UNIVERSITY OF MASSACHUSETTS DARTMOUTH

ECE544: Fault-Tolerant Computing & Reliability Engineering

Midterm Exam Review

Instructor: Dr. Liudong Xing

Fall 2022

Administrative Issues (10/12, Wednesday)

- Homework#4 due **Today**
- Midterm Exam
  - On **Oct. 17, Monday**
  - Review session **Today**
- Project Meeting
  - Due **Oct. 28, Friday**

Midterm Exam

- **Time & Place:**
  - 3:30pm-4:55pm on October 17 (Monday)
  - SENG212
- **Form:**
  - Open book and open notes
  - Individual work
- **Preparation**
  - Lecture notes # 1 – 9
  - Homework #1 - #4
  - Relevant readings

Midterm Exam Review

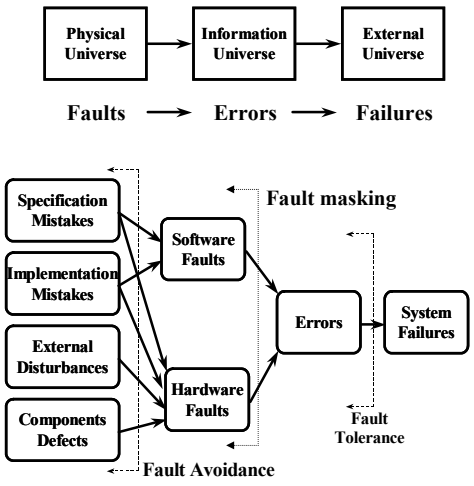
I. Fundamental Concepts
II. Fault Tolerance Techniques
III. Reliability Modeling and Analysis Techniques

Part I: FTC & RE Overview

- **Basic concepts**: Fault-tolerant systems, fault-tolerance, fault-tolerant computing, fault avoidance; reliability, availability, safety, testability, maintainability, performability, graceful degradation, dependability
- **Applications**: Long-life; Critical computation; High availability applications
- **General motivation (4 examples)**
  - **Why fault tolerance?** To increase length of time a system will operate correctly; to minimize amount of time a system is down; to ensure safe operation; to meet certain design requirements
  - **Why reliability analysis?** To predict the reliability of a system for a specified period of time; compare alternative architecture design solution; facilitate trade-off studies for various FT techniques

Part I: Faults, Errors, and Failures

- Concepts
  - Fault: a physical defect, imperfection, or flaw that occurs in HW or SW comp.
  - Error: the occurrence of an incorrect value in some unit of information; the manifestation of a fault; a deviation from accuracy or correctness
  - Failure: a deviation from the expected performance of a system
- Three universe model and cause-and-effect relationship



Midterm Exam Review

I. Fundamental Concepts

II. Fault Tolerance Techniques

III. Reliability Modeling and Analysis Techniques

Part II: Fault Tolerance Techniques

Hardware redundancy	Information redundancy	Time redundancy	Software redundancy
-- Passive	-- Parity	-- Transient	-- Consistency check
-- Active	-- m-of-n	-- Permanent *Alter. Logic *RESO *RESWO	-- Capability check
-- Hybrid	-- Berger		-- RB
	-- Checksum		-- NVP
	-- Cyclic		-- NSCP
	-- Arithmetic		

Hardware Redundancy Techniques (1)

- **Passive** redundancy uses fault masking to hide the occurrence of faults and prevent the faults from resulting in errors and failures
  - **TMR** is the most common form, (multi-stage) triplicated TMR can overcome the effects of the single-point of failure (voter)
  - **Hardware and software voting** have their pros and cons, the decision must be made based on several factors
  - **Mid-value select** and **Voting on part of data** techniques can be used to alleviate the problem of disagreeing results in a NMR system

Hardware Redundancy Techniques (2)

- **Active** redundancy uses detection, location, and recovery techniques (reconfiguration)
  - **Duplication with comparison** can only detect faults, not tolerate them
  - **Hot standby sparing** can minimize the disruption in performance but consume more power than **cold standby sparing**
  - **Warm standby sparing is a tradeoff between hot and cold**

Hardware Redundancy Techniques (3)

- **Hybrid** redundancy employs both fault masking and reconfiguration
  - Requires enough hardware to use voting & for spares
  - The most expensive in terms of hardware required to implement a system, used when highest levels of reliability are desired
  - **NMR with spare** technique can accomplish the same results using fewer hardware modules than passive approaches, but with fault detection/location/recovery schemes
  - **Self-purging redundancy** technique uses the system output to remove modules whose output disagrees with the system output

Information Redundancy Techniques (1)

- **Basic concepts**
  - Code, code word, binary code, error detecting/correcting code, encoding /decoding process
  - Bit, symmetric, asymmetric, unidirectional, and byte errors
  - Hamming distance, code distance, error detection/correction capabilities (3 theorems)

Information Redundancy Techniques (2)

- Parity codes
  - Single-bit parity codes: Detects all errors which involve an odd number of bits
  - Multiple-bit parity codes: **Hamming SEC codes**
    - Calculate number of check bits K
    - Arrange bit positions
    - Generate the check bits
    - Correct the erroneous bit according to the syndrome word
  - Horizontal and vertical parity codes: can correct (detect&locate) any single-bit errors in groups of data words

