life.augmented

Functional Safety in Electronic Systems: Principles and Applications

Alessandro Bastoni
Functional Safety Expert
STMicroelectronics

Lesson #6

Formal safety architectures (IEC61508, ISO13849) + ISO26262 methods (ASIL decomposition). Mapping on automotive and robotics use cases

Summary:

- General concepts (HFT)
- IEC 61508 architectures (1oo1,1oo2,2oo2)
- ASIL decomposition, theory and examples
- Considerations on role of the software

life.augmented

About “diversity” concept

Diversity is defined in IEC61508-4 “different means of performing a required function“

Diversity can be required in some circumstances (e.g. to use the composition rule SC+1 with redundant channels)

Diversity is the key asset to fight common cause failures (and to cope with their potential incomplete analysis)

Diversity is possible in hardware and software.

For software, also timing diversity can be achieved (same computation executed in different moments) providing protection against soft errors impacting the CPU

Diversity is an asset but also a cost (design time, hardware resources, memory space, complexity, increased verification, potential availability issues).

Diversity can be used as mitigation factor for tool failures (tool assessment procedure for T3 tools)

life.augmented

Hardware Fault Tolerance (HFT)

Hardware Fault Tolerance of N means that N+1 is the minimum number of faults that could cause a loss of the safety function.

To determine HFT no considerations can be done on other measure controlling the effect of faults (like diagnostics)

Some faults can be excluded from the considerations, under specific rationale based on their probability.

➔HFT is basically a property of subsystems, not of components.

Remember: HFT is on of the entries to establish SFF targets for hw systems

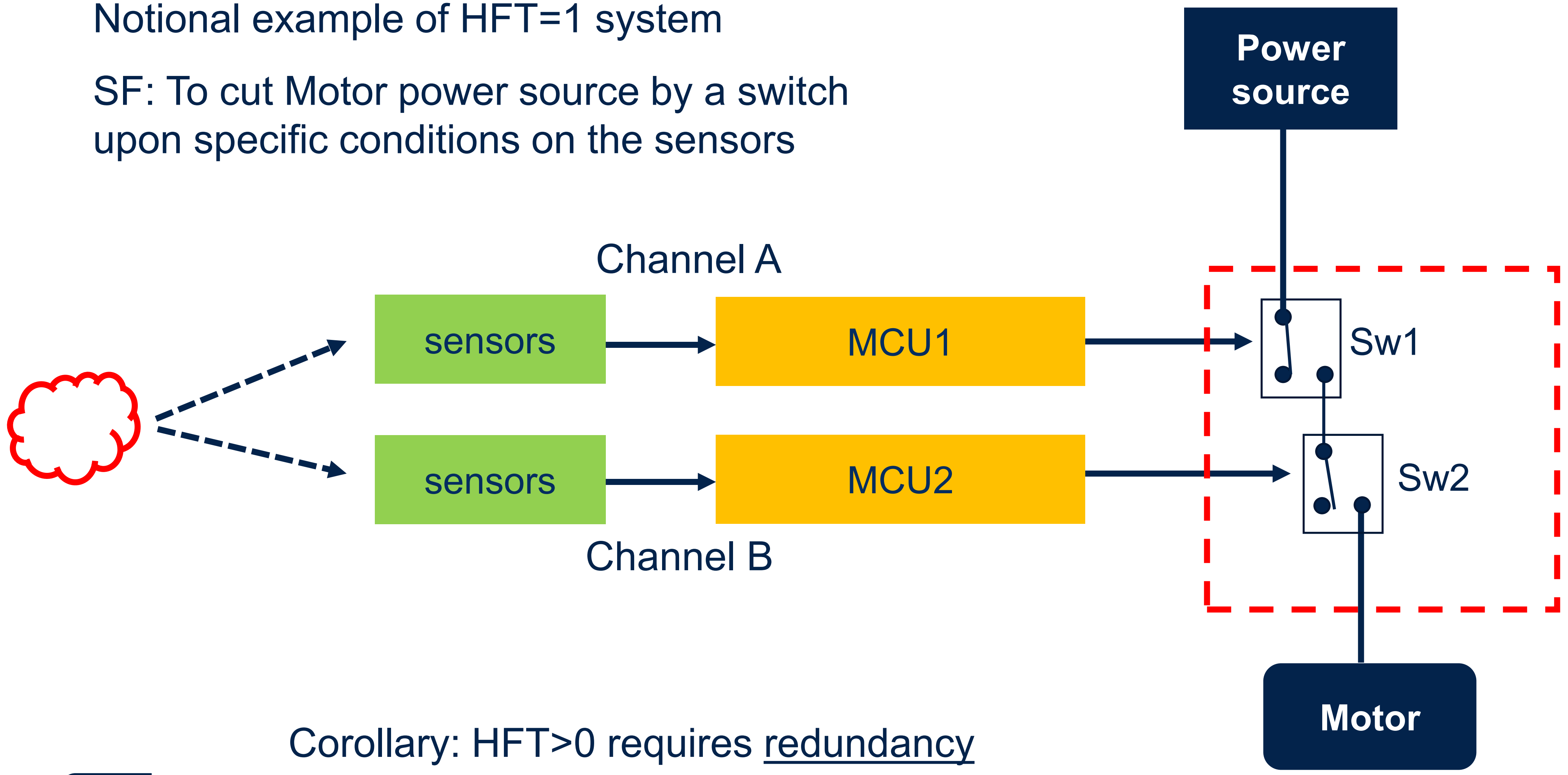
Note: IEC 61508 includes special architecture requirements for ICs with on-chip redundancy

