CSE423 (Fault tolerant systems)

Text: ① Dependable Computing: A Multilevel Approach — By Behrooz Parhami (Professor, ECE, UCSB) (softcopy will be given)

② Design and analysis of fault tolerant digital systems — Johnson

Class tests: ① 2 in-class ② 2 from project — coding, final report

Link for the course: https://sites.google.com/site/cse423ftsfeb2013/home

Link for project: www.ecs.umass.edu/ece/koren/FaultTolerantSystems/simulator

Dependable computing in recent years

- Ambitious projects in space and elsewhere
  - Subject to harsh environments (vibration, temperature, pressure), external influence (radiation), etc.
- More complex systems (system-on-chip) and critical applications (medicine, transportation)

Few examples of some memorable system failures

① April 2000: Computer failure halts the trading for nearly 3 hours at the London Stock Exchange on its busiest day (end of financial year)

② February 2012: Programming error doomed Russian Mars Probe, that failed to escape earth orbit due to simultaneous reboot of two subsystems.

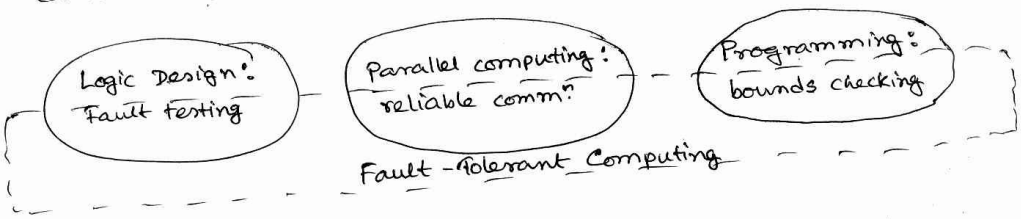
How we benefit from failures

"When a complex system succeeds, that success masks its proximity to failure..."

— Henry Petroski, "Success through Failure: The Paradox of Design"

Why the course should not be needed?

- In an ideal world, methods of dealing with faults, errors, and other impairments in h/w and s/w would be covered within every computer engineering course that has a design component.



Why the course is needed?

- The world is not ideal

Why the world in not ideal in the realm of computer engineering? (2)

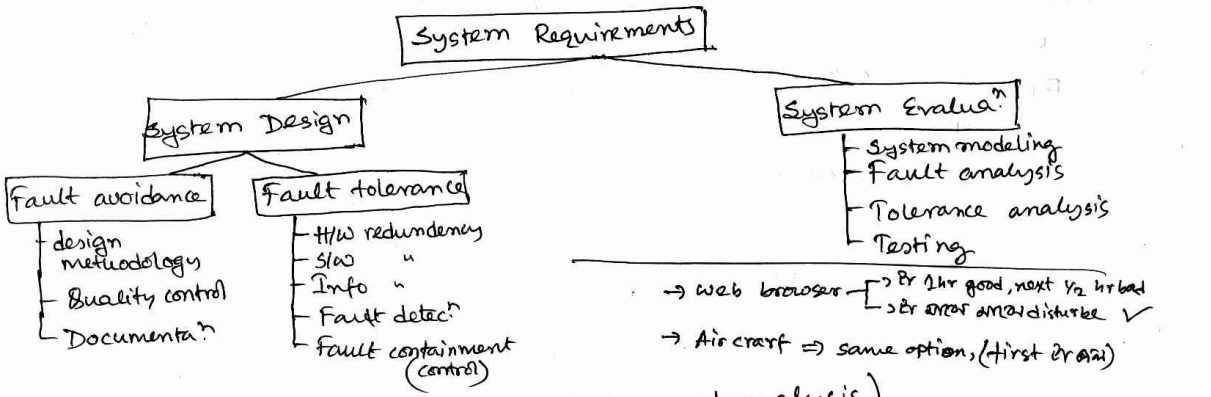
- Due to the curse of complexity (more than a billion transistors in a quad core processor)
  - ⇒ Computer engineering is the art and science of
    - translating user requirements that we do not fully understand
    - into h/w and s/w that cannot precisely analyse
    - to operate in environments that we cannot actually predict
    - all in such a way that the society at large is given no reason to suspect the extent of our ignorance.
  - ⇒ Complex systems almost certainly contain multiple design flaws: "To Engineer is human"
- Result: ~~Bad~~ Failures
  - ~~Fail~~ Failure is an unacceptable difference between expected and observed performance of a system
    - Ex: A car need not completely stop moving to show a failure

