Reliability

Prof. Dr. Akash Kumar

Chair for Processor Design

(Ack: my past and current students/PostDocs)
(Some slides adapted from Koren, Krishna, Anand)
Why do we care about Reliability?

- “The most expensive hyphen in history”
  - Mariner 1

- “The processor bug that shook the world”
  - Intel Pentium floating point division

- “Mixing up units”
  - Mars Climate Orbiter

[1] https://en.wikipedia.org/wiki/Mariner_1
What effects Reliability?

- The **Human** factor
  - Bugs
  - Malicious attacks

- **More Natural** factors
  - The Process
    - Manufacturing defects
  - Aging
    - Circuit Wearout
  - Operating Environment
    - Electromagnetic Interference
    - Internal Electronic Noise

These phenomenon always existed ... why bother now?
Need For Fault-Tolerance

- **Critical applications** require extreme fault tolerance (e.g., aircrafts, nuclear reactors, medical equipment, and financial applications)
  - A malfunction of a computer in such applications can lead to catastrophe
  - Their probability of failure must be extremely low, possibly one in a billion per hour of operation

- **System operating in a harsh environment** with high failure possibilities
  - electromagnetic disturbances
  - particle hits and alike

- **Complex systems** consisting of millions of devices
Failures during Lifetime

- Three phases of system lifetime
  - Infant mortality (imperfect test, weak components)
  - Normal lifetime (transient/intermittent faults)
  - Wear-out period (circuit aging)
The Impact of Technology Scaling

- More leakage
- More process variability
- Smaller critical charges
  - Trends show soft-error rates incr. exp., 8% per tech generation
- Weaker transistors and wires

Burn-in test less effective

Higher random failure rate

Faster wear-out
Outline

Reliability … should we really care?

Nomenclature of reliability
- Faults, Errors and Failures
- Fault types
- Fault mechanisms

Living with faults
- Application view of reliability
- Fault mitigation
- Redundancy, Redundancy, Redundancy

Research: Reliable System Design
- Improving System Lifetime
- QoS-aware Reliability
- Cross-layer Reliability
Outline

Reliability … should we really care?

Nomenclature of reliability
- Faults, Errors and Failures
- Fault types
- Fault mechanisms

Living with faults
- Application view of reliability
- Fault mitigation
- Redundancy, Redundancy, Redundancy

Research: Reliable System Design
- Improving System Lifetime
- QoS-aware Reliability
- Cross-layer Reliability
Faults, Errors and Failures

- **FAILURE** is an event that occurs when the delivered service deviates from the correct service.

- **ERROR** is the part of the total state of the system that may lead to its subsequent failure.

- **FAULT** is an erroneous state of software or hardware resulting from failures of its components.
Types of Faults

FAULTS

DEVELOPMENTAL
- Occur during development
- Design and manufacturing defects

PHYSICAL
- Faults that affect hardware
- Deterioration -- wear, fatigue and corrosion
- External factors

INTERACTION
- All external faults
- System misuse or external disturbances
- Harsh environment, mechanical disturbances
Physical Fault Mechanisms: Soft-Errors

Soft-errors are **transient** hardware malfunctions that are not reproducible and can usually be removed by resetting the system.

- Interaction of charged particles
  - Alpha particles
    - Impurities in chip packages
  - Neutrons
    - Cosmic radiation + Earth's atmosphere
- Changes in logical value
  - Collected charge > $Q_{crit}$
  - Combinational circuits: computation error
  - Storage elements: Incorrect till refresh
Physical Fault Mechanisms: Soft-Errors

- **Masking**
  - **Logical Masking**
    - Sensitivity path from origin to latch
  - **Electrical Masking**
    - Attenuation during gate propagation
  - **Latching-window Masking**
    - Appearance of glitch in latching-window

