Software/firmware activities @ MZ

- Working concurrently on support of existent hardware and pre-phase1 circuitry
  - Software (Jan) and firmware (Uli) for JEM processors
  - Software (Jan, Christian) and firmware (Christian, Patric, Stephan) for L1Topo

- Software for current systems is VME/HDMC based
  - Mainz in charge of JEM module services / register maps …
  - Some contribution made to test vector generation in the not so recent past
  - Probably just minor modifications required during LS1
- Software for L1Topo makes use of IPBus suite
  - Need to write module services from scratch
  - Register model, test vectors, GUIs, …, …
- L1Topo firmware under way
  - Regina: physics / algo firmware coordination
  - Stephan: algorithms
  - Patric: infrastructure – MGTs, real-time path before algos, including masking, error checking, play/spy, firmware based test vectors
  - Christian: IPBus
  - (Marek: readout)

Software

- Upgrade of JetProcessor pre-phase 1
  - Using updated firmware register map from Stockholm

|	Reserved			
Offset + 200	RoI0:Threshold1/Size1	12	0x3FF	Read/Write
Offset + 202	RoI0:Threshold2/Size2	12	0x3FF	Read/Write
|	|	12	0x3FF	Read/Write
Offset + 214	RoI3:Threshold1/Size1	12	0x3FF	Read/Write
Offset + 216	RoI3:Threshold2/Size2	12	0x3FF	Read/Write
|	|	12	0x3FF	Read/Write
Offset + 228	RoI7:Threshold1/Size1	12	0x3FF	Read/Write
Offset + 230	RoI7:Threshold2/Size2	12	0x3FF	Read/Write

Bits	Function
Threshold / FCAL Threshold	
0 - 9	Threshold value
10 - 11	Cluster size

- Upgrade of EnergySumProcessor pre-phase 1
  - Firmware not yet updated
  - Updates to playback and spy memory
  - changes in the module services, simulation and test vectors
- Integration of IPBus sw/register model for L1Topo
  - In cooperation with Christian and Murrough
  - Trying to setup IPBus software and L1Calo package on a virtual machine in Mainz
  - Write module services from scratch
  - Register model, test vectors, … to be done

L1Topo Module Control
IPBus Protocol

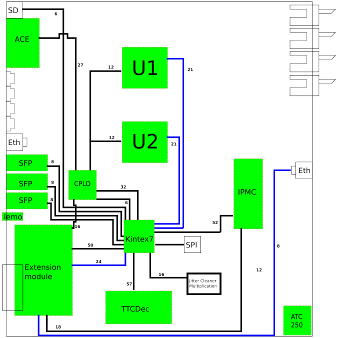
Christian Kahra

Institut für Physik - ETAP
Johannes Gutenberg-Universitaet Mainz

27. Juni 2013

L1Topo Module Control

- Communication between PC and Control-FPGA via Ethernet
- read-, write-, ... requests on module registers transacted by IPBus
- on prototype module the Processor FPGAs are connected with Control FPGA by 21 LVDS-pairs respectively
  - these 21 LVDS-pairs are shared by Control, ROD and TTC



Control-Processor - Connection

Standard approach: Every FPGA is IPBus-Endpoint

- sharing the Ethernet MAC via AXI4-Stream (David Sankey)
- AXI4 signals:
  - receive side: mac_rx_data[7:0], mac_rx_error, mac_rx_last, mac_rx_valid
  - transmit side: mac_tx_data[7:0], mac_tx_error, mac_tx_last, mac_tx_valid, mac_tx_ready
- every Endpoint sees whole traffic, discarding unwanted packets
- not possible on the prototype module because of limited bandwith

Alternative: Processor FPGAs are IPBus-Slaves

- IPBus-Slave on the Control FPGA represents Processor FPGA
  - IPBus data-, address- and control- lines are serial connected to the Processor FPGA
  - transaction is holded until acknowledge from the Processor FPGA arrives

IPBus address space

Description of the IPBus address space

- not coherent on firmware and software side
- a common XML description of the whole L1Topo firmware is in discussion
  - IPBus addresses would have to be part of this description

Currently:

Textfile

```
# num   name         base     addr_width

0       ctrl_reg     0x0      1
1       reg          0x2      0
2       ram          0x1000   12
3       pulse_reg    0x0100   1
```

↓ Python-Skript

VHDL-File → Firmware

```vhdl
function ipbus_addr_sel(signal addr : in std_logic_vector(31 downto 0))
return integer is
  variable sel : integer;
begin
    if    std_match(addr, "-------------------0---0------0-") then
       sel := 0; -- ctrl_reg / base 00000000 / mask 00000001
    elsif std_match(addr, "-------------------0---0------1-") then
       sel := 1; -- reg / base 00000002 / mask 00000000
    elsif std_match(addr, "-------------------1------------") then
       sel := 2; -- ram / base 00001000 / mask 00000fff
    elsif std_match(addr, "-------------------0---1------0-") then
       sel := 3; -- pulse_reg / base 00000100 / mask 00000001
    else
       sel := 99;
    end if;
    return sel;
 end ipbus_addr_sel;
```

