Safety-Critical Real-Time Systems

ARTES PhD course

Lecture 4: Formal Analysis of Fault Tolerance

Simin Nadjm-Tehrani ©

Real-time Systems Laboratory

simin@ida.liu.se

Recall from earlier...

- Removing/containing certain faults enhances safety
- How about faults that do not originate from digital component being designed?

Formal techniques

- The least we can do:
  - **Avoid**/**remove** (design) faults that lead to obvious bad things
  - **Analyse behaviour** in presence of selected combinations of potential external faults

To illustrate...

- Two case studies:
  - Hydraulic electronic control unit in JAS Gripen

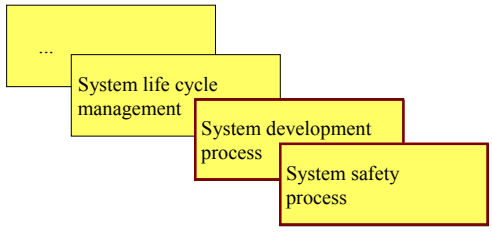
    [Hammarberg & Nadjm-Tehrani 04]
  - Distributed flight control

    [Forsberg, Nadjm-Tehrani, Torin 05]

Case study I

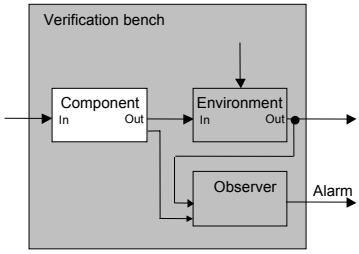
- Masters Thesis by Jerker Hammarberg studied shut down system for JAS hydraulic system
- Illustrates a design process covering
  - Functional verification of FPGA
  - Effects of failure modes at system level
  - Formal verification of FTA/FMEA like analyses

Parallel processes

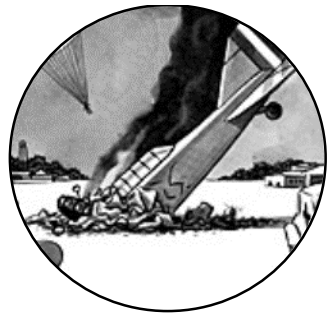


Can we use the same system model for these two?

Pattern: Functional verification



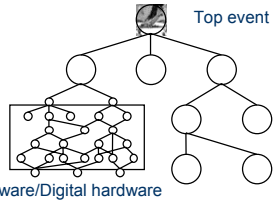
Non-occurance of Catastrophic failures



Patterns for safety analysis?

Traditional FTA/FMEA

FTA:



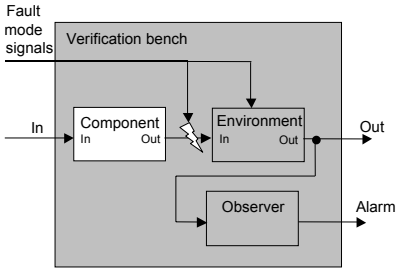
FMEA:

What are the consequences of some particular component's failure?

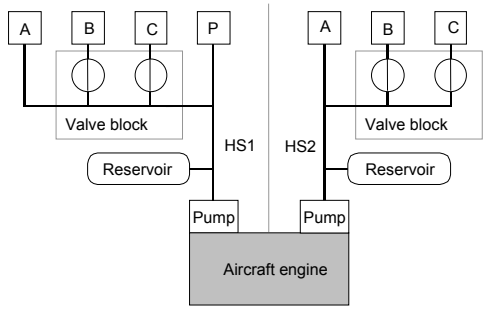
Generic approach

- Design a digital component in language X
  - Software and hardware
- Build verification bench in development environment for language X
  - Physical models
  - Observer
- Iterate until functional requirements are met:
  - Verify properties
  - Change design, add constraints
- Add fault mode analysis and study safety related properties

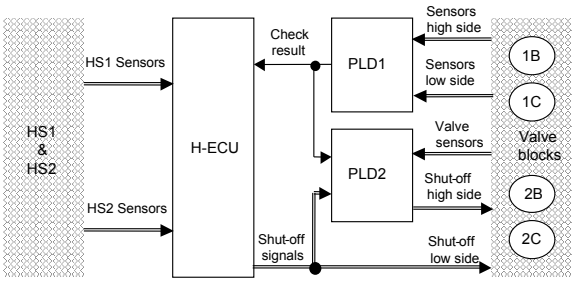
Pattern: Fault mode modelling



Example: Hydraulics monitoring



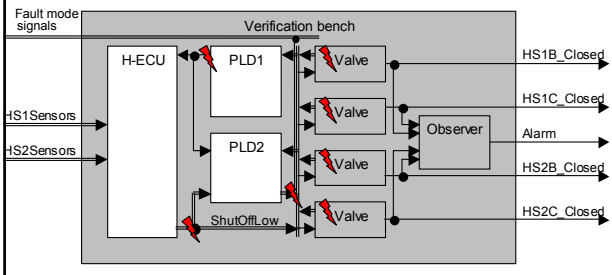
Safety-related hardware



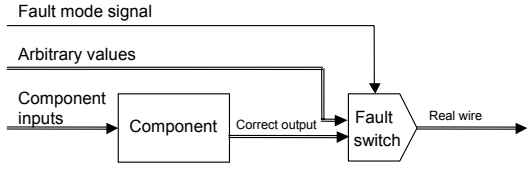
Hazards/Faults

- Hazard:
  - Closing more than one valve at a time
- Potential faults
  - Faults in the components, radiation-induced bit flips
  - Faults in the valves, stuck-at faults

