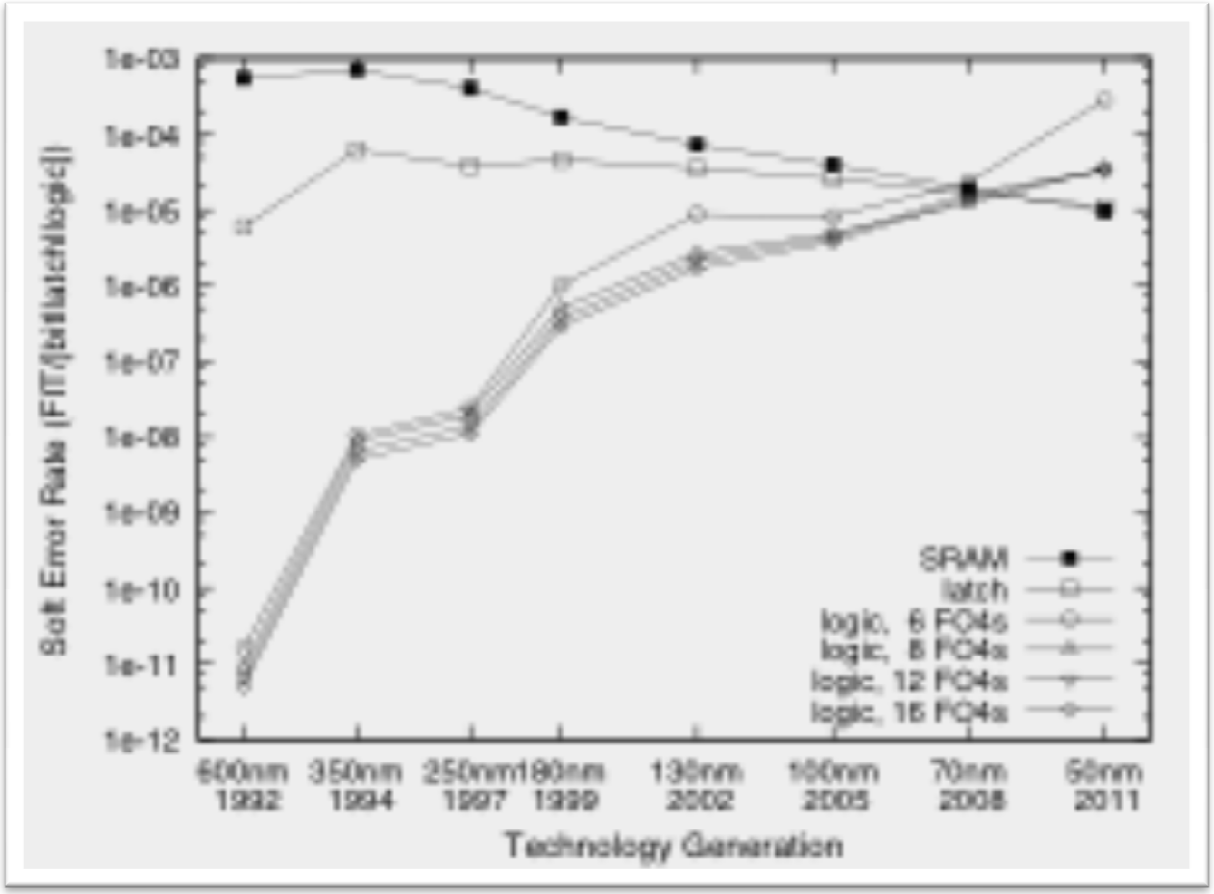
Robustness in Real-Time Systems

VáclavMikolášek

The Need for Robustness

- Many embedded systems are developed under the assumptions that
  - Hardware works always as described by documentation
  - Software is free of design errors
  - Clients use the system as specified
    - » Fragile system
- On a chip, soft error rate increased by *nine orders of magnitude* during the shift from 600ηm to 50ηm technology! [Shivakumar et al.]

The Need for Robustness



Taken from [Shivakumar et al.]
2002

Robustness – Definition

- The capability of a system to deliver an *acceptable* level of services despite the occurrence of internal or external perturbations
  - » We must always talk about a concrete property and a concrete perturbation
- Perturbations are essentially *unpredictable*

Stability and Fault-tolerance vs. Robustness

- Stability is a narrower concept
  - Studies quantifiable parameters of a system
    - temperature in a room, requests load, etc.
  - Not everything can be easily quantified: organisational structure of a system
- Fault-tolerance “only” masks failures.
- We need also *recovery*, *adaptation*, and subsequently architectural paradigms – some recipe for robustness

Main Ingredients

- Resilience
  - Components tolerate temporal degradation or lack of a requested service (e.g. one missing message in a row of correct messages for an actuator)
- Adaptation
  - Controlled changes in a system's structure or behaviour can counteract *severe changes* in the environment
  - Reconfiguration (e.g. routing)
- Recovery
  - Components which crashed due to transient faults can be recovered
- Sustainability
  - Whatever the configuration of a system, the provided service must be at least acceptable

Old friends to be reconsidered

- Assumptions
  - "Any system that attempts to gain robustness solely through precognition is prone to fragility" [Gribble]
  - We have to give up some traditional assumptions
    - » asynchronous RT systems?
- Constraints
  - Don't place constraints on environment
  - Take "no" as an answer
  - Let the environment constraint you (e.g. link failures -> Internet routing mechanism -> good connectivity)
- Models
  - Independent failures?

How to go for Robustness

Some good design practices (nothing new):

- Distributed
- Hierarchical
- Simple (Cognitive complexity ensues design errors)
- Fault containment regions
- Redundancy
- Good error detection

Recovery

- Ground State:

  A state which a component enters periodically*, and which can be externalized and used for a recovery if the component crashes*

  – Ground state identification
  – Ground state monitoring
  – Recovery & Reintegration
- Recursive Restartability:

  Restartable components on all levels of a system's hierarchy – a design paradigm by [Candea et al.]

  – Fast and cheap
  – Implication: No-as-an-answer design

Thank you!

References

[Gribble] “Robustness in complex systems“, Gribble, S.D. *Hot Topics in Operating Systems, 2001. Proceedings ,* May 2001 Page(s):21 – 26

[Shivakumar et al.] “Modelling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic”, PremkishoreShivakumar, Michael Kistler, Stephen Keckler, Doug Burger and Lorenzo Alvisi, *International Conference on Dependable Systems and Networks (DSN), June 2002*

[Candea et al.] “Recursive restartability: turning the reboot sledgehammer into a scalpel“, Candea, G.; Fox, A., *Hot Topics in Operating Systems, 2001. Proceedings,* May 2001 Page(s):125 - 130