Fault-tolerant systems – a short introduction

The improvement of reliability can be increased by two different approaches, *perfectness or tolerance*, [11.4]. Perfectness refers to the idea of avoiding faults and failures by means of an improved mechanical or electrical design. This includes the continued technical advancement of all components that increase the operational life. During operation the intactness of the component must be maintained by regular maintenance and replacement of wearing parts. Methods that facilitate fault detection at an early stage allows one to replace the regular maintenance schedule with a maintenance-on-demand scheme.

Tolerance describes the notion of trying to contain the consequences of faults and failures such that the components remain functional. This can be reached by the principle of *fault tolerance*. Herewith, faults are compensated in such a way that they do not lead to system failures. The most obvious way to reach this goal is *redundancy* in components, units or subsystems called modules. However, the overall systems then become more complex and costly. In the following, various types of fault-tolerant methods are reviewed briefly, for more details see [11.3] and [11.8].

Fault-tolerance methods generally use *redundancy*. This means that in addition to the considered module, one or more modules are connected, usually in parallel. These redundant modules are either *identical* or *diverse*. Such redundant schemes can be designed for hardware, software, information processing, and mechanical and electrical components like sensors, actuators, microcomputers, buses, power supplies, etc.

### 11.1 Basic redundant structures

There exist mainly two basic approaches for fault tolerance: static redundancy and dynamic redundancy. The corresponding configurations are first considered for *electronic hardware* and then for other components. Figure 11.1a) shows a scheme for *static redundancy*. It uses three or more parallel modules that have the same input signal and are all active. Their outputs are connected to a voter, which compares these signals and decides by majority which signal value is the correct one. If a triple
modular-redundant system is applied, and the fault in one of the modules generates a wrong output, this faulty module is masked (i.e. not taken into account) by the two-out-of-three voting. Hence, a single faulty module is tolerated without any effort for specific fault detection and \( n \) redundant modules can tolerate \((n - 1)/2\) faults (\( n \) odd).

To improve the fault tolerance also the voter can be made redundant, [11.8]. Disadvantages of static redundancy are high costs, more power consumption and extra weight. Furthermore, it cannot tolerate common-mode faults, which appear in all modules because of common fault sources.

Dynamic redundancy needs fewer modules at the cost of more information processing. A minimal configuration consists of two modules, Figure 11.1b) and c). One module is usually in operation and, if it fails, the standby or back-up unit takes over. This requires fault detection to observe if the operating modules become faulty. Simple fault-detection methods only use the output signal for, e.g. consistency checking (range of the signal), comparison with redundant modules or use of information redundancy in computers like parity checking or watchdog timers. After fault detection, it is the task of the reconfiguration to switch to the standby module and to remove the faulty one.

![Fault-tolerant schemes](image)

**Fig. 11.1.** Fault-tolerant schemes: a) static redundancy: multiple-redundant modules with majority voting and fault masking, \( m \) out of \( n \) systems (all modules are active); b) dynamic redundancy: standby module that is continuously active, “hot standby”; c) dynamic redundancy: standby module that is inactive, “cold standby”

In the arrangement of Figure 11.1b), the standby module is continuously operating, called hot standby. Then, the transfer time is small at the cost of operational aging (wear-out) of the standby module.