ECE454/544: Fault-Tolerant Computing & Reliability Engineering

Lecture #6 –

Time & Software Redundancy Techniques

Instructor: Dr. Liudong Xing

Fall 2022

Administrative Issues

- Homework#2 due 9/28 (W)
- Homework#3 assigned
  - Please download the problems from the course website: https://xingteaching.sites.umassd.edu/
  - Due **Oct. 5, Wednesday**
- Project Proposal
  - Due **Oct. 5, Wednesday**
  - Refer to Proposal Guideline on the course website

Review of Lecture #5

- 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
- Cyclic codes are invariant to the end-around shift operation; are best represented and analyzed using polynomial algebra; can be separable and non-separable
- AN codes are invariant to addition and subtraction, but not multiplication and division
- Both residue and inverse-residue codes are separable codes
- Factors considered in codes selection

Outline

- Time redundancy techniques
- Software redundancy techniques

Time Redundancy (Agenda)

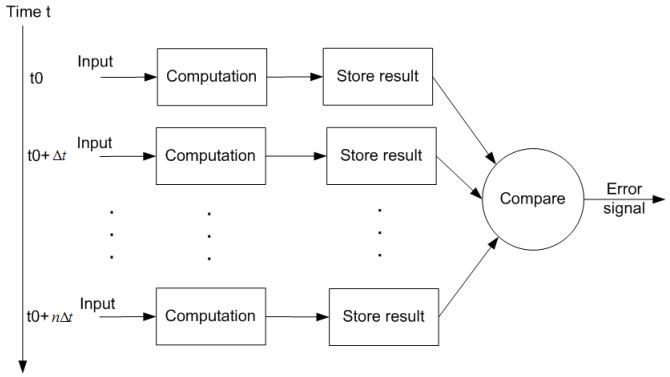
- Motivations
- Transient fault detection
- Permanent fault detection
- Error correction

Motivations & Basic Concept

- Both hardware and information redundancy can require large amounts of extra hardware for the implementation
- In many applications, time is of much less importance than hardware and is readily available because hardware is a physical entity impacting weight, size, power consumption, and cost
- Basic concept of time redundancy is the repetition of computations in ways that allow faults to be detected
- Time redundancy can reduce the amount of extra hardware at the cost of using additional time for achieving fault detection / correction

Transient Fault Detection

- Computations are repeated at different points in time and then compared
- If an error is detected, the computation can be performed again to see if the discrepancy remains or disappear



Transient Fault Detection (Cont'd)

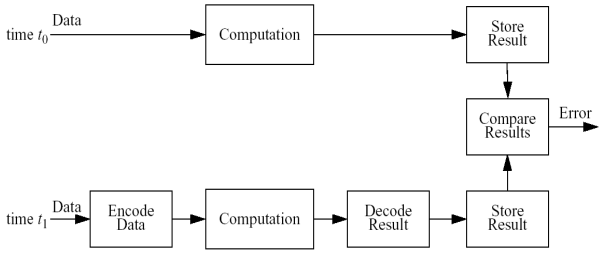
- Often used to distinguish between permanent and transient faults
  - Repeat the computation after an error is detected
  - Transient if error conditions clears
  - Permanent if problem continues to exist

Permanent Fault Detection

- Time redundancy combined with coding schemes (information redundancy) can detect permanent faults
  - Perform the same computation multiple times using different coding schemes in each case
  - **Example**: the arithmetic operations performed on a processor protected by some AN arithmetic code
  - *Case 1:*
    - First without using any code
    - Second use 3N code
  - *Case 2:*
    - First use 3N code
    - Second use 5N code

Basic Strategy

- Operands used as presented during 1st computation/transmission
- Operands encoded using some encoding function, results decoded and compared with those obtained during the first operation



Example Approaches

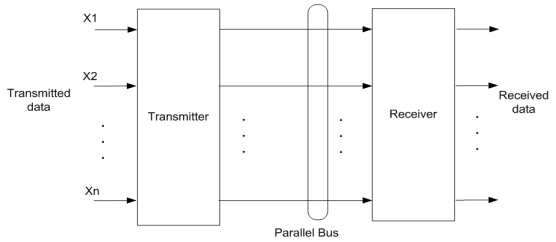
- Alternating logic
  - Complementation
- Recomputing with shifted operands (RESO)
  - Arithmetic shift
- Recomputing with swapped operands (RESWO)
  - Swapping function

Different encoding functions employed!

