Balancing Lifetime and Soft-Error Reliability to Improve System Availability

Junlong Zhou\textsuperscript{1*}, Xiaobo Sharon Hu\textsuperscript{2}, Yue Ma\textsuperscript{2}, Tongquan Wei\textsuperscript{1}

\textsuperscript{1}Department of Computer Science and Technology
East China Normal University, China

\textsuperscript{2}Department of Computer Science and Engineering
University of Notre Dame, USA

*Presenter

ASPDAC 2016, Macao, China
Outline

● Introduction
  ▪ Availability

● Preliminary Knowledge
  ▪ Permanent Fault
  ▪ Transient Fault

● Our Method
  ▪ Formula to Calculate the MTTF due to Transient Fault
  ▪ Framework to Maximize System Availability
  ▪ Heuristic Algorithm to Improve Reliability

● Simulation Setup and Results

● Summary and Future Work
What’s availability?

Availability is the state if an application being accessible to the end user, when running on a specific platform.

Unavailability (or called outage/downtime) is the time that a system is not available to an end user.
Cost of Outage/Downtime

Financial Cost of Outage Per Hour among Various Industries

<table>
<thead>
<tr>
<th>Industry</th>
<th>Cost ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brokerage Retail</td>
<td>6.5</td>
</tr>
<tr>
<td>Credit Card Sales Authorization</td>
<td>2.6</td>
</tr>
<tr>
<td>Airline Reservation Centers</td>
<td>90,000</td>
</tr>
<tr>
<td>Package Shipping Services</td>
<td>28,250</td>
</tr>
<tr>
<td>Manufacturing Industry</td>
<td>26,761</td>
</tr>
<tr>
<td>Banking Industry</td>
<td>17,093</td>
</tr>
<tr>
<td>Transportation Industry</td>
<td>9,435</td>
</tr>
</tbody>
</table>

Source: Contingency Planning Research & Strategic Research Corp.

Thus, we want to decrease outage (increase availability).
Types of Outages

- Two main types
  - Planned outages: Planned Maintain and Updates
  - Unplanned outages: Hardware Failure + System Software Bugs + Application Software Bugs

Source: Standish Group
Reasons of Outages

● Planned outages (controllable)
  - Such as database reorganization, release changes, network reconfiguration

● Unplanned outages (unexpected)
  - Such as hardware failure and software failure

*more than 50%, important*
Questions:
1. What are the permanent fault and transient fault?
2. How to improve the two reliabilities for achieving high availability?
What’s permanent fault? (or hard error)
- A type of failure that continues to exist until the faulty hardware is repaired or replaced

What causes permanent fault?
- Four main IC-dominant failure mechanisms: EM, TDDB, SM, and TC
IC-Dominant Failure Mechanisms

- **Electromigration (EM)**
  - Dislocation of metal atoms caused by momentum imparted by electrical current in wires and vias

- **Time-dependent dielectric breakdown (TDDB)**
  - Deterioration of the gate oxide layer

- **Stress migration (SM)**
  - Caused by the directionally biased motion of atoms in metal wires due to mechanical stress

- **Thermal cycling (TC)**
  - Wear due to thermal stress induced by mismatched coefficients of thermal
IC-Dominant Failure Mechanisms

- Failure rate $\lambda$ for EM, TDDB, and SM can be computed as

$$\lambda = K_1 \cdot e^{-\frac{k_2}{T}}$$

$T$: the temperature
$K_1, K_2$: temperature-independent constants
IC-Dominant Failure Mechanisms

- Number of cycles to failure $N_{TC}$ can be computed as

$$N_{TC} = K_3 \cdot (\Delta T - \Delta T_0)^{K_4} \cdot e^{-\frac{k_5}{T_{max}}}$$

$K_3$, $K_4$, $K_5$: temperature-independent constants

$\Delta T$: the thermal cycle amplitude

$\Delta T_0$: the temperature at which inelastic deformation begins

$T_{max}$: the maximal temperature during the cycle
Estimate Lifetime Reliability

- Mean Time to Failure (MTTF)
  - A common metric to quantify lifetime reliability

- Xiang’s tool to derive the MTTF due to permanent fault
  - Integrates three levels of models: device-, component-, and system-level models
  - Consider four IC-dominant failure mechanisms: EM, TDDB, SM, and TC
  - Output: the system-level MTTF

Estimate Lifetime Reliability

Device specification

Component specification

Device-level modeling

Selected distribution

Component-level modeling

System-level Monte Carlo simulation

Lifetime reliability

Flowchart of Xiang’s tool to estimate lifetime reliability
### Transient Fault

- **What’s transient fault? (soft error)**
  - A type of failure that appears for a short time and then disappears without damage to the device

