Dependability Modelling and Assessment of Avionics Systems with Altarica.

P. Bieber, Ch. Castel, G. Durrieu, Ch. Seguin, C. Pagetti, L. Sagaspe

General Problem

- Avionics are complex systems
  - A380 (safety critical avionics):
    - +100 computers connected to the main Aicraft network,
    - ~10 000 data flows transmitted over the network
- Structured Design
  - Modular design
    - Systems : Flight Control, Flight Management, Flight parameters, …
  - Layered design
    - functional architecture/allocation/ hardware architecture
- Complex Design Process
  - Several actors:
    - System designers -> functional architecture
    - Platform designers -> hardware architecture
    - Integrator -> allocation

General Goal

- Support the safety assessment of avionics systems
  - using Altarica models
  - and taking into account the current design process

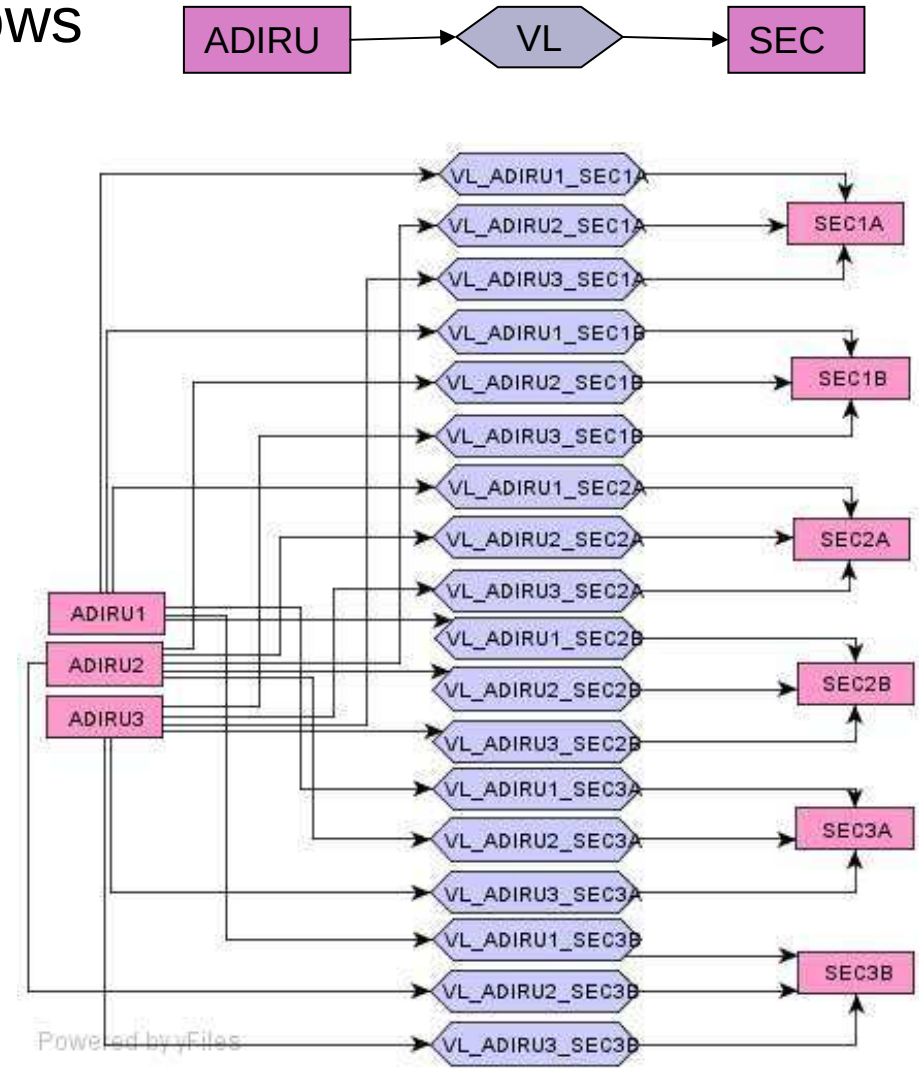
- Apply the approach on case-studies
  - Dassault Mirage Terrain Following/Terrain Avoidance
  - Airbus systems (ADIRS, Fuel On Board,…)
  - Astrium ATV (Automatic Transfer Vehicle)

Overview

- Avionics Platform Design
  - Functional and hardware description
  - Allocation
- Safe Resource Allocation Process
  - Failure Propagation Modelling
  - Safety Requirements Validation
  - Independence requirement derivation
- Advanced Topics
  - Allocation Generation by Constraint Solving
  - Installation related risks
  - Automatical production of Altarica models
  - Middleware Modelling