Example I: Alternating Logic

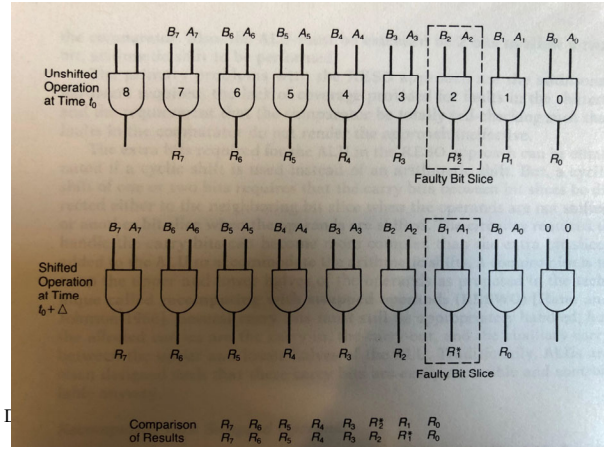
- Encoding function is the complementation operation
- Applied to the transmission of digital data over wire media and fault detection in digital circuits

- An example: a transmission system protected using time redundancy
  - Data is transmitted over a parallel bus
  - At t0, original data is transmitted
  - At t1, the complement of the data is transmitted
  - If one line of the bus is stuck at either a 1 or a 0, the two versions of information received won't be complements of each other → fault detected!



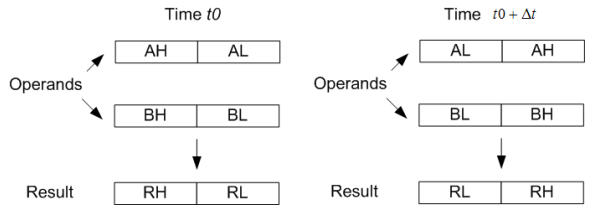
Example II: Recomputing with Shifted Operands (RESO)

- Encoding function is a left shift operation
- Decoding function is thus a right shift operation
- Originally developed as a method to detect concurrent errors in ALU
- Assume bit-sliced organization of hardware
- Ex: Suppose bit slice 2 of the circuit is faulty (Johnson89: Figure 3.67)
  - The two results will disagree in both 1st and 2nd bits → fault detected!



Example III: Recomputing with Swapped Operands (RESWO)

- Encoding function is **swapping operation**
- During 1st computation, operands are manipulated in standard form
- During 2nd computation, upper and lower halves of the operands are swapped
- If a bit slice is faulty, it operates on either the lower or upper half of the operands during 1st computation and the opposite half of the operands during 2nd computation
- An example



Error Correction

- Time redundancy can correct errors in logic operations if the computations are repeated 3 or more times
- The errors resulted from the faulty bit slice can be corrected by performing a **majority voting**
- An example
  - Perform a logic AND operation on two 8-bit operands
  - Johnson89: Figure 3.75

Error Correction Example

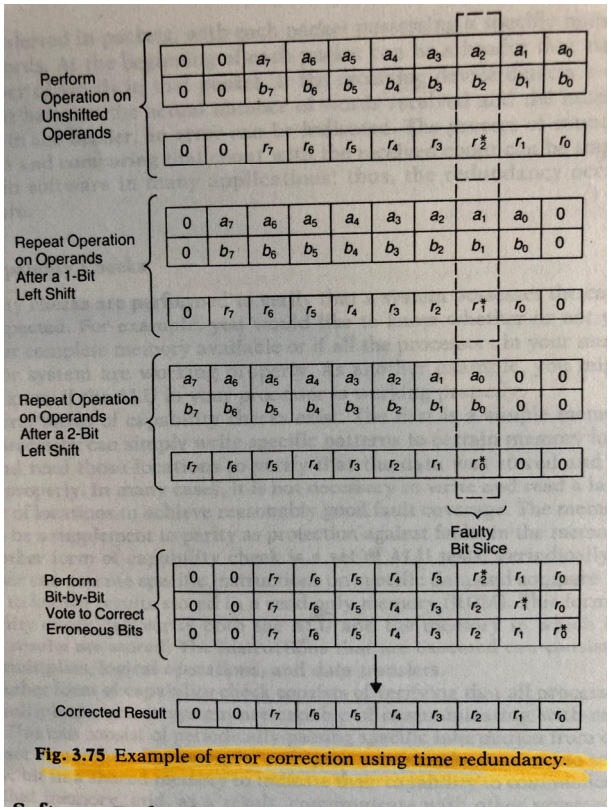


Fig. 3.75 Example of error correction using time redundancy.

