Status and Perspectives of the Real-Time Control of VIRGO Gravitational Waves Detector

Alberto Gennai

on behalf of the VIRGO Collaboration

RICAP 2024 – Rome International Conference on AstroParticle Physics 23–27 Sept 2024

Abstract

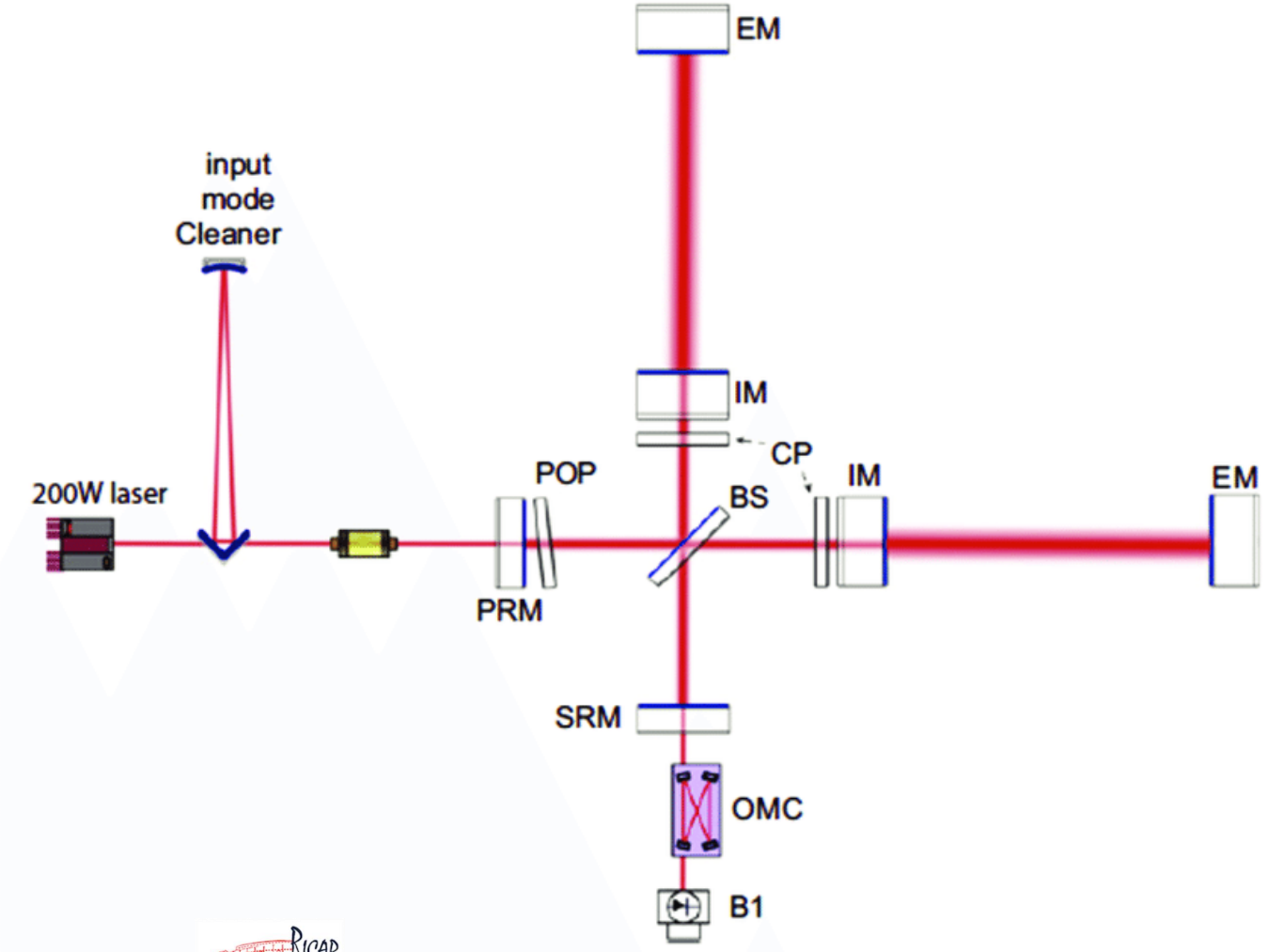
- The Advanced VIRGO detector is currently taking data during the second part of the fourth observation run called O4b, which began last April and is scheduled to finish in spring 2025.
- VIRGO uses a custom complex real-time control system consisting of 135 multicore DSP processors and more than 1,000 channels of high-resolution analog inputs and outputs.
- In this talk we summarize the main requirements with associated technical choices focusing mainly on electronic hardware, entirely designed and produced by INFN. We also presents our plans for upcoming upgrades targeting next VIRGO observation run O5 and that will also lead to the conceptual design of the control system for the third-generation ground based detector Einstein Telescope.

- VIRGO is a large interferometer located near Pisa in Tuscany, designed and built by a collaboration between the French CNRS and the Italian INFN.
- Today, there are more than 1000 Members in the Virgo Collaboration, representing more than 100 Institutions from several countries.

The VIRGO Collaboration

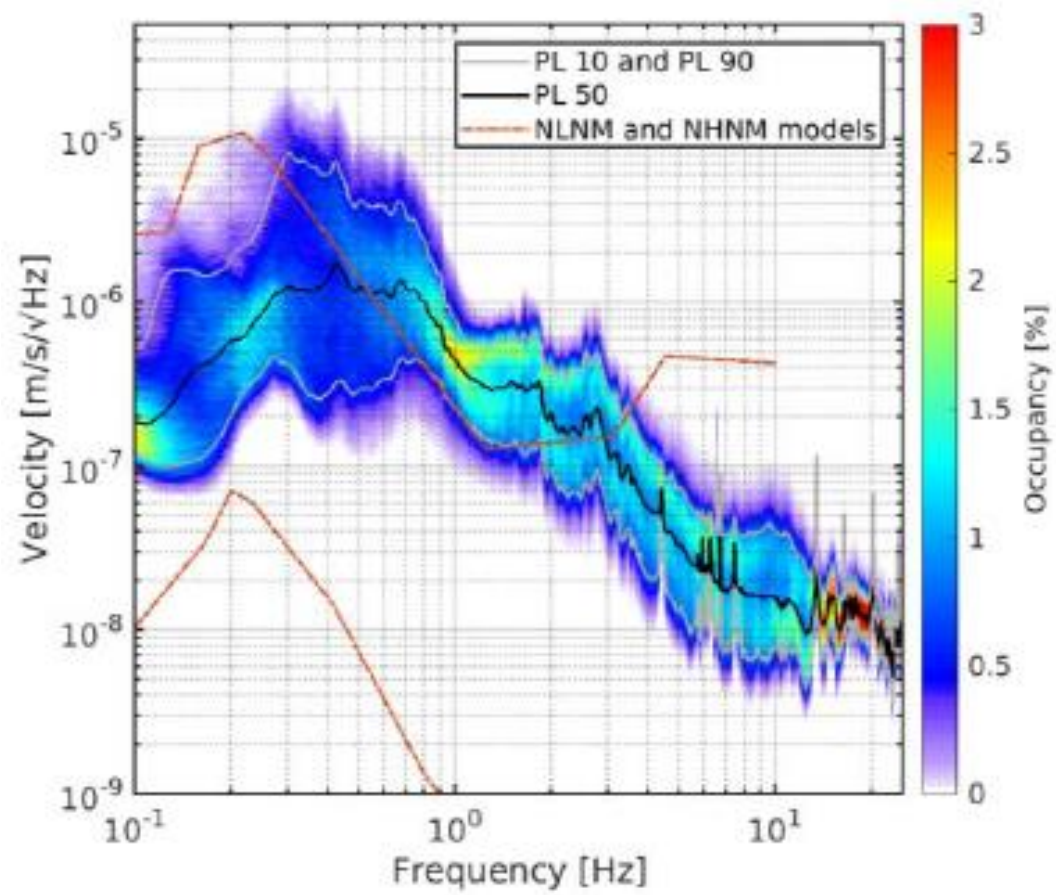
Optical Layout

- VIRGO interferometer
- The two arms are Fabry Perot cavities 3km long
- Makes use of Power Recycling technique to increase the total circulating power
- Signal Recycling was recently introduced, before the start of running data taking (O4b)

