Acknowledgments

- Prof. Edward J. McCluskey
- Dr. Nahmsuk Oh
- Naval Research Laboratory (NRL)
- Funding: BMDO/IST, NASA
The Challenge – Stanford CRC ARGOS Project

- Determine to what extent commercial electronics, e.g., microprocessors and RAMs, can be used in space
The Approach

1. Design an Experiment to Collect Data
   - Actual satellite
   - Compare rad-hard and COTS boards
   - Evaluate fault tolerance (FT) techniques

2. Develop Techniques for Fault Tolerance
   - Software based – no special hardware
     - EDDI, CFCSS
     - Software-implemented EDAC

3. Develop a Method
   - Estimate distribution of errors
     - In various functional units
Outline

- Motivation and Background
- ARGOS Space Experiment Setup
- Software-Implemented Hardware Fault Tolerance
- Experiment Results
- Conclusion
Motivation

- Reliable Computing in Space
  - Failures Caused by Radiation
    - e.g., Single-Event Upsets (SEUs)
- Problems
  - Costly classical solutions
    - Hardware duplication
    - Radiation-hardening
  - Increasing sensitivity to radiation
    - Deep submicron technologies
Radiation Sources

- Sources in Space
  - Radiation belts
    - Particles trapped in Earth’s magnetic field
  - Solar winds
  - Galactic cosmic rays
- Sources on Earth
  - \(\alpha\)-particles from radioactive material
  - Secondary particles from cosmic rays
  - Thermal neutrons
Radiation-Matter Interaction

- Electronic Charge Displacement (Ionization)
  - Electron-hole pair production
    - Short current pulse (causing an SEU)
    - Trapped holes in dielectrics

![Diagram of radiation interaction with a gate, substrate, and ion track with electric field]
Radiation Effects

- Cumulative Long-Term Degradation
  - Total Ionizing Dose (TID)
  - Displacement Damage Dose (DDD)
- Single-Event Effects (SEEs)
  - Single incident ionizing particle
  - Permanent or transient effects
  - Global or local effects
  - e.g., Single-Event Upsets (SEUs)
    - Soft errors, soft fails, transients
Mitigating Radiation Effects

- Fault Avoidance
  - Shielding
    - Heavy, large volume
  - Radiation hardening
    - Expensive, limited availability, old designs
- Fault Tolerance
  - Redundancy (hardware or software)
    - Overhead: price, performance, power, ...
Problem Definition

- Computation in Radiation Environments
  - Without customized or rad-hard components

Solutions

- Commercial Off-The-Shelf (COTS) Components
  - Relatively cheaper and higher performance
- Software based FT Techniques
  - Reliability improvement for COTS
Outline

- Motivation and Background
- ARGOS Space Experiment Setup
- Software-Implemented Hardware Fault Tolerance
- Experiment Results
- Conclusion
Advanced Research and Global Observation Satellite

- Launch: Feb. 23, 1999
- Polar LEO orbit
  - 800 km Altitude, Sun Synchronous, 98° Inclination
- 9 Experiments
  - Including USA (Unconventional Stellar Aspect) experiment of NRL
    - Computing testbed
The ARGOS Project – Computing Testbed

- Reliable Computing in Space
  - Autonomous navigation and data processing
- Goals
  - Comparison of rad-hard & COTS components
  - Evaluation of software-based FT techniques
  - Collection of error data
    - From a real space experiment
    - No simulation or fault injection
Previous Work

- **Ground Testing**
  - Artificial fault injection

- **Space Testing**
  - University of Surrey [Underwood 98]
    - COTS SRAMs for micro-satellites
  - MPTB [Dale 95]
    - RAMs, microprocessor, photonic devices
  - Hiten Satellite [Takano 96]
    - Hardware FT technique
Computing Testbed

- Hard Board
  - Harris RH3000 rad-hard chip set
  - SOI SRAMs
  - Hardware FT techniques
    - Self-checking pair processors
    - EDAC for memory
- COTS Board
  - IDT R3081
  - No error detection hardware
    - No EDAC
  - Only software FT techniques
Outline

- Motivation and Background
- ARGOS Space Experiment Setup
- **Software-Implemented Hardware Fault Tolerance**
- Experiment Results
- Conclusion
Software-Implemented Hardware Fault Tolerance (SIHFT)

- Software-Implemented EDAC [Shirvani 00]
  - Error Detection And Correction (EDAC)
  - SEU protection for main memory
- Error Detection by Duplicated Instructions (EDDI) [Oh 02-1]
- Control Flow Checking by Software Signatures (CFCSS) [Oh 02-2]
- Software-Implemented Error Recovery
Software-Implemented EDAC

