THALES

Time predictability and reliability of multi-core processors for critical embedded systems

ARNAUD GRASSET – MARCH 15TH, 2016

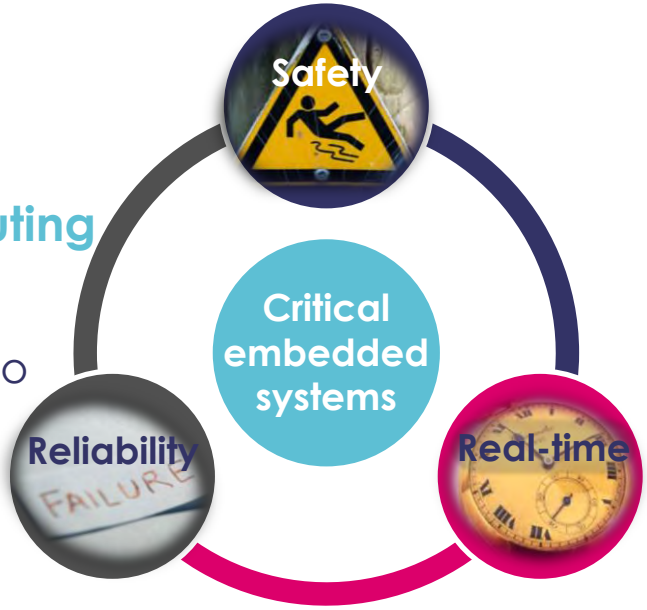
JOURNÉE THÉMATIQUE "*SYSTÈMES EMBARQUÉS POUR LES TRANSPORTS DE DEMAIN*"

Introduction

Safety-critical embedded systems have specific requirements

Challenged by the evolution of computing architectures

- High performance multi-core processors to fulfill ↗ performance requirements



Need to provide guarantees on dependability and hard real-time requirements

Contents

The example of cockpit evolution

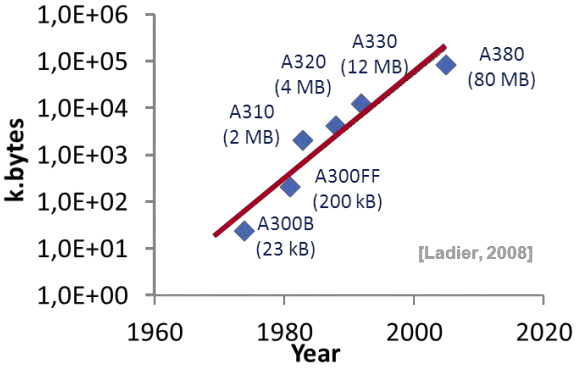
Performance growing needs for improved safety and comfort!

(growing automation, anti-collision systems, improvement of passengers comfort …)

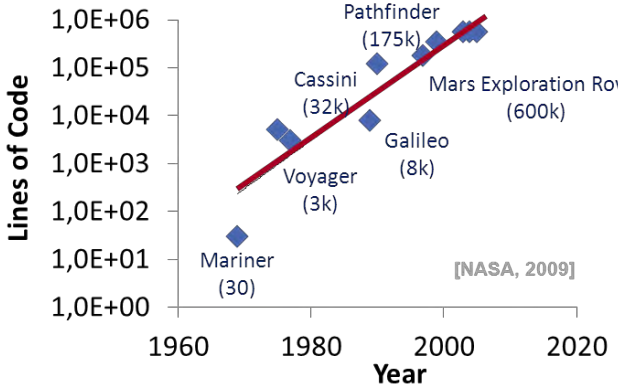
Mission critical systems: high-perf. requirements

Code size evolution for critical embedded systems

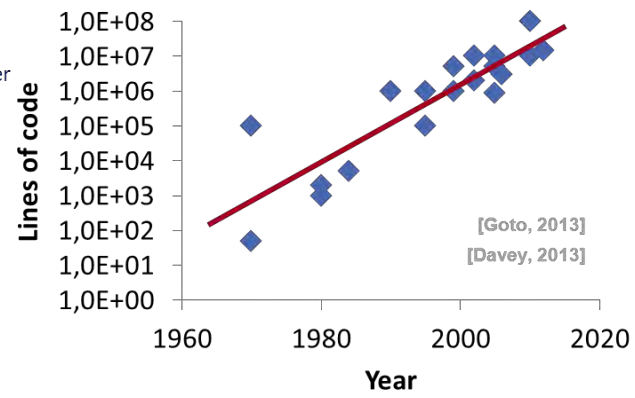
Code size for Airbus aircraft



Code Size for Space Missions



Code size for automotive



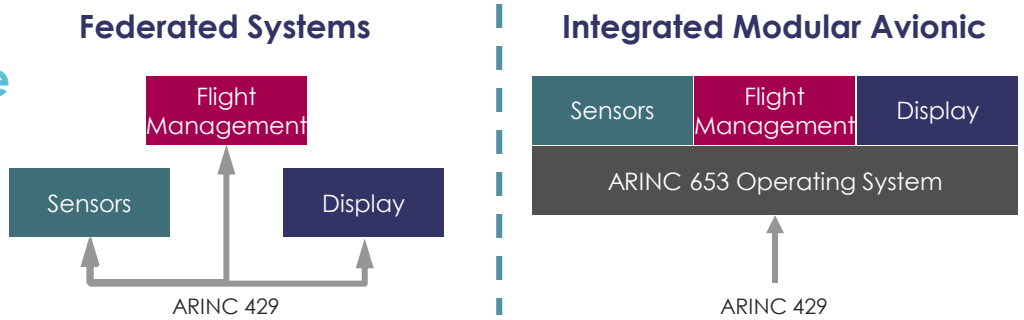
Low-power low-performance processors are not sufficient anymore for mission-critical systems

Avionics bay

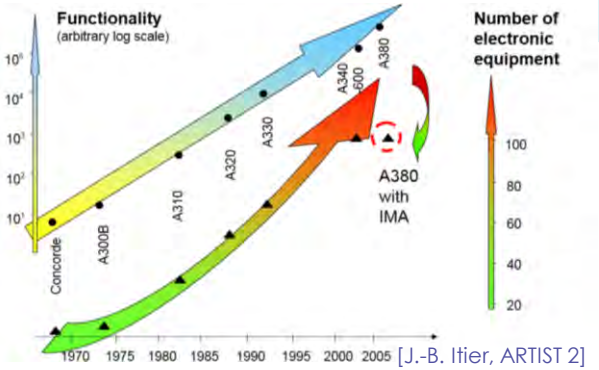
Size, Weight and Power (SWAP) …

Evolution of computing architectures: the case for avionics

Evolution from Federated to Integrated Modular Architecture (IMA)



