UNIVERSITY OF MASSACHUSETTS DARTMOUTH

ECE454/544: Fault-Tolerant Computing & Reliability Engineering

Final Review

Instructor: Dr. Liudong Xing

Fall 2022

Administrative Issues

(Nov. 30, Wednesday)

- Last lecture today: **Final Review**
- Class Project
  - Final report due **Today**
  - Presentation slides due by **Dec. 5, Monday**
  - Please check out the Report, Presentation Guidelines for requirements

Slot	Dec. 5 (Mon)	Slot	Dec. 7 (Wed)
1	Team 3	4	Team 1
2	Team 2	5	Team 4
3	Team 5	6	Team 6

25 minutes per presentation

Final Exam

- **Time & Place:**
  - Dec. 15, Thursday
  - 3pm ~ 6pm @ SENG 212
- **Form:**
  - Open book and open notes
  - Individual work
- **Preparation**
  - Lecture #8, 9, 11–17
  - Homework #4 - #7
  - Relevant readings

Final Review

I. Overview of fault tolerant computing & reliability engineering (FTC & RE) (*L#1-2*)

II. Fault tolerance and avoidance techniques (*L#3-6*)

III. Reliability modeling and analysis (*L#7 – 18, except L#10*)

Part I: FTC & RE Overview

1. Fundamental concepts & general motivations (*L#1*)
2. Faults, errors, and failures & cause-and-effect relationship (*L#2*)

Part I: FTC & RE Overview

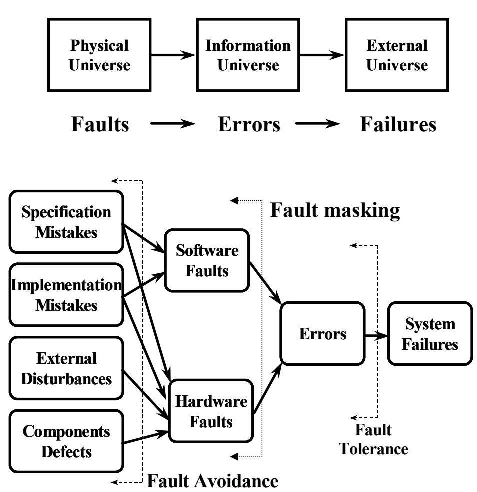
- **Basic concepts**: Fault-tolerant systems, fault-tolerance, fault-tolerant computing, fault avoidance; reliability, availability, safety, testability, maintainability, performability, graceful degradation, dependability
- **Applications**: Long-life; Critical computation; High availability applications
- **General motivation**
  - **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

Review Questions (True/False, L#2)

- ____ Reliability differs from availability in that reliability depends on an interval of time whereas availability is taken at an instant of time
- ____ A fault-tolerant system must have a high reliability
- ____ A highly reliable system must be fault-tolerant
- ____ A system's safety is usually larger than a system's reliability value
- ____ A TMR system is more reliable than a standby sparing design with one spare

Part II: 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



Final Review

I. Overview of fault tolerant computing & reliability engineering (FTC & RE) (*L#1-2*)

II. Fault tolerance and avoidance techniques (L#3-6)

1. Hardware redundancy (*L#3*)
2. Information redundancy (*L#4 & 5*)
3. Time redundancy (*L#6*)
4. Software redundancy (*L#6*)

III. Reliability modeling and analysis (*L#7 –18, except L#10*)

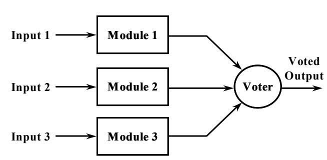
Fault Tolerance Techniques

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

Reliability of TMR (L#3, Hands-on Problem)



- Reliability of each module: *p*
- Reliability of the voter: *w*
- Reliability of TMR?