Difference between failure and fault

- Fault: A problem within system design or development that may cause the system to behave incorrectly.

Failure: A system performing unexpectedly due to having any ~~design~~ fault or/and any other reason, within or beyond control.

- Fault tolerant system: continues to perform correctly in presence of h/w or s/w faults
  - Achieved through fault avoidance and fault tolerance



Some terminologies (Needed in modeling and analysis)

⊛ Reliability, R(t): Conditional probability that a system performs ~~not~~ correctly throughout the interval $[t_0, t]$

→ conditional: depends on system being operational at beginning of interval (at t_0)

⊛ Unreliability, Q(t): Conditional probability that a system perform incorrectly throughout the interval $[t_0, t]$, given it was correct at t_0

- momentary periods of incorrect performance are unacceptable
- impossible to repair

⇒ fault tolerance is a technique of improving reliability

⇒ A fault tolerant system does not necessarily have high reliability [as R depends on t_0 and $t - t_0$]

(*) Availability, A(t) : Probability that a system is operating correctly at a particular instant of time t [available at t]

⇒ Reliability → over an interval
Availability → at an instant

→ System can be highly available yet frequently unreliable.
[if there are numerous shortly-lived failures]

- Availability is related to the extent that how quickly a system can be repaired.
  → measure of availability: fraction of time a system operates correctly

(*) Maintainability, M(t) : Probability that a failed system will be restored to operation within time t.
- ease with which a system can be repaired (important s/w goal)

(*) Safety : Probability that a system either performs correctly or discontinues without disruption to other systems or safety
- Fail-safe capability : fail in a safe manner (e.g., robust s/w)

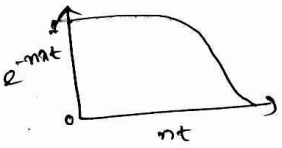
(*) Testability : Ability to test or verify
- Testing measures determines existence and quality of attributes, and thus distinguishes between faulty and fault-free systems.

Motivating case studies :

(A) How reliability gets impacted?

→ Reliability of n-transistor system, each having failure rate λ is:

$$R(t) = e^{-n\lambda t}$$

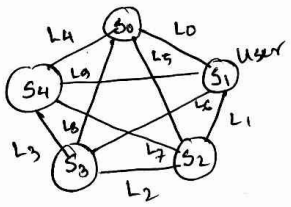


→ Apparent 3 ways of making the system more reliable:
① Reduce λ
② Reduce n
③ Reduce t

⇒ Alternative : change the reliability formula by introducing redundancy in the system

(B) How availability gets impacted? [worst case availability]

- Consider a distributed DB system having 5 sites
  - Full connectivity with dedicated links
  - Only direct comm'ns are allowed
  - Sites and links may malfunction
  - Redundancy improves availability



⇒ S : Prob of a site being available
L : Prob of a link be active

Sample values : S = 0.99, L = 0.95

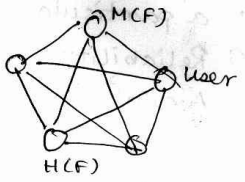
① Single copy link availability (Single copy is stored in the system and accessed through a link)

$$A = S * L = 0.94$$

(2) Data duplication: A data file F has a home or primary site H(F) and a mirror site M(F)

