Hardware Acceleration
LAB

DAQ-0 and FE EMC system status

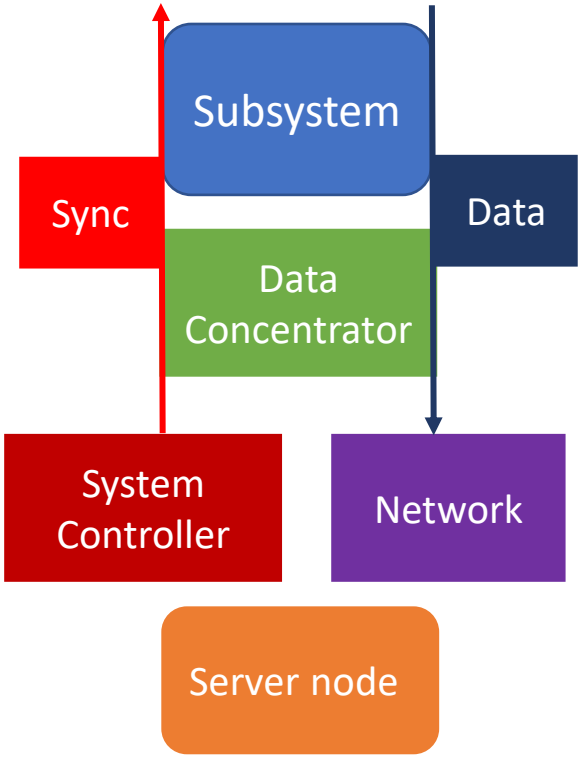
Grzegorz Korcyl

Panda Collaboration Meeting 23/2

12.06.2023

DAQ-0

- Actual detector subsystem setup
  - A single segment or more complex setup
- Synchronization
- Existing Data Concentrator hardware platform
- Framework for:
  - Subsystem integration with the DAQ
  - Development of subsystem preprocessing on DC
  - Development of control and monitoring procedures
  - DAQ performance and scalability evaluation
  - Available for beam tests of subsystems
- First candidate -> Forward Endcap EMC beam at COSY
  - ~32 SADC Endpoints
  - 1 Data Concentrator
  - 1 System Controller



DAQ-0 setup

Controller Board

- Generic platform, development board

- Common clock source

- Syncronization pulse (SB update) generation

- Can accept external input reference signals

- Synchronous transc.

