Error Detection and Collection Software

The CRC ARGOS Project:
Error Detection and Collection Software

by

Philip P. Shirvani

RATS
January 11, 1999

Outline

- Introduction
- System Setup
- Software Design
- Error Detection Software
- Summary

Introduction

- Collect Data
  - Processor errors in space
  - Hard vs. COTS board
  - Software error detection techniques
- Determine Tradeoffs
  - Fault avoidance
    - Radiation hardening
  - Fault tolerance
    - COTS components
    - Hardware and software techniques

ARGOS Experiment

Uplink Flow Chart

Downlink Flow Chart
Error Detection and Collection Software

Processor Boards
- Rad-Hard Board
  - 10 MHz, no cache memory
  - Self-Checking Pair, EDAC for Main RAM
  - VxWorks ver. 5.0.1
- COTS Board
  - 32 MHz, 16KB I-cache, 4KB D-cache
  - No error detection hardware
  - VxWorks ver. 5.3.1

Requirements
- Error Detection
  - Exercise functional units
  - Counter for each technique (h.w. and s.w.)
- Error Collection
  - Log errors safely
  - Short and long logs
- Correct Transmission of Log

Constraints
- Limited Memory (2MB)
  - Small programs and data structures
- Limited Communication Bandwidth
  - Uplink: 1.1kbps
  - Downlink: 40kbps; 128kbps
  - 8 min. window every 101.6 min.
  - Object code < 40KB
- Readiness for Uplink
  - Quick response time

Software Modules
- Incremental Upload

Main Control Program
- Command Interpreter
  - Adding/deleting subroutines
  - Change parameters
- Calling/Forking Subroutines
  - Frequency controlled loops
  - Controlling the watchdog task
- Logging Error
- Telemetry

Modular Design
- Dynamic Linking
- Upload Submodules
  - New, update, repair
  - Guard function calls with flags
- Advantages
  - Efficient use of bandwidth
  - Concurrent with program execution
- (Tested on RH3000)
Main Loop

```c
main()
|
 initialize();
 while (main_status != SHUTDOWN)
 |
 process_ground_command();
 run_next_test();
 send_heartbeat();
 |
 |
```

Test Loops

```c
run_next_test()
 |
 if (test_present[t] == TRUE)
 for(i=0; i<test_freq[t]; i++)
 |
 test_healthy[t] = ((void*)TEST[t])();
 |
 if (test_healthy[t] == FALSE)
 break;
 |
 t++;
 |
```

Ground Commands

- **8-Byte Commands (Including Checksum)**
  - Command: **TF n, f**  |  Description: Set the frequency of test 'n' to 'f' times
  - Command: **RT n**  |  Description: Re-transmit log number 'n'
  - Command: **SI n**  |  Description: Send information on test 'n'
  - Command: **PF**  |  Description: Send profile
  - Command: **HL**  |  Description: Halt (quit)

Error Distribution

- Occurrence Probability
  - Memory and CPU use
- Computation Modules
  - FFT, compression, sort, etc.
  - Large and time consuming
  - Detect error and abort
- All Other Modules
  - Small and quick
  - Not fully fault-tolerant

Software Error Detection

- Time Redundancy
  - Stutter-Step Mode execution (SSM)
  - Software duplication/TMR
- Control Flow Checking
  - Signature Analysis by Instructions (SAI)
  - Watchdog task and timers
- Other
  - Algorithm-Based Fault Tolerance (ABFT)
  - Assertions
  - Diversified values, duplicated data

Programming Practices

- Detect Error in Control Variables
  - Loop counters, flags
- Avoid Common Values
  - 0, 1, -1, small integers, TRUE, FALSE
  - e.g.: `if (test_present[t] == TRUE)`
- Check Both Values of Flag
  - Both ‘if’ and ‘else’ condition
Error Detection and Collection Software

Error Handling and Logging
- Main Control Module
  - Quick handlers
  - Quick report (error count)
- Collector
  - Complete log
  - Doubly link list
  - Error detection and correction
  - Scrubbing

Summary
- Maximize Error Detection Coverage
- Modular Design
- Error Detection Techniques
- Fault Tolerance for Critical Data

Project Plan

<table>
<thead>
<tr>
<th>Module</th>
<th>Error Detection</th>
<th>Status / Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Control</td>
<td>Manual, assertions</td>
<td>Done; incremental changes till 3/1</td>
</tr>
<tr>
<td>Collector</td>
<td>Manual, robust data structures</td>
<td>Written; to be merged by 1/31</td>
</tr>
<tr>
<td>Watchdog</td>
<td>Manual, assertions</td>
<td>1/18</td>
</tr>
<tr>
<td>Diagnostic</td>
<td>Manual</td>
<td>3/1</td>
</tr>
<tr>
<td>Preloader</td>
<td>Manual, assertions</td>
<td>2/1</td>
</tr>
<tr>
<td>Telemetry</td>
<td>Manual, assertions</td>
<td>2/15</td>
</tr>
<tr>
<td>Ground Program</td>
<td>None</td>
<td>1/31</td>
</tr>
</tbody>
</table>

Project Plan (cont.)

<table>
<thead>
<tr>
<th>Computation Modules</th>
<th>Error Detection</th>
<th>Status / Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT (5 variants)</td>
<td>ABFT, EDI, software duplication and TMR</td>
<td>Under test; 1/18</td>
</tr>
<tr>
<td>Insert Sort</td>
<td>EDI</td>
<td>Under test; 1/25</td>
</tr>
<tr>
<td>LZW (2 variants)</td>
<td>EDI, TMR</td>
<td>Under test; 1/25</td>
</tr>
<tr>
<td>Quick Sort</td>
<td>EDI</td>
<td>Under test; 1/25</td>
</tr>
<tr>
<td>SortTMR</td>
<td>Manual</td>
<td>Under test; 1/25</td>
</tr>
<tr>
<td>Unit targeting tests</td>
<td>N/A</td>
<td>Under test; 1/31</td>
</tr>
<tr>
<td>ALU test</td>
<td>Hook up by 2/15</td>
<td></td>
</tr>
<tr>
<td>VxWorks diagnostics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other Candidates</td>
<td>TBD</td>
<td>TBD</td>
</tr>
</tbody>
</table>

Future Work
- Coding and Debugging
- Testing on Brass Board
- First Real Experiment!
  - Late February, early March
- Detection Techniques
- More Fault Tolerance