Test und Verlässlichkeit 4: Test und Überwachung

Prof. G. Kemnitz

Institut für Informatik, TU Clausthal (TV_F4.pdf)

16. Januar 2024

Inhalt Foliensatz 4

Vorlesung	1	2	3
ca. ab Folie	2	56	94

Wiederholung

Tests, Überwachung und Kontrollen

Mit der unterstellten Fehlerkultur, dass alle erkannten Probleme beseitigt und der Beseitigungserfolg kontrolliert wird, hängt die Verlässlichkeit hauptsächlich von den durchgeführten Tests und Kontrollen ab.

Test: Einstellung von Testbedingungen und Kontrolle zuzusichernder Merkmale

- statisch: ohne Funktionsausführung
- dynamisch: Ausprobieren der Funktion
- hierarchisch: Bausteine, Schnittstellen, Gesamtsystem, ...

Überwachung: Kontrolle zuzusichernder Merkmal im Betrieb:

- Eingaben, Ausgaben, interne Signale,
- Sensorwerte aus der Systemumgebung,
- Antwortzeiten, Energieverbrauch, ...

Kontrollen als Bestandteile von Tests und Überwachung:

- Nur Format oder auch Werte.
- Automatisiert oder manuell (Inspektion).
- Kontrolle Testergebnisses vorzugsweise Soll-Ist-Vergleich.

Hardware und Software

IT-Systeme bestehen aus Software und Hardware.

Hardware:

- Gegenständliche Bestandteile, die gefertigt werden müssen, verschleißen, ausfallen und gestört werden können.

Software:

- Nichtgegenständlichen Bestandteile (Programme, Know-How, ...).
- Beliebig oft replizierbar ohne nenneswerte Fehlerneuentstehung.
- Keine Aufälle, keine Störungen.

Gemeinsam haben HW und SW aus Sicht der Verlässlichkeit:

- Entwurfs- und Reparaturprozesse, in denen Fehler entstehen.
- Fehlervermeidung als Iteration aus Prozessüberwachung und Beseitigung von Fehlern und Schwachstellen.
- Fehlerbeseitigung als Iterarion aus Tests und Fehlerbeseitigung vor der Einsatzfreigabe.
- Viele Arten von Tests und Kontrollen, ...

1. Wiederholung

Besonderheiten von Hardware:

- Behält Funktionsumfang und Entwurfsfehler bis zur Erneuerung.
- Umgehung erkannter Fehler per SW und die Art der Nutzung.
- Fehlerarmut viel höhere Priorität als bei SW.
- Zusatzmaßnahmen Umgang mit Ausfällen der HW: Voralterung, Einschalttests, Wartung, Redundanzen, ...
- Zusatzmaßnahmen Umgang mit Störungen: Überwachung, Informationsredundanz, Wiederholung, ...
- ...

Besonderheiten von Software:

- Viel mehr funktionale Möglichkeiten incl. für MF-Behandlung.
- Möglichkeit der Beseitigung störender Probleme und Funktionserweiterung in der Nutzungsphase.
- Zusatzfunktionalität hat oft höhere Priorität als Verlässlichkeit.
- ...

Für HW ist Verlässlichkeit schon immer extrem wichtige Einsatzvoraussetzung. Konzepte für die strikte Sicherung der Verlässlichkeit haben sich deshalb oft zuerst für HW etabliert.

Dieser Foliensatz fokusiert auf Tests und Kontrollen, die nicht unbedingt spezifisch für HW oder SW sind.

Die Folgefoliensätze 5 und 6 gehen anschließend stärker auf die auf Besonderheiten von HW und SW ein.

Wiederholung Kenngrößen von Tests

Fehlerüberdeckung, Anteil der nachweisbaren Fehler:

$$FC = \frac{\#DF}{\#F}\Big|_{\mathrm{ACR}} \tag{1.52}$$

Phantom-MF-Rate während des Tests:

$$\zeta_{\mathrm{PhanT}} = \frac{\#PM}{N}\Big|_{\mathrm{ACR}} \tag{1.53}$$

Defektüberdeckung, bei Ersatz erkannter defekte Einheiten:

$$DC = \frac{\#D_{\mathrm{D}}}{\#D}\Big|_{\mathrm{ACR}}$$

Davon auch befasst mit μ, σ, wahrscheinlichen Bereichen.

FC	Fehlerüberdeckung (fault coverage), Anteil der nachweisbaren Fehler.
$\#DF$	Anzahl der erkennbaren Fehler (Number of detectable faults).
$\#F$	Anzahl der Fehler (Number of faults).
ζ_{PhanT}	Phantom-MF-Rate des Tests (Phantom MF rate during test).
$\#PM$	Anzahl der Phantom-MF, d.h. der korrekten DS, die als MF klassifiziert werden.
N	Anzahl der Tests.
ACR	Geeignete Zählwertgrößen, typ. 100 ... 1000 ein- und nicht eingetretene Zählereignisse.
DC	Defektüberdeckung (defect coverage), Anteil der erkennbar defekten Produkte.
$\#DD$	Anzahl der als defect erkannten defekten Produkte (Number of detectable faults).
$\#D$	Anzahl der defekten Produkte (Number of detectable defective products).

Wiederholung der Überwachungskenngrößen

MF-Überdeckung (MF coverage), Anteil nachweisbare MF:

$$MC = \frac{\#DM}{\#MF}\Big|_{\mathrm{ACR}} \tag{1.20}$$

Phantom-MF-Rate (Phantom-MF: als MF klassifizierte CS):

$$\zeta_{\mathrm{Phan}} = \frac{\#PM}{\#DS}\Big|_{\mathrm{ACR}} \tag{1.21}$$

Ideale Formatkontrolle mit r redundanten Bits: $\zeta_{\mathrm{Phan}} = 0$ und

$$MC \geq 1 - 2^{-r} \tag{1.24}$$

Wertekontrolle mit Master-Checker-Paar:

$$MC = \eta_{\mathrm{Div}} \tag{1.25}$$

$$\zeta_{\mathrm{Phan}} = \zeta_{\mathrm{MS}} \cdot (1 - \eta_{\mathrm{Div}}) \tag{1.26}$$

MC	Fehlfunktionsüberdeckung (malfunction coverage), Anteil nachweisbare Fehlfunktionen.
$\#DM$	Anzahl der erkannten Fehlfunktionen (Number of detected MFs).
$\#MF$	Anzahl der Fehlfunktionen (Number of malfunctions).
ζ_{Phan}	Phantom-Fehlfunktionsrate.
$\#PM$	Anzahl der Phantom-MF, d.h. der korrekten DS, die als MF klassifiziert werden.
$\#DS$	Anzahl der erbrachten Service-Leistungen (Number of delivered services).
ζ_{MS}	Übereinstimmende Fehlfunktionsrate von Master und Checker.

Wiederholung Kenngrößen Fehlerbeseitigung

Eine vernünftige Fehlerbeseitigungsiteration reduziert die Fehleranzahl bzw. den Defektanteil nahezu auf den Anteil der erkannten Probleme:

- Abnahme der zu erwartendenen Fehleranzahl bei Reparatur:

$$\mu_{\mathrm{F}} = \frac{(1-p_{\mathrm{FD}})\cdot\mu_{\mathrm{CF}}}{1-\eta_{\mathrm{RF}}} \tag{2.15}$$

- Defektanzahl bei Ersatz erkannter defekter Bauteil:

$$DL = \frac{DL_{\mathrm{M}}\cdot(1-DC)}{1-DL_{\mathrm{M}}\cdot DC} \tag{1.85}$$

- Zu erwartende Anzahl defekter Produkte bei poisson-verteilter Fehleranzahl:

$$\mu_{\mathrm{DL}} = 1 - \mathrm{e}^{-\mu_{\mathrm{F}}} \tag{3.45}$$

μ_{F}	zu erwartende Fehleranzahl nach Test und Beseitigung aller erkennbaren Fehler.
p_{FD}	Fehlererkennungswahrscheinlichkeit (zu erwartende Fehlerüberdeckung).
μ_{CF}	Zu erwartende Anzahl der Fehler aus den Entstehungsprozessen.
η_{RF}	Erwartete Anzahl der bei der Reparatur entstehenden Fehler je ursprünglicher Fehler.
DL	Defektanteil nach Ersatz der Produkte mit erkannten Fehlern.
DL_{M}	Defektanteil nach der Fertigung vor Ersatz erkannter defekter Bauteile.
DC	Defektüberdeckung (defect coverage), Anteil der erkennbar defekten Produkte.
μ_{DL}	Zu erwartender Defektanteil.

Zuverlässigkeit ohne Fehlfunktionsbehandlung

Fehleranzahl und Zuverlässigkeit nach den statischen und einigen dynamischen Tests, insbesonderer aller fehlerorientiert gewählten*:

$$\mu_{\mathrm{F}}\left(N_0\right) = \mu_{\mathrm{FCR}} \cdot \left(1 - FC_{\mathrm{PT}}\right) \tag{1.62}$$

$$R_{\mathrm{F}}\left(N_0\right) = \frac{N_0}{K \cdot \mu_{\mathrm{F}}(N_0)} \tag{1.11, 1.63}$$

Weitere Abnahme mit weiteren zufällig ausgewählten Tests und pareto-verteilter Nachweislänge mit Formfaktor K:

$$R_{\mathrm{F}}\left(N_2\right) = R_{\mathrm{F}}\left(N_1\right) \cdot \left(\frac{N_2}{N_1}\right)^{K+1} \tag{1.66}$$

$\mu_{\mathrm{F}}\left(N_0\right)$	Zu erwartende Anzahl Fehler, die nach N_0 Vortests nicht erkannt und beseitigt sind.
N_0	Anzahl der dynamischen Tests aller Vortests zusammen.
μ_{FCR}	Zu erwartende Fehleranzahl aus den Entstehungs- und Reparaturprozessen insgesamt.
FC_{PT}	Fehlerüberdeckung der Vortests (Fault coverage of the pre tests).
$R_{\mathrm{F}}\left(N\right)$	Fehlerbezogene Teilzuverl. nach Beseitigung aller mit N Tests nachweisbaren Fehlern.
K	Formfaktor der Verteilung der Fehlfunktionsrate $(0 < K < 1)$.
N_1, N_2	Testanzahl mit bekannter oder gesuchter Zuverlässigkeit.
*	Vorausgesetzt, aller erkennbaren Fehler werden auch beseitigt.

Wiederholung Verlässlichkeit mit MT

Bei geringenFehlerfunktionsraten (ζ und ζ_{Phan}) werden fast alle Service-Leistungen ohne nennswerte Verfügbarkeitminderung erbracht. Zuverlässigkeitsverbesserung mit Kehrwert des Anteils der nicht erkennbaren DS:

- Abbruch ohne Service bei MF (einfachste MT):

$$\eta_{\mathrm{DS}} = 1 - (\zeta \cdot MC + \zeta_{\mathrm{Phan}}) \tag{1.29}$$

$$R_{\mathrm{MT}} = \frac{1}{(1-MC)\cdot\zeta} \tag{1.32}$$

- 3-Versionen-Mehrheitsentscheid (aufwändigste MT):

$$\eta_{\mathrm{DS}} = 1 - \zeta_{\mathrm{MS}} \cdot (1 - \eta_{\mathrm{Div}}) \tag{1.41}$$

$$R_{\mathrm{MT}} = \frac{R_{\mathrm{MS}}}{(1-\eta_{\mathrm{Div}})} - 1 \tag{1.44}$$

η_{DS}	Anteil der erbringbaren Service-Leistungen.
$\zeta_{[\mathrm{MS}]}$	Fehlfunktionsrate ohne MT, MS - jeweils für Master und Checker.
MC	Fehlfunktionsüberdeckung (malfunction coverage), Anteil nachweisbare Fehlfunktionen.
ζ_{Phan}	Phantom-Fehlfunktionsrate.
R_{MT}	Zuverlässigkeit mit Fehlfunktionsbehandlung (Reliability with malfunction treatment).
η_{Div}	Diversitätsrate, Anteil der nicht übereinstimmenden MF bei Mehrfachberechung.

Wiederholung Sicherheit

Fehlfunktionsbehandlung ohne zusätzliche Sicherheitsfunktionen erhöht die Sicherheit proportional mit der Zuverlässigkeit:

$$S_{\mathrm{MT}} = \frac{R_{\mathrm{MT}}}{\eta_{\mathrm{SE}}} \tag{1.49}$$

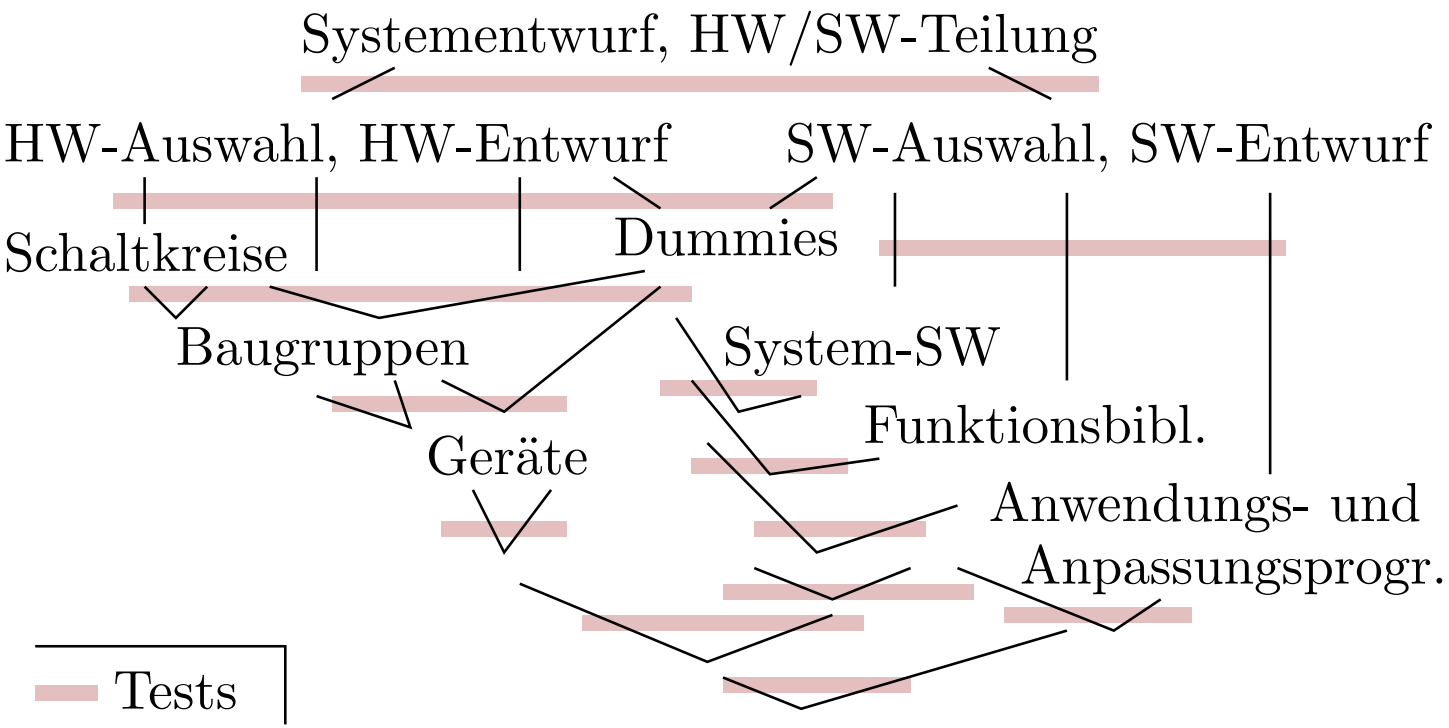
Eine Verringerung des Anteils der sicherheitskritischen MF von η_{SE} auf einen geringeren Wert $\eta_{\mathrm{SESF}} < \eta_{\mathrm{SE}}$ verlangt in der Regel zusätzliche Sicherheitsfunktionen, die die MF-Rate erhöhen und die Zuverlässigkeit auf einen Anteil $\eta_{\mathrm{RSF}} < 1$ verringern. Sicherheit insgesamt:

$$S_{\mathrm{MTSF}} = \frac{R_{\mathrm{MT}} \cdot \eta_{\mathrm{RSF}}}{\eta_{\mathrm{SESF}}} \tag{1.50}$$

S_{MT}	Sicherheit mit Fehlfunktionsbehandlung.
R_{MT}	Zuverlässigkeit mit Fehlfunktionsbehandlung (Reliability with malfunction treatment).
η_{SE}	Anteil der sicherheitsgefährdenden Fehlfunktionen.
S_{MTSF}	Sicherheit mit Fehlfunktionsbehandlung und zusätzlichen Sicherheitsfunktionen.
η_{RSF}	Zuverlässigkeitsverringerung durch MF der zusätzliche Sicherheitsfunktionen.
η_{SESF}	Verringerter Anteil sicherheitsgefährdender MF mit Sicherheitsfunktionen.

