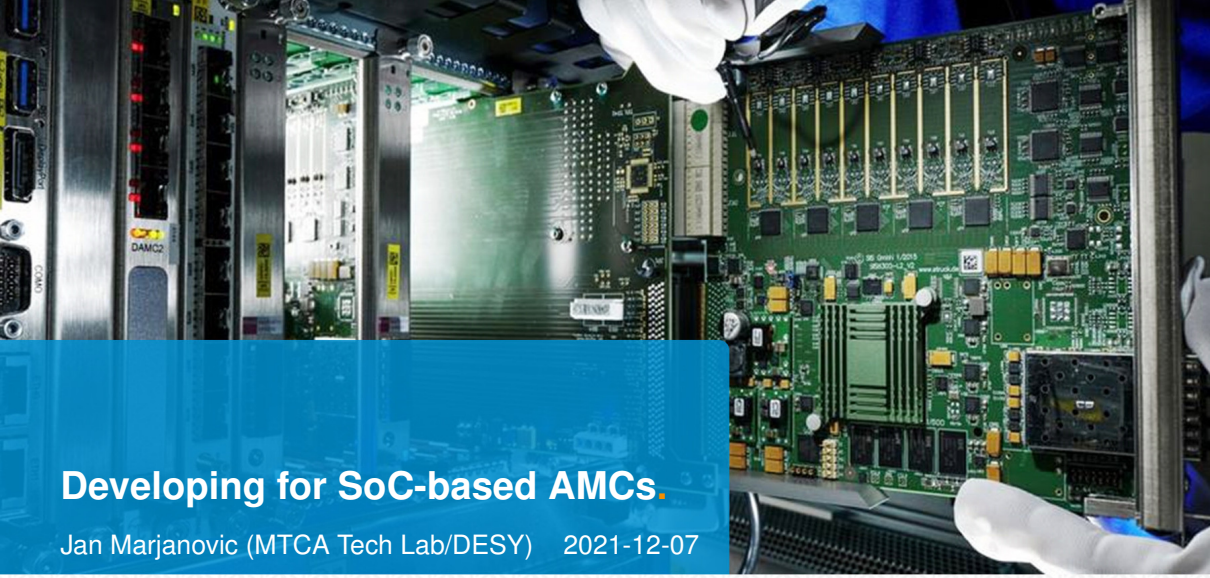


Developing for SoC-based AMCs.

Jan Marjanovic (MTCA Tech Lab/DESY) 2021-12-07

10th MicroTCA Workshop, Hamburg

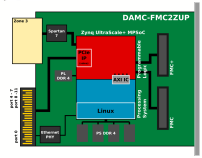
MicroTCA TECHNOLOGY LAB

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

DESY.

Presentation outline

- Hardware Introduction
- Device and Board Architecture
- Advantages of SoC-based AMCs
- Examples
- Tips and Tricks
- Conclusion



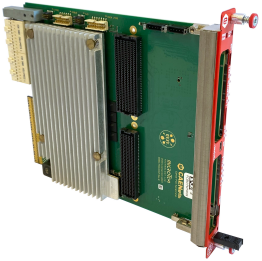
Hardware Introduction

System on a Chip, Hardware

*The Zynq MPSoC [...] comprises a number of different processing elements, each optimised for particular purposes — for instance, a set of **applications processors**, **real-time processor**, and a graphics processor, as well as **Field Programmable Gate Array (FPGA)** programmable logic.*

from L.H. Crockett, D. Northcote, C. Ramsay: Exploring Zynq® MPSoC, 2019, https://www.zynq-mpsoc-book.com/

DAMC-FMC1Z7IO
Cost-optimized FMC carrier
(Xilinx Zynq®-7000)



DAMC-FMC2ZUP
High-performance FMC/FMC+ carrier
(Xilinx Zynq® UltraScale+™ MPSoC)

DAMC-DS812ZUP
Low-latency high-speed 8-ch digitizer
(Xilinx Zynq® UltraScale+™ MPSoC)

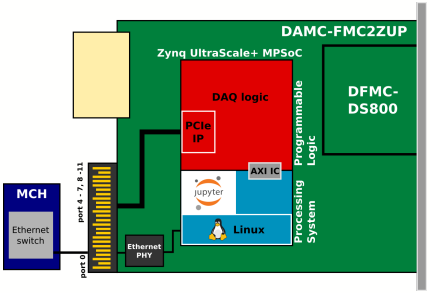
Hardware

The ”standard” part:

- a large FPGA, lots of DSPs, fast transceivers
- FMC and FMC+ slot, Zone 3 Class D1.1
- backplane connections: PCIe, LLL, MLVDS, TCLK
- flexible clocking scheme with White Rabbit support

The ”computer” part:

- processors
- Ethernet (next slide)
- USB C and DisplayPort
- SATA on AMC port 2 and port 3



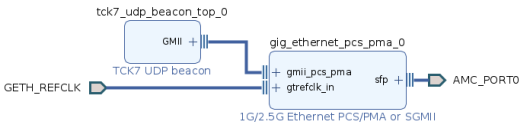
Example with DFMC-DSx00 on DAMC-FMC2ZUP; the board can be interfaced either through PCI Express or through Ethernet (Jupyter notebooks)

Evolution of Ethernet handling

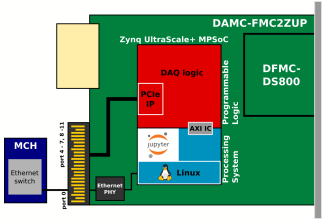
Backplane Ethernet (port 0) is . . .

