Fault-Tolerant Computer System Design
ECE 60872/CS 590

Topic 4: Reliable Hardware Design

Saurabh Bagchi
ECE/CS
Purdue University

Outline

- Basic approaches to hardware redundancy
- Series/parallel, non-series parallel structures
- Voting
- Hardware voter example

Basic Forms of Hardware Redundancy

- Error masking
  - relies on voting to mask the occurrence of errors
  - can operate without need for error detection or system reconfiguration
  - triple modular redundancy (TMR)
  - N-modular redundancy (NMR),
- Dynamic redundancy
  - achieves fault tolerance by error detection, error location, and error recovery
  - standby sparing
    - one module is operational and one or more modules serve as standbys or spares
- Hybrid hardware redundancy
  - Fault masking used to prevent the system from producing erroneous results
  - Fault detection, location, and recovery used to reconfigure the system in the event of an error.
  - N-modular redundancy with spares.

Hardware Masking Redundancy

- Masking employs redundancy to isolate or correct faults before they reach the output
- Logical interconnection of the modules is fixed, hence called “static redundancy”
- When masking redundancy is exhausted, any further fault will cause an error at the output
- Gives no indication of deteriorating hardware state until enough faults have accumulated to cause an error

Dynamic Redundancy

- Involves reconfiguration of system in response to faults
- Reconfiguration often involves disconnecting damaged units from system and doing on-line or off-line repair
- Reconfiguration triggered by internal detection of faults or detection of errors in output
- Success of reconfiguration depends on coverage of detection, diagnosis and confinement, expressed as a combined coverage measure
- Some techniques for detection
  - Self-checking
  - Diagnostic program
  - Watch-dog timer
  - Run sample workload

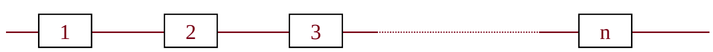
Evaluation Criteria

- A method of evaluation is required in order to compare the redundancy techniques and make subsequent design tradeoffs
- Modeling techniques are a vital means for obtaining reasonable predictions of system reliability and availability
  - Combinatorial: series/parallel, M-of-N, nonseries/nonparallel
  - Markov: time invariant, discrete time, continuous time, hybrid
  - Queuing
- Using these techniques probabilistic models of systems can be created and used to evaluate system reliability and/or availability

Combinatorial Modeling

- System is divided into non-overlapping modules
- Each module is assigned either a probability of working, P_i, or a probability as function of time, $R_i(t)$
- The goal is to derive the probability, P_{sys}, or function $R_{sys}(t)$ of correct system operation
- Assumptions:
  - module failures are independent
  - once a module has failed, it is always assumed to yield incorrect results
  - system is considered failed if it does not satisfy minimal set of functioning modules

Series Systems



$$R_{sys} = R_1 \bullet R_2 \bullet R_3 \bullet \ldots \bullet R_n$$

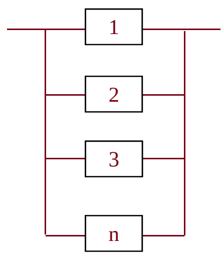
- For exponential failure rate of each component

$$R_{series}(t) = e^{-\sum_{i=1}^{n} \lambda_i t} = e^{-\lambda_{system} t}$$

  - Effect is summation of failure rates of components

$$\lambda_{system} = \sum_{i=1}^{n} \lambda_i$$

Parallel Systems



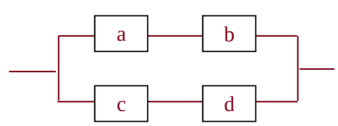
- Assume system with spares
- As soon as fault occurs a faulty component is replaced by a spare
- Only one component needs to survive for the system to operate correctly
- Reliability of the parallel system

$$R_{parallel}(t) = 1.0 - \prod_{i=1}^{n} (1.0 - R_i(t))$$

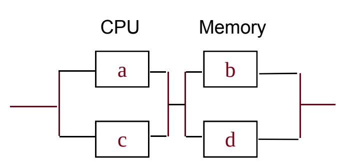
Series-Parallel Systems

- Consider combinations of series and parallel systems
- Example, two CPUs connected to two memories in different ways

$R_{sys} = 1- (1-R_a R_b) (1-R_c R_d)$

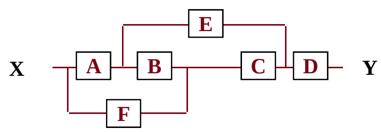


$R_{sys} = (1-(1-R_a)(1-R_c)) (1-(1-R_b)(1- R_d))$



Non-Series-Parallel-Systems

- Often a “success” diagram is used to represent the operational modes of the system

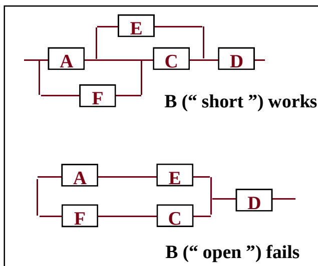


Each path from X to Y represents a configuration that leaves the system successfully operational

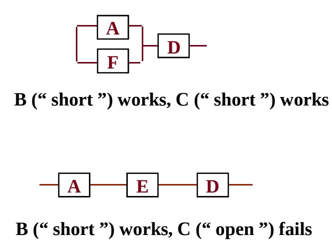
$R_{sys} = R_m$ P (system works|m works) + $(1-R_m)$ P (system works|m fails)
where the notation P(s|m) denotes the conditional probability “s given m has occurred”

- Reliability of the system can be derived by expanding around a single module *m*

Non-Series-Parallel-Systems (cont.)



