Reliability and safety in real-time systems

Examples and basic principles

Overview

- Real-Time Systems need to be reliable!
- In this slide set, we will talk about some of the techniques to make fault-tolerant systems (this is a pre-requisite to making a safe system).
- Rather than focus on the large amount of theory in this area, we'll emphasize a few examples:
  - Therac-25
  - CANDU Reactors
  - Space Shuttle
  - Modern Passenger Jets

What should you learn?

- What is the difference between reliability, security and safety?
- What are the steps from the time an error occurs to when a system fails?
- What are some of the causes of errors?
- What are some of the approaches to fault tolerance?
  - What are the differences between hardware and software schemes?

FT.com

Financial Times: March 31, 2005

Mercedes recalls 1.3m cars for quality issues

>By James Mackintosh in London

>Published: March 31 2005 17:49 | Last updated: March 31 2005 17:49

Mercedes-Benz recalled 1 in 3 of the cars it produced in the past 4 years to fix electronic problems…

Mercedes, owned by DaimlerChrysler, has seen profits plummet as a 1.2bn provision last year for the costs of fixing problems with cars already sold added to pressure from the weak dollar and losses at its Smart small car operations. In the final quarter of last year Mercedes profits dropped 97 per cent to 20m, and Eckhard Cordes, head of the division, said this

…Final quarter of last year, Mercedes's profits dropped 97%…

part of a

… sharp rise in breakdowns was due to the failure of the complex electronics in its cars…

The loss of Mercedes' coveted position near the top of quality measures has added to pressure on sales from the ageing of several models. But the launching of new cars and offroaders and a 3bn cost-saving and revenue plan is designed to get the company back to 7 per cent profit margins in two years.

…it is extremely rare to recall more cars than a company builds in a year.

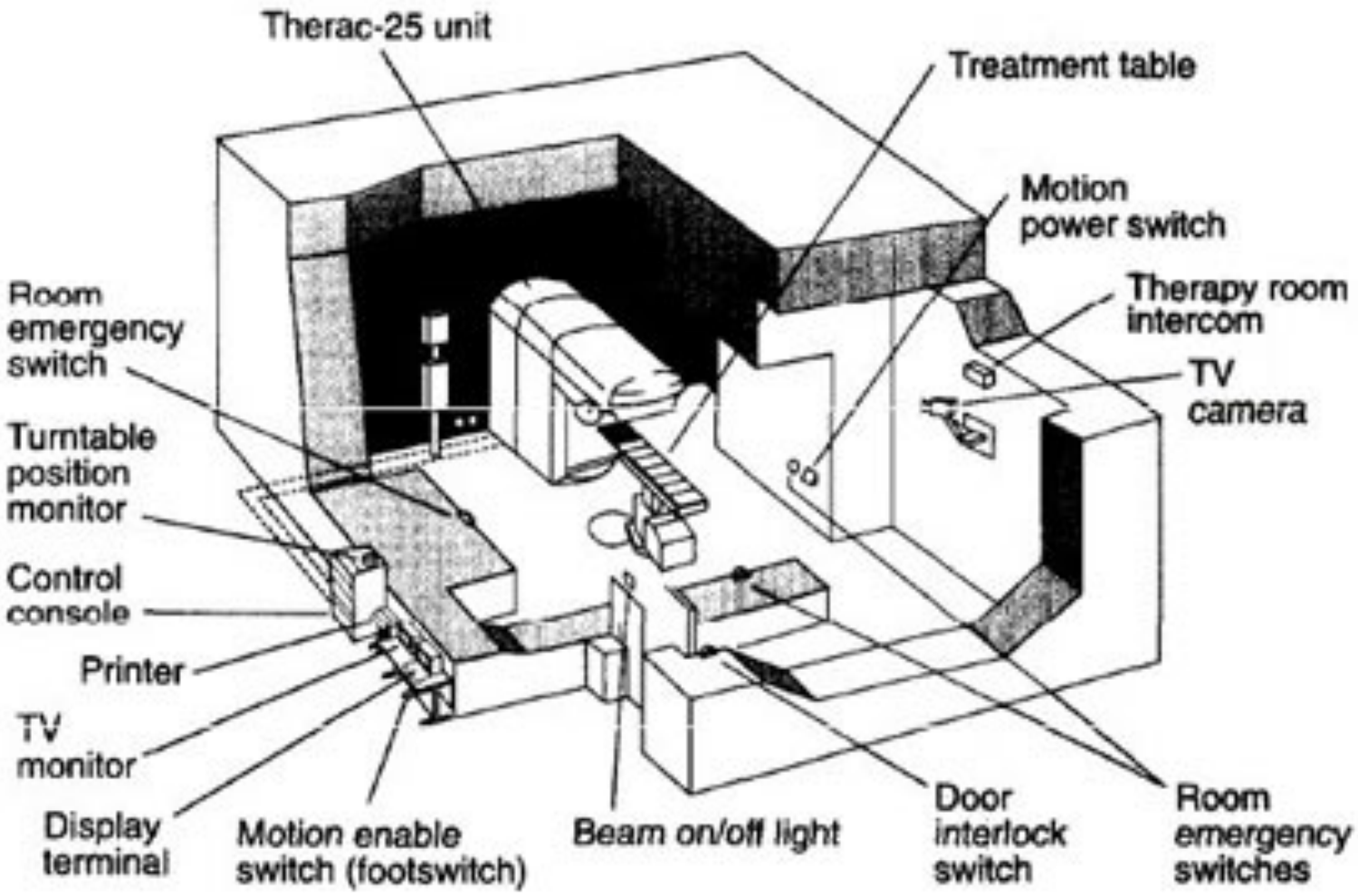
Mercedes declined to say how much the recall would cost, but analysts said it was likely to be covered by the extra warranty provisions taken last year.

Large recalls are becoming increasingly common at big carmakers as regulators and customers become more demanding. But it is extremely rare to recall more cars than a company builds in a year.

Motivating example: Therac-25

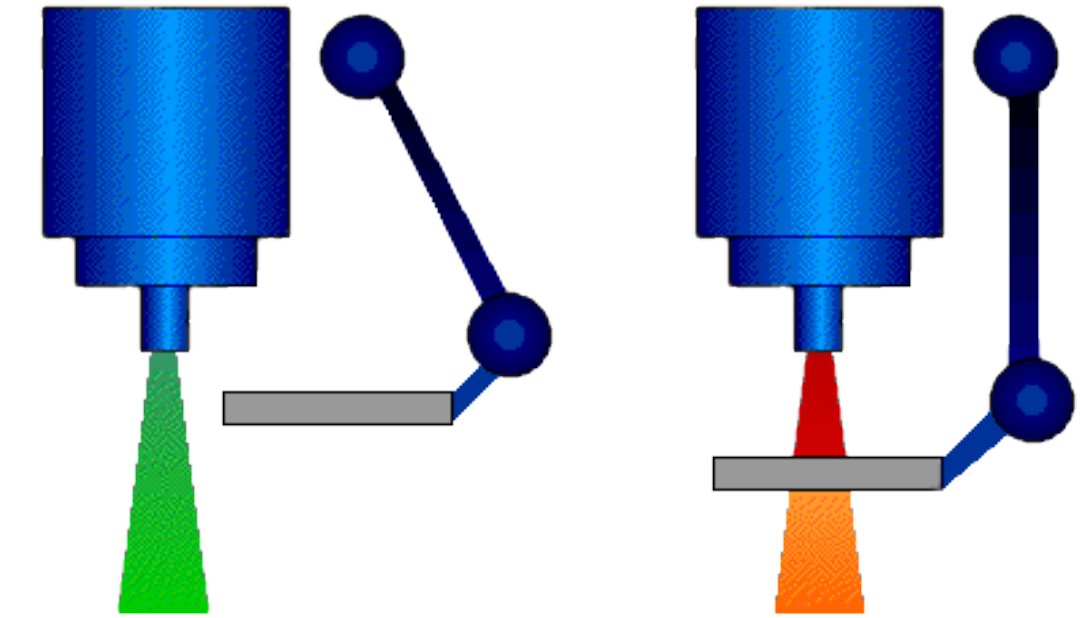
- Medical linear accelerator
  - Used to treat tumors with either:
    - Electron beams for shallow tissue
    - X-Ray beams for deep tissue
- Eleven Therac-25s were installed
  - Six in Canada
  - Five in the United States

- Developed by Atomic Energy of Canada Limited (AECL).


Therac-25 unit
Treatment table
Motion power switch
Therapy room intercom
TV camera
Room emergency switch
Turntable position monitor
Control console
Printer
TV monitor
Display terminal
Motion enable switch (footswitch)
Beam on/off light
Door interlock switch
Room emergency switches

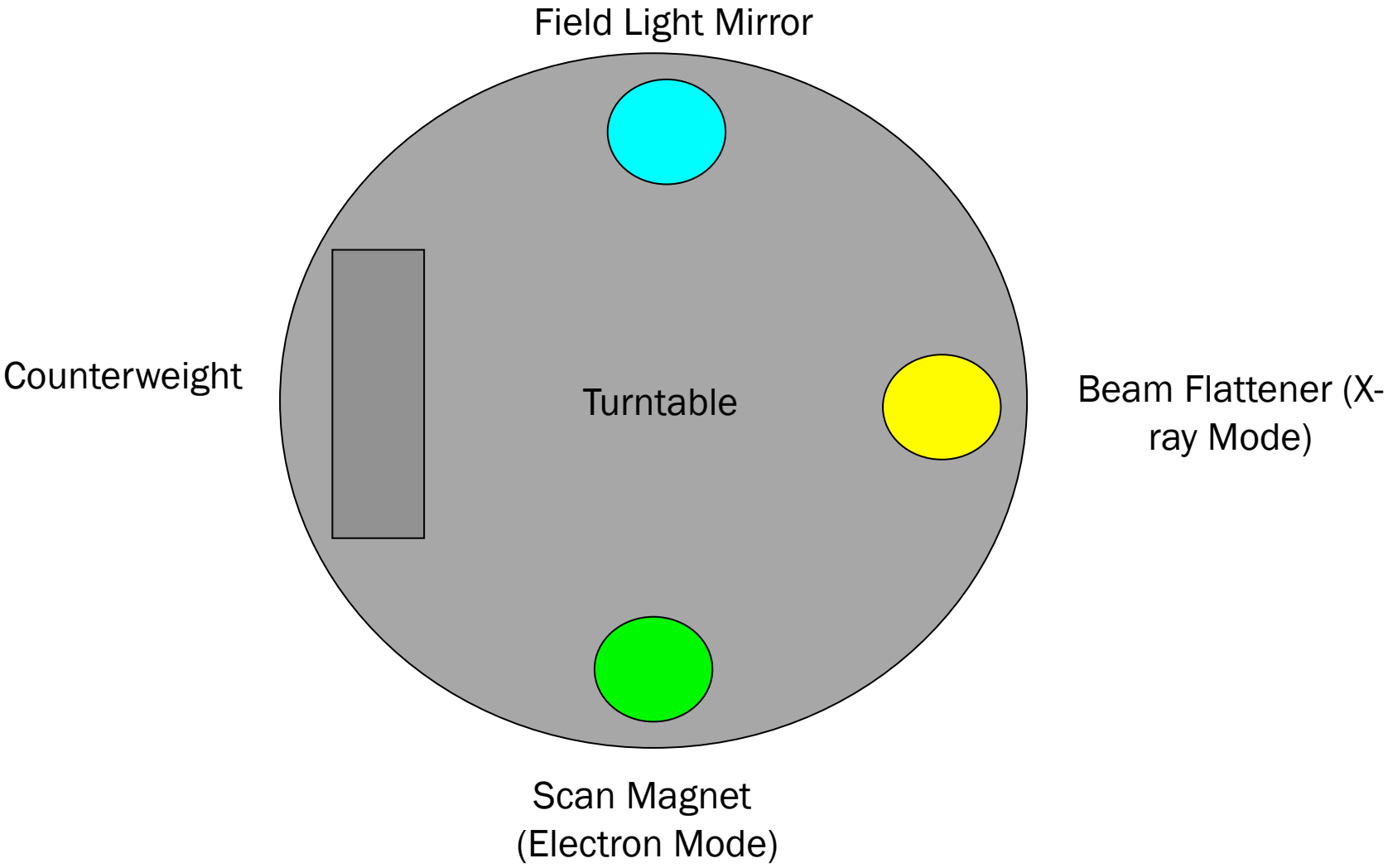
Therac-25

- Could deliver either electron therapy or X-ray therapy



Electron Mode

X-Ray Mode



Source: W. Leubke, 2003

- Six patients were delivered severe overdoses of radiation between 1985 and 1987.
  - Four of these patients died.
- Why?
  - The turntable was in the wrong position.
  - Patients were receiving x-rays without beam-scattering.

