IPbus: status & plans

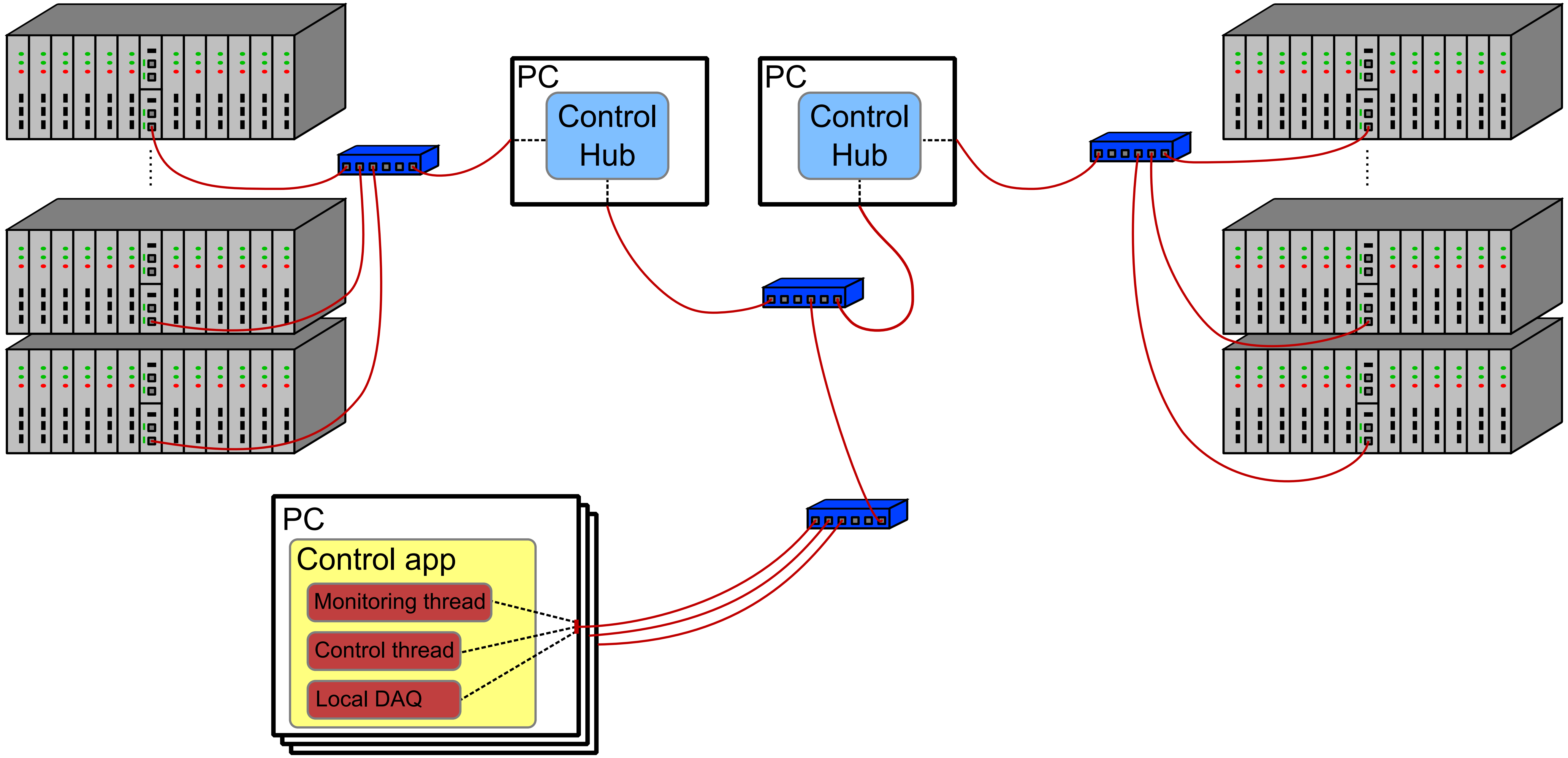
D. Newbold, A. Rose, D. Sankey, A. Thea, T. Williams
Bristol, Imperial, RAL

What is IPbus?

