About This Presentation

This presentation is intended to support the use of the textbook *Dependable Computing: A Multilevel Approach* (traditional print or on-line open publication, TBD). It is updated regularly by the author as part of his teaching of the graduate course ECE 257A, Fault-Tolerant Computing, at Univ. of California, Santa Barbara. Instructors can use these slides freely in classroom teaching or for other educational purposes. Unauthorized uses, including distribution for profit, are strictly prohibited. © Behrooz Parhami

<table>
<thead>
<tr>
<th>Edition</th>
<th>Released</th>
<th>Revised</th>
<th>Revised</th>
<th>Revised</th>
<th>Revised</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Oct. 2020</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5 Defect Avoidance
“They found a defect in the new chip. Looks like someone was asleep at the itty-bitty, teeny-weeny switch.”

“Thinking outside of the box didn’t work. Thinking inside of the box didn’t work. Maybe it’s a defective box!”

“The whole keyboard isn’t damaged, just the colon. I’m referring you to a proctologist.”
### STRUCTURE AT A GLANCE

<table>
<thead>
<tr>
<th>Part I — Introduction: Dependable Systems (The Ideal-System View)</th>
<th>Goals</th>
<th>Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Background and Motivation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Dependability Attributes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Combinational Modeling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. State-Space Modeling</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part II — Defects: Physical Imperfections (The Device-Level View)</th>
<th>Methods</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>5. Defect Avoidance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Defect Circumvention</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Shielding and Hardening</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Yield Enhancement</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part III — Faults: Logical Deviations (The Circuit-Level View)</th>
<th>Methods</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>9. Fault Testing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. Fault Masking</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11. Design for Testability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12. Replication and Voting</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part IV — Errors: Informational Distortions (The State-Level View)</th>
<th>Methods</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>13. Error Detection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14. Error Correction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15. Self-Checking Modules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16. Redundant Disk Arrays</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part V — Malfunctions: Architectural Anomalies (The Structure-Level View)</th>
<th>Methods</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>17. Malfunction Diagnosis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18. Malfunction Tolerance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19. Standby Redundancy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20. Resilient Algorithms</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part VI — Degradations: Behavioral Lapses (The Service-Level View)</th>
<th>Methods</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>21. Degradation Allowance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22. Degradation Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23. Robust Task Scheduling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24. Software Redundancy</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part VII — Failures: Computational Breaches (The Result-Level View)</th>
<th>Methods</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>25. Failure Confinement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26. Failure Recovery</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27. Agreement and Adjudication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28. Fail-Safe System Design</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Appendix: Past, Present, and Future

Oct. 2020

Part II – Defects: Physical Imperfections

BParhani
5.1 Types and Causes of Defects

Resistive open due to unfilled via
[R. Madge et al., IEEE D&T, 2003]

Particle embedded between layers
Process and Operational Variations

Even if there isn’t a complete short or open, resistance and capacitance variations can lead to trouble.

Chip temperature map
Analogy: Ideal vs. Real Clock Signals

Ideal clock signal has sharp edges and an exact constant period

Real clock signal is quite different
Disk Memory Defects

The tiniest particle or scratch can wipe out many thousands of bits
Protective Error Coding in Disk Memories

Disks typically use CRC or similarly strong error-correcting codes. It is virtually impossible for data to become contaminated.
When a sector repeatedly violates the code, it is assumed to be bad.
Bad sectors reduce the disk’s capacity.
Disk crashes and other catastrophic failures are a different story.

Image credit: Hitachi

Black: Sector data
Red: Sector code
Blue: Block code
Green: Super-block code

Section of a track
Learning from Failed Disk Drives

Analyses of failed disk drives have led to the following monitoring suggestions to predict when a disk drive is about to go, thus allowing a preemptive replacement before a hard failure

Head flying height: Downward trend often precedes a head crash

Number of remapped sectors: A bad sector is remapped to a different physical location on disk to avoid repeat errors, so having too many remapped sectors signal persistent problems

Frequency of error correction via the built-in code: Disks routinely use CRC and other coding schemes to protect against data loss, but as errors accumulate, they may go beyond the code’s tolerance limit

The following are signs of mechanical or electrical problems:
- Changes in spin-up time
- Rising temperatures in the unit
- Reduction in data throughput
5.2 Yield and Its Associated Costs