life.augmented

Hardware Fault Tolerance (HFT)

Notional example of HFT=1 system

SF: To cut Motor power source by a switch upon specific conditions on the sensors



Corollary: HFT>0 requires redundancy at a certain level

What is a safety architecture

Generally speaking, a good definition for an ***architecture*** is “representation of the structure of a system that allows identification of building blocks, their boundaries and interfaces, and includes the allocation of requirements to these building blocks. This is akind of gereal definition.

There is no exact definition of “safety architecture” in current safety standards. That can be inferred, looking to an IEC 61508 framework, this way:

The safety architecture defines the structure and organization of the safety-related system, including the safety functions, safety mechanisms, safe state(s), and their allocation to hardware and software elements to achieve the required safety integrity.

life.augmented

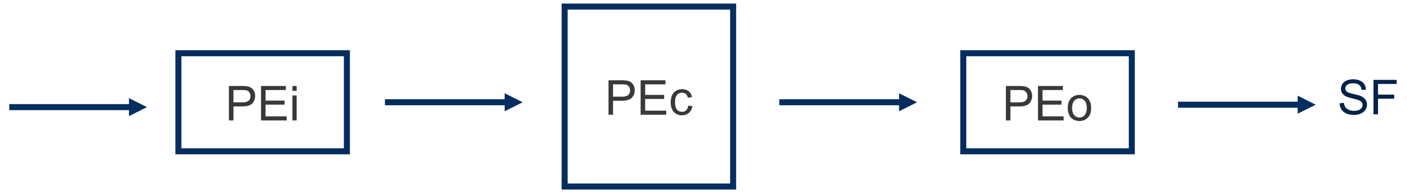
IEC 61508: 1oo1

1oo1 is the single channel architecture (PE = Processing Element)

It’s the most simple but also the most challenging / NO benefits from architecture scheme! Everything is solved inside the simple structure.

DC must be guaranteed by intrinsic diagnostic (HW/SW)

HFT = 0



Safe state transition could be complex

No diversity in hardware is possible → hardware SC to be reached at component level

Diversity in software poorly possible → software SC to be reached at component level

IEC 61508: 1oo1 - formulas

Low Demand mode:

$$PFD_G = (\lambda_{DU} + \lambda_{DD})t_{CE} \qquad t_{CE} = \frac{\lambda_{DU}}{\lambda_D}\left(\frac{T_1}{2} + MRT\right) + \frac{\lambda_{DD}}{\lambda_D} MTTR$$

High Demand/Continuous Mode:

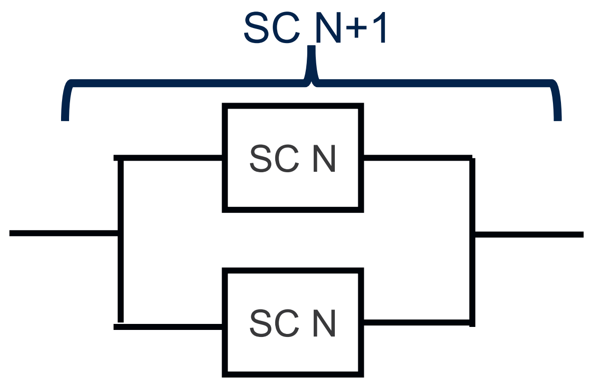
If it is assumed that the safety system puts the EUC into a safe state on detection of any failure, for a 1oo1 architecture the following is obtained

$$PFH_G = \lambda_{DU}$$

IEC 61508: 1oo2

Multiple types of diversity in hardware are possible → hardware SC can be easier thanks to N+N = N+1 rule

Diversity in software is possible → software SC is easier thanks to N+N = N+1 rule



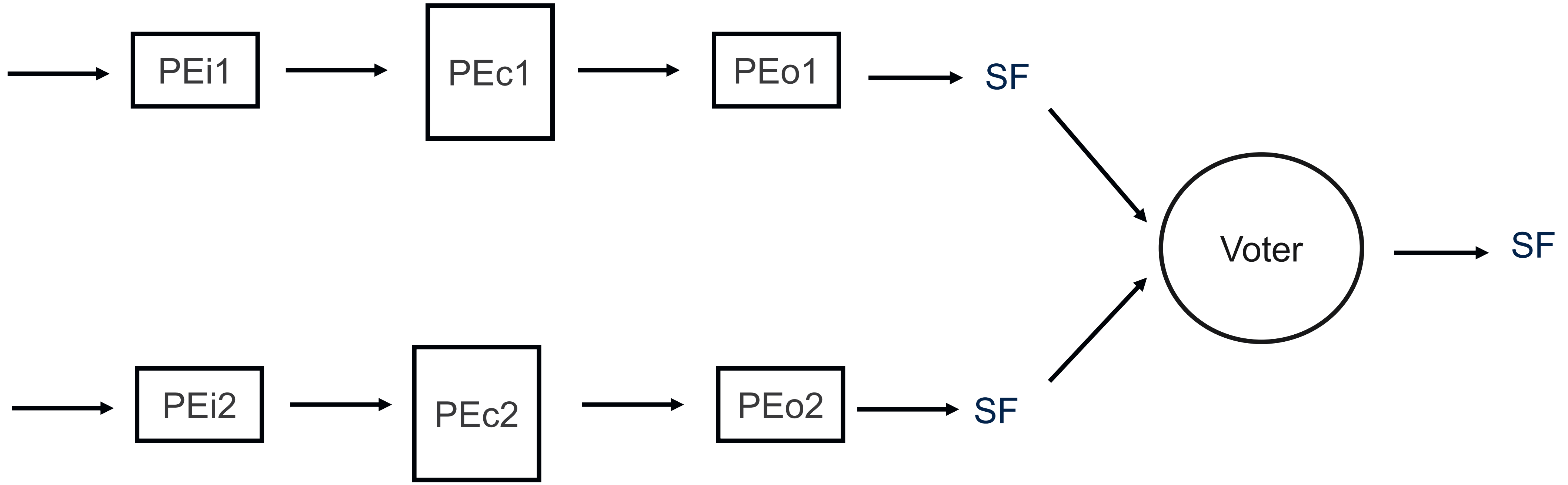
Attention:
- Independence between elements is required
- Allowed only once (no more cascading compositions)

BUT still some complications

Common cause failures to be explored (impact on-safety metrics!)