- Linux + master AXI

Data Concentrator

- Common clock recovery

- Common clock forwarding

- Endp. data aggregation

- Ethernet output

- AXI forwarding and local

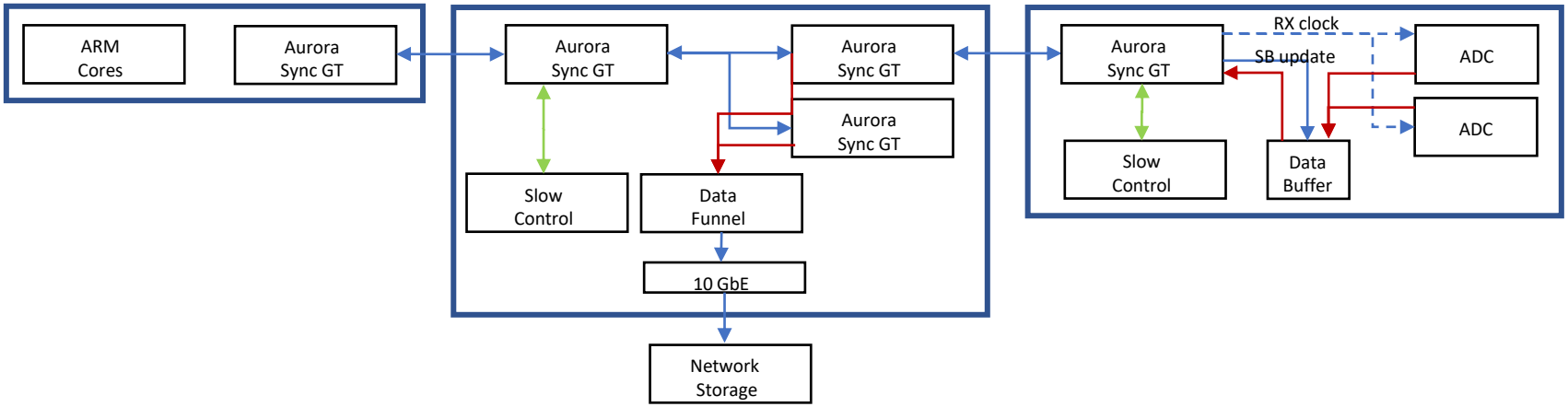
Endpoint

- Common clock recovery

- SB update pulse recovery

- Data buffering

- Slow control

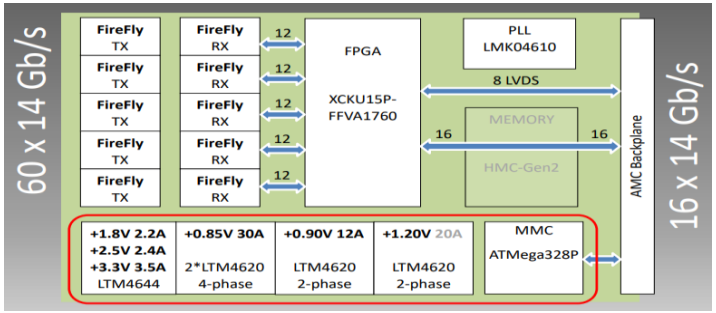


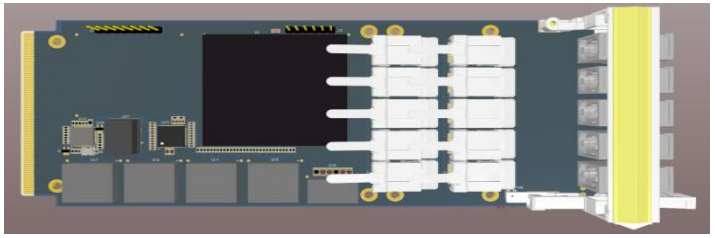
System Controller

- Xilinx development board ZCU102
  - ZYNQ ARM Processing System with Petalinux
    - Bootable from SD card
    - Gigabit Ethernet to server required
    - Root filesystem on an NFS mountable directory – ease of management
    - Bitfile upload from Petalinux – flexible reconfiguration
    - Control/Monitoring server
  - 3x GTH transceivers activated on 2x2 SFP+ cage
    - Configurable reference clock frequency
    - Master Aurora Sync interfaces
    - Possibility to extend with FMC addons, up to 16x links
  - Synchronization pulse generator
    - Configurable SYNC pulse frequency <-> readout frequency

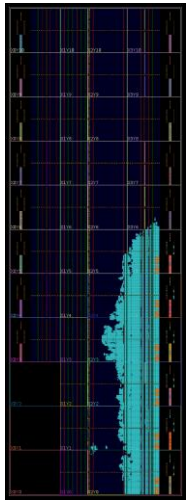
Data Concentrator

- 2 units produced and equipped
  - 1 u. operational, 1 u. power supply failure
  - New production batch submitted
- FPGA programmable
- Module Management Controller operational
- Initial firmware (P. Marciniewski & M. Caselle)
  - PLL HDL controller – clock source for MGTs
  - Evaluation of MGT channels
- Development firmware
  - 100Mhz local reference clock
  - Quads 224, 225, 226, 227, 228, 229 activated
    - 24 links in total
    - 1x Aurora Sync Slave, 1x Ethernet, 22x Aurora Sync Masters
  - GbE/UDP data output
  - 10% LUT, 10% BRAM, 32% GT, 6% BUFG, 9% MMCM
  - ~3 hours compilation time