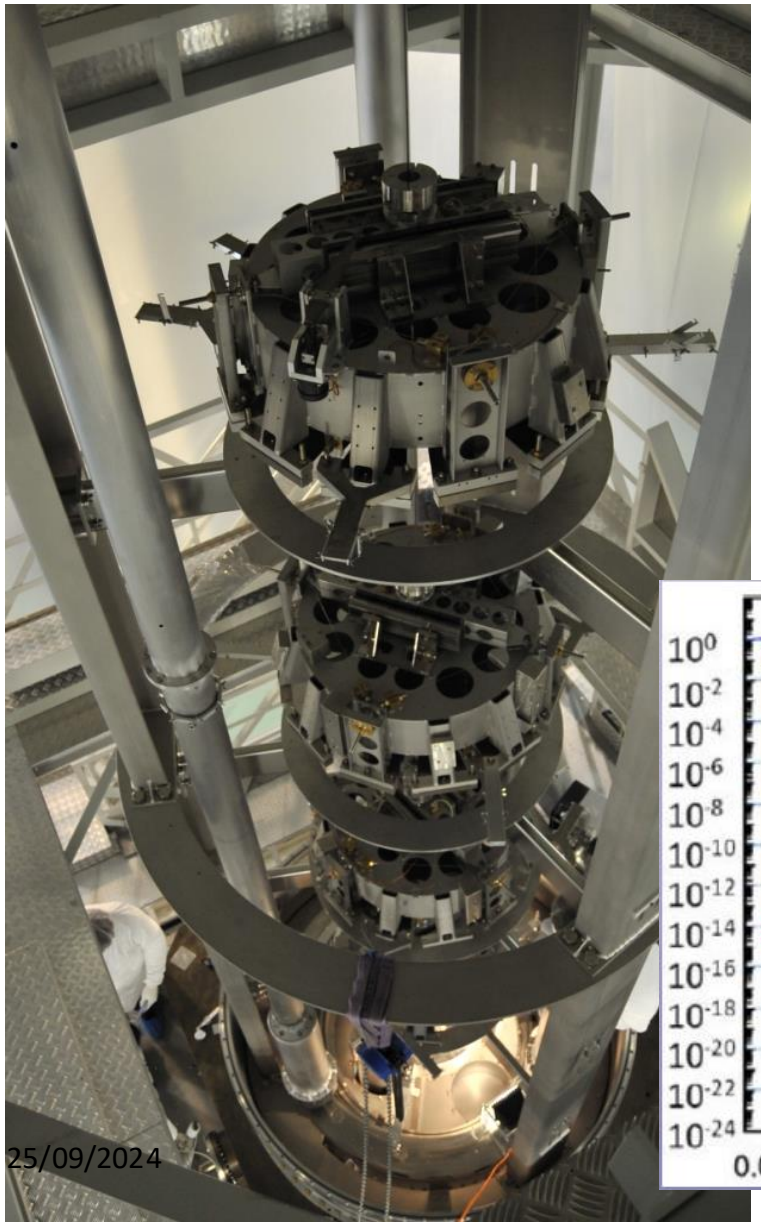


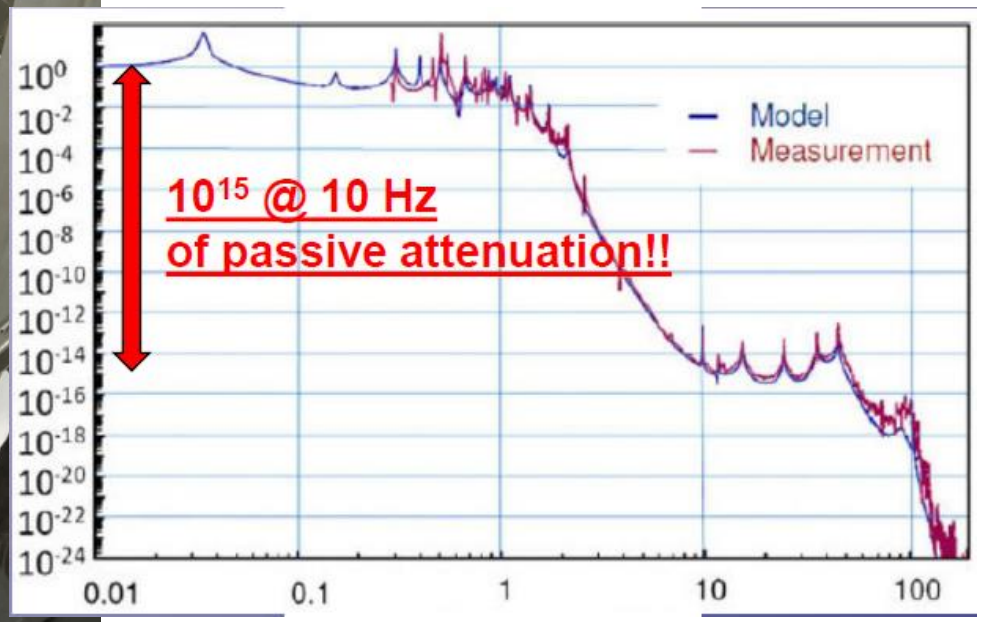
Seismic Noise

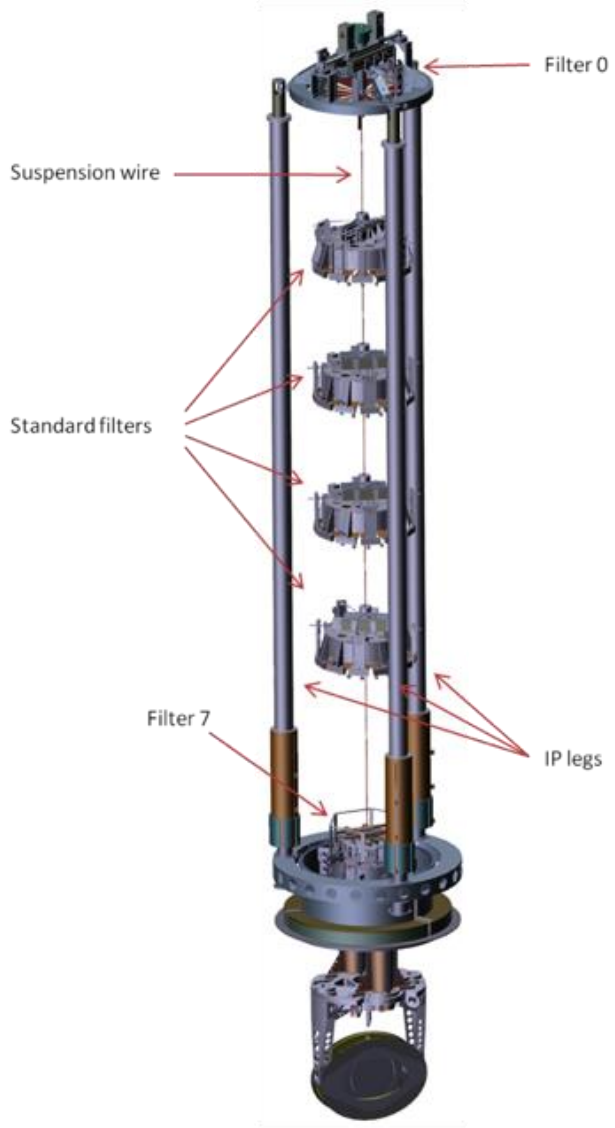
- Ground displacement is the largest noise source.
- Different sources:
  - Tidal strain (about 200 µm peak-to-peak between input and end mirrors)
  - Earthquakes
  - Seismic noise
    - Microseism: dominant between 0.1 and 1 Hz mainly due to sea waves
    - Anthropogenic:
      - between 1 and 5-10 Hz, due to heavy vehicles traveling along near roads and bridges
      - Between 10 and 40 Hz, due to onsite ‘traffic’.



