



CPLD Developement & Nuclear Safety (NS) Constraints

October 6 - 2016

SUMMARY



- NEXEYA
- EQUIPEMENT ARCHITECTURE
- DEVELOPEMENT
- VALIDATION – TEST – MAINTENANCE
- TEST BENCH
- REX (return of experience)



- COTS...





NEXEYA

- **Staff > 1 000 on 3 main industrial sites (Paris, Bordeaux, Toulouse)**
- **Sales 2015 : 120M€**
- **5 Business Lines**
 - **Testing and Integration Solutions (TIS) : system for validation, test and integration of embedded electronic equipment.**
 - **Mission Management Systems (MMS) : systems embedded on aircrafts or ships:**
 - EDF project: GMPP pump supervision,
 - **Power Conversion (PC) : energy conversion and motor control.**
 - **Space Systems (SS) : spacecraft and launcher onboard equipment, and the ground station part.**
 - **Customer Support (CS) : system exploitation technical assistance, operational maintenance and obsolescence treatment (revamping...)**

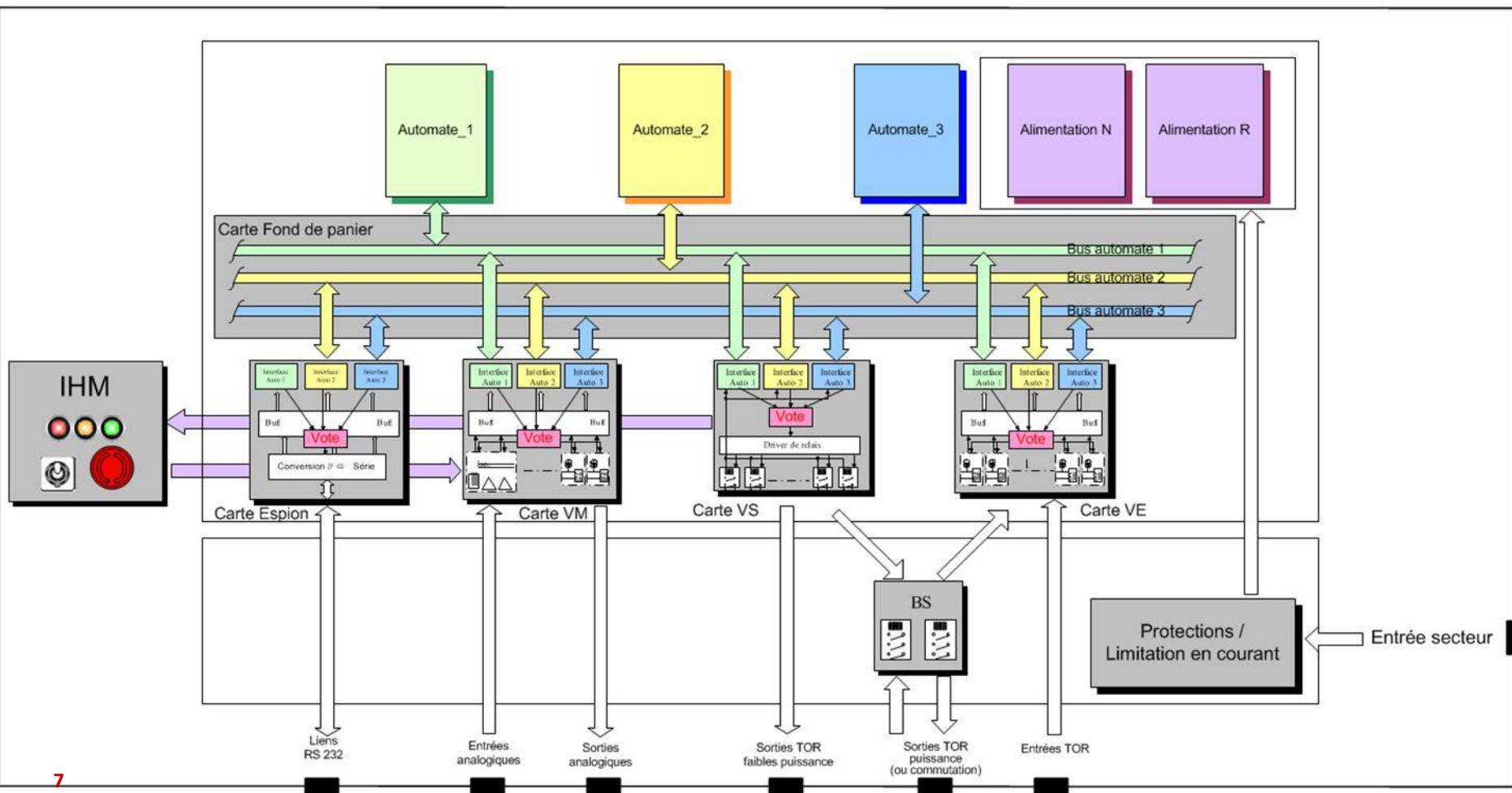


ARCHITECTURE

- **Safety machine functioning 24h/24 and 365d/year**
 - no software
 - no FPGA (use of CPLD of Xilinx manufacturer, CoolRunner-2 family)
 - More than 350 I/O in small volume (19"/ 7U rack) :
 - VE : Discrete in
 - VS : Discrete out
 - VM : Analogue in or Resistive in (few VM analog out)
 - **Military environment**
- **Nuclear safety constraints**
 - **Standard applied: none BUT:**
 - NS analysis according ASTRIUM process validated by the DGA NS authorities
 - « purely » NS requirements > 50 → more than 100 fault trees
 - **Some Potentially Dangerous Events with occurrence probability < 10⁻⁹**
 - ~ SIL3 / DO254 DAL B

