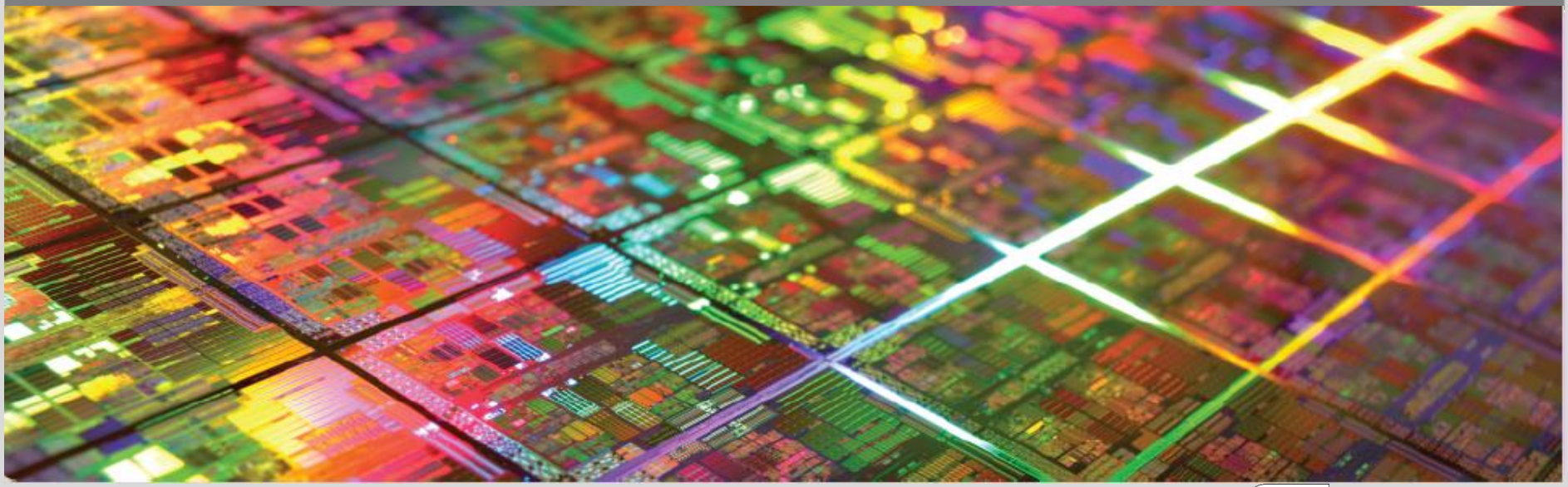
Karlsruher Institut für Technologie

Rechnerstrukturen

Vorlesung im Sommersemester 2017

Prof. Dr. Wolfgang Karl

Institut für Technische Informatik (ITEC), Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung



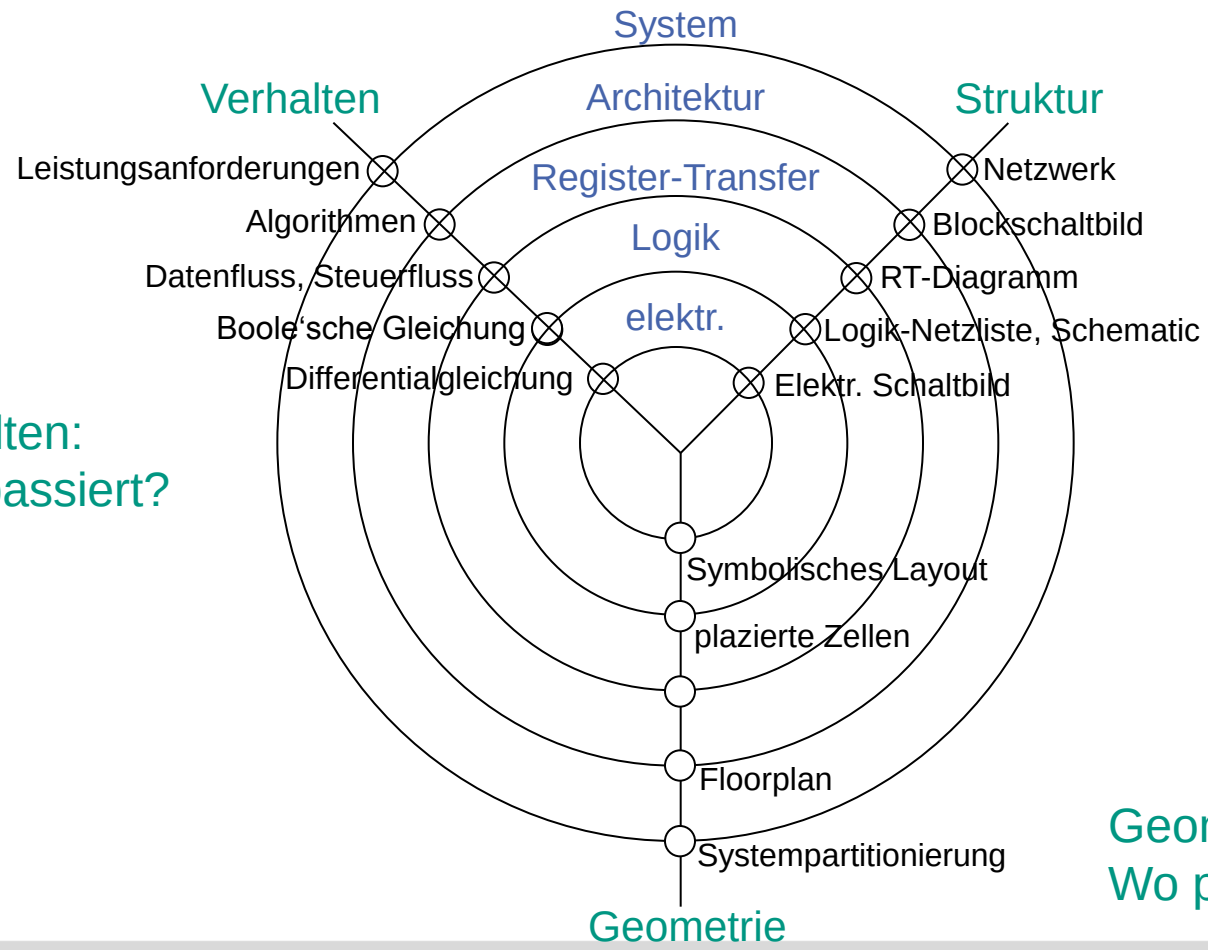
Vorlesung Rechnerstrukturen

Kapitel 1: Grundlagen

- 1.1 Einführung, Begriffsklärung
- 1.2 Entwurf von Rechenanlagen – Entwurfsfragen
- 1.3 Bewertung der Leistungsfähigkeit eines Rechners
- 1.3 Einführung in den Entwurf eingebetteter Systeme

Entwurf von eingebetteten Systemen

- Abstraktionsebenen (Chip-Entwurf):



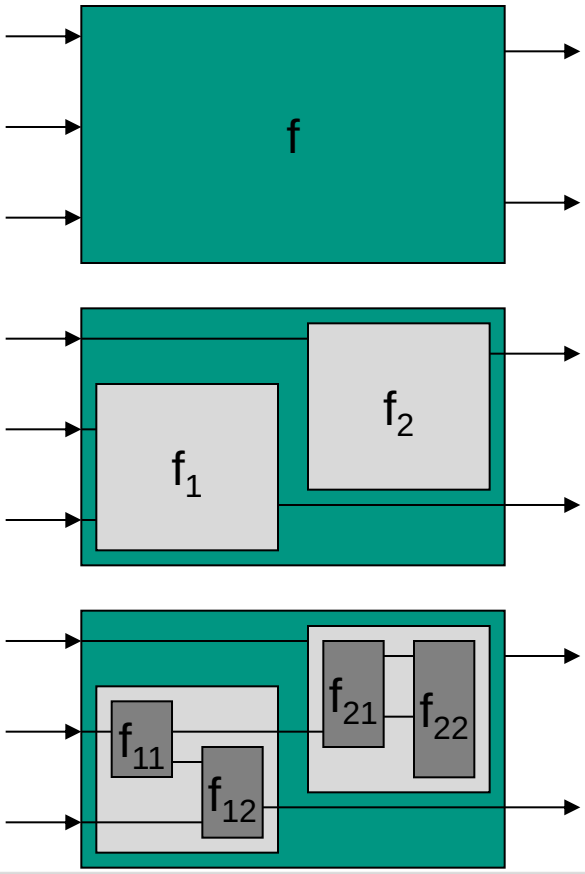
Struktur:
Wie passiert es?

Verhalten:
Was passiert?

Geometrie:
Wo passiert es?

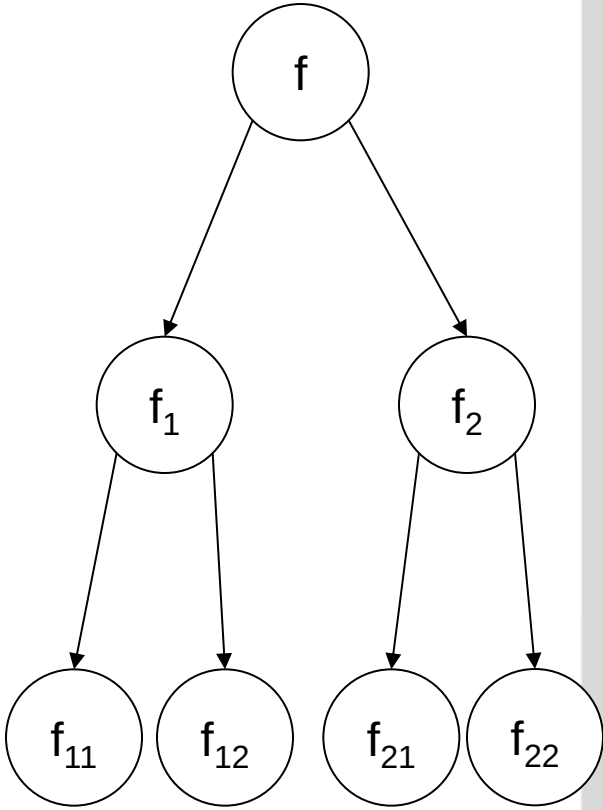
Entwurf von eingebetteten Systemen

- Top-Down-Entwurf (Chip-Entwurf)
  - Schrittweise Verfeinerung, ausgehend von einer hohen Abstraktionsebene



komplexes Verhalten
wenig Struktur

viel Struktur
einfaches Verhalten



Entwurf von eingebetteten Systemen

- Bottom-Up-Entwurf (Rechnerentwurf)
  - Umgekehrte Vorgehensweise wie bei Top-Down
    - Ausgehend von den zur Verfügung stehenden Platinen oder Chips wird in mehreren Entwurfsschritten festegelegt, wie die Funktionen einer Entwurfsebene zu Funktionen der jeweils darüber liegenden Ebene zusammengesetzt werden

Entwurf von eingebetteten Systemen

- Automatische Synthese
  - Vorteile
    - Eingabespezifikation auf höherer Ebene
      - Kürzere Entwurfszeit
      - Komplexere Entwürfe möglich
      - Weniger Entwurfsfehler
    - Ausschöpfung des Entwurfsraums
      - Mehrere Entwürfe können durchgespielt werden
      - Nutzung des Optimierungspotentials
    - Flexibilität
      - Änderung der Spezifikation
      - Änderung der Zieltechnologie
    - Weniger fehleranfällig

Entwurf von eingebetteten Systemen

- Automatische Synthese
  - Nachteile
    - Auswirkung von Randbedingungen
      - Constraint propagation
      - Formulierung auf einer Ebene möglich, aber Auswirkung andere Ebenen nur schwer zu beurteilen!
    - Qualität des Syntheseergebnisse
    - Integration verschiedener Werkzeuge schwierig

Entwurf von eingebetteten Systemen

- Die Hardware-Bescheibungssprache VHDL
  - VHSIC-Programm der Vereinigten Staaten (Very High Speed Integrated Circuits)
  - Standardisierte Hardware-Beschreibungssprache:
    - VHDL wurde 1987 IEEE-Standard, mittlerweile in überarbeiteter Form
  - Die verschiedenen Schaltungsbeschreibungen des gesamten Entwurfsablaufs können dargestellt werden – von der algorithmischen Spezifikationen bis hin zu realisierungsnahen Strukturen.
  - Ursprünglich als Modellierungssprache nur für die Simulation konzipiert
  - Heute zunehmend auch als Sprache für die Synthese und die Verifikation eingesetzt
  - Eingesetzt zum ASIC- und FPGA-Entwurf
  - Enthält alle Elemente einer klassischen Programmiersprache (ADA), erweitert um Konstrukte für den Schaltungsentwurf