Seismic Isolation



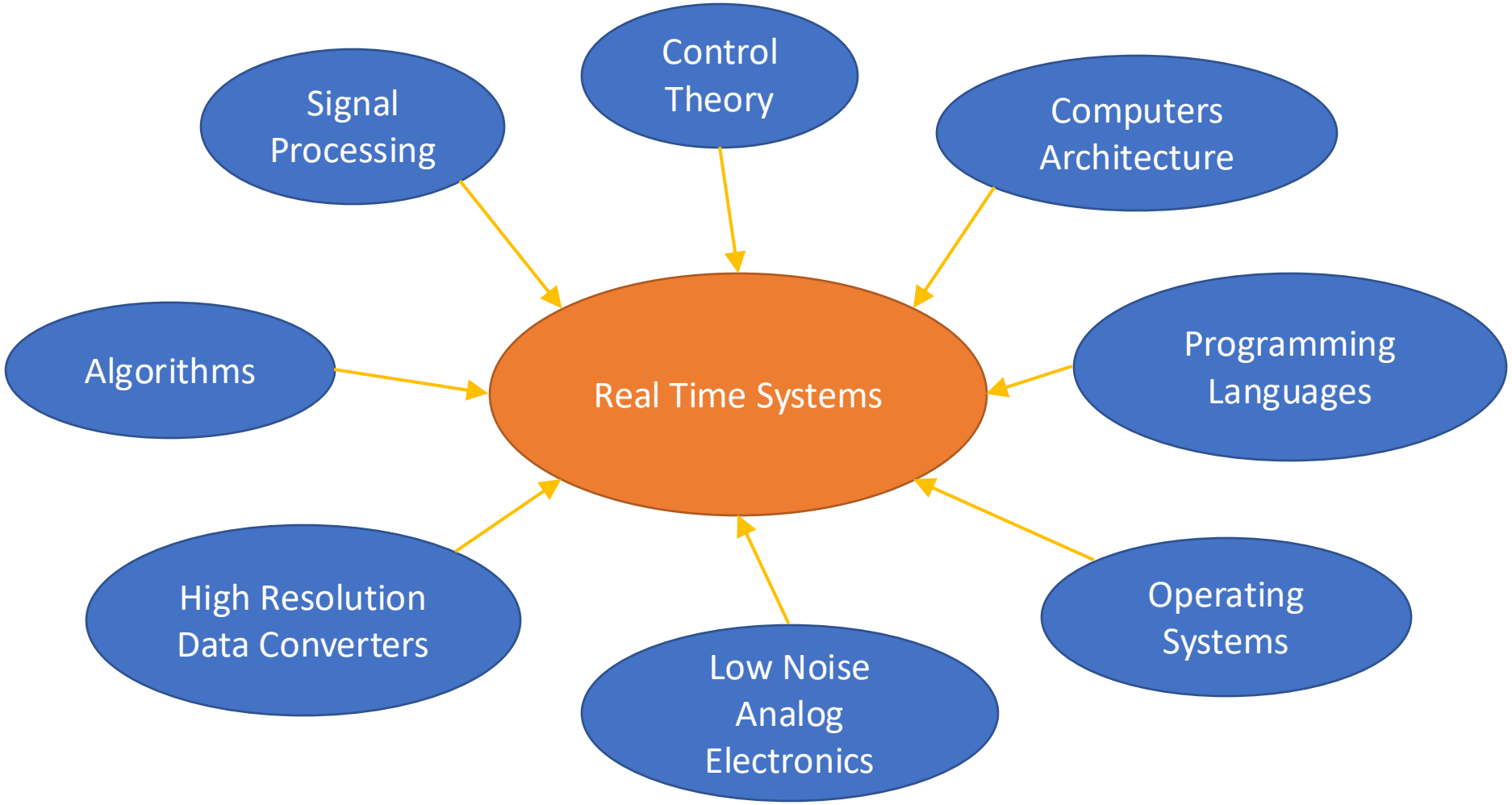




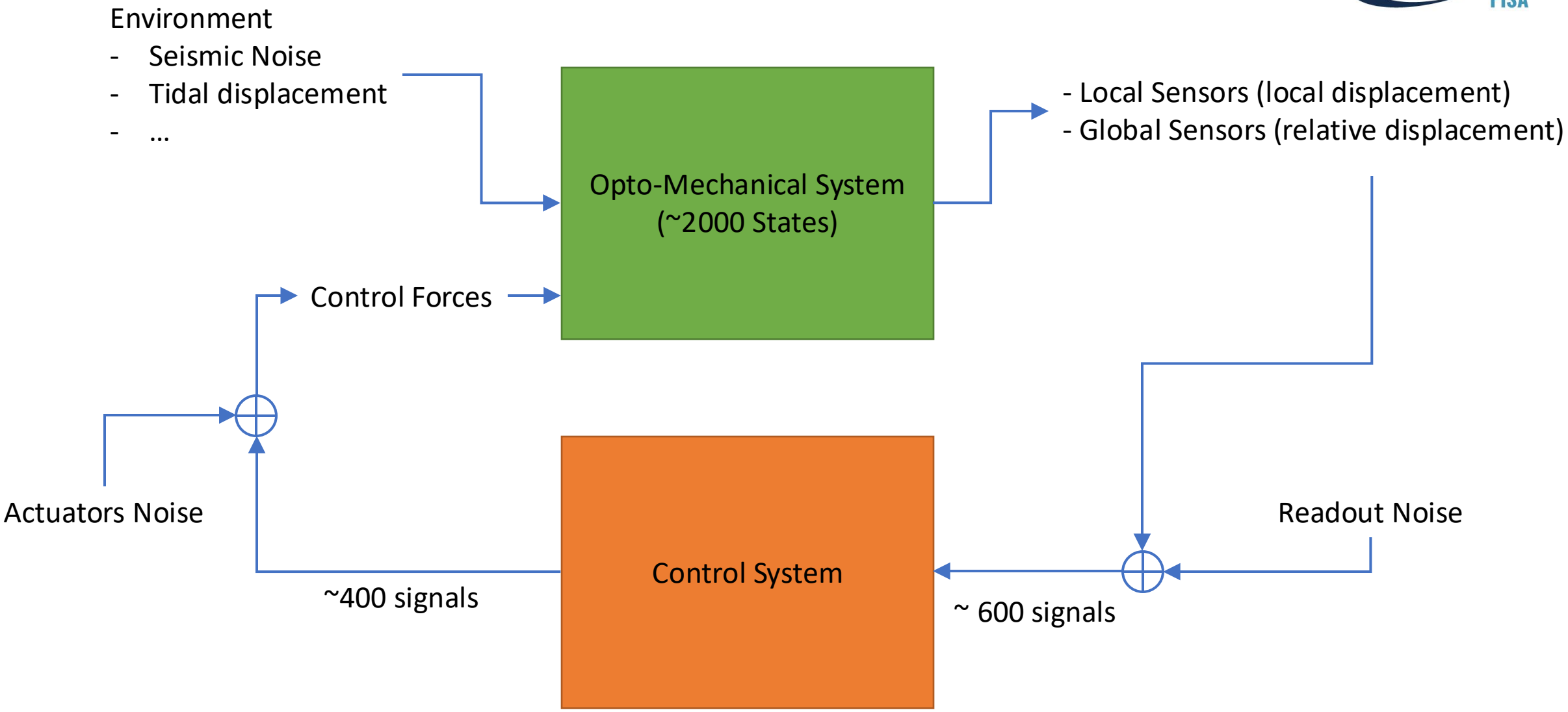
Control System

- All the elements of the interferometer are suspended from chains of pendulums. The spectral displacement above the frequencies of the main modes of the chains is effectively reduced.
- The displacement at the resonance frequencies is amplified, and since we want systems with low thermal noise, the quality factors are always very high. Moreover, the relative displacement between the various elements must be kept within the dynamic range of the readout system.
- To operate the interferometer, it is therefore necessary to use a complex control system that aims to reduce the residual displacement of the mirrors at the system's resonant modes and ensure precise control of the mirrors' position.

