Gen-Z Atomics

April 2019

This presentation covers Gen-Z Atomic operations.

Disclaimer

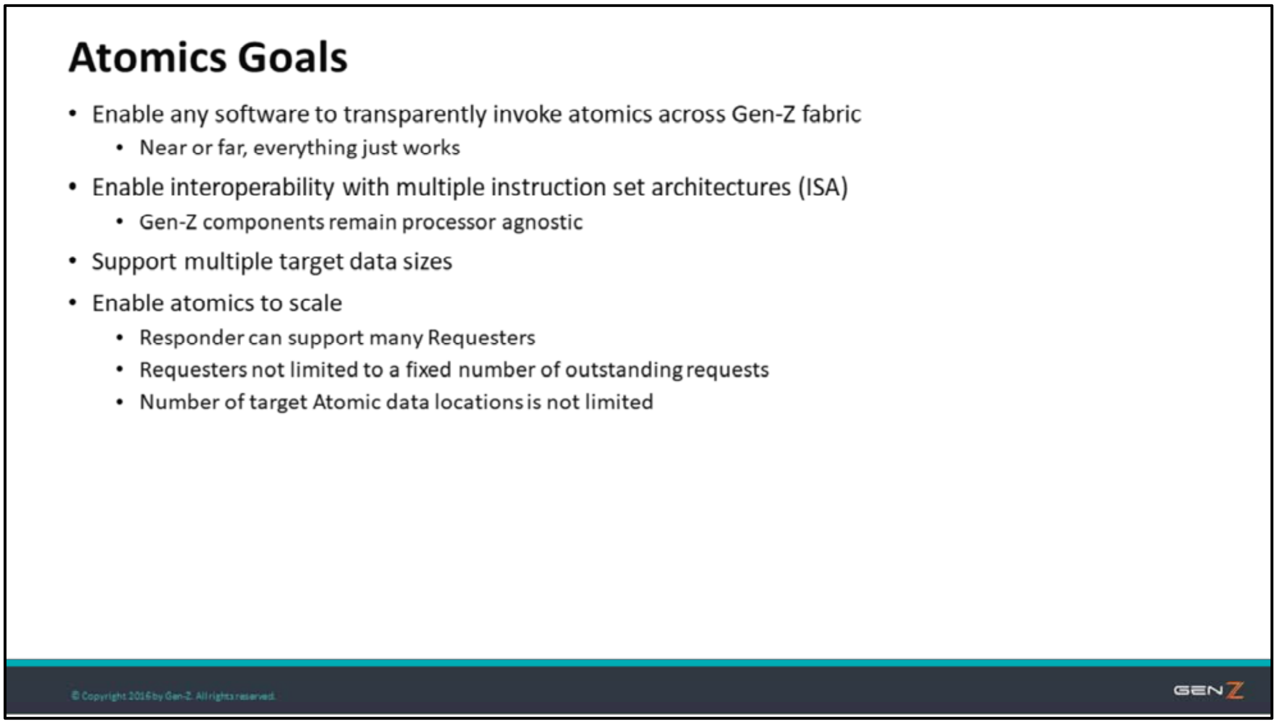
This document is provided 'as is' with no warranties whatsoever, including any warranty of merchantability, noninfringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Gen-Z Consortium disclaims all liability for infringement of 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/



Gen-Z architecture enables atomic operations to be executed on any component. Processor atomic operations can be transparently transported across a Gen-Z fabric to another component for execution. This enables software to seamlessly operate across Gen-Z, i.e., software compatibility is transparently provided.