- . . . on previous-gen boards[1] connected to **FPGA MGT**
- Ethernet implementation in FPGA:
  - with hard IP: requires high effort, not very flexible
  - with soft core: low performance, limited capabilities, complicated integration

https://github.com/MicroTCA-Tech-Lab/damc-tck7-fpga-bsp



- . . . on SoC-based boards[2] connected to **Processing System (ARM® CPU)**
- Link is operational even when FPGA is not programmed
- Leverages Linux network stack
- SSH for development
- HTTP(S), EPICS, ... for deployment



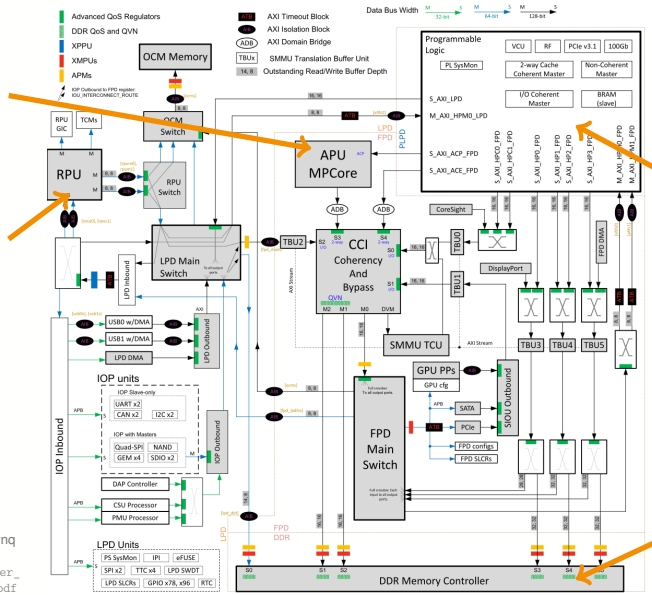
[1] DAMC-FMC20, DAMC-FMC25, DAMC-TCK7
[2] DAMC-FMC1Z7IO, DAMC-FMC2ZUP, DAMC-DS812ZUP

Architecture

Device architecture - Xilinx Zynq UltraScale+ MPSoC

APU (ARM® Cortex®-A53)

RPU (ARM® Cortex®-R5)



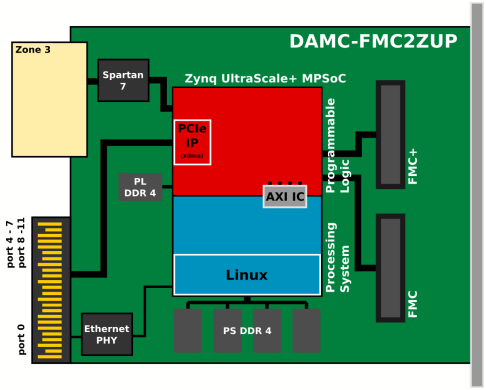
Programmable Logic (= ”FPGA”)

DDR4 memory controller

System Block Diagram from UG1085: Zynq UltraScale+ Device TRM, https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf

Board architecture - DAMC-FMC2ZUP

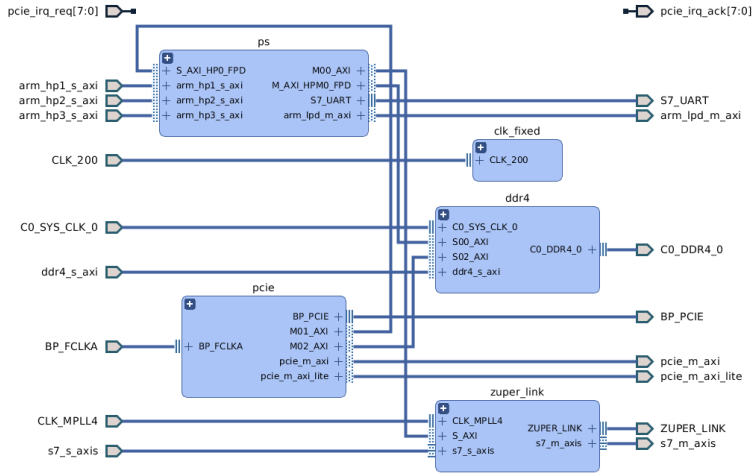
- PCIe connected to Programmable Logic (PL)
- Ethernet connected to Processing System (PS)
- Independent memories for PL and PS



Omitted for clarity: LLL (port 8 - 11, 12 - 15), MLVDS, SATA, USB-C, White Rabbit, clocking, IPMI/management

FPGA project organization

Vivado project in the Board Support Package, each subsystem is handled in a separate hierarchy:

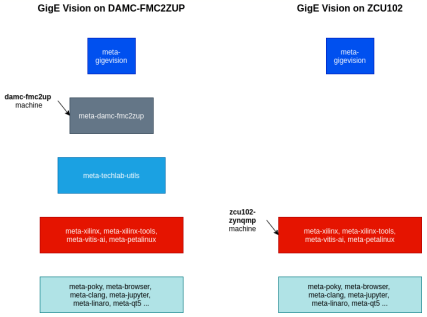


Components of an Embedded Linux:

- **First Stage BootLoader**
  initializes Zynq internals (PLL, PS DDR4, ...)
- **bootloader (u-boot)** and **bootloader commands/script**
  copies kernel image and Device Tree Blob (DTB) into memory, prepares environment (e.g. MAC address for Ethernet), optionally programs PL (FPGA)
- **device tree blob**
  describes HW to Linux (Zynq internals, on-board periphery)
- **kernel**
- **root filesystem**
  init, coreutils, ssh server, glibc, Python, application programs, ...