Functional Architecture

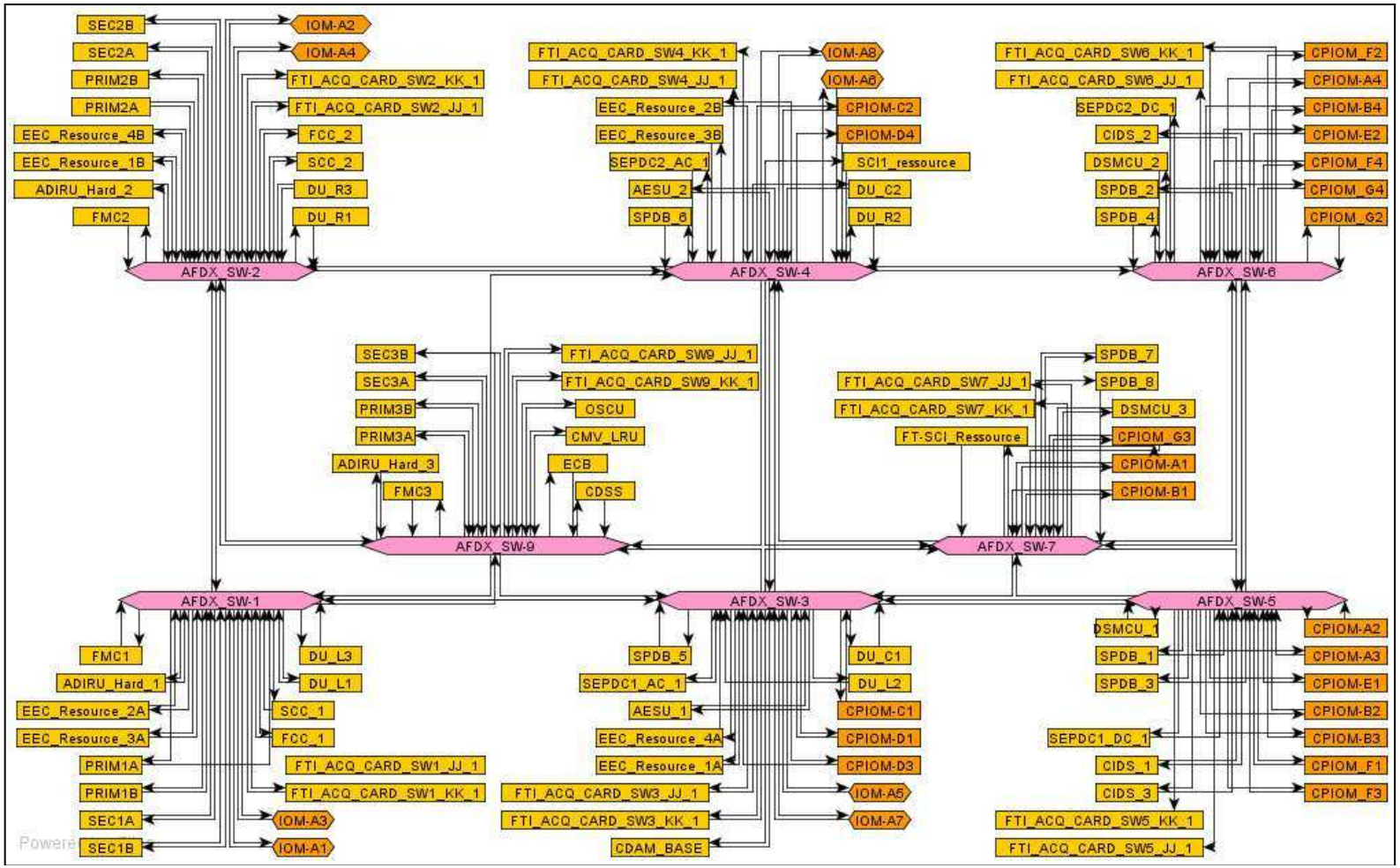
- Function and Data flows
  - ADIRU: x3
  - SEC: x6
  - VL : x18