Motivations for IMA



[J.-B. Itier, ARTIST 2]

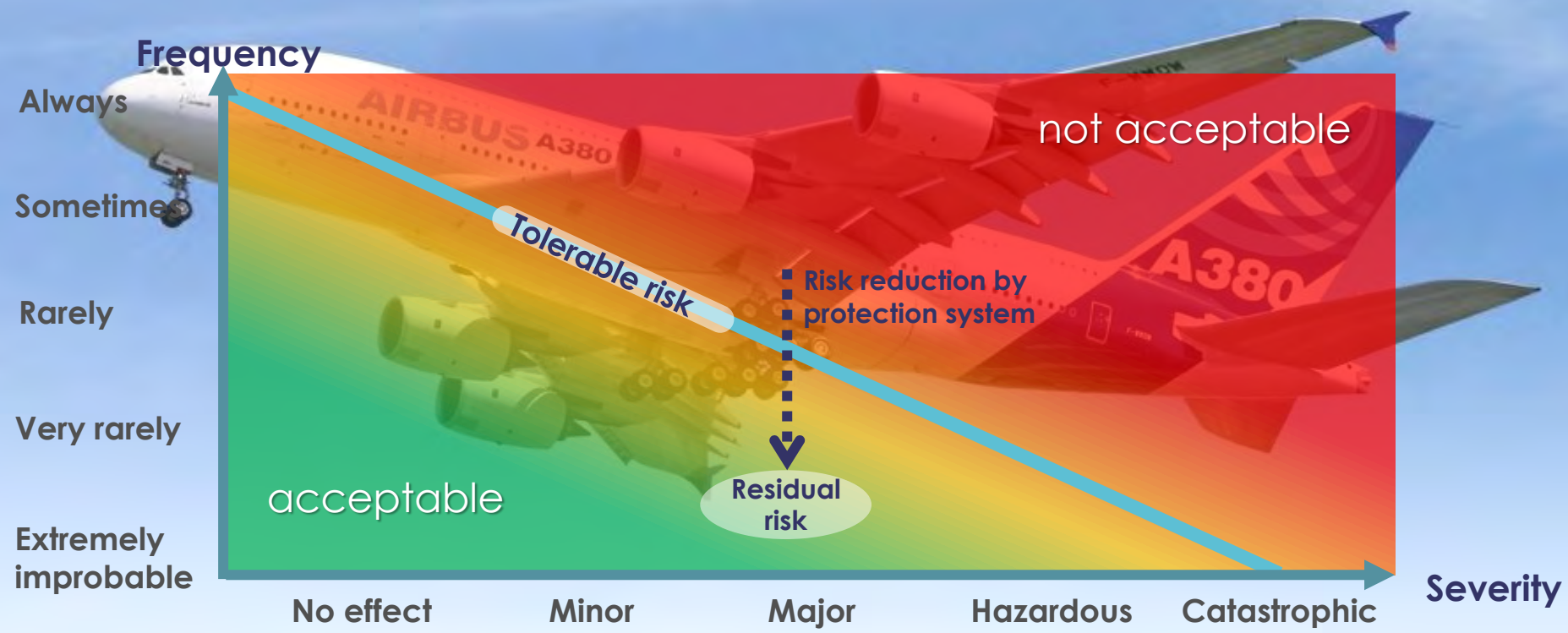
Multiple applications share the same processing resource

- Computing and networking element reduction costs
- About 30 avionic functions per Computing Unit(CPU)
- Aims to double/triple on new CPU modules

Challenging non-functional requirements

OPEN

Diversity of applications and safety levels



Design Assurance Level (DAL):

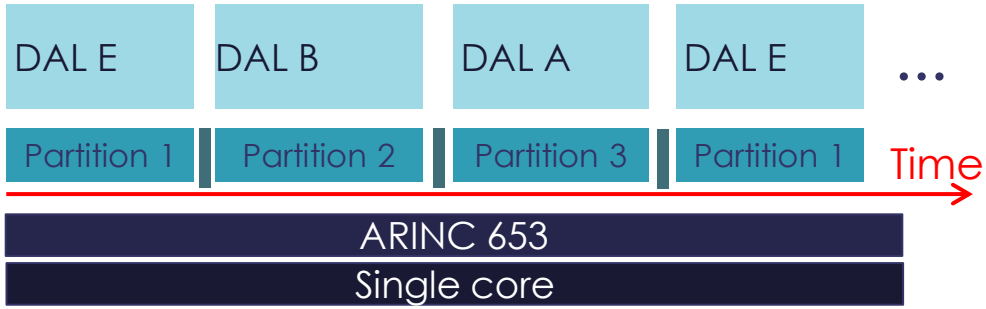
E	D	C	B	A/B	A
Passenger *IFE Communication*	**Cabin** *Cabin Light Water control Pressure*	**Maintenance** *Maintenance Logging*	**FMS** *Localization Trajectory Guidance*	**Critical** *Engine Control Flight Control Breaking*	**Cockpit** *Display Autopilot*

Software partitioning in critical systems

Composability and incremental certification ➔ evolution toward mixed-critical systems

Strict spatial and temporal partitioning (ARINC 653)

- Through MMU management
- Dedicated time slots

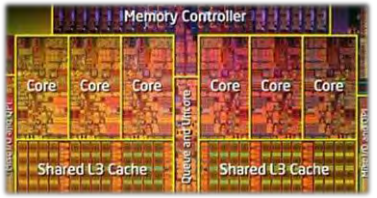


Concurrent execution with the guarantee that low critical applications DO NOT perturb high critical ones

Evolution of ES & dependability in the multi-core era

Use of COTS to benefit from the huge processing power, and low cost allowed by huge mainstream markets