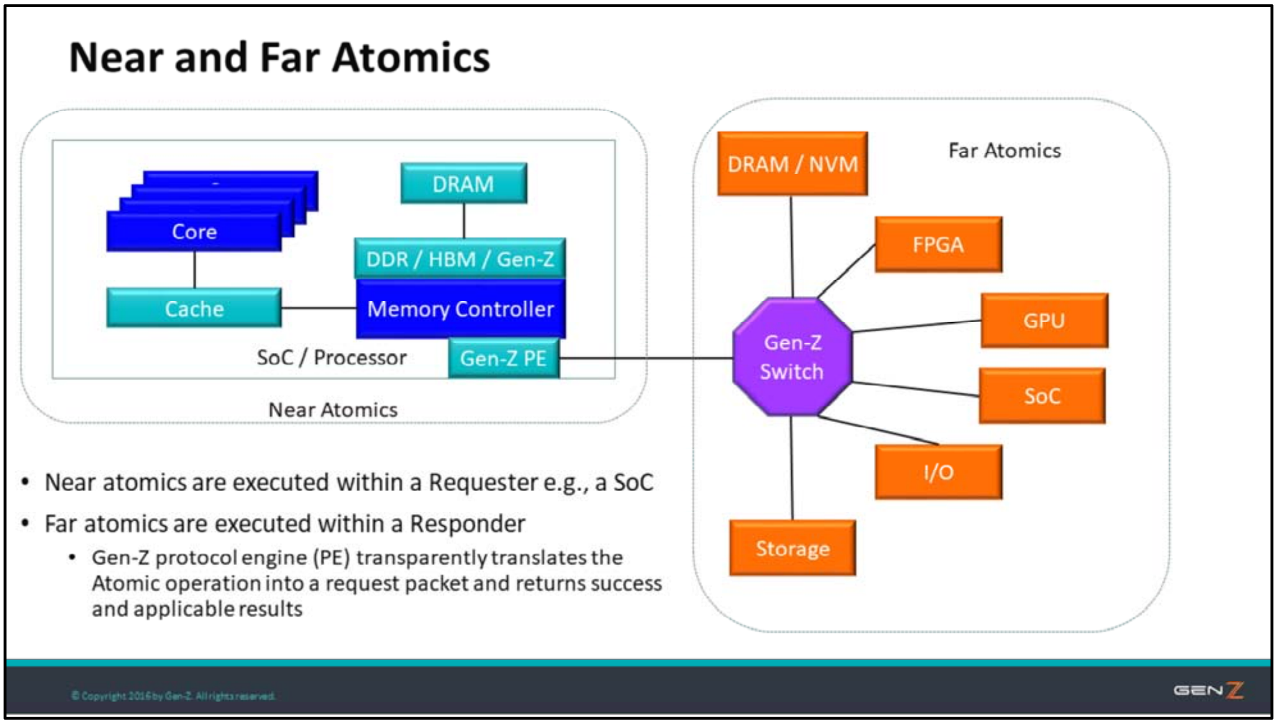
Gen-Z Atomics provide a complete set of operations capable of supporting atomics used by multiple processor ISAs including x86, ARM, and Power. This enables Gen-Z components to remain processor-agnostic and provide full software compatibility.

Gen-Z Atomics support multiple data sizes including: 8-bit, 16-bit, 32-bit, 64-bit, 128-bit, 256-bit, and 512-bit atomics.

Gen-Z Atomics can be scaled in multiple ways:

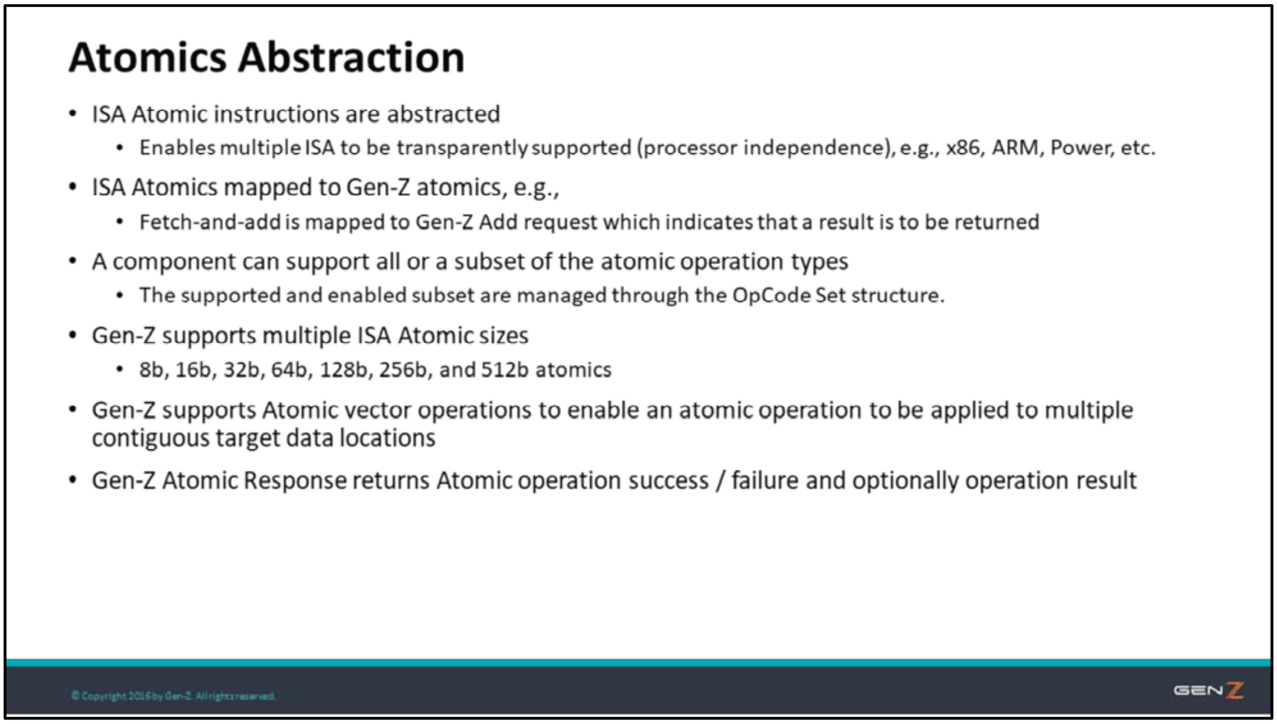
- A Responder can execute Atomics requests from multiple Requesters.
- Requesters are not limited to a fixed number of outstanding requests per Requester (as some technologies are). If a Responder supports 1000 Atomic operations, then one or more Requesters can take advantage of some or all of the supported operations, e.g., burst or random execution. Gen-Z's forward progress screens (FPS) ensure all Requesters can make forward progress.
- Though a Responder might support a limited number of outstanding Atomic requests,

this does not limit the number of memory locations that Atomic operations can be applied.



Near Atomics are operations executed within a processor. The target memory can be any directly-attached memory, e.g., DDR, HBM, or Gen-Z memory.

Far Atomics are operations executed outside of the processor. Software maps addressable resources in one or more Responders. Once mapped, a processor can execute an Atomic operation which is transparently transported across the Gen-Z fabric by a Gen-Z protocol engine (PE) to be executed by the Responder component. The Responder returns success or failure and applicable results. Any component type can initiate and / or execute Atomic operations.



Gen-Z Atomics abstract ISA-specific Atomic operations, e.g., a fetch-and-add atomic is translated into a Gen-Z Add operation with a flag to indicate the results are to be returned, i.e., fetched. Abstraction enables Gen-Z to support a variety of processor ISAs and to be easily extended / adapted to meet future needs.

