First release of

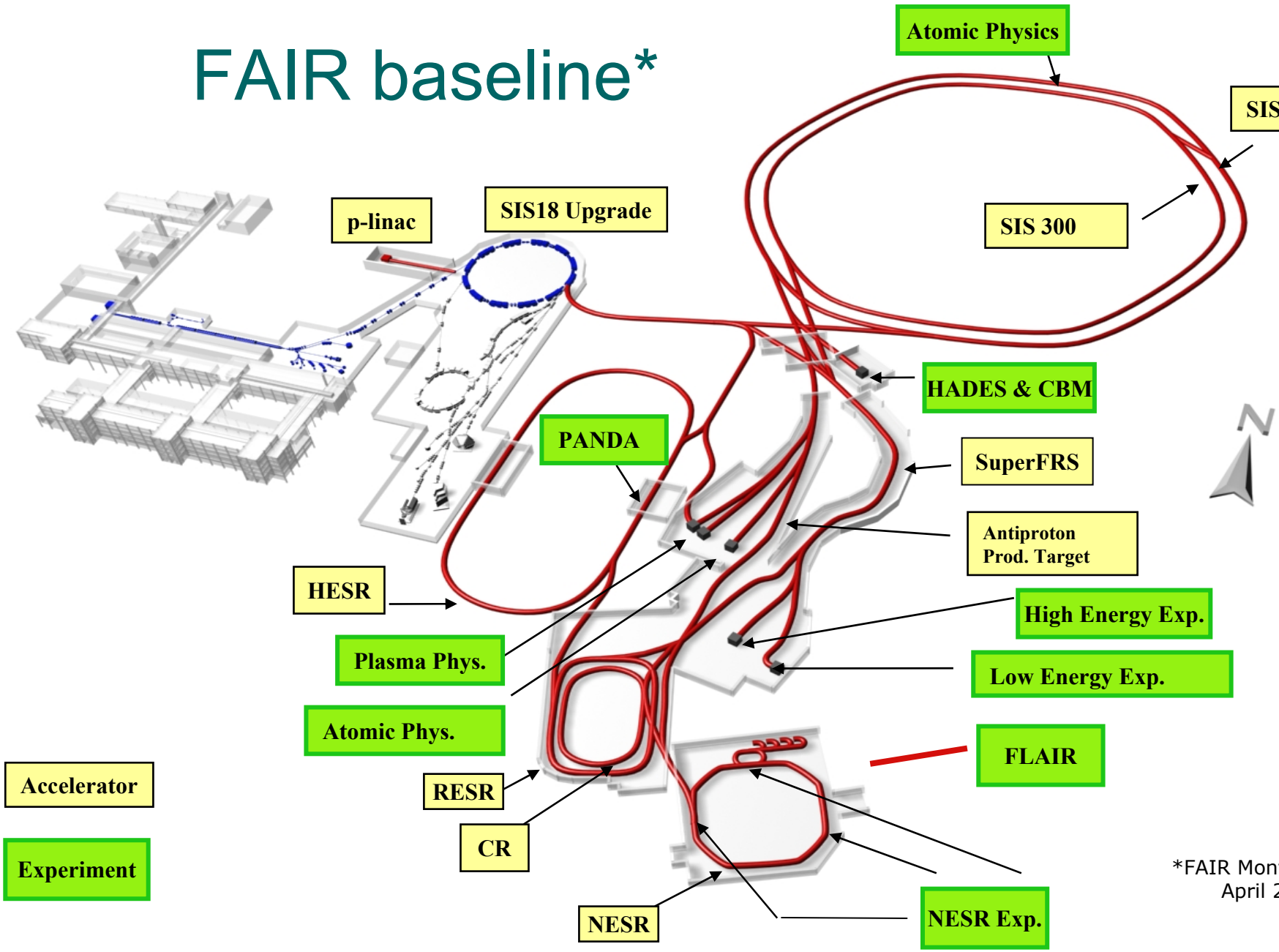
Data Acquisition Backbone Core

J. Adamczewski-Musch, H.G. Essel, N. Kurz, S. Linev

GSI Darmstadt, Germany
Experiment Electronics: Data processing group

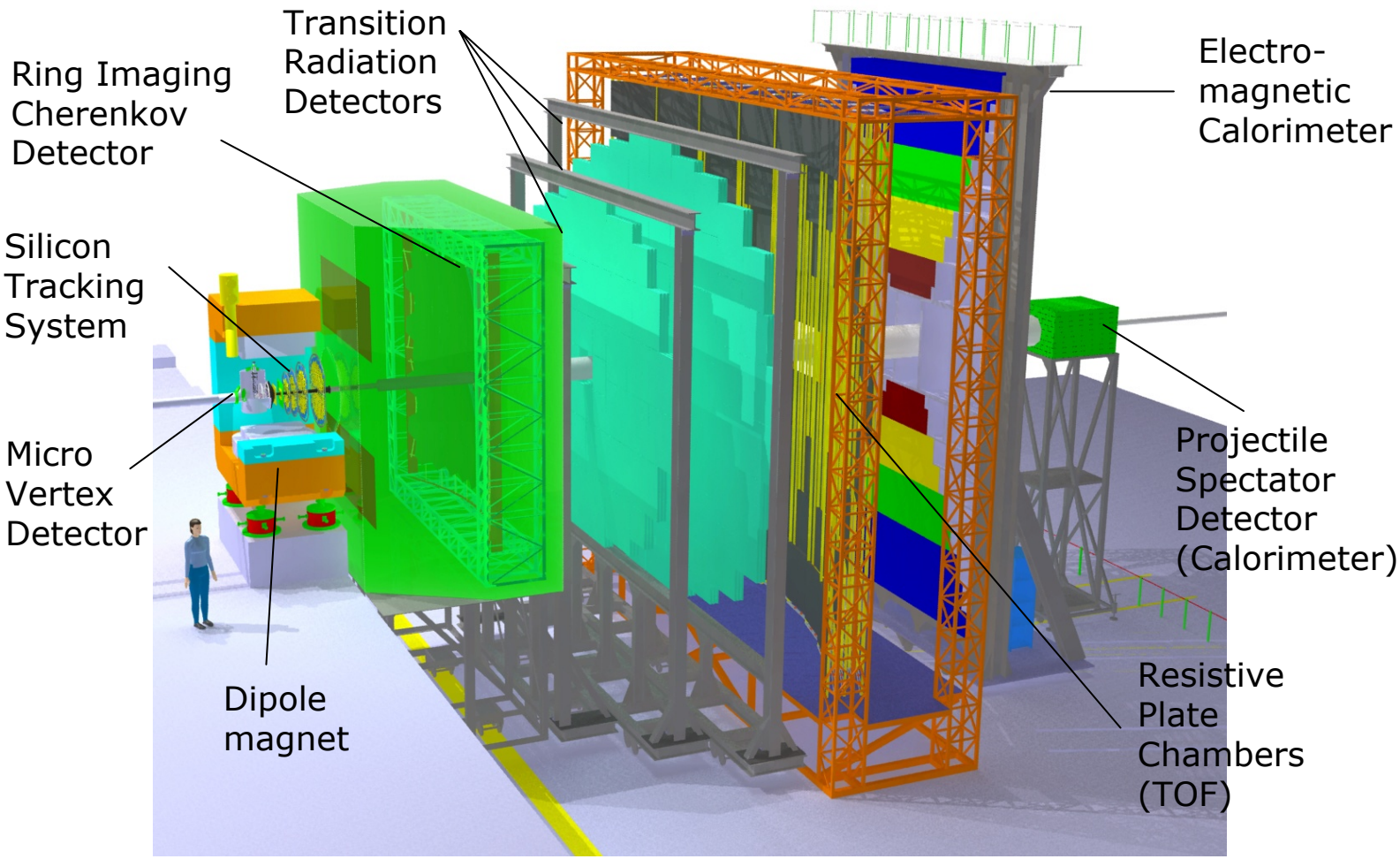
Work supported by EU RP6 project JRA1 FutureDAQ RII3-CT-2004-506078

