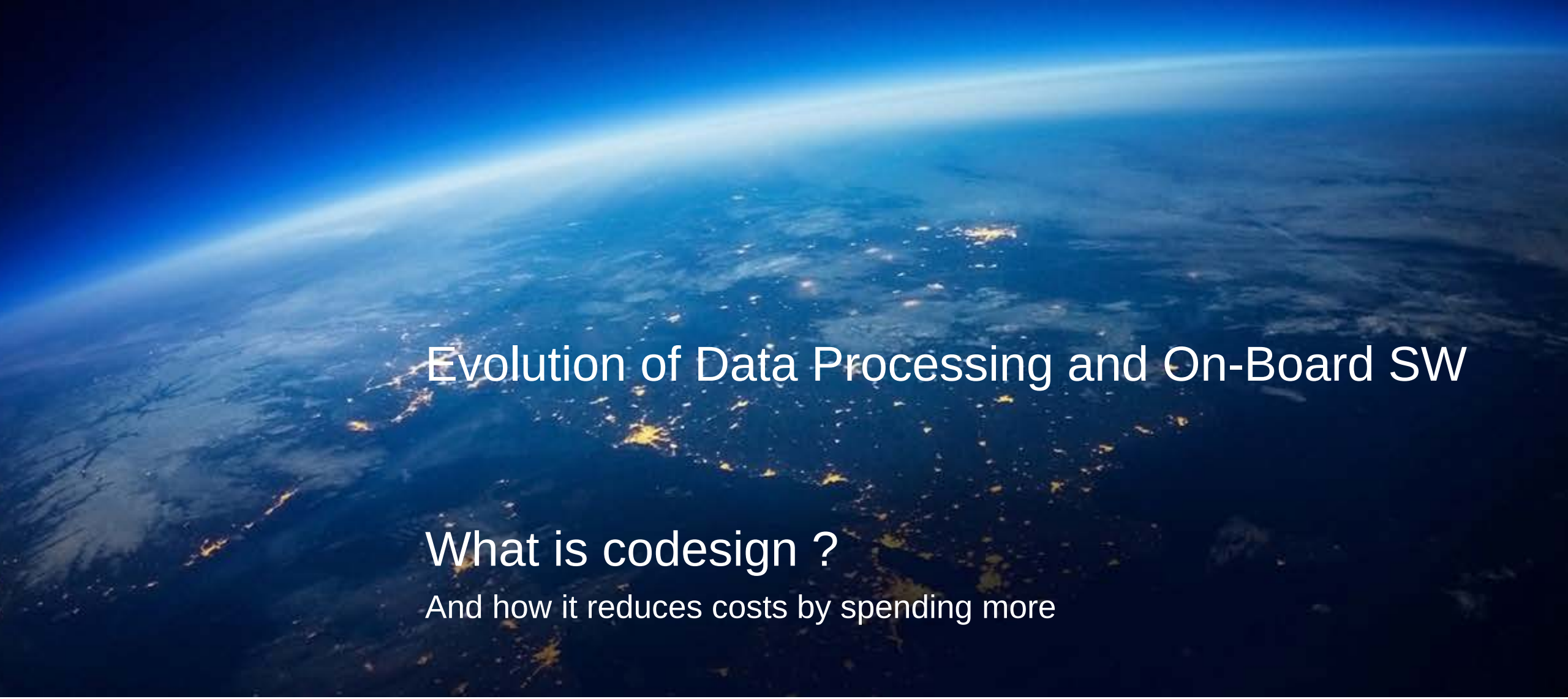


# Evolution of Data Processing and On-Board SW

## What is codesign ?

And how it reduces costs by spending more

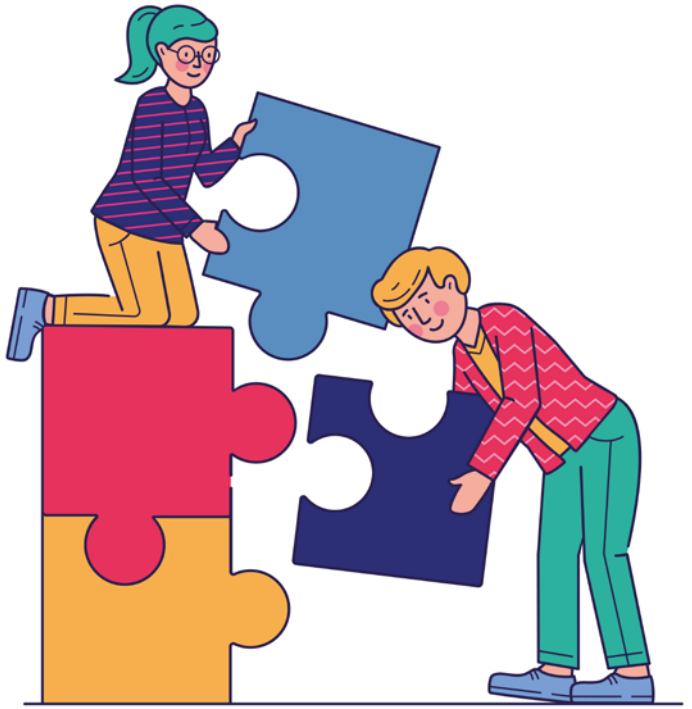
DEFENCE AND SPACE

LACHAIZE Jérôme  Data Processing and On-Board SW  Expert
October 21st  2020

AIRBUS

# Outline

- Where is codesign  ?
- What is codesign ?
- Workflow impacts
- Definition of activities covered
- Cons and Pros
- How to help its adoption
- Conclusion and recommendation