Therac-25

- How could this happen?
  - Race conditions in the software
  - Multiple threads did not lock variables properly
- Overflow error.
  - The turntable position was not checked every 256th time a certain variable was incremented.
- No hardware safety interlocks.
- User interface errors, and wrong information on console.
- Non-descriptive error messages.
  - “Malfunction 54”
  - “H-tilt”
- Too easy to just hit “P” (Proceed)

For details on this, there is a very complete article linked to the course web site.

Some definitions

- Security: A measure of confidence that the system can resist attempts to modify its behaviour.
- Reliability: A measure of confidence that the system produces accurate and consistent results.
- Safety: A measure of confidence that the system will not cause accidents.
- Security and Reliability are necessary, but not sufficient conditions of safety.

Note that safety and reliability can be in conflict.

"The safest plane is one that never leaves the ground".

Good engineering always involves a tradeoff between safety and reliability.

What should we be worried about?

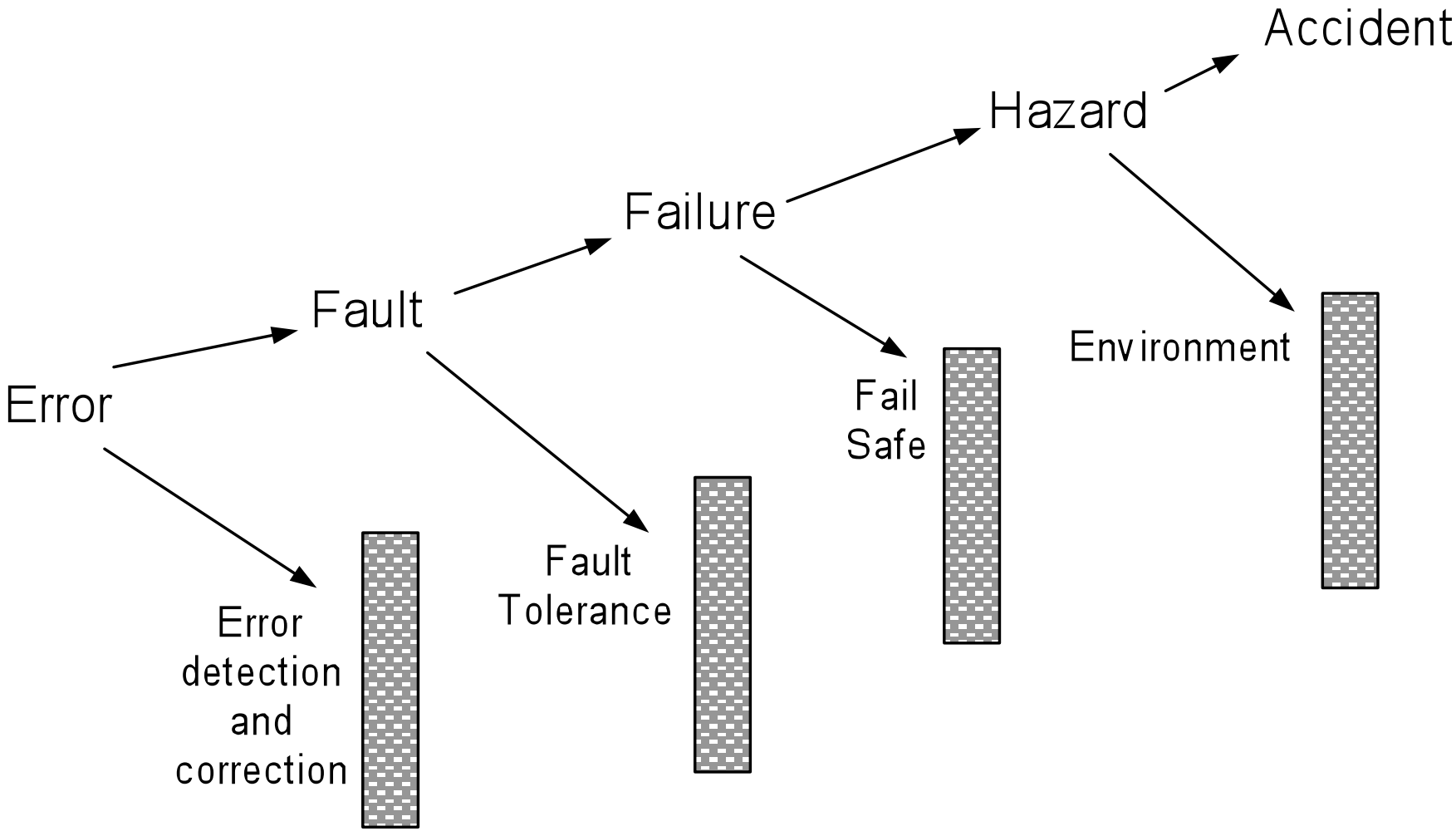
- Computers are composed of hardware, software, and data
  - The software defines the operations
  - The hardware performs the operations
  - The data records the results of the operations
- We have to worry about all three of these

	Hardware	Software	Data
Cause of Failure	Deficiencies in design, production or maintenance	Design (logic) errors	Transient Events
Occurrences	Will eventually fail	May never fail	May never fail
Failure Rates	Can be predicted in theory from physical principles	Can not be predicted from physical principles	Some upset rates can be predicted from test
Redundancy	Will improve reliability, but may be susceptible to common cause failures	Will not improve reliability, since this will only replicate same failure	May improve reliability
Diversity	Will improve reliability, should be less susceptible to common cause failures	Will improve reliability since it minimizes possibility of same error occurring in separate modules	Will improve reliability since it minimizes possibility of same error occurring in separate modules

Modified from A. Tribble, 23rd Digital Avionics System Conference

	Hardware	Software	Data
Environmental Factors	Dependant on temperature, humidity, stress, etc.	Dependent on internal environment of computer (memory, clock speed, etc.)	Dependent on both external (radiation, EMI, etc.) and internal environment (memory, clock speed, etc.)
Time Dependence	Is time dependent. Failures can be increasing, constant or decreasing	Not time dependent. Failures occur when path that contains error is executed	Is time dependent. Failures can be increasing, constant or decreasing
Wear-Out	Responsible for some failures. May be preceded by a warning.	Not responsible for any failures	Not responsible for any failures
Preventative Maintenance	Can improve reliability	Will not improve reliability, and may actually worsen it	Will not improve reliability

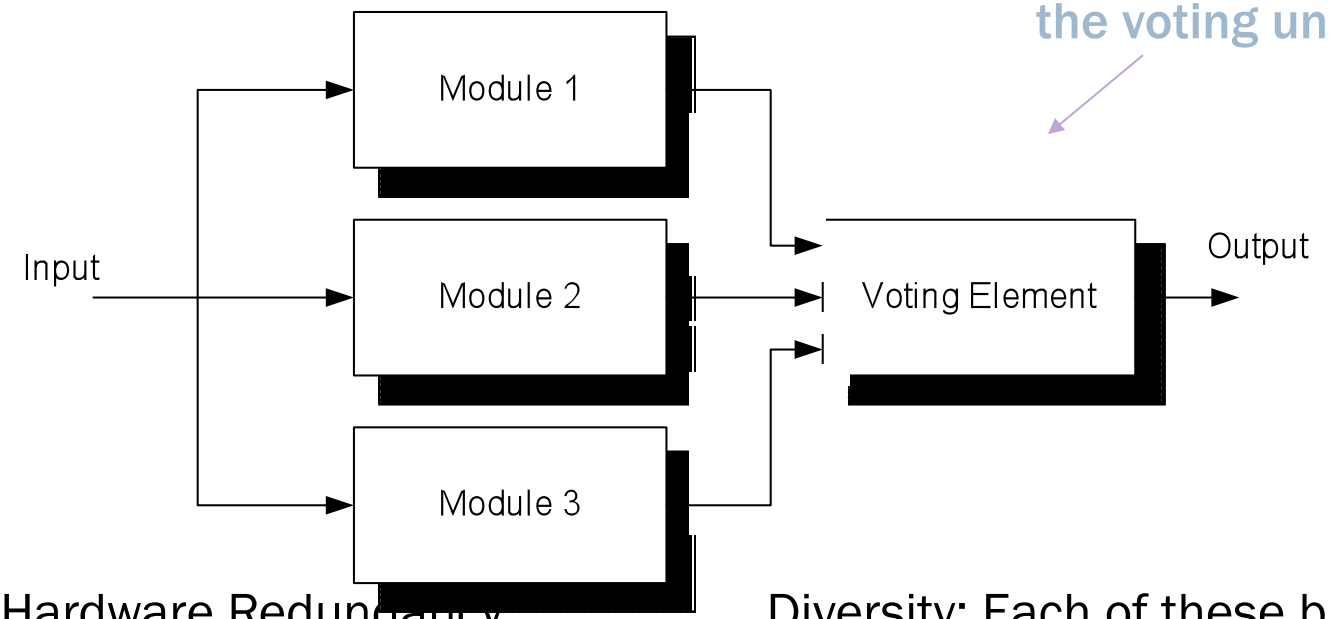
Modified from A. Tribble, 23rd Digital Avionics System Conference