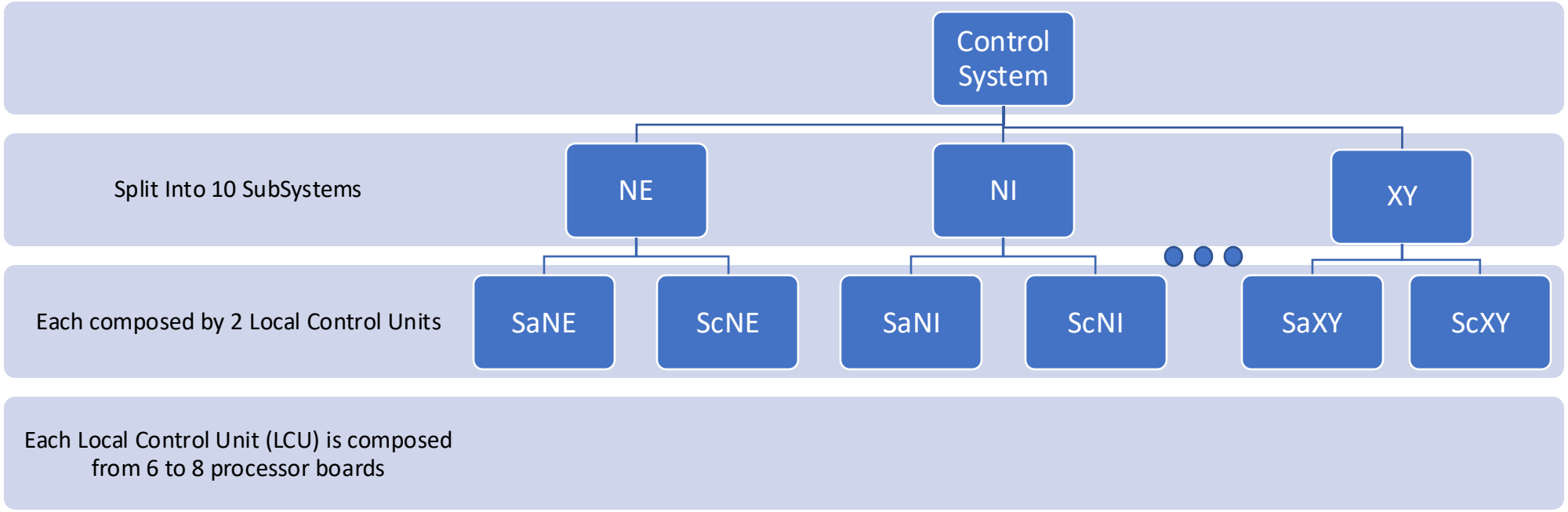
Disciplines Impacting Hard Real Time Systems Design