Summary

- 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
  - Alternating logic, Recomputing with shifted operands, Recomputing with swapped operands
- Time redundancy can provide error correction if computation is repeated 3 or more times!

Agenda

✓ Time redundancy techniques

- **Software redundancy techniques**

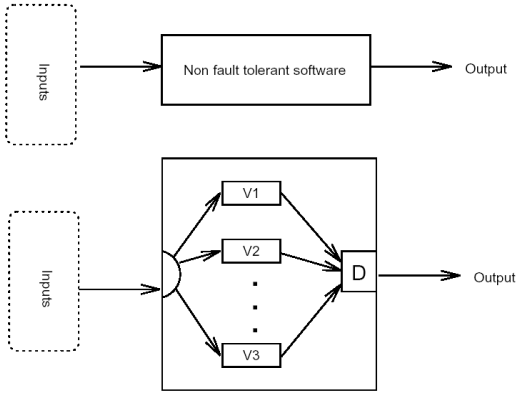
Software Redundancy

- The addition of redundant software to a system, for the purpose of achieving fault tolerance
  - Extra code lines or routines
  - Extra versions of the complete program
- Software redundancy techniques
  - Consistency checks
  - Capability checks
  - Recovery blocks (RB)
  - N-version programming (NVP)
  - N-self-checking programming (NSCP)

Software Redundancy Techniques (1)

- Consistency checks
  - Use a priori knowledge about the characteristics of information to verify the correctness of the information
  - Examples:
    - Sensor readings are often checked to verify that they lies within an acceptable range of values
    - Arithmetic operations
- Capability checks
  - Performed to verify if a system possesses the capability expected
  - Examples:
    - Run a set of arithmetic/logic operation instructions to test if the ALU works properly

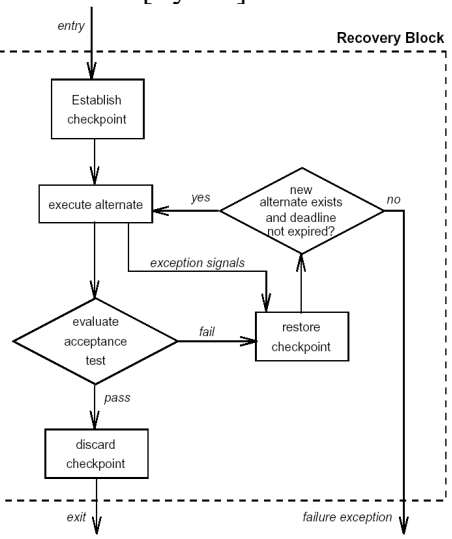
Software Redundancy Techniques (2)



- Redundancy: multiple (diverse) versions of a software module
- A decision mechanism: to detect errors and determine a correct result
- Three popular approaches
  - RB (Recovery block)
  - NVP (N-version programming)
  - NSCP (N-self-checking programming)

Recovery Block (RB)

- Three software elements
  - A primary routine (PR) executing a critical function
  - An acceptance test checking the results of the PR after each execution
  - One or more secondary/alternate routines
    - Performing the same function as the PR
- Implementation [Lyu96]



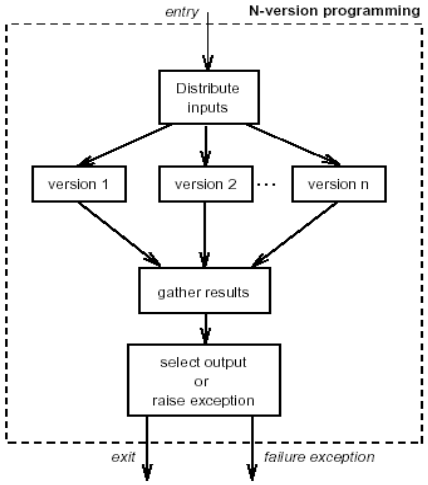
Implementation of RB

- Current state of computation and inputs are saved by a checkpoint
- The PR calculates a result, which is checked by the acceptance test
  - If test is passed, the check point is discarded and the RB has performed successfully
  - Otherwise, the system state is rolled back to the checkpointed values → an alternate routine attempts the computation, and the result is checked by the acceptance test → if test is passed, the RB has performed successfully
  - If all the alternates fail the test, the RB has failed

Combine RB with Hardware Redundancy

- Distributed recovery block (DRB)
  - One processor executes the PR while the other executes the secondary
  - If an error is detected in the primary results, the results from the secondary are immediate available