Entwurf von eingebetteten Systemen

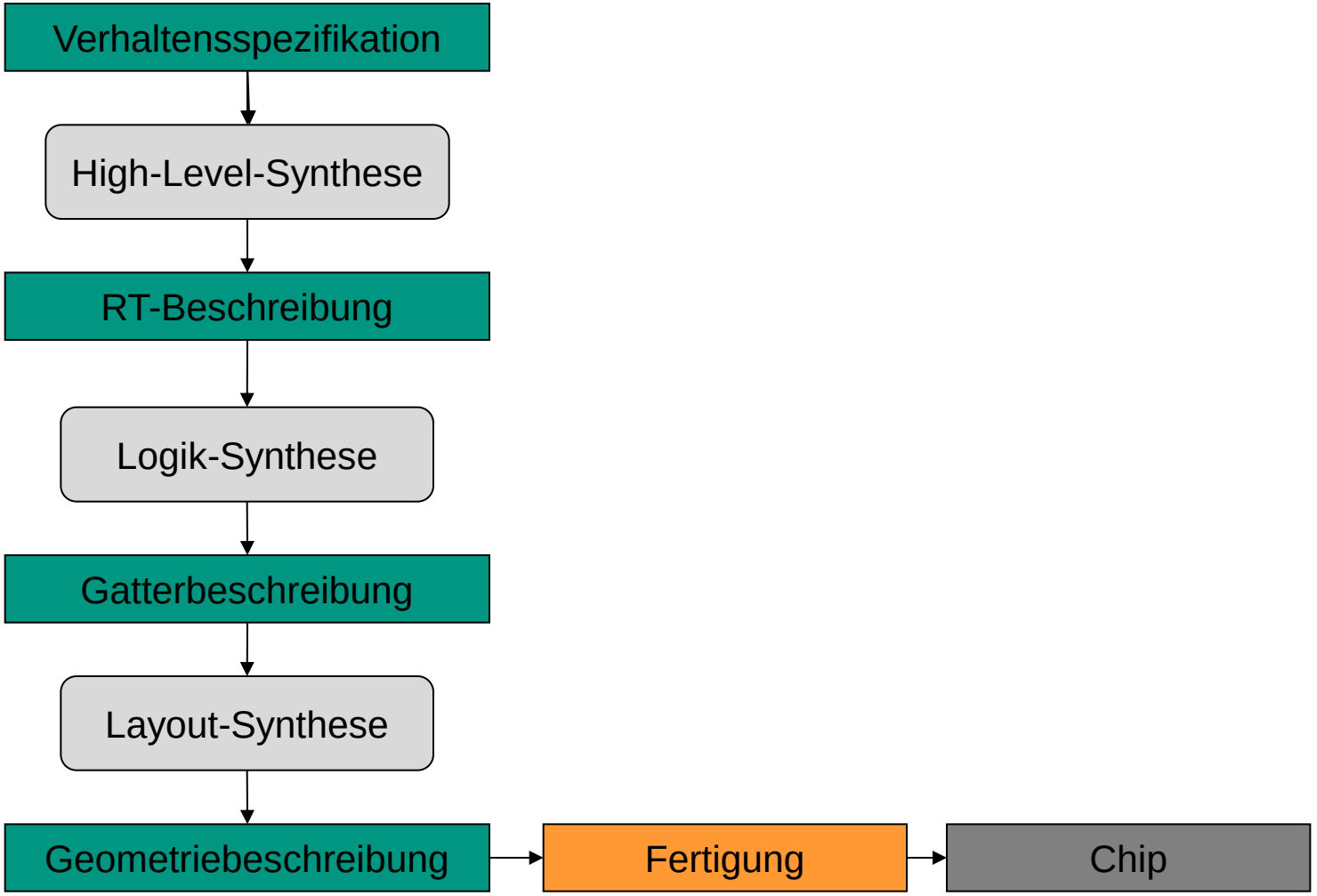
- Chip-Entwurf mit VHDL
  - Grundlage des Entwurfs ist die Spezifikation der Schaltung:
    - das gewünschte Verhalten,
    - die Schnittstellen (Zahl und Art der Ein-/Ausgänge)
    - Vorgaben bezüglich Geschwindigkeit, Kosten, Fläche, Leistungsverbrauch etc.

Entwurf von eingebetteten Systemen

- Chip-Entwurf mit VHDL
  - Entwurfsschritte
    - Verhaltensverfeinerung
    - Strukturverfeinerung
      - wie eine spezifizierte Funktion durch eine Verschaltung von Komponenten mit einfacherer Funktionalität realisiert werden kann.
    - Datenverfeinerung
      - Realisierung abstrakter Datentypen durch einfachere Typen.

Entwurf von eingebetteten Systemen

- Chipentwurf mit VHDL

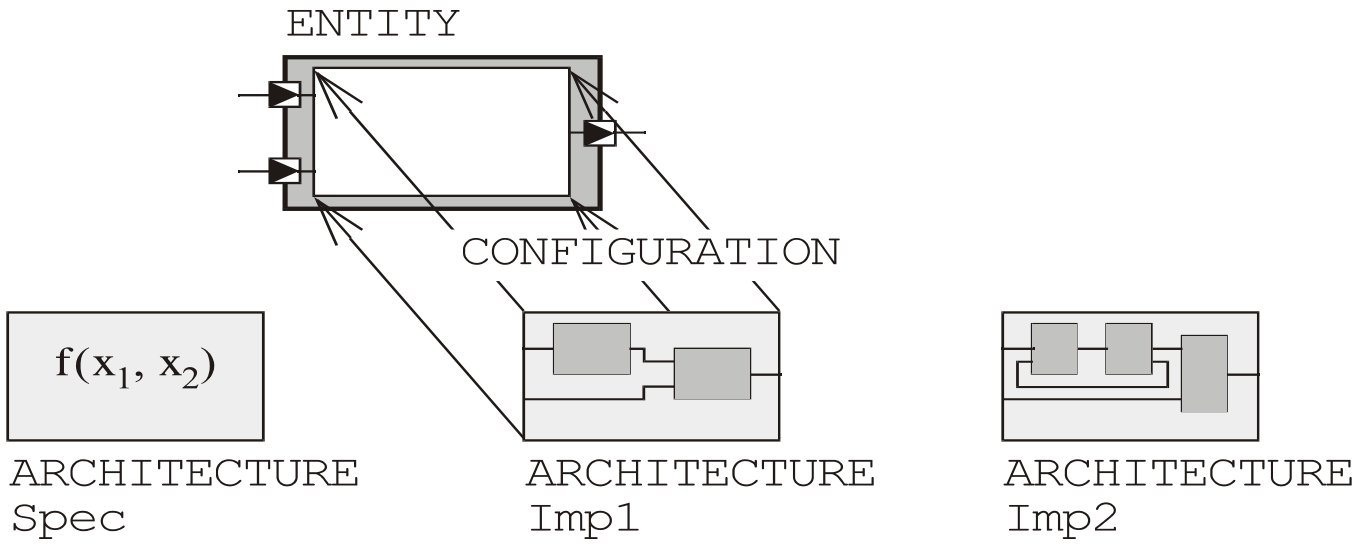


Entwurf von eingebetteten Systemen

- Chip-Entwurf mit VHDL
  - Ein zu entwerfender Chip oder ein Modul ist durch seine Schnittstellen nach außen sowie durch seinen internen Aufbau festgelegt.
  - Der innere Aufbau ist zu Beginn des Entwurfs im allgemeinen nur durch eine funktionale Spezifikation des Verhaltens repräsentiert, die im weiteren Verlauf zu einer strukturellen Implementierung, bestehend aus Submodulen, verfeinert wird.

Entwurf von eingebetteten Systemen

- Chip-Entwurf mit VHDL
  - VHDL erlaubt die getrennte Definition:
    - der Schnittstellen eines Moduls (ENTITY),
    - der internen Verhaltens- oder Strukturrealisierungen (ARCHITECTURE) sowie
    - der Zuordnung, die angibt, welche interne Realisierung für das Modul aktiv ist (CONFIGURATION) und beispielsweise für eine Simulation oder für eine Synthese verwendet wird.



Entwurf von eingebetteten Systemen

- Chip-Entwurf mit VHDL
  - Beispiel: Schnittstellendefinition eines NAND-Gatters

```
ENTITY Nand2 IS
                    PORT(
                X1, X2: IN Std_Logic;
                Y  : OUT Std_Logic);
END Nand2;
```

  - Für einen Baustein darf es nur eine Schnittstellendefinition, jedoch beliebig viele interne Realisierungen (ARCHITECTURE) geben
  - Vorsicht: der ARCHITECTURE-Begriff von VHDL hat nichts mit der Definition von „Architektur“ vs. „Mikroarchitektur“ bei Prozessoren zu tun!

Entwurf von eingebetteten Systemen

- Chip-Entwurf mit VHDL
  - Beispiel: Schema einer ARCHITECTURE

```
ARCHITECTURE Architecture-Name OF Entity-Name IS
        <Daten-, Komponenten- und
        Unterprogrammdeklarationen>
BEGIN
        <Realisierung, z.B. durch Prozesse>
END Spec;
```

Entwurf von eingebetteten Systemen

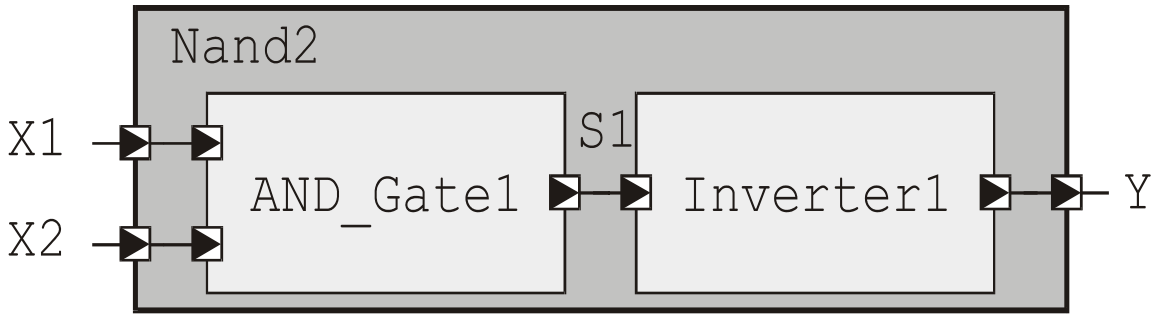
- Chip-Entwurf mit VHDL
  - Strukturbeschreibung einer ARCHITECTURE
    - besteht aus verschiedenen Submodulen und deren Verschaltung.
    - Variable Zuordnung einer ARCHITECTURE („Implementierung“) zu einer ENTITY („Schnittstellenbeschreibung“).
    - Die in einer ARCHITECTURE verwendeten Submodule sind im allgemeinen nicht direkte Kopien einer ENTITY. Man verwendet vielmehr „leere Hülsen“ von Modulen, so genannte components oder Komponenten.

Entwurf von eingebetteten Systemen

- Chip-Entwurf mit VHDL
  - Strukturbeschreibung einer ARCHITECTURE
    - Komponenten werden zu Beginn einer ARCHITECTURE bekannt gemacht (*component declaration*).
    - Anschließend werden Kopien (*instances*) der Komponente erzeugt (*component instantiation*) und die Verbindungsstruktur angegeben.
    - Abbildung Konfigurationen (component configuration), d. h. welche COMPONENT durch welche ENTITY mit welcher ARCHITECTURE realisiert werden soll (separat in einer *configuration unit*).

Entwurf von eingebetteten Systemen

- Chip-Entwurf mit VHDL
  - Beispiel: ein NAND-Gatter soll für die bereits deklarierte ENTITY Nand2 aus einem AND-Gatter und einem Inverter realisiert werden.
    - Modulstruktur von Nand2:



Entwurf von eingebetteten Systemen

- Chip-Entwurf mit VHDL
  - Beispiel: ARCHITECTURE

```
ARCHITECTURE Structure of Nand2 IS
     COMPONENT Inverter
        PORT (
           In1 : IN Std_Logic;
           Out1 : OUT Std_Logic);
     END COMPONENT
     COMPONENT And_Gate
        PORT (
           In1, In2: IN Std_Logic;
           Out1 : OUT Std_Logic);
     END COMPONENT
SIGNAL S1: Std_Logic;
BEGIN
     And_Gate1 : And_Gate PORT MAP (X1, X2, S1);
     Inverter1 : Inverter PORT MAP (S1, Y);
END Structure;
```

Entwurf von eingebetteten Systemen

- Chip-Entwurf mit VHDL
  - Beispiel: Realisierung der Komponenten Inverter und And_Gate

```
ENTITY An2 IS
     GENERIC Delay : Time;
     PORT(
        X1, X2 : IN Std_Logic;
        Y : OUT Std_Logic);
END An2;

ENTITY Inv IS
     PORT(
     Y : OUT Std_Logic;
     X1 : IN Std_Logic);
END Inv;
```

Entwurf von eingebetteten Systemen

- Chip-Entwurf mit VHDL
  - CONFIGURATION-Deklaration
    - Möchte man diese Elemente für die Komponenten von Nand2 benutzen, wobei für beide jeweils eine ARCHITECTURE „Behavior“ ausgewählt werden soll, so wird dies durch eine CONFIGURATION vereinbart.
    - In einer CONFIGURATION wird
      - die Zuordnung von ENTITY und ARCHITECTURE zu konkreten Instanzen jeder COMPONENT gegeben
      - eventuell eine „Umverdrahtung“ der Signale mit PORT MAP oder eine Verfügung von Parametern mit GENERIC MAP durchgeführt.
    - Diejenigen Schnittstellensignale und Parameter, die in COMPONENT und zu verwendender ENTITY in Zahl und Anordnung übereinstimmen, müssen nicht explizit angegeben werden.

