CprE 458/558: Real-Time Systems

Lecture 17

Fault-tolerant design techniques

Fault Tolerant Strategies

- Fault tolerance in computer system is achieved through redundancy in hardware, software, information, and/or computations. Such redundancy can be implemented in static, dynamic, or hybrid configurations.
- Fault tolerance can be achieved by many techniques:
  - **Fault masking** is any process that prevents faults in a system from introducing errors. Example: Error correcting memories and majority voting.
  - **Reconfiguration** is the process of eliminating faulty component from a system and restoring the system to some operational state.

Reconfiguration Approach

- **Fault detection** is the process of recognizing that a fault has occurred. Fault detection is often required before any recovery procedure can be initiated.
- **Fault location** is the process of determining where a fault has occurred so that an appropriate recovery can be initiated.
- **Fault containment** is the process of isolating a fault and preventing the effects of that fault from propagating throughout the system.
- **Fault recovery** is the process of remaining operational or regaining operational status via reconfiguration even in the presence of faults.

The Concept of Redundancy

- *Redundancy* is simply the addition of information, resources, or time beyond what is needed for normal system operation.
- **Hardware redundancy** is the addition of extra hardware, usually for the purpose either detecting or tolerating faults.
- **Software redundancy** is the addition of extra software, beyond what is needed to perform a given function, to detect and possibly tolerate faults.
- **Information redundancy** is the addition of extra information beyond that required to implement a given function; for example, error detection codes.

The Concept of Redundancy (Cont'd)

- **Time redundancy** uses additional time to perform the functions of a system such that fault detection and often fault tolerance can be achieved. Transient faults are tolerated by this.

- The use of redundancy can provide additional capabilities within a system. But, redundancy can have very important impact on a system's performance, size, weight, power consumption, and reliability.

Hardware Redundancy

- **Passive techniques** use the concept of fault masking. These techniques are designed to achieve fault tolerance without requiring any action on the part of the system. Relies on voting mechanisms.
- **Active techniques** achieve fault tolerance by detecting the existence of faults and performing some action to remove the faulty hardware from the system. That is, active techniques use fault detection, fault location, and fault recovery in an attempt to achieve fault tolerance.

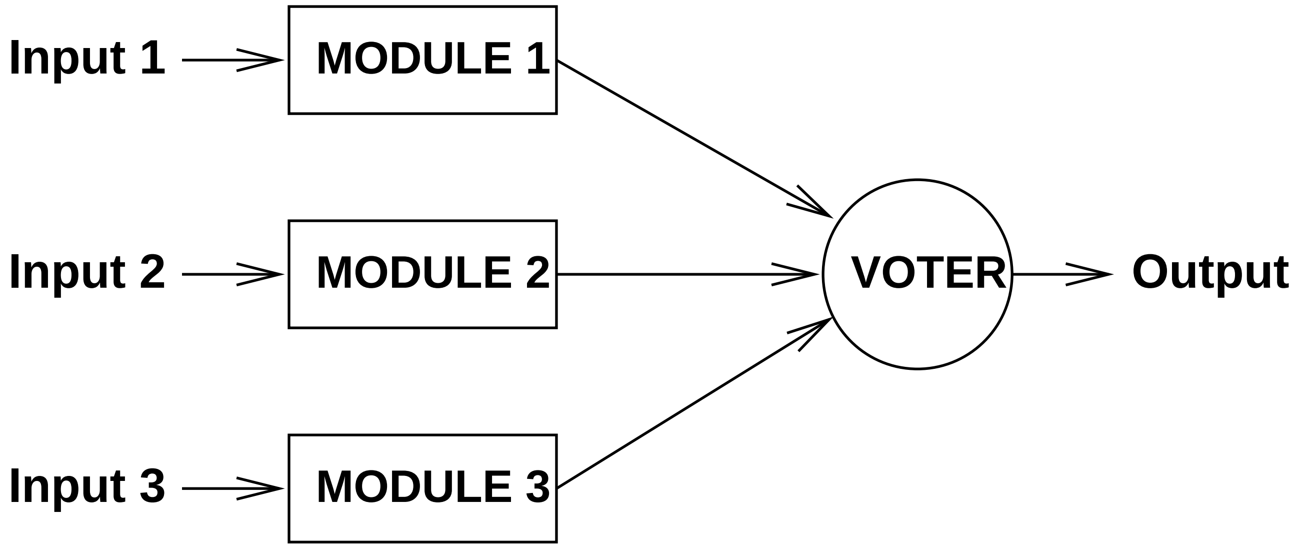
Hardware Redundancy (Cont'd)

- **Hybrid techniques** combine the attractive features of both the passive
and active approaches.
  - Fault masking is used in hybrid systems to prevent erroneous results from being generated.
  - Fault detection, location, and recovery are also used to improve fault tolerance by removing faulty hardware and replacing it with spares.