FAIR baseline*

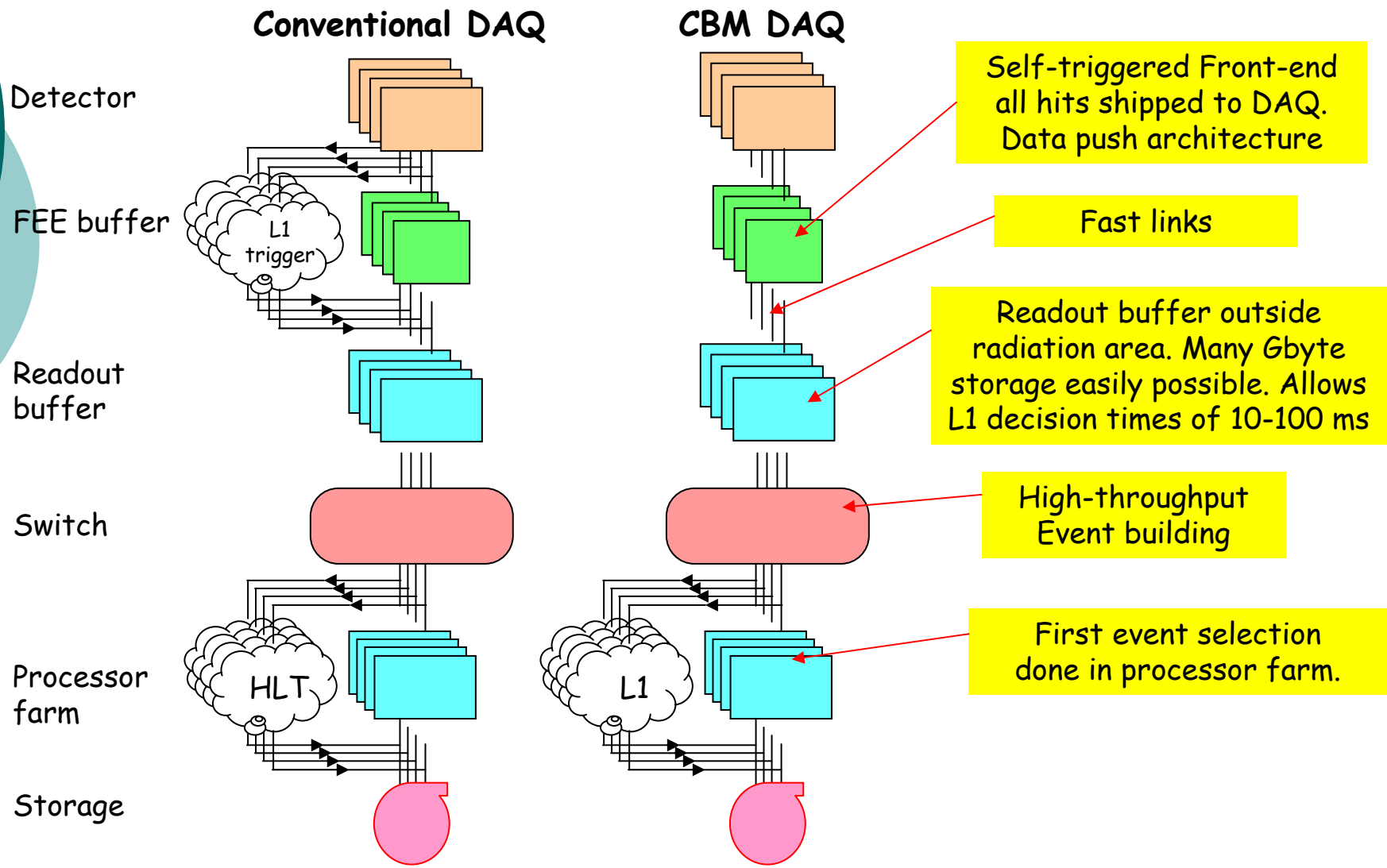


*FAIR Monthly, April 2009

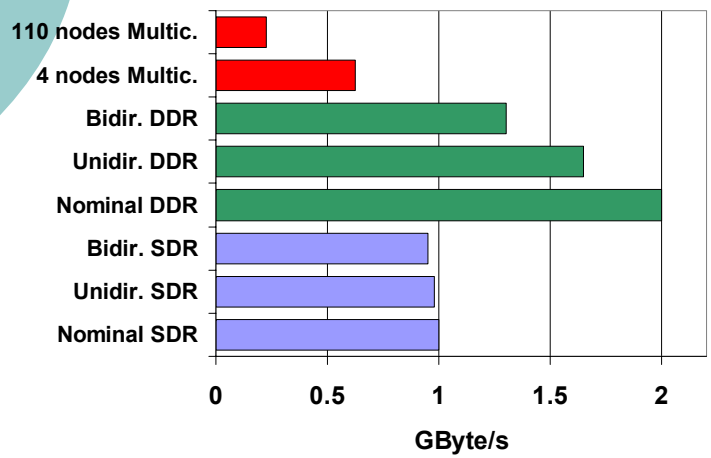
CBM experiment at FAIR



DAQ architectures

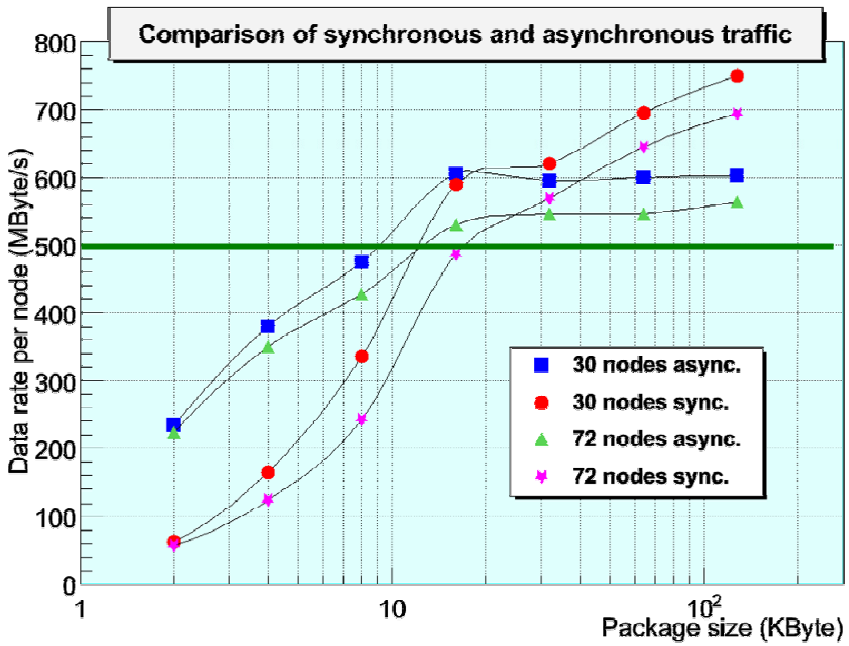


InfiniBand performance studies



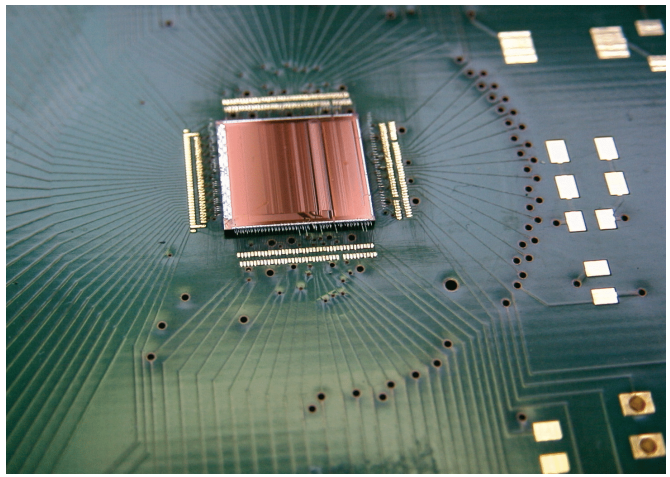
> 500 MBytes/s bidirectional

+ 25%

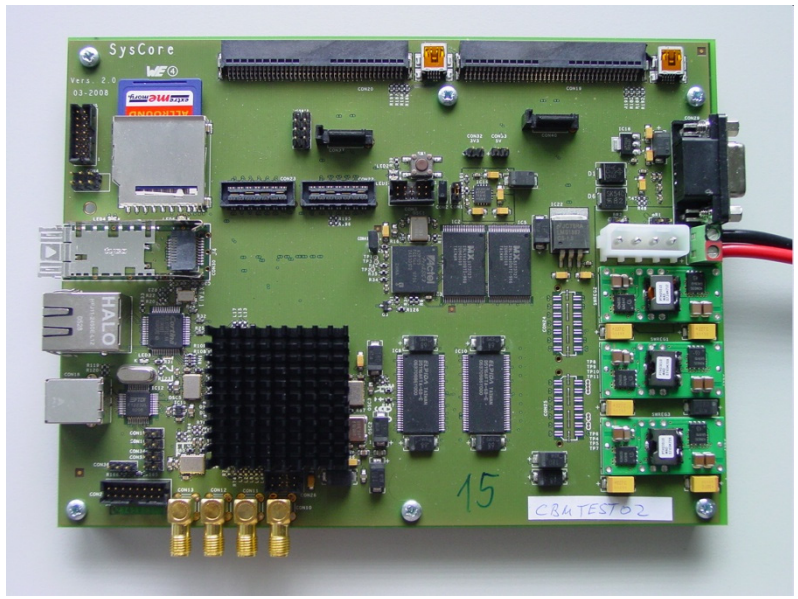