Built using Yocto from:

- ”standard” layers
- Xilinx layers
- AMC BSP layers
- application layer



Software, demonstration

With Bitbake (Yocto) images derived from `petalinux-image-full` (e.g. `zup-image-demo-full`) the result is a full-blown Linux distribution:

```
jan@ZUP-0555 ~ lsb_release -a
LSB Version:    n/a
Distributor ID: petalinux
Description:    PetaLinux 2020.1
Release:        2020.1
Codename:       zeus
jan@ZUP-0555 ~ uname -a
Linux ZUP-0555 5.4.0-xilinx-v2020.1 #1 SMP Thu Mar 4 22:37:31 UTC 2021 aa
ch64 GNU/Linux
jan@ZUP-0555 ~ free -h
              total        used        free      shared  buff/cache   ava
Mem:          3.8Gi       318Mi       3.3Gi       0.0Ki       267Mi
Swap:            0B          0B          0B
jan@ZUP-0555 ~ python3 --version
Python 3.7.6
jan@ZUP-0555 ~ gcc --version
gcc (GCC) 9.2.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is N
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOS

jan@ZUP-0555 ~ cat /sys/class/fpga_manager/fpga0/state
operating
jan@ZUP-0555 ~ file /mnt/sd-mmcblk0p1/download-damc-fmc2zup.bit
/mnt/sd-mmcblk0p1/download-damc-fmc2zup.bit: Xilinx BIT data - from damc_
D=0XFFFFFFFF;Version=2020.1 - for xczu11eg-ffvc1760-2L-e - built 2021/03/
ta length 0x167d0ec
jan@ZUP-0555 ~
```

Advantages of SoC-based AMCs

Advantages of SoC-based AMCs

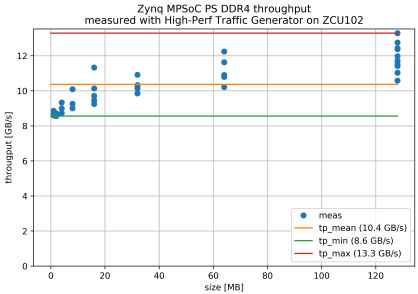
- Application partitioning
- Stand-alone products
- Configuration, monitoring

Application partitioning

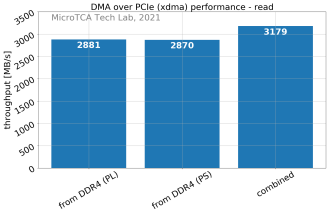
SoC-based design vs PCIe-based designs:

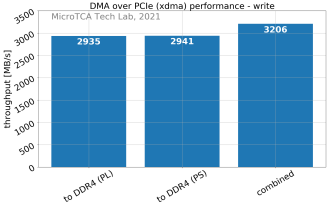
- ▲ data bandwidth between FPGA and CPU [1]
- ▲ interrupt latency, real-time performance
- ▼ computing power

Data bandwidth



PCIe throughput on DAMC-FMC2ZUP in gen3 x4 config



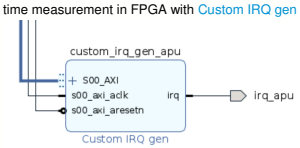


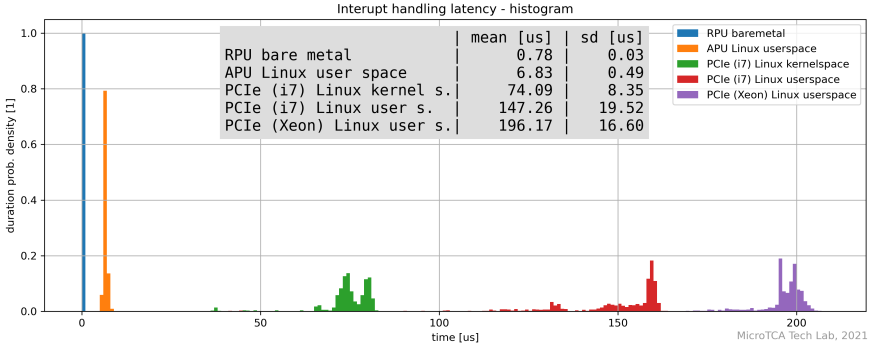
[1] assuming PCIe gen3 x4 or x8

Application partitioning

Interrupt latency

- Real-time processing unit (with Arm Cortex™-R5F) unsurprisingly performs really well
- Application Processing Unit (with Arm Cortex™-A53) in user space provides a good compromise between latency and ease of use
- PCIe interrupt latency higher and less deterministic
  → still OK for some applications (e.g. pulsed accelerators)





Application partitioning

Computing power

- On everyday tasks Cortex-A53 is approx. 10 times slower than a dedicated CPU AMC (with Intel i7)

```
In [1]:
import platform
import numpy as np

In [2]:
platform.machine()
Out[2]:
'aarch64'

In [3]:
xs = np.random.random(1024)

In [4]:
%timeit np.fft.fft(xs)
110 µs ± 89.5 ns per loop (mean ± s
td. dev. of 7 runs, 10000 loops eac
h)
```

```
In [1]:
import platform
import numpy as np

In [2]:
platform.machine()
Out[2]:
'x86_64'

In [3]:
xs = np.random.random(1024)

In [4]:
%timeit np.fft.fft(xs)
11.5 µs ± 38.3 ns per loop (mean ±
std. dev. of 7 runs, 100000 loops e
ach)
```