[Sheng2009]
Physical Faults: Transient

**Caused by…**
- Charged particles
  - Outer space
  - Packaging materials

**Manifest as…**
- Computation errors
- Data Corruption
- System Failure

**Rising rate…**
- Reduced *Q*<sub>critical</sub>
- Reduced Electrical Masking
- Reduced Latching-window Masking

- **Transistor Scaling**
- **Voltage Scaling**

© Akash Kumar
Physical Fault Mechanisms: Aging

Aging is the degradation of semiconductor devices due to continued electrical stress. “Usually, the aging process merely slows the speed at which transistors turn on and off. But it can also break them outright.” [Keane2011]

Electrical stress can lead to one or more among – Bias Temperature Instability, Gate Oxide Breakdown, Hot Carrier Injection and Electro-migration.
Physical Fault Mechanisms: Aging

- **Bias Temperature Instability (BTI)**
  - Migration of charge carriers into gate-oxide
  - Increased $V_{th}$
  - Partly reversible

- **Gate-Oxide Breakdown (GOB)**
  - Defects/Traps in gate-oxide
  - May lead to gate-channel short circuit
  - Can lead to chip failure
Physical Fault Mechanisms: Aging

- **Hot-carrier Injection (HCl)**
  - High energy charge carriers stray into dielectric
  - Increases the $V_{th}$
  - Leads to slower switching

- **Electro-migration (EM)**
  - Current surges lead to atomic displacement
  - Increased resistance of interconnects
  - May lead to shorts/opens
Physical Faults: *Intermittent & Permanent*

**Caused by...**
- Intermittent & Permanent
  - Gate-oxide Breakdown
  - Hot Carrier Injection
  - Electromigration
  - NBTI

**Manifest as...**
- Reduced Lifetime
  - System Failure

**Rising rate...**
- Transistor Scaling
  - Manufacturing defects
  - Increased Variability
  - High Power Density
  - Higher Temperature
  - Faster aging

*Physical Faults: Intermittent & Permanent*
Outline

1. Reliability ... should we really care?
2. Nomenclature of reliability
   - Fault types
   - Fault mechanisms
3. Living with faults
   - Application view of reliability
   - Fault mitigation
   - Redundancy, Redundancy, Redundancy
4. Research: Reliable System Design
   - Improving System Lifetime
   - QoS-aware Reliability
   - Cross-layer Reliability

© Akash Kumar
# Reliability: An application point of view

## Timing
- **Adherence to deadlines**
- **Real-time systems** *(Hard, Firm, Soft)*
- **Metrics**
  - WCET
  - Completion Probability
  - Average makespan

## Functional
- **Computational correctness**
- **Financial systems**
- **Security applications**
- **Metrics**
  - Error probability
  - Mean time between errors

## Lifetime
- **Operational longevity**
- **Safety/Mission critical systems**
- **Consumer products**
- **Metrics**
  - Mean time to failure
  - Mean time to crash *(MTTF, MTTC)*

© Akash Kumar
Fault Mitigation

- Fault prevention
- Fault tolerance
- Fault removal
- Fault forecasting
Levels of Fault Tolerance

- N-VP
  - Code Triplication
  - Virtualization
  - Task migration
  - Redundant multithreading
  - Fault-tolerant
  - Mapping/scheduling

- Core-level redundancy
  - TMR/ DWC
  - Dynamic verification & correction

- Block-level redundancy
  - ECC for memory
  - Circuit hardening
Fault Tolerance $\rightarrow$ Redundancy

- N-VP, Code Triplication
- Virtualization
- Task migration
- Redundant multithreading
- Fault-tolerant
- Mapping/scheduling

- Core-level redundancy
- TMR/ DWC
- Dynamic verification & correction
- Block-level redundancy
- ECC for memory
- Circuit hardening