Entwurf von eingebetteten Systemen

- Chip-Entwurf mit VHDL
  - Beispiel CONFIGURATION

```
CONFIGURATION Nand_Conf OF Nand2 IS

     -- Angabe der ENTITY
          -- Angabe der ARCHITECTURE
     FOR Structure
        FOR Inverter1 : Inverter
           USE ENTITY Work.Inv1(Behavior);
           PORT MAP (X1 => In1, Y => Out1);
        END FOR;
        FOR And_Gate1: And_Gate
           USE ENTITY Work.An2(Behavior);
           GENERIC MAP (10 ns)
        END FOR;
     END FOR;
END Nand_Conf;
```

Work ist hierbei die Bibliothek, in der Inverter und AND_Gate abgelegt werden.

Die **FOR**-Anweisung gibt hier nicht eine Iterationsschleife an, sondern legt fest, wie bestimmte Einheiten realisiert werden sollen.

Vorlesung Rechnerstrukturen

Kapitel 1: Grundlagen

- 1.1 Einführung, Begriffsklärung
- 1.2 Entwurf von Rechenanlagen – Entwurfsfragen
- 1.3 Einführung in den Entwurf eingebetteter Systeme
- 1.4 Energieeffizienter Entwurf - Grundlagen

Elektrische Leistung und Energie

Motivation

- Mobile Geräte
  - verfügbare Energiemenge durch Batterien und Akkumulatoren begrenzt
  - möglichst lange mit vorhandener Energie auskommen
  - möglichst wenig Energie soll in Wärme umgesetzt werden, um eine Überhitzung zu vermeiden
- Green IT
  - Rechnerhersteller bieten „green HW“ an:
  - niedriger Energieverbrauch
  - ökologische Produktion
  - einfaches Recycling

Elektrische Leistung und Energie

Motivation

- Einige Fakten zum Energieverbrauch
  - Beobachtungen seit 1992:
    - Steigerung der Rechenleistung: Faktor 10000
    - Steigerung der Rechenleistung/Watt: Faktor 300
  - Leistungsaufnahme / Energieverbrauch von Rechenzentren
    - im Bereich zwischen 0,5 MW und 15 MW, oder höher
    - Laufender Betrieb mit 1 MW bedeutet: ~ 8.700.000 kWh/Jahr
    - mit Energiekosten von 0,20 €/kWh: ≈ 1.750.000 €/y, ≈ 5.000 €/d, ≈ 200 €/h
  - CO2-äquivalent mit 500 g/kWh
    - 4,350,000 kg CO2
    - approx. 3,000 2-Personenhaushalte oder
    - approx. 1,000 Automobile mit ~15,000 km/Jahr

Elektrische Leistung und Energie

Motivation

- Einige Fakten zum Energieverbrauch
  - HPC-Bereich
    - Rangliste unter www.green500.org: (November 2012): Maß: MFlops/W
      - Top1: Green500-Liste:
        - National Institute for Computational Sciences/University of Tennessee, Beacon - Appro GreenBlade GB824M, Xeon E5-2670 8C 2.600GHz, Infiniband FDR, Intel Xeon Phi 5110P: 44.89 KW, 2,499.44 MFlops/W
    - hoher Energiebedarf für die Kühlung:
      - zusätzlich 50% - 70% der Leistung

Elektrische Leistung und Energie

Definitionen:

- **Elektrische Leistung** bezeichnet den Energiefluss pro Zeit
- Zusammenhang zwischen **Energie E**, **Leistung P** und **Zeit t**:

$$P = \frac{E}{t} \text{ bzw. } E = P \times t$$

  - Auf elektrische Geräte übertragen:
    - Leistung bezeichnet die aufgenommene bzw. verbrauchte Energie pro Zeit
    - Verbale Unterscheidung der Rechenleistung von der elektrischen Leistung: Leistungsaufnahme oder Verlustleistung

Elektrische Leistung und Energie

Ziele beim Entwurf

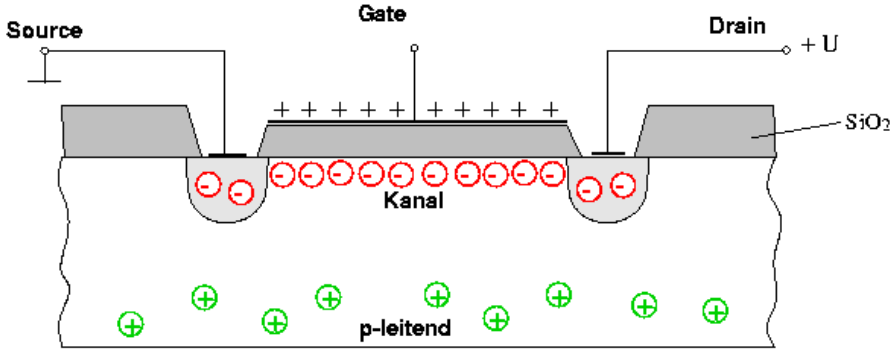
- Verringerung des Energieverbrauches
  - Erhöhung der Betriebszeit eines batteriebetriebenen Gerätes

- Reduktion der Temperatur
  - Reduktion der Leistungsaufnahme (Verlustleistung)
  - Hochleistungsmikroprozessoren:
    - Prozessortemperatur begrenzt möglicherweise die Verarbeitungsgeschwindigkeit
    - Prozessortemperatur beeinflusst die Zuverlässigkeit
    - als Vergleichsmaß wird die auf die Leistungsaufnahme normierte Verarbeitungsgeschwindigkeit verwendet (MIPS/W oder MFlops/W)

Elektrische Leistung und Energie

Grundlagen

- Selbstsperrende nMOS-Transistoren (siehe VL Digitaltechnik und Entwurfsverfahren)
  - der MOS Transistor arbeitet zur Steuerung der Strecke zwischen Source und Drain allein mit elektrischen Feldern (praktisch kein Stromfluss am Gate)
  - je nach Spannung am Gate und dem daraus resultierenden Feld im Kanal können Ladungsträger den Kanal passieren oder nicht.
  - MOS Transistor als Schalter

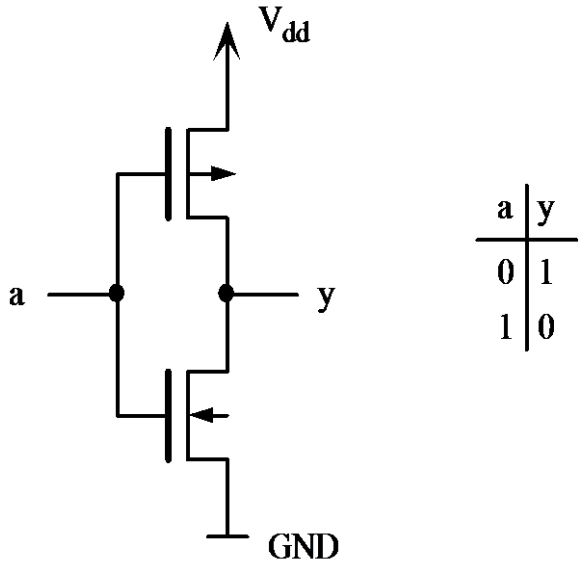


Spannung $\mathbf{U_{GS}}$ zwischen Gate und Source: $\mathbf{U_{GS} = +U}$
Positive Ladungsträger auf der Gate-Elektrode, die
negative Ladungsträger unter der Isolationsschicht induzieren.

Elektrische Leistung und Energie

Grundlagen

- CMOS-Schaltung: Beispiel Inverter
  - Funktionsweise
    - Ist *a = 0*, so wird der nMOS Transistor gesperrt, der pMOS Transistor leitet: am Ausgang liegt $V_{dd} = 1$
    - Ist *a = 1*, so leitet der nMOS Transistor, der pMOS Transistor ist gesperrt: am Ausgang liegt *GND = 0*
    - Weder für *a = 1* noch für *a = 0* existiert ein leitender Pfad von V_{dd} zu GND:
    - kein Stromverbrauch bei konstanten Eingangsvariablen.
    - Stromverbrauch nur beim Übergang

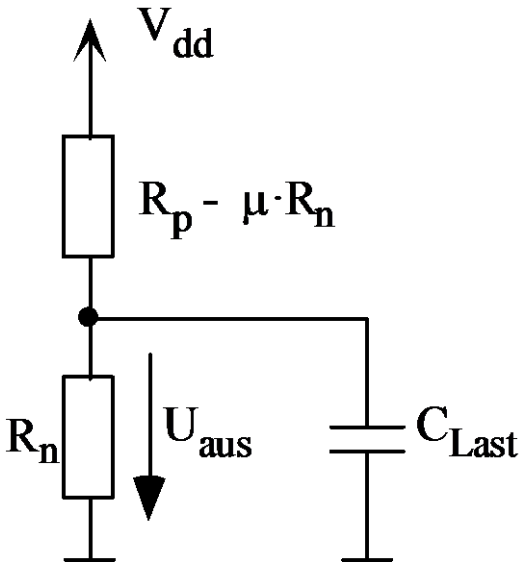


a	y
0	1
1	0

Elektrische Leistung und Energie

Grundlagen

- CMOS-Schaltung: Ersatzschaltbild
  - realitätsnäheres Bild mit Widerständen und Kapazitäten
    - R_p: Widerstand des p-Netzes leitet das p-Netz, so ist R_p klein, ansonsten groß
    - R_n: Widerstand des n-Netzes leitet das n-Netz, so ist R_n klein, ansonsten groß
    - C_{Last}: Lastkapazität der am Ausgang angeschlossenen Leitungen und weiteren Schaltungen



Elektrische Leistung und Energie

Grundlagen

- CMOS-Schaltung: Leistungsaufnahme
  - $\mathbf{P_{total} = P_{switching} + P_{shortcirciut} + P_{static} + P_{leakage}}$
  - Leistungsverbrauch bei Zustandsänderung
    - $\mathbf{P_{switching}}$: Laden oder Schalten einer kapazitiven Last
    - $\mathbf{P_{shortcircuit}}$: Leistungsverbrauch während des Übergangs am Ausgang in einem CMOS Gatter, wenn sich die Eingänge ändern
  - Statischer Leistungsverbrauch (unabhängig von Zustandsänderungen)
    - $\mathbf{P_{static}}$: Statischer Leistungsverbrauch
    - $\mathbf{P_{leakage}}$: Leistungsverbrauch durch Kriechströme

Elektrische Leistung und Energie

Grundlagen

- CMOS-Schaltung: Leistungsaufnahme
  - $\mathbf{P_{switching}}$: Wesentlicher Anteil am Leistungsverbrauch
  - Vereinfacht:
    - $\mathbf{P_{switching} = C_{eff} * V_{dd}^2 * f}$, mit
      - $\mathbf{C_{eff}}$: effektive Kapazität: C * a
      - $\mathbf{V_{dd} = V_{swing}}$

Elektrische Leistung und Energie

Grundlagen

- CMOS-Schaltung: Leistungsaufnahme
  - $\mathbf{P_{shortcircuit}}$: Während des Wechsels des Eingangssignals tritt eine überlappte Leitfähigkeit der nMOS und pMOS-Transistoren auf, die einen CMOS-Transistorgatter ausmachen
  - Vereinfacht:
    - $\mathbf{P_{shortcircuit} = I_{mean} * V_{dd}}$, mit
      - $\mathbf{I_{mean}}$: mittlerer Strom während des Wechsels des Eingangssignals
      - Kann für ein Gatter mit kurzen Eingangsflanken minimiert werden, auf Kosten von langen Übergangszeiten am Ausgang
      - Für eine Menge von Gatter wird versucht, gleiche Zeiten für steigende und fallende Flanken am Eingang und am Ausgang zu erhalten

Elektrische Leistung und Energie

Grundlagen

- CMOS-Schaltung: Leistungsaufnahme
  - **$P_{leakage}$**: Bei realen CMOS-Schaltungen kommt zu dem Stromfluss beim Wechsel des logischen Pegels ein weiterer ständiger Stromfluss hinzu: **Leckströme (Leakage)**
    - Leckströme entstehen, da die Widerstände zwischen den Leiterbahnen der integrierten Schaltkreise nicht unendlich hoch sind.
    - Leckströme wachsen mit zunehmender Integrationsdichte
    - bei Integrationsdichten mit Strukturen kleiner als 100 nm kann die Leistungsaufnahme aufgrund von Leckströmen nicht mehr vernachlässigt werden.