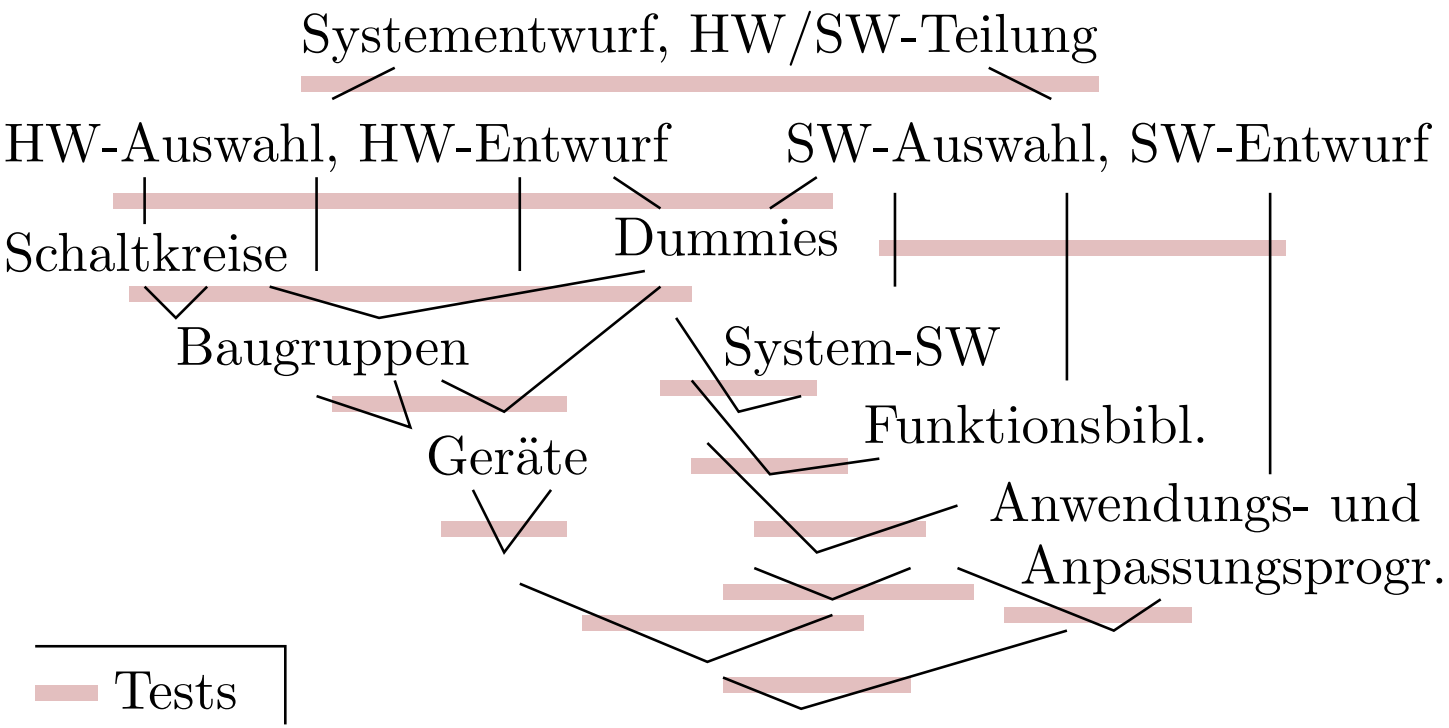
Test

Tests in Entwurfsprozessen (HW, SW)



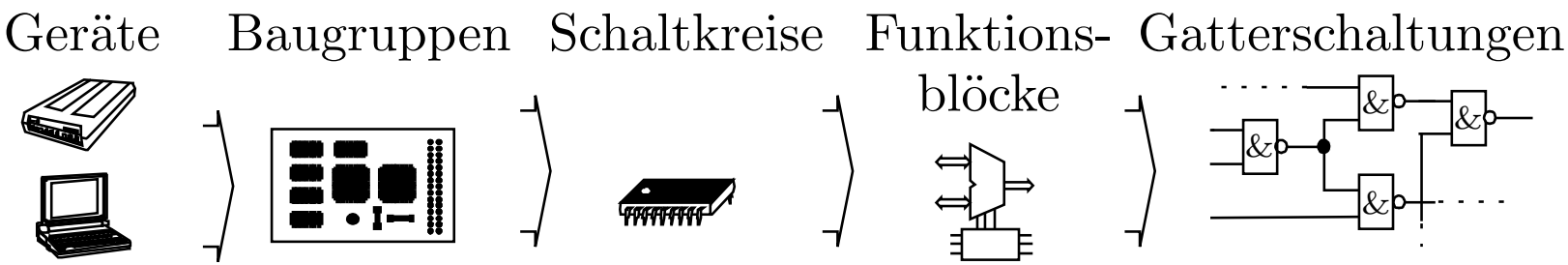
Entstehungsprozess sind komplexe Abläufe, in denen Know-How angereichert wird, also Software entsteht. Fehler manifestieren sich in aufgeschrieben Entwurfszwischen- und Endergebnissen. Vor ihrer Weiterverwendung Fehlerbeseitigungsiteration.

Tests in Entwurfsprozessen (HW, SW)



Dynamische Tests verlangen, dass das Entwurfsergebnis in ein ausführbares oder simulierbares System transferiert werden kann. Zu Beginn ausgeschlossen, mit Fortschreiten des Entwurfsprozesses tendentiell immer einfacher/häufiger. Davor in der Regel nur Inspektion (manuelle Sichtung).

Fertigungs- und Wartungstests der HW



- IT-Hardware ist modular aus austauschbaren Einheiten aufgebaut.
- Austauschbare Module max. so groß, dass mit akzeptabeler Ausbeute herstellbar.
- Interne Fehler werden vorzugsweise mit isolierten Tests gesucht. Die Tests übergeordneter Einheiten zielen auf Verbindungsfehler.
- Am schwierigsten ist die Testauswahl für den Kern der HW, hochintegrierte Schaltkreise aus tausenden oder millionen von Gattern, deren Anschlusswerte nur durch die umgebende Schaltung steuer- und beobachbar sind.
- Wartungstests sind idealerweise die Herstellertests. Praktisch oft gekürzte Versionen mit geringerer Fehlerüberdeckung, auch als Einschalttests, auch mit Lokalisierung zu tauschender Einheiten.

Inspektion

Inspektion (Review)

Inspektion, Sichtprüfungen (von lat. inspicere = besichtigen, betrachten). Angewendet auf:

- Dokumentationen (Spezifikation, Nutzerdokumentation, ...),
- Programmcode, Testausgaben,
- Schaltungsbeschreibungen, Konstruktionspläne, ...

besonders in frühen Entwurfsphasen, bevor die Zielfunktionen in simulierbarer oder ausprobierbarer Form beschrieben sind oder bei Testausgaben, bevor Sollwerte festgelegt oder Kontrollen programmiert sind.

Eigenenschaften von Inspektionen als Kontrollen:

- großer manueller Arbeitsaufwand,
- geringere Güte als maschinelle Kontrollen.
- Nachweis auch von nicht funktionalen Fehlern: Verstößen gegen Vereinbarungen und Standards, Antipattern*.
- Know-How-Weitergabe als positiver Zusatzeffekt.

* Beschreibungselemente, die die Übersichtlichkeit beeinträchtigen, die Kontrolle erschweren und die Fehlerentstehung begünstigen.

Kenngrößen einer Inspektion

Kenngrößen wie bei jedem anderen Test:

- Fehlerüberdeckung:

$$FC = \frac{\#DF}{\#F}\Big|_{\mathrm{ACR}} \tag{1.52}$$

- Phantom-MF-Rate des Tests:

$$\zeta_{\mathrm{PhanT}} = \frac{\#PM}{N}\Big|_{\mathrm{ACR}} \tag{1.53}$$

Weitere Kenngrößen zur Bewertung von Inspektionsprozessen [3]:

- Effizienz (EFC): Gefundene Fehler pro Mitarbeiterstunde.
- Effektivität (EFT): Gefundene Fehler je 1000 NLOC.

Kenngrößenschätzung oft getrennt für funktionale und andere Fehler.

$\#DF$	Anzahl der erkennbaren Fehler (Number of detectable faults).
$\#F$	Anzahl der Fehler (Number of faults).
ζ_{PhanT}	Phantom-MF-Rate des Tests (Phantom MF rate during test).
$\#PM$	Anzahl der Phantom-MF, d.h. der korrekten DS, die als MF klassifiziert werden.
N	Anzahl der Tests.
ACR	Geeignete Zählwertgrößen, typ. 100 ... 1000 ein- und nicht eingetretene Zählereignisse.
NLOC	Netto Lines of Code, Anzahl der Code-Zeilen ohne Kommentar und Leerzeilen.

Beispiel 4.1: Inspektion

Programmgröße: 10.000 NLOC, Arbeitsaufwand: 200 Stunden, 228 gefundene Fehler, davon 156 funktionale. Geschätzte Gesamtfehleranzahl (vor der Inspektion): 300, davon 200 funktionale. Wie groß sind:

a) *Inspektionsfehlerüberdeckung* FC*?*

b) *Effizienz?*

c) *Effektivität?*

NLOC Netto Lines of Code, Anzahl der Code-Zeilen ohne Kommentar und Leerzeilen.

Programmgröße: 10.000 NLOC, Arbeitsaufwand: 200 Stunden, 228 gefundene Fehler, davon 156 funktionale. Geschätzte Gesamtfehleranzahl (vor der Inspektion): 300, davon 200 funktionale. Wie groß sind:

a) *Inspektionsfehlerüberdeckung FC?*

b) *Effizienz?*

c) *Effektivität?*

	gesamt	funktionale Fehler	sonstige Fehler
$FC = \frac{\#DF}{\#F}$	$\frac{228}{300}$	$\frac{156}{200}$	$\frac{72}{100}$
Effizienz (EFC)	$\frac{228\ \text{Fehler}}{200\ \text{h}}$	$\frac{156\ \text{Fehler}}{200\ \text{h}}$	$\frac{72\ \text{Fehler}}{200\ \text{h}}$
Effektivität (EFT)	$\frac{228\ \text{Fehler}}{10.000\ \text{NLOC}}$	$\frac{156\ \text{Fehler}}{10.000\ \text{NLOC}}$	$\frac{72\ \text{Fehler}}{10.000\ \text{NLOC}}$

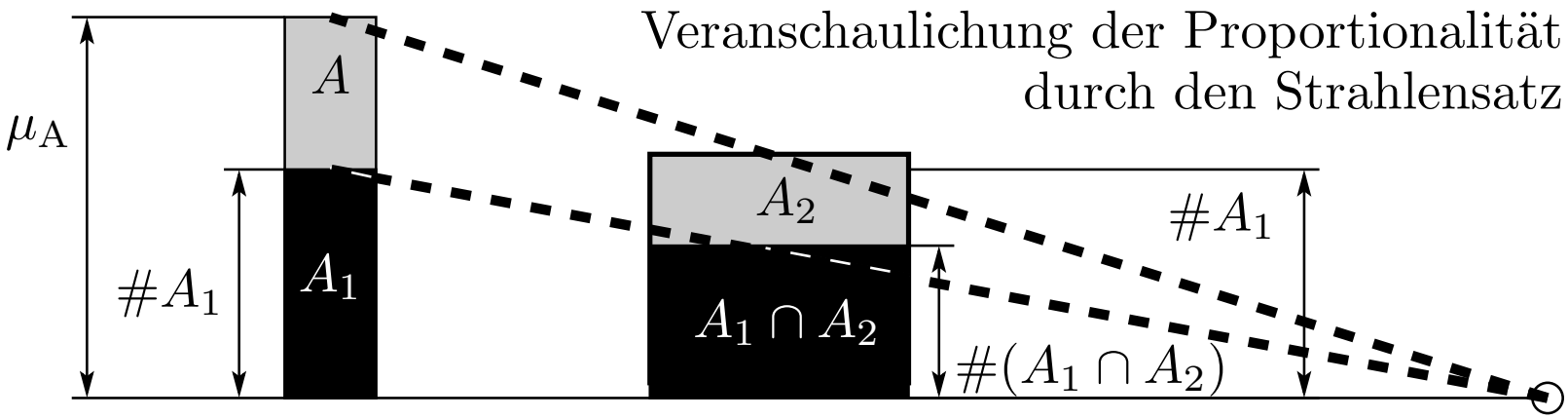
- Die Fehlerüberdeckung FC basiert auf schlecht überprüfbaren Schätzwerten für die Gesamtfehleranzahl.
- Die Angaben zur Effizienz und Effektivität sind objektiver und bewerten die Prozessgüte der Inspektion.

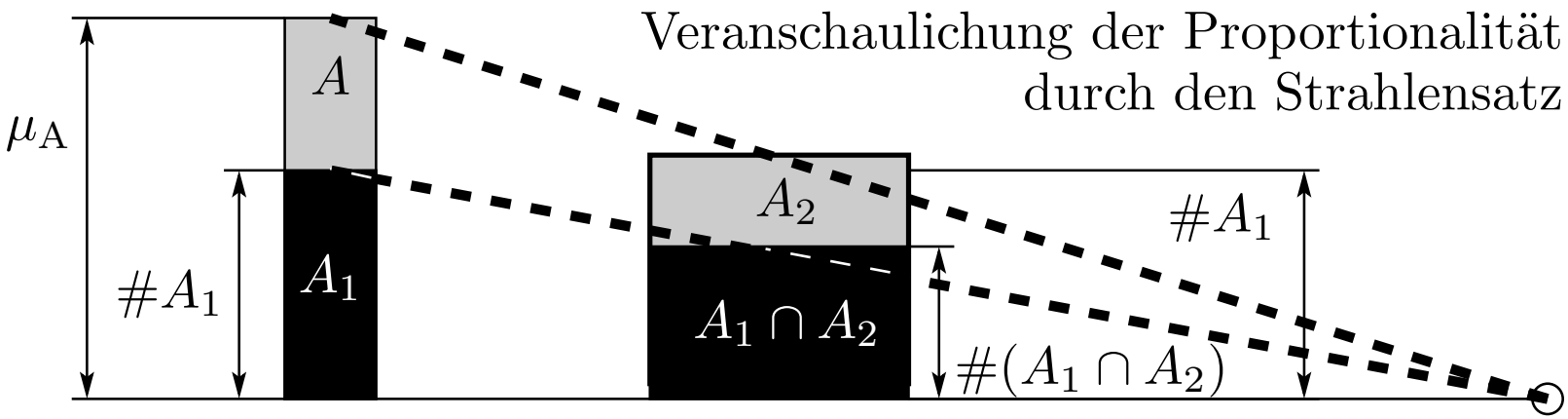
Capture-Recapture

Abgeleitet von einem Schätzer für die Größe von Tierpopulationen (z.B. von Vögeln in einem Gebiet) [2, 7, 4].

- Aus einer Menge A unbekannter Größe wird eine Menge A_1 von Tieren eingefangen, gekennzeichnet und freigelassen.
- Nach Vermischung der Population wird eine Menge A_2 von Tieren einfangen. Gekennzeichnete Tiere werden gezählt.

Bei tierunabhängiger Einfangwahrscheinlichkeit ergibt sich der Anteil der Tiere, die beim zweiten Einfangen gekennzeichnet sind, über den Strahlensatz:





$$\left.\frac{\#A_1}{\mu_A} = \frac{\#(A_1 \cap A_2)}{\#A_2}\right|_{\text{ACR}}$$

Zu erwartende Größe der Tierpopulation:

$$\left.\hat{\mu}_A = \frac{\#A_1 \cdot \#A_2}{\#(A_1 \cap A_2)}\right|_{\text{ACR}}$$

μ_A	Zu erwartende Anzahl aller Tiere.
A_1, A_2	Menge der beim ersten bzw. zweiten mal eingefangene Tiere.
$A_1 \cap A_2$	Menge der Tiere, die beim ersten und zweiten eingefangen werden.
#...	Anzahl der Elemente der Mengen.
ACR	Brauchbare Schätzwerte nur bei geeigneten Zählwertgrößen.

Fehler statt Tiere

Zwei Inspekteure i finden jeweils eine Menge von F_i Fehlern:

$$\hat{\mu}_{\mathrm{F}} = \left.\frac{\#F_1 \cdot \#F_2}{\#\left(F_1 \cap F_2\right)}\right|_{\mathrm{ACR}} \tag{1.1}$$

Die geschätzte Fehlerüberdeckung ist das Verhältnis der Anzahl der insgesamt von beiden Inspekteuren gefundenen Fehler $\#\left(F_1 \cup F_2\right)$ zur geschätzten Geamtfehleranzahl $\#F$:

$$\hat{\mu}_{\mathrm{FC}} = \left.\frac{\#\left(F_1 \cup F_2\right)}{\hat{\mu}_{\mathrm{F}}}\right|_{\mathrm{ACR}} = \left.\frac{\#\left(F_1 \cap F_2\right) \cdot \#\left(F_1 \cup F_2\right)}{\#F_1 \cdot \#F_2}\right|_{\mathrm{ACR}} \tag{1.2}$$

Gebunden an die Annahmen:

- Inspekteure erkennen die Fehler unabhängig voneinander.
- Alle Fehler haben dieselbe Erkennungswahrscheinlichkeit.

μ_{F}	Zu erwartende Gesamtfehleranzahl.
$\#F_1, \#F_2$	Anzahl der von Inspektor 1 bzw. Inspektor 2 gefundenen Fehler.
$\#(F_1 \cap F_2)$	Anzahl von beiden Inspekteuren gefundenen Fehler.
μ_{FC}	Zu erwartende Fehlerüberdeckung.

Beispiel 4.2: Capture-Recapture

- Inspekteur 1: 228 gefundene Fehler.
- Inspekteur 2: 237 gefundene Fehler.
- Schnittmenge: 105 Fehler.

Wie groß sind Gesamtfehleranzahl und Inspektionsfehlerüberdeckung?

- Inspekteur 1: 228 gefundene Fehler.
- Inspekteur 2: 237 gefundene Fehler.
- Schnittmenge: 105 Fehler.

Wie groß sind Gesamtfehleranzahl und Inspektionsfehlerüberdeckung?

$$\hat{\mu}_{\mathrm{F}} = \left.\frac{\#F_1 \cdot \#F_2}{\#(F_1 \cap F_2)}\right|_{\mathrm{ACR}} \tag{4.1}$$

$$\hat{\mu}_{\mathrm{FC}} = \left.\frac{\#(F_1 \cap F_2) \cdot \#(F_1 \cup F_2)}{\#F_1 \cdot \#F_2}\right|_{\mathrm{ACR}} \tag{4.2}$$

$$\hat{\mu}_{\mathrm{F}} = \frac{228 \cdot 237}{105} = 515$$

$$\hat{\mu}_{\mathrm{FC}} = \frac{\#F_1 + \#F_2 - \#(F_1 \cap F_2)}{\mu_{\mathrm{F}}} = \frac{228 + 237 - 105}{515} = 70\%$$

μ_{F}	Zu erwartende Gesamtfehleranzahl.
$\#F_1, \#F_2$	Anzahl der von Inspektor 1 bzw. Inspektor 2 gefundenen Fehler.
$\#(F_1 \cap F_2)$	Anzahl von beiden Inspekteuren gefundenen Fehler.
μ_{FC}	Zu erwartende Fehlerüberdeckung.

Vertrauenswürdigkeit der Schätzung

Die zu erwartende Fehleranzahl und Fehlerüberdeckung nach Gl. 4.1 und (Gl. 4.2) ergibt sich durch Multiplikation und Division von drei zufälligen Zählwerten. Bei Multiplikation und Division von Zuvallsvariablen addieren sich die Quadrate der Varianzkoeffizienten (siehe Folie 3.23 *Produkte und Quotienten von Zufallsvariablen*).

Grob überschlagen sind die relativen Intervallradien bei vergleichbarer Anzahl der nicht nachweisbaren Fehler etwa $\sqrt{3}$ mal so groß, wie bei einer Abschätzung aus einem Zählwert, z.B. mit einer Modellfehlermenge. Gleiche relative Intervallradien verlangen 3-mal so große Zählwerte.

Hinzu kommen systematische Schätzfehler:

- Capture-Recaptur unterstellt für alle Fehler dieselbe Erkennungswahrscheinlichkeit. In der Praxis reicht diese aber von *»Fehler kaum übersehbar«* bis *»Fehler fast nicht erkennbar«*.
- ...

- Capture-Recaptur verbietet Informationsaustausch zwischen den Inspekteuren. Falls es doch einen Informationsaustausch gibt, vergrößert der die Menge der von beiden Inspekteuren gefundenen Fehler $F_1 \cap F_2$ gegenüber einer unabhängigen Suche.
- Wenn die Inspekteure ihre Fehlerlisten voneinander abschreiben

$$\#F_1 = \#F_2 = \#\left(F_1 \cap F_2\right) = \#\left(F_1 \cup F_2\right)$$

ergibt sich nach

$$\hat{\mu}_{\mathrm{FC}} = \left.\frac{\#(F_1 \cap F_2) \cdot \#(F_1 \cup F_2)}{\#F_1 \cdot \#F_2}\right|_{\mathrm{ACR}} \tag{4.2}$$

eine Inspektionsfehlerüberdeckung von 100%, d.h. ein völlig aussagefreier Schätzwert.