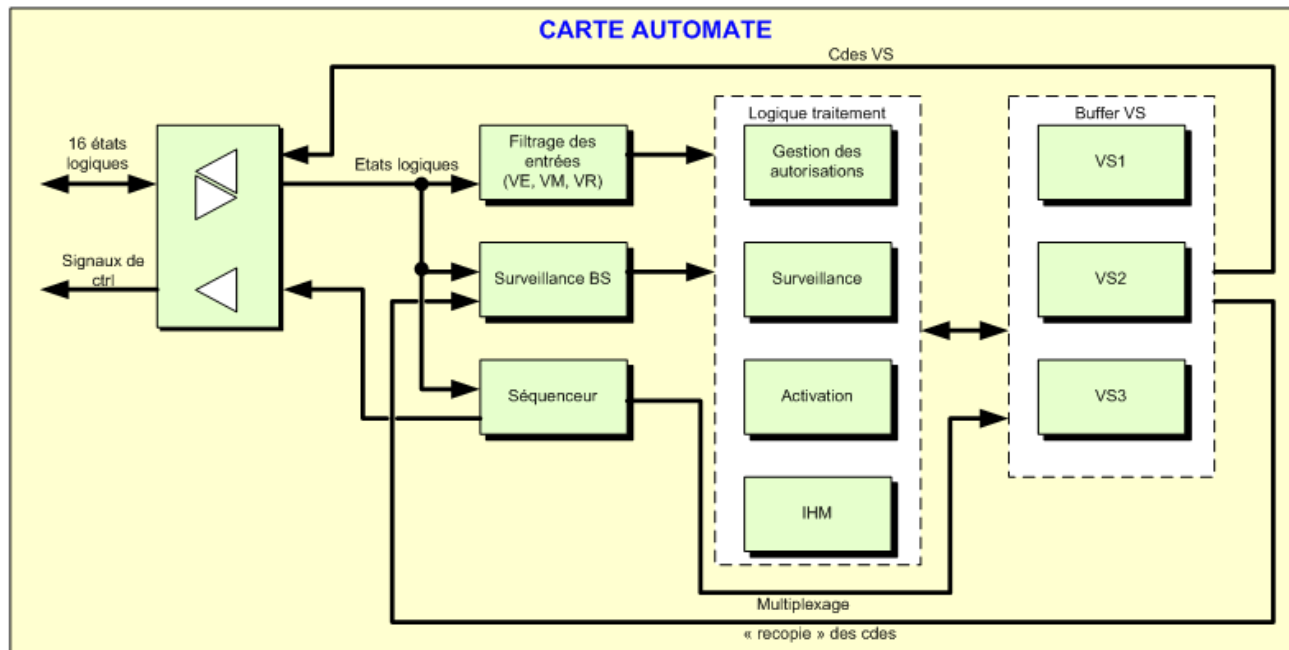
SAFETY MACHINE ARCHITECTURE

■ Triplicated « Core » with majority voting circuit (2oo3)



« Core » ARCHITECTURE

- the « core » function is duplicated on 3 boards
- The logic functionality is identical on each board
- Functional separation with logic blocs (21 CPLDs per board)
- BIST : periodic autotest of the IO boards via the “Automate” board (Automate => automatic machine)

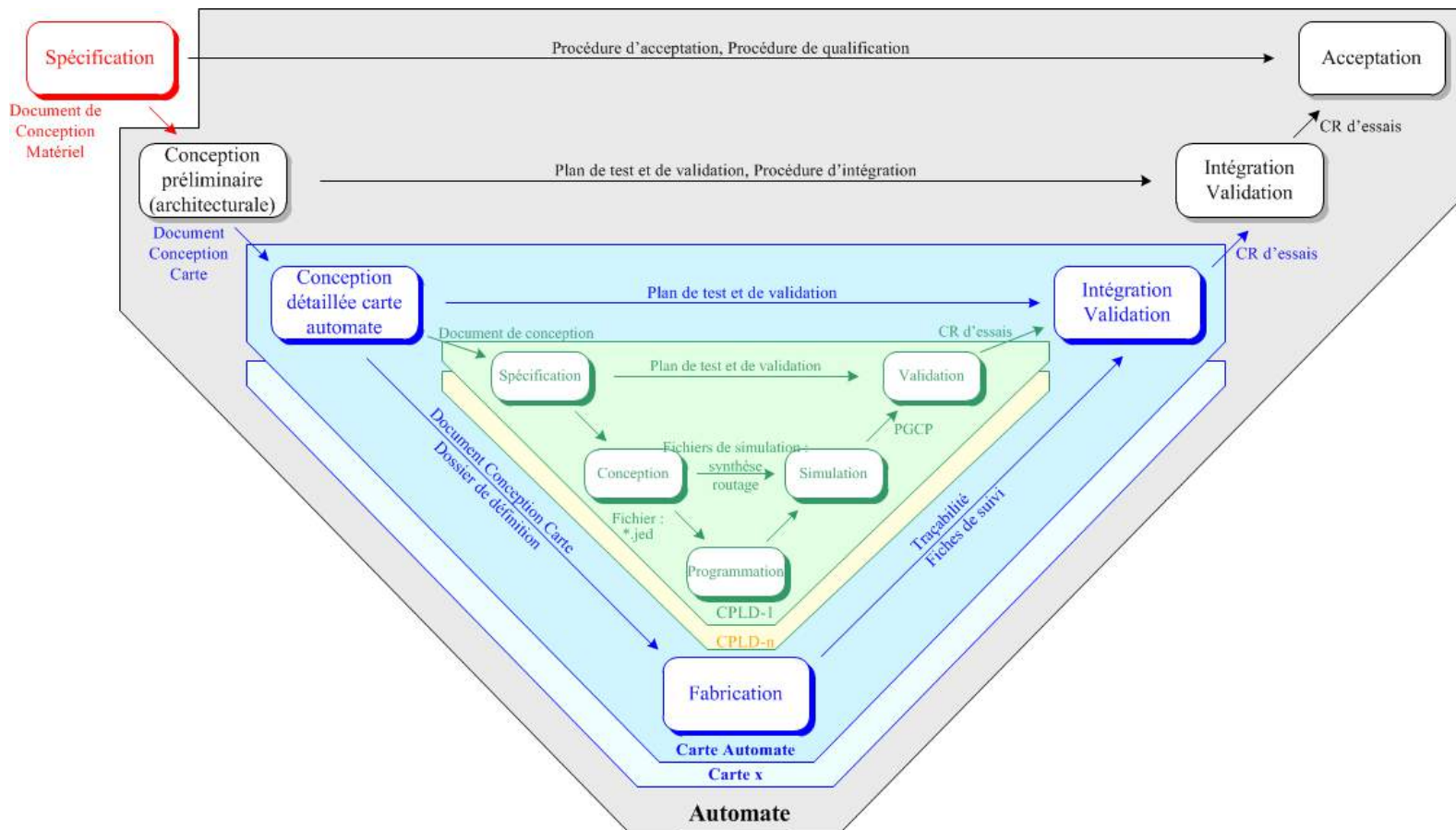




DEVELOPEMENT

GLOBAL DEVELOPEMENT

- **Process of realization : design → operational maintenance**
- **V-model methods : from the equipment to each CPLD**



■ Constraints & rules

- **Each CPLD is designed unitarily (Hardware Design Document - DCM)**
- **Schematic entry instead of textual entry (VHDL)**
- **Encoding rules :**
 - Use of primary functions of the chip manufacturer (AND, OR, D-latch, Counters, Comparator)
 - No-use of “complex” functions (RAM, FIFO, DLL, ...)
 - ...