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**
  - **Pair-and-a-spare** combines both

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)

Final Review

I. Overview of fault tolerant computing & reliability engineering (FTC & RE) (*L#1-2*)

II. Fault tolerance and avoidance techniques (*L#3-6*)

III. Reliability modeling and analysis (*L#7 – 18*)

1. Probability theory review (*L#7*)
2. Time-to-failure models (*L#8*)
3. Fault tree analysis (*L#9*)
4. Reliability block diagrams (*L#11*)
5. Binary decision diagrams (*L#12*)
6. Component sensitivity analysis (*L#13*)
7. Dynamic fault trees (*L#14*)
8. Markov analysis (*L#15, 16*)
9. Network reliability analysis (*L#17*)
10. Trust sensitivity analysis for social networks (*L#18*)

Quantitative Evaluation Measures (Component Level: 1)

- 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 (Component Level: 2)

- 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, L#9-18)

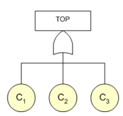
- Static fault tree (*L#9*)
- Reliability block diagram (RBD) (*L#11*)
- Minimal cut-set, minimal path-set (*L#9 &11*)
  - Inclusion-exclusion (I/E)
  - Sum of disjoint products (SDP)
- Binary decision diagram (BDD) (*L#12*)
- Dynamic fault tree & Markov analysis (*L#14-16*)
- Component sensitivity analysis (*L#13*)
- Network reliability analysis (*L#17*)
- Trust sensitivity analysis for social networks (*L#18*)

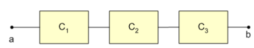
Fault Trees (*L#9*)

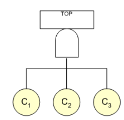
- A **failure-oriented** model, expressing combinations of component failures that can lead to system failure
- **Top-down construction**: consists of a top event (system failure), basic events (component failures), and gates that connect the events.
- **Static fault trees** consist of only static gates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

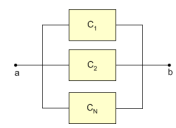
Reliability Block Diagrams (RBD) (*L#11*)

- A RBD is a **success-oriented** network describing the function of the system
- In terms of modeling capability, RBD is equivalent to the static/traditional fault trees, and they can be converted into each other easily
  - Fault tree → RBD: starting from the TOP event and replacing the gates successively; OR-gates are replaced by series structures of the components directly beneath the gate; AND-gates are replaced by parallel structures of the components directly beneath the gate
- Quantitative analysis of RBD using minimal cut-sets and path-sets









Minimal Cutsets and Pathsets (*L#9 & 11*)

- **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
- **Path-sets**: a set of components which by functioning ensures that the system is functioning; a path set is **minimal** if it cannot be reduced without loosing its status as a path set

- Path-sets and cut-sets can be generated from both RBD and fault trees
  - From the RBD, you can identify the pathsets by enumerating all paths between the end points and cutsets by enumerating all sets of components which can interrupt/disconnect the path between the two end points
  - Apply the top-down algorithm (L#9) to original fault tree to obtain minimal cutsets; to dual fault tree for obtaining minimal pathsets

- Inclusion/Exclusion (IE) and Sum of Disjoint Products (SDP) can be applied to the quantitative analysis based on both path sets and cut sets

Minimal Cutsets and Pathsets (*L#9 &11*)

- **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)

$$
\begin{aligned}
Unreliability &= \Pr\{\bigcup_{i=1}^{n} C_i\} \\
&= P(C_1) + P(\overline{C_1} C_2) + P(\overline{C_1}\,\overline{C_2} C_3) \\
&+ \ldots + P(\overline{C_1}\,\overline{C_2}\,\overline{C_3} \ldots \overline{C_{n-1}} C_n)
\end{aligned}
$$

- **Reliability** analysis of fault trees using **minimal pathsets**
  - Inclusion/Exclusion (IE)

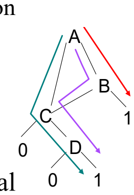
$$
\begin{aligned}
Reliablity = \Pr\{\bigcup_{i=1}^{n} P_i\} &= \sum_{i=1}^{n} \Pr\{P_i\} \\
&- \sum_{i<j} \Pr\{P_i \cap P_j\} \\
&+ \sum_{i<j<k} \Pr\{P_i \cap P_j \cap P_k\} \\
&\mp \ldots\ldots \\
&\pm \Pr\{\bigcap_{j=1}^{n} P_j\}
\end{aligned}
$$

  - Sum of Disjoint Products(SDP)

$$
\begin{aligned}
Reliablity &= \Pr\{\bigcup_{i=1}^{n} P_i\} \\
&= P(P_1) + P(\overline{P_1} P_2) + P(\overline{P_1}\,\overline{P_2} P_3) + \ldots + P(\overline{P_1}\,\overline{P_2}\,\overline{P_3} \ldots \overline{P_{n-1}} P_n)
\end{aligned}
$$

