Logical PCI Devices Overview

Introduction/Motivation

Gen-Z as a new data access technology offers a number of compelling benefits, including higher bandwidth, decreased latency, consolidated fabrics, extreme scalability, fabric-attached memory, and data-centric computing. In order for Gen-Z to deliver these capabilities, there must be a transition plan for software and Operating Systems (OSs). To enable and accelerate migration from PCI-based systems, PCI Express® (PCIe®) achieved remarkable success by implementing key hardware features that enabled unmodified OSs to discover and configure PCIe devices and switches as if they were PCI devices and bridges. Over time, OSs evolved to take advantage of PCIe's many advanced features, but PCIe's compatibility with basic OS infrastructure software enabled immediate system migration to PCIe. Gen-Z is using a similar strategy via the concept of *logical PCI devices*.

Mixture of Gen-Z and PCIe

Many Gen-Z systems will contain a mix of Gen-Z I/O components and PCIe I/O devices, as illustrated below.

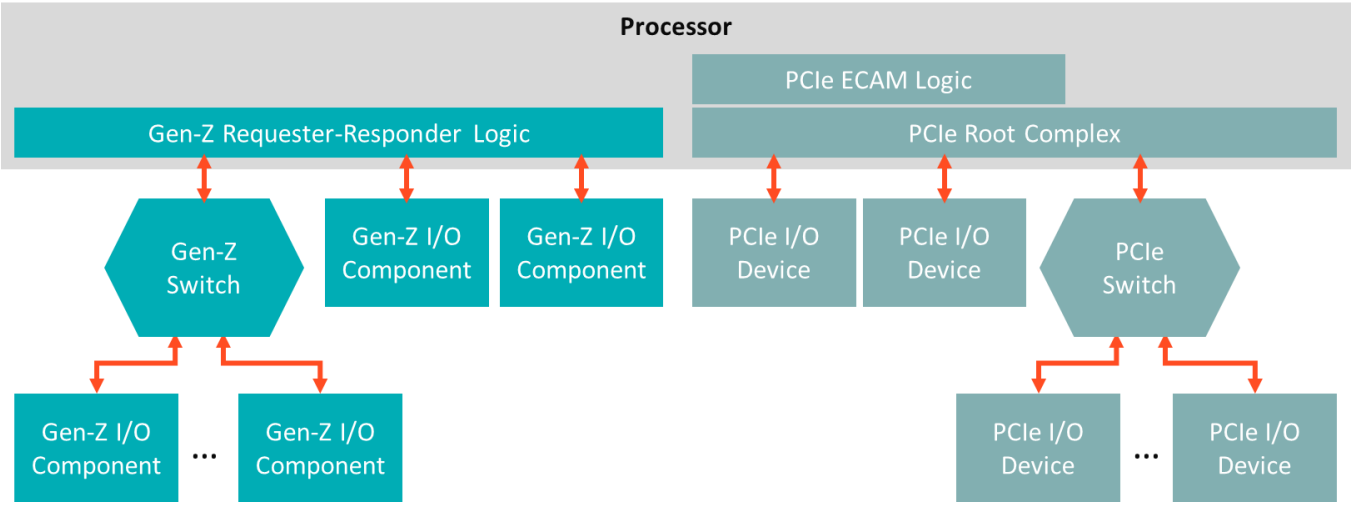

Figure 1: Example Gen-Z System with Gen-Z I/O Components and PCIe I/O Devices

The standard PCIe ECAM[1] hardware provides memory-mapped access to PCI Configuration Space, enabling OSs to discover and configure PCIe devices and switches[2]. Notably, OSs use the standard PCI Configuration Space header and capability structures for each PCIe device to determine its device class, map its run-time control and status registers, bind an appropriate device driver, and configure its interrupts.

Gen-Z supports a new PECAM[3] hardware mechanism to enable unmodified OSs to discover and configure suitable Gen-Z I/O components as if they were PCI devices. The same OS PCI infrastructure software handles both PCI and PCIe devices. In fact, some I/O devices integrated by modern SoCs are PCI devices instead of PCIe devices, since they may be simpler and provide more cost-effective functionality when integrated within the processor.