Voter must be HFT = 1 intrinsically

IEC 61508: 1oo2



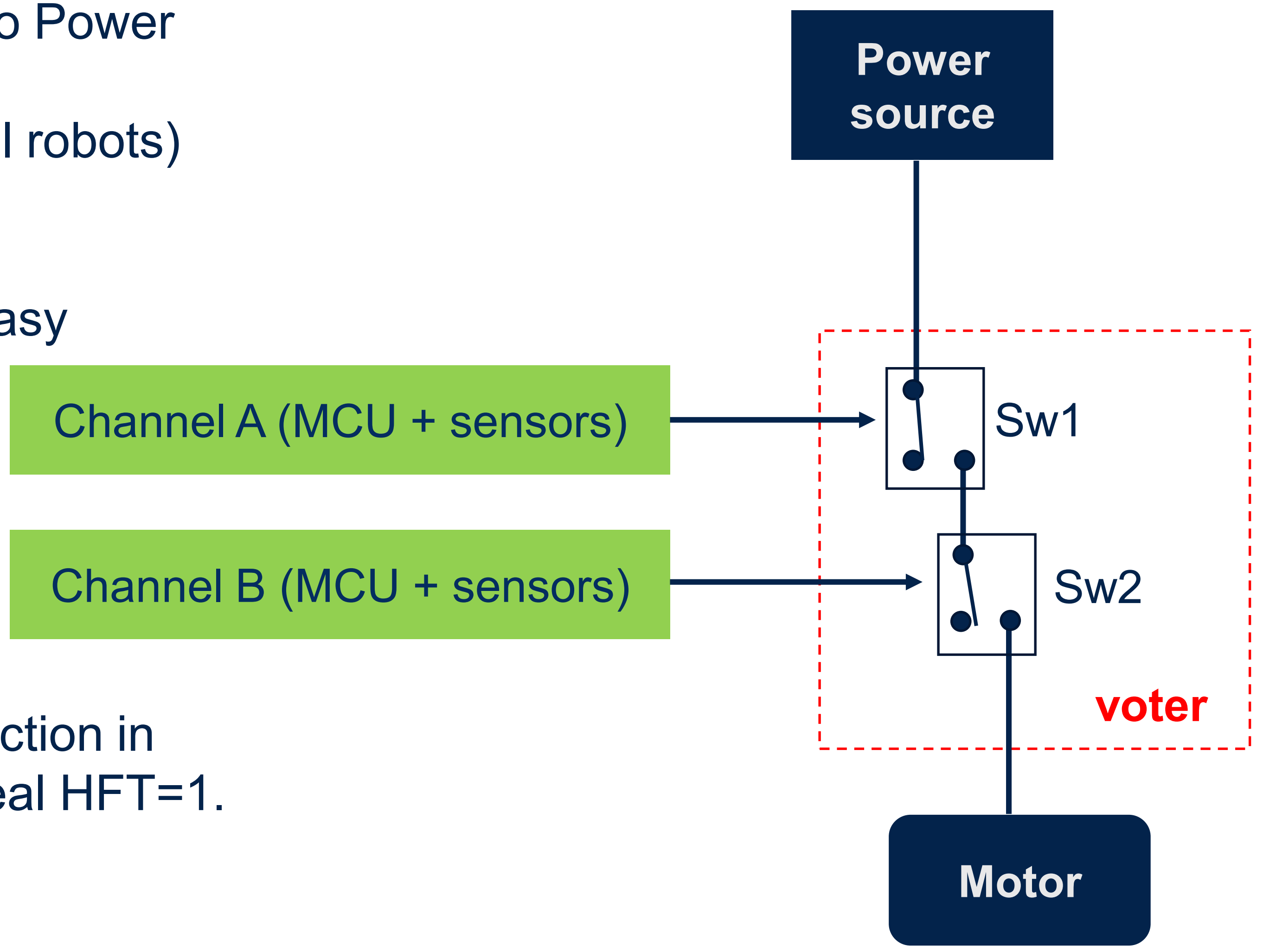
Note: be aware that cross-collaboration between the two MCUs may lead to increase of the penalty factors β and βD (common cause failures between MCUs

1oo2 notional example

Safety function: “Open Motor connection to Power source on specific input signals condition” (application e.g. active barrier for industrial robots)