XML parsing with Cortex-A53

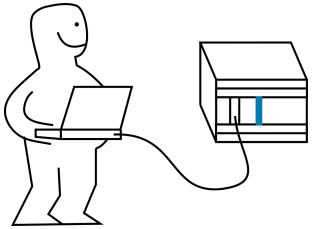
```
Running tests...
Test project
    Start 1:
1/2 Test #1:                                  0.10 sec
    Start 2:
2/2 Test #2:                                  6.41 sec

100% tests passed, 0 tests failed out of 2

Total Test time (real) =   6.54 sec
```

XML parsing with Intel i7

```
Running tests...
Test project
    Start 1:
1/2 Test #1:                                  0.01 sec
    Start 2:
2/2 Test #2:                                  0.98 sec

100% tests passed, 0 tests failed out of 2

Total Test time (real) =   0.99 sec
```

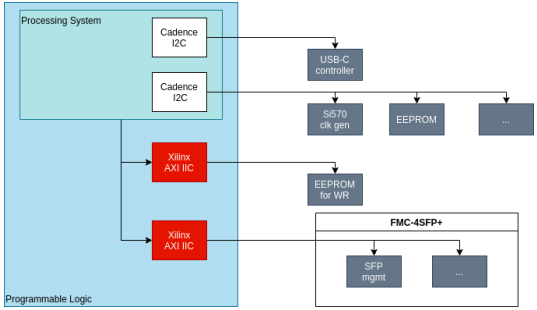
Stand-alone product

- User software can run on the board, e.g. Web server, EPICS IOC or Jupyter notebook
- No installation needed $\rightarrow$ better out-of-box experience
- Easier deployment of FPGA images (sync between software and firmware)
- Vendor has a better control of the environment
- Advanced diagnostics tools can be ”hidden” on the board



Configuration

Typically a board has several on-board components which need to be configured or initialized at the startup according to the application.



Old approach (non-SoC-based AMCs):

- state machines in the FPGA
- soft-core CPU (MicroBlaze, Nios II, LM32, ...)

New approach (SoC-based AMCs):

- a lot of I2C and SPI devices have Linux drivers [1]
- standard GPIO (`libgpio`) and I2C (`i2c-tools`) utilities
- Python or other high-level languages can be used

```
# i2cdetect -r -y 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: UU -- -- -- -- -- -- -- 58 -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- 6a -- -- --
70: -- -- -- -- -- 75 76 77
```

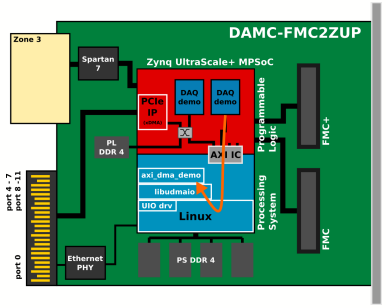
[1] press release from a couple of days ago:
https://www.analog.com/en/about-adi/news-room/press-releases/2021/11-30-2021-analog-devices-expands-linux-distribution.html

Examples

Example design

- BSP comes with Data Acquisition example
- Two transfer paths:
  - from FPGA into the PS memory (allocated with `u-dma-buf`)
  - from FPGA into the PL memory, then with PCIe to the CPU
- `libudmaio` and `pyudmaio` used to handle the data
- Xilinx XDMA driver for PCIe (`https://github.com/MicroTCA-Tech-Lab/xdma-metapackage`)

TechLab presentation on Thu at 13:30

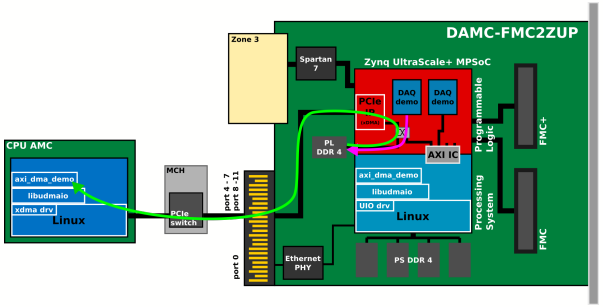



```
$ axi_dma_demo_cpp \
    --mode uio \
    --pkt_pause 45000 \
    --nr_pkts 1000
Counters: OK = 8192000,
total = 8192000
```

Example design

- BSP comes with Data Acquisition example
- Two transfer paths:
  - from FPGA into the PS memory (allocated with `u-dma-buf`)
  - from FPGA into the PL memory, then with PCIe to the CPU
- `libudmaio` and `pyudmaio` used to handle the data
- Xilinx XDMA driver for PCIe (`https://github.com/MicroTCA-Tech-Lab/xdma-metapackage`)

TechLab presentation on Thu at 13:30

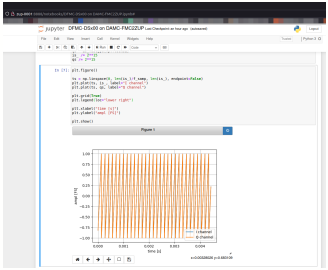



```
$ axi_dma_demo_cpp \
    --mode xdma \
    --dev_path \
    /dev/xdma/slot11 \
    --pkt_pause 35000 \
    --nr_pkts 1000
Counters: OK = 8192000,
total = 8192000
```