Hardware Redundancy - A Taxonomy

- Hardware Redundancy
  - Passive Techniques
    - Triple Modular Redundancy
    - N-Modular Redundancy
  - Active Techniques
    - Duplication with Comparison
    - Standby Sparing
      - Hot standby
      - Cold standby
    - Pair-and-a-Spair
    - Watchdog timer
  - Hybrid Techniques
    - NMR with Spares
    - Slef-Purging Redundancy
    - Sift-Out Redundancy
    - Triple-Duplex Architecture

Triple Modular Redundancy (TMR)



Software Redundancy - to Detect Software Faults

- There are two popular approaches: **N-Version Programming** (NVP) and **Recovery Blocks (RB)**.

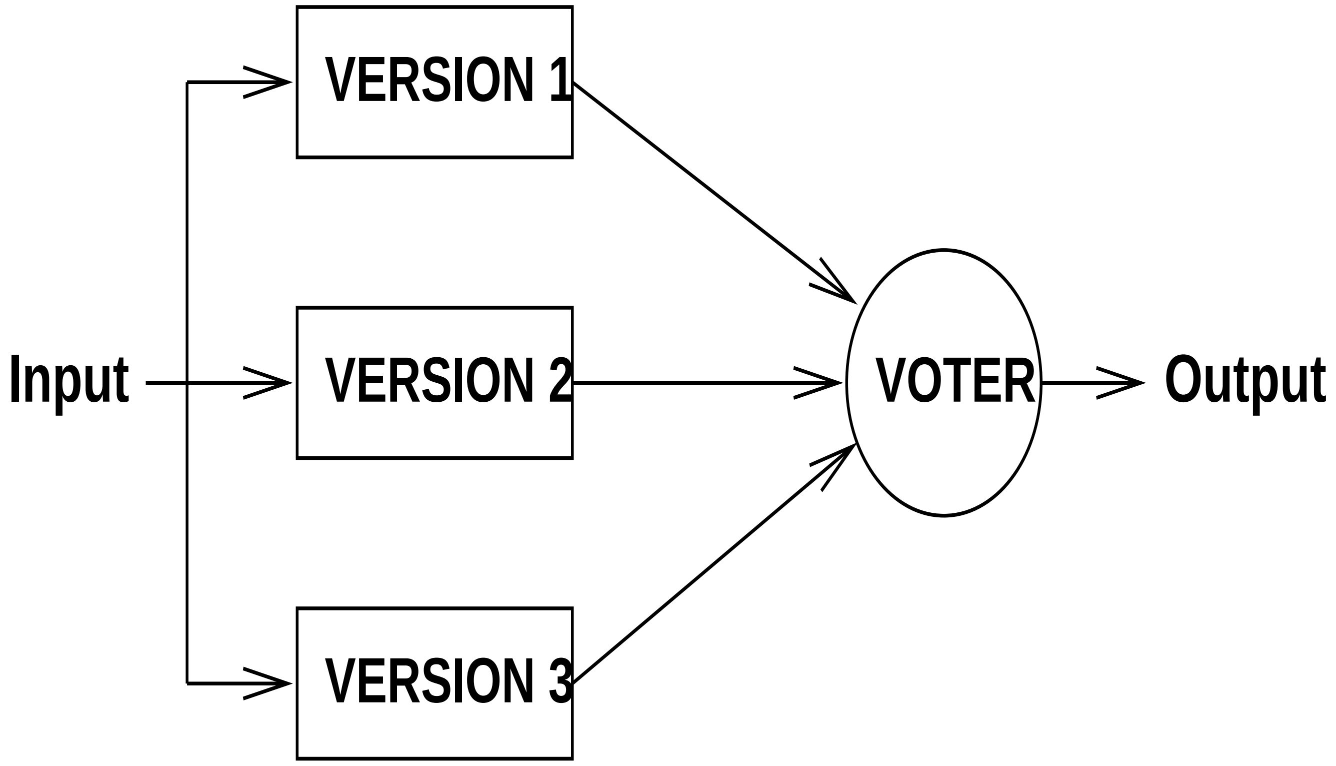
- NVP is a forward recovery scheme - it masks faults.
- NVP: multiple versions of the same task is executed concurrently.
- NVP relies on *voting.*

- RB is a backward error recovery scheme.
- RB: the versions of a task are executed serially.
- RB relies on *acceptance test.*

N-Version Programming (NVP)

- NVP is based on the principle of **design diversity**, that is coding a software module by different teams of programmers, to have multiple versions.
- Diversity can also be introduced by employing different algorithms for obtaining the same solution or by choosing different programming languages.
- NVP can tolerate both hardware and software faults.
- Correlated faults are not tolerated by the NVP.
- In NVP, deciding the number of versions required to ensure acceptable levels of software reliability is an important design consideration.

