OpenPOWER™

A software developer's introduction to OpenCAPI

Andrew Donnellan, Software Engineer

IBM Linux Technology Centre

OpenCAPI

Open Coherent Accelerator Processor Interface

Coherent
High bandwidth
Low latency
Designed for FPGAs
Open standard
Available today on POWER9

skiboot

github.com/open-power/skiboot
(hw/npu2-opencapi.c)

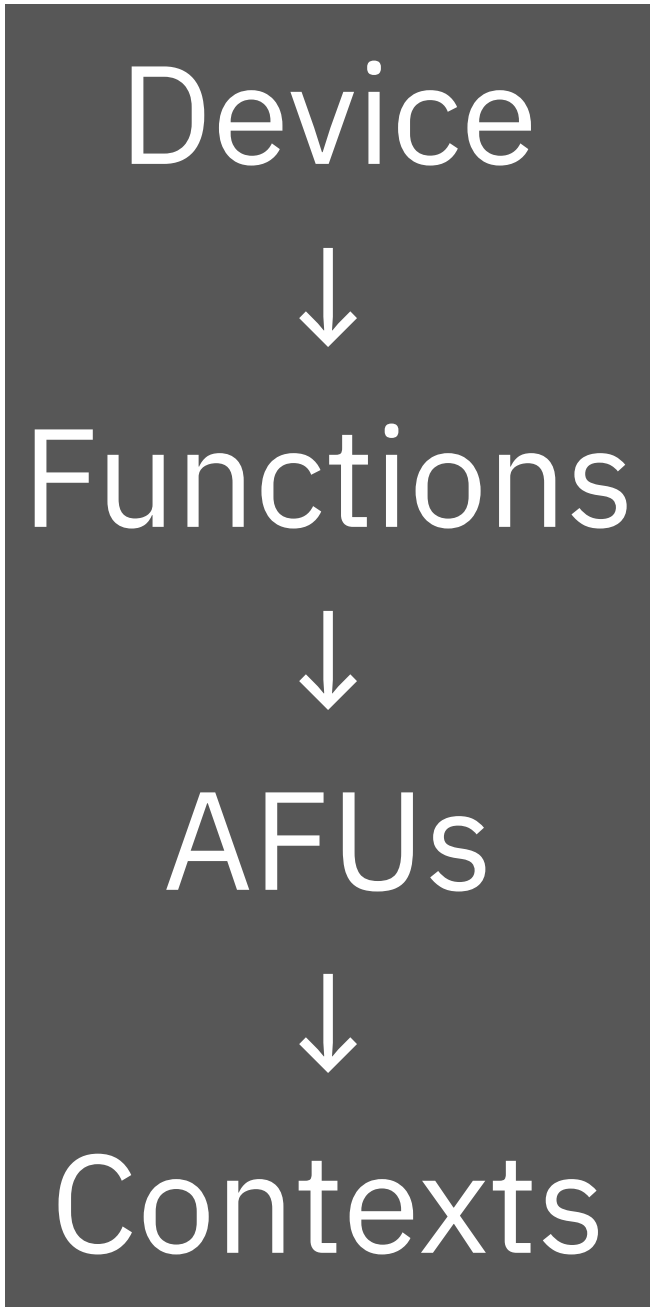
```
[   66.300124422,5] OPAL v6.1-115-g23d5532076ba-debug starting...
...
[   84.378110109,6] NPU: Chip 8 Found NPU2#1 (6 links) at
                    /xscom@623fc00000000/npu@5011000
...
[   85.211655161,7] PLAT: Chip 8 GPU#0 slot present
[   85.211689090,7] PLAT: Chip 8 GPU#0 is OpenCAPI
...
[   88.454427431,6] OCAPI[8:3]: link trained in 108 ms
...
[   88.460301163,7] PHB#0008:00:00.0 Link up at x8 width
[   88.460302626,7] PHB#0008:00:00.0 Scanning (upstream+downsteam)...
[   88.460327984,7] PHB#0008:00:00.0 Found VID:1014 DEV:062b TYP:1 MF+ BR- EX-
[   88.460341774,7] PHB#0008:00:00.1 Found VID:1014 DEV:062b TYP:1 MF+ BR- EX-
```

OpenCAPI looks like PCI

```
ubuntu@wcapilp4:/dev/ocxl$ sudo lspci -v

0006:00:00.0 Processing accelerators: IBM Device 062b
        Subsystem: IBM Device 060f
        Flags: fast devsel
        Memory at 600e804000000 (64-bit, non-prefetchable) [size=16]
        Memory at 600e804010000 (64-bit, non-prefetchable) [size=16]
        Memory at 600e804020000 (64-bit, non-prefetchable) [size=16]
        Expansion ROM at <ignored> [disabled]
        Capabilities: [40] Vital Product Data
        Kernel driver in use: ocxl
        Kernel modules: ocxl

0006:00:00.1 Processing accelerators: IBM Device 062b
        Subsystem: IBM Device 060f
        Flags: fast devsel
        Memory at 600e800000000 (64-bit, non-prefetchable) [size=64M]
        Memory at 600e804030000 (64-bit, non-prefetchable) [size=16]
        Memory at 600e804040000 (64-bit, non-prefetchable) [size=16]
        Expansion ROM at <ignored> [disabled]
        Kernel driver in use: ocxl
        Kernel modules: ocxl
```


