Gen-Z ZMMU and Memory Interleave

April 2019

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.

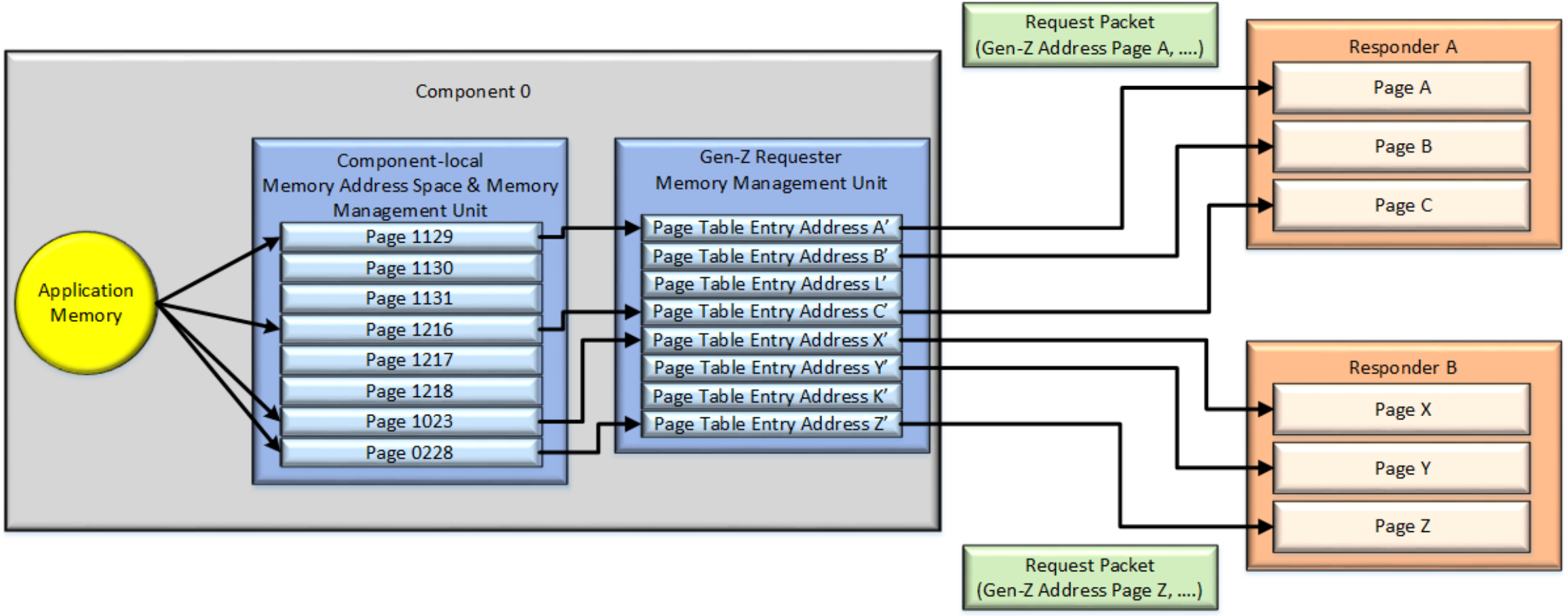
All material is subject to change at any time at the discretion of the Gen-Z Consortium

