ASAR: Application-Specific Approximate Recovery to Mitigate Hardware Variability

Presenter: Manish Gupta

Collaborators: Abbas Rahimi, Daniel Lowell, John Kalamatianos,

Advisors: Dean Tullsen, Rajesh Gupta
Variability

From Chiseled Objects to Molecular Assemblies

Temperature Clock

Actual circuit delay

Guardband

Aging

\[ V_{CC} \]

Droop

Across wafer Frequency

Process Temperature Aging Vcc Droops

Greater Performance
Lower Power

Higher Variations

The paradigm simulation yesterday

At 32nm (physical gate length) MOSFET in production today

At 9nm MOSFET in production by 2023

Courtesy A. Asenov
Univ. of Glasgow

2
Mitigating Variability Effects

Detect Errors in Hardware

- Ignore
- Fix
Today,

- Today, we detect and fix in hardware.
- Using hardware error recovery.
• In this work, we champion software recovery [1].

Ignore (Discard) vs. Fix (Rerun)

Detect Errors in Hardware

Ignore (Discard)

Fix (Rerun)

Software

Discard may lead to unacceptable output quality.

Rerun suffers from high recovery overhead.
• Variability Problem
• Hardware/Software Recovery
• Software Recovery (Rerun or Discard) [Relax ISCA 10]
• Emerging Approximate Applications
• ASAR: Application Specific Approximate Recovery
• Hybrid Recovery (ASAR + Rerun)
• Summary
Emerging Approximate Applications

Data-intensive

Scientific-Computing

Vision & media

BIG DATA ANALYTICS

audio & video

Analogue

Digital

Time
Approximate Recovery

Detect Errors in Hardware

- Ignore (Discard)
- Approximate Recovery
- Fix (Rerun)

Software

ASAR: Application-Specific Approximate Recovery

- Worst output quality
- Best recovery performance
- Good output quality
- Good recovery performance
- Best output quality
- Worst recovery performance
HW/SW Organization

Main contributions of this work is SW-based recovery using:
1) Rerun
2) Discard
3) ASAR (Approx. recovery)
4) Hybrid (Rerun + ASAR)

<table>
<thead>
<tr>
<th>HW error detection [2, 3]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Safe mode</strong></td>
</tr>
</tbody>
</table>

Software

| Hardware |

2. S. Agarwal “BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data” EuroSys 2013
3. Y. Ding et al., “A Compiler Scheme for Reusing Intermediate Computation Result” CGO, 2004
Try-Catch Refactoring: ASAR Reuse

K-means for N points \([P_1, P_2, \ldots, P_N]\).

Assign the closest of M center \([C_1, C_2, \ldots, C_M]\).

Next, two questions we explore for all application are:


2) Absolute performance gain of using approximate recovery over rerun.

In each iteration

For \(P_i\) in \([P_1, \ldots, P_N]\) {

Try {
    \(d_{\text{min}} = \text{INT\_MAX};\)
    \text{for} \ c \ \text{in} \ [C_1, C_2, \ldots, C_M]:\)
    \(d = \text{distance}(P_i, c);\)
    \text{if} (d < d_{\text{min}}) \{
        \text{Cen}[P_i] = c;
        \text{ReuseBuf}[P_i] = c;
    \}
}

Catch {
    \text{Cen}[P_i] = \text{ReuseBuf}[P_i]
}

} //for \(P_i\) in \([P_1, \ldots, P_N]\)
Answering 1) and 2)

1) Critical vs. Non-critical

2) Catch { ASAR }

Catch { Rerun }
Evaluation

• Six application from various domains.

• SW-based using 1) rerun, 2) discard, 3) ASAR, and 4) hybrid (ASAR + Rerun) recovery.

• Random fault-injection simulated.

• Finally, we measure cpu cycles spent in critical, non-critical, and recovery code regions.
ASAR vs. Rerun
K-Means
K-Means

![Graph showing normalized execution time vs. failure rate with QoS metrics.]

- Normalized execution time
- Failure rate (Try block)
- QoS (%)
K-Means
ASAR vs. Rerun

- WordCount
- K-means
- SOR
- MonteCarlo
- A2time
- FIR

Normalized execution time vs. Failure rate of try block (%)

QoS (%) vs. Normalized execution time

ASAR vs. Rerun

QoS(Rerun) vs. QoS(ASAR)
Hybrid (ASAR + Rerun) Recovery

- ASAR only provides one additional operating point.
- Only approximate recovery could degrade below user acceptable level.
- Hybrid recovery to explore performance-quality at a finer-granularity and allow user to specify desired output quality.
Hybrid (ASAR + Rerun) Recovery
K-means
K-means
Hybrid (ASAR + Rerun) Recovery
Summary

• We propose **ASAR**, application-specific approximate recovery.

• To guarantee the output acceptability and explore performance-quality curve at a finer-granularity we also propose a **hybrid recovery** mechanism.

• We show that, depending on the application characteristics, approximate recovery can reduce recovery overhead and improve output quality relative to rerun and discard, respectively.
Thanks!