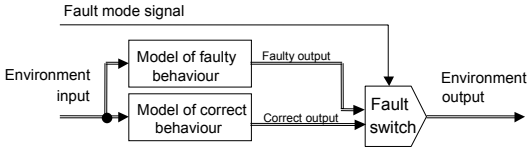
Example: Fault mode modelling



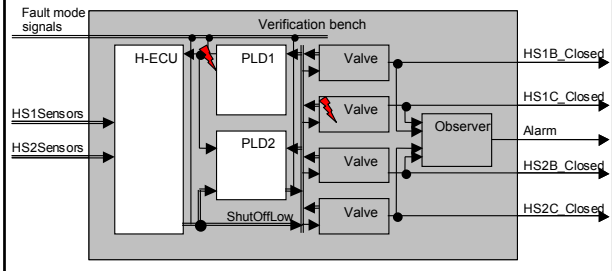
Fault modes: Component faults



Fault modes: Physical system



Which combinations?



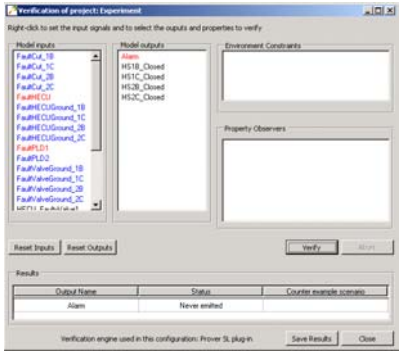
FTA/FMEA like analysis

- Without building the trees
- Can top level failure appear with:
  - Single faults?
  - Multiple faults?
- Proved by a SAT-based model checker (Prover plugin)
- Countermodels are *sequences* of combinations of input and fault modes

Hydraulic system

- Fifteen fault modes
  - Three component faults
  - Three possible faults on each of the four valves
- Results:
  - No single faults lead to top event
  - No combination of valve faults lead to top event
  - One combination of HECU & PLD2 faults violates the safety property

In Esterel Studio

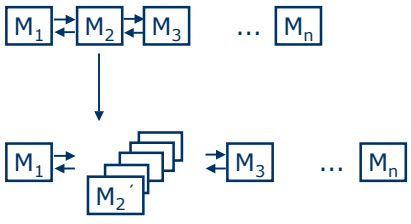


What next?

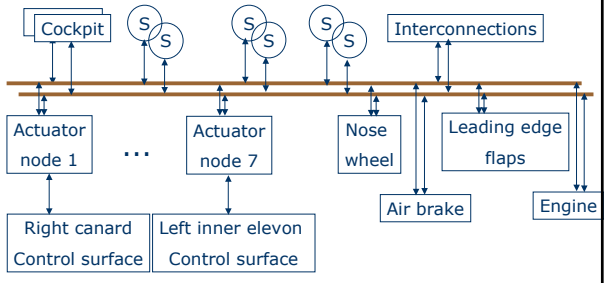
- In progress:
  - Automatic generation of cut sets for better support to safety analyser
- Formal models of (upgradeable) components
  - What does a component interface need to capture to provide enough information for current (and future) safety analyses?

Case Study II

Distributed vs. central

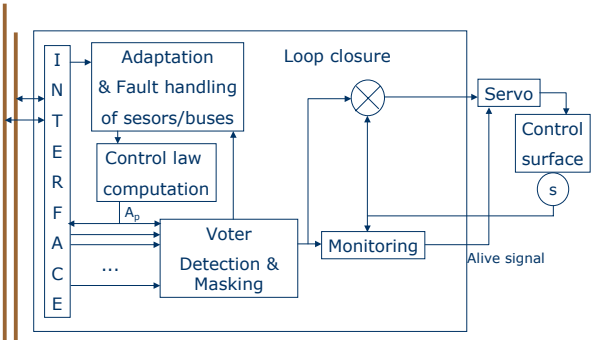


Distributed flight control



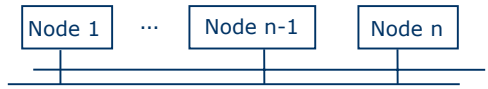
7 primary control surfaces 3 secondary control surfaces

Actuator node



Architectural support

- Timed-triggerd architecture [Kopetz et al.]



- Time division multiple access (TDMA)

Bus traffic

Duplicated critical sensors broadcast on both buses

Actuators broadcast their results

BUS1	...	S_i^1	...	S_i^2	...	A_1	A_2	...	A_7	
BUS2	...	S_i^1	...	S_i^2	...	A_1	A_2	...	A_7	

TDMA cycle

Message Ap from node p may be: $[v_1,...,v_7, mode_m^n, \alpha, \varepsilon^p, ...]$

Errors in Actuators

- May be caused by permanent or transient faults in:
  - Sensors or buses
    - Wrong or missing value
      - All actuators (all rounds or one round)
      - Some actuators
  - Communication interface
    - Corrupted or missing value
  - Processor/memory
    - Crash
    - Value error (all rounds or one round)

Requirements

1. No combinations of transient faults lead to streamlining.
2. The distributed actuators behave as one wrt discrete signals:
   1. Mode changes will get reflected in decisions by all actuators in the same TDMA cycle, and within pre-defined interval.
   2. If none of the control surfaces are streamlining then none of the computations are in this mode.

Correctness claim

- Can be used as an informal evidence in constructing safety cases
- Can be seen as a skeleton for a formal proof (is subject to checking by mechanical tool).

Where *time* comes into it...

- Semi-synchronous architecture
- The change of common state can be asynchronous in various nodes
- Design leads to eventual synchrony and common state

Next course day: Architecture issues