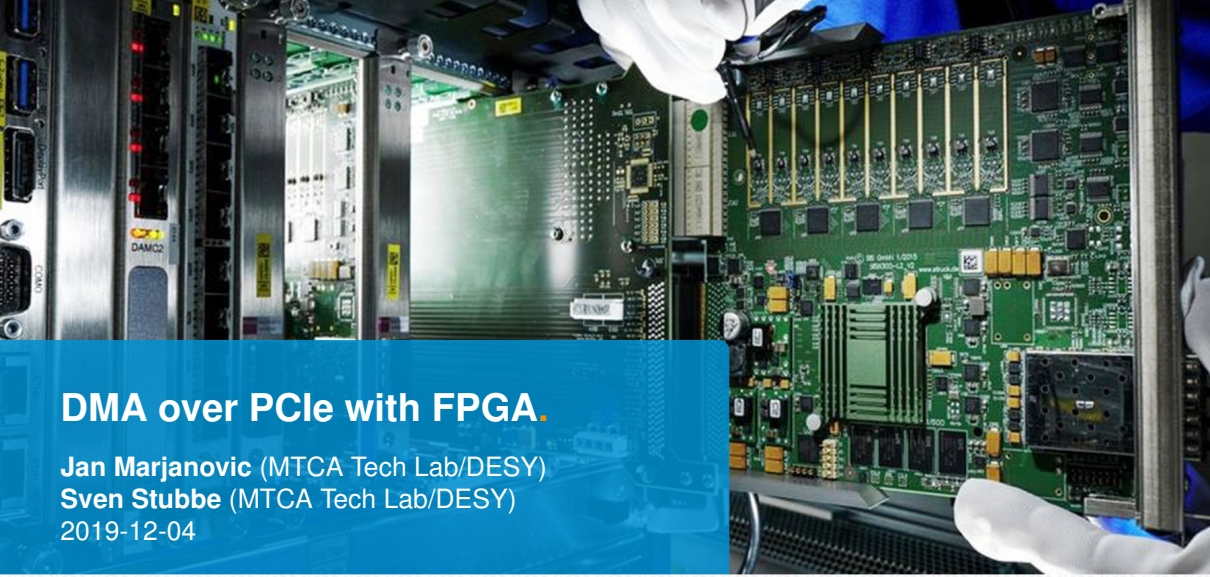


DMA over PCIe with FPGA.

Jan Marjanovic (MTCA Tech Lab/DESY)
Sven Stubbe (MTCA Tech Lab/DESY)
2019-12-04

MicroTCA Workshop 2019, Hamburg

MicroTCA TECHNOLOGY LAB
HELMHOLTZ RESEARCH FOR GRAND CHALLENGES
DESY.

Introduction

- MicroTCA and PCI Express - introduction
- PCI Express protocol
- PCI Express protocol - physical layer
- PCI Express protocol - configuration
- FPGA (in second part)
- Linux Drivers (in second part)
- MicroTCA and PCI Express
- Hot Swap and Hot Plug

MicroTCA and PCI Express Introduction

A typical ADC system in MicroTCA is shown here:

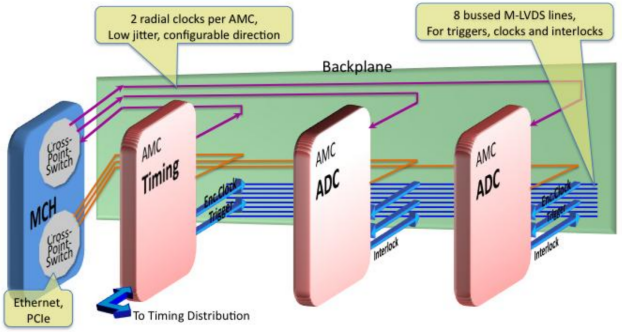


Figure 6-6: A 'typical' analog front-end.
CPU and further AMC's are not shown

from PICMG® MicroTCA.4 Enhancements for Rear I/O and Timing R1.0

A typical board in such a system is shown here:

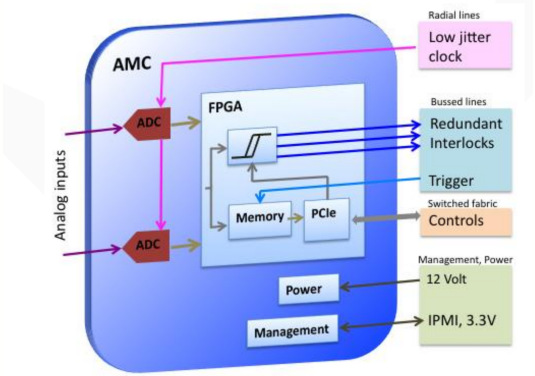


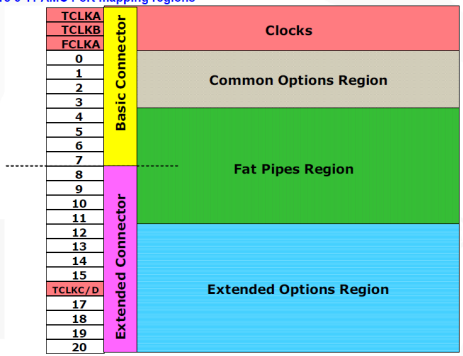
Figure 6-7: An example of a fast front-end (ADC AMC0)

from PICMG® MicroTCA.4 Enhancements for Rear I/O and Timing R1.0

Ports on an Advanced Mezzanine Cards

AMC standard provides some guidance on how the ports should be assigned:

Figure 6-11 AMC Port mapping regions



from PICMG® Advanced Mezzanine Card AMC.O Specification R2.0

AMC Family of Specification

from https://www.picmg.org/openstandards/advanced-mezzanine-card/

Family of Specifications

AdvancedMC for Serial Rapid I/O

PICMG#	NAME	CURRENT REVISION	DATE	DESCRIPTION
PICMG AMC.4	AdvancedMC for Serial Rapid I/O	1.0	2009-07-09	Defines additional requirements for Serial Rapid I/O

AdvancedMC for Storage

PICMG#	NAME	CURRENT REVISION	DATE	DESCRIPTION
PICMG AMC.3	AdvancedMC for Storage	Rev 1.0	2005-08-05	Defined additional requirements for Fibre Channel

AdvancedMC for Ethernet

PICMG#	NAME	CURRENT REVISION	DATE	DESCRIPTION
PICMG AMC.2	AdvancedMC for Ethernet	Rev 1.0	2007-03-01	Defines additional requirements for Ethernet interconnects

AdvancedMC for PCI Express

PICMG#	NAME	CURRENT REVISION	DATE	DESCRIPTION
PICMG AMC.1	AdvancedMC for PCI Express	Rev 2.0	2008-10-08	Defines additional requirements for PCI Express interconnects

AdvancedMC® Mezzanine Module

PICMG#	NAME	CURRENT REVISION	DATE	DESCRIPTION
PICMG AMC.0	AdvancedMC® Mezzanine Module	Rev 2.0	2006-11-15	Defines all electrical, mechanical, and system management requirements for building AMC's

MicroTCA backplane

MicroTCA backplane connections as specified by MicroTCA.4 standard:

REQ 6-6 Port 0 **should** be used for base Ethernet interface. Port 1 is connected to the optional (redundant) MCH.

REQ 6-7 Port 4 to 7 **should** be used for PCIe.

REQ 6-8 Port 12 to 15 **may** be used for application specific wire-ring.

REQ 6-9 Ports 17- 20 **should** be wired as a bus according Section 6.4.

REQ 6-10 FCLKA (Clk3) **should** be used for PCIe clock distribution as defined in MTCA.O R1.0

...

from PICMG® Specification MTCA.4 Revision 1.0: MicroTCA Enhancements for Rear I/O and Precision Timing

MicroTCA backplane

MicroTCA backplane connections as specified by MicroTCA.4 standard:

REQ 6-6 Port 0 **should** be used for base Ethernet interface. Port 1 is connected to the optional (redundant) MCH.

REQ 6-7 Port 4 to 7 **should** be used for PCIe.

REQ 6-8 Port 12 to 15 **may** be used for application specific wire-ring.

REQ 6-9 Ports 17- 20 **should** be wired as a bus according Section 6.4.

REQ 6-10 FCLKA (Clk3) **should** be used for PCIe clock distribution as defined in MTCA.O R1.0

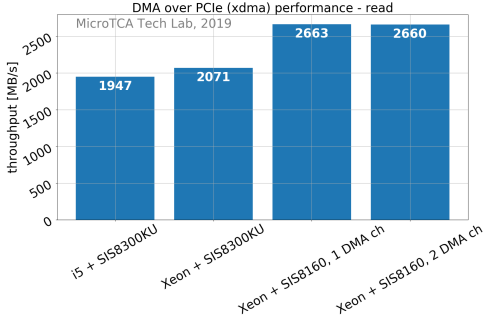
...

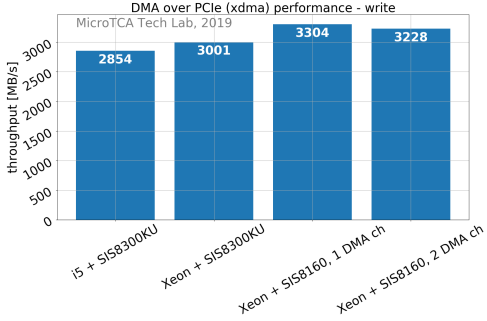
from PICMG® Specification MTCA.4 Revision 1.0: MicroTCA Enhancements for Rear I/O and Precision Timing

Reasons to use PCI Express

- High-throughput (up to 64 Gbps (gen 3, x8 link))
- Low-latency
- Software model (memory-based transaction, method to detect devices on the bus)

Example of the throughput achieved with two of the boards in MicroTCA:





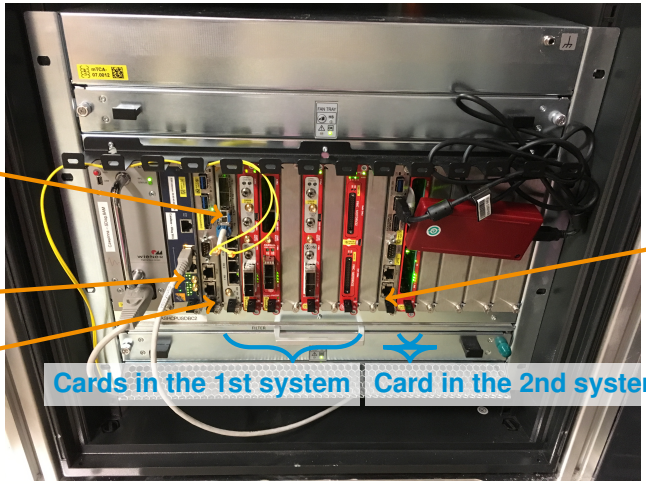
Reasons to use PCI Express

A lot of boards available on the market use PCI Express as main/only communication protocol.
FPGAs also allow implementation of other protocols (e.g. 10 and 40 Gb Ethernet and Serial Rapid IO).





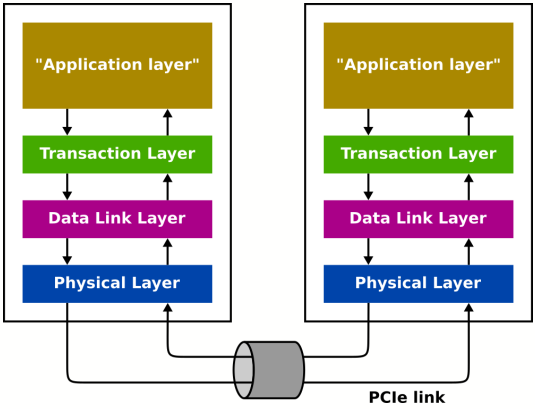

Example from FLASH: PCI Express switch enables splitting crate in two separate systems, but with shared timing and interlocks.



PCI Express protocol

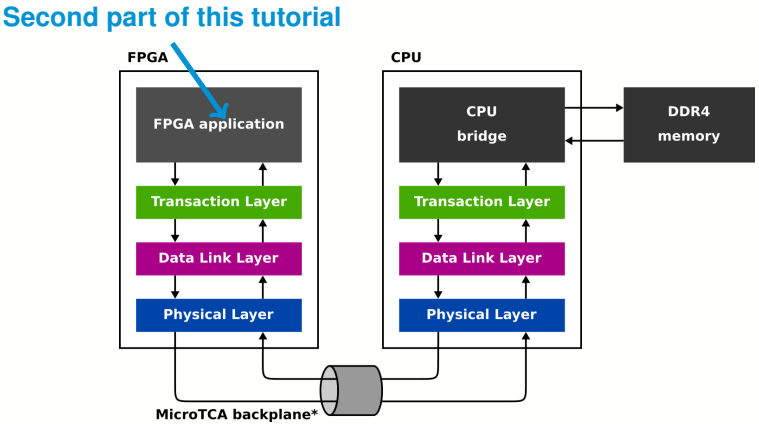
PCI Express

PCIe express is high-speed protocol for PC extension cards. Different form-factors are also available. Link out of 1, 2, 4, 8 or 16 lanes, lane rate 2.5 GT/s, 5.0 GT/s, 8.0 GT/s and 16.0 GT/s.



inspired by Figure 2-12 from PCI Express Technology 3.0

Concrete example:



* real PCIe link in MicroTCA system includes a PCIe switch in MicroTCA Carrier Hub (MCH)

Physical Layer

- two differential pairs (TX and RX) per lane
- system clock (typical 100 MHz)
- 8b/10b (2.5 and 5.0 GT/s) or 128b/130b encoding (8.0 GT/s)
- Link Training and Status State Machine (LTSSM)
- in FPGAs implemented with transceivers + some logic

Data Link Layer

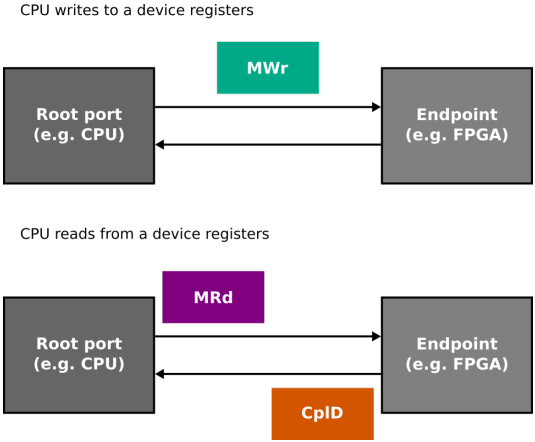
- Exhange of flow-control credits (= free buffers in receiver)
- CRC check, ACK/NAK, re-transmission

Transaction Layer

- ”Real application” packets - memory read and write, ...

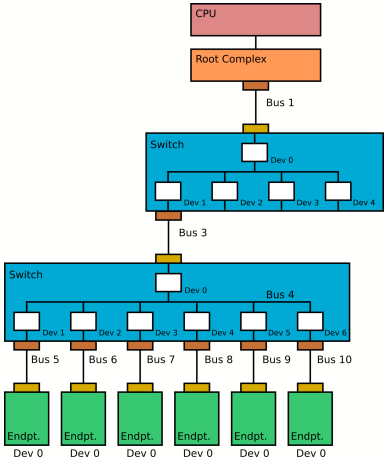
PCIe TLP transaction types

Packets can be one of several types: Memory Read, Memory Write, Completion, Completion with Data, Messages, Configuration Accesses and some legacy types.

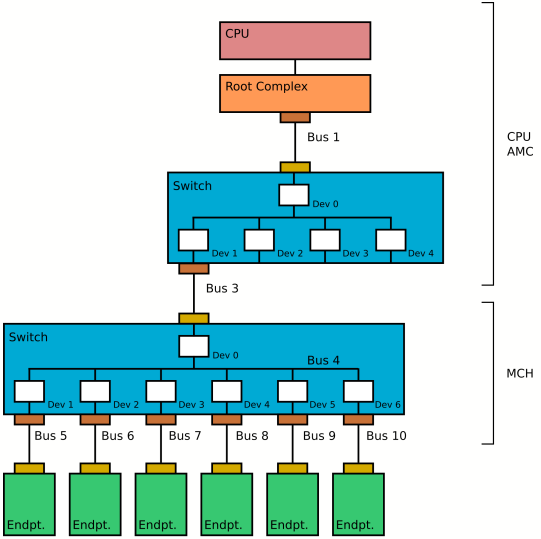


PCIe switches, buses, devices

Since PCIe is a point-to-point link, switches are used to connect more than 1 device to the CPU. Each PCIe device has its own address (bus:device.function), assigned during the enumeration process.