[1] ECAM: Enhanced Configuration Access Method, defined in the PCIe Base specification
[2] Modern OSs also use the ECAM to discover and configure PCI devices and bridges.
[3] PECAM: PCI Enhanced Configuration Access Method, defined in the Gen-Z Core specification

Logical PCI Devices

For suitable Gen-Z I/O components, the PECAM presents a view of logical PCI devices as shown in Figure 2.

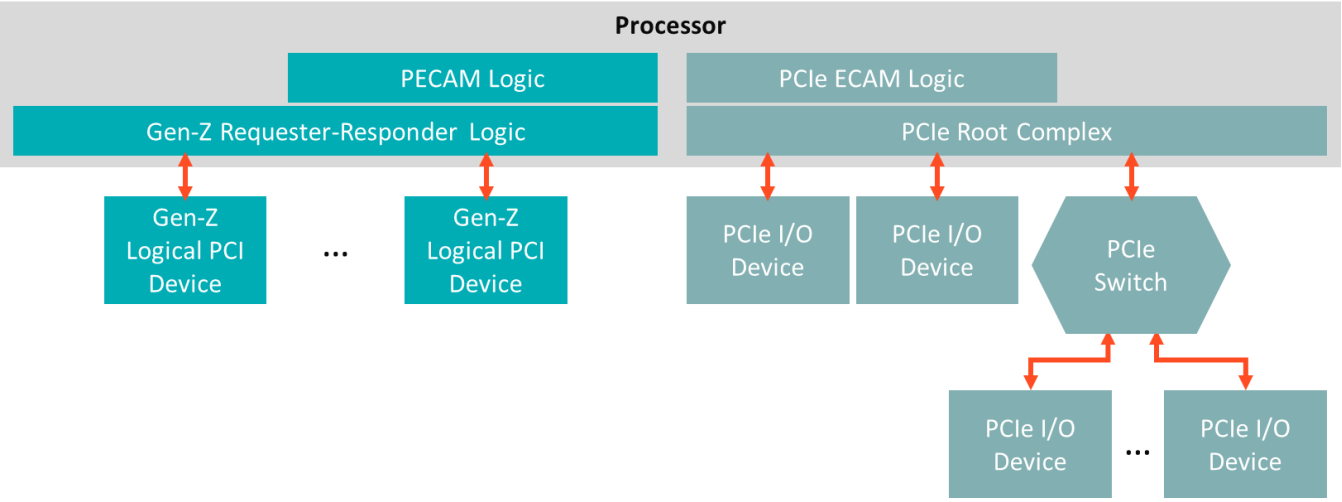

Figure 2: Example Gen-Z System Showing PECAM View of Gen-Z Logical PCI Devices

Since Gen-Z fabrics support advanced topologies and are capable of connecting multiple systems, the PECAM presents a simplified topology view for the logical PCI devices, not showing any Gen-Z switches that might be present. The Gen-Z PECAM hardware may be implemented as a part of the PCIe ECAM hardware or may be implemented separately.

Each Gen-Z I/O component designed to appear as logical PCI device implements a Gen-Z Component PCI structure[4], which among other things, contains a PCI Configuration Space header and one or more PCI capability structures. At power-up, system firmware configures the local Gen-Z fabric (if present) and scans for Gen-Z I/O components that contain a Component PCI structure. For each suitable Gen-Z I/O component, system firmware creates one or more entries in the PECAM, each corresponding to a virtual PCI bus/device/function (BDF) routing tuple[5]. System firmware also maps each logical PCI device's run-time control and status registers.

After the OS boots, it discovers each logical PCI device represented in the PECAM, observing that the device's run-time control and status registers have already been mapped by system firmware. The OS then selects and binds an appropriate driver, using the PCI device class, Vendor ID, and other information contained in the PCI Configuration Space header. When the driver initializes, it uses OS services to configure the device for interrupts, using an MSI or MSI-X capability[6] structure.