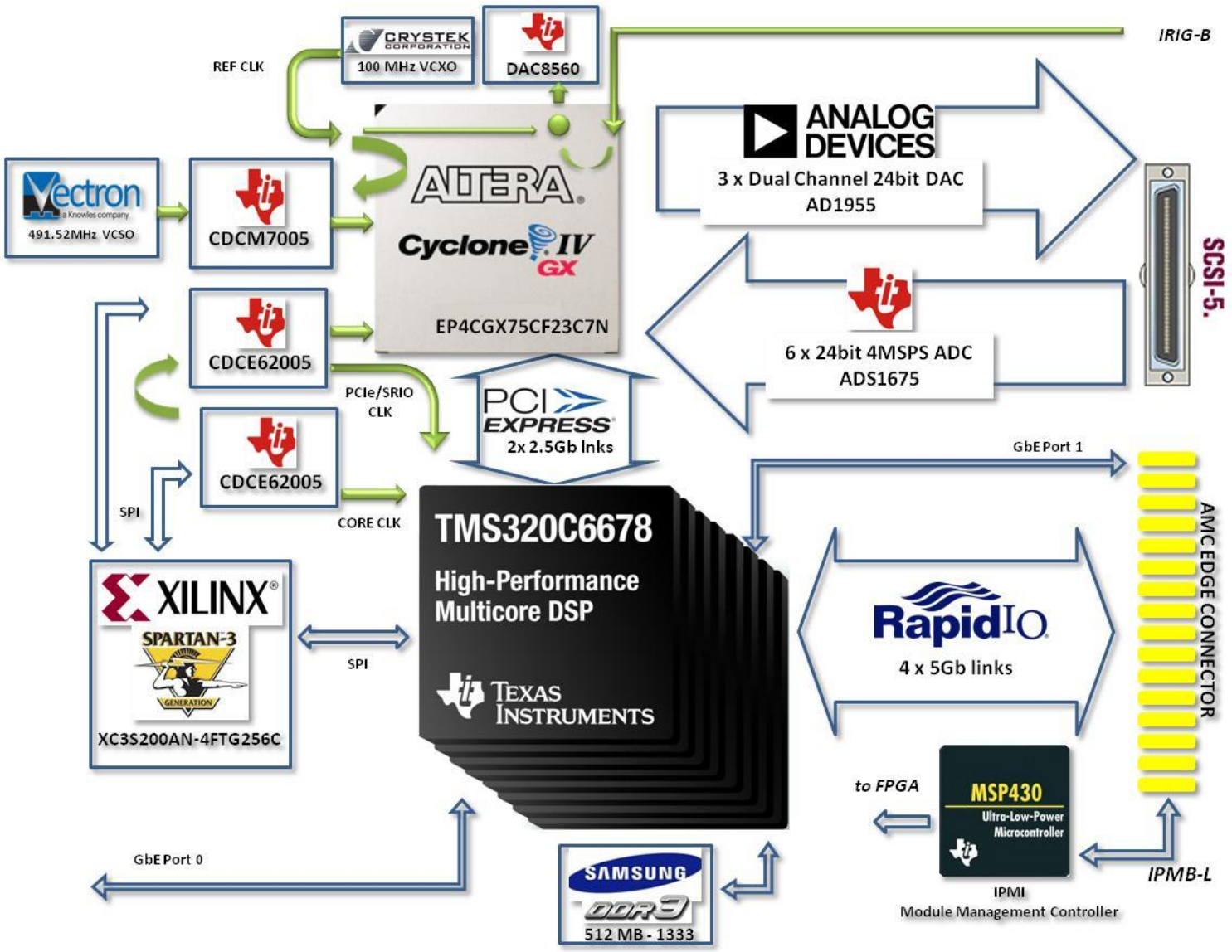
The Global Picture



Control System Breakdown



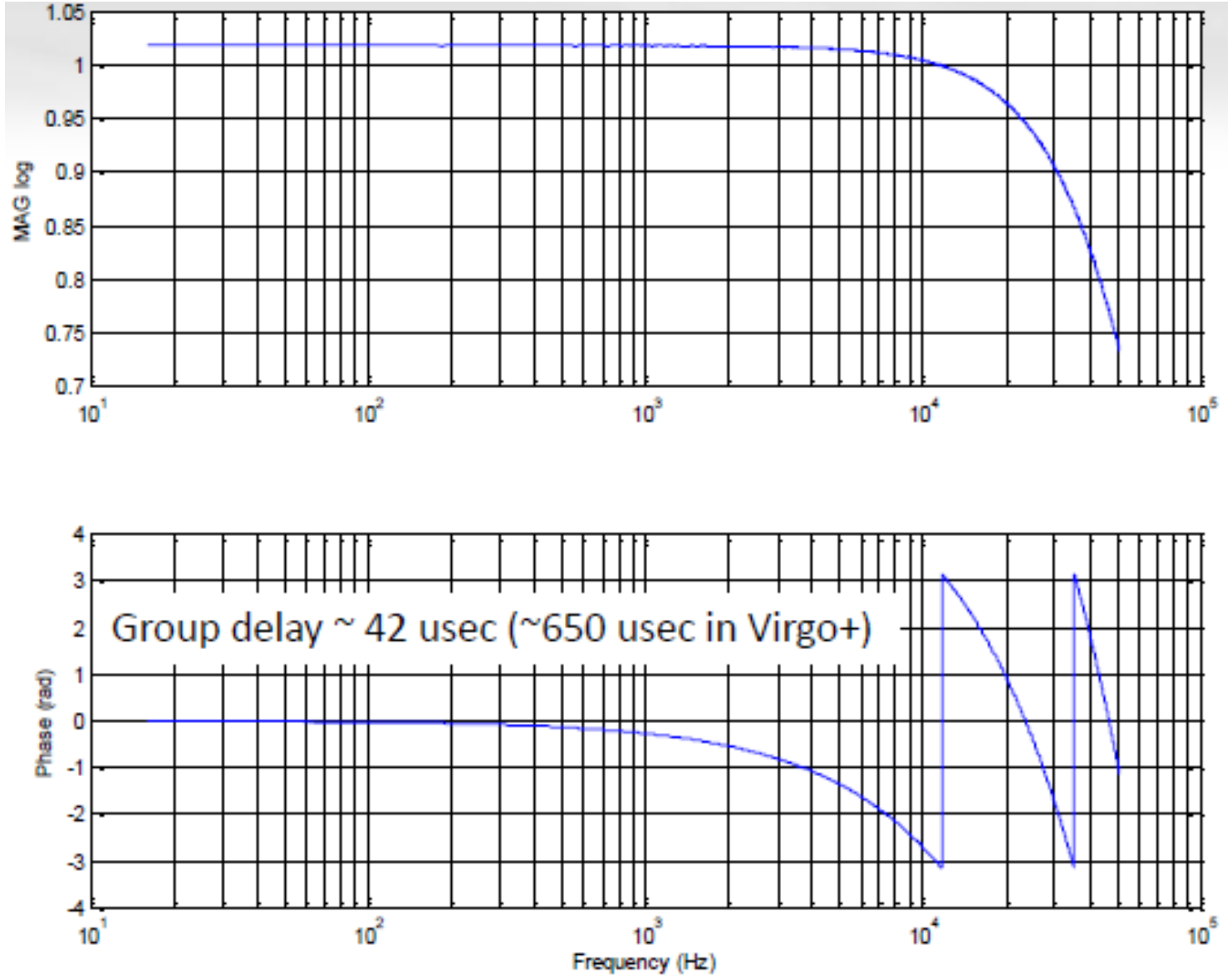
UDSPT Board



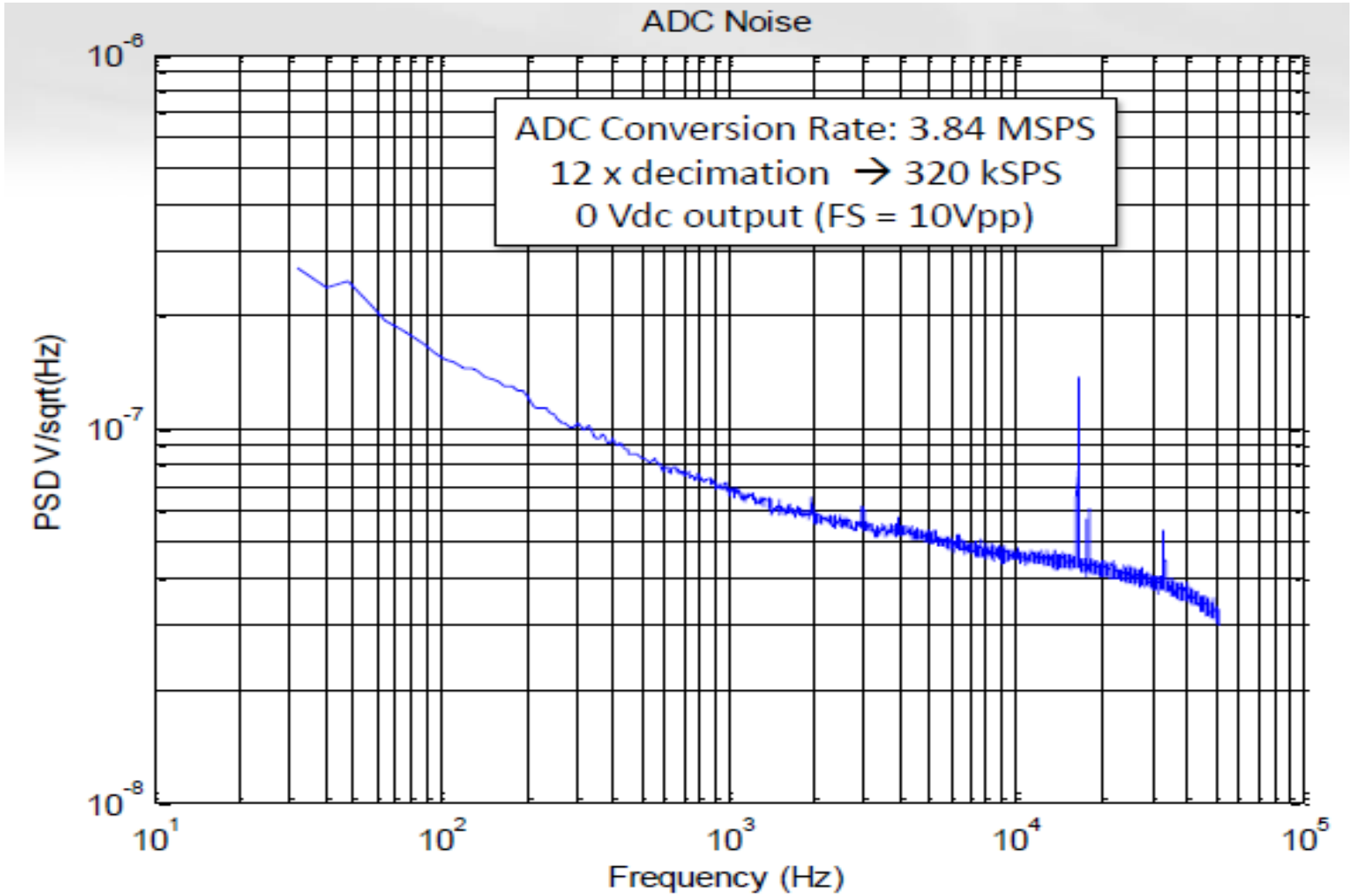
- 8 core DSP TMS3206678
- 6ch ADC (24bit, 3.84MSPS)
- 6ch DAC (24bit 320 kSPS)
- Analog Front End electronics
- Timing and DAQ interfaces

ADC-DSP-DAC Transfer Function

- The total time delay introduced by our control system can be measured evaluating the transfer function. We inject white noise into one ADC channel and we propagate the signal to a DAC channel.

