Virtual Platform

Cristian Cucchiella
Ferruccio Foti

Context

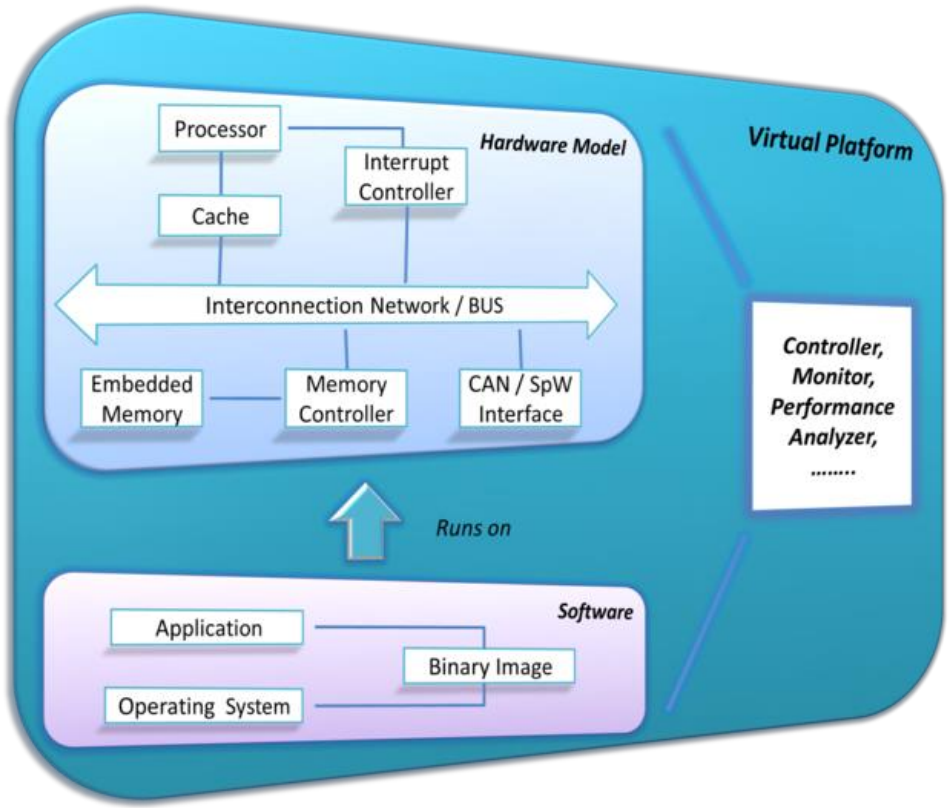
In Embedded Systems, Software and Firmware components (SWCI & FWCI) are tightly coupled.

SWCIs are developed when FWCIs a r e n ô t a v a i l a b l e .

IV&V usually takes place at Unit or System level: problems discovered at this level can have heavy impact on Project Schedule.

Software is difficult to test against regressions: FW and HW must be available and tests are not performed in an automatic way (i.e. Continuous Integration environments)

Virtual Platform



Source: esa.int

Software based system that can fully mirror the functionality of a target System-on-Chip or board.

Binary compatible

Can be enhanced adding mo as missing firmware components

Many VPs available commercially and not.

