ET Data acquisition & real time control

Kick-off meeting
24/01/2024

Loïc Rolland

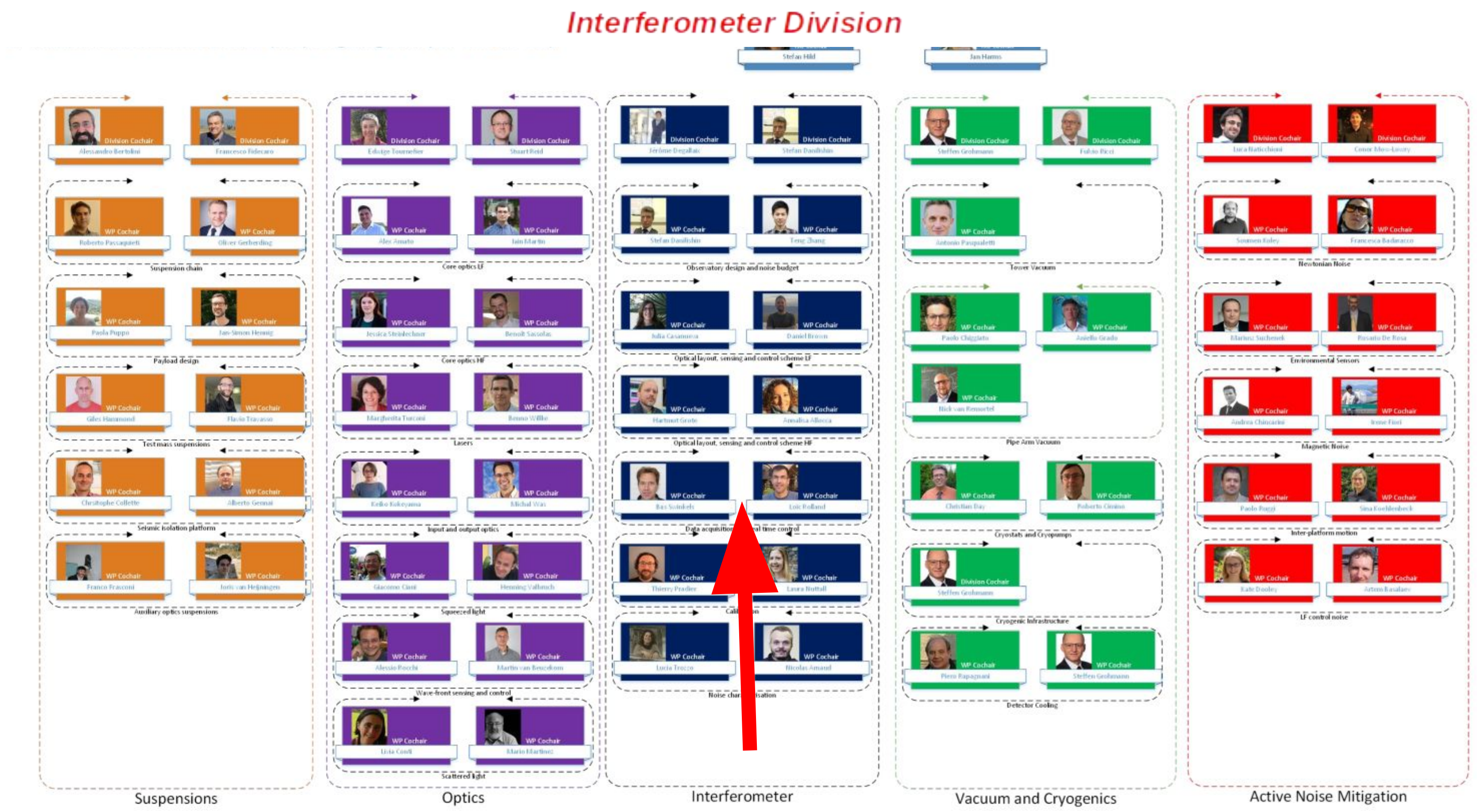
Bas Swinkels

ET-0020A-24

Overview

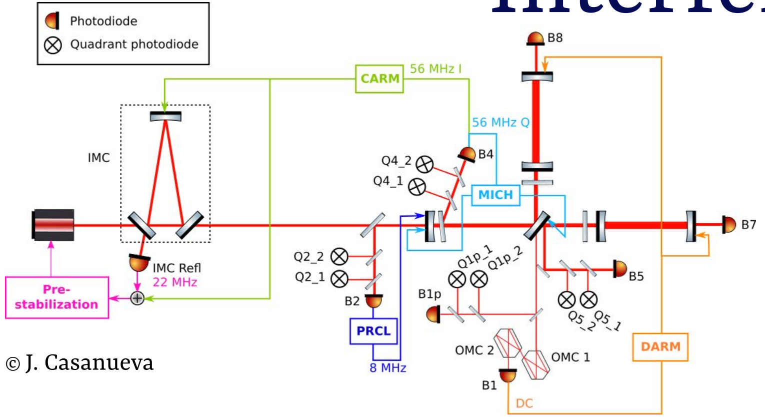
- Quick introduction of participants (time permitting)
- Quick overview: DAQ and control for GW detectors
- Organizational matters
- Drafting the requirements and Project Breakdown Structure
- Short summaries of work at some groups:
  - LAPP
  - Hamburg
  - Nikhef
  - Pisa
  - Geneva
- Discussion

Instrument Science Board / Interferometer Division Data acquisition and real-time control Work Package