Safe state: power connection OPEN

The nature of this safety function allows easy implementation for voter



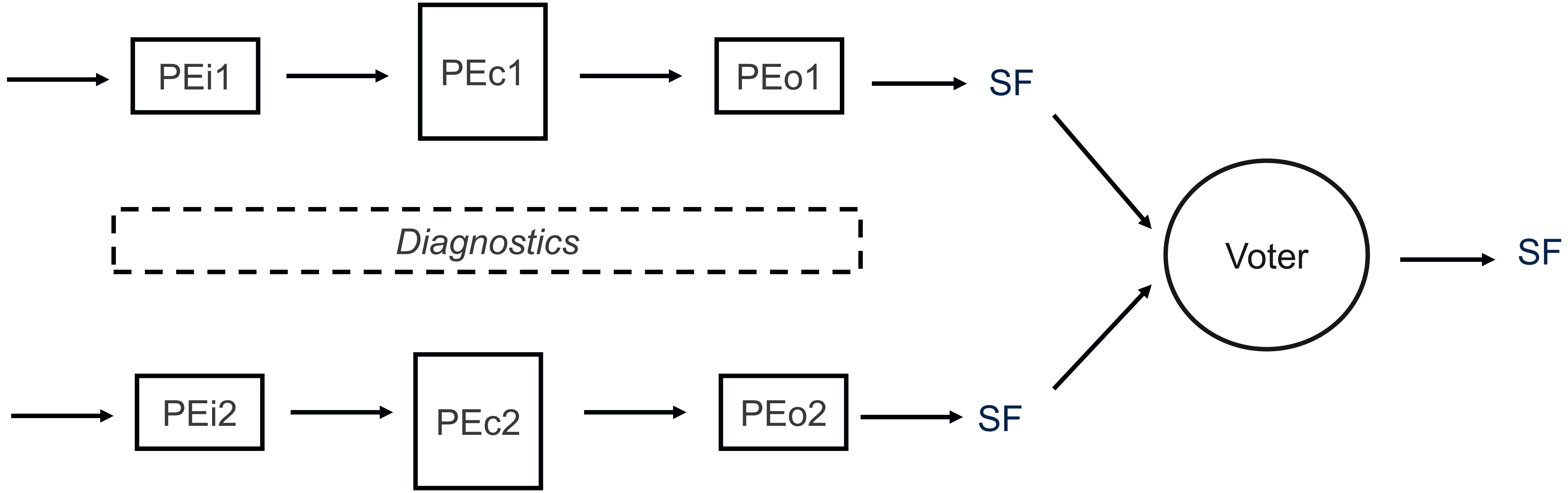
Note: switch structure forces the disconnection in case of failure for Voter local supply For real HFT=1. structure of the voter, switches need to be implemented in redundant way

IEC 61508: 2oo2

2oo2 consists of two channels connected in parallel so that both channels need to demand the safety function before it can take place.

DC must be guaranteed by intrinsic diagnostic for each channel. Diagnostics would report faults but not changing voting.