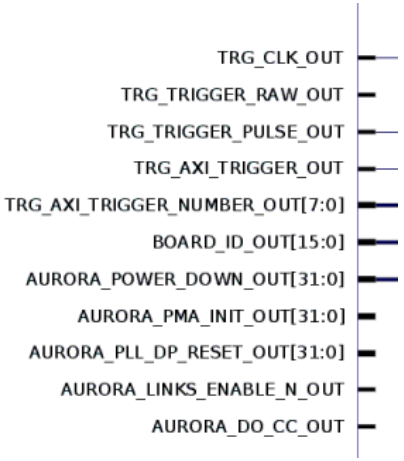


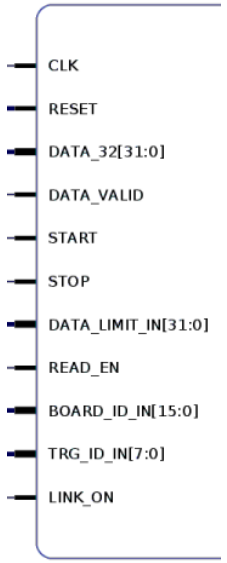
P. Marciniewski



Subsystem Endpoint

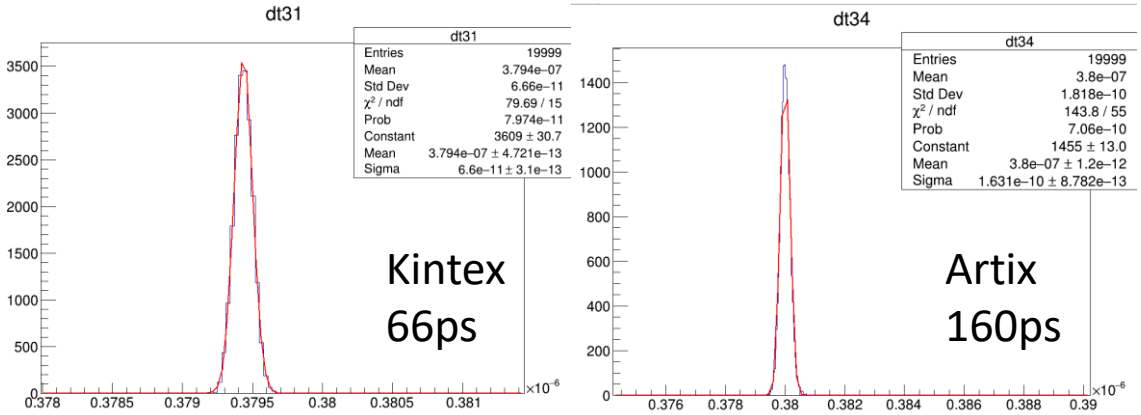
- Basic functionality
  - Aurora Sync Slave link
    - RX clock domain recovery
  - Synchronization Pulse recovery
    - RX clock domain pulse for digitization
    - Async. pulse for the readout procedure
  - Readout data transport
    - FIFO-like interface on local clock
  - Control/Monitoring
    - Memory-mapped AXI-Lite components
    - Addressable set of registers

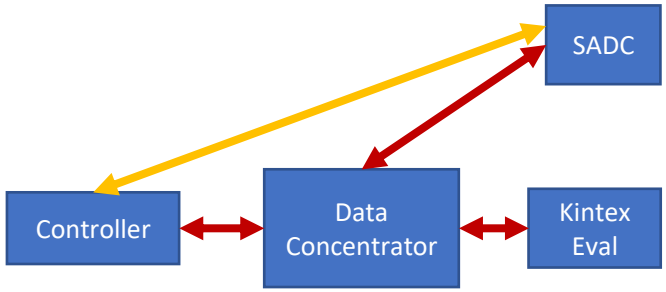




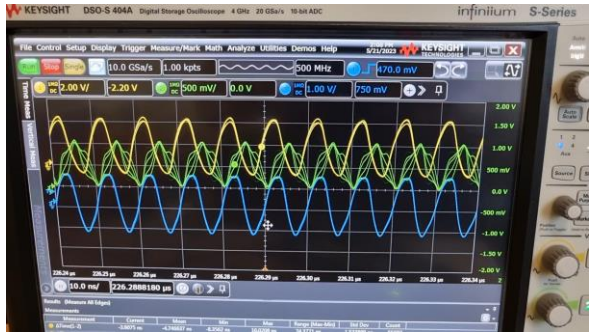
Subsystem Endpoint

- Evaluated on multiple platforms (via the Data Concentrator)
  - Virtex, Kintex, Artix, …
  - Running sync pulse jitter, offset stability over reboots





