Cross-Project IV&V Software Testing
Tools & Technologies

ivv-itc@lists.nasa.gov

Internal Website: http://itc.ivv.nasa.gov

ITC Team Members
Justin Morris    Jeff Joltes
Steven Seeger    Justin McCarty
Brandon Bailey   Dan Nawrocki
Shawn Carroll    Mark Pitts
                David Soto
Agenda

JUNO SUROM TESTING

ITCSB Middleware

GDS Integration
Background
RAD750 Boot Anomaly

Project reported difficulties with boot on flight-like boards
Episodic – 10% of boot attempts failed
Not always the same location
Not at a location where a write was expected
Generally shortly after virtual addressing started

Discovered by developer to be an interaction between uninitialized BAT registers (MMU) and cache controller. (RAD750 erratum)
Corrections successfully applied
Significant changes also made to other FSW components

IV&V team investigated anomaly including board support packages and startup code. ITC was called in to evaluate simulation as part of the investigation.
# Overview

## Capabilities

- Memory page fault injection
- Forced execution of corrupt FSW image
- CRC verification
- Seamless integration with target system
  - Produces no unintended side effects
- Monte-Carlo testing
- Fully customizable

## Environment

- C++ test application
  - Handles interactions with Juno system components
  - Configurable
- Python test control script
  - Allows for consecutive independent test runs to be executed
  - Configurable
- Wind River Simics
  - Hardware on software simulation suite
Test Environment

Python Test Control Script

Juno Environment

Wind River Simics

C++ Test App

Logging Mechanism
Page Fault Injection

- SUROM sequentially tests entire memory space
- SUROM marks page as bad if memory test fails within the page

<table>
<thead>
<tr>
<th>Memory Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
</tr>
<tr>
<td>0x00000001</td>
</tr>
<tr>
<td>0x00000002</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>0x07FFFFFFF9</td>
</tr>
<tr>
<td>0x07FFFFFFA</td>
</tr>
<tr>
<td>0x07FFFFFFB</td>
</tr>
<tr>
<td>0x07FFFFFFC</td>
</tr>
<tr>
<td>0x07FFFFFFD</td>
</tr>
<tr>
<td>0x07FFFFFFE</td>
</tr>
<tr>
<td>0x07FFFFFFF</td>
</tr>
</tbody>
</table>

**Memory Test**
1. Write value
2. Read value
3. Compare
• SUROM sequentially tests entire memory space  
• SUROM marks page as bad if memory test fails within the page

<table>
<thead>
<tr>
<th>Memory Test</th>
<th>Memory Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Write value</td>
<td>0x00000000</td>
</tr>
<tr>
<td>2. Read value</td>
<td>0x00000001</td>
</tr>
<tr>
<td>3. Compare</td>
<td>0x00000002</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x07FFFFFFF9</td>
</tr>
<tr>
<td>0x07FFFFFFF8</td>
</tr>
<tr>
<td>0x07FFFFFFF7</td>
</tr>
<tr>
<td>0x07FFFFFFF6</td>
</tr>
<tr>
<td>0x07FFFFFFF5</td>
</tr>
</tbody>
</table>

Test App Interaction  
1. Capture written value  
2. Apply data transform  
3. Store values for logging
FSW Image Corruption

- SUROM validates FSW image correctness via CRC
- Simple image corruption
  - Verifies CRC evaluation
- CRC correction
  - Forcibly loads corrupted FSW image
System Inputs/Outputs

- Python Control Options
- Configuration File
- Juno Test Environment
- Log File
# System Inputs

## Configuration File
- Memory modification
  - Location (range, all)
  - Number of faults
- FSW image modification
  - Location (header, body, all, range)
  - Number of faults
- Data transform
  - Mask (and, or, xor, not)
  - Random
- Fix CRC
- Success logging

## Python Control Options
- Number of test runs
- Specify configuration file
- Specify log file
Log File

- Captures (for each test run)
  - Run ID
  - Date/time
  - Memory modifications
  - FSW image modifications
  - Image CRC values
  - Noteworthy register and memory contents
  - Successful load indication
- System state components logged as requested by Juno IV&V team
Benefits of JUNO SUROM Testing

• Configurable test environment
  – Memory page fault injection
  – Forced execution of corrupt FSW image
  – CRC verification
  – Seamless integration with target system
  – Monte-Carlo testing
  – Test result logging

• Repeatable

• Applicability to other NASA IV&V supported projects
ITCSB Middleware
Steven Seeger

JUNO SUROM TESTING
ITCSB Middleware
GDS Integration

NASA IV&V Facility Independent Test Capability
Background

• ITCSB (ITC Synchronous Bus)
  – Distributed simulation components
    • Communication is necessary
    • Separate notions of time
    • Synchronization points are needed to keep system “in sync”
  – Focus on new technology rather than interconnectivity
    • Ease of development
    • Ease of integration

• Rationale of Change
  – Last year’s shared memory peer-to-peer approach
    • Deadlocks on crash
    • Misbehaving nodes tie up the system
ITCSB Middleware Overview

- ITCSB Implementation
- Cross platform C++ implementation
- Transport-agnostic
- Shared library for nodes
- Server application
- Basis for 1553, SpaceWire, time broadcast
ITCSB_1553 GO-SIM Application

Remote Terminal
(C++ exe)

ITCSB_1553 API

SPW App (GPMCW, GDS, etc.)
(C++ exe)

ITCSB_SPW API

ITCSB

Simics
- RAD750 (WR)
- 1553 PCI (WR)
- Spacewire (EFS)