Modellfehlerbasierte Abschätzung

Einbau von Kontrollfehlern (Mutationen) in das zu inspizierende Datenmaterial und Abschätzung der zu erwartenden Fehlerüberdeckung aus dem Anteil der gefundenen Kontrollfehler nach Gl. 1.52:

$$\hat{\mu}_{\mathrm{FC}} = \left.\frac{\#DF_{\mathrm{M}}}{\#F_{\mathrm{M}}}\right|_{\mathrm{ACR}} \tag{1.3}$$

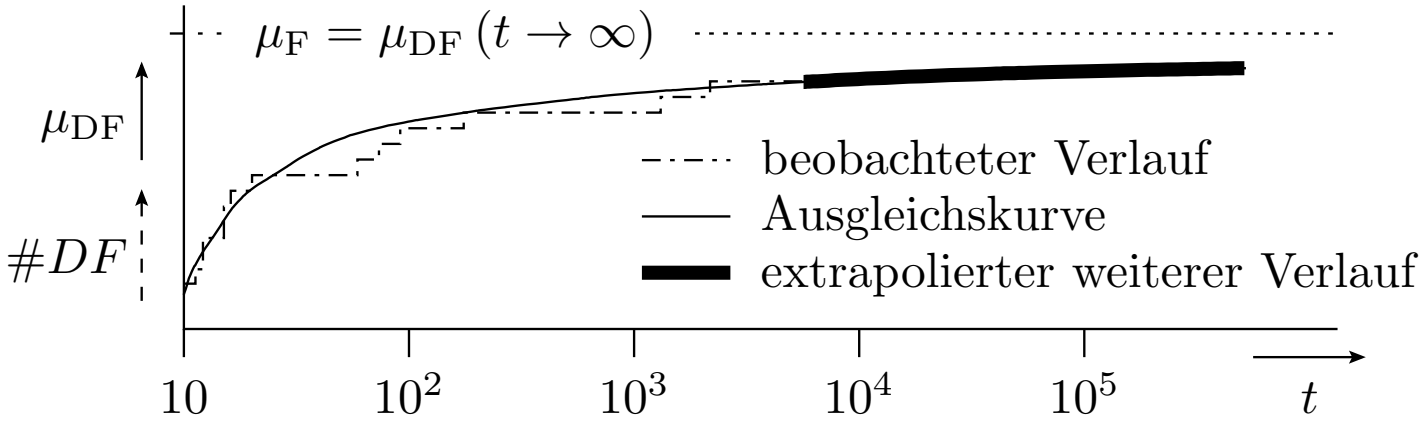
Geschätzte Anzahl der vorhandenen Fehler:

$$\hat{\mu}_{\mathrm{F}} = \left.\frac{\#DF}{\hat{\mu}_{\mathrm{FC}}}\right|_{\mathrm{ACR}} = \#DF \cdot \left.\frac{\#DF_{\mathrm{M}}}{\#F_{\mathrm{M}}}\right|_{\mathrm{ACR}} \tag{1.4}$$

Vergleich mit Capture-Recapture:

- Modellfehleranzahl exakt bekannt. Für $\hat{\mu}_{\mathrm{FC}}$ nur ein und für $\hat{\mu}_{\mathrm{F}}$ zwei zur Streuung beitragende Zählwerte.
- Aufwand für zusätzliche Fehlersuche: Capture-Recapture für $F_1 \cap F_2$, modellfehlerbasiert für Modellfehlermenge.
- Systematische Schätzfehler einfacher vermeidbar ...

$\hat{\mu}_{\mathrm{FC}}$	Geschätzte Fehlerüberdeckung.
$\#DF_{\mathrm{M}}$	Anzahl der erkennbaren Modellfehler (Number of detectable modeled faults).
$\#F_{\mathrm{M}}$	Anzahl der Modellfehler.
$\#DF$	Anzahl der nachweisbaren Fehler.



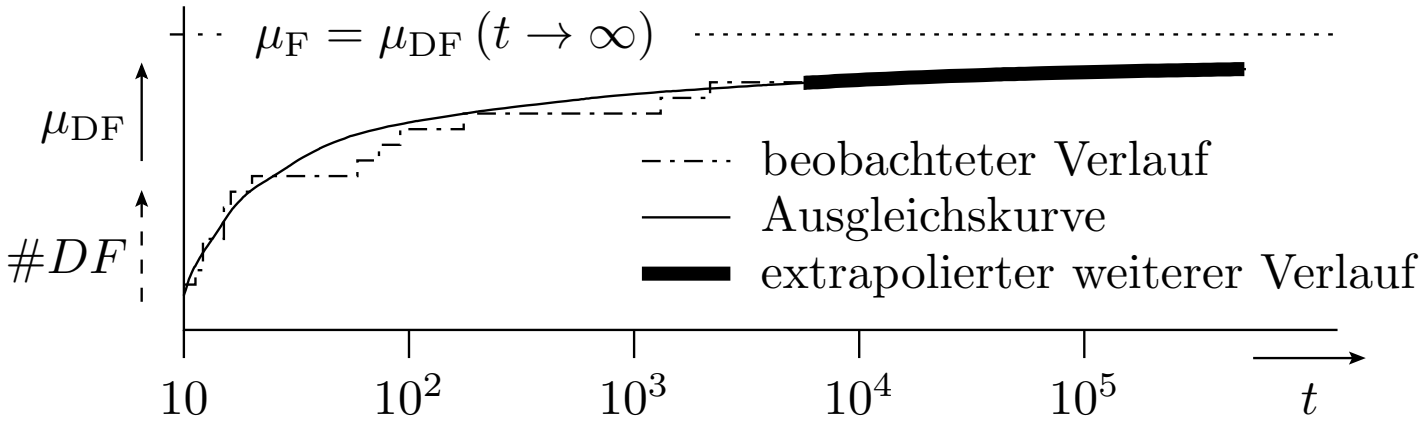
Maßnahmen zur Vermeidung systematischer Schätzfehler:

- Wahl einer Modellfehlerstichprobe mit ähnlicher Verteilung der Inspektionsfehlernachweiszeit.
- Keine Informationsweitergabe über die eingefügten Modellfehler, insbesondere auch nicht die unverfälschten Beschreibungen an die Inspekteure.

Theoretisch könnte man auch die Inspektionsüberdeckung ähnlich wie beim Zufallstest über die Verteilung der Nachweiszeit abschätzen, ...

μ_F	Zu erwartende Anzahl der vorhandenen Fehler.
μ_{DF}	Zu erwartende Anzahl der nachweisbaren Fehler.
$\#DF$	Anzahl der gefundenen Fehler (Number of detectable faults).
t_{Insp}	Inspektionsdauer in Mitarbeiterstunden.

Verteilung der Nachweiszeit



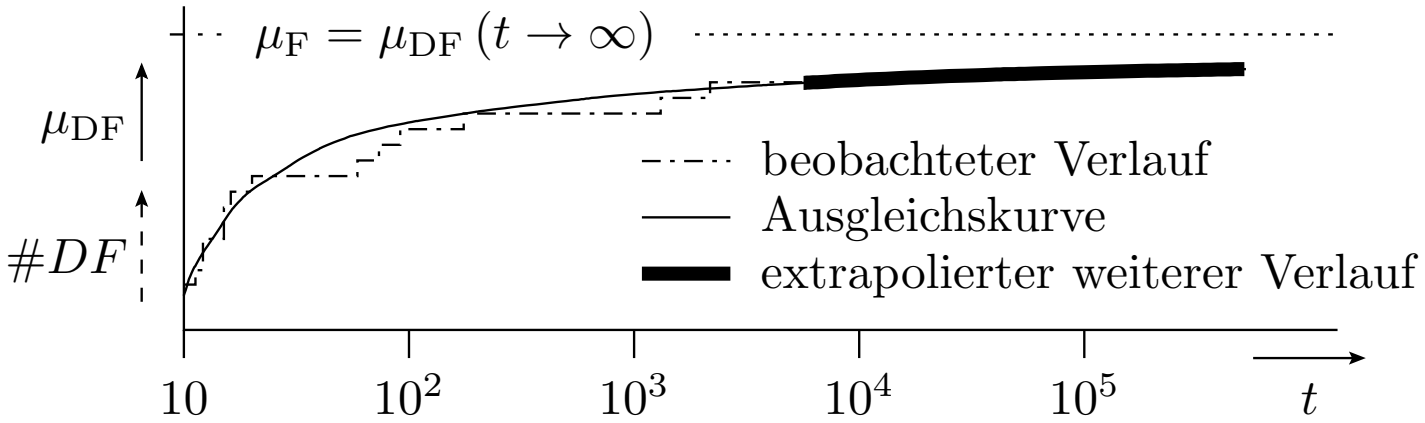
Auch für Inspektionen gilt in der Regel das Pareto-Prinzip. In einem kleinen Teil der Inspektionszeit wird die Mehrheit der Fehler gefunden. Deutet auf eine Pareto-Verteilung der Nachweiszeit (vergl. Gl. 3.78):

$$F_X(t) = \mathbb{P}[X \leq t] = \mu_{FC}(t) = \begin{cases} 0 & t \leq t_{min} \\ 1 - \left(\frac{t_{min}}{t}\right)^K & \text{sonst} \end{cases} \quad (1.5)$$

Schätzung von $\mu_{FC}(t)$ für länger Inspektionszeiten aus dem Verlauf für kürzere, wie im Bild ersichtlich, unsicher.

$K > 0$	Formfaktor der Pareto-Verteilung.
t_{min}	Skalenparameter der Pareto-Verteilung.

Inspektionsdauer und Effizienz



Die Effizienz ist proportional zum Anstieg und damit zur Dichtefunktion:

$$\frac{EFC(t)}{\mu_F \cdot 1\,\mathrm{h}} = f_X(t) = \frac{\mathrm{d}F_X(t)}{\mathrm{d}t} = \frac{K \cdot t_{\min}^K}{t^{K+1}} \tag{1.6}$$

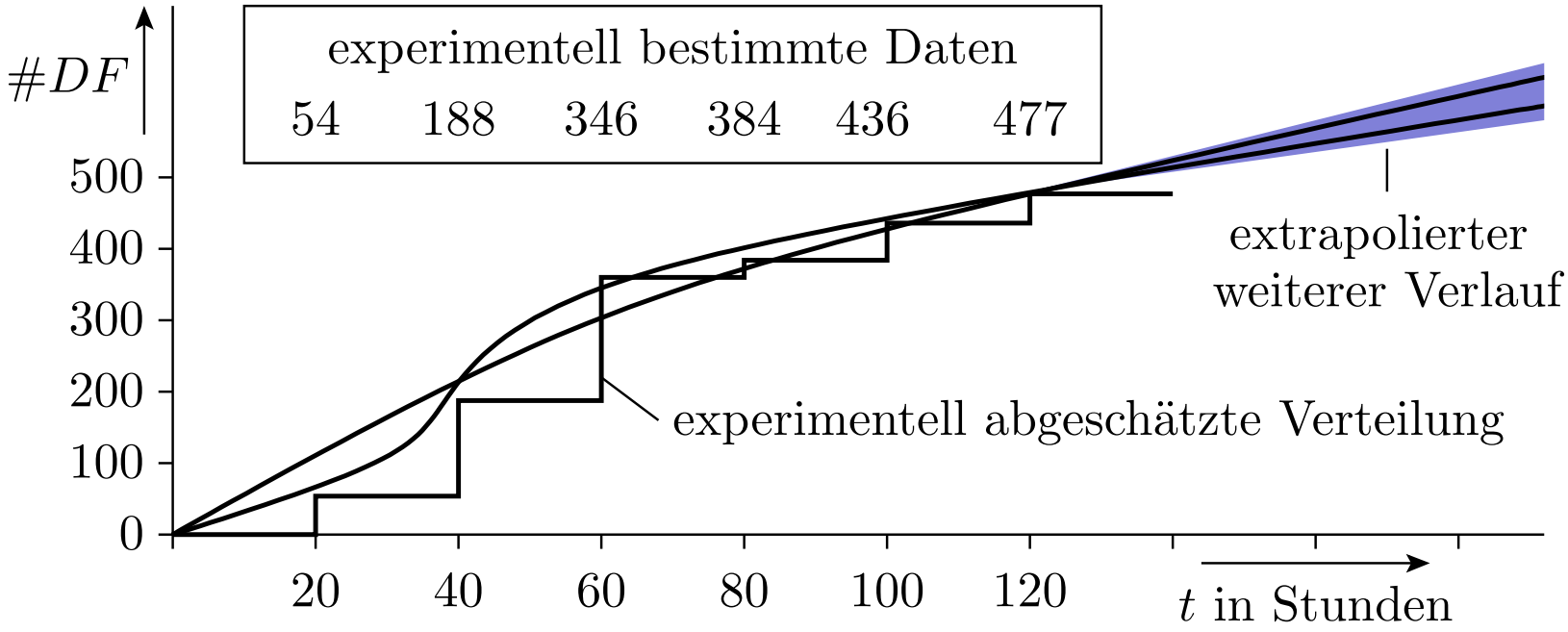
Mehr als umgekehrt proportionale Abnahme mit Inspektionsdauer t.
Begrenzung auf eine max. Inspektionsdauer sinnvoll.

EFC	Effizienz, gefundene Fehler pro Mitarbeiterstunde.
$f_X(t)$	Dichtefunktion der Inspektionfehlernachweiszeit.
μ_F	Zu erwartende Anzahl der vorhandenen Fehler.
$1\,\mathrm{h}$	Eine Mitarbeiterstunde.

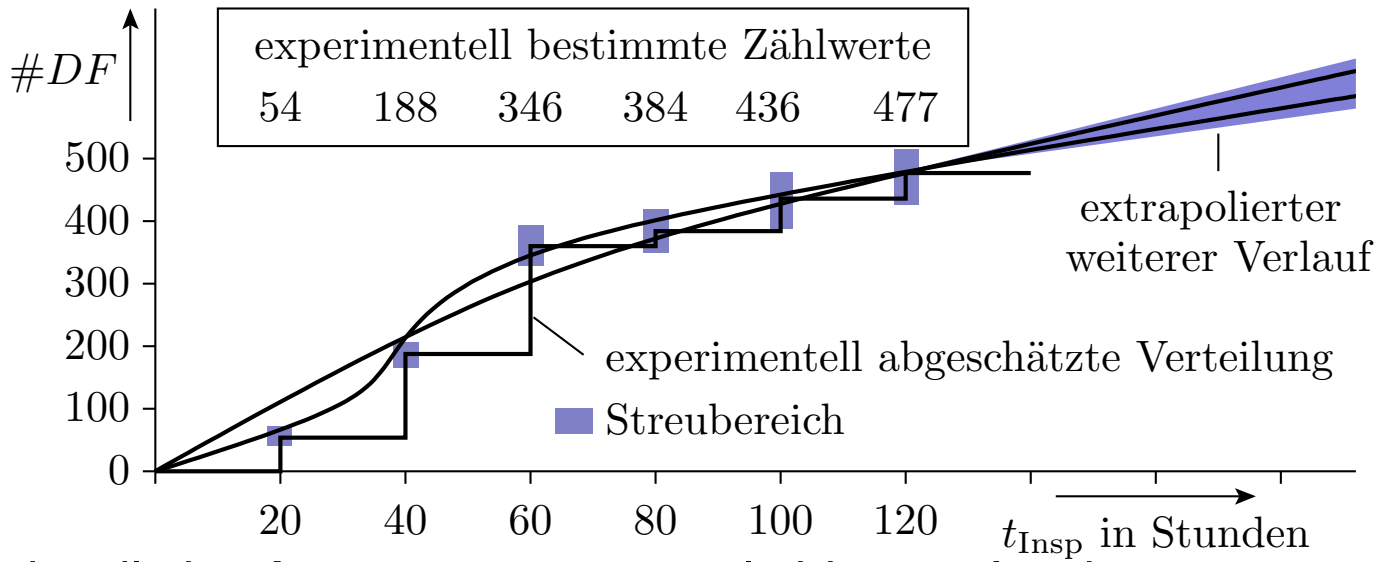
Experiment mit einem Inspekteur

Inspektion des Buchmanuskripts* plus Beispielprogramme:

- Anzahl der gefunden Fehler in Abhängigkeit von der Inspektionsdauer.



$\#DF$	Anzahl der gefundenen Fehler (Number of detectable faults).
t_{Insp}	Inspektionsdauer in Mitarbeiterstunden.
*	Bachelor-Arbeit von Yu Hong.



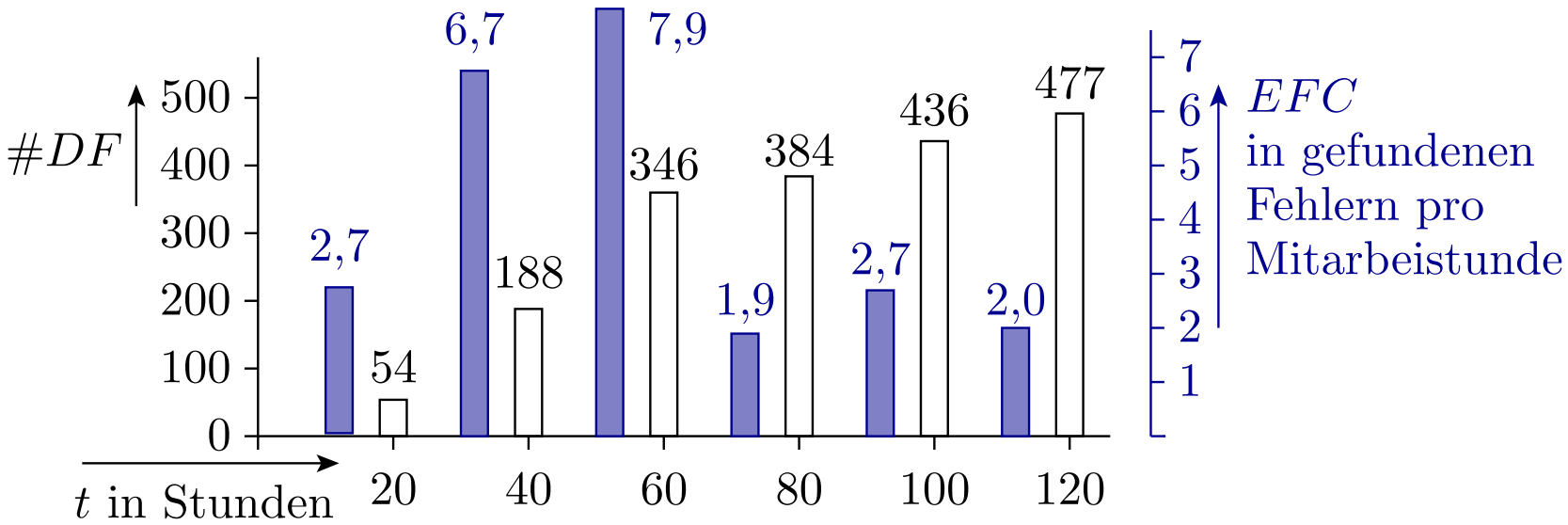
Unterschiedliche Approximationsmöglichkeiten für die weitere Abnahme der zu erwartenden Anzahl der nicht gefundenen Fehler, z.B. bei pareto-verteilter Nachweiszeit (Gl. 4.5):

$$\mu_{\text{DF}}(t) = \mu_{\text{F}} \cdot \left(1 - \left(\frac{t + t_{\min}}{t_{\min}}\right)^{-K}\right)$$

μ_{DF}	Zu erwartende Anzahl der nachweisbaren Fehler.
μ_{F}	Zu erwartende Anzahl der vorhandenen Fehler.
t_{Insp}	Inspektionsdauer in Mitarbeiterstunden.
$K > 0$	Formfaktor der Pareto-Verteilung.
$t_{\min}$	Skalenparameter der Pareto-Verteilung.