■ Preliminary design of the programmable component

- **Hierarchical description (top down approach)**
- **List of critical points with analog simulation if needed (i/o ambiguity resolution from the very start)**

- **Detailed design of the programmable component**
 - **Hierarchical description by level (top down approach):**
 - Top level (hierarchical top)
 - Primary functions (Main functions)
 - Secondary functions (Sub functions)
 - **Identification of the necessary resources (CPLD number definition)**
 - **Realization of the sub functions (with the encoding rules constraints)**
 - **Simulation of the sub functions**
 - **Realization of the main functions (only use of the sub-functions)**
 - **Simulation of the main functions**
 - **Realization of the hierarchical top**
 - **Simulation of the hierarchical top**

- **Placement/Routing**
 - **Constraints definition**
 - **Placement / routing**
 - **Simulation**

- **Generation of the binary programing file**

- **Simulation (behavioral) on the schematic capture for validation of :**
 - Sub function
 - Main functions
 - Hierarchical top (all the logic circuit)

- **Simulation post implantation and routing phases for validation of :**
 - Real functionality of the chip (state and timing)

- **Development/simulation tools**
 - ISE for schematic capture (Xilinx chip manufacturer)
 - ISE for placement/routing (Xilinx chip manufacturer)
 - Modelsim for simulation (with Xilinx simulations libraries)

...



VALIDATION – TEST - MAINTENANCE

■ CPLD

- see previous simulations slides

■ BOARDS

▪ Functional qualification

According PTV (Functional validation Procedure)

Nominal test cases validation

Erroneous test cases validation (with non expected entries stimulus)

AMDEC (~FMEA : *Failure Modes and Effects Analysis*) results verification with fault injection/creation on the boards

Electrical stress test on the I/O (NS requirements)

▪ Test

According functional hardware TEST BENCH

For “automate” board (core) :

The board is tested in its operational mode (with nominal stimuli vs STB and with non expected entries = behavioral verification in several test cases – time consuming => limited by the power of the workstation

...

■ **Equipment (safety machine - «Automate»)**

- **Functional qualification**
- **According the PTV (Functional validation Procedure)**
- **Nominal test cases**
- **Degraded test cases (non expected entries)**

~ 3 000 test scripts are run (nearly one week in automatic mode)

Functional verification in the following configurations :

- **3 safety machines operational and running**
- **Only 2 safety machines operational and running**
- **Detection of wrong (unexpected) CPLD version**
- ...

- **Electrical stress test**

- **Environment qualification (climatic, mechanical - shock & vibrations, EMC, ..)**

- **Test**

According functional automatic hardware TEST BENCHes (test scripts with nominal cases, all the I/O and functions are covered)

...

■ **BOARDS**

■ **Test**

TAKAYA (mobile probes) at the end of the assembly line

Functional test on board with a specific automatic board test bench

The “automate” board is tested in its operational mode (with nominal stimulus vs STB and with non expected entries = behavioral verification in several test cases – time consuming => limited by the power of the workstation

■ **Equipment (safety machine - «Automate»)**

■ **Tests**

- **Integration tests**

- **Endurance tests**

- **Climatic tests**

- **Functional Tests (>150 scripts)**

- **Acceptance Procedures**

■ Preventive maintenance

■ Safety parameters monitoring

→ Periodic preventive maintenance

Boards test according the series process

“Safety machine” test according the series process (except climatics)

Corrective maintenance

■ Corrective maintenance

■ Functional entry control and analysis of “confirmed or not” failure on a NS level

■ According the replaced part we run all or parts of the preventive maintenance process



TEST BENCH

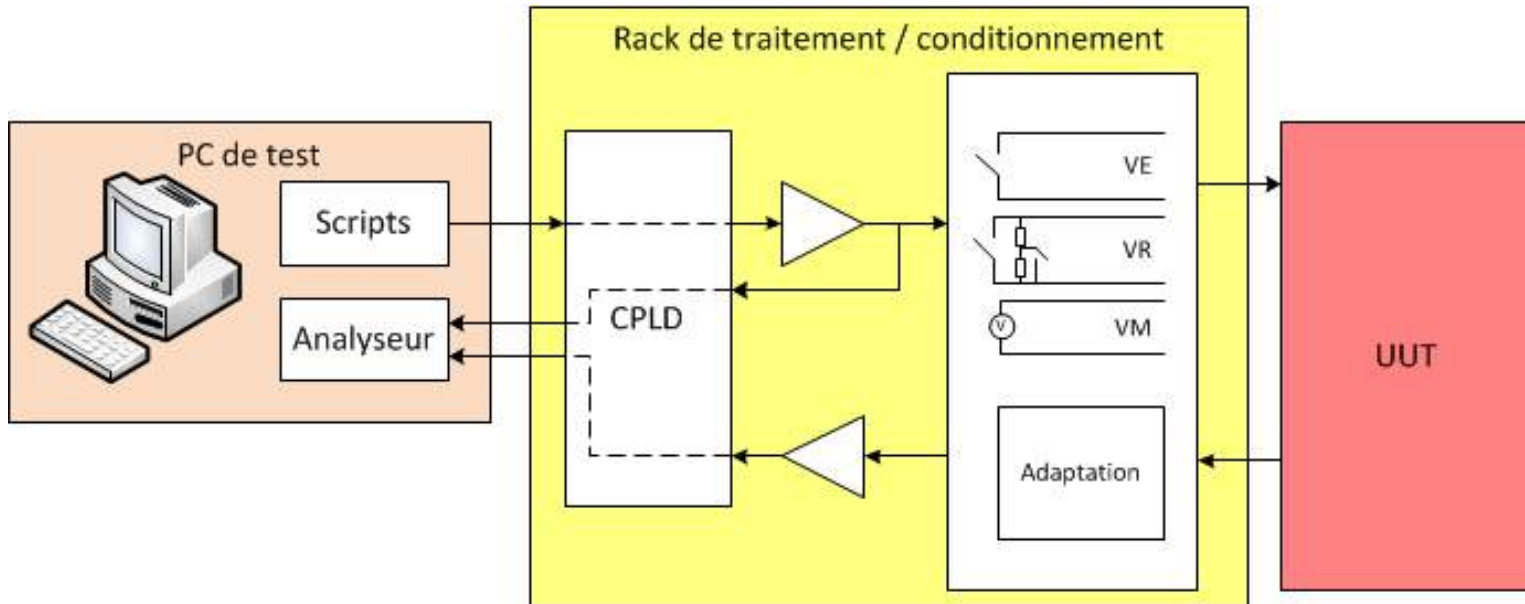
- **Manual and automatic TEST BENCH**
 - **Manual** : for development and possibly maintenance
 - **Automatic** : Production and preventive maintenance

