

---

# Grundlagen zuverlässiger fehlertoleranter Systeme



## Sicherheit von Flugzeugtechnik und Autotechnik (USA)

---

	Flugtechnik	Autotechnik
<b>Einheiten</b>	<b>10 k</b>	<b>100 Mio</b>
<b>Betriebsstunden/Jahr</b>	<b>55 Mio</b>	<b>30000 Mio</b>
<b>Kosten/Einheit</b>	<b>65 Mio</b>	<b>20 k</b>
<b>Unfälle/Jahr</b>	<b>170</b>	<b>21 Mio</b>
<b>Todesfälle/Jahr</b>	<b>350</b>	<b>42 k</b>
<b>Todesfälle/10<sup>6</sup> Betr.Std.</b>	<b>6,4</b>	<b>0,71</b>
<b>Bediener-Training</b>	<b>hoch</b>	<b>niedrig</b>
<b>Redundante Komponenten</b>	<b>alle flug- kritischen Systeme</b>	<b>Bremsen</b>

**Zuverlässigkeitsanforderungen für sicherheitskritische Systeme der Flugzeugtechnik: 10<sup>-9</sup> Fehler/h für eine Missionszeit von 10 h.**



# Kann man die Ansätze der Flugzeugtechnik übernehmen?

---

**Zu teuer.**

**Unterschiedliche Betriebsbedingung.**

**Schwer durchzusetzende Wartungsintervalle.**

**Schlecht ausgebildete Benutzer.**



# Anforderungen an die Autoelektronik

<b>Anfangszuverlässigkeit:</b>	<b>(0 km / 0 h) Fehler: <math>&lt; 500 \cdot 10^{-9}</math> im 1. Jahr Fehler: <math>&lt; 1000 \cdot 10^{-9}</math></b>
<b>System-Lebenszeit:</b>	<b>3500 h (ca. 5Jahre bei 2h/Tag)</b>
<b>Garantie:</b>	<b><math>\geq 1</math> Jahr, Ersatzteile <math>\geq 10</math> Jahre</b>
<b>Umgebungsbedingungen:</b>	<b>- 40 bis +85 °C</b>
<b>Vibration:</b>	<b>10 Hz bis 1 kHz, zufällig 5g, Sinus 2-5g</b>
<b>Shock:</b>	<b>30 g</b>
<b>Versorgungsspannung:</b>	<b>8-16V Motorstart mit 6V (-40 bis +85 °C), 18V für 2h, 24V für 1 min umgekehrte Polarität 13,5V für 1 min</b>

## Streßtest für Autoelektronik

- **Funktionstest:** 8, 13.5, 16 V bei -40, 25, 85 °C
- **Hitzetest:** 85 ± 2 °C für 16h bei 16 V und 6000 upm
- **Kältetest:** - 40 ± 3 °C für 2h
- **Lagerung:** 85 ± 2 °C für 504h
- **Temperaturschock:** - 40 bis 85 °C Übergang in 30 sek. 25 mal
- **Temperaturänderung:** - 40 bis 85 °C , 3 ± 0.6 °C /min für 2 Zyklen



# Definitionen:

---

## **Verlässlichkeit (Dependability): "Doing the right thing at the right time!"**

Die **Verlässlichkeit (Dependability)** eines Systems ist die Qualität einer vom System erbrachten Funktion (Service), in die begründbar und berechtigterweise Vertrauen (reliance) gesetzt werden kann.

Die **Funktion (Service)** ist das an der Schnittstelle zu anderen Systemen, die mit dem betrachteten System interagieren, beobachtbare Systemverhalten. Die **Qualität** bezieht sich auf die Übereinstimmung der erbrachten mit der spezifizierten Systemfunktion.

## **Fragestellungen:**

- **Fehler:** Welche Klassen von Fehlern werden berücksichtigt?
- **Attribute:** Welche Aspekte der Verlässlichkeit werden besonders hervorgehoben?
- **Maße:** Wie läßt sich die Verlässlichkeit quantitativ erfassen?

Laprie, J.-C. : Dependability: A unifying concept for reliable, safe, secure computing.  
In IFIP Congress, volume 1, (1992)pages 585-593.



# Aspekte der Verlässlichkeit

**Überlebensfähigkeit (Reliability)** bedeutet Zuverlässigkeit in Hinblick auf ununterbrochenes korrektes Systemverhalten. Es ist als die Wahrscheinlichkeit definiert, daß ein zu Beginn fehlerfreies System bis zu einem bestimmten Zeitpunkt fehlerfrei bleibt.

**Verfügbarkeit (Availability)** bedeutet Zuverlässigkeit in Hinblick auf die momentane Bereitschaft eines Systems zur Erbringung eines Service. Verfügbarkeit wird als quantitatives Maß definiert, das die Ausfalldauer zu der Dauer korrekten Systemverhaltens in Beziehung setzt, d.h. die Wahrscheinlichkeit, das System zu einem beliebigen Zeitpunkt fehlerfrei anzutreffen.

**Prozeßsicherheit (Safety)** bedeutet Zuverlässigkeit in Hinblick auf das Verhindern katastrophaler Auswirkungen eines Systemverhaltens auf seine Umgebung, wobei mit Umgebung meist die physikalische, reale Umgebung gemeint ist, wie z.B. industrielle Prozeßsteuerungsanlagen, Kraftwerke, Verkehrslenkungssysteme, u. s. w.

**Informationssicherheit (Security)** bedeutet Zuverlässigkeit in Hinblick auf die Erhaltung der Vertraulichkeit (Confidentiality) und Integrität (Integrity) von Information in einem Computersystem.

der Ansatz in der  
Flugzeugindustrie

der Ansatz in der  
Autoindustrie

der Ansatz in der  
industriellen  
Automatisierung



# Wie häufig fallen Komponenten aus?

---

$\lambda$ : Ausfälle/ $10^6$  Betriebsstunden (~115 Jahre)

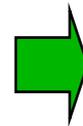
Militärischer Microprozessor	0,022	(Daten von 1987)
Automotiver Microprozessor	0,12	
Elektromotor	2,17	
Bleibatterie	16,9	
Ölpumpe	37,3	

zum Vergleich:

einzelner menschlicher Operateur:  $100/10^6$  Aktionen

Mensch in der Krisenbewältigung:  $300000/10^6$  Aktionen

$1 \times 10^{-9}$  Ausfälle/h in der Flugtechnik  
 $\sim 1 \times 10^{-9}$  Ausfälle/h in der Autotechnik



**Basiszuverlässigkeit unzureichend**



# Mechanismen zur Verlässlichkeit

---

## Fehlervermeidung Fehlertoleranz

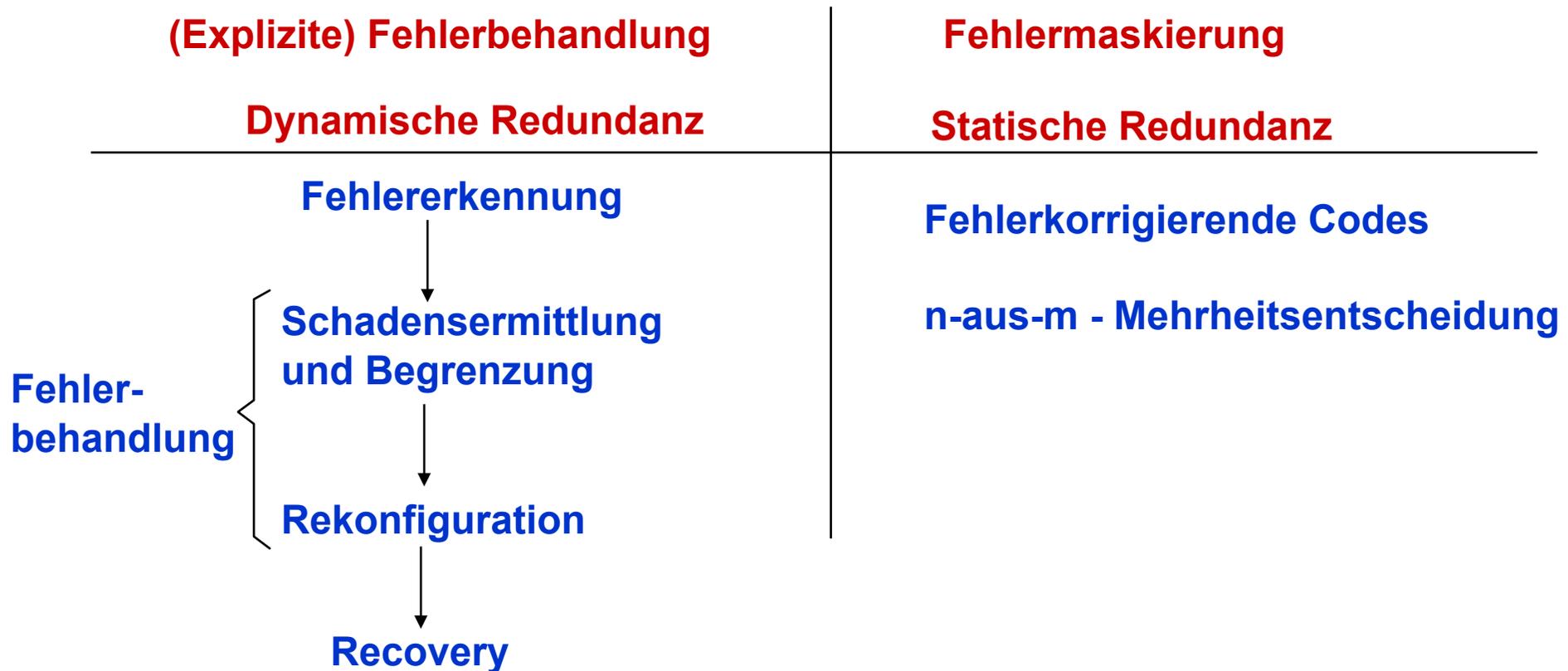
**Alle Mechanismen der Fehlertoleranz beruhen auf Redundanz**

- Informationsredundanz
- Komponentenredundanz
- Zeitredundanz

**Die Wahl der Redundanzmethode ist davon abhängig, welche Fehlerklasse berücksichtigt wird!**



# Mechanismen der Fehlertoleranz



# Fehlerklassifizierung

kann prinzipiell nicht vollständig vermieden werden.

**Fehlerursache (Fault)**

Ausfall einer physischen Komponente, fehlerhaftes Programm(statement)

Methoden der Fehlervermeidung

bewirkt möglicherweise eine fehlerhafte Änderung des Zustands

muß behandelt werden. Der fehlerhafte Zustand muß in einen fehlerfreien Zustand zurückgesetzt werden.

**Fehler (Error)**

fehlerhafter Zustand, z.B. Speicherinhalt, Registerinhalt

Methoden der Fehler-toleranz

bewirkt möglicherweise eine Abweichung vom spezifizierten Verhalten

kann nicht toleriert werden, sondern muß unter allen Umständen vermieden werden.

**Funktionsausfall (Failure)**

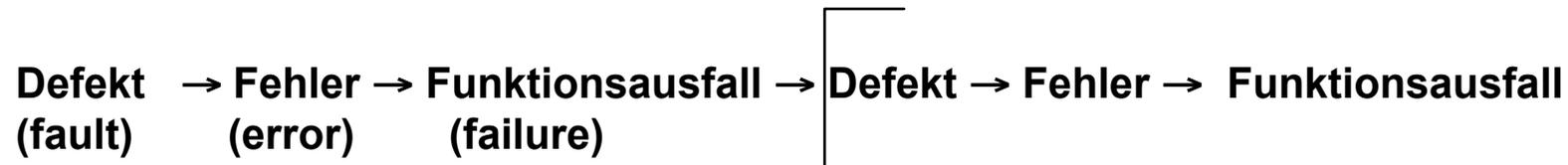
Abweichung vom spezifizierten Systemverhalten

Kann vom System nicht mehr behandelt werden.

Äußerer Eingriff, Katastrophe



# Ursache - Wirkung von Fehlern



**Defekt:** Ereignis (Fehlerursache)  
**Fehler:** Auswirkungen der Fehlerursache auf den Systemzustand  
**Funktionsausfall:** Abweichung des Systems von seinem spezifizierten Verhalten

## Defekt → Fehler

- ein Defekt, der durch den Berechnungsvorgang (noch) nicht aktiviert wurde, heißt *ruhend (dormant)*.
- Ein Defekt ist *aktiv*, wenn er einen Fehler verursacht.

## Fehler → Funktionsausfall

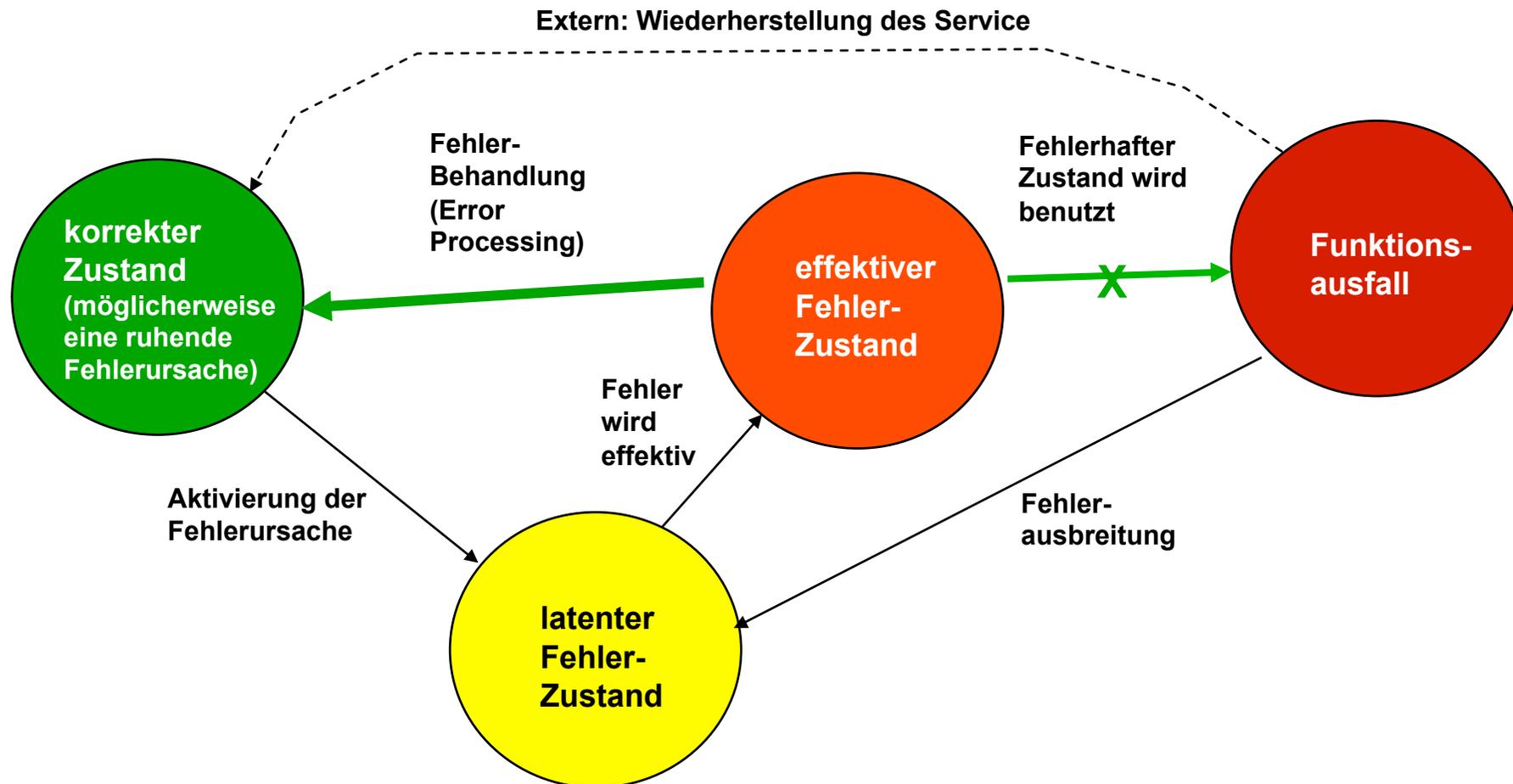
- ein Fehler heißt *latent*, wenn er noch nicht zu einem Funktionsausfall geführt hat (oder noch nicht durch Erkennungsmaßnahmen entdeckt wurde).
- ein Fehler heißt *effektiv*, wenn er zu einem Funktionsausfall führt.