thanks to Klaus Merle and Markus Tacke at the Zentrum für Datenverarbeitung in Uni Mainz

Hardware tests



nXYTER, 128-channel self-triggered readout chip with few ns time resolution, *DEFNI collaboration*

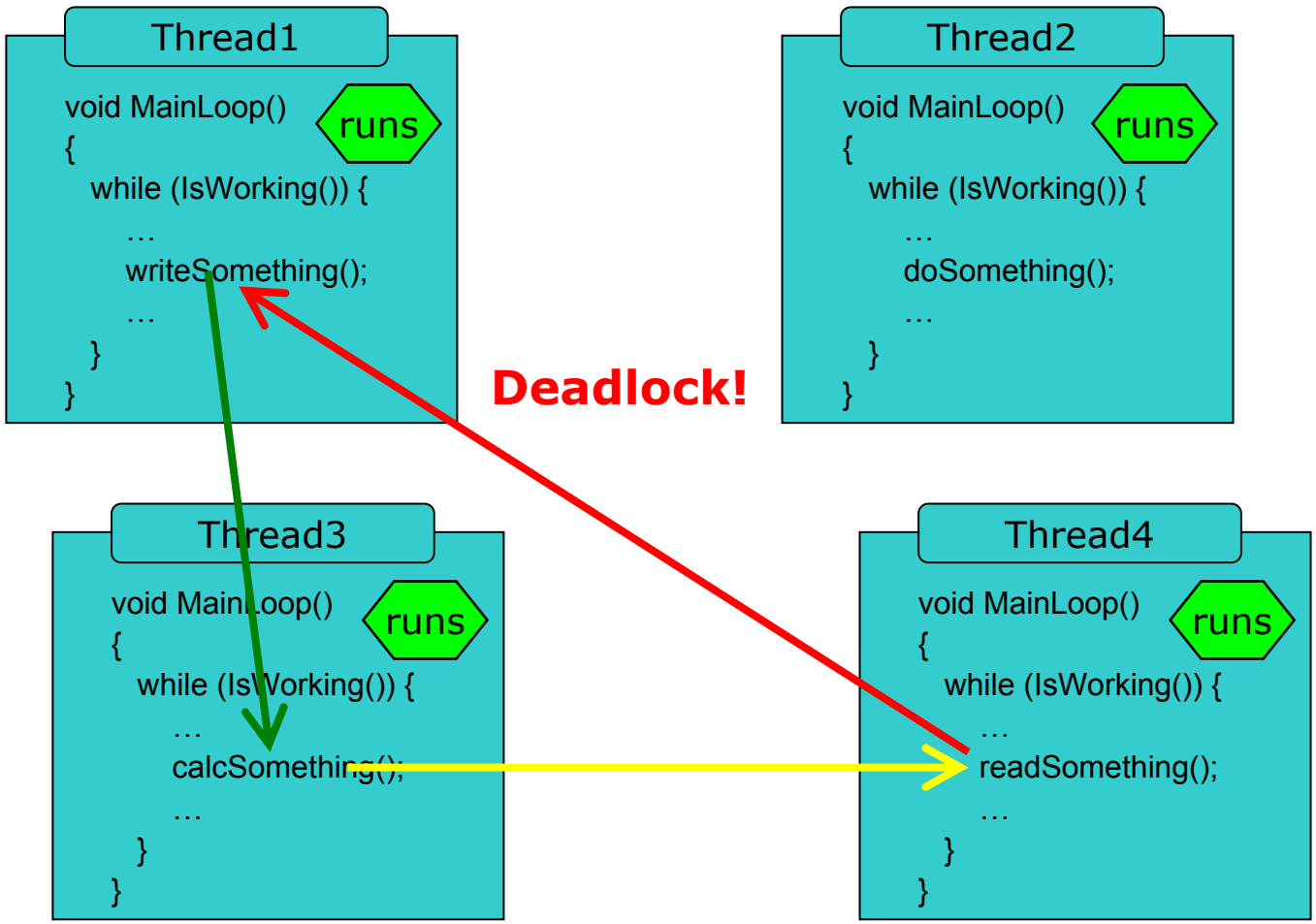


CBM Readout controller (ROC), *Kirchhoff Institut für Physik, Heidelberg (see Norbert Abel talk TDA4-4)*