- IPbus: A simple (Ethernet-based) control protocol
  - Designed for controlling Ethernet-/IP-based hardware - i.e. read/write registers, etc.
- Suite of firmware and software:
  1. **IPbus firmware core**
     - Reference VHDL implementation of IPbus 2.0 UDP server
     - Complete system-on-chip implementation
     - Interprets and implements IPbus transactions (read, write, …) on FPGA
  2. **uHAL library**
     - C++ and Python end-user Hardware Access Library
     - Design mimics recursive modularity of firmware blocks
  3. **ControlHub**
     - Software application analogous to VME crate controller
     - Mediates/Arbitrates simultaneous hardware access from multiple clients
     - Implements IPbus reliability mechanism
  - Widely used: CMS, ATLAS & ALICE upgrades; FNAL g-2, SOLID, COMET, COMPASS
  - **Reference:** JINST 10 (2015) 02, C02019 - DOI 10.1088/1748-0221/10/02/C02019
  - Documentation, installation instructions, etc - http://cactus.web.cern.ch

Example system layout

- Full-scale system, large experiment:

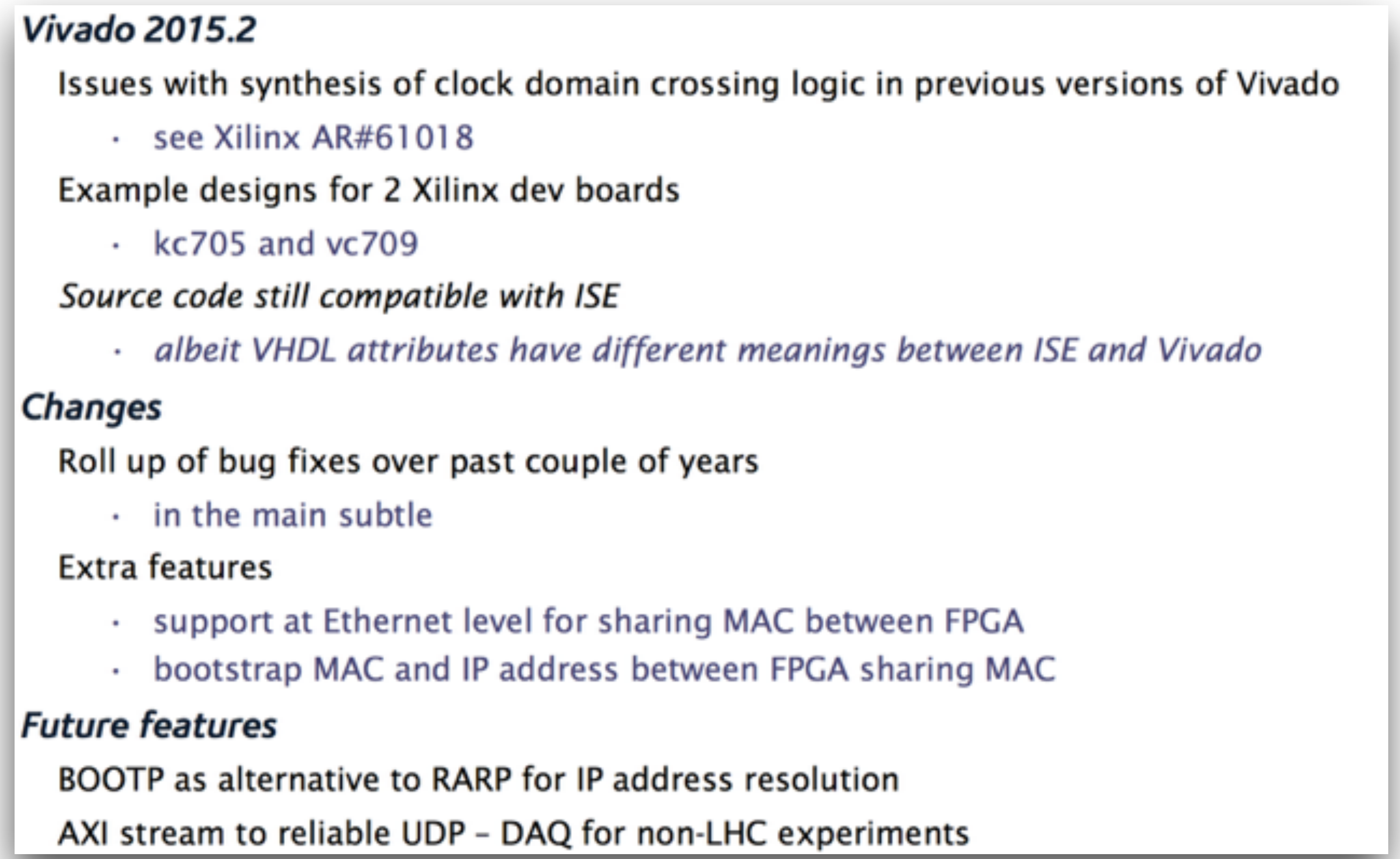


Software status & plans

- Since last xTCA Interest Group presentation (April 2013):
  - Various releases:
    - 2013/2014: Performance improvements, bugfixes - v2.1.0 (Nov 2013), v2.2.0 (Jan 2014)
    - 2014/2015: Minor improvements for large projects & deployment at Points 1 & 5
      - v2.3.x (April - Oct 2014):
        - uHAL: Update versions of external dependencies; fixes for compilation on OS X
        - ControlHub: Run as Linux service (i.e. control via /sbin/service)
      - v2.4.x (Feb 2015 onwards):
        - uHAL: C++ code cleanup for newer g++ versions
        - ControlHub: Added configuration files (e.g. for TCP port number, timeouts, etc.); start / stop without being root; log via syslog
  - **Latest release: v2.4.2**
    - **Main supported platform** (tests before release, RPMs + YUM repo): **SLC6**
    - Help with other platforms on best effort basis.
- Plans for near future:
  - Add **CERN CentOS 7** as supported platform
  - Review documentation (user reports that some areas slightly out of date)