## Funktionsausfall → Defekt

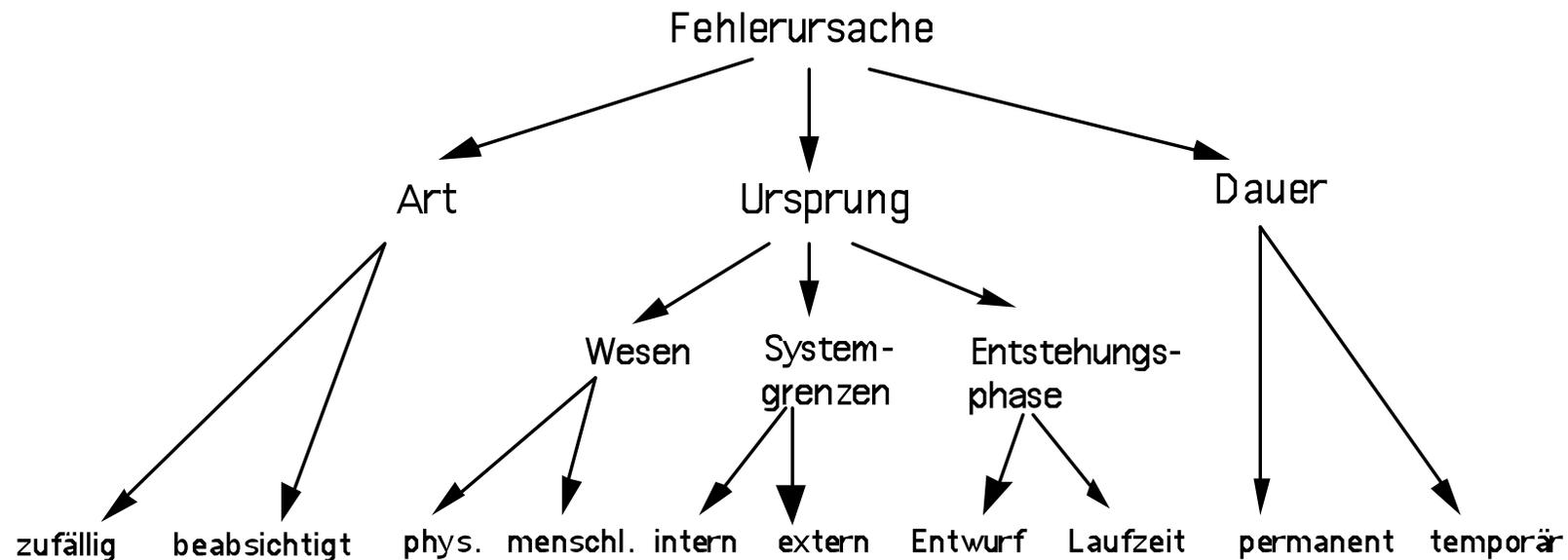
- ein Funktionsausfall tritt ein, wenn ein Fehler effektiv wird und den erbrachten Service verfälscht.
- ein Funktionsausfall kann die Fehlerursache für eine höhere Systemebene darstellen.



# Zustandsdiagramm der Fehlerursache-Fehler-Funktionsausfall -Kette



# Fehlerklassifizierung nach der Fehlerursache



# Beispiel: Physische Fehlerursachen

Ursprung des Fehlers: physikalisches Ereignis								
Art		Systemgrenzen		Entstehungsphase		Dauer		Bezeichnung
•		•			•	•		permanenter phys. Fehler
•		•			•		•	intermittierender Fehler
•			•		•		•	transienter Fehler

**Typische permanente Fehler: Stuck-at- {0,1}, Stuck-together.**

**Typische intermittierende Fehler: Muster-abhängige Fehler, Temperatur- und Zeitfehler.**

**Typische transiente Fehler:  $\alpha, \beta, \gamma$ -Teilchen, temperaturabhängige Fehler.**



# Beispiel: Unbeabsichtigte menschliche Fehlerursachen

---

Ursprung des Fehlers: menschliches Fehlverhalten								
Art		Systemgrenzen		Entstehungsphase		Dauer		Bezeichnung
		intern	extern	Entwurf	Laufzeit	permanent	temporär	
•		•		•		•		Entwurfsfehler
•			•		•		•	Bedienungsfehler



# Beispiel: Beabsichtigte menschliche Fehlerursachen

Ursprung des Fehlers: menschliches Fehlverhalten								
Art		Systemgrenzen		Entstehungsphase		Dauer		Bezeichnung
		intern	extern	Entwurf	Laufzeit	permanent	temporär	
	•		•		•	•		Intrusion
	•		•		•		•	
	•	•			•	•		Virus
	•	•		•	•	•		Trojanisches Pferd
	•	•		•		•		maliziöse Logik

## Problem der Informationssicherheit !



# Möglichkeiten der Fehlererkennung

---

**Diagnostische Tests:** Überprüfen einzelne Komponenten des Systems, indem sie aus der Struktur der Komponenten Eingabewerte ableiten, die bestimmte Fehler in eine Komponente aktivieren und zum Ausgang propagieren.

**Code Tests:** Basieren auf fehlererkennenden Codes.

**Timing Tests:** Überprüfen der bekannten Zeitbedingungen.

**Replikations-Tests:** Mehrfache Ausführung einer Berechnung und Vergleich der Ergebnisse.

**Reversive-Tests:** Umkehrung der Berechnung. Aus den Ergebnissen werden die Eingaben abgeleitet verglichen.

**Plausibilitäts-Tests:** überprüfen Ergebnisse auf ihre Plausibilität hinsichtlich der beabsichtigten Nutzung.

**Strukturelle Tests:** Überprüfen die Struktur von Datenstrukturen.



# Testen permanenter Hardwarefehler

---

**Was testen?** Chips auf einem Board, chipinterne Komponenten.

**Wie testen?**

**Generelle Probleme:** Komplexe Schaltungen erfordern ausführliche Tests. Große Mengen von Testdaten, Probleme mit sequentiellen Schaltungen, Auswertung schwierig.

**Probleme auf Board-Ebene:** Isolation von Chips vom Rest der elektronischen Schaltungen, Anlegen von Testmustern an die Pins der Testkandidaten.

**Probleme auf Chip-Ebene:** Isolation von Komponenten, Anlegen von Testmustern an die internen Strukturen, eklatant schlechtes Verhältnis von verfügbaren Pins zu Anzahl der internen Komponenten.

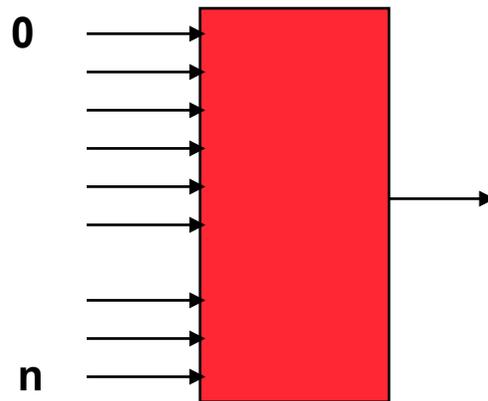
**Problem verschärft sich durch "System-on-a-Chip" Technik.**



# Problem des ausführlichen Testens

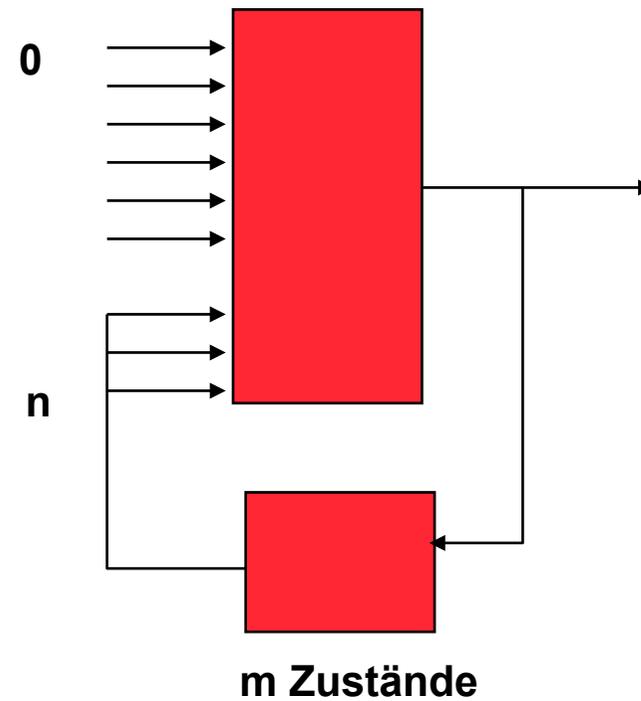
---

**Kombinatorische  
Schaltung**



**$2^n$  Testmuster**

**Sequentielle  
Schaltung**

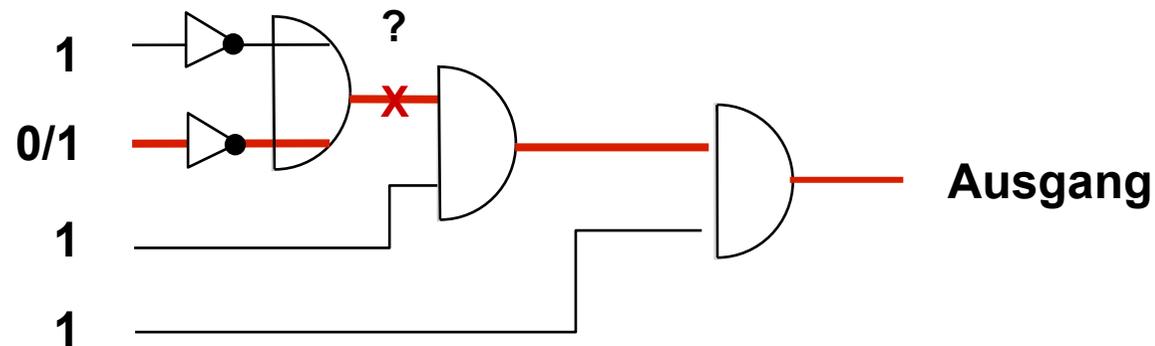


**$2^{n+m}$  Testmuster**



# Aktivierung eines Fehlers

---



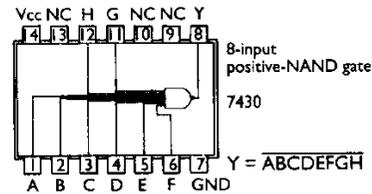
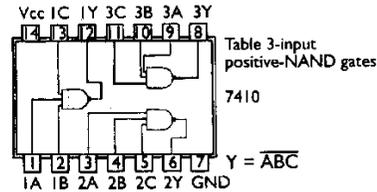
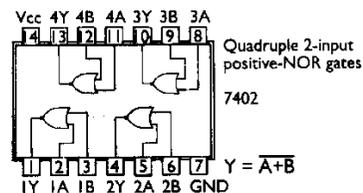
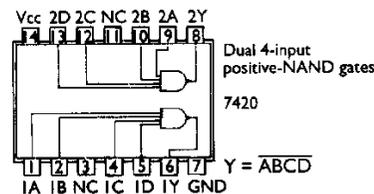
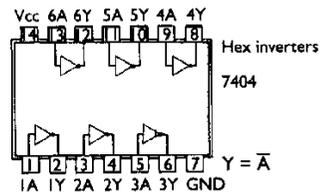
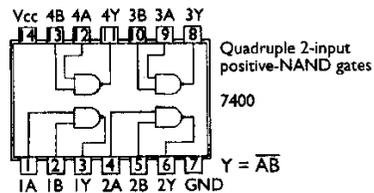
**Pfadsensitivierung:**

**Anlegen eines Musters, das es erlaubt, einen bestimmten Fehler zu aktivieren und auf einem Pfad zum Ausgang zu propagieren.**