- SADC
  - Endpoint implemented – unable to properly evaluate
  - 4/5 Gbps Aurora Sync variants tested
  - Improve measurement procedure

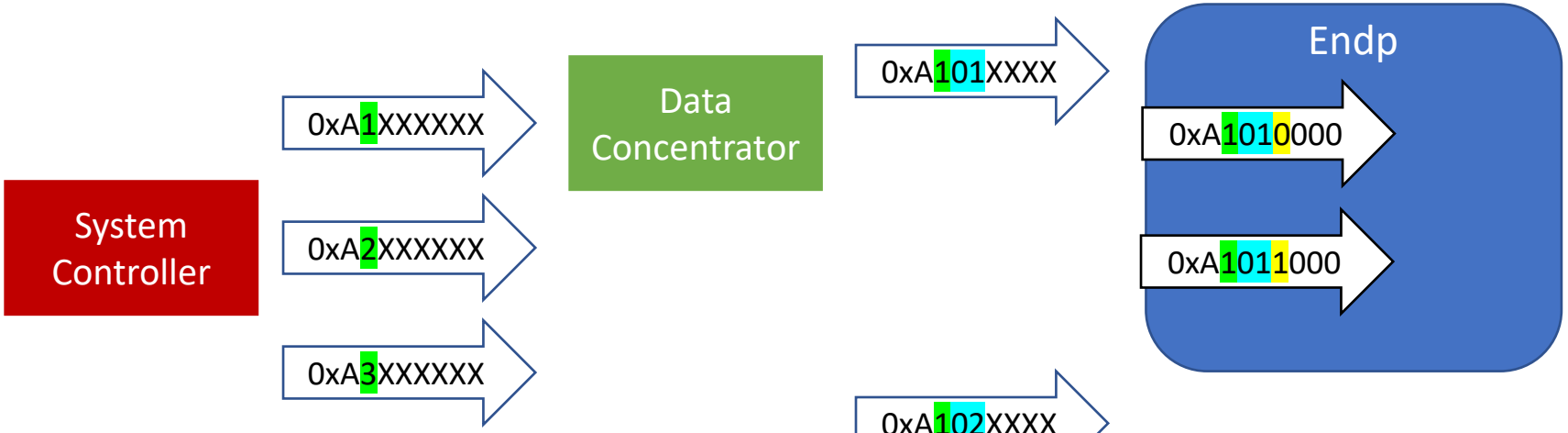


SADC Integration

- Major tasks:
  - Verification of the synchronization
  - Control/Monitoring interfacing
  - Inter-FPGA communication over GT
    - Aurora Sync Master on FPGA A and Slave on FPGA B
    - Readout data first gathered on FPGA A and then forwarded to the DC

Control and Monitoring

- Hierarchical addressing scheme
  - Each link has a sub-address
  - Each component is addressable by its location in the system
  - 32b addresses down to a single byte addressable



Local address space on the Controller:
0xAC00_0000 – 0xACFF_FFFF

Local address space on the DC:
0xA1FF_0000 – 0xA1FF_FFFF

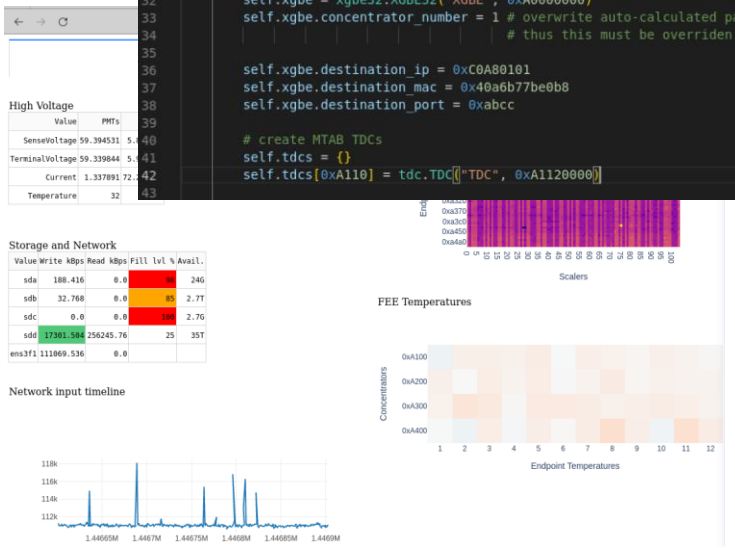
Local address space on the Endpoint
0xA101_0000 – 0xA101_FFFF

```
root.dev.write_mem(0xA1010000, 0xabcd)
root.dev.read_mem(0xA1010000, 1)
```