Elektrische Leistung und Energie

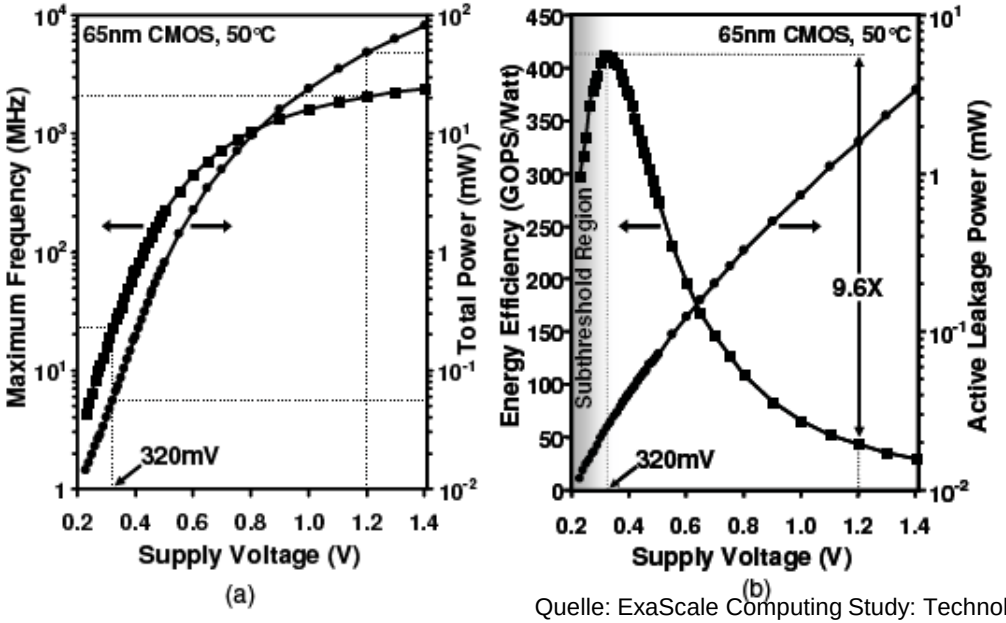
Grundlagen

- CMOS-Schaltung: Leistungsaufnahme
  - unter idealen Voraussetzungen ist $P \sim f$, d.h. Reduktion der Taktfrequenz bedeutet Reduktion der Leistungsaufnahme, aber eine Verlangsamung der Ausführungsgeschwindigkeit
  - unter idealen Voraussetzungen ist $P \sim V^2_{dd}$, d.h. eine Reduktion der Versorgungsspannung um beispielsweise 70% bedeutet eine Halbierung der Leistungsaufnahme. Bei Beibehaltung der Taktfrequenz keine Verlangsamung der Ausführungsgeschwindigkeit!
  - Aber: Versorgungsspannung und Taktfrequenz sind keine voneinander unabhängige Größen: je geringer die Versorgungsspannung desto geringer die maximale Frequenz. Näherungsweise kann ein linearer Zusammenhang angenommen werden: $f \sim V_{dd}$
  - Kubus-Regel: $P \sim V^3_{dd}$ oder $P \sim f^3$

Elektrische Leistung und Energie

Grundlagen

- CMOS-Schaltung: Leistungsaufnahme
  a) Zusammenhang zwischen Versorgungsspannung, Taktfrequenz und Leistung
  b) Zusammenhang zwischen Versorgungsspannung, Energieeffizienz und Leckströme



Experimentelle Untersuchungen an einem Testchip (65nm, 50°C)

Quelle: ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems

Elektrische Leistung und Energie

Grundlagen

- CMOS-Schaltung: Energieverbrauch
  - unter idealen Voraussetzungen ist für eine konstante Zeit t_k der **Energieverbrauch *E*** proportional zur Taktfrequenz ***f: E~f***
  - unter idealen Voraussetzungen ist bezogen auf eine zu erfüllende Aufgabe (z.B. Durchführung einer Berechnung) die dafür benötigte Zeit t_a umgekehrt proportional zur Taktfrequenz. Damit wird der Energieverbrauch zur Erfüllung einer Aufgabe unabhängig von der Taktfrequenz.
  - unter Berücksichtigung der statischen Leistungsaufnahme wächst der Energieverbrauch bezogen auf eine zu erfüllende Aufgabe mit abnehmender Taktfrequenz!
    - Dies wird verursacht durch den statischen Teil der Leistungsaufnahme, der umso längere Zeit anliegt, je länger die Ausführung der Aufgabe durch Verringerung der Taktfrequenz benötigt.

Elektrische Leistung und Energie

Energiespar-Techniken

- Senkung der Leistungsaufnahme ohne Einbußen in der Verarbeitungsgeschwindigkeit und damit auch Senkung des Energiebedarfs für die Bearbeitung einer Aufgabe
- Optimierung der Systemarchitektur
  - Sinnvolles Zusammenwirken aller Komponenten einer Systemarchitektur (HW, Betriebssystem, Kommunikationsschnittstelle, Middleware, Anwendung), um unnötigen Energieverbrauch zu erkennen und zu vermeiden
- Energieoptimierung für Desktop- und Serversysteme
  - Einsatz von Multicore-CPUs
    - Ausnützen der Parallelverarbeitung anstelle Erhöhung der Taktfrequenz
  - Einsatz energiesparender spezialisierter Prozessorkerne (Koprozessoren)
- Energiespartechniken auf den verschiedenen Ebenen des Entwurfs

Zuverlässigkeit und Fehlertoleranz

Begriffsbildung

- **Zuverlässigkeit (dependability)**
  - bezeichnet die Fähigkeit eines Systems, während einer vorgegebenen Zeitdauer bei zulässigen Betriebsbedingungen die spezifizierte Funktion zu erbringen.
  - Ziel

- **Fehlertoleranz (fault tolerance)**
  - bezeichnet die Fähigkeit eines Systems, auch mit einer begrenzten Anzahl fehlerhafter Subsysteme die spezifizierte Funktion (bzw. den geforderten Dienst) zu erbringen.
  - Technik

Vorlesung Rechnerstrukturen

Kapitel 1: Grundlagen

- 1.1 Einführung, Begriffsklärung
- 1.2 Entwurf von Rechenanlagen – Entwurfsfragen
- 1.3 Bewertung der Leistungsfähigkeit eines Rechners
- 1.3 Einführung in den Entwurf eingebetteter Systeme
- 1.5 Energieeffizienter Entwurf – Grundlagen
- 1.6 Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeit und Fehlertoleranz

Begriffsbildung

- **Sicherheit (safety)**
  - bezeichnet das Nichtvorhandensein einer Gefahr für Menschen oder Sachwerte. Unter einer Gefahr ist ein Zustand zu verstehen, in dem (unter anzunehmenden Betriebsbedingungen) ein Schaden zwangsläufig oder zufällig entstehen kann, ohne dass ausreichende Gegenmaßnahmen gewährleistet sind.
- **Vertraulichkeit (security)**
  - betrifft Datenschutz, Zugangssicherheit.

Zuverlässigkeit und Fehlertoleranz

Begriffsbildung

- Zuverlässigkeit ist durch **Zuverlässigkeitskenngrößen** zu quantifizieren:
  - Beispiele: Verfügbarkeit, Überlebenswahrscheinlichkeit, …

- **Anforderung des Benutzers**
  - Bei Erneuerung, Erweiterung oder Wechsel des Systems sollen die Auswirkungen der Änderungen auf die Anwendungen nicht nennenswert wahrnehmbar sein.
  - **Wartungsfreundlichkeit**
    - System: Instandhaltung notwendig?
    - Komponenten: Ersatz?
    - Datenbestände: langfristig lesbar?

Zuverlässigkeit und Fehlertoleranz

Begriffsbildung

- **Ausfall** durch
  - Hardwarekomponenten
  - Software (Programmfehler)
  - Menschliche Eingriffe

- **Sicherheitsrelevante Anwendungen**
  - erfordern hohe **Verfügbarkeit**

Zuverlässigkeit und Fehlertoleranz

Begriffsbildung

- **Nutzungsdauer** eines Rechners
  - Mindestens 5 Jahre oder 40000 h, Dauerbetrieb notwendig?
  - Sehr kleine Ausfallraten der Komponenten ($< 10^{-9}h^{-1}$)
  - Probleme: Nachweisbarkeit, Testmöglichkeiten

- **Beispiel: Wahrscheinlichkeit des Auftretens eines Fehlers**
  - Bei einem Prozessor treten nicht behebbare Fehler einmal alle fünf Jahre auf (im Mittel)
  - Es sind eine Million Prozessoren im Mittel jedes Jahr im Einsatz
  - Daraus folgt: Jedes Jahr wird erwartet, dass 1/5 der Prozessoren ausfallen
  - Mehr als 500 nicht behebbare Fehler treten im Mittel jeden Tag auf
    - Katastrophale Konsequenzen für einen Hersteller
  - Eine Fehlerrate von einem Fehler alle fünf Jahre ist nicht akzeptabel, obwohl es für einen Benutzer akzeptabel erscheint

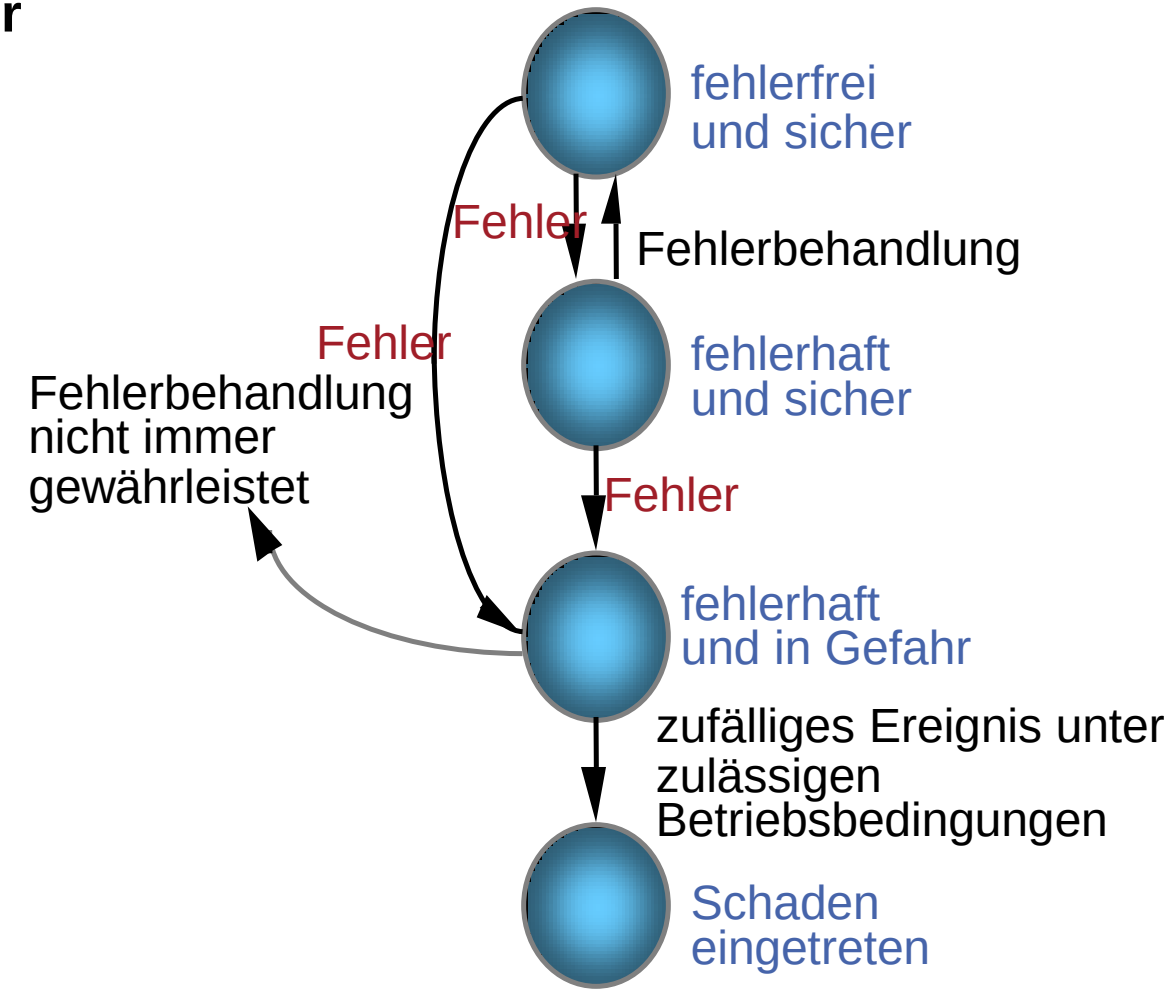
Zuverlässigkeit und Fehlertoleranz

Fragen:

- Wie zuverlässig sind heutige Rechensysteme?
- Nutzen redundante, also fehlertolerante Strukturen zur Verbesserung der Zuverlässigkeit?
- Welche Verbesserung der Zuverlässigkeit lässt sich durch derartige Fehlertoleranzmaßnahmen überhaupt erreichen?