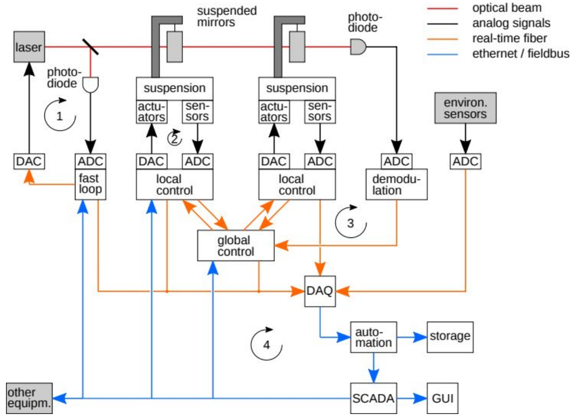


Data Acquisition & control
for
Gravitational Wave detectors

Interferometer control

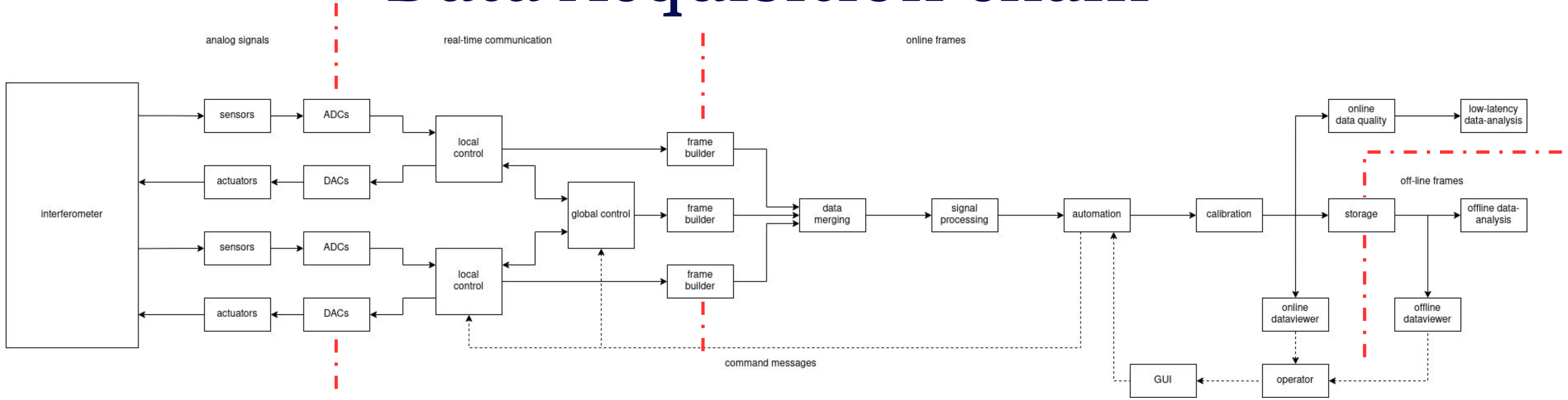


© J. Casanueva



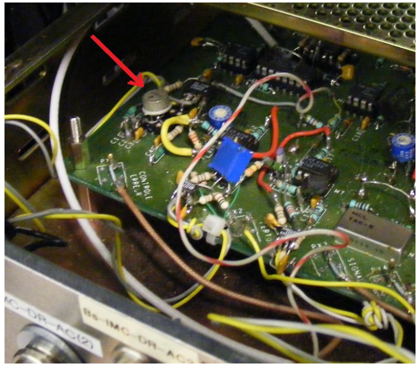
- Control system is a key part of a measuring GW with interferometers: keep mirrors quiet and cavities on resonance at picomoter level
- Various levels of control loops:
    1) very fast analog/digital loops (~MHz)
    2) fast local control of suspensions (~10 kHz)
    3) fast global control of whole interferometer (~10 kHz) — hard real time
    4) slow automation: lock acquisition (~1 Hz) — soft real time
    5) 'human-in-the-loop' monitoring and operation (minutes)
- Hard real time, distributed, hierarchical control. Sample frequency/delays limit obtainable control bandwidth
- All signals from control system and environmental monitoring recorded by data-acquisition (DAQ) chain

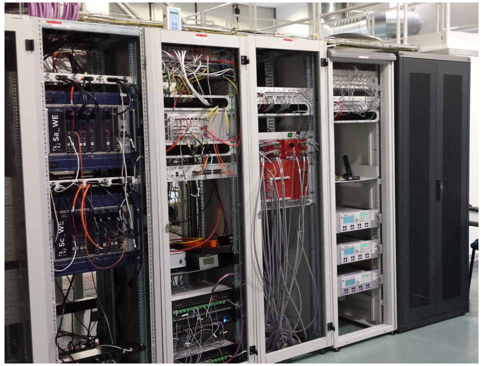
Data Acquisition chain



- purpose: provide inputs to calibrated GW strain, provide auxiliary channels for validating/vetoing claims of GW detections, debugging/fine tuning the experiment
- analog signals of sensors/actuators converted from/to digital, sent around for real-time control
- fast signals collected by ‘*frame builders*’ into in chunks of data (0.1 to 1.0 sec), merged with slow data
- optional signal processing (decimation, image processing, ...)
- slow automation nodes that change parameters of fast processes based on data in frames
- data viewers, GUIs for human interaction with the interferometer
- calibrated strain signal sent to online and offline data analysis
- data flux relatively modest compared to e.g. CERN (Virgo stores 55 MB/s on disk)

Control hardware



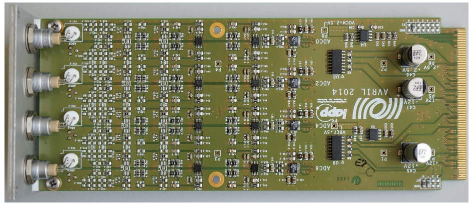


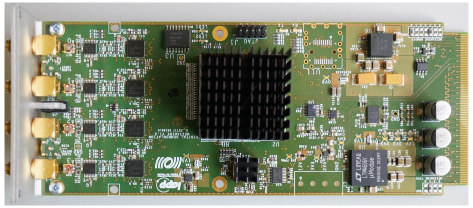
- Historic progression of control electronics:
  analog -> barely working custom digital -> comfortable custom digital -> (mostly) off-the-shelf?
- Key components:
  - analog <--> digital conversion
  - computing element: FPGA/DSP/CPU, can be part of mezzanine, crate or remote PC
  - real-time communication
  - timing
  - software (real-time code, DAQ chain, supervisory control, ...)
- Different experiments/groups came to quite different solutions
- Opportunity to start from scratch for Einstein Telescope: learn from past experience, use best modern solutions

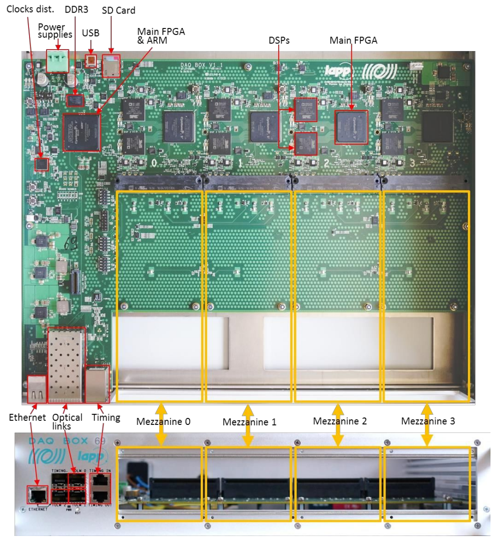
DAQ crates

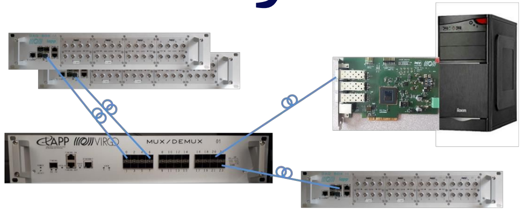
- Located in electronics racks, connected to the experimental hardware
- Virgo does everything propriety:
  - DAQboxes from LAPP Annecy for photodiodes, bench suspensions
  - Pisa DSPs crates for local control of big mirror suspensions
- LIGO uses mostly commercial solutions:
  - PCI expansion crates with commercial ADCs/DACs from General Standards
  - EtherCAT hardware from Beckhoff for slower controls (up to ~ 1kHz?)
- Other possibilities:
  - MicroTCA
  - Commercial: NI/Dspace/SpeedGoat/... (specs usually not good enough)
- My ideal:
  - commercially available open standard crate (no vendor/group lock-in)
  - commercial ADC/DAC/IO boards for non-critical signals
  - custom low noise ADC/DAC/FPGA boards for the critical photodiodes/mirror actuators.
  - possibly integrate ADC/DAC directly into analog front-ends of some sensors/actuators?
- Important decision seems where to divide the interface between crate and (one or more) mezzanines: between ADC and FPGA, between FPGA and DSP, between DSP and CPU?