Untschiede zwischen Inspektion und Zufallstest

Bei einem Zufallstest nimmt die zu erwartende Anzahl der erstmalig nachweisbaren Fehler je Testschritt mit der Testdauer ab, weil die mittlere Fehlfunktionsrate je nicht nachweisbarer Fehler abnimmt.



Die Beispielinspektion hatte offenbar eine »Anlernphase«, in der die Effizienz, sprich die Anzahl der nachweisbaren Fehler pro Zeit mit der Inspektionsdauer zugenommen hat.

$\#DF$	Anzahl der nachweisbaren Fehler.
EFC	Effizienz, gefundene Fehler pro Mitarbeiterstunde.

- Beim dritten und vierten mal »Lesen des Buchs und der Aufgabentexte« nahm im Experiment nicht nur die Effizienz, sondern auch die Zeit dafür deutlich ab, obwohl ein erheblicher Anteil (ca. 25%) der Fehler noch nicht gefunden war.

Anzahl, wie oft gelesen	1	2	3	4
Anzahl der gefundenen Fehler	251	126	79	4
Zeitaufwand	50 h	70 h		

- Ein Mensch als Inspekteur ermüdet offenbar nach einiger Zeit und wird blind für Fehler, ...

These

Ein gute Inspektionstechnologie vermeidet die uneffizienten Einarbeitungs- und Ermüdungsphasen.

Inspektionstechniken

- Arbeit »geschickt« auf mehrere Inspekteure mit unterschiedlichen Rollen verteilen.
- Know-How-Weitergabe (Inspekteur ungleich Autor).
- Diversität ausnutzen »Vier Augen sehen mehr als zwei«.

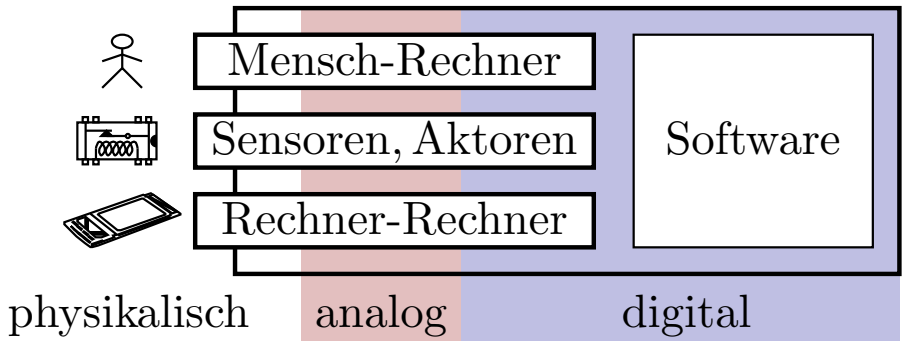
Einteilung der Inspektionstechniken

- Review in Kommentartechnik: Korrekturlesen und Dokument mit Anmerkungen versehen.
- Informales Review in Sitzungstechnik: Lösungsbesprechung in der Gruppe, Vier-Augen-Prinzip. Nimmt die Monotonie, steigert die Aufmerksamkeit, fördert den Wissensaustausch.
- Formales Review in Sitzungstechnik: Festlegen von Rollen (Leser, Moderator, Autor, Inspekteure) und Abläufen, ...

[Lesegeschwindigkeit, Rollendisziplin, Vermeidung Langeweile, überhitzte Emotionen; Reifen Gruppennormen, Sägezahnverlauf]

Funktionstest

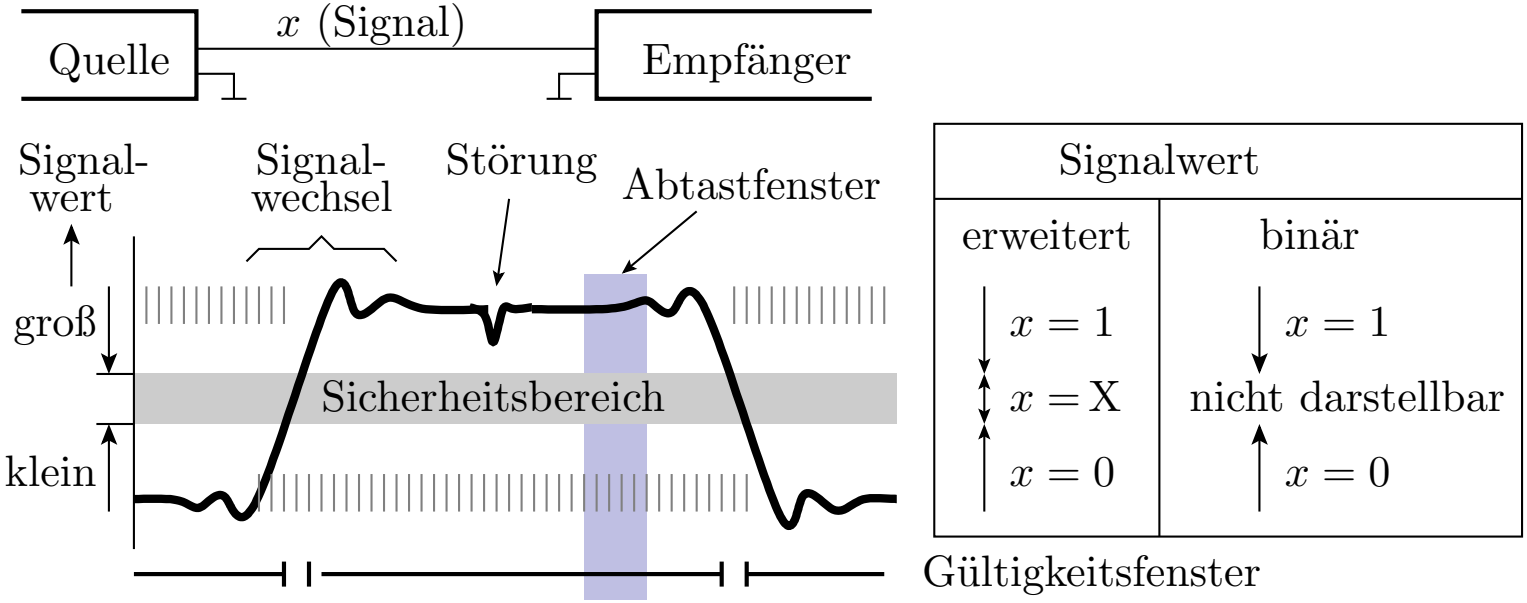
Typ Systemstruktur von IT-Systemen



Ein IT-System kommuniziert mit der Welt über analoge Signale, arbeitet aber intern überwiegend digital.

- Ausprobieren in der Anwendungsumgebung sind Tests mit analoger Ein- und Ausgabe, aber
- Analoge Verarbeitung:
  - Nur an den Schnittstellen zur physikalischen Umgebung.
  - Überschaubare Funktionsvielfalt: Wandlung physikalisch $\Leftrightarrow$ elektrisch, Verstärkung, Bandbegrenzung, Wandlung analog $\Leftrightarrow$ digital.
- Kompliziertere Verarbeitungsfunktionen werden digital realisiert,
- noch kompliziertere Funktionen in SW.

Digitalisierung



Digitalisierung: Informationstdarstellung durch Bits

- Werteunterteilung in groß, klein und ungültig,
- Abtastung im Gültigkeitsfenster.
- MF-Toleranz gegenüber Rauschen, induktivem und kapazitivem Übersprechen, Fertigungsstreuungen, Alterung, ...

0, 1, X Logische Signalwerte für klein, groß und ungültig.

Vorteile digitaler Verarbeitung

- Toleranz gegenüber Fertigungsstreuungen, ...

Trotz der viel größeren Anzahl von Berechnungen und Signalen (z.B. für Addition einen Volladdierer je Bit statt Summationsverstärker):

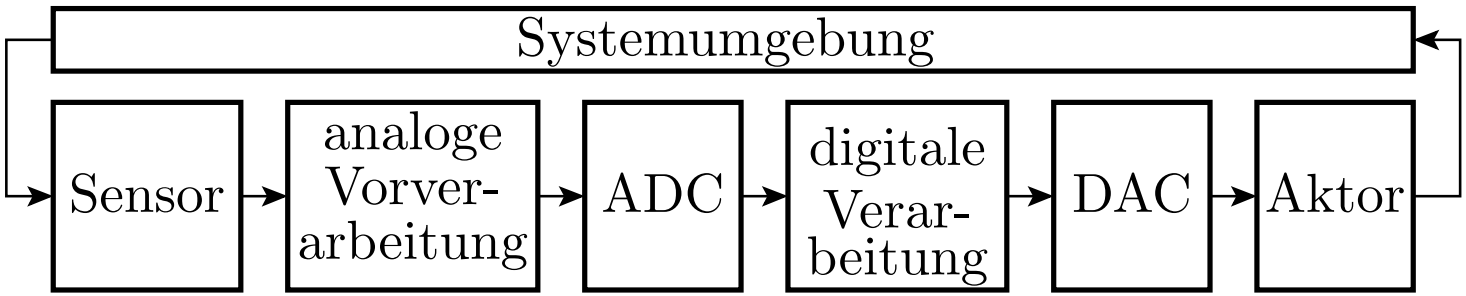
- geringere Fertigungskosten und Entwurfskosten*,
- kleiner, schneller, genauer zuverlässiger.

Zusätzlicher funktionaler Gestaltungsspielraum:

- Datenspeicherung,
- sequentielle und SW-gesteuerte Abarbeitung, ...
- Fehlerbeseitigung durch Programmänderung,
- Einschalttest, einprogrammierbare Testhilfen, Überwachungs- und Fehlerbehandlungsfunktionen.
- keine Akkumulation der Verfälschungen durch Rauschen und anderer Störungen bei der Verarbeitung und Übertragung,
- ...

* Komplexe Digitalschaltungen werden heute wie Software entworfen.

Typische Verarbeitungskette

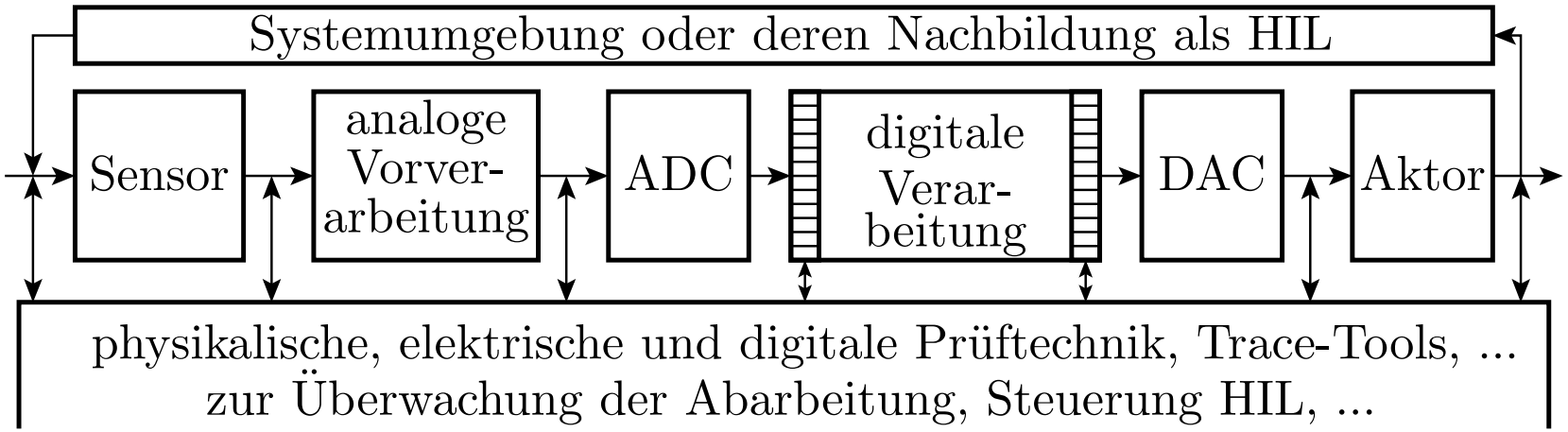


- Sensor, analoge Vorverarbeitung, analog-digital-Wandlung,
- digitale Verarbeitung
  - in der Regel mit Rechnern, durch Software,
  - die Hardware stellt für die SW die Grundfunktionen, zeitkritische Funktionen, ... bereit.
  - Zwischen der HW und der Anwendungs-SW liegen in der Regel noch Schichten: Hardware-Abstraktion, Betriebssystem, ...
- digital-analog-Wandlung,
- analoge Nachbearbeitung, Aktor.

Die oberste Testebene ist das Ausprobieren in der Systemumgebung:

- Bereitstellung physikalischer Eingaben und
- Kontrolle der Ausgaben.

Systemtest, HIL

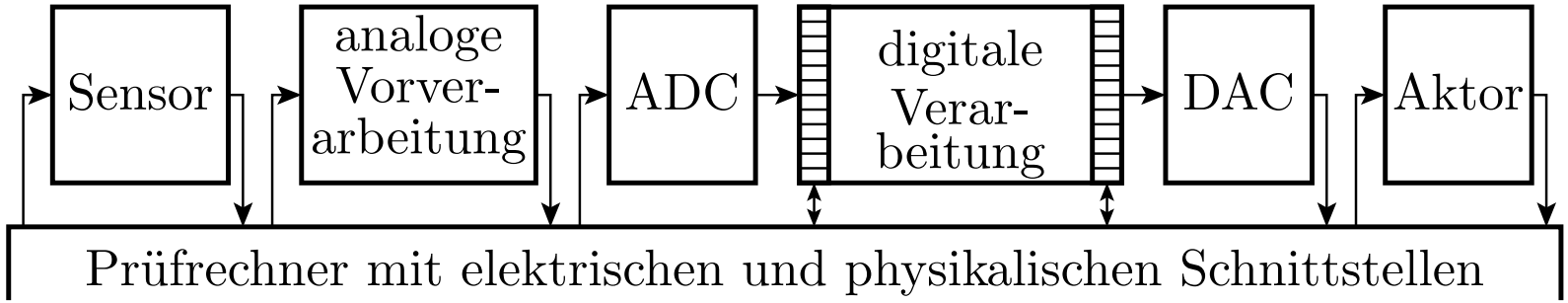


Das Ausprobieren eines ungetesteten Systems (z.B. Motorsteuerung) in der realen Umgebung kann gefährlich sein. Alternative HIL (Hardware in the Loop, Attrappe für die Systemumgebung).

Die Überwachung der Tests und die Fehlersuche erfordert in der Regel zusätzliche Mittel und Testhilfen zur Beobachtung systeminterner Signale, Daten und Verarbeitungsabläufe ($\longleftrightarrow$ zur Prüftechnik).

ADC	Analog-Digital-Umsetzer.
DAC	Digital-Analog Umsetzer.
HIL	Nachbildung der Systemumgebung.

Isolierte Tests



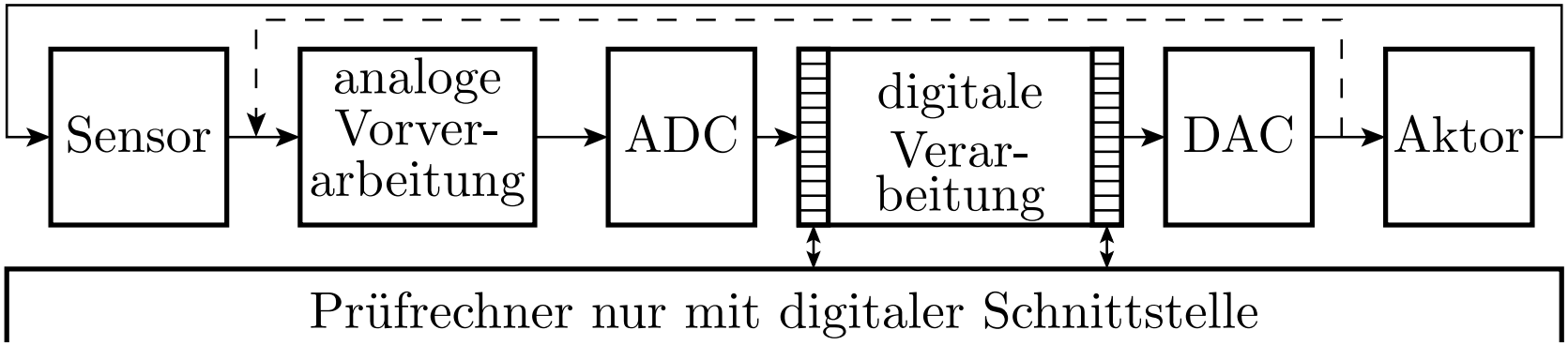
Gründliche isolierter Funktionstests nicht digitale Komponenten (Sensoren, Vorverarbeitung, ADU, DAU und Aktoren):

- durch Kontrolle der Kennlinien, Frequenzgänge, Verzerrungen, ...
- mit analoger Messtechnik

vorzugsweise isoliert von der digitalen Verarbeitung. Erfordert:

- Kontaktierungsmöglichkeiten der Ein- und Ausgänge zum Anschluss geeigneter Prüftechnik,
- Unterbrechungsmöglichkeit des normalen Signalflusses an Testpunkten, ...
- Der Zugriffe auf die bitparallelen digitalisierten Signale erfolgt in der Regel über Testbus seriell (siehe Folie 4.48 *Boundary-Scan*).

Loop-Test



Statt externer Prüftechnik Nutzung der digital-analog- (DAC) und analog-digital-Wandler (ADC) für die Bereitstellung und Auswertung der analogen Testsignale. Erfordert Testmodi mit Signalfluss:

- DAC, (analoge Vorverarbeitung,) ADC und
- DAC analoge Vorverarbeitung, Aktor, Sensor, ADC.

Kontrollen Übertragungsfunktion, Frequenzgang, Verzerrung, Rauschen, ... der entstehenden Loops in den Testmodi erfordert:

- ausreichend schnelle und genaue ADC und DAC,
- Umschalter im Signalfluss, die die Signale nicht verfälschen.
- Auch nutzbar für Selbst-, Einschalt- und Wartungstests.