Hardware Architecture

Interconnected resources

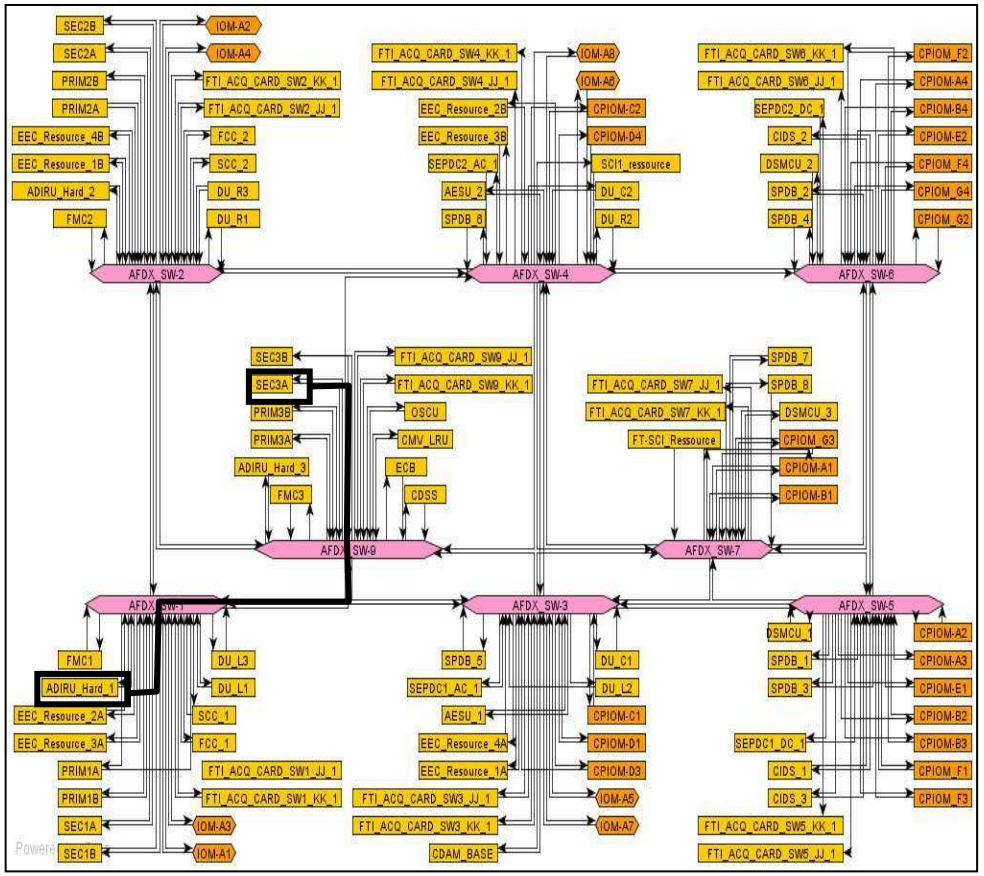
- Bus, Switch, CPU, …



Allocation

- Described as tables
  - well formalized at detailed design stages
  - but often missing at earlier design stages

```
VL_ADIRU1_SEC3A :
   ADIRU_Hard_1,AFDX_SW-
   1,AFDX_SW-1,AFDX_SW-
   9,SEC3A
```

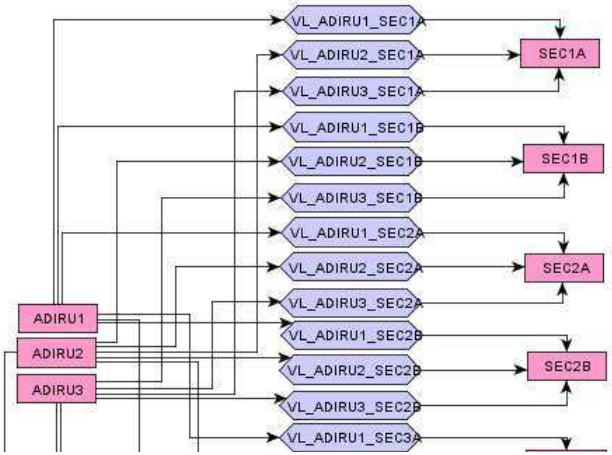


Overview

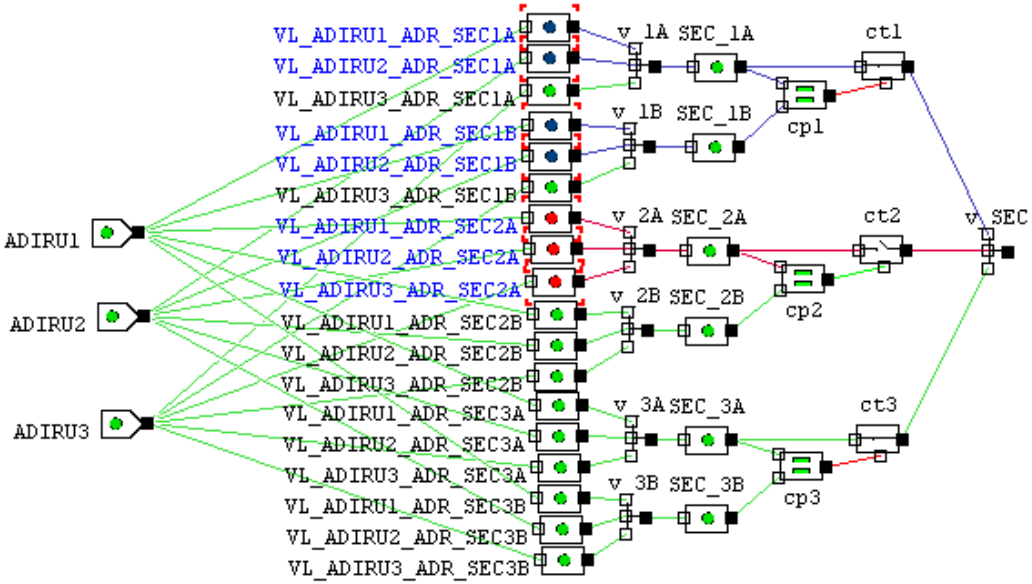
- Avionics Platform Design
  - Functional and hardware description
  - Allocation
- Safe Resource Allocation Process
  - Failure Propagation Modelling
  - Safety Requirements Validation
  - Independence requirement derivation
- Advanced Topics
  - Allocation Generation by Constraint Solving
  - Installation related risks
  - Automatical production of Altarica models
  - Middleware Modelling