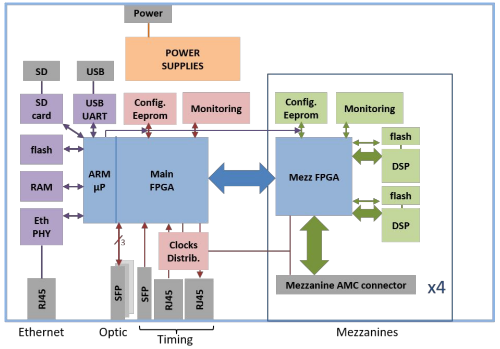
DAQboxes from LAPP Annecy





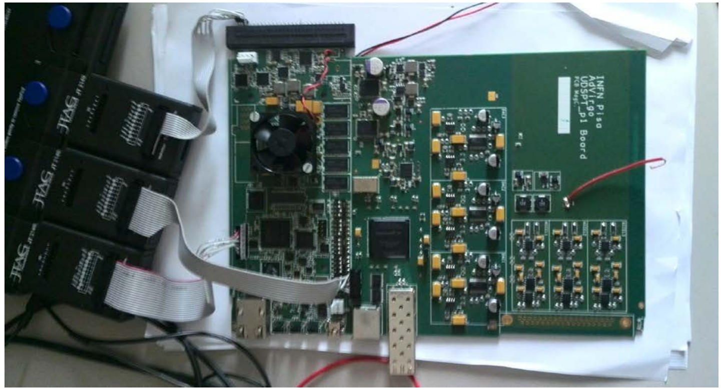


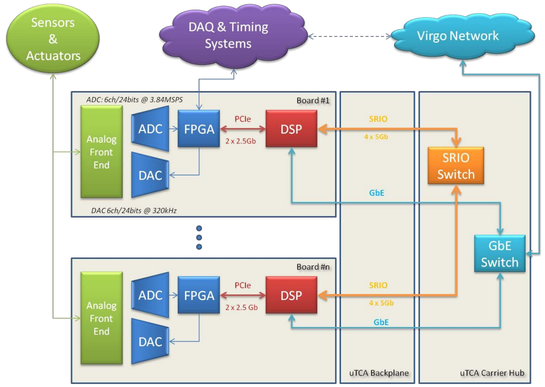




- Custom DAQbox with slots for 4 mezzanines, on-board DSPs for decimation, FPGAs, TOLM connection
- Standard mezzanines are relatively simple, just analog front-end and ADCs/DAC chip
- Digital demodulation mezzanines have big on-board FPGA
- Real-time PCs in remote computing room (fast PC with real-time Linux)
- Was relatively cheap: about 250 € for simple ADC/DAC channel (before pandemic, excluding personpower!)
- Needs redesign if we want more, some components obsolete
- See VIR-0750A-19 for details

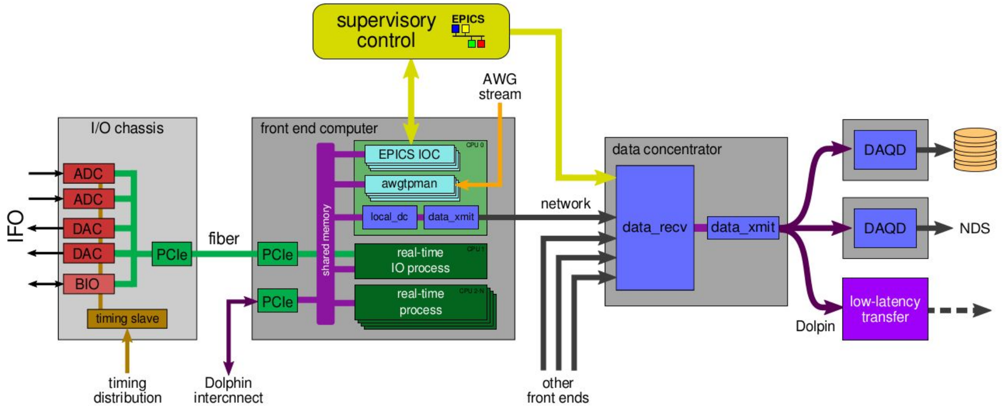
Pisa DSPs

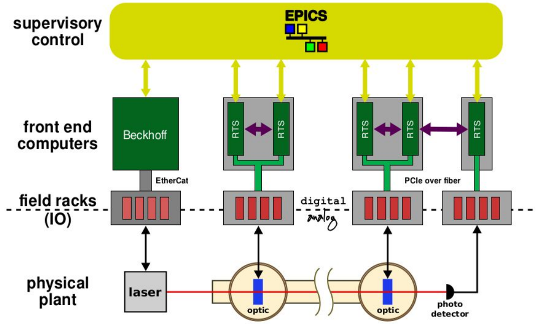




- Single board with pretty powerful multi-core DSP, FPGA, 6x ADC, 6x DAC, analog front-ends
- microTCA physical crate, rapid-IO communication to other boards via backplane, MCH
- TOLM for real-time communication with LAPP RTPC
- ~720 GFLOPs per crate. Loops run at 10 kHz, except digital demodulation of LVDTs at 100 kHz
- Crates located in rack right next to the suspensions
- See LIGO-G1501131 for more details

LIGO solution





- all real-time computations done with off-the-shelf real-time PCs located in remote computing room
- ~20 meter fiber to PCI extension crates, which contain commercial ADCs/DACs (General Standards) total costs per channel ~2000$
- Analog AA filtering/whitening in separate home-built 19-inch boxes
- Dolphin for real-time communication with remote PCs
- ~20-50 meter long analog cables to the suspension towers
- See arXiv:2005.0021 for more details

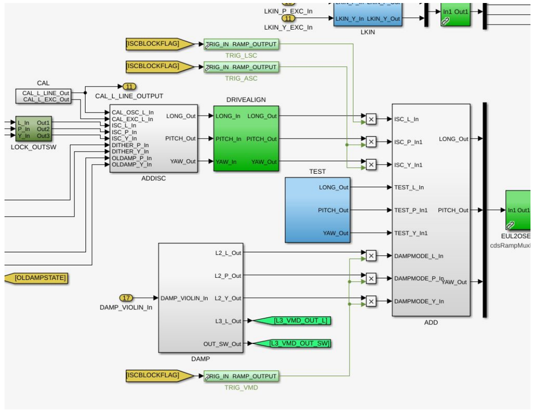
Real time computing elements

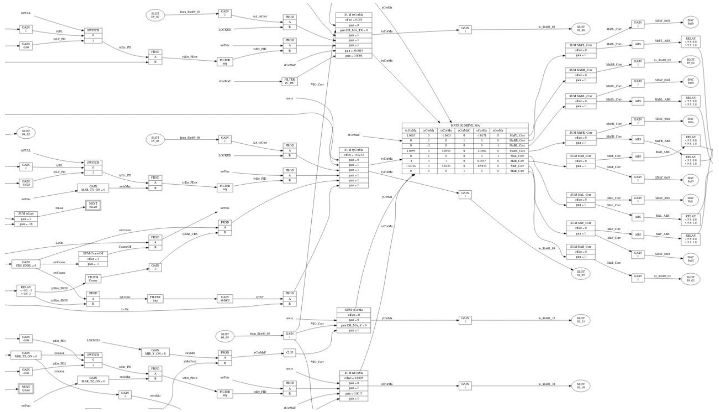
- FPGA for fastest computations, main use at Virgo is digital demodulation of RF photodiode signals
  - LAPP demodulation mezzanines: 400 MHz decimated down to 400 kHz
  - Nikhef phase camera: 500 MHz decimated down to 16 kHz
- DSPs:
  - at Virgo used for control of main suspensions (20 kHz normal loops, ~100 kHz for LVDT demod)
  - used in DAQbox for decimation (e.g. 1MHz down to 10 kHz)
  - handful of fast loops: laser frequency stabilization at 500 kHz (hand-coded assembler)
- Real-time PCs:
  - off-the-shelf fast Linux server with real-time kernel patches
  - control algorithms with arbitrary complexity
  - fast control loops need to run on single core!
  - typically runs at 10 kHz (Virgo) or 64 kHz (LIGO?). Probably cannot be increased a lot
  - noisy, so typically in remote server room

Real-time fiber communication

- Hard real time
- Global (~10 km) communication of sensor and actuator values at >= 10 kHz rate
- Amount of signals not very high:
  - ~20 photodiode signals from optical benches to global control
  - ~10 corrections from global control to each suspension (longitudinal, alignment, some flags)
- Virgo:
  - proprietary TOLM protocol: 2.5 Gbit SFPs (-> 10 Gb?), custom routers using FPGAs, Comes with PCIe card for in RTPCs, built into DAQbox. Sample rate ~10 kHz, but some special loops at 500 kHz
  - various ModBus/BACnet/... for interfacing with slow devices like air-conditioning
- LIGO uses commercial solution:
  - Reflective Memory from Dolphin for fast stuff, comes with PCIe cards, custom routers
  - EtherCAT hardware from Beckhoff for slower controls (< 1kHz?)
- Alternatives:
  - just use Ethernet with overkill of bandwidth (> 40 Gbit)? But can you guarantee latency?
  - whatever they use in High-Frequency Trading?