The shift to COTS continues with the move to multi-cores processors

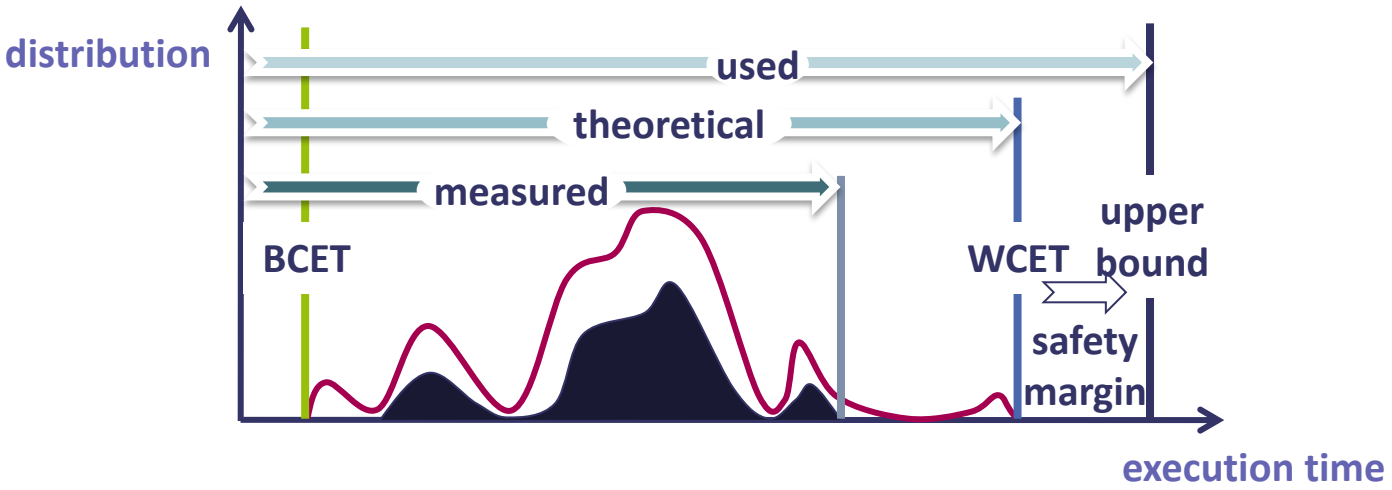


Certification of COTS HW impacted by increasing integration

Contents

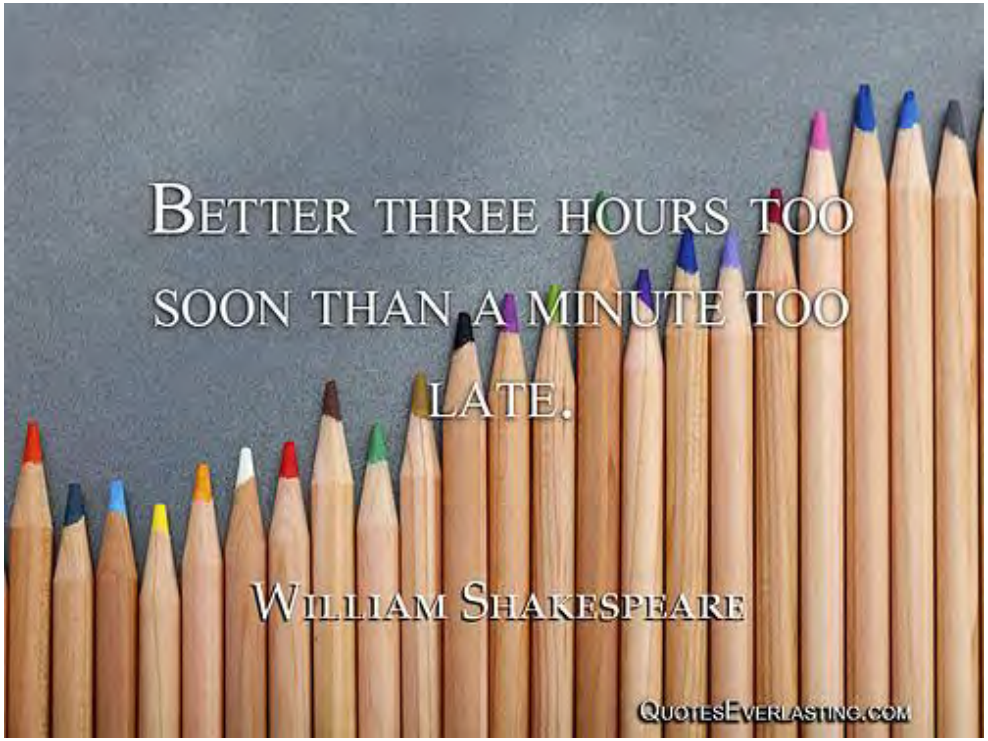
Predictability of complex architectures

Need of absolute guarantees on the timing (DO-178B) but design of commodity processors for average performance



Temporal behavior as important as **logical function** in ES

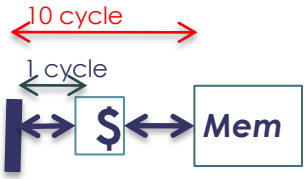
Safety margins



But the timing margins are increasing …

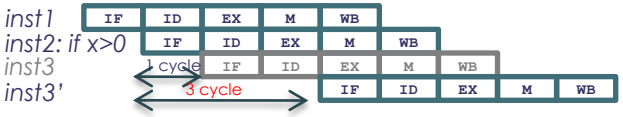
Shift to multi-core architectures

Cache



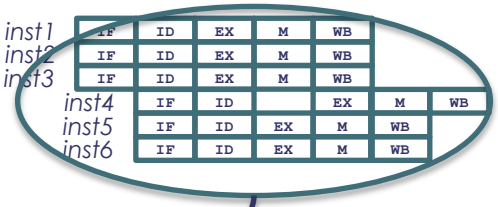
Avg. Time
WCET

Pipelines



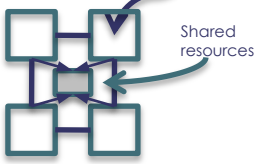
Avg. Time
WCET

Superscalar Out of Order



Avg. Time
WCET

Multicore



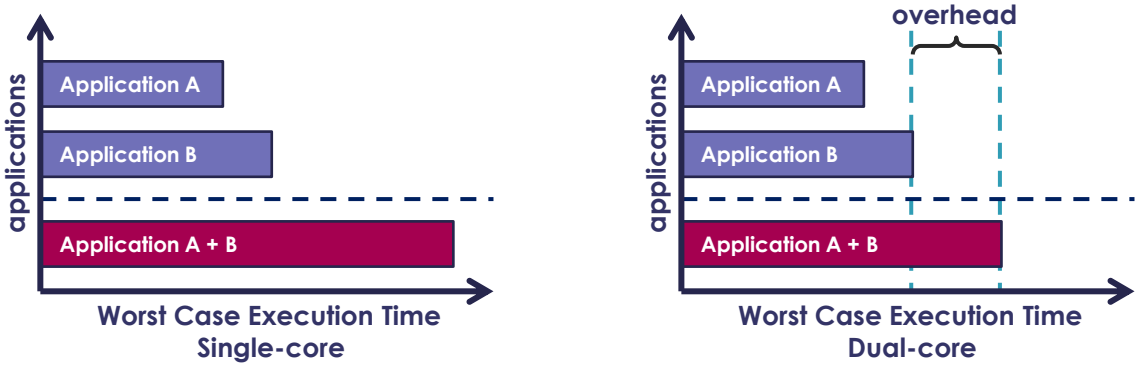
Avg. Time
WCET

Difficulties in certification: WCET analysis, deterministic behavior, complexity of architectures, thermal constraints

