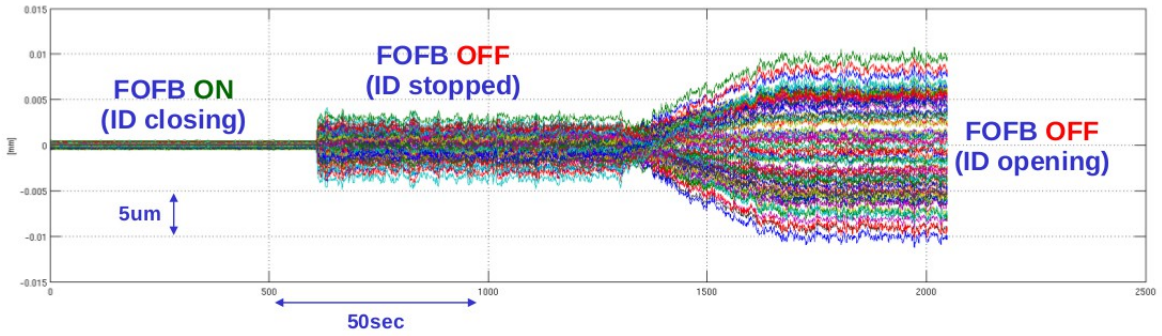
ALBA Fast Orbit FeedBack
Soft real time interface with Tango

28th Tango collaboration meeting
May 19 and 20, 2014

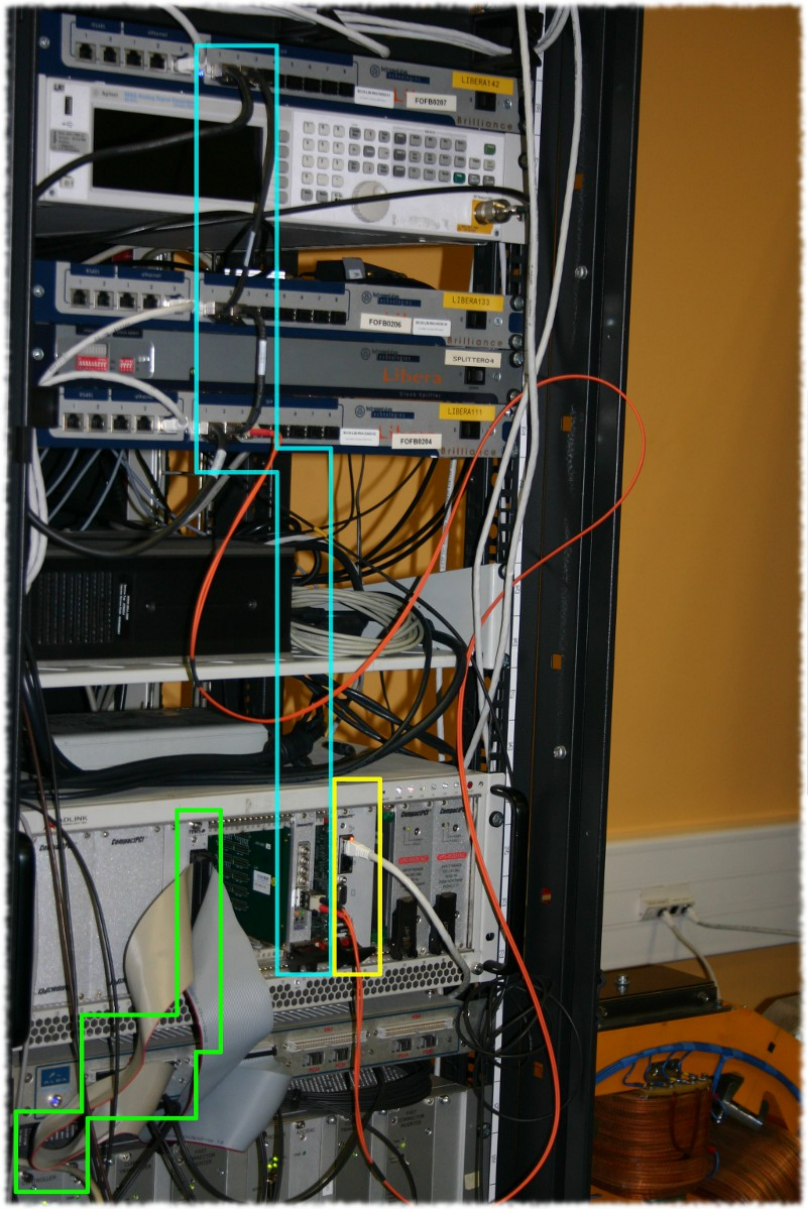
Jairo Moldes Fuentes
Computing division
ALBA

Introduction



- Motivation: increase beam stability by fast orbit corrections at 10 KHz (5 KHz achieved)
- This means that in 100 µsecs (200 µsecs for 5KHz) we have to:
  - **Read** the position of all the 88 BPMs included in the loop
  - **Compute** the correction to be applied to each of the 88x2 corrector power supplies
  - **Write** that correction into the power supply controllers
- How we do this? Problem is divided in 16 sectors. Each sector has a cPCI machine with a cPCI sniffer card and cPCI power supply controller interface card
  - **Reading** the 88 BPM position. This data is in each of the cPCI sniffer cards. 88 Libera BPM units + 16 sniffer cards each running DCC on its own FPGA, all connected with special fiber optics network, sharing their positions at 10 KHz
  - **Computing** corrections. Per sector, this is basically a ([10|12]x88) * (88x1) matrix multiplication using SSE optimization. Each CPU takes care of 10 or 12 correctors
  - **Writing**. That same CPU writes these computed values into the correctors