- **No specific designed software**

- **Based on scripts with EXCEL (Visual basic)**

- **Philosophy :**
 - **Scenario loading**
 - **Pattern generation**
 - **Acquisition of the pattern generated (read at the outputs of the bench)**
 - **Acquisition of the output data of the safety machine**
 - **Display of the read states on the generated script**
 - **Verification :**
 - Pattern send = pattern expected to be send
 - Read results = Expected results

General synoptic





REX (return of experience)

- **Time period : 2005 -> today (mainly since 2009)**
- **Park : ~ 100 Equipment (300 “automates” boards → 6 300 CPLDs) functioning 24h/24**
 - 1 component failure (not a CPLD) on the “Automates” boards detected in preventive maintenance
 - No functional default (bug) detected by the operators
 - Zero operational default not-detected by the safety machine
- **On the 100 equipment park → 1 600 boards (I/O & “automate”)**
 - 30 failures detected



COTS

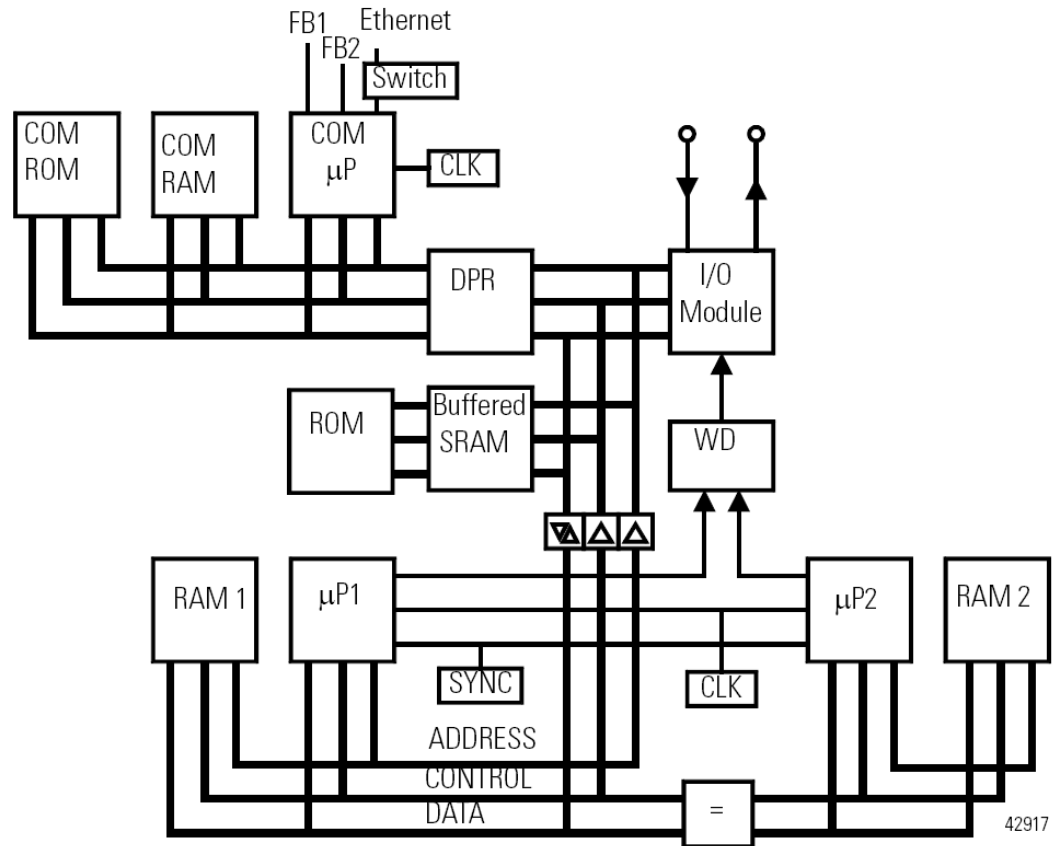
3 « safety » boards examples

- **ROCKWELL**

- **MEN**
 - Safe Railway
 - Power PC safe

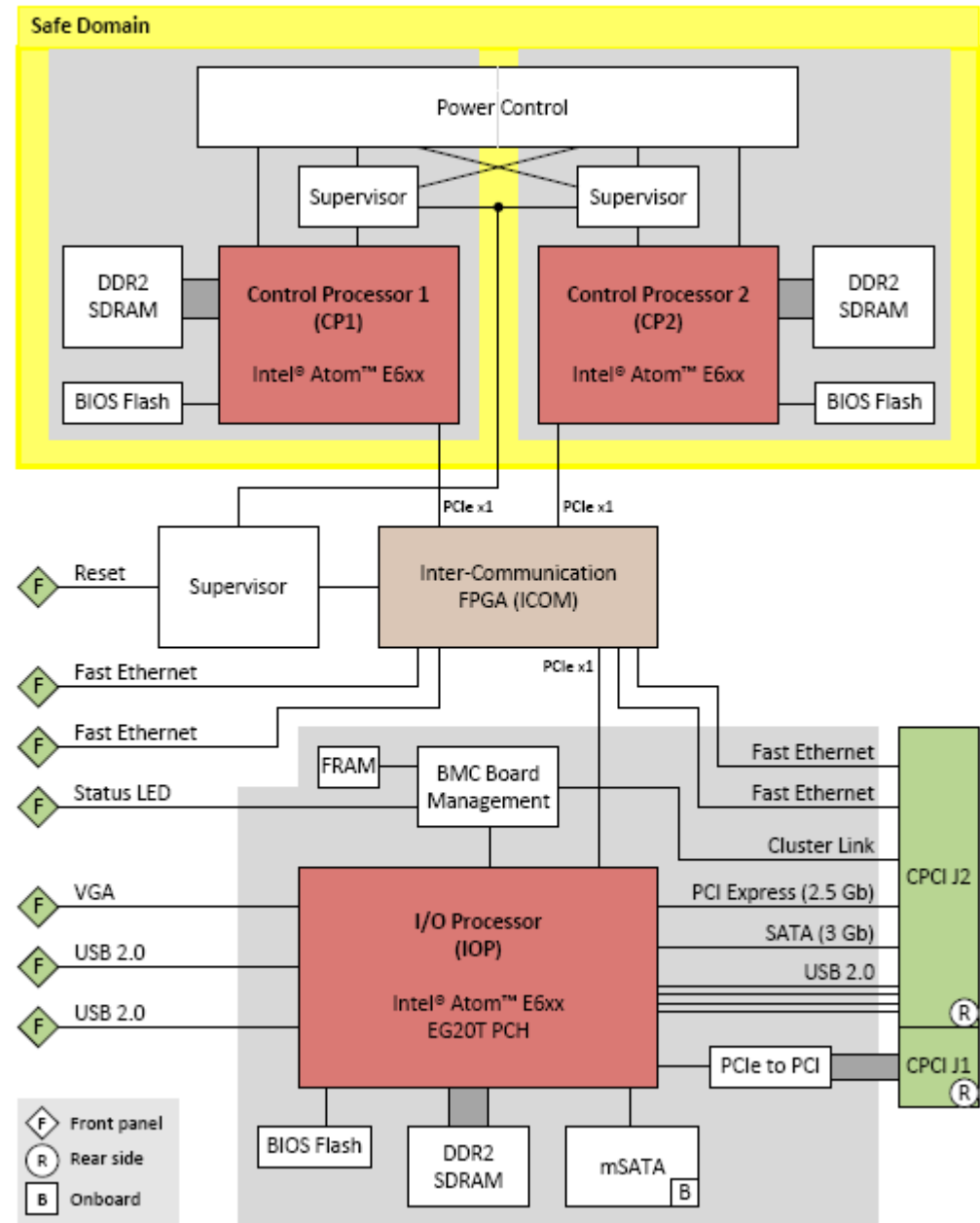
- **These 3 examples are at minima SIL3 / DO254 DAL B «certifiable»**