- Intercepting all Reads and Writes
  - Infeasible in software
- Periodic Scrubbing
  - Scrubbing:
    - Reading out memory and correcting errors
  - Periodic:
    - e.g., every 30 seconds
  - Limited to memory blocks with fixed contents:
    - Code segments
    - Read-only data segments
Self-Repair for EDAC Software

● Issue
  ▪ SEU in code segment of EDAC software
    ● Cannot repair itself

● Solution
  ▪ Cross-Checking Pair
    ● Each copy scrubs the other one
    ● Assuming single error
Error Detection by Duplicated Instructions (EDDI)

- Duplicate Instructions
  - Master and shadow instructions
- Compare Master and Shadow Results
  - Detect transient errors in computations

```
ADD R3, R1, R2 ; R3 <- R1 + R2
MUL R4, R3, R5 ; R4 <- R3 * R5
ST 0(SP), R4 ; store R4 in location pointed by SP

ADD R3, R1, R2 ; R3 <- R1 + R2 master
ADD R23, R21, R22 ; R23 <- R21 + R22 shadow
MUL R4, R3, R5 ; R4 <- R3 * R5 master
MUL R24, R23, R25 ; R24 <- R23 * R25 shadow
BNE R4, R24, ErrorHandler ; compare master and shadow results
ST 0(SP), R4 ; store master result
ST offset(SP), R24 ; store shadow result
```
Control Flow Checking by Software Signatures (CFCSS)

- Assigned Signature Analysis Method
  - Unique signature for each basic block
- Interblock Control Flow Checking
  - Correct sequence of blocks followed
- Signature Comparison
  - Pure software
  - No extra hardware
Flow for Adding EDDI and CFCSS

- C source
- CC (gcc)
- Assembly code
- Post Processor
- Assembly code with EDDI/CFCSS
- Assembler
- Object code
COTS Board – Error Collection & Recovery Software

- EDAC1
- EDAC2
- Profiler
- Collector
- Watchdog
- Main Control
- Test Programs with EDDI & CFCSS
- Ground Program
- OS
Software Modules

- Main Control
  - Overall coordination
- Watchdog Timer
  - Detect hang-ups
- Profiler
  - Measure each task’s CPU time
- Collector
  - Store information about errors
- Cross-Checking EDAC Pair
  - Detect and correct memory errors
Error Recovery

- **Goal**
  - Automatic recovery
    - Without assistance from ground station

- **Mechanism**
  - Separate task for each module (multitasking)
    - Independent contexts

- **Steps**
  1. Error is detected
  2. The erroneous task is terminated
  3. Code segment of task is checked by EDAC
  4. The task is restarted
Outline

- Motivation and Background
- ARGOS Space Experiment Setup
- Software-Implemented Hardware Fault Tolerance
- Experiment Results
- Conclusion
Hard Board – Test Programs

- Memory Test
  - Write a pattern in a block of memory
  - Loop
    - Read back and check for correct pattern

- Sine Table Generation
  - Load table
  - Loop
    - Calculate a sine table entry
    - Compare with entry in table
Hard Board – Experiment Results

<table>
<thead>
<tr>
<th>Program</th>
<th>Data Size (KB)</th>
<th>Running Period (days)</th>
<th>Num. of Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Test</td>
<td>256</td>
<td>140</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>349</td>
<td>4</td>
</tr>
<tr>
<td>Sine Table</td>
<td>128</td>
<td>191</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>250</td>
<td>9</td>
</tr>
</tbody>
</table>

- All Errors Detected by Software
- No Parity or EDAC Errors
- Agreement in Self-Checking Pairs
  - Suspected Source of Errors
    - Shared components, e.g., data buffers
Hard Board SEU Map

South Atlantic Anomaly (SAA)
COTS Board SEU Map
COTS Board – Experiment Results

- Mostly Memory SEUs
  - 5.55 SEUs/MByte per day
- Software-Implemented EDAC
  - Protected memory size: 450KByte
  - Running time: 329 days
  - Errors Detected and Corrected: 831
- Reliability Improvement
  - Time to crash:
    - 2 days without software EDAC
    - 20 days with software EDAC
Memory SEUs

- Fixed Pattern Test: 55 hex = 01010101 binary

<table>
<thead>
<tr>
<th>Data Size</th>
<th>Running Period</th>
<th>Num. of Errors</th>
<th>SEU / MB per Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>512KB cached</td>
<td>18 days</td>
<td>41</td>
<td>4.56</td>
</tr>
<tr>
<td>256KB cached</td>
<td>36 days</td>
<td>48</td>
<td>5.33</td>
</tr>
<tr>
<td>128KB cached</td>
<td>84 days</td>
<td>54</td>
<td>5.14</td>
</tr>
<tr>
<td>128KB non-cached</td>
<td>84 days</td>
<td>59</td>
<td>5.62</td>
</tr>
</tbody>
</table>

