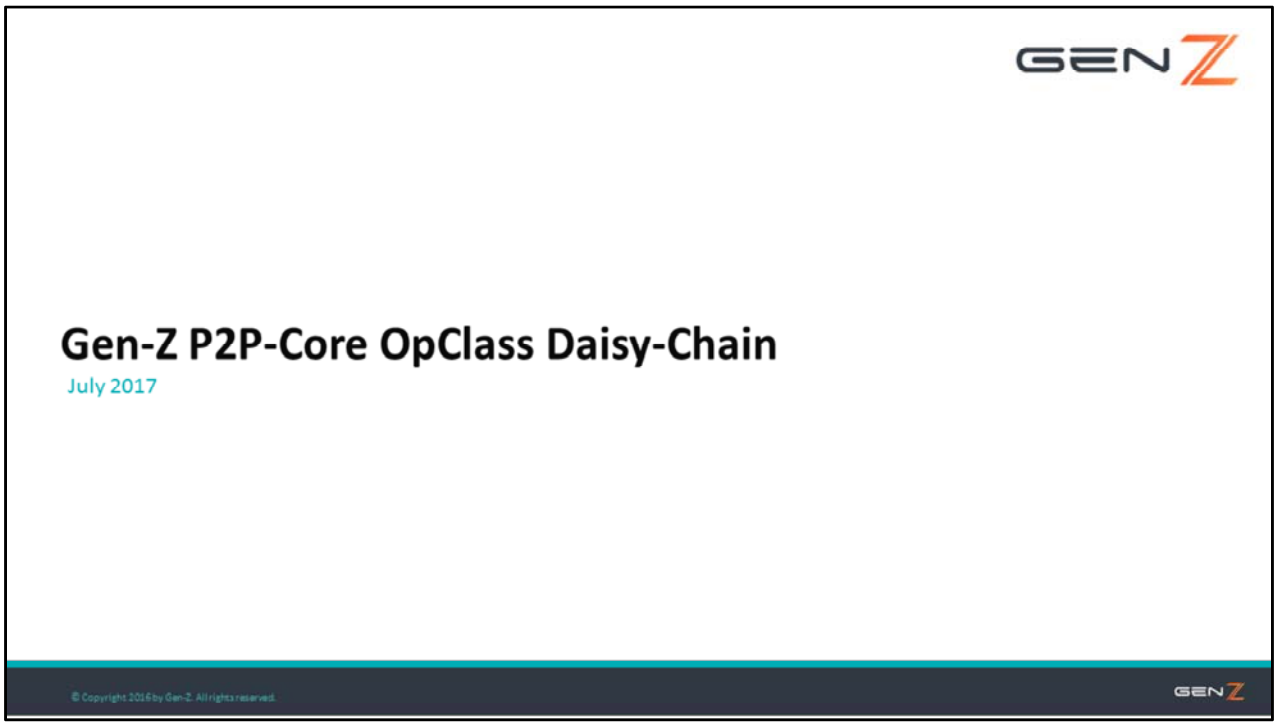


This presentation covers P2P-Core OpClass Daisy-chain topologies.