Modified from A. Tribble, 23rd Digital Avionics System Conference

Fault Tolerance – Relies on Redundancy

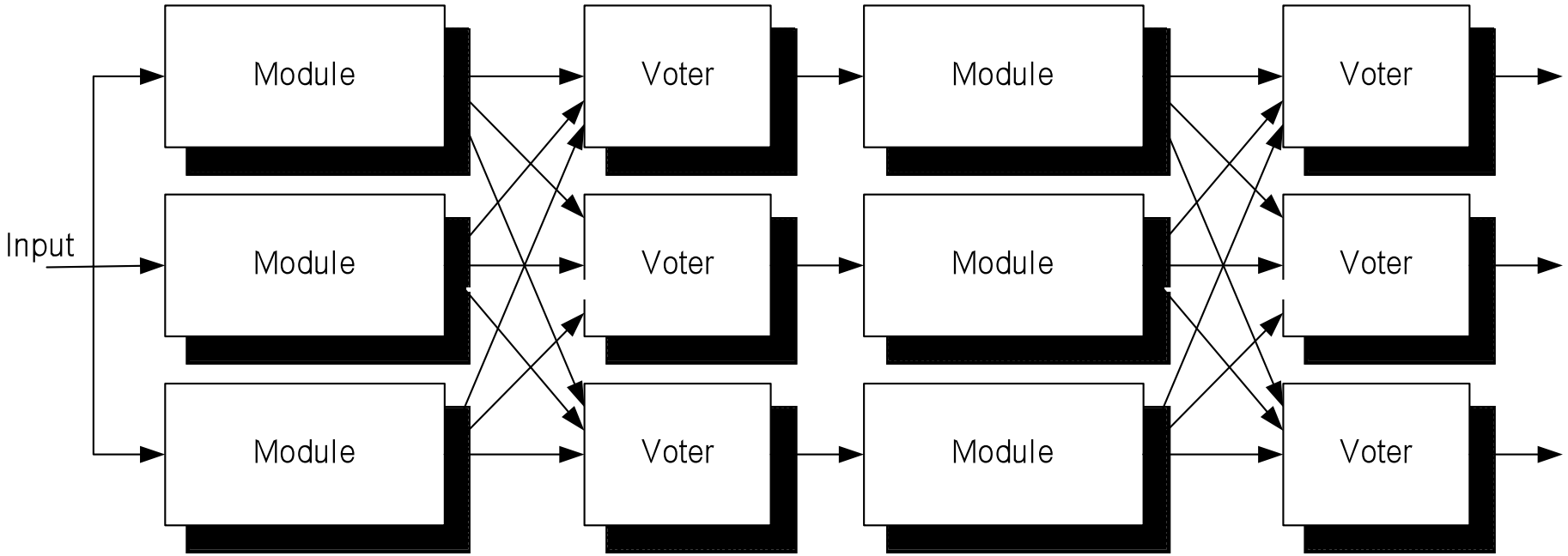
- Triple Modular Redundancy (TMR)



- Hardware Redundancy
- Software Redundancy
- Information Redundancy
- Temporal (Time) Redundancy

Diversity: Each of these blocks could be designed by a different design team using different techniques.

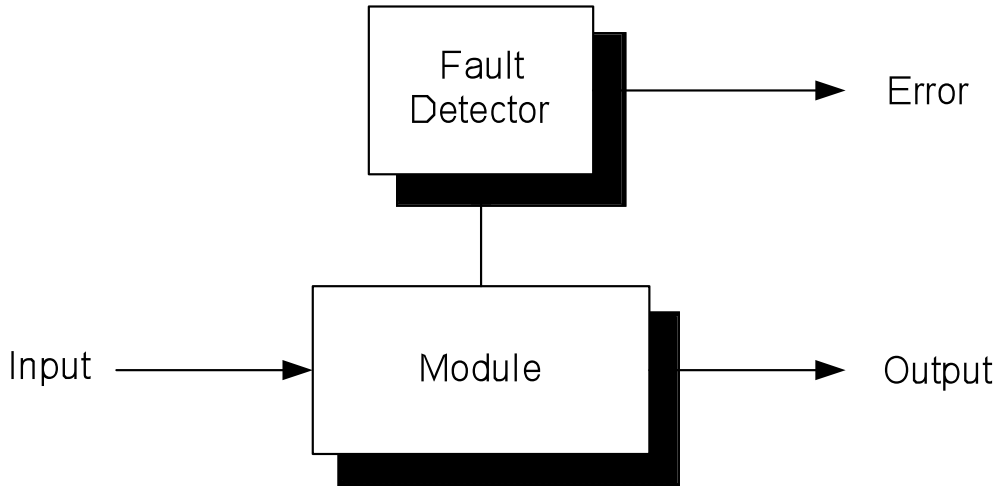
Multi-stage TMR



Problem with voters:

- Module outputs may be come valid at slightly different times
  - Differences in hardware paths
  - Differences in sensor locations

Detecting Faults



- For some applications, this might be.
- But, how do we detect a fault?

Detecting Faults

1. Functionality Checks:
   - Periodically execute code and check results
   - For example, write and read from a RAM
2. Consistency Checking:
   - Example: Range checking
3. Signal Comparison:
   - In some systems, you can compare signals at various points within a module
4. Information Redundancy:
   - Checksums, parity checking

Detecting Faults

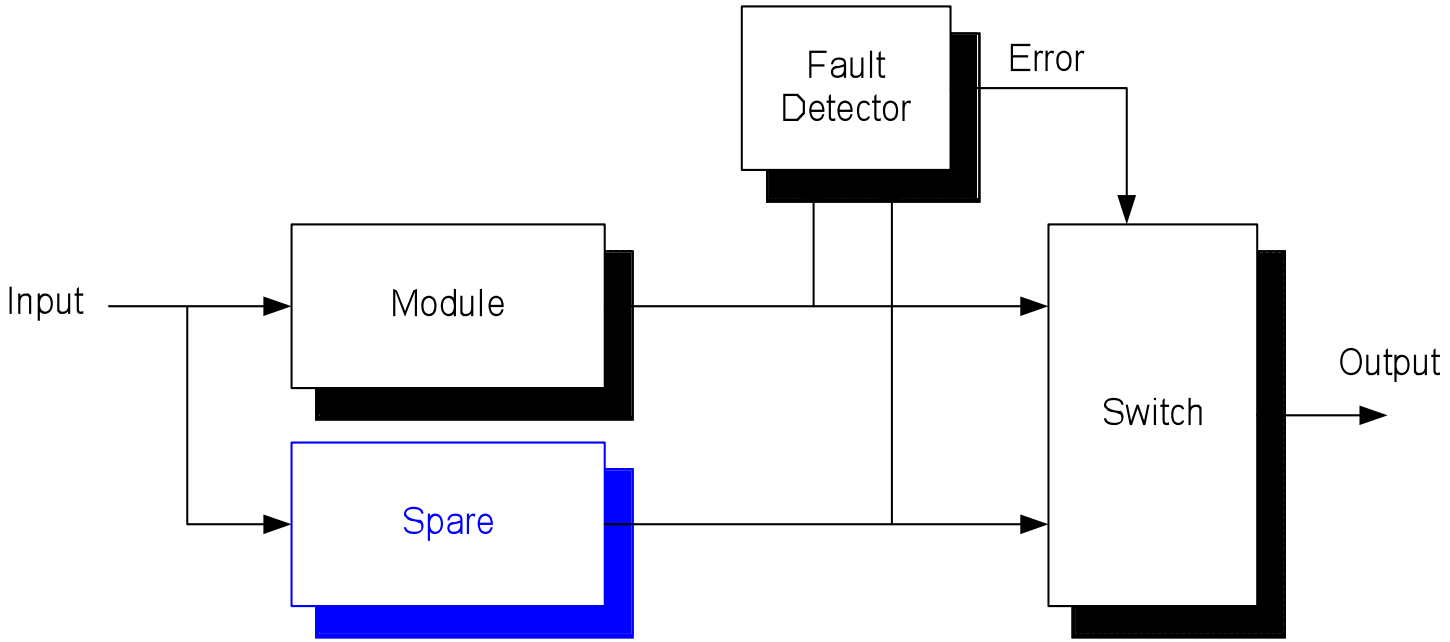
5. Instruction Monitoring:
   - If the processor fetches an illegal instruction, something is probably wrong
6. Loop-back Testing:
   - Useful for testing communication channels. Make sure what is received is the same as what is sent
7. Bus Monitoring:
   - Watch the bus and make sure that the program accesses memory within an allowable range
8. Power Supply Monitoring:
   - Possibly a dead system will draw less power
   - Also, the power supply may fail: this would cause major system failure. There might be a warning you could watch for

Detecting Faults

- 9. Watchdog Timers:
  - Detect the crash of a microprocessor by arranging a timer such that it will cause a reset (or error condition) if it is allowed to time-out
  - While the processor is operating normally, it periodically loads a value into this register
  - Problem: Time delay until fault is detected
  - Problem: It is conceivable that the system could crash in such a way that the timer is still loaded with a value, or it is possible that the watchdog timer fails

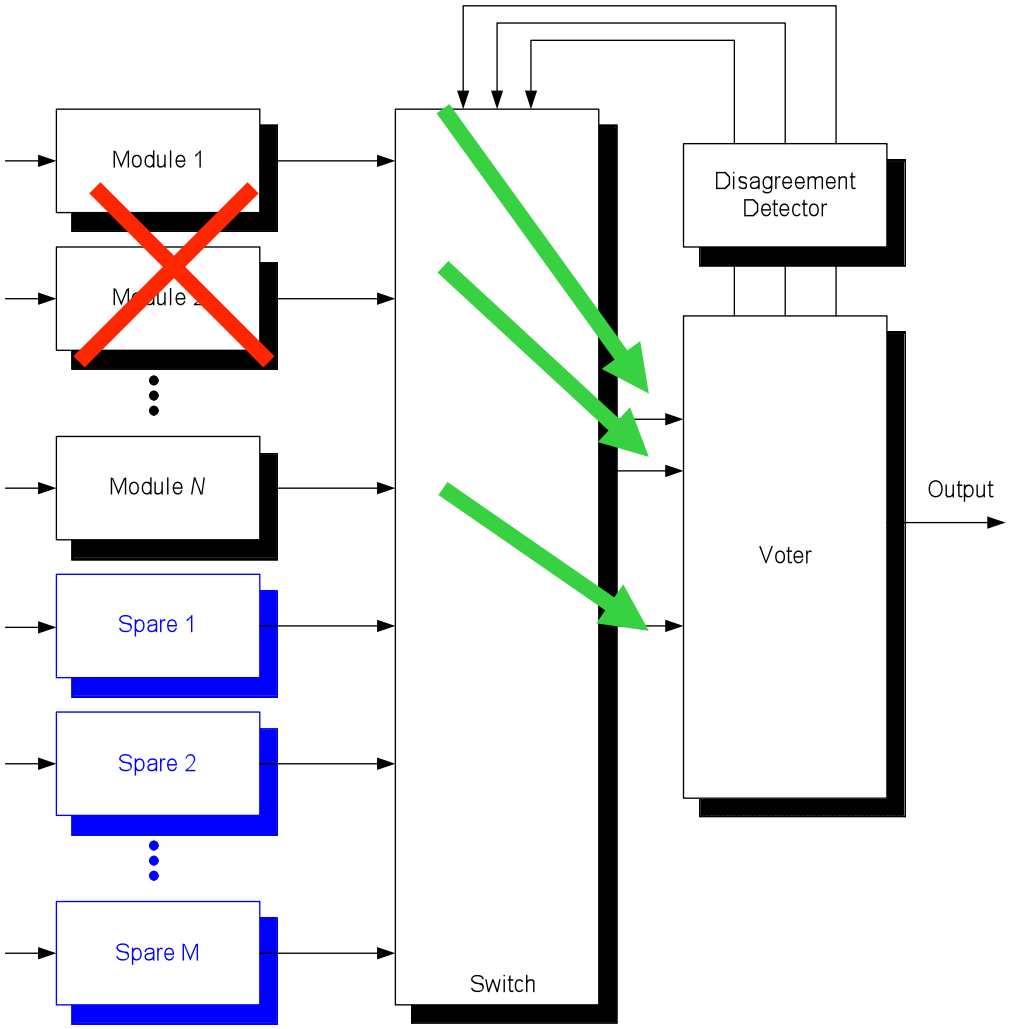
Standby Spares

- When an error is detected, switch in a spare:



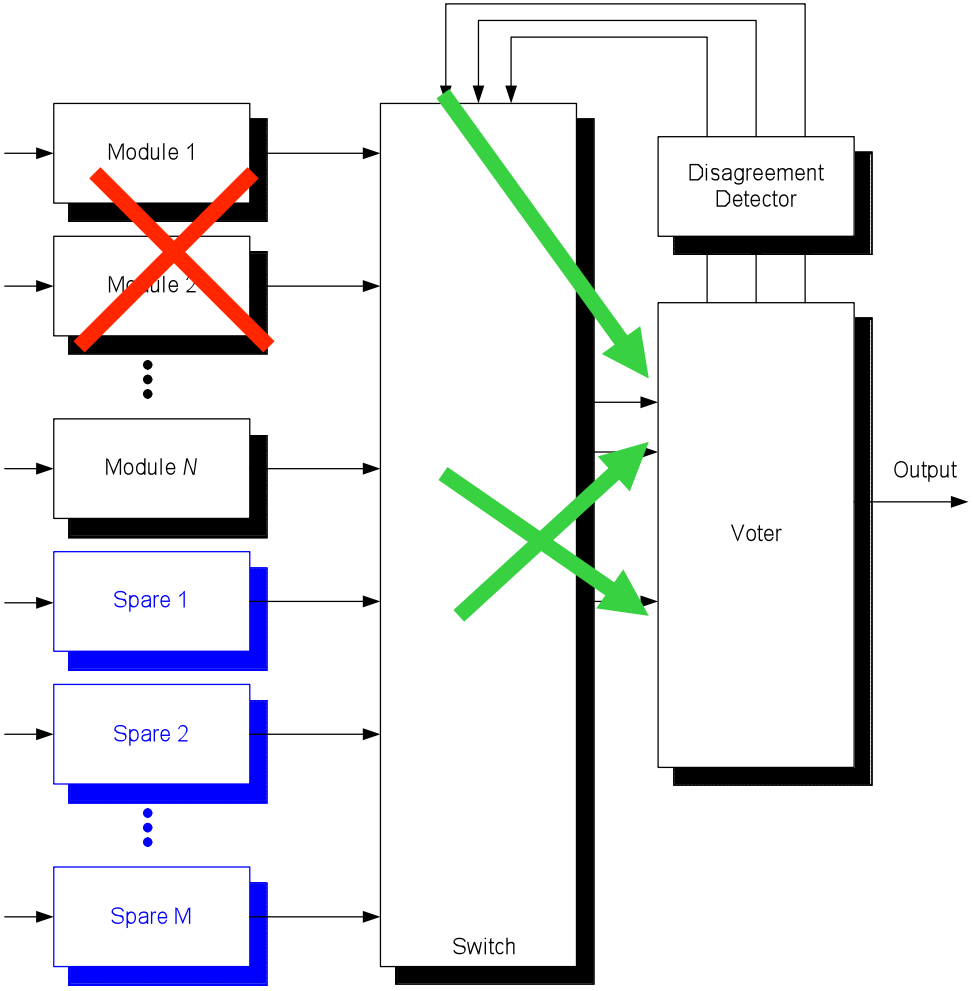
- Hot Standby: during normal operation, run the spare in parallel with the active unit. Allows for a fast transfer of control with minimum of delay

N-Modular Redundancy with Spares



Normal Operation:
Use first N modules

N-Modular Redundancy with Spares

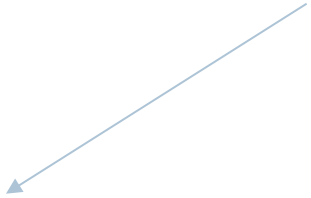


- Normal Operation: Use first N modules
- When one module fails, can switch in one of the spares.
- Note: we still are protected from faults

Software Fault-Tolerant Techniques

- Main difference between S/W and H/W fault tolerance schemes:
  - Simply replicating code and executing it 3 times doesn't help
  - If there is a bug in the code, it will happen each time
- Thus, with software, it really only makes sense if you have *different* implementations of the module
- *N*-Versions programming: have *N* different versions of the software, and execute all *N* versions
  - Significant Run-time overhead
  - Development time/cost overhead

Recovery Blocks



Problem: primary module might
have changed the state of the system
need to checkpoint at the start so
we can “roll back” system state

```
primary module
acceptance test
if (acceptance test failed) {
   secondary module
   acceptance test
   if (acceptance test failed)  give up
}
```

Data Errors

- Even if the hardware and software is fine, data can be corrupted:

- One mechanism: Single-Event Upset faults
  - Radiation continuously strikes earth
  - It is possible that a bit can be flipped

- Flipping a bit can cause memory errors, or if you are using reconfigurable logic, it can even cause circuit/ processor errors

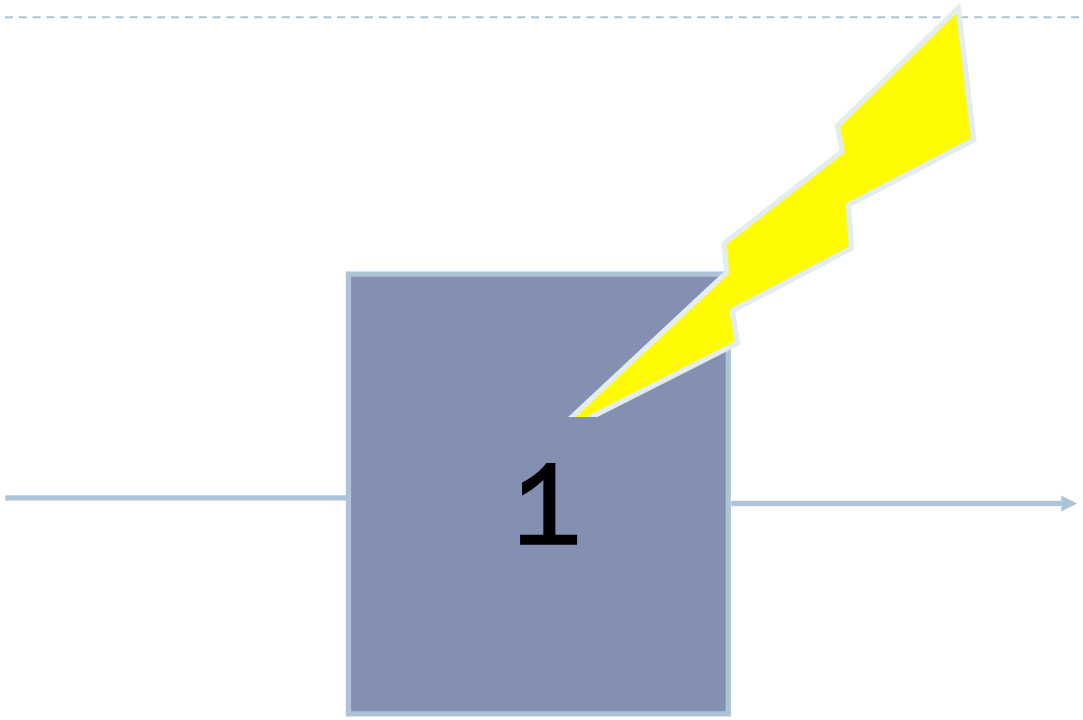
Does it really happen?

- Documented strikes in large servers found in error logs
  - Normand, “Single Event Upset at Ground Level,” IEEE Transactions on Nuclear Science, Vol. 43, No. 6, December 1996.

- Sun Microsystems, 2000 (R. Baumann, Workshop talk)
  - Cosmic ray strikes on L2 cache with defective erro protection
    - caused Sun’s flagship servers to suddenly and mysteriously crash!
  - Companies affected
    - Bell, America Online, Ebay, & dozens of other corporations