$A = \underbrace{S*L}_{\text{Home site available}} + \underbrace{(1-S*L)}_{\text{Home site unavailable}} * \underbrace{(S*L)}_{\text{Mirror site available}}$

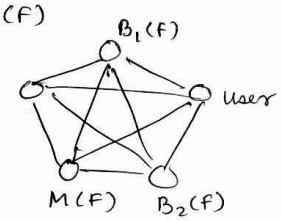
≈ 0.9965



(3) Data triplication: Home and two backups B(F)

$A = \underbrace{S*L}_{\text{Home site available}} + \underbrace{(1-S*L)}_{\text{Home site unavailable}} * \underbrace{(S*L)}_{\text{Backup1 available}} + \underbrace{(1-S*L)^2}_{\text{Home and Backup1 unavailable}} * \underbrace{(S*L)}_{\text{Backup2 available}}$

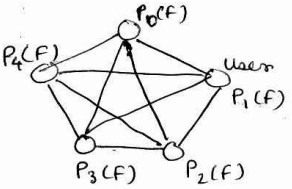
≈ 0.9998



(4) Data dispersion: Each file F is a bit string of length ℓ, which can be encoded into $5\ell/3$ bits. Encoded bits are divided equally into five pieces each of $\ell/3$ bits. Any three of the $\ell/3$-bit pieces are sufficient to reconstruct the original file.

$A = \underbrace{(S*L)^4}_{\text{all 4 pieces can be reached}} + \underbrace{4C_1(1-S*L)(S*L)^3}_{\text{Exactly 3 pieces can be reached}} + \underbrace{4C_2(1-S*L)^2(S*L)^2}_{\text{only 2 pieces can be reached}}$

≈ 0.99992



Comparison:

A system having Moderate unavailability with smaller redundancy

Scheme	No redundancy	Duplication	Triplication	Dispersion
Availability	0.95 (5%)	0.9965 (0.35%)	0.9998 (002%)	0.9992 (0.08%)
Redundancy	0%	100%	200%	67%

[unavailability in brackets]

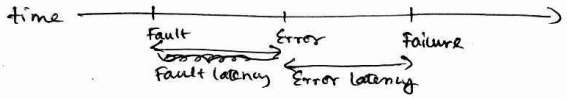
Some terminologies (needed for fault tolerant systems)

* Fault: Problem in design of development
* Error: Condition (or state) of a system containing a fault
* failure: Unexpected performance of a system

Example:

Fault → A hole in spare tire (Break fluid starts leaking)
Fault effect → flat spare tire
Error → Having a car with a flat spare tire (Break fluid pressure gets low)
failure → When a running tire becomes flat

Aspect	Impairment
structure	Fault
↓	↓
state	Error
↓	↓
Behavior	failure



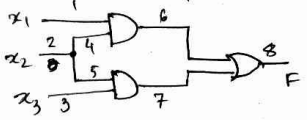
* Fault Avoidance: Technique of preventing fault. (Quality control methods)
* Fault Masking: Technique of preventing faults from introducing errors (error correcting codes)
* Fault Tolerance: Ability to continue performing tasks correctly even in the presence of a number of errors.

Fault Classes :

- Based on origin of the fault:
  ① S/W, ② H/W, ③ External, ④ Human error, ⑤ Unknown
- Based on temporal persistence:
  ① Permanent: Faults whose presence is cont^{s} and stable
  ② Transient: Faults whose presence is temporary (may resulted from environmental conditions)
  ③ Intermittent: Faults whose presence is occassional and may have recurring presences over time (due to unstable H/W or S/W (overheating of laptop)).