Disclaimer

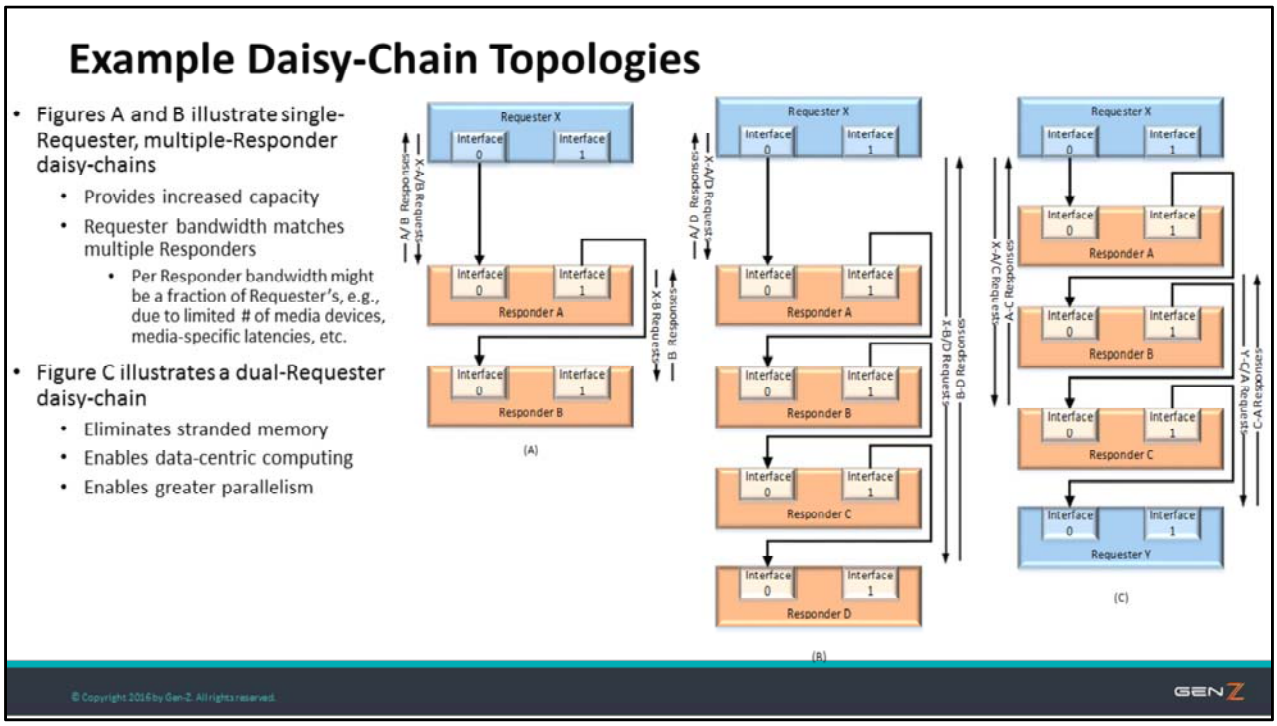
This document is provided 'as is' with no warranties whatsoever, including any warranty of merchantability, noninfringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Gen-Z Consortium disclaims all liability for infringement of proprietary rights, relating to use of information in this document. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

Gen-Z is a trademark or registered trademark of the Gen-Z Consortium.

All other product names are trademarks, registered trademarks, or servicemarks of their respective owners.

All material is subject to change at any time at the discretion of the Gen-Z Consortium

http://genzconsortium.org/



The P2P-Core OpClass is optimized for point-to-point and daisy-chain topologies composed of a processor and memory components. These figures illustrate example daisy-chain topologies. A Requester transmits a request packet (e.g., a read or write request) to Responder A. If the request is for Responder A, it executes the request and returns the response packet. If the request is not for Responder A, then it relays the request packet out interface 1 to Responder B, and the process repeats itself.

Daisy-chain topologies provide the following benefits:

- Provide increased capacity without requiring additional links / pins / larger packaging on the Requester.
- Enables the Requester to provision a higher-bandwidth link to match the aggregate bandwidth of a daisy chain. For example, a Requester can provision a 25-400 GB/s link that matches the aggregate bandwidth of multiple Responders.
- Though there is incremental latency (~5 ns) per hop, if the media is a slower SCM media or the data is organized such that latency-sensitive data is placed in the first component and less latency-sensitive data is placed in the subsequent components, then he incremental latency impact on application performance could be negligible.
- A daisy-chain can be used to ensure data is not stranded. Example C illustrates two Requesters that can access any memory component within the daisy-chain. If either fails, then the other retains data access. This is not possible in alternative technologies

which are generally limited to a single component interface. This becomes even more critical when using SCM media where multiple TBs of data could become inaccessible if alternative paths are not available.