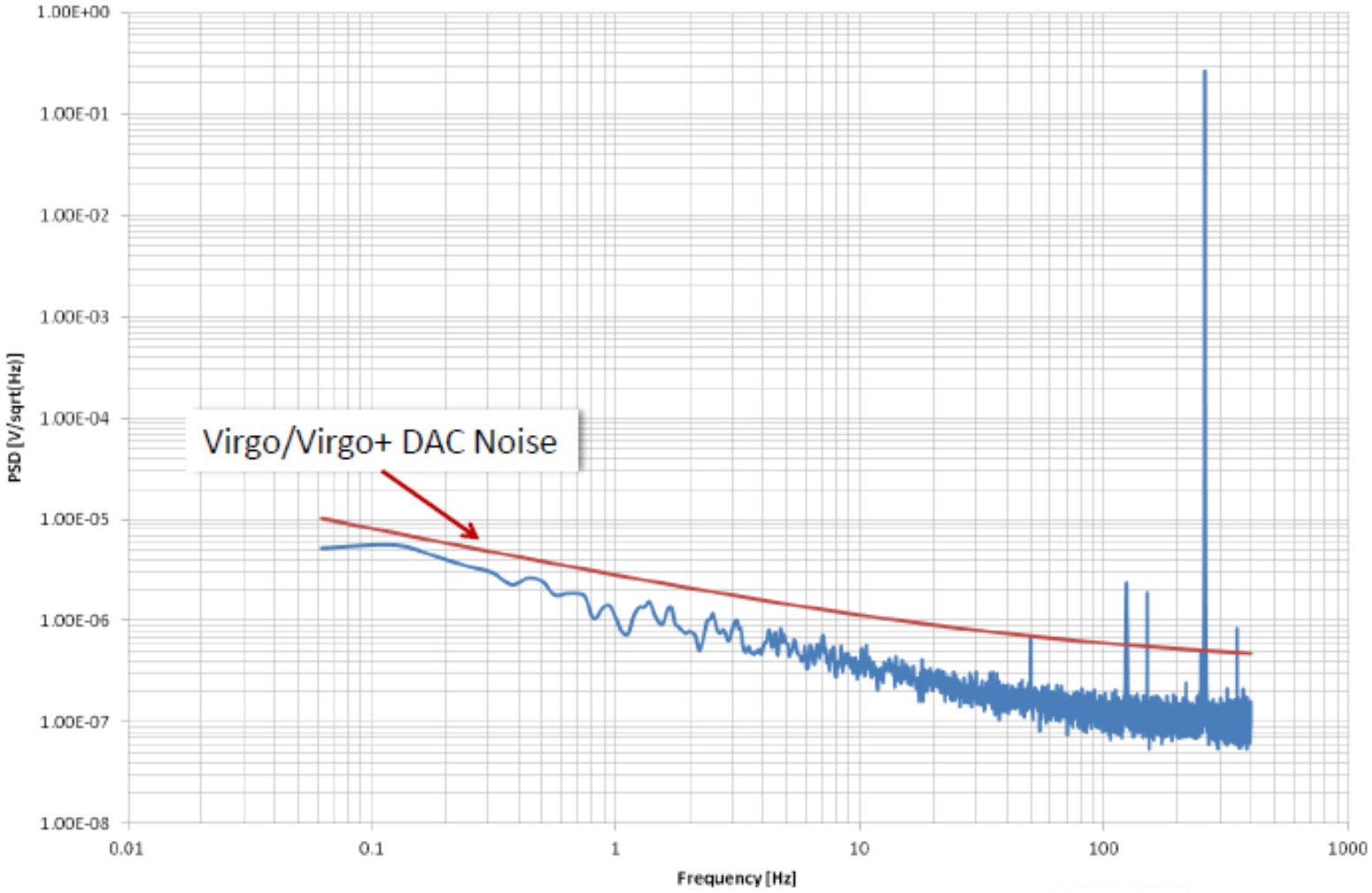


VIRGO Pisa - ADC Noise



- Most of local sensors have a 50 kHz modulated output to minimize flicker noise contribution.

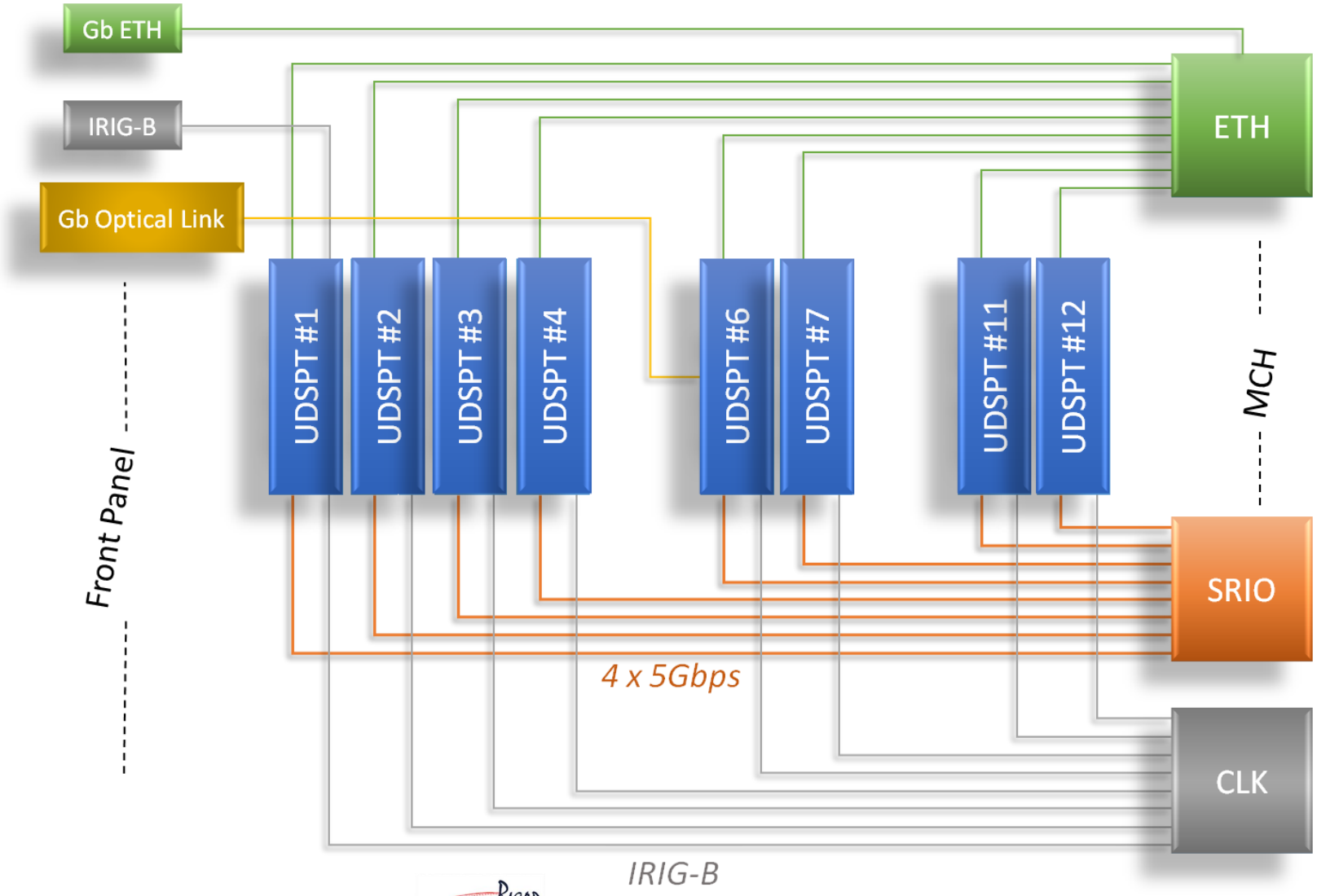
VIRGO Pisa - DAC Noise



- Only DC actuation → need to deal with flicker noise

Digital Signals Path

- Boards in each LCU are connected via point-to-point 20 Gbps links
- Boards can communicate with other LCUs via dedicated point-to-point optical link (1 Gb) and with external world via Gb Ethernet
- Boards synchronization is provided by connection to VIRGO timing distribution