http://genzconsortium.org/

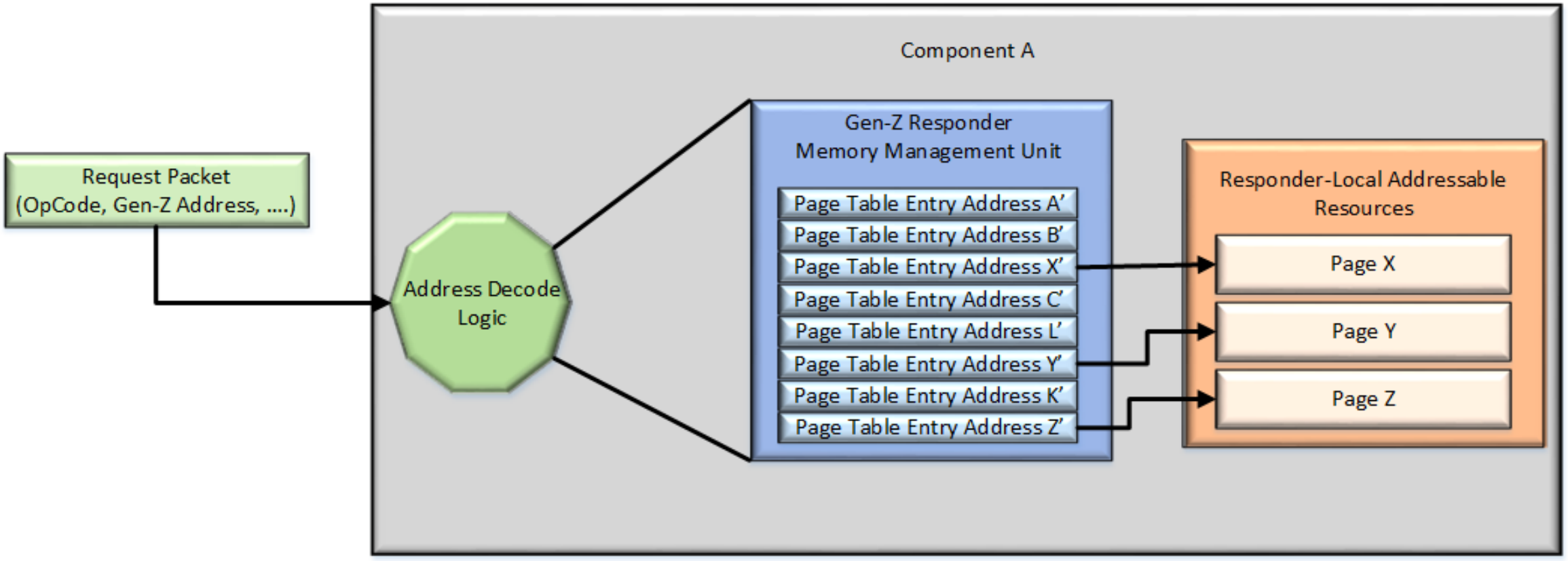
ZMMU

- Gen-Z supports two MMU (Memory Management Unit) types:
  - Requester ZMMU
    - Used to map Responder addressable resources
    - Enables application-transparent access, e.g., processor load-store operations are transparently translated to Gen-Z read and write operations that target a specific memory component
    - Applicable only to explicit OpClass operations
      - Gen-Z assumes memory-controller integration to optimize load-to-use latency for P2P-Core OpClass memory. Hence, P2P-Core relies exclusively upon Requester-specific decoding and interleaving
  - Responder ZMMU
    - Used to translate the packet's Address into Responder-specific media device addresses
    - Used to enforce page-level access permission (R-Keys)
- Components that act as both a Requester and as a Responder can support both ZMMU types
  - For example, processors, I/O components, memory / storage modules that support third-party data moves, etc.
  - Much of the underlying ZMMU IP can be re-used to implement Requester and Responder ZMMUs
- Gen-Z specifies two ZMMU designs:
  - Page Table-based ZMMU
  - Page Grid-based ZMMU