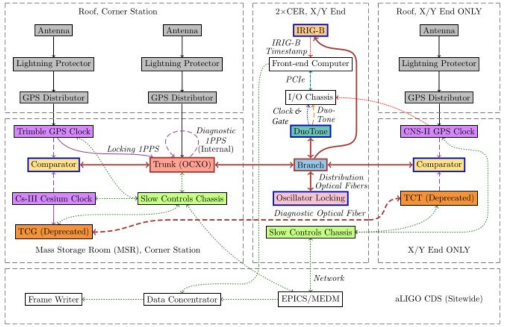
Real time control software

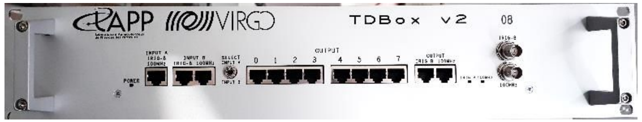


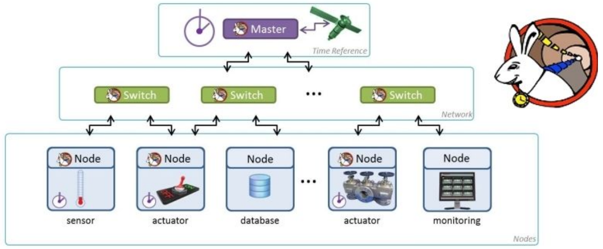


- Typically consists of two parts:
  - low-level code that talks to hardware
  - user algorithm written by commissioners in control room
- Virgo:
  - in the past (Gc by LAL): hand-coded algorithms in C, only few people could modify it
  - ACL (formerly known as Pr): text-based, used by many in control room
  - DSPs: command line interface to modify code, only used by few people
- LIGO
  - abuses Simulink as GUI only, custom compiler (BorkSpace?)
- Alternatives:
  - most commercial systems (dSPACE, SpeedGoat, ...) based on Simulink + matching compiler
  - LabView: uniformly hated?

Timing synchronization

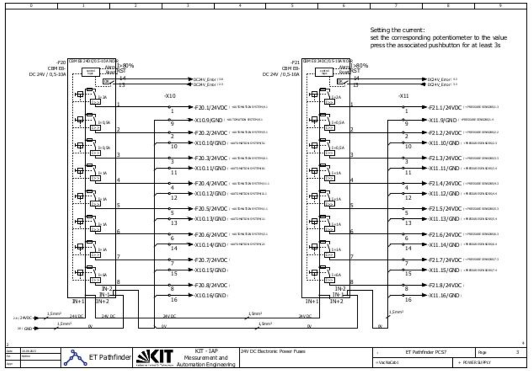


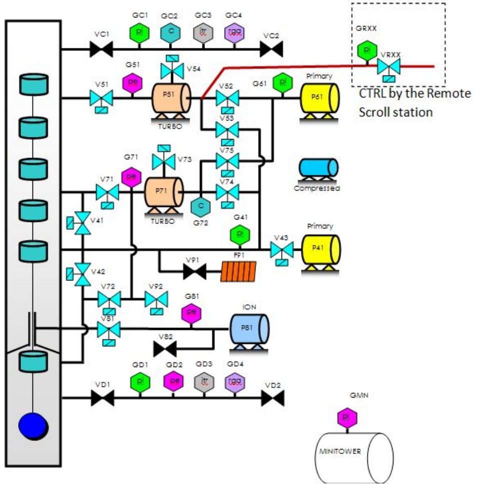




- Need global synchronization for astronomical observations: ~ 1 usec global accuracy
- Synchronization of real-time controls: <1 usec relative accuracy within whole experiment
- Fast sampling of RF signals: pretty hard requirements on timing jitter
- Virgo:
  - custom solution based on distributing IRIG-B/10 MHz/100 MHz signals via copper/fiber. obsolete ...
  - Nikhef is currently working on replacement based on White Rabbit
- LIGO uses custom built solution: DuoTone, see arXiv:2304.01188. No digital demodulation
- Sample rate: 2/5/10*10^n (Virgo, common 10 MHz equipment) vs 2^n (LIGO, easier for signal processing)

Safety critical stuff


Setting the current:
set the corresponding potentiometer to the value
press the associated pushbutton for at least 3s
ET Pathfinder
KIT
KIT - IAP Measurement and Automation Engineering
24V DC Electronic Power Fuses
ET Pathfinder PCS7

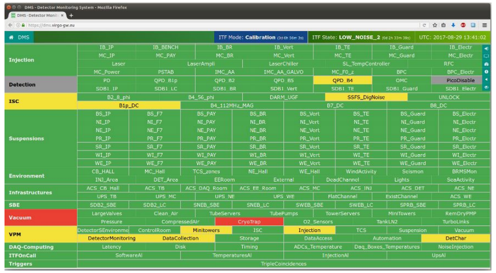


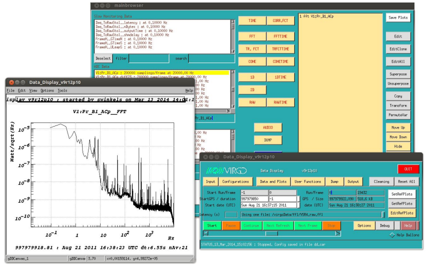
- Dedicated hardware for protecting safety/hardware:
  - vacuum system
  - laser interlocks
  - over-temperature protection
  - fast shutters to protect photodiode in case of unlock, ...
- Hardware: typically PLCs (Siematic, Beckhoff, NI, ...)
- Personal opinion: this should be responsibility of corresponding subsystems, not DAQ. Don't care what system they use as long as they provide read-only data (status, sensor values) to the DAQ chain. Project should probably enforce a single platform to allow experts to work on multiple subsystems
- Would be good to standardize on single vendor for whole ET

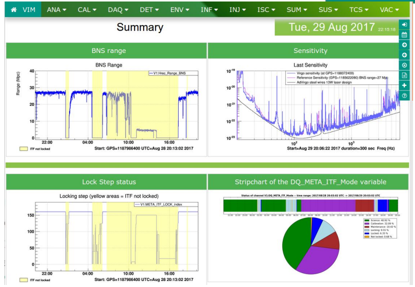
SCADA

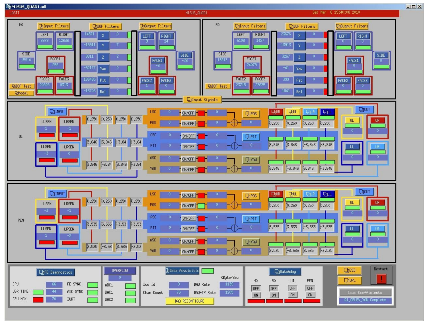
- Supervisory control and data acquisition (SCADA), used for big experiments, factory automation
- Consists of many parts: inter-process communication (start/stop process, change parameters), interface with commercial hardware (signal generators, HVAC, relays, ...), database for configuration management, process supervision, logging, data viewers, monitoring ...
- Virgo does everything custom:
  - Cm library for inter-process communication : very obsolete, no security, replace by OPC??
  - VPM: home built web interface for starting/stopping programs, web editor of config files
  - Home built data viewer, web pages for trend plots
  - SMS/mail alerts when parameters go out of range: custom Detector Monitoring
  - GUIs: some custom based on PyQT, some web interfaces
  - Failed attempt at using TANGO: not everyone was on board
- LIGO uses EPICS, GUIs based on MEDM (bit dated look), custom built data viewers
- Use full SCADA systems with all bells & whistles?: WinCC, Tango, EPICS
- Mix and match to get just what you need?
  - communication: MQTT/OPC/zeroMQ/...,
  - monitoring: Nagios, Grafana, ...
  - GUIs: Labview-like front panels, any open source alternative?