Deployment

- 20 Crates equipped with a total number of 134 boards are in operation since 2015. Data exchange rate between boards is more that 4 Gbyte/sec (corresponding to about 8’000 simultaneous HD movies). Total number of DSP instructions is about 300’000’000’000 per second

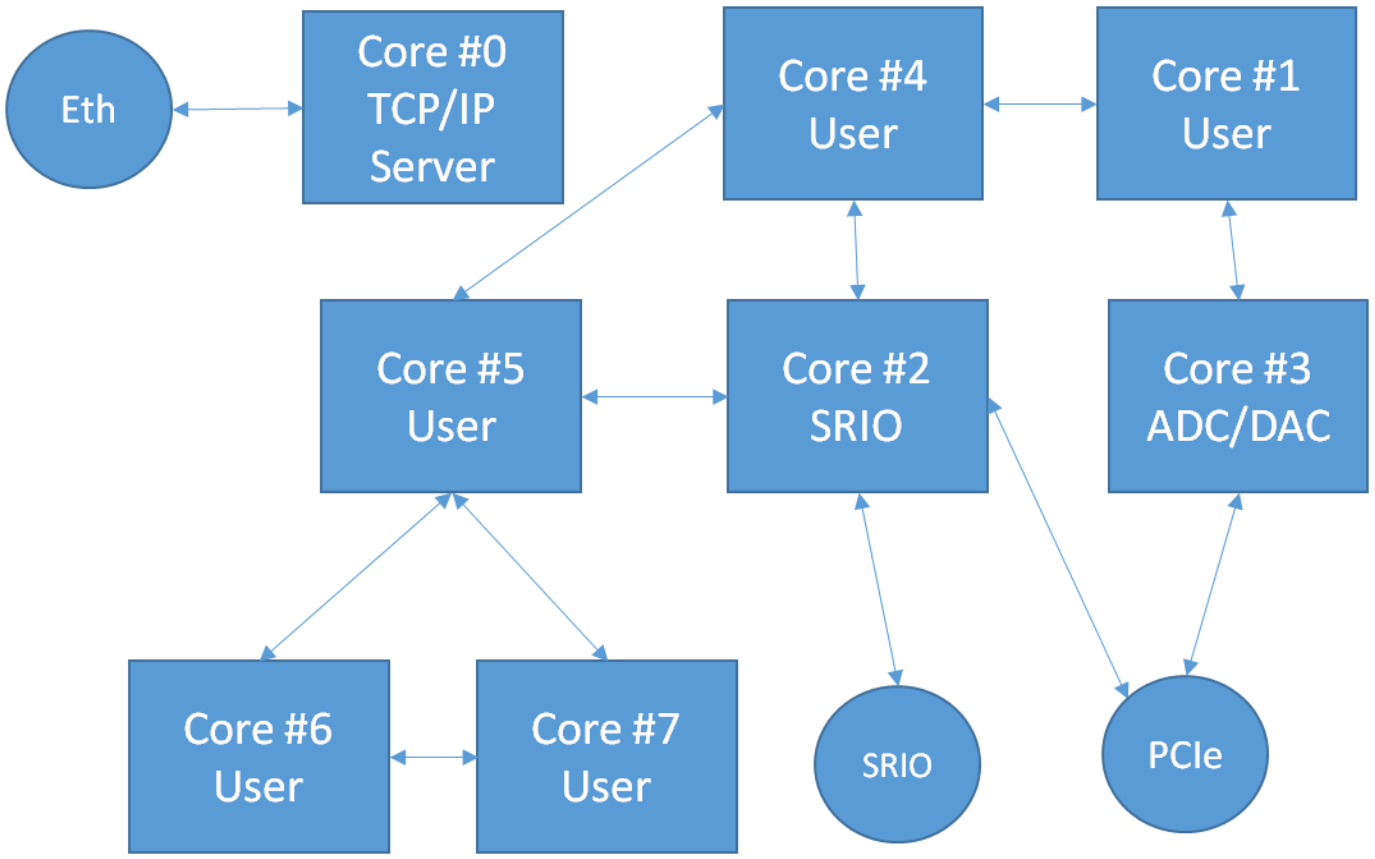
ScNE LCU Tasks List

Task Name	*Activation Freq (kHz)*	*Max exec time (ns)*	*Max Exec Time %*	*System Poles*	*MFLOPS (DP)*	*MFLOPS ADC decim.*	*MFLOPS Probes decim.*	
NE_LN	40	3181	13%	29	19.96	23.04	0	
NE_Mar_LN	40	5219	21%	85	40.52	23.04	0	
NE_PSDf	40	3699	15%	0	15.6	23.04	80.64	
NE_PSDi	40	3313	13%	136	22.72	23.04	0	
NE_PSDm	40	637	3%	0	2.76	23.04	0	
NE_PSDt	40	608	2%	0	2.68	23.04	0	
NE_F7_CD	40	1929	8%	21	11	23.04	0	
NE_F7_LVDT	40	699	3%	0	1.88	23.04	0	
NE_F7_LVDT_Demod	320	1865	60%	66	134.4	0	0	**Total MFLOPS**
					251.52	***184.32***	***80.64***	***516.48***

- 9 user tasks are running. Activation frequency is 40 kHz for all of them except for the last one activated 320’000 times per second. Execution is parallel: each tusk runs on a dedicated DSP core.
- 24 system task are running simultaneously to user tasks (ADC decimation, DAC inputs interpolation, Data Acquisition signals pre-processing, communication, …)

DSP Cores

- Each DSP core, excluding C#0, runs a single task activated at a fixed frequency lower or equal to 320 kHz
- Communication is memory mapped using a 2 stages FIFO (ping-pong table)
- 5 out of the 8 cores available are user programmable



