Sporadic failures
Implications for Multicore

Franz Münz, COESA3 – Eurofighter Software Development

25.November 2013

CASSIDIAN

Agenda

1. Personal information presenter
2. Sporadic failures:
   - Definition and background
   - Example
3. Implications for multicore:
   - Evolvement of avionic systems
   - Evolvement of critical sporadic failures
   - Consequence of critical sporadic failure
4. Conclusion

Personal information

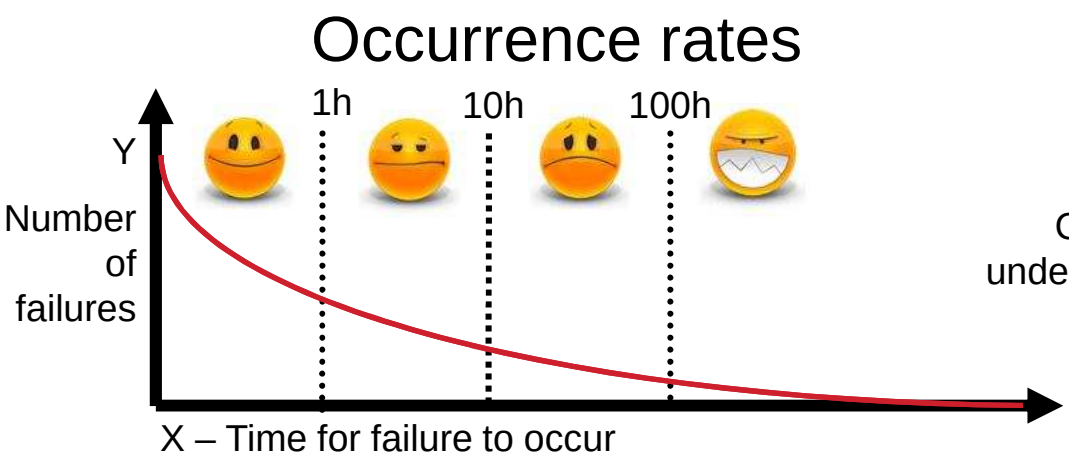
Franz Muenz

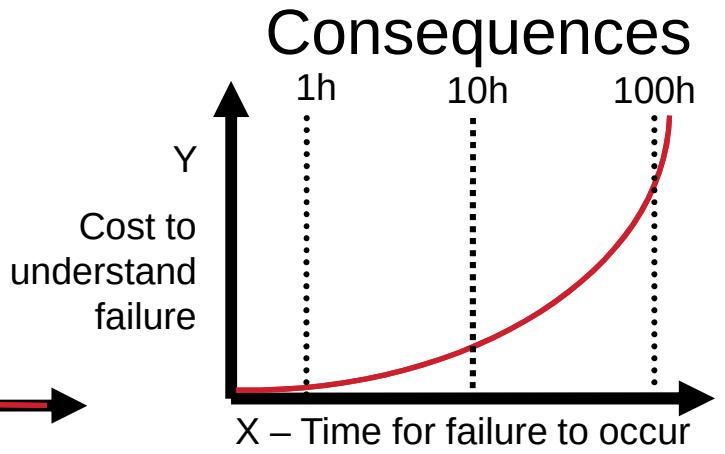
- CASSIDIAN employee since 2006
- Software design engineer
- Experience:
  - Attack Computer Software (esp. bus controller)
  - HW/SW and SW/SW integration support
  - Equipment engineering support for Attack/Navigation Computer (AC/NC)
  - Flight test and production support
  - Critical error analysis, debugging and assessment
  - Multicore Debugging Infrastructure technology project
- Contact
  - +49 (0) 8459-81-66708
  - Franz.Muenz@cassidian.com

Sporadic failures

Definition and background

Sporadic = Not easy to reproduce
Failure = Observable adverse effect





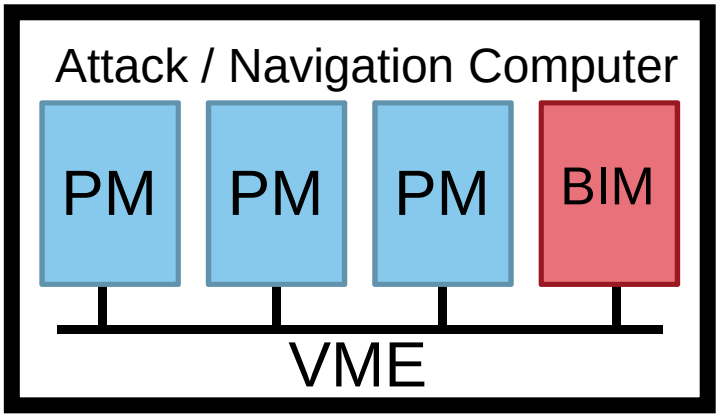
- Reasons for sporadic failures:
  - Signal / Software timings
  - System load
  - Specific scenario / input
  - Environmental conditions e.g. temperature
  - Disturbances e.g. Single Event Upsets (SEU) due to radiation

