Techniques for fault tolerance

- Fault masking “hides” faults that occur. Do not require detecting faults, but require containment of faults (the effect of all faults should be local)
- Another approach is to first detect, locate and contain faults, and then recover from faults using reconfiguration
Redundancy

- hardware redundancy
  - 2nd CPU, 2nd ALU, ...
- software redundancy
  - validation test...
- information redundancy
  - error-detecting and correcting codes, ...
- time redundancy
  - repeating tasks several times, ...

Example

- FT digital filter
  - acceptance test [0 - 255]
    - SW: detect overflow
    - HW: memory for test
    - time: to execute test
  - transients: via re-execution
    - time to re-execute
Redundancy (5)

• NOTHING FOR FREE!
• costs
  – HW: components, area, power, ...
  – SW: development costs, ...
  – information: extra HW to code / decode
  – time: faster CPUs, components
• trade-off against increase in dependability

Types of redundancy

• hardware redundancy
• information redundancy
• software redundancy
• time redundancy
HW redundancy: overview

- passive redundancy techniques
  - fault masking
- active redundancy techniques
  - detection, localisation, containment, recovery
- hybrid redundancy techniques
  - static + dynamic
  - fault masking + reconfiguration

Passive HW redundancy

Triple Modular Redundancy (TMR)

input 1 → M1
input 2 → M2
input 3 → M3

voter → output
Passive HW redundancy

• Triple Modular Redundancy (TMR)
  – 3 active components
  – fault masking by voter
• Problem: voter is a single point of failure

restoring organ
Passive HW redundancy

- N-modular redundancy (NMR)
  - N active components (N A)
  - N odd, for majority voting
  - tolerates \( \lceil N/2 \rceil \) module faults

- example Apollo
  - N=5
  - 2 faults can be tolerated (masked)

HW voting

 hardware realisation of 1-bit majority voter

\[ f = ab + ac + bc \]

n-bit majority voter: n times 1-bit

requires 2 gate delays
SW voting

- Voting can be performed using software
- voter is software implemented by a microprocessor
- voting program can be as simple as a sequence of three comparisons, with the outcome of the vote being the value that agrees with at least one on the other two

HW vs. SW Voting

- HW: fast, but expensive
  - 32-bit voter: 128 gates and 256 flip-flops
  - 1 TMR level = 3 voters

- SW: slow, but more flexible
  - use existing CPUs
Problem with voting

- Major problem with practical application of voting is that the three results may not completely agree
  - sensors, used in many control systems, can seldom be manufactured so that their values agree exactly
  - analog-to-digital converter can produce quantities that disagree in the least significant bits

Problems with voting

- (1) When values that disagree slightly are processed, the disagreement can grow larger
  - small difference in inputs can produce large differences in outputs
- (2) A single result must ultimately be produced
  - potential point where one failure can cause a system failure
How to cure problem 1

• Mid-value select technique
  – choose a value from the three which lies between the other two

How to cure problem 1

• Ignore the least-significant bits of data
  – disagreement which occurs only in the least-significant bits is acceptable
  – disagreement which affects the most-significant bits is not acceptable and must be corrected
Types of HW redundancy

• static techniques (passive)
  – fault masking

• dynamic techniques (active)
  – detection, localisation, containment and recovery

• hybrid techniques
  – static + dynamic
  – fault masking + reconfiguration

Active HW redundancy

• dynamic redundancy
  – actions required for correct result
    • detection, localization, containment, recovery
    • no fault masking
  – does not attempt to prevent faults from producing errors within the system
Active HW redundancy

• most common in applications that can tolerate temporary erroneous results
  – satellite systems - preferable to have temporary failures that high degree of redundancy

• types of active redundancy:
  • duplication with comparison
  • standby sparing
  • pair-and-a-spare
  • watchdog timer

Duplication with comparison

• Two identical modules perform the same computation in parallel and their results are compared
Duplication with comparison

• The duplication concept can only detect faults, not tolerate them
  – there is no way to determine which module is faulty

Duplication with comparison

• Problems:
  – if there is a fault on input line, both modules will receive the same erroneous signal and produce the erroneous result
  – comparator may not be able to perform an exact comparison
    • synchronisation
    • no exact matching
  – comparator is a single point of failure
Implementation of comparator