- Silicon crystal ingot (30-60 cm)
- Slicer (15-30 cm)
- Blank wafer with defects
- Processing: 20-30 steps
- Patterned wafer

- Dicer
- Die ~1 cm
- Die tester
- Good die ~1 cm
- Mounting
- Microchip or other part
- Part tester
- Usable part to ship

(100s of simple or scores of complex processors)
The dramatic decrease in yield with larger dies

Die yield = \( \frac{\text{def (Number of good dies)}}{\text{(Total number of dies)}} \)

Die yield = Wafer yield \( \times [1 + (\text{Defect density } \times \text{Die area}) / a]^{-a} \)

Die cost = (Cost of wafer) / (Total number of dies \( \times \) Die yield)  
\[ = (\text{Cost of wafer}) \times (\text{Die area / Wafer area}) / (\text{Die yield}) \]

The parameter \( a \) ranges from 3 to 4 for modern CMOS processes
Effects of Yield on Testing and Part Reliability

Assume a die yield of 50%

Out of 2,000,000 dies manufactured, \(\approx 1,000,000\) are defective

To achieve the goal of 100 defects per million (DPM) in parts shipped, we must catch 999,900 of the 1,000,000 defective parts

Therefore, we need a test coverage of 99.99%

Testing is imperfect: missed defects/faults (coverage), false positives

Going from a coverage of 99.9% to 99.99% involves a significant investment in test development and application times

False positives are not a source of difficulty in this context
Discarding another 1-2% due to false positives in testing does not change the scale of the loss
5.3 Defect Modeling

Defect are of two main types:

Global or gross-area defects are due to:
- Scratches (e.g., from wafer mishandling)
- Mask misalignment
- over- and under-etching

Local or spot defects are due to:
- Imperfect process (e.g., extra or missing material)
- Effects of airborne particles

Not every spot defect leads to structural or parametric damage
Actual damage depends on location and size (relative to feature size)
Excess-Material and Pinhole Defects

Extra-material defects are modeled as circular areas

Pinhole defects are tiny breaches in the dielectric between conducting layers

From: http://www.see.ed.ac.uk/research/IMNS/papers/IEE_SMT95_Yield/IEEAbstract.html
Defect Size Distribution

Sample random defect size distribution, assuming 0.3 defects per cm²

\[ f(x) = \begin{cases} 
    kx^{-p} & \text{for } x_{\text{min}} < x < x_{\text{max}} \\
    0 & \text{otherwise} 
\end{cases} \]

- \( x \) = Defect diameter
- \( f(x) \) = Defect density
- \( k \) = Normalizing constant
- \( p \) is typically in [2.0, 3.5]

From: http://www.design-reuse.com/articles/10164/model-based-approach-allows-design-for-yield.html
5.4 The Bathtub Curve

Many components fail early on because of residual or latent defects. Components may also wear out due to aging (less so for electronics). In between the two high-mortality regions lies the useful life period.
Survival Probability of Electronic Components

Infant mortality

No significant wear-out

Bathtub curve

5.5 Burn-in and Stress Testing

Burn-in and stress tests are done in accelerated form.

Difficult to perform on complex and delicate ICs without damaging good parts.

Expensive “ovens” are required.

Burn-in Oven Example

From: http://www.goldenaltos.com/environmental_options.html
5.6 Active Defect Prevention

Other than initial or manufacturing imperfections, defects can develop over the course of a device’s lifetime.

Defects induced by harsh operating environment
- Temperature control
- Load redistribution
- Clock scaling

Radiation-induced defects

Defects due to shock and vibration

Defects due to mishandling (e.g., scratch or smudge on disk)