REDUNDANCY ... in some form

- Spatial
- Temporal
- Informational
Case-study: Matrix-Matrix Multiplication

\[ C = AB \]

\[
\begin{bmatrix}
    c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} \\
    c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} \\
    c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} \\
    c_{4,1} & c_{4,2} & c_{4,3} & c_{4,4}
\end{bmatrix}
= \begin{bmatrix}
    a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\
    a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\
    a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\
    a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4}
\end{bmatrix}
\times \begin{bmatrix}
    b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4} \\
    b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} \\
    b_{3,1} & b_{3,2} & b_{3,3} & b_{3,4} \\
    b_{4,1} & b_{4,2} & b_{4,3} & b_{4,4}
\end{bmatrix}
\]

\[ c_{i,j} = \sum_{k=1}^{N} a_{i,k} \times b_{k,j} \quad \forall \, i, j \in [1, N] \text{ and } i, j \in \mathbb{N} \]

\[ N^2 \text{ operations} \]
Spatial Redundancy: For Soft-error

- Dual Modular Redundancy (DMR)
  - Error Detection

\[ a_{i,k} = b_{k,j} \]

\[ c_{i,j} \]

\[ clk \]

\[ rst \]

\[ error? \]
Spatial Redundancy: For Soft-error

- **Triple Modular Redundancy (TMR)**
  - Error correction ... to a large extent
Quantifying Error Probability: TMR

- $p_m$: Error probability of each module
- $p_v$: Error probability of voting

Then, Reliability ($Rel$)

\[
Rel = 3 \times (1 - p_v) \times (1 - p_m)^3 \\
+ 3 \times (1 - p_v) \times (1 - p_m)^2 \times p_m
\]

Neglecting $p_v$: $p_v \ll p_m$

\[
Rel \approx 3 \left( (1 - p_m)^3 + p_m (1 - p_m)^2 \right)
\]
Estimation of Probability of Soft-Error

- Single Event Errors
  - Soft error-rate (SER): $\lambda$
  - Poisson point process ... assuming independent events

$$p_{error}(t) = 1 - e^{-\lambda t}$$
$$p_{ne}(t_0) = e^{-\lambda t_0}$$

![Graph showing the probability of error over time with different lambda values]
Spatial Redundancy: For Permanent Faults

- **$k$-out-of-$n$:G** system model
  - Generic representation of systems with spatial redundancy
  - **Good system**: At least $k$ out of $n$ units are functionally correct


- Improving Lifetime Reliability: Adding spares
  - **$k$-out-of-$n$:G** system with cold/warm spares

  4-out-of-4:G → 4-out-of-5:G (with 1 cold spare)
Temporal Redundancy

- Retry

Error Detection
Temporal Redundancy

- Checkpointing with rollback

```
\begin{align*}
[ c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} ] \\
[ c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} ] \\
[ c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} ] \\
[ c_{4,1} & c_{4,2} & c_{4,3} & c_{4,4} ]
\end{align*}
\times
\begin{align*}
[ a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} ] \\
[ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} ] \\
[ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} ] \\
[ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} ]
\end{align*}
```

save state/results $\rightarrow$ Create checkpoints

Feasible equidistant checkpoints

Timing Overheads

Useful Computation

Error Detection

Checkpoint Creation

© Akash Kumar
Informational Redundancy

☐ Checksum

“A checksum is a small-sized datum derived from a block of digital data for the purpose of detecting errors that may have been introduced during its transmission or storage.” … or during computation!!

--Wikipedia

☐ ABFT (Algorithm-based Fault Tolerance)

\[
\begin{bmatrix}
c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} \\
c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} & c_{2,5} \\
c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} & c_{3,5} \\
c_{4,1} & c_{4,2} & c_{4,3} & c_{4,4} & c_{4,5} \\
c_{5,1} & c_{5,2} & c_{5,3} & c_{5,4} & c_{5,5}
\end{bmatrix}
= \begin{bmatrix}
a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\
a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\
a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\
a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \\
a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4}
\end{bmatrix}
\times \begin{bmatrix}
b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4} & b_{1,5} \\
b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} & b_{2,5} \\
b_{3,1} & b_{3,2} & b_{3,3} & b_{3,4} & b_{3,5} \\
b_{4,1} & b_{4,2} & b_{4,3} & b_{4,4} & b_{4,5}
\end{bmatrix}
\]

\[C_{fc} = A_{cc} \times B_{rc}\]

\[C_{fc}: (N + 1)^2 \text{ operations, }\]
+ Error detection,
+ Error correction

\[a_{N+1,j} = \sum_{i=1}^{N} a_{i,j}, \text{ for } 1 \leq j \leq N\]
\[b_{j,N+1} = \sum_{j=1}^{N} b_{i,j}, \text{ for } 1 \leq i \leq N\]
Impact of Redundancy on Timing Reliability

