Predicting and Measuring System Error Rates for Designs Incorporating Upset Mitigation based on Triple Modular Redundancy

Gary M. Swift and Larry D. Edmonds
Jet Propulsion Laboratory / California Institute of Technology

NASA JPL

The work described in this presentation was conducted at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

Introduction and Overview

Upset Mitigation Basics

- Upsets are NOT errors
- Upset-error relation for memories (Hamming codes and the like)

TMR Basics

New "fitting" equation for TMR

Examples of application to real data

Use JPL format

Basics of Upset Mitigation

Redundancy -

Extra information (bits) prevents all upsets from yielding system errors.

Scrubbing required –

Accumulation of errors rapidly kills mitigation effectiveness.

Effective –

Most spacecraft now fly large arrays of upset-soft memories with few or no errors.

Typically, uncorrectable errors are detectable.

Basics of Upset Mitigation - cont'd

Common sense says -

At some point, upsets will occur too rapidly and the mitigation will be "overwhelmed."

In fact, Edmonds approx. equation says –

There's not really a "cliff."

The relationships are known; the error rate:

(1) increases with the square of upset rate
(2) decreases linearly with faster scrub rates
(3) is directly proportional to EDAC word size†

† EDAC word size = data bits + check bits ; EDAC=error detection and correction

Basics of Upset Mitigation - Examples

32 data bits + 7 check bits -

- Cassini Solid State Recorders with 2+ Gb DRAM array is working well, in spite of architecture "flaw."

128 data bits + 9 check bits –

- This hidden EDAC word inside IBM Luna-C 16Mb DRAMs used on RAD6000 boards on many missions requires external accesses to prevent accumulation of upsets.

64 data bits + 16 check bits –

- A specially design cyclical parity scheme on the RAD750 board corrects up to 4 upsets, if confined to a nibble, allowing correct operation with a bad DRAM chip.

TMR Basics

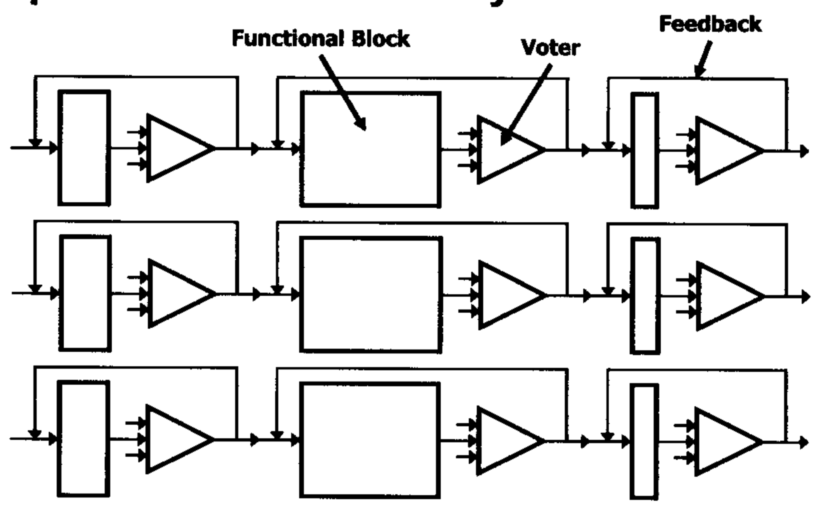
TMR = triple-module redundancy

- Three independent "legs" or domains performing identical functions
- Voters are inserted – typically at feedback points
- Voters are triplicated also
  – they are *not* a single point of failure

Error-free operation with any *single* upset

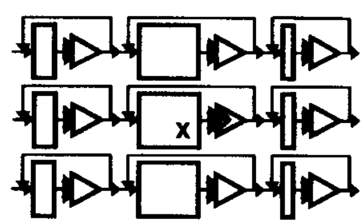
- Two upsets might cause system failure
- Scrubbing is again required to reduce the chance of co-resident upsets.