- Overall Results
  - Average SER = 5.55 SEUs/MB per day
  - MBUs: 1.44%
# Pattern Sensitivity of SEUs

<table>
<thead>
<tr>
<th>Pattern (hex)</th>
<th>Errors</th>
<th>0 to 1 bit-flips</th>
<th>1 to 0 bit-flips</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Num</td>
<td>%</td>
<td>Num</td>
</tr>
<tr>
<td>00,00,00,00,…</td>
<td>45</td>
<td>15.0</td>
<td>45</td>
</tr>
<tr>
<td>FF,FF,FF,FF,…</td>
<td>45</td>
<td>15.0</td>
<td>0</td>
</tr>
<tr>
<td>00,FF,00,FF,…</td>
<td>34</td>
<td>11.3</td>
<td>12</td>
</tr>
<tr>
<td>FF,00,FF,00,…</td>
<td>36</td>
<td>12.0</td>
<td>15</td>
</tr>
<tr>
<td>AA,AA,AA,AA,…</td>
<td>43</td>
<td>14.3</td>
<td>23</td>
</tr>
<tr>
<td>AA,55,AA,55,…</td>
<td>52</td>
<td>17.3</td>
<td>23</td>
</tr>
<tr>
<td>55,55,55,55,…</td>
<td>46</td>
<td>15.3</td>
<td>17</td>
</tr>
<tr>
<td>Total</td>
<td>301</td>
<td>100.0</td>
<td>135</td>
</tr>
</tbody>
</table>


COTS Board – Error Detection Coverage

- Test Programs
  - Insert sort, Quick sort, and FFT
- Error Detection Techniques
  - EDDI + CFCSS + Watchdog Timer
- Checking for Undetected Errors
  - Sort check
  - Checksum of FFT results
## Error Detection Coverage

<table>
<thead>
<tr>
<th>Test Program</th>
<th>Num. of Errors</th>
<th>Errors Detected</th>
<th>Undetected Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>EDDI</td>
<td>CFCSS</td>
</tr>
<tr>
<td>Insert Sort – Int.</td>
<td>156</td>
<td>156</td>
<td>–</td>
</tr>
<tr>
<td>Insert Sort – FP</td>
<td>21</td>
<td>21</td>
<td>–</td>
</tr>
<tr>
<td>Quick Sort – Int.</td>
<td>43</td>
<td>31</td>
<td>5</td>
</tr>
<tr>
<td>FFT – FP</td>
<td>102</td>
<td>99</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>322</strong></td>
<td><strong>307</strong></td>
<td><strong>6</strong></td>
</tr>
</tbody>
</table>

- **99.7% Detection Coverage**
- **98.8% Successful Recovery**
Throughput Comparison

- Hard Board
  - 10 MHz, no cache memory
- COTS Board
  - 25 MHz, 4KB I-cache, 16KB D-cache
    - 25 times faster without SIHFT
  - SIHFT Overhead
    - EDDI & CFCSS: 170%
    - Software EDAC: 3%
  - One order of magnitude faster
Outline

- Motivation and Background
- ARGOS Space Experiment Setup
- Software-Implemented Hardware Fault Tolerance
- Experiment Results
- Conclusion
ARGOS Conclusions

● Rad-Hard Board
  - Failures despite all hardware FT techniques
    - Single points of failure
● COTS Board
  - Effective software FT techniques
    - Error detection, correction and recovery
● COTS + SIHFT
  - Viable techniques
    - Low radiation environments (such as LEO)
The Challenge

- Determine to what extent commercial electronics, e.g., microprocessors and RAMs, can be used in space

The Answer

- COTS + SIHFT
  - Viable for low radiation environments

Demonstration

- Successful operation of COTS + SIHFT in ARGOS
  - Inspite of 5.55 SEUs/MByte per day
References

Publications


<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARGOS</td>
<td>Advanced Research and Global Observation Satellite</td>
</tr>
<tr>
<td>CFCSS</td>
<td>Control Flow Checking by Software Signatures</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial Off-The-Shelf</td>
</tr>
<tr>
<td>EDAC</td>
<td>Error Detection And Correction</td>
</tr>
<tr>
<td>EDDI</td>
<td>Error Detection by Duplicated Instructions</td>
</tr>
<tr>
<td>FT</td>
<td>Fault Tolerance</td>
</tr>
<tr>
<td>LEO</td>
<td>Low Earth Orbit</td>
</tr>
<tr>
<td>MBU</td>
<td>Multiple-Bit Upset</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>SAA</td>
<td>South Atlantic Anomaly</td>
</tr>
<tr>
<td>SEU</td>
<td>Single Event Upset</td>
</tr>
<tr>
<td>SIHFT</td>
<td>Software-Implemented Hardware Fault Tolerance</td>
</tr>
</tbody>
</table>