OpenCAPI (Open Coherent Accelerator Processor Interface)

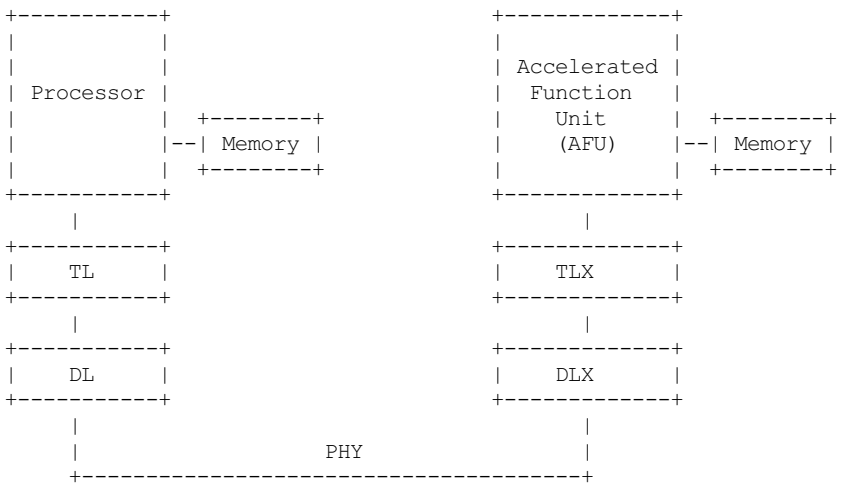
OpenCAPI is an interface between processors and accelerators. It aims at being low-latency and high-bandwidth. The specification is developed by the OpenCAPI Consortium.

It allows an accelerator (which could be an FPGA, ASICs, ...) to access the host memory coherently, using virtual addresses. An OpenCAPI device can also host its own memory, that can be accessed from the host.

OpenCAPI is known in linux as 'ocxl', as the open, processor-agnostic evolution of 'cxl' (the driver for the IBM CAPI interface for powerpc), which was named that way to avoid confusion with the ISDN CAPI subsystem.

High-level view

OpenCAPI defines a Data Link Layer (DL) and Transaction Layer (TL), to be implemented on top of a physical link. Any processor or device implementing the DL and TL can start sharing memory.



Device discovery

OpenCAPI relies on a PCI-like configuration space, implemented on the device. So the host can discover AFUs by querying the config space.

OpenCAPI devices in Linux are treated like PCI devices (with a few caveats). The firmware is expected to abstract the hardware as if it was a PCI link. A lot of the existing PCI infrastructure is reused: devices are scanned and BARs are assigned during the standard PCI enumeration. Commands like 'lspci' can therefore be used to see what devices are available.

The configuration space defines the AFU(s) that can be found on the physical adapter, such as its name, how many memory contexts it can work with, the size of its MMIO areas, ...

MMIO

OpenCAPI defines two MMIO areas for each AFU:

- the global MMIO area, with registers pertinent to the whole AFU.
- a per-process MMIO area, which has a fixed size for each context.

AFU interrupts

OpenCAPI includes the possibility for an AFU to send an interrupt to a host process. It is done through a 'intrp_req' defined in the Transaction Layer, specifying a 64-bit object handle which defines the interrupt.

The driver allows a process to allocate an interrupt and obtain its 64-bit object handle, that can be passed to the AFU.

char devices

The driver creates one char device per AFU found on the physical device. A physical device may have multiple functions and each function can have multiple AFUs. At the time of this writing though, it has only been tested with devices exporting only one AFU.

Char devices can be found in /dev/ocxl/ and are named as: /dev/ocxl/<AFU name>.<location>.<index>

where <AFU name> is a max 20-character long name, as found in the config space of the AFU. <location> is added by the driver and can help distinguish devices when a system has more than one instance of the same OpenCAPI device. <index> is also to help distinguish AFUs in the unlikely case where a device carries multiple copies of the same AFU.

Sysfs class

An ocxl class is added for the devices representing the AFUs. See /sys/class/ocxl. The layout is described in Documentation/ABI/testing/sysfs-class-ocxl

User API

open

Based on the AFU definition found in the config space, an AFU may support working with more than one memory context, in which case the associated char device may be opened multiple times by different processes.

ioctl

OCXL_IOCTL_ATTACH:

Attach the memory context of the calling process to the AFU so that the AFU can access its memory.

OCXL_IOCTL_IRQ_ALLOC:

Allocate an AFU interrupt and return an identifier.

OCXL_IOCTL_IRQ_FREE:

Free a previously allocated AFU interrupt.

OCXL_IOCTL_IRQ_SET_FD:

Associate an event fd to an AFU interrupt so that the user process can be notified when the AFU sends an interrupt.

OCXL_IOCTL_GET_METADATA:

Obtains configuration information from the card, such at the size of MMIO areas, the AFU version, and the PASID for the current context.

OCXL_IOCTL_ENABLE_P9_WAIT:

Allows the AFU to wake a userspace thread executing 'wait'. Returns information to userspace to allow it to configure the AFU. Note that this is only available on POWER9.

OCXL_IOCTL_GET_FEATURES:

Reports on which CPU features that affect OpenCAPI are usable from userspace.

mmap

A process can mmap the per-process MMIO area for interactions with the AFU.