Firmware status (1)

- Since last xTCA Interest Group presentation (April 2013):
  - Nov 2013: Firmware tag (ipbus_2_0_v1)
    - Extensive soak testing on Virtex 5, 6 & 7 -based boards
  - Changes since then, will be integrated into "v2" tag:
    - Support for Xilinx ultrascale; bug fixes (mostly for rare scenarios)



Vivado 2015.2

Issues with synthesis of clock domain crossing logic in previous versions of Vivado

- see Xilinx AR#61018

Example designs for 2 Xilinx dev boards

- kc705 and vc709

Source code still compatible with ISE

- *albeit VHDL attributes have different meanings between ISE and Vivado*

Changes

Roll up of bug fixes over past couple of years

- in the main subtle

Extra features

- support at Ethernet level for sharing MAC between FPGA
- bootstrap MAC and IP address between FPGA sharing MAC

Future features

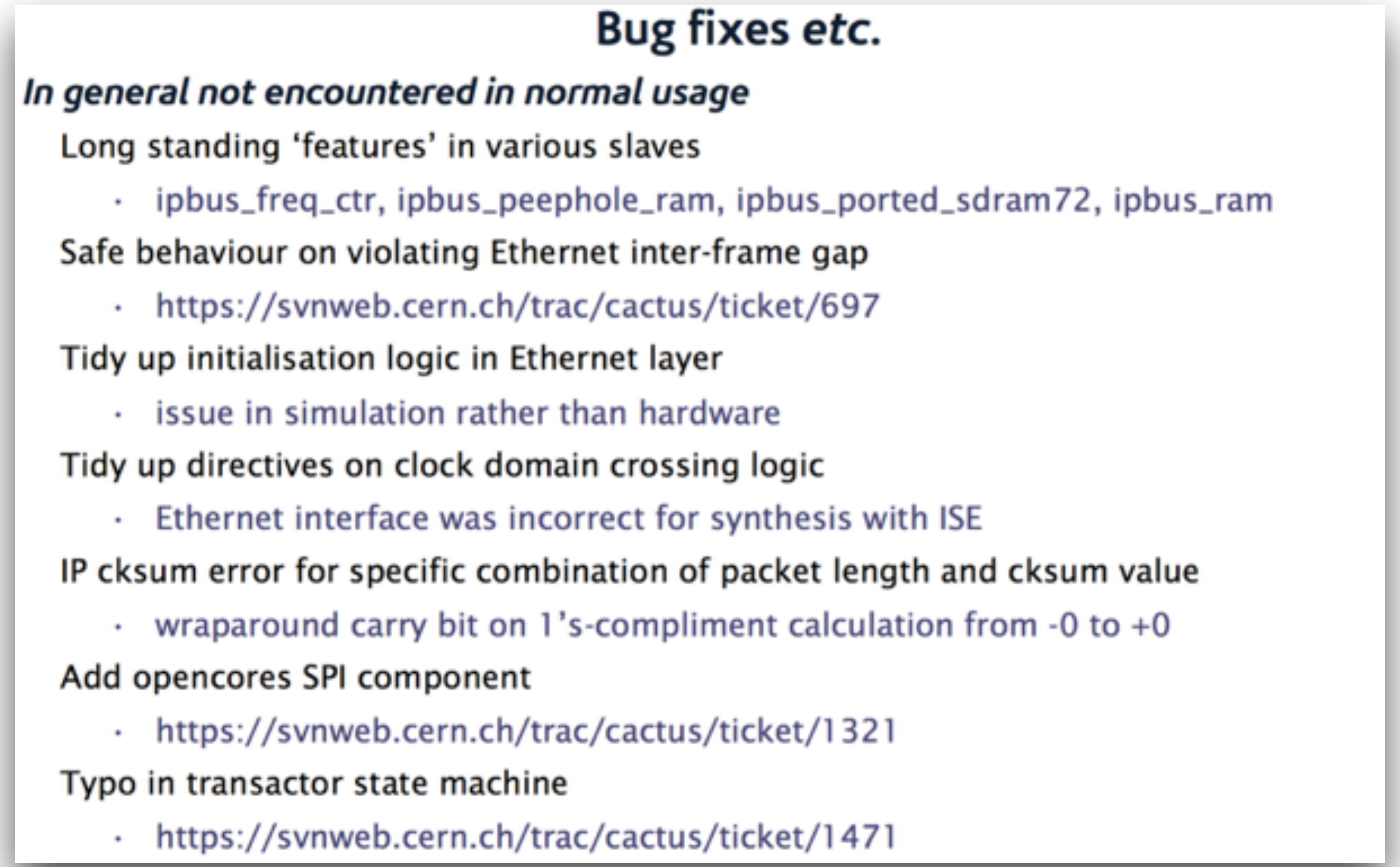
BOOTP as alternative to RARP for IP address resolution