- Redundant architecture of the « executive »
- Simple « communication » architecture



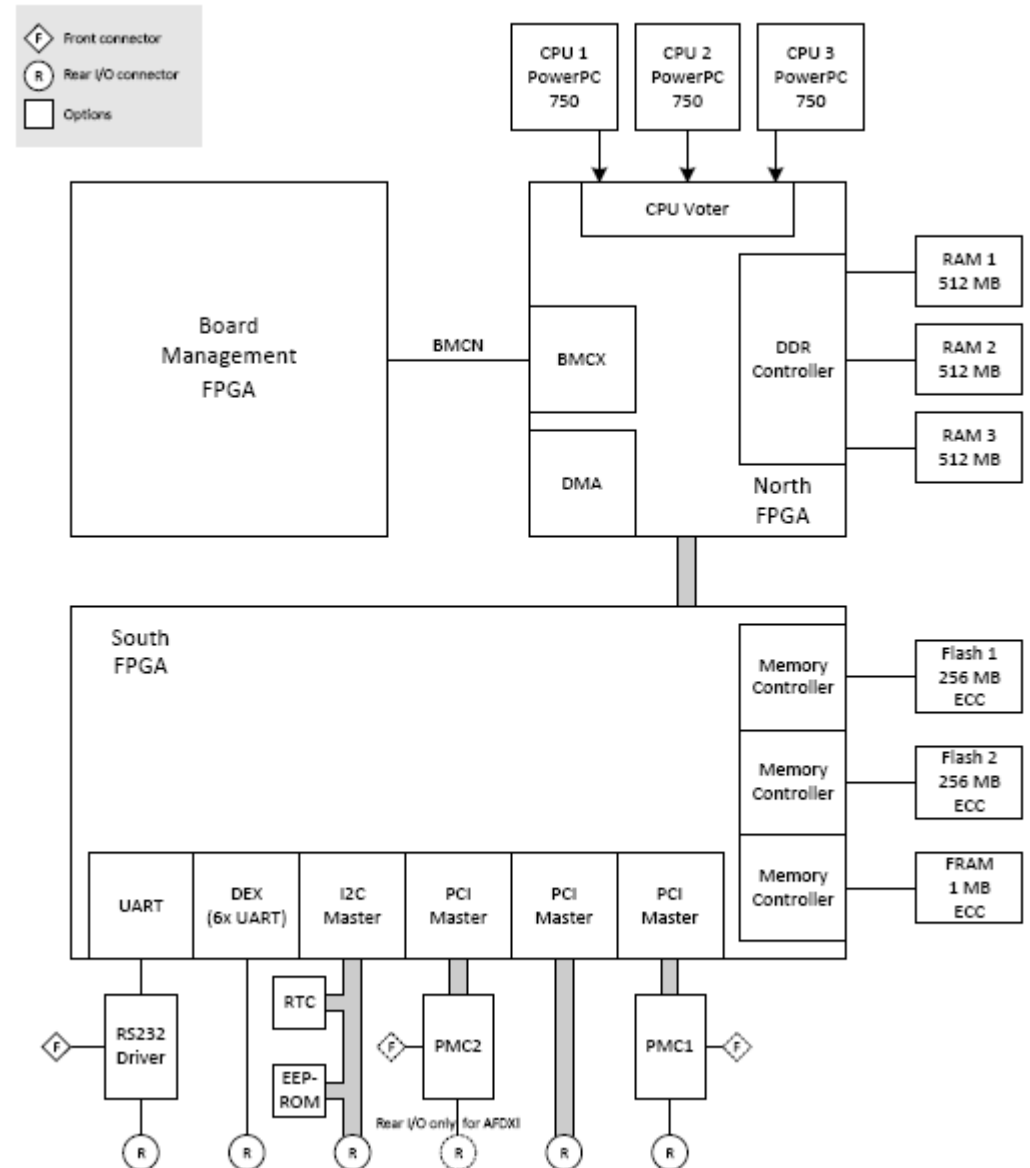
MEN –Safe Railway board

- Redundant architecture of the « executive »
- Simple « communication » architecture



MEN –Power PC - D602 board

- Triplicated architecture of the « executive »
- Simple « communication » architecture



2 Potentially Dangerous Events analysis

- **PDE_1 : Generation of a wrong order (but coherent)**

The syntax of the order sent is correct but is not the one expected to be sent

- **PDE_2: Absence of generation of system safe keeping**

The safety machine must send to the equipment an order to set the system in a safe state

PDE_1 – « COTS » triplicated architecture

The Potentially Dangerous Events is generated via :