Control and monitoring software

- Linux running on the Controller Board
  - All AXI components, local and remote, are memory mapped
  - Accessible from the Linux memory address space
  - Specific board/component address scheme
- C++ backend for reading and writing the memory space, with Python server, running on embedded Linux
- Python set of frontend classes representing particular functional components, runs on client PC, connects to the server on embedded Linux
- Complete setups can be combined out of these classes
- Python WEB server
- Linux with C++ backend can be easily integrated with EPICS
- Performance and throughputs for large-scale setups have to be evaluated

```python
from djpet_client.utils import setup, device
from djpet_client.statistics import scalers, fsf_stats
from djpet_client.trigger import trigger_generator, trigger_receiver
from djpet_client.xgbe import xgbe32
from djpet_client.config import ftab_config, controller_config, mtab_config
from djpet_client.mtab import tdc
from djpet_client.ftab import temp_sens
import time


class PandaTest(setup.Setup):

    def __init__(self, **kwargs):
        super().__init__("tcp://192.168.0.52:4242", # use default connection map
                **kwargs)

        # create controller config block
        self.ctrl_config = controller_config.ControllerConfig("Ctrl Config", 0xAC000000)

        # create trigger generator
        self.trg_gen = trigger_generator.TriggerGenerator("Trigger Gen", 0xAC000000)

        # create trigger receivers
        self.trg_rec = {}
        # run loop over endpoints
        for i in range(1,2):
            base_addr = 0xA1000000 + (i << 20)
            dk = base_addr >> 16 # use shortenned address as dict key
            self.trg_rec[dk] = trigger_receiver.TriggerReceiver("Trigger E Rec", base_addr)

        # create XGBE subsystem, use xgbe32 for gbe, as the register map is the same
        self.xgbe = xgbe32.XGBE32("XGBE", 0xA0000000)
        self.xgbe.concentrator_number = 1 # overwrite auto-calculated parameter. There are no
                                          # thus this must be overriden.

        self.xgbe.destination_ip = 0xC0A80101
        self.xgbe.destination_mac = 0x40a6b77be0b8
        self.xgbe.destination_port = 0xabcc

        # create MTAB TDCs
        self.tdcs = {}
        self.tdcs[0xA110] = tdc.TDC("TDC", 0xA1120000)
```



FE EMC setup status

Controller Board

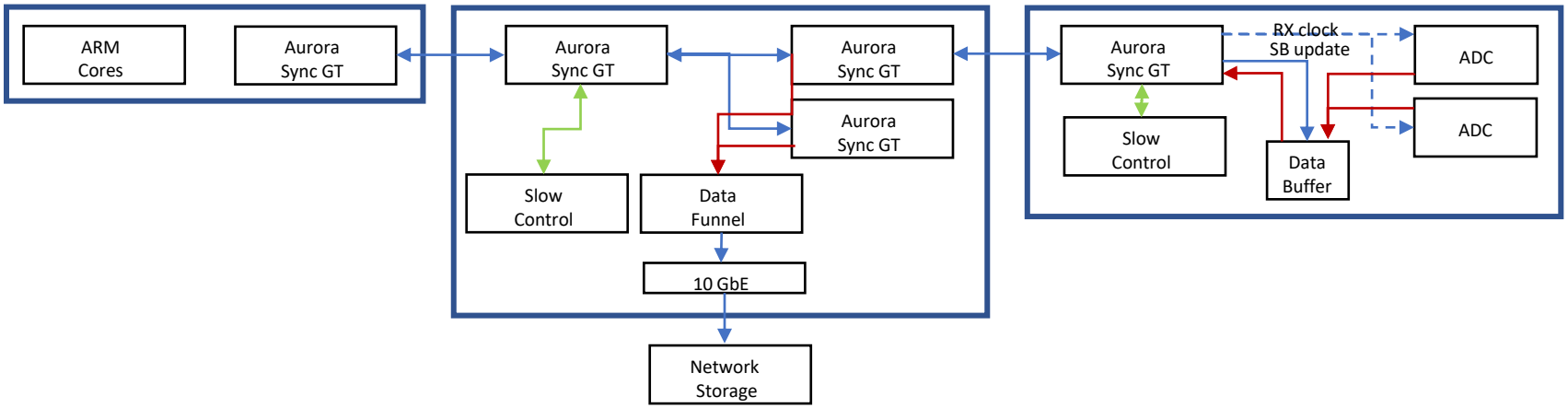
- Developed on ZCU102 platform
- Tested on multiple setups
- Firmware with 3x Sync. links
- Embedded linux
- Control and monitoring software stack on C++/Python with CLI and WEB interfaces