- **What causes transient fault?**
  - Electromagnetic interference or cosmic radiation
Transient Fault

- Soft-error reliability can be determined by the exponential failure law

\[ R(f) = e^{-\lambda(f)\frac{C}{f}} \]

\( \lambda(f) \): the fault rate when task \( \tau \) operating at frequency \( f \)
\( C \): the number of task cycles
Need a uniform metric to quantify the two reliabilities

- User’s concern: mean time to first failure, regardless of the type of failure

- Difficult to gauge how tradeoffs should be made to achieve overall high system reliability without a uniform metric
  - Certain design decisions (e.g., task mapping and voltage scheduling) may increase lifetime reliability but decrease soft-error reliability or vice versa
Existing Reliability-Aware Methods

• Most only focus on one of the two reliability concerns
  - Lifetime reliability: e.g., Chantem et al., DATE 2013; Amrouch et al., ICCAD 2014; Duque et al., DATE 2015
  - Soft-error reliability: e.g., Li et al., ISCA 2008; Sridharan et al., ISCA 2010

• A few focus on handling permanent and transient faults simultaneously
  - E.g., Chou et al., DATE 2011, Das et al., DATE 2014: use separate metrics to perform reliability evaluation
Our Contributions

1. An analytical approach to calculate the MTTF due to transient fault
   - Enable the quantification of two reliabilities by a uniform metric

2. A single-objective optimization problem to maximize system availability
   - Consider both transient and permanent faults

3. A framework and a heuristic algorithm
   - Framework: solve the optimization problem
   - Heuristic algorithm: improve reliability for a specific scenario
System Model & Assumptions

- A uniprocessor platform
  - Supports a discrete set of frequencies

- A set of tasks repeatedly running on the processor
  - Tasks are non-preemptive and independent
  - Execution of task set in different runs are independent

- Replication to tolerate transient faults
  - Single-fault-tolerance
  - Occurrence of faults in tasks are independent
  - No fault propagation
Calculate the MTTF due to Transient Fault

The time to failure due to a transient fault of task $\tau_i$ in the $k$th run of task set $\mathcal{T}_n$

Illustration: the task set $\mathcal{T}_n = \{\tau_1, \tau_2, \ldots, \tau_n\}$ executes successfully during the prior $k - 1$ runs, but fails in the $k$th run due to the transient fault in task $\tau_i$.

- $T_{\text{exe}}(\mathcal{T}_n) = \sum_{i=1}^{n} t_i$: total execution time of task set $\mathcal{T}_n$
- $T_{\text{exe}}(\mathcal{T}_i) = \sum_{j=1}^{i} t_j$: execution time of tasks $\tau_1$ to $\tau_i$
Then, the MTTF due to transient fault, denoted by $MTTF_T$, can be calculated as

$$MTTF_T = \sum_{k=1}^{\infty} \sum_{i=1}^{n} \{(k - 1)T_{exe}(T_n) + T_{exe}(T_i)\} \cdot P_{succ}(T_{n,k-1}) \cdot P_{fail}(\tau_i)$$

- $P_{succ}(T_{n,k-1})$: probability that the first $k - 1$ runs of $T_n$ are all successful
- $P_{fail}(\tau_i)$: probability that $\tau_i$ is erroneous but $\tau_1 - \tau_{i-1}$ in the same run of $T_n$ are successful
Calculate the MTTF due to Transient Fault

Through a series of algebraic transformation, $MTTF_T$ can be derived as

$$MTTF_T = \frac{T_{exe}(T_n) + T_{exp}(T_n)}{P_{fail}(T_n)} - T_{exe}(T_n)$$

$T_{exp}(T_n) = \sum_{i=1}^{n} T_{exe}(T_i) \cdot P_{fail}(\tau_i)$ denotes the expected time to failure when the fault occurs in the first run.
The $MTTF_T$ is derived when assuming a workload $\mathcal{T}_n$ is being repeatedly executed forever.

What the $MTTF_T$ would be if treat two or more runs of $\mathcal{T}_n$ as the given workload being repeated forever?