In the switch on the MCH one or more slots are configured as Upstream AMC.

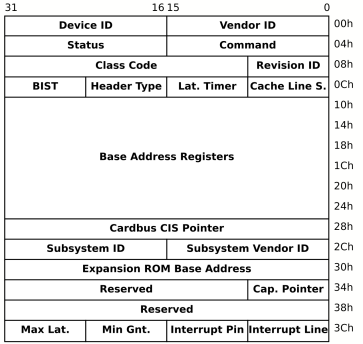


PCIe configuration space

Configuration Register Space defined for each Function. The first 256 bytes are the same as for PCI, a total of 4 kB for PCIe Extended Configuration Area.

The configuration space contains information such as: Vendor and Device ID, Link Status, Base Address Registers, ...

No driver [1] is needed to read this space. Drivers in Linux use Vendor ID and Device ID (and Subsystem Vendor ID and Subsystem Device ID) to claim devices.

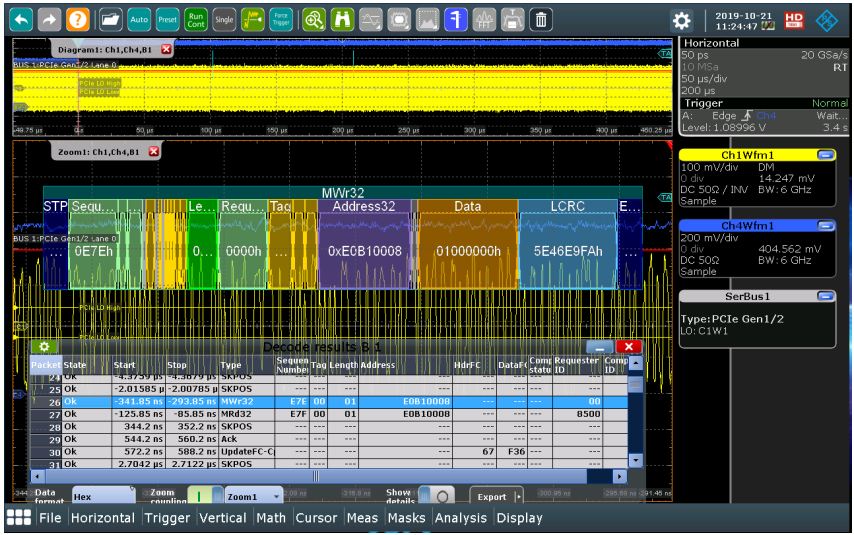


By Vijay Kumar Vijaykumar - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=3181779

[1] there is a driver embedded in OS, usually, i.e. in GNU/Linux

TLP examples

MWr - Memory Write

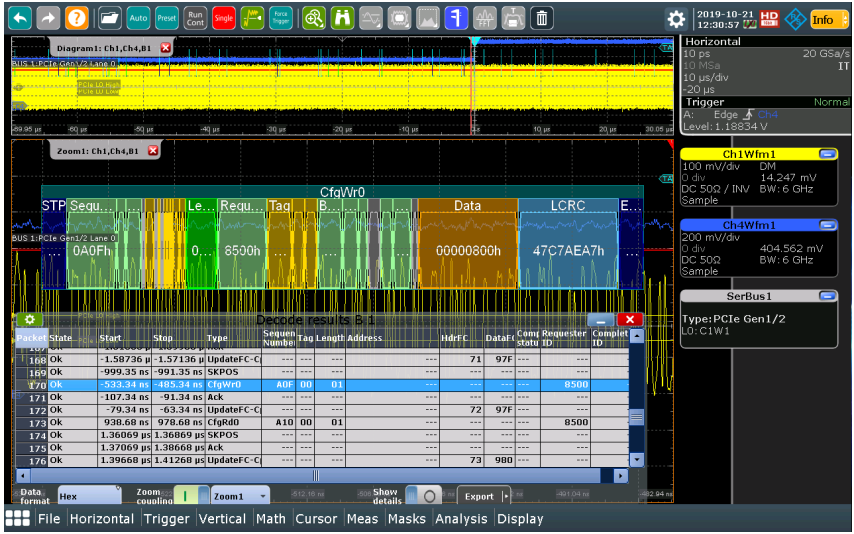


MRd - Memory Read

CplD - Completion with Data

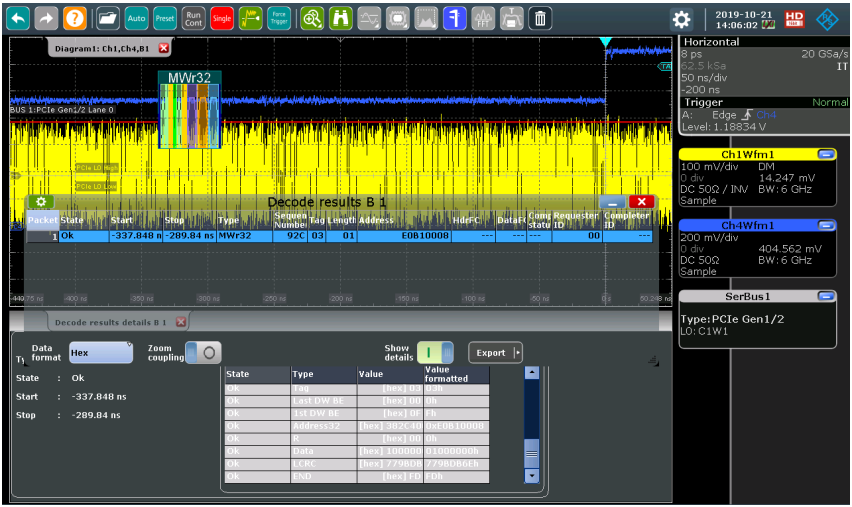
CfgRd - Configuration Read

CfgWr - Configuration Write

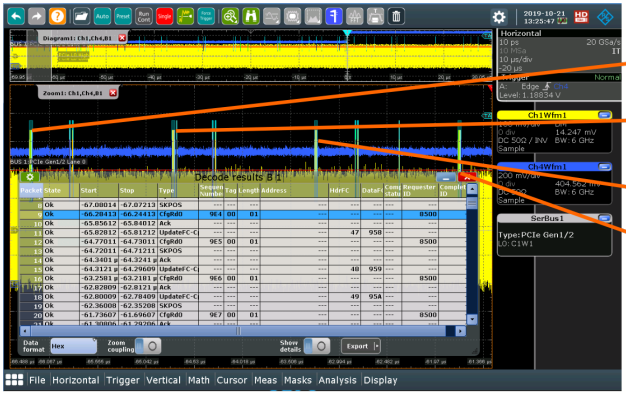


Memory write → LED

Turn on an LED with Memory Write



Software ”walking” Configuration Area



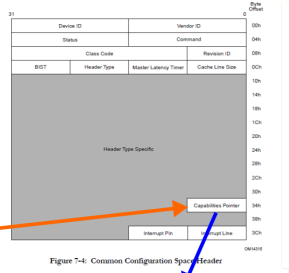


Figure 7-4: Common Configuration Space Header

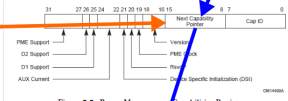
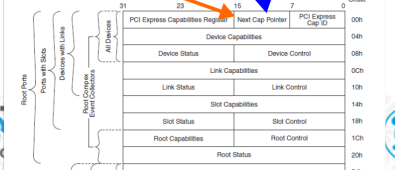


Figure 7-7: Power Management Capabilities Register

Capability Structure for 64-bit Message Address

Figure 6-11: MSI Capability Structure

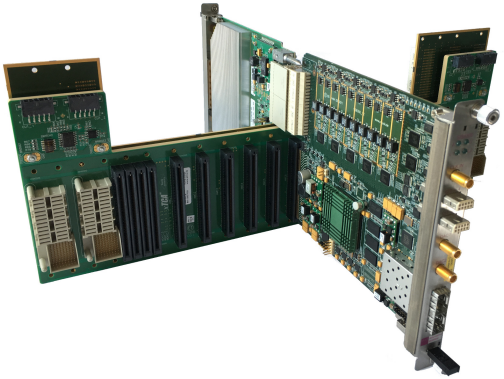


PCI Express protocol
Physical Layer

Signal integrity

In a typical MicroTCA systems a signal integrity is an important topic.

- Each high speed signal passes 2 card-edge connectors.
- Components from different vendors are plugged together



Indication of poor signal integrity

Link not running at the maximum possible rate:

PCIe Link Status Menu

	AMC1	AMC2	AMC3	AMC4	AMC5	AMC6	AMC7	AMC8	AMC9	AMC10	AMC11	AMC12	OPT1	RTM
	4..7	4..7	4..7	4..7	4..7	4..7	4..7	4..7	4..7	4..7	4..7	4..7		
	-	-	-	-	-	-	-	-	x4	x4	-	-	x8	-
Link Speed	-	-	-	-	-	-	-	-	5 GT/s	2.5 GT/s	-	-	8 GT/s	-

High rate of DLLP errors (rule of thumb: 1 correctable (re-trans) error per minute is OK):

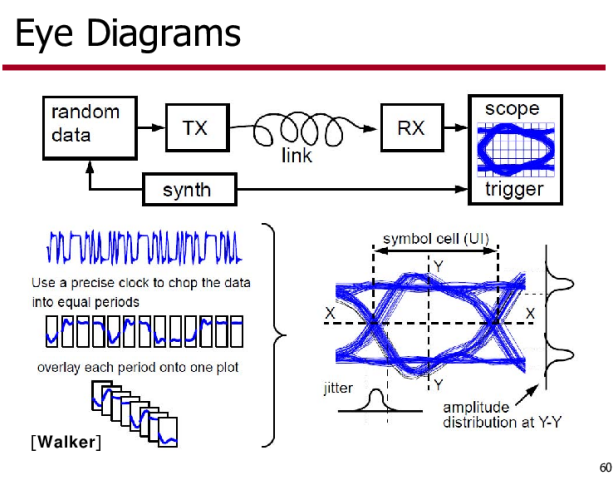
PCIe Error Counters Menu

	AMC1	AMC2	AMC3	AMC4	AMC5	AMC
	4..7	4..7	4..7	4..7	4..7	4..
	x4	x4	x4	x4	x4	x4
RCV_CNT	0	255	0	0	0	0
Bad TLP	0	0	0	0	0	0
Bad DLLP	0	65	0	0	0	0

Reset ☐ Refresh

Eye diagram

Error counters are only the high level indication that something is not OK. To understand more about the quality of the signal, one needs to look at the eye diagram.

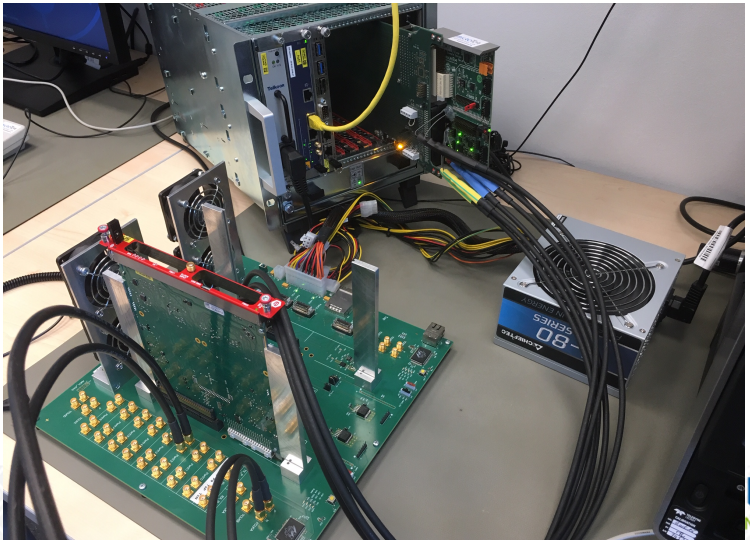


from: Sam Palermo, ECEN689: Special Topics in Optical Interconnects Circuits and Systems, http://ece.tamu.edu/~spalermo/ecen689_oi/lecture4_ee689_rx_analysis.pdf

Signal integrity measurement

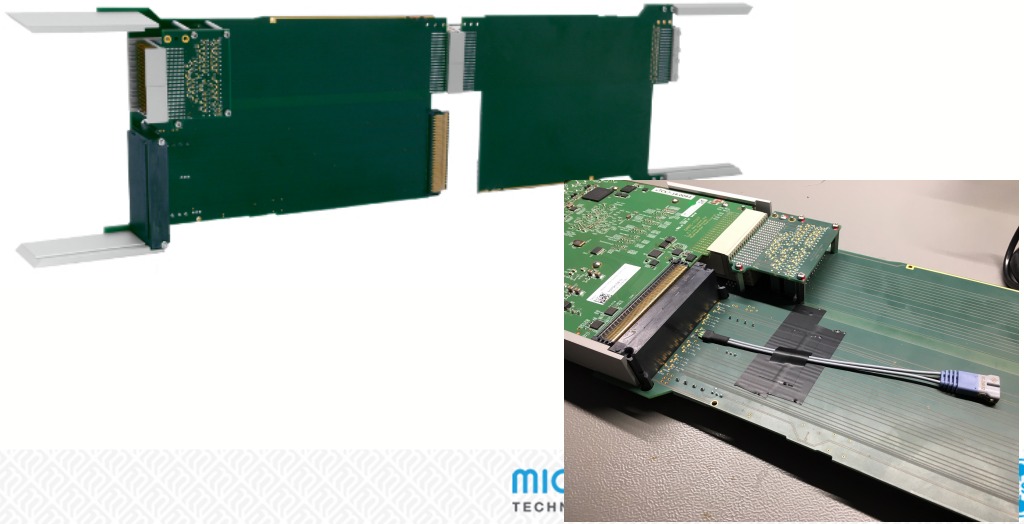
When developing or debugging boards in MicroTCA (or any other form factor) having test harnesses is extremely useful.