Functional Architecture Safety Model

- Failure Propagation Model built using predefined nodes in an Altarica Library

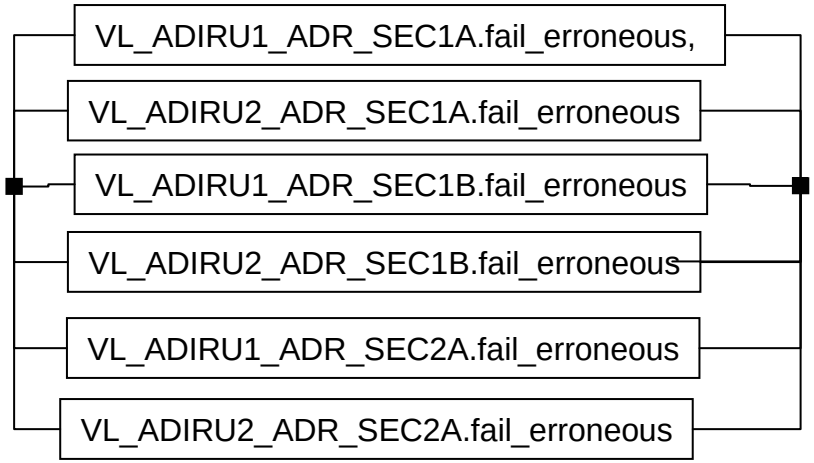


- Qualitative Safety Requirement:
  - « No double failure of dataflows between ADIRU and SEC shall cause the loss of all SEC functions »
  - « No double failure of dataflows between ADIRU and SEC shall cause the undetected erroneous behaviour of all SEC functions »



Safety Requirement Assessment

- Automatic Generation of the fault-tree from the model
  - Generation of minimal cut sets

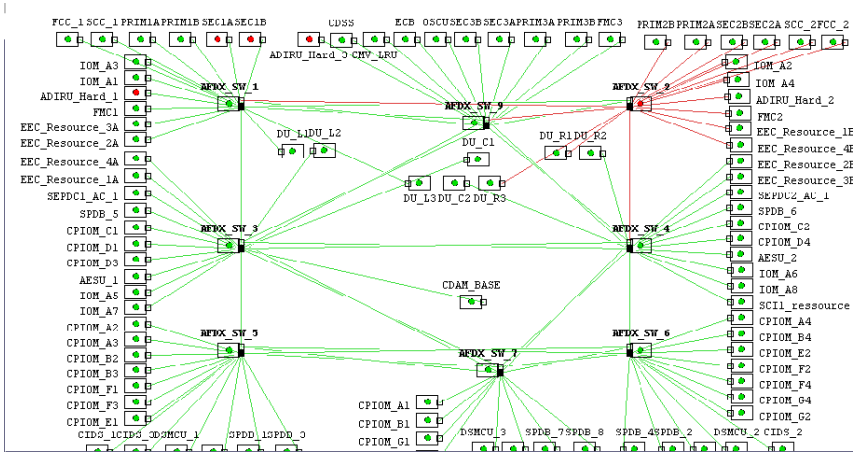


Size	Loss	Erroneous
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	5832	8748
7	1944	972
8	216	0
9	8	0
Total	8000	9720
Proba	2.0 e-24	3.0 e-24