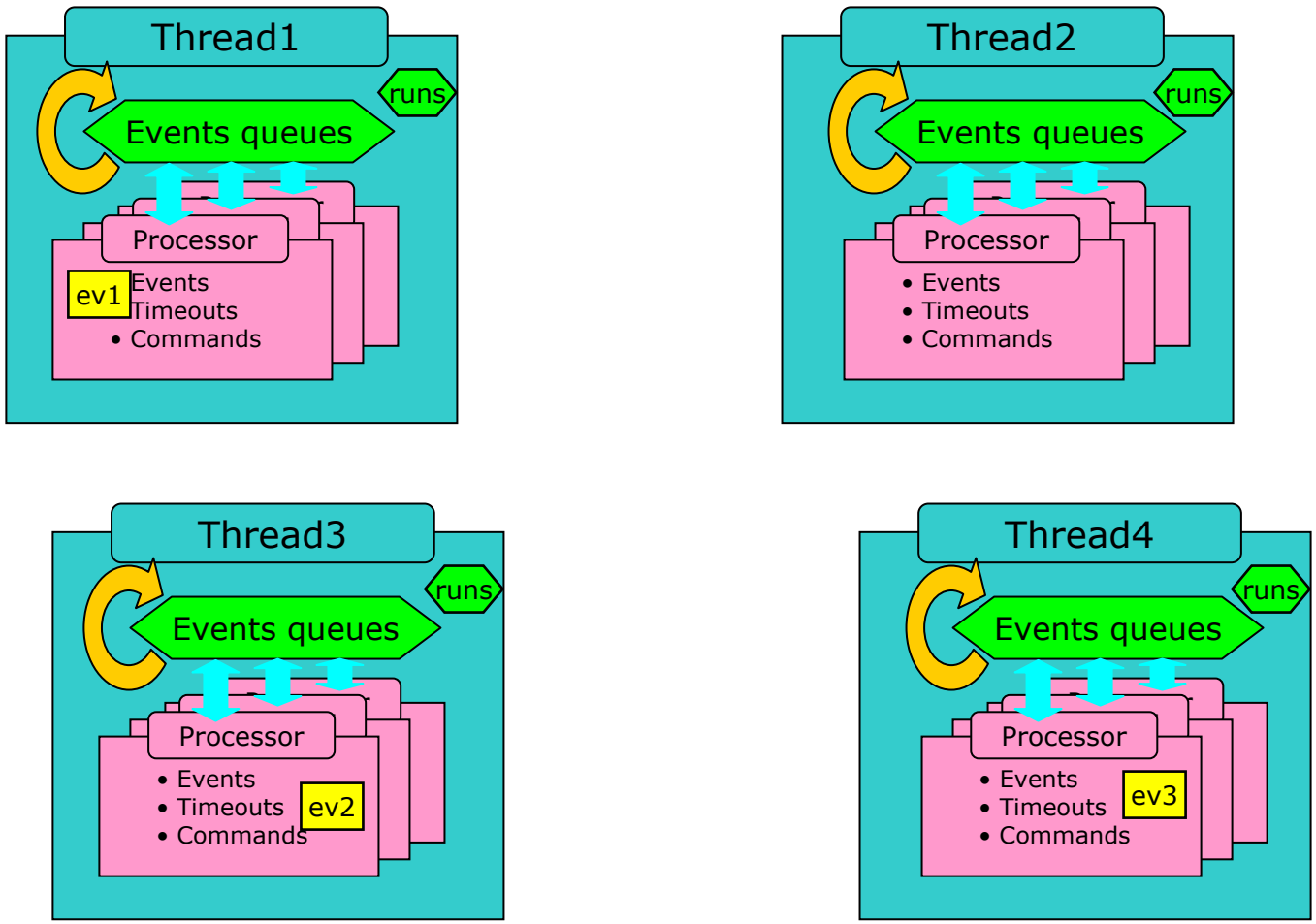
Future DAQ requirements

- Flexible – adopt different kinds of soft- and hardware
- Compact – use only necessary code
- Scalable – from small detector tests to 100-nodes clusters
- Performing – low framework overhead
- Multiprocessing & multithreading