# Where is codesign ?

## Everywhere, since software is everywhere

Soft CPU are commonly used in hardware design with « firmware » but it is really software more and more complex => new needs in terms of qualification E.G. A RTG4 design may embed a cortex-M running at more than 40 MHz with 128 KiB of memory. It is more than a E3000 OBC
*RTG4 is already proven in Space with soft core,*

The Zynq (Cortex A9) has been used in Space in nanosat and more,
NanoXplore develops FPGA and APSoC for Space :

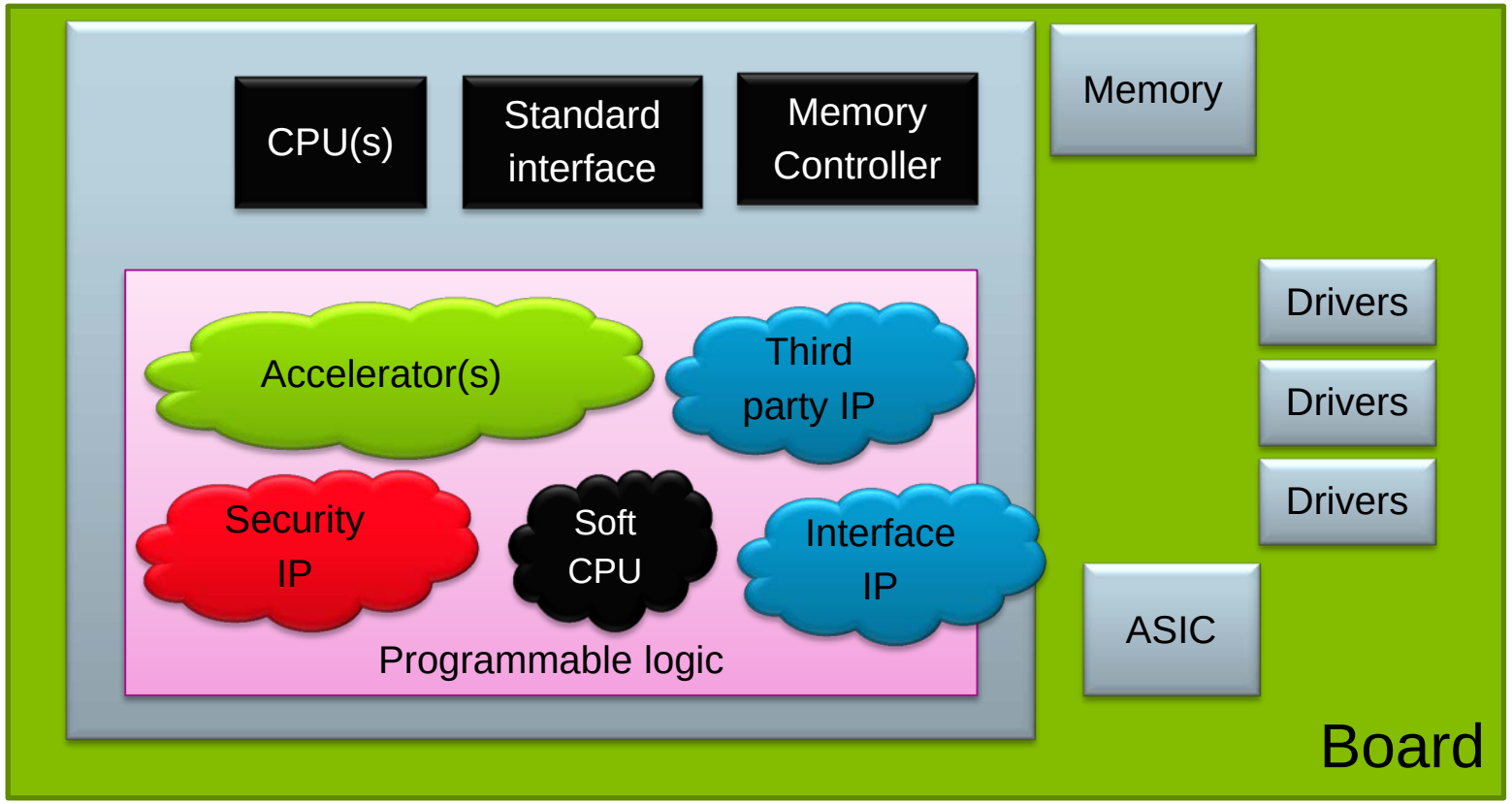
- NG-Ultra (Cortex R52),
- NG-Large (Cortex R5),