Zuverlässigkeit und Fehlertoleranz

Fehler



Zuverlässigkeit und Fehlertoleranz

Fehler:

- **Funktionsausfälle**
  - Unzulässige bzw. aussetzende Funktion einer Komponente
- **Fehlzustände** (unzulässiger Zustand) einzelner Komponenten des Rechensystems oder Störungen
  - Sind verantwortlich für den Ausfall
  - Werden durch verschiedenste Fehlerursachen erzeugt
- **Wirkungskette**:
  - **Fehler → Fehlzustand → Ausfall**
  - Fehlerausbreitung verhindern!
- **Ziel der Fehlertoleranz**:
  - Tolerierung der Fehlzustände von Teilsystemen (Komponenten)
  - Erhöhung der Zuverlässigkeit
  - Behebung der Fehlzustände vor dem Ausfall des Systems

Zuverlässigkeit und Fehlertoleranz

Fehler:

- **Ursachen**
  - **Fehler beim Entwurf**
    - Führen dazu, dass ein von vornherein fehlerhaftes System konzipiert wird
      - Spezifikationsfehler
      - Implementierungsfehler
      - Dokumentationsfehler
  - **Herstellungsfehler**
    - Verhindern, dass aus einem korrekten Entwurf ein fehlerfreies Produkt entsteht

Zuverlässigkeit und Fehlertoleranz

Fehler:

- **Ursachen**
  - **Betriebsfehler**
    - Erzeugen während der Nutzungsphase eines Rechensystems einen fehlerhaften Zustand in einem vormals fehlerfreien System
    - **Störungsbedingte Fehler**
      - Störungen mechanischer, elektrischer, magnetischer, elektromagnetischer oder thermischer Art sind auf äußere Einflüsse zurückzuführen, denen keine Ursachen im Rechensystem selbst zugrunde liegt
    - **Verschleißfehler**
      - Treten mit zunehmender Betriebsdauer in der HW auf
    - **Zufällige physikalische Fehler**
    - **Bedienungsfehler**
      - Bewusste oder unbewusste Fehleingaben des Benutzers
    - **Wartungsfehler**

Zuverlässigkeit und Fehlertoleranz

Fehler:

- **Fehlerentstehungsort**
  - **Hardwarefehler**
    - Umfassen alle Entwurfs-, Herstellungs- und Bedienfehler
  - **Softwarefehler**
    - Umfassen alle Fehler, die in Programmteilen entstehen

- **Fehlerdauer**
  - **Permanente Fehler**
    - bestehen ab ihrem Auftreten so lange ununterbrochen auf, bis geeignete Reparatur- oder Fehlertoleranzmaßnahmen ergriffen werden
  - **Temporäre Fehler**
    - Treten nur vorübergehend auf
    - Entstehen eventuell mehrmals spontan und verschwinden wieder

Zuverlässigkeit und Fehlertoleranz

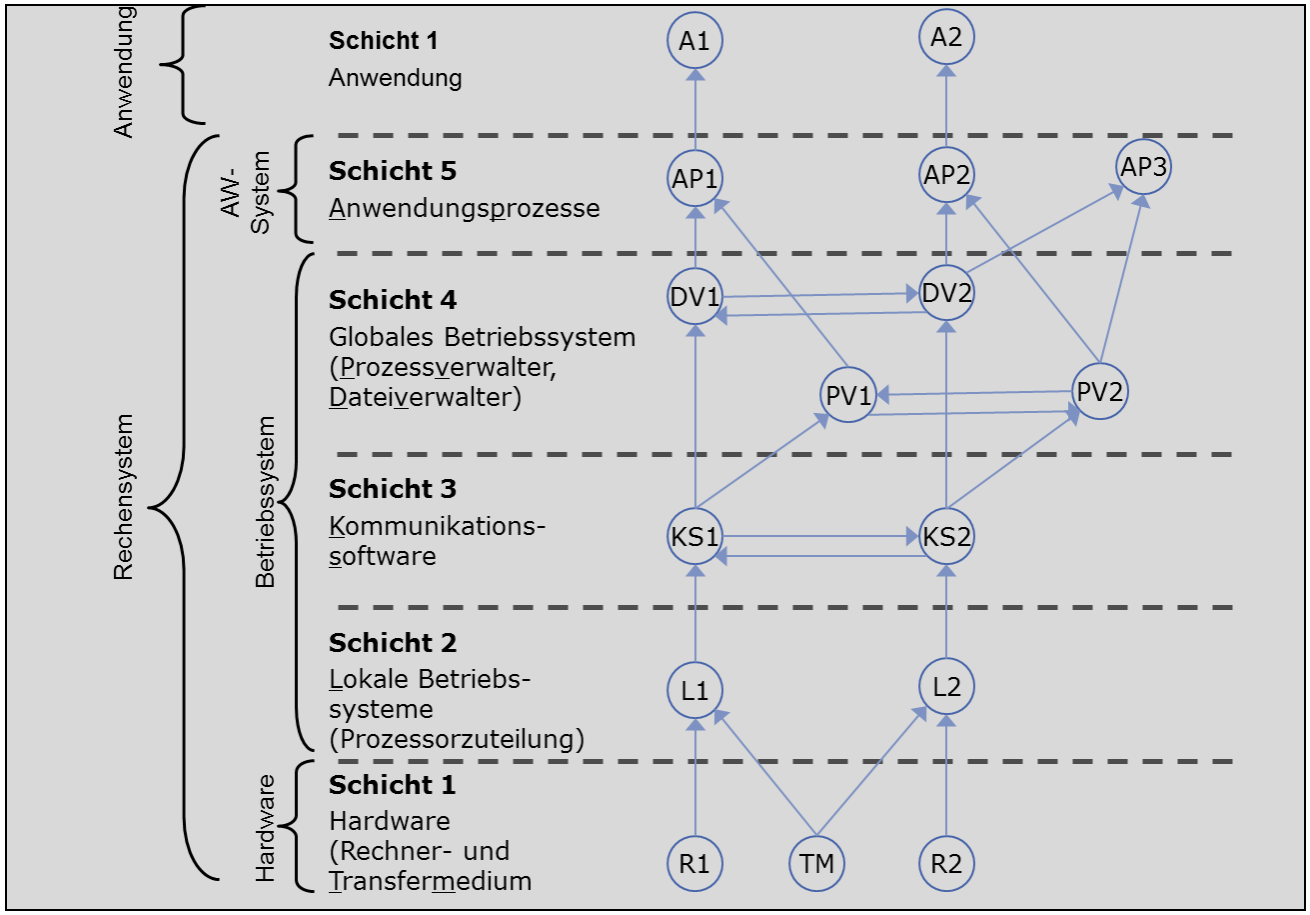
Struktur-Funktions-Modell

- Definition:
  - Das **Struktur-Funktions-Modell** ist ein gerichteter Graph, dessen Knoten die Komponenten und dessen Kanten die Funktionen eines Systems repräsentieren. Eine gerichtete Kante von der Komponente K_i zur Komponente K_j bedeutet, dass K_i eine Funktion erbringt, die von K_j benutzt wird.
  - Eine **Komponentenmenge** heißt **System**, wenn die nach außen erbrachten Funktionen in einer äußeren Spezifikation festgelegt sind. Ein System, das Teilmenge eines anderen ist, heißt **Subsystem**.
  - **Schichtenmodell**:
    - Die Komponenten werden in disjunkte Schichten partitioniert, für die es eine Totalordnung gibt. Funktionszuordnungen sind nur von niedrigeren an höhere Schichten (eine Halbordnung) und innerhalb von Schichten möglich.

Zuverlässigkeit und Fehlertoleranz

Struktur-Funktions-Modell

- **Schichtenmodell** eines 2-Rechensystems



Zuverlässigkeit und Fehlertoleranz

Definitionen:

- Ein **Fehlermodell** beschreibt die möglichen Fehlzustände eines Systems, beispielsweise durch Angabe der Komponentenmengen, die zugleich von einer Fehlerursache betroffen sein können und durch Angabe des möglichen fehlerhaften Verhaltens dieser Komponenten.
- **Binäres Fehlermodell**:
  - Binäre **Fehlerzustandsfunktion Z** gibt für jede Komponente und das System an, ob sie fehlerfrei sind (wahr = kein Fehler, falsch = Fehler):

$$Z : (S \cup \{S\}) \rightarrow \{wahr, falsch\}$$

  - Ein System, das nur dann fehlerfrei arbeitet, wenn es seit der Inbetriebnahme fehlerfrei war, erfüllt:

$$Z(S,t) = \bigwedge_{t_0 \leq t} Z(S,t_0)$$

Zuverlässigkeit und Fehlertoleranz

Binäres Fehlermodell:

- **Nichtredundantes System**
  - Ein System, das nur dann fehlerfrei ist, wenn alle seine Komponenten fehlerfrei sind, wird charakterisiert durch

$$Z(S) = Z(K_1) \wedge \ldots \wedge Z(K_n)$$

- **Systemfunktion** $f(K_1, \ldots, K_n)$
  - gibt an, wie sich die Funktion des Systems aus den Funktionen der einzelnen Komponenten ableitet.
  - Systemfunktion für ein nichtredundantes System

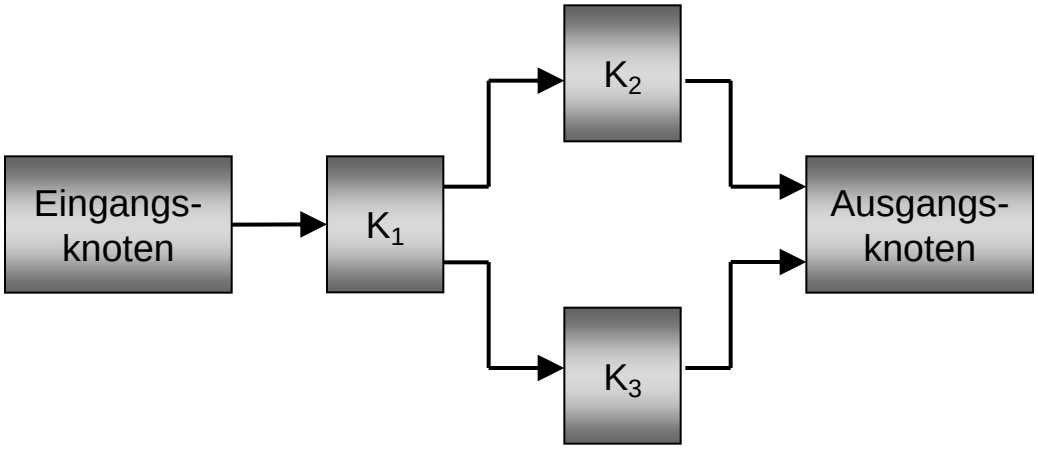
$$S = K_1 \wedge \ \ldots \ \ \wedge K_n$$

Zuverlässigkeit und Fehlertoleranz

Binäres Fehlermodell:

- **Zuverlässigkeitsblockdiagramm**
  - Die Systemfunktion lässt sich grafisch durch ein Zuverlässigkeitsblockdiagramm darstellen:
  - Gerichteter Graph mit einem Eingangs- und einem Ausgangsknoten
  - Beispiel für Systemfunktion

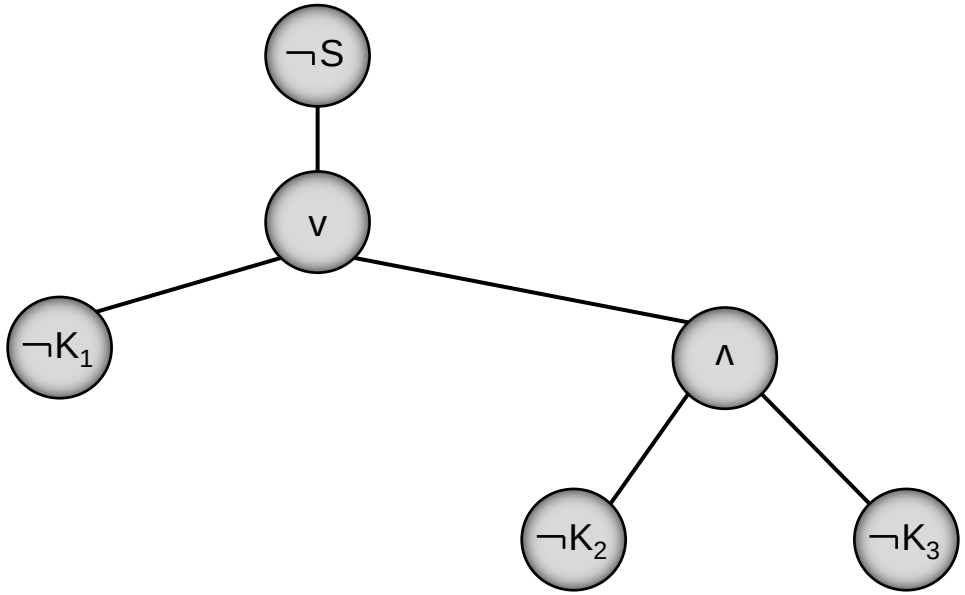
$$S = K_1 \wedge (K_2 \vee K_3)$$
$$Z(S) = Z(K_1) \wedge (Z(K_2) \vee Z(K_3))$$



Zuverlässigkeit und Fehlertoleranz

Binäres Fehlermodell:

- **Fehlerbaum**
  - Strukturbaum der Negation der Systemfunktion
  - Stellt graphisch dar, wie sich Fehler des Systems auf Fehler der Komponenten zurückführen lassen
  - Beispiel: **Fehlerbaum** für $S = K_1 \wedge (K_2 \vee K_3)$ d. h. $\neg S = \neg K_1 \vee (\neg K_2 \wedge \neg K_3)$



Zuverlässigkeit und Fehlertoleranz

Binäres Fehlermodell:

- **Fehlerbereich B:**
  - Ein **Fehlerbereich** B $\subset$ S ist eine Menge von Komponenten, die zugleich fehlerhaft sein können, ohne dass das System S insgesamt fehlerhaft wird.
  - D. h.: aus $\forall K \in S - B : Z(K) =$ wahr

    folgt: $Z(S) =$ wahr
  - Beispiel: $S = K_1 \wedge (K_2 \vee K_3) \rightarrow B_1 = \{K_2\}$ und $B_2 = \{K_3\}$

Zuverlässigkeit und Fehlertoleranz

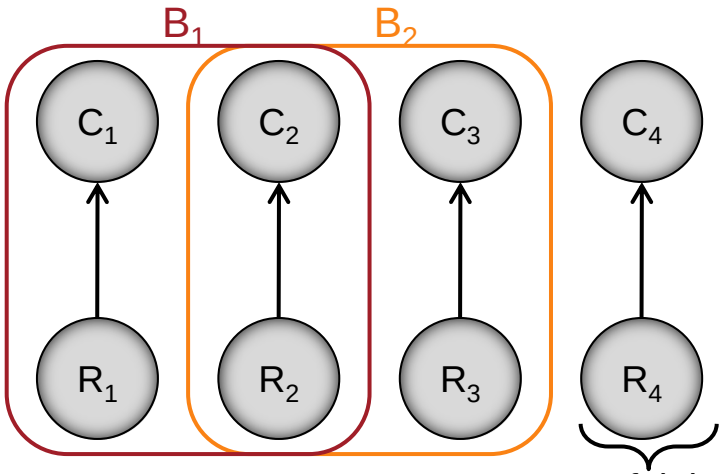
Binäres Fehlermodell:

- **Fehlerbereich B:**
  - **Einzelfehlerbereich:**
    - Wenn für ein System eine Menge von Fehlerbereichen Γ definiert ist, so bezeichnen wir eine Menge von Komponenten, die genau den gleichen Fehlerbereichen angehören, als Einzelfehlerbereich.
  - **Perfektionskern**
    - das Komplement der Vereinigung aller Fehlerbereiche
  - Es wird von den einzelnen Komponenten dieser Bereiche abstrahiert. Dadurch können sich Systemmodelle erheblich vereinfachen.

Zuverlässigkeit und Fehlertoleranz

Binäres Fehlermodell:

- **Fehlerbereich B:**
  - Beispiel: Für ein System S = {R_1,R_2,R_3,R_4,C_1,C_2,C_3,C_4} könnte die Fehlerbereichsannahme wie folgt lauten:
    - Γ = {B_1, B_2
    - Fehlerbereiche B_1 = {R_1,R_2,C_1,C_2} und B_2 = {R_2,R_3,C_2,C_3}
    - Einzelfehlerbereiche E_1 = {R_1,C_1}, E_2 = {R_2,C_2} und E_3 = {R_3,C_3}
    - Perfektionskern P_1 = {R_4,C_4}



Zuverlässigkeit und Fehlertoleranz

Ausfallverhalten

- Steht das mögliche Ausfallverhalten bestimmter Komponenten eines Rechensystems im Vordergrund der Betrachtung, dann ist das eingeführte binäre Fehlermodell zu allgemein
- Es treten nur bestimmte Fehlfunktionen auf.

- Die aus bestimmten Fehlfunktionsannahmen hervorgehenden Einschränkungen des fehlerhaften Verhaltens können den Redundanzaufwand für ein Fehlertoleranzverfahren teilweise erheblich reduzieren.

- Um zu erreichen , dass auf einzelne Komponenten nur bestimmte Fehlfunktionsannahmen zutreffen, kann es notwendig sein, die Komponenten selbst fehlertolerant zu gestalten.

Zuverlässigkeit und Fehlertoleranz

Ausfallverhalten

- **Teilausfall**: Von einer fehlerhaften Komponente fallen eine oder mehrere, aber nicht alle Funktionen aus.
- **Unterlassungsausfall**: Eine fehlerhafte Komponente gibt eine Zeit lang keine Ergebnisse aus. Wenn jedoch ein Ergebnis ausgegeben wird, dann ist dieses korrekt.
- **Anhalteausfall**: Eine fehlerhafte Komponente gibt nie mehr ein Ergebnis aus.
- **Haftausfall**: Eine fehlerhafte Komponente gibt ständig den gleichen Ergebniswert aus.
- **Binärstellenausfall**: Ein Fehler verfälscht eine oder mehrere Binärstellen des Ergebnisses.

Zuverlässigkeit und Fehlertoleranz

Ausfallverhalten

- Systeme, die nur eine bestimmte Art von Ausfallverhalten aufweisen
  - **Fail-stop-System**:
    - Ein System, dessen Ausfälle nur *Anhalteausfälle* sind
  - **Fail-silent-System**:
    - Ein System, dessen Ausfälle nur *Unterlassungsausfälle* sind
  - **Fail-safe-System**:
    - Ein System, dessen Ausfälle nur *unkritische Ausfälle* sind

Zuverlässigkeit und Fehlertoleranz

Ausfallverhalten

- Der Ausfall einer ursächlich fehlerhaften Komponente K kann auch die Fehlerursache für Fehler in anderen Komponenten darstellen, wenn diese Funktionen auf K zugreifen: **Folgefehler**

Zuverlässigkeit und Fehlertoleranz

Ausfallverhalten

- **Maßnahmen der Fehlereingrenzung**
  - **Vertikale Fehlereingrenzung von höheren auf niedrigere Schichten**
    - Niedrigere Schichten prüfen die Funktionsaufrufe vor ihrer Ausführung
    - Beispiel: jeder unzulässige Befehlscode führt zu einer Fehlermeldung.
  - **Vertikale Fehlereingrenzung von der Hardware auf höhere Schichten**
    - Beispiel: Fehlerkorrekturcode im Arbeitsspeicher
  - **Vertikale Fehlereingrenzung von niedrigeren auf höhere Software-Schichten**
    - Durchführen von Plausibilitäts- und Konsistenzprüfungen der Ergebniswerte in höheren Schichten.
    - Es können viele, aber nicht alle Fehler erkannt werden
  - **Horizontale Fehlereingrenzung in lokalen Schichten:**
    - **z.B. (räumliche, elektrische, thermische, ...) Isolierung der Knoten.**
  - **Horizontale Fehlereingrenzung in globalen Schichten:**
    - Hauptproblem der Fehlereingrenzung
    - erfordert mitunter aufwendige Fehlertoleranzverfahren.

Zuverlässigkeit und Fehlertoleranz

Fehlertoleranzanforderungen

- Hohe **Überlebenswahrscheinlichkeit**
  - Beispielsweise zwecks Erfolg bei einer kurzzeitige Mission (z.B. 10-stündiger Flug)
- Hohe **mittlere Lebensdauer**
  - z.B. bei begrenzten Reparaturmöglichkeiten in unzugänglichen Rechensystemen
- Hohe **Verfügbarkeit**
  - z.B. im interaktiven Rechenzentrums- oder Nutzerbetrieb
- Hohe **Sicherheitswahrscheinlichkeit**
  - Schutz von Menschen, Maschinen, Daten
- Hohe **Sicherheitsdauer**

Zuverlässigkeit und Fehlertoleranz

Fehlertoleranzanforderungen

- **Vorgehensweise zur Erfüllung der Anforderungen**
  - **Fehlervermeidung**
    - Perfektionierung, Verwendung von zuverlässigen Komponenten, sorgfältiger Entwurf
  - **Fehlertoleranz**
    - Erfordert Redundanz und damit Zusatzaufwand

Zuverlässigkeit und Fehlertoleranz

Fehlertoleranzverfahren

- **Gesichtspunkte bei der Konstruktion**
  - Ableiten einer **Fehlervorgabe**
    - aus den angenommenen Fehlerraten der Komponenten und den aus den Zuverlässigkeitsanforderungen an das Gesamtsystem
    - Fehlervorgabe besteht aus **Fehlermodell** und der **Menge der zu tolerierenden Fehler**
  - **Menge der zu tolerierenden Fehler**
    - gibt an, welche der im Fehlermodell vorgesehenen Fehler zu tolerieren sind.
    - Häufig wird die Menge der zu tolerierenden Fehler bezüglich einer ***Fehlerbereichsannahme*** formuliert;
      - in diesem Fall ist festzulegen, wie viele ***Einzelfehlerbereiche*** gleichzeitig fehlerhaft werden können und
      - welche ***Fehlfunktionen*** zu behandeln sind.

Zuverlässigkeit und Fehlertoleranz

Fehlertoleranzverfahren

- **Gesichtspunkte bei der Konstruktion**
  - Zur Behandlung von mehreren nacheinander auftretenden Fehlern muss jeweils ein Zeitintervall gegeben sein, in dem keine zusätzlichen Fehler auftreten, bevor die Fehlerbehandlung abgeschlossen ist
  - **Zeitredundanz**
    - Zeitintervall in dem keine weiteren Fehler auftreten, bevor die Fehlerbehandlung abgeschlossen ist
  - **Fehlerbehandlungsdauer**
    - Zeit, die das Fehlerbehandlungsverfahren benötigt, um den Fehler zu behandeln
    - Fehlerbehandlungsdauer muss kleiner als die Zeitredundanz sein

Zuverlässigkeit und Fehlertoleranz

Fehlertoleranzverfahren

- **Zusätzliche Anforderungen**
  - Nachweis der Fehlertoleranzfähigkeit
    - Verifikation, Validierung, Durchführung einer Anfälligkeitsanalyse
  - Geringer Betriebsmittelbedarf (geringe Kosten)
  - Schnelle Ausführung von Fehlertoleranzverfahren (Leistung)
  - Unabhängigkeit von der Anwendungssoftware (Transparenz)
  - Unabhängigkeit vom Rechensystem

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Zuverlässigkeit, Sicherheit einer Rechensystems**
  - Quantifizierbar mittels stochastischer Modelle
  - Man betrachtet die kontinuierliche Variable Zeit zwischen dem Zeitpunkt, ab dem die Zuverlässigkeitsbetrachtung beginnen soll (Zeitpunkt Null), bis zum Auftreten eines betrachteten Effekts

  - Nichtnegative Zufallsvariablen:
    - **Lebensdauer L** – besitzt die **Dichte $f_L(t)$**
    - **Fehlerbehandlungsdauer B** – besitzt die **Dichte $f_B(t)$**
    - **Sicherheitsdauer D** – besitzt die **Dichte $f_D(t)$**

  - Korrespondierende **Verteilungsfunktionen**

$$F_x(t) := \int_0^t f_x(s)ds \quad \text{mit x = L, B und D}$$

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Fehlerwahrscheinlichkeit $F_L(t)$**
  - Bezeichnet die Wahrscheinlichkeit, dass ein zu Beginn fehlerfreies System im Zeitintervall [0,t] fehlerhaft wird

$$F_L(t) = \frac{N_f(t)}{N}$$

$N_f(t)$: Anzahl der Komponenten, die bis zum Zeitpunkt t fehlerhaft sind

N: Gesamtzahl der Komponenten

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Überlebenswahrscheinlichkeit (component reliability) R(t)**
  - Gibt an, mit welcher Wahrscheinlichkeit ein zu Beginn (also zum Zeitpunkt t=0) fehlerfreies System bis zum Zeitpunkt t ununterbrochen fehlerfrei bleibt

$$R(t) = \frac{N_s(t)}{N}$$

$N_s(t)$: Anzahl der Komponenten, die bis zum Zeitpunkt t überleben

N: Gesamtzahl der Komponenten

$$N_f(t) = N - Ns(t)$$

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

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Fehlerwahrscheinlichkeit $F_L(t)$**
  - Verteilungsfunktion der nichtnegativen Zufallsvariablen $F_L(t)$
  - Dichte $f_L(t)$ ist gegeben durch die Ableitung von $F_L(t)$

$$f_L(t) = \frac{d}{dt} F_L(t) = -\frac{d}{dt} \mathrm{R(t)} = -\frac{1}{N} \times \frac{d}{dx} N_s(t)$$

  - Es gilt für die Verteilungsfunktionen nichtnegativer Zufallsvariablen, dass diese in t monoton wachsen und es gilt:
    - $\mathrm{F_L(t)} = 0$ und $\lim_{n \to \infty} \mathrm{F_L(t)} = 1$
    - $\mathrm{R(0)} = 1$ und $\lim_{n \to \infty} \mathrm{R(t)} = 0$