Reduced model with B replaced



Reduction with B and C replaced

$R_{sys} = R_B$ P(system works|B works)
$+ (1 - R_B)\ \{R_D[1 - (1 - R_A R_E)(1 - R_F R_C)]\}$

P(system works|B works) =
$R_C\{R_D[1 - (1 - R_A)(1 - R_F)]\}$
$+ (1 - R_C)(R_A R_D R_E)$

Letting all R’s = R_m yields $\mathbf{R_{sys} = R^6_m - 3R^5_m + R^4_m + 2R^3_m}$

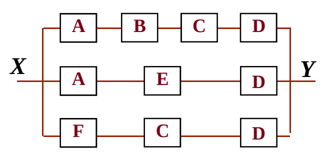
Non-Series-Parallel-Systems (cont.)

- For complex success diagrams, an upper-limit approximation on R_{sys} can be used

$$R_{sys} \le 1 - \prod \left(1 - R_{path\ i}\right)$$

R_{path} is the serial reliability of path i

- An upper bound on system reliability is:



Reliability block diagram (RBD) of a system

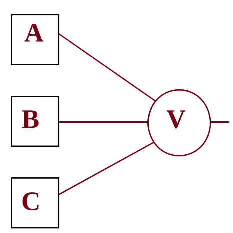
The above equation is an upper bound because the paths are not independent.
That is, the failure of a single module affects more than one path.

$$R_{sys} \le 1 - (1 - R_A R_B R_C R_D)(1 - R_A R_E R_D)(1 - R_F R_C R_D)$$

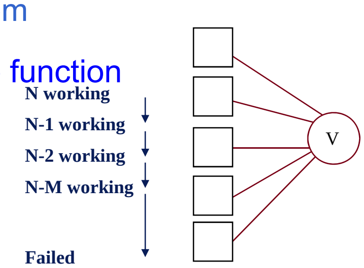
$$R_{sys} \le 2R_m^3 + R_m^4 - R_m^6 - 2R_m^7 + R_m^{10}$$

M-out-of-N Systems

- Static or masking redundancy



- For general *M-out-of-N* system
- Out of N modules, need M to function



$$R_{MN} = \sum_{i=0}^{N-M} \binom{N}{i} R_m^{N-i} (1 - R_m)^i$$

Cascading TMR Systems

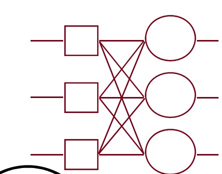
- Consider n stages of original system
- Each stage replaced by TMR with Voter

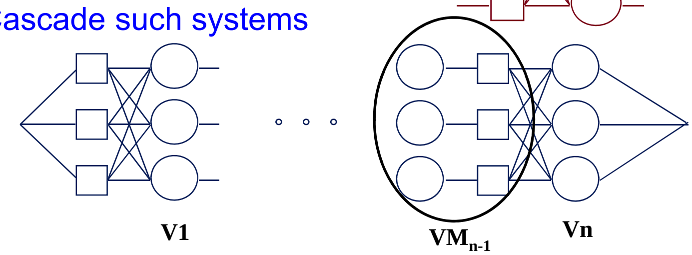
Reliability of the system

$$R_{cascade} = \left(R_V \left(R_m^3 + \binom{3}{2} R_m^2 (1 - R_m) \right) \right)^n$$

TMR with 3 Voters

- Remove single point of failure
- Use TMR with 3 voters
- Cascade such systems





Consider (n-1) voter-module combinations in the middle

$R_{n-1\ stages}$ = (0 or 1 voter-module combinations have failed)$^{n-1}$

= $(3R_{vm}^2 - 2R_{vm}^3)^{n-1}$, where $R_{vm} = R_v.R_m$

$R_{sys} = (3R_m^2 - 2R_m^3) \cdot R_{n-1\ stages} \cdot (3R_v^2 - 2R_v^3)$

Coding based Detection and Correction

- Parity
  - Can detect an odd number of bits in error
  - Cannot correct any error
- Single error correction, Double error detection code
  - 4 data bits
  - 4 parity bits
- Theorems
  - To detect all *d* bit errors (or fewer), Hamming distance of code ≥ *d*+1
  - To correct all *c* bit errors (or fewer), Hamming distance of code ≥ 2*c*+1
  - To correct all *c* bit errors (or fewer) and detect all *c*+*d* bit errors (or fewer), Hamming distance of code ≥ 2*c*+*d*+1