One example is an AMC test harness (PCIe connected with coax cables - 6 in total):



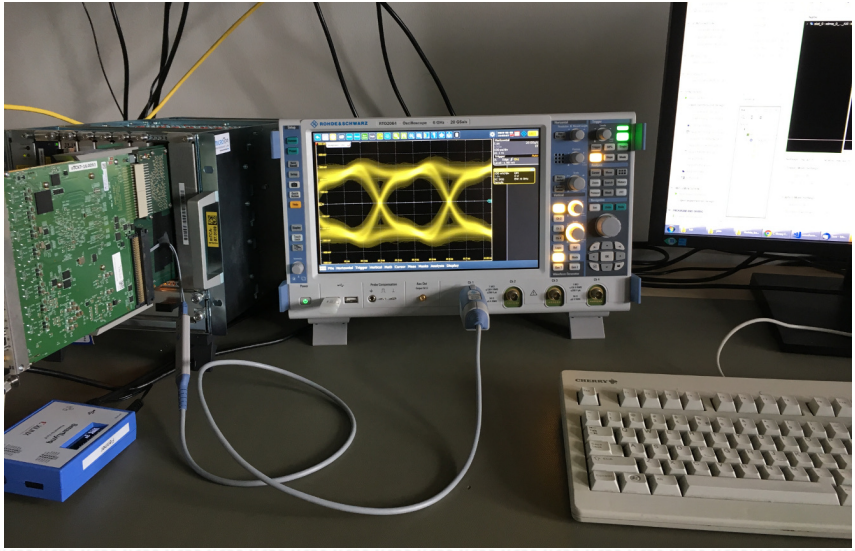
Signal integrity measurement

Extender boards (e.g. from N.A.T. - `https://www.nateurope.com/products/NAMC-EXT-RTM.html`) is a good equipment when we want to observe the data on the backplane:



Signal integrity measurement

With oscilloscope, PCIe 2.5 GT/s:



In System IBERT allows measuring link quality during operation (everything in grey is not needed for normal operation of the link and is there only to be used for eye scan):

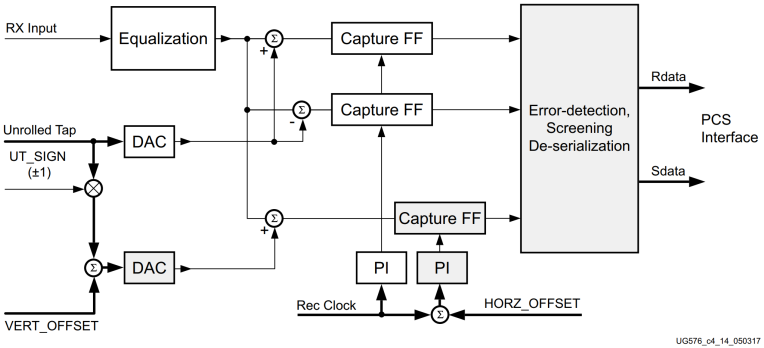
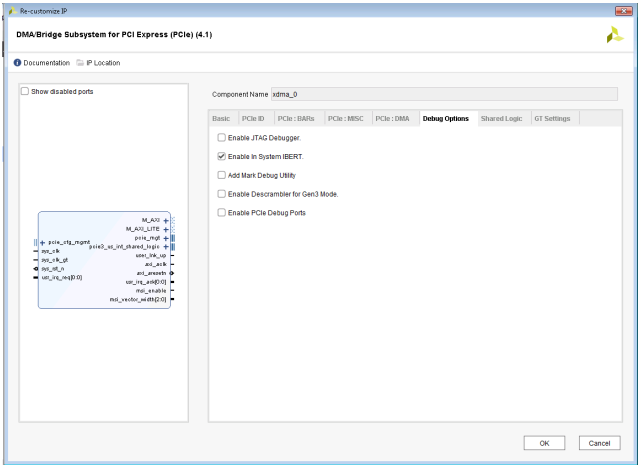


Figure 4-19: **PMA Architecture to Support Eye Scan**

from

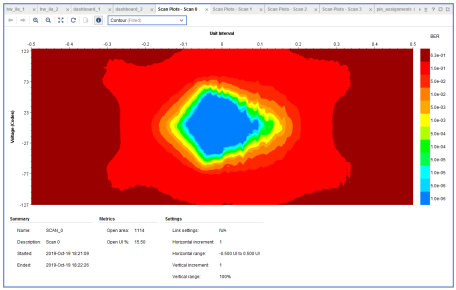
https://www.xilinx.com/support/documentation/user_guides/ug576-ultrascale-gth-transceivers.pdf

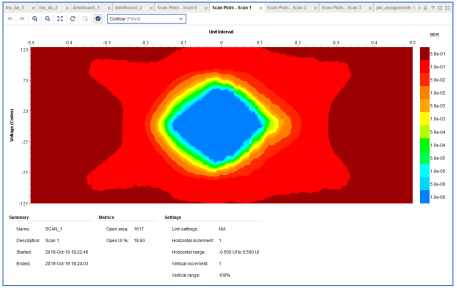
In System IBERT is enabled in IP configuration:

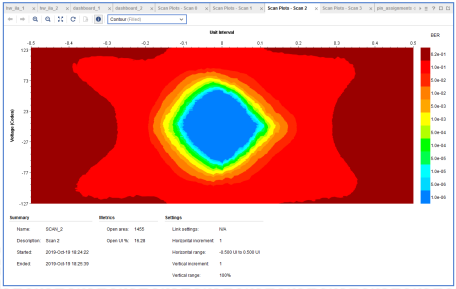


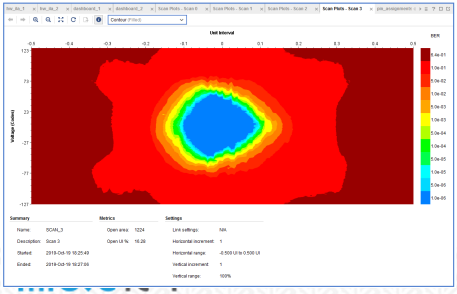
Measurements: IBERT

ELMA crate (12 slot) with SIS8300-KU at 8 GT/s



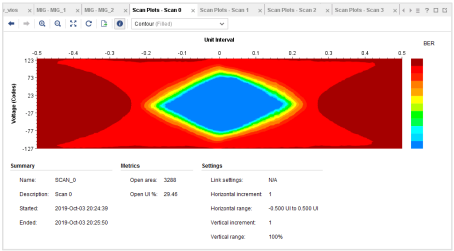


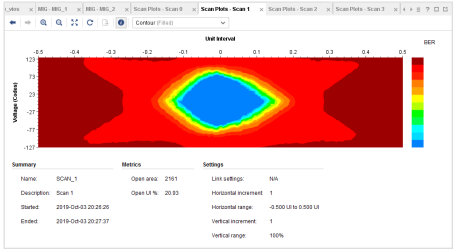


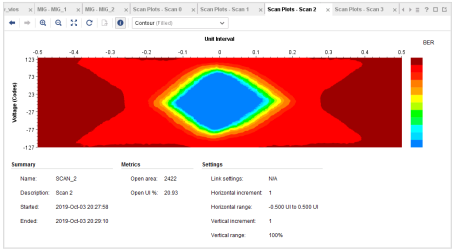


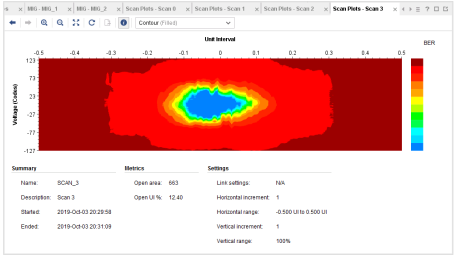
Measurements: IBERT

Schroff crate (12 slot) with SIS8160 at 8 GT/s



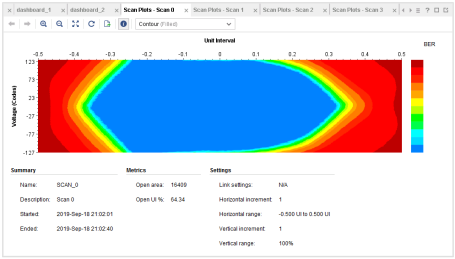


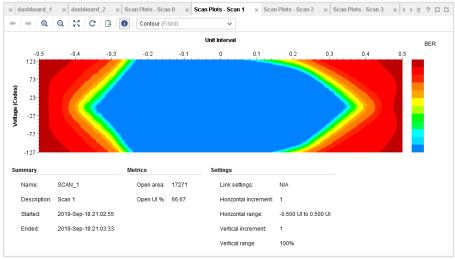


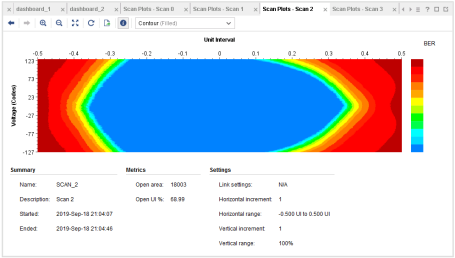


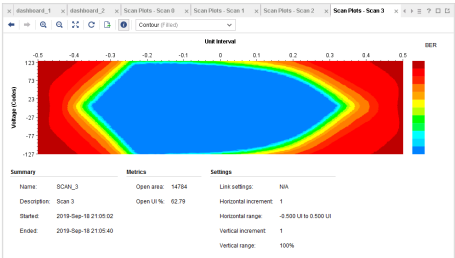
Measurements: IBERT

Schroff crate (7 slot) with SIS8300-KU at 5 GT/s (limited to 5 GT/s from slot configuration)

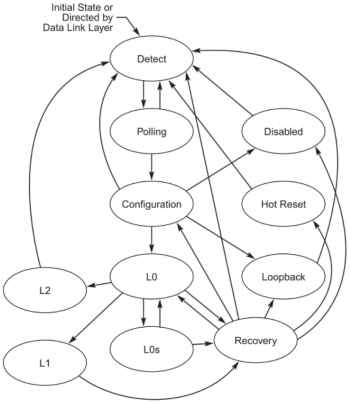








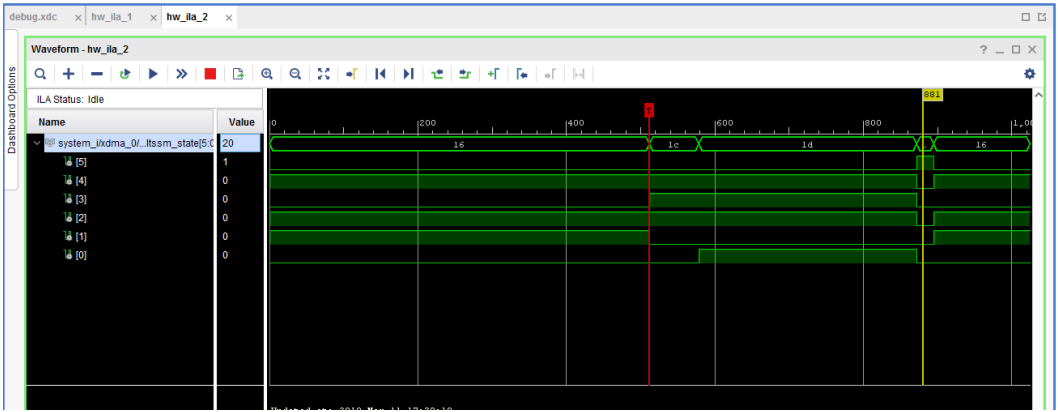
Link Training and State State Machine is a state machine running on both devices on the link. At the start up it negotiates link speed and link width, and removes the skew between lanes.



Very useful PCIe debug guide from Xilinx (recomends observing LTTSM state with Integrated Logic Analyzer - ILA):
`https://www.xilinx.com/Attachment/Xilinx_Answer_56616_7_Series_PCIe_Link_Training_Debug_Guide.pdf`

Here is an example of LTSSM state observed with ILA.

This example shows LTSSM leaving state L0 (the normal working state, encoded as 0x16) and cycling through states:
0x1C - Recovery Rcvrlock, 0x1D - Recovery Rcvrcfg, 0x20 - Recovery Idle.



PCI Express protocol Configuration

Tools

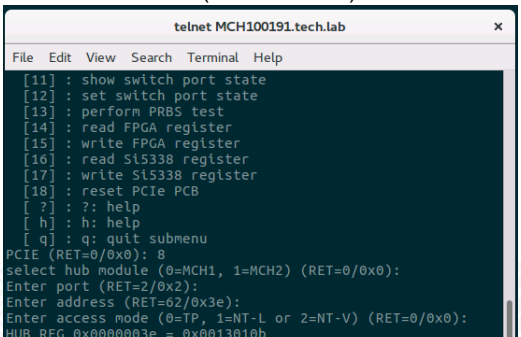
Two utilities from pciutils package (`https://github.com/pciutils/pciutils`) are useful to inspect and configure PCI Express system (on GNU/Linux):

- `lspci` - list PCI and PCIe devices, their organization and current status
- `setpci` - read and write individual registers in Configuration Space

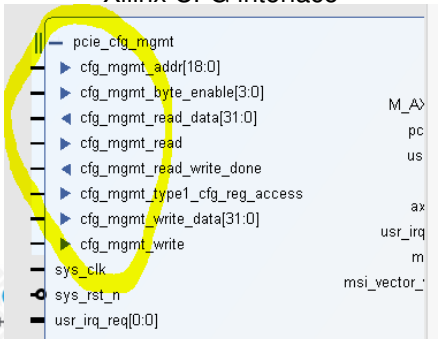
+

Tools to access the interface from the other (back) side:

N.A.T. MCH (PCIe switch) interface



Xilinx CFG interface

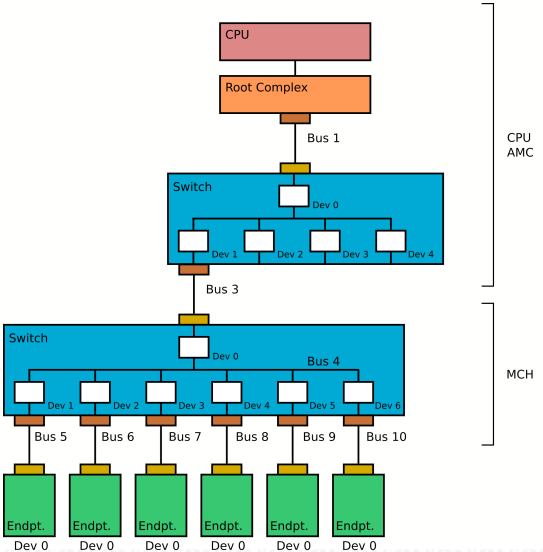


Example on Concurrent Tech AM G6x board:

```
$ lspci
00:00.0 Host bridge: Intel Corporation Xeon E3-1200 v6/7th Gen Core Processor Host Brid   /
00:01.0 PCI bridge: Intel Corporation Xeon E3-1200 v5/E3-1500 v5/6th Gen Core Processor
...
00:1f.3 Audio device: Intel Corporation CM238 HD Audio Controller (rev 31)
00:1f.4 SMBus: Intel Corporation 100 Series/C230 Series Chipset Family SMBus (rev 31)
01:00.0 PCI bridge: PLX Technology, Inc. Device 8725 (rev ca)
01:00.1 System peripheral: PLX Technology, Inc. Device 87d0 (rev ca)
...
02:01.0 PCI bridge: PLX Technology, Inc. Device 8725 (rev ca)
...
03:00.0 PCI bridge: PLX Technology, Inc. PEX 8748 48-Lane, 12-Port PCI Express Gen 3 (8    / )
04:01.0 PCI bridge: PLX Technology, Inc. PEX 8748 48-Lane, 12-Port PCI Express Gen 3 (8    / )
04:03.0 PCI bridge: PLX Technology, Inc. PEX 8748 48-Lane, 12-Port PCI Express Gen 3 (8    / )
...
05:00.0 Serial controller: Xilinx Corporation Device 7024
0d:00.0 Ethernet controller: Intel Corporation Ethernet Controller X710 for 10GbE SFP+ (    0
...
```

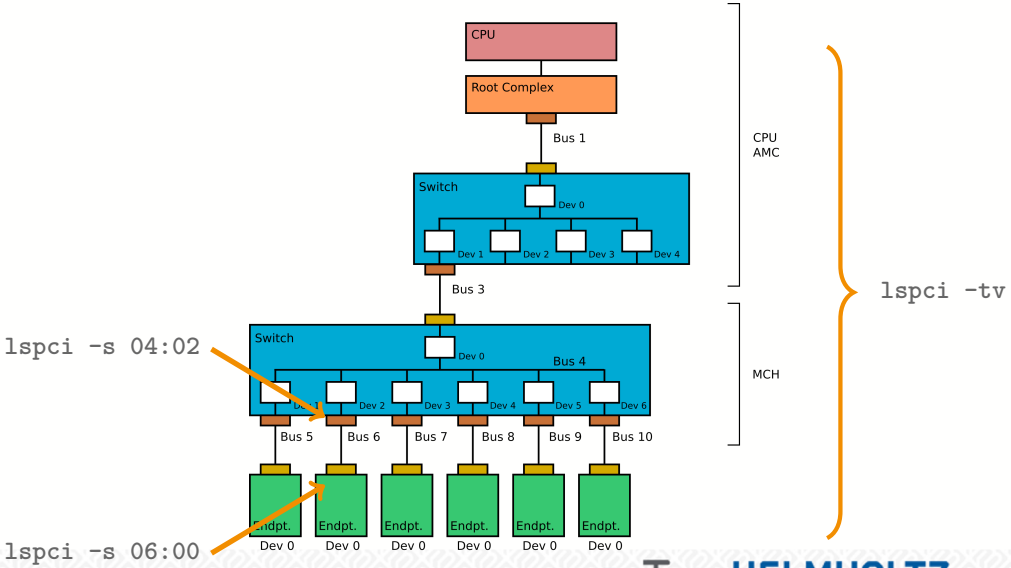
Various outputs from lspci

Several arguments can be passed to lspci to obtain different information from the system.