Fault Models :

① Stuck-At : (s.a.)
- Most commonly used fault models
- The effect of the fault is modeled by having a line segment stuck at 0 or 1
- May consider single or multiple stuck-at faults



F → correct output func^{n}
F* → Output in presence of fault(s)

$F = x_1x_2 + x_2x_3$

Line 1 s.a.0 ⇒ $F^* = x_2x_3$

Line 2 s.a.1 ⇒ $F^* = x_1 + x_3$

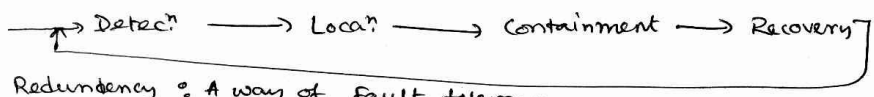
② Bridging Fault Model :

- Occurs when 2 or more lines are shorted together
- Operational effect is of a wired AND or OR

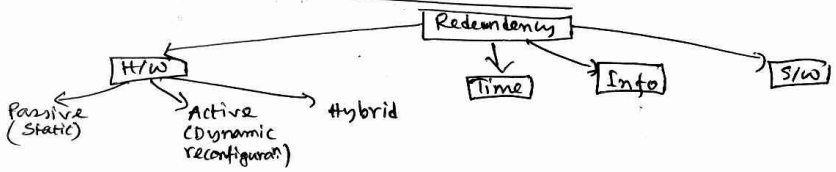
⇒ For an AND bridging fault occurring bet^{n} lines 6 and 7,

$F^* = x_1x_2x_3$

Fault tolerance cycle :

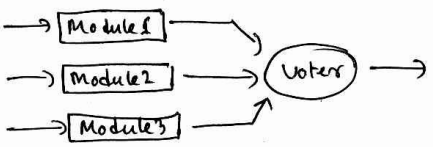


Redundancy : A way of fault tolerance

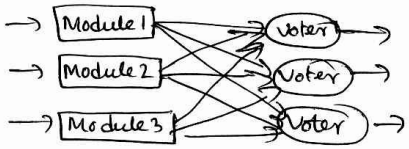


Passive redundancy (using majority voting)

Triple Modular Redundancy (TMR):

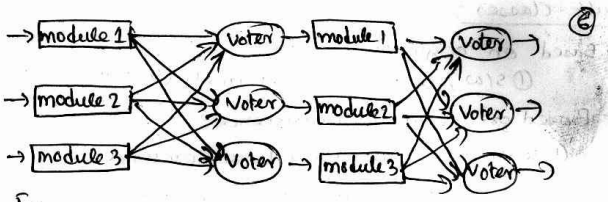


Triplicated TMR:



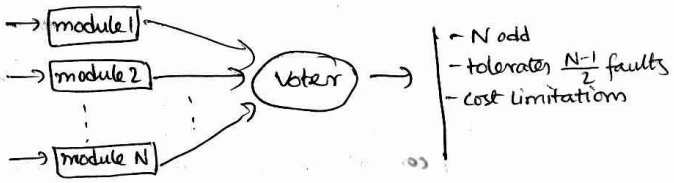
2 out of 3 system outputs are correct iff 2 out of 3 replicated voter pairs function correctly.

Multiple stages triplicated TMR:



[Errors are corrected (masked) between stages, and fault contained to one stage]

N-Modular redundancy (NMR):



Advantages of TMR

- Immediate fault masking
- Covers both permanent and transient faults in a module
- No separate fault detection needed before masking
- Conversion from simple system to TMR is easy

Challange of implementing TMR: *Synchronization between modules

- Results of 3 modules may not agree even when correct
- Margin of errors occurs when inputs come from sensors (ADC)
- If modules do any computation at all, round off error gets compounded.

Voting techniques:

- How far back one pushes the voting determines the amount of fault containment in the system
  => The sooner and the more often, the voting is applied → the less fault propag. is likely

Trade off bet. H/W and S/W fault tolerance

Technique	H/W	S/W
Speed	↑	↓
Overhead	↑	↓
Additional complexity	Must check for races (timing fault tolerance)	Must have a fault tolerant interface for each module

[Other redundancies => later in this course]