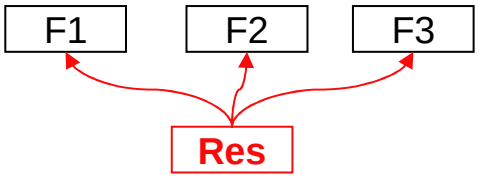
  - Computation of probabilities

Hardware + Allocation models

- Hardware model
  - very basic model

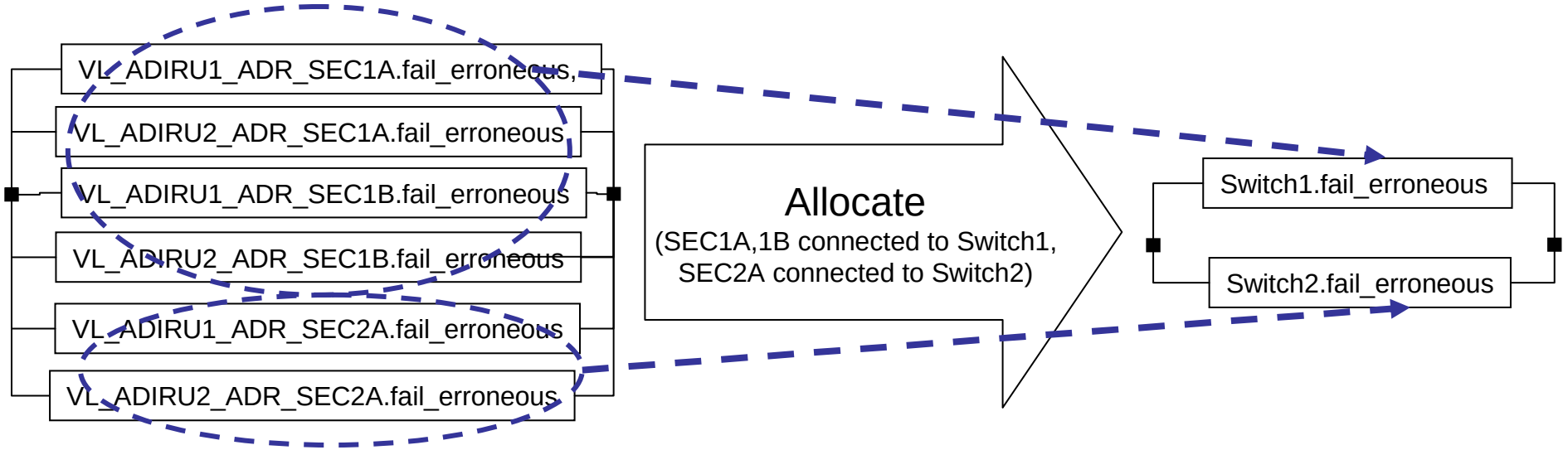


- Allocation model
  - Common cause failure
  - Use Broadcast to group failure event of the resource with failure events of all supported functions and data flows



Impact of allocation on Safety requirements

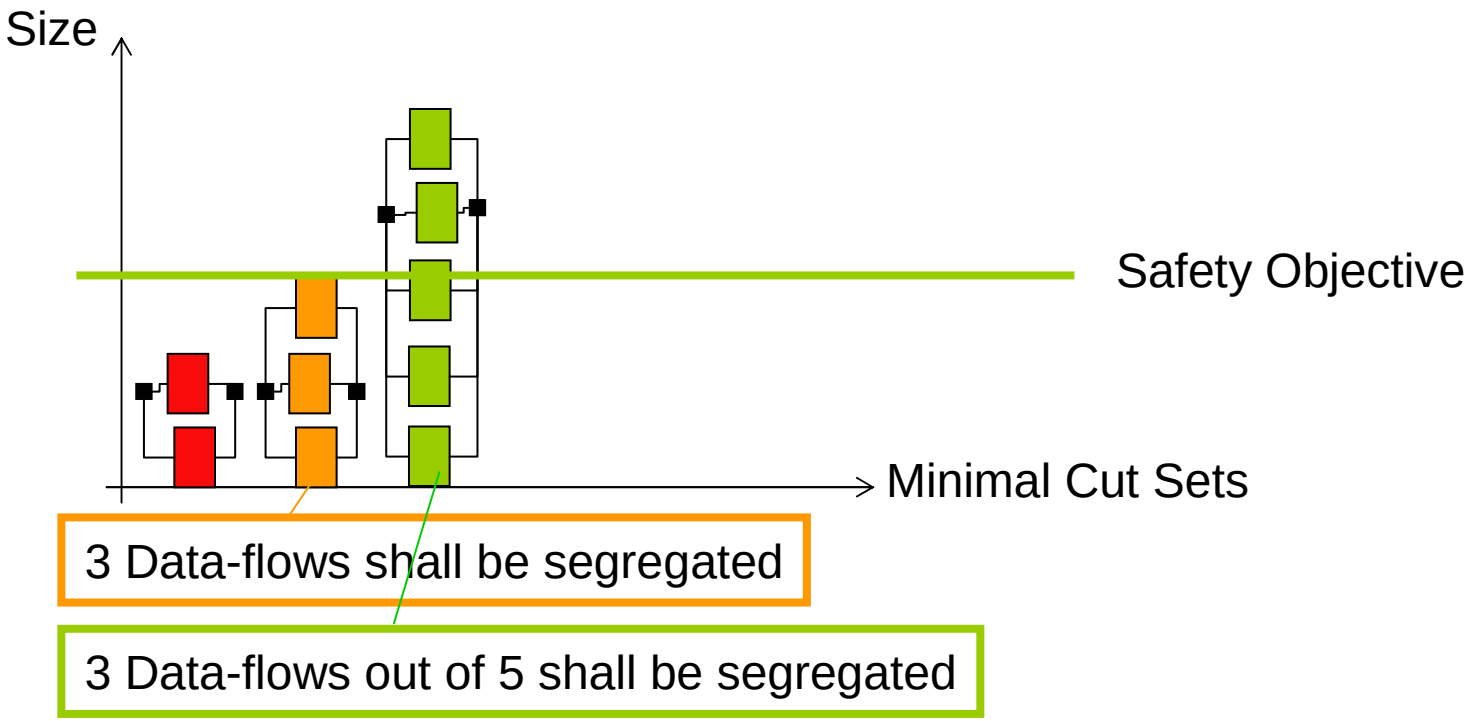
- Allocation of shared resources to functions and data-flows creates Common Mode Failures.



