Embedded Systems Engineering

System Reliability - 3

- Processor Problems
  - Power-on aspects
  - Run-time issues
- Hardware-based Fault Tolerance
  - Fault-tolerant structures
  - Matching structures to requirements

Michael Brockway
Processor Problems

- Techniques to answer these questions:
  - Is it OK to use the processor system at all?
  - Is it safe to continue using it in its normal running mode?
  - If problems are detected, can we recover?

- We want as far as possible graceful recovery from failures

- Power-on Aspects
  - CPU Tests
    - Verify processor correctly executes its range of instructions
    - Ensure condition flags can be set and read correctly
    - Check on-chip registers etc can be written and read correctly
    - Verify numeric coprocessors
  - You can write your own code to perform these test insofar as applicable to your application code
Processor Problems

● Power-on Aspects
  ■ ROM Tests
    ◆ ROM contents could become corrupted by EM pulses / radiation
    ◆ Flash can be “accidentally” reprogrammed
  ■ Write tests to check for validity on power-up

● Run-time Issues
  ■ Stack overflow
    ◆ Inhibit interrupts until current one is serviced (not always feasible)
    ◆ Service the interrupt before the next arrives (not always possible)
    ◆ Monitor with software
    ◆ Hardware-based monitoring (eg ARM)
Processor Problems

● Run-time Issues
  ■ Corruption of critical variables
    ◆ Causes by electrical noise, power supply fluctuations, …
    ◆ Use check summing
    ◆ Make a (check summed copy), compare with original
    ◆ Make 3 copies, use majority voting
  ■ Instruction pointer corruption
    ◆ PC may point to unprogrammed locations
      ♦ Fill with No-op codes
    ◆ or it may point to programmed memory locations
      ♦ More difficult to deal with
Hardware-based Fault Tolerance

operational requirements

continuous operation required

full performance needed

fail-operational

no repair possible

automated repair permissible

reduced performance acceptable

fail-active

automatic re-config

manual repair without re-config

non-continuous operation acceptable

stops & locks out safely

fail-safe

manual repair with auto re-config

manual repair with manual recovery

must be returned quickly to full service

high-availability

Automatic reconfig with automatic recovery
Hardware-based Fault Tolerance

- Fault-tolerant structures
  - Passive redundant systems

```
Sensor 1
  ↓
  ↓
  ↓
Processor 1
  ↓
  ↓
  ↓
Processor 2
  ↓
  ↓
  ↓
Processor 3
  ↓
  ↓
  ↓
Processor 4
  ↓
  ↓
  ↓
Majority voter
  ↓
  ↓
  ↓
Control actuator
```
Hardware-based Fault Tolerance

- Fault-tolerant structures
  - Active redundant systems
    - Standby processor could be running “cold” or “hot”, in the latter case perhaps fully synchronised with the active processor
Hardware-based Fault Tolerance

- Fault-tolerant structures
  - Hybrid redundant systems

![Diagram of a fault-tolerant system with a sensor, processors, switching circuit, and majority voter.]
Hardware-based Fault Tolerance

- Fault-tolerant structures
  - Very high availability systems
    - Active redundant structures
    - DB servers etc
    - Very fast “failover” times
    - Ready availability of disk data to standby systems
  - Hot-swap systems
    - Replacement of boards while system is on-line, without disturbing operations
    - CompactPCI standard: four levels of capability
      - Basic hot-swap
      - Full hotswap
      - High-availability hot-swap (single processor)
      - High-availability hot-swap (multiple processor)
Hardware-based Fault Tolerance

- Fault-tolerant structures
  - CompactPCI - Basic hot-swap
Hardware-based Fault Tolerance

- Fault-tolerant structures
  - CompactPCI - Full & high-availability (single processor) hot-swap
    - Full: interface to system software allowing it to oversee swap of boards
    - HA: system monitors/controls boards with platform management software
Hardware-based Fault Tolerance

- Fault-tolerant structures
  - CompactPCI - Full & high-availability (multiple processor) hot-swap
    - Distributed computing:
      - load balancing
      - Re-allocation of work from a failed processor
    - Fault diagnosis, repair managed by the host SBC
Hardware-based Fault Tolerance

operational requirements

continuous operation required

full performance needed

fail-operational

no repair possible 1

automated repair permissible 2

reduced performance acceptable

fail-active

automatic re-config 3

manual repair without re-config 4

manual repair with auto re-config 5

manual repair with manual recovery 6

Automatic reconfig with automatic recovery 7

non-continuous operation acceptable

stops & locks out safely

fail-safe

must be returned quickly to full service

high-availability
## Hardware-based Fault Tolerance

Matching structures to Requirements

<table>
<thead>
<tr>
<th></th>
<th>Hybrid redundancy</th>
<th>Passive redundancy</th>
<th>Active redundancy</th>
<th>High avail hot-swap</th>
<th>Full hot-swap</th>
<th>Basic hot-swap</th>
<th>Fail-safe</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>