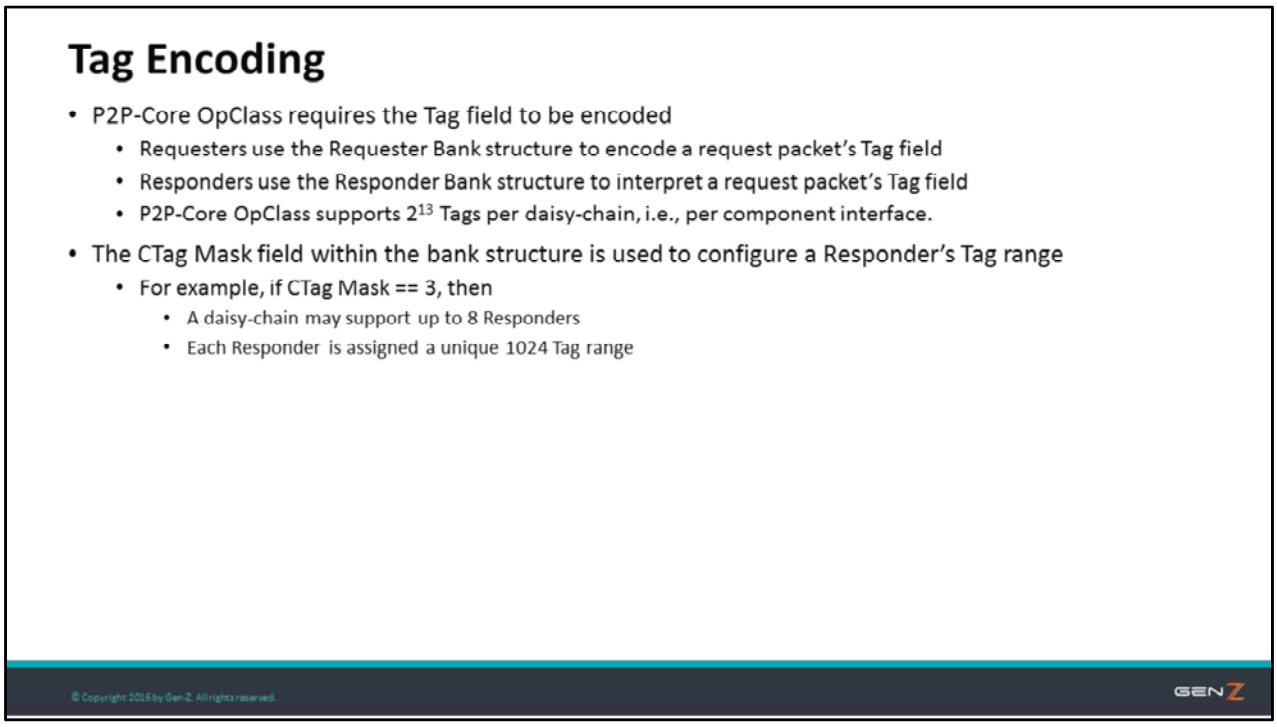
- Daisy-chain topologies enable greater parallelism and the opportunity to insert accelerators within the memory components to enable data-centric computing.
- Responders can support multiple daisy-chains, i.e., multiple pairs of interfaces. This increases aggregate bandwidth, enables higher-speed signaling, provides natural resiliency, etc.

Daisy-Chain Requirements

- Daisy-chain components shall support:
  - P2P-Core OpClass—protocol optimized for point-to-point and daisy-chain topologies
  - Explicit flow-control—to prevent buffer overflows
  - Link-level Reliability—ensures packets are reliably delivered across each link
  - Requester shall support a single interface per daisy-chain
    - May support multiple daisy-chains
  - Responders shall support two interfaces per daisy-chain—one to transmit between each component pair
    - Requester and Responder or Responder and Responder
    - May support multiple daisy-chains
- First component in daisy-chain is a Requester
  - Requesters can be any component type—SoC, FPGA, GPU, DSP, etc.
  - Shall limit the maximum number of outstanding requests to the smallest Max Outstanding Requests supported by any component in the daisy-chain
  - If two Requesters, then the maximum number of outstanding requests for both Requesters shall be limited to the smallest Max Outstanding Requests supported by any component in the daisy-chain
- Second component in a daisy-chain is typically a Responder
  - Typically, Responders are memory components, but can be any type that acts as a Responder

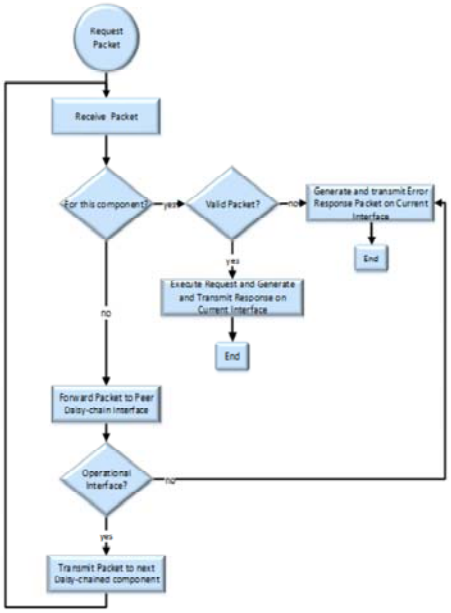
GEN Z

This slides details the high-level requirements to support daisy-chain topologies. In general, since Gen-Z supports and designs are strongly encouraged to implement multiple component interfaces, daisy-chain support requires only incremental logic and resources.



