Data acquisition in MicroTCA.

Jan Marjanovic (DESY) 9.6.2018

IEEE RT2018, MTCA pre-workshop

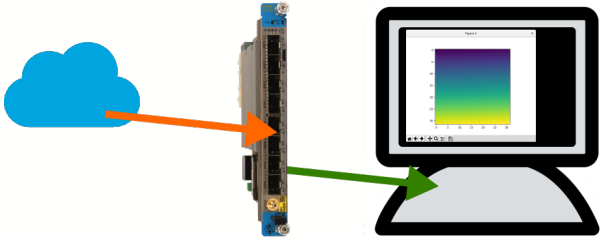
MICROTCA TECHNOLOGY LAB
A HELMHOLTZ INNOVATION LAB

DESY

General idea

A hypothetical detector is transmitting 32x32x16-bit frames, with 16-byte header.

Task: Capture the data with a MicroTCA board, store it in on-board memory, transfer it to CPU over PCIe and display it.



For this minimal example, DAMC-TCK7 acts both as our hypothetical detector and DAQ unit. In this example, a lot of simplifications have been made to highlight the main points of a typical daq system.

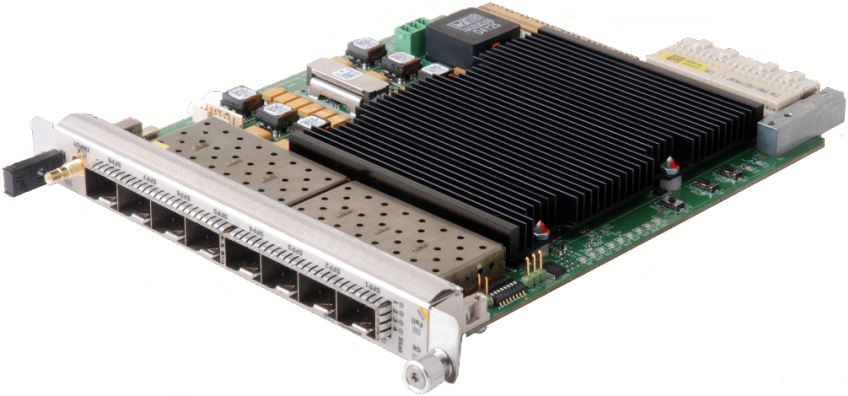
Target audience

Target audience:

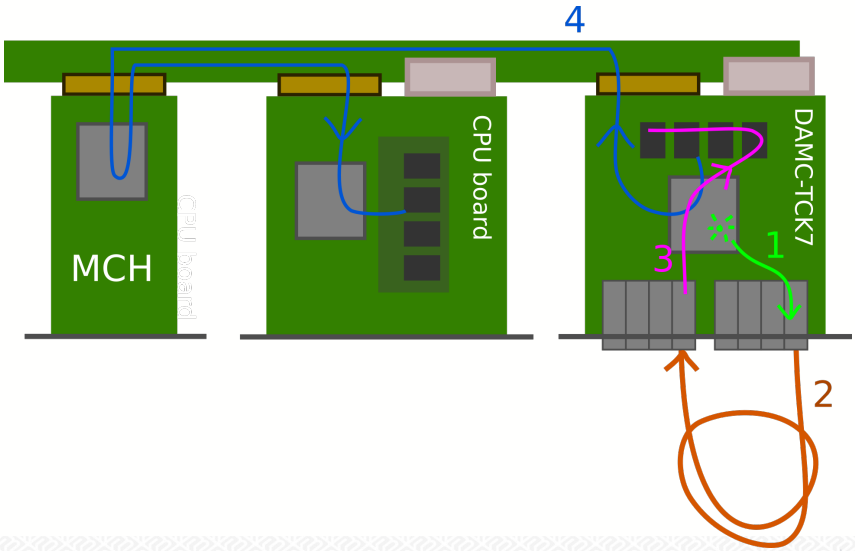
People who are at least a little bit familiar with FPGA development.