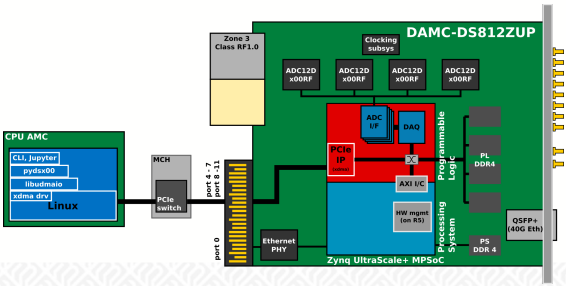
Standalone product

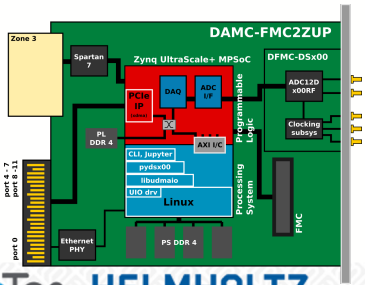
DSx00 family

- 8-channel digitizer, talks:
  - Zink: "RF Performance of Z3 class RF1.0 . . . ", Wed 16:00
  - Fenner: "Latest Hardware Developments . . . ", Thu 12:35
- Python library (`pydsx00`) provides a high-level interface to the hardware
- Runs both on x86_64 (over PCIe, with xdma driver) and on aarch64 (ARM)
- Easy start with Jupyter notebooks
- CLI for advanced users (long captures, . . .)




```
$ dsx-cli AMC \
    --xdma /dev/xdma/slot4 \
    plot_fft --nr_samp 1e6


$ dsx-cli AMC \
    --xdma /dev/xdma/slot4 \
    dump \
    --nr_samp 250e6 \
    meas_data0.npy
Written meas_data0.npy
```





MicroTCA TECHNOLOGY LAB

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

DESY.

Monitoring and partitioning, example #1

DAMC-DS812ZUP - communication between the app and the RPU

- PLL and ADCs are managed by the firmware (`HW mgmt`)
  running on Cortex-R5 (RPU)
- Application software (`pydsx00`) needs to communicate with the `HW mgmt`
- Solution: IPI[1] Messages
- Managing other processors from Cortex-A53 (APU) - still to be implemented on DS812ZUP:
  `remoteproc` - `https://www.kernel.org/doc/Documentation/remoteproc.txt`

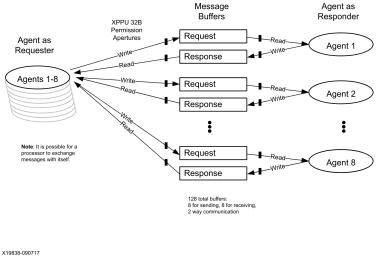


Figure 13-6: **IPI Message Passing Architecture**

> *Processor communications include both an IPI interrupt structure and memory buffers to exchange* ***short private 32B messages*** *between eight IPI agents — the PMU, RPU, APU, and PL processors. Access to the interrupt registers and message buffers is protected by the XPPU to give exclusive access to the AXI transactions of the agents.*
>
> from UG1085, `https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf`

[1] IPI = InterProcessor Interrupt

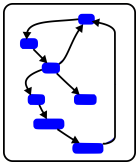
Configuration, example #1

FMC116 with DAMC-FMC1Z7IO - PLL configuration

- Used in a LISA phasemeter prototype → "Prof. Gerberding: Update on LISA Phasemeter ..." at 16:00 today
- Code migrated from Virtex-6 to Zynq-7000

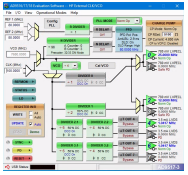
Before: `.coe` file to initialize BRAM, state machine to program the PLL over SPI

```
;*****************
;** AD9517 Init **
;*****************
0107C,
01101,
01200,
01300,
<...>
```



Round-trip-time for a change:
approx. 30 minutes

After:
Use of vendor tool to generate a `.stp` file



CLI to program the PLL, used in `init` script:

```
$ fmc116_cli.py --ad9517-program-stp iqdemod.stp
$ fmc116_cli.py --ad9517-info
AD9517 part_id = 0x53
AD9517 PLL readback = 0x4f
  Digital lock detect     = 1
  REF1 freq > threshold   = 1
```

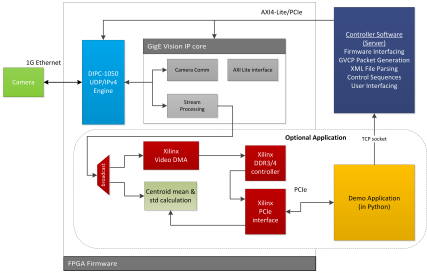
Round-trip-time for a change:
approx. 1 minute, easier to use