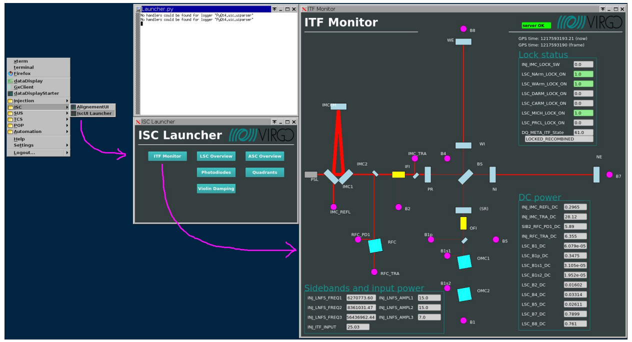
Human interface





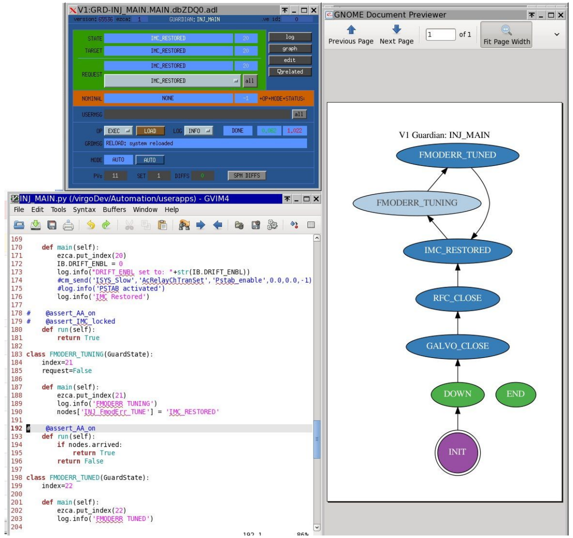




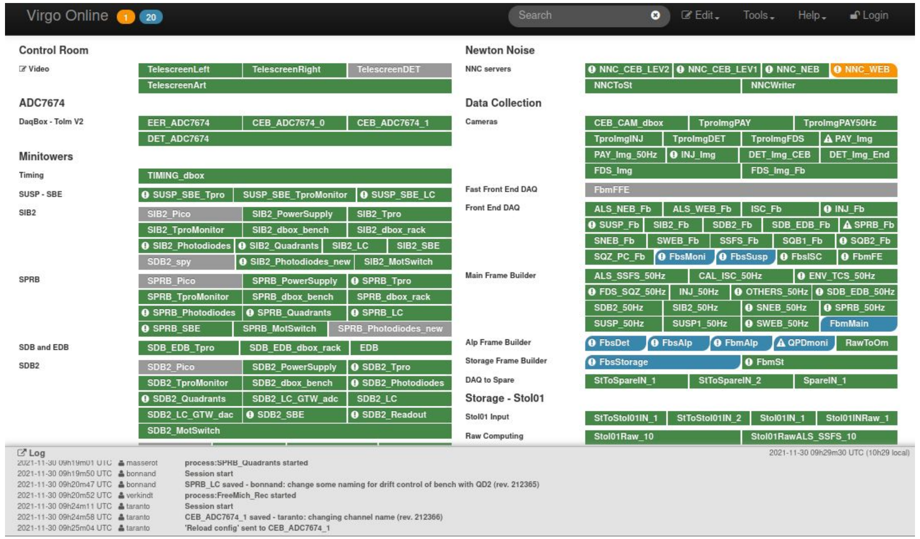


- GUIs, data viewers for human monitoring and control of instrument
- Web-pages with predefined plots for monitoring on scales of hours to days
- Supervising process to check that thousands of parameters are at their expected values, send alarm messages via mails/SMS when something breaks at night: Detector Monitoring System (DMS)

Automation



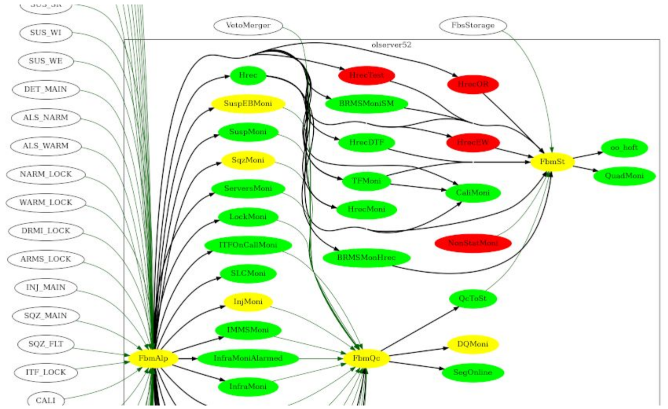
```python
    def main(self):
        ezca.put_index(20)
        IB.DRIFT_ENBL = 0
        log.info("DRIFT_ENBL set to: "+str(IB.DRIFT_ENBL))
        #cm_send('ISYS_Slow','AcRelayChTranSet','Pstab_enable',0.0,0.0,-1)
        #log.info('PSTAB activated')
        log.info('IMC Restored')

#    @assert_AA_on
#    @assert_IMC_locked
    def run(self):
        return True

class FMODERR_TUNING(GuardState):
    index=21
    request=False

    def main(self):
        ezca.put_index(21)
        log.info('FMODERR TUNING')
        nodes['INJ_FmodErr_TUNE'] = 'IMC_RESTORED'

#    @assert_AA_on
    def run(self):
        if nodes.arrived:
            return True
        return False

class FMODERR_TUNED(GuardState):
    index=22

    def main(self):
        ezca.put_index(22)
        log.info('FMODERR TUNED')

```



```
2021-11-30 09h29m30 UTC (10h29 local)
2021-11-30 09h19m01 UTC  masserot   process:SPRB_Quadrants started
2021-11-30 09h19m50 UTC  bonnand    Session start
2021-11-30 09h20m47 UTC  bonnand    SPRB_LC saved - bonnand: change some naming for drift control of bench with QD2 (rev. 212365)
2021-11-30 09h20m52 UTC  verkindt   process:FreeMich_Rec started
2021-11-30 09h24m11 UTC  taranto    Session start
2021-11-30 09h24m58 UTC  taranto    CEB_ADC7674_1 saved - taranto: changing channel name (rev. 212366)
2021-11-30 09h25m04 UTC  taranto    'Reload config' sent to CEB_ADC7674_1
```

- Automation processes are embedded in the DAQ chain, so they have access to all data (with a latency of a few frames). Responsible for sequencing lock acquisition, various slow loops, basic safety checks
- Python-based hierarchical set of state machines: Guardian (LIGO) / Metatron (Virgo), see arXiv:1604.01456
  Needs to be adapted to future inputs/outputs, but in principle good enough for ET
- Process supervision: Virgo Process Monitoring (VPM)

Frame distribution



- Virgo:
  - 1 sec frames, ~ 55 Mbyte/s
  - propriety Fd library that works via Cm/shared-memory
- LIGO:
  - 1/16 sec frames
- Alternatives:
  - various open source frameworks (Apache Kafka, ...)
  - some of the low-latency data analysis is based on Gstreamer (open source multimedia pipeline). Note that detector output is 1 signal at audio frequencies
- Frame format itself: stick with GWF, or use something else (hd5, ...)?