C_1553_BC

C_SPW

ITCSB_SPW API

ITCSB_1553 API
Features

• Unlimited busses and nodes
• Manages all ITCSB logic
• Built in TCP/IP
  – Transport is decoupled from logic for easy addition of other transports
• Concurrent operation of all busses

Features

Node

1. Request send token
2. Give send token
3. Send data to server
4. Send data to node(s)
5. Acknowledge transfer
6. Notify sender
7. Yield send token

Server

2. Give send token
4. Send data to node(s)
5. Acknowledge transfer
6. Notify sender
7. Yield send token
ITCSB Libraries

ITCSB Client Library

- Cross-platform C++ library
- Custom transport
- Nodes referred to by name
- Readers and Writers
- Send callback does not return until all nodes received or timed out
- Data is sent as array of bytes opaque to the middleware
- Nodes can be forcefully removed
## ITCSB Libraries

<table>
<thead>
<tr>
<th>ITCSB_1553</th>
<th>ITCSB_SPW</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Exposes own callbacks</td>
<td>• ITCSB_SPW Router application handles routing logic</td>
</tr>
<tr>
<td>• Bus owns multiple RTs</td>
<td>– XML config file defines topology</td>
</tr>
<tr>
<td>• RTs own multiple subaddresses</td>
<td>– Multiple networks and configs</td>
</tr>
<tr>
<td>– TX or RX</td>
<td>– Configuration registers</td>
</tr>
<tr>
<td>• Register nodes by RT number</td>
<td>– Uses two ITCSB busses</td>
</tr>
<tr>
<td></td>
<td>• Exposes own callbacks</td>
</tr>
<tr>
<td></td>
<td>• All communication done with router app</td>
</tr>
<tr>
<td></td>
<td>• Node applications must know topology</td>
</tr>
</tbody>
</table>
Benefits of ITCSB

• Makes development of new simulation technology easier
• Allows for concurrent development of new simulation technologies
• Allows fault injection and logging of traffic
• No license requirements
• Potential to be utilized by development teams and IV&V teams
GDS Integration
Dan Nawrocki
What is GDS?

• GDS = Goddard Dynamic Simulator

• Used by SDO, LRO, MMS, and GPM

• Physics Simulator
  – Ephemeris, Equations of Motion, Gravity, Magnetic Fields, Heating/Cooling, GPS

• MIL-STD-1553 Remote Terminal Simulator
  – Mission Agnostic
    • Honeywell IRU, Goddard RWA, Selex Galileo Star Tracker, GSFC GPS Navigator
  – Mission Dependent (GPM)
    • ATB, IIC, MAC, MCE, PDE, PSE, Northrop Grumman IRU
Why Use GDS?

- GPM main CPU handles both Science and Spacecraft systems
- IV&V team would like to verify functionality of Spacecraft health software (sun-pointing, star tracking, attitude control, thruster control, etc)
- IV&V of numerical algorithms, such as the “C” in GN&C, is difficult/expensive using manual code examination or static code analysis
<table>
<thead>
<tr>
<th>Challenge</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDS is Ada, GOSIM system is C/C++</td>
<td>Ada wrappers written around C functions in shared library calling C++ library methods</td>
</tr>
<tr>
<td>GDS requires Windows XP, GOSIM requires Linux or Windows 7</td>
<td>Must support distributed simulations</td>
</tr>
<tr>
<td>Three new versions of GDS code have been released between GPM build 3.0 and 4.0</td>
<td>Develop new Ada packages conforming to existing interfaces to add functionality. Minimize GDS codebase changes to reduce risk and increase portability between GDS versions</td>
</tr>
<tr>
<td>Target EM (Engineering Model) system for higher fidelity simulation of spacecraft software system</td>
<td>Just do it</td>
</tr>
</tbody>
</table>
GDS Main Execution Thread

- Wait for Cycle_Start Interrupt [FPGA]
- Read inputs for next frame; write outputs from previous frame [1553/SpW Card]
- Wait for Communicate_1_Start Interrupt [FPGA]
- Execute Models [GDS Internal]

NASA IV&V Facility Independent Test Capability
Time

GDS Main Thread
  gds-clocks
    -- Wait_Tick
  gds-cards-itr_time_broadcast
    -- Wait_Tick
  gds_time_broadcast_receiver.cpp
    -- GdsTimeBroadcastReceiverWaitTick
    libitcsb_time_broadcast

Waits for time tick from Simics instead of local clock

Ada wrapper

NASA IV&V Facility Independent Test Capability
1553

GDS Main Thread

gds-hardware-bus_1553-wrapper
  -- Read_Inputs
  -- Write_Outputs

Ada wrapper

gds-mil_std_1553-itcsb_1553
  -- Read_Message
  -- Write_Message

gds_1553.cpp
  -- Gds1553ReadMessage
  -- Gds1553WriteMessage

libitcsb_1553

Buffers messages similar to SBS 1553 card
Timing – GDS Too Fast
Timing – GDS Slow

Legend
- Running Thread
- Blocked Thread
- Blocked for Synchronization
- Signal
- Critical Path

NASA IV&V Facility Independent Test Capability
• IV&V has GPM simulator similar in functionality and fidelity to GSFC labs

• New IV&V test capabilities for GPM:
  – GN&C: does the spacecraft maintain the correct attitude?
  – Power: does the spacecraft turn component X off after Y seconds?
  – Safety: does the spacecraft enter safe-hold mode after encountering out-of-bounds thermal or voltage readings?

• Will be integrated with graphical tools to provide high-quality visualizations
Closing Remarks

- Cross-Project Software Testing Tools & Technologies
  - JUNO SUROM Testing
  - ITCSB Middleware
  - GDS Integration
QUESTIONS?