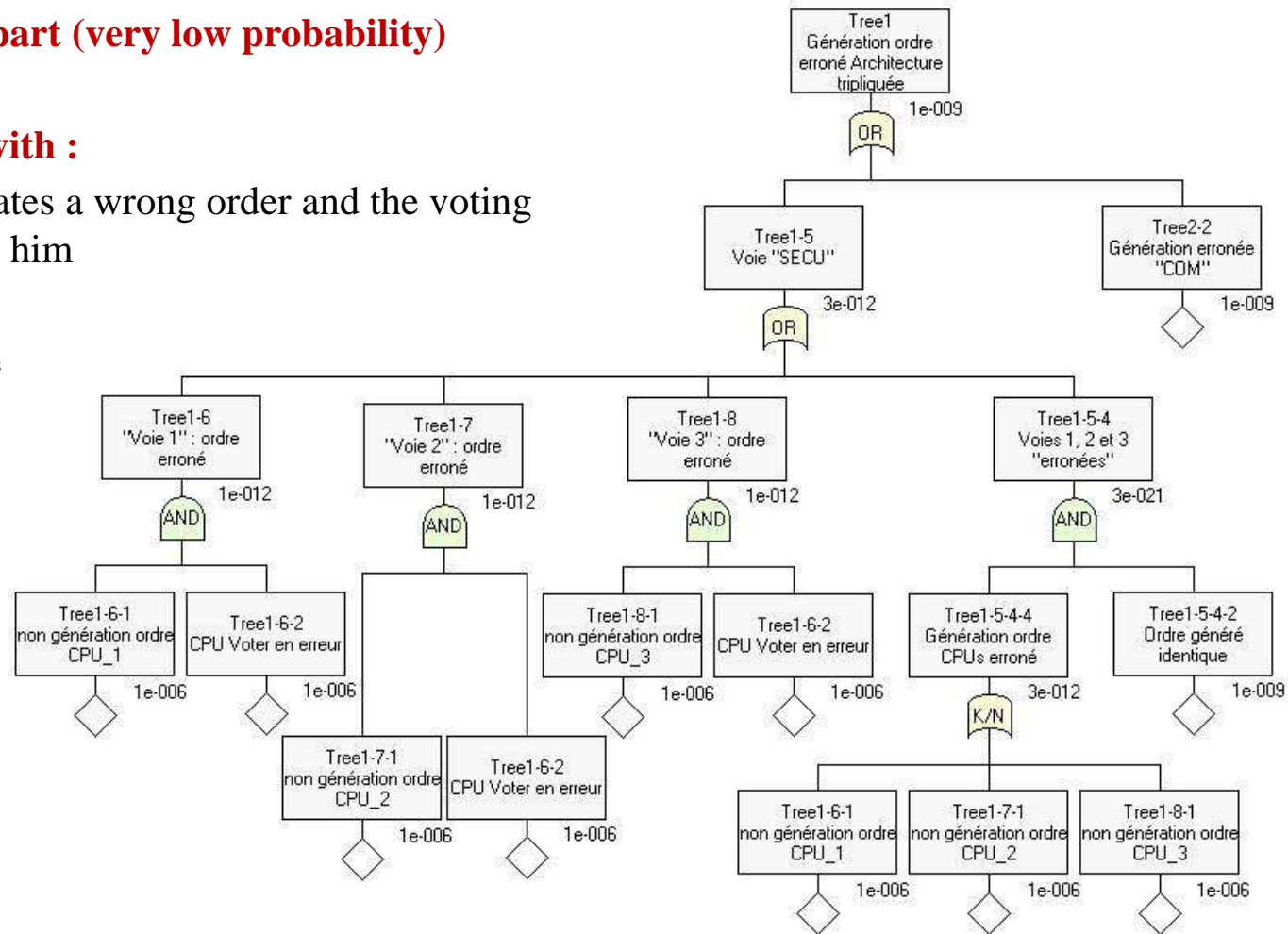
- The communication part (very low probability)

Or via

- The « safe » blocs with :
 - one CPU generates a wrong order and the voting does not filtrate him

or

- 2 CPU sent the same wrong order



■ Number of « safety barriers »

FTA - Minimal Cut Sets

Result for top event: FTA Name:

Minimal Cut Sets: Number of MCS: / Order of MCS: Min Max

N	Q(mean)	%	Order	Event 1	Event 2	Event 3
1	1e-009	99.7	1	Tree2-2		
2	1e-012	0.1	2	Tree1-6-1	Tree1-6-2	
3	1e-012	0.1	2	Tree1-6-2	Tree1-8-1	
4	1e-012	0.1	2	Tree1-6-2	Tree1-7-1	
5	1e-021	0.0	3	Tree1-6-1	Tree1-8-1	Tree1-5-4-2
6	1e-021	0.0	3	Tree1-6-1	Tree1-7-1	Tree1-5-4-2
7	1e-021	0.0	3	Tree1-7-1	Tree1-8-1	Tree1-5-4-2

The default can emerge directly from the Tree2-2 (« COM » wrong generation). It is highly improbable that functional blocs used for communication generates PDE.

Improvement : double the CPU boards to have a redundant communication channel

Remark : This analysis doesn't take into consideration the autotest

The Potentially Dangerous Events is generated via :

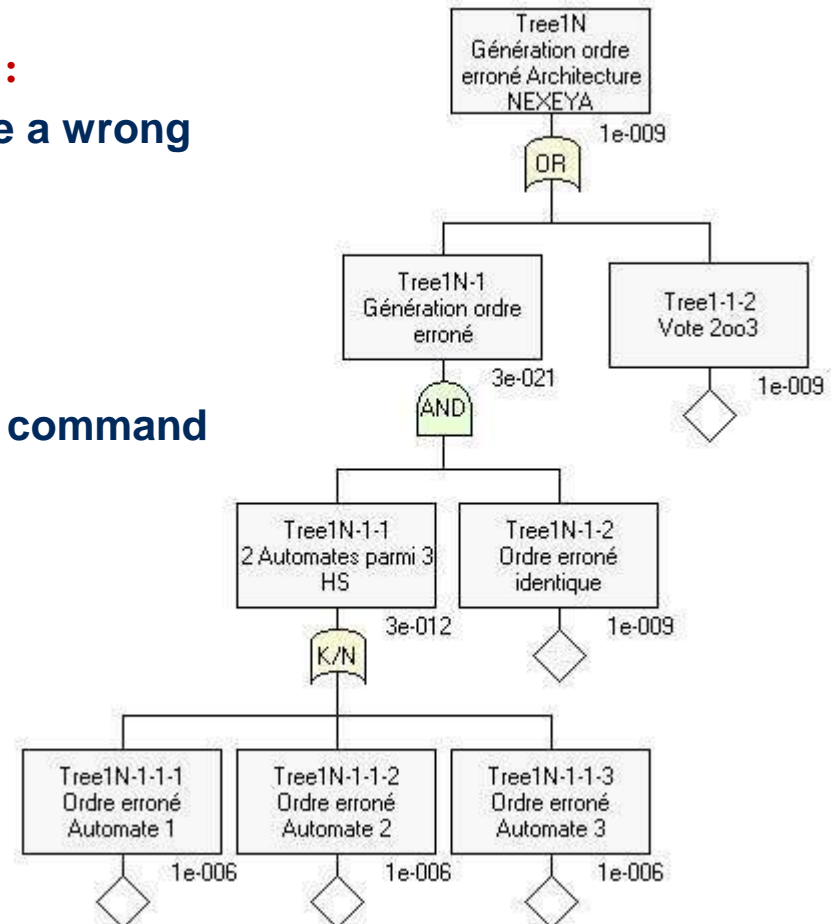
- When 2 out of the 3 machines generate a wrong order