... discussed in Chap. 7 dealing with shielding and hardening
6 Defect Circumvention
“This just in: the inhabitants of planet Earth are being recalled for the correction of a major defect.”
### STRUCTURE AT A GLANCE

| Part I — Introduction: Dependable Systems (The Ideal-System View) | Goals | 1. Background and Motivation  
2. Dependability Attributes  
3. Combinational Modeling  
4. State-Space Modeling |
| Part II — Defects: Physical Imperfections (The Device-Level View) | Methods | 5. Defect Avoidance  
6. Defect Circumvention  
7. Shielding and Hardening  
8. Yield Enhancement |
| Methods | Examples |
| Part III — Faults: Logical Deviations (The Circuit-Level View) | 9. Fault Testing  
10. Fault Masking  
11. Design for Testability  
12. Replication and Voting |
14. Error Correction  
15. Self-Checking Modules  
16. Redundant Disk Arrays |
| Methods | Examples |
| Part V — Malfunctions: Architectural Anomalies (The Structure-Level View) | 17. Malfunction Diagnosis  
18. Malfunction Tolerance  
19. Standby Redundancy  
20. Resilient Algorithms |
| Methods | Examples |
22. Degradation Management  
23. Robust Task Scheduling  
24. Software Redundancy |
| Methods | Examples |
| Part VII — Failures: Computational Breaches (The Result-Level View) | 25. Failure Confinement  
26. Failure Recovery  
27. Agreement and Adjudication  
28. Fail-Safe System Design |
| Methods | Examples |

Appendix: Past, Present, and Future

Oct. 2020  
Part II – Defects: Physical Imperfections  
Bparhami  
Slide 25
Defect Avoidance vs. Circumvention

Defect Avoidance
Defect awareness in design, particularly floorplanning and routing
Extensive quality control during the manufacturing process
Comprehensive screening, including burn-in and stress tests

Defect Circumvention (Removal)
Built-in dynamic redundancy on the die or wafer
Identification of defective parts (visual inspection, testing, association)
Bypassing or reconfiguration via embedded switches

Defect Circumvention (Masking)
Built-in static redundancy on the die or wafer
Identification of defective parts (external test or self-test)
Adjustment or tuning of redundant structures
6.1 Detection of Defects

Visual or optical inspection:
Focus on more problematic areas, such as edge of wafer

Photo from: http://www.semiconductor.net/article/327100-Defect_Detection_Drives_to_Greater_Depths.php
6.2 Redundancy and Reconfiguration

Works best when the system on die has regular, repetitive structure:
- Memory
- FPGA
- Multicore chip
- CMP (chip multiprocessor)

Irregular (random) logic implies greater redundancy due to replication:
- Replicated structures must not be close to each other
- They should not be very far either (wiring/switching overhead)
Avoiding Bad Sectors on a Disk

**P-List:**
Permanent or primary defect table

- Does not affect drive speed

**G-List:**
Growth or post-use defect table

- Affects drive performance

Image source: http://www.myharddrivedied.com/img4A.jpg
6.3 Defective Memory Arrays

Defect circumvention (removal)
Provide several extra (spare) rows and/or columns
Route external connections to defect-free rows and columns

Defect circumvention (masking)
Error-correcting code

With \( m \) rows and \( s \) spares, can model as \( m \text{-out-of-}(m + s) \)

Somewhat more complex with both spare rows and columns (still combinational, though)

Modeling with coded scheme to be discussed at the info level
6.4 Defects in Logic and FPGAs

Moore and Shannon’s pioneering work:
Building arbitrarily reliable relay circuits out of “crummy” relays

Prob. that a relay device closes when it is supposed to be open = $p$

Prob. that a relay circuit closes when it is supposed to be open = $h(p)$

If we can achieve $h(p) < p$, then repeated application of the composition scheme will lead to arbitrarily small $h(h(h(\ldots h(p))))$

$$h(p) = 4p^2 - 4p^3 + p^4$$

$h(p) < p$ for $p < 0.382$

$h(p) > p$ for $p > 0.382$
Defect Circumvention in FPGAs

Defect circumvention (removal)
Provide several extra (spare) CLBs, I/O blocks, and connections
Route external connections to available blocks

Defect circumvention (masking)
Not applicable
Routing Resources in FPGAs

Simple 3 × 3 switch box
Limited configurability

More elaborate switch boxes
Highly flexible connections

Defect circumvention is quite natural because it relies on the same mechanisms that are used for layout constraints (e.g., use only blocks in the upper left quadrant) or for blocks and interconnects that are no longer available due to prior assignment.
Defects in Multicore Chips or CMPs

Defect circumvention (removal)
Similar to FPGAs, except that processors are the replacement entities

Interprocessor interconnection network is the main challenge

Will discuss the switching and reconfiguration aspects in more detail when we get to the malfunction level in our multilevel model
6.5 Defective 1D and 2D Arrays

Multiple resources on a chip not a challenge if they are independent in logic and I/O connections.