Triple-module redundancy

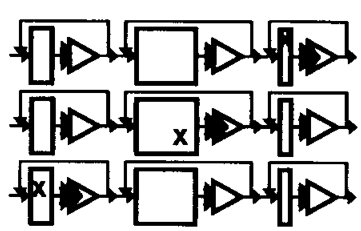


Feedback from the voters corrects state errors inside blocks

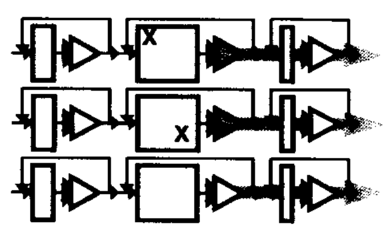
TMR stops error propagation



Single upsets cannot cause errors

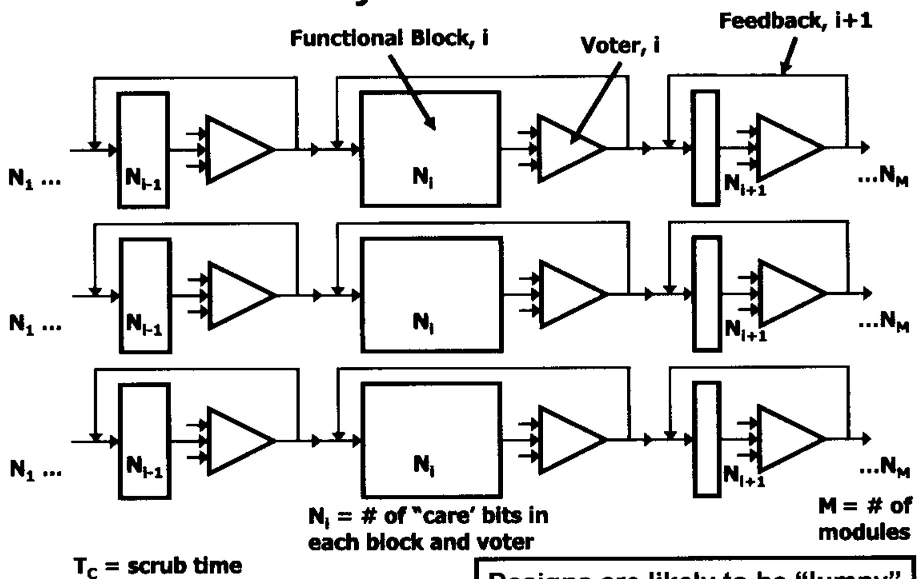


Even multiple upsets *may* not cause errors



Error propagation requires upsets in two parallel modules.

Model of TMR System



Designs are likely to be "lumpy"

Edmonds TMR Equation – small r approx.

$$R \approx 3M\,T_C\,(\mathcal{M}_2\,r)^2$$

- R: System Error Rate
- M: Total Modules
- T_C: Scrub Time
- $\mathcal{M}_2$: "Fitting" Parameter
- r: Underlying Upset Rate

Edmonds TMR Equation – small r approx.

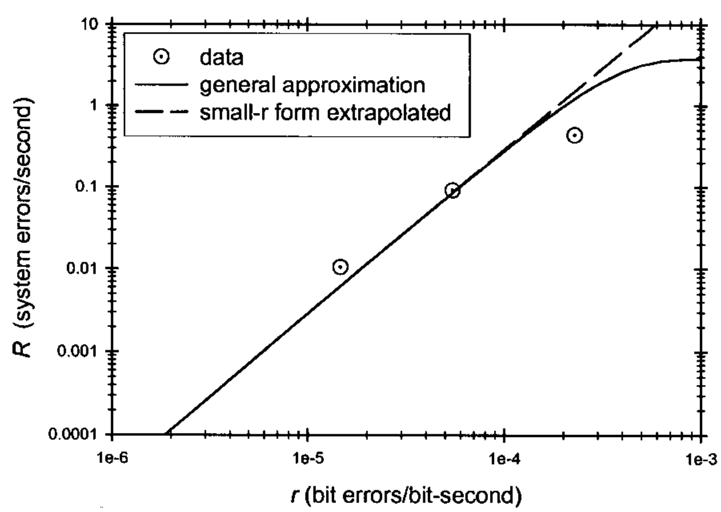
$$R \approx 3M\,T_C\,(\mathcal{M}_2\, r)^2$$