Prüfgerechter Entwurf

- Anschluss von Prüftechnik zur Überwachung von Systemtests,
- HIL (Hardware-in-the-Loop),
- isolierte Tests analoger Komponenten,
- Loop-Tests, ...

verlangen Vorkehrungen, die ab Beginn des Entwurfsprozesses mit berücksichtigt werden müssen. Das gilt auch, wie im weiteren gezeigt, für den Test der digitalten Verarbeitung, SW-Tests und Selbsttests.

Prüfgerechter Entwurf

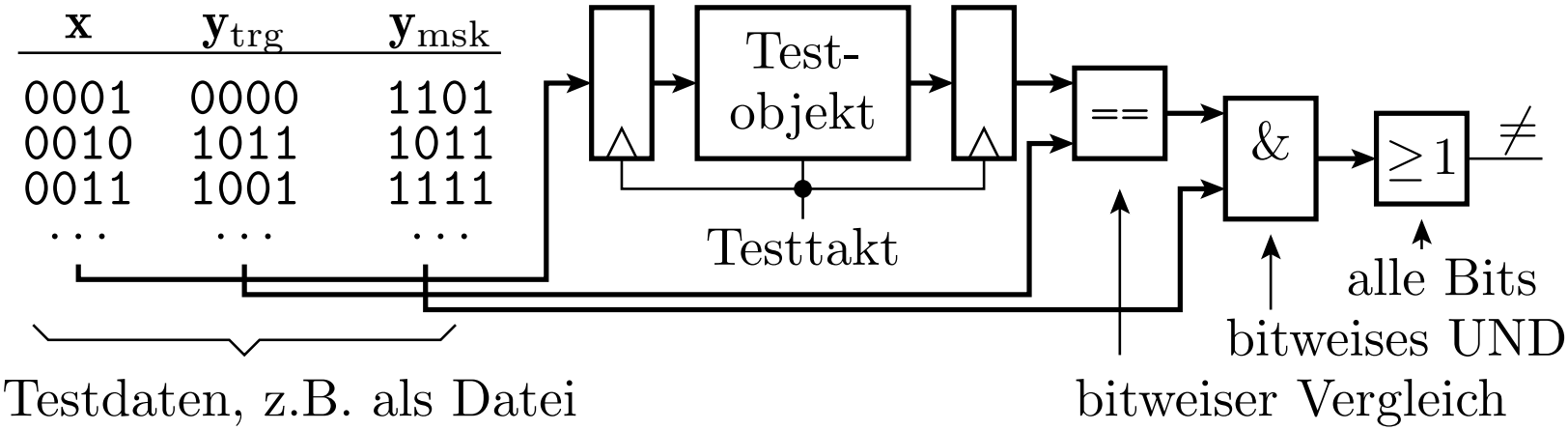
Sammlung von Maßnahmen, um Tests entwerfen und durchführen sowie die Testergebnisse auswerten zu können.

Die Art der Testdurchführung muss schon sehr früh in den Entwurfsprozess für ein neues System einfließen.

Ohne prüfgerechten Entwurf kein ausreichender Test, keine ausreichende Verlässlichkeit und damit nicht nutzbar.

Digitale Schaltung

Test digitaler Bausteine

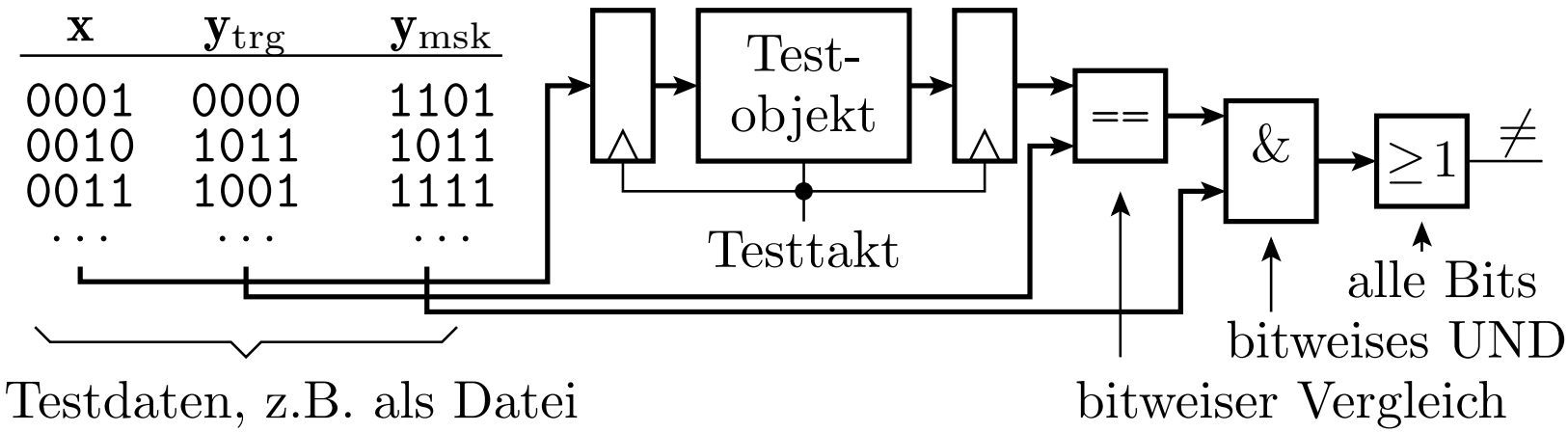


Wiederhole für jeden Taktschritt des Tests:

- Bereitstellung logischer Eingabewerte und
- Abtasten und Auswertung der vorherigen Ausgaben.

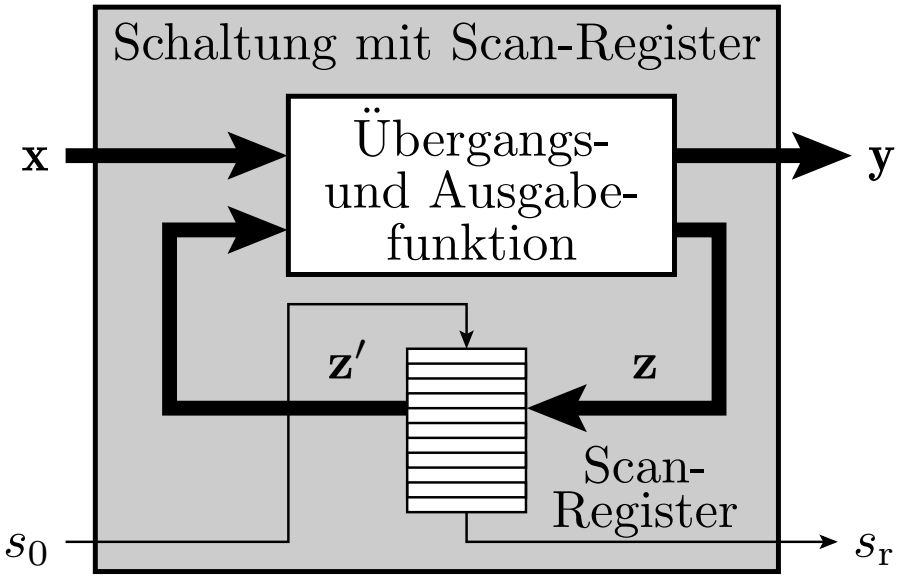
Auswertung vorzugsweise Vergleich mit Sollwerten unter Ausmaskierung von Ausgabebits ohne definierten Sollwert.

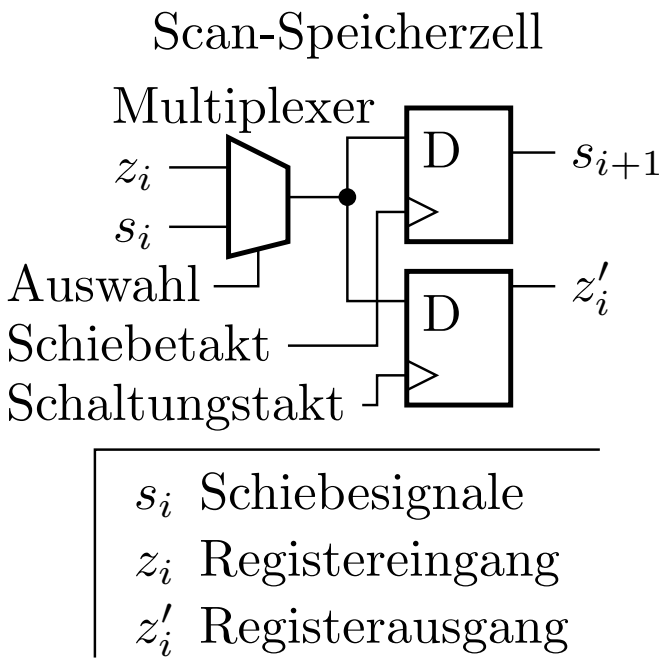
$\mathbf{x}$	Testeingaben.
$\mathbf{y}_{\mathrm{trg}}$	Sollwerte der Testausgaben.
$\mathbf{y}_{\mathrm{msk}}$	Maskenwerte zum Ausschluss von Testausgaben vom Soll-Ist-Vergleich.
$\neq$	Vergleichsfehler.



- Speicherelemente im Testobjekt sind vor dem Test zu initialisieren.
- Kontrolle der Signalverzögungen durch Ergebnisabtastung mit mehrfacher Aufzeichnungsfrequenz.
- Zusätzliche Stimulierung oder Beobachtung schaltungsinterner Signale.
- Tester auch als Kombination Signalgenerator Logikanalysator.

Scan-Verfahren

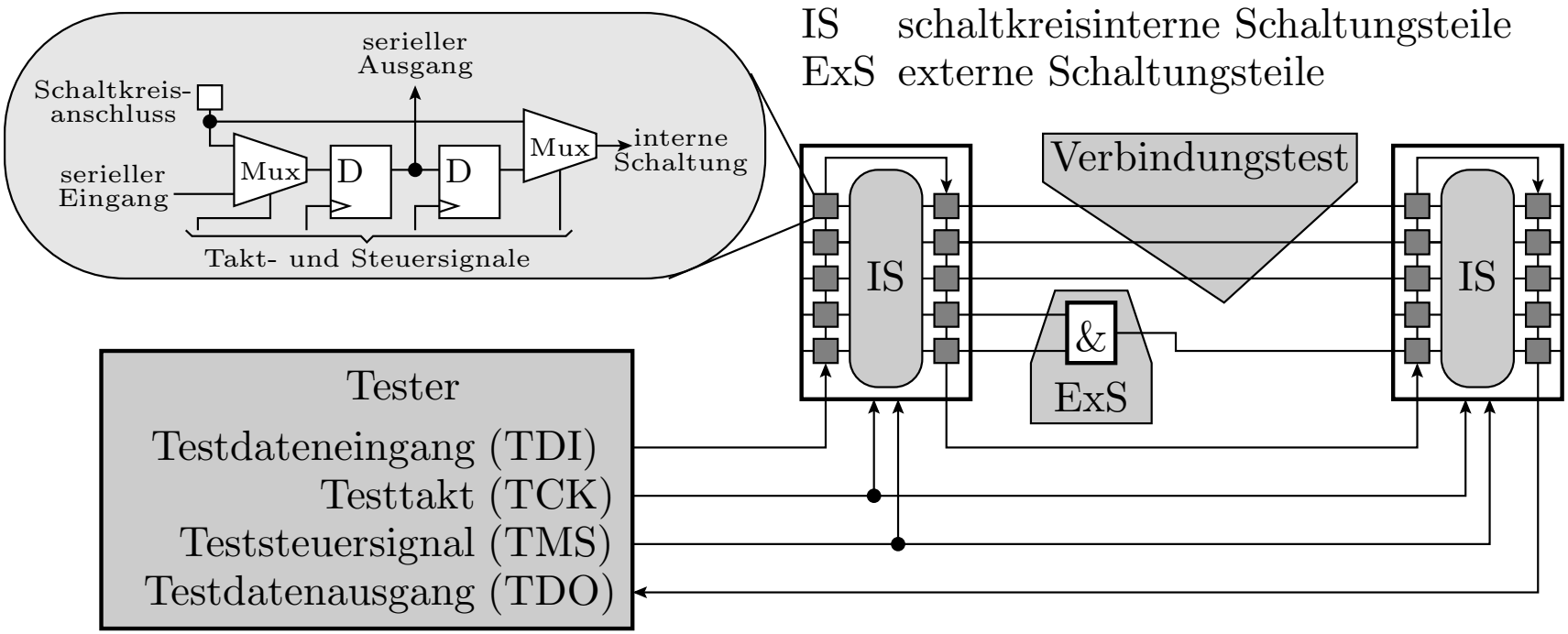




Vereinfachung von Fehlernachweis und -lokalisierung durch Lese- und Schreibzugriff auf interne Signale und Speicher. Datentransfer vorzugsweise seriell. Im Bild ist jede Speicherzelle um einen Multiplexer und eine Schiebezelle erweitert zur Bereitstellung der Funktionen:

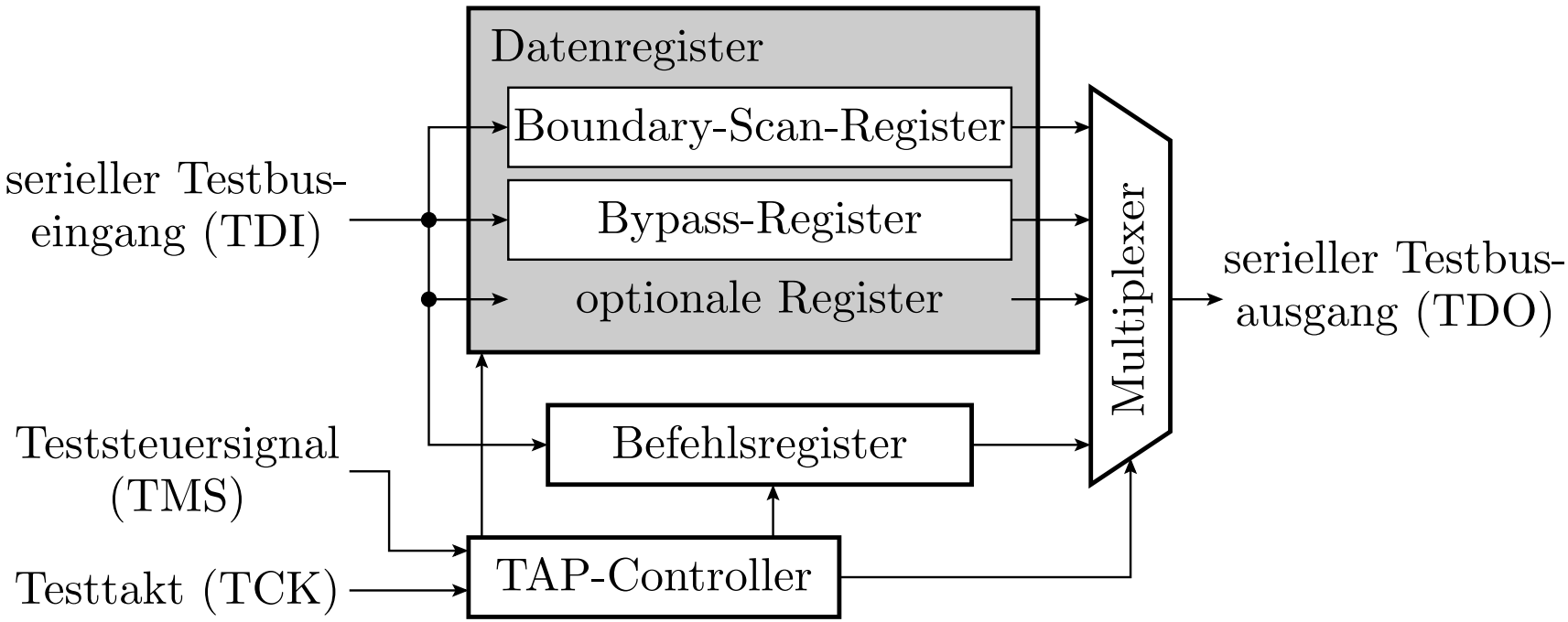
- Capture: Übernahme aus der Schaltung in die Schiebezellen,
- Shift: serielles Auslesen und neu beschreiben und
- Update: Übergabe seriell eingelesene Daten an Schaltung.

Boundary-Scan



- Scan-Register an den Anschlüssen digitaler Schaltkreise.
- Ermöglicht den isolierten Test der schaltkreisinternen Funktion und Schaltkreisumgebung auf Baugruppen (siehe Abschn. 4.1.5).
- Der standardisierte JTAG-Testbus besitzt 4 Bussignale: TDI, TDO, TCK und TMS zur Verkettung der Testbusse vieler Schaltkreise.

JTAG-Testbusarchitektur der Schaltkreise

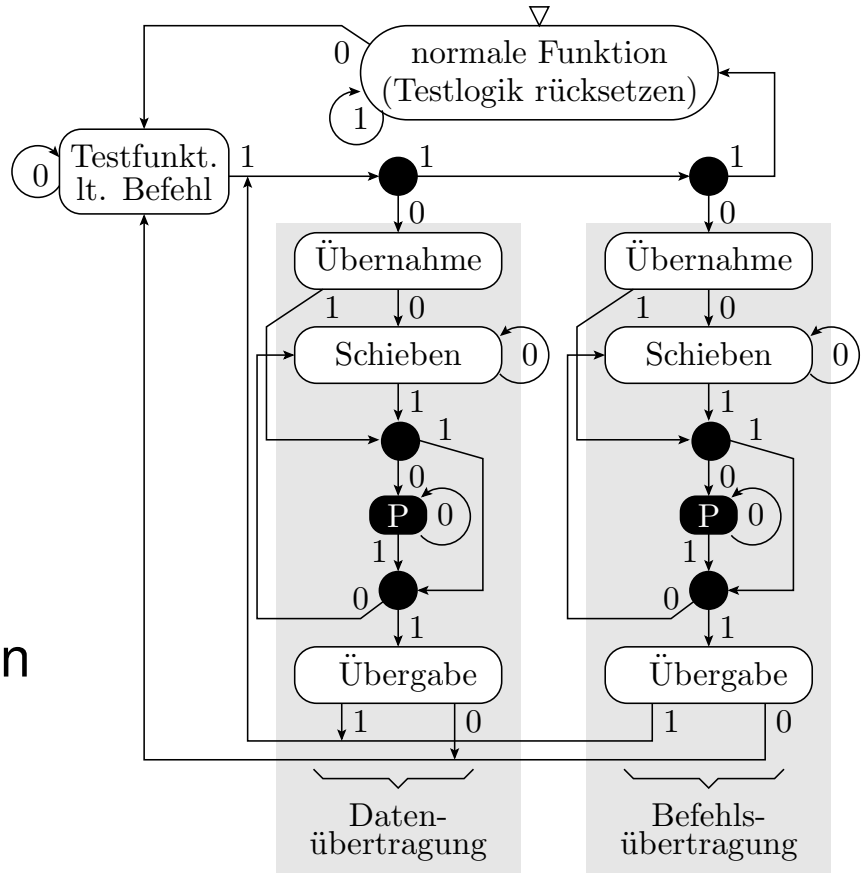


Eine Boundary-Scan-Implementierung umfasst:

- den TAP- (Test Access Port) Controller
- ein Befehlsregister
- mehrere Testdatenregister: mindestens Boundary-Scan- und Bypass-Register, optional Vendor-, ID-, BIST-, Programmier-Register, ..., Auswahl mit TAP).