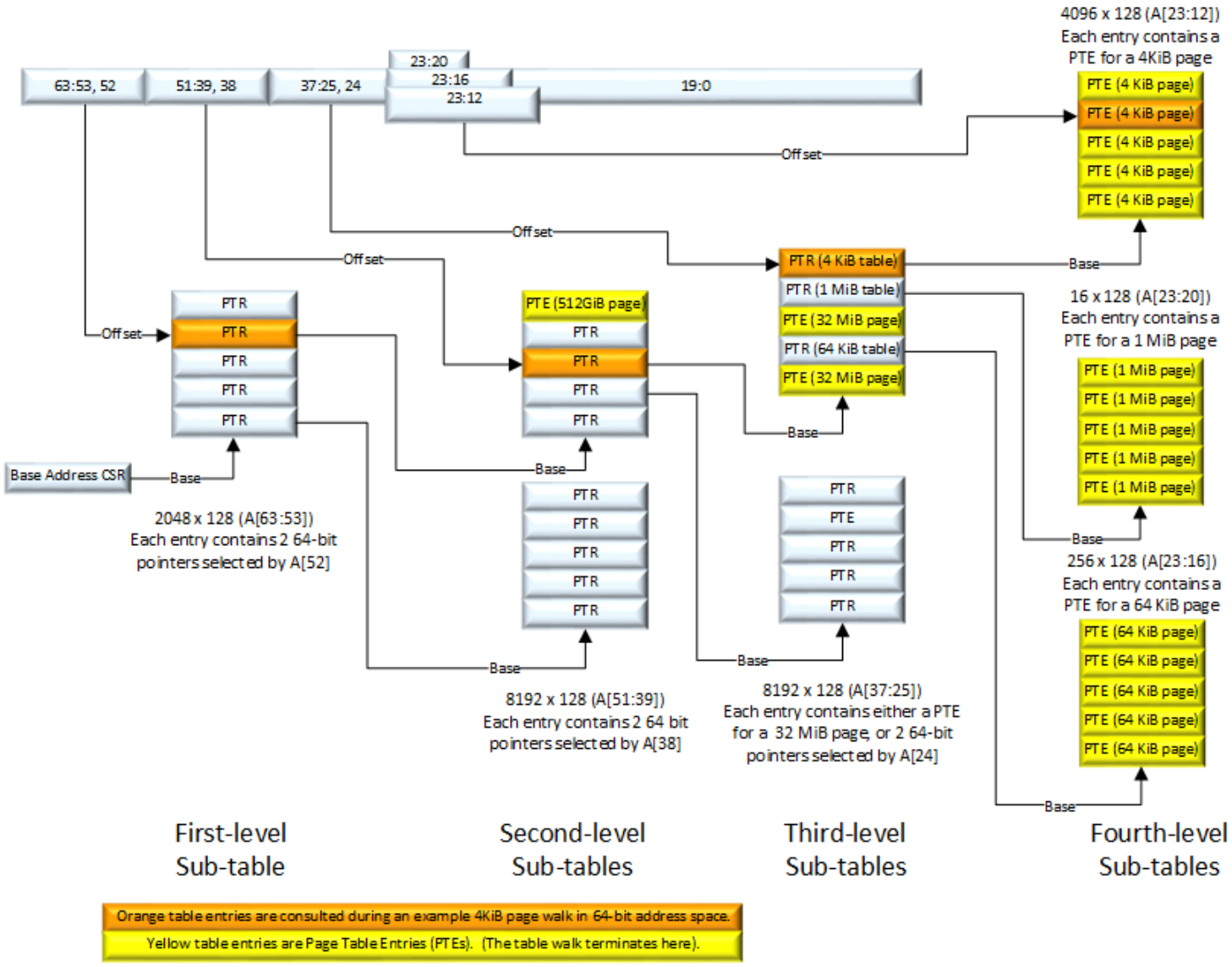
Example Component with Requester ZMMU



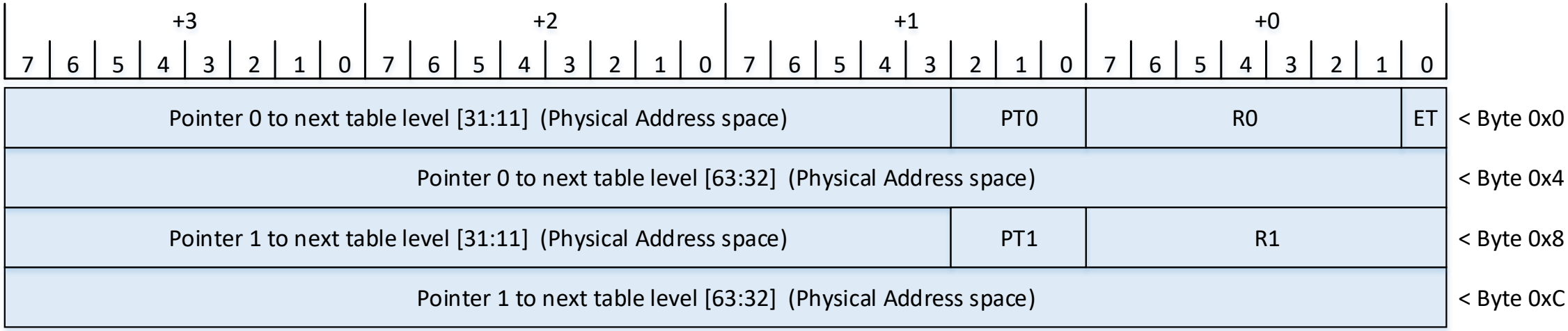
Example Component with Responder ZMMU



Page Table-based ZMMU Structure

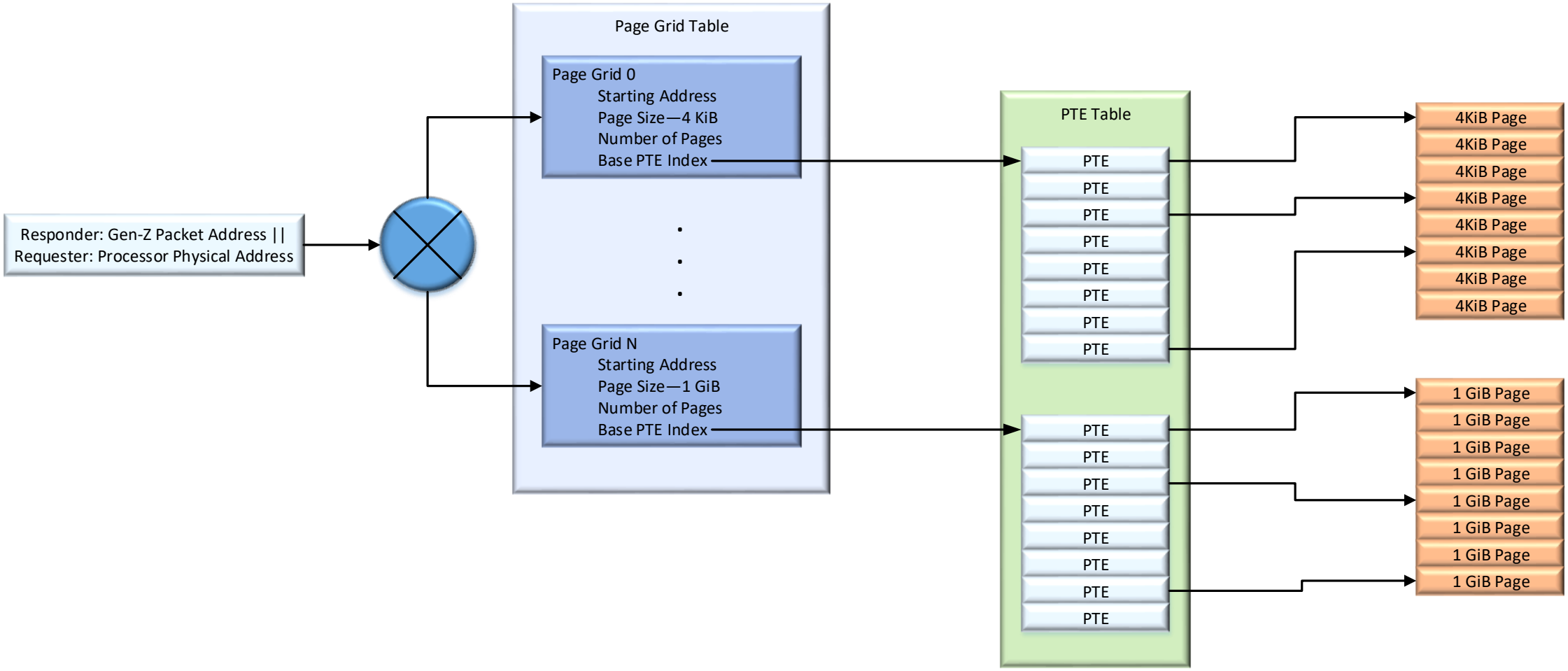


Page Table Pointer-Pair ZMMU Table Entry