Virtual Platform Technology

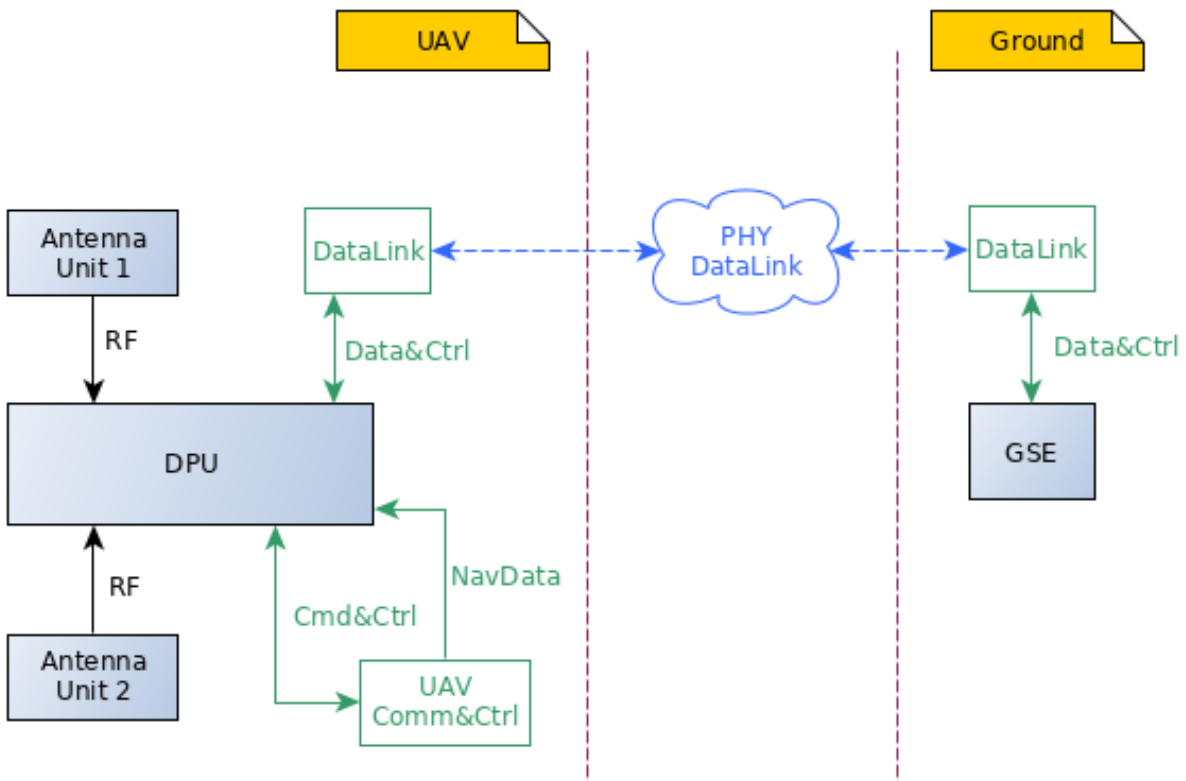
Early software development which can start in advance of RTL simulation, or FPGA prototype availability.

High controllability: physical hardware cannot usually be stopped at once and it cannot be completely and un-intrusively inspected. A VP, being a software program, can be fully managed and customized by the hardware designed to address its needs.

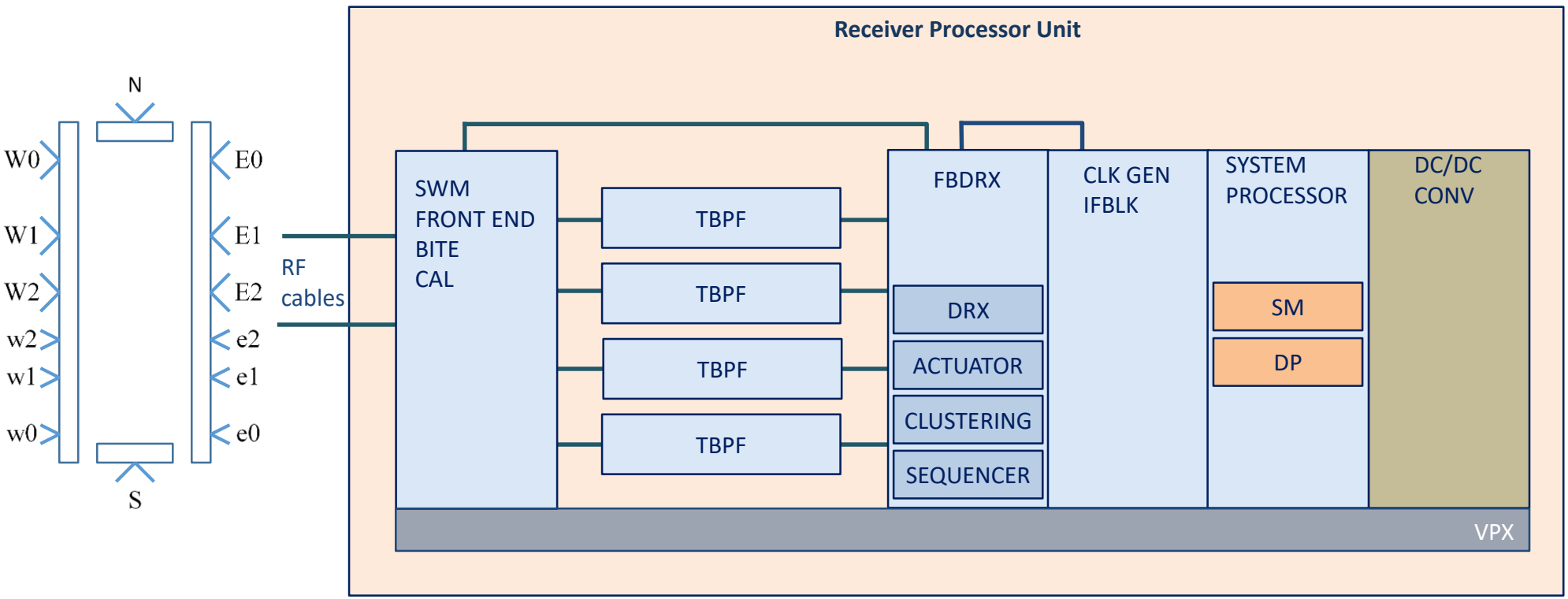
Determinism: being able to reproduce intermittent problems or complex timing-related problems on physical hardware is difficult. A VP, instead, provides determinism in the form of repeatability of a test case where the user will get the exact same behavior over and over again.