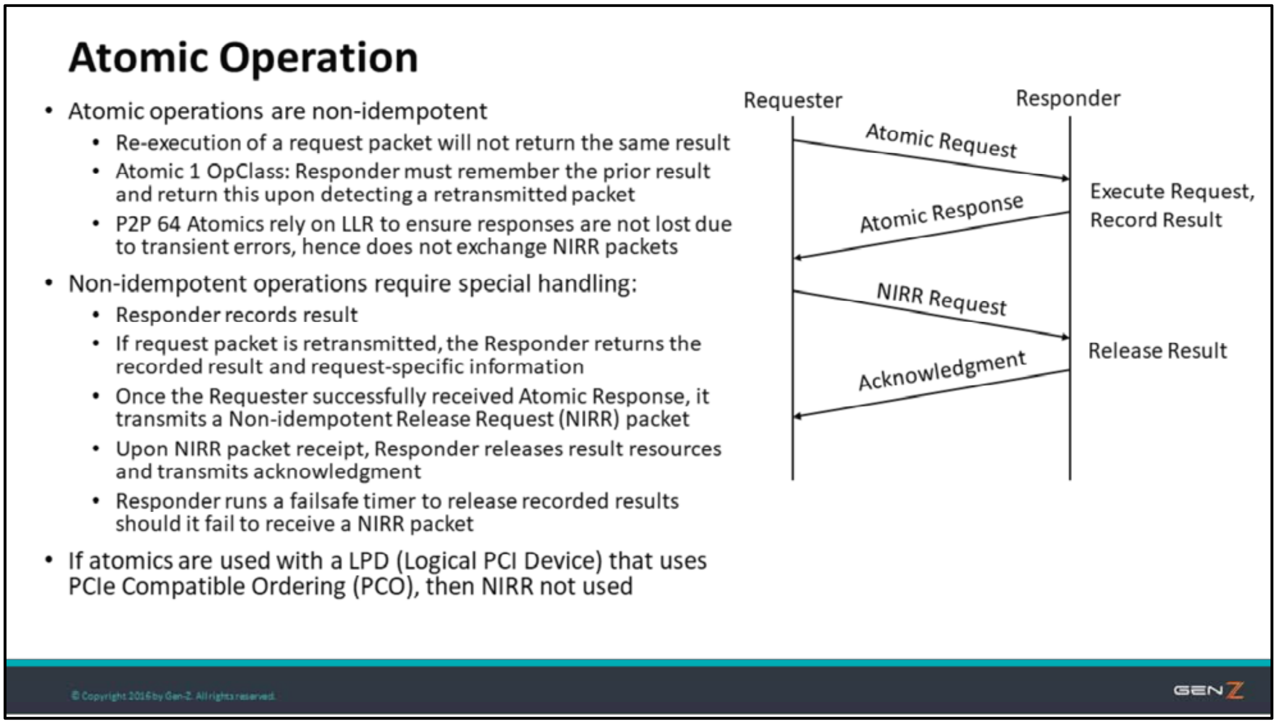
Gen-Z supports integer and floating point data types. Gen-Z supports comparison, arithmetic, logical, etc. operations.

Gen-Z Vector Atomics are used to apply an Atomic arithmetic or logic operation to a contiguous range of memory locations (up to 256 bytes of data). Conceptually, this is similar to a SIMD operation, e.g., application of 32 8b fetch-and-add operations.

Atomic Operation Types

- Gen-Z supports a wide range of Atomic operation types, e.g.:
  - Add
  - Sum
  - Swap
  - Compare-and-Swap (CAS)
  - CAS Not Equal
  - Logical (AND, OR, XOR)
  - Load Max
  - Load Min
  - Test-Zero-and-Modify
  - Increment Bounded
  - Increment Equal
  - Decrement Bounded
  - Compare-Store-Twin
  - Atomic Vector Sum
  - Atomic Vector Logical
  - Atomic Fetch
- Multiple ISA Atomics can be mapped onto a given Gen-Z Atomic operation type

Gen-Z supports a very wide-range of Atomic operations compared to other technologies which often are limited to just two. This ensures software compatibility and portability which simplifies solution development and amplifies the benefits of using Gen-Z technology.