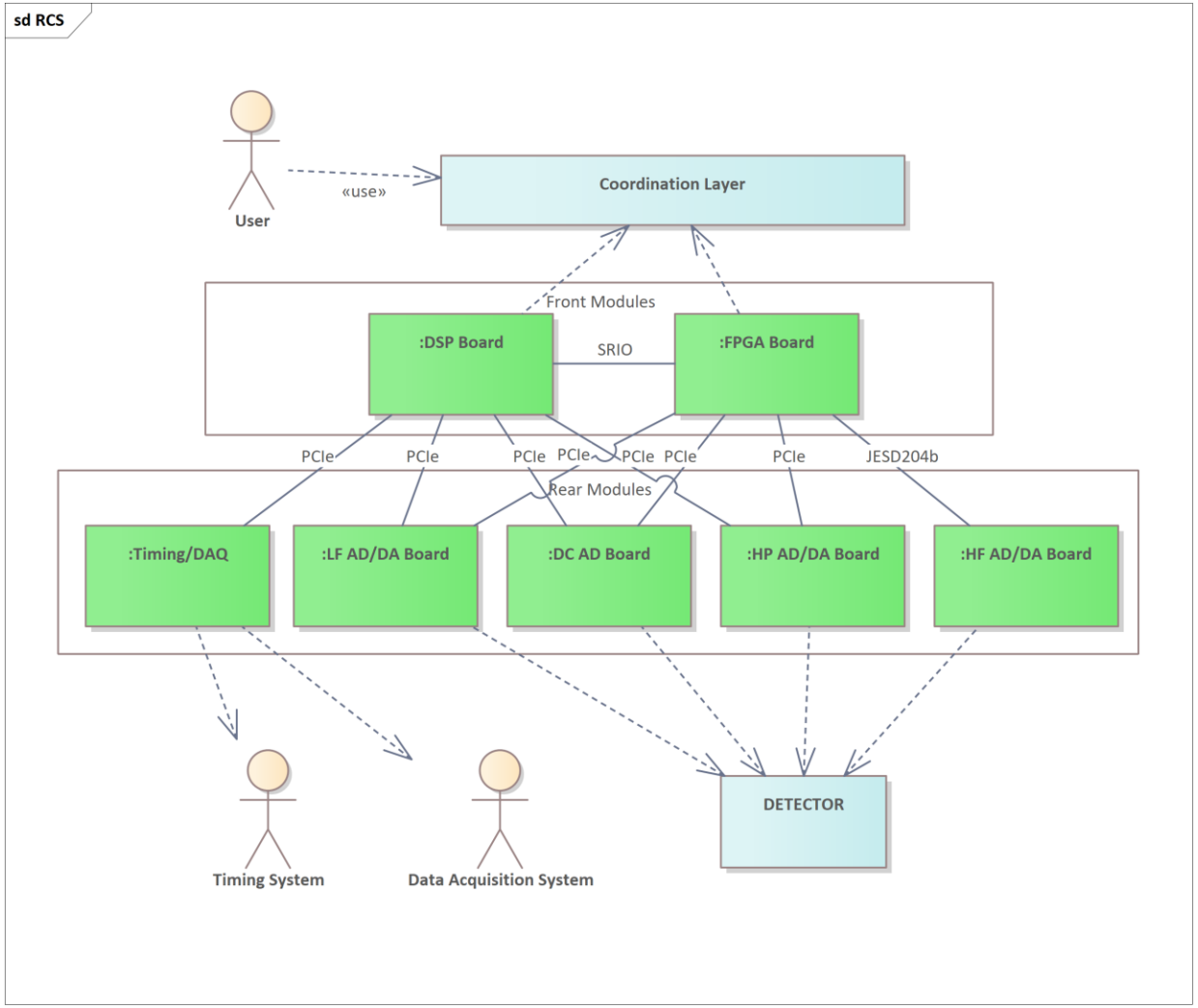
DSP Code

```
#include <asm/header.net>
NETLIST {

  START {
    SAMPLING_FREQ = 10000;
    OVER_SAMPLING = 1;
    USE_ADC       = 1;
    DOWN_SAMPLING = 1;
    W_COEFF = internal;
    SOURCE = parent;
    DOL_ADDR = f94c0000;
    GRD_ADDR = Guard4_to_Core0;
    DICT_NAME = Sc_NI;
    KEEP_VALUES = no;
    DECAY_TIME  = 0x00000000;

    TOLM_OUT_N = 1;
    TOLM_OUT = {
      {
        PAGE = 0x00000000;
        NAME = PRB;
        ROUTING = 0xf94c0000;
        TOLM = TOLM1;
      }
    }
  }

  ASM_START {}

  DAC_SETUP {
    NDAC = 6;
    OUT = ($DAC_0xf8, $DAC_0xf9, $DAC_0xfa, $DAC_0xfb,
           $DAC_0xfc, $DAC_0xfd);
  }

  ASM_CONTINUE {}

  CALL {
    NAME   = NI_F7_LVDT_Demod.hrd;
    COMM   = " ";
    NINP   = 0;
    NOUT   = 9;
    OUT    = (lvdt1, lvdt2, lvdt3, vlvdt1, vlvdt2, vlvdt3, ratio_l1, ratio_l2,
              ratio_l3);
  }
```

```
    .tab 6
DAQ_header1   .equ  00055045h
DAQ_header2   .equ  00055046h
DMA7_chain    .equ  000517e8h
tolm_in_dpm_buffer  .equ  14831a00h
tolm_in_ser_buffer  .equ  14831c00h
tolm_in_glb_buffer  .equ  14831800h
tolm1_out_buffer    .equ  00058300h
tolm2_out_buffer    .equ  00058700h
Guard1_to_Core0 .equ  1087f000h
Guard4_to_Core0 .equ  1483c400h
        .sect       .textasm         ; program section

SP      .set        b15

        .global dsp_irq
        .global i_list
        .global c_list
        .global w_list
        .global noise_table
        .global cosine_table
        .global input_list
        .global output_list
        .global Guard4_to_Core0
dsp_irq:
        .asmfunc

        mvc     CSR, B2                     ; save old interrupt status
||      add     -8,SP,SP
        DINT                                ; disable interrrupts
        stdw        A11:A10, *SP--[1]       ; a10 = call_in
        stdw        A13:A12, *SP--[1]       ; a12 = tolm_out_address
        stdw        A15:A14, *SP--[1]
        stdw        B11:B10, *SP--[1]       ; b10 = call_out
        stdw        B13:B12, *SP--[1]       ; b12 = tolm_in_address
        stdw        B9:B8, *SP--[1]         ; b8 = From_Slot buffer address
        stdw        B3:B2, *SP--[1]
        stw         A8, *SP--[1]            ; save To_Slot buffer address
        stw         B6, *SP--[1]            ; save signals buffer address
        stw         A6, *SP--[1]            ; save probe buffer address
        stw         B4, *SP--[1]            ; save DAC address
```

- From a source code written in a simple object-oriented language we generate asm code for the cross-compiler that produces DSP binary code

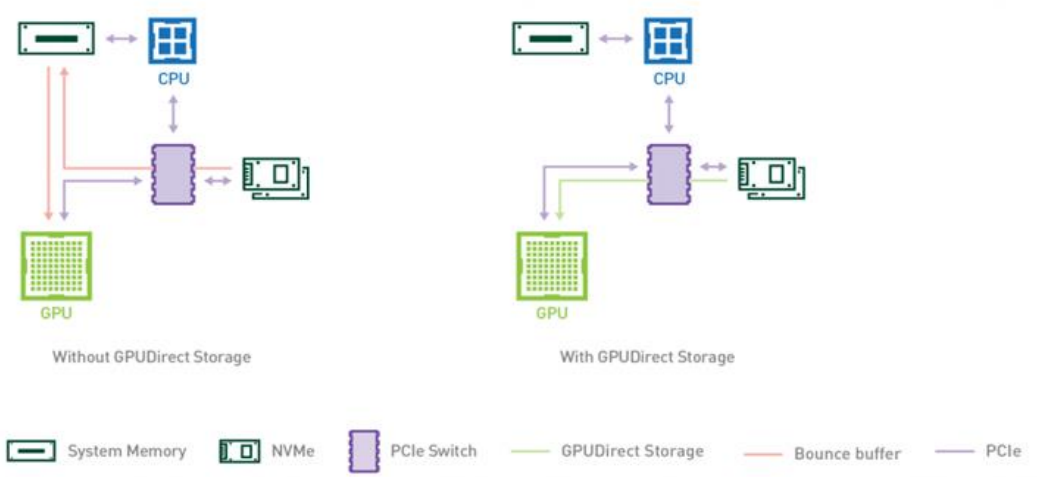
RCS – KATANE (AdV+ Phase II)



- MTCA.4
- DSP processor with additional 4 ARM cores-
- Board-to-board communication via RapidIO
- Processor-data converters communication via PCIe
- INFN Bologna and Perugia contribution

RCS - ZANCLE

- Future upgrades of our control system: Virgo_nEXT, Einstein Telescope



- We are starting from existing VIRGO system and evolve gradually towards future architecture where presumably A.I. will play a key role.
- DSP+FPGA architecture would evolve in GPU+FPGA architecture but we are still far in terms of Real-Time performances. Key focusing is in high resolution converters and in lower low-frequency noise in general.
- Whatever solution, software will play a major role.
- For initial ET a mixed approach could be the best choice.


THANK YOU