Binary Decision Diagrams (BDD) (*L#12*)

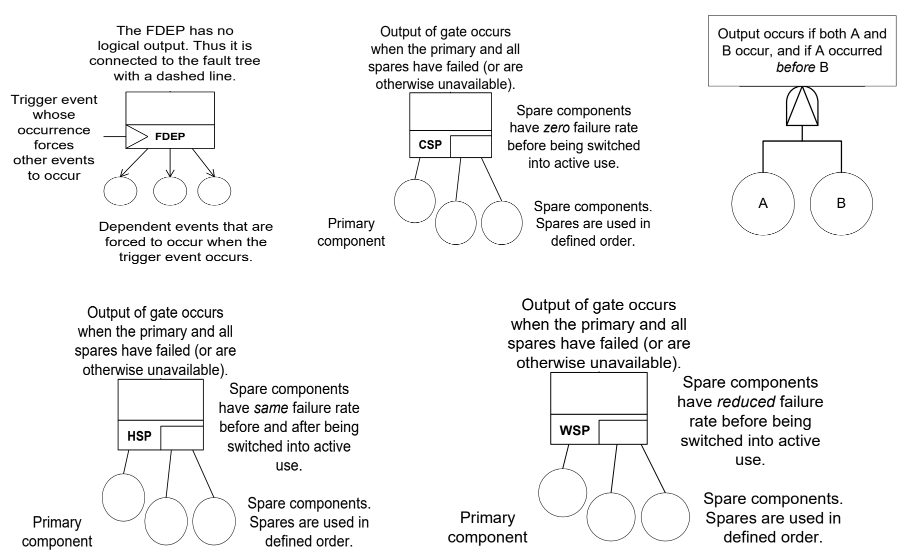
- A BDD is a binary tree based on Shannon decomposition
  - two sink nodes labeled with constants 1 and 0
  - each non-sink node is labeled with a Boolean variable x and has two outgoing edges called 1-edge (then-edge) and 0-edge (else-edge), respectively
  - 1-edge represents the Boolean expression when x=1: $F_1 = f_{x=1}$ in Shannon decomposition
  - 0-edge represents the Boolean expression when x=0: $F_0 = f_{x=0}$ in Shannon decomposition
- The size of the BDD depends heavily on the input variable ordering
  - Heuristics can usually be used to find a reasonable variable ordering
- BDD can be used to efficiently and accurately solve the combinatorial reliability models without the use of cutsets
  - System unreliability is given by the sum of the probabilities for all paths from the root to a leaf node labeled 1



Dynamic Fault Trees (*L#14*)

- **Dynamic fault trees** consist of at least one dynamic gate (CSP, WSP, HSP, FDEP, PAND)
- Model dynamic systems whose failures depend not only on the logical combinations of component failures but also on the order in which the events occur
- Markov model and modular solution (L#15, 16)

A Review of Dynamic Gates (*L#14*)



Markov Model Based Solution (*L#15-16*)

- A Markov process is a stochastic process with Markov property: probabilities of future states depend only on the current state and not on the history
- Both static and dynamic systems with exponential component failure distribution can be solved as a Markov chain
- System states **S** can be grouped into two sets: operational state **O** and failed state **F**

$$\text{System reliability} = R_{sys} = \sum_{i \in O} P_i(t)$$

$$\text{System unreliability} = U_{sys} = \sum_{i \in F} P_i(t)$$

Markov Model Based Solution (*L#15-16*)

- Unique solution can be found by solving

$$\begin{cases} \boldsymbol{A} \bullet \boldsymbol{P}(\boldsymbol{t}) = \dot{\boldsymbol{P}}(\boldsymbol{t}) \\ \sum_{j=0}^{r} P_j(t) = 1, \quad \text{the system must be in one of (r + 1) states} \\ P_i(0) = 1,\ P_k(0) = 0,\ \text{for } k = 0, \ldots, r \text{ and } k \neq i \quad \text{initial condition} \end{cases}$$

Markov Model Based Solution (*L#15-16*)