Pitfalls Using Single Metric

- Compare reliability of simplex and TMR systems

$R_{simplex}(t) = e^{-\lambda t}$

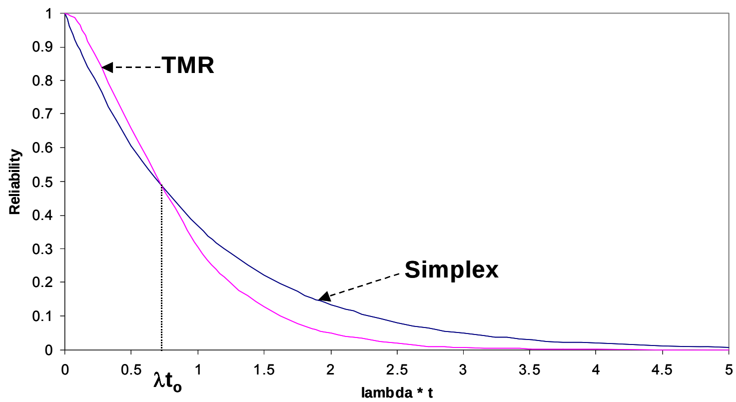
$MTTF_{simplex} = \int_0^\infty e^{-\lambda t} dt = 1/\lambda$

$$R_{TMR}(t) = e^{-3\lambda t} + \binom{3}{2} e^{-2\lambda t}(1 - e^{-\lambda t})$$

$$MTTF_{TMR} = \frac{3}{2\lambda} - \frac{2}{3\lambda} = \frac{5}{6\lambda}$$

$MTTF_{simplex} > MTTF_{TMR}$

Pitfalls Using Single Metric (cont.)



$$R_{TMR}(t) \geq R(t) \qquad 0 \leq t \leq t_0$$

$$R_{TMR}(t) \leq R(t) \qquad t_0 \leq t < \infty$$

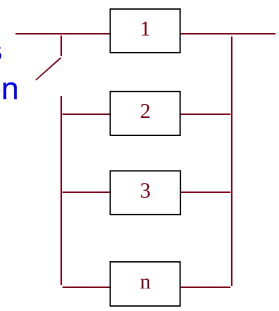
$$\text{where} \quad t_0 = \frac{\ln 2}{\lambda} \approx \frac{0.7}{\lambda}$$

Pitfalls Using Single Metric (cont.)

- Instead of MTTF, look at mission time
- Reliability of M-out-of-N systems very high in the beginning
  - spare components tolerate failures
- Reliability sharply falls down in end
  - system exhausted redundancy, more hardware can possibly fail
- Such systems useful in aircraft control
  - very high reliability, short time
  - 0.99999 over 10 hour period
- Used in FTMP and SIFT multiprocessors

Effect of Coverage

- Failure detection is not perfect
- Reconfiguration may not succeed
- Attach a coverage "c" includes chance for successful detection and switching



One spare system

$R_{sys} = R_1 + c\,(1-R_1)\,R_2$

n-1 spare system

$$R_{sys} = R_m \sum_{i=0}^{n-1} c^i (1 - R_m)^i$$

Effect of Coverage (cont.)

- If coverage is 100%, then given low module reliability, can increase system reliability arbitrarily

With low coverage, reliability saturates

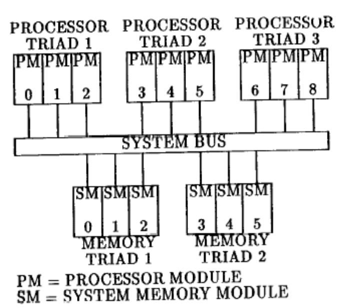
	Rm = 0.9	Rm = 0.7	Rm = 0.5
C=0.99, n=2	0.989	0.908	0.748
C=0.99, n=4	0.999	0.988	0.931
C=0.99, n=inf	0.999	0.996	0.990
C= 0.8 , n=2	0.972	0.868	0.700
C= 0.8 , n=4	0.978	0.918	0.812
C=0.8, n=inf	0.978	0.921	0.833

Voting in Hardware & Software

- Guarantee majority vote on the input data to the voter
- Ability of detecting own errors (self-checking)
- Determine the faulty replica/node (building the exclusion logic)
- Voting in networked systems (software)
  - requires synchronization of inputs to the voter
  - may be difficult to determine voter timeout
    - different relative speed of machines
    - varying network communication delays
- Voting in hardware systems
  - generally does not require an external synchronization of inputs to the voter
  - lock step mode or loosely synchronized mode
  - CPUs internally can be out of synch because of non-deterministic execution of instructions

Example: FTMP (Fault Tolerant Multi Processor)

- Triads of processor-cache do processing
- Triads of memory store data
- Voting done on bus for memory accesses
- System bus is made redundant
- If failure detected
  - Triads recreated with spares
  - Triads broken up and good ones returned to spare pool



Reference

- Today's material from Siewiorek-Swarz Chapter 3, pp. 138-146, pp. 169-193