Optimized group interaction: VPs, representing the current status of the system being designed, can easily be shared across distributed teams; VPs also improve communication and interaction between hardware, software design and V&V teams. No last minute surprises and big design changes.

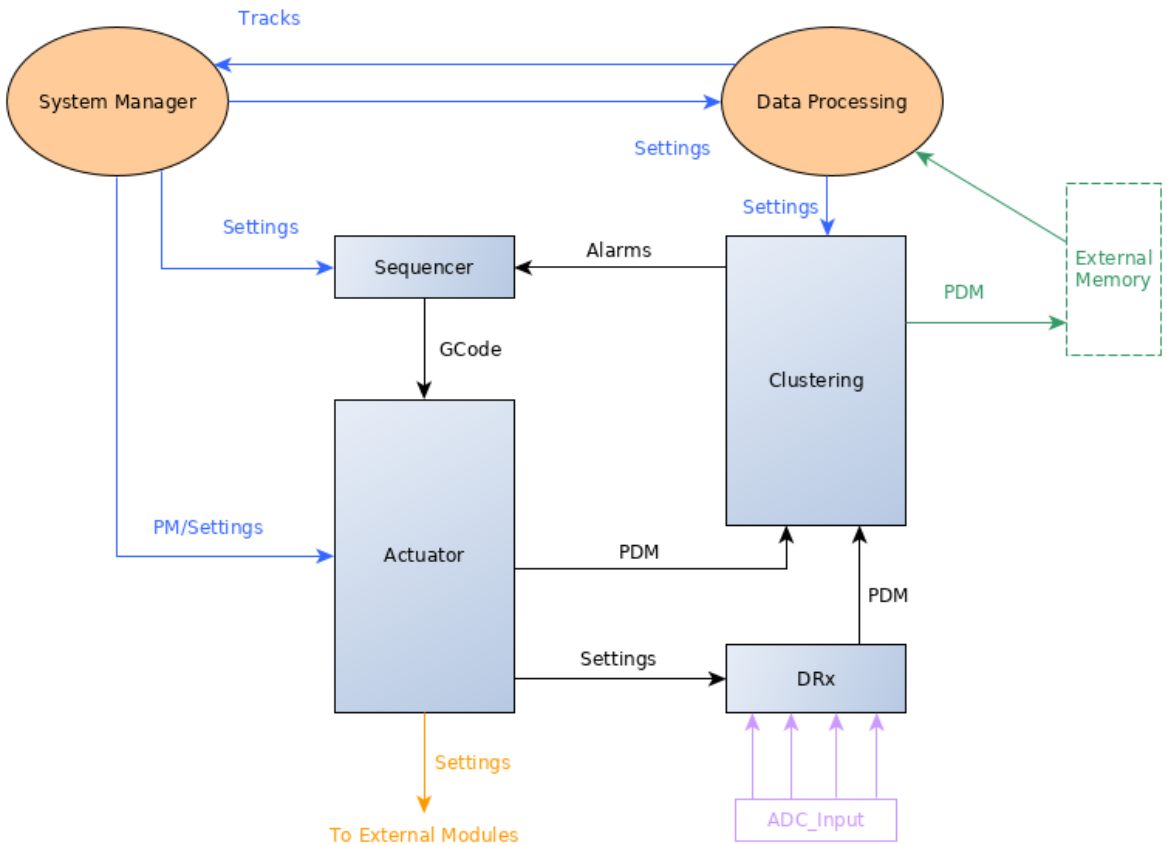
Use Case


UAV
Ground
Antenna Unit 1