Various outputs from lspci

Several arguments can be passed to lspci to obtain different information from the system.



`lspci -tv` can be used to show the tree structure of the PCIe bus:

Example on Concurrent Tech AM G6x board:

```
$ lspci -tv
-[0000:00]-+-00.0  Intel Corporation Xeon E3-1200 v6/7th Gen Core Processor Host Bridge/DRAM Registers
           +-01.0-[01-0e]--+-00.0-[02-0e]--+-01.0-[03-0a]----00.0-[04-0a]--+-01.0-[05]--
           |               |                |                               +-03.0-[06]----00.0  Xilinx Corp [.] 7024
           |               |                |                               +-08.0-[07]--
           |               |                |                               +-0a.0-[08]--
           |               |                |                               +-11.0-[09]--
           |               |                |                               \-13.0-[0a]--
           |               |                +-02.0-[0b]--
           |               |                +-08.0-[0c]--
           |               |                +-09.0-[0d]--+-00.0  Intel Corporation Ethernet Controller X710 for 10GbE SFP+
           |               |                |            \-00.1  Intel Corporation Ethernet Controller X710 for 10GbE SFP+
           |               |                \-0a.0-[0e]--
           |               +-00.1  PLX Technology, Inc. Device 87d0
           |               +-00.2  PLX Technology, Inc. Device 87d0
           |               +-00.3  PLX Technology, Inc. Device 87d0
           |               \-00.4  PLX Technology, Inc. Device 87d0
           +-01.1-[0f]--
           +-02.0  Intel Corporation HD Graphics P630
           +-08.0  Intel Corporation Xeon E3-1200 v5/v6 / E3-1500 v5 / 6th/7th Gen Core Processor Gaussian Mixture Model
           +-14.0  Intel Corporation 100 Series/C230 Series Chipset Family USB 3.0 xHCI Controller
           +-14.2  Intel Corporation 100 Series/C230 Series Chipset Family Thermal Subsystem
           +-15.0  Intel Corporation 100 Series/C230 Series Chipset Family Serial IO I2C Controller #0
           +-15.1  Intel Corporation 100 Series/C230 Series Chipset Family Serial IO I2C Controller #1
           +-16.0  Intel Corporation 100 Series/C230 Series Chipset Family MEI Controller #1
           +-17.0  Intel Corporation Q170/Q150/B150/H170/H110/Z170/CM236 Chipset SATA Controller [AHCI Mode]
           +-1c.0-[10]----00.0  Intel Corporation I210 Gigabit Network Connection
           +-1c.1-[11]----00.0  Intel Corporation I210 Gigabit Backplane Connection
           +-1c.2-[12]----00.0  Intel Corporation I210 Gigabit Backplane Connection
           ...
```

lspci - slot

```
$ sudo lspci -s 04:03.0 -vv
04:03.0 PCI bridge: PLX Technology, Inc. PEX 8748 48-Lane, 12-Port PCI Express Gen 3 (8    / )
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- S    -
        [...]
        Bus: primary=04, secondary=06, subordinate=06, sec-latency=0
        [...]
        Capabilities: [68] Express (v2) Downstream Port (Slot+), MSI 00
                DevCap: MaxPayload 512 bytes, PhantFunc 0
                        ExtTag- RBE+
                DevCtl: Report errors: Correctable+ Non-Fatal+ Fatal+ Unsupported+
                        RlxdOrd+ ExtTag- PhantFunc- AuxPwr- NoSnoop+
                        MaxPayload 256 bytes, MaxReadReq 128 bytes
                DevSta: CorrErr+ UncorrErr- FatalErr- UnsuppReq+ AuxPwr- TransPend-
                LnkCap: Port #3, Speed 8GT/s, Width x4, ASPM L1, Exit Latency L0s <2us,     <4
                        ClockPM- Surprise+ LLActRep+ BwNot+ ASPMOptComp+
                LnkCtl: ASPM Disabled; Disabled- CommClk-
                        ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
                LnkSta: Speed 5GT/s, Width x4, TrErr- Train- SlotClk- DLActive+ BWMgmt-
                SltCap: AttnBtn+ PwrCtrl+ MRL+ AttnInd- PwrInd- HotPlug+ Surprise-
                        Slot #3, PowerLimit 25.000W; Interlock- NoCompl-
                SltCtl: Enable: AttnBtn+ PwrFlt- MRL- PresDet- CmdCplt+ HPIrq+ LinkChg+
                        Control: AttnInd Unknown, PwrInd Unknown, Power- Interlock-
                SltSta: Status: AttnBtn- PowerFlt- MRL- CmdCplt- PresDet+ Interlock-
                        Changed: MRL+ PresDet- LinkState-
                [...]
        [...]
        Kernel driver in use: pcieport
        Kernel modules: shpchp
```

lspci - endpoint

```
$ sudo lspci -s 06:00 -vv
06:00.0 Serial controller: Xilinx Corporation Device 7024 (prog-if 01 [16450])
        Subsystem: Xilinx Corporation Device 0007
        Physical Slot: 3
        Control: I/O- Mem- BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- S    -
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >
        Interrupt: pin A routed to IRQ 16
        Region 0: Memory at 92800000 (32-bit, non-prefetchable) [size=8M]
        Region 1: Memory at 93000000 (32-bit, non-prefetchable) [size=64K]
        [...]
        Capabilities: [60] Express (v2) Endpoint, MSI 00
                DevCap: MaxPayload 512 bytes, PhantFunc 0, Latency L0s <64ns, L1 unlimi
                        ExtTag+ AttnBtn- AttnInd- PwrInd- RBE+ FLReset-
                DevCtl: Report errors: Correctable- Non-Fatal- Fatal- Unsupported-
                        RlxdOrd+ ExtTag+ PhantFunc- AuxPwr- NoSnoop+
                        MaxPayload 256 bytes, MaxReadReq 512 bytes
                DevSta: CorrErr- UncorrErr- FatalErr- UnsuppReq- AuxPwr- TransPend-
                LnkCap: Port #0, Speed 5GT/s, Width x4, ASPM L0s, Exit Latency L0s unli
                        ClockPM- Surprise- LLActRep- BwNot- ASPMOptComp-
                LnkCtl: ASPM Disabled; RCB 64 bytes Disabled- CommClk-
                        ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
                LnkSta: Speed 5GT/s, Width x4, TrErr- Train- SlotClk+ DLActive- BWMgmt-
                DevCap2: Completion Timeout: Range B, TimeoutDis-, LTR-, OBFF Not Suppo
                DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis-, LTR-, OBFF Disa
                LnkCtl2: Target Link Speed: 5GT/s, EnterCompliance- SpeedDis-
                         [...]
        Kernel driver in use: xdma
        Kernel modules: xdma
```

Using `setpci` to enter the compliance mode

Another system: PCIe up-link (Fujitsu server) and Struck SIS8300-KU AMC

```
$ lspci -tv
[...]
 +-[0000:85]-+-00.0-[86-8f]--+-00.0-[87-8f]--+-08.0-[88]--
 |           |                |               \-09.0-[89-8f]----00.0-[8a-8f]--+-00.0-[8b]--
 |           |                |                                               +-02.0-[8c]----00.0  Xilinx 8021
 |           |                |                                               +-04.0-[8d]--
 |           |                |                                               +-06.0-[8e]--
 |           |                |                                               \-08.0-[8f]--
 |           |                +-00.1  PLX Technology, Inc. Device 87d0
 |           |                +-00.2  PLX Technology, Inc. Device 87d0
 |           |                +-00.3  PLX Technology, Inc. Device 87d0
 |           |                \-00.4  PLX Technology, Inc. Device 87d0
[...]
```

Relevant registers

To enter the compliance mode, we must set Enter Compliance bit and then perform the Hot Reset.

PCI EXPRESS BASE SPECIFICATION, REV. 3.0

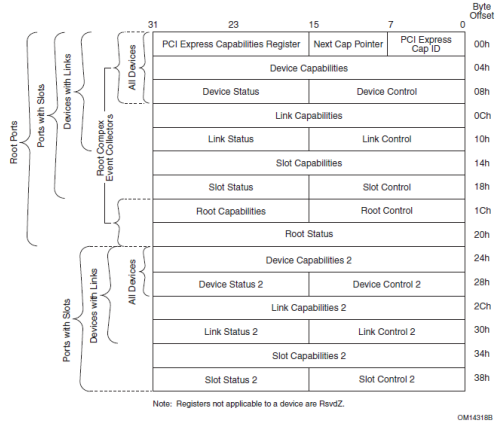


Figure 7-10: PCI Express Capability Structure

7.8.19. Link Control 2 Register (Offset 30h)

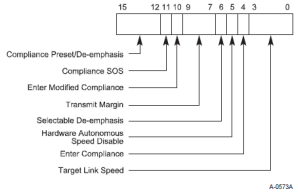


Figure 7-28: Link Control 2 Register

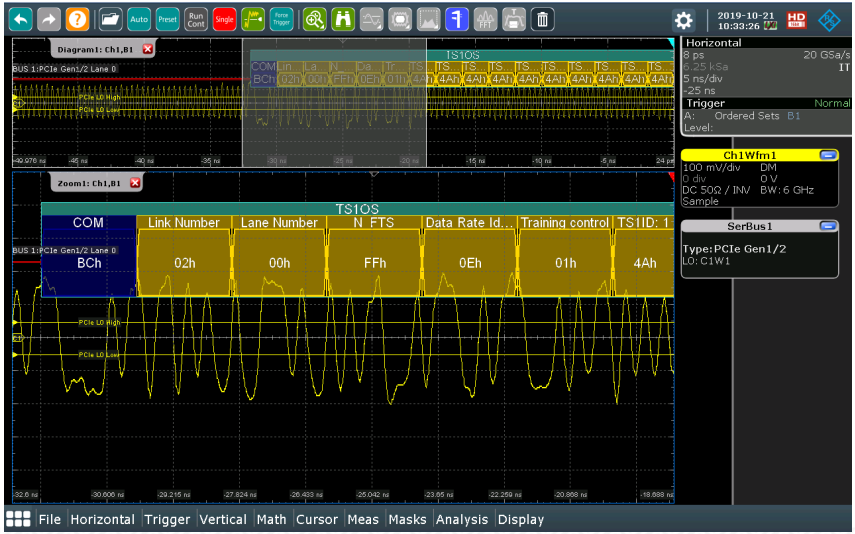
Commands to enter the compliance mode:

```
$ sudo setpci -s 8a:02 CAP_EXP+0x30.W
0003
$ sudo setpci -s 8c:00 CAP_EXP+0x30.W
0002
$ sudo setpci -s 8a:02 BRIDGE_CONTROL.W
0013

$ sudo setpci -s 8a:02 CAP_EXP+0x30.W=0x13
$ sudo setpci -s 8c:00 CAP_EXP+0x30.W=0x0012
$ sudo setpci -s 8a:02 BRIDGE_CONTROL.W=0x53
```

Compliance mode

We are now in compliance mode (both link partners are transmitting TS1 Ordered Sets):



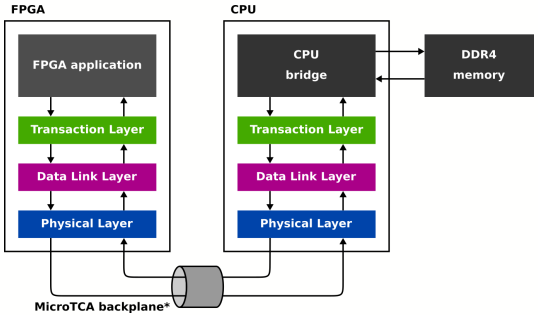
FPGA side

Explained in detail in second part of this tutorial session.

Drivers

CPU board

From ”FPGA side” we have solved the left part, now it is time to look at the right part:



- Typically there is one (or more) CPU AMCs is MicroTCA systems
- Usually (in experimental physics) the CPU runs GNU/Linux
- DESY runs Ubuntu 16.04 LTS, Ubuntu 18.04 LTS is also in use
- PCIe hotplug was modernized in Linux kernel 4.19[1] → available in Ubuntu 20.04 LTS

[1] `https://lwn.net/Articles/767885/`

Linux pci subsystem

- Linux has a subsystem responsible for managing PCI/PCIe device driver
- Hot-plug/hot-swap module (pciehp) is also part of Linux kernel
- Device-specific driver register by providing pci_driver structure to pci_register_driver() function

Linux pci subsystem

https://elixir.bootlin.com/linux/v4.15/source/include/linux/pci.h#L744

```
struct pci_driver {
    struct list_head node;
    const char *name;
    const struct pci_device_id *id_table;   /* must be non-NULL for probe to be called */
    int  (*probe)  (struct pci_dev *dev, const struct pci_device_id *id);   /* New device
        inserted */
    void (*remove) (struct pci_dev *dev);   /* Device removed (NULL if not a hot-plug
        capable driver) */
    int  (*suspend) (struct pci_dev *dev, pm_message_t state);  /* Device suspended */
    int  (*suspend_late) (struct pci_dev *dev, pm_message_t state);
    int  (*resume_early) (struct pci_dev *dev);
    int  (*resume) (struct pci_dev *dev);                    /* Device woken up */
    void (*shutdown) (struct pci_dev *dev);
    int (*sriov_configure) (struct pci_dev *dev, int num_vfs); /* PF pdev */
    const struct pci_error_handlers *err_handler;
    const struct attribute_group **groups;
    struct device_driver    driver;
    struct pci_dynids dynids;
};
```

Xilinx driver

Xilinx driver only provides probe() and remove() functions - enough to perform hot-swap.

from https://github.com/Xilinx/dma_ip_drivers/blob/8b8c70b697f049649d5fa99be9c6bc4302d89ac9/XDMA/linux-kernel/xdma/xdma_mod.c#L316:

```
static struct pci_driver pci_driver = {
        .name = DRV_MODULE_NAME,
        .id_table = pci_ids,
        .probe = probe_one,
        .remove = remove_one,
        .err_handler = &xdma_err_handler,
};
```

Interface to user space

On the other side (towards the user space), Xilinx DMA driver provides several char devices:

```
$ ll /dev | grep xdma
drwxr-xr-x  3 root root           60 Jun 20 17:19 xdma
crw-rw-rw-  1 root root     238,  36 Jun 20 17:19 xdma0_c2h_0
crw-rw-rw-  1 root root     238,   1 Jun 20 17:19 xdma0_control
crw-rw-rw-  1 root root     238,  10 Jun 20 17:19 xdma0_events_0
crw-rw-rw-  1 root root     238,  11 Jun 20 17:19 xdma0_events_1
[...]
crw-rw-rw-  1 root root     238,  24 Jun 20 17:19 xdma0_events_14
crw-rw-rw-  1 root root     238,  25 Jun 20 17:19 xdma0_events_15
crw-rw-rw-  1 root root     238,  32 Jun 20 17:19 xdma0_h2c_0
crw-rw-rw-  1 root root     238,   0 Jun 20 17:19 xdma0_user
crw-rw-rw-  1 root root     238,   2 Jun 20 17:19 xdma0_xvc
```