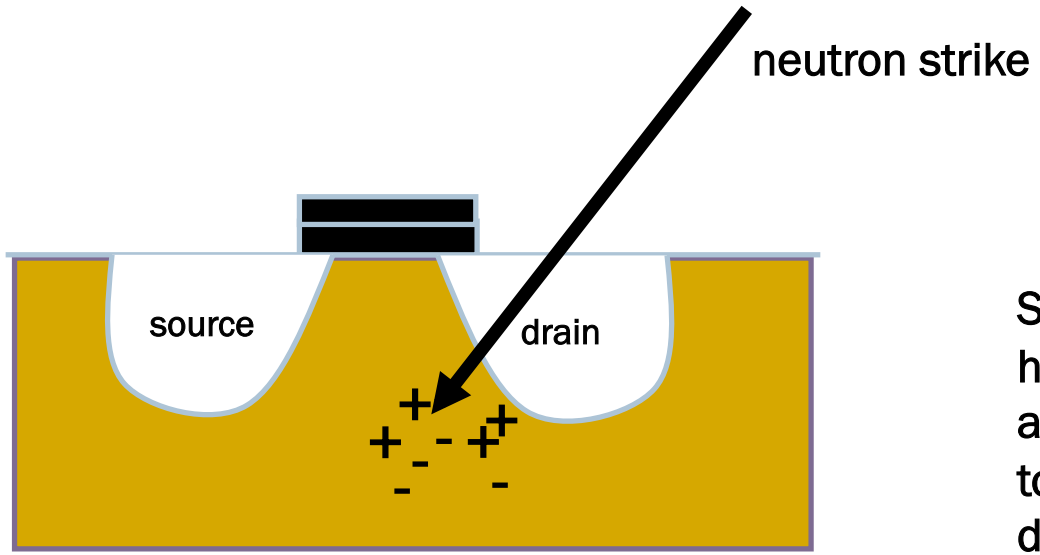
From: S. Mukherjee, Intel, HPCA 2005

Strike Changes State of a Single Bit



From: S. Mukherjee, Intel, HPCA 2005

Impact of Neutron Strike on a Si Device

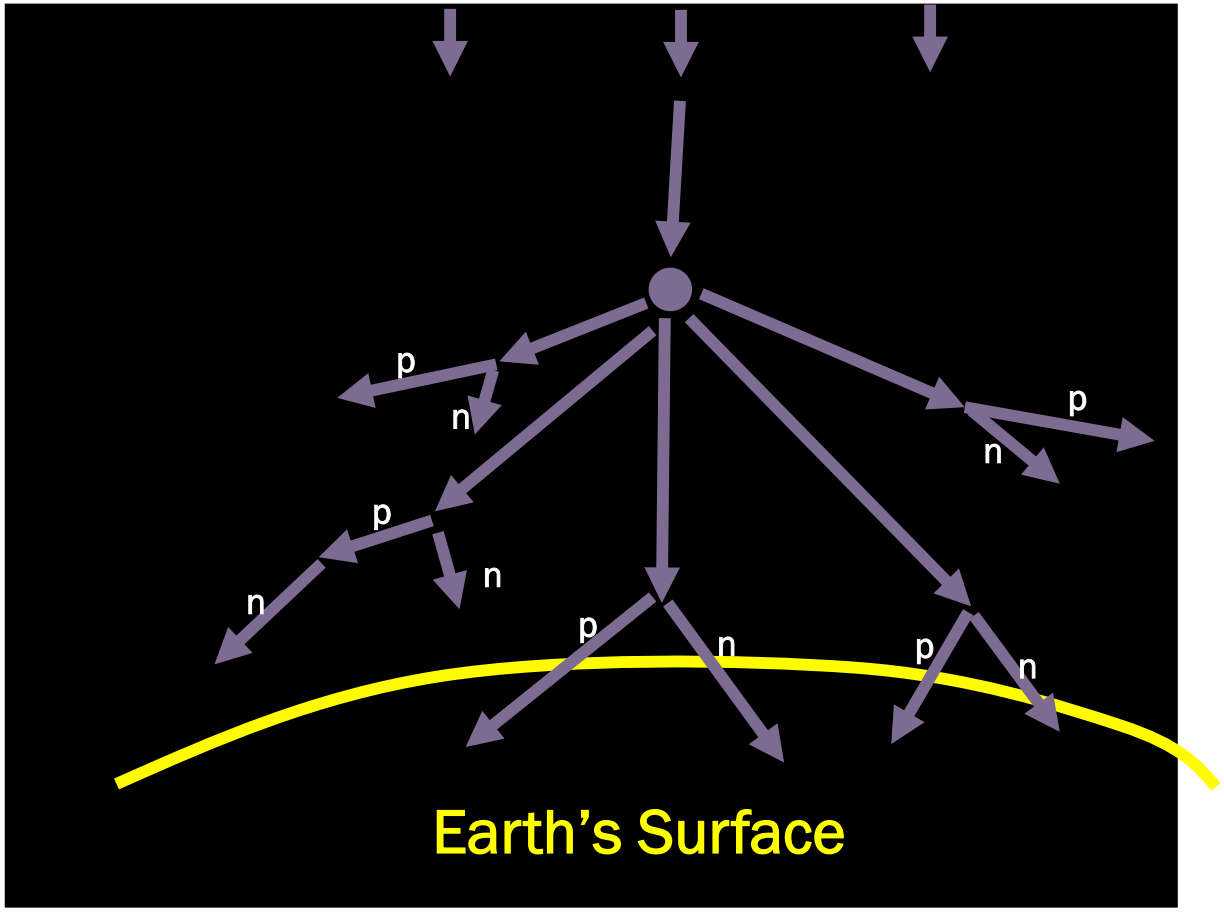


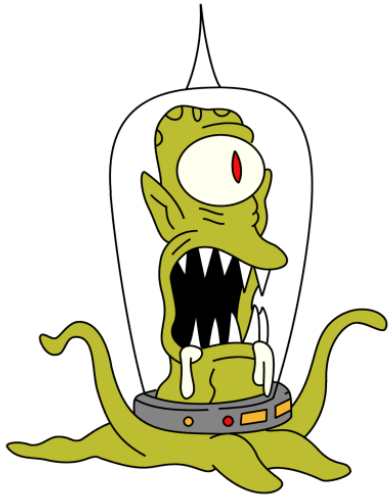
Strikes release electron & hole pairs that can be absorbed by source & drain to alter the state of the device

Secondary source of upsets: alpha particles from packaging

From: S. Mukherjee, Intel, HPCA 2005

Cosmic Rays Come From Deep Space





Neutron flux is higher in higher altitudes

From: S. Mukherjee, Intel, HPCA 2005

Impact of Elevation

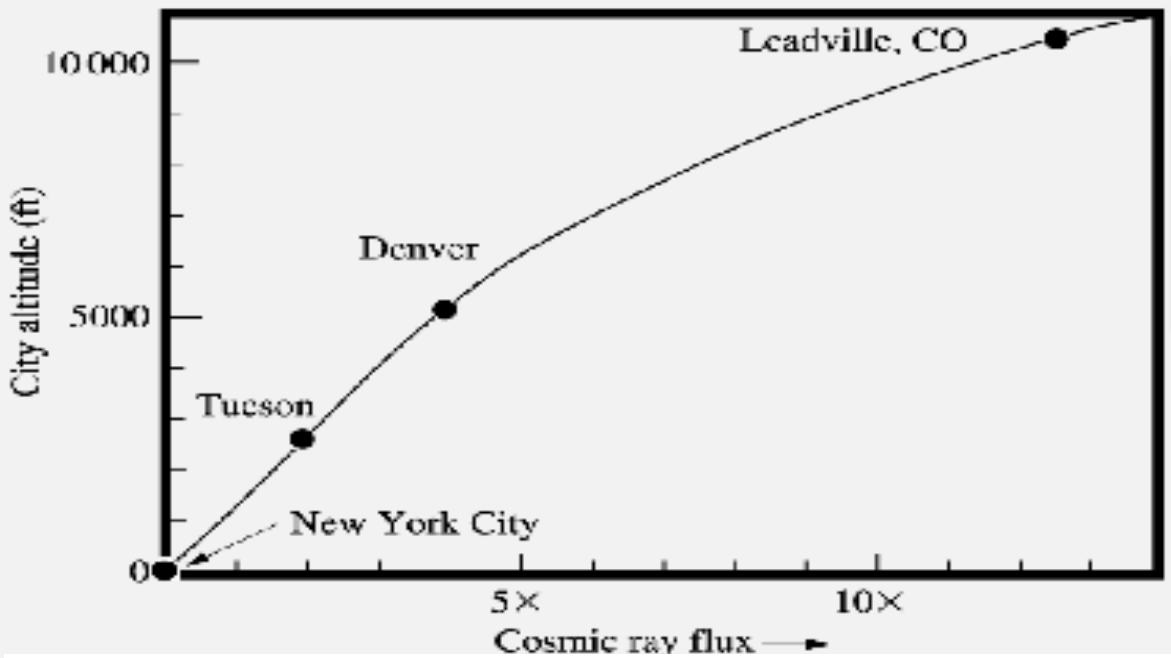


Figure 8, Ziegler, et al., "IBM experiments in soft fails in computer electronics (1978 - 1994)," IBM J. of R. & D., Vol. 40, No. 1, Jan. 1996.

3x - 5x increase in Denver at 5,000 feet

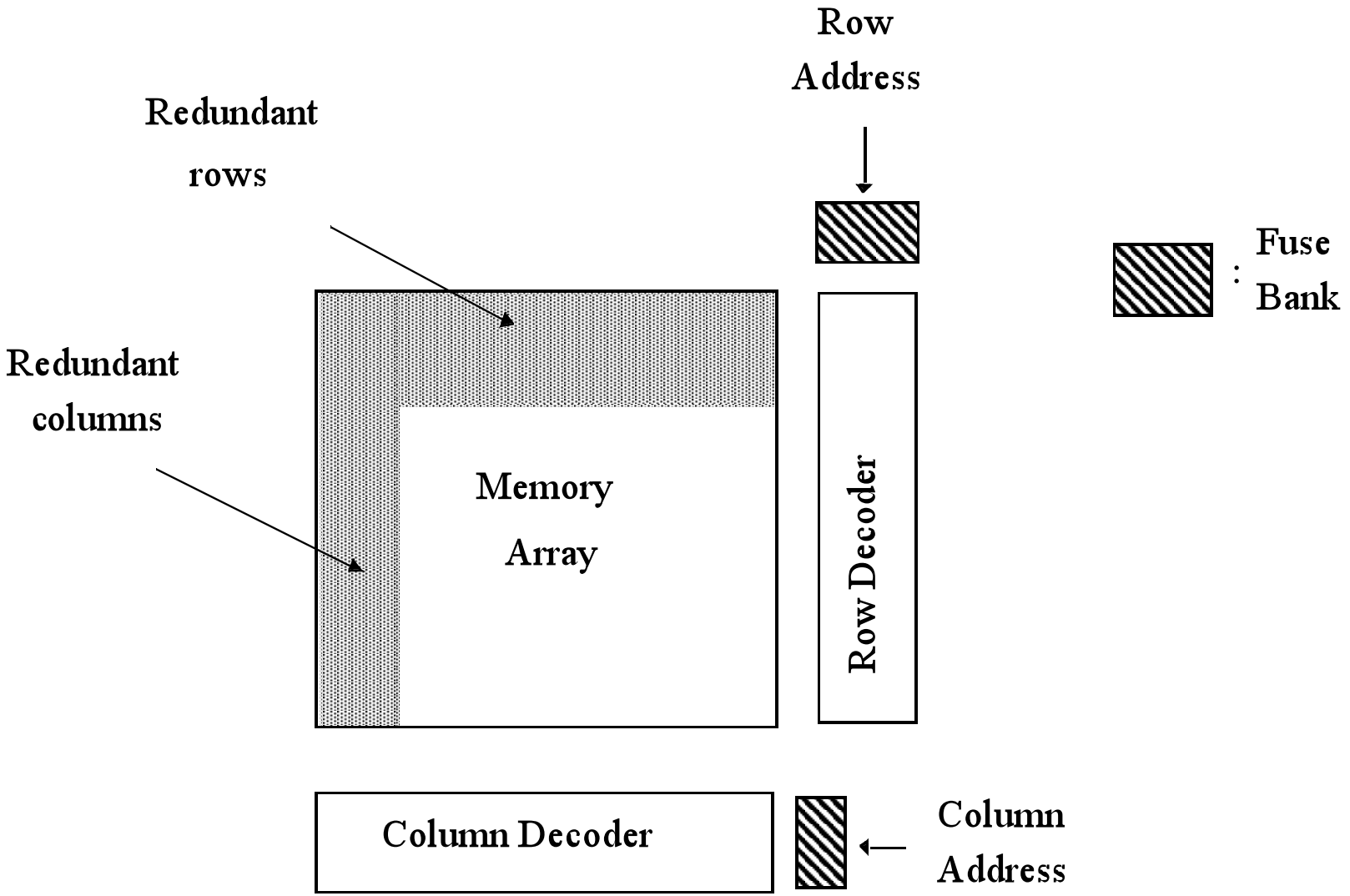
100x increase in airplanes at 30,000+ feet

From: S. Mukherjee, Intel, HPCA 2005

Error Correction/Detection

- Most memories have extra bits to detect when there has been
- an error (and possibly correct)
  - Every time you read a word, the parity bit(s) are checked
  - If there is an error, and it can not be corrected, inform the processor via an interrupt
- Note that it can affect your registers too, and registers often do not have any parity bits.
  - Any system will fail eventually
  - Quantified by “Mean Time Between Failure” (MTBF)
- FPGAs have an extra problem: the configuration (the circuit implemented in the FPGA) is stored in memory bits
  - A neutron strike can change the circuit!
  - Is this a problem? Some researchers are working on it, some say no big deal
- Actel says it is an advantage of their Anti-fuse parts