The scheme is HFT = 0.   PFH = 2λDU

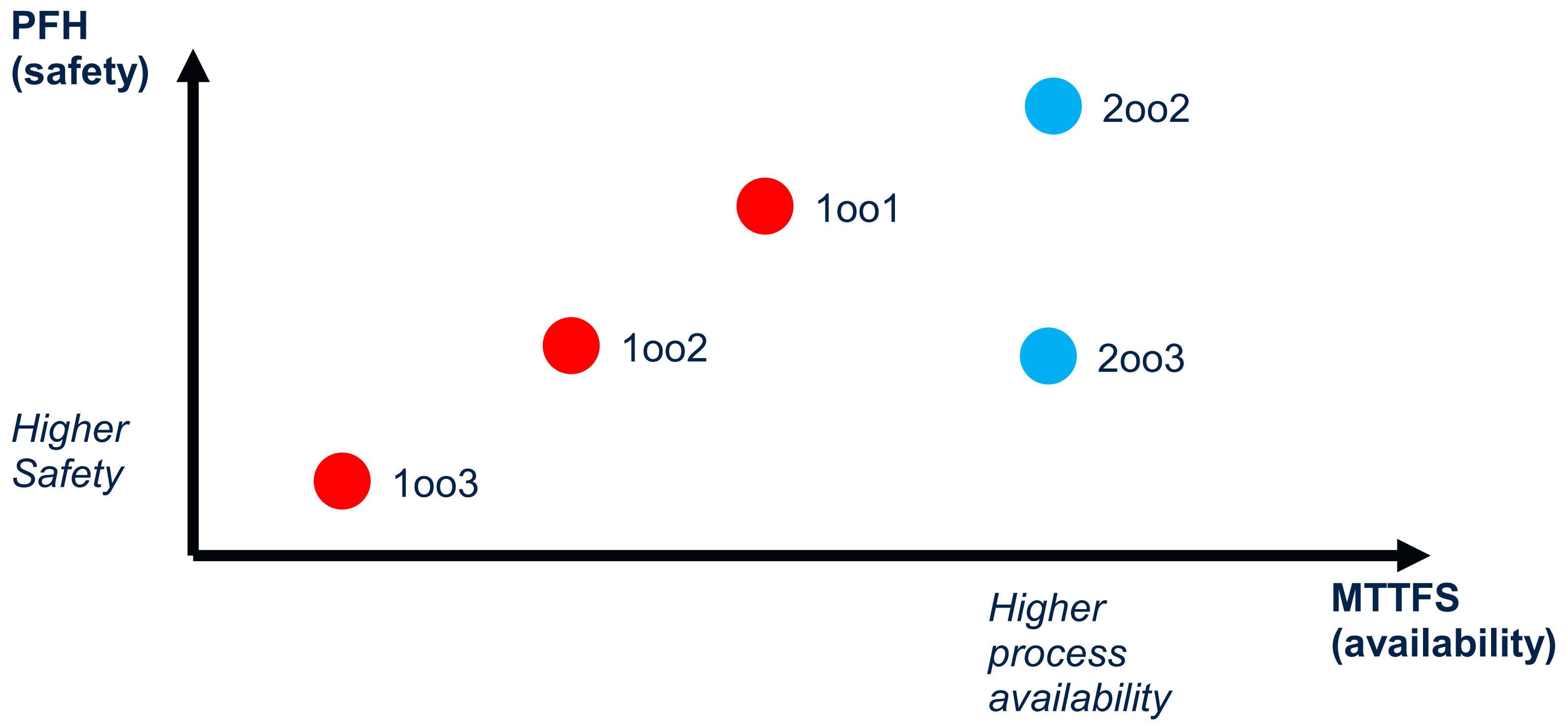


1oo2 vs 2oo2 vs 2oo3

1oo2: priority is on safety (lower PFH)

2oo2: priority is on availability (higher PFH but less false positive safe state transitions)

2oo3: keeps the advantages of two above schemes, at higher cost



Notes: MTFS = Mean Time to Fail Spurious; above graph is qualitative only

ASIL decomposition

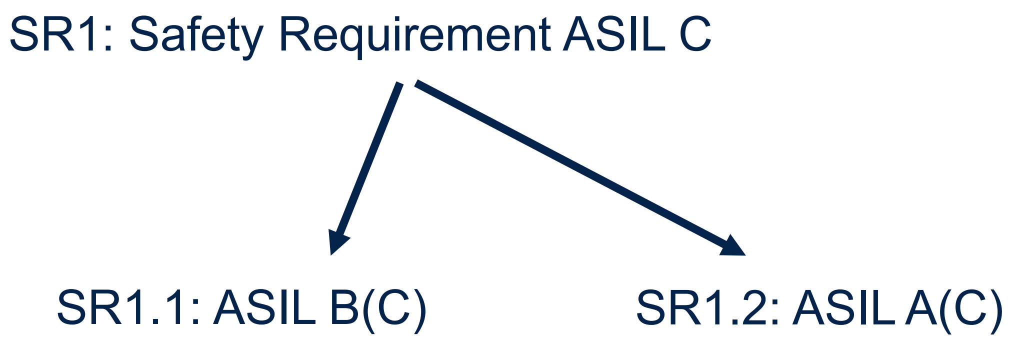
ASIL decomposition is described in ISO 26262-9:2018 clause 5.

The main purpose of implementing ASIL decomposition is to lower the ASIL level.

A safety requirement is decomposed into redundant safety requirements, which are then allocated to sufficiently independent elements.

The ASIL decomposition process must adhere to the scheme given by Table 1 of ISO 26262-9:2018.