Interface to user space

On the other side (towards the user space), Xilinx DMA driver provides several char devices:

```
$ ll /dev | grep xdma
drwxr-xr-x  3 root root           60 Jun 20 17:19 xdma
crw-rw-rw-  1 root root     238,  36 Jun 20 17:19 xdma0_c2h_0
crw-rw-rw-  1 root root     238,   1 Jun 20 17:19 xdma0_control
crw-rw-rw-  1 root root     238,  10 Jun 20 17:19 xdma0_events_0
crw-rw-rw-  1 root root     238,  11 Jun 20 17:19 xdma0_events_1
[...]
crw-rw-rw-  1 root root     238,  24 Jun 20 17:19 xdma0_events_14
crw-rw-rw-  1 root root     238,  25 Jun 20 17:19 xdma0_events_15
crw-rw-rw-  1 root root     238,  32 Jun 20 17:19 xdma0_h2c_0
crw-rw-rw-  1 root root     238,   0 Jun 20 17:19 xdma0_user
crw-rw-rw-  1 root root     238,   2 Jun 20 17:19 xdma0_xvc
```

c2h
AXI4 port (DMA from device)

h2c
AXI4 port (DMA to device)

user
AXI4-Lite port

events_N
Interrupts

Interface to user space

On the other side (towards the user space), Xilinx DMA driver provides several char devices:

```
$ ll /dev | grep xdma
drwxr-xr-x  3 root root           60 Jun 20 17:19 xdma
crw-rw-rw-  1 root root     238,  36 Jun 20 17:19 xdma0_c2h_0
crw-rw-rw-  1 root root     238,   1 Jun 20 17:19 xdma0_control
crw-rw-rw-  1 root root     238,  10 Jun 20 17:19 xdma0_events_0
crw-rw-rw-  1 root root     238,  11 Jun 20 17:19 xdma0_events_1
[...]
crw-rw-rw-  1 root root     238,  24 Jun 20 17:19 xdma0_events_14
crw-rw-rw-  1 root root     238,  25 Jun 20 17:19 xdma0_events_15
crw-rw-rw-  1 root root     238,  32 Jun 20 17:19 xdma0_h2c_0
crw-rw-rw-  1 root root     238,   0 Jun 20 17:19 xdma0_user
crw-rw-rw-  1 root root     238,   2 Jun 20 17:19 xdma0_xvc
```

c2h
AXI4 port (DMA from device)

h2c
AXI4 port (DMA to device)

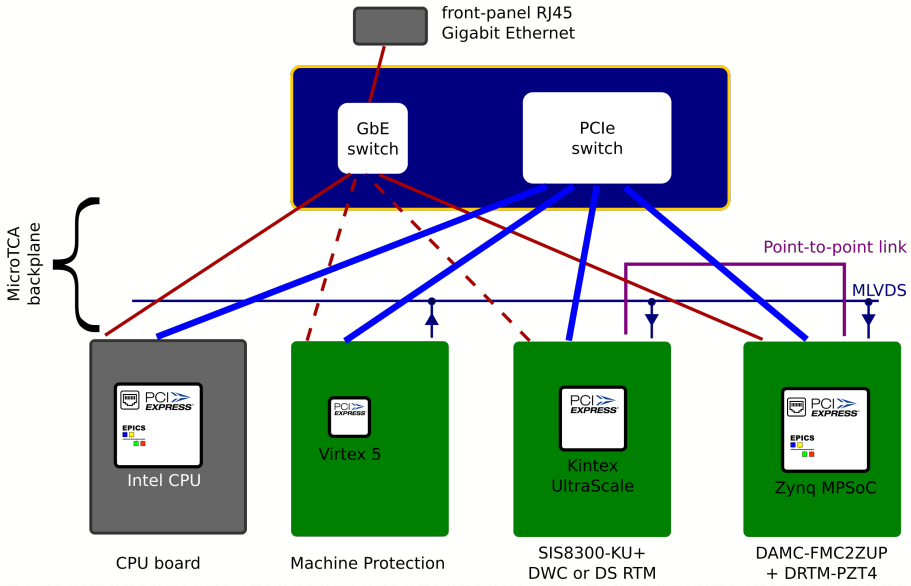
user
AXI4-Lite port

events_N
Interrupts

Demo on a real HW in second part

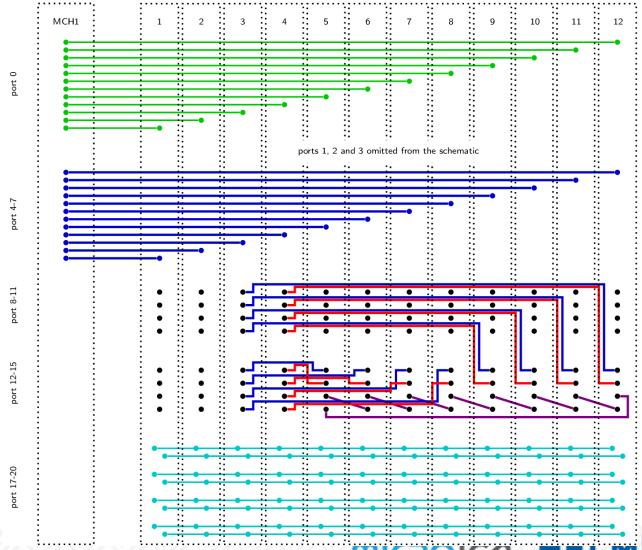
MicroTCA and PCI Express

Example of a MicroTCA system

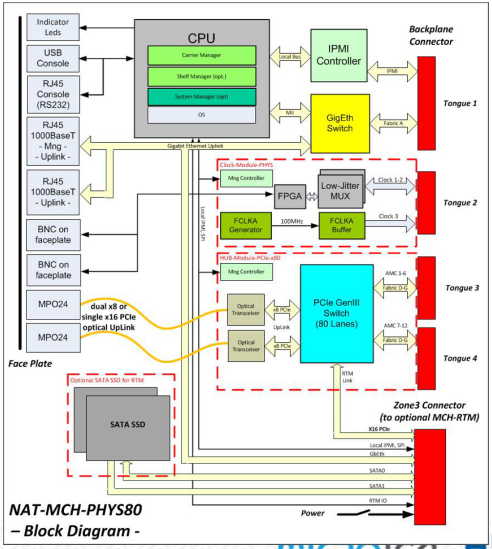


Backplane

On a typical backplane there are 4 ports (= 2 differential pairs) connected between AMC slots and MCH slot.



MCH contains a PCIe switch with N lanes (80 in the example shown below, 4*12 for AMC slots, 16 for PCIe up-link and 16 for MCH-RTM).



Hot Plug and Hot Swap

Definitions (from Wikipedia):

Hot swapping [...] is replacing or adding components without stopping or shutting down the system.
Hot plugging on the other hand describes only the addition of components that would expand the system without significant interruption to the system.

- In MicroTCA hot-swap provides a possibility to replace individual cards in a running crate, without rebooting the crate or CPU
- Speeds up development time - no need to reboot CPU every time FPGA image is changed

PCIe hot-swap and hot-plug is getting more popular/needed for various applications:

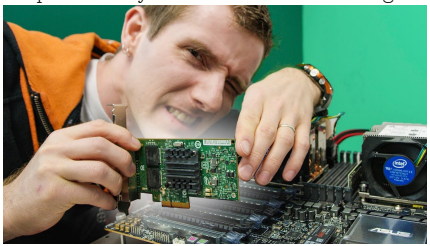
> *... PCI Express (PCIe), instead, supported hotplug from the get-go in 2002, but its embodiments have changed over time. Originally intended to hot-swap PCIe cards in servers or ExpressCards in laptops, today it is commonly used in data centers (where NVMe flash drives need to be swapped at runtime) and by Thunderbolt (which tunnels PCIe through a hotpluggable chain of converged I/O switches, together with other protocols such as DisplayPort).*

from "The modernization of PCIe hotplug in Linux" (`https://lwn.net/Articles/767885/`)

A couple of interesting (but not very informative) videos on YouTube:

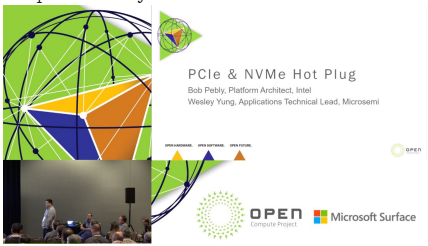
I've NEVER been so FRUSTRATED... Hot-Swapping PCIe Cards

https://www.youtube.com/watch?v=YigN2mkQMPc



Lightning (PCIe JBOF): Update, Challenges, and Solutions

https://www.youtube.com/watch?v=GJ6B0xzgvlM



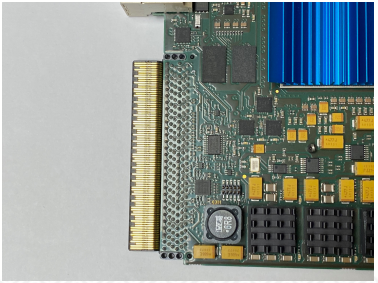
Facilities needed

- MicroTCA hot-swap
  - Mechanical
  - Management
- PCIe hot-plug/hot-swap
  - PCIe switch
  - Interface between PCIe switch and MCMC on MCH
  - Operating system support

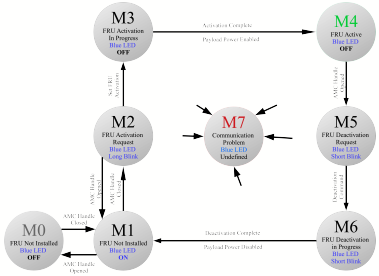
Facilities needed

- MicroTCA hot-swap
  - Mechanical
  - Management
- PCIe hot-plug/hot-swap
  - PCIe switch
  - Interface between PCIe switch and MCMC on MCH
  - Operating system support

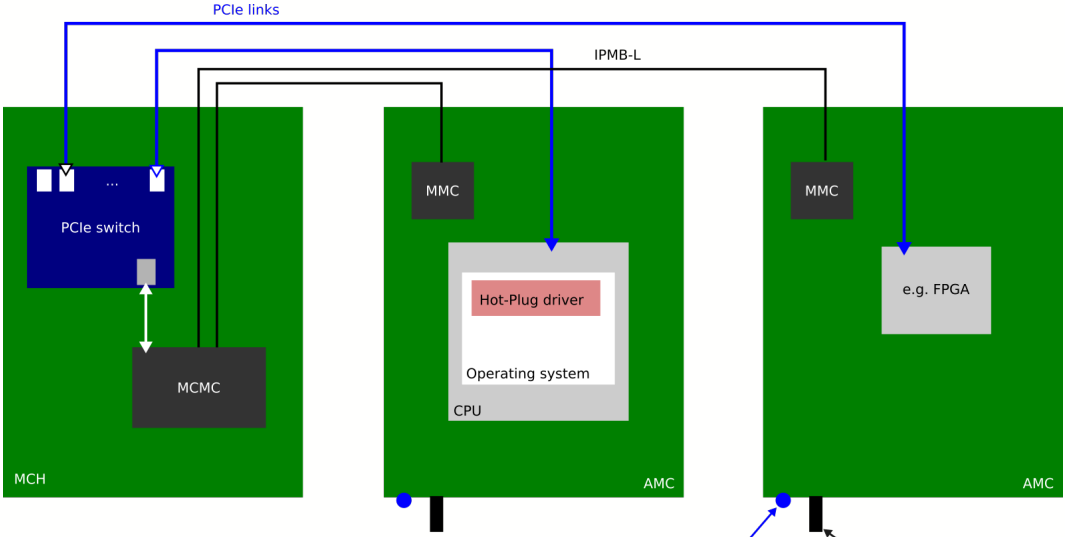
AMC connector providing first, ..., last mate contacts



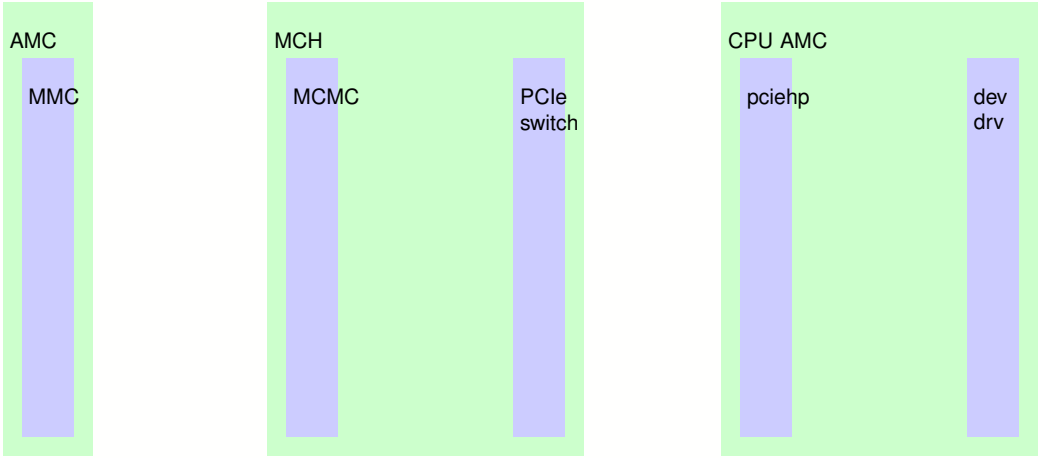
M-state finite state machine



Components which are involved in hot-swap/hot-plug in MicroTCA:

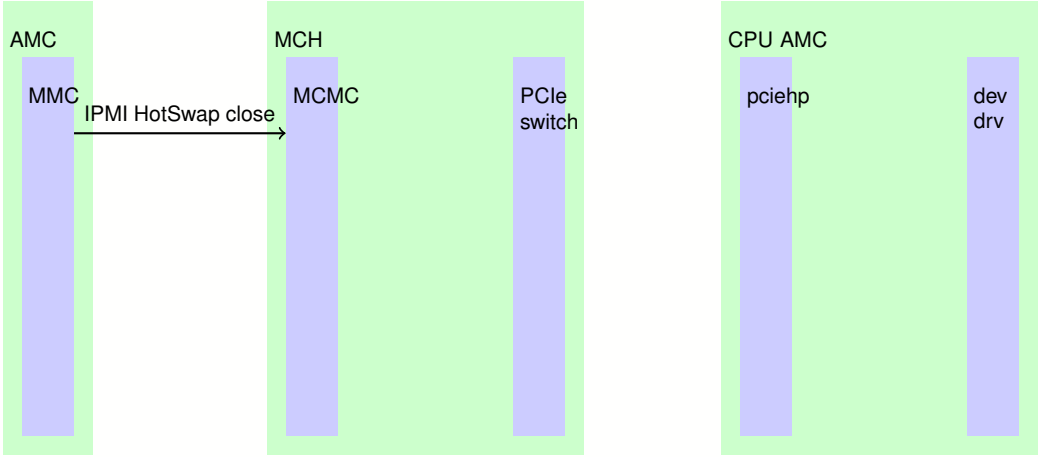


AMC insertion sequence



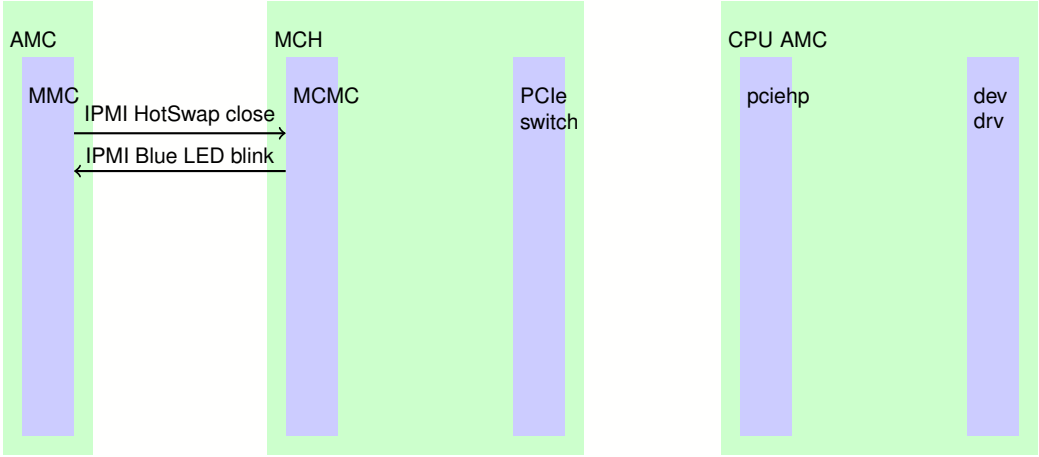
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC insertion sequence