Example: To build an MPP out of 64-processor chips, one might place 72 processors on each chip to allow for up to 8 defective processors.

Given the probability of a processor (including its external connections) being defective, the chip yield can be modeled as a 64-out-of-72 system.

In practice, we interconnect such processors on the chip to allow higher-bandwidth interprocessor communication and I/O.
Defect Circumvention in Regular Arrays

Extensive research done on how to salvage a working array from one that has been damaged by defects

Proposed methods differ in

- Types and placement of switches (e.g., 4-port, single/double-track)
- Types and placement of spares
- Algorithms for determining working configurations
- Ways of effecting reconfiguration
- Methods of assessing resilience

The next few slides show some methods based on 4-port, 2-state switches
Defect Circumvention in Linear Arrays

A linear array with a spare processor and reconfiguration switches

A linear array with a spare processor and embedded switching
Assumption: A defective unit can be bypassed in its row/column by means of a separate switching mechanism (not shown)
A Reconfiguration Scheme for 2D Arrays

A 5 × 5 working array salvaged from a 6 × 6 redundant mesh through reconfiguration switching

Seven defective processors in a 5 × 5 array and their associated compensation paths
6.6 Other Circumvention Methods

Nanoelectronics with “crummy” components: Hybrid-technology FPGA, with CMOS logic elements and crossbar nanoswitches that are very compact, but highly unreliable

Allows 8-fold increase in density, while providing reliable operation via defect circumvention

Image source: W. Robinett et al., *Communications of the ACM*, Sep. 2007
Highly Redundant Nanoelectronic Memories

**Memory with block-level redundancy:**
Based on hybrid semiconductor/nanodevice implementation

Error-correcting code applied for defect tolerance, as opposed to operational or “soft” errors

7 Shielding and Hardening
“Miss Simms, activate my deflector shield.”

“No wonder our rifles aren’t effective! The deer are wearing bullet proof vests.”

“The halo and wings I understand. The bullet-proof jacket I don’t.”

“I want you folks to turn all your bows and arrows and swords and shields and stuff over to the Government — you might hurt yourselves with them.”
### STRUCTURE AT A GLANCE

<table>
<thead>
<tr>
<th>Part I — Introduction:</th>
<th>Goals</th>
<th>1. Background and Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependable Systems</td>
<td>Models</td>
<td>2. Dependability Attributes</td>
</tr>
<tr>
<td>(The Ideal-System View)</td>
<td></td>
<td>3. Combinatorial Modeling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. State-Space Modeling</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part II — Defects:</th>
<th>Methods</th>
<th>5. Defect Avoidance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Imperfections</td>
<td>Examples</td>
<td>6. Defect Circumvention</td>
</tr>
<tr>
<td>(The Device-Level View)</td>
<td></td>
<td>7. Shielding and Hardening</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8. Yield Enhancement</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical Deviations</td>
<td>Examples</td>
<td>10. Fault Masking</td>
</tr>
<tr>
<td>(The Circuit-Level View)</td>
<td></td>
<td>11. Design for Testability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12. Replication and Voting</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part IV — Errors:</th>
<th>Methods</th>
<th>13. Error Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>informational Distortions</td>
<td>Examples</td>
<td>14. Error Correction</td>
</tr>
<tr>
<td>(The State-Level View)</td>
<td></td>
<td>15. Self-Checking Modules</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16. Redundant Disk Arrays</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part V — Malfunctions:</th>
<th>Methods</th>
<th>17. Malfunction Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architectural Anomalies</td>
<td>Examples</td>
<td>18. Malfunction Tolerance</td>
</tr>
<tr>
<td>(The Structure-Level View)</td>
<td></td>
<td>19. Standby Redundancy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20. Resilient Algorithms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part VI — Degradations:</th>
<th>Methods</th>
<th>21. Degradation Allowance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavioral Lapses</td>
<td>Examples</td>
<td>22. Degradation Management</td>
</tr>
<tr>
<td>(The Service-Level View)</td>
<td></td>
<td>23. Robust Task Scheduling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24. Software Redundancy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part VII — Failures:</th>
<th>Methods</th>
<th>25. Failure Confinement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computational Breaches</td>
<td>Examples</td>
<td>26. Failure Recovery</td>
</tr>
<tr>
<td>(The Result-Level View)</td>
<td></td>
<td>27. Agreement and Adjudication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28. Fail-Safe System Design</td>
</tr>
</tbody>
</table>