AXI stream to reliable UDP – DAQ for non-LHC experiments

D. Sankey

Firmware status (2)

- Since last xTCA Interest Group presentation (April 2013):
  - Nov 2013: Firmware tag (ipbus_2_0_v1)
    - Extensive soak testing on Virtex 5, 6 & 7 -based boards
  - Changes since then, will be integrated into "v2" tag:
    - Support for Xilinx ultrascale; bug fixes (mostly for rare scenarios)



Bug fixes *etc.*

In general not encountered in normal usage

Long standing 'features' in various slaves
- ipbus_freq_ctr, ipbus_peephole_ram, ipbus_ported_sdram72, ipbus_ram

Safe behaviour on violating Ethernet inter-frame gap
- https://svnweb.cern.ch/trac/cactus/ticket/697

Tidy up initialisation logic in Ethernet layer
- issue in simulation rather than hardware

Tidy up directives on clock domain crossing logic
- Ethernet interface was incorrect for synthesis with ISE

IP cksum error for specific combination of packet length and cksum value
- wraparound carry bit on 1's-compliment calculation from -0 to +0

Add opencores SPI component
- https://svnweb.cern.ch/trac/cactus/ticket/1321

Typo in transactor state machine
- https://svnweb.cern.ch/trac/cactus/ticket/1471

D. Sankey

Community-driven changes

- Variety of contributions & suggestions from the community, esp. related to firmware, e.g:
  - Shims for Altera devices
  - Generating address decoder/slave VHDL down to level of registers & bit masks
  - Interface to soft CPU

- Currently in CMS Level-1 Trigger repository (CACTUS):
  - With community-driven changes, it may make more sense to move IPbus firmware to experiment-independent repo
  - Two phase process (after “v2” firmware release):
    - First move to independent part of CACTUS repo (as a test bench)
    - Then, separate repo
  - Watch this space!