# Problem: Komplexe interne Struktur, wenige externe Verbindungen (PINS)

1980



3 Pins/Gatter

2002



> 0,0001 Pins/Gatter

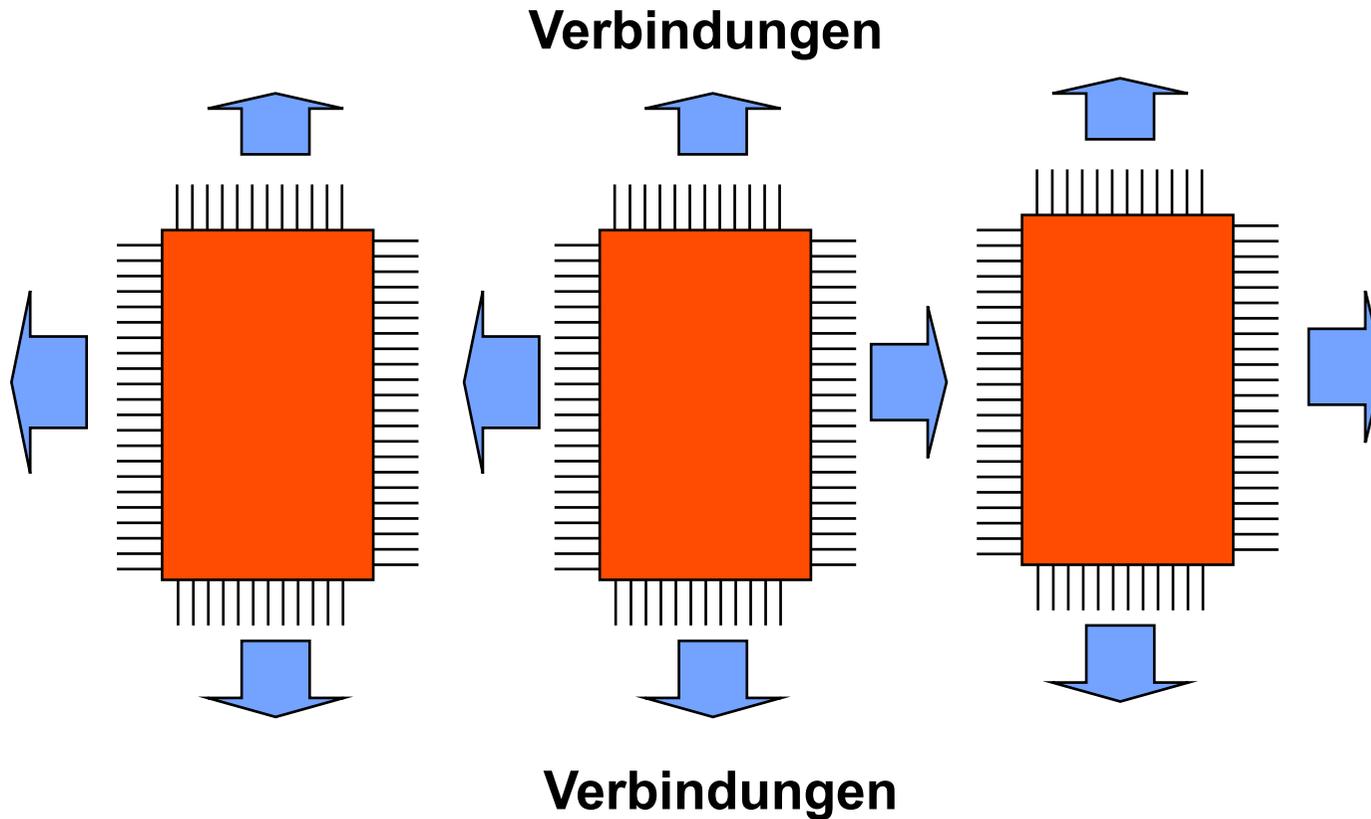
**Kontrollierbarkeit:** Wie kann man einen bestimmten Zustand herbeiführen, damit der Fehler aktiviert wird?

**Beobachtbarkeit:** Wie kann man einen Fehler zum Ausgang propagieren?



# Boundary Scan: Der JTAG Standard

(IEEE 1149)



**Wie isoliert man einen Chip für den Test?  
Wie bringt man Testmuster an die Eingänge?**



# Verfahren zur Testunterstützung

---

## **Boundary Scan:**

**Isolation von Chips auf dem Board,  
Heranführen beliebiger Testmuster an  
den Chip, Auslesen der Ergebnisse**

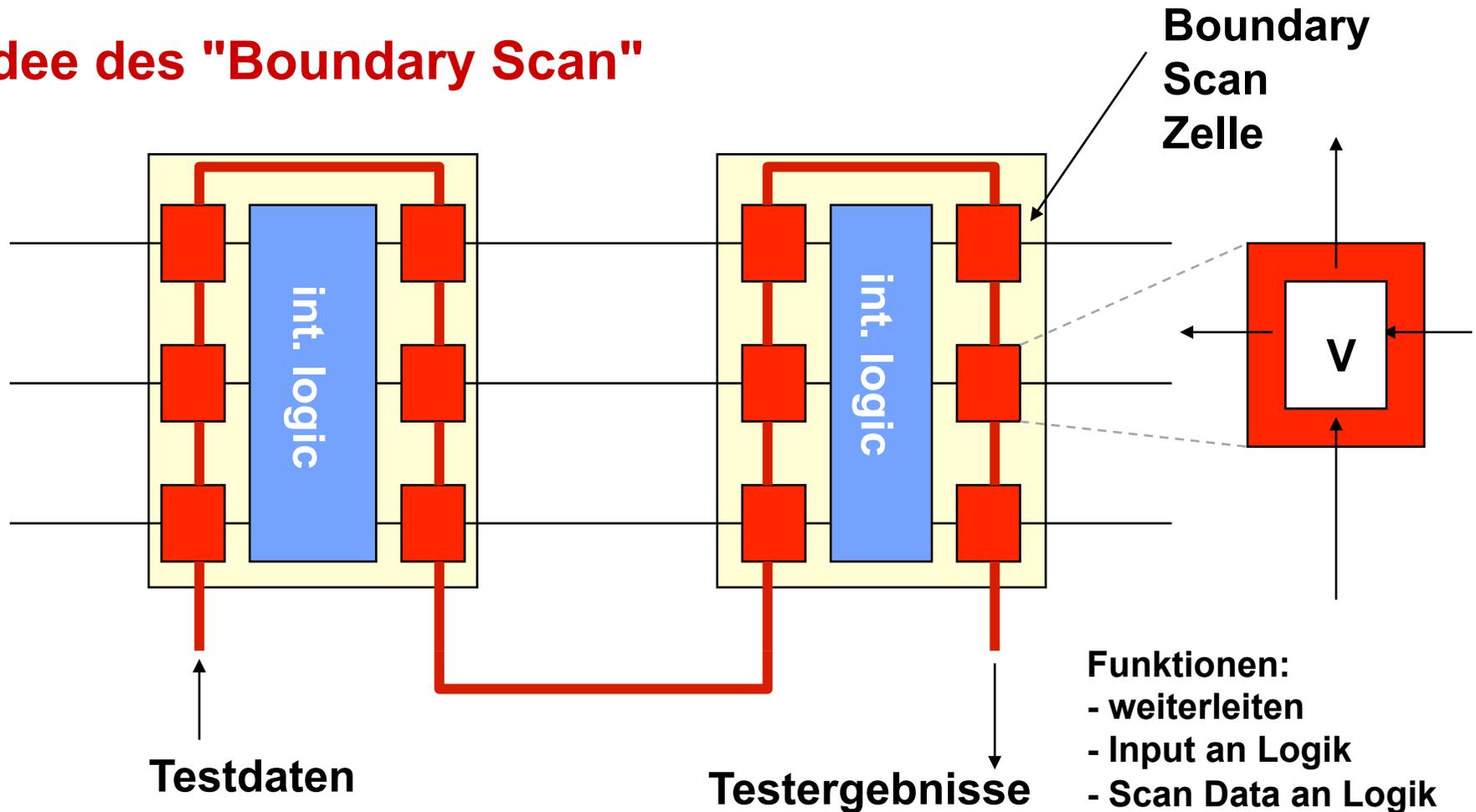
## **Scan Path Design:**

**Heranführen beliebiger Testmuster an  
interne Komponenten auf dem  
Chip, Auslesen der Ergebnisse**



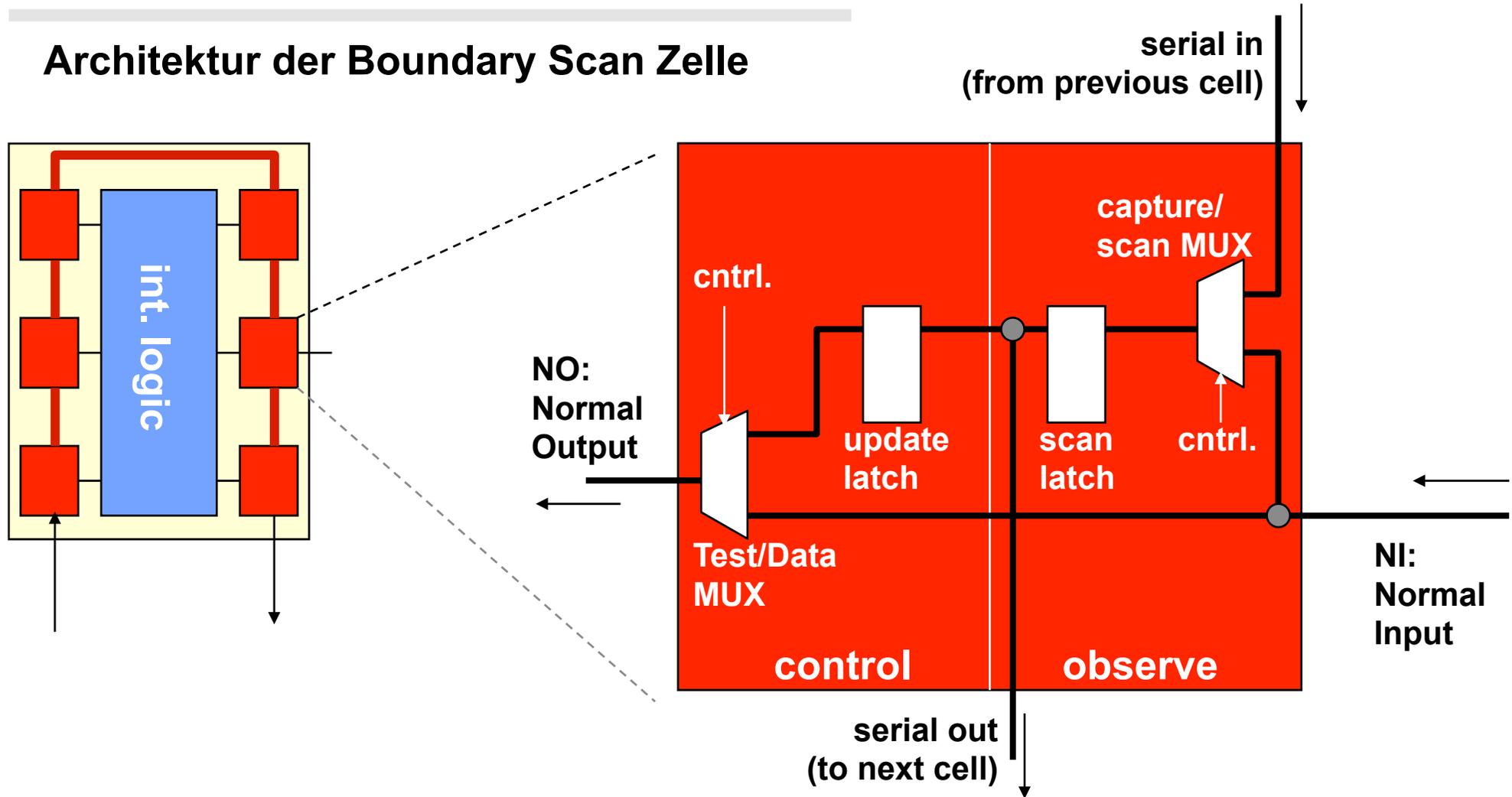
# Der JTAG Standard (IEEE 1149)