Scaling up from Virgo to ET

- No major technological breakthroughs needed to control ET, could be evolution of current hardware
  - data flux might increase by order of magnitude: Virgo produces 5 TB/day, ET will have multiple interferometers that are more complex
  - ADCs and DACs noise close to limiting for some critical sensors/actuators
  - slightly faster digital loops (replace some more analog loops)
  - slightly better timing: lower phase noise
  - get rid of all obsolete stuff, upgrade software/hardware to state-of-the-art
  - make things more uniform/maintainable: don’t have multiple devices for the same job

Conclusions

- in the past almost all software/hardware custom built by individual groups, not a sustainable model for ET
  - use existing open standards where possible
  - get all the boring stuff that is good enough off-the-shelves, concentrate R&D on where it matters, e.g. integrating ADC/DAC directly into low-noise sensors/actuators
  - Collaborate with LIGO?

Organizational matters

- Regular meetings? not too often, ~ every 2 months?
- Wiki: https://wiki.et-gw.eu/ISB/Interferometer/DataAcquisitionAndReaTimeControl (needs an EGO account)
  - Slides to be put in the ET TDS: https://apps.et-gw.eu/tds/
- Build a WP mailing list?
- Still need to clarify the boundaries of our work package
  - Definitely inside: real-time hardware and software, and their interface with digital electronics, Supervisory Control and Data Acquisition, automation, (+ general purpose ADC/DAC)
  - Definitely outside: storage, general purpose computing hardware/networking, sensors/actuators
  - Boundary on front side: analog signal sensor/actuator and ADC/DAC? Digital link with some sensor/actuator?
    - For example, digital demodulation more in the I/O optics work package or in this WP?
  - Boundary on back side: somewhere between automation and low latency searches/storage/calibration?
- Need to fill in Project Breakdown Structure (see next slides) ASAP
- Available personpower? → fill the ET Member Database: https://apps.et-gw.eu/etmd/?c=1

ET PBS deliverables

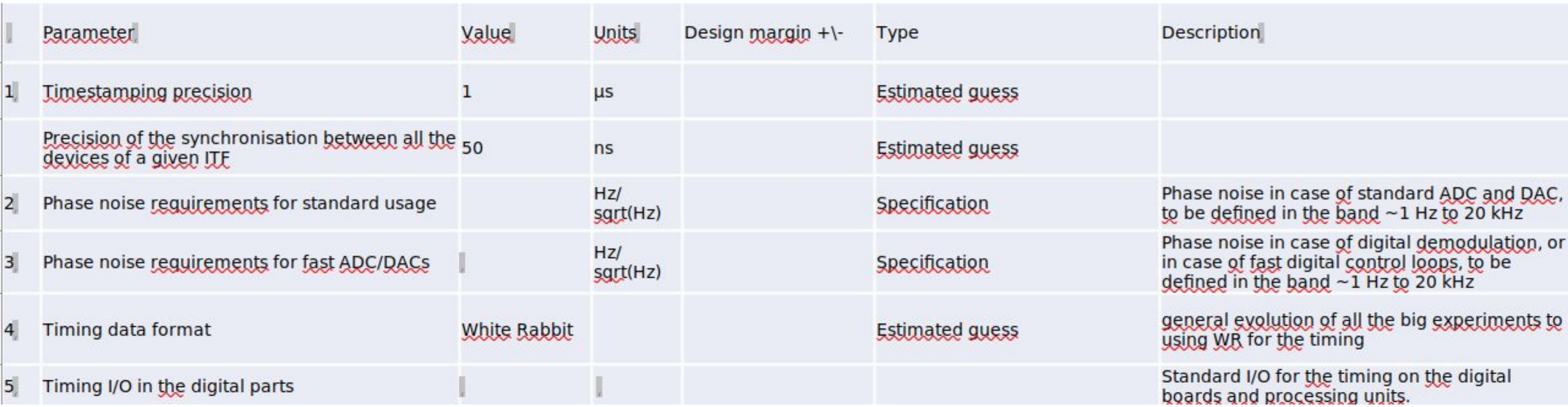
Goal: the "product breakdown structure" (PBS), which will act as THE backbone for many activities (requirements, timeline, WBS, costing ... you name it) to follow.
Not asked to develop full sets of requirements, but simply to take stock,

- 6 mains deliverables defined in the ET PBS:
  - Timing distribution
  - Real-time data distribution
  - Real-time computing
  - DAQ hardware (i.e. timing and real-time networks hardware)
  - Data collection pipeline
  - Data collection software

→ 1st draft in next slides, your input will be valuable!
1/ to set the list of requirements/choices
2/ then to fill the values

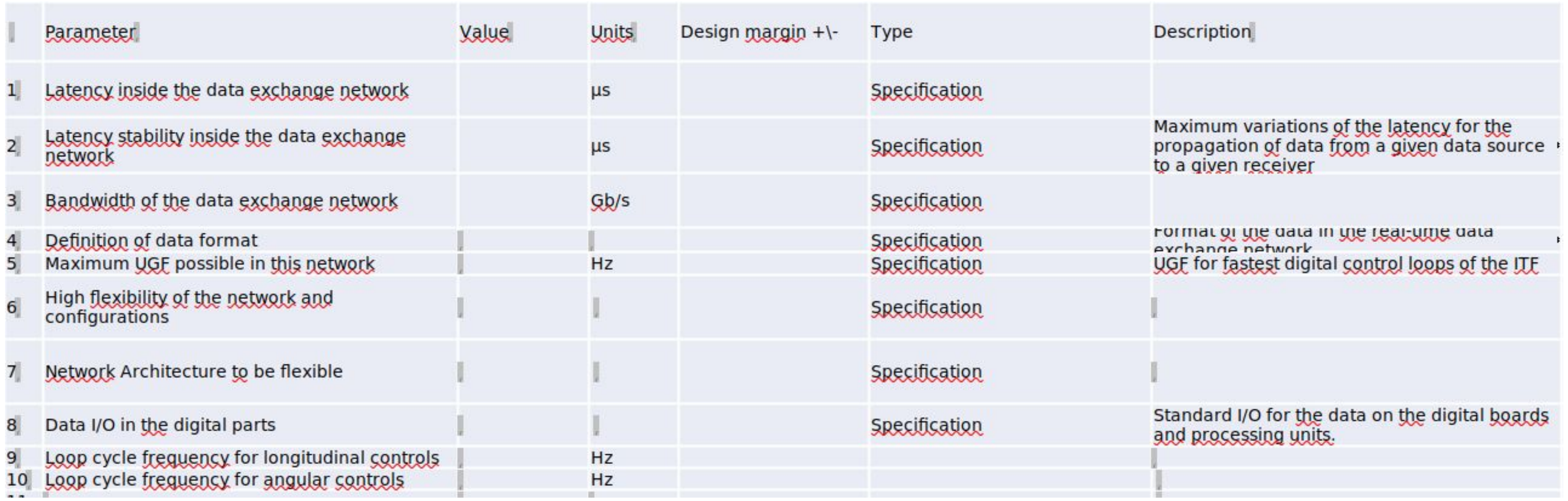
Mainly nothing about digital controls and data collection in the “ET design report update 2020”, but:
“All currently operating interferometers employ digital control loops wherever possible, for the benefit of flexibility, stability and transient noise reduction of control transfer functions. Typical sample frequencies are around 10-64 kHz, but even for faster loops digital control is now used (at Virgo the fastest digital loop is running at 500 kHz.)”