Summary

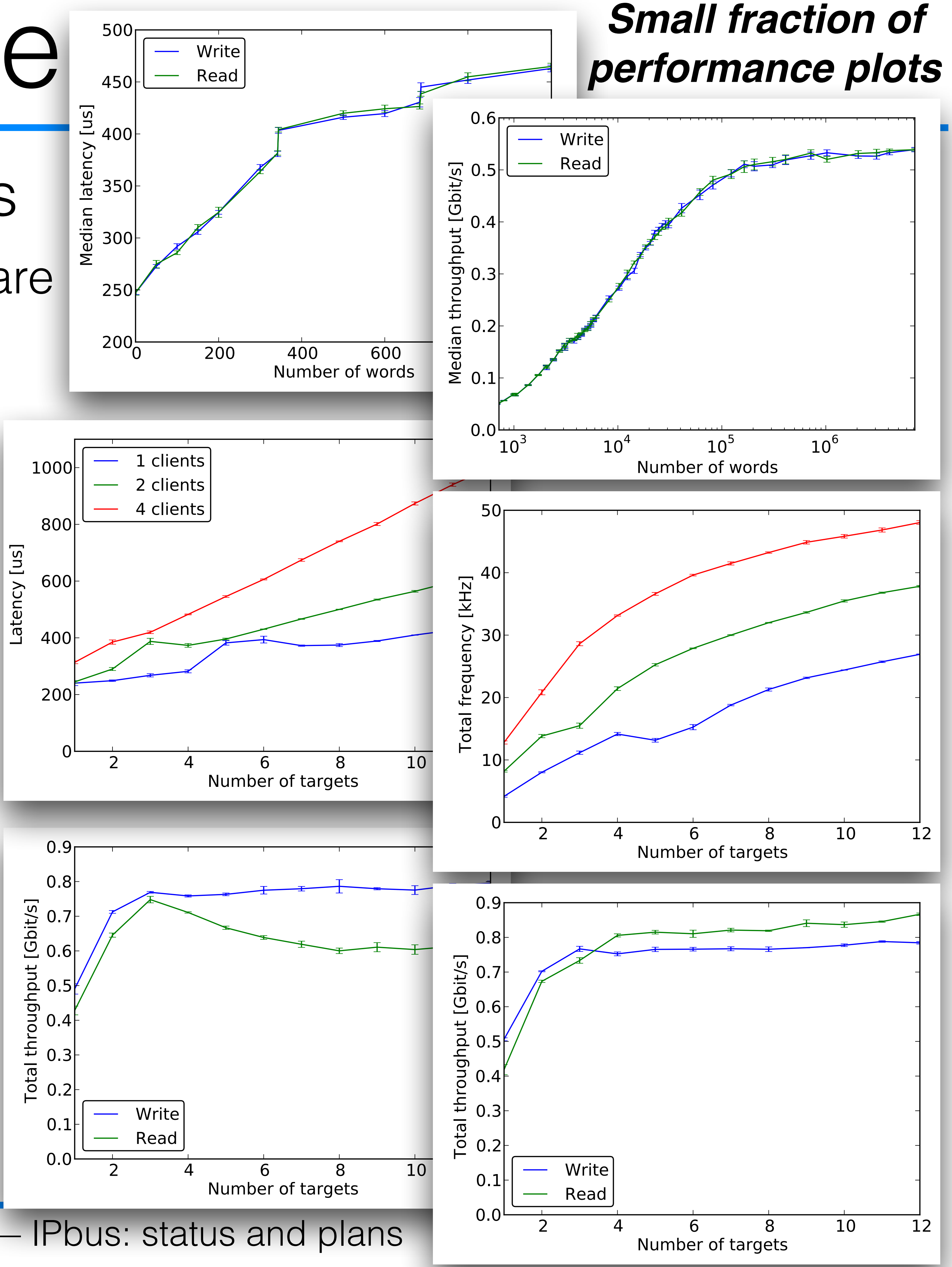
- Roadmap over coming months:
  - Software:
    - New release for CERN CentOS 7
  - Firmware:
    - New tag
    - Incorporate community-driven changes
  - Documentation:
    - Cleanup campaign

- To stay informed:
  - Sign up to IPbus user announcements egroup: ipbus-users@cern.ch

BACKUPS

Performance

Small fraction of performance plots



- Took many measurements
  - Using realistic layout & hardware
  - Understand the results
- Repeated measurements using hardware from diff vendors
  - i.e. NAT vs VadaTech MCH
  - Mitigate with different ControlHub config file
- Optimised performance
  - Large block read/write throughput significantly increased
  - Medium-perf DAQ systems

The IPbus ecosystem (1)

- Goal: Easy to use IPbus from day 1
  - Equally so for VHDL simulation, board on bench top in lab, or at Point 5
  - Achieved with scalable end-to-end chain of common software / firmware components
- Firmware:
  - IPbus master
  - IPbus fabric — Can define **hierarchical address-decoding trees** that **match** the **modularity of firmware blocks** (transceivers, buffers, trigger algorithm, DAQ, TTC)
  - IPbus slaves — Ready-made, plug-and-play entities for interfacing with registers, RAMs, FIFOs, … (even across clock domain crossing)
- Address decode script (`gen_ipbus_addr_decode`)
- Software (uHAL + ControlHub):
  - 1-line installation — RPMs, from YUM repository: `sudo yum groupinstall uhal`

The IPbus ecosystem (2)

- Address table decode script (`gen_ipbus_addr_decode`)
  - Automatically generates the firmware "address decode" logic corresponding to the XML address files used by software
    - i.e. generates VHDL that specifies IPbus fabric tree
  - **Crucial** to allow **common firmware modules** (e.g. TTC, DAQ, transceivers) to be **easily & quickly integrated** into different firmware designs

