Implementing Redundancy

Issues:
- Placement of *error containment filter* in space/time.
- Synchronization of redundant processes
- Redundant analog/asynchronous signals: not totally identical in value/timing
- Concurrent failure detection
- Policy: handling a failed module
- Avoiding correlated errors
Voter Placement

Voter can be cause of *single-source failures*

Non-redundant

TMR-version
A TMR System: Wakerly

- Errors in data-flow:
  - Info from memory filtered
  - Bad info in registers will be eventually replaced
- Errors in control (instruction sequencing)
  - Periodic software reset request (SRR)
  - At reset, registers & memory initialized

Why is voter after memory?

Containment in space

Containment in time

One-Third of the system

Fault Tolerant Computing
©Y.K. Malaiya
Synchronization Issues

- A TMR system may have
  - Clock-level synchronization
  - Non-synchronous implementation
    - Voting in software
    - Staggered job segments (Kameyama & Higuchi) to avoid correlation

```
\begin{array}{cccccc}
  n & n+1 & n+2 & n+3 & n+2 & n+3 & n+4 \\
  n-1 & n & n+1 & n+2 & - & n+2 & n+3 \\
  n-2 & n-1 & n & n+1 & - & - & n+2 \\
\end{array}
```

OK  OK  x  Based on voted result (context)
TMR Asynchronous/Analog Inputs

• What if an input is asynchronous (like interrupt request)?

  \[
  \text{Int1} \quad \text{Int2} \quad \text{Int3} \quad \text{voter} \quad \text{Voter waits until all signals have arrived, or an error is suspected}
  \]

• An input (from redundant sensors) is analog?
  • Allow a margin within which they are considered equal.

Spares:
- Unpowered: lower failure rates
- Powered: no switching transients
TMR Synchronization: Info, Clock

Clock choices:

- **Single clock**: common source failure; skew due to uneven load
- **Independent clocks synchronized initially**: synchronization not guaranteed over a long period
- **Separate clocks interlocked by voting**:

![Clock Diagram]

Info synchronization: scrubbing persisting errors

- Copy from clean module
- Automatic periodic initialization of all modules
- Wait until bad info is eventually replaced by good (may need to save SP etc periodically)
On-line Testing

- On-line fault detection:
- Periodic scheduled testing
- Concurrent testing:
  - a. self-testing logic
  - b. duplex configuration

Contain some redundancy in space or time
Tests for some invariant property

- Only selected nodes need to be compared.
- Active and Shadow need to stay synchronized
References