Future projects with RISC-V + FPGA ,

# Whats is co-design ?

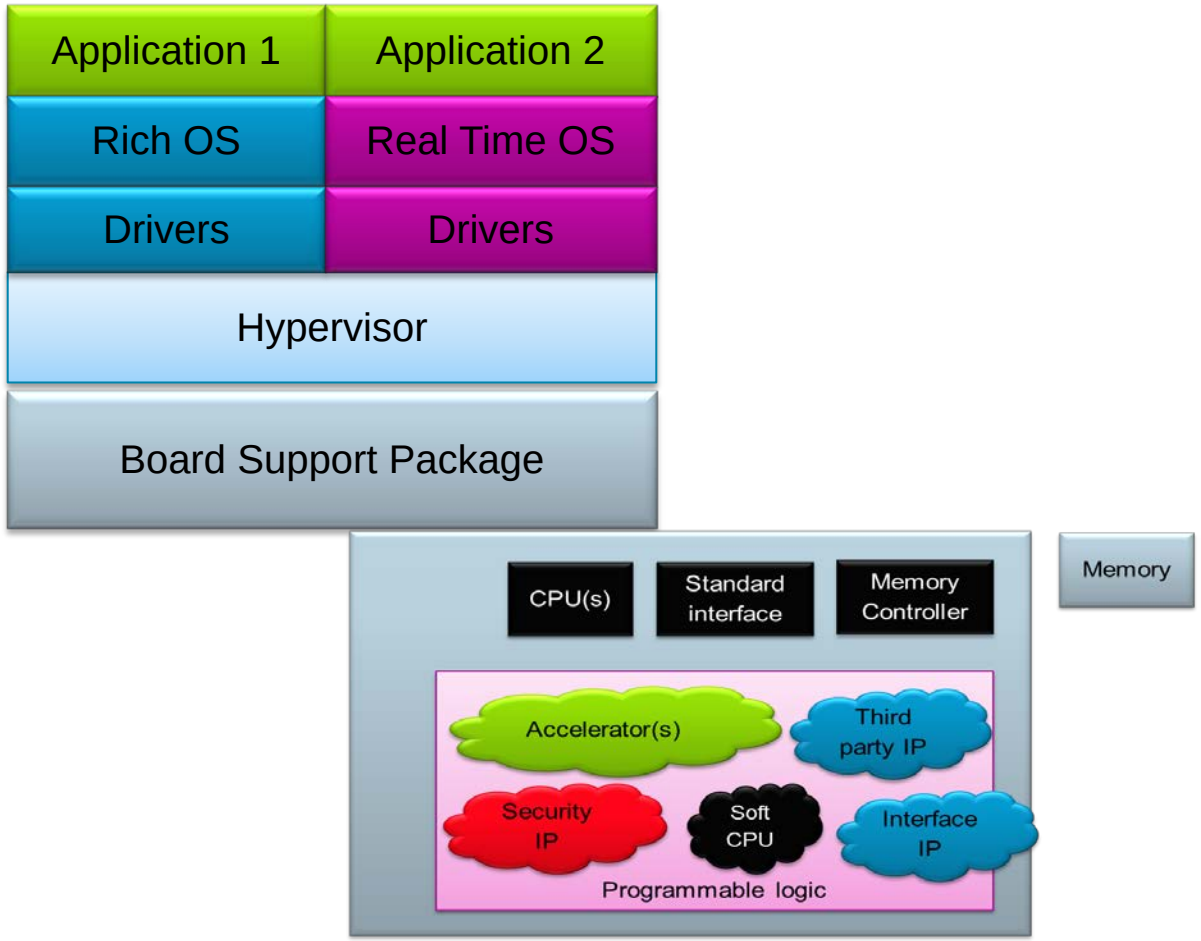
Goal is to improve the efficiency of the design process when both hardware and software are customized.