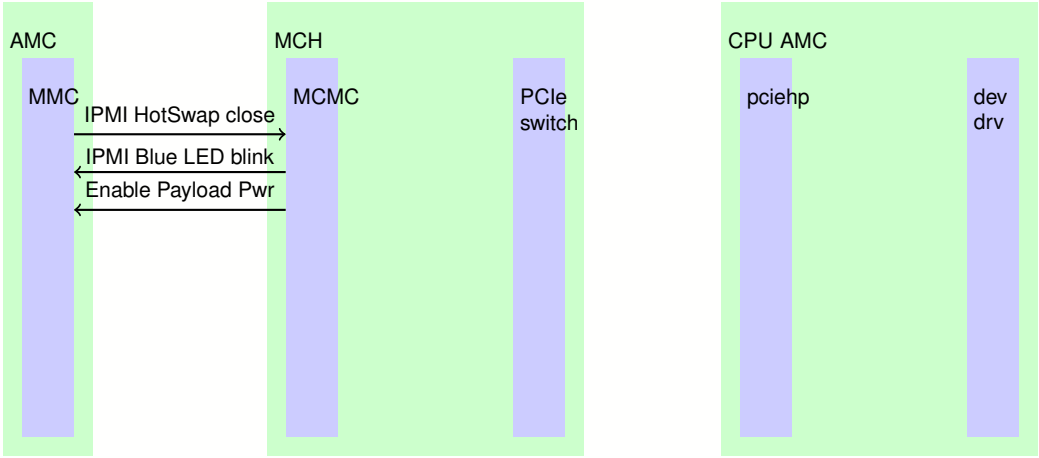
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
`https://webcast.desy.de/index.jsp?m=201312`

AMC insertion sequence



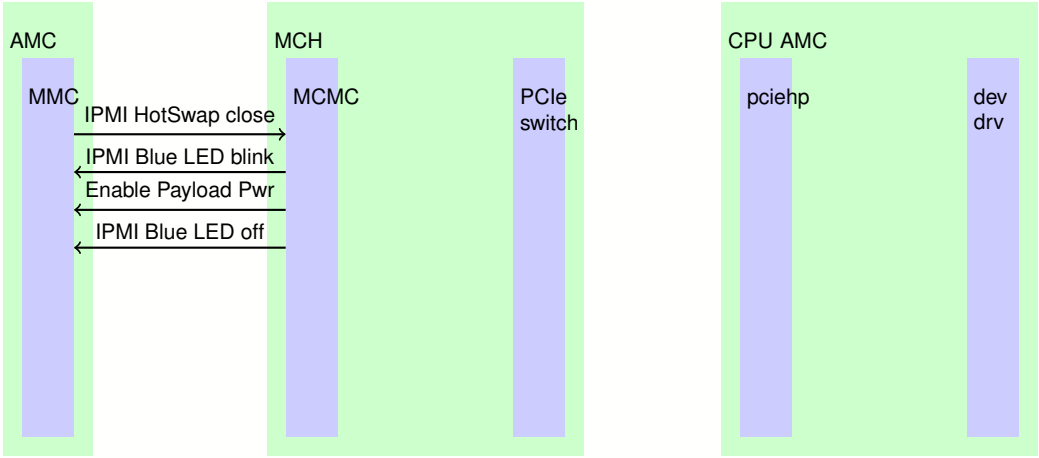
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC insertion sequence



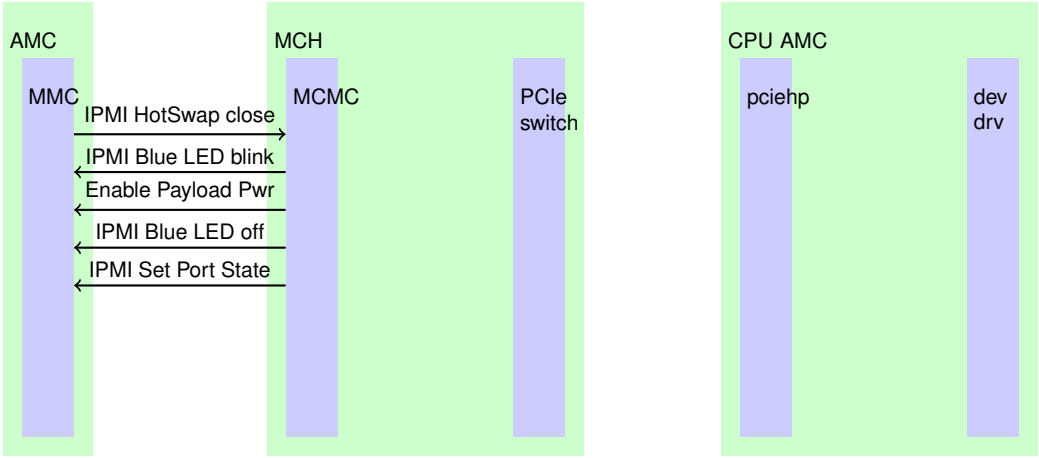
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC insertion sequence



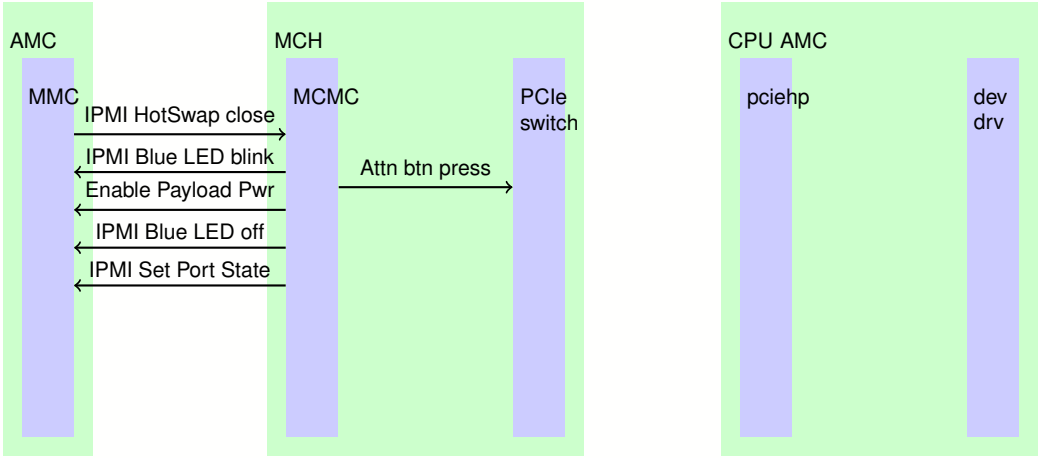
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC insertion sequence



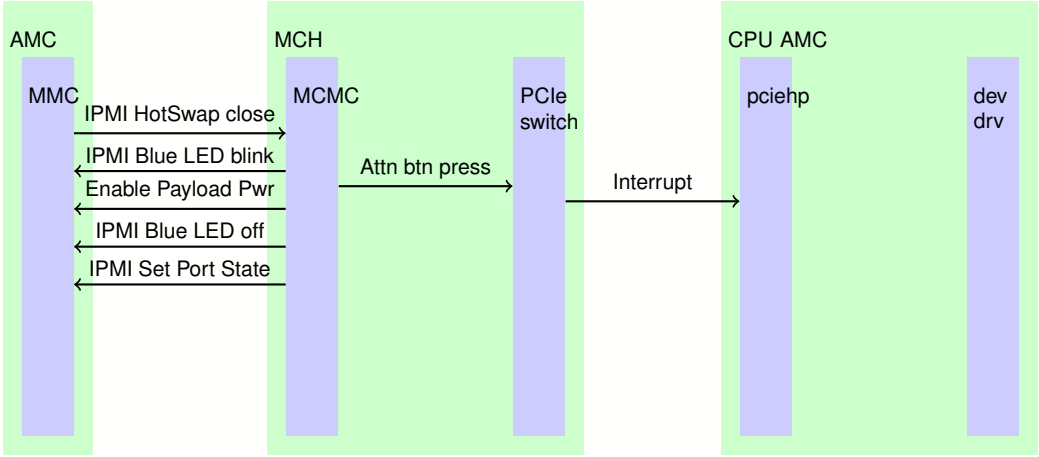
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC insertion sequence



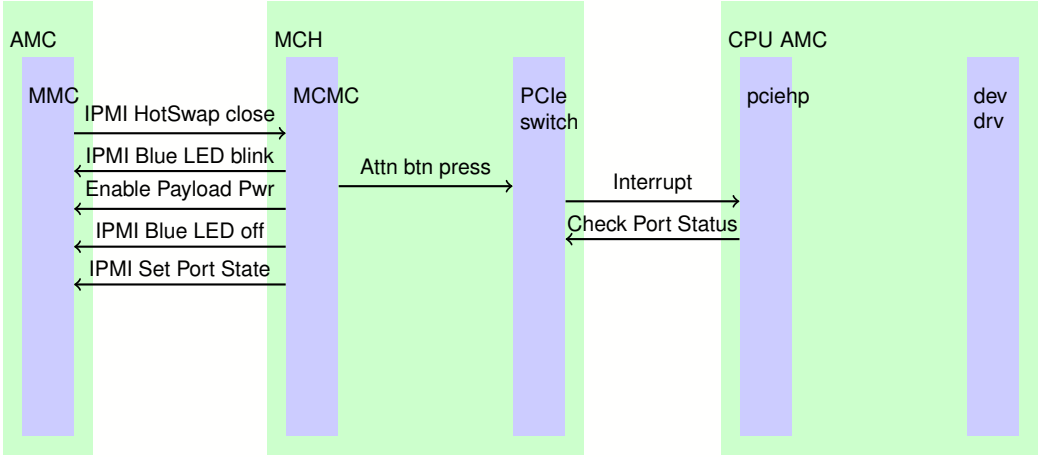
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC insertion sequence



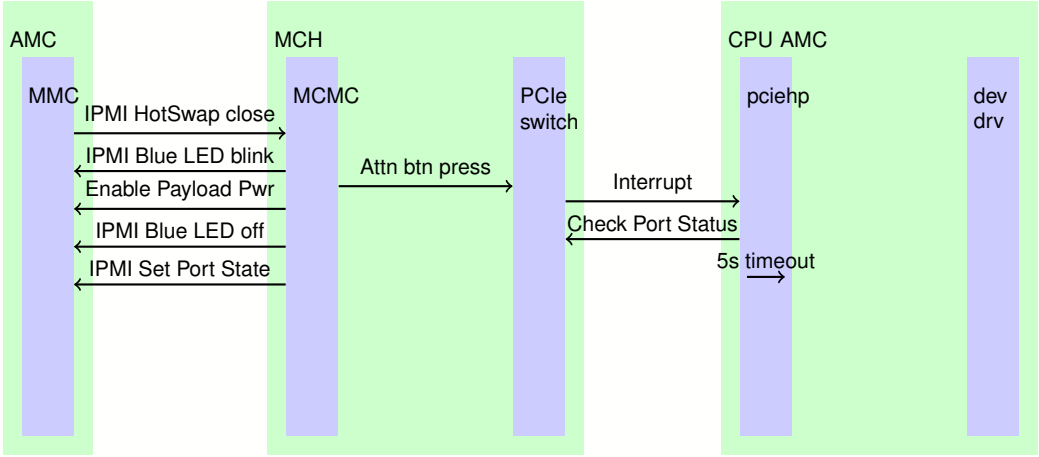
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC insertion sequence



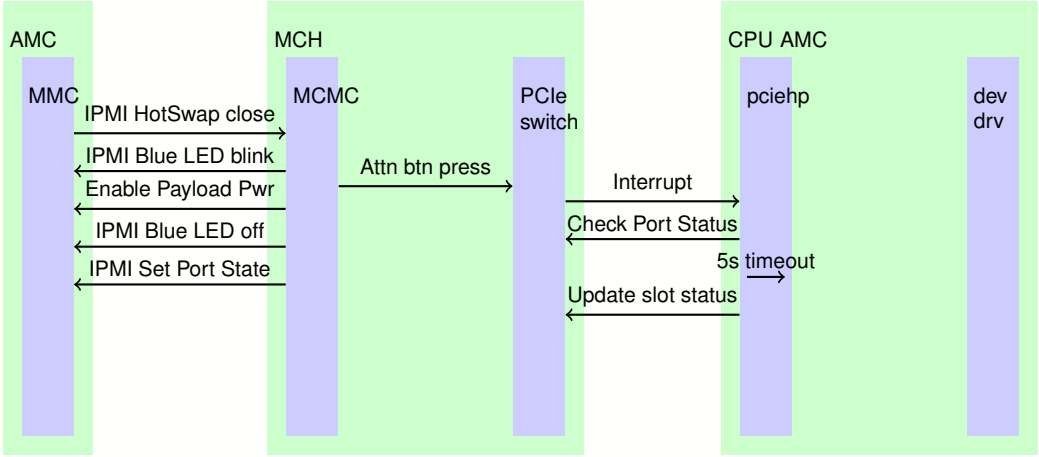
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC insertion sequence



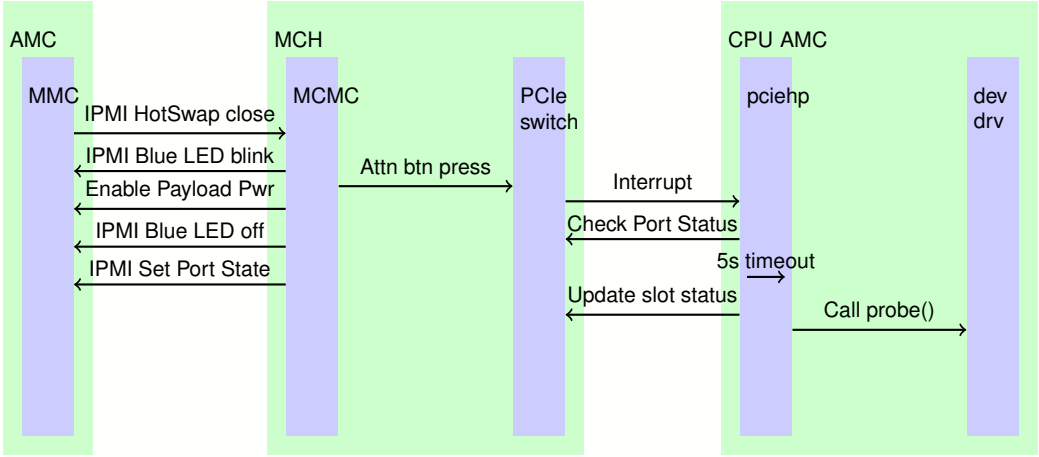
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC insertion sequence



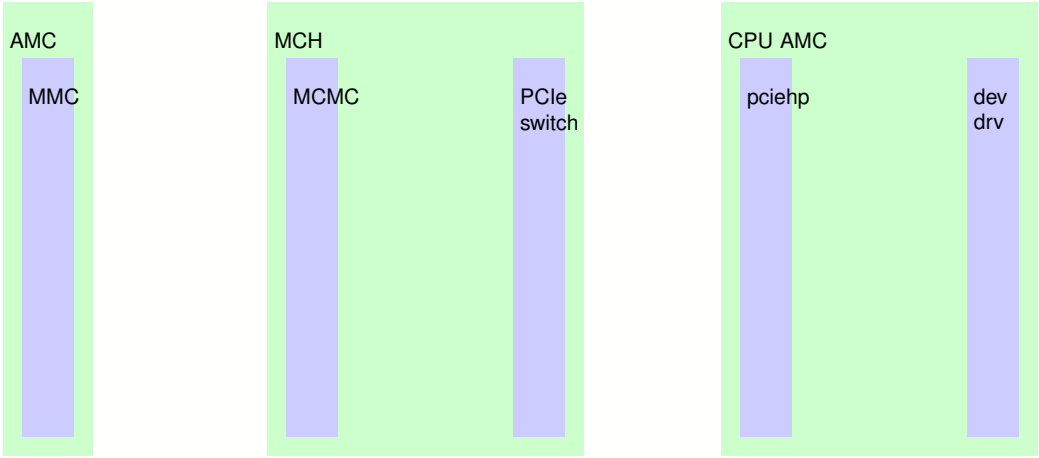
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC insertion sequence