DataLink
PHY DataLink
DataLink
RF
Data&Ctrl
Data&Ctrl
DPU
GSE
RF
NavData
Cmd&Ctrl
Antenna Unit 2
UAV Comm&Ctrl

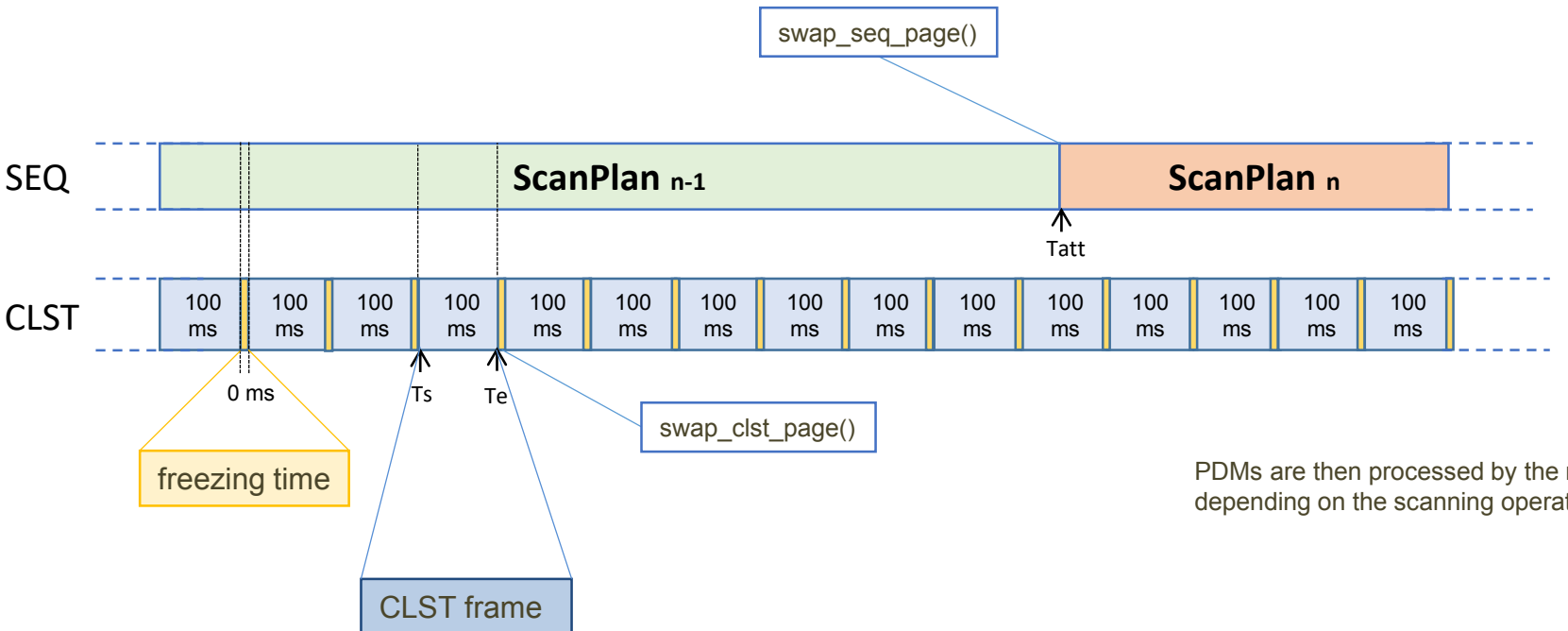
Use Case


N
W0
W1
W2
w2
w1
w0
E0
E1
E2
e2
e1
e0
S
RF
cables
Receiver Processor Unit
SWM
FRONT END
BITE
CAL
TBPF
TBPF
TBPF
TBPF
FBDRX
DRX
ACTUATOR
CLUSTERING
SEQUENCER
CLK GEN
IFBLK
SYSTEM
PROCESSOR
SM
DP
DC/DC
CONV
VPX