CASSIDIAN

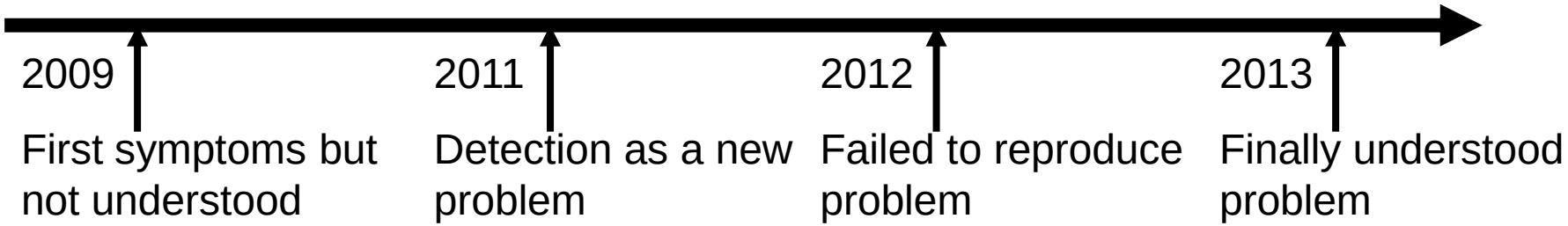
Sporadic failures

Example

Sporadic dead AC/NC bus interface module



- Symptoms:
  - Communication to bus interface module (BIM) fails
  - Hardware watchdog timeout on processing module (PM)
  - Software task monitoring timeouts on PMs
  - Processor on BIM seems to stop from PM point of view
  - No logging / detection on BIM
- Known hints:
  - Some AC/NC show the failures more often, some less often (=> HW problem)
  - Failure does not happen without involvement of other real computers
  - Heavily scenario dependent
- Timeline:



Sporadic failures

Restrictions

Why it took so long

- Very low occurrence rate on average
  - Delayed detection of systematic failure
- Moderate occurrence rate worst case
  - Prevented reproducing failure in economical time frame
- Only occurs in environments where instrumentation is not allowed
  - System integration RIG (rarely available to isolate a single failure)
  - On aircraft (don’t touch this... ever)
- Formalizing instrumentation is also not allowed / feasible

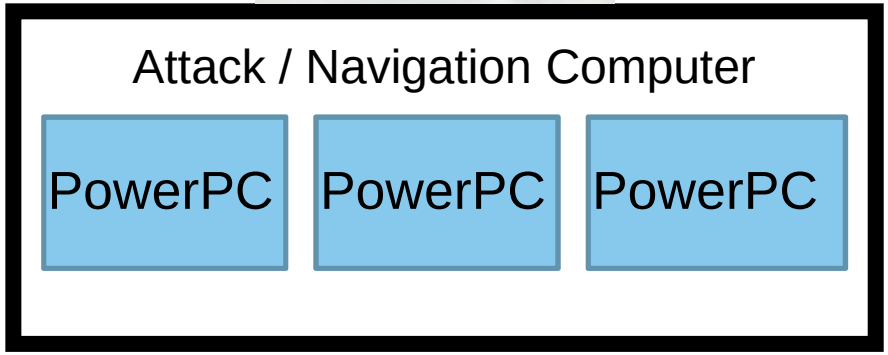
CASSIDIAN

Implications for multicore

Evolvement of avionic systems

Today

- Federated avionic architecture
- 40+ different mission computer
- One function per mission computer
- Example Attack / Nav Computer



Future

- All current Eurofighter **mission functions** (except Sensors and flight safety critical functions) could potentially fit on a single Freescale T4240 multicore processor

Avionic

-

Sensors

=

CASSIDIAN

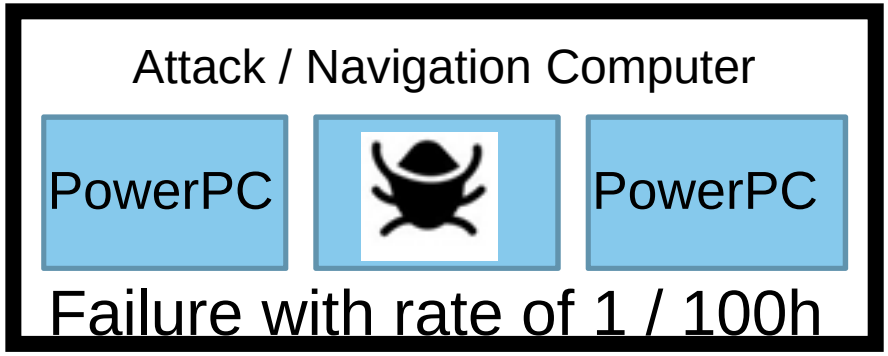
Implications for multicore

Evolvement of critical sporadic failures

Today

- Critical sporadic failures acceptable* for mission avionic if their rate is very low

  (* as effect is local and probably recoverable without affecting the rest of the system)