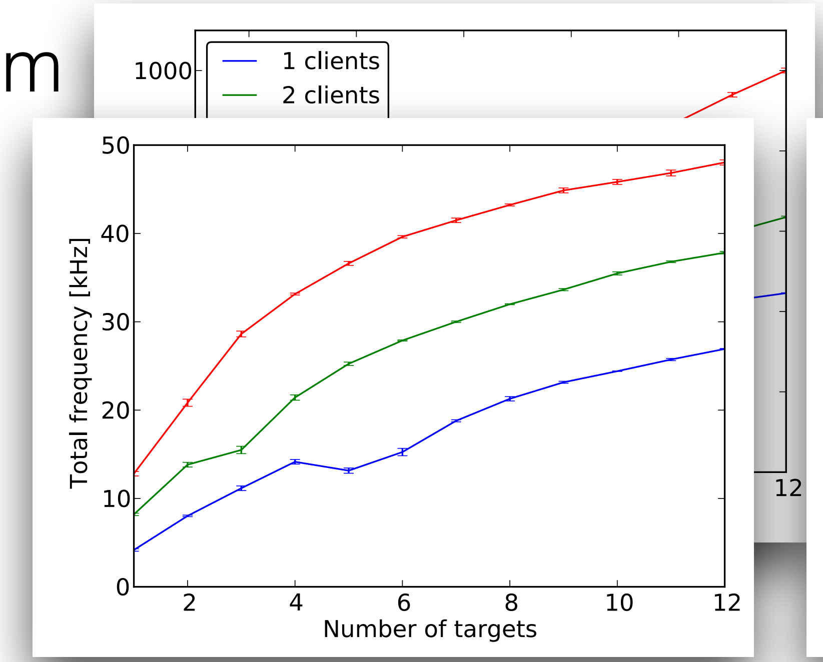
Code management: Firmware

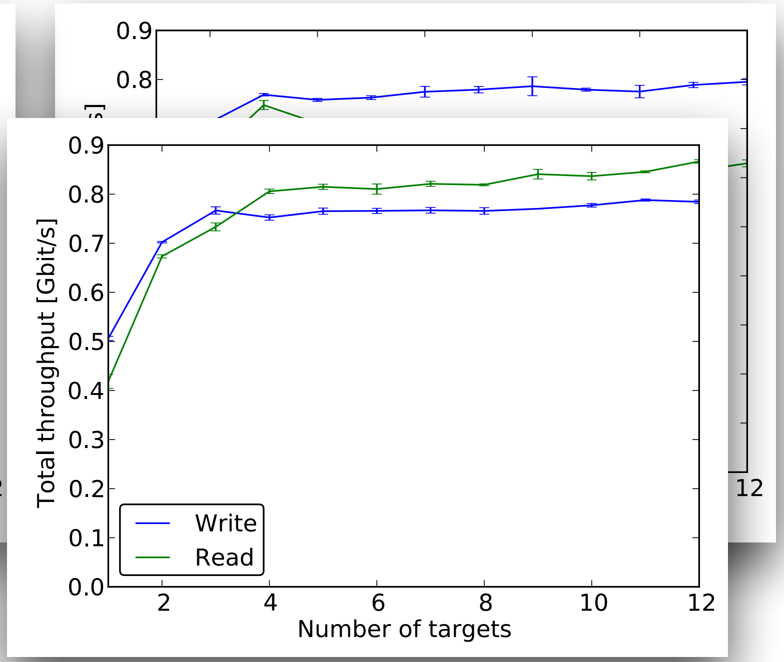
- Phase-1 trigger upgrade:
  - From MP7: **Common packages** for links, DAQ, TTC, …; used in multiple boards, for multiple purposes (calo trigger, muon trigger, global trigger)
    - ~ 130k lines of VHDL
    - Matching **modular structure of slow control firmware** encoded in IPbus design
  - Firmware image now depends on multiple packages, all with different release cycles, from different areas of repository … how do we manage this effectively?
    - **Need:** Reproducible build in few commands, without editing any source code
    - **Solution**: **Build tool** ("ProjectManager.py"):
      - E.g. For stage-1 calo trigger:

```
> ./ProjectManager.py create tags/mp7/stable/firmware/mp7fw_v1_8_0 -u MY_USERNAME
> ./ProjectManager.py addproj s1calol2 tags/s1calol2/fw_01010100
> ./ProjectManager.py vivado projects/s1calol2/processor
> cd processor
> make project
> make bitfile
```

BACKUPS: More information

- IPbus protocol and suite
  - Extensively-tested, tightly-integrated suite with Gigabit performance
  - Easily scalable control system
  - Applicable to any hardware with Ethernet interface
- For more information …
  - Main page: http://cactus.web.cern.ch/
  - Firmware: https://svnweb.cern.ch/trac/cactus/wiki/IPbusFirmware
  - Software (uHAL + ControlHub):
    Quick start tutorial (Easy installation on SL(C) 6)
    https://svnweb.cern.ch/trac/cactus/wiki/uhalQuickTutorial
  - Bug reports, feature requests, clarifications
    https://svnweb.cern.ch/trac/cactus/newticket