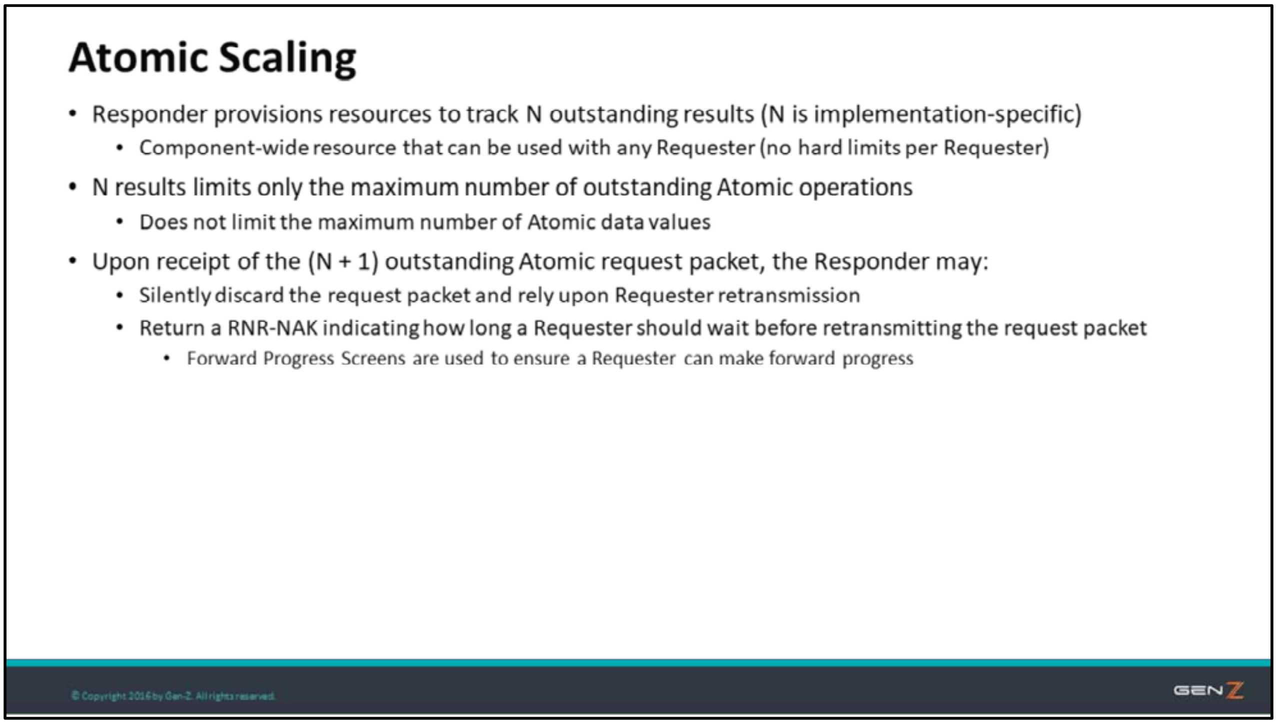


A non-idempotent operation is one where re-execution does not return the prior results, e.g., if a fetch-and-add operation is retransmitted and re-executed, then the result will differ due to the add being performed twice.

To ensure that a retransmission returns the prior results, a Responder maintains a non-addressable resource that contains the results of outstanding requests. A Responder does not execute an Atomic request unless it has a result slot available. Each result slot contains sufficient space to store the result, the Requester's identity, and the Tag; these values uniquely identify each request packet. Upon receipt of an Atomic request packet, the Responder verifies if it is a duplicate request. If not, it allocates a result slot and executes the request packet. If a duplicate, then it does not execute the request packet; instead, it returns the prior results.