## Die Idee des "Boundary Scan"

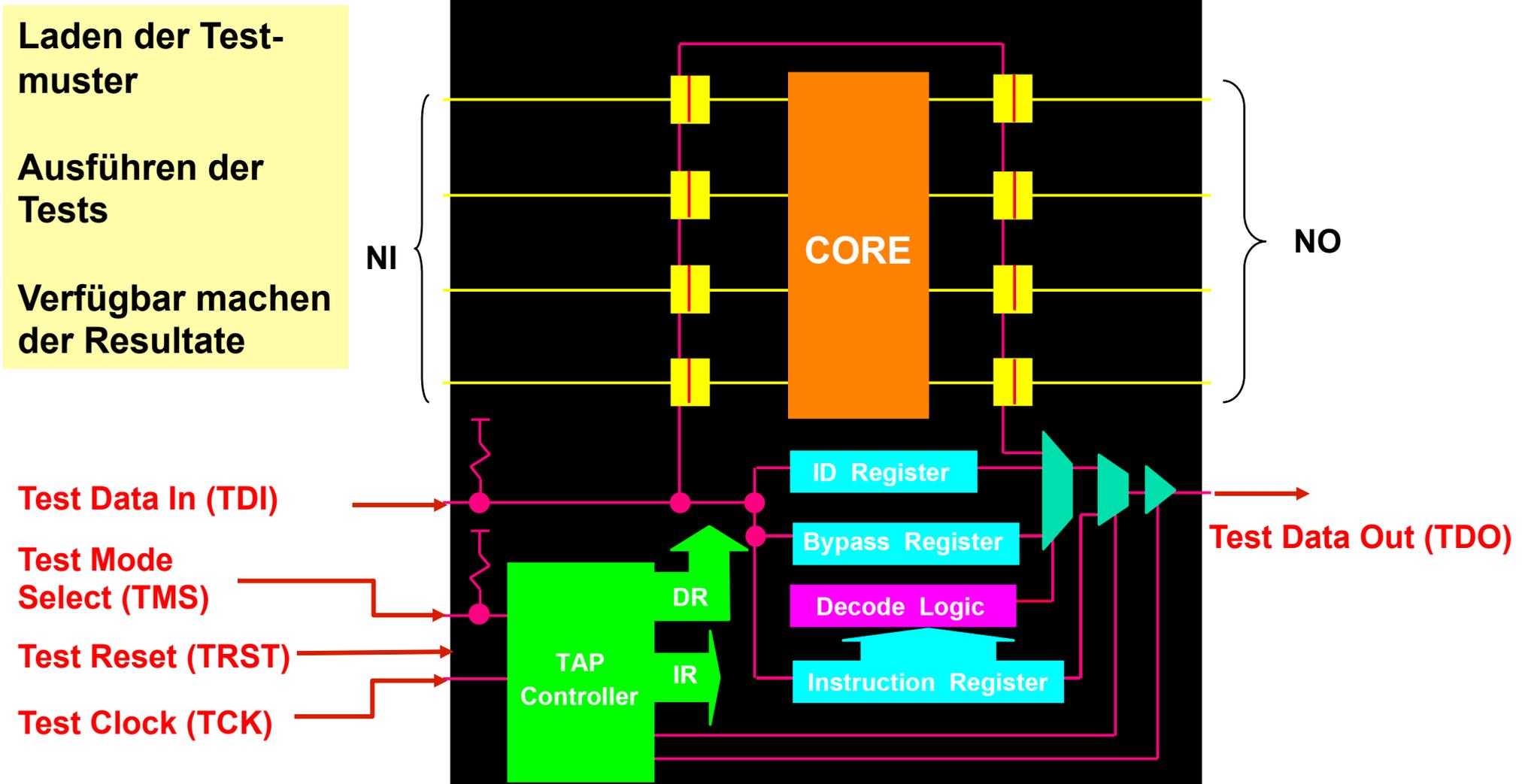


# Der JTAG Standard (IEEE 1149)

## Architektur der Boundary Scan Zelle



# Die Architektur der JTAG Kontrolleinheit



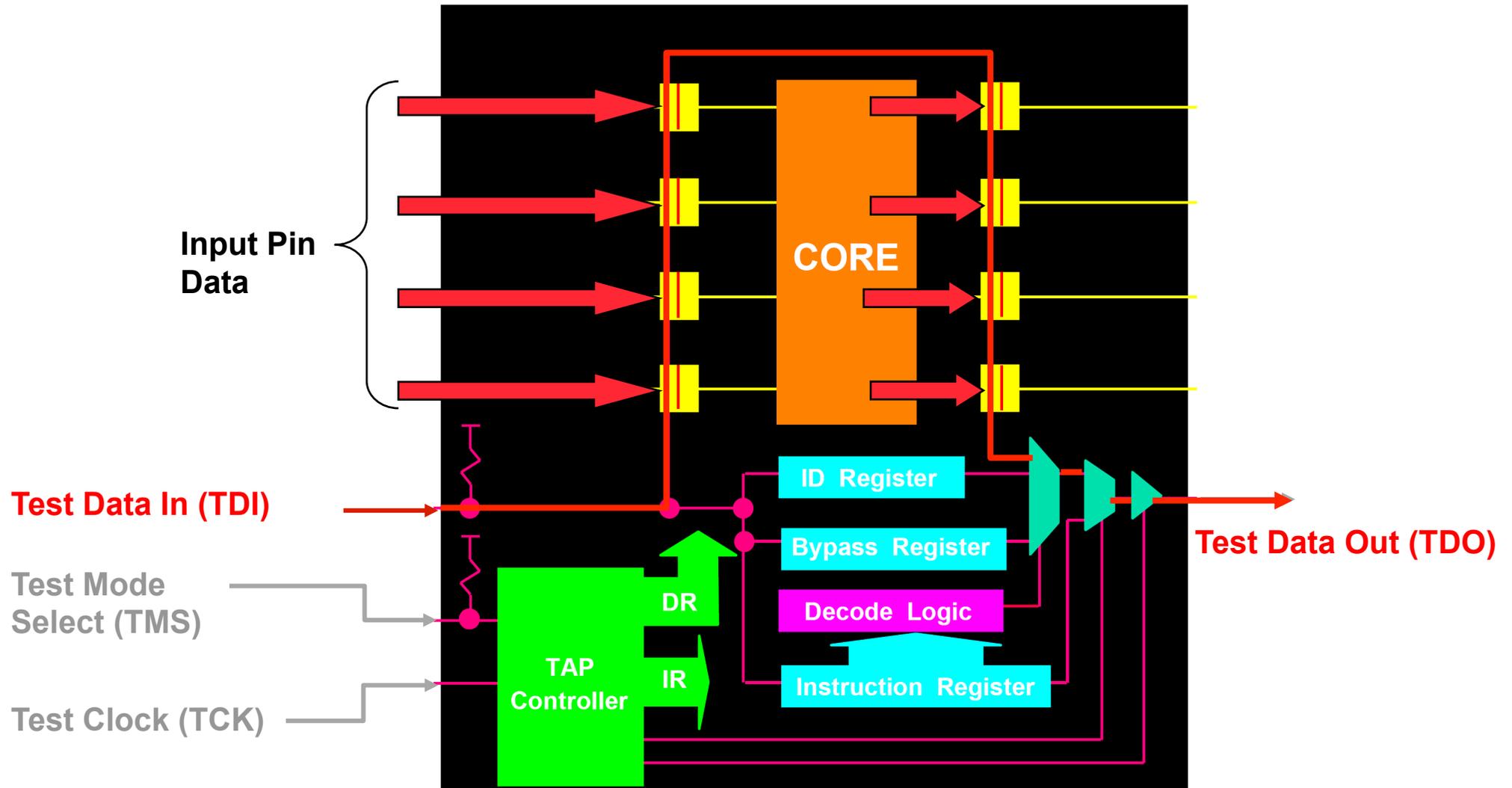
# JTAG Befehle

---

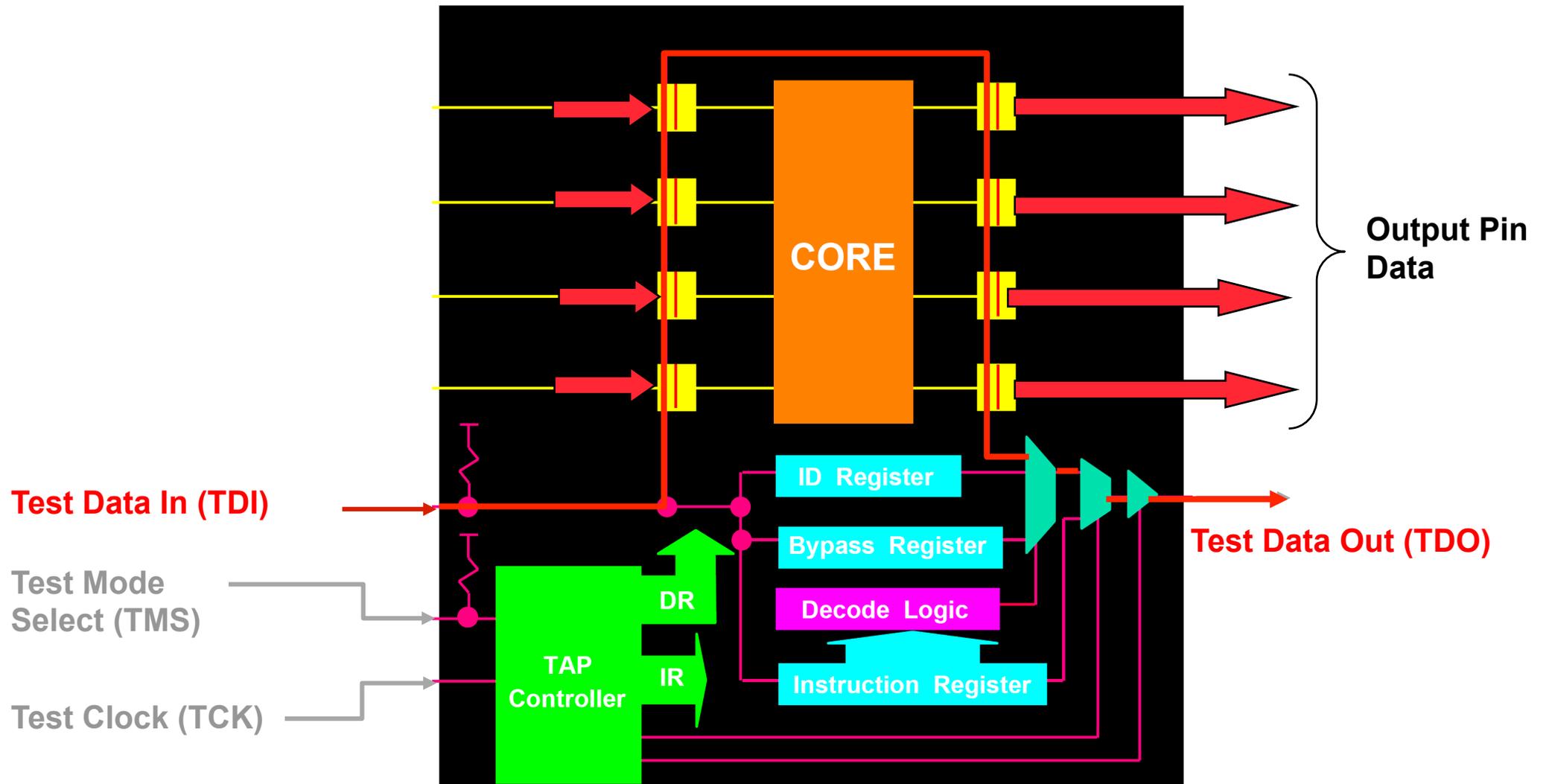
- 0000 = EXTEST:** Bei einem EXTEST werden TDI und TDO mit dem Boundary Scan Register (BSR) verbunden. Der Zustand der Pins wird geladen (sample) durch die "capture dr" Instruktion. Neue Werte können durch im "shift dr" Zustand in das BSR geladen werden. Diese Werte werden dann im Zustand "update dr" an die Pins gelegt.
- 0010=SAMPLE\_** Zu Beginn dieses Modus werden alle Inhalt der I/O-Pins in das BSR kopiert.  
**PRELOAD:** Danach wird das BSR mit TDI und TDO verbunden. Dadurch lassen sich die Daten durch TDO auslesen. Gleichzeitig ist es auch möglich, durch TDI neue Daten in die (Boundary Scan Cell) BSC zu schreiben und damit einen EXTEST oder INTEST vorzubereiten.
- 1111=BYPASS:** Beim BYPASS wird ein 1-Bit-Schieberegister zwischen TDI und TDO geschaltet. Dadurch werden die Daten, welche an TDI anliegen, mit nur 1 Takt Verzögerungen an TDO weitergeleitet.
- 0001=IDCODE:** In diesem Modus wird das ID-Register des Chips mit TDO verbunden Durch die ID lassen sich Informationen über Baureihen spezifische Eigenschaften enthalten.
- 0011=INTEST:** Im Gegensatz zum EXTEST wird hier die interne Chiplogik getestet. Zuerst wird die Logik deaktiviert, danach wird wie beim EXTEST der Inhalt der BSC an die Pins kopiert. Nachdem die Logik kurz aktiviert wurde, werden die Daten der Pins wieder in das BSC kopiert und können dann durch SAMPLE\_PRELOAD ausgelesen werden. Dieser Modus wird zwar durch den Standard definiert, ist jedoch nicht zwingend vorgeschrieben.



# Die "Sample/Preload" Instruktion

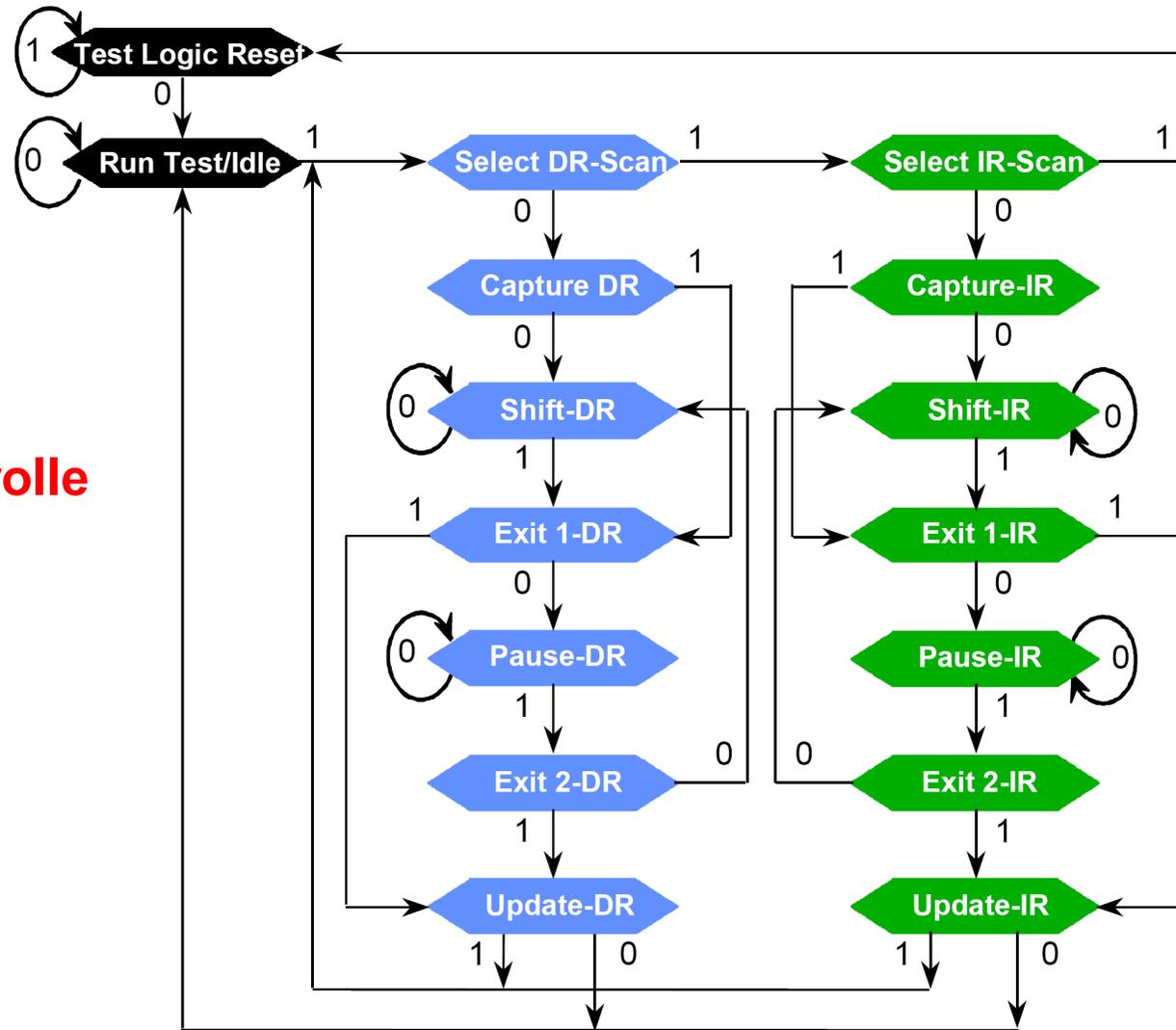


# Die "Exttest" Instruktion



# Test Access Port (TAP) Controller

TMS- Eingang:



Zustandsautomat zur Kontrolle der Testfunktionen





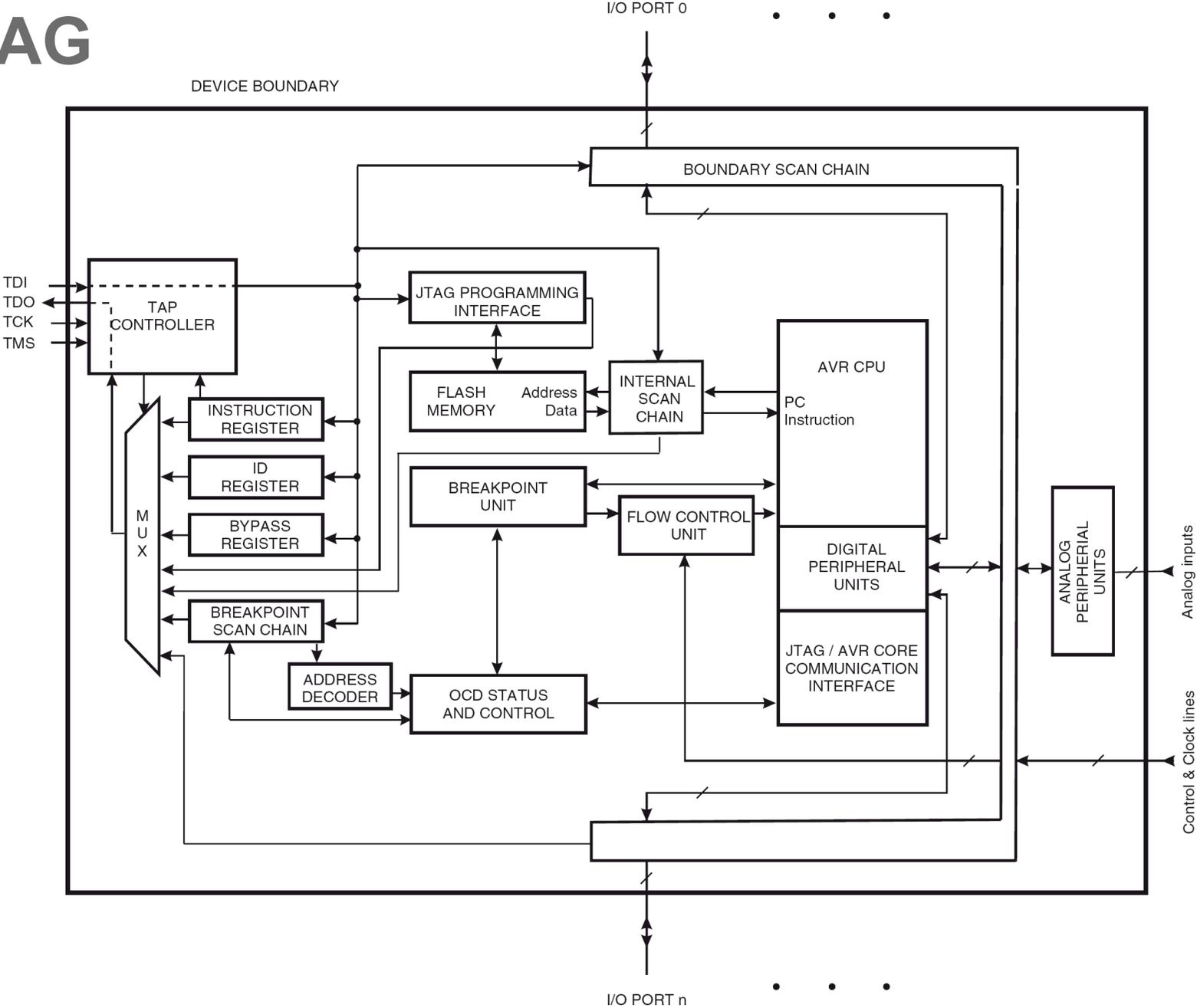
# AVR: JTAG Interface and On-chip Debug System

---

- **JTAG (IEEE std. 1149.1 Compliant) Interface**
- **Boundary-scan Capabilities According to the IEEE std. 1149.1 (JTAG) Standard**
- **Debugger Access to:**
  - All Internal Peripheral Units
  - Internal and External RAM
  - The Internal Register File
  - Program Counter
  - EEPROM and Flash Memories
- **Extensive On-chip Debug Support for Break Conditions, Including**
  - AVR Break Instruction
  - Break on Change of Program Memory Flow
  - Single Step Break
  - Program Memory Breakpoints on Single Address or Address Range
  - Data Memory Breakpoints on Single Address or Address Range
- **Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface**
- **On-chip Debugging Supported by AVR Studio**



# AVR: JTAG



# Quantitative Ermittlung der Zuverlässigkeit

---

## Strukturbasierte Modellierung:

- **identifizierbare unabhängige Komponenten**
- **jede Komponente besitzt eine bestimmte Zuverlässigkeit**
- **die Konstruktion des Modells basiert auf der Verbindungsstruktur zwischen den Komponenten**

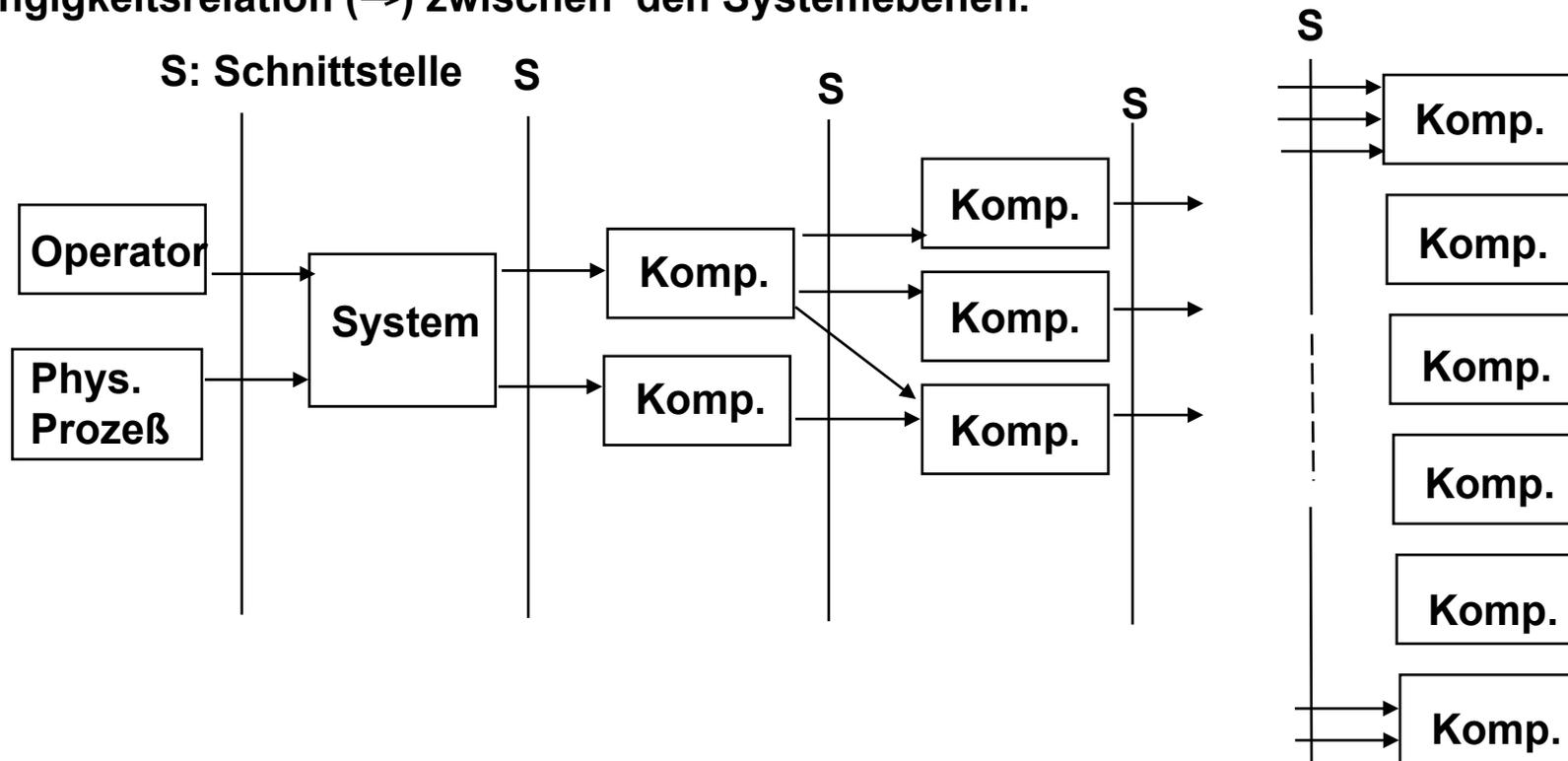


# Systemmodellierung

Ein System wird definiert durch:

- seine Struktur, d.h. die Topologie seiner Komponenten
- sein Verhalten, d.h. durch die Gesamtheit des Verhaltens seiner Komponenten

Systemkomponenten sind hierarchisch organisiert. Dadurch ergibt sich eine Abhängigkeitsrelation ( $\rightarrow$ ) zwischen den Systemebenen.



# Zuverlässigkeitsschaltbilder

Intaktwahrscheinlichkeiten:

Für jeden Teil eines Systems werden zwei Zustände betrachtet:

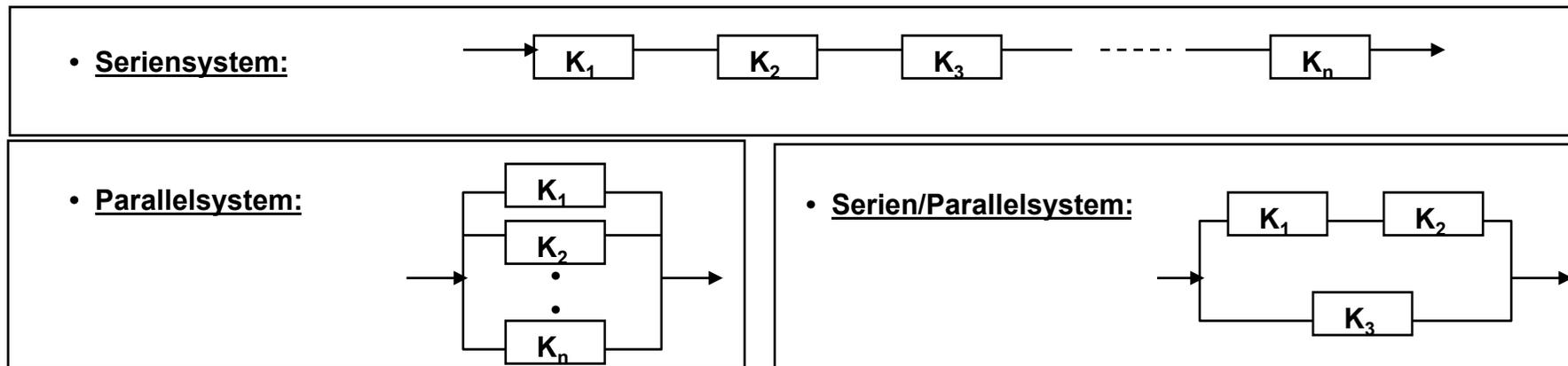
- **intakt** (funktionstüchtig)
- **defekt** (ausgefallen)

**Intakt-Wahrscheinlichkeit** einer Komponente oder eines Systems von Komponenten:  
Wahrscheinlichkeit, dass die Komponente oder das System das spezifizierte Verhalten zeigen.

Ein System ist **fehlertolerant**, wenn es intakt sein kann, ohne dass alle Komponenten intakt sind.

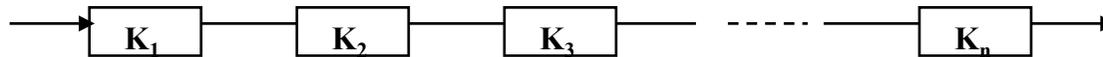
**Zuverlässigkeits-Schaltbilder** (nicht zu verwechseln mit elektrischen Schaltbildern!):

Abstraktion eines Systems in Komponenten,  
denen jeweils eine spezifische Zuverlässigkeit zugeordnet wird.



# Intaktwahrscheinlichkeit für ein Seriensystem:

---



$$P_{\text{serie}} = P(K_1 \text{ intakt}) \wedge P(K_2 \text{ intakt}) \wedge \dots \wedge P(K_n \text{ intakt})$$

Annahme: Die Eigenschaften  $(K_i \text{ intakt})$  ( $i=1, \dots, n$ ) sind unabhängig.

➔ 
$$P_{\text{serie}} = P(K_1 \text{ intakt}) \cdot P(K_2 \text{ intakt}) \cdot \dots \cdot P(K_n \text{ intakt})$$

mit  $p_i$  : Intaktwahrscheinlichkeit der Komponente  $i$ :

➔ 
$$P_{\text{serie}} = p_1 \cdot p_2 \cdot \dots \cdot p_n$$

Beispiel:

$n$  identische Komponenten:

$$P_{\text{serie}} = p_i^n, \quad n = 5, \quad p_i = 0,99: \quad P_{\text{serie}} = 0,99^5 = 0,95$$

$$P_{\text{serie}} = p_i^n, \quad n = 5, \quad p_i = 0,70: \quad P_{\text{serie}} = 0,70^5 = 0,16$$



# Intaktwahrscheinlichkeit für ein Parallelsystem (1-aus-n)

**Defektwahrscheinlichkeit = 1 - Intaktwahrscheinlichkeit**

(Intakt und defekt sind zueinander komplementäre Ereignisse).

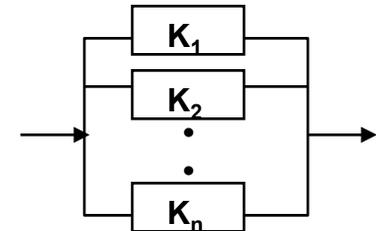
$$P_{\text{parallel}} = P(K_1 \text{ defekt}) \text{ und } P(K_2 \text{ defekt}) \text{ und } \dots P(K_n \text{ defekt})$$

**Annahme: Die Eigenschaften ( $K_i$  defekt) ( $i=1, \dots, n$ ) sind unabhängig.**

➔  $P_{\text{parallel}} = P(K_1 \text{ defekt}) \cdot P(K_2 \text{ defekt}) \cdot \dots \cdot P(K_n \text{ defekt})$

**Mit  $p_i$  : Defektwahrscheinlichkeit der Komponente  $i$ :**

➔  $P_{\text{parallel}} = p_1 \cdot p_2 \cdot \dots \cdot p_n$

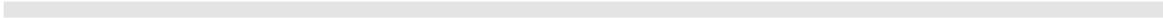
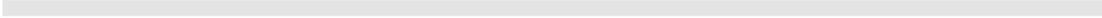


**Beispiel Defektwahrscheinlichkeit:**

**n identische Komponenten:**

$$P_{\text{serie}} = p_i^n, \quad n = 5, \quad p_i = 1 - 0,99: \quad P_{\text{serie}} = 0,01^5 = 0,0000000001 \quad \text{Intaktw.: } 0,9999999999$$
$$P_{\text{serie}} = p_i^n, \quad n = 5, \quad p_i = 1 - 0,70: \quad P_{\text{serie}} = 0,30^5 = 0,00243 \quad \text{Intaktw.: } 0,99757$$





# K - aus - n - Systeme

---

Systeme aus n Komponenten von denen mindestens k der Komponenten intakt sind.