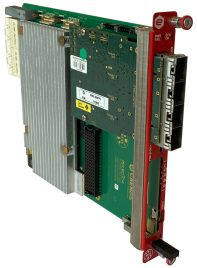
Monitoring, example #2

GigE Vision on ZUP

- Protocol implementation split between HW (high-performance part) and SW (initialization, XML parsing, link management)
- MAC addresses, S/N, production date stored in on-board EEPROM
- SFP modules (EEPROM at `0x50` for identification, DDI at `0x51` for monitoring)

```
# i2cdetect -y 5
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
<...>
50: 50 51 -- -- -- -- -- -- -- -- -- -- -- -- -- --
<...>
# i2cdump -y 5 0x50 b
     0  1  2  3           9  a  b  c  d  e  f    0123456789abcdef
00: 03 04 07 10          0c 00 06 67 00 00 00    ????...@.?.?g...
10: 03 01 00 00          20 20 20 20 20 20 20    ??..FS
20: 20 20 20 20          46 50 2d 31 30 47 2d        ....SFP-10G-
30: 54 20 20 20  <...>   20 20 20 03 52 00 28    T           ?R.(
40: 00 1a 0a 58          30 30 39 31 35 33 20    .??XG1904009153
```



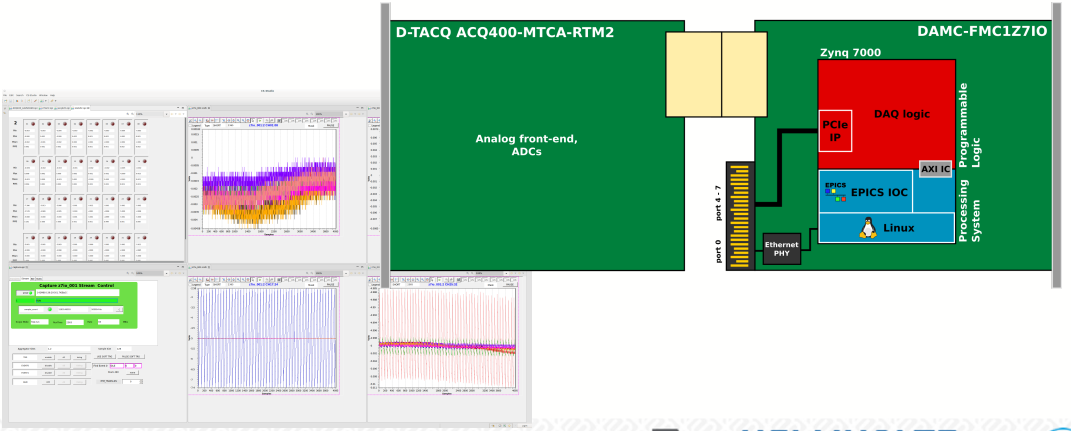


Examples from our partners

D-TACQ ACQ400 RTM with DAMC-FMC1Z7IO

D-TACQ Solutions Ltd (`https://www.d-tacq.com/`) is using Z7IO in combination with ACQ400-MTCA-RTM-2.

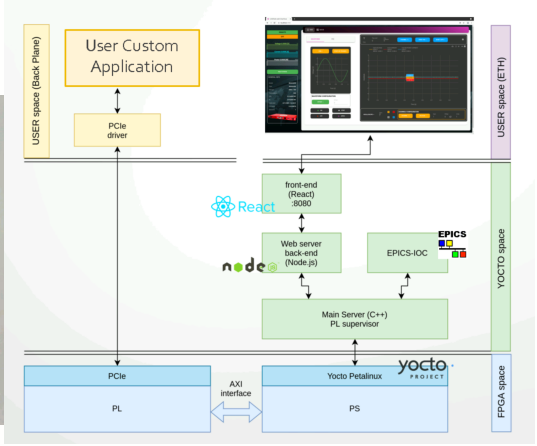
Example: EPICS IOC running on Z7IO



Examples from our partners

CAEN ELS picoammeter on DAMC-FMC2ZUP

Web server (Node.js) with React front-end and EPICS IOC running on the board.



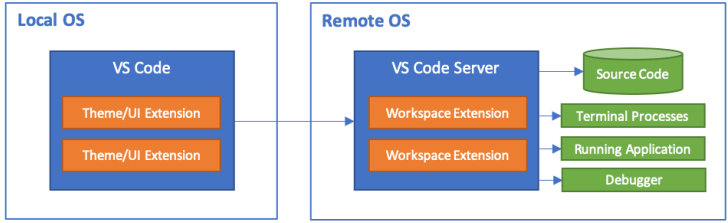
CAENels presentation on Wed at 13:40

Tips and Tricks

Cross-development

- SDK can be generated with Yocto (`bitbake -c populate_sdk`)
- `vim`, `nano`, `ed`[1] and `git` are installed on the board by default
- for more complex development, several cross-development options are available, for example: Visual Studio Code Remote Development

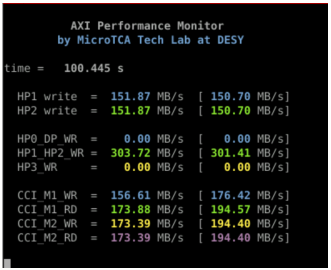
`https://code.visualstudio.com/docs/remote/remote-overview`

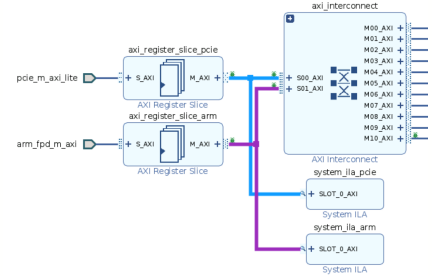


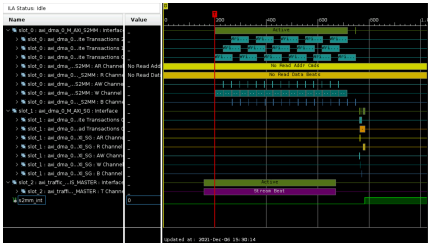
[1] `https://www.gnu.org/fun/jokes/ed-msg.en.html`

