Design and Implementation of Triple Modular Redundant (TMR) System on Linux-Based On-Board Computer for CubeSat

Dr. Emir Husni  Angga Putra  Nazmi Febrian

School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Need and Objective

The Largest Archipelago
Need and Objective

Major Volcanoes of Indonesia
(with eruptions since 1900 A.D.)

Ring of Fire

Topinka, USGS/CVO, 2001; base map modified from: CIA map, 1997; volcanoes from: Simkin & Siebert, 1994
Problem Statement:
Non-Destructive & Destructive Event in Space Environment

Non-destructive events
• Single Event Transient
• Single Bit Upset
• Multi Bit Upset

Destructive events
• Single Event Latch-up
• Single Event Gate Rupture (SEGR)
• Single Event Burnout
Non-Destructive Event
Solution: Error Correction Coding on TMR system communication inter on-board computer (OBC)

Data communication between OBC on TMR system
- Using CAN-over-IP as packet data protocol
- CAN frame implemented on UDP data

*CAN = Controller Area Network
Non-Destructive Event
Solution: Error Correction Coding on TMR system
communication inter on-board computer

Error Correction Coding
1. Error detection
2. Error correction

1. Error detection using CRC (Cyclic Redundancy Check)

```
  1010
1011\sqrt{1001000}
   1011
   0100
   0000
  
  1000
  1011
   0110
   0000
  
  110
```

Data | CRC
---   | ---
1001  | 110

Remainder: 3 bits

2. Error correction
Using ARQ (Automatic Repeat Request)
Non-Destructive Event Error Correction Coding Implementation

Communication inter OBC
- Linux socket programming
- Using C language

**SENDER SIDE**
Non-Destructive Event Error Correction Coding implementation

Communication inter OBC
- Linux socket programming
- Using C language

RECEIVER SIDE
Destructive Event

Single-ion induced dielectric failure
MOSFETs, Capacitors, FG Devices

I-V following biased irradiation of
3.3 nm SiO₂ capacitors

Lum, et al., IEEE TNS 51 3263 (2004)
Massengill, et al., IEEE TNS 48 1904 (2001)
Destructive Event

Solution: Fail-over TMR system on on-board computer

OBC-1 (communicate with GS)

OBC-2 (sensor processing)

OBC-3 (sensor processing, back-up mode)

Sensor Module

Diagram showing the connection between the Sensor Module, OBC-1, OBC-2, and OBC-3.
Destructive Event
Solution: Fail-over TMR system on on-board computer

- Sensor Module
- OBC-1 (communicate with GS)
- OBC-2 (die)
- OBC-3 (sensor processing)
Destructive Event
Solution: Fail-over TMR system on on-board computer

- OBC-1 (communicate with GS)
- OBC-2 (sensor processing)
- OBC-3 (die)
Destructive Event

Solution: Fail-over TMR system on on-board computer

Sensor Module

OBC-1
(sensor processing, communicate with GS)

OBC-2
(die)

OBC-3
(die)
Destructive Event
Solution: Fail-over TMR system on on-board computer
Destructive Event
Solution: Fail-over TMR system on on-board computer

Sensor Module

OBC-1
(die)

OBC-2
(die)

OBC-3
(sensor processing, communicate with GS)

Computer
Destructive Event
Solution: Fail-over TMR system on on-board computer

- Sensor Module
- OBC-2 (sensor processing communicate with GS)
- OBC-3 (die)
- OBC-1 (die)
Destructive Event
Fail-over TMR system implementation on on-board computer

TMR System
• Linux command line interface
• Using bash shell script

Hardware (single board computer)
• Intel Galileo Generation 1
Systems Implementation

- Each OBC connected through passive switch
- Communicate with GS through ION-DTN protocol

Specification
- Data corrupt resistance up to 10%
- Error control coding
- Hardware triple modular redundancy
- Remote software update
## Experiment Results

### Error Correction Coding

<table>
<thead>
<tr>
<th>Corrupt</th>
<th>Transmission Time</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 %</td>
<td>1:04:04.72</td>
<td>76,894</td>
</tr>
<tr>
<td>2 %</td>
<td>1:05:51.924</td>
<td>79,038</td>
</tr>
<tr>
<td>4 %</td>
<td>1:07:09.74</td>
<td>80,595</td>
</tr>
<tr>
<td>6 %</td>
<td>1:08:54.15</td>
<td>82,683</td>
</tr>
<tr>
<td>8 %</td>
<td>1:09:44.11</td>
<td>83,682</td>
</tr>
<tr>
<td>10 %</td>
<td>1:10:23.6</td>
<td>84,472</td>
</tr>
</tbody>
</table>