Redundancy in Memory Arrays



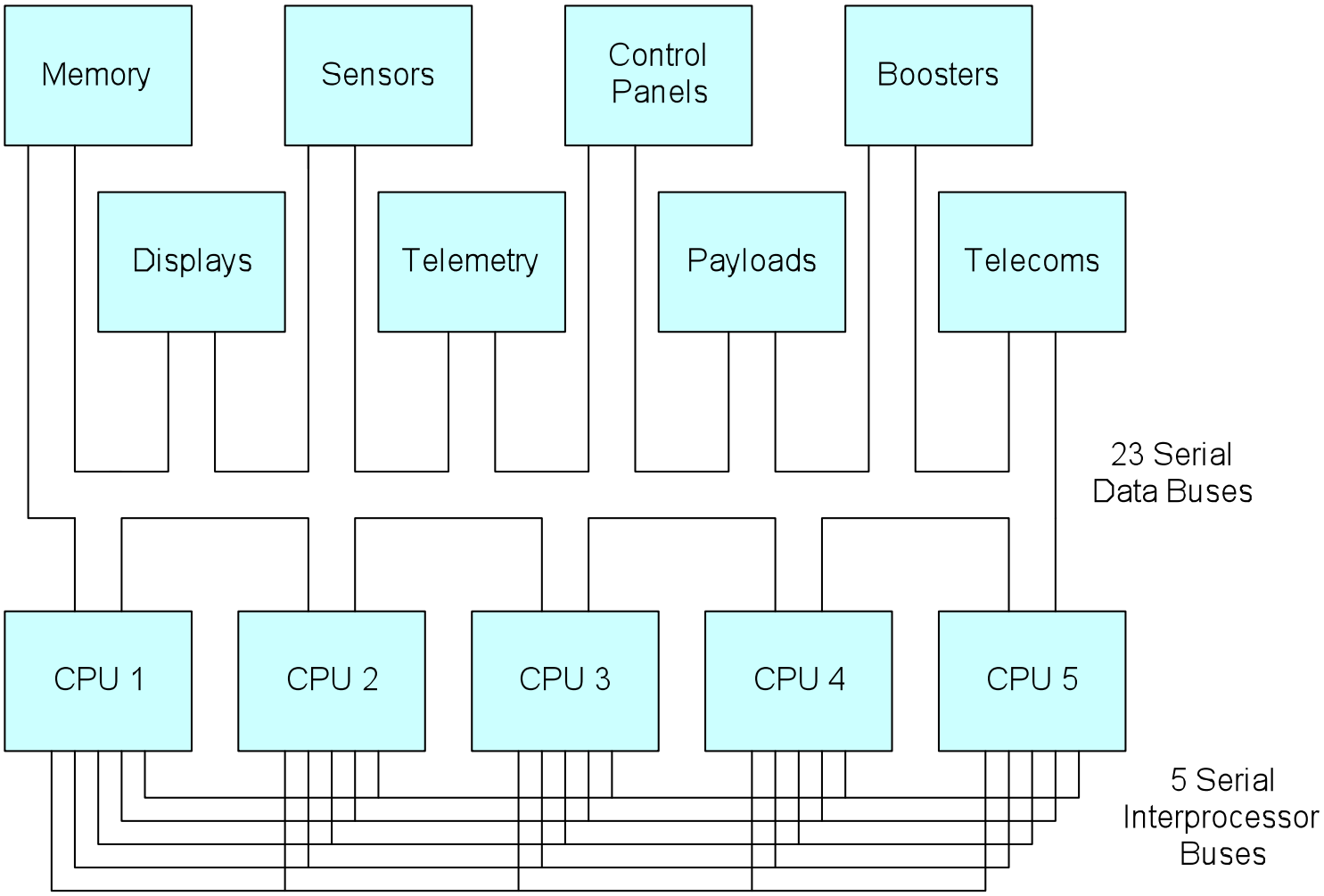
Some Example Fault-Tolerant Systems

Example: Darlington Nuclear Power Plant

- Two ways of shutting down reaction:
  - SDS1: Drop Neutron-absorbing shut-off rods into the reaction
  - SDS2: Injects liquid Gadolinium Nitrate into the reaction
- Both systems use separate sensors and separate software:
  - SDS1: 7000 lines of Fortran
  - SDS2: 13000 lines of Pascal
- Written by two different teams (but managed by the same person)

Example: Space Shuttle

Space Shuttle Computer Systems



Space Shuttle Computer Systems

- Four CPUs are configured in a N-Way Modular Redundancy scheme
  - Each CPU executes the same code
  - Hardware voting is done, but each processor also compares its results to those from its neighbour
  - If there is a disagreement, voting is used to remove the offending computer
- When one computer fails, there are three left
  - Use TMR (Triple Modular Redundancy) techniques
- When another computer fails, there are two left:
  - Two remaining computers compare their results to detect failure
- When another computer fails, there is one left:
  - Inform the crew, try to detect what the problem is and maybe fix it?

Space Shuttle Computer Systems

- The fifth computer is normally used for non-critical functions such as communications.

- In an emergency, it can take over critical operations
  - It contains flight control software written by a different contractor
  - Provides some software diversity
  - However, all processors are of the same type (potential problem)

Modern Passenger Jet Processor Architecture

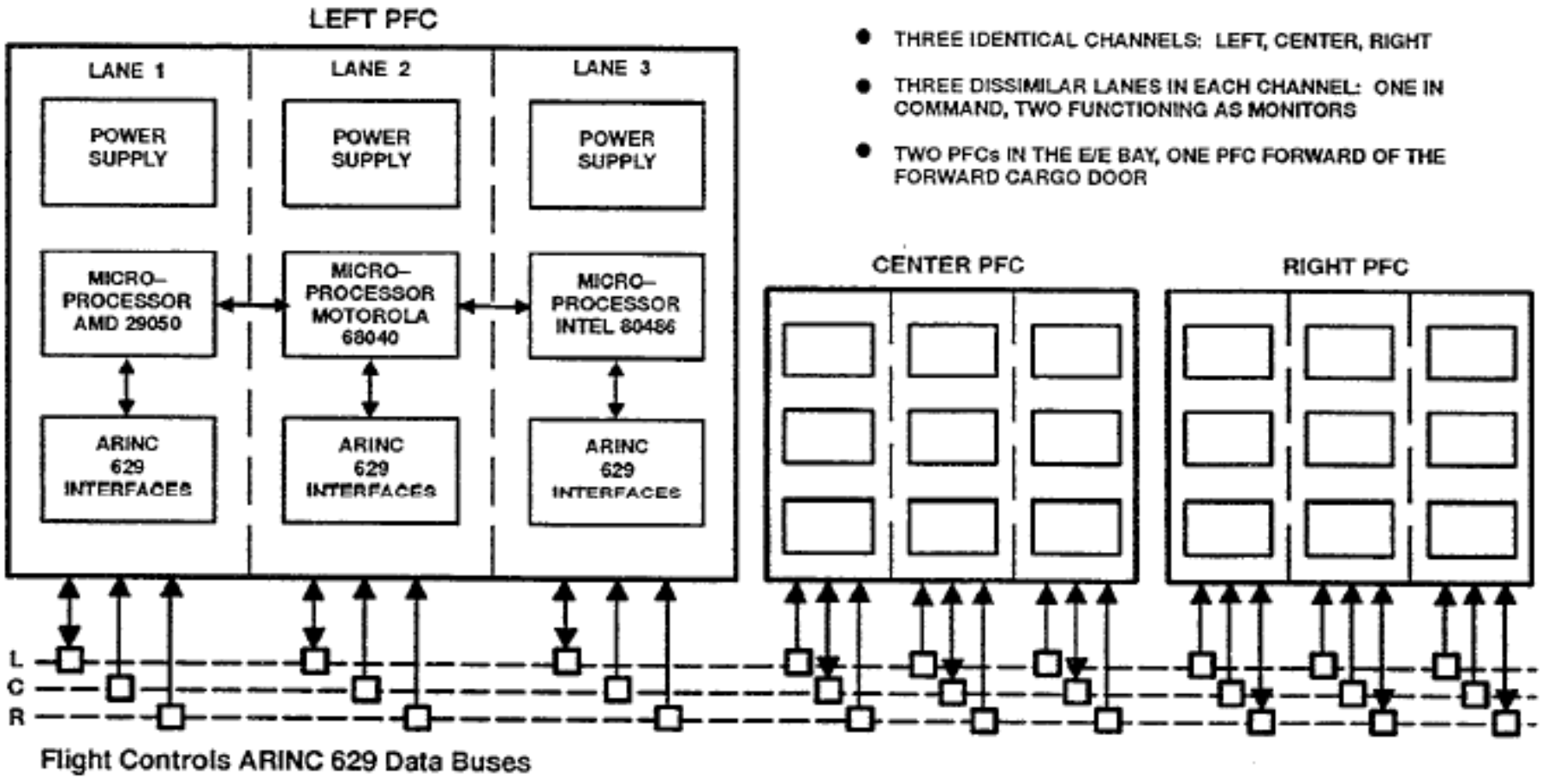


Figure 5 Primary Flight Computer Channel Architecture

Mariner 1 Venus Probe

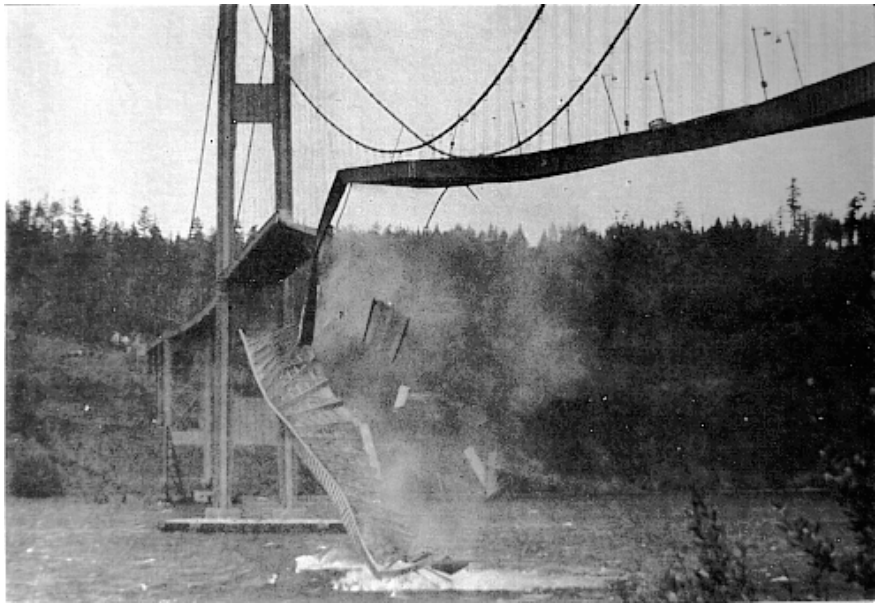
- Hardware/Software problems go way back…
- Mars 1 Venus Probe: Launched in July of 1962: “The first American attempt to send a probe to Venus. Guidance instructions from the ground stopped reaching the rocket due to a problem with its antenna, so the onboard computer took control. However, a bug in the guidance software caused the rocket to veer off course and it was destroyed by the range safety officer.”
- The problem was traced to the following line of Fortran code:
  - DO 5 K = 1. 3
- The period should have been a comma.
- An $18.5 million space exploration vehicle was lost

Learning from Other Fields…

In the 1800's, ¼ of iron truss railroad bridges failed!

Today: safety is now part of the Civil Engineering culture
- margin of safety: 3x-6x vs. calculated load
- What is the EE/CE margin of safety?

What will people in the future think of our computers?