Wahrscheinlichkeit, daß genau k ausgewählte Komponenten intakt  
(die Komponenten 1,...,k), die anderen Komponenten defekt sind  
(die Komponenten k+1,...,n).

$$P_{k\text{-aus-}n} = p_1 \cdot p_2 \cdot \dots \cdot p_k \cdot (1 - p_{k+1}) \cdot (1 - p_{k+2}) \cdot \dots \cdot (1 - p_n)$$

Es gibt  $\binom{n}{i}$  Möglichkeiten, i Komponenten aus n Komponenten auszuwählen:

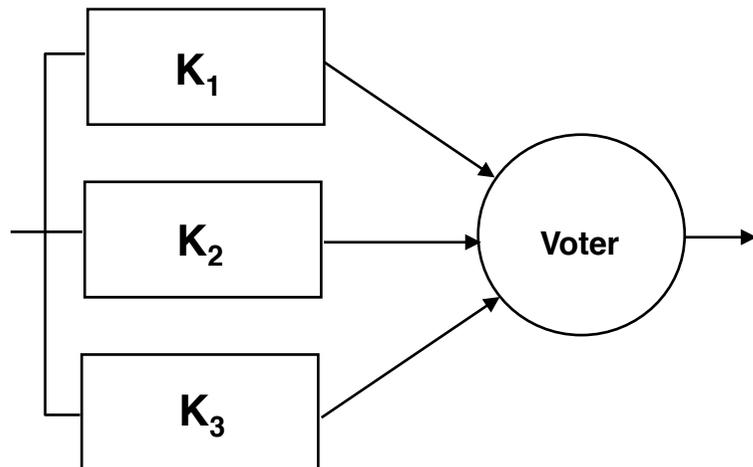
$$P_{k\text{-aus-}n} = \sum_{i=k}^n \binom{n}{i} p^i \cdot (1 - p)^{n-i}$$

Beisp.: ein 2-aus-3 System:  $\binom{3}{2} p^2 \cdot (1 - p)^{3-2} + \binom{3}{3} p^3 \cdot (1 - p)^{3-3} = 3 \cdot p^2 \cdot (1 - p) + p^3 \cdot 1$

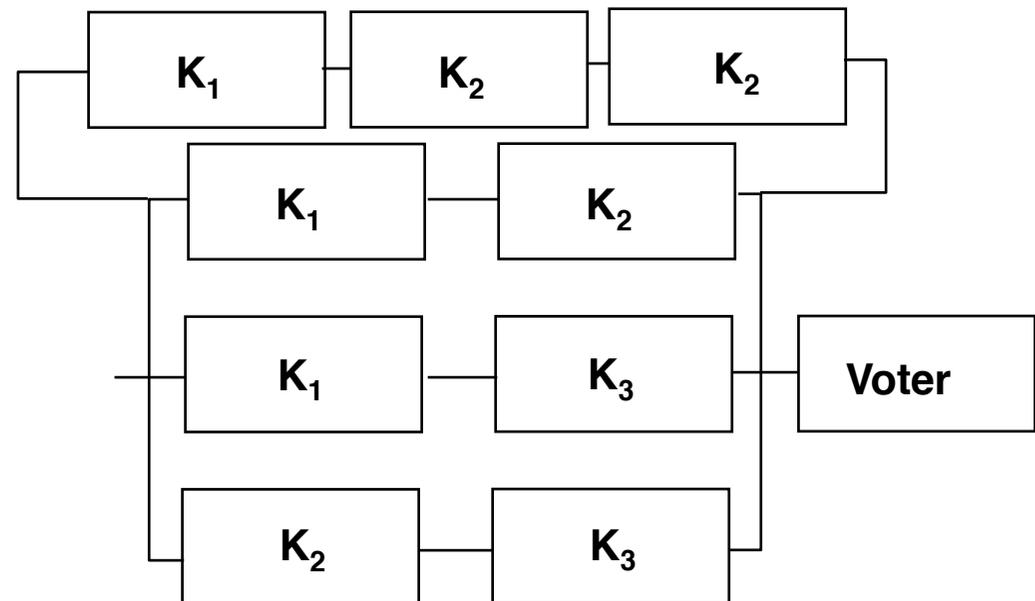


# Beispiel TMR (Triple Modulare Redundanz: 2-aus-3-System)

(Elektr.) Schaltbild



Zuverlässigkeits-Schaltbild



$$P_{\text{TMR}} = (p^3 + 3 p^2 \cdot (1 - p)) \cdot p_{\text{voter}}$$

$$p = 0,9, p_{\text{voter}} = 0,99: P_{\text{TMR}} = (0,9^3 + 3 \cdot 0,9^2 \cdot (1 - 0,9)) \cdot 0,99$$

$$= (0,729 + 3 \cdot 0,81 \cdot (1 - 0,9)) \cdot 0,99$$

$$= (0,729 + 2,43 \cdot 0,1) \cdot 0,99 = 0,972 \cdot 0,99$$

$$= 0,96228$$



---

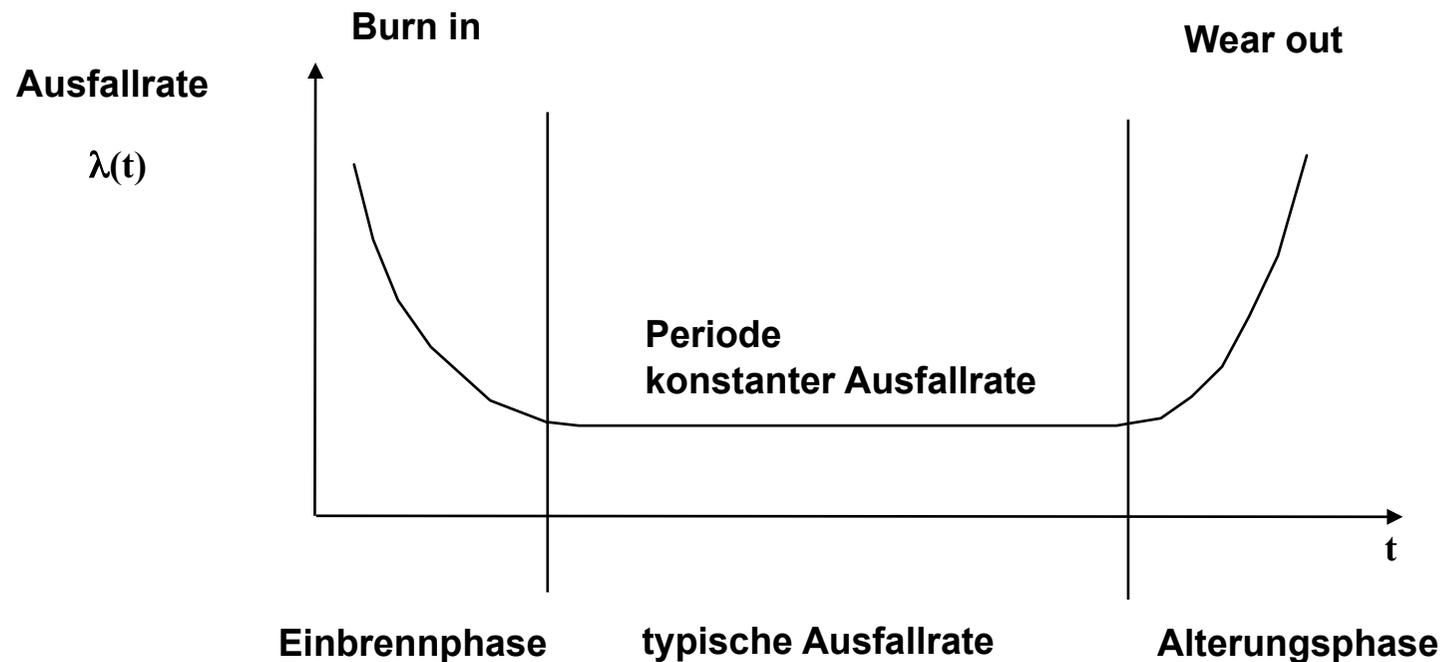
**HAL: "My Fault-Prediction Unit tells me that I will fail within the next 36 hours....."**

**Wie werden die  
Ausfallwahrscheinlichkeiten über  
die Zeit ermittelt ?**



# Ausfallrate über die Zeit

---



Infant  
Mortality

Typische Ausfallrate:  
VLSI-Chip:  $10^{-8}$  Ausfälle/h = 1 Ausfall in 115000 Jahren



# Maße der Fehlertoleranz

---

## *Lebensdauer T*

Zeit vom Beanspruchungsbeginn (DIN 40 042) bis zum Totalausfall (nicht mehr reparierbar)

## *Ausfallwahrscheinlichkeit F(t)*

ist die Wahrscheinlichkeit für eine Komponente bis zum Zeitpunkt  $T < t_i$  auszufallen.

## *Überlebenswahrscheinlichkeit R(t) (Reliability)*

Wahrscheinlichkeit, daß eine Komponente zum Zeitpunkt  $t_i$  noch nicht ausgefallen ist.  $F(t)$  ist das Komplement zu  $R(t)$ .

$$R(t) = 1 - F(t)$$

Für nicht reparierbare Systeme ist  $R(t)$  eine monoton fallende Funktion.  $R(0) \leq 1$ ,  $R(\infty) = 0$

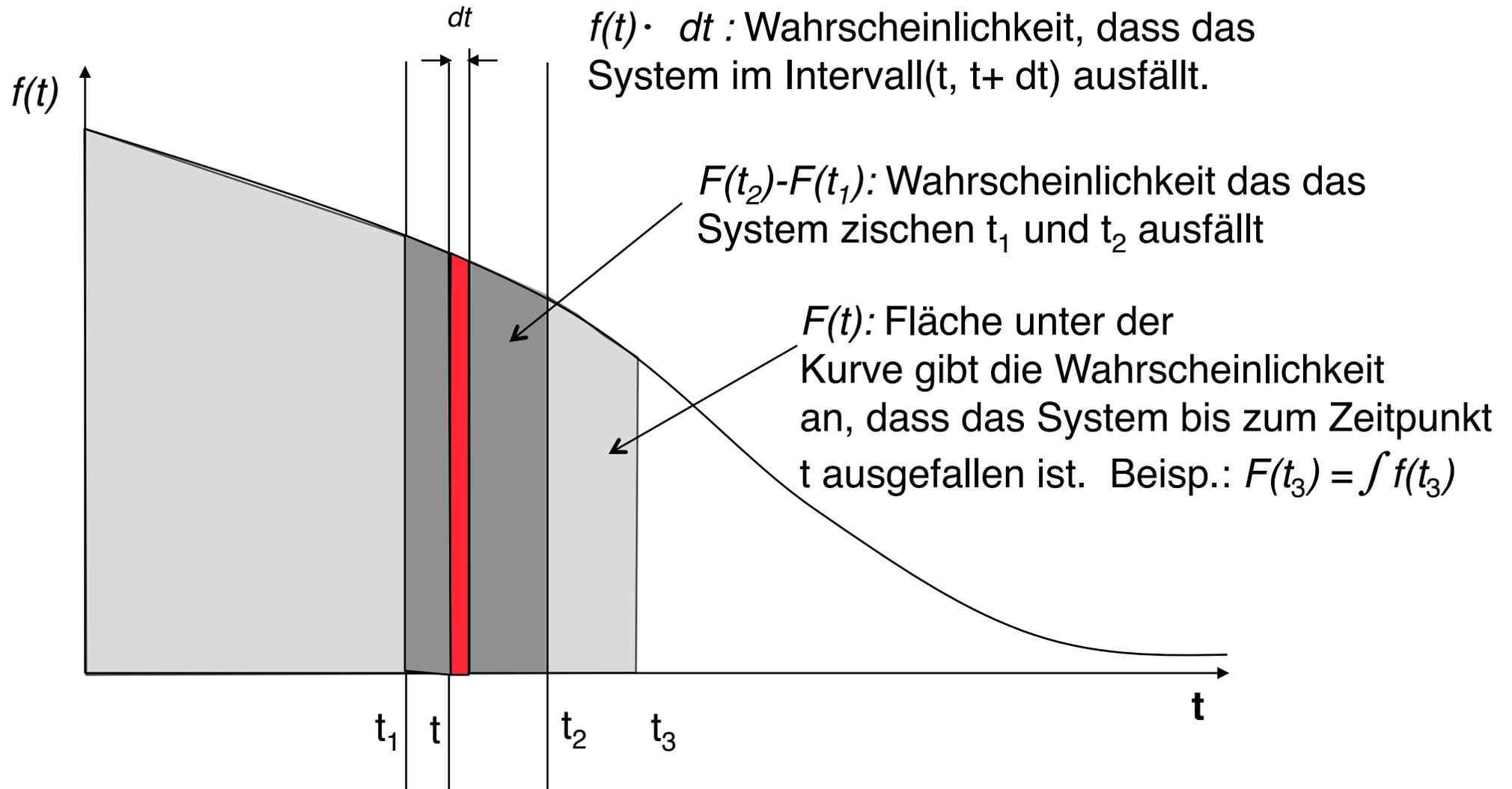
## *Ausfallwahrscheinlichkeitsdichte f(t)*

$f(t)$  ist die Wahrscheinlichkeit, mit der in dem Zeitintervall  $(t, t+dt)$  Ausfälle erwartet werden können.

$$f(t) = \frac{dF(t)}{dt} = - \frac{dR(t)}{dt}$$



# Lebensdauer-Modellierung



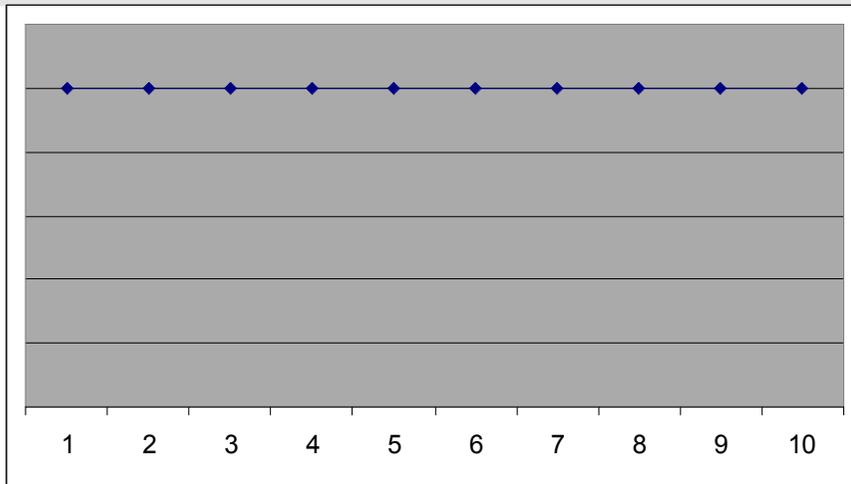
$f(t)$ : PDF: Probability Distribution Function