Evolving PCIe Devices into Gen-Z Components

A wide range of PCIe devices are good candidates for evolving into Gen-Z I/O components and being represented as logical PCI devices on Gen-Z systems. Notable examples are those devices needing increased bandwidth, higher scalability, and/or increased resiliency. Gen-Z supports multiple signaling rates (16, 25, 28, 56, and 112 GT/s) on each lane over copper and optical media. Common link widths will be 1, 2, 4, 8, and 16 lanes. This enables Gen-Z to scale in bandwidth from 10s to 100s of GB/s, covering a wide range of I/O solutions.

Gen-Z can address up to 4096 components per subnet, scaling far beyond device counts achievable with traditional PCIe systems. Further, Gen-Z decouples virtual device instances from component addressing, allowing a single Gen-Z component to support up to thousands of virtual devices (e.g., for direct assignment to virtual machines) without being constrained by

[4] A Gen-Z Control Space structure defined in the Gen-Z Core specification
[5] Gen-Z does not use PCI bus, device, and function numbers for actual routing.
[6] MSI & MSI-X are capabilities supporting message-signaled interrupts, defined in the PCI Local Bus specification

addressing units like PCI function numbers. Assignment of logical PCI devices by system firmware also enables unmodified OSs to support Gen-Z I/O component sharing by multiple systems without incurring the cost and complexity of PCIe MR-IOV[7], which was never widely adopted by the industry.

The Gen-Z PECAM's use of virtual BDFs and its simplified device topology view avoids the inefficient and unnecessary consumption of virtual BDFs for interconnect switch ports. This avoids scaling barriers that exist on some PCIe systems for supporting high device counts.

The use of virtual BDFs also enables increased flexibility with hot plug. Whereas some PCIe systems may require the OS to "rebalance" bus number assignments when adding additional PCIe switches or SR-IOV[8] devices that consume multiple bus numbers, the PECAM can readily and non-obtrusively handle hot plug additions or deletions without requiring the unmodified OS to rebalance bus number assignments, a capability that was never widely supported by OSs.

As I/O fabric scaling increases, so does the importance of resiliency against errors. Besides automatic recovery from most transient link errors, Gen-Z limits the impact of most uncorrectable I/O transaction errors to the single device that initiates the transaction; other devices can generally continue their normal operation without impact. With PCIe, an uncorrectable I/O transaction will often require shutting down a portion of a PCIe tree hierarchy or even an entire tree hierarchy in some cases, in order to contain the error and avoid data corruption.

Gen-Z Solid-State Disk (SSD) components using SCSI or NVMe-like protocols are excellent candidates for logical PCI devices, taking immediate advantage of Gen-Z's increased bandwidth, massive scalability, increased hot plug flexibility, and increased resiliency. SSD block storage protocols will work well over Gen-Z[9], and existing PCIe SSD drivers are highly leverageable for creating Gen-Z SSD drivers for these logical PCI devices.

Other excellent candidates for logical PCI devices include Gen-Z Virtual Network Interface Controllers (vNICs), which can be mapped into an unmodified OS and unmodified IP network stack, enabling traditional network applications to operate transparently across a Gen-Z topology. For high-speed, low-latency communications, a Gen-Z provider library can be mapped underneath the OpenFabrics OFI Libfabric infrastructure. This enables a broad set of advanced messaging applications to operate across Gen-Z, e.g., MPI, SHMEM, Sockets, and many more.

Summary

Using the approach of logical PCI devices, Gen-Z is well positioned for immediate support by unmodified OSs, following the successful example of PCIe. Important classes of I/O devices can take immediate advantage of key Gen-Z benefits (bandwidth/scalability/resiliency) to deliver compelling I/O solutions. Even advanced functionality like I/O sharing between multiple systems can be handled by unmodified OSs with reduced cost and complexity.

[7] MR-IOV: Multi-Root I/O Virtualization, defined by the Multi-Root I/O Virtualization and Sharing specification
[8] SR-IOV: Single-Root I/O Virtualization, defined by the Single Root I/O Virtualization and Sharing specification
[9] Covered in detail by the Gen-Z *Block Storage Overview* paper

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.