Unlike explicit OpClasses, the P2P-Core OpClass requires the Tag field to be encoded. The encoding is used to quickly identify if the request packet is for a Responder. If it is, then the Responder further decodes the Tag field to identify the logical bank. Encoding and decoding are performed using bit masks, thus are very lightweight and fast to execute.

Request Packet Forwarding

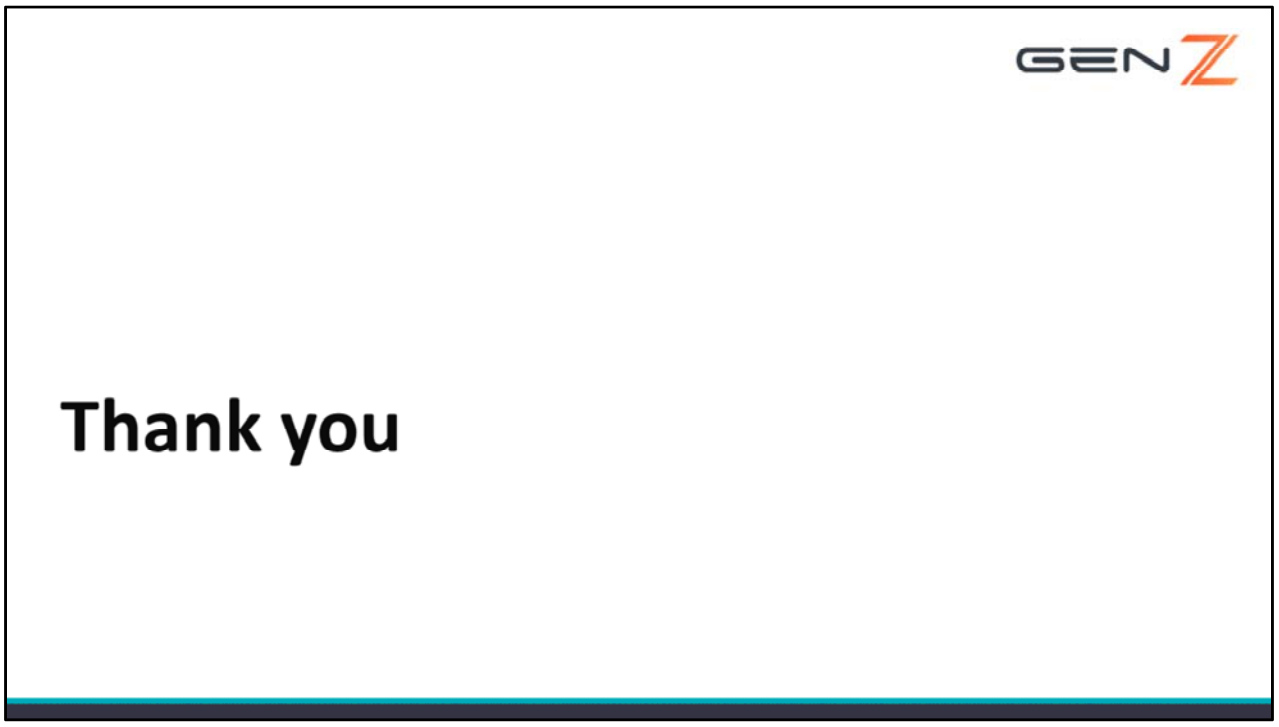


GEN Z

A Requester transmits a request packet to the first Responder

A request packet flows from the first Responder to the next until it reaches it destination or an error is detected

- The Responder compares the request packet's Tag field to the range of Tags assigned to the Responder
    - If the Tag falls within the range, then the Responder is the destination component
    - If the Tag falls outside of the range, then the Responder forwards the request packet to the next hop
        - If a next hop is not available, then the Responder generates an error
- A response packet flows from the Responder that executed the request packet or detected an error towards the Requester
    - If not a request packet, then the Responder forwards the packet to the peer daisy-chain interface for transmission
    - Upon receipt at a Requester, the response packet is validated and executed



This concludes this presentation. Thank you.