$F(t)$ : CDF: Cumulative Distribution Function. Für  $t \rightarrow \infty$  :  $F(t) = 1$

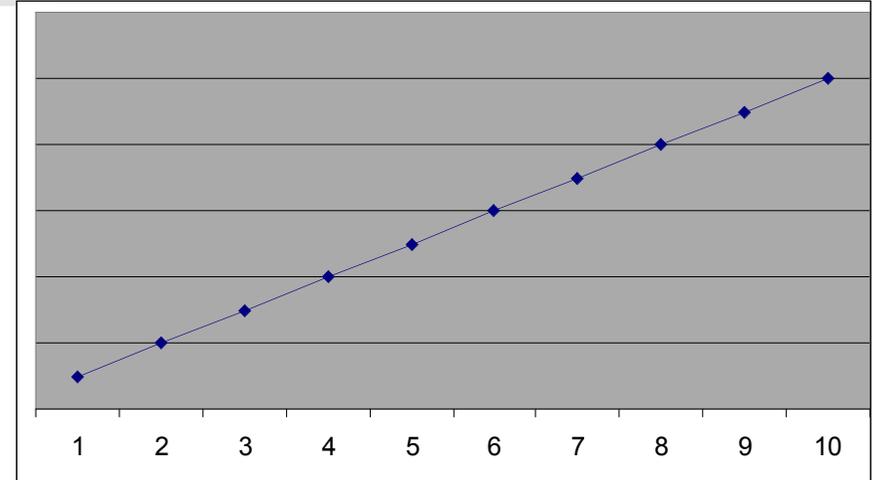


# Konstante Ausfallwahrscheinlichkeitsdichte

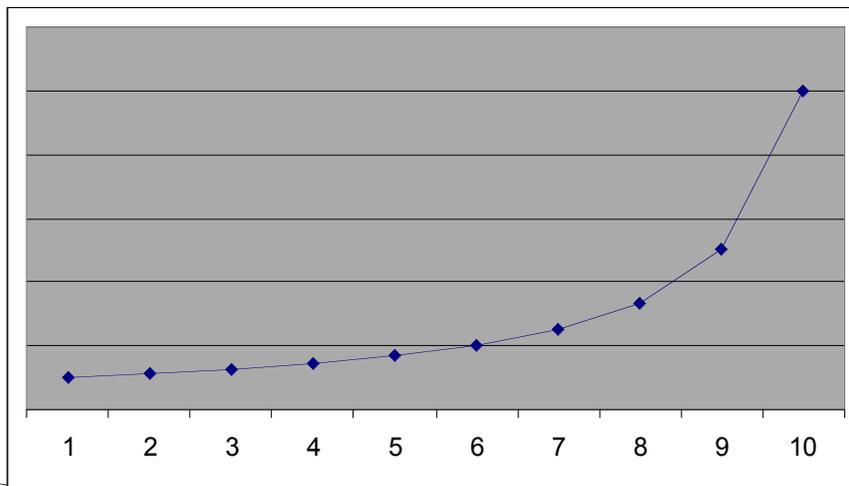
$f(t)$



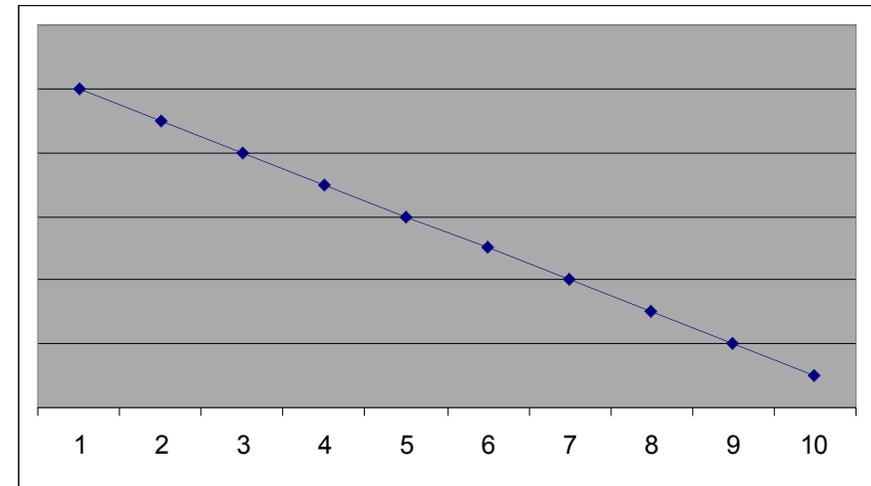
$F(t)$



Ausfallrate:  $\lambda$



$R(t)$



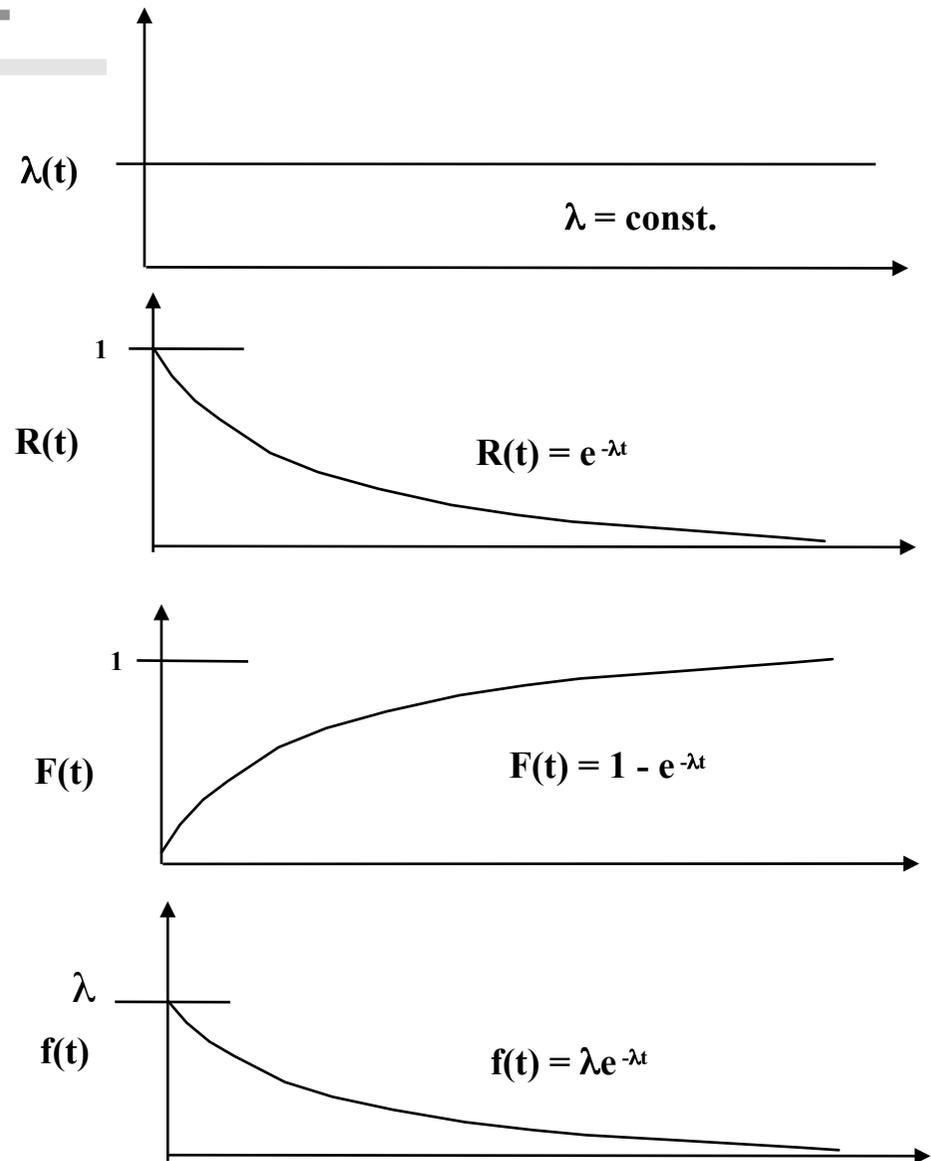
# Maße der Fehlertoleranz

## Ausfallrate $\lambda(t)$

Anzahl der Ausfälle pro Zeiteinheit

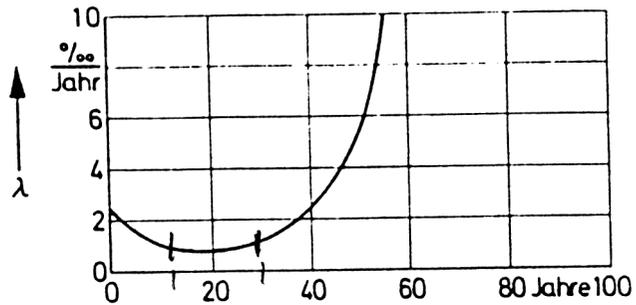
**Bemerkung:** Die Ausfallrate ist relativ zum Bestand definiert. Fallen pro Zeiteinheit immer gleich viele Komponenten aus, steigt die Ausfallrate relativ zum Bestand an, der ja immer kleiner wird.

**Bleibt die Ausfallrate relativ zum Bestand konstant, ergibt sich daraus eine Exponentialverteilung für die Überlebenswahrscheinlichkeit  $R(t)$ .**

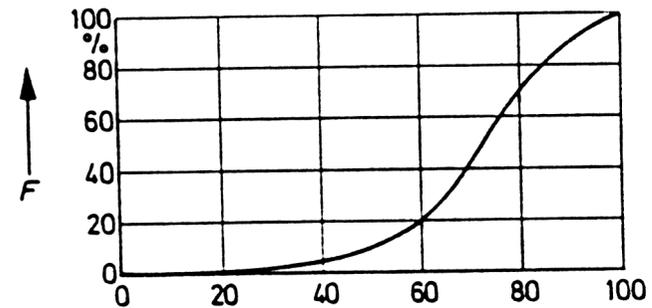


# Lebensdauererverteilung beim Menschen

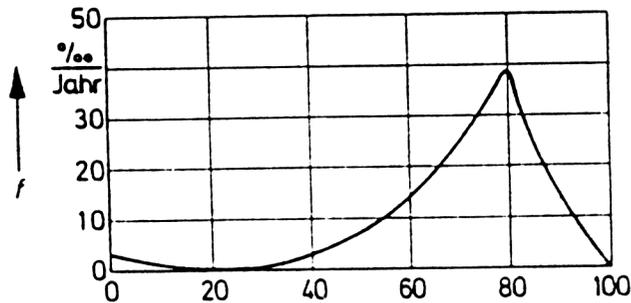
Ausfallrate  $\lambda(t)$



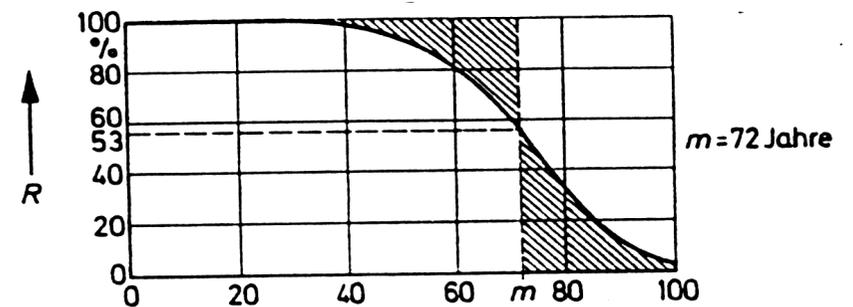
Ausfallwahrscheinlichkeit  $F(t)$



Ausfallwahrscheinlichkeitsdichte  $f(t)$



Überlebenswahrscheinlichkeit  $R(t)$  (Reliability)



# Kenngrößen (Zusammenfassung)

---

Kenngröße	Symbol	Einheit
Lebensdauer	$T$	h
Ausfallwahrscheinlichkeit	$F$	%
Überlebenswahrscheinlichkeit	$R$	%
Ausfallwahrscheinlichkeitsdichte	$f$	%/h
Ausfallrate	$\lambda$	1/h

**Da die Ausfallrate über die Zeit als konstant angenommen wird, lohnt sich häufiges Testen. Je kürzer das Intervall zwischen den Tests, desto geringer die Wahrscheinlichkeit, dass das System in diesem Intervall ausfällt!**



# Maße der Fehlertoleranz

---

Unter der Annahme von  $\lambda(t) = \text{const.}$  gilt:

$$\frac{1}{\lambda} = \text{MTBF} = \text{MTTFF} = \text{MTTF}$$

**MTBF : Mean Time Between Failures**

**MTTFF: Mean Time To First Failure**

**MTTF : Mean Time To Failure**



# Verfügbarkeit (Availability)

---

Zeit in der ein System intakt ist bezogen auf die gesamte Missionszeit

$$A = \frac{U \text{ (Up time)}}{M \text{ (Mission time)}}$$

$$M = U + TR \text{ (Repair time)}$$

$$A = \frac{MTBF}{MTBF + MTTR}$$



# Maße der Fehlertoleranz

---

## Beispiele:

- **Telefonvermittlungssysteme**  
Nichtverfügbarkeit 2h/a bis 3 Min /a

**Klasse 5**

**Aber: in den letzten Jahren waren mehrere große Ausfälle:**  
1 USA-nationsweiter Fehler: 8h  
1 Midwest: 4 Tage

- **Starkstromüberwachung**  
Nichtverfügbarkeit typ.: 2h/a

**Klasse 4**

- **AAS (Advanced Automation System) IBM**  
3 sek/a für kritische Dienste (A = 0,9999999)  
156 sek/a für weniger kritische Dienste (A = 0,9999950)

**Klasse 7++**

**Klasse 5**



# Prozeßsicherheit (Betriebssicherheit, Safety)

---

- ist die Überlebenswahrscheinlichkeit in Beziehung zu kritischen (Funktions-) Ausfällen
- ist die Realisierung eines bestimmten Verhaltens beim Auftreten bestimmter Fehler (Ausschuß Steuerungstechnik VDE/VDI). Grundlage dafür ist eine Sicherheitsvereinbarung, die einen Fehlerkatalog und einen Verhaltenskatalog spezifiziert.
- ist die Wahrscheinlichkeit, dass das System ein bestimmtes antizipiertes Verhalten zeigt, d.h. wenn es von einem kritischen Funktionsausfall betroffen ist, in einen Zustand überführt werden kann, in dem der kritische Funktionsausfall sich nicht katastrophal auswirken kann.