N-Version Programming (Cont'd)



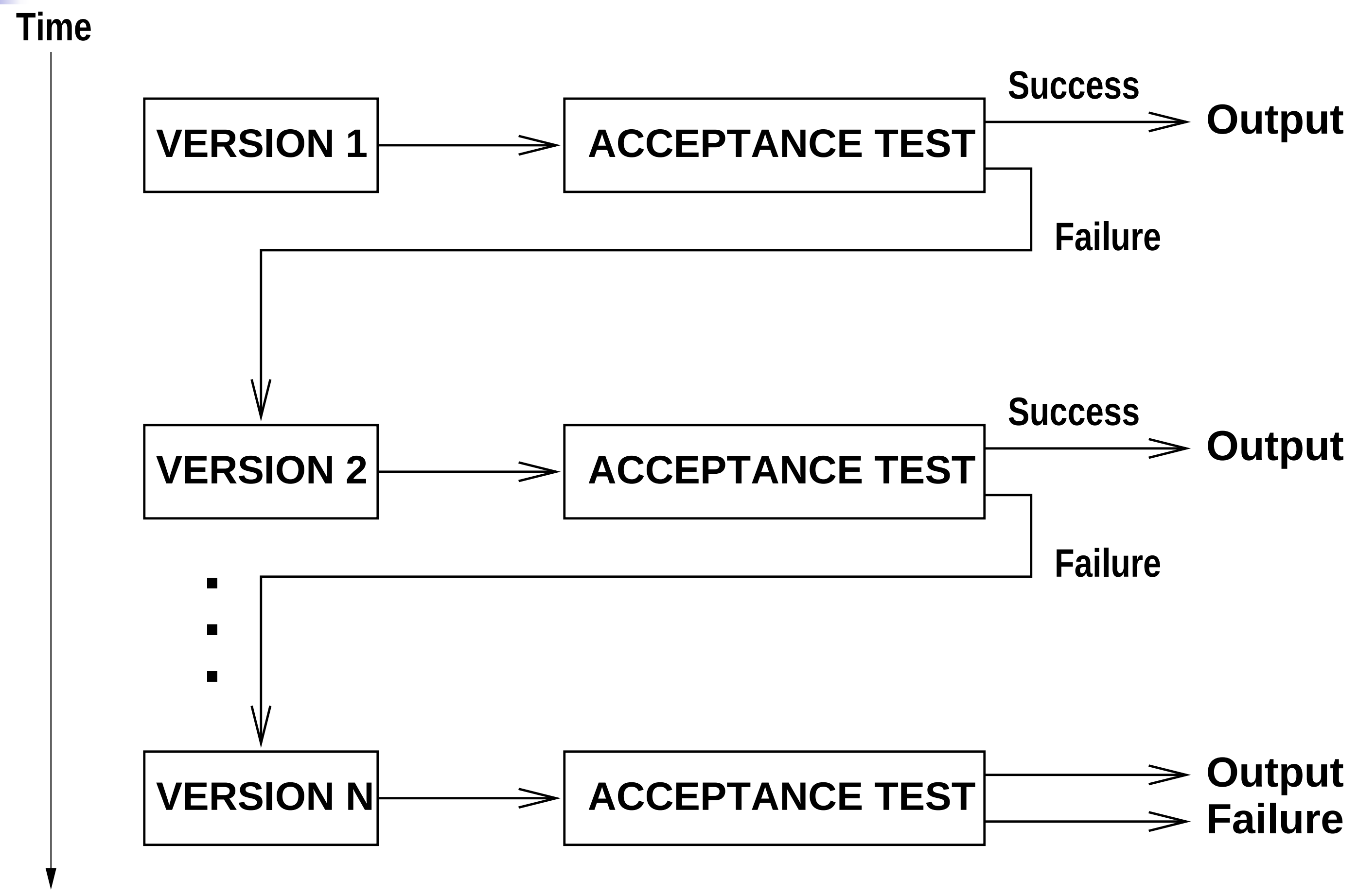
Recovery Blocks (RB)

- RB uses multiple alternates (backups) to perform the same function; one module (task) is primary and the others are secondary.

- The primary task executes first. When the primary task completes execution, its outcome is checked by an **acceptance test.**

- If the output is not acceptable, a secondary task executes after undoing the effects of primary (i.e., rolling back to the state at which primary was invoked) until either an acceptable output is obtained or the alternates are exhausted.

Recovery Blocks (Cont'd)



Recovery Blocks (Cont'd)

- The acceptance tests are usually sanity checks; these consist of making sure that the output is within a certain acceptable range or that the output does not change at more than the allowed maximum rate.
- Selecting the range for acceptance test is crucial. If the allowed ranges are too small, the acceptance tests may label correct outputs as bad. If they are too large, the probability that incorrect outputs will be accepted is more.
- RB can tolerate software faults because the alternates are usually implemented with different approaches; RB is also known as **Primary-Backup approach.**