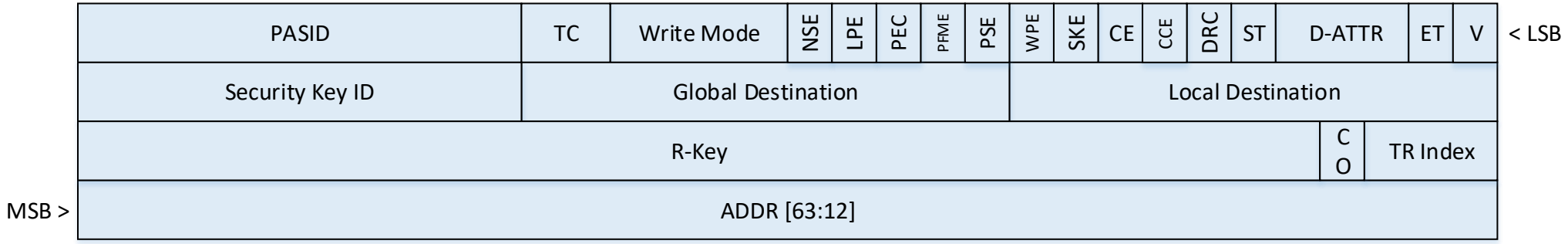
- This table entry is used to locate either another level of tables or a table of PTE entries
  - ET indicates if the table entry is valid
  - PT0 and PT1 indicate if the pointer is valid and to what is points