JTAG-Busprotokoll

- Automat mit 16 Zuständen
- Kantenauswahl über TMS (**T**est **M**ode **S**elect)



Typischer Testbeginn:

- Befehlsregister lesen (Kontrolle JTAG-Kette*),
- Bauteilnummern lesen (Bestückungskontrolle),
- Einen Teil der Schaltkreise auf Bypass setzen. Für die anderen ein Datenregister auswählen.

Verbindungstest:

- BS-Register auswählen,
- Wiederhole alle Testschritte
  - BS-Register Übernahme, Schieben, Übergabe

* Übernahme 01 in Bit0 und Bit1 zur Kontrolle auf Unterbrechungen der Schiebekette.

Über das Befehlsregister lässt sich eine

- beliebige Anzahl von Datenregistern adressieren und
- eine beliebige Anzahl von »Testfunktionen laut Befehl« steuern.

Typische zusätzliche Testfunktionen:

- lesbare Hersteller- und Bauteilidentifikationsregister für den Bestückungstests.
- Laden und Lesen von Programm- und Konfigurationsdaten.
- Steuerung integrierter Debugger-Funktionen,
- Steuerung von Selbsttests, ...

Software

Testrahmen

Zu testende Programmbausteine werden für den Test in einem Programmrahmen eingebettet, der Testeingaben bereitstellt und Testausgaben auswertet und mit dem er gemeinsam in ein ausführbaren Programm übersetzt wird.

Im einfachsten Fall ist der Testrahmen ein ausführbares Programm und das Testobjekt ein aufgerufenes Unterprogramm.

Nützliche Funktionen zur Unterstützung von Test und Fehlersuche, bereitzustellen von der Systemsoftware oder Hardware:

- Schrittbetrieb, Haltepunkte,
- Lesen und Verändern von Variablen im Haltezustand,
- Trace-Aufzeichung von Variablen und Ausgabesignalen, ...

Ein Testobjekt und sein Testrahmen

Beispieltestsobjekt: Unterprogramm zur Quadrierung:

```
uint32_t quad(int16_t a){ // Quadratberechnung
  return (uint32_t)a * a; // Warum mit Typcast?
};
```

Testbeispiele sind Tupel aus Eingaben und Sollausgaben. Man kann dafür einen neuen Datentyp definieren:

```
typedef struct{
  int16_t x;                    //Eingabe
  uint32_t y;                   //Sollausgabe
} test_t;
```

Ein »struct« ist eine Zusammenfassung aus bereits definierten Datentypen.

Testsatz

Eine Testsatz als Menge von Tests ist im einfachsten Fall ein initalisiertes Feld von Testbeispielen

```
test_t testsatz[] ={{<Tupel1>},{...},...};
```

Testbeispiele für die Quadratberechnung:

```
test_t testsatz[] ={        //Eingabe   Sollwert
  {0, 0},                   // 0x0000   0x00000000
  {1, 1},                   // 0x0001   0x00000001
  {9, 81},                  // 0x0009   0x00000051
  {-5, 25},                 // 0xFFFB   0x00000019
  {463, 214369},            // 0x01CF   0x00034561
  {0x7FFF, 1073676289}      // 0x7FFF   0x3FFF0001
};
```

Welche Testbeispiele führt das Programm falsch aus?*

* Vermutlich ergibt -5 konvertiert in uint32_t mal -5 keinen positiven Wert.

Testrahmen

Programm, das in einer Schleife alle Testbeispiele abarbeitet und die Ergebnisse kontrolliert oder zur Kontrolle ausgibt:

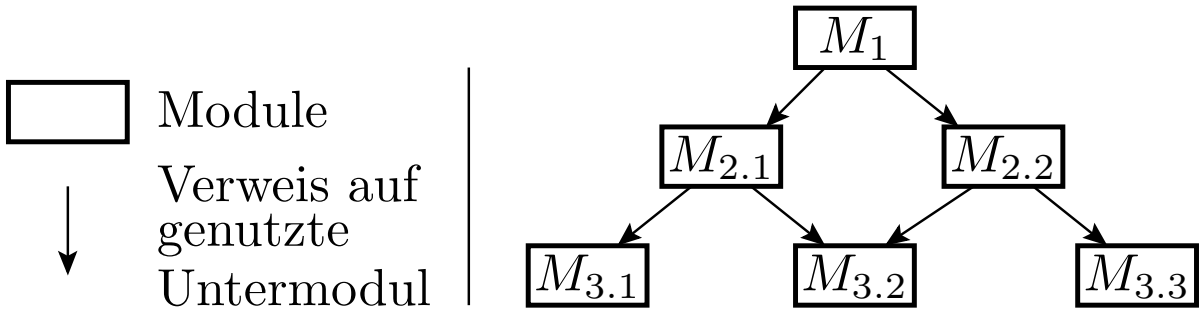
```
int main(){
 uint8_t idx, err_ct=0;
 uint32_t erg;
 for (idx=0; idx<6;idx++){
 erg = quad(testsatz[idx].x);//Istwert
 if (erg != testsatz[idx].y) //Soll/Ist-Vergl.
  err_ct++;                  //Fehlfkt. zählen
 }
}
```

Testdurchführung mit dem im Simulator im Debug-Modus:

- Unterbrechungspunkt vor dem Fehlerzähler.
- Bei jeder Fehlfunktion Halt am Unterbrechungspunkt.

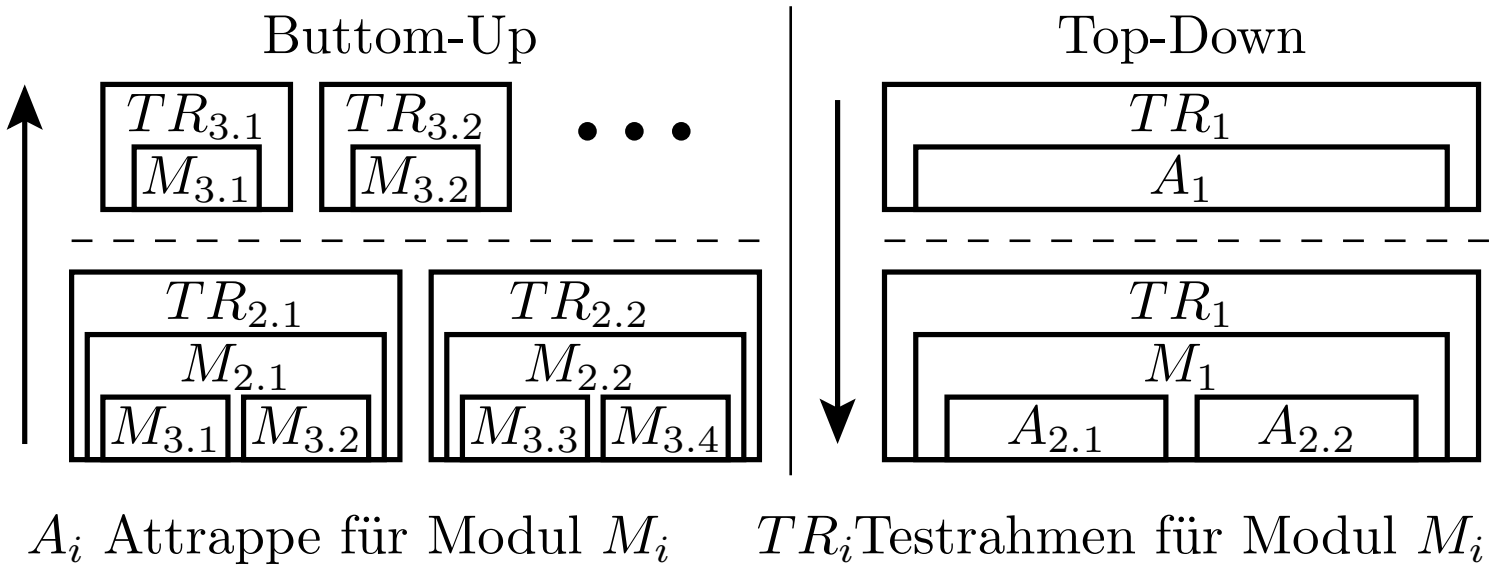
Modularisierung und Testrahmen

Jeder Code-Baustein muss ausprobiert werden:



Strategien für die Entwurfs- und Testreihenfolge:

- Bottom-Up: Beginn mit dem Entwurf und Test der untersten Module. Test der übergeordneten Module mit den bereits getesteten Untermodulen.
- Top-Down: Beginn mit dem Entwurf übergeordneter Module und Test mit Attrappen für die Untermodule. Schrittweise Ersatz der Attrappen durch getestete Untermodule.

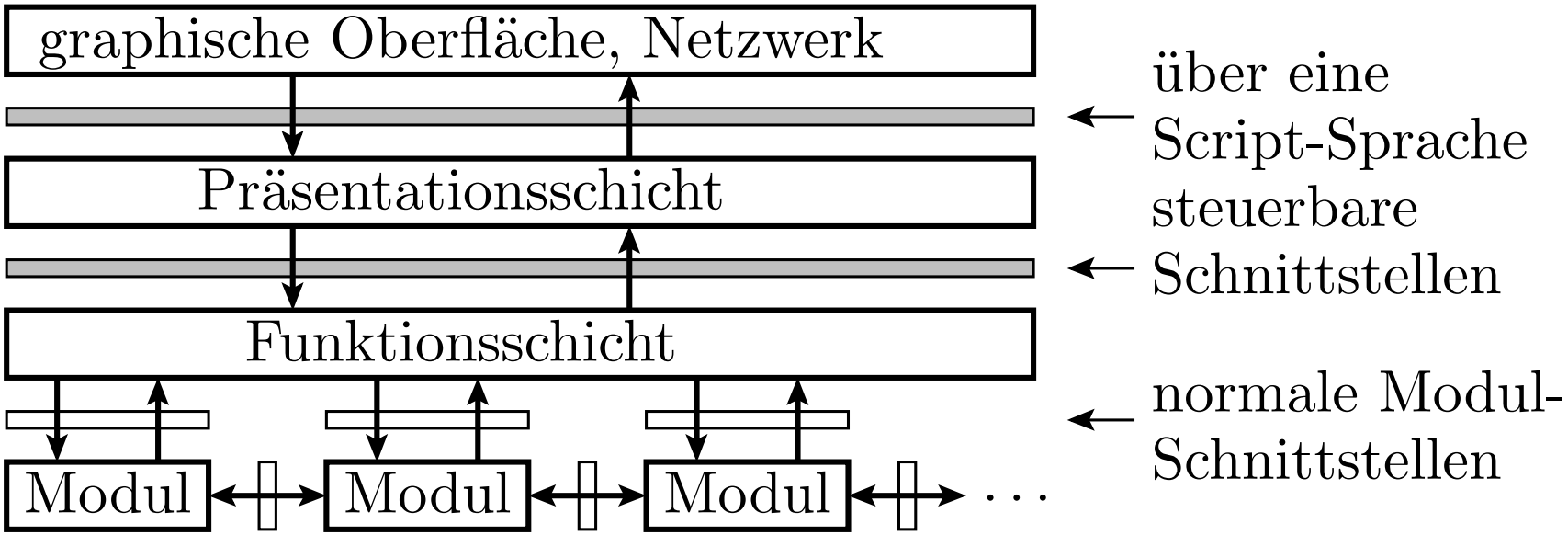


Praktisches Vorgehen bei der Erstellung der Testbeispiele:

- Beginn mit typischen Eingaben und manueller Ergebniskontrolle, um das Testobjekt zu untersuchen.
- Erweiterung der Tests um automatische Ergebniskontrolle.
- Erweiterungen um Tests zur zielgerichteten beispielhaften Kontrolle aller zuzusichernden Eigenschaften.
- Erweiterung um Tests für die Fehlfunktionsbehandlung.
- Fussifizierung (lange Zufallstests) für Fehler mit geringer MF-Rate.

Je mehr Attrappen der Test erfordert, um so schlechter ist der Code.

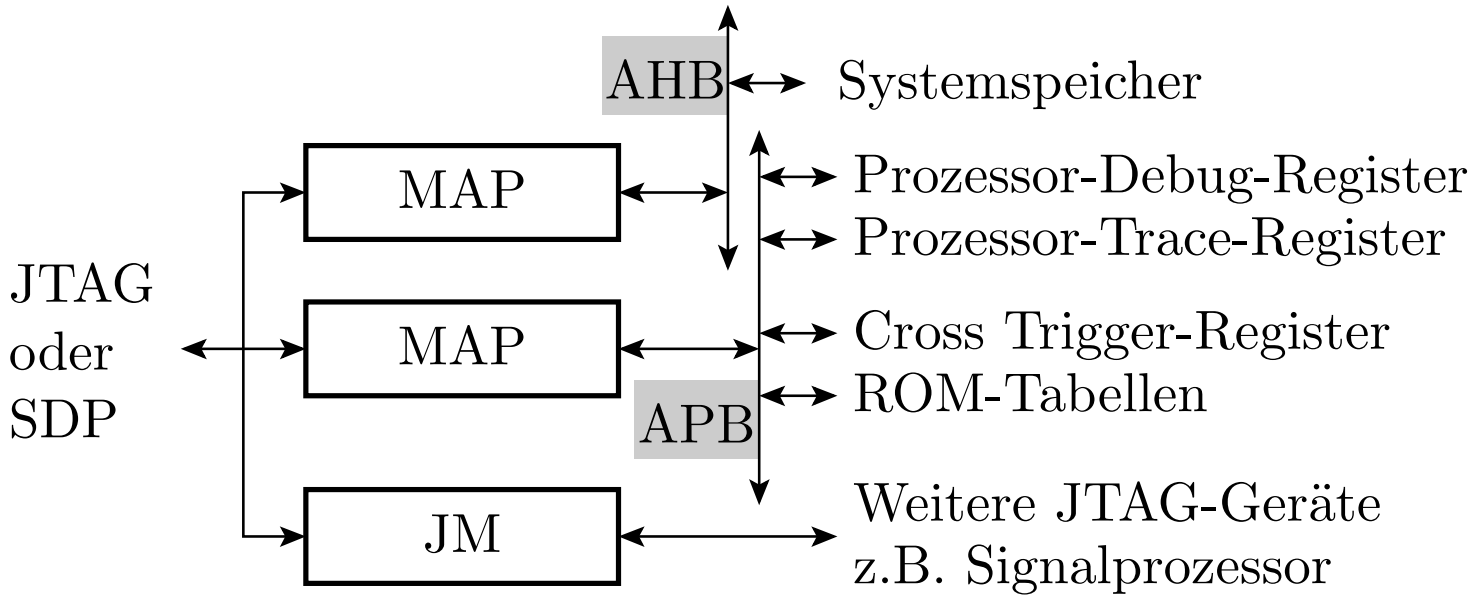
Schichtenmodell



- Alle Funktionsbausteine sind einer Schicht zugeordnet.
- Einer höhere Schicht - z.B. Benutzeranwendung wie Excel oder Word - kann nur Prozeduren der Schicht darunter über eine wohl definierte Schnittstelle (API) nutzen.
- Ein Kommunikationsprogramm kann z.B. nicht direkt, sondern nur über eine Betriessystemfunktion auf den COM-Port zugreifen.
- Die Schichten bieten oft Script-Sprachen, um Tools (Funktionen der Schicht) zu steuern und damit auch Testskripte zu schreiben.

ARM-Testbusarchitektur

Einige Trace- und Debug-Funktionen ... verlangen HW-Unterstützung.



Über AHB Debug-Lese- und Schreibzugriff auf den kompletten Speicher, über APB auch Zugriff auf Debug-Register, ROM-Tabellen, ...

ARM	Verbreitetste Mikroprozessor-Architektur für Embedded-Systems (Smartphones, ..).
SDP	Serieller Debug-Port.
JM	JTAG-Master für weiterer Rechnerbausteine auf dem Chip, z.B. Co-Prozessoren.
MAP	Speicherzugriffsport mit serieller Adress- und Datenübergabe.
AHB, APB	Advanced High Performance Bus, Advanced Peripherial Bus.

JTAG-Datenregister für den Baugruppentest:

- Boundary-Scan, Bypass und 40-Bit ID-Code

JTAG-Datenregister für den SW-Test:

- 32-Bit Debug-Control und Status-Register (DSCR).
- Instruction-Transfer-Register (ITR): 32 Befehlsbit + ein Statusbit, zur Ausführung von Prozessorbefehlen in einem speziellen Debug-Modus.
- Debug Communications Channel (DCC): 32-Bit-Datenwort + 2 Statusbits für den bidirektionalen Datentransfer mit Prozessorkern.
- Embedded Trace Module (ETM): 7 Adressbits + 32-Bit-Datenwort + 1 R/W-Bit zur Steuerung von Trace-Operationen*.
- Debug-Modul: 7 Adressbits + 32-Bit-Datenwort + 1 R/W Bit. Zugriff auf die Register für Hardware Breakpoints, Watchpoints etc..

ARM Befehle für Zusammenarbeit Programm Debugger: HALT (Übergang in Debug-Modus, in dem über das ITR Befehle eingefügt werden können) und RESTART zum Verlassen des Debug-Modus.

* Trace-Aufzeichnung entweder in einen eingebetten Trace-Buffer (ETB) auf dem Chip oder Ausgabe über einen High-Speed-Port.

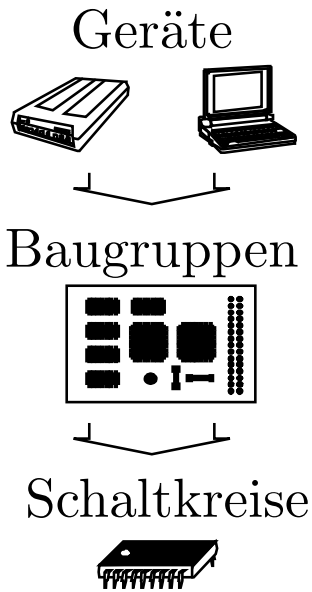
Leiterplatten

Hierarchie und Test

Rechnerhardware besteht aus tauschbaren Komponenten:

- tauschbare Teilsysteme.
- tauschbare Leiterplatten,
- austauschbare Bauteile.

Die Komponenten werden vor Einbau in das übergeordnete System gründlich getestet.

