Implementing Redundancy

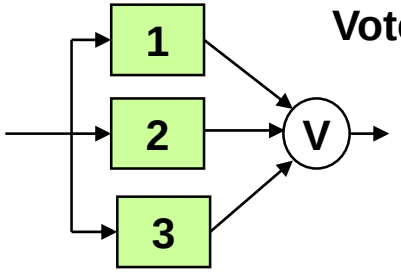
Yashwant K Malaiya

Implementing Redundancy

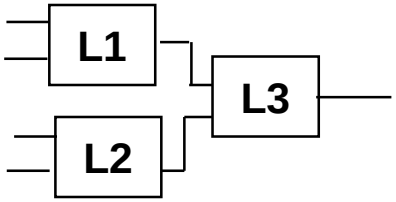
Issues:

- **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**

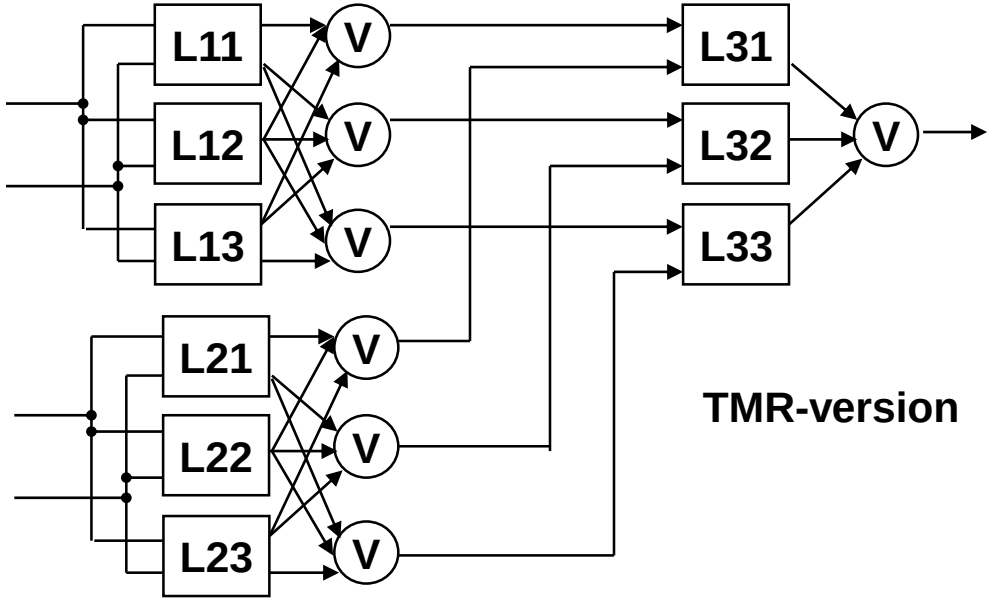
Voter Placement



Voter can be cause of *single-source failures*

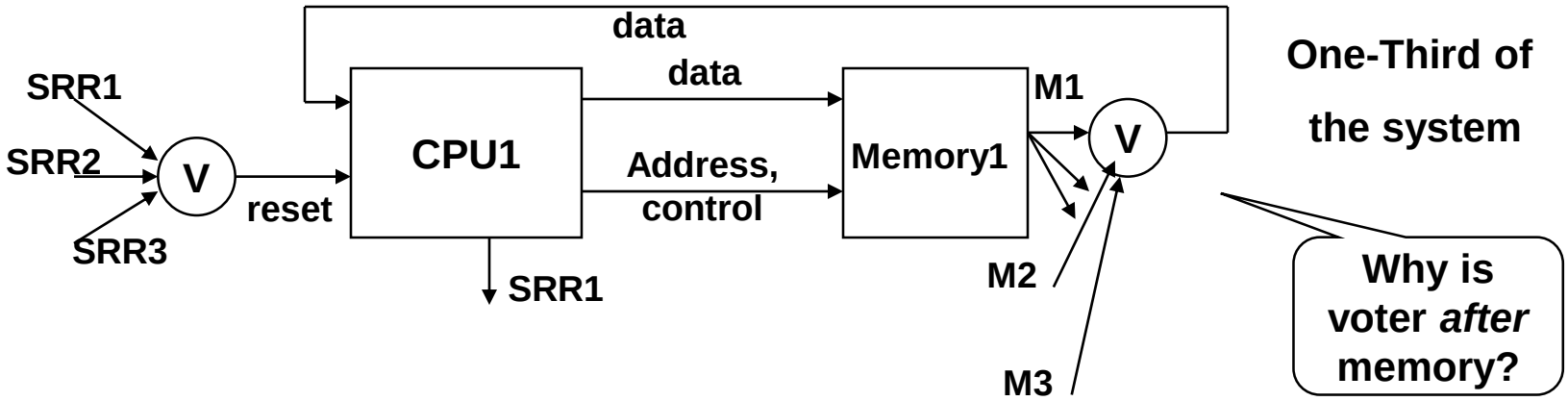


Non-redundant



TMR-version

A TMR System: Wakerly



- **Errors in data-flow:**
  - **Info from memory filtered**
  - **Bad info in registers will be eventually replaced**