Information Redundancy Techniques (3)

- m-of-n codes (separable/non-separable) can detect all single-bit errors and all multiple, unidirectional errors
- Berger codes are separable unidirectional error detecting codes; which can be manipulated so that they are invariant to the arithmetic/logical operations
- Checksum (SPC/DPC/Honeywell/Residue) codes are separable codes and can only detect errors but not locate/correct errors

Information Redundancy Techniques (4)

- Cyclic codes (separable/non-separable)
  - Cyclic codes are invariant to the end-around shift operation; are best represented and analyzed using polynomial algebra
  - Cyclic coding can be implemented using combinatorial circuit composed of exclusive-OR gates
- Arithmetic codes
  - AN codes are invariant to addition & subtraction, but not multip. & division
  - Both residue and inverse-residue codes are separable codes

Information Redundancy Techniques (5)

- To select a proper coding scheme in designing the system, three major decisions must be made
  - Whether or not the code needs to be separable
  - Whether error detection, error correction, or both are required
  - Number of bit errors needs to be detected or corrected

Time Redundancy Techniques

- Time redundancy can reduce the amount of extra hardware at the cost of using additional time in achieving fault detection/correction
- Often employed to distinguish between **permanent** and **transient** faults
- Time redundancy **combined with coding schemes** can detect permanent faults (different encoding functions)
  - Alternating logic
  - Recomputing with shifted operands
  - Recomputing with swapped operands
- Time redundancy can provide **error correction** if computation is repeated 3 or more times!

Software Redundancy Techniques

- The addition of redundant software to a system, for the purpose of achieving fault tolerance
  - **Extra code lines or routines**
    - Consistency checks: Use a priori knowledge about the characteristics of information to verify the correctness of the information
    - Capability checks: Performed to verify if a system possesses the capability expected
  - **Extra versions of the complete program**
    - Recovery blocks (RB)
    - N-version programming (NVP)
    - N-self-checking programming (NSCP)

Midterm Exam Review

I. Fundamental Concepts

II. Fault Tolerance Techniques

III. Reliability Modeling and Analysis Techniques

- Reliability measures
- Fault trees and cutsets-based analysis

Quantitative Evaluation Measures

- Time to failure (T): a r.v. describing the time elapsing from when a component is put into operation until it fails for the first time
  - $F(t)$: cumulative distribution function (c.d.f) of the r.v. T; failure function
  - $f(t)$: probability density function (p.d.f.) of T
- Reliability/survivor function $R(t)=1-F(t)$
- Failure rate (hazard rate/function) $z(t)$
  - The bathtub curve: burn-in/infant mortality period, **useful-life period**, wear-out period

Quantitative Evaluation Measures

- Mean time to failure (MTTF)
  - Mean time to repair (MTTR), Mean time between failure (MTBF)
  - MTBF=MTTF+MTTR
- Mean residual life (MRL) at age *t*:

$$MRL\ (t) = \int_0^\infty R(x \mid t)\,dx = \frac{1}{R(t)} \int_t^\infty R(x)\,dx$$

- Relationship between F(t), f(t), z(t), R(t), and MTTF
- Time to failure distributions: exponential distributions with constant failure rate and memory-less property

Reliability Modeling and Analysis (System-Level)

- Fault tree
- Minimal cut-set
  - Inclusion-exclusion (I/E)
  - Sum of disjoint products (SDP)

……More after Midterm Exam

Fault Trees

- A **failure-oriented** model, expressing combinations of component failures that can lead to system failure
- **Top-down construction**: consists of a top event (system failure), basic events (component failures), and gates that connect the events.
- **Fault trees** consist of only static gates (AND, OR, K/N) and model static systems whose failures are simply logical combinations of component failures
- Analysis of fault trees is based on **minimal cutsets**

Minimal Cutsets and Analysis

- **Cut-sets**: a set of components which by failing causes the system to fail; a cut set is **minimal** if it cannot be reduced without loosing its status as a cut set
- **Unreliability** analysis of fault trees using **minimal cutsets**

Inclusion/Exclusion (IE)

$$\begin{aligned} Pr\{\bigcup_{i=1}^{n} C_i\} &= \sum_{i=1}^{n} Pr\{C_i\} \\ &- \sum_{i<j} Pr\{C_i \bigcap C_j\} \\ &+ \sum_{i<j<k} Pr\{C_i \bigcap C_j \bigcap C_k\} \\ &\mp \cdots \\ &\pm Pr\{\bigcap_{i=1}^{n} C_i\} \end{aligned}$$

Sum of Disjoint Products(SDP)

$$\begin{aligned} Unreliability &= \Pr\{\bigcup_{i=1}^{n} C_i\} \\ &= P(C_1) + P(\overline{C}_1 C_2) + P(\overline{C}_1 \overline{C}_2 C_3) \\ &\quad + \ldots + P(\overline{C}_1 \overline{C}_2 \overline{C}_3 \ldots \overline{C}_{n-1} C_n) \end{aligned}$$

Midterm Exam

- **Time & Place:**
  - 3:30pm-4:55pm on October 17 (Monday)
  - SENG212
- **Form:**
  - Open book and open notes
  - Individual work
- **Preparation**
  - Lecture notes # 1 – 9
  - Homework #1 - #4
  - Relevant readings

Good Luck!!!