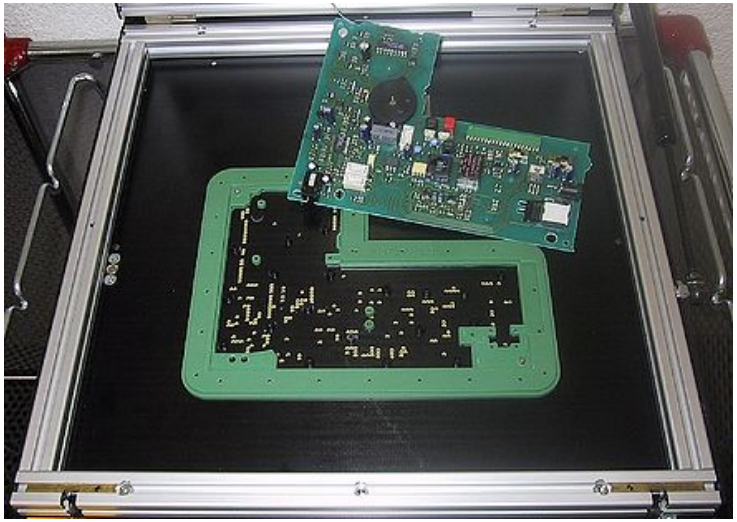


Wiederholungsfragen

1. Warum tauschbare Komponenten bzw. unter welcher Bedingung nicht erforderlich (nicht zweckmäßig)?
2. Warum ist ein gründlicher Komponententest zu fordern?
3. Warum beginnt ein BG Test mit einem statischen Bestückungs- und Verbindungstest?

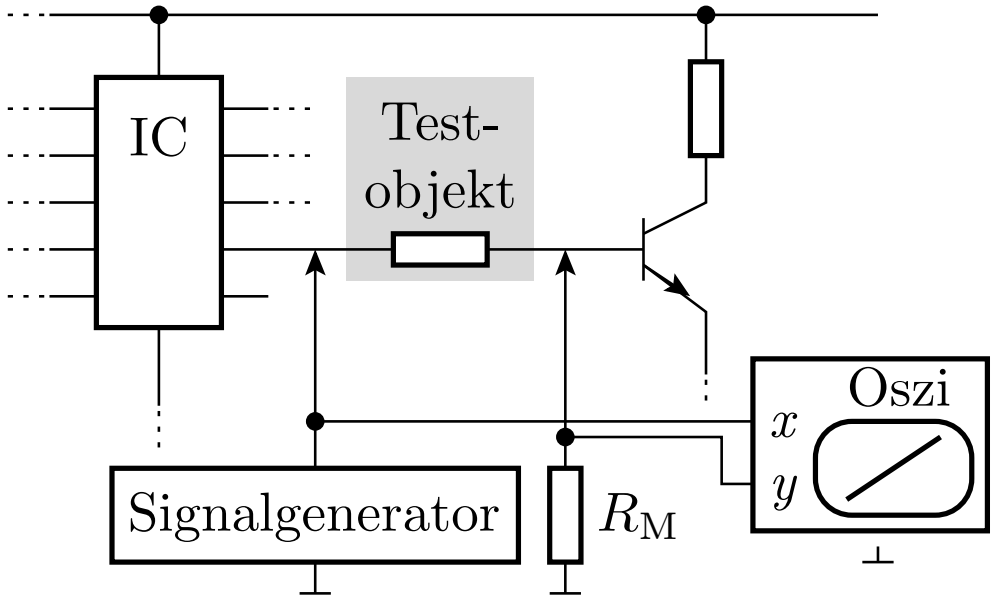
Bestückungs- und Verbindungstests

Hauptfehler auf Baugruppen sind Kurzschlüsse und Unterbrechungen. Nachweis durch Widerstandsmessungen zwischen und entlang der Verbindungen.
In der Serienfertigung erfolgt die Kontaktierung mit einem mit Unterdruck angesaugten Nadeladapter.



Die Nadeln sind mit einer Relais-Matrix verbunden, über die vom Testprogramm Prüfgeräten angeschlossen werden. Auch Bestückungsfehler lassen sich überwiegend mit Zweipunktmessungen von Strom-Spannungsbeziehungen erkennen. Fehlerlokalisierung im Vergleich zur Rückverfolgung falsche Ausgaben von dynamischen Tests sehr einfach (siehe Folie 1.120 *Rückverfolgung von Datenverfälschungen*).

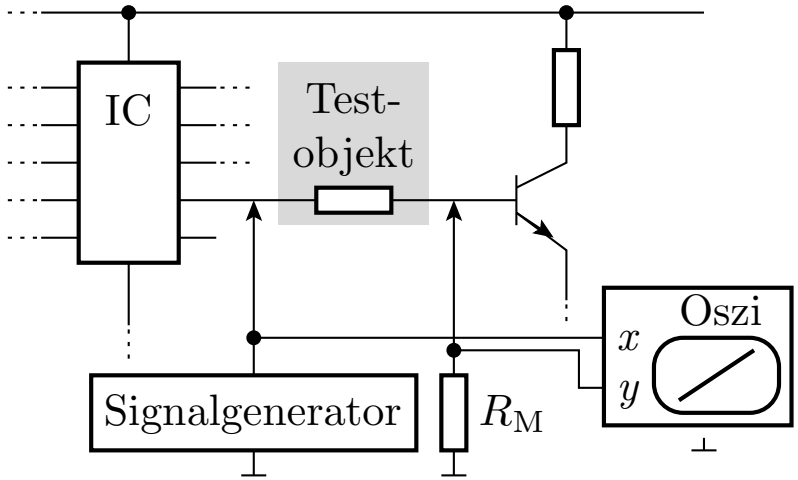
Bestückungstests



Kontrolle auf Bestückungsfehler durch Überprüfung ausgewählter Zweipunktmerkmale:

- Widerstandswerte,
- Kapazitäten,
- Flussspannungen von Dioden, ...

Bestückungstests für Schaltkreise

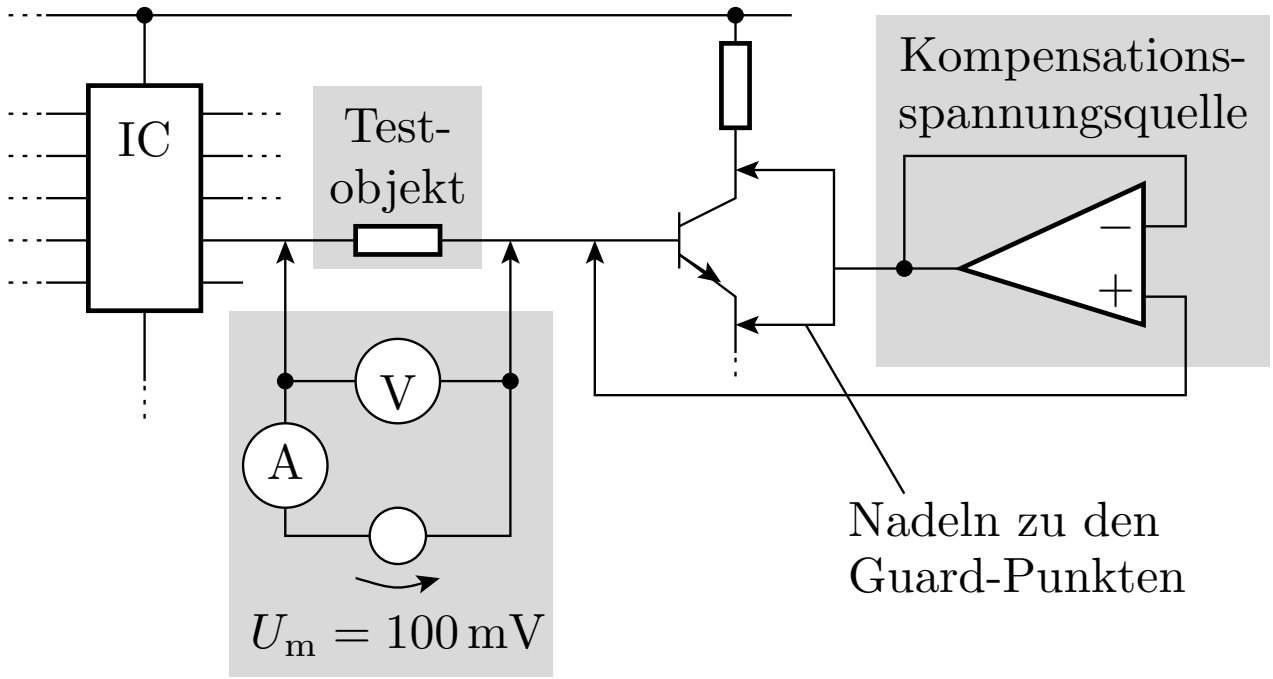


Verbindungskontrolle IC-Anschlüsse zu den Kontakt-Pads durch Ausmessen der Schutzdioden zur Versorgungsspannung oder Masse.

Die Strom-Spannungs-Beziehung zwischen zwei Punkten hängt nicht nur vom Bauteil zwischen den Nadeln, sondern von allen Strompfaden, im Beispiel durch Transistor und Schaltkreis ab. Problematisch:

- die Toleranzbereiche der Sollwerte mit allen Bauteilstreuungen,
- die Erkennungssicherheit für Fehlbestückungen, z.B. bei sehr kleinen Kapazitäten.

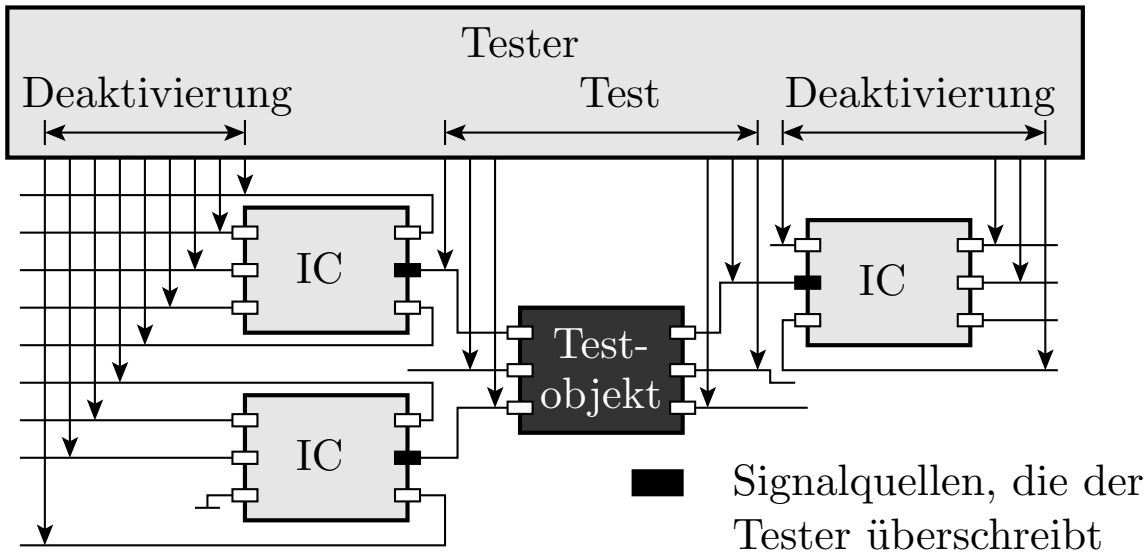
Analoger In-Circuit Test



Unterdrückung von Parallelströmen zum Testobjekt durch Kompensation der Spannungsabfälle über den wegführenden Bauteilen auf einer Testobjektseite auf null über »Guard-Punkte«:

- Vereinfacht die Testprogrammerstellung, insbesonderen die Sollwert- und Sollwerttoleranzfestlegung.
- Mindert die Häufigkeit von Fehlklassifikationen.

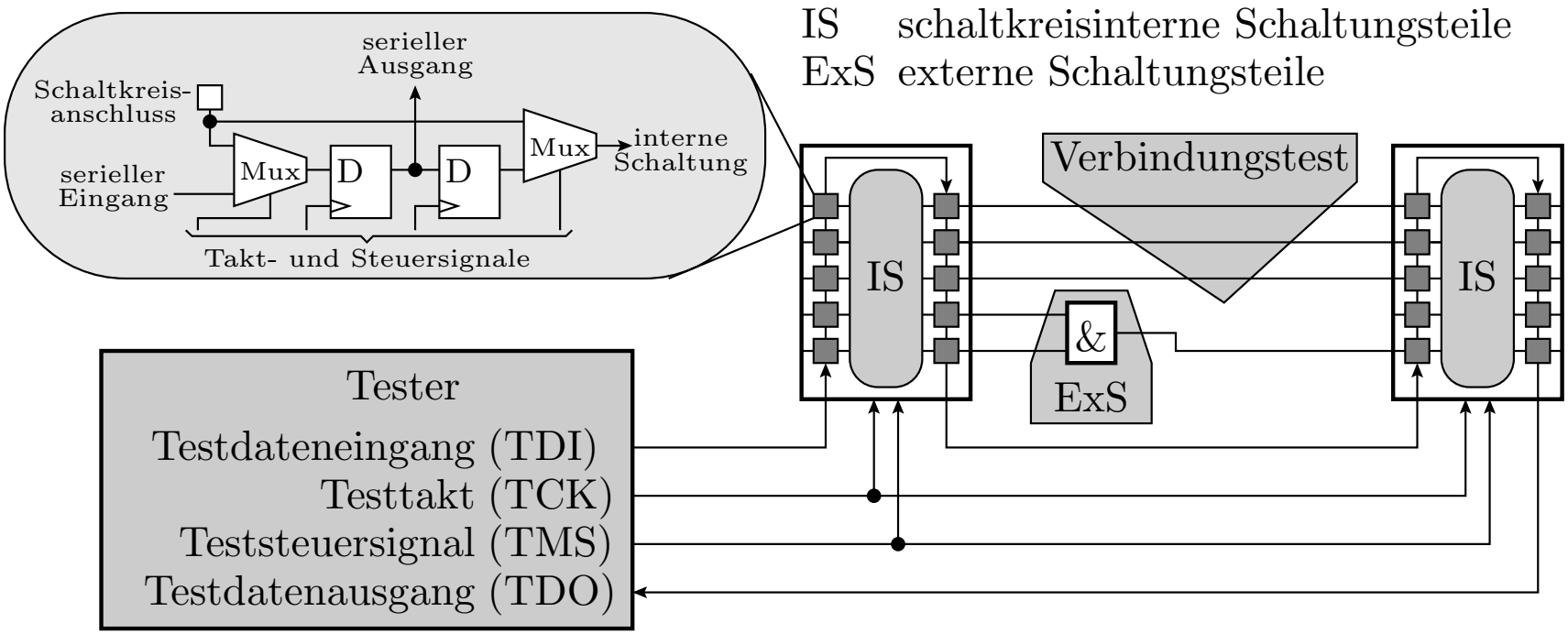
Digitaler In-Circuit-Test



- Kontaktierung der Baugruppe über ein Nadelbetadapter.
- Isolierter Test der Schaltkreise durch Überschreiben der digitalen Schaltkreiseeingaben mit stromstarken Treibern.
- Im Gegensatz zum analogen ICT unter Spannung.
- Andere Schaltkreise werden möglichst deaktiviert (Anschlüsse hochohmig).

Erlaubt auch den isolierten Test der Bausteine, aber im Gegesatz zum analogen In-Circuit-Test unter Spannung.

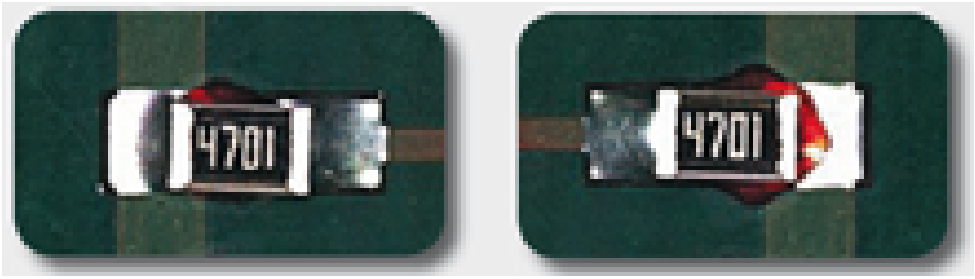
Boundary-Scan



Die ursprüngliche Ideee von Boundary-Scan war der Ersatz der mechanischen Nadeln durch »silicon nails« als Alternative zu den teuren, für jede Baugruppe speziell anzufertigenden Nadeladaptern. Verbindungs- und Bestückungstest über die seriell les- und beschreibbaren Scan-Register an den Schaltkreisanschlüssen (siehe Folie 4.48 *Boundary-Scan*).

Optische Inspektion

Es gibt Bestückungsfehler, die sind optisch, aber nicht elektrisch erkennbar. Bild links korrekt bestückter SMD-Widerstand, rechts Lötfläche durch Kleber verschmutzt. Elektrisch leitende aber keine feste Lötverbindung:



Nachweis nur durch visuelle Kontrolle möglich. Nach Ausfall der Baugruppe z.B. durch Vibration in einem Fahrzeug ist sofort erkennbar, dass es sich um einen optisch nachweisbaren Fertigungsfehler handelt.

Wenn ein solcher Fertigungsfehler Schaden verursacht, z.B. einen Unfall, greift die Produkthaftung, d.h. der Hersteller muss für den entstandenen Schaden aufkommen.

Kontrollfragen

- Zählen Sie Maßnahmen des prüfgerechten Entwurfs für den Baugruppentest auf.
- Was bedeutet isolierter Test von Komponenten und wie setzen der analoge und der digitale In-Circuit-Test dieses Prinzip um?
- Warum ist für SMD-bestückte Baugruppen von Steuergeräten für KFZ eine optische Inspektion zwingend?
- Welche prüftechnischen Probleme des Baugruppentests löst Boundary-Scan?

Zusammenfassung

Inspektion

Sichtprüfungen, anwendbar auf Entwurfsbeschreibungen, Programmcode, Testausgaben, ... Kenngrößen:

- Testgüte: Fehlerüberdeckung, Phantom-MF-Rate.
- Technologiegüte: Effizienz (gefundene Abweichungen pro Mitarbeiterstunde) und Effektivität (gefundene Abweichungen je NLOC).

Inspektionen sind

- in frühen Entwurfsphasen oft einzige Kontrollmöglichkeit,
- aufwändige manuelle Tätigkeiten,
- ein gutes Mittel für den Knowhow-Austausch zwischen Entwerfern.

Ein Experiment und die allgemeine Erfahrung haben gezeigt, dass die Effizienz einer manuellen Inspektion nicht wie bei automatischen Kontrollen monoton abnimmt, sonderen durch ineffiziente Einarbeitungs- und Ermüdungsphasen gekennzeichnet ist.

Schätzen der Kenngrößen

- Capture-Recapture:

$$\hat{\mu}_{\mathrm{F}} = \left.\frac{\#F_1 \cdot \#F_2}{\#(F_1 \cap F_2)}\right|_{\mathrm{ACR}} \tag{4.1}$$

$$\hat{\mu}_{\mathrm{FC}} = \left.\frac{\#(F_1 \cap F_2) \cdot \#(F_1 \cup F_2)}{\#F_1 \cdot \#F_2}\right|_{\mathrm{ACR}} \tag{4.2}$$

  Abschätzung als Produkt bzw. Quotient von drei Zählwerten. Für dieselbe Genauigkeit ≈ 3-mal so große Zählwerte wie für Abschätzungen aus nur einem Zählwert erforderlich. Erhebliche systematische Fehler bei stark abweichenden Fehlernachweiswahrscheinlichkeiten und Informationsaustausch zwischen den Inspekteuren.
- Modellfehlerbasiert:

$$\hat{\mu}_{\mathrm{FC}} = \left.\frac{\#DF_{\mathrm{M}}}{\#F_{\mathrm{M}}}\right|_{\mathrm{ACR}} \tag{4.3}$$

$$\hat{\mu}_{\mathrm{F}} = \left.\frac{\#DF}{\hat{\mu}_{\mathrm{FC}}}\right|_{\mathrm{ACR}} \tag{4.4}$$

  Ähnlicher Aufwand. Geringere zu erwartende zufällige und systematische Schätzfehler.