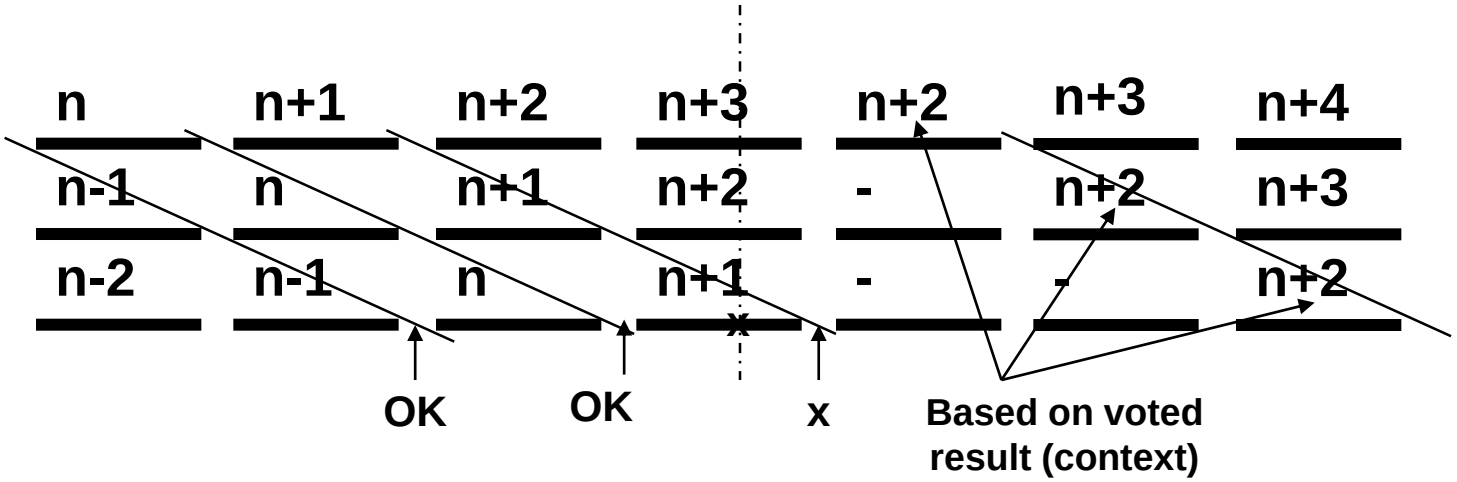
Containment in space

- **Errors in control (instruction sequencing)**
  - **Periodic software reset request (SRR)**
  - **At reset, registers & memory initialized**

Containment in time

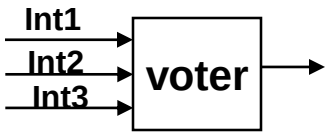
Synchronization Issues

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



TMR Asynchronous/Analog Inputs

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



Voter waits until all signals have arrived, or an error is suspected

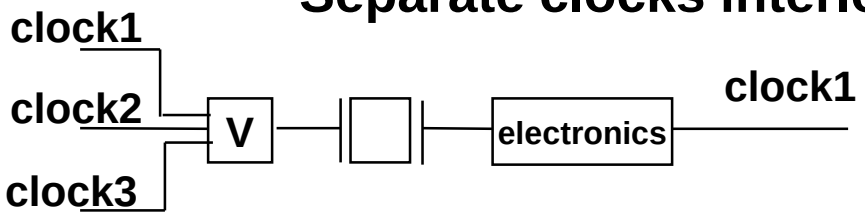
- **An input (from redundant sensors) is analog?**
  - **Allow a margin within which they are consider equal.**

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:**

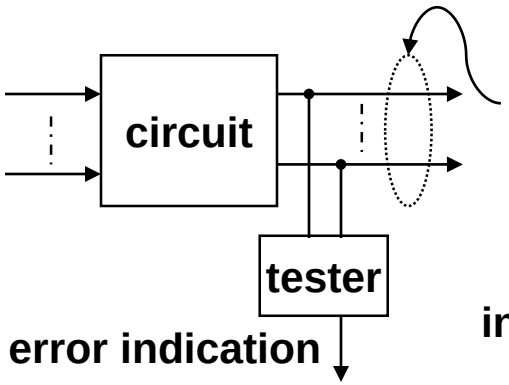


Info synchronization: scrubbing persisting errors

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

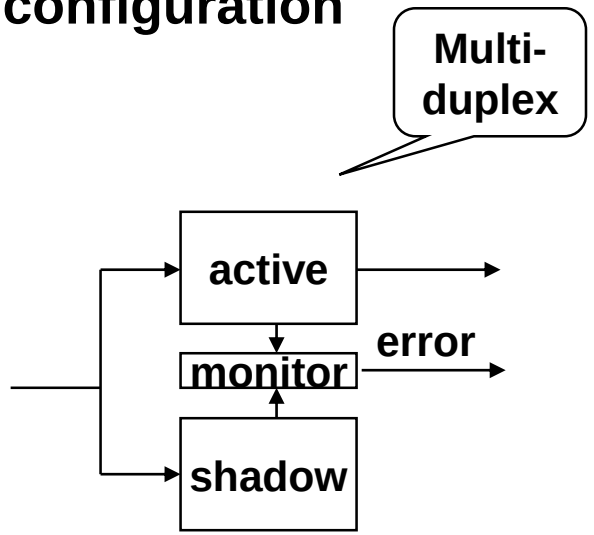
On-line Testing

- **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

- J.F. Wakerly, “Microcomputer Reliability Improvement Using Triple-Modular Redundancy,” Proceedings of the IEEE 64 (6), June 1976, pp.889-895
- M. Kameyama and T. Higuchi, "Design of Dependent-Failure-Tolerant Microcomputer System Using Triple-Modular Redundancy," in IEEE Journal of Solid-State Circuits, vol. 15, no. 1, pp. 138-142, Feb 1980.
- Wirthlin, M.J., Keller, A.M., McCloskey, C., Ridd, P., Lee, D. and Draper, J., 2016, February. SEU Mitigation and Validation of the LEON3 Soft Processor Using Triple Modular Redundancy for Space Processing. In Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2016, pp. 205-214.