PBS - Timing distribution



	Parameter	Value	Units	Design margin +\-	Type	Description
1	Timestamping precision	1	µs		Estimated guess	
	Precision of the synchronisation between all the devices of a given ITF	50	ns		Estimated guess	
2	Phase noise requirements for standard usage		Hz/ sqrt(Hz)		Specification	Phase noise in case of standard ADC and DAC, to be defined in the band ~1 Hz to 20 kHz
3	Phase noise requirements for fast ADC/DACs		Hz/ sqrt(Hz)		Specification	Phase noise in case of digital demodulation, or in case of fast digital control loops, to be defined in the band ~1 Hz to 20 kHz
4	Timing data format	White Rabbit			Estimated guess	general evolution of all the big experiments to using WR for the timing
5	Timing I/O in the digital parts					Standard I/O for the timing on the digital boards and processing units.

PBS - Real-time data distribution

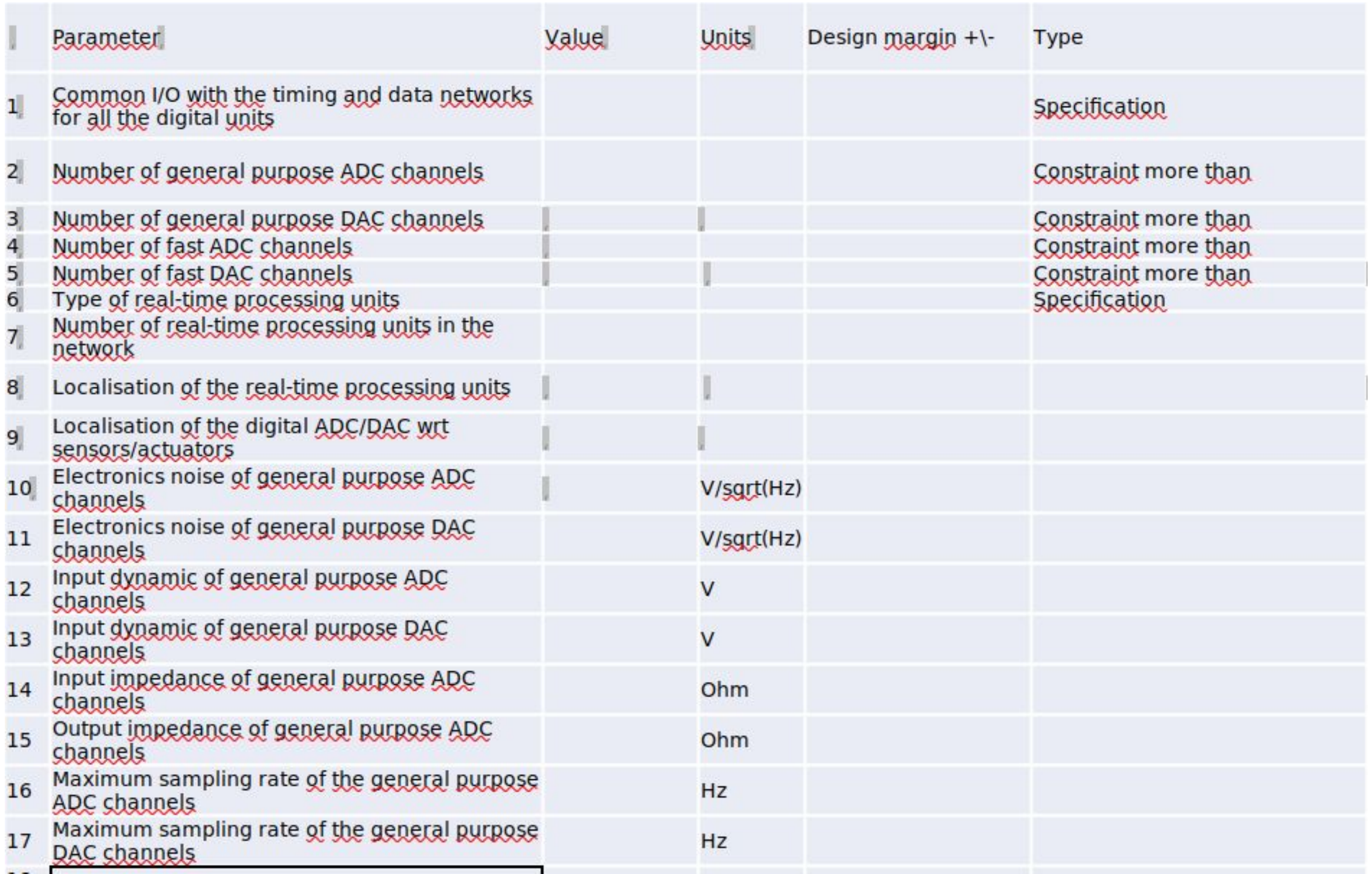


	Parameter	Value	Units	Design margin +\-	Type	Description
1	Latency inside the data exchange network		µs		Specification	
2	Latency stability inside the data exchange network		µs		Specification	Maximum variations of the latency for the propagation of data from a given data source to a given receiver
3	Bandwidth of the data exchange network		Gb/s		Specification	
4	Definition of data format				Specification	Format of the data in the real-time data exchange network
5	Maximum UGF possible in this network		Hz		Specification	UGF for fastest digital control loops of the ITF
6	High flexibility of the network and configurations				Specification	
7	Network Architecture to be flexible				Specification	
8	Data I/O in the digital parts				Specification	Standard I/O for the data on the digital boards and processing units.
9	Loop cycle frequency for longitudinal controls		Hz			
10	Loop cycle frequency for angular controls		Hz			

PBS - Real-time computing

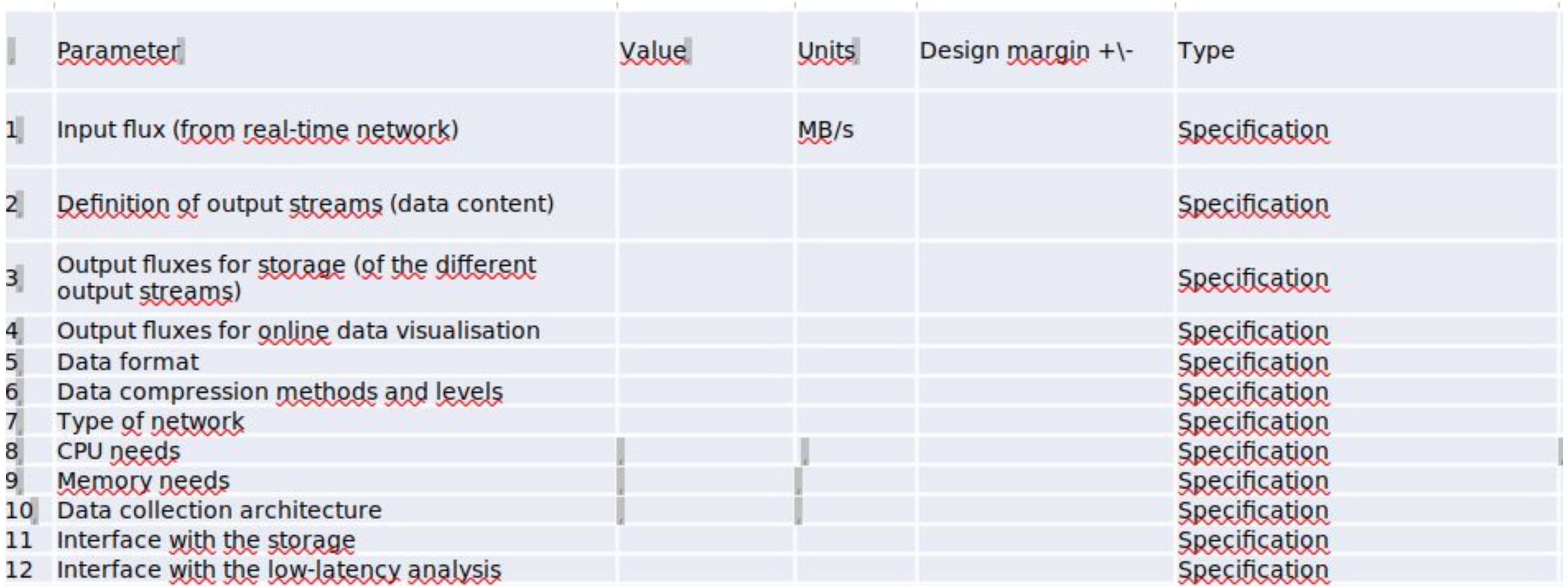
Loop cycle and maximum duration of RT tasks?
Synchronisation/sequence of RT tasks/dependencies?