This experiment is done by repeating CRC test **50 times** for each corrupt percentage to get the average time.
Experiment Results
Error Correction Coding

Transmission Time

Corrupt (%)

Time (s)
Experiment Results
Triple Modular Redundant System

Case 1: Un-connected from GS

<table>
<thead>
<tr>
<th>OBC-1</th>
<th>OBC-2</th>
<th>OBC-3</th>
<th>Request Hit</th>
<th>Request Miss</th>
<th>Total Packet Acquired (byte)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
<td>FAIL</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>ACTIVE</td>
<td>FAIL</td>
<td>ACTIVE</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>ACTIVE</td>
<td>FAIL</td>
<td>FAIL</td>
<td>no request</td>
<td>no request</td>
</tr>
<tr>
<td>5</td>
<td>FAIL</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
<td>no request</td>
<td>no request</td>
</tr>
<tr>
<td>6</td>
<td>FAIL</td>
<td>ACTIVE</td>
<td>FAIL</td>
<td>no request</td>
<td>no request</td>
</tr>
<tr>
<td>7</td>
<td>FAIL</td>
<td>FAIL</td>
<td>ACTIVE</td>
<td>no request</td>
<td>no request</td>
</tr>
</tbody>
</table>

Assumption: All scenario happen in one rotation
Testing scenario: 90 minutes/7 scenario ≈ 13 minutes/scenario

RESULTS
Maximum packet data acquired ≈ 6.3 MB
Maximum packet data acquired from scenario #1

LEO Orbit

Rotation Period
62’ sun phase + 38’ eclipse = 100 minutes

GS-connected duration estimation
≈ 10 minutes

GS-unconnected duration
100 minutes – 10 minutes ≈ 90 minutes
Experiment Results
Triple Modular Redundant System

Case 2: Connected from GS

<table>
<thead>
<tr>
<th></th>
<th>OBC-1</th>
<th>OBC-2</th>
<th>OBC-3</th>
<th>First packet Sent Time (s)</th>
<th>Update Success?</th>
<th>Packet received in GS during process (byte)</th>
<th>Size of packet acquired from GS-unconnected period (First packet)</th>
<th>Total packet delivered (byte)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
<td>15.03</td>
<td>yes</td>
<td>985,924</td>
<td>44.1 MB</td>
<td>45,085,924</td>
</tr>
<tr>
<td>2</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
<td>FAIL</td>
<td>14.57</td>
<td>yes</td>
<td>985,952</td>
<td>44.1 MB</td>
<td>45,085,952</td>
</tr>
<tr>
<td>3</td>
<td>ACTIVE</td>
<td>FAIL</td>
<td>ACTIVE</td>
<td>14.81</td>
<td>yes</td>
<td>352,818</td>
<td>44.1 MB</td>
<td>44,452,818</td>
</tr>
<tr>
<td>4</td>
<td>ACTIVE</td>
<td>FAIL</td>
<td>FAIL</td>
<td>14.21</td>
<td>yes</td>
<td>981,536</td>
<td>44.1 MB</td>
<td>45,081,536</td>
</tr>
<tr>
<td>5</td>
<td>FAIL</td>
<td>ACTIVE</td>
<td>ACTIVE</td>
<td>15.03</td>
<td>yes</td>
<td>1,542,597</td>
<td>44.1 MB</td>
<td>45,642,597</td>
</tr>
<tr>
<td>6</td>
<td>FAIL</td>
<td>ACTIVE</td>
<td>FAIL</td>
<td>15.45</td>
<td>yes</td>
<td>1,542,557</td>
<td>44.1 MB</td>
<td>45,642,557</td>
</tr>
<tr>
<td>7</td>
<td>FAIL</td>
<td>FAIL</td>
<td>ACTIVE</td>
<td>15.57</td>
<td>yes</td>
<td>1,191,887</td>
<td>44.1 MB</td>
<td>45,291,887</td>
</tr>
</tbody>
</table>

Testing scenario: 10 minutes/scenario
Size of packet accumulated on GS-unconnected period (first packet data to be delivered to GS) are calculated from the maximum packet data acquired from scenario #1 on Case 1 in 90 minutes duration.

RESULTS
Software update successfully done remotely from GS
All packet data acquired from GS-unconnected period successfully delivered to GS
During GS-connected period, the system is able to acquiring and delivering data to GS.
Conclusion

- By using TMR system, probability of ‘total system fail’ was significantly reduced. Sensor data is safely kept and delivered to GS, except when all OBC fail simultaneously.

- Error correction coding reduced packet data retransmission, thus increased data transmission speed and amount of data delivered.

- By using Linux operating system, internet based communication can be used as protocol for data communication inter OBC.