Isolation of applications on multi cores : new challenges

Isolation/partitioning on multi cores

- How to ensure sufficient hardware isolation without performance loss to ensure partitioning



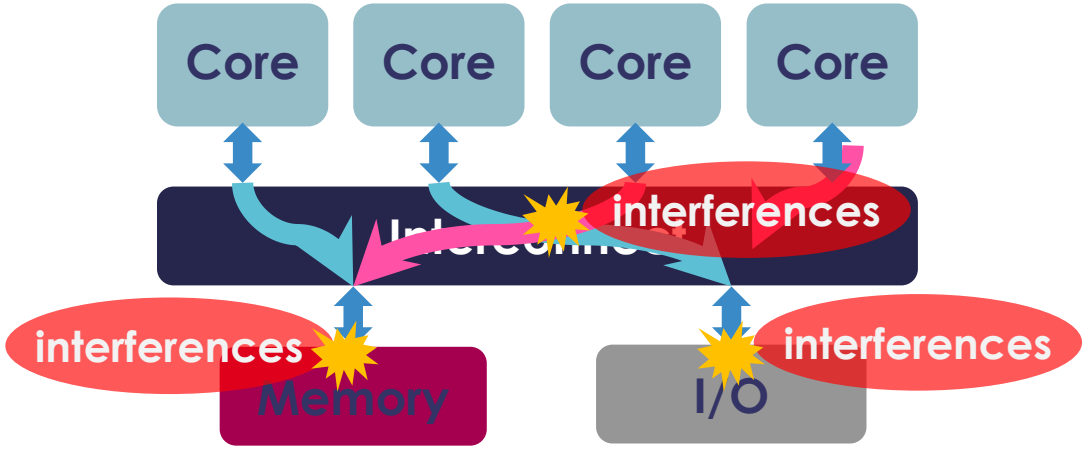
Existing Partitioning/Isolation principles are not scalable to multi cores

- Insufficient hardware segregation for shared resources (access to the shared memory)
- Time composability is not ensured

What are the interferences?

Interferences are unscheduled events that happen when two different cores simultaneously try:

- to access the same shared resource
- to access different resources but both need to go through a shared resource



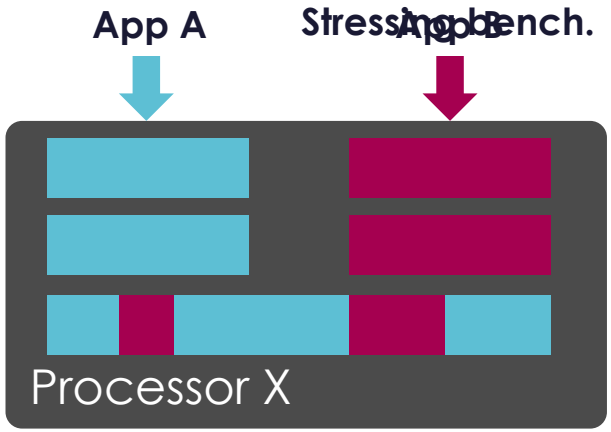
Interferences can have a significant impact and have to be evaluated carefully

Characterization of the interferences

How App A and App B will interfere when simultaneously execute on Processor X?

Idea: can we have a framework for

- characterizing the sensitivity of an application to shared resources
- evaluating interferences on a specific architecture

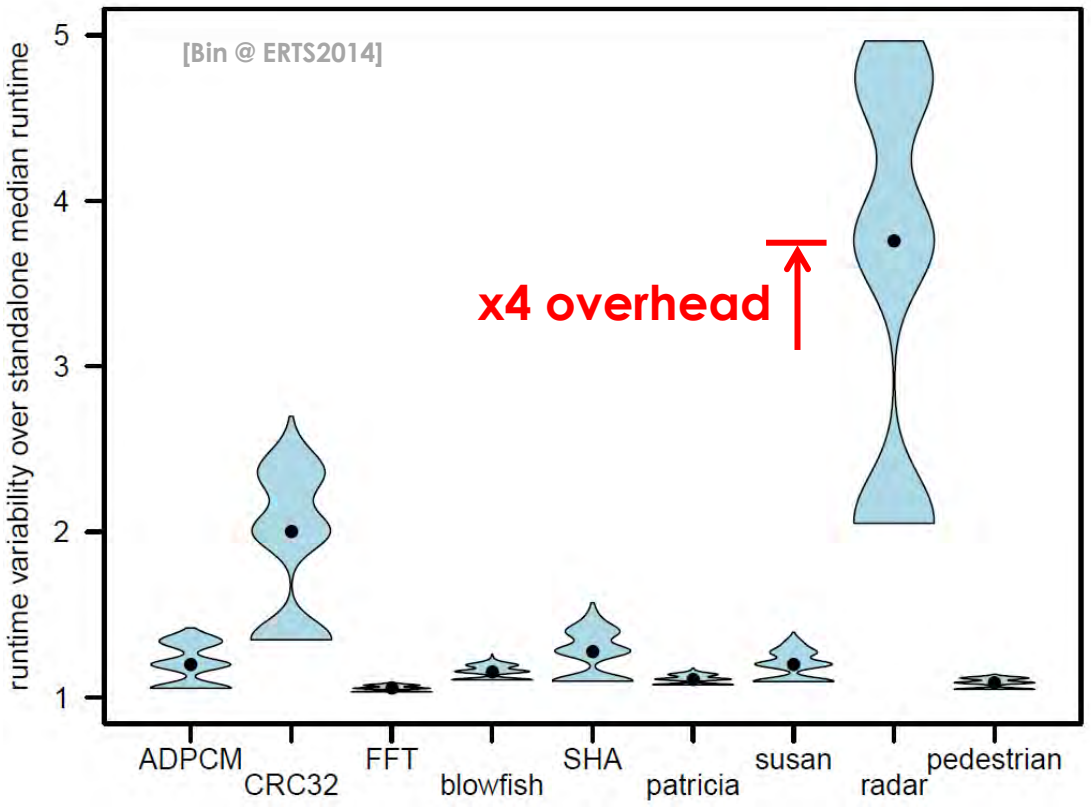


Use of worst stressing benchmarks to stress specific shared-resources

- Experimented on multithreaded and multi-core processors
- Written in assembly code for the target architecture maximize the stress on different architectural elements