N-Version Programming (NVP)



- A direct software analog of NMR
- Software elements
  - 3 or more independent versions of a software module
  - A decision algorithm, usually a majority voting mechanism
  - Multiple versions can be executed sequentially on a single processor or concurrently when sufficient processing power is available

NVP (Cont'd)

- To develop independent versions
  - Each version is designed and coded by a separate group of programmers
  - Each group works independently of the others, and communication between groups is not permitted except by passing message (which can be edited or blocked) via the contracting organization
  - Each group designs the software from the same set of specifications / requirements
  - Each version is subjected to the same set of comprehensive acceptance tests

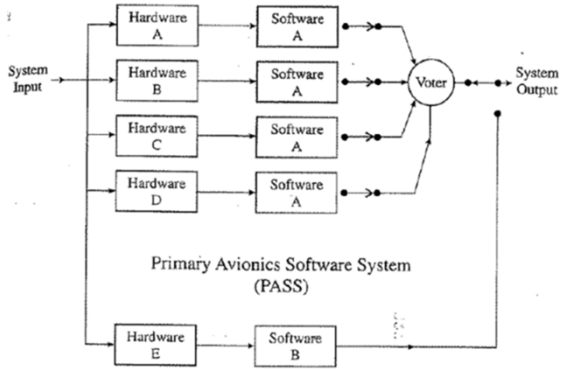
NVP (Cont'd)

- Primary difficulties
  - Software designers and programmers can tend to make similar mistakes
    - Identical misinterpretation of the specifications
    - Identical, incorrect designs for difficult portions of the problem
  - NVP cannot detect specification mistakes

Real Applications of NVP

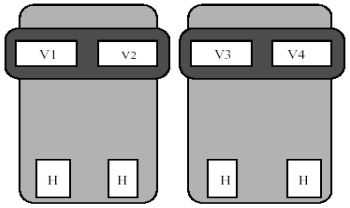
- Point switching, signal control, and traffic control in the Goteborg area of the Swedish State Railway
- Nuclear reactor control systems
- Space shuttle orbiter flight control system (Shooman Ch 5.9.3 – Figure 5.19)
  - Refer to extra note on the course website for details

Space shuttle orbiter flight control system



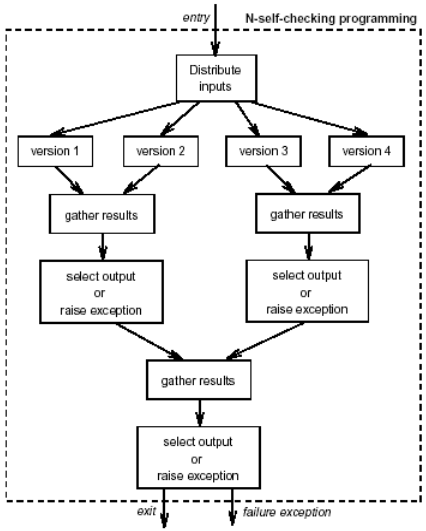
- Five identical computers (Hardware A-E)
- Two different software systems (A, B)
  - A developed by IBM Federal Systems Division
  - B developed by Rockwell and draper Labs
- Primary Avionics Software System (PASS)
  - Computers A-D, each using software A, are connected in a voting arrangement
  - Can sustain two failures (**2-out-of-4 system**): If one computer fails, such computer is disconnected from the voting arrangement, the remaining ones form a TMR system
  - Vulnerable to common-mode software failures in Software A
- Backup Flight Control System (BFS)

N-Self-Checking Programming (NSCP)



- 4 software versions and 4 hardware components, grouped into two pairs
- Hardware pairs operate in hot standby redundancy with each hardware component supporting one software version
- The four software versions are executed and results of V1 and V2 are compared, results of V3 and V4 are compared

General Structure of NSCP



- If either pair of results do not match, that pair is discarded and only remaining pair is used
- If each pair of results matches, the results of the two pairs are then compared
- A hardware fault causes the software version running on it to produce incorrect results, as would a fault in the software version itself!

Summary

- Five software redundancy techniques are discussed:
  - Consistency checks
  - Capability checks
  - Recovery blocks (RB)
  - N-version programming (NVP)
  - N-self-checking programming (NSCP)

Things to Do

- Homework
- ECE544 Project Proposal
  - Check out the guidelines from course website
  - Due **Wednesday, Oct. 5**