- Time-dependent solution based on Laplace transform

$$\begin{bmatrix} -\alpha_{00} & \alpha_{10} & \alpha_{20} & \ldots\alpha_{r0} \\ \alpha_{01} & -\alpha_{11} & \alpha_{21} & \ldots\alpha_{r1} \\ \alpha_{02} & \alpha_{12} & -\alpha_{22} & \ldots\alpha_{r2} \\ : & : & : & : \\ \alpha_{0r} & \alpha_{1r} & \alpha_{2r} & \ldots-\alpha_{rr} \end{bmatrix} \bullet \begin{bmatrix} P_0^*(s) \\ P_1^*(s) \\ P_2^*(s) \\ : \\ P_r^*(s) \end{bmatrix} = \begin{bmatrix} sP_0^*(s) \\ sP_1^*(s) \\ sP_2^*(s) \\ : \\ sP_r^*(s) \end{bmatrix} - \begin{bmatrix} P_0(0) \\ P_1(0) \\ P_2(0) \\ : \\ P_r(0) \end{bmatrix} \qquad \sum_{i=0}^{r} P_i^*(s) = \frac{1}{s}$$

- Asymptotic / long-run solution

$$\begin{bmatrix} -\alpha_{00} & \alpha_{10} & \alpha_{20} & \ldots\alpha_{r0} \\ \alpha_{01} & -\alpha_{11} & \alpha_{21} & \ldots\alpha_{r1} \\ \alpha_{02} & \alpha_{12} & -\alpha_{22} & \ldots\alpha_{r2} \\ : & : & : & : \\ \alpha_{0r} & \alpha_{1r} & \alpha_{2r} & \ldots-\alpha_{rr} \end{bmatrix} \bullet \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ : \\ P_r \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ : \\ 0 \end{bmatrix} \qquad \sum_{j=0}^{r} P_j(t) = 1$$

Markov Model Based Solution (*L#15-16*)

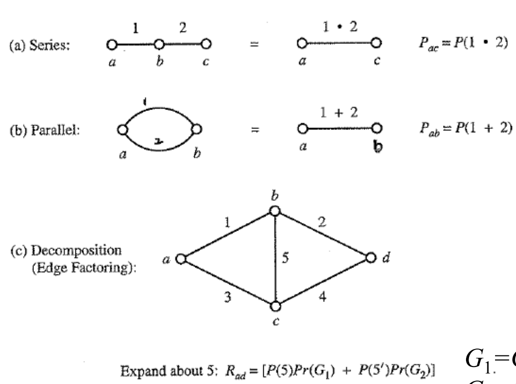
- Powerful in terms of modeling capability, as compared with combinatorial models (RBD, BDD, static fault trees, cut-sets, etc), allowing for easily modeling
  - various dependencies (FDEP, HSP, WSP, CSP, PAND)
  - rather complicated repair strategies
  - fault imperfect coverage
- Solution to state explosion problem
  - Modularization
  - Bounding method

Component Sensitivity Analysis (*L#13*)

- Component importance/sensitivity analysis measures the sensitivity of the system unreliability to the component failure parameters, which has two classes
  - Structure/deterministic importance measures
  - Reliability/probabilistic importance measures
- Applications
  - Improvement Oriented: helps identify which components contribute most to the system reliability and thus they will be good candidates for efforts leading to improving system reliability, e.g.: **Birnbaum's measure**, improvement potential
  - Maintenance Oriented: helps identify the component that has the largest probability of being the cause of system failure → set up a repairperson's checklist, e.g.: criticality importance factor, diagnostic importance factor, Fussel-Vesley measure

Two-Terminal Network Reliability Analysis (*L#17*)

- State space enumeration method
- Cut-set ad tie-set method
- Binary decision diagrams-based method
- Graph transformation method



- Use series and parallel transformations first
- Resort to edge-factoring only when no more series or parallel transformations can be made!

$G_1 = G \cdot 5$ (*G* contract *edge 5*)
$G_2 = G - 5$ (*G delete edge 5*)

Trust Sensitivity Analysis for Social Networks (*L#18*)

Two-party trust	All-party trust	*K*-party trust
• Trust relationship between two particular parties	• Trust relationship between all parties	• Trust relationship between a subset of *K* parties

- An application of sensitivity analysis (L#13) and network reliability analysis (L#17)

39

Final Exam

- **Time & Place:**
  - Dec. 15, Thursday
  - 3pm ~ 6pm @ SENG 212
- **Form:**
  - Open book and open notes
  - Individual work
- **Preparation**
  - Lecture #8, 9, 11–17
  - Homework #4 - #7
  - Relevant readings

Good luck to your finals!!!
and
Have a happy & safe holiday!