Measurement of runtime variability

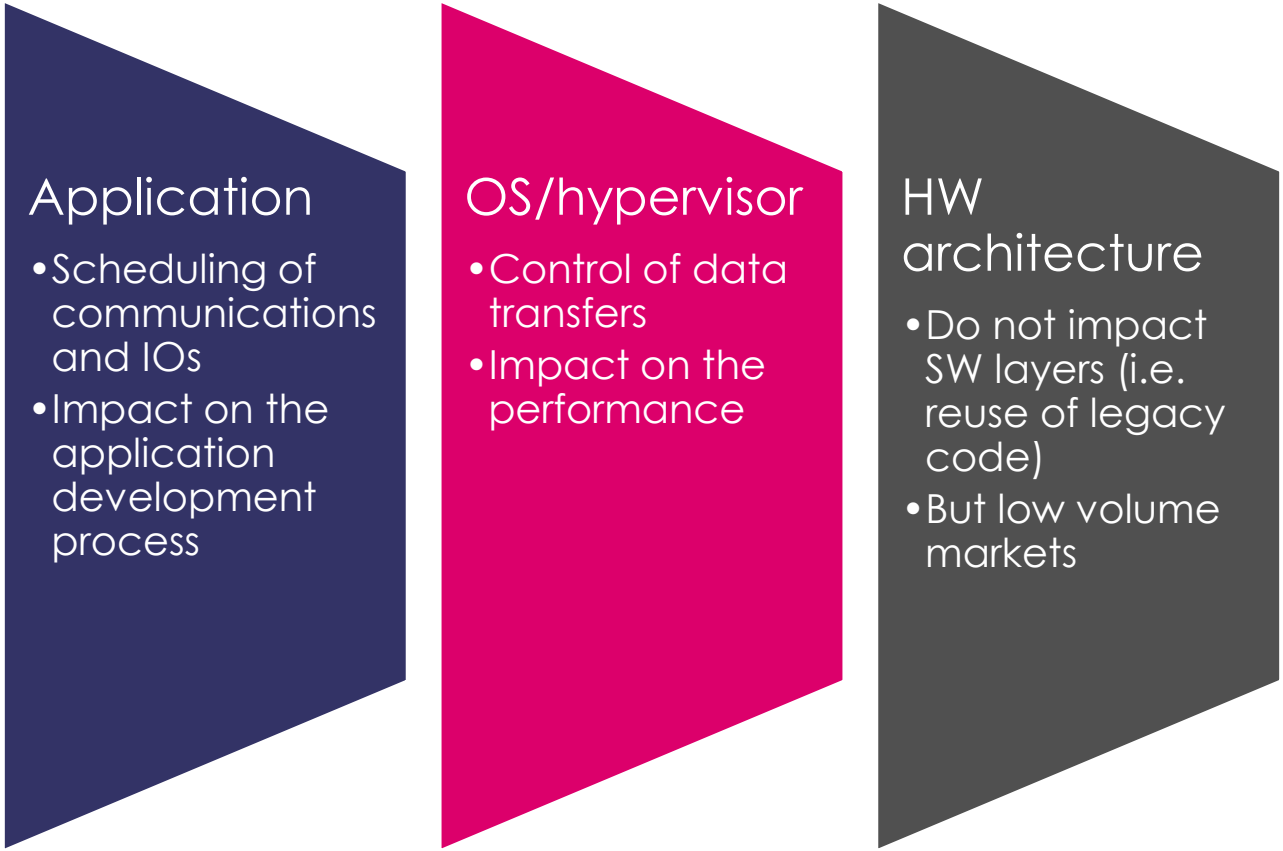
- Freescale P4080 platform
- 1 reference applications + 7 benchmarks stressing the shared memory path
- 600 iterations



Safety margins to guarantee proper operating are no longer sustainable

Rethinking isolation mechanisms in the multi-cores era …

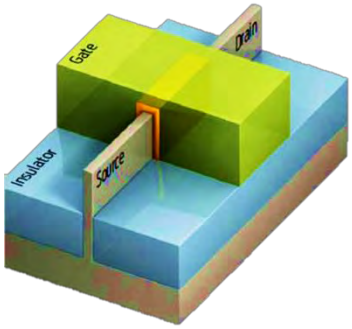
Need of new partitioning mechanisms respecting isolation to the hardware shared resources …



Contents

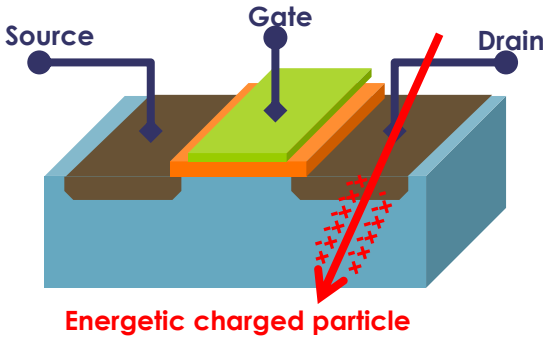
Technology scaling brings new challenges!