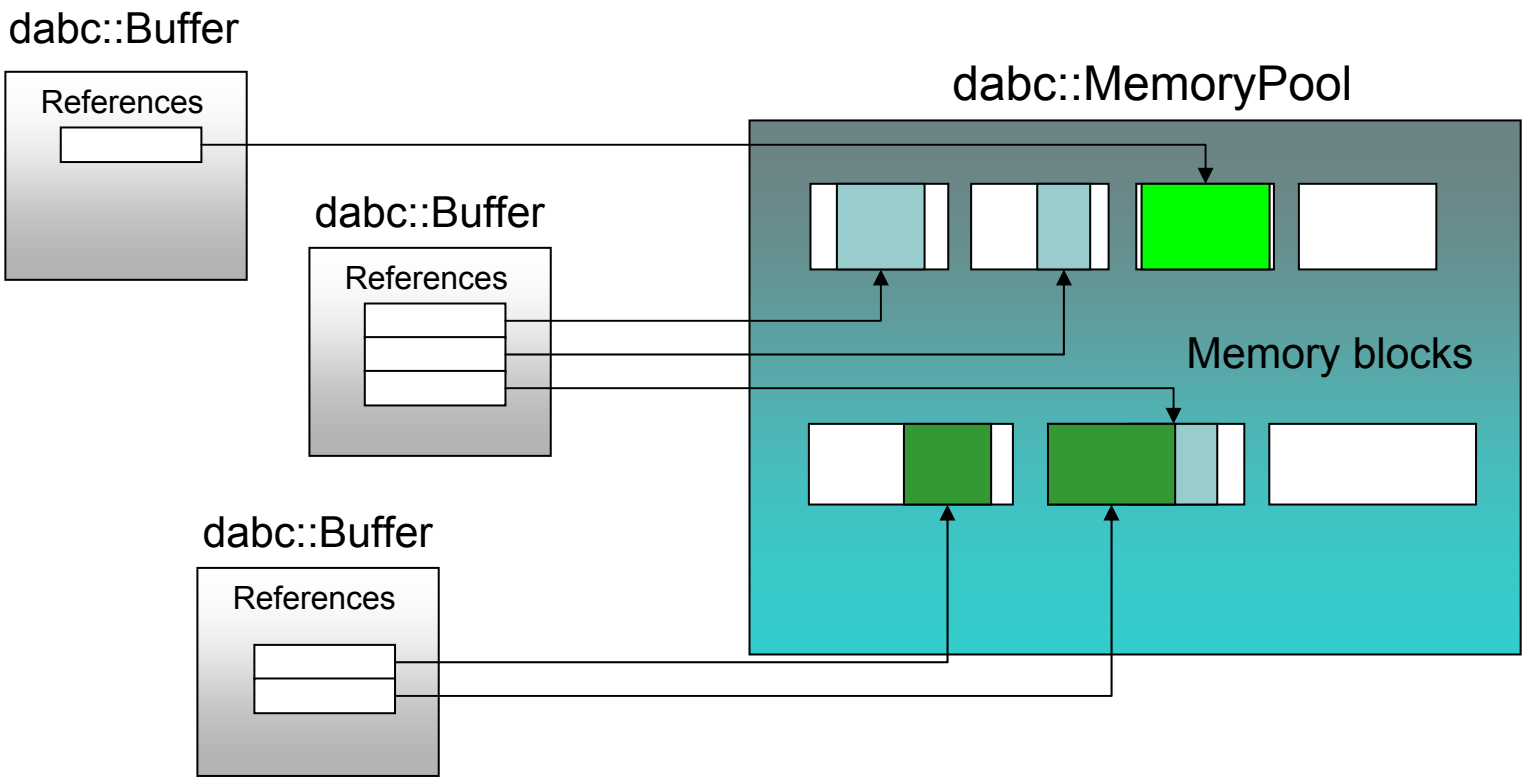
Multithreading



Multithreading in DABC

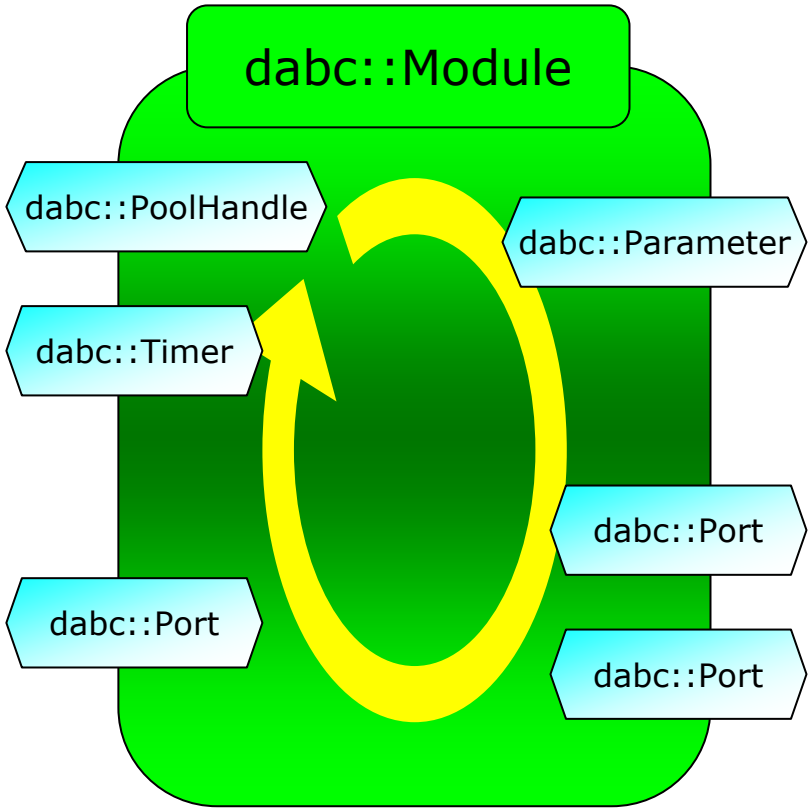


Memory management



Multiple references on resources (blocks)
Copy of reference without copy of data
Zero copy transfer: PCI/DMA → buffer → IB/DMA

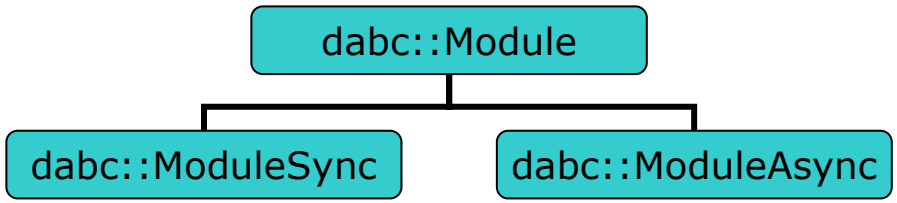
User code - modules

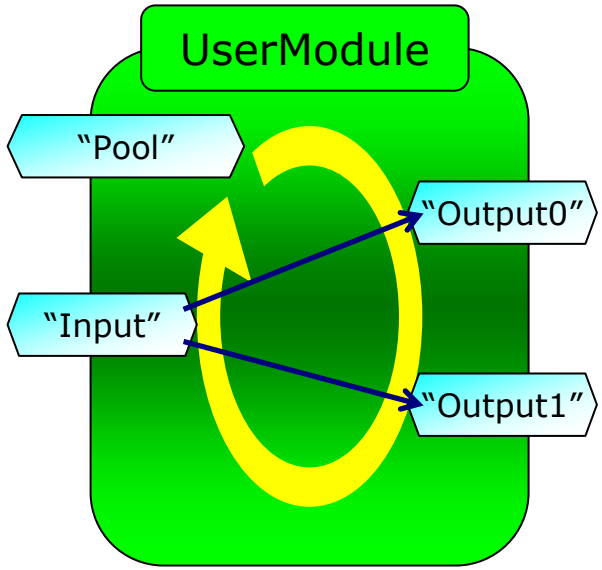


dabc::Module provides:

- I/O ports for communications
- Pools handles to request memory
- Timers for timeouts processing
- Configuration & monitoring parameters

Two kind of modules: sync & async





Constructors mainly the same

```
CreatePoolHandle("Pool", 2048, 1);
CreateInput("Input", Pool(), 5);
CreateOutput("Output0", Pool(), 5);
CreateOutput("Output1", Pool(), 5);
fCnt = 0;
```