Hardware

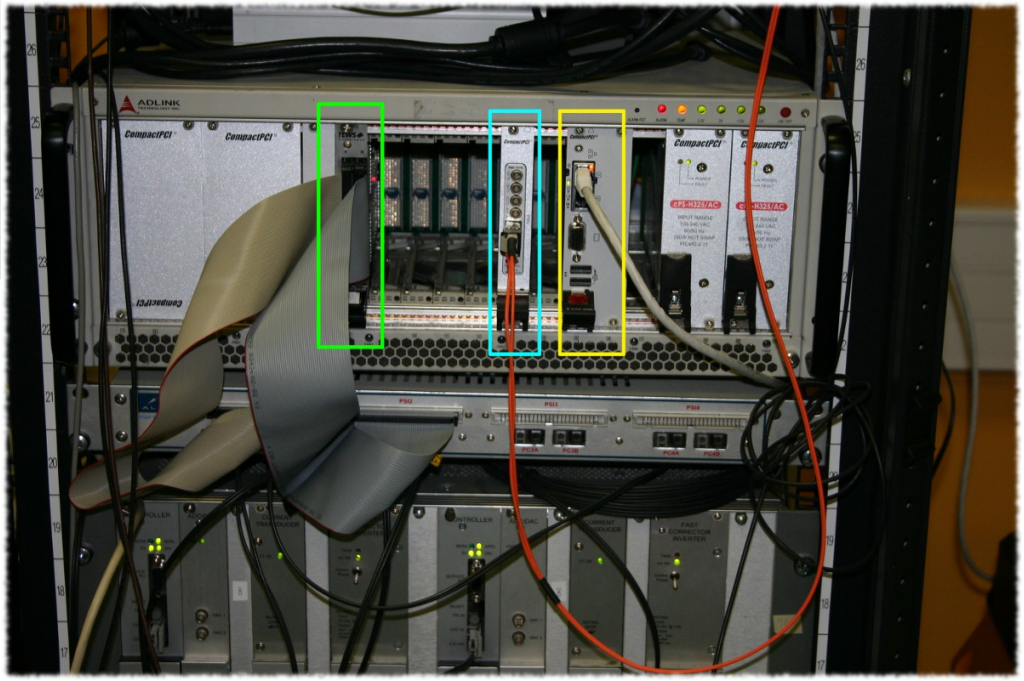


Read

- i-tech libera BPM (88x)
- cPCI sniffer (16x): reused MRF Event Receiver

Compute (16x) Adlink cPCI-3970 (4-core Intel i7)

Write (16x3) cPCI IP carrier for OCEM PS controller



Software

- Hard real time: critical applications (a timeout may be catastrophic, e.g. airbag)
- Soft real time: non critical applications (sporadic timeout is not catastrophic, e.g. FOFB)
- Linux kernel 2.6.27.29 compiled with SMP and PREEMPT flags (not PREEMPT_RT patches)
- 2 soft real time processes with shared memory (C, no Tango used here):
  - Process 1:
    - Wait for interruption from the sniffer card driver (synchronization)
    - Read the data into shared memory
    - Signal process 2
  - Process 2:
    - Wait for signal
    - Compute corrections with data from shared memory
    - Write those corrections into the power supplies
- To achieve soft real time performance we have to set:
  - RT scheduler (PREEMPT flag) and max priority for both processes
  - CPU and IRQ affinities for both processes (SMP flag)
- PyTango DS for controlling a sector (16): shared memory with the 2 non Tango processes
- PyTango DS for controlling the 16 device servers
- User GU (to be done)

Status and conclusions

- Very satisfactory results in tests (still not in production)
- Advantages:
  - [Almost] no need for extra new hardware (new CPUs, sniffers are reused receivers)
  - Faster development (C vs VHDL)
  - Flexibility: changes could be done [almost] online for tests (no need to develop/change hardware's firmware, drivers...)
  - Tango easy and quick integration: only modifications in shared memory structure are needed
- Disadvantages:
  - No hardware reliability (e.g. general OS may crash)
  - No hardware determinism (but timeouts are detected)
- Shared memory + soft real time process(es) + Tango DS: poor man's (but effective) approach for interfacing real time with Tango (maybe something more sophiscated inside Tango?)

Special thanks to

- David Fernández, who first thought of the software solution
- Albert Gutiérrez, who started integration of drivers and development of the algorithm
- Xevi Serra, who took care at Alba of the sniffer firmware
- Michael Abbott and Isa Uzun, who provided both the Diamond Communication Controller firmware and driver for the sniffers
- Alejandro Ohms for his assistance in software development
- Roberto Petrocelli and Domingo Alloza, for the help with the OCEM power supplies
- Ángel Olmos, user of the system
- An many others...