AND

- The wrong order are identical

OR

- The 2oo3 vote generates the expected command output (very low probability)



- **Number of « safety barriers »**

FTA - Minimal Cut Sets

Result for top event: FTA Name:

Minimal Cut Sets: Number of MCS: / Order of MCS: Min Max

N	Q(mean)	%	Order	Event 1	Event 2	Event 3
1	1e-009	100.0	1	Tree1-1-2		
2	1e-021	0.0	3	Tree1N-1-1-1	Tree1N-1-1-2	Tree1N-1-2
3	1e-021	0.0	3	Tree1N-1-1-2	Tree1N-1-1-3	Tree1N-1-2
4	1e-021	0.0	3	Tree1N-1-1-1	Tree1N-1-1-3	Tree1N-1-2

- **The default can emerge directly from the Tree1-1-2 (2oo3 Vote). It is highly improbable => function realized with hard-wired logical gates.**
- **Improvement : double the corresponding command channel**

Remark : This analysis doesn't take into consideration the autotest.

PDE_2 – «COTS» triplicated architecture

The Potentially Dangerous Events is generated via :

- The communication part

Or via

- The « safe » blocs with :

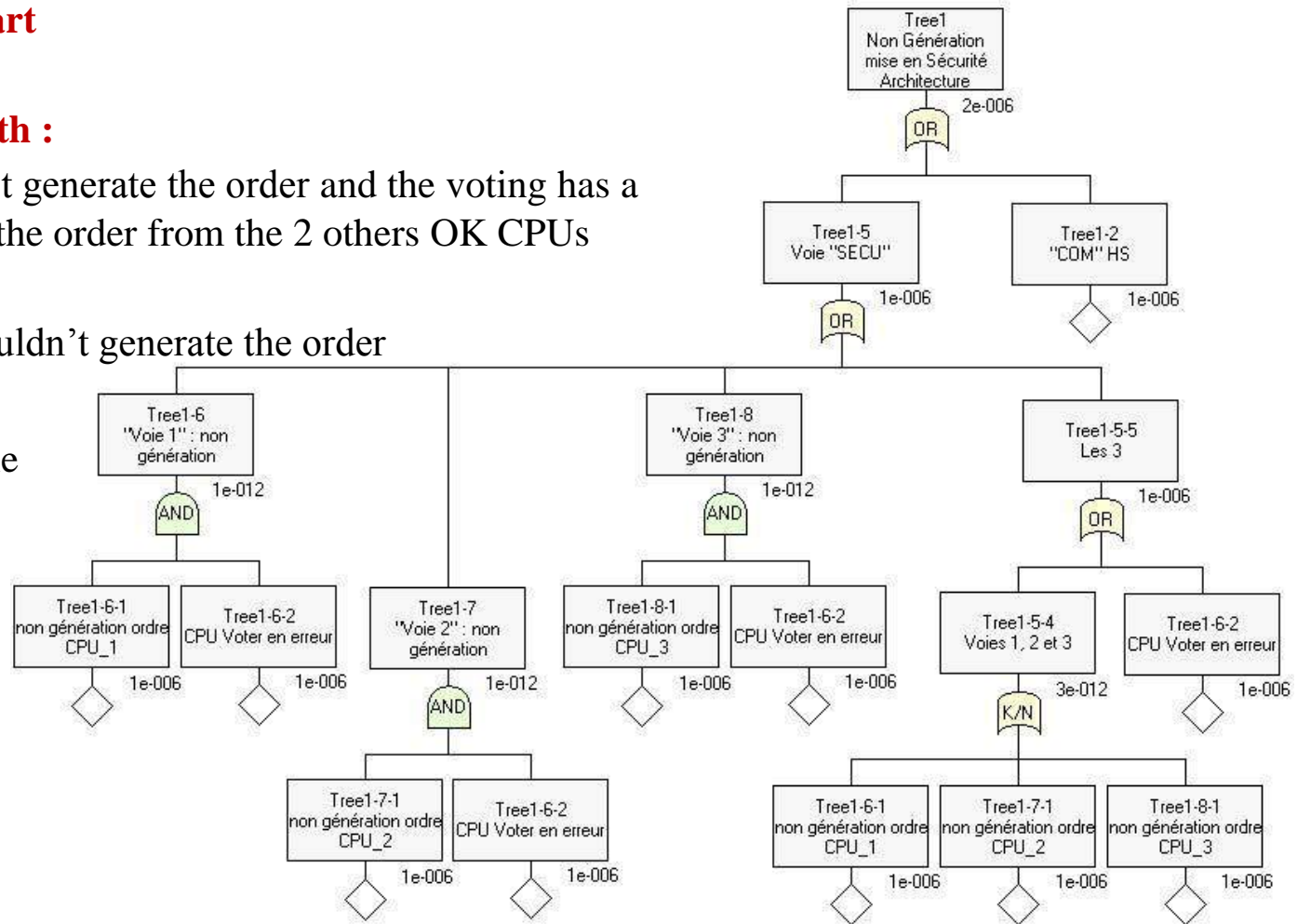
- one CPU couldn't generate the order and the voting has a failure to enable the order from the 2 others OK CPUs

or

- 2 CPU out of the 3 couldn't generate the order

or

- The CPU vote machine is out of order



■ Number of « safety barriers »

FTA - Minimal Cut Sets

Result for top event: FTA Name:

Minimal Cut Sets: Number of MCS: / Order of MCS: Min Max

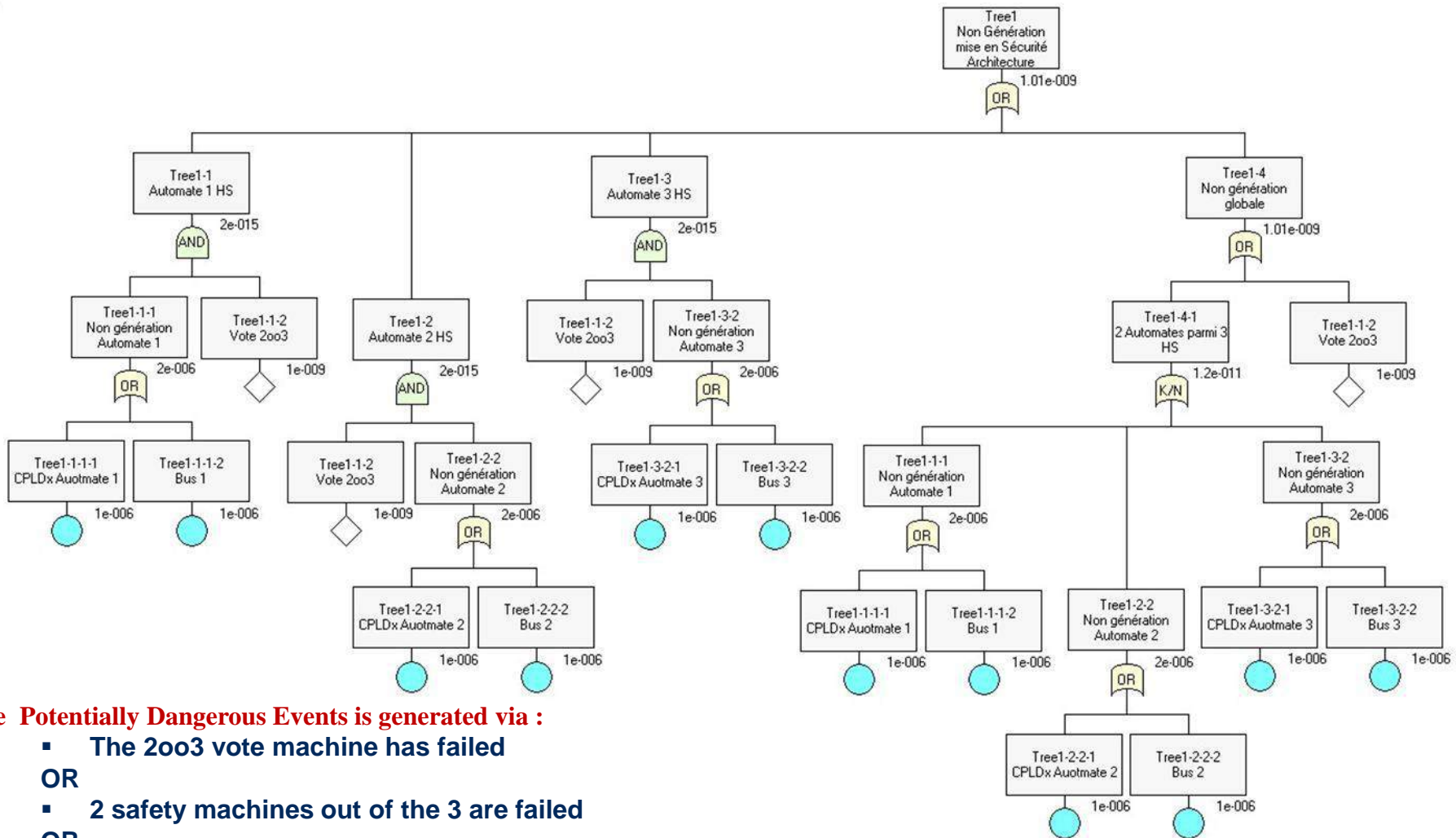
N	Q(mean)	%	Order	Event 1	Event 2
1	1e-006	50.0	1	Tree1-2	
2	1e-006	50.0	1	Tree1-6-2	
3	1e-012	0.0	2	Tree1-6-1	Tree1-8-1
4	1e-012	0.0	2	Tree1-6-1	Tree1-7-1
5	1e-012	0.0	2	Tree1-7-1	Tree1-8-1

The default can emerge directly from the Tree1-2 (« COM » ko) or Tree1-6-2 (CPU Vote failed).

For the 2 « Tree » the occurrence probability is not negligible because any failure of a chip of these blocs interfere.

- Improvement : double the CPU boards to have a redundant communication channel and vote machine

Remark : This analysis doesn't take into consideration the autotest.



The Potentially Dangerous Events is generated via :

- The 2oo3 vote machine has failed
- OR
- 2 safety machines out of the 3 are failed
- OR
- 1 safety machine out of the 3 has failed
- AND
- One of the 2 entries corresponding to the 2 others safety machines has failed

- **Number of « safety barriers »**

FTA - Minimal Cut Sets

Result for top event: FTA Name:

Minimal Cut Sets: Number of MCS: / Order of MCS: Min Max

N	Q(mean)	%	Order	Event 1	Event 2
1	1e-009	98.8	1	Tree1-1-2	
2	1e-012	0.1	2	Tree1-1-1-1	Tree1-2-2-1
3	1e-012	0.1	2	Tree1-1-1-2	Tree1-2-2-2
4	1e-012	0.1	2	Tree1-1-1-1	Tree1-2-2-2
5	1e-012	0.1	2	Tree1-2-2-1	Tree1-3-2-2
6	1e-012	0.1	2	Tree1-2-2-1	Tree1-3-2-1
7	1e-012	0.1	2	Tree1-2-2-2	Tree1-3-2-2
8	1e-012	0.1	2	Tree1-2-2-2	Tree1-3-2-1
9	1e-012	0.1	2	Tree1-1-1-2	Tree1-3-2-2
10	1e-012	0.1	2	Tree1-1-1-2	Tree1-3-2-1
11	1e-012	0.1	2	Tree1-1-1-1	Tree1-3-2-2
12	1e-012	0.1	2	Tree1-1-1-1	Tree1-3-2-1
13	1e-012	0.1	2	Tree1-1-1-2	Tree1-2-2-1

The default can emerge directly from the Tree1-1-2 (2oo3 Vote). It is highly improbable => function realized with hard-wired logical gates

Improvement : double the corresponding command channel

Remark : This analysis doesn't take into consideration the autotest.