Hardware

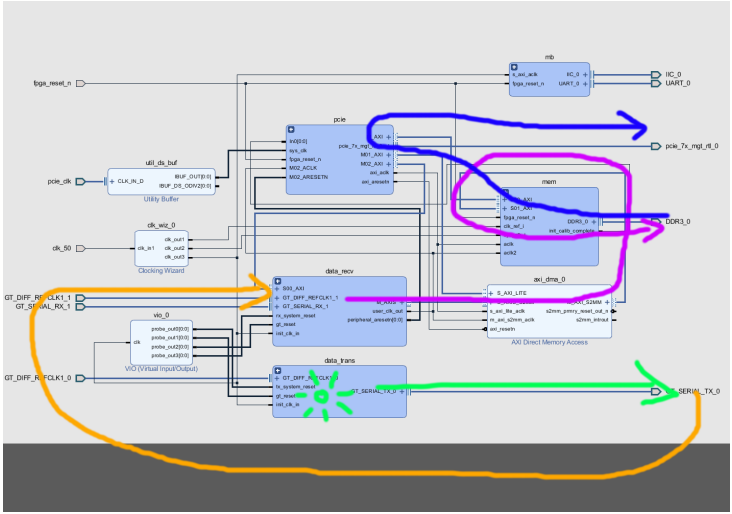
We will be using **DAMC-TCK7** (commercially available as NAMC-TCK7) with Kintex 7 FPGA, DDR3 memory, PCIe x4 gen 2 link and SFP+ slots.



Overview of the system



Overview of the architecture for FPGA



Linux driver

We are using Xilinx DMA for PCI Express (PCIe) Subsystem which requires corresponding drivers.

More about PCIe subsystem (including introductory video) at:
https://www.xilinx.com/products/intellectual-property/pcie-dma.html

Driver is available at:
https://www.xilinx.com/support/answers/65444.html

System check

Before we start with example, let's check if everything is OK with the system:

- ☑ FPGA programmed

System check

Before we start with example, let's check if everything is OK with the system:

- ☑ FPGA programmed
- ☑ PCIe link is up

System check

Before we start with example, let's check if everything is OK with the system:

- ☑ FPGA programmed
- ☑ PCIe link is up (**show_link_state**, **show_ekey**, front-panel LED)

System check

Before we start with example, let's check if everything is OK with the system:

- ☑ FPGA programmed
- ☑ PCIe link is up (**show_link_state**, **show_ekey**, front-panel LED)
- ☑ PCIe device enumerated

System check

Before we start with example, let's check if everything is OK with the system:

- ☑ FPGA programmed
- ☑ PCIe link is up (**show_link_state**, **show_ekey**, front-panel LED)
- ☑ PCIe device enumerated (**lspci**)

System check

Before we start with example, let's check if everything is OK with the system:

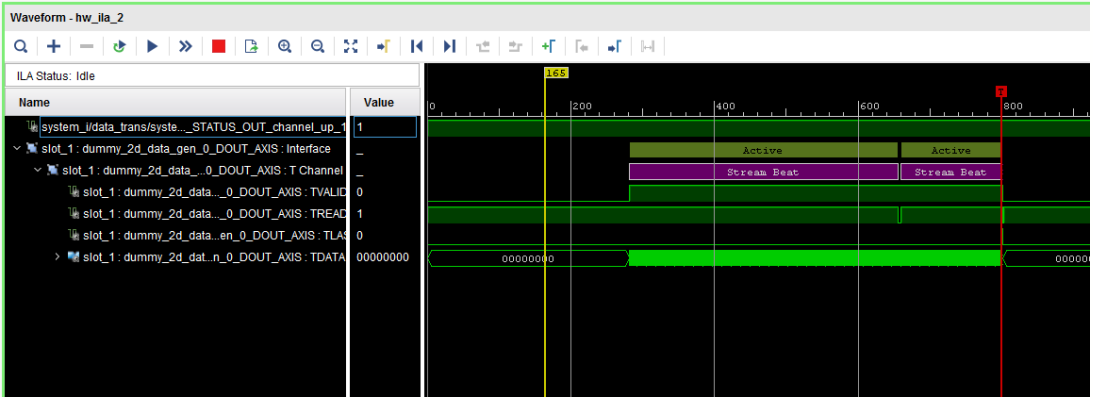
- ☑ FPGA programmed
- ☑ PCIe link is up (**show_link_state**, **show_ekey**, front-panel LED)
- ☑ PCIe device enumerated (**lspci**)
- ☑ Driver loaded

System check

Before we start with example, let's check if everything is OK with the system:

- ☑ FPGA programmed
- ☑ PCIe link is up (**show_link_state**, **show_ekey**, front-panel LED)
- ☑ PCIe device enumerated (**lspci**)
- ☑ Driver loaded (**lspci -v**)

Our sensor is generating data



Register map

Three modules accessible on AXI4-Lite Master (control and status interfaces):

```
BAR0
├── 0x0000 AXI GPIO
│   └── 0x0000 bit[0] = DDR3 ready
├── 0x1000 AXI DMA
│   ├── 0x1030 S2MM_DMACR
│   ├── 0x1034 S2MM_DMASR
│   └── 0x1058 S2MM_LENGTH
└── 0x4000 Data recv module
    ├── 0x4000 ID_REG
    ├── 0x4004 STATUS_REG
    └── 0x4008 CTRL_REG
```

Software part

We will be using reg_rw utility from Xilinx to access registers; we are closer to the ”metal”.

1. Check DDR3 status

```
./reg_rw /dev/xdma/card0/user 0x0 w
```

2. Check if DMA is in idle

```
./reg_rw /dev/xdma/card0/user 0x1034 w
```

Software part

Get the data from sensor into on-board DDR3 memory

1. Enable DMA

```
./reg_rw /dev/xdma/card0/user 0x1030 w 0x1
```

2. Check if DMA is not idle anymore

```
./reg_rw /dev/xdma/card0/user 0x1034 w
```

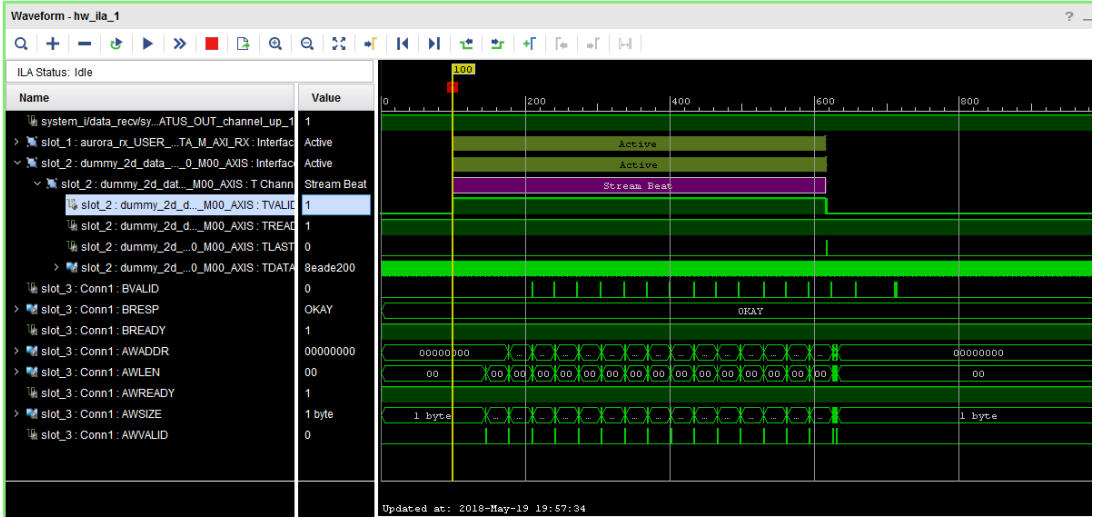
3. Start transfer

```
./reg_rw /dev/xdma/card0/user 0x1058 w 0x2000
```

4. Start data capture

```
./reg_rw /dev/xdma/card0/user 0x4008 w 0x1
```

Data was transmitted



Software part #2

Get the data from on-board DDR3

1. Check recv status

```
./reg_rw /dev/xdma/card0/user 0x4004 w
```

2. Check DMA (nr bytes transferred)

```
./reg_rw /dev/xdma/card0/user 0x1058 w
```

3. Copy data to file

```
./dma_from_device -d /dev/xdma/card0/c2h0 -s 0x810 -f data.bin
```

Plotting the image

```
#! /usr/bin/env python3

import numpy as np
import matplotlib.pyplot as plt

bs = open("pkt_dump3.bin", "rb").read()

# strip header
data = bs[4*4:]

img = np.frombuffer(data, np.uint16)
w = h = int(np.sqrt(len(img)))
img = img.reshape((w, h))
img_float = img.astype(np.float32)

plt.imshow(img_float)
plt.show()
```

Final result