Page Grid-based ZMMU

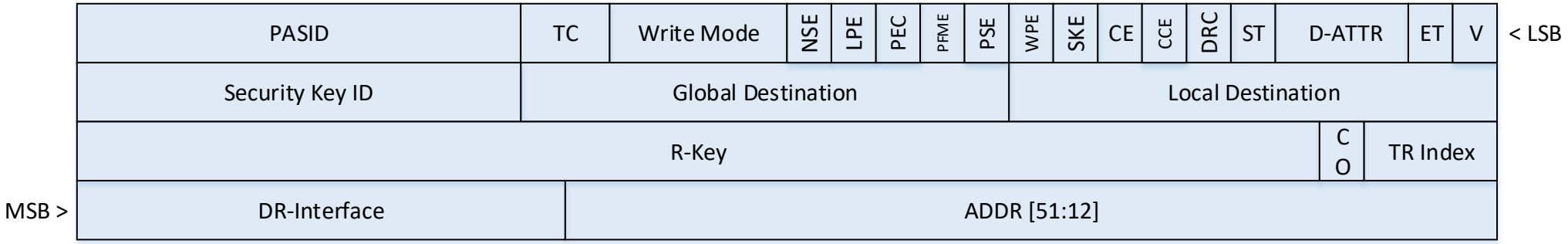


- A Page Grid is a highly-efficient alternative to a Page Table-based ZMMU that uses on-die resources instead of a hardware page table walker and caching resources to provide optimal performance