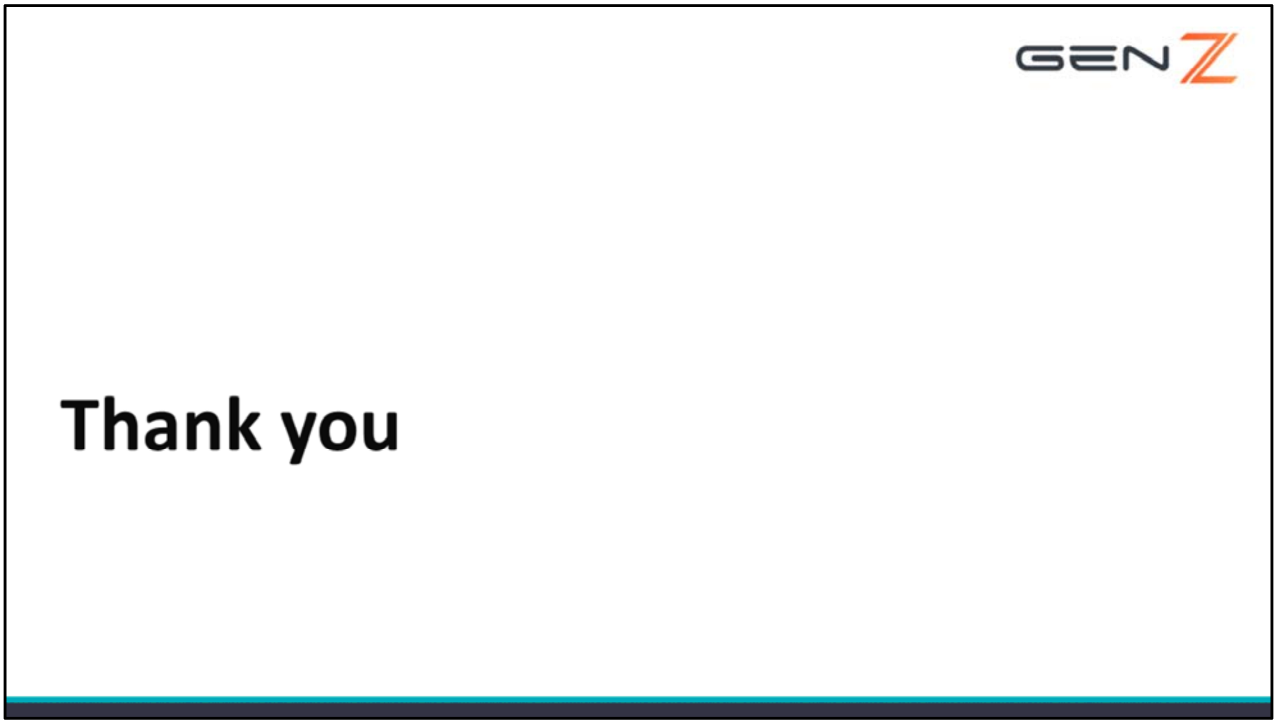
Once the Requester receives the response packet, it transmits a non-idempotent release request packet to the Responder. This is used to release the results slot at the Responder. To ensure result slots are eventually released in the event of hardware failure, the Responder maintains a failsafe timer.

Components that support LPDs that use PCIe PCO do not exchange NIRR. If the component is configured to use PCO, then communications are constrained to a single path that requires LLR (link-level reliability). This ensures that packets arrive in the order transmitted and any failures are raised as exceptions which handled as though operating over a native PCIe topology. Though LPDs may support PCO in order to re-use existing PCIe device drivers, designers are encouraged to not use PCO since the component will not be able to take full advantage of Gen-Z's architectural capabilities, e.g., multipath for aggregate performance and natural resiliency.



If a Responder receives more Atomic request packets than it has result resources, then it can take one of two actions:

- It can silently discard the request packet and rely upon the Requester to retransmit the packet. Gen-Z support packet Deadline semantics which enable reasonably aggressive retransmission timers. Though simple and effective, the primary issue with this approach is it does not guarantee the Requester will make forward progress.
- Alternatively, a Responder can return a Responder-Not-Ready Negative Acknowledgment (RNR-NAK). This packet indicates how long a Requester should wait before retransmitting a request packet (could be immediate in which case, this might be less than the retransmission time, or could be longer due to heavy load). The Reason in an RNR-NAK indicates which forward progress screen (FPS) epoch that the Requester is now associated. When the Requester retransmits the Atomic request, it includes the epoch. The epoch acts as an input to the Responder for it to determine how to prioritize this request or to calculate a new wait value in order to ensure the Requester makes forward progress.



This concludes this presentation. Thank you.