This presentation gives an overview of Logical PCIe Devices (LPDs). This optional LPD mechanism is a critical feature for Gen-Z that enables native Gen-Z I/O components to present themselves as one or more PCIe devices to a host system running an unmodified OS. Moreover, LPDs on Gen-Z go far beyond PCIe capabilities, creating immediate compelling I/O solutions with Gen-Z.

A related presentation on PCIe-Compatible Ordering (PCO) for LPDs covers in detail how PCO works.
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/
This drawing depicts an example Gen-Z system that contains both Gen-Z and PCIe® I/O components. Such I/O components might include HDDs, SSDs, HBAs, NICs, cluster interconnects, GPUs, computational accelerators, etc.

Note that Gen-Z supports advanced fabric topologies beyond simple tree-based ones, providing increased bandwidth, scalability, and availability. Bandwidth increases can come via higher link signaling rates, multiple aggregated links, or both. Scalability increases can come from larger fabrics with advanced topologies and congestion management. Availability increases can come from multipath link failover, end-to-end retry, and more precise error containment.

In 2001, the launch of PCIe was eased and accelerated dramatically by PCIe devices and switches presenting themselves to software as conventional PCIe devices and bridges, allowing the new PCIe ecosystem to be supported by unmodified OSs. Over time, OSs added support for advanced PCIe features.

Gen-Z is using a similar strategy with LPDs. LPDs enable immediate support of Gen-Z I/O components by unmodified OSs. Over time, OSs can add support for certain advanced Gen-Z features, though LPDs can immediately take advantage of many advanced Gen-Z features without waiting for OS changes.
Though LPDs emulate PCIe Root Complex Integrated Endpoints (RCIEPs), Gen-Z components with LPDs can fully exploit Gen-Z’s revolutionary fabric and architecture.

Much of this is facilitated by Gen-Z-aware system firmware discovering and configuring local Gen-Z components before the OS boots. Some of this is enabled by Gen-Z “PECAM” logic (covered on a subsequent slide) presenting LPDs as being directly connected to the host, and not making the Gen-Z fabric visible to an unmodified OS.

Many performance benefits come from Gen-Z links supporting higher signaling rates, Gen-Z switches inherently providing lower latency through their architecture, and Gen-Z fabrics supporting advanced topologies with link aggregation and dynamic congestion management.

Future performance benefits will come from applications and OS infrastructure becoming Gen-Z aware and utilizing new software paradigms or hardware primitives supported by Gen-Z. Examples include load/store access to storage and use of processor-specific atomic operations.
Gen-Z’s PECAM mechanism is compatible with PCIe’s standard ECAM mechanism, which supports device/function discovery, enumeration, and configuration. Mentioned earlier, a PECAM presents LPDs as being directly attached to the host, regardless of their actual Gen-Z fabric connections. This avoids an unmodified OS from attempting to manage a Gen-Z fabric, which the unmodified OS doesn’t comprehend.

MMIO apertures are a clever use of existing ACPI mechanisms to constrain MMIO (memory-mapped I/O) mappings within specified ranges (“apertures”). This enables Gen-Z hardware to tolerate if (unmodified) OSs reprogram LPD function Base Address registers (BARs) from the values configured earlier by system firmware.

PCIe and Gen-Z have fundamentally different ordering models. Notably, PCIe requires its fabric to do ordering enforcement while Gen-Z does not, which enables Gen-Z fabrics to perform substantially better, taking advantage of multipath, congestion avoidance, and reduced head-of-line blocking. PCO is an optional feature for LPDs that enables PCIe driver and infrastructure software to work without changes to accommodate Gen-Z’s different ordering model. PCO basically works by configuring the Gen-Z fabric to deliver all PCO packets between an LPD and its host in order. Due to fundamental fabric ordering differences, PCO uses totally different and innovative approaches to guarantee forward progress when avoiding deadlock.
A Gen-Z component that supports LPDs implements a Component LPD structure for each LPD. One Gen-Z component can support thousands of LPDs. Each LPD function has a 4096-byte region in Gen-Z Control Space that looks like PCI/PCIe Config Space.

Logical PCIe Configuration Space content:
- Type 0 Configuration Space Header
  - BARs that map the LPD function’s run-time CSRs
  - Base Address for optional Expansion ROM
- Interrupt controls (in MSI or MSI-X capability)
- Optional PCI/PCIe Capability structures; e.g., PCIe Capability, SR-IOV, ATS, etc
  - Page Request Group (PRG) services for LPDs is enhanced to provide better paging controls

System firmware uses the Gen-Z Requester ZMMU to map PCIe Configuration Space for the PECAM. Similarly, system firmware uses the Gen-Z Requester ZMMU to map PCI Memory Space for the function’s run-time CSRs.

How do LPDs work?

- A Gen-Z component that supports LPDs implements a Component LPD structure for each LPD.
- One Gen-Z component can support thousands of LPDs.
- Each LPD function has a 4096-byte region in Gen-Z Control Space that looks like PCI/PCIe Config Space.

Each function contains a 4096-byte region that looks like PCI/PCIe Configuration Space to the host system. Using its PECAM, the host reads & writes the function’s Configuration Space to discover its attributes, bind a suitable driver to it, and configure its essential resources like MMIO space, interrupts, and advanced PCIe capabilities.
Some functionality associated with advanced PCIe capabilities requires information not carried by normal Gen-Z transactions. LPDs supporting these advanced PCIe capabilities use special versions of Gen-Z transactions that carry the required additional information. For example, LPDs that support PCIe Address Translation Services (ATS) must use Gen-Z transactions that carry the TA field. The slide gives other key examples of additional information and how it is used.

Gen-Z architects a much broader superset of the atomics supported by PCIe, and LPDs can use any atomics from that superset.

Gen-Z transactions carrying this additional information are still highly efficient, native Gen-Z transactions. For cases where the LPD isn’t using functionality that requires the additional information, the LPD can use Gen-Z transactions without it, for even greater efficiency.
LPDs support massive scaling!

Bus/device/function tuples (BDFs) mapped by PECAMs are virtual, allowing a single Gen-Z component to support an arbitrary number of virtual devices, consuming an arbitrary number of virtual busses. This enables a single Gen-Z component to support up to thousands of LPDs, and each LPD can contain up to 256 functions.

Bus and device numbers assigned for LPDs within a Gen-Z component don’t need to be consecutive or even monotonically increasing. This avoids the need for OSs to ever “rebalance” bus numbers during run time, which many OSs do not support.

PECAMs present no bridges or switches, so bridges/switches consume no virtual bus numbers.
Gen-Z fabrics can readily connect multiple systems, and LPDs are able to be shared by multiple host systems, given suitable Gen-Z fabric management software.

Component sharing provides significant benefits for availability. For example, if multiple systems are providing a service and one of the systems fails, the remaining systems can take over the failed system’s I/O components, allowing continued operation of the service.

OSs do not need to be aware that they are sharing a common Gen-Z I/O component. Each system running an unmodified OS will only “see” the LPDs assigned to it by Gen-Z fabric management software and mapped by its PECAM, so there’s no problem with unmodified OSs attempting to claim the same logical devices.

I/O component sharing does not mean a loss of security. Host systems and LPDs can be configured to use R-Keys, ensuring that host systems can access only the LPDs that they own within a shared I/O component, and LPDs can access only the host systems that own and control them.

As shown in the figure, the BDF space for each host system is orthogonal to the BDF spaces in other host systems. A single component can have the same bus/device number tuple assigned to multiple LPDs, as long as each LPD belongs to a different host system.
Here are the key takeaways for LPDs, each with their associated benefit.
Thank you