**Theorem 1**: For any given task set $\mathcal{T}_n$ and any integer $m \geq 2$, let $\mathcal{T}_n^m$ be the task set containing $m$ runs of $\mathcal{T}_n$, i.e., $\mathcal{T}_n^m = \{\tau_1, \tau_2, \ldots, \tau_n, \tau_1, \tau_2, \ldots, \tau_n, \ldots, \tau_1, \tau_2, \ldots, \tau_n\}^{n \times m}$. Then $MTTF_T(\mathcal{T}_n^m) = MTTF(\mathcal{T}_n)$ holds.
Problem Formulation

- Definition of availability: \( A = \frac{MTTF}{MTTF + MTTR} \)

- For a system that may suffer from both transient and permanent faults, maximizing system availability (objective) becomes

\[
\max: \left\{ \frac{MTTF_T}{MTTF_T + MTTR_T} \cdot \frac{MTTF_P}{MTTF_P + MTTR_P} \right\}
\]

same goal

\[
\max: \{ \gamma MTTF_T, MTTF_P \}
\]

equivalent to

\[
\max \min: \{ \gamma MTTF_T, MTTF_P \}
\]

where \( \gamma = \frac{MTTR_P}{MTTR_T} \) is assumed to be a given constant.
Four Scenarios in Our Problem

- Determine the $MTTF_T$ using our formula and the $MTTF_P$, using Xiang’s tool
  - Identify which reliability dominates

- Group the relationship between $\Upsilon MTTF_T$ and $MTTF_P$ into four scenarios
  1. $\Upsilon MTTF_T \ll MTTF_P$
  2. $\Upsilon MTTF_T < MTTF_P$
  3. $\Upsilon MTTF_T > MTTF_P$
  4. $\Upsilon MTTF_T \gg MTTF_P$
The Existence of Four Scenarios

Setup
- the same core, benchmarks, and parameter settings as in GVLSI’s work and $\gamma = 1$
Framework to Maximize Availability

Start

Calculate $MTTF_T$ and $MTTF_P$

$YMTTF_T < MTTF_P$

$YMTTF_T \ll MTTF_P$

$Y$

Full replication and speedup

Partial replication and speedup

$N$

$N$

DVS-based strategy

Lifetime reliability-aware strategy

Workload, processor specification

$Y$

soft-error reliability dominates

End

lifetime reliability dominates
Countermeasures for Four Scenarios

- $Y M T T F_T < M T T F_P$
  - Full replication and speedup
    - Every task with a recovery, at maximal frequency

- $Y M T T F_T < M T T F_P$
  - Partial replication and speedup
    - A part of tasks are replicated or sped up

- $Y M T T F_T > M T T F_P$
  - DVS-based strategy
    - Reduce the temperature by scaling frequency

- $Y M T T F_T \geq M T T F_P$
  - Lifetime reliability-aware strategy
    - E.g., mitigate aging speed
Our Focus

- Full replication and speedup
  - Simple and easy to implement
- DVS-based strategy
  - Widely explored
- Lifetime reliability-aware strategy
  - Widely explored
- Partial replication and speedup
  - A few works, lack a specific strategy

So Partial replication and speedup is our concern
Our Heuristic Algorithm

1. Start
2. Initialize the system current setup, denoted by $\text{Setup}_{\text{cur}}$.
3. Calculate $MTTF_P(\text{Setup}_{\text{cur}})$ and $MTTF_T(\text{Setup}_{\text{cur}})$.
4. If $MTTF_T(\text{Setup}_{\text{cur}}) < MTTF_P(\text{Setup}_{\text{cur}})$, go to step N; otherwise, continue.
5. Determine which task’s choice should be actually adopted to maximize $\Delta YMTTF_T$.
6. Decide the best choice (be replicated or sped up) for every task in terms of increasing $YMTTF_T$.
7. Update $\text{Setup}_{\text{cur}}$.
8. Workload, frequency, wear state, etc.
9. End.
Simulation Setup

- **Hardware platform**
  - Alpha 21264 microprocessor, 5 frequency levels
- **Synthetic, real-world app. based benchmarks**
  - Five sets of 20 randomly generated tasks
  - Embedded System Benchmark Suite [Univ. of Michigan] including
    - Autom.-industrial, consumer-networking, telecom, mpeg
- **Simulation tools**
  - HotSpot 5.0 [Univ. of Virginia], Xiang’s tool
- **Algorithms used for comparison**
  - Random, full speed, full replication algorithms
  - Energy-efficient and reliability-aware algorithm [ACM TAES 2013]
Simulation Results

RA: random algorithm, FSA: full speed algorithm, FRA: full replication algorithm, ERA: energy-efficient and reliability-aware algorithm, PRS: Partial replication and speedup

Up to 85%
Simulation Results

RA: random algorithm, FSA: full speed algorithm, FRA: full replication algorithm, ERA: energy-efficient and reliability-aware algorithm, PRS: Partial replication and speedup

Up to 45%
Summary & Future Work

- Existing reliability-aware methods lack a uniform metric to quantify lifetime and soft-error reliability.

- We proposed an analytical approach to enable the evaluation of two reliabilities.

- We presented a framework and a heuristic algorithm to maximize system availability.
  - Our framework: designed for all scenarios
  - Our algorithm: designed for a specific scenario

- In the future, we will consider real-time systems and multicore platforms.
Thank you!