Inspektionstechniken

Eine gute Inspektionstechnik vermeidet ineffiziente Einarbeitungs- und Ermüdungsphasendiese durch geschickte Arbeitsorganisation.

Es gibt unterschiedliche Inspektionstechniken, insbesondere auch mit unterschiedlichen Kompromissen zwischen Kreativitätsbeschränkung, Effizienz und Ergebnisvorhersagbarkeit:

- max. kreative Freiräume: Review in Kommentartechnik
- max. Ergebnisvorhersagbarkeit: formales Review in Sitzungstechnik.

Funktionstest

- IT-Systeme werden überwiegend digital realisiert, vor allem auch wegen Toleranz gegenüber Störungen und andere Probleme.
- Analoge Verarbeitung ist im wesentlichen auf die Ein- und Ausgabe begrenzt und wird beim Systemtest und isoliert vom übrigen System überprüft.
- Also Test mit analogen Ein- und Ausgaben nur ganzheitlich und EA-Schnittstellen.
- Wenn Ausprobieren in der Systemumgebung problematisch (gefährlich, schwierig, ...) auch Nachbildung der Systemumgebung durch Attrappen (HIL, Simulation).
- Prüfgerechter Entwurf: Sammlung von Voraussetzungen, gute Tests durchführen zu können. Kontaktiermöglichkeit für Messtechnik, Testmodi, ...
- Für den Test der analogen Verabeitungsketten bieten sich Loop-Tests mit Ein- und Ausgabe an der digitalen Peripherie an. Später in der Einsatzphase auch als Einschalt- und Wartungstests nutzbar.

Test digitaler Schaltungen

- Bereitstellung logischer Eingabewerte und Kontroll der Ausgabebits, vorzugsweise durch Vergleich mit Sollwerten.
- Kontrolle von Signallaufzeiten durch mehrfache Abtastung nach jeder Eingabeänderung.
- Zur Erhöhung der Fehlerüberdeckung und Vereinfachung der Testauswahl Einbau von Lese- und Schreibfunktionen für interne Speicher und Signale z.B. mit Scan-Registern.
- Datenaustausch mit Tester vorzugsweise seriell über Testbus.
- Boundary-Scan: Scan-Ring um den Schaltkreis als Ersatz für die Stecker oder Prüfspitzen zur mechanischen Kontaktierung.

JTAG-Testbus:

- Serieller Testbus für den Testerzugriff auf die Boundary-Scan-Ketten und andere Testhilfen .
- Außer für den isolierten Test der Leiterplattenverdrahtung, auch Steuerung interner Test- und Debug- und Programmierfunktionen.

Software-Test

Einbettung der Testobjekte in Programmrahmen, die Eingaben bereitstellen, Ausgaben kontrollieren und den Ablauf steuern.

Nützlich Hilfmittel für gründliche Kontrollen und die Fehlersuche:

- Debugger mit Schrittbetrieb, Haltpunkten, Lese- und Schreibzugriff auf interne Daten.
- Trace-Aufzeichung von Variablen und Ausgabesignalen, ...

Modularisierung, Testrahmen und Attrappen:

- Botton-Up: Entwurf und Test beginnend von den kleinsten zu immer größeren Bausteinen mit je einem zu entwerfenden Testrahmen je Baustein.
- Top-Down: Entwurf und Test beginnend mit dem Gesamttestrahmen, Testrahmen mit oberster Verarbeitungsschicht, ... Ersatz der fehlenden Bausteine durch Attrappen.

SW-Test: Schichten und ARM-Testbus

Schichten bieten vereinheitlichte Testschnittstellen für der Schicht zugeordnete Funktionen, teilweise sogar Script-Sprachen zur Programmierung von Tests.

Die beispielhaft besprochene ARM-Testbus-Architektur bietet sämtliche Test-, Programmier- und Debug-Schnittstellen für die Inbetriebnahme und die hardware-nahe Fehlersuche:

- Speicher- und Registerzugriff,
- Debugger-Funktionen,
- Trace-Aufzeichung, ...

Leiterplattentest

- Leiterplatten bestehen aus gründlich vorher getesteten Bauteilen. Test hauptsächlich auf Bestücks- und Verbindungsfehler.
- Bei größeren Stückzahlen Kontaktierung mit Nadelbettadapter. Bestückungs- und Verbindungstests spannungsfrei über elektrische Zweipunkmessungen.
- Der elektrische In-Circuit-Test verwendet zusätzlich Guard-Punkte zur Unterdrückung von Parallelströmen zum zu testenden Bauteil.
- Digitaler In-Circuit-Test ist im Gegensatz dazu ein Test unter Betriebsspannung, bei dem digitale Signalwerte auf den Verbindungen mit stromstarken Treibern auch kurzzeitig überschrieben werden.
- Wegen zunehmender mechanischer und elektrischer Probleme mit wachsender Schaltungsgröße und Packungsdichte Trend zum Ersatz der mechanischen Nadeln durch Boundary-Scan, d.h. in die HW eingebauten Lese- und Schreibfunktionen für die logischen Werte an den Schaltkreisanschlüssen.

- Bestückungskontrolle durch Lesen der Hersteller- und Bauteilidentifikationsnummern.
- Optische Inspektion für typische elektrisch nicht nachweisbare aber die Lebenserwartung beeinträchtigende Verbindungsfehler.

Überwachung

Format- und Wertekontrollen

Im Rechner dargestellte Daten repräsentieren Werte in einem Format (Bitanzahl, Codierung, Prüfmerkmale, ...). Meist werden nur Formateigenschaften überwacht. Bisher behandelt:

- Informationsredundanz bei gleichmäßiger Abbildung von Verfälschungen auf zulässige und unzulässige Werte kombiniert mit Kontrollverfahren, die alle unzulässigen Werte erkennen.

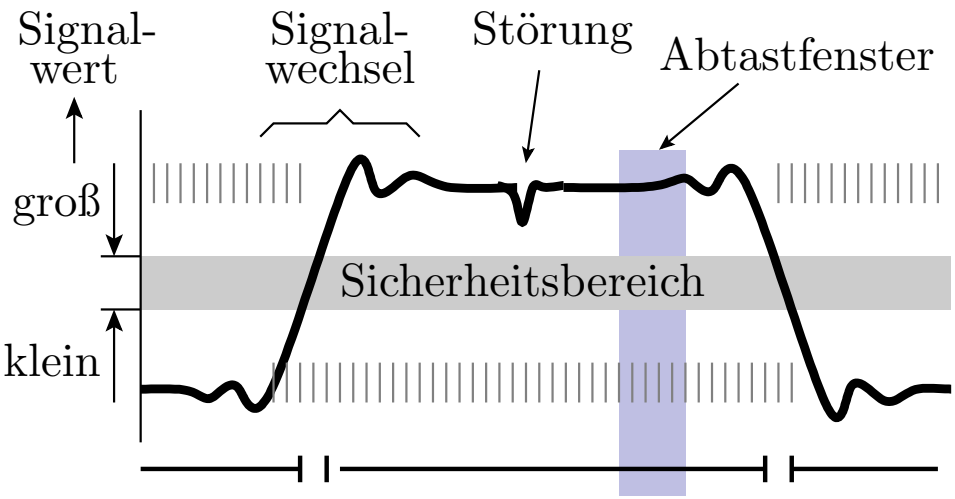
Behandelten Verfahren für Werteüberwachung

- Soll-Ist-Vergleiche zur Ergebniskontrolle dynamischer Tests,
- Mehrfachberechnung und Vergleich,
- Loop-Test und
- aufgabenspezifische Korrektheitstests.

Bei jedem dieser Verfahren werden Istwerte mit Sollwerten verglichen (Formatmerkmale, Prüfkennzeichen, Datenwerten, ...) .

Vergleich

Bitweiser Vergleich mit Sollwerten



Signalwert	
erweitert	binär
$x = 1$	$x = 1$
$x = \mathrm{X}$	nicht darstellbar
$x = 0$	$x = 0$

Gültigkeitsfenster

Die Werteeinteilung in »kleine«, »unzulässig« und »groß« macht die Bitverarbeitung incl. Vergleich extrem robust gegen Störungen, aber nicht gegen Fehler:

- falsche Sollwerte oder
- Fehler in der Vergleichfunktion.

Eine Vergleichfunktion muss wie jede andere Funktion getestet und oder sogar überwacht werden.

Wertebereichstest

Im Gegensatz zu Bitwerten unterliegt die dargestellte Information oft Störungen und anderen zufälligen Einflüssen:

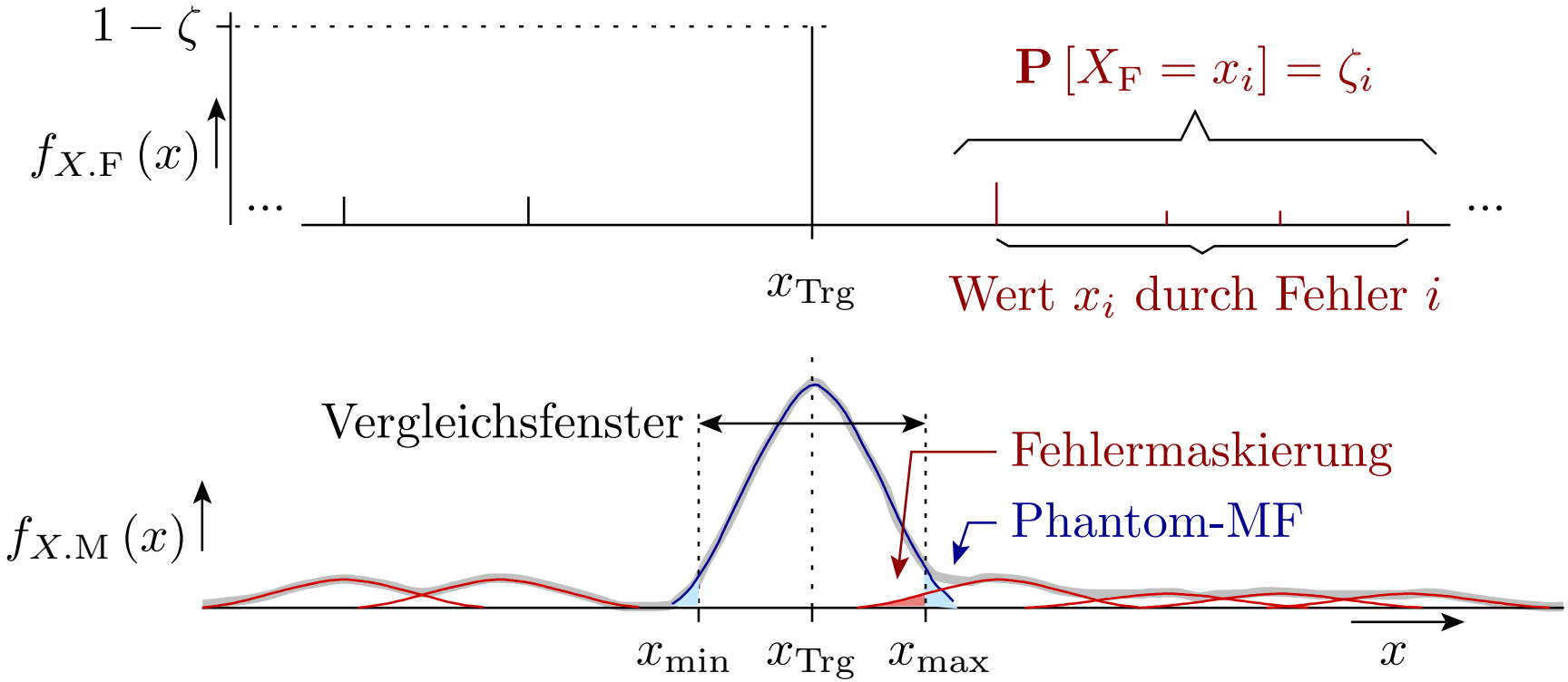
- Rauschen und andere Störungen bei der Informationserfassung,
- Quantisierungsfehler bei der Digitalisierung und
- Rundungsfehler bei der Berechnung.

Ein auszuwertendes ungestörtes Signal hat eine Mischverteilung aus

- der Verteilung fehlerfreier Werte, im Idealfall ein exakter Wert und
- den Werteverteilungen möglicher Verfälschungen:

$$f_{X.\mathrm{F}}(x) = \sum_{i=1}^{\#F} \zeta_i \cdot f_{X.i}(x) \cdot \left(1 - \sum_{i=1}^{\#F} \zeta_i\right) \cdot f_X(x)$$

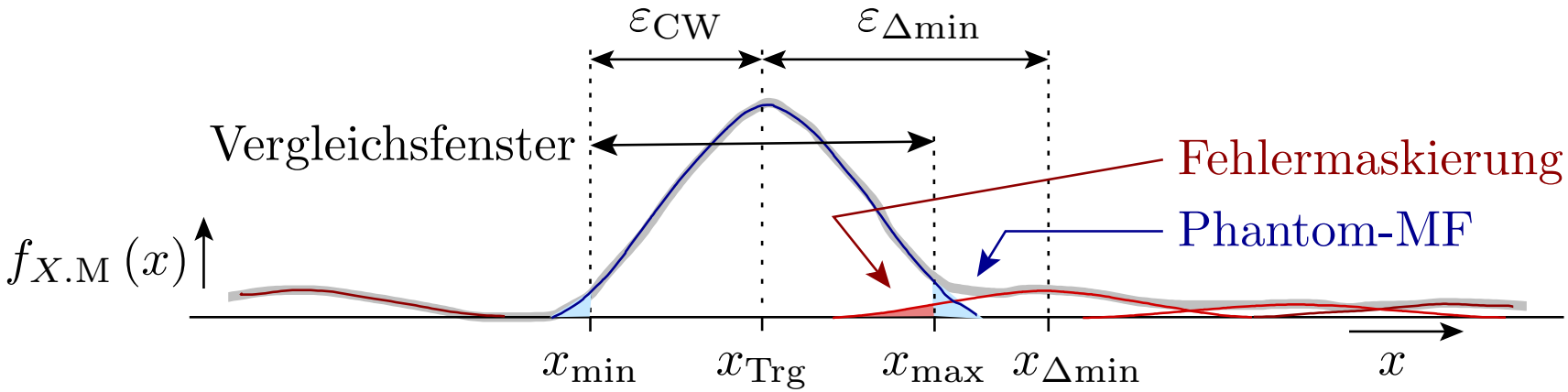
X	Zufallsvariable für den unverfälschten Wert, im Beispiel $\mathbb{P}[X = x_{\mathrm{Trg}}] = 1$, sonst 0.
x_{Trg}	Sollwert.
X_i	Zufallsvariable für durch Fehler i verfälschte Werte, im Beispiel $\mathbb{P}[X = x_i] = 1$.
X_{F}	Zufallsvariable der durch Fehler verfälschte Werte insgesamt, im Beispiel diskret verteilt.
ζ_i	MF-Rate verursacht durch Fehler i.
$\#F$	Anzahl der möglichen Fehler (number of possible faults).



Der zu kontrollierende Wert X_M ist die Summe der Zufallsvariable X_F »nur Fehler« und X_D »zufällige Einflüsse«:

$$X_\mathrm{DF} = X_\mathrm{F} + X_\mathrm{D}$$

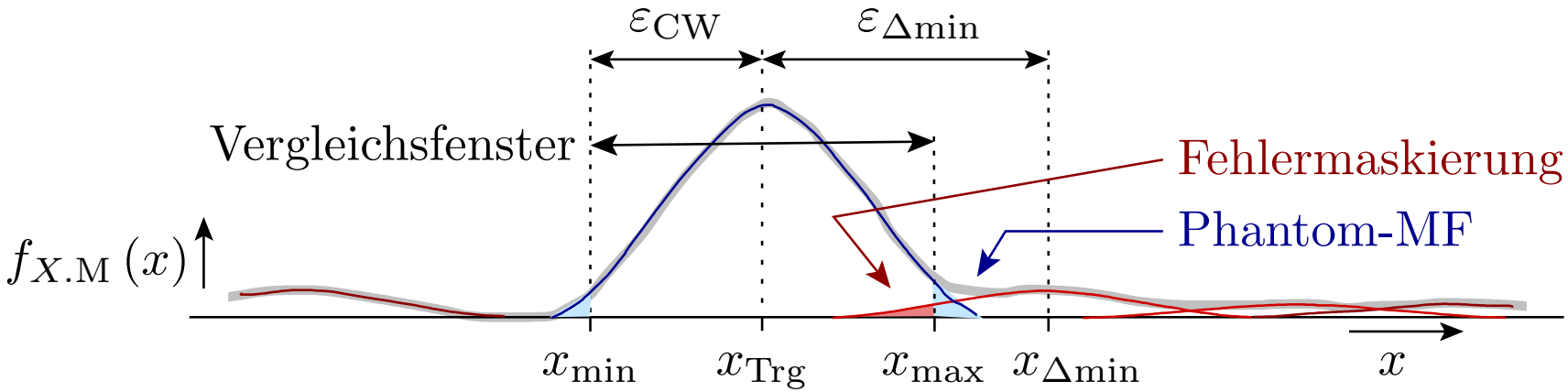
x_Trg	Sollwert.
X_F	Zufallsvariable der durch Fehler verfälschte Werte insgesamt, im Beispiel diskret verteilt.
X_D	Zufallsvariable der zufälligen Einflüsse (Rauschen, Quantisierung, Rundung, ...).
X_DF	Zufallsvariable der durch Fehler und Störungen verfälschten zu kontrollierenden Werte.



Im Bild sind die zufälligen Einflüsse normalverteilt. Erwartungswert $\mu_{\mathrm{D}} = 0$, Standardabweichung σ_{D}. Die Faltung beider Verteilungen ist multimodal. Hauptgipfel beim Sollwert x_{Trg}, Nebengipfeln an den einzelnen fehlerverursachten Werten x_i. Phantom-MF-Rate bei symmetrischem Vergleichsfenster:

$$\zeta_{\mathrm{Phan}} = 2 \cdot \left(1 - \Phi\left(\frac{\varepsilon_{\mathrm{CW}}}{\sigma_{\mathrm{D}}}\right)\right) \tag{1.7}$$

ζ_{Phan}	Phantom-Fehlfunktionsrate.
$\Phi(z)$	Verteilungsfunktion der standardisierten Normalverteilung.
$\varepsilon_{\mathrm{CW}}$	Intervallradius des symmetrischen Vergleichsfensters.
σ_{D}	Standardabweichung der normalverteilten zufälligen Einflüsse.