Modified from Patterson, 2004

Estimating system reliability

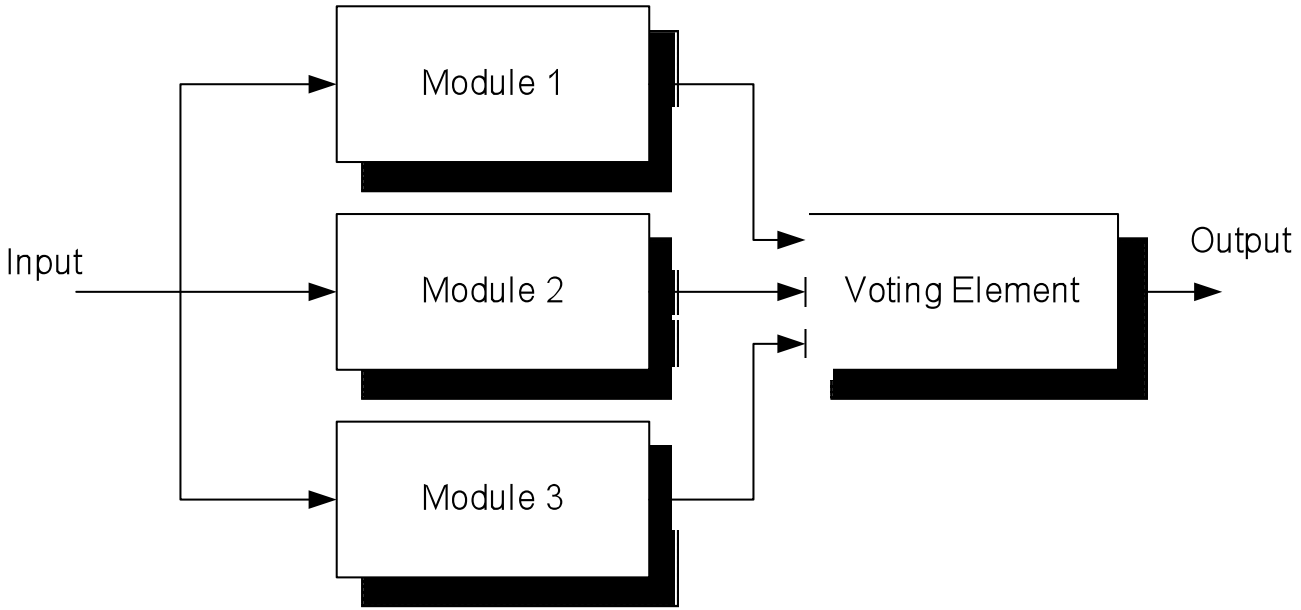
Definitions and basic mathematical modeling

EECE 494 – Design of Real-Time Systems

Overview

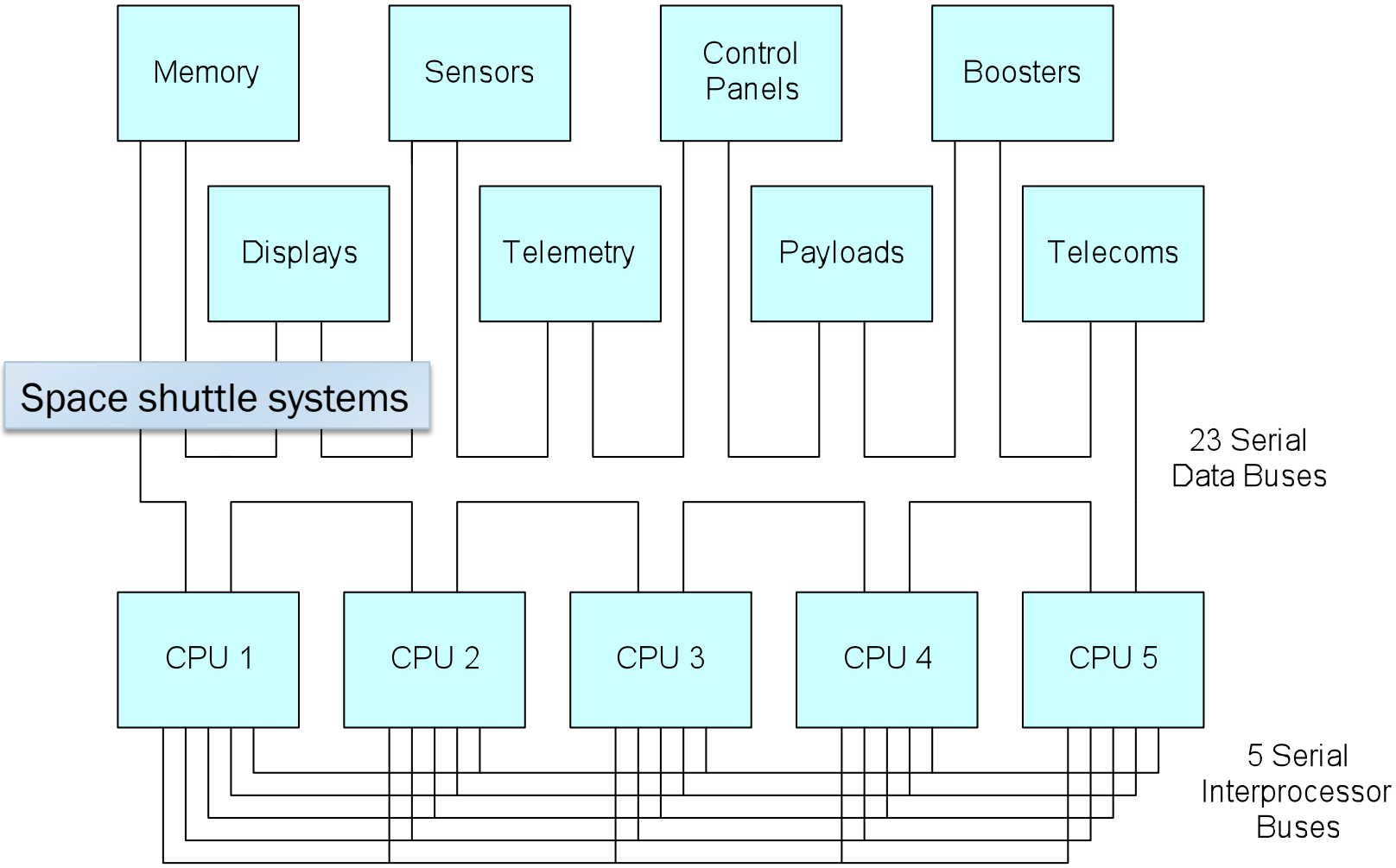
- Basic concepts
- Reliability expressions
- Mean time to failure (& mean time between failures)
- Availability
- Maintainability

Improving reliability via redundancy



Triple modular redundancy

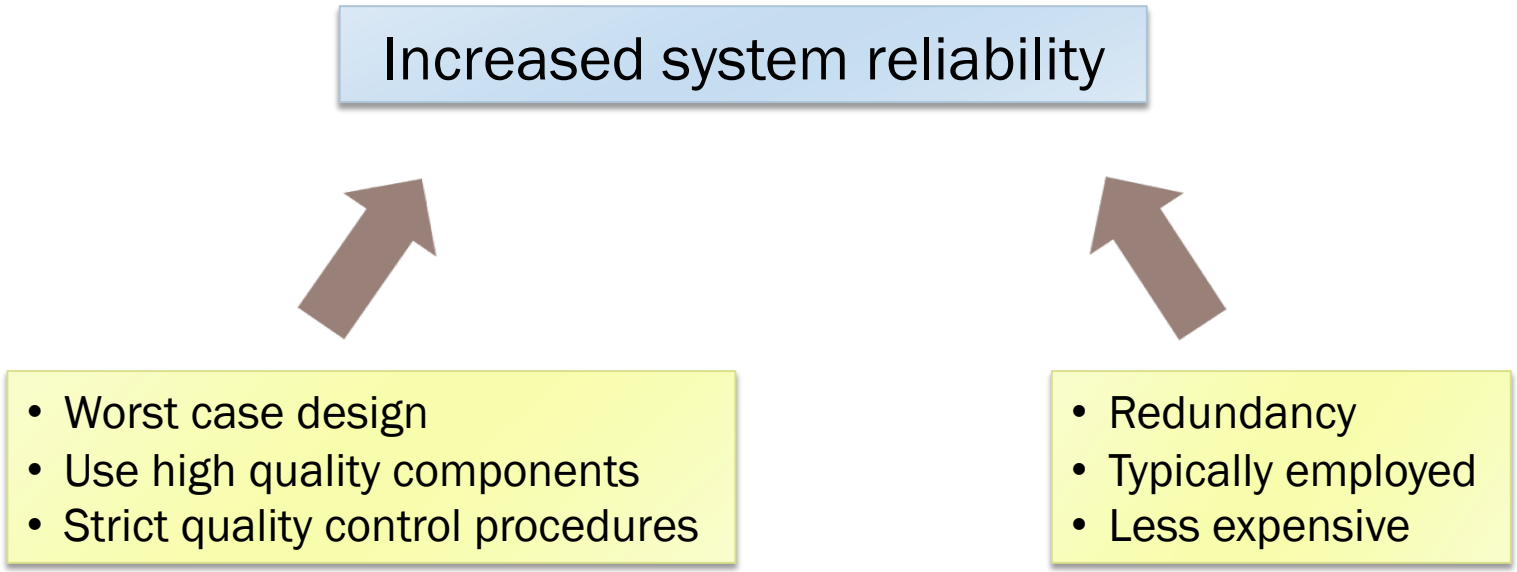
Improving reliability via redundancy



Definitions

- **Reliability**: The probability that the given system will perform its required function under specified conditions for a specified period of time.

- **MTBF (Mean Time Between Failures)**: Average time a system will run between failures. The MTBF is usually expressed in hours. This metric is more useful to the user than the reliability measure.


Increased system reliability
• Worst case design
• Use high quality components
• Strict quality control procedures
• Redundancy
• Typically employed
• Less expensive

Reliability expressions

- Exponential failure law
  - Mostly applicable to hardware components
  - The reliability of a system is modeled as

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

  - where λ is the failure rate expressed as the number of failure per time unit (the time unit could be hours, days, …).
  - When the product λt is small, the expression can be approximated as

$$R(t) = 1 - \lambda t$$

Mean time between failures

- MTBF is the average time a system will run between failures.

$$MTBF = \int_0^\infty R(t)dt = \int_0^\infty e^{-\lambda t}dt = \frac{1}{\lambda}$$

- The mean time between failures is the reciprocal of the failure rate.
- If λ is the number of failures per hour, the MTBF is expressed in hours.
- If, after repair, a system behaves like it was new, there is no difference between MTTF and MTBF. Else there may be some difference.

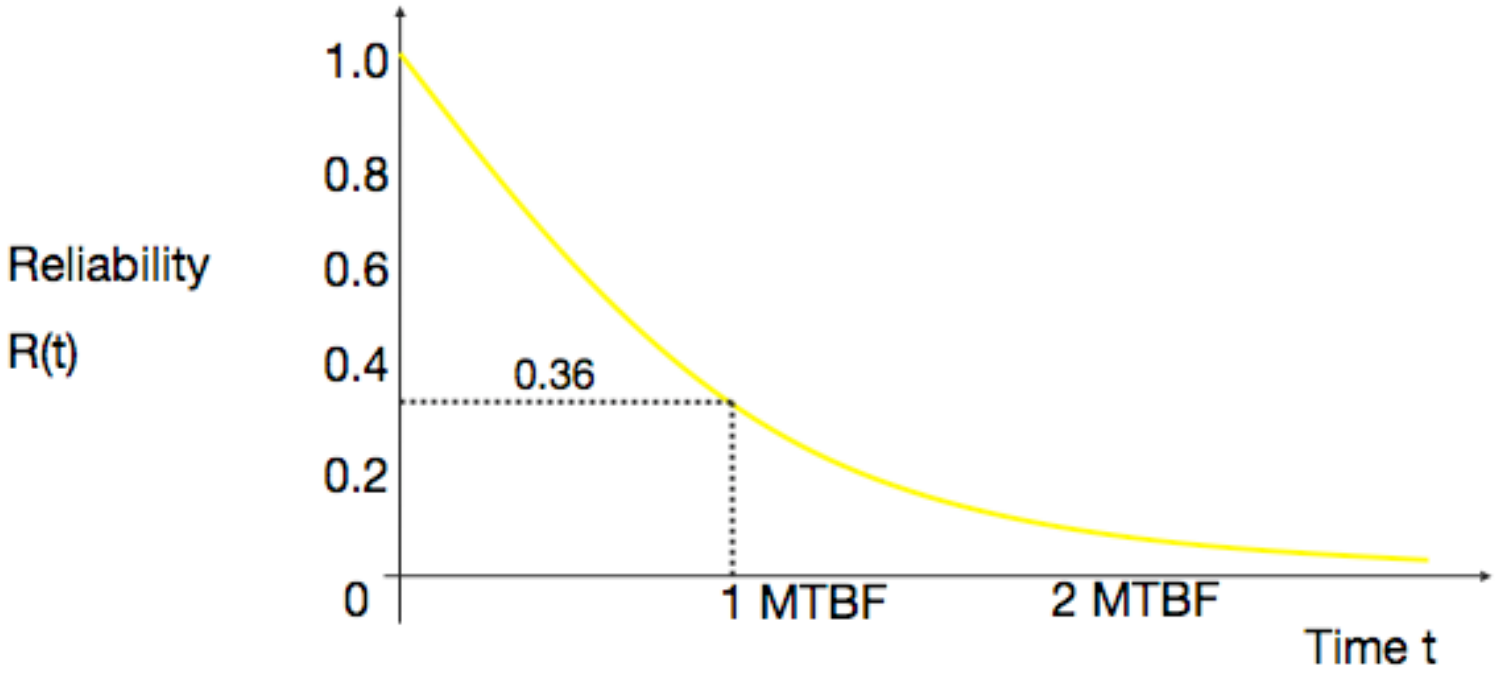
Example

- A system has 4000 components, each with a failure rate of 0.02% per 1000 hours. Calculate λ and the MTBF.