Device
↓
Functions
↓
AFUs
↓
Contexts

AFU

Attached Functional Unit

Contexts → PASIDs

Process Address Space ID


MMIO

ocxl

drivers/misc/ocxl/

```
ubuntu@wcapilp4:/dev/ocxl$ ls -l
total 0
crw------- 1 root root 242, 0 Jan 28  2018 IBM,MEMCPY3.0006:00:00.1.0
```

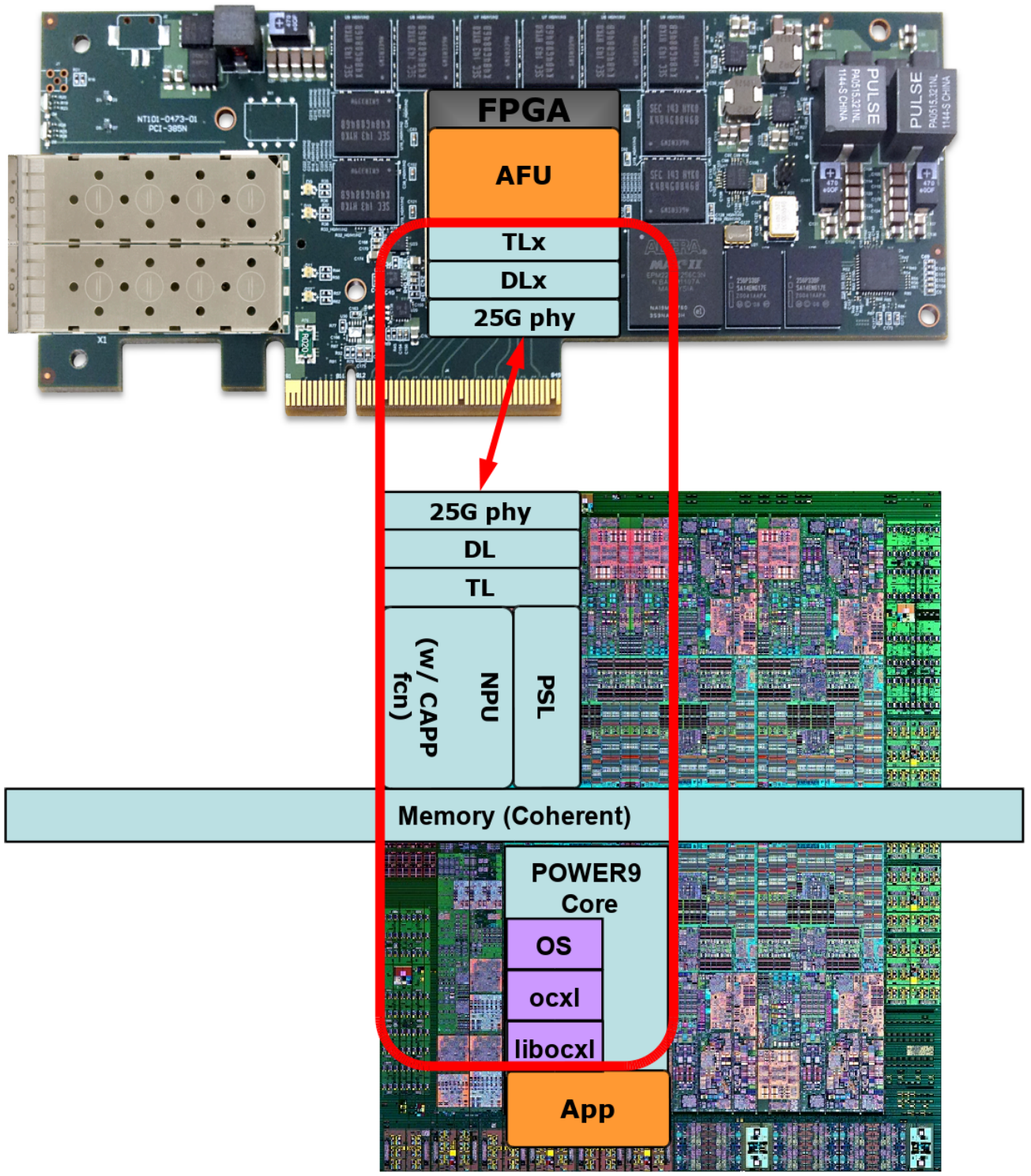
libocxl

github.com/opencapi/libocxl

```
$ sudo {apt,yum} install libocxl-dev
```

OCSE

OpenCAPI Simulation Engine


FPGA
AFU
TLx
DLx
25G phy
25G phy
DL
TL
NPU (w/ CAPP fcn)
PSL
Memory (Coherent)
POWER9 Core
OS
ocxl
libocxl
App
POWER9 Processor