- APSoC  Application Programmable System On Chip
- Deal with complex architecture both in hardware…



# Whats is co-design ?

…and in software.



# Why co-design ?

- Solve interactions between  Hardware and Software
- Solve misunderstanding at the very beginning of the system
- Solve issue before it appears  because of observability reduced in highly integrated design
- Early assesment of functional perfomance and major bottleneck
  - Starting at high abstraction level
  - Refinement  of models theoricaly possible but most probably shoot and forget approach

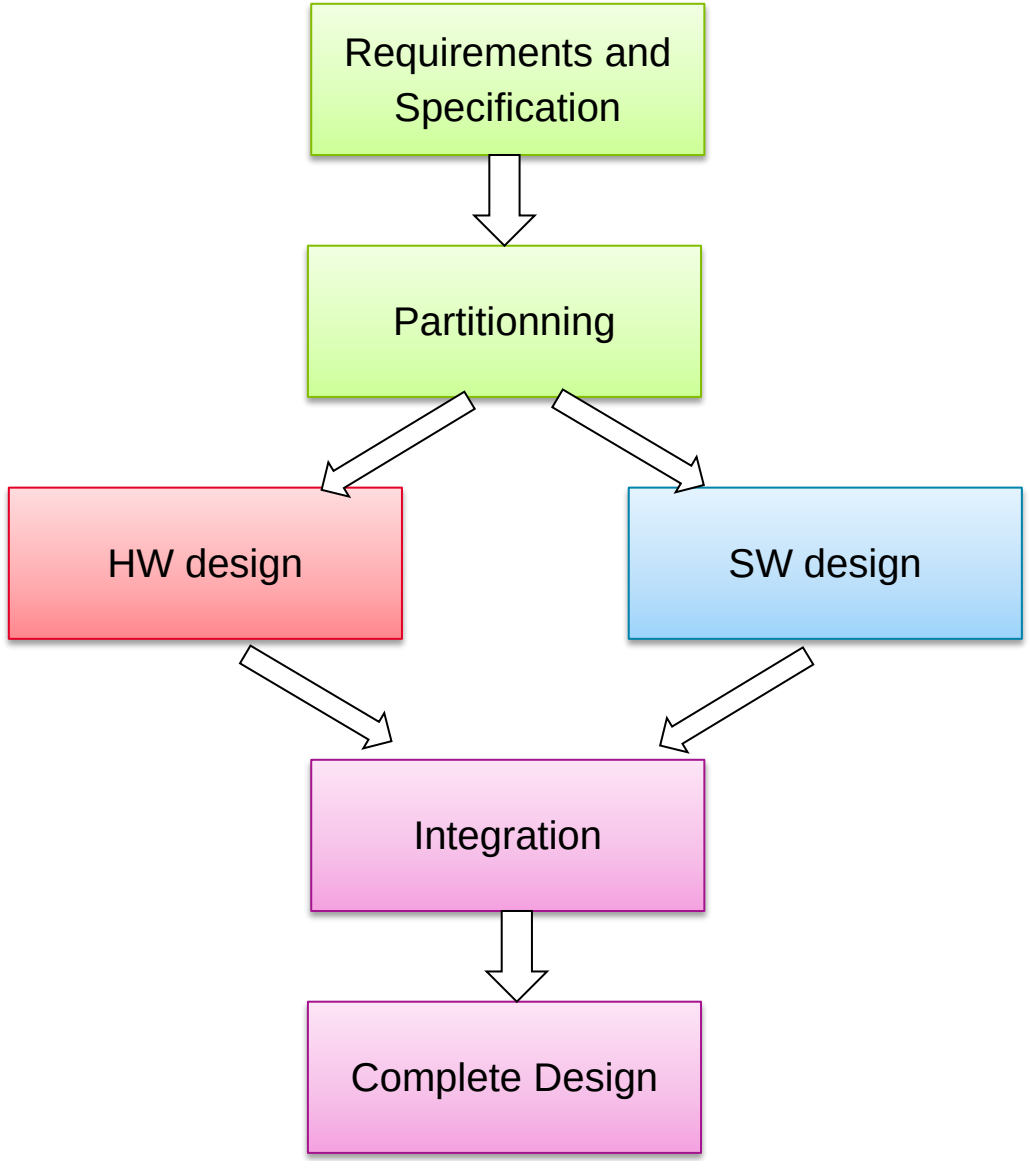
- Use of engineering models for :
  - requirements consolidations
  - Model cross validation