Appendix: Past, Present, and Future

Oct. 2020

Part II – Defects: Physical Imperfections

UCSB

Bparhani

Slide 44
7.1 Interference and Cross-Talk

**Electromagnetic or radio-frequency interference** (EMI, RFI) is a disturbance that affects an electrical circuit due to either electromagnetic conduction or electromagnetic radiation emitted from an external source. The disturbance may interrupt, obstruct, or otherwise degrade or limit the effective performance of the circuit.

Interference can occur through the air or via shared power supply

**Crosstalk** (XT) refers to any phenomenon by which a signal transmitted on one circuit or channel of a transmission system creates an undesired effect in another circuit or channel. Crosstalk is usually caused by undesired capacitive, inductive, or conductive coupling from one circuit, part of a circuit, or channel, to another.

On-Chip Cross-Talk

Shrinking feature sizes have made on-chip crosstalk a major problem

The interwire capacitance $C_I$ can easily exceed the load + parasitic capacitance $C_L$ for long buses, affecting power dissipation, speed, and signal integrity

Wires with taller cross sections (required for speed with scaling) make crosstalk problems worse

From: [Duan09]
Cross-Talk Mitigation Methods

Spacing and staggering of wires that tend to produce heavier cross-talk

On-chip twisted pair [Yu09]

Aggressor

Victim

Bus encoding: Details to be supplied

For a discussion of crosstalk noise modeling and reduction, see:
http://users.ece.utexas.edu/~dpan/2009Fall_EE382V/notes/lecture10_crosstalk.ppt/
7.2 Shielding via Enclosures

Materials and techniques exist for shielding hardware from a variety of external influences

- Shielded cable
- NASA’s EAFTC computers
- RF-shielded packaging
- Static-shield package
7.3 The Radiation Problem

**Electromagnetic radiation:**

Ultraviolet (UV) radiation is nonpenetrating and thus easily stopped.

X-ray and gamma radiations can be absorbed by atoms with heavy nuclei, such as lead.

Nuclear reactors use a thick layer of suitably reinforced concrete.

**Particle radiation:**

Alpha particles (helium nuclei), least penetrating, paper stops them.

Beta particles (electrons), more penetrating, stopped by aluminum sheet.

Neutron radiation, difficult to stop, requires bulky shielding.

Cosmic radiation, not a problem on earth, important for space electronics.

**Secondary radiation:** Interaction of primary radiation and shield material.

Radiation Effect on CMOS ICs

Impact by high-energy particles, such as protons or heavy ions

Radiation ionizes the oxide, creating electrons and holes; the electrons then flow out, creating a positive charge which leads to current leak across the channel.

It also decreases the threshold voltage, which affects timing and other operational parameters.

One-way mission to Mars:
Exposes the electronics to about 1000 kilorad of radiation, which is near the limit of what is now tolerable by advanced space electronics.

From: http://ajnoyola.com/RHBD_primer.html
Heavy-Ion and Proton Radiations

Each particle produces an ionization track

A few protons cause nuclear reactions

Most protons pass through the device with little effect

Short-range recoil produces ionization

a) Heavy Ions (ionization by each particle)
b) Protons (nuclear reaction needed to produce recoil)

More Details Regarding Radiation Effects

Negative Impacts of Radiation

**Single-event upset (SEU):** A single ion changing the state of a memory or register bit; multiple bits being affected is possible, but rare

**Single-event latchup (SEL) or snapback:** A heavy ion or a high-energy particle shorting the power source to substrate (high currents may result)

**Single-event transient (SET):** The discharge of collected charge from an ionization event creating a spurious signal

**Single-event induced burnout (SEB):** A drain-source voltage exceeding the breakdown threshold of the parasitic structures

**Single-event gate rupture (SEGR):** A heavy ion hitting the gate region, combined with applied high voltage, as in EEPROMs, creates breakdown
7.4 Radiation Hardening

**Use of insulating or wide-band-gap substrate:** Instead of common, and fairly inexpensive, semiconductor substrate

**Shielding the package or the chip itself:** Radioactive-resistant packaging or use of more resilient material in the chip’s composition