memcpy

libocxl/samples/memcpy/

```
#include "libocxl.h"
#define AFU_NAME "IBM,MEMCPY3"
#define MEMCPY_SIZE 2048

int main(int argc, char *argv[])
{
    ocxl_afu_h afu;

    ocxl_enable_messages(OCXL_ERRORS | OCXL_TRACING);
    ocxl_afu_open(AFU_NAME, &afu);
    global_setup(afu);

    char *src = aligned_alloc(64, MEMCPY_SIZE);
    char *dst = aligned_alloc(64, MEMCPY_SIZE);
    fill_buffer(src, MEMCPY_SIZE);
    memset(dst, '0', MEMCPY_SIZE);
    afu_memcpy(afu, src, dst, MEMCPY_SIZE, 0, 10); /* Use polling, 10ms timeout */

    if (memcmp(dst, src, MEMCPY_SIZE))
        LOG_ERR("Memory contents do not match\n")

    ocxl_afu_close(afu);
}
```

```
struct memcpy_work_element {
    volatile uint8_t cmd; /* valid, wrap, cmd */
    volatile uint8_t status;
    union {
        uint16_t length;
        uint16_t tid;
    };
    uint8_t cmd_extra;
    uint8_t reserved[3];
    uint64_t atomic_op;
    uint64_t src;  /* also irq EA or atomic_op2 */
    uint64_t dst;
} __packed;

struct memcpy_weq {
    struct memcpy_work_element *queue;
    struct memcpy_work_element *next;
    struct memcpy_work_element *last;
    int wrap;
    int count;
};
```

```
static bool afu_memcpy(ocxl_afu_h afu, const char *src, char *dst,
                       size_t size, int completion, int timeout)
{
        uint64_t wed;
        struct memcpy_weq weq;

        memcpy3_init_weq(&weq, QUEUE_SIZE);

        // Point the work element descriptor (wed) at the work queue
        wed = MEMCPY_WED(weq.queue, QUEUE_SIZE / CACHELINESIZE);

        // Setup a work element in the queue
        struct memcpy_work_element memcpy_we;
        memset(&memcpy_we, 0, sizeof(memcpy_we));
        memcpy_we.cmd = MEMCPY_WE_CMD(0, MEMCPY_WE_CMD_COPY);
        memcpy_we.length = htole16((uint16_t) size);
        memcpy_we.src = htole64((uintptr_t) src);
        memcpy_we.dst = htole64((uintptr_t) dst);
```

```
ocxl_afu_attach(afu, OCXL_ATTACH_FLAGS_NONE);

// Map the per-PASID MMIO space
ocxl_mmio_h pp_mmio;
ocxl_mmio_map(afu, OCXL_PER_PASID_MMIO, &pp_mmio);

// Allocate an IRQ to report errors
ocxl_irq_h err_irq;
ocxl_irq_alloc(afu, NULL, &err_irq);

// Let the AFU know the handle to trigger for errors
uint64_t err_irq_handle = ocxl_irq_get_handle(afu, err_irq);

ocxl_mmio_write64(pp_mmio, MEMCPY_AFU_PP_IRQ,
                  OCXL_MMIO_LITTLE_ENDIAN, err_irq_handle);
```

```
// Write the address of the work element descriptor to the AFU
ocxl_mmio_write64(pp_mmio, MEMCPY_AFU_PP_WED,
                  OCXL_MMIO_LITTLE_ENDIAN, wed);

// setup the work queue
struct memcpy_work_element *memcpy_element =
                memcpy3_add_we(&weq, &memcpy_we);
struct memcpy_work_element *stop_element = NULL;
struct memcpy_work_element stop_we;
memset(&stop_we, 0, sizeof(stop_we));
stop_we.cmd = MEMCPY_WE_CMD(1, MEMCPY_WE_CMD_STOP);

stop_element = memcpy3_add_we(&weq, &stop_we);
```

```
        // memory barrier to ensure the descriptor is written to
        // memory before we ask the AFU to use it
        __sync_synchronize();

        // Initiate the memcpy
        memcpy_element->cmd |= MEMCPY_WE_CMD_VALID;

        // Wait for the AFU to be done
        wait_for_status(timeout, afu, memcpy_element, err_irq_handle);
        wait_for_status(timeout, afu, stop_element, err_irq_handle);
}
```

```
ubuntu@wcapilp4:~/libocxl/sampleobj$ sudo ./memcpy
AFU config = 0x0
traces reset and rearmed
WED=0x745140490fff  src=0xa85f14208c0  dst=0xa85f1421140 size=2048
Memory contents match
```

opencapi.org
opencapi.github.io

Thank you

andrew.donnellan@au1.ibm.com
@ajdlinux
sthbrx.github.io