Data Concentrator

- Hardware platform operational
- PLL clock generator operational
- Aurora Sync Slave on GTH implemented
- Aurora Sync Master on GTH implemented
- Ethernet migrated, to be evaluated
- Data aggregation
- 24 links activated

Endpoint

- Communication evaluated on multiple hardware platforms
- Migration to SADC
- Evaluation of DAQ functionalities on SADC
- Inter-FPGA communication
- Adaptation of software package



FE EMC setup status

- Several components left to develop/evaluate
- Time is critical

- Plan B – short-term backup solution
  - Allow multiple SADCs to operate without precise time synchronization
  - Ethernet connection with fixed, individual addressing
    - Use FPGA unique ID to generate MAC, IP, UDP addresses
    - Gather the IDs over JTAG and build a look-up-table
  - Network switch with multiple interfaces
  - Adaptation of the software packages

Summary

- Development is advancing but
  - Not fast enough for the FE EMC beamtimes
  - Single issues, problems can significantly affect the plan
    - Measurement of the SADC synchronization
    - Digilent/Xilinx programmer… !
  - Greater reuse of components/knowledge required
    - Multiple parallel development tracks for: GbE, PLLs,…
- Core components of the system operational
  - SADC integration in progress

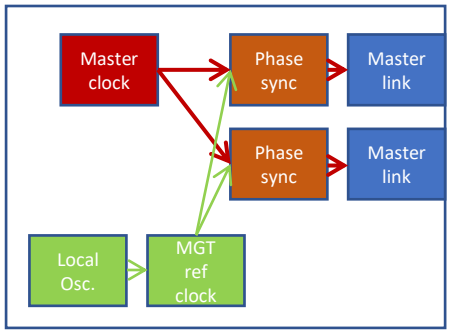
Communication infrastructure

- **Aurora**: low-level, point-to-point, network protocol for high-speed links
  - Ready to use, free-of-charge components in the IP library
  - 8b/10b encoded data transport in framing mode, introduces 10% overhead
  - Transceiver agnostic
  - Fixed timing, can be combined with synchronous transceivers
  - Adapted and implemented for Xilinx GTP, GTX, GTH, GTY transceiver families, all in synchronous mode
  - Tested with 5 Gbps, higher values possible, to be evaluated

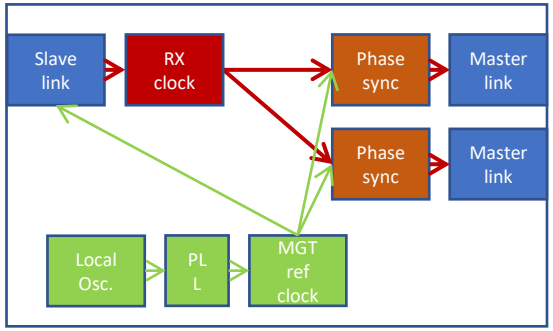
- **AXI**: on-chip communication bus protocol
  - Standard from ARM, adapted by Xilinx, natural for on-chip component communication
  - Many ready-to-use components from the IP library
  - Interface standard for embedded processors (ARM, Microblaze)

System clocking

Controller



Data Concentrator



SADC