$$\begin{aligned} \lambda &= \frac{0.02}{100} \times \frac{1}{1000} \times 4000 = 8 \times 10^{-4} \text{ failures per hour} \\ MTBF &= \frac{1}{8 \times 10^{-4}} = 1250 \text{ hours} \end{aligned}$$

Relation between MTBF and reliability

- When λt is small: **R(t) = 1-λt = 1-(t/MTBF)**
- **MTBF = t/(1-R(t))**



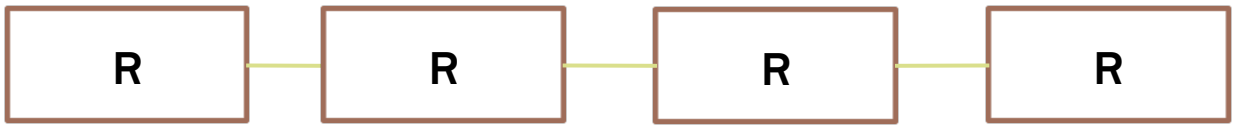
Another example

- A first generation computing system contains 10,000 components, each with λ=0.5% per 1000 hours. What is the period of 99% reliability? (All components have to function correctly.)

$$
\begin{aligned}
N &= 10,000 \text{ (the number of components)} \\
\lambda &= N \times \frac{0.5\%}{1000} = 5 \times 10^{-2} \text{ per hour} \\
MTBF &= \frac{1}{\lambda} = 20 \text{ hours} \\
&\text{Using the approximation:} \qquad t_{0.99} \\
MTBF &= \frac{t}{1 - R(t))} = \frac{t}{1 - 0.99} = 100t \\
t_{0.99} &= 0.2 \text{ hours } = 12 \text{ minutes}
\end{aligned}
$$

Reliability of different system configurations

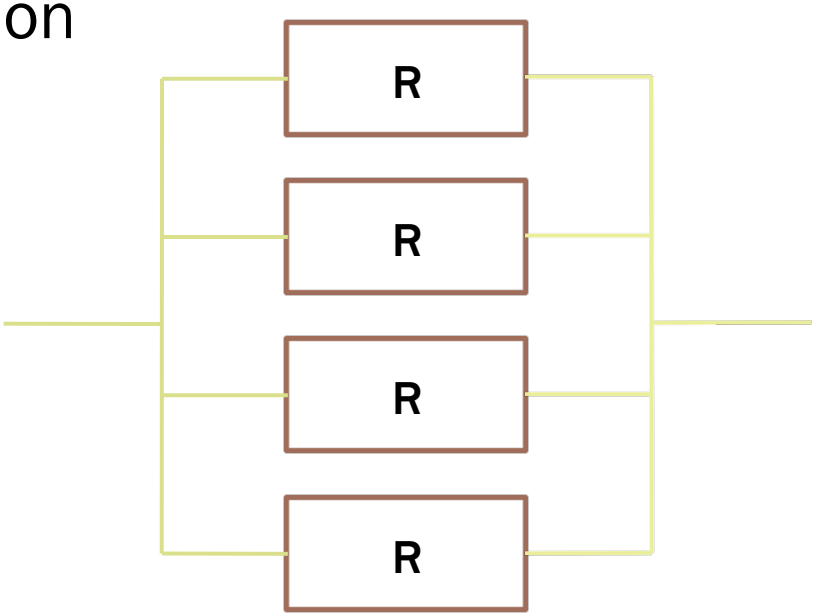
- Series configuration



Overall reliability, $R_o = R \times R \times \ldots \times R = R^n$

Reliability of different system configurations

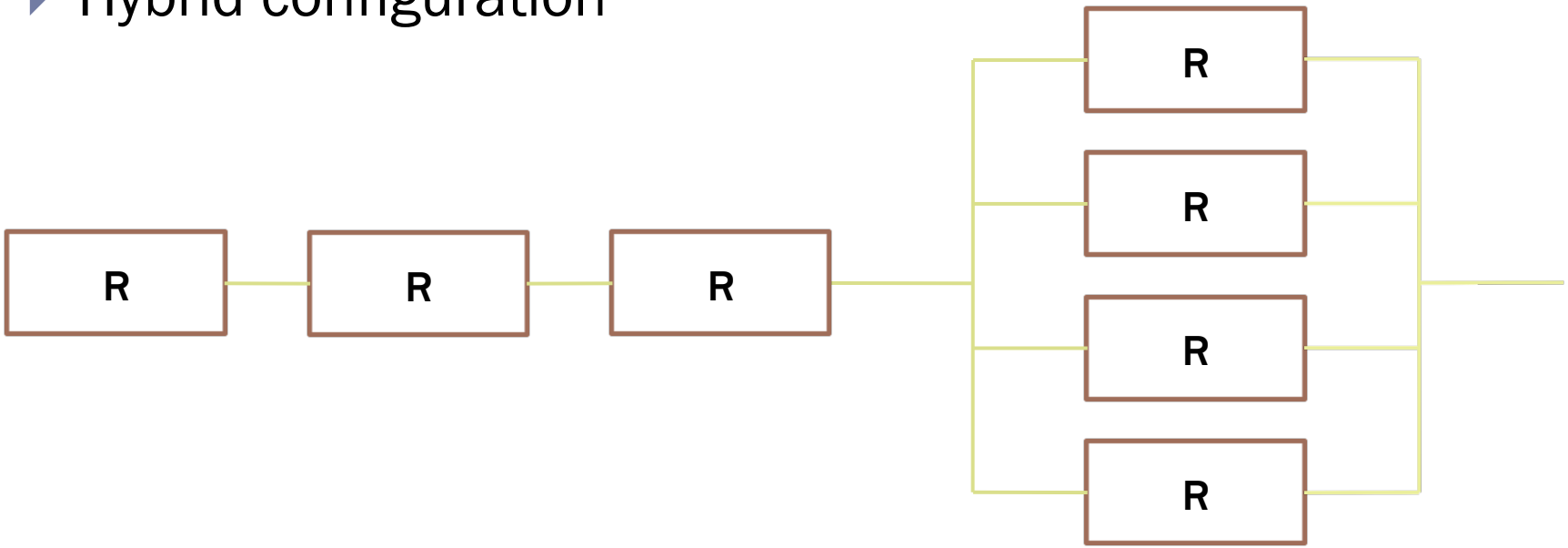
- Parallel configuration



Overall reliability, R_o = 1-(probability of all failures) = $1-(1-R)^n$

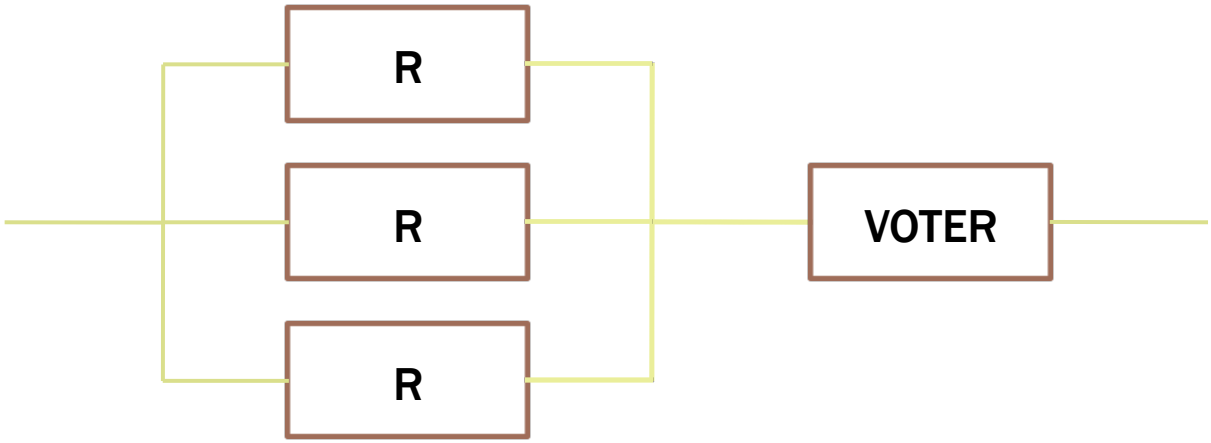
Reliability of different system configurations

- Hybrid configuration



Reliability of different system configurations

- Triple modular redundancy



Assuming an always reliable voter, and assuming at least two modules have to function correctly at any time.

$$R_o = \left[\binom{3}{2} \times R^2 \times (1-R)\right] + R^3$$

Maintainability

- Maintainability of a system is the probability of isolating and repairing a fault in the system within a given time duration.

- **M(t) = 1-e$^{-\mu t}$** where **μ** is the repair rate and **t** is the permissible time for the maintenance action.
- **M(t)** is the probability that the system has been repaired at time t.

- **μ = 1/MTTR**
  - MTTR is the mean time to repair

Availability

- Availability of a system is the probability that the system will be functioning according to expectations at any time during its scheduled working period.

- **Availability = (Uptime)/(Operation Time) = (Uptime)/(Uptime+Downtime)**

- Downtime = (number of failures) x MTTR
- Downtime = (uptime) x λ x MTTR

- Availability = (Uptime)/(uptime + ((uptime) x λ x MTTR)) = 1/(1+ (λ x MTTR))
- **Availability = MTBF/(MTBF+MTTR)**

An exercise

- You have five modules of the same type to design a fault-tolerant system. Each module has a failure rate λ_m. You have several possibilities for building redundancy. You can use a 5-input majority voter (with failure rate λ_5), a 3-input majority voter (λ_3), a 2-to-1 selection circuit (λ_{2s}), and a 2-input comparator (λ_{2c}). Assume that
  - $\lambda_5 = 2\lambda_3$
  - $\lambda_3 = 2\lambda_{2s}$
  - $\lambda_{2s} = \lambda_{2c}$

- What would be the best configuration to connect the modules?
  - You may make reasonable assumptions to ease calculations.

Software systems

- How do we improve reliability?
- Apart from N-versions programming and recovery blocks…

- Better operating systems support
- Stricter programming language primitives
- Rigorous engineering practice
- Formal verification of correctness

Redundancy… but at what cost?

- Discussion: What is an important factor affecting most engineering decisions that has been excluded in reliability estimation? How would you account for the impact of this factor?

- **The price of redundancy.** If we had a finite budget, is it better to improve the reliability of a component or rely on redundant (and less reliable) versions of that component?

Summary

- Building a reliable system involves integrating several redundant components.
- In this lecture we discussed:
  - Reliability and failure rates;
  - MTTF and MTBF;
  - Maintainability;
  - Availability.
- We also analyzed the reliability of different configurations.
  - Parallel, Serial, Hybrid, TMR.