BACKUPS: IPbus firmware core

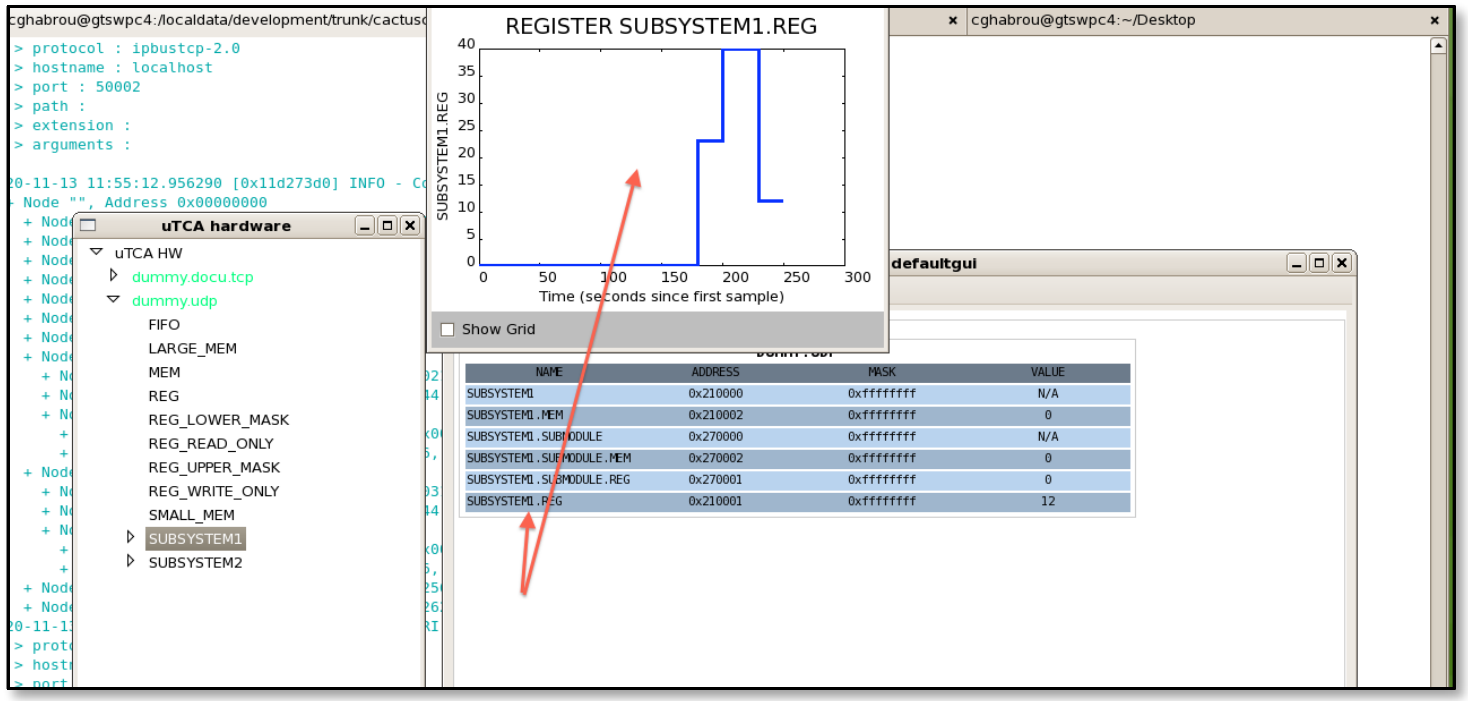
- Reference VHDL SoC implementation of IPbus 2.0 UDP server
  - Currently Xilinx-specific; but successfully adapted for Altera devices & custom ASICs
- Interprets IPbus transactions (read, write, …) on FPGA
- Transport protocol: UDP vs TCP
  - Main processing logic firmware (e.g. trigger algos) must fit on same FPGA
  - TCP: complex algorithm
    - Implementing full protocol in FPGA => high resource usage
  - UDP: Much simpler algorithm
    - Can implement in firmware with low resource usage
  - Use UDP, correct for packet loss with IPbus-level reliability mechanism
- Also, ICMP (unix ping command), ARP, and RARP (IP address assignment

Resource usage	FFs	Slices	BRAMs
Fully-featured	3500	2900	17
Minimal config	2000	1000	5

25% of smallest Spartan-6 chip

BACKUPS: uHAL

- C++ library providing end-user API for reads, writes, etc.
  - Also has Python bindings
- Register layout specific in XML files
  - Reflect hierarchical and modular nature of firmware
  - Promotes reuse and modularity of address table files
- Includes example GUI for hardware development
- Fast and scalable in conjunction with ControlHub



cghabrou@gtswpc4:/localdata/development/trunk/cactusc

```
> protocol : ipbustcp-2.0
> hostname : localhost
> port : 50002
> path :
> extension :
> arguments :

20-11-13 11:55:12.956290 [0x11d273d0] INFO - Co
+ Node "", Address 0x00000000
```

uTCA hardware

```
uTCA HW
  dummy.docu.tcp
  dummy.udp
    FIFO
    LARGE_MEM
    MEM
    REG
    REG_LOWER_MASK
    REG_READ_ONLY
    REG_UPPER_MASK
    REG_WRITE_ONLY
    SMALL_MEM
    SUBSYSTEM1
    SUBSYSTEM2
```

defaultgui

NAME	ADDRESS	MASK	VALUE
SUBSYSTEM1	0x210000	0xffffffff	N/A
SUBSYSTEM1.MEM	0x210002	0xffffffff	0
SUBSYSTEM1.SUBMODULE	0x270000	0xffffffff	N/A
SUBSYSTEM1.SUBMODULE.MEM	0x270002	0xffffffff	0
SUBSYSTEM1.SUBMODULE.REG	0x270001	0xffffffff	0
SUBSYSTEM1.REG	0x210001	0xffffffff	12