**Replace DRAM with the more rugged SRAM:** Capacitor-based DRAM is particularly susceptible to upset events

**Fault- and error-level methods:** Circuit duplication/triplication with comparison/voting, or coding, lead to area and power penalties

**System and application-level methods:** On-line or periodic testing, liveness checks, frequent resets
Packaging Solutions to the Radiation Problem

Shielding much less effective against proton radiation

Packaging can be a partial solution to slow down the particles

7.5 Vibrations, Shocks, and Spills

Hundreds of patents on the topic, but very little published material

Shock-resistant or ruggedized computers are useful for military personnel, law enforcement, emergency response teams, and children.

Ruggedized can mean:
Shock- or drop-resistant
Heat-resistant
Water-resistant (e.g., for water rescue)

Most common accidents for laptops:
1. Drops (detection, followed by securing the disk drive; hardened case)
2. Spills (spill-proof keyboard)
Rugged Laptop for NASA’s Space Shuttles

The GRiD (Graphical Retrieval Information Display) Compass

First laptop in orbit
First with a clamshell design
21.6-cm bright plasma display
In use through the early 1990s
Weight: 4.5 kg
Cost: $8150, at the time

Reportedly survived the 1986 Space Shuttle Challenger crash

Image credit: IEEE Spectrum
7.6 Current Practice and Trends

This section to be completed

Nemoto, N., et al.  
“Evaluation of Single-Event Upset Tolerance on Recent Commercial Memory ICs”  

**Abstract:** Single-event upset (SEU) tolerance for commercial 1Mbit SRAMs, 4Mbit SRAMs, 16Mbit DRAMs and 64Mbit DRAMs was evaluated by irradiation tests using high-energy heavy ions with an LET range between 4.0 and 60.6 MeV/(mg/cm²). The threshold LET and the saturated cross-section were determined for each device from the LET dependence of the SEU cross-section. We show these test results and describe the SEU tolerance of highly integrated memory devices in connection with their structures and fabrication processes. The SEU rates in actual space were also calculated for these devices.
8 Yield Enhancement
Economists blame the "inverted yield curve"... today the rich got poorer and the poor got richer!

You're too strict

"I'm afraid we need someone with a little more backbone for this position."

**STRUCTURE AT A GLANCE**

<table>
<thead>
<tr>
<th>Part I — Introduction: Dependable Systems (The Ideal-System View)</th>
<th>Goals Models</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Part II — Defects: Physical Imperfections (The Device-Level View)</th>
<th>Methods Examples</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Part III — Faults: Logical Deviations (The Circuit-Level View)</th>
<th>Methods Examples</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Part IV — Errors: Informational Distortions (The State-Level View)</th>
<th>Methods Examples</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Part V — Malfunctions: Architectural Anomalies (The Structure-Level View)</th>
<th>Methods Examples</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Part VI — Degradations: Behavioral Lapses (The Service-Level View)</th>
<th>Methods Examples</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Part VII — Failures: Computational Breaches (The Result-Level View)</th>
<th>Methods Examples</th>
</tr>
</thead>
</table>

Appendix: Past, Present, and Future

---

Oct. 2020  Part II – Defects: Physical Imperfections  
Slide 61
8.1 Yield Models

Highly simplified example, with only extra-material defects

Consider a square chip area of side 1 cm with parallel, equally spaced nodes of 1 \( \mu \)m width and separation. Let there be an average of 10 random defects per cm\(^2\). Assume extra-material defects are of two kinds: 80\% are small defects of diameter 0.5 \( \mu \)m, and 20\% are larger defects of diameter 1.5 \( \mu \)m. What is the expected yield of this simple chip?

Expected number of defects = 10 (8 small, 2 large)
Small defects cannot lead to shorts, so we can ignore them.
A large defect leads to a short if its center is within a 0.5-\( \mu \)m band halfway between two nodes.
So, we need to find the probability of at least 1 large defect appearing within an area of 0.25 cm\(^2\), given an average of 2 such defects in 1 cm\(^2\).
8.2 Redundancy for Yield Enhancement

Ideally, with $n$ cells and $s$ spares on a chip or die, the yield can be modeled as an $n$-out-of-$(n + s)$ structure.