- Compare before/after allocation:
  - Decrease size of minimal cut sets,
  - increase probability of FC occurrence
  - Is this impact acceptable ?

Derivation of Segregation Requirements

- Extract segregation requirements from the safety assessment results in order to avoid allocation common mode failures



Overview

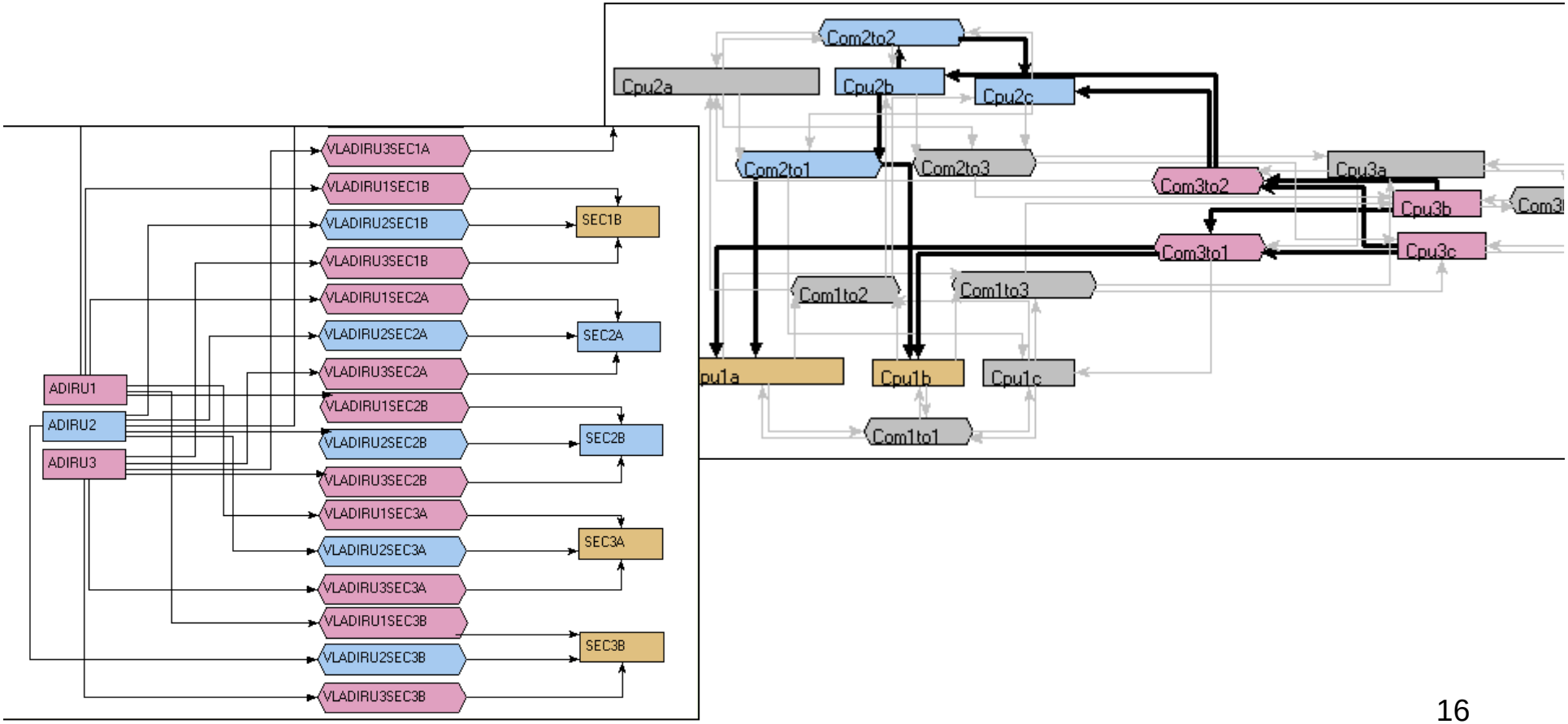
- Avionics Platform Design
  - Function and architecture description
  - Allocation
- Safe Resource Allocation Process
  - Failure Propagation Modelling
  - Safety Requirements Validation
  - Independence requirement derivation
- Advanced Topics
  - Allocation Generation by Constraint Solving
  - Installation related risks
  - Automatical production of Altarica models
  - Middleware Modelling