PBS - Timing and data networks hardware



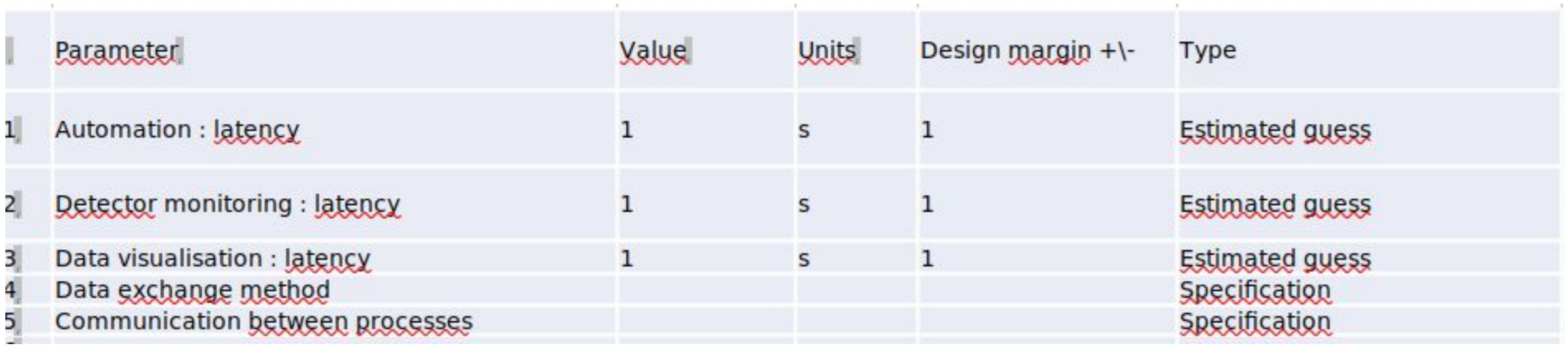
	Parameter	Value	Units	Design margin +\-	Type
1	Common I/O with the timing and data networks for all the digital units				Specification
2	Number of general purpose ADC channels				Constraint more than
3	Number of general purpose DAC channels				Constraint more than
4	Number of fast ADC channels				Constraint more than
5	Number of fast DAC channels				Constraint more than
6	Type of real-time processing units				Specification
7	Number of real-time processing units in the network				
8	Localisation of the real-time processing units				
9	Localisation of the digital ADC/DAC wrt sensors/actuators				
10	Electronics noise of general purpose ADC channels		V/sqrt(Hz)		
11	Electronics noise of general purpose DAC channels		V/sqrt(Hz)		
12	Input dynamic of general purpose ADC channels		V		
13	Input dynamic of general purpose DAC channels		V		
14	Input impedance of general purpose ADC channels		Ohm		
15	Output impedance of general purpose ADC channels		Ohm		
16	Maximum sampling rate of the general purpose ADC channels		Hz		
17	Maximum sampling rate of the general purpose DAC channels		Hz		

PBS - Data collection pipeline



	Parameter	Value	Units	Design margin +\-	Type
1	Input flux (from real-time network)		MB/s		Specification
2	Definition of output streams (data content)				Specification
3	Output fluxes for storage (of the different output streams)				Specification
4	Output fluxes for online data visualisation				Specification
5	Data format				Specification
6	Data compression methods and levels				Specification
7	Type of network				Specification
8	CPU needs				Specification
9	Memory needs				Specification
10	Data collection architecture				Specification
11	Interface with the storage				Specification
12	Interface with the low-latency analysis				Specification

PBS - Data collection software



	Parameter	Value	Units	Design margin +\-	Type
1	Automation : latency	1	s	1	Estimated guess
2	Detector monitoring : latency	1	s	1	Estimated guess
3	Data visualisation : latency	1	s	1	Estimated guess
4	Data exchange method				Specification
5	Communication between processes				Specification

BACK-UP and NOTES

Some notes

- Timing network
  - Most probably WhiteRabbit
  - Need to define constraints on phase noise for the different parts
  - One single network with low phase noise everywhere, or low-phase noise only where critical and high phase noise/cheaper where non-critical?
- Real-time data network. Data used in-loop, so needs to be deterministic, and with low latency,
  - Which constraints on latency and bandwidth? (will put constraints on speed/buffer size for the network switches and processing units…)
  - Which type of real-time processing units/platforms → link with real-time software
  - Which data format?
  - Which maximum UGF should be possible in this network? (high UGF only possible if sensing/driving are close together, could be short “direct” links, but hopefully within same network infrastructure?)
  - High flexibility to be kept as in Virgo network (adding/removing channels on-the-fly, changing sampling frequencies, 1 source can send to multiple receivers, …)
  - Architecture to be flexible also, to allow real-time communication between most/all of the systems. What about the communication between the different ITFs ?
  - Real-time software
- Data collection
  - Which data format ? (gwf in Virgo now) Same for online and offline is very practical to use the same tools in both cases, data visualisation, …
  - Ethernet network -> ok ?
  - Which input flux, can it be centralized on a single machine or not, if not, how to share streams?
  - Software : Automation, monitoring, data visualisation, data exchange and process communication/configuration
  - Which streams (for online visualization, for storage, …)
  - Which data flow at the storage level ?
  - More robust to separate timing and data networks, more difficult to debug in case of networks based shared hardware
- Boundaries and interfaces
  - Define a limit at the interface with timing/data networks → users can develop specific boards for their systems
  - Include general purpose ADC and DAC in the WP, so that the same hardware can be used in most of the experiment
  - Digital demodulation: in DAQ or in another WP?
  - Fast ADC and DAC for loops with UGF ~<10 kHz (as currently SSFS, Squeezing CC, ALS, PCal in Virgo). Could also be common, inside the DAQ WP?
  - Very fast ADC and DAC for loops with UGF 100-500 kHz ? is it needed/possible, for some laser/IMC/INJ loops,... → common or very specific ?

List for PBS deliverables (1)

- General
  - determine number of required ADCs, DACs, loops etc (interface with DET, PSL, SUSP, ENV, ...)
  - determine data flux
  - determine ADC/DAC requirements
  - determine dig demod requirements (if we use it)
- facilities: determine space, power, cooling requirements for DAQ hardware (interface with infrastructure)
  - next to experiment (interface with DET/PSL/SUSP/ENV/...)
  - in separate underground DAQ rooms
  - on the surface
- timing:
  - support RF demodulation? (big impact on phase noise requirements)
  - determine requirement (some input from calibration, DET/ISC for dig demod)
  - choose timing framework
  - determine sample rates: 2^n at LIGO vs vs {1,2,5}*10^n at Virgo

List for PBS deliverables (2)

- SCADA:
  - determine requirements
  - choose framework
- Crates
  - crate physical/electrical format
- ADCs/DACs
- Real time communication
- frame distribution framework (with calibration/low latency)
- frame format