Characterization of Error Tolerant Applications When Protecting Control Data

Darshan D. Thaker
John Oliver
Tzvetan S. Metodi
UC Davis

Derek Lockhart
Diana Franklin
CalPoly
San Luis Obispo

Susmit Biswas
Frederic T. Chong
UC Santa Barbara
With decrease in feature size, soft errors on the rise.

Most research primarily at architectural level or at circuit level.

Understanding degradation of fidelity in an application, in presence of errors, is critical.

Use static analysis to protect instructions affecting control flow.
Overview of recent work.

Static analysis mechanism.

Fidelity measure of an application.

Experiments and results.

Concluding remarks.
Soft Errors in $\mu$-Processors

Studies of architectural vulnerability factors.
(Biswa 05, Weaver 04, Mukherjee 03)

Circuit level mitigation techniques.
(Mitra 05, Iyer 05, Wang 04)

Concurrent thread execution.
(Smolens 05, Gomaa 03, Reinhardt 00)

Analyzed behavior of algorithms from stereo vision and speech recognition.
(Vong 06)
Perceptual applications can tolerate soft errors.

Extreme example to make errors obvious.

Fidelity measure of an application can quantify degraded output.

Example: Mpeg 2 decoder.
Mpeg with Errors
Overview of recent work.

Static analysis mechanism.

Fidelity measure of an application.

Experiments and results.

Concluding remarks.
Programmer identifies functions tolerant to errors.

Static Analysis: start at a branch and use Def-Use chain to identify instructions that effect the outcome of that branch.

Tag instructions that do not effect the branch. These instructions can be exposed to errors.

All other instructions are assumed to be protected.
Static Analysis Example

B.Blk 0:

I0: \( \$2 = \$4 + 1 \)
I1: \( \text{LD} \ $3, \text{addr} \)
I2: \( \$2 = \$3 + 2 \)
I3: \( \$3 = \$3 + 8 \)
I4: \( \$10 = \$8 - \$4 \)

\[ \{ \$3, \$10 \} \in \text{CVAR} \]

B.Blk 1:

I5: \( \$10 = \$3 \ll \$2 \)
I6: \( \$4 = \$3 + \$6 \)
I7: \( \$3 = \$3 + 1 \)
I8: \( \text{BNE} \ $3, \$10, \text{label} \)
Static Analysis Example

B.Blk 0:

..

I0:  $2 = $4 + 1
I1:  LD  $3, addr
I2:  $2 = $3 + 2
I3:  $3 = $3 + 8
I4:  $10 = $8 - $4

[ $3, $10] ∈ CVAR

B.Blk 1:

I5:  $10 = $3 << $2
I6:  $4 = $3 + $6

I7:  $3 = $3 + 1
I8:  BNE  $3, $10, label
B.Blk 0:

\[ I_0: \quad \$2 = \$4 + 1 \]
\[ I_1: \quad \text{LD} \quad \$3, \text{addr} \]
\[ I_2: \quad \$2 = \$3 + 2 \]
\[ I_3: \quad \$3 = \$3 + 8 \]
\[ I_4: \quad \$10 = \$8 - \$4 \]

\[ [\$3, \$10] \in \text{CVAR} \]

B.Blk 1:

\[ I_5: \quad \$10 = \$3 << \$2 \]
\[ I_6: \quad \$4 = \$3 + \$6 \]
\[ I_7: \quad \$3 = \$3 + 1 \]
\[ I_8: \quad \text{BNE} \quad \$3, \$10, \text{label} \]
Static Analysis Example

B.Blk 0:

I0:   $2 = $4 + 1
I1:  LD $3, addr
I2:   $2 = $3 + 2
I3:   $3 = $3 + 8
I4:   $10 = $8 - $4

[ $3, $2] \in CVAR

B.Blk 1:

I5:   $10 = $3 \ll $2
I6:   $4 = $3 + $6
I7:   $3 = $3 + 1
I8:  BNE $3, $10, label
B.Blk 0:

I0: $2 = $4 + 1
I1: LD $3, addr
I2: $2 = $3 + 2
I3: $3 = $3 + 8
I4: $10 = $8 - $4

[ ] ∈ CVAR

B.Blk 1:

I5: $10 = $3 << $2
I6: $4 = $3 + $6
I7: $3 = $3 + 1
I8: BNE $3, $10, label
Overview of recent work.

Static analysis mechanism.

Fidelity measure of an application.

Experiments and results.

Concluding remarks.
Compare SNR of every frame.
Difference > 2dB, implies bad frame.
Fidelity measure: number of bad frames.
Susan (MiBench):
Edge detection. Compare SNR of outputs.

MCF (SPEC 2000int):
Generates optimal schedules.
Fidelity measure: Distance from optimal schedule.

Art (SPEC 2000int):
Neural net to identify objects in an image.
Fidelity measure: % objects identified correctly.
Blowfish (MiBench):
Encryption - Decryption algorithm.
Fidelity measure: % matching bytes post-decryption.