# Classical workflow

- System engineer split systems in equipment and map functions inside, define communication links and interface, estimate estimate the needs in power processing, memory

- Equipment maker receive  partitioned design as HW specification
- Software developer receive partitioned design as SW specification
- HW/SW interface not yet defined
  - Need a definition from HW side due to non reprogrammable aspect of HW ➔ delay
  - Software constraints (latency, cyclicality) not known by HW

- Integration shows poor interactions
  - Performance issue may endanger the design
  - Misunderstanding
  - Ineffective SW development for complex HW Ips
  - Complex debugging  HW or SW bugs ?






AIRBUS

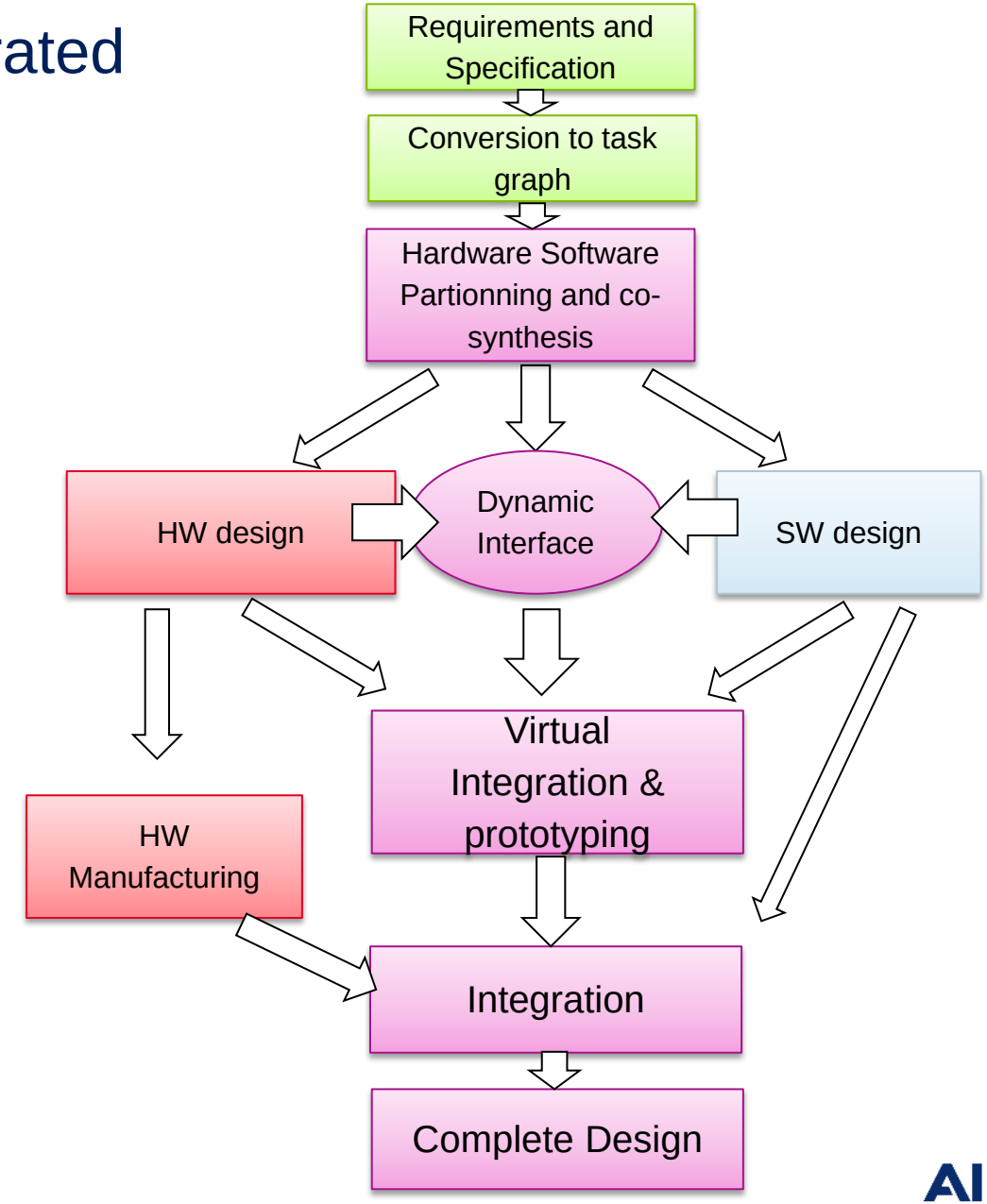
# Co-design workflow : iterative and integrated