D	**ASIL D(D) + QM(D)**
	ASIL C(D) + ASIL A(D)
	ASIL B(D) + ASIL B(D)
C	**ASIL C(C) + QM(C)**
	ASIL B(C) + ASIL A(C)
B	**ASIL B(B) + QM(B)**
	ASIL A(B) + ASIL A(B)
A	**ASIL A(A) + QM(A)**



ASIL decomposition – general rules

Homogenous redundancy (e.g., by duplicated device or software) is, in general, not sufficient for reducing the ASIL due to the lack of independence between the elements

Safety goals cannot be decomposed

The decomposed requirements must independently meet the original requirement (definition of redundancy)

ASIL decomposition may be applied more than once (!) (big difference with combination of elements in IEC 61508)

What is the bet: the decomposed safety requirements sre simpler and/orcheaper in terms of implementation. So: ASIL decomposition is not always a winning solution…

life.augmented

ASIL decomposition – general rules

(INVARIANTS)

The resulting decomposed ASIL must be expanded with the original ASIL before decomposition, e.g., ASIL A(C)

The functional safety assessment of the item is defined by the original ASIL

The independence of decomposed elements shall be evaluated by analysis of dependent failures

Requirements for test and integration are defined by the ASIL of the relevant integration level

Target values for HW-metrics are defined by the original ASIL at the item level

life.augmented

Safety related software vs non-safety related

In articulated architectures, software is often part of the safety concept.

Coexistence between safety related software and non-safety related software may happen in microcontroller-based systems (often for cost reasons e.g. reuse of open source sw for non-safety related tasks).

Main issue is separation i.e. to guarantee that non-safety related software can’t interfere with the safety function:

- Spatial interference: memory overwrite, peripherals access etc
- Temporal interference: determinism violation, resource occupation, jitter, interrupt masking etc.

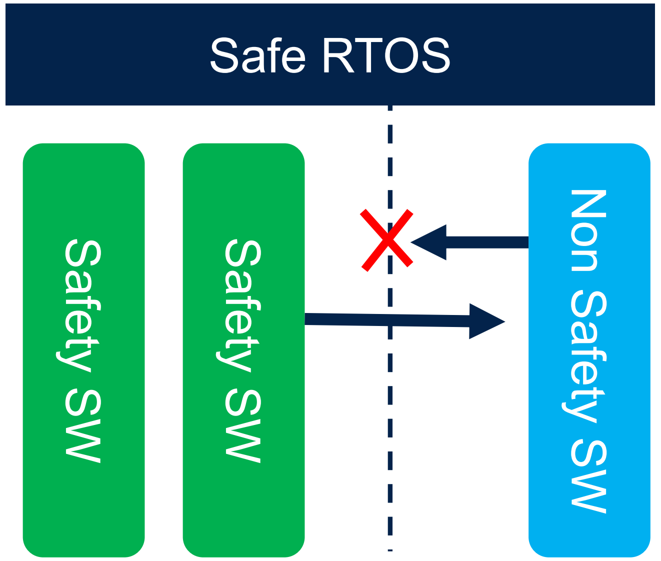
Hardware segregation can be difficult as typical microcontroller don‘t have complex MMUs (often just basic MPU is available)

Safety related software vs non-safety related

Possible vectors for coexistence issue solution:

Non-safety related software is simple: to repatriated it inside the certified V-model for application software could be viable

Non-safety software is complex: software segregation by a Safe RTOS could guarantee a) isolation of the potential spatial interferences b) determinism of the safety software regardless the non-safety one. Overhead is cost and avauilability of the Safe RTOS.