Without Fault-mitigation

Information Redundancy

Spatial Redundancy

Temporal Redundancy
Impact of Redundancy on Functional Reliability

- **Spatial Redundancy**
  - Single-events upsets
  - Number of modules

- **Temporal Redundancy**
  - Imperfect error-detection
  - Imperfect recovery

- **Informational Redundancy**
  - Recovery of Single errors
  - Detection of multiple errors
Impact of Redundancy on *Lifetime Reliability*

- **Spatial Redundancy**
  - N-MR: Increased power $\rightarrow$ Increased temperature $\rightarrow$ Accelerated aging
  - k-out-of-n:G with spares: Increased time to system crash

- **Temporal Redundancy**
  - Increased average execution time
  - Increased aging per unit computation

- **Informational Redundancy**
  - Increased computation
  - Increased aging for unit computational load. e.g. Product Matrix
Outline

- Reliability ... should we really care?
- Nomenclature of reliability
  - Faults, Errors and Failures
  - Fault types
  - Fault mechanisms
- Living with faults
  - Application view of reliability
  - Fault mitigation
  - Redundancy, Redundancy, Redundancy
- Research: Reliable System Design
  - Improving System Lifetime
  - QoS-aware Reliability
  - Cross-layer Reliability
Improving Lifetime Reliability: Task-Remapping

- Task-mapping in HMPSoCs

- Aging-aware Task-remapping
  - Re-map more stressful tasks to lesser aged Processing Elements (PEs)

Source: Singh et al. DAC, 2013
Source: Sahoo et al. DATE, 2016
Improving Lifetime Reliability: HW/HW Partitioning

- Hardware-Hardware partitioning
  - Dynamic Partial Reconfiguration (DPR)
    - Task-mapping equivalent for FPGAs

![Diagram](image-url)

© Akash Kumar
Improving Lifetime Reliability: HW/HW Partitioning

- Aging-aware partitioning and scheduling

- Scheduling

- Partitioning
Improving Lifetime Reliability: HW/HW Partitioning

- Research Results

- Aging-aware Partitioning

Maximize system MTTF with deadline constraints using homogeneous PRRs

Maximize system MTTF with deadline constraints using heterogeneous PRRs

Source: Sahoo et al. ASP-DAC, 2018
Application-specific Design for Reliability

- Diversity in application areas of electronic systems

- Niche applications (1940s-1960s)
  - Intel 4004
  - 1971

- Commercial usage (1970s-2000s)

- Everyday usage (2008--)
  - IoTs
  - 2008-09

© Akash Kumar
Application-specific Design for Reliability

- Diversity in application areas of electronic systems

System Design Goals

- Commercial usage (1970s-2000s)
- Everyday usage (2008-)

© Akash Kumar
## Application areas and requirements

- **Variation**

<table>
<thead>
<tr>
<th>Application Area</th>
<th>Functional Reliability</th>
<th>Timing Reliability</th>
<th>Lifetime Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banking</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Multimedia</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Portable multimedia</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Health monitoring</td>
<td>High</td>
<td>Medium ~ High</td>
<td>High</td>
</tr>
<tr>
<td>Satellites / Space Missions</td>
<td>Medium</td>
<td>Medium ~ High</td>
<td>High</td>
</tr>
</tbody>
</table>

Not all applications require the same level of reliability.
Single-layer Reliability

- The usual “phenomenon-based” approach
- Provide a “perfect” hardware to upper layers
Cross-layer Approach

- Inter-layer information exchange
- Leverage application-specific tolerances to degradation
- Exploit implicit masking of multiple layers

Cross-layer Approach

Need to do a cost-benefit analysis!!