- System engineer split systems in tasks and map functions with HW and SW architects, define communication links and interface, estimate the needs in power processing, memory

- Equipment maker define HW/SW ICD document
  - From concept to implementation e.g. Event driven ➔ Interrupt, status register, action duration

- SW and HW architects define interaction between HW and SW
  - Sequence of concept and timing assertion e.g. An event per character on a serial line, HW operation => early User Manual

- Virtual platform support build the complement for each domain
  - Pseudo processor + register access for FPGA prototyping
  - IP models for tests development
  - Virtual platform including IP interface accurate and fonctionnal model for software functional validation






AIRBUS

# Hardware Software Co-design definition



- Co-design of embedded computer systems encompasses the following parts :
  - ➢ Co-Specification

    Developing system specification that describes hardware and software modules and relationship between the hardware and software

  - ➢ Co-Synthesis

    Automatic and semi-automatic design of hardware and software modules to meet the specification

  - ➢ Co-simulation and Co-verification

    Simultaneous simulation of hardware and software

# Pro and Cons of co-design : the Cons

- Higher cost due to new team involved (virtual platform) and models developments
- Poor representativeness of virtual platform for performance assessment
- Model maintenance may be costly
- Non cost efficient if hardware evaluation board exists with high observability
- Complex components (CPU core , interconnect)  are time-consuming and expensive to develop and validate require support teams from both HW and SW.
- Change in organization, responsibilities and design “secrets” are more shared

# Pro and Cons of co-design : the Pros

- Early stabilization of interface
- Sharing of a common understanding, early user manual.
- Software Functional validation independent from Hardware platform and electrical issue
- Works without solving every detail, with preliminary design without data significance
- Additional review of requirements (virtual platform team)
- Reduce risks of ineffective design (risk mitigation)
- Shared virtual integration platform
- Virtual platform support incremental design
- Re-use of system simulation
- Seamless integration at system level (in the whish list)
- Compatible with reprogrammable FPGA-based design

# How to help to co-design deployement

- Co-nsolidate the team
  - Co-location : unformal and continuous exchange(

    coffee machine, ~~Open space~~ https://www.inc.com/jessica-stillman/new-harvard-study-you-open-plan-office-is-making-your-team-less-collaborative.html
  - Co-ncerns sharing : sensitize, get tendencies

    latency, reactivity, virtualization
  - Co-mmon exchange language : standardization, share description and allow job-specific generation tools

    IP-XACT, RDL
  - Co-mmon modelling language

    SysML for capture , SystemC for execution

# How to help to co-design deployement

- Give AGILEity to the process
  - Regular delivery of draft interface documents
    - Reduce serialization of the developmentS,
    - Work on abstract concept instead of bitfields
  - Frequent meetings between System(s), HW and SW team
    - Meeting agenda mandatory,
    - Limited time
    - Order of the day built from anyone request,
    - Meeting Minute : recalling key point and answer to order of the day,
    - Action list if any additional investigation is required to provide an answer,

# Conclusion

- The major risk in reprogrammable FPGA design is that one may consider that correction could be done late but any architectural issue impact both hardware and software design.
- The process is applicable for complex ASIC when they embed software.  In this case missing issue could drive to a major failure of the project. E.g.
  - ➢ The coupling on a die of a Big Endian processor with a Little Endian DSP shows large software overhead to exchange data.
  - ➢ iMX component is unable to reach the gigabit traffic due to limitation on the internal bus.

The development of new integrated systems will cost less in term of power, mass, size  but will be more risky if there is no investment in the design process.

Spend more to lesser risks

Thank you

AIRBUS