Requester PTE

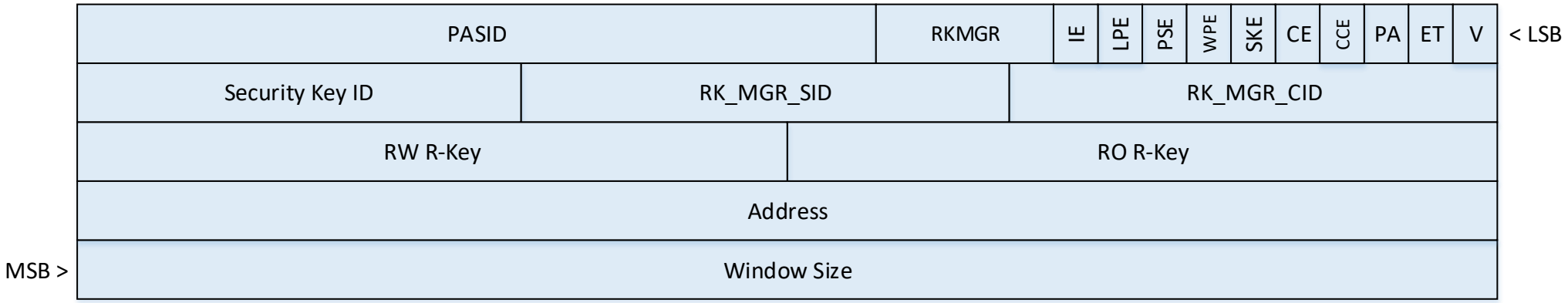


Data Space Requester PTE

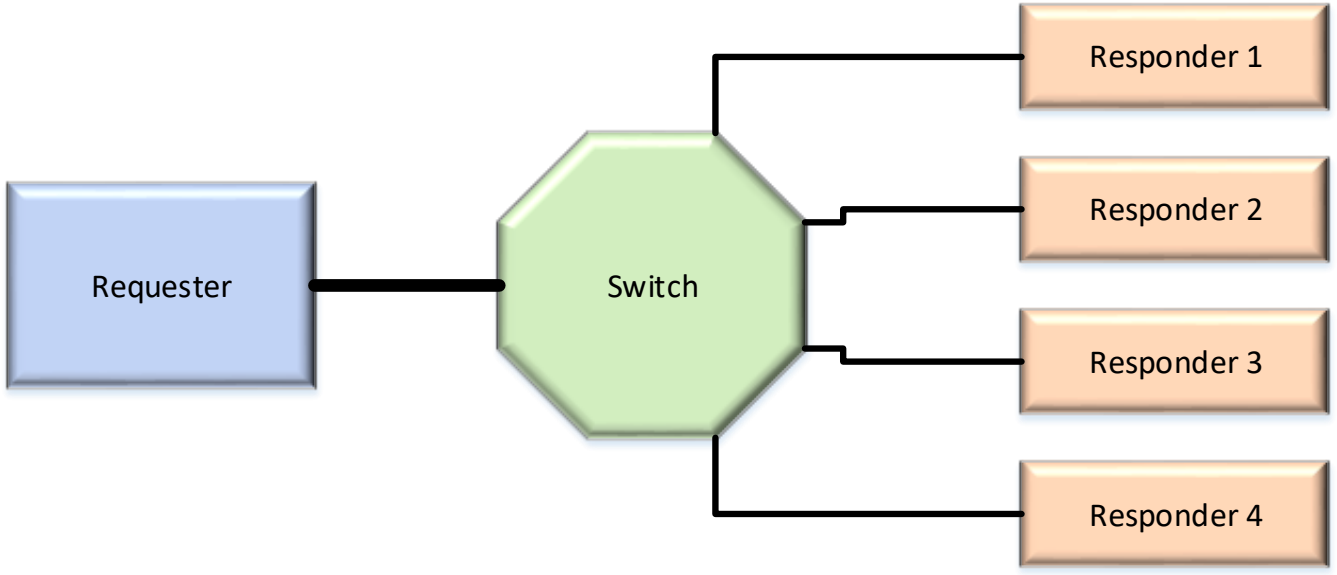


Control Space Requester PTE

Responder PTE

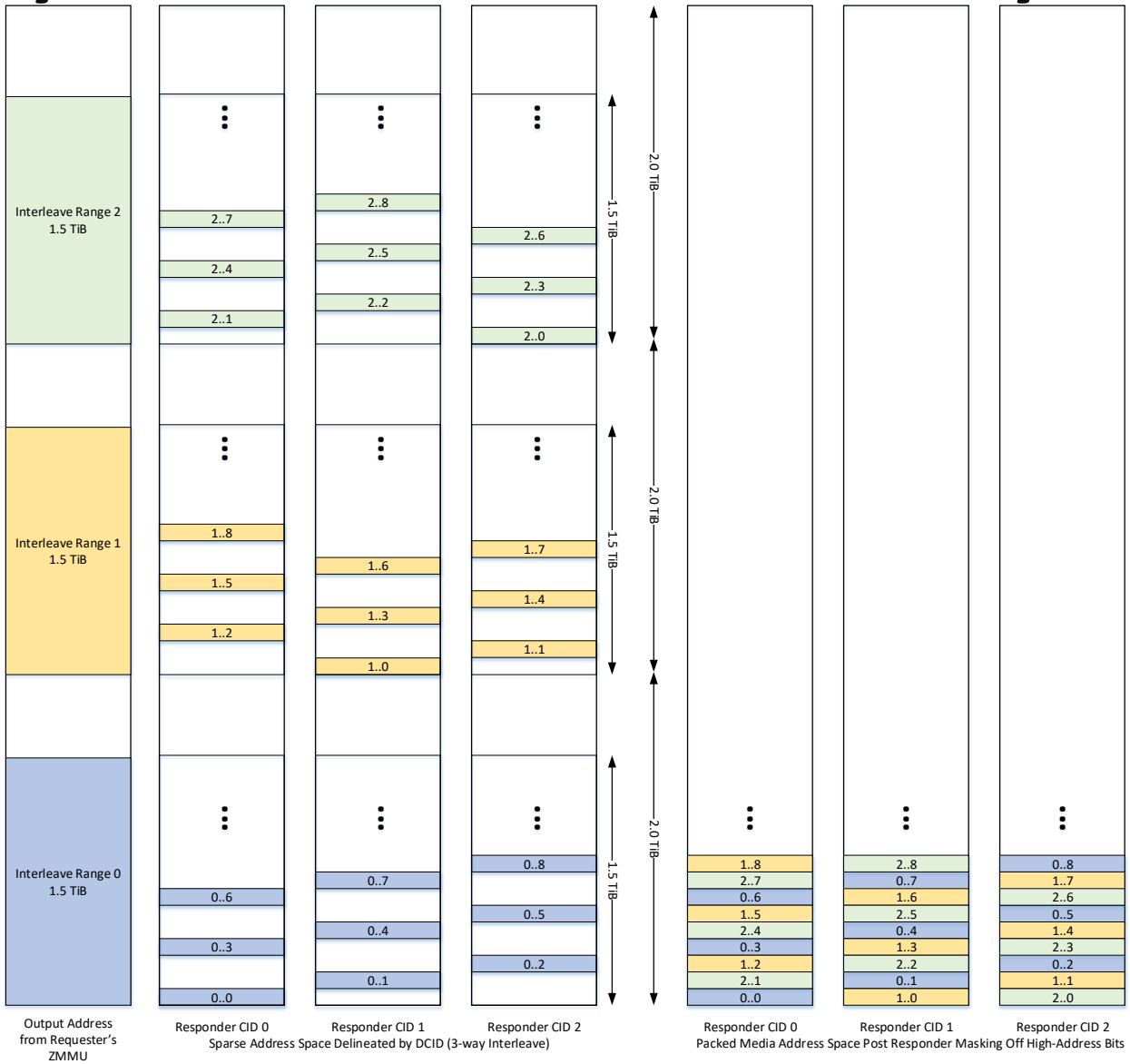


Memory Interleaving

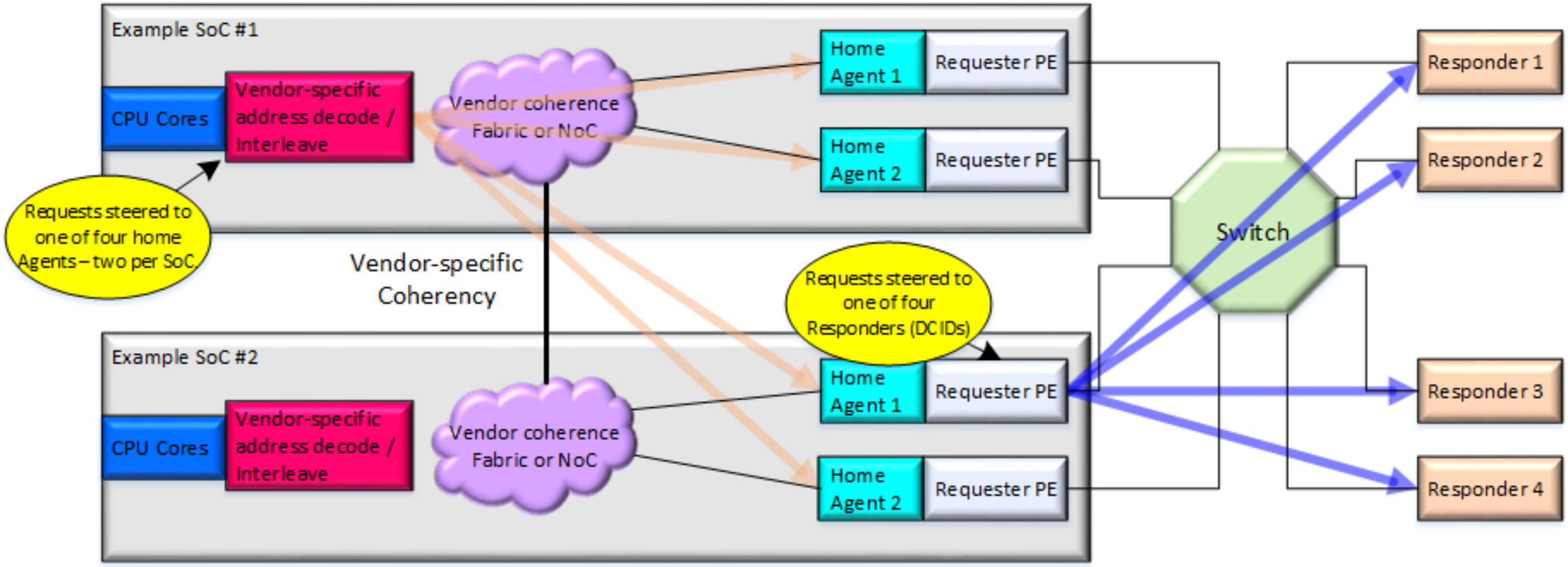


- Memory interleaving is used to improve aggregate bandwidth and latency by distributing memory across multiple components.
  - In this example, memory is distributed across four Responders providing up to 4x improvement in aggregate bandwidth
- Memory interleaving can be enhanced using RAID, erasure codes, etc. to distribute resiliency information across multiple Responders to improve data availability
- Gen-Z specifies a barber pole interleave scheme that supports up to 64-way interleaves