Use Case


Tracks
System Manager
Data Processing
Settings
Settings
Settings
Sequencer
Alarms
External Memory
PDM
GCode
Clustering
PM/Settings
Actuator
PDM
PDM
Settings
DRx
Settings
To External Modules
ADC_Input

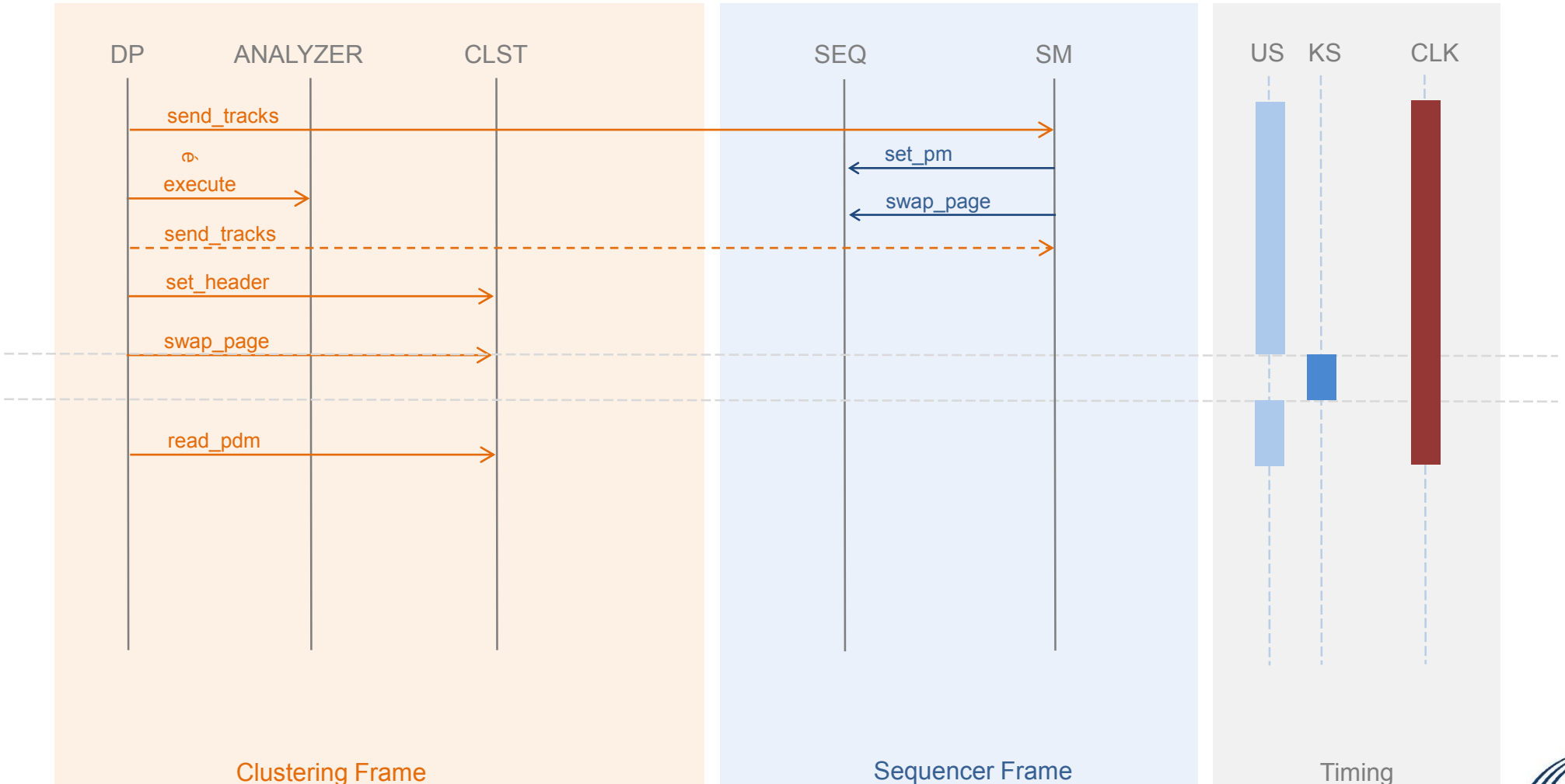
Use Case

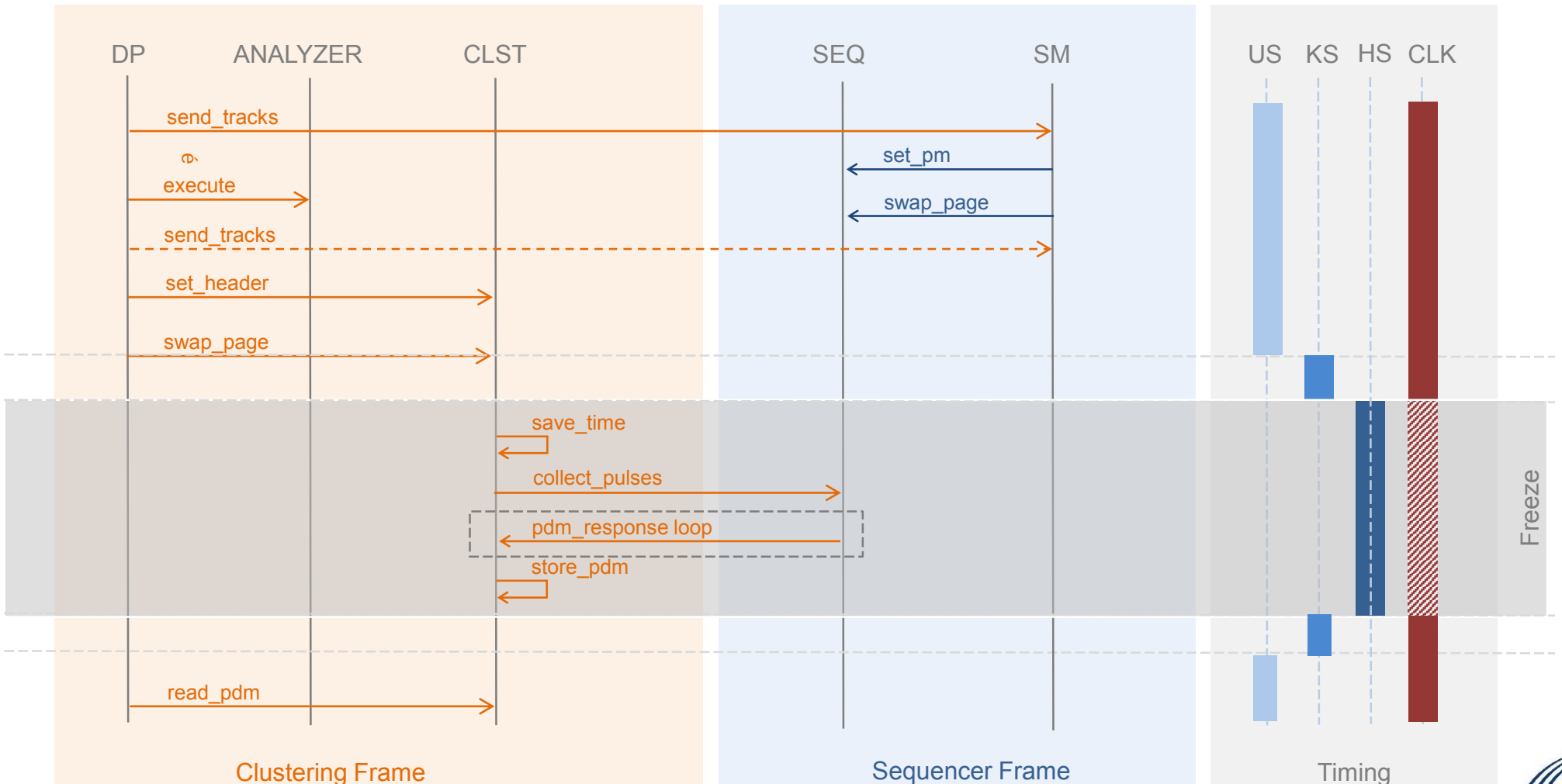


For each tuning, the PDWs, characterized by a TOA included in the tuning actuation interval (Ts õTe) are extracted from the CEESIM file.