R: System Error Rate
M: Total Modules
T_C: Scrub Time
r: Underlying Upset Rate

Parameter is really the second moment of the distribution of N's. This is a "cousin" of the standard deviation.

$$\mathcal{M}_2 \equiv \left[\frac{1}{M}\sum_{i=1}^{M} N_i^{\,2}\right]^{1/2}$$

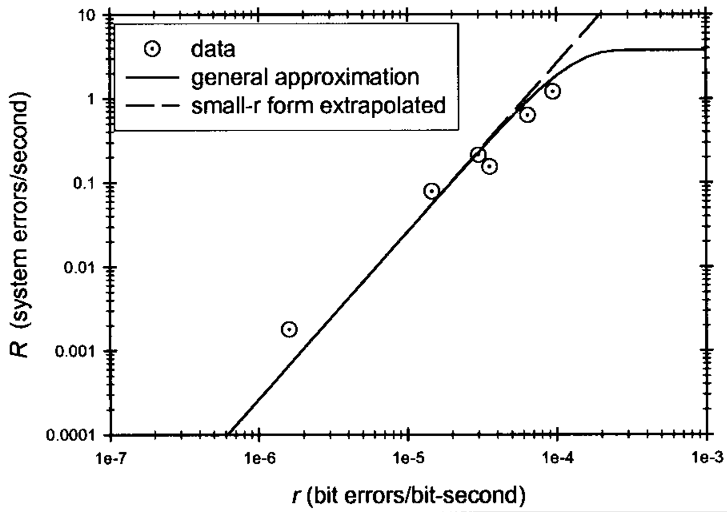
Example Application - Multipliers



Given parameters: T=0.266 s, M=900

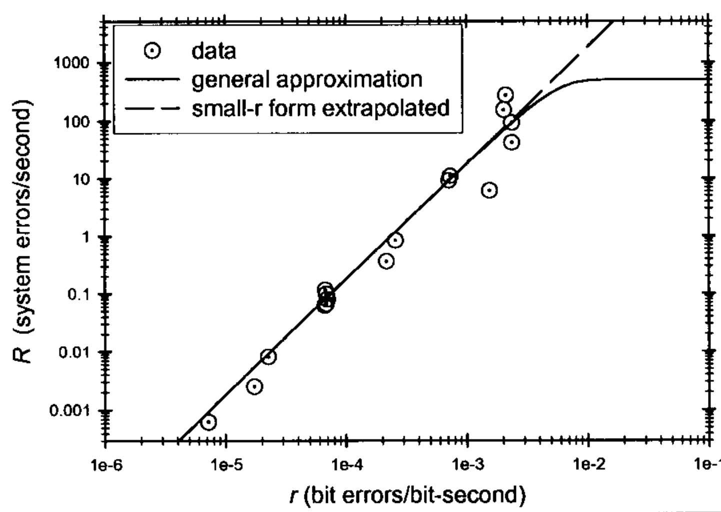
Fit parameter: M2=M3=M4= 200

Example Application - Counter Design



Given parameters: T=0.266 s, M=8224 **Fit parameter: M2=M3=M4= 200**

Example Application - BRAM Scrubber



Given parameters: T=2 ms, M=48000 **Fit parameter: M2=M3=M4= 250**

In Review ...

New Edmonds Equation for TMR is

- General (for TMR-ed systems)
- Powerful
  - × Works over many orders-of-magnitude
- Based on moments which are
  - × Statistically meaningful
  - × Of rapidly diminishing importance so only one (or two) adjustable parameters are enough
  - × Calculable, in theory anyway; in practice, probably not.
- Useful
  - × In predicting system error rates in space
  - × In designing appropriate in-beam testing