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

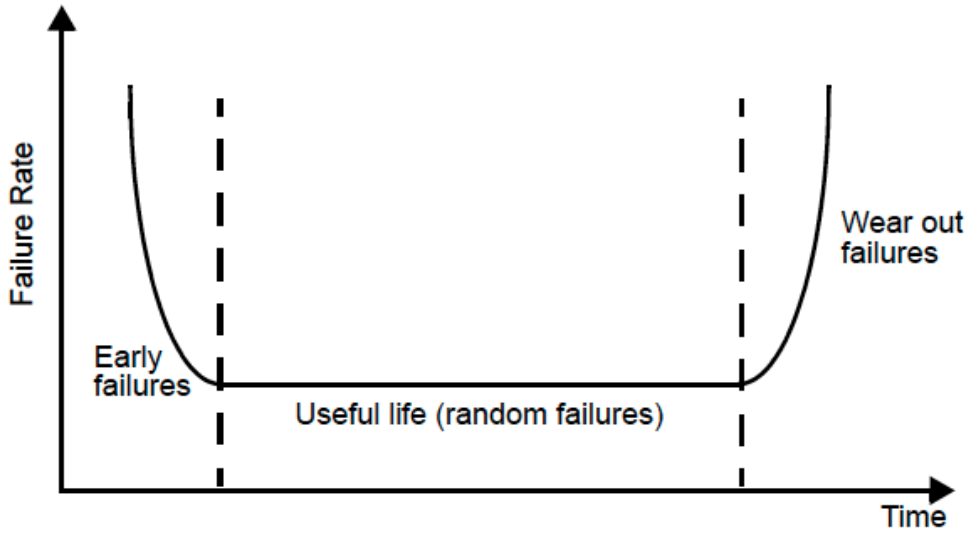
- **Ausfallrate z(t)**
  - Die Ausfallrate bezeichnet den Anteil der in einer Zeiteinheit ausfallenden Komponenten bezogen auf den Anteil der noch fehlerfreien Komponenten
  - Die Gesamtzahl der zu erwartenden ausgefallenen Komponenten zum Zeitpunkt t ist: $f_L(t) \times N$
  - Eine Komponente kann zum Zeitpunkt t nur dann ausfallen, wenn sie bis dahin überlebt hat. Zum Zeitpunkt t verbleiben dann $N_s(t) = R(t) \times N$.
  - **Ausfallrate** ist dann:

$$z(t) = \frac{f_L(t)}{R(t)} = \frac{1}{R(t)} \times \frac{d}{dt} F_L(t) = -\frac{1}{R(t)} \times \frac{d}{dt} R(t)$$

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Ausfallrate z(t)**
  - Badewannenkurve: Ausfallrate über die Lebenszeit eines Systems



Quelle: M. Dubois, M. Annavaram, P. Stenström:
Parallel Computer Organization and Design.
Cambridge University Press, 2012

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Fehlerwahrscheinlichkeit $F_L(t)$:**
  - Ist nur die Ausfallrate bekannt, so ergibt sich die Fehlerwahrscheinlichkeit aus der Anfangswertaufgabe

$$\frac{d}{dt}F_L(t) = fL(t) = z(t) \times R(t) = z(t) \times (1 - FL(t))$$

mit der Anfangsbedingung $F_L(0) = 0$

  - Die Anfangswertaufgabe hat die Lösung:

$$F_L(t) = 1 - e^{-\int_0^t z(s)ds}$$

  - Bei einer konstanten Ausfallrate $z(t) = \lambda$ ist die Fehlerwahrscheinlichkeit folglich exponentialverteilt mit Parameter λ

$$F_L(t) = 1 - e^{-\lambda t}$$

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Verfügbarkeit V:**
  - Die Verfügbarkeit bezeichnet die Wahrscheinlichkeit, ein System zu einem beliebigen Zeitpunkt fehlerfrei anzutreffen.
  - Es interessiert der zeitliche Anteil der Benutzbarkeit des Systems an der Summe der Erwartungswerte von Lebensdauer L und Behandlungsdauer B, wenn während B das System repariert und wieder funktionsfähig wird.
  - Es gilt:

$$v := \frac{E(L)}{E(L) + E(B)}$$

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Sicherheit einer Rechensystems**
  - Geht man Ausfällen aus, die die Sicherheit beeinträchtigen, dann ergeben sich analog zu den bisher betrachteten Größen solche mit Sicherheitsrelevanz
  - **Gefährdungswahrscheinlichkeit $F_D(t)$**
    - Wahrscheinlichkeit, dass ein zu Beginn sicheres System im Zeitintervall [0,t] in einen gefährlichen Zustand gerät
  - **Sicherheitswahrscheinlichkeit $S(t):=1- F_D(t)$**
    - Wahrscheinlichkeit, dass ein zu Beginn sicheres System bis zum Zeitpunkt t ununterbrochen in einem sicheren Zustand bleibt
  - **Mittlere Sicherheitsdauer E(D)**

$$E(D) = \int_0^\infty t \cdot f_D(t)dt = \int_0^\infty S(t)dt$$

    - Erwartungswert der Zeitdauer, bis ein gefährlicher Zustand auftritt

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Funktionswahrscheinlichkeit φ**
  - Da die Zuverlässigkeitsbewertung für die Überlebenswahrscheinlichkeit R und die Verfügbarkeit V einer Komponente bzw. eines Systems analog erfolgt, führen wir für beide Größen den Oberbegriff Funktionswahrscheinlichkeit ein.
  - Ausgehend von gegebenen Funktionswahrscheinlichkeiten $\varphi(K_1)$, …, $\varphi(K_n)$ der Komponenten ist die Funktionswahrscheinlichkeit $\varphi(S)$ des Systems S zu bestimmen. Diese muss alle möglichen Kombinationen von Werten der Fehlerzustandsfunktion aller Komponenten berücksichtigen
  - **Funktionswahrscheinlichkeit des Systems S = f(K_1, …, K_n)**

$$\varphi(S) = \sum_{(K_1,\ldots,K_n)\in f^{-1}(wahr)} \varphi(\wedge_{i=1}^{n} K_i)$$

    - wobei $K_i \in$ {wahr,falsch} den Fehlzustand der jeweiligen Komponente angibt und f^{-1}(wahr) die Menge der Kombinationen von Fehlzuständen der Komponenten des Systems S beschreibt, für die der Fehlerzustand „wahr“ ist

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Nichtfunktionswahrscheinlichkeit**

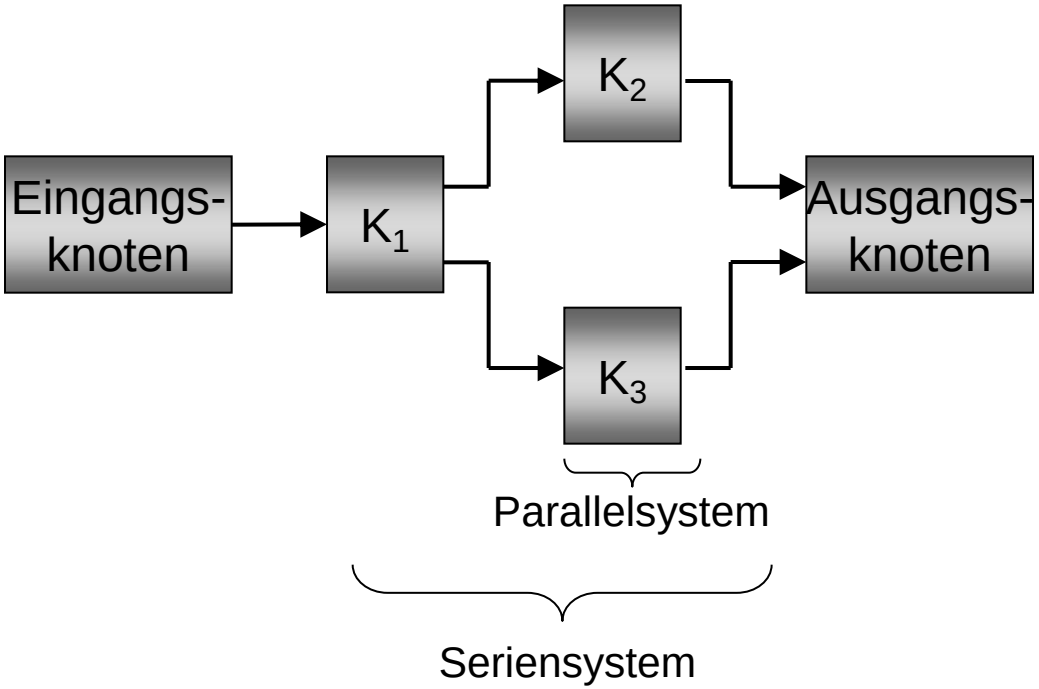
$$\varphi(\neg K) = 1 - \varphi(K)$$

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Funktionswahrscheinlichkeit für Seriensystem und Parallelsystem**
  - Zuverlässigkeitsdiagramm

$$S = K_1 \wedge (K_2 \vee K_3)$$



Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Funktionswahrscheinlichkeit**
  - **Seriensystem**

$$\varphi(\wedge_{K\in\Lambda}) = \prod_{K\in\Lambda} \varphi(K)$$

  - **Parallelsystem**

$$\varphi(\vee_{K\in\Lambda}) = \sum_{\varnothing\neq A\in\Lambda} (-1)^{1+\#A} \cdot \varphi(\wedge_{K\in A} K)$$

K steht für einzelne Komponenten und Λ für eine endliche Menge von Komponenten oder Systemfunktionen

  - **System** $S = K_1 \vee K_2$

$$\varphi(S) = \varphi(K_1 \vee K_2) = \varphi(K_1) + \varphi(K_2) - \varphi(K_1 \wedge K_2)$$

Zuverlässigkeit und Fehlertoleranz

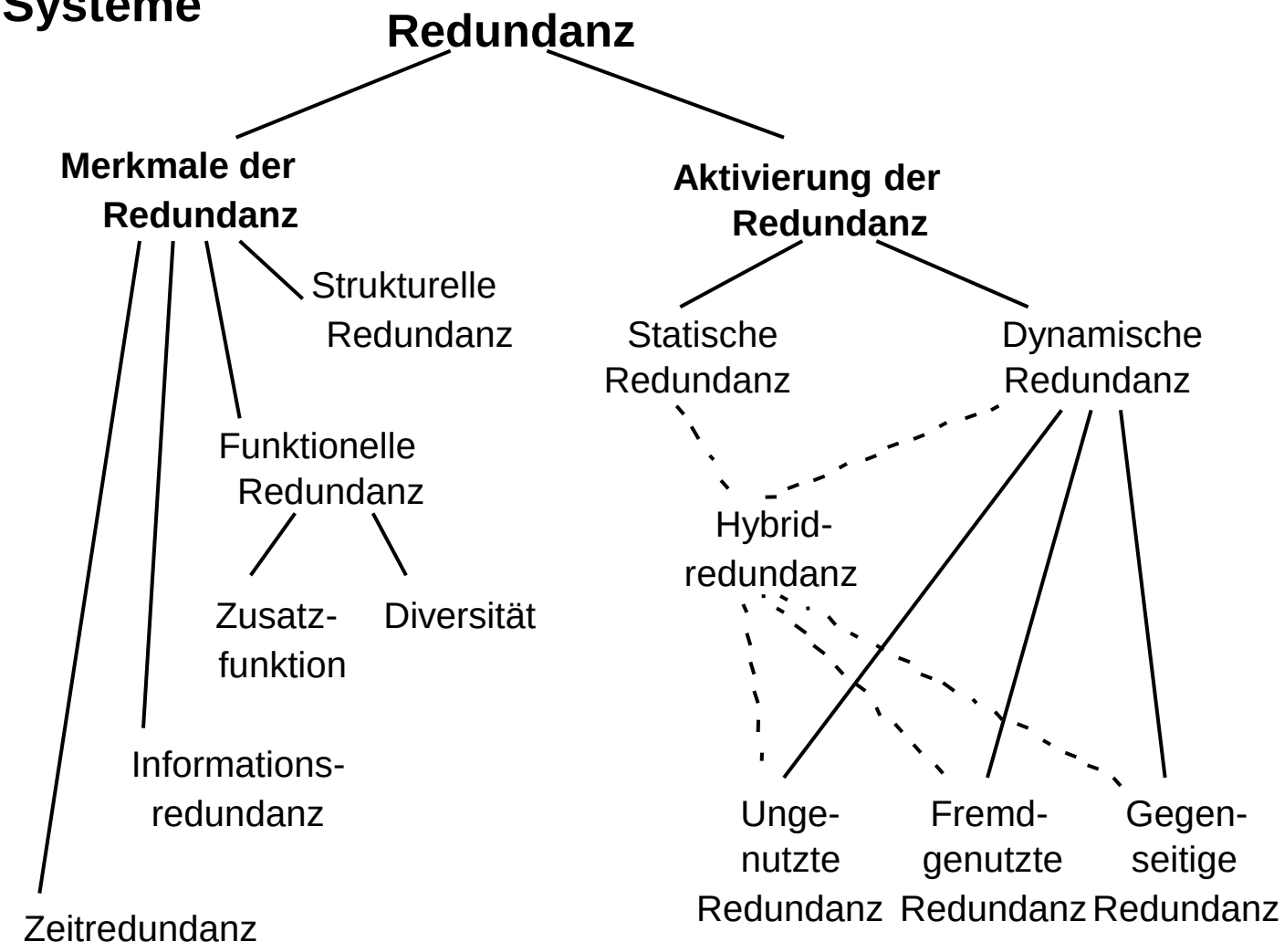
Zuverlässigkeitskenngrößen

- **Funktionswahrscheinlichkeit**
  - **Zuverlässigkeitsverbesserung**

$$\Phi_{S_1 \to S_2} = \frac{\varphi(\neg S_1)}{\varphi(\neg S_2)} = \frac{1-\varphi(S_1)}{1-\varphi(S_2)}$$