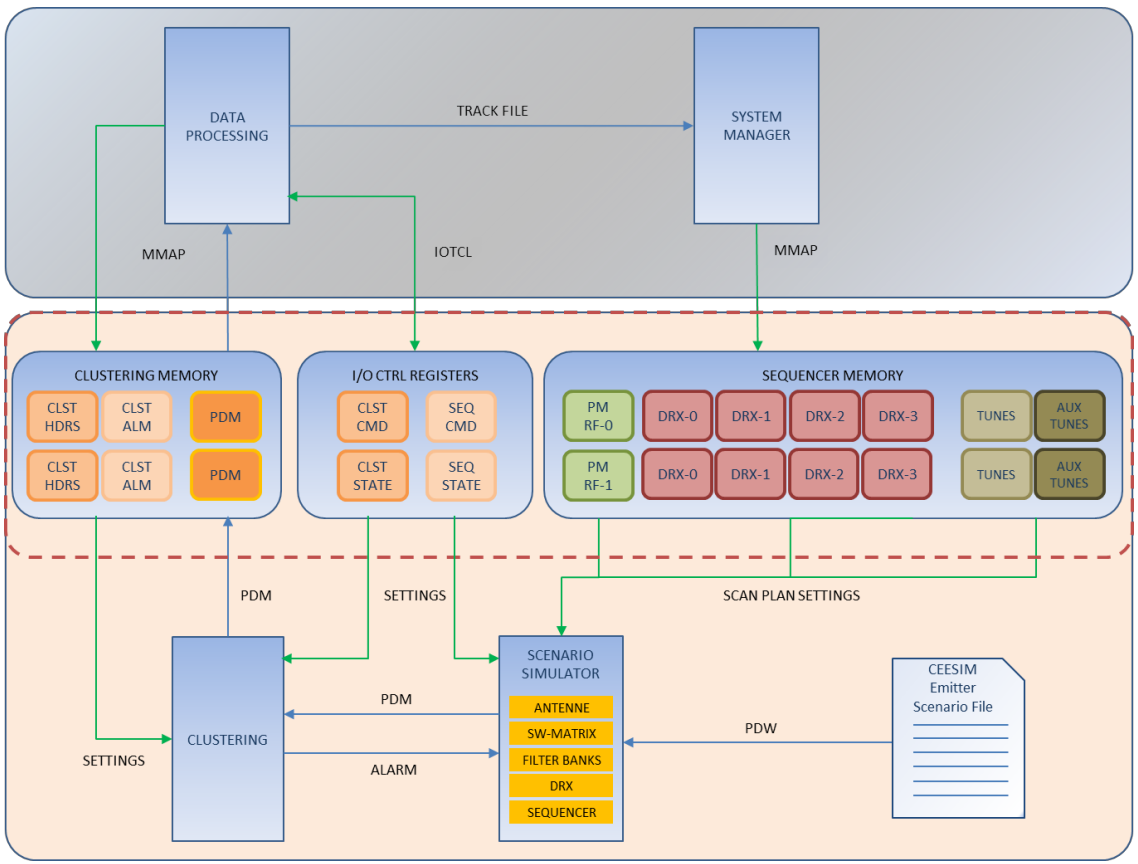
PDMs are then processed by the receiving chain (antennas, filters, DRx), depending on the scanning operating mode.

Detected PDMs are sent to the Clustering one at a time, sorted according to TOA. Clustering, collects them in chains.


DP
ANALYZER
CLST
SEQ
SM
US
KS
CLK
send_tracks
set_pm
execute
swap_page
send_tracks
set_header
swap_page
read_pdm
Clustering Frame
Sequencer Frame
Timing


DP
ANALYZER
CLST
SEQ
SM
US
KS
HS
CLK
send_tracks
set_pm
execute
swap_page
send_tracks
set_header
swap_page
save_time
collect_pulses
pdm_response loop
store_pdm
read_pdm
Freeze
Clustering Frame
Sequencer Frame
Timing