New technological process and process variability



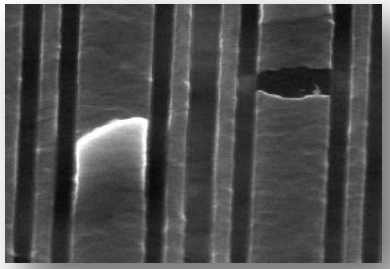
- New technologies: FD-SOI, FinFET, 3D integration, …
- Extensive process variations
- Infant mortality and intermittent faults

Increased susceptibility to environment



- Soft errors (SEU, MBU), EMC increased sensitivity
- Transient faults

Increased aging of devices

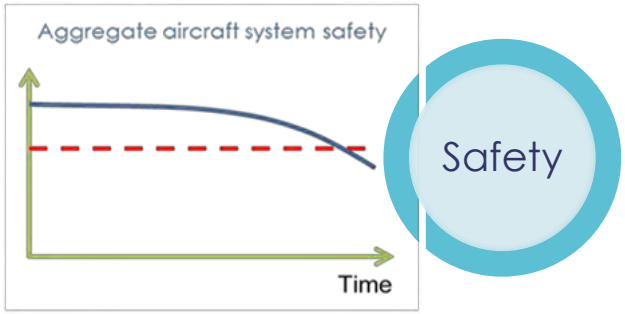


[IAlam, 2007]

- Early wear-out effects: BTI, TDDB, HCI, electromigration, …
- Permanent faults

Reduced lifetime/reliability of electronic components with the shrinking technology size

Impact on the design of critical embedded systems

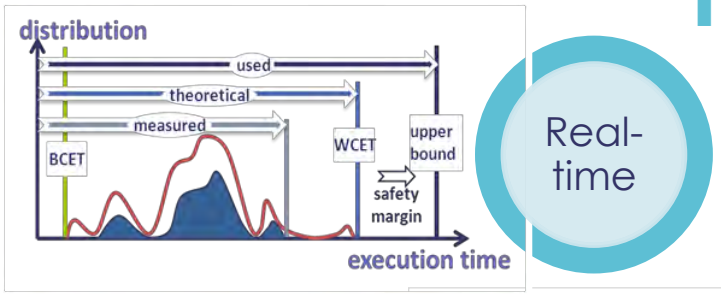


Operational reliability and safety impact

- Failure rate as low as 10-9 failure per hour
- Increasing soft-errors failure rates and MBU more prominent
- Current practices based on the hypothesis of constant failure rates

Transistor aging and service life guarantee

- Preventive maintenance and management of components obsolescence
- Reduction of the operating domains and components derating



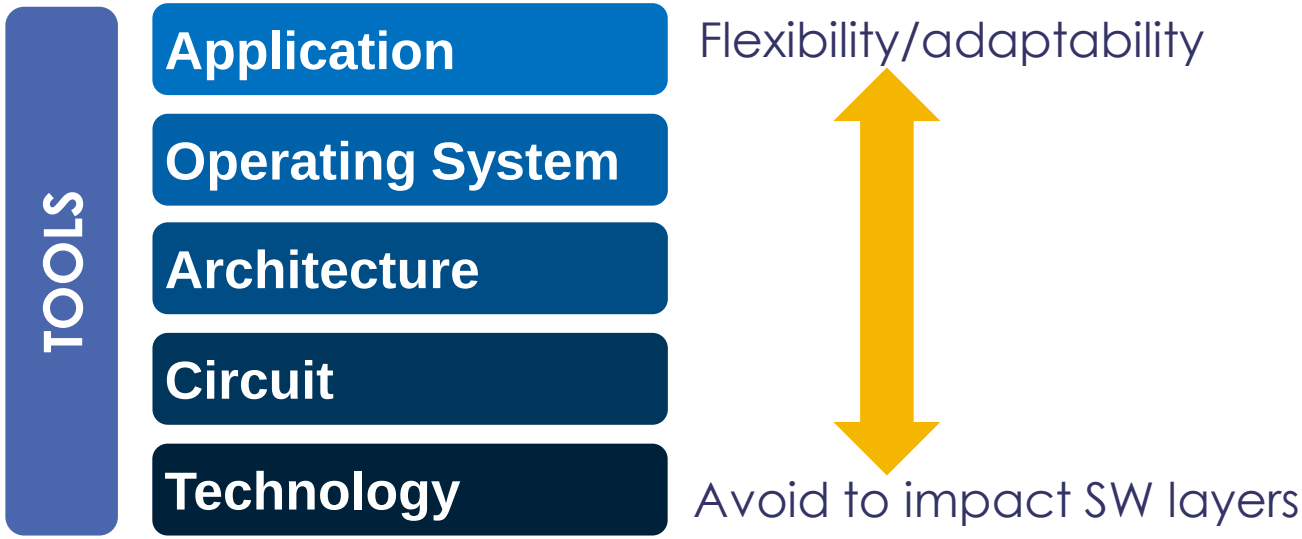
Critical safety = predictability + reliability (timing and behavior)

- Timing predictability of error detection & recovery mechanisms?
- Impact of faults on timing and applications partitioning

The need of new system design approaches

Challenge for future technologies: building "dependable" systems on top of unreliable components

- which will degrade and even fail during normal lifetime of the chip
- while providing guarantees on reliability, timing …



Reliability assessment is a critical point to support new methodologies

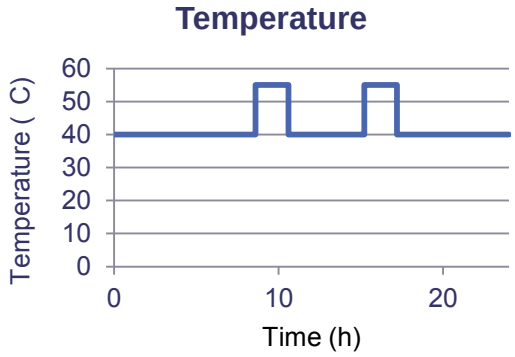
Operating conditions of critical embedded systems

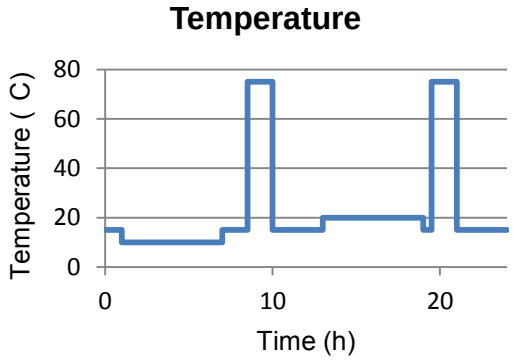
Harsh environmental conditions ➔ strong impact on the reliability

➢ Extreme temperature range (-55°C - +125°C), temperature cycling, 24/7 ON time, high level of radiations and vibrations

But operating conditions depends on the environment, the mission profile, the location of the equipment in the system