Zuverlässigkeit und Fehlertoleranz

Redundante Systeme



Zuverlässigkeit und Fehlertoleranz

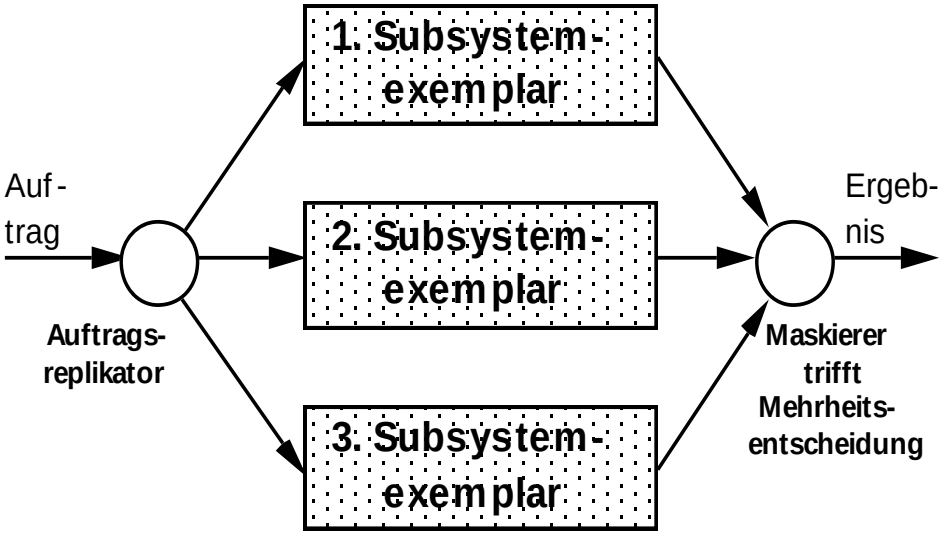
- **Redundanz**
- **Dynamische Redundanz (dynamic redundancy)**
  - bezeichnet das Vorhandensein von redundanten Mitteln, die erst nach Auftreten eines Fehlers aktiviert werden, um eine ausgefallene Nutzfunktion zu erbringen.
  - Typisch für dynamische strukturelle Redundanz ist die Unterscheidung in Primär- und Ersatzkomponenten (bzw. Sekundär- oder Reservekomponenten).
  - Grundstruktur eines dynamisch strukturell redundanten Systems

Zuverlässigkeit und Fehlertoleranz

- **Redundanz**
- **Dynamische Redundanz (dynamic redundancy)**
  - Bevor Ersatzkomponenten aktiviert werden, lassen diese sich auf eine der folgenden Arten verwenden:
  - **Ungenutzte Redundanz**
    - Ersatzkomponenten führen keine sonstigen Funktionen aus und bleiben bis zur fehlerbedingten Aktivierung passiv.
  - **fremdgenutzte Redundanz:**
    - Ersatzkomponenten erbringen nur Funktionen, die nicht zum betreffenden Subsystem gehören und im Fehlerfall bei niedrigerer Priorisierung ggf. verdrängt werden.
  - **gegenseitige Redundanz:**
    - Ersatzkomponenten erbringen die von einer anderen Komponente zu unterstützenden Funktionen, die Komponenten stehen sich gegenseitig als Reserve zur Verfügung.
      Dies ermöglicht einen abgestuften Leistungsabfall (graceful degradation).

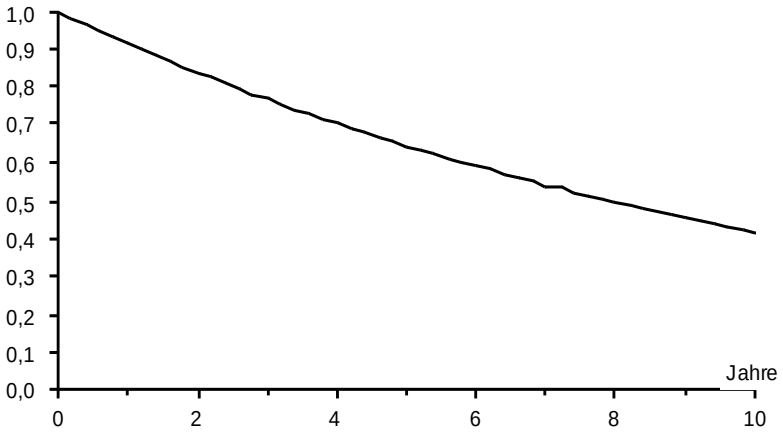
Zuverlässigkeit und Fehlertoleranz

- **Redundanz**
- **Statische Redundanz (static redundancy)**
  - bezeichnet das Vorhandensein von redundanten Mitteln, die während des gesamten Einsatzzeitraums die gleiche Nutzfunktion erbringen.
  - Beispiel der statischen strukturellen Redundanz: **n-von-m-System**
    - 2-von-3-System:



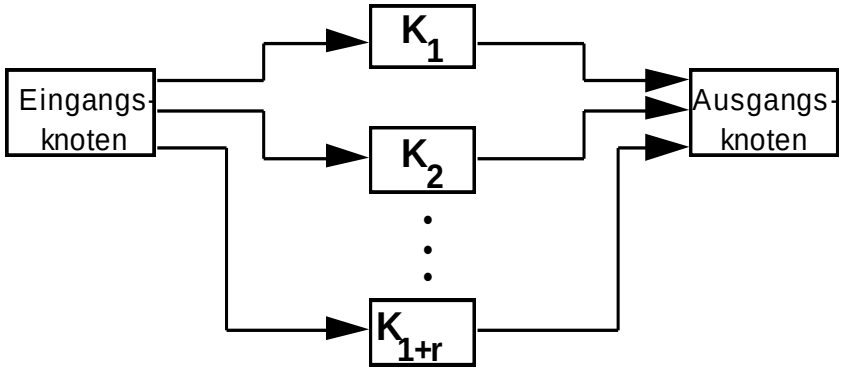
Zuverlässigkeit und Fehlertoleranz

- **Verbesserung der Zuverlässigkeit durch Redundanz**
  - Nichtredundantes Einfachsystem: S1=K1
  - Bei konstanter Ausfallrate beschreibt man die Zeitabhängigkeit der Funktionswahrscheinlichkeit φ (S_1, t) durch eine Exponentialverteilung
    - mit $z(t)=\lambda$, $\varphi(S_1, t) = e^{-\lambda \cdot t}$.
  - Beispiel:
    - Funktionswahrscheinlichkeit φ (S_1, t) mit $\lambda=10^{-5/h}$



Zuverlässigkeit und Fehlertoleranz

- **Verbesserung der Zuverlässigkeit durch Redundanz**
- Parallelsystem (Einfachsystem mit ungenutzter oder fremdgenutzter Redundanz)



Systemfunktion

$$S_{1+r} = K_1 \vee \ldots \vee K_{1+r}$$

Funktionswahrscheinlichkeit

$$\varphi(S_{1+r}, t) = 1 - \prod_{i=1}^{1+r} (1 - \varphi(K_i, t))$$

gleiche konstante Ausfallrate λ

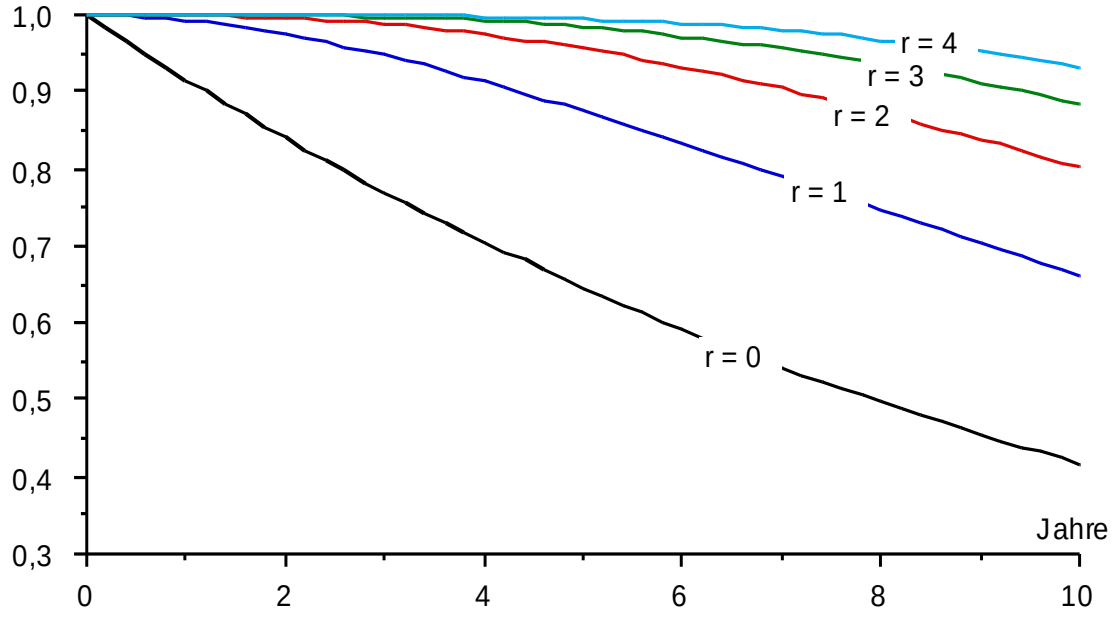
$$\varphi(S_{1+r}, t) = 1 - (1 - e^{-\lambda \cdot t})^{1+r}$$

Zuverlässigkeitsverbesserung

$$\Phi_{S1 \to S1+r} = (1 - e^{-\lambda \cdot t})^{-r}.$$

Zuverlässigkeit und Fehlertoleranz

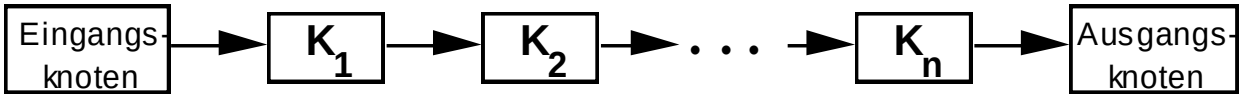
- **Verbesserung der Zuverlässigkeit durch Redundanz**
- Funktionswahrscheinlichkeit für Parallelsystem



Annahme einer Komponentenausfallrate von $\lambda = 10^{-5}/\mathrm{h}$

Zuverlässigkeit und Fehlertoleranz

- **Verbesserung der Zuverlässigkeit durch Redundanz**
- Seriensystem

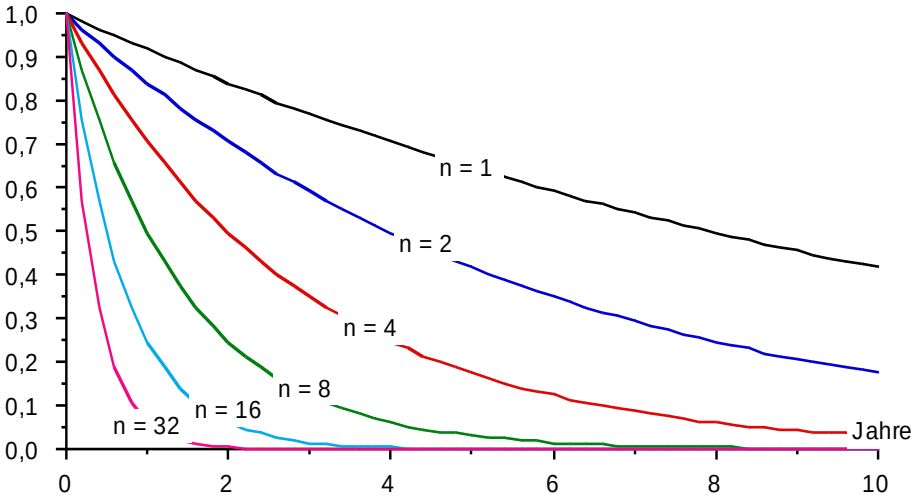


Seriensystem $S_n = K_1 \wedge \ldots \wedge K_n$

Zuverlässigkeit $\varphi(S_n, t) = \prod_{i=1}^{n} \varphi(K_i, t)$

Funktionswahrschein-lichkeit $\varphi(S_n, t)$

für $\lambda = 10^{-5}/h$



Zuverlässigkeit und Fehlertoleranz

- **Statisch redundantes System**
  - Ist die Fehlererfassung zu gering oder verbieten sich wiederholte Berechnungen wegen den geforderten maximalen Antwortzeiten, so kann statische Redundanz eingesetzt werden.
  - Dabei führen mehrere Komponenten die gleiche Berechnung aus, um anschließend die errechneten Ergebnisse zu vergleichen und ein mehrheitliches auszuwählen.
  - Bis zu f fehlerhafte Komponenten können überstimmt werden, wenn mindestens n=f+1 fehlerfreie, insgesamt also m=2·f+1 Komponenten vorhanden sind.

$$S_{m\ von\ m} = \bigvee_{1 \le i_1 < \ldots < i_n \le m} K_{i_1} \wedge \ldots \wedge K_{i_n}$$

Zuverlässigkeit und Fehlertoleranz

- **Statisch redundantes System**