Use Case


DATA PROCESSING
TRACK FILE
SYSTEM MANAGER
MMAP
IOTCL
MMAP
CLUSTERING MEMORY
CLST HDRS
CLST ALM
PDM
CLST HDRS
CLST ALM
PDM
I/O CTRL REGISTERS
CLST CMD
SEQ CMD
CLST STATE
SEQ STATE
SEQUENCER MEMORY
PM RF-0
DRX-0
DRX-1
DRX-2
DRX-3
TUNES
AUX TUNES
PM RF-1
DRX-0
DRX-1
DRX-2
DRX-3
TUNES
AUX TUNES
PDM
SETTINGS
SCAN PLAN SETTINGS
SETTINGS
CLUSTERING
PDM
ALARM
SCENARIO SIMULATOR
ANTENNE
SW-MATRIX
FILTER BANKS
DRX
SEQUENCER
PDW
CEESIM Emitter Scenario File

Virtual Platform Architecture in ELT

Forked version of QEMU which is integrated in a development environment that helps the users with common tasks (defining hardware interface, implementing hardware behavior).

The QEMU image is expanded with a QEMU module devoted to the new hardware implementation.

The platform also provides a generated Linux kernel module (Yocto Project) and a Software API for interfacing the real and emulated hardware (Eclipse Acceleo).

Docker containers wrap up QEMU and its dependencies into a standardized unit for software development that includes everything it needs to run: code, runtime, system tools and libraries.

Virtual Platform Framework, QEMU

Full software MMU for maximum portability.

Optionally, use an in-kernel accelerator, like kvm. The accelerators execute most of the guest code natively, while continuing to emulate the rest of the machine.

Various hardware devices can be emulated and in some cases, host devices (e.g. serial and parallel ports, USB, drivers) can be used transparently by the guest Operating System.

Symmetric multiprocessing (SMP) support. Currently, an in-kernel accelerator is required to use more than one host CPU for emulation.

Virtual Platform Framework, YOCTO PROJECT ©

Open source collaboration project that helps developers create custom Linux-based systems for embedded products, regardless of the hardware architecture.

The project provides a flexible set of tools and a space where embedded developers worldwide can share technologies, software stacks, configurations and best practices which can be used to create tailored Linux images for embedded devices.

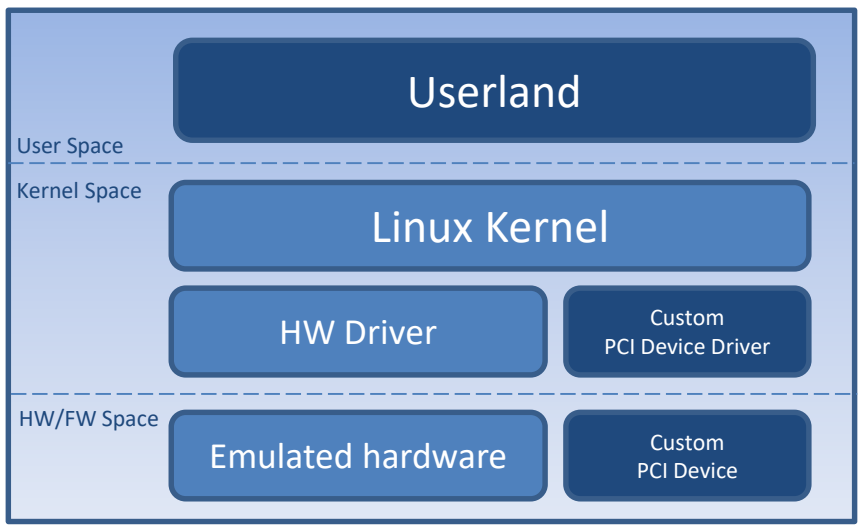
A standard to delivering hardware support and software stacks, allowing the interchange of software configurations and builds.

Virtual Platform Framework, Eclipse© Acceleo

Template-based technology including authoring tools to create custom code generators. It allows you to automatically produce any kind of source code from any data source available in EMF format.

A pragmatic implementation of the Object Management Group (OMG) MOF Model to Text Language (MTL) standard, Acceleo is the result of several man-years of R&D started in the Obeo company.

Virtual Platform Framework, Eclipse Acceleo



Virtual Platform Framework, Docker

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

Containers share the machineôs OS system kernel a driving higher server efficiencies and reducing server and licensing costs.