System ILA and AXI Performance Monitor

- systems are becoming increasingly complex
- good debugging tools are available, e.g. System ILA[1]
- System ILA can be used to observe AXI transactions
- three System ILAs are included in the example design
  - ARM interface, PCIe interface, DMA operation
- AXI Performance Monitor can be used to observe interconnects, e.g. when acquiring data:







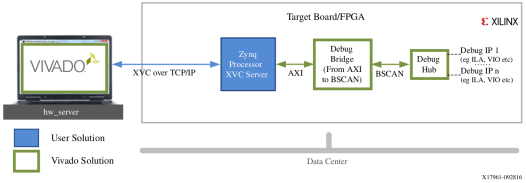
[1] ILA = Integrated Logic Analyzer

Xilinx Virtual Cable

Xilinx Virtual Cable (XVC) is a TCP/IP-based protocol that acts like a JTAG cable and provides a means to access and debug your FPGA or SoC design without using a physical cable.

from `https://www.xilinx.com/products/intellectual-property/xvc.html`

Figure 15: **AXI to BSCAN Debug Bridge**



- Software-side of the XVC is included in the BSP
- two packages in `meta-techlab-utils` layer: `xilinx-xvc-driver` and `xilinx-xvc-server`
- Useful links:
  - `https://support.xilinx.com/s/article/974879`
  - `https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_2/ug908-vivado-programming-debugging.pdf`

FPGA Manager

*The Zynq UltraScale+ MPSoC Programmable Logic (PL) can be programmed either using First Stage Boot-loader(FSBL), U-Boot or **through Linux**.*

from https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841847/Solution+ZynqMP+PL+Programming

Load the FPGA image (and the device tree overlay) from Linux userspace (init script from `fpgautil-init.bb`):

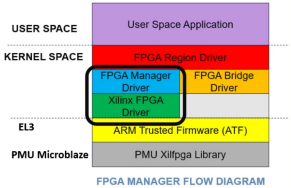
```
# cat /etc/init.d/fpgautil-init.sh
echo "fpgautil-init: downloading FPGA image"
fpgautil -b /lib/firmware/base/damc_fmc2zup_top.bit.bin -o /lib/firmware/base/base.dtbo
echo "fpgautil-init: FPGA image download done"
```

Un-load the FPGA image and the device tree overlay:

```
# fpgautil -R
```

Get the FPGA ID code (device identifier):

```
# fpgautil -r
Readback contents are stored in the file readback.bin
# grep IDCODE readback.bin
IDCODE -->        4a4e093
```



FPGA MANAGER FLOW DIAGRAM

local.conf should include:

```
IMAGE_FEATURES += " fpga-manager"
```

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES
DESY.

Device Tree

- Two separate files on the board (when using FPGA manager):
  - system: `/mnt/sd-mmcblk0p1/damc-fmc2zup-system.dtb`
  - PL: `/lib/firmware/base/base.dtbo`
- when using our BSP (`meta-techlab-utils`) add `DT_FROM_BD_ENABLE = "1"` to `local.conf` to get the IPs from the application into the device tree
- User-generated IPs can be handled by the UIO driver (`https://www.kernel.org/doc/html/v5.4/driver-api/uio-howto.html`)
- `lsuio` (`https://www.osadl.org/UIO-Archives.uio-archives.0.html`) can be used to list all IPs present in the device tree

```
root@ZUP-0001:~/lsuio-0.2.0# ./lsuio
uio9: name=hier_daq_pcie_axi_bram_ctrl, version=devicetree, events=0
  map[0]: addr=0xA0220000, size=8192
uio8: name=hier_daq_arm_axi_traffic_gen, version=devicetree, events=0
  map[0]: addr=0xA0110000, size=65536
uio7: name=hier_daq_arm_axi_dma, version=devicetree, events=0
  map[0]: addr=0xA0122000, size=4096
uio6: name=hier_daq_arm_axi_bram_ctrl, version=devicetree, events=0
  map[0]: addr=0xA0120000, size=8192
uio5: name=clk_monitor_clk_monitor_0, version=devicetree, events=0
  map[0]: addr=0xA0060000, size=4096
```

Package Management

- A single command required to create a repo: `bitbake package-index`
- Configuration file in `/etc/yum.repos.d/`