Application Design
Performance metrics, Acceptable miss-rate, Error Tolerance, Profiled data, Acceptance test time …

System Software Design
Masking factor, Execution overhead, Error detection and/or correction time, other overheads …

Hardware
Masking factor, Power/Energy overheads, Fault detection/correction overhead …

Compilation

System Software

Platform Design

Architecture Design

Synthesis

Place and Route

Device and Cell Design

Resilience Mechanism

© Akash Kumar
## Varying Reliability Schemes – Multimedia

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Baseline</th>
<th>HW-only</th>
<th>Configurable redundancy</th>
<th>Cross-Layer (at Ground)</th>
<th>Cross-Layer (at Altitude)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Time</td>
<td></td>
<td></td>
<td>Configurable error-detection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deadline</td>
<td>$\lambda_0$</td>
<td>$\lambda_0$</td>
<td>$\lambda_0$</td>
<td>$\lambda_0$</td>
<td>$10\times\lambda_0$</td>
</tr>
<tr>
<td>SEU rate</td>
<td>$\lambda_b$</td>
<td>$\lambda_b$</td>
<td>$\lambda_b$</td>
<td>$\lambda_b$</td>
<td>$10\times\lambda_b$</td>
</tr>
<tr>
<td>Mitigation methods</td>
<td>None</td>
<td>Full TMR</td>
<td>Checkpointing (30% coverage)</td>
<td>Partial TMR + Checkpointing (30% coverage)</td>
<td>Full TMR + Checkpointing (10% coverage)</td>
</tr>
<tr>
<td>HW Masking</td>
<td>$M_f_{HW0}$</td>
<td>$10\times M_f_{HW0}$</td>
<td>$M_f_{HW0}$</td>
<td>$5\times M_f_{HW0}$</td>
<td>$10\times M_f_{HW0}$</td>
</tr>
<tr>
<td>Functional Reliability (errors/sec)</td>
<td>54 bad frames /hour</td>
<td>5.4 bad frames /hour</td>
<td>38 bad frames /hour</td>
<td>7.5 bad frames /hour</td>
<td>48.6 bad frames /hour</td>
</tr>
<tr>
<td>Timing Reliability (probability)</td>
<td>No lags</td>
<td>No lags</td>
<td>54 lagging frames/hour</td>
<td>11 lagging frames/hour</td>
<td>18 lagging frames/hour</td>
</tr>
<tr>
<td>Cost (Area)</td>
<td>$A_0$</td>
<td>$3\times A_0$</td>
<td>$A_0$</td>
<td>$2\times A_0$</td>
<td>$3\times A_0$</td>
</tr>
<tr>
<td>Energy Usage</td>
<td>$E_{SYS0}$</td>
<td>$3\times E_{SYS0}$</td>
<td>$1.1\times E_{SYS0}$</td>
<td>$2.2\times E_{SYS0}$</td>
<td>$3.13\times E_{SYS0}$</td>
</tr>
</tbody>
</table>
Cross-layer Reliability

- Design-time task-mapping

### Increased number of design choices can cause design space explosion!!!
Conclusions

Reliability

Living with faults
- Faults, Errors and Failures
- Fault types: Transient, Intermittent, Permanent
- Fault Mechanisms: Soft-errors, Aging

QoS-aware

Application-specificity matters
- Leveraging applications’ inherent tolerances
- Joint consideration of multiple objectives and constraints
- Optimizations across design activities: Task-mapping, HW/HW Partitioning, etc.

Cross-layer

Breaking down abstractions
- Enables customization across multiple dimensions
- Exponential increase in design space
- Reconfiguration cost-aware dynamic Cross-layer Reliability

Looking ahead...

Research problems
- Cross-layer Reliability: Dealing with Vast Design Space, Using AI for improving run-time decision-making
- Reliability for Emerging Technologies: Silicon Nanowires, Carbon Nanotubes
- Reliability across Computing Paradigms: Machine Learning, Neuromorphic Computing
- Reliability in Internet-of-things: Billions of systems .. Not all regularly replaceable