GSM (MiBench):
Audio compression scheme.
Fidelity measure: SNR of output.
Overview of recent work.
Static analysis mechanism.
Fidelity measure of an application.

Experiments and results.

Concluding remarks.
Errors randomly inserted as single bit-flips.
Every error alters the result of an ALU operation.
Errors inserted only in tagged instructions.
Used SimpleScalar for our simulations and for error insertions.
## Benefits of Static Analysis

<table>
<thead>
<tr>
<th>Application</th>
<th>Errors inserted</th>
<th>Total instructions (millions)</th>
<th>Failures with static analysis</th>
<th>Failures without static analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Susan</td>
<td>2200</td>
<td>144</td>
<td>0 %</td>
<td>10 %</td>
</tr>
<tr>
<td>Mcf</td>
<td>320</td>
<td>201</td>
<td>6 %</td>
<td>100 %</td>
</tr>
<tr>
<td>Mpeg</td>
<td>120</td>
<td>2740</td>
<td>0 %</td>
<td>100 %</td>
</tr>
<tr>
<td>GSM</td>
<td>40</td>
<td>892</td>
<td>0 %</td>
<td>100 %</td>
</tr>
<tr>
<td>Blowfish</td>
<td>20</td>
<td>507</td>
<td>19 %</td>
<td>48 %</td>
</tr>
<tr>
<td>Art</td>
<td>4</td>
<td>42770</td>
<td>0 %</td>
<td>0 %</td>
</tr>
</tbody>
</table>
Results: Mpeg and Susan

Mpeg
(0.80 billion cycles)

Susan
(0.51 billion cycles)
Results: Blowfish and GSM

Blowfish
(0.18 billion cycles)

GSM
(0.38 billion cycles)
Results: Mcf and Art

For Mcf (0.3 billion cycles):
- Num. Failures: Decrease with increasing errors inserted.
- % Distance from optimal: Decrease with increasing errors inserted.

For Art (15.27 billion cycles):
- Num. Failures: Decrease with increasing errors inserted.
- % Items recognized: Decrease with increasing errors inserted.
Overview of recent work.

Static analysis mechanism.

Fidelity measure of an application.

Experiments and results.

Concluding remarks.
Certain applications have an inherent ability to tolerate soft errors.

Simple static analysis protects code especially susceptible to errors.

Loss of fidelity, when errors inserted, within acceptable limits.
## Instructions Tagged

<table>
<thead>
<tr>
<th>Application</th>
<th>Errors inserted</th>
<th>Total instructions (millions)</th>
<th>Tagged instructions</th>
<th>Failures with static analysis</th>
<th>Failures without static analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Susan</td>
<td>2200</td>
<td>144</td>
<td>91.3%</td>
<td>0 %</td>
<td>10 %</td>
</tr>
<tr>
<td>Mcf</td>
<td>320</td>
<td>201</td>
<td>8.9%</td>
<td>6 %</td>
<td>100 %</td>
</tr>
<tr>
<td>Mpeg</td>
<td>120</td>
<td>2740</td>
<td>50.3%</td>
<td>0 %</td>
<td>100 %</td>
</tr>
<tr>
<td>GSM</td>
<td>40</td>
<td>892</td>
<td>19.6%</td>
<td>0 %</td>
<td>100 %</td>
</tr>
<tr>
<td>Blowfish</td>
<td>20</td>
<td>507</td>
<td>62.4%</td>
<td>19 %</td>
<td>48 %</td>
</tr>
<tr>
<td>Art</td>
<td>4</td>
<td>4277</td>
<td>70.8%</td>
<td>0 %</td>
<td>0 %</td>
</tr>
</tbody>
</table>
Differentiate between data, pointer and stack variables during static analysis.

Feedback-directed optimizations driven by programmer-defined fidelity measures.

Architectural mechanisms to protect non-tagged instructions.
Thank you!

Your questions...
## Error Rate

<table>
<thead>
<tr>
<th>Application</th>
<th>Errors inserted</th>
<th>Total instructions (millions)</th>
<th>Tagged instructions</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Susan</td>
<td>2200</td>
<td>144</td>
<td>91.3%</td>
<td>1.6 e-5</td>
</tr>
<tr>
<td>Mcf</td>
<td>320</td>
<td>201</td>
<td>8.9%</td>
<td>1.7 e-5</td>
</tr>
<tr>
<td>GSM</td>
<td>40</td>
<td>892</td>
<td>19.6%</td>
<td>2.28 e-7</td>
</tr>
<tr>
<td>Blowfish</td>
<td>20</td>
<td>507</td>
<td>62.4%</td>
<td>6.32 e-8</td>
</tr>
<tr>
<td>Mpeg</td>
<td>120</td>
<td>2740</td>
<td>50.3%</td>
<td>8.7 e-8</td>
</tr>
<tr>
<td>Art</td>
<td>4</td>
<td>4277</td>
<td>70.8%</td>
<td>1.3 e-9</td>
</tr>
</tbody>
</table>