Serving the repository (on the build server):

```
jan@HSKTEGHLABEXTCPU0004 deploy/rpm  7c2c51b  python3 \
    -m http.server \
    --bind EXTCPU-100-0004.tech.lab 8123
Serving HTTP on 192.168.1.149 port 8123 (http://192.168.1.149:8123/) ...
192.168.1.87 - - [06/Oct/2021 20:18:59] "GET /repodata/repond.xml HTTP/1.1" 200 -
192.168.1.87 - - [06/Oct/2021 20:18:59] "GET /repodata/repond.xml HTTP/1.1" 200 -
192.168.1.87 - - [06/Oct/2021 20:18:59] "GET /repodata/5375abad73abc6a4b10a20358a03fd739416e5a62a1eed
c01f60dcd20b1376f7-primary.xml.gz HTTP/1.1" 200 -
192.168.1.87 - - [06/Oct/2021 20:18:59] "GET /repodata/0a48513bdbabb2eb35df4a9075af4d822bbcb920cda187
298c2ad24001aa402e-filelists.xml.gz HTTP/1.1" 200 -
192.168.1.87 - - [06/Oct/2021 20:19:18] "GET /repodata/repomd.xml HTTP/1.1" 200 -
192.168.1.87 - - [06/Oct/2021 20:19:26] "GET /damc_fmc2zup/device-tree-lic-xilinx-v2020.2+git0+f725aa
ecff-r0.damc_fmc2zup.rpm HTTP/1.1" 200 -
192.168.1.87 - - [06/Oct/2021 20:19:26] "GET /damc_fmc2zup/device-tree-xilinx-v2020.2+git0+f725aaecff
-r0.damc_fmc2zup.rpm HTTP/1.1" 200 -
192.168.1.87 - - [06/Oct/2021 20:19:26] "GET /damc_fmc2zup/fpga-manager-util-base-xilinx+git0+bc84458
333-r0.damc_fmc2zup.rpm HTTP/1.1" 200 -
192.168.1.87 - - [06/Oct/2021 20:19:26] "GET /damc_fmc2zup/fpga-manager-util-lic-xilinx+git0+bc844583
33-r0.damc_fmc2zup.rpm HTTP/1.1" 200 -
192.168.1.87 - - [06/Oct/2021 20:19:26] "GET /damc_fmc2zup/fpga-manager-util-xilinx+git0+bc84458333-r
0.damc_fmc2zup.rpm HTTP/1.1" 200 -
```

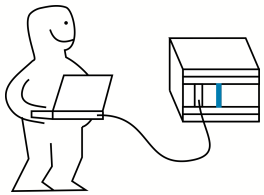
Updating the packages (on the board):

```
root@ZUP-0552:~# dnf --refresh update                                                    [0/172]
extcpu0004                                              607 kB/s | 3.0 kB     00:00
Dependencies resolved.
================================================================================================
 Package              Architecture    Version                          Repository       Size
================================================================================================
Reinstalling:
 device-tree-lic      damc_fmc2zup    xilinx+v2020.2+git0+f725aaecff-r0   oe-packages     13 k
 device-tree          damc_fmc2zup    xilinx+v2020.2+git0+f725aaecff-r0   oe-packages     15 k
 fpga-manager-util-base damc_fmc2zup  xilinx+git0+bc84458333-r0           oe-packages    2.5 M
 fpga-manager-util-lic damc_fmc2zup   xilinx+git0+bc84458333-r0           oe-packages    7.2 k
 fpga-manager-util    damc_fmc2zup    xilinx+git0+bc84458333-r0           oe-packages    5.8 k

Transaction Summary
================================================================================================

Total download size: 2.6 M
Installed size: 35 M
Is this ok [y/N]: y
Downloading Packages:
(1/5): device-tree-lic-xilinx+v2020.2+git0+f725aaecff-r0.damc_fmc2zup 2.0 MB/s |  13 kB     00:00
(2/5): device-tree-xilinx+v2020.2+git0+f725aaecff-r0.damc_fmc2zup.rpm 2.1 MB/s |  15 kB     00:00
(3/5): fpga-manager-util-lic-xilinx+git0+bc84458333-r0.damc_fmc2zup.r 1.8 MB/s | 7.2 kB     00:00
(4/5): fpga-manager-util-xilinx+git0+bc84458333-r0.damc_fmc2zup.rpm   1.2 MB/s | 5.8 kB     00:00
(5/5): fpga-manager-util-base-xilinx+git0+bc84458333-r0.damc_fmc2zup. 46 MB/s | 2.5 MB     00:00
------------------------------------------------------------------------------------------------
Total                                                 43 MB/s | 2.6 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                              1/1
  Reinstalling     : fpga-manager-util-lic-xilinx+git0+bc84458333-r0.damc_fmc2zup  1/10
  Reinstalling     : fpga-manager-util-base-xilinx+git0+bc84458333-r0.damc_fmc2zup 2/10
  Reinstalling     : device-tree-lic-xilinx+v2020.2+git0+f725aaecff-r0.damc_fmc2zup 3/10
  Running scriptlet: device-tree-xilinx+v2020.2+git0+f725aaecff-r0.damc_fmc2zup    4/10
%prein(device-tree-xilinx+v2020.2+git0+f725aaecff-r0.damc_fmc2zup): scriptlet start
%prein(device-tree-xilinx+v2020.2+git0+f725aaecff-r0.damc_fmc2zup): execv(/bin/sh) pid 1158
+ set -e
+ rm -rf /usr/lib/opkg/alternatives/device-tree
%prein(device-tree-xilinx+v2020.2+git0+f725aaecff-r0.damc_fmc2zup): waitpid(1158) rc 1158 status 0
```

Conclusion

Conclusion

- Several new and interesting AMC boards were developed; available to the community and partners for their projects
- All important features of previous-gen boards are still present (PCIe, LLL, MLVDS, TCLK, Zone3)
- Tight integration between a CPU and FPGA $\rightarrow$ solutions leveraging all components
- Good eco-system (driven by Xilinx); smooth integration between different components
- The board can act as a standalone product
- Improved quality-of-life for the developers and users



Thank you

https://techlab.desy.de

mtca-techlab@desy.de

Deutsches Elektronen-Synchrotron DESY
A Research Centre of the Helmholtz Association
Notkestr. 85, 22607 Hamburg, Germany