Realistic mission profile analysis to avoid worst-case approach

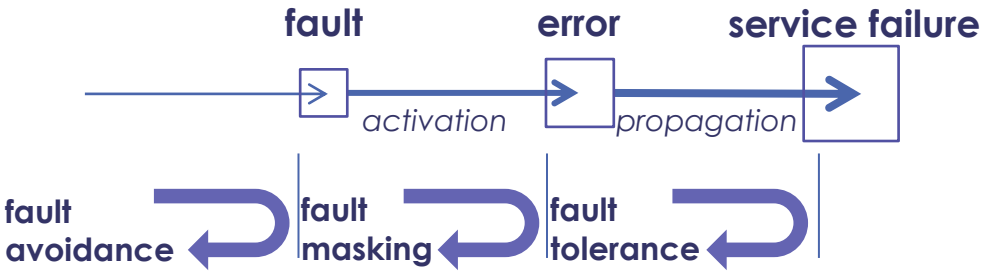




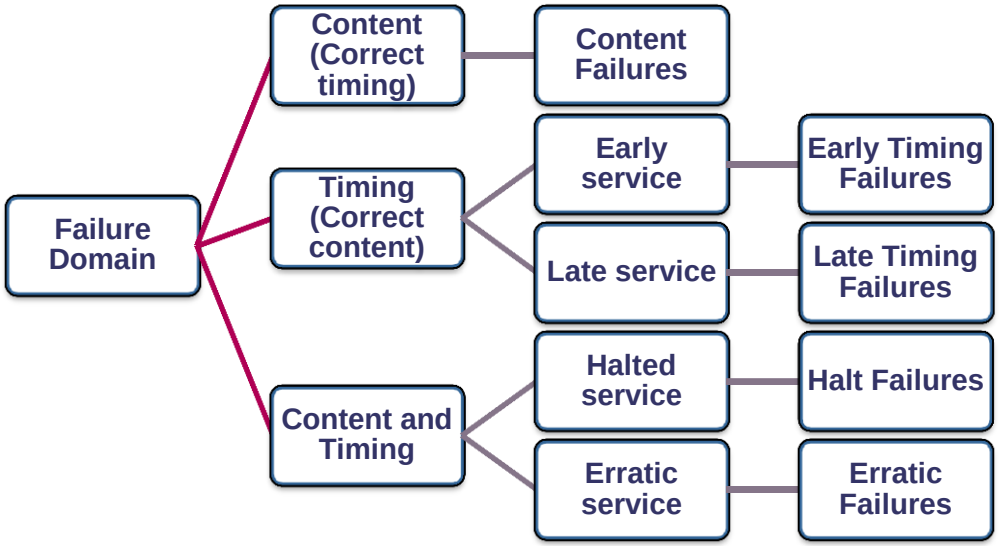
Critical systems subject to harsh environmental conditions that have to be accurately evaluated

The system failure modes

From faults to failures



Need to consider the different failure modes of the system



[Avizienis et al., 2004]

e.g. silent data corruption, detected unrecoverable errors

e.g. deadline miss

e.g. fatal HW trap, abnormal application exit, hangs

e.g. high OS activity

Cross-layer early reliability estimation

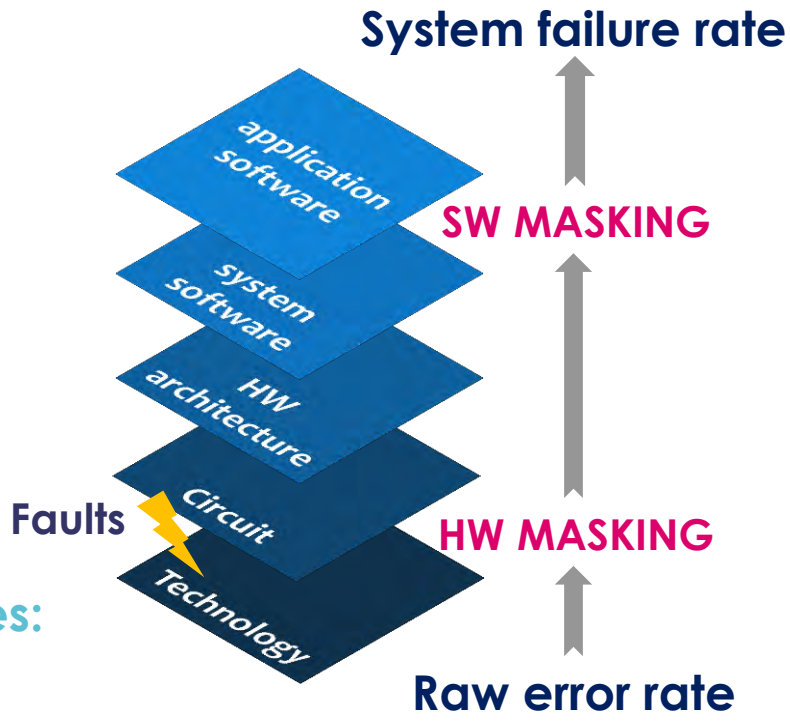
The CLERECO FP7 Collaboration Project

- http://www.clereco.eu

Cross-layer estimation:

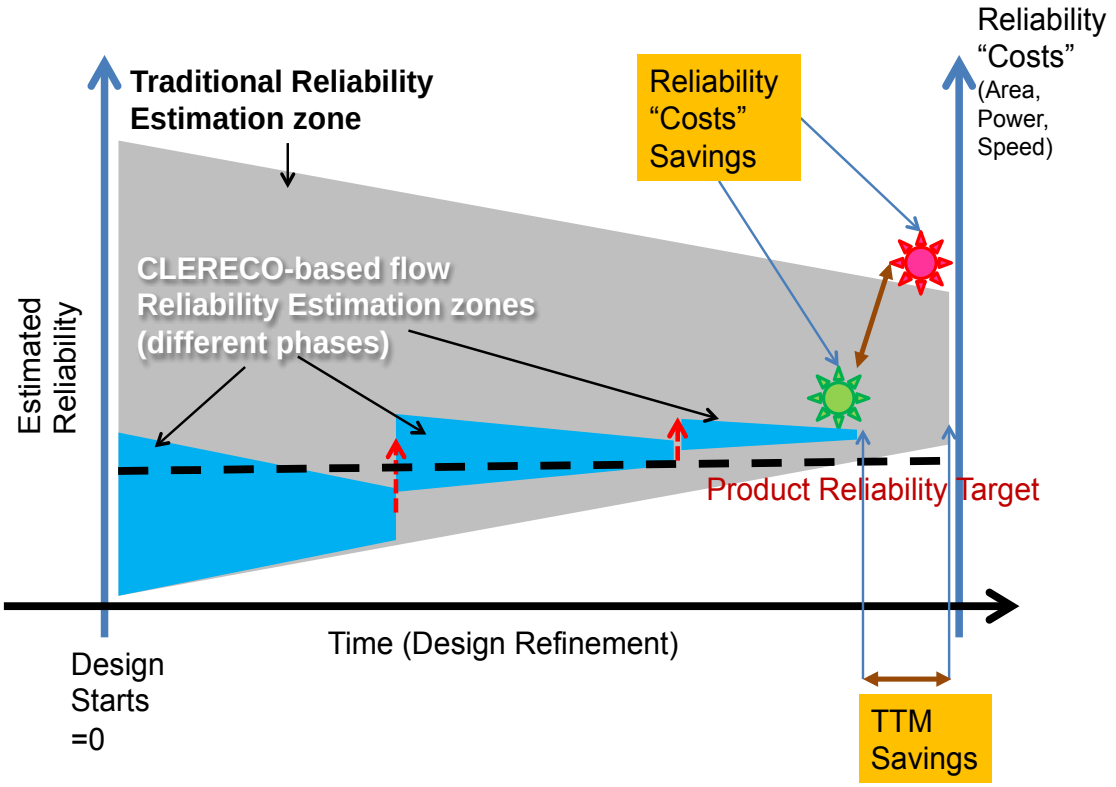
- Reliability is evaluated at system level
- Considering the hardware structure as well as the software stack (application, OS, …)

Standard reliability evaluation approaches: **massive** and **time-consuming** simulations and/or fault injection campaigns



The CLERECO objectives

- EARLY: reliability evaluation performed in every phase of the design cycle even when only high-level specifications are available
  - Reduction of area and design effort (dedicated to reliability)
  - Reduction of performance and energy lost for reliability



Masking effects and faults propagation

At SW level

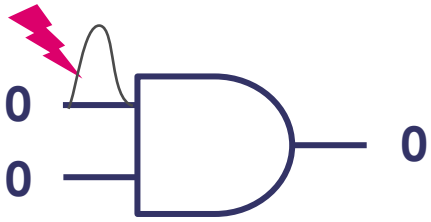
- Dead instruction
- Dead data bits
- Masking by an instruction

At micro-architecture level

- Masking by ECC mechanism
- Error in predictor structures
- Inactive hardware resources

At gate level

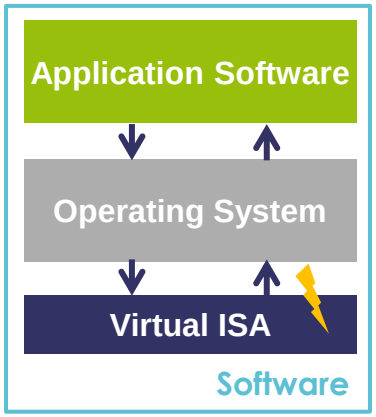
- Electrical masking
- Logical masking
- Latching-window masking



The CLERECO methodology in a nutshell

Software fault-injection

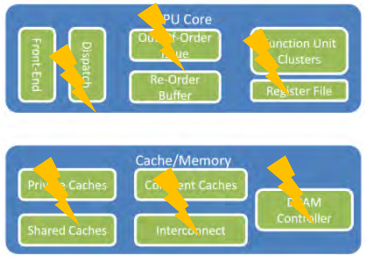
- At the intermediate code level in the LLVM framework
- Using a SW fault model defined at the ISA level



application + OS characterization

Micro-architectural simulator fault injection

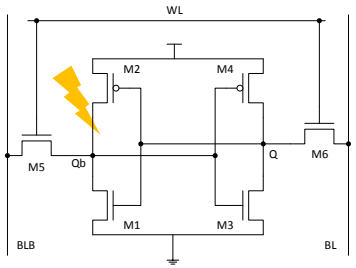
- Best trade-off between simulation time & accuracy
- Use of gem5 and MARSSx86 simulators
- Fault injection in all the major parts of the processor



SW fault rates

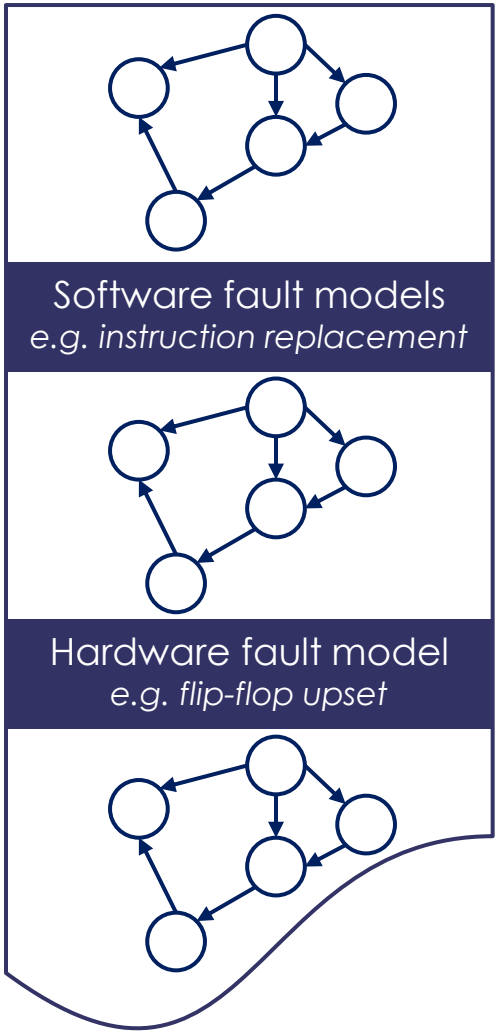
Technology characterization

- Evaluation of raw error rates for different technologies and different operating conditions
- Resorting to Spice simulations



raw error rates

System model based on Bayesian network



Contents

Conclusion

Avionics systems become more and more complex ➔ requires a shift to multi-cores processors

Real-time and safety requirements ➔ new development methods for a predictable and deterministic behavior

↘ reliability of CMOS technologies ➔ need of tools for an early and accurate reliability evaluation

The end …

Time predictability and reliability of multi-core processors for critical embedded systems

arnaud.grasset@thalesgroup.com