Allocation Generation by Constraint Solving

- Formalisation of allocation constraints
  - {0,1} linear inequalities.
- Variables :
  - `allotc(task,cpu) : {0,1}`
  - `allodb(data,bus) : {0,1}`
  - `connected(cpu,bus) or connected(bus,cpu) : {0,1}`
- Inequalities
  - Any task has to be allocated to one and only cpu
    `allotc(t,c1) +…+ allotc(t,cn) = 1`
  - Two segregated tasks should not be allocated to the same cpu
    `allotc(t1,c) + allotc(t2,c) + segregated(t1,t2)` $\leq$ `2`
  - A connection (C,B) is used if there exists a data flow D and its producing task T such D is allocated to B and T is allocated to C.
- Criterion
  - Minimise the number of used connections

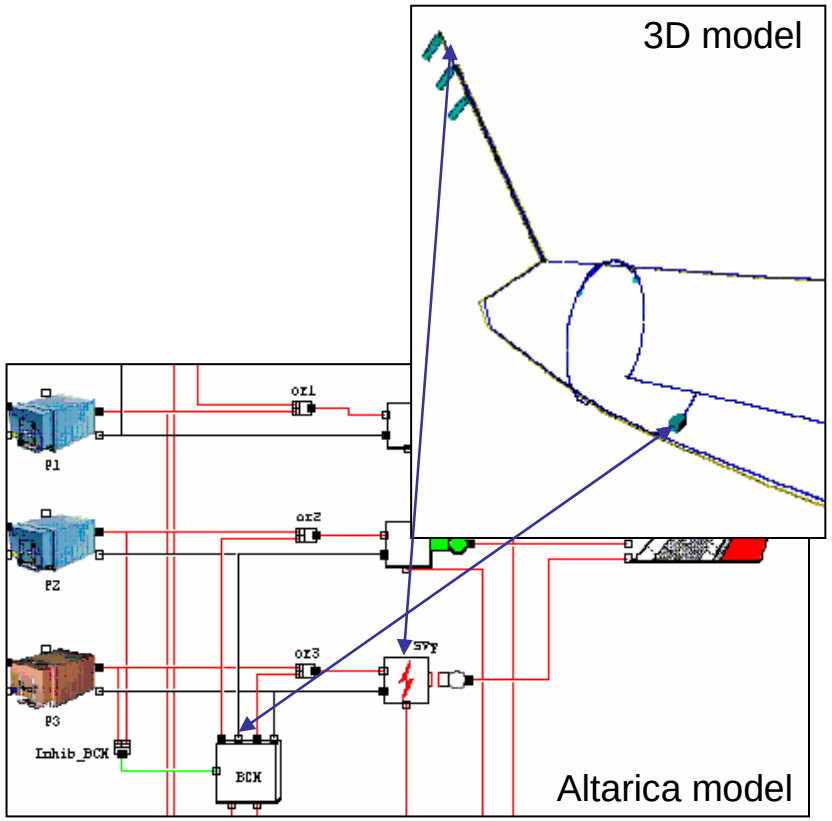
Tool Support for Constraint Solving

- Generation of constraints
- Call to solvers (ILOG solver, satzoo)
- Visualisation of allocations