Future

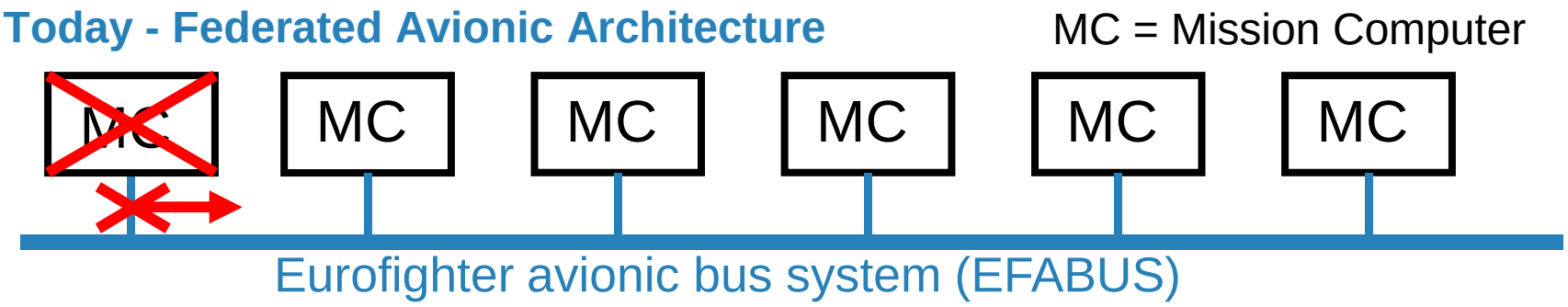
- If all mission avionic functions run on a single multicore processor all sporadic failures accumulate
- Increased complexity of multicore SoC also increases number sporadic failures
- > 40x failure with rate of 1 / 100h is **unacceptable**

CASSIDIAN

Implications for multicore

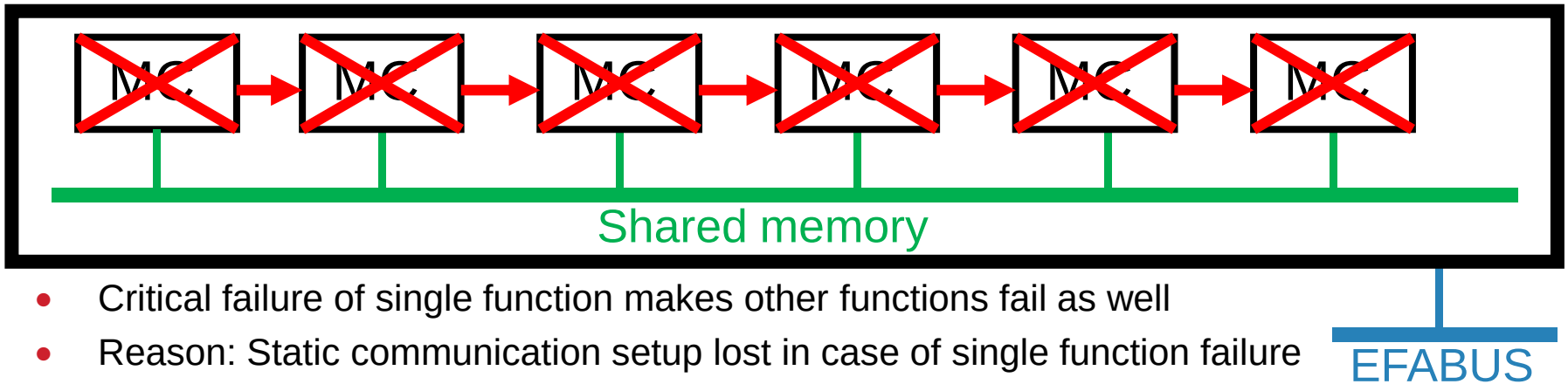
Consequence of critical sporadic failure

Today - Federated Avionic Architecture



- Failure of single mission computer does not make other mission computers fail

Future – Integrated Avionic Architecture on Multicore



- Critical failure of single function makes other functions fail as well
- Reason: Static communication setup lost in case of single function failure
- If function crashes any blocking communication towards it cannot be resolved

CASSIDIAN

Conclusion

How to deal with sporadic failures in the future?

Multicore debugging requirements

- Root causes of sporadic failures must be identified quicker and more efficiently
- New multicore observation method which needs to:
  - support analysis of trace data in real-time
  - be non-intrusive
  - trace processors for many hours not just a few seconds
  - be capable of triggering on complex conditions which may have a high temporal spread
  - adaptable in terms of observation focus without changing the software
  - be capable to observe multiple failures concurrently due to low occurrence rate of single failure
  - support high level description of trigger conditions and post-processing actions of trace extracts
  - be part of the system, not an external device – without affecting reliability of the host system
- Multicore processors useable for mission avionic must support observation
  - Example: Freescale QorIQ processors with NEXUS/Aurora

CASSIDIAN

Thank you

for your attention!

CASSIDIAN