XML-File

```xml
<?xml version="1.0" encoding="LATIN1"?>
<chip name="FPGA1">
        <branch name="Level_1" address="0x0200" mask="0x01FF">
                <branch name="Level_2" address="0x0100" mask="0x00FF">
                        <leaf name="register" address="0x0000"/>
                        <leaf name="register1" address="0x0001"/>
                        <leaf name="register2" address="0x0002"/>
                        <leaf name="register3" address="0x0003"/>
                        <array count="256" spacing="0x0000001" >
                                <leaf name="register"/>
                        </array>
                </branch>
        </branch>
</chip>
```

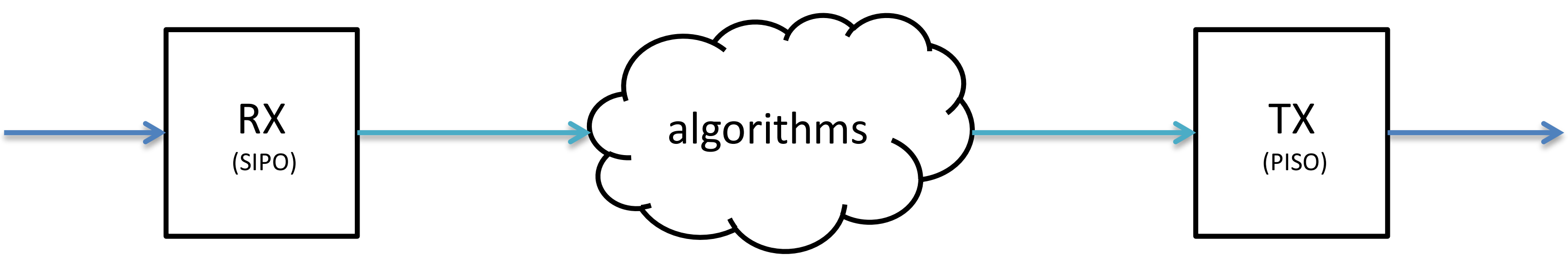
uHAL(Boost XML-Parser) ↓

Software using uHAL

```cpp
#include <iostream>
#include "uhal/uhal.hpp"

using namespace uhal;

int main()
{
  try
  {
    ConnectionManager connectionManager( "file://~/exampleForManuel/cfg/connections.xml;" );

    HwInterface hw = connectionManager.getDevice("Board1");

    std::cout << "Attempting single-word write of '0xcafebabe' to register1..." << std::endl;
    hw.getNode ("register1").write(0xcafebabe);
    hw.dispatch();
    std::cout << "Write of '0xcafebabe' to register1 performed successfully!\n" << std::endl;

    std::cout << "Reading back the content of register1..." << std::endl;
    ValWord<uint32_t> result = hw.getNode ("register1").read();
    hw.dispatch();
    std::cout << "register1 contains the value: 0x" << std::hex << result.value() << std::dec <<std::endl;
  }
  catch ( const std::exception& e ){
              std::cout << "Something went wrong: " << e.what() << std::endl;
  }
  return 0;
}
```

Current MGT status

- Work on Top-Level-VHDL module with instantiated MGTs and connected algorithms
- C++ based instantiation and mapping of MGTs in development
- Currently also busy with BERT on AVAGO-MINIPOD mezzanine board based on Wojciechs format



MET,HT and Meff

- MET=$\sqrt{E_x{}^2 + E_y{}^2}$
  → implemented with CORDIC

resource use [%]	latency [ns]
0.11	~50

- HT=Σp_T(jets) and Meff= Σp_T(jets,MET) with 64 jet TOBs

resource use [%]	latency [ns]
0.44	23.1

- all implementations on xc7vx690tffg1927-2

ϕ_{MET}

- $\phi_{\mathrm{MET}}=\arctan\left(\frac{E_x}{E_y}\right)$
  - CORDIC implementation (without format change)

resource use [%]	latency [ns]
0.12	17.2

  - use sign of Ex and Ey to calculate quadrant,
  - use relative size to calculate octant,
  - use $E_x > E_y \cdot$threshold (with 7 thresholds) to get 0.1 granularity

resource use [%]	DSP use [%]	latency [ns]
0.06	0.19	12.5

Transverse/Contratransverse Mass

- $M_T = \sqrt{2 E_T MET (1 - \cos(\Delta\phi))}$
  $\rightarrow M_T^2 = 2 E_T MET (1 - \cos(\Delta\phi))$

resource use [%]	DSP use [%]	latency [ns]
0.03	0.06	13.1

- $M_{CT} = \sqrt{2 E_T MET (1 + \cos(\Delta\phi))}$
  $\rightarrow M_{CT}^2 = 2 E_T MET (1 + \cos(\Delta\phi))$

resource use [%]	DSP use [%]	latency [ns]
0.02	0.06	12.5

Testimplementation with multiple algorithms

- maximum number of cluster TOBs
  - with sorting 6
  - with selection 6
- maximum number of jet TOBs
  - with sorting 10
  - with selection 10
- whole algorithm module has a latency of 3 BC
- currently reworking selection algorithm