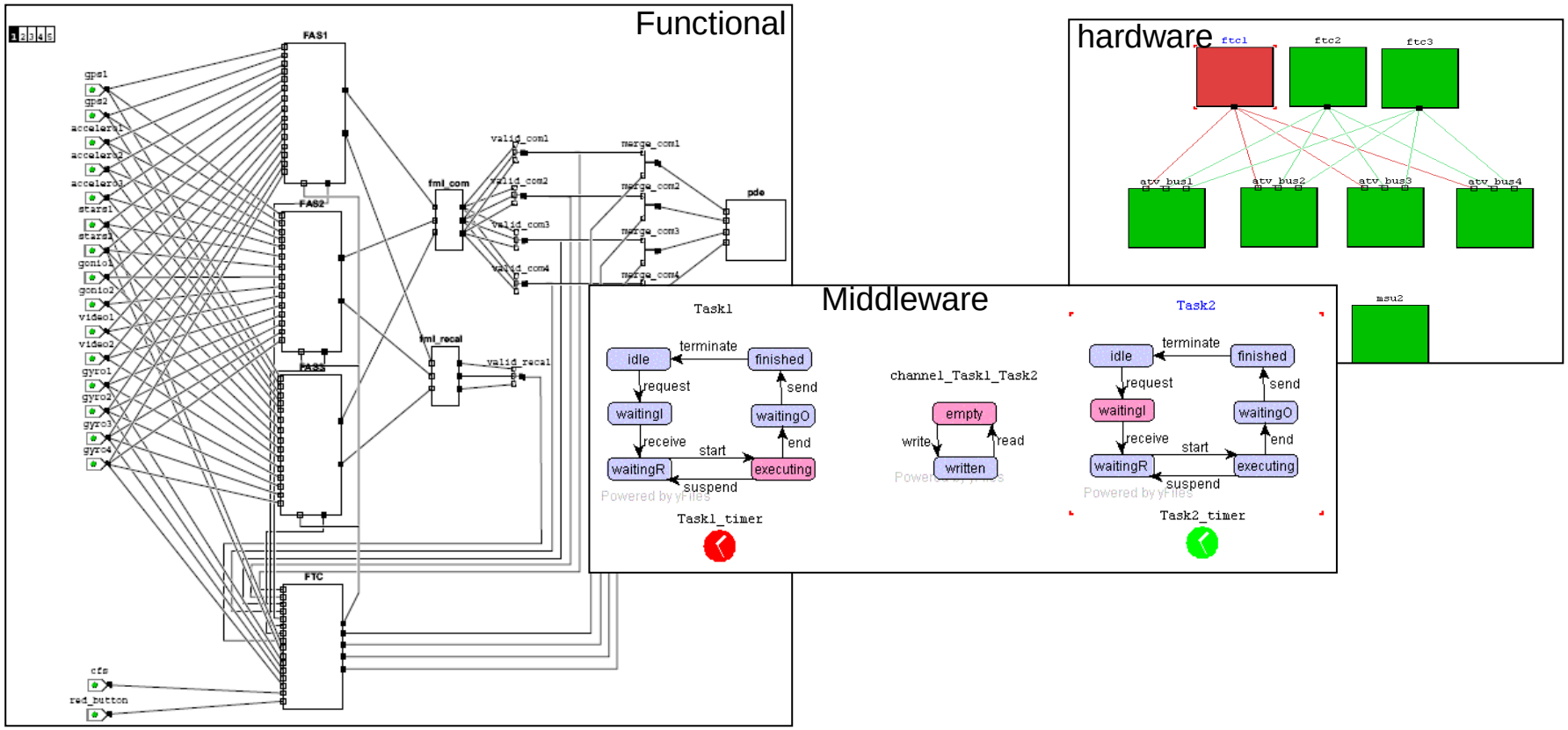


Installation Related Assessment

- Assess the impact of equipment installation on Safety Requirements
- Link functional architecture model with Digital Aircraft mockup (CATIA, IRIS)
  - Similar to the modelling of allocation of functions on hardware
- Study the effect of tyre or engine burst on functions



ATV Case Study



- Software dependability oriented model:
  - More detailed functional Architecture, simpler hardware model
  - Add a model of middleware services « between » functional view and Hardware architecture view to study new kind of failure propagations in the temporal domain

Automated Production of Altarica models

- Generate dependability models
  - Industrial need : decrease the modelling effort
  - AADL (Avionics Architecture Description Language) to Altarica model transformation
  - AADL models structured in layers
    - Hardware and allocation : similar to Altarica, easy to transform
    - Functional architecture : more expressive, not so easy to transform…
  - AADL Error Annex
    - AADL special notation for failure propagation models
    - Adapted for Software failure propagation modelling
    - Limited tool-support (by now)

Conclusion – Further work

- Requirement driven engineering
  - Organize the design activities
  - Define what models should be built and what analysis should be performed
- Models for software dependability
  - Model more accurately software
- Optimise avionics architecture with respect to several viewpoints :
  - real-time performances, operational reliability, installation, Electro-magnetic Interference, …