Example 3-way Stack Barber Pole Memory Interleave

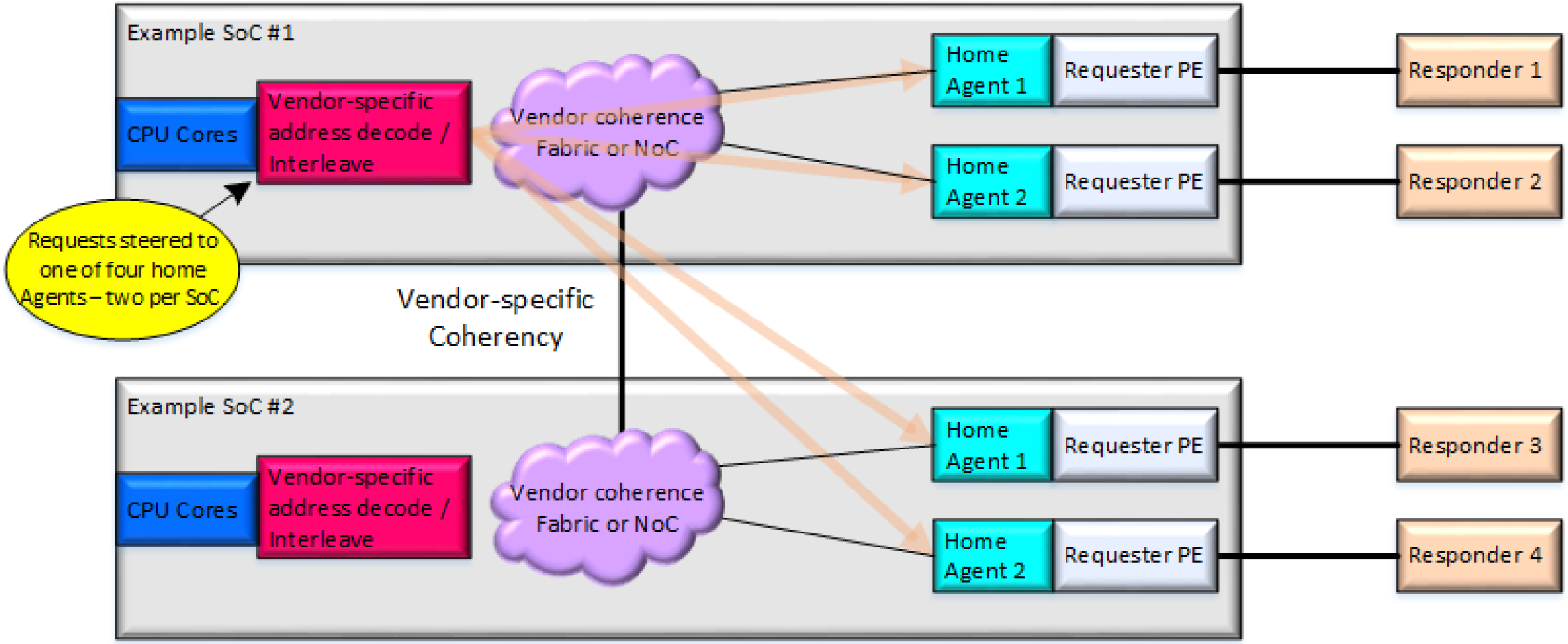


Component-specific and Fabric Memory Interleave

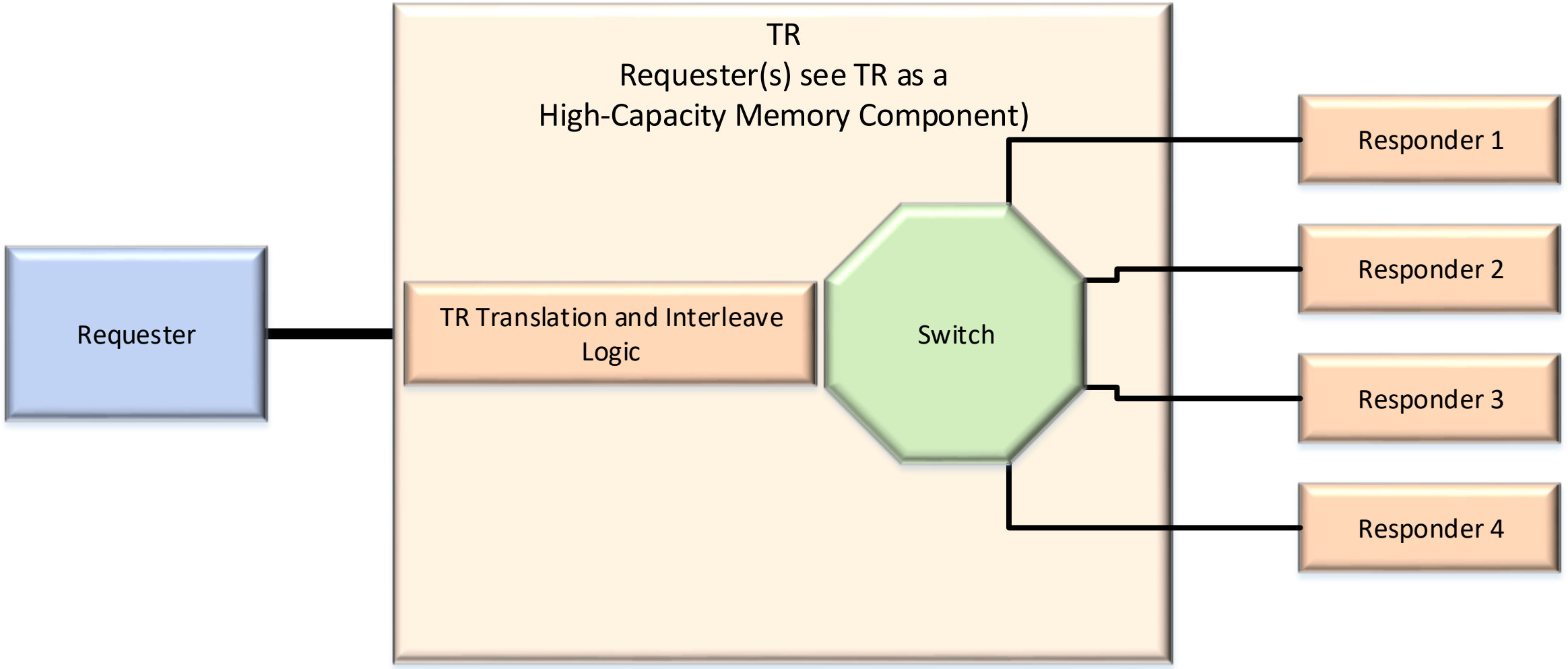


- A component such as a SoC may impose component-specific requirements and constraints on interleaves. For example, interleave between SoC home agents, vendor-specific coherency links, cache line slices, etc.

Vendor-specific P2P-Core Memory Interleave



Transparent Memory Interleave



Thank you