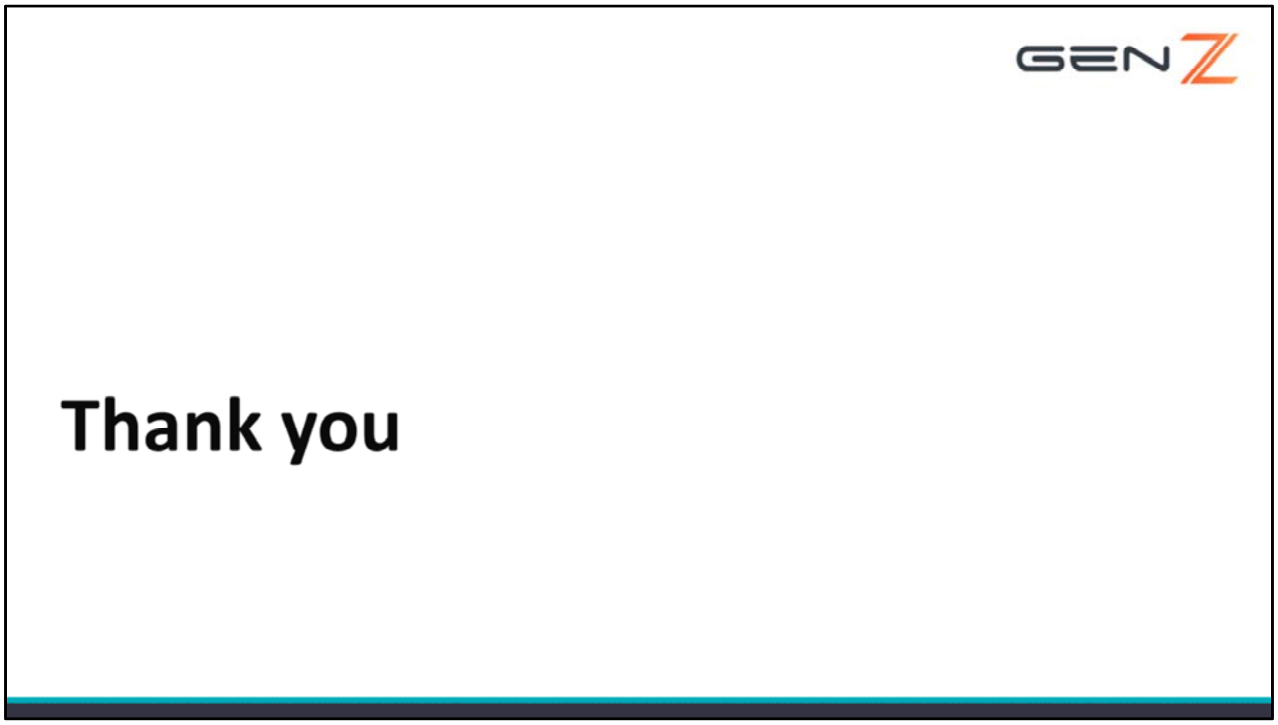


Requester Bank Structure

- Each Requester Bank structure describes a given Responder memory component
  - For each media type (primary and secondary), the structure describes the following:
    - Number of logical banks—maximum of 2^{13} logical banks
    - Logical row size (minimum of 128 bytes and maximum of 8192 bytes)
    - Bank size
  - Structure also describes the following:
    - How Tags are encoded:
      - Maximum number of unique Tags per logical bank
      - Component Tag ID that uniquely identifies a Responder within a daisy-chain
    - Requester-local addresses associated with each Responder media
    - Requester egress interface through which the Responder is attached
- To target a media address (primary media), the Requester takes the following steps:
  - To determine the logical bank, the Requester masks off:
    - Lower bits of the Responder-local address that correspond to the bank size
    - Upper (64 – MLEN) bits of the Responder-local address
    - The logical bank acts as an input to encode the request packet's Tag
  - The offset within a logical bank is the lower Bank Size bits of the Responder-local address
    - The offset is placed into the request packet's Address field

The Requester Bank structure describes each Responder component. A Requester, e.g., a SoC, provides one Requester Bank structure per supported P2P-Core Responder. Requester Bank structures are linked together to enable scalability and flexibility. Software configures each Requester Bank structure with a set of values it reads from the corresponding Responder Bank structure (number of logical banks, logical row size, and bank size. Once the topology is ascertained, software configures this structure with the values that describe the encoded Tag, the Requester-local address that is mapped to the Responder, and the Requester egress interface identifier to reach the Responder (a Requester can support multiple egress interfaces).

To target a specific media address, the Requester applies bit masks to the Responder-local address. This enables the Requester to encode the Tag, identify the Logical Bank, and calculate the bank offset in parallel with minimal latency.



Thank you

This concludes this presentation. Thank you.