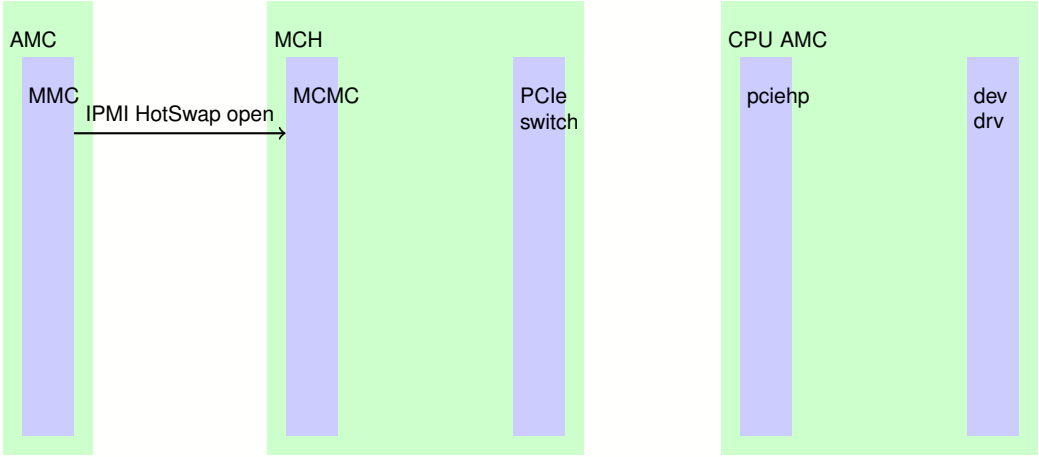
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC removal sequence



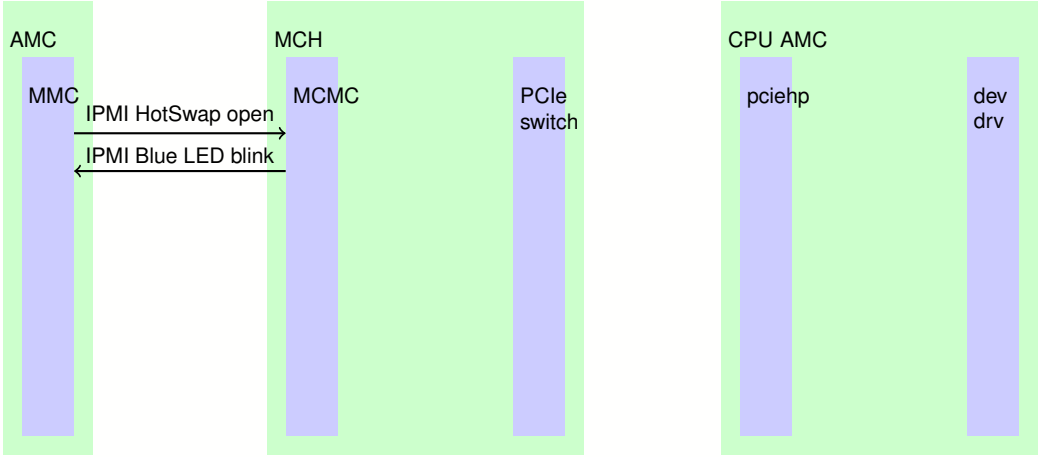
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC removal sequence



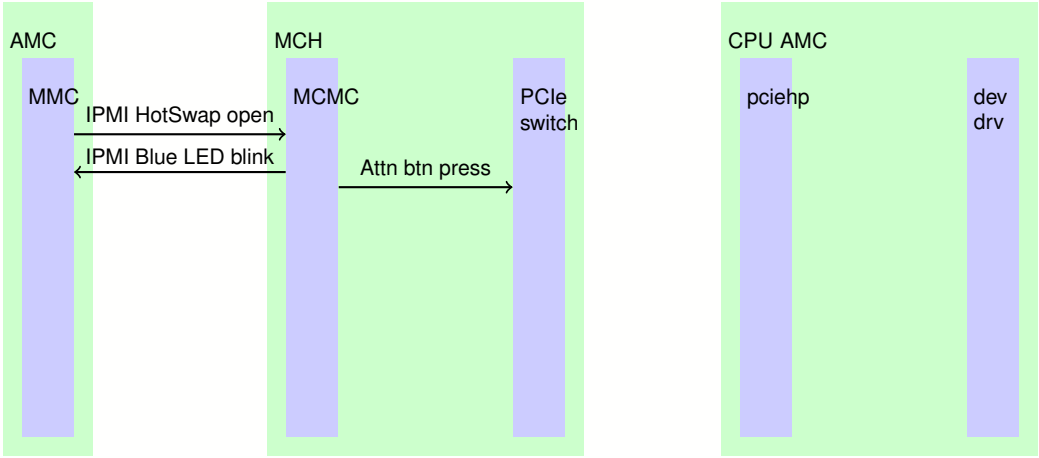
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
`https://webcast.desy.de/index.jsp?m=201312`

AMC removal sequence



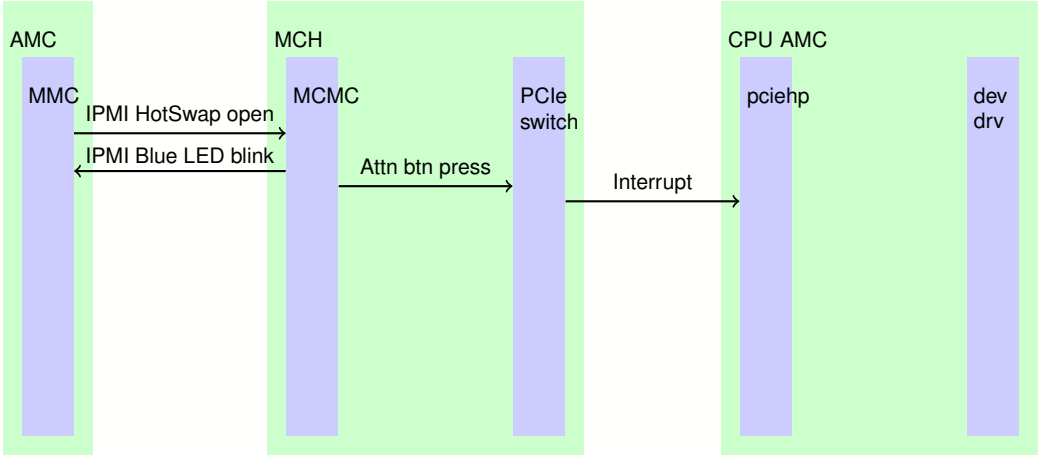
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC removal sequence



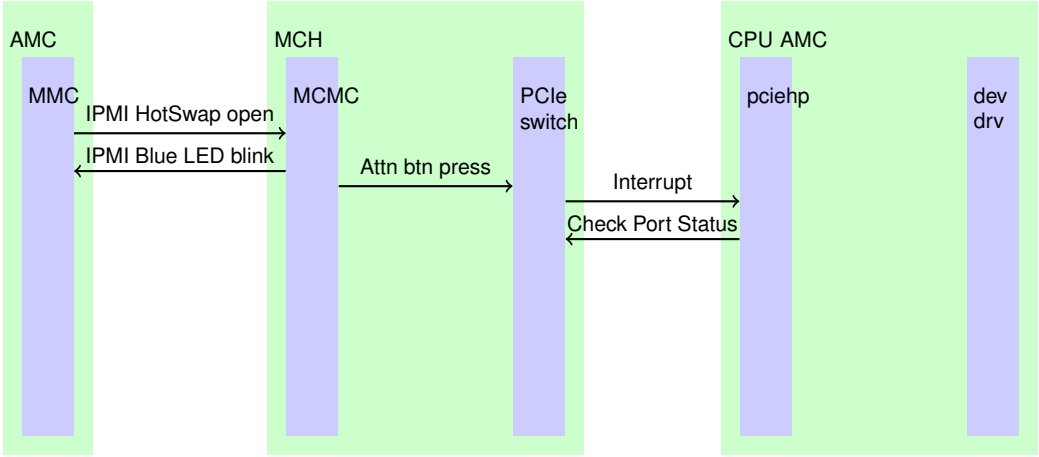
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
`https://webcast.desy.de/index.jsp?m=201312`

AMC removal sequence



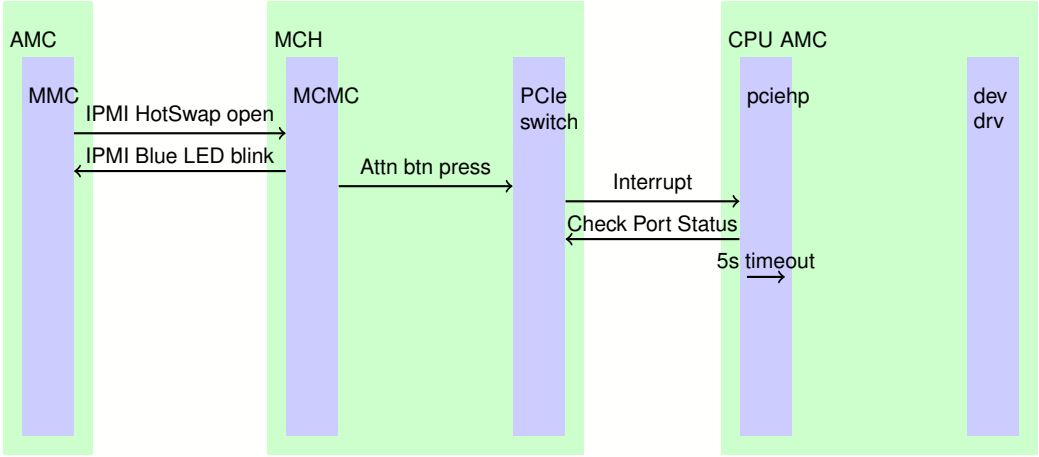
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC removal sequence



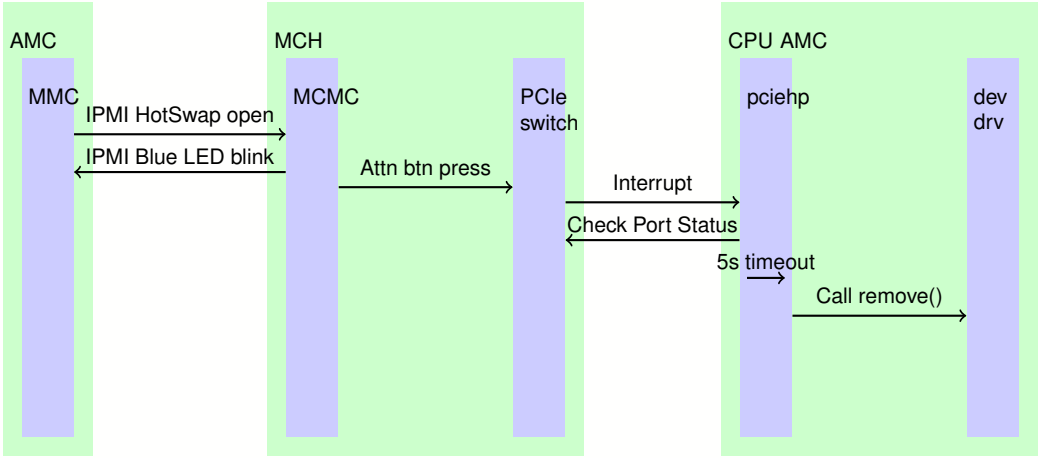
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC removal sequence



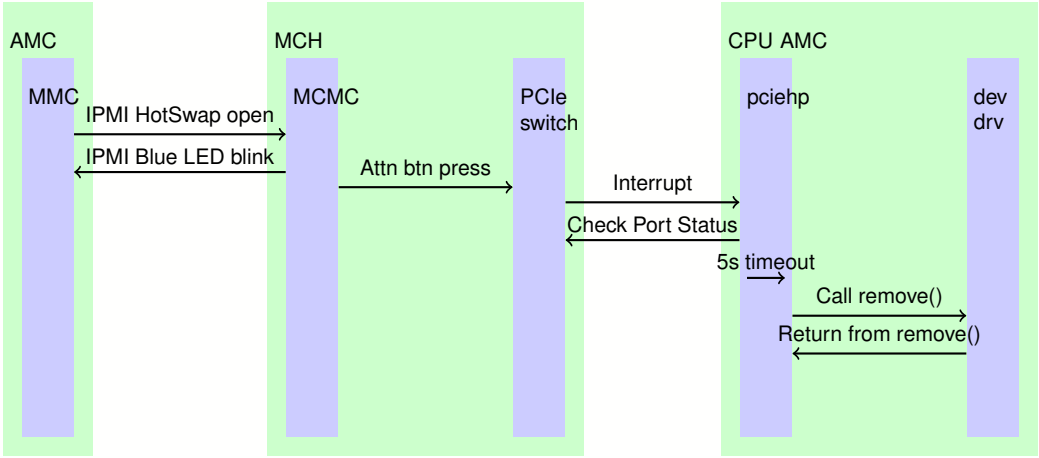
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC removal sequence



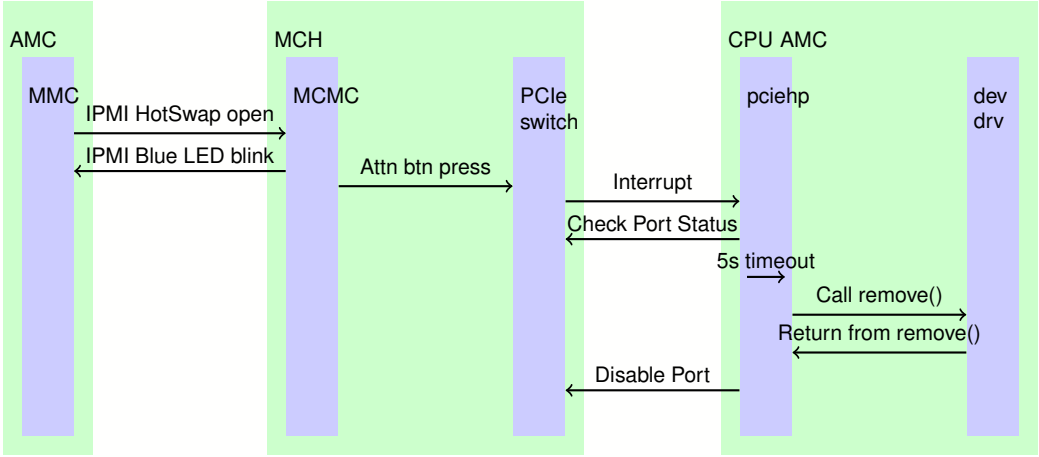
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC removal sequence



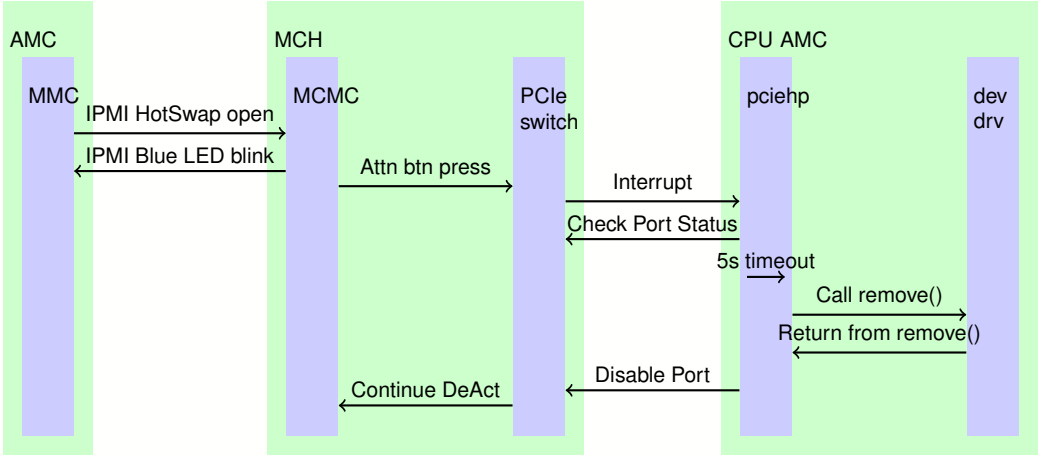
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC removal sequence