This is usually not the correct model because:

A defective cell may not be replaceable by an arbitrary spare; there are often severe restrictions on what can replace what.

Replacement may have to be done in blocks (such as rows or columns) rather than single cells.

For examples, see Sections 8.4 and 8.5.
8.3 Floorplanning and Routing

Designers can mitigate the effects of extra- and missing-material defects by adjusting the floorplanning and routing

Wider wires are less sensitive to missing-material defects

Narrower wires are less likely to be shorted to others by extra material

Therefore, an optimal point may exist with regard to yield optimization

Different chip layout/routing designs differ in their sensitivity to various defect classes

Because of defect clustering, one good idea is to place modules with similar sensitivities to defects apart from each other
Sensitivity of Layouts to Defects

VLSI layout must be done with defect patterns and their impacts in mind.

A balance must be struck with regard to sensitivity to different defect types.

Actual photo of a missing-material defect

http://www.midasvision.com/v3.htm
Effects of Defect Sizes and Distribution

Derivation of critical areas for various defect sizes, combined with defect size distribution data allows accurate modeling of defects

From: http://www.see.ed.ac.uk/research/IMNS/papers/IEE_SMT95_Yield/IEEAbstract.html
8.4 Improving Memory Yield

Example: $6 \times 6$ memory array, with 2 spare rows and 2 spare columns

Can we circumvent the defect pattern shown?

With $r$ spare rows and $c$ spare columns, $r + c$ defects can always be circumvented, but here we have 4 spare rows/columns and 7 defects

The problem of assigning spares to defectives rows and columns is NP-complete

From: [Kore07], p. 265
A Formulation of the Assignment Problem

Example: $6 \times 6$ memory array, with 2 spare rows and 2 spare columns

Defect pattern shown as a bipartite graph

Select a set of vertices that together “touch” all edges

A variety of heuristics are available for this bipartite graph *edge covering problem*

They usually start by doing a feasibility check and making some mandatory assignments (e.g., the 3 defects in column 2 cannot all be covered by row spares)
8.5 Regular Processor Arrays

**Linear array with spares**

Given $s$ spares, any $s$ defects can be reconfigured around.

Model as $n$-out-of-$(n + s)$ system.

Switches can be dealt with like voting elements in TMR or they can be distributed and made part of somewhat more complex modules.
Linear Array with Redundant Switching

Inaccessible processor

Inoperative switch
Limits of Reconfigurability in 2D Arrays

Extension: May go beyond the 3-defect limit by providing spare rows on top and bottom and spare columns on either side.

A set of three defective nodes, one of which cannot be accommodated by the compensation-path method.

No compensation path exists for this defect.

Seven defective processors in a 5 × 5 array and their associated compensation paths.
Combinational Modeling for 2D Arrays

No compensation path exists for this defect

**Pessimistic/Easy:** Any 3 bad cells lead to failure
Model $m \times m$ array as $(m^2 - 2)$-out-of-$m^2$ system

**Realistic/Hard:** Enumerate all combinations of bad cells that cannot be reconfigured around and assess the probability of at least one of these combinations occurring
Shift-Switching at the 2D Array’s Edges

**Two-way shift switch:**
Connect outside link for row $i$ to row $i$ or $i + 1$

**Three-way shift switch:**
Connect outside link for row $i$ to row $i - 1$, $i$, or $i + 1$ (larger defect patterns become circumventable)
Multiple Redundancy Schemes

Multiple forms of redundancy can be effective for defect circumvention, if each method covers the others’ weaknesses

**Example:** Memory yield enhancement

- ECC quite good in confronting isolated random defects
- Spare rows/columns/blocks good for correlated or large-area defects
Combined Sparing/ECC for Memory Arrays

Example of IBM’s experimental 16 Mb memory chip
Combines the use of spare rows/columns in memory arrays with ECC

Four quadrants, each with 16 spare rows & 24 spare columns

ECC corrects any single error via 9 check bits (137 data bits)

Bits assigned to the same word are separated by 8 bit positions
8.6 Impact of Process Variations

Small feature sizes and high densities of modern VLSI circuits make slight manufacturing variations quite significant in their correct functioning and performance.

Additionally, there may be massive numbers of defects in nanoelectronic circuits and a single physical defect may affect more components than before.
Looking Back and Forward

Next step: The fault-level view