Backup slides

life.augmented

IEC 61508: 1oo2 - formulas

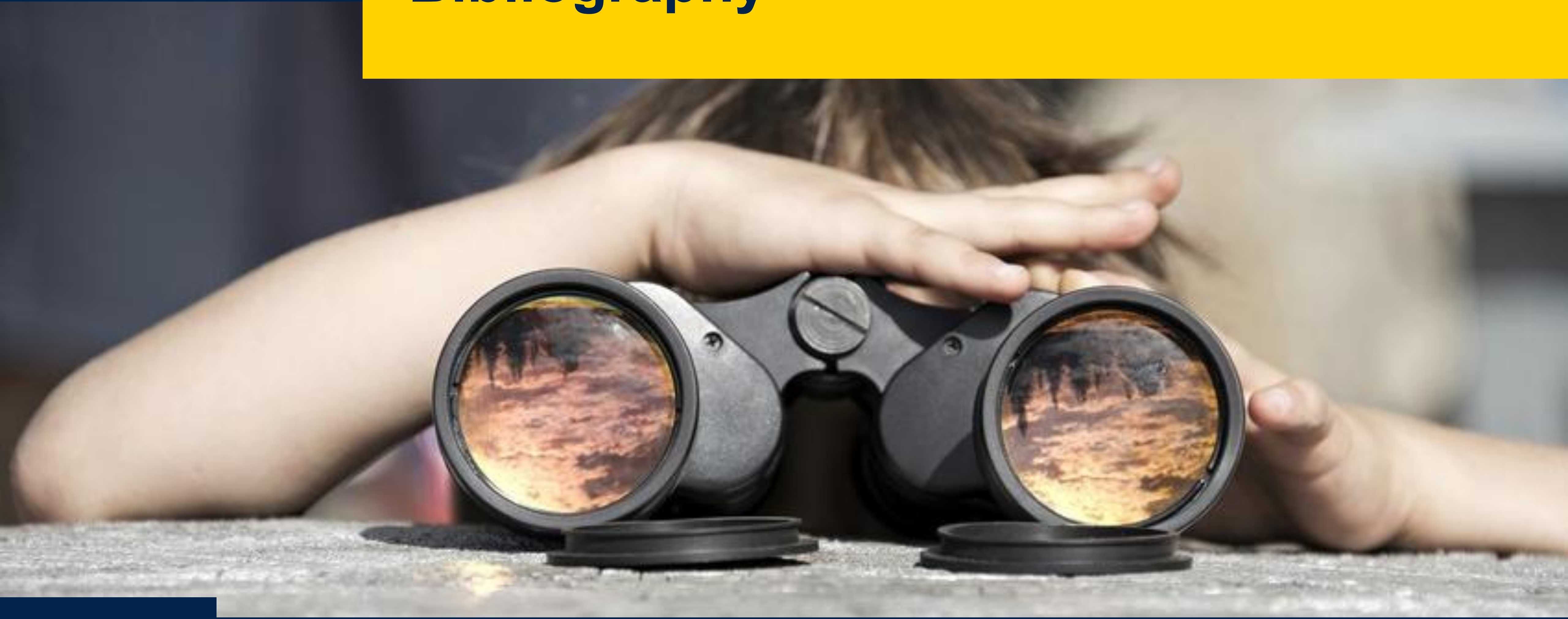
Low Demand Mode:

$$PFD_G = 2\left((1-\beta_D)\lambda_{DD} + (1-\beta)\lambda_{DU}\right)^2 t_{CE} t_{GE} + \beta_D \lambda_{DD} MTTR + \beta\lambda_{DU}\left(\frac{T_1}{2} + MRT\right)$$

High Demand/Continuous Mode:

$$PFH_G = 2\left((1-\beta_D)\lambda_{DD} + (1-\beta)\lambda_{DU}\right)(1-\beta)\lambda_{DU} t_{CE} + \beta\lambda_{DU}$$

Bibliography



life.augmented

Reference documents 1/2

[R1]: Microelectronics Reliability:Physics-of-Failure Based Modeling and Lifetime Evaluation - Jet Propulsion LaboratoryCalifornia Institute of Technology Pasadena, California

[R2]: : Semiconductor Reliability Handbook – Renesas Electronics, Rev.2.50 Jan. 2017

[R3]: ExoMars 2016 - Schiaparelli Anomaly Inquiry (ESA) downloaded from https://exploration.esa.int/web/mars/-/59176-exomars-2016-schiaparelli-anomaly-inquiry

[R4]: Fault Tree Handbook with Aerospace Applications - NASA Office of Safety and Mission Assurance, V 1.1 , 2002

[R5]: open FTA software can be found on the wed, e.g. https://www.fault-tree-analysis.com/free-fault-tree-analysis-software, or check for OpenFTA download

life.augmented

Reference documents 2/2

[R6]:Safety Manual for TMS570LS31x and TMS570LS21x Hercules™ ARM®-Based Safety Critical Microcontrollers

[R7]: UM2331- STM32H7 singlecore series safety manual STMicroelectronics – from https://www.st.com/en/embedded-software/x-cube-stl.html#documentation

[R8]: Safety Manual for TPS65919-Q1 Power Management Unit (PMU)

life.augmented

Thank you

life.augmented