L1Topo Module Control
IPBus Protocol

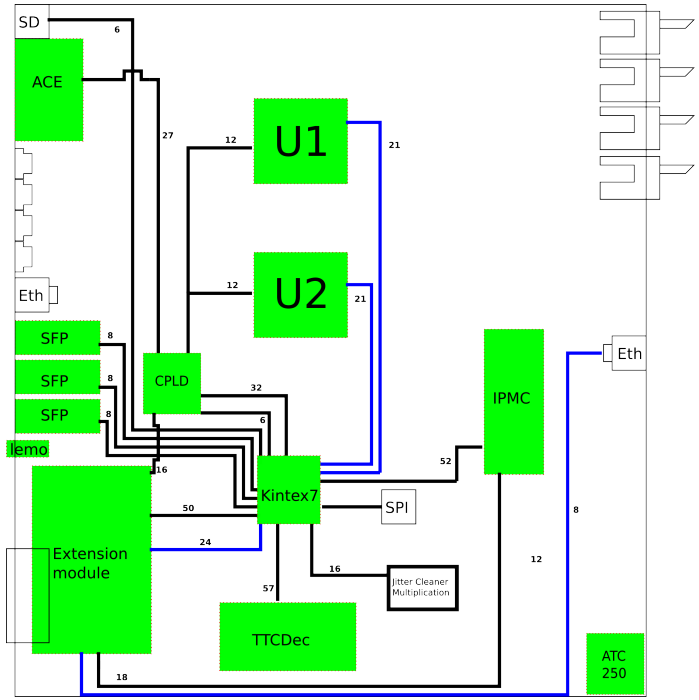
Christian Kahra

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

```
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 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

```
#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;
}
```