BACKUPS: ControlHub

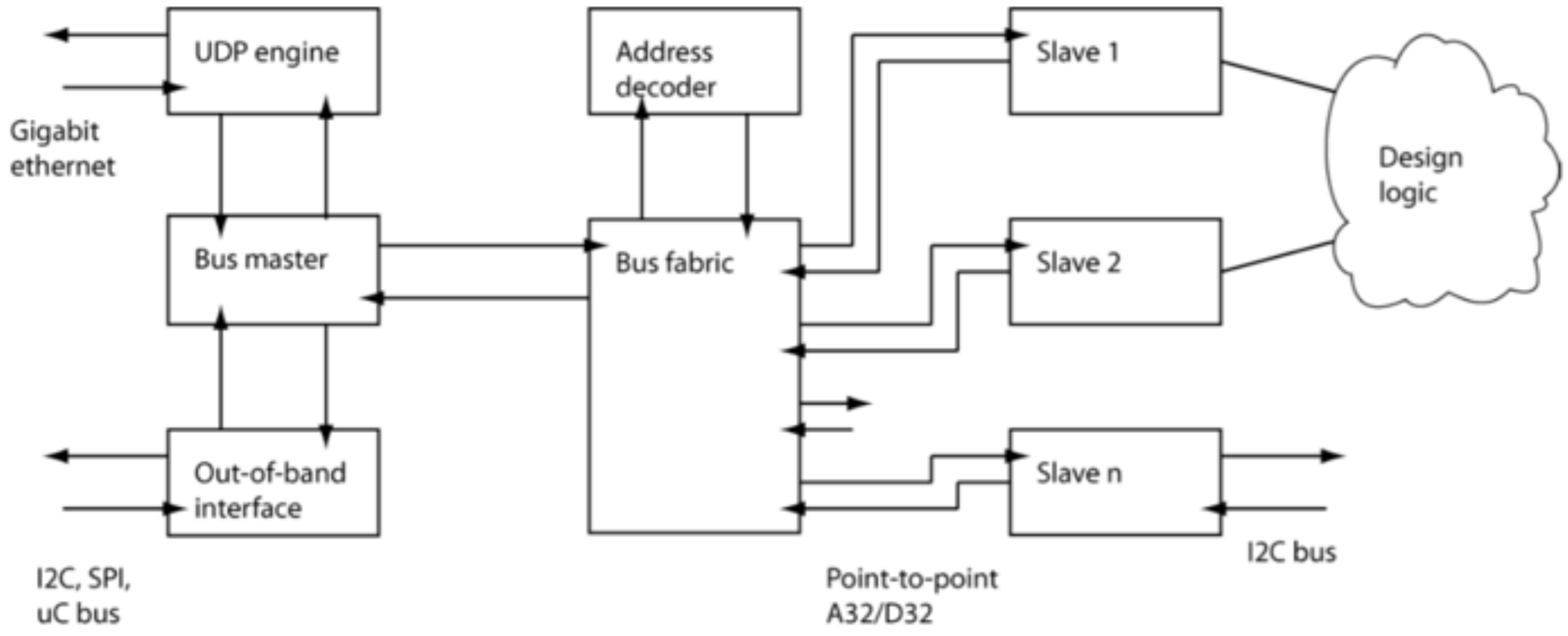
- Software application analogous to VME crate controller

- Purpose:
  - Route IPbus traffic from multiple control applications to single board
  - Implement packet-loss recovery over UDP
  - Also, implementation must allow multiple clients to communicate with multiple targets reliably, efficiently & independently

- Implemented in Erlang:
  - Concurrent programming language developed by Ericsson (J Armstrong et al)
  - Scales transparently across multiple CPU cores
  - Standard libraries for creating high-availability, fault-tolerant applications
  - Efficient, mature network protocol implementations

Backup: IPbus firmware design

- Bus topology


UDP engine
Gigabit ethernet
Bus master
Out-of-band interface
I2C, SPI, uC bus
Address decoder
Bus fabric
Point-to-point A32/D32
Slave 1
Slave 2
Slave n
I2C bus
Design logic