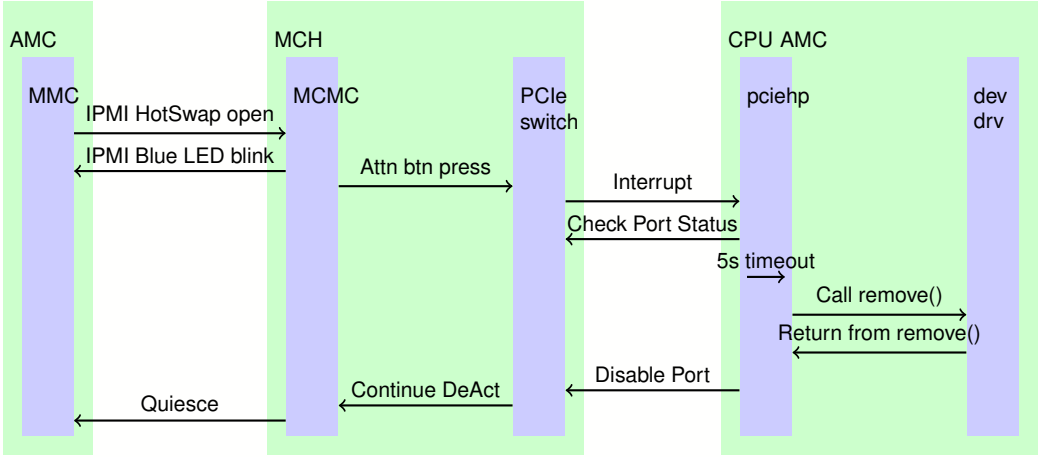
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC removal sequence



Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

AMC removal sequence



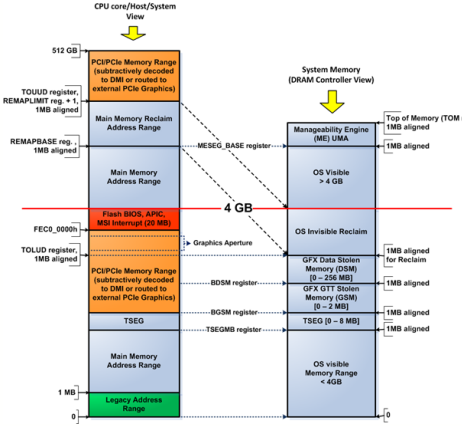
Inspired by L. Petrosyan: MicroTCA and PCIe HotSwap under Linux,
https://webcast.desy.de/index.jsp?m=201312

Hot Plug

- From MicroTCA perspective Hot Plug performs the same steps as for Hot Swap insertion
- The main concern here is the allocation of the x86 memory space
- CPU addresses $\neq$ memory addresses
- cca 1 GB for PCIe devices (in the lower 4 GB = 32-bit addr)

```
[  0.070024] [mem 0x7f800000-0xdfffffff]
available for PCI devices
```

- Enough space needs to be assigned to each slot to accommodate the hot-plugable card's BARs



from https://resources.infosecinstitute.com/system-address-map-initialization-x86x64-architecture-part-2-pci-express-based-systems/

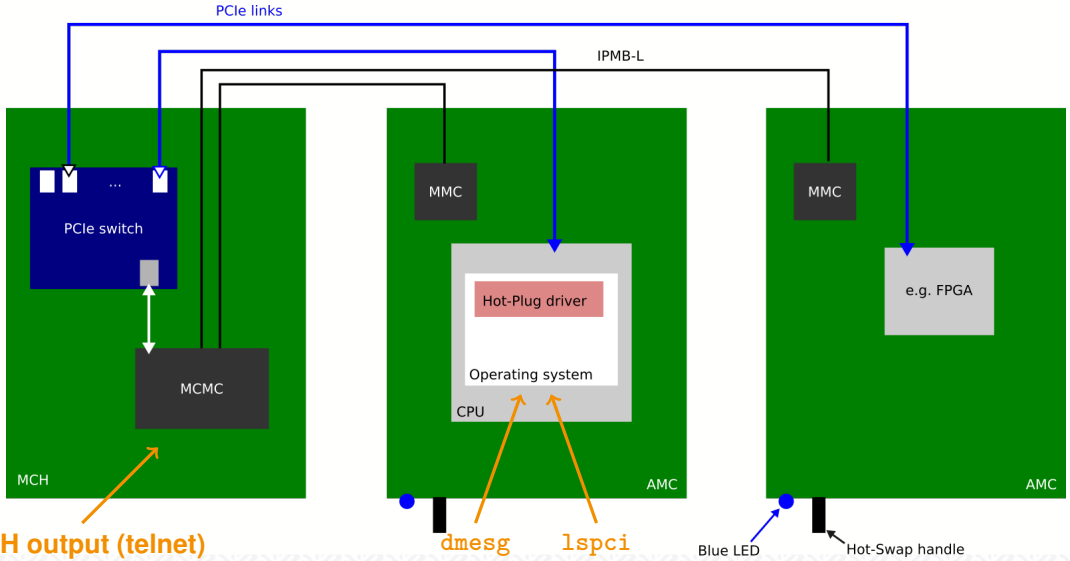
Linux kernel arguments for Hot Plug

- Linux provides options to change the behavior of the kernel
- Full list available at
  `https://www.kernel.org/doc/html/v4.14/admin-guide/kernel-parameters.html`
- To control the amount of space allocated per each slot, hpmemsize is used
- Be careful with this argument, setting it too high can leave no space in 32-bit region for later slots
- `realloc` is also needed
- Procedure:
  - add `pci=realloc,hpmemsize=16M` to `GRUB_CMDLINE_LINUX_DEFAULT` in `/etc/default/grub`
  - `sudo update-grub`
  - `reboot`

```
$ uname -a
Linux CPU-110219 4.15.0-70-generic #79~16.04.1-Ubuntu SMP Tue Nov 12 14:01:10 UTC
    2019 x86_64 x86_64 x86_64 GNU/Linux
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-4.15.0-70-generic root=UUID=3664d7dd-8a29-4e82-90a5-
    cab8157b94e5 ro quiet pci=realloc,hpmemsize=16M
```

Observation of the hot-plug sequence

We can observe the status change at several different points in the system:



Observation of the hot-plug sequence

At CPU (lspci for slot) - before AMC insertion:

```
Capabilities: [68] Express (v2) Downstream Port (Slot+), MSI 00
    DevCap:     MaxPayload 512 bytes, PhantFunc 0
        ExtTag- RBE+
    DevCtl:     Report errors: Correctable+ Non-Fatal+ Fatal+ Unsupported+
        RlxdOrd+ ExtTag- PhantFunc- AuxPwr- NoSnoop+
        MaxPayload 256 bytes, MaxReadReq 128 bytes
    DevSta:     CorrErr+ UncorrErr- FatalErr- UnsuppReq+ AuxPwr- TransPend-
    LnkCap:     Port #1, Speed 8GT/s, Width x4, ASPM L1, Exit Latency L0s <4us, L1 <4us
        ClockPM- Surprise+ LLActRep+ BwNot+ ASPMOptComp+
    LnkCtl:     ASPM Disabled; Disabled- CommClk-
        ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
    LnkSta:     Speed 8GT/s, Width x0, TrErr- Train- SlotClk- DLActive- BWMgmt- ABWMgmt-
    SltCap:     AttnBtn+ PwrCtrl+ MRL+ AttnInd- PwrInd- HotPlug+ Surprise-
        Slot #4, PowerLimit 25.000W; Interlock- NoCompl-
    SltCtl:     Enable: AttnBtn+ PwrFlt- MRL- PresDet- CmdCplt+ HPIrq+ LinkChg+
        Control: AttnInd Unknown, PwrInd Unknown, Power+ Interlock-
    SltSta:     Status: AttnBtn- PowerFlt- MRL+ CmdCplt- PresDet+ Interlock-
        Changed: MRL+ PresDet- LinkState-
    DevCap2: Completion Timeout: Not Supported, TimeoutDis-, LTR+, OBFF Via message ARIFwd+
    DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis-, LTR-, OBFF Disabled ARIFwd-
    LnkCtl2: Target Link Speed: 8GT/s, EnterCompliance- SpeedDis-, Selectable De-emphasis: -6dB
             Transmit Margin: Normal Operating Range, EnterModifiedCompliance- ComplianceSOS-
             Compliance De-emphasis: -6dB
    LnkSta2: Current De-emphasis Level: -6dB, EqualizationComplete+, EqualizationPhase1+
             EqualizationPhase2+, EqualizationPhase3+, LinkEqualizationRequest-
```

Observation of the hot-plug sequence

At MCH - AMC insertion:

```
AMC3(7): Handle=0x01 - closed
AMC3(7): PCIe Hot Plug Delay active !
LSHM(0): FRU 7 sensor 105 LUN 0 'HS 007 AMC3' hotswap M1->M2
LSHM(0): FRU 7 sensor 105 LUN 0 'HS 007 AMC3' hotswap M2->M3
LSHM(0): FRU 7 sensor 128 LUN 0 'Hot Swap' hotswap M1->M2
LSHM(0): FRU 7 sensor 128 LUN 0 'Hot Swap' hotswap M2->M3
processMsg: No request pending for I2C=0xc2 - PM_HEARTBEAT_RSP
EKey(AMC3): Channel(0): Port[4 5 6 7] - used:[4 5 6 7] PCIe downstream Gen 1, no SSC
EKey(AMC3): Channel(1): Port[0 - - -] - used:[0] Eth  1000Base-BX
LSHM(0): FRU 7 sensor 105 LUN 0 'HS 007 AMC3' hotswap M3->M4
LSHM(0): FRU 7 sensor 128 LUN 0 'Hot Swap' hotswap M3->M4
LS_MON: PCIe - AMC3/4-7 up
```

Observation of the hot-plug sequence

At CPU (Linux kernel output - dmesg) - AMC insertion:

```
[ 1580.279023] pciehp 0000:04:03.0:pcie204: Slot(3): Attention button pressed
[ 1580.279070] pciehp 0000:04:03.0:pcie204: Slot(3) Powering on due to button press
[ 1580.422133] pciehp 0000:04:03.0:pcie204: Slot(3): Link Up
[ 1580.636478] pci 0000:06:00.0: [10ee:7024] type 00 class 0x070001
[ 1580.636517] pci 0000:06:00.0: reg 0x10: [mem 0x00000000-0x007fffff]
[ 1580.636529] pci 0000:06:00.0: reg 0x14: [mem 0x00000000-0x0000ffff]
[ 1580.636591] pci 0000:06:00.0: Max Payload Size set to 256 (was 128, max 512)
[ 1580.636597] pci 0000:06:00.0: enabling Extended Tags
[ 1580.636721] pci 0000:06:00.0: PME# supported from D0 D1 D2 D3hot
[ 1580.636914] pci 0000:06:00.0: BAR 0: assigned [mem 0x99000000-0x997fffff]
[ 1580.636923] pci 0000:06:00.0: BAR 1: assigned [mem 0x98c00000-0x98c0ffff]
[ 1580.636931] pcieport 0000:04:03.0: PCI bridge to [bus 06]
[ 1580.636935] pcieport 0000:04:03.0:   bridge window [io  0x3000-0x3fff]
[ 1580.636942] pcieport 0000:04:03.0:   bridge window [mem 0x98c00000-0x9abfffff]
[ 1580.636948] pcieport 0000:04:03.0:   bridge window [mem 0x92c00000-0x93bfffff 64bit pref]
[ 1580.637021] serial 0000:06:00.0: enabling device (0000 -> 0002)
[ 1580.637627] xdma:xdma_device_open: xdma device 0000:06:00.0, 0x00000000f9f371d8.
[ 1580.637898] xdma:map_single_bar: BAR0 at 0x99000000 mapped at 0x00000000c9cd89ae, length=8388608(/8388608)
[ 1580.637981] xdma:map_single_bar: BAR1 at 0x98c00000 mapped at 0x000000007e36fe64, length=65536(/65536)
...
```

Observation of the hot-plug sequence

At CPU (lspci for slot) - after AMC insertion:

```
Capabilities: [68] Express (v2) Downstream Port (Slot+), MSI 00
        DevCap:     MaxPayload 512 bytes, PhantFunc 0
            ExtTag- RBE+
        DevCtl:     Report errors: Correctable+ Non-Fatal+ Fatal+ Unsupported+
            RlxdOrd+ ExtTag- PhantFunc- AuxPwr- NoSnoop+
            MaxPayload 256 bytes, MaxReadReq 128 bytes
        DevSta:     CorrErr+ UncorrErr- FatalErr- UnsuppReq+ AuxPwr- TransPend-
        LnkCap:     Port #1, Speed 8GT/s, Width x4, ASPM L1, Exit Latency L0s <4us, L1 <4us
            ClockPM- Surprise+ LLActRep+ BwNot+ ASPMOptComp+
        LnkCtl:     ASPM Disabled; Disabled- CommClk-
            ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
        LnkSta:     Speed 8GT/s, Width x4, TrErr- Train- SlotClk- DLActive+ BWMgmt- ABWMgmt-
        SltCap:     AttnBtn+ PwrCtrl+ MRL+ AttnInd- PwrInd- HotPlug+ Surprise-
            Slot #4, PowerLimit 25.000W; Interlock- NoCompl-
        SltCtl:     Enable: AttnBtn+ PwrFlt- MRL- PresDet- CmdCplt+ HPIrq+ LinkChg+
            Control: AttnInd Unknown, PwrInd Unknown, Power- Interlock-
        SltSta:     Status: AttnBtn- PowerFlt- MRL- CmdCplt- PresDet+ Interlock-
            Changed: MRL- PresDet- LinkState-
        DevCap2: Completion Timeout: Not Supported, TimeoutDis-, LTR+, OBFF Via message ARIFwd+
        DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis-, LTR+, OBFF Disabled ARIFwd-
        LnkCtl2: Target Link Speed: 8GT/s, EnterCompliance- SpeedDis-, Selectable De-emphasis: -6dB
                 Transmit Margin: Normal Operating Range, EnterModifiedCompliance- ComplianceSOS-
                 Compliance De-emphasis: -6dB
        LnkSta2: Current De-emphasis Level: -6dB, EqualizationComplete+, EqualizationPhase1+
                 EqualizationPhase2+, EqualizationPhase3+, LinkEqualizationRequest-
```

Observation of the hot-plug sequence

At MCH - AMC removal:

```
AMC3(7): Handle=0x02 - opened
AMC3(7): disable RTM...
AMC3(7): waiting for PCIe HPPwrEn signal
ekey_DisableAMCPorts(7): disabling all ports
AMC3: channel 1 disabled, Ports[0 ]
AMC3: channel 0 disabled, Ports[4 5 6 7]
LSHM(0): FRU 7 sensor 128 LUN 0 'Hot Swap' hotswap M4->M5
LSHM(0): FRU 7 sensor 105 LUN 0 'HS 007 AMC3' hotswap M4->M5
LSHM(0): FRU 7 sensor 128 LUN 0 'Hot Swap' hotswap M5->M6
Handle state=0x06 - quiesced
clk_disableFCLK(60): disabling FCLK-A for AMC3
LSHM(0): FRU 7 sensor 105 LUN 0 'HS 007 AMC3' hotswap M5->M6
LSHM(0): FRU 7 sensor 128 LUN 0 'Hot Swap' hotswap M6->M1
LSHM(0): FRU 7 sensor 105 LUN 0 'HS 007 AMC3' hotswap M6->M1
LS_MON: PCIe - AMC3/4-7 down
```

Observation of the hot-plug sequence

At CPU (Linux kernel output - dmesg) - AMC removal:

```
[ 1562.926931] pciehp 0000:04:03.0:pcie204: Slot(3): Attention button pressed
[ 1562.926979] pciehp 0000:04:03.0:pcie204: Slot(3): Powering off due to button press
[ 1568.076460] xdma:remove_one: pdev 0x00000000ddccb157, xdev 0x000000007d418c5c, 0x000000008a9a58db.
[ 1568.076464] xdma:xpdev_free: xpdev 0x000000007d418c5c, destroy_interfaces, xdev 0x000000008a9a58db.
[ 1568.077917] xdma:xpdev_free: xpdev 0x000000007d418c5c, xdev 0x000000008a9a58db xdma_device_close.
[ 1568.186289] pciehp 0000:04:03.0:pcie204: Slot(3): Link Down
[ 1569.868459] pciehp 0000:04:03.0:pcie204: Slot(3): Already disabled
```

Conclusion 1st part

- PCI Express is widely used in MicroTCA
- Good support in FPGAs, low latency, high throughput
- A lot of diagnostics/configuration:
  - `lspci` from Linux
  - MCH outputs (LEDs, web interface, telnet)
- Hot Plug and Hot Swap
  - increases up-time
  - reduces development time