Der Sicherheitsgrad kann angegeben werden mit:

$$S = 1 - \frac{k}{u+k}$$

S : Sicherheitsgrad

k: Anzahl der kritischen Funktionsausfälle

u: Anzahl der unkritischen Funktionsausfälle

**Ein Funktionsausfall ist kritisch, wenn seine Konsequenzen die normalen Installations- und Betriebskosten eines Systems bei weitem (mehrere Größenordnungen) übersteigen.**



# Maße der Fehlertoleranz

## Verfügbarkeit: Einteilung in Systemklassen

$$\text{Klasse: } \lfloor \log_{10} (1/(1-A)) \rfloor$$

1 Jahr = 525600 Minuten = 8760 h

Systemtyp	Nicht-Verfügbarkeit Minuten/Jahr	Verfügbarkeit %	Klasse
Nicht verwaltete Systeme	50 000	~ 90	1
Verwaltete Systeme	5 000	99	2
Gut verwaltete Syst.	500	99,9	3
Fehlertolerante Syst.	50	99,99	4
Hochverfügbare Syst.	5	99,999	5
Sehr hochverf. Syst.	0,5	99,9999	6
Ultra-hochverf. Syst.	0,05	99,99999	7



# Organisation der redundanten Komponenten

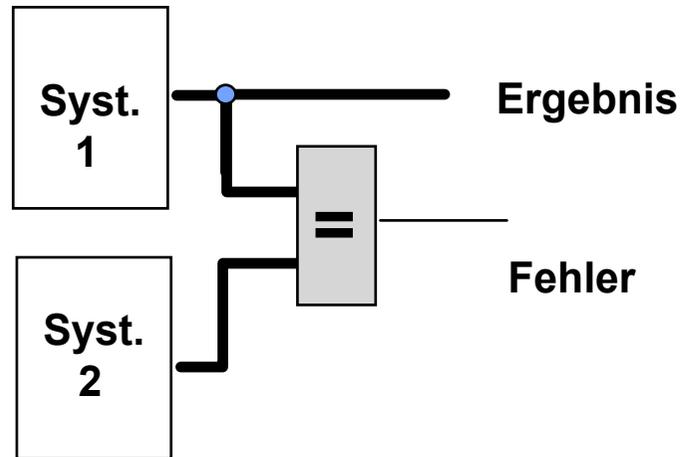
---

## Arten der Redundanz:

- **Komponentenredundanz**
- **Zeitredundanz**
- **Aktive Redundanz:** Mehrere Komponenten erbringen dieselbe Dienstleistung nebenläufig.
- **Passive Redundanz:** Redundante Komponenten sind nicht an der Erbringung der Dienstleistung beteiligt.
- **Cold Standby:** die redundante(n) Komponente(n) werden erst aktiviert, wenn eine aktive Komponente ausgefallen ist. Der Zustand der Berechnung zum Zeitpunkt des Ausfalls der aktiven Komponente muss auf der redundanten Komponente rekonstruiert werden.
- **Hot Standby:** die redundante(n) Komponente(n) ist aktiv, erzeugt aber keine Ausgaben. Die redundante Komponente enthält beim Ausfall der aktiven Komponente bereits deren Zustand und kann sie sofort ersetzen.

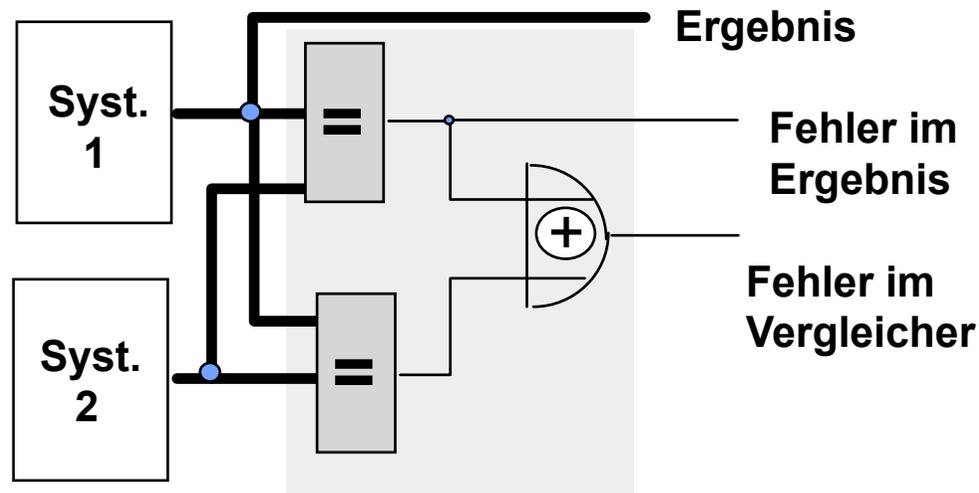


# Organisation der redundanten Komponenten



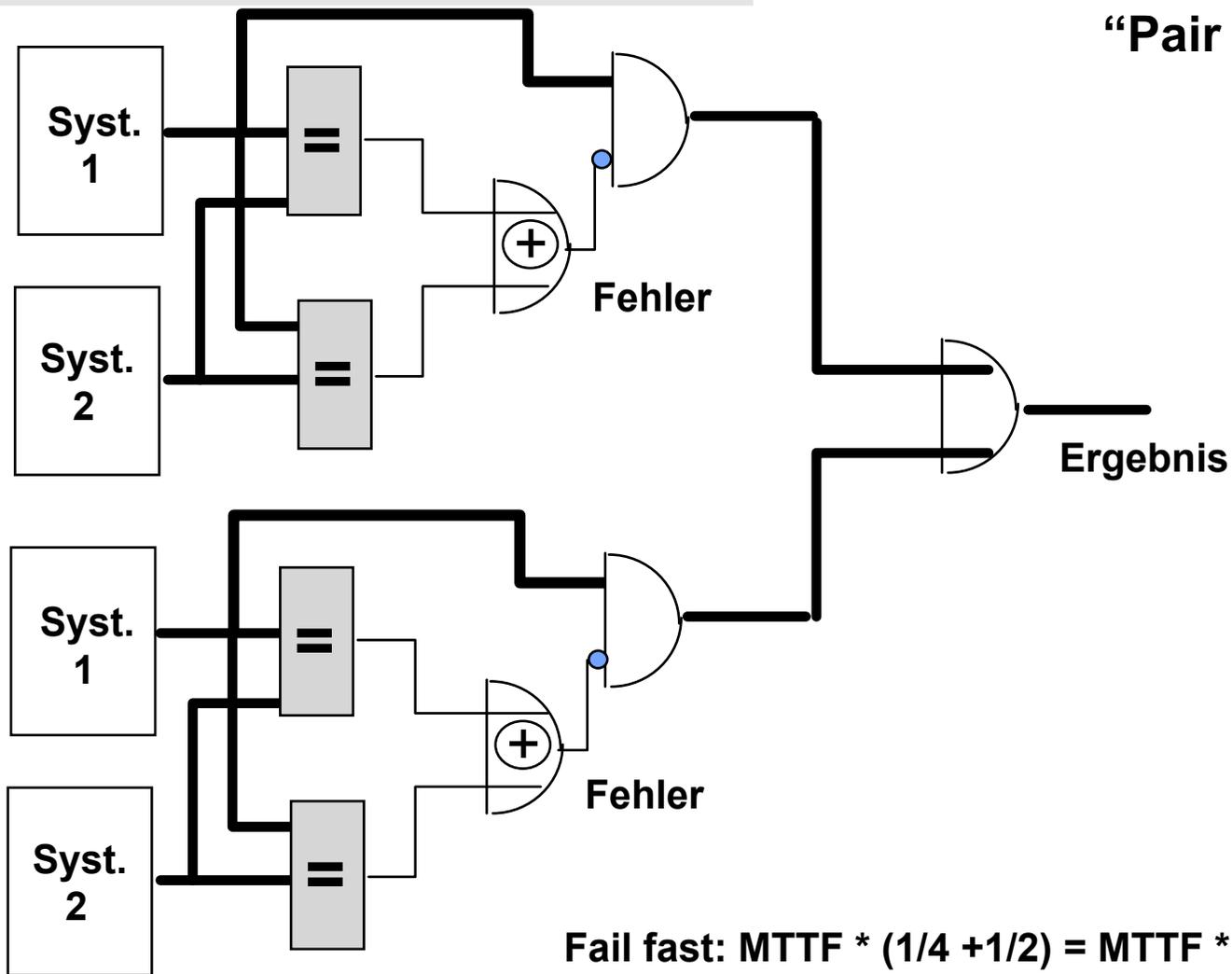
MTTF für ein Modul ist:  $1 * \text{MTTF}$   
Für ein System mit 2 Moduln gilt dann:  
 $\text{MTTF} / 2$ .

Fail Fast:  $\text{MTTF}/2$   
Fail Soft:  $\text{MTTF} * (1+1/2) = \text{MTTF} * 3/2$



# Organisation der redundanten Komponenten

“Pair and Spare”



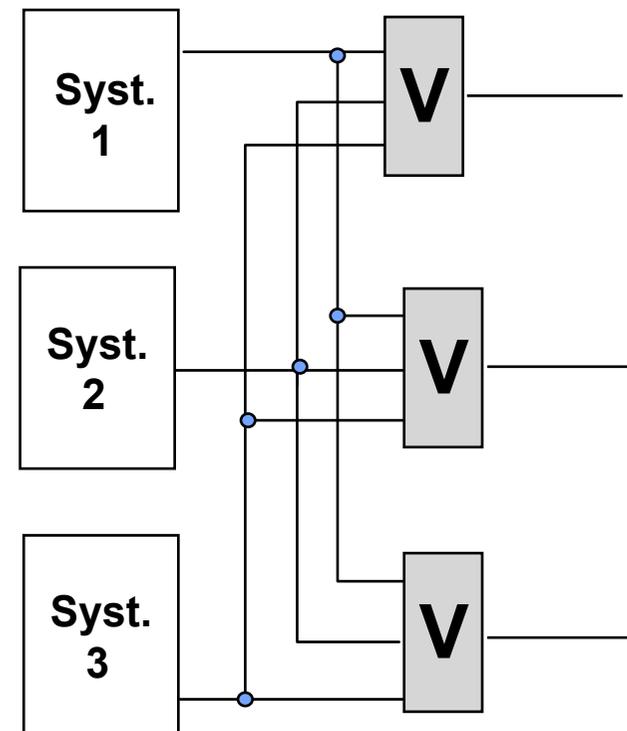
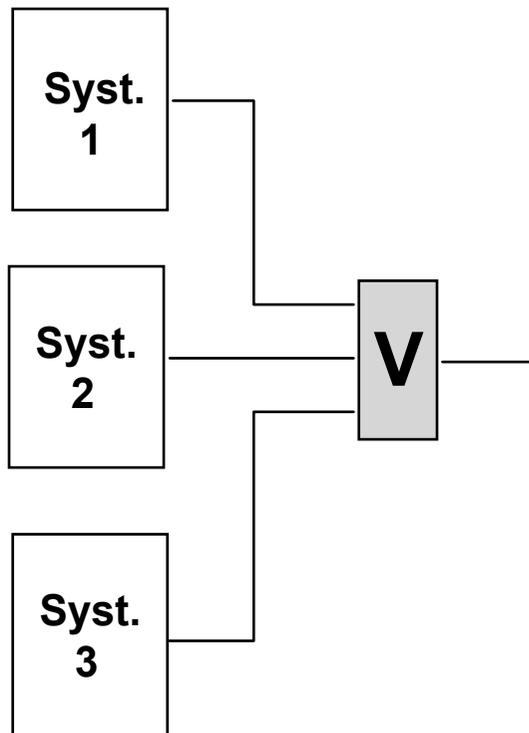
$$\text{Fail fast: } \text{MTTF} * (1/4 + 1/2) = \text{MTTF} * (3/4)$$

$$\text{Fail soft: } \text{MTTF} * ((3/4) + (1/2) + 1) = \text{MTTF} * 2,25$$



# Organisation der redundanten Komponenten

## Triple Modular Redundancy (TMR)



**Fail Fast:  $MTTF/3 + MTTF/2 = MTTF(5/6)$**

**Fail Soft:  $MTTF (1 + 5/6)$**



# Organisation der redundanten Komponenten

Annahme: MTTF = 1 Jahr

Organisation	MTTF (Jahre)	Klasse	Gleichung	Kosten
Simplex	1	3	$MTTF * 1$	1
Duplex (FF)	$\sim 0,5$	3	$MTTF/2$	$2 + \epsilon$
Duplex (FS)	$\sim 1,5$	3	$MTTF(3/2)$	$2 + \epsilon$
TMR (FF)	$\sim 0,8$	3	$MTTF(5/6)$	$3 + \epsilon$
TMR (FS)	$\sim 1,8$	3	$MTTF(1+5/6)$	$3 + \epsilon$
Pair&Spare (FF)	$\sim 0,75$	3	$MTTF(3/4)$	$4 + \epsilon$
Pair&Spare (FS)	$\sim 2,25$	3	$MTTF((3/4) + (1/2) + 1)$	$4 + \epsilon$

**FF: Fail fast**

**FS: Fail soft**

Quelle: J.Gray



# Verfahren zur Konstruktion verlässlicher Systeme

## ➔ Fehleranalyse (Forecasting)

Methoden, um die Auswirkung von Fehlern, ihre Anzahl und ihre Kritikalität abzuschätzen.

**komplementäre Ansätze, um Verlässlichkeit zu erhöhen:**

## ➔ Fehlervermeidung (Fault Avoidance = Fault Prevention + Fault Removal)

umfaßt alle Methoden, die einen fehlerfreien Entwurfs eines Systems zum Ziel haben, d.h. (formale) Spezifikation, Verifikation sowie konstruktive Maßnahmen zur Vermeidung von Fehlern, z.B. besondere Qualität der Bauelemente, Überdimensionierung.

### ➔ perfektionistischer Ansatz

## ➔ Fehlertoleranz

umfaßt alle Maßnahmen, die das System in die Lage versetzen, trotz auftretender Fehler die spezifizierte Funktion kontinuierlich zu erbringen.

### ➔ illusionsloser Ansatz

**Die Ansätze sind komplementär, d.h. sie ergänzen sich und sind zur Konstruktion hochverlässlicher Systeme ALLE unerlässlich.**