- In hardware, a bit-by-bit comparison can be done using two-input exclusive-or gates
- In software, a comparison can be implemented with a COMPARE instruction
  – commonly found in instruction sets of almost all microprocessors

Standby sparing

- One module is operational and one or more serve as stand-bys, or spares
- Error detection is used to determine when a module has become faulty
- Error location is used to determine which module is faulty
- Faulty module is removed from operation and replaced with a spare
**Standby sparing**

- M1
- M2
- ... 
- Mn

**Error detection**

N to 1 switch

**Switch**

- The switch examines error reports from the error detection circuitry associated with each module
  - if the module is error-free, the selection is made using a fixed priority
  - any module with errors is eliminated from consideration
  - momentary disruption in operation occur while the reconfiguration is performed
**Hot standby sparing**

- In hot standby sparing spares operate in synchrony with on-line module and are prepared to take over any time

```
<table>
<thead>
<tr>
<th>in</th>
<th>M1</th>
<th>err</th>
<th>err</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>spare</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>out</td>
</tr>
</tbody>
</table>
```

**Cold standby sparing**

- In cold standby sparing spares are unpowered until needed to replace a faulty module

```
<table>
<thead>
<tr>
<th>in</th>
<th>M1</th>
<th>err</th>
<th>err</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>out</td>
</tr>
</tbody>
</table>
```
+ and - of cold standby sparing

• (-) time is required to bring the module to operational state
  – time to apply power to spare and to initialize it
  – not desirable in applications requiring minimal reconfiguration time (control of chemical reactions)

• (+) spares do not consume power
  – desirable in applications where power consumption is critical (satellite)

Pair-and-a-spare technique

• Combines standby sparing and duplication with comparison

• like standby sparing, but two instead of one modules are operated in parallel at all times
  – their results are compared to provide error detection
  – error signal initiates reconfiguration
Pair-and-a-spare technique

- As long as two selected outputs agree, the spares are not used.
- If they disagree, the switch uses error reports to locate the faulty module and to select the replacement module.
Watchdog timer

- watchdog timer
  - must be reset an on a repetitive basic
  - if not reset - system is turned off (or reset)
  - detection of
    - crash
    - overload
    - infinite loop
  - frequency depends on application
    - aircraft control system - 100 msec
    - banking - 1 sec

HW redundancy: overview

- static techniques (passive)
  - fault masking
- dynamic techniques (active)
  - detection, localisation, containment, recovery
- hybrid techniques
  - static + dynamic
  - fault masking + reconfiguration
Hybrid HW redundancy

- combines
  - static redundancy
    - fault masking
  - dynamic redundancy
    - detection, location, containment and recovery
- very expensive but more FT

Types of hybrid redundancy

- Self-purging redundancy
- N-modular redundancy with spares
- Triple-duplex architecture
Self-purging redundancy

• All units are actively participate in the system
• each module has a capability to remove itself from the system if its faulty
  – very attractive feature: maintenance personnel can disable individual modules and replace them without interrupting the system
Basic structure of a switch

- If output of a module disagrees with the output of the system, its contribution to the voter is forced to be 0 (threshold voter).

N-modular redundancy with spares

- M1, M2, ..., Mn, S1, ..., Sm inputs to switch, voter, and output.
NMR with spares

- System remains in the basic NMR configuration until the disagreement vector determines a fault
- The output of the voter is compared to the individual outputs of the modules
- Module which disagrees is labeled as faulty and removed from the NMR core
- Spare is switched to replace it

NMR with spares

- The reliability is maintained as long as the pool of spares is not exhausted
- 3-modular redundancy with 1 spare can tolerate 2 faults
- To do it in a passive approach, we would need to have 5 modules
**Triple-duplex architecture**

- Combines duplication with comparison and triple modular redundancy

---

**Diagram:**

- M1a, M1b
- M2a, M2b
- M3a, M3b
- comp nodes
- voter
- out
**Triple-duplex architecture**

- TMR allows faults to be masked
  - performance without interruption
- duplication with comparison allows faults to be detected and faulty module removed from voting
  - removal of faulty module allows to tolerate future faults
- two module faults can be tolerated

**Summary**

- application-dependent choice
  - critical-computation - momentary erroneous results are not acceptable
    - passive or hybrid
  - long-life, high-availability - system should be restored quickly
    - active
  - very critical applications - highest reliability
    - hybrid
Next lecture

• Information redundancy

Read chapter 5
of the text book