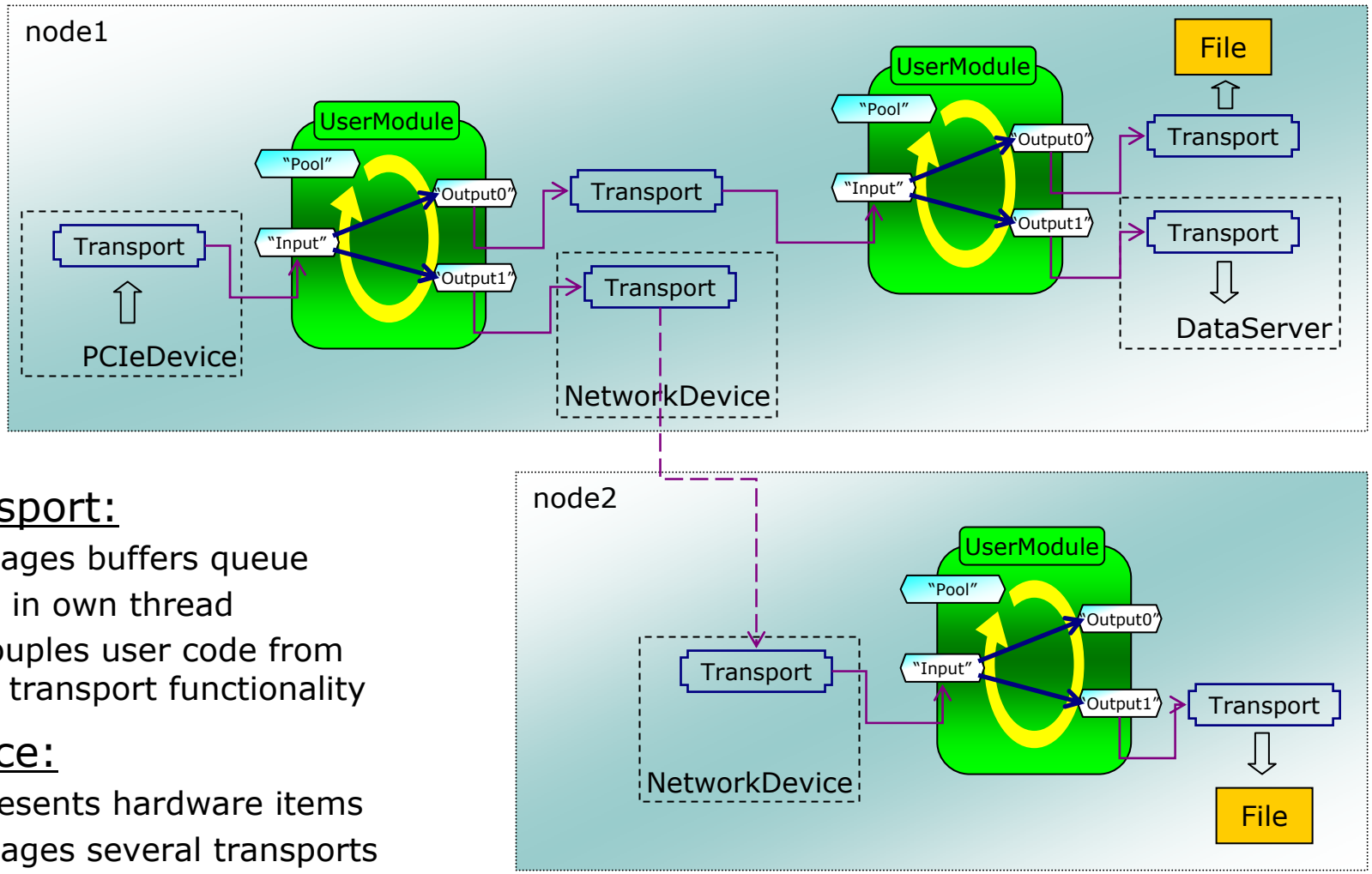
Explicit loop in ModuleSync

```
void UserModule::MainLoop()
{
  while (ModuleWorking()) {
    dabc::Buffer* buf = Recv(Input());
    if (fCnt++ % 2 == 0) Send(Output(0), buf);
                    else Send(Output(1), buf);
  }
}
```

Event processing in ModuleAsync

```
void UserModule::ProcessIOEvent(dabc::Port*)
{
  while (Input()->CanRecv() &&
         Output(fCnt % 2)->CanSend()) {
    dabc::Buffer* buf = Input()->Recv();
    Output(fCnt++ % 2)->Send(buf);
  }
}
```

Transport & device – dataflow



Transport:

- manages buffers queue
- runs in own thread
- decouples user code from actual transport functionality

Device:

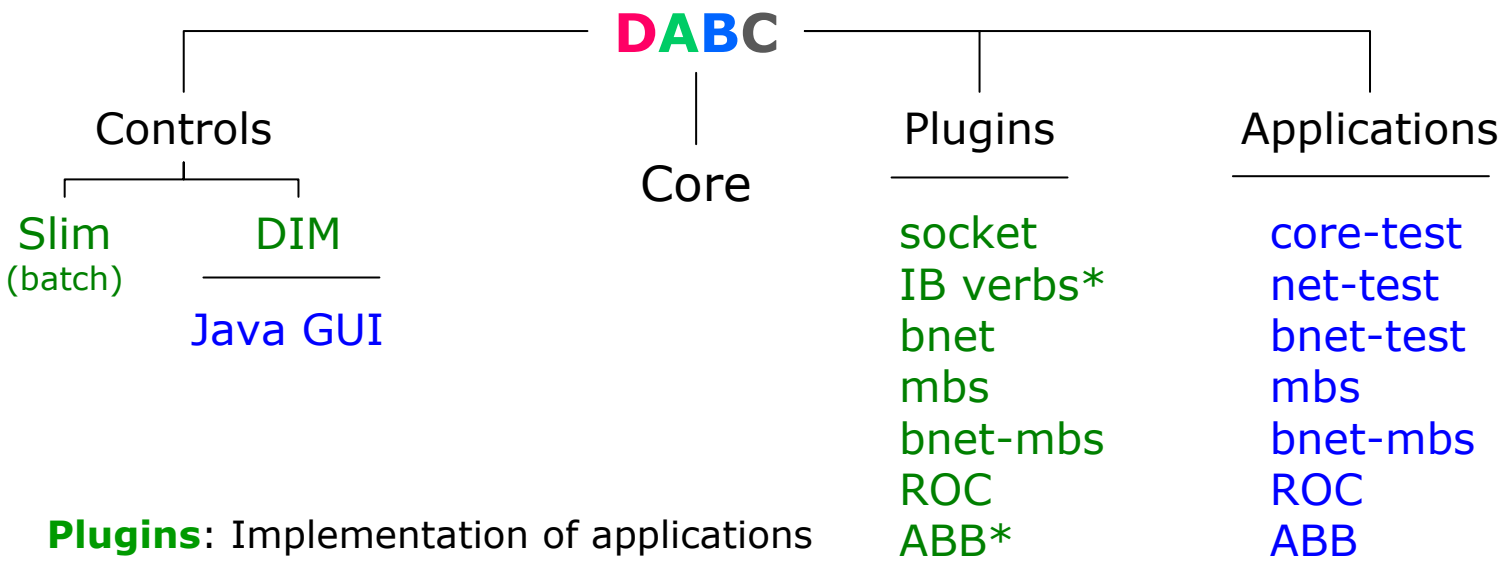
- represents hardware items
- manages several transports

Many other features

- Central dabc::Manager class
- Configuration with XML files
- Controlling interface and GUI
- dabc::Commands
- Factories (plugins) architecture
- Application and state machine
- Event building network (BNET)

DABC release 1.0

Download via http://dabc.gsi.de

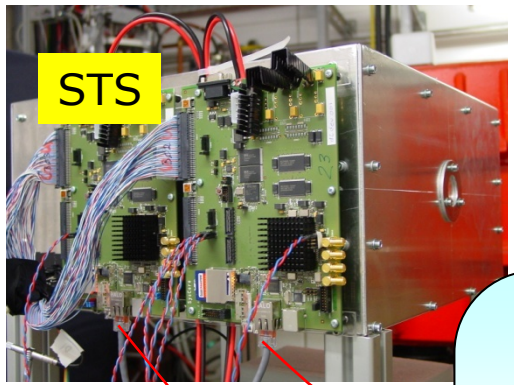


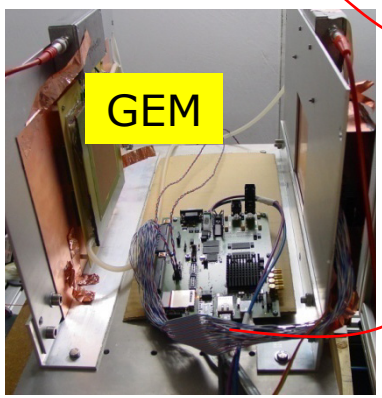
Plugins: Implementation of applications (programmers)
Applications: Mainly setup or testing programs (users)

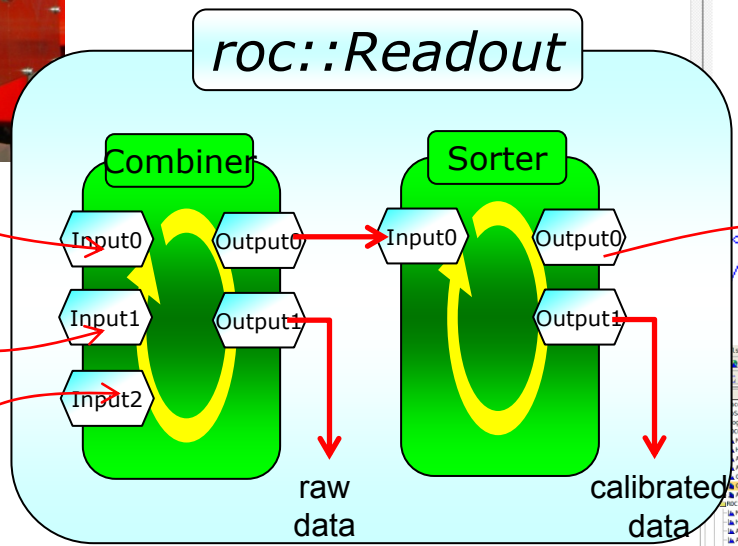
ROC: CBM Readout Controller board
ABB: Active buffer board (PCIe)
IB: InfiniBand
MBS: Multi Branch System (GSI DAQ)

* external packages needed

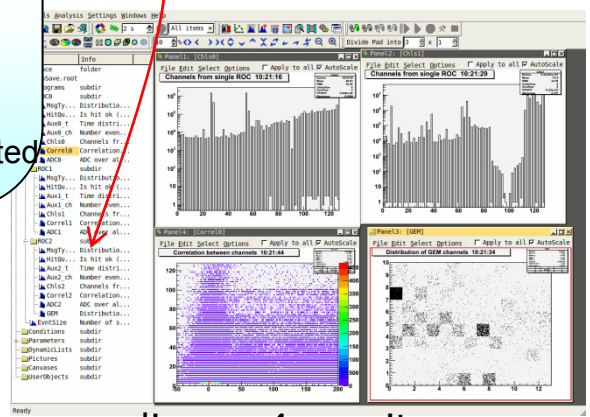
DABC in CBM test beam 08





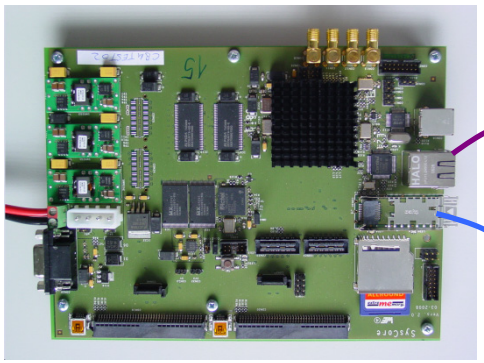



```
<?xml version="1.0"?>
<dabc version="1">
<Context name="Readout">
  <Run>
    <lib value="libDabcMbs.so"/>
    <lib value="libDabcKnut.so"/>
  </Run>
  <Application class="roc::Readout">
    <DoCalibr value="0"/>
    <NumRocs value="3"/>
    <BufferSize value="65536"/>
    <NumBuffers value="100"/>
    <RawFile value="run090.lmd"/>
    <MbsFileSizeLimit value="110"/>
    <RocIp0 value="cbmtest01"/>
    <RocIp1 value="cbmtest02"/>
    <RocIp2 value="cbmtest04"/>
    <TransportWindow value="30"/>
    <MbsServerKind value="Stream"/>
  </Application>
</Context>
/dabc>
```



online go4 monitor

DABC as access layer to ROC



Ethernet

ROC/udp plugin

DABC

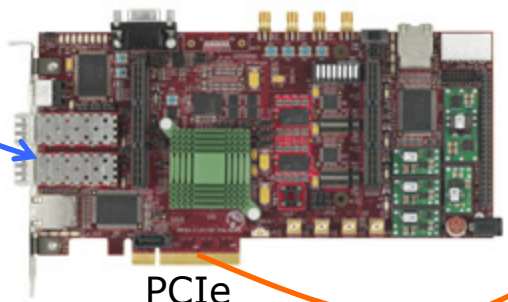
ROC/PCIe plugin

User access layer

Users scripts, GUIs

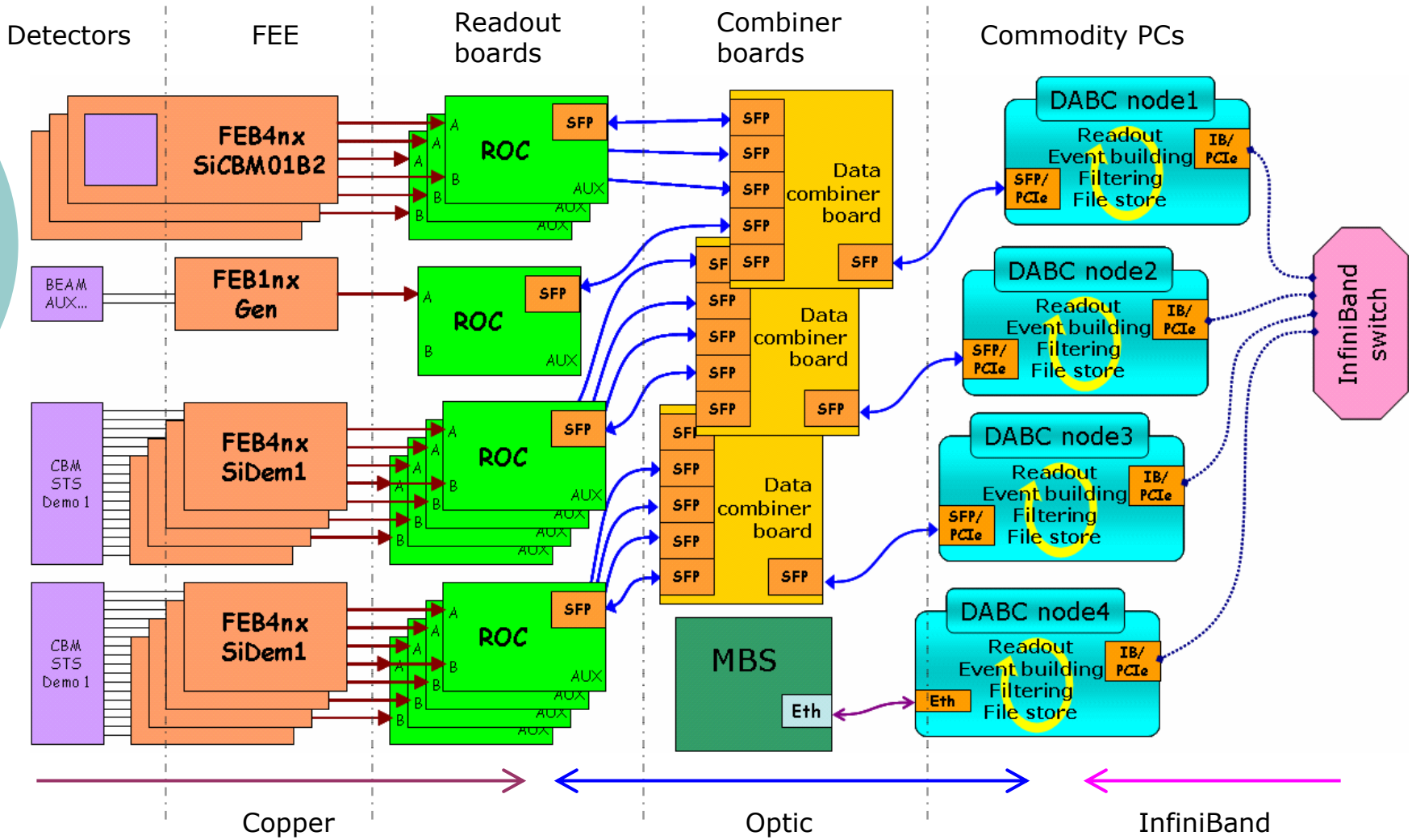
file I/O
online monitor

optic

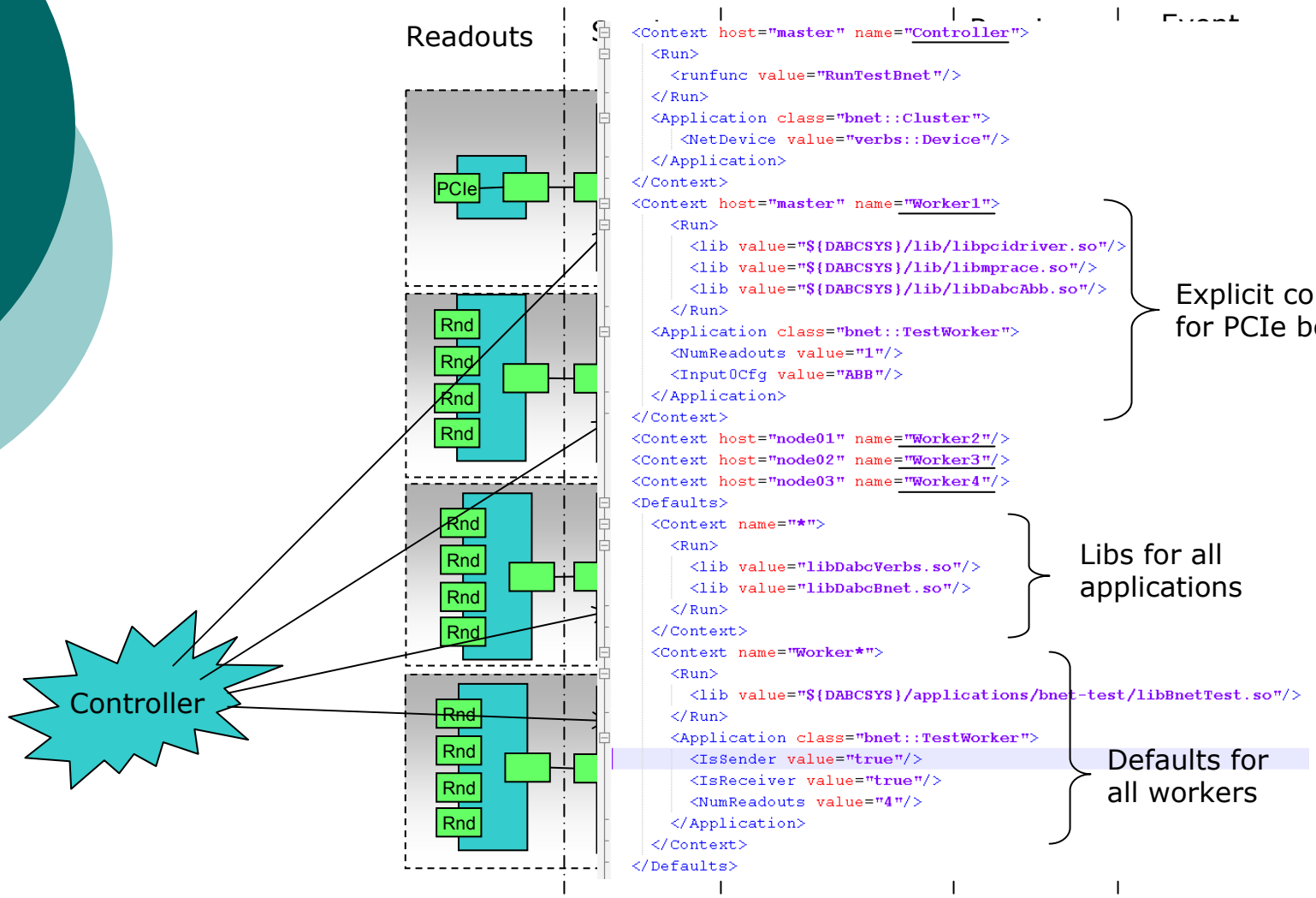


PCIe

Planned CBM beam tests



Event building network test



Conclusion

- DABC is general-purpose DAQ framework
- Can be used for different purposes – from small detectors tests to multi-nodes application
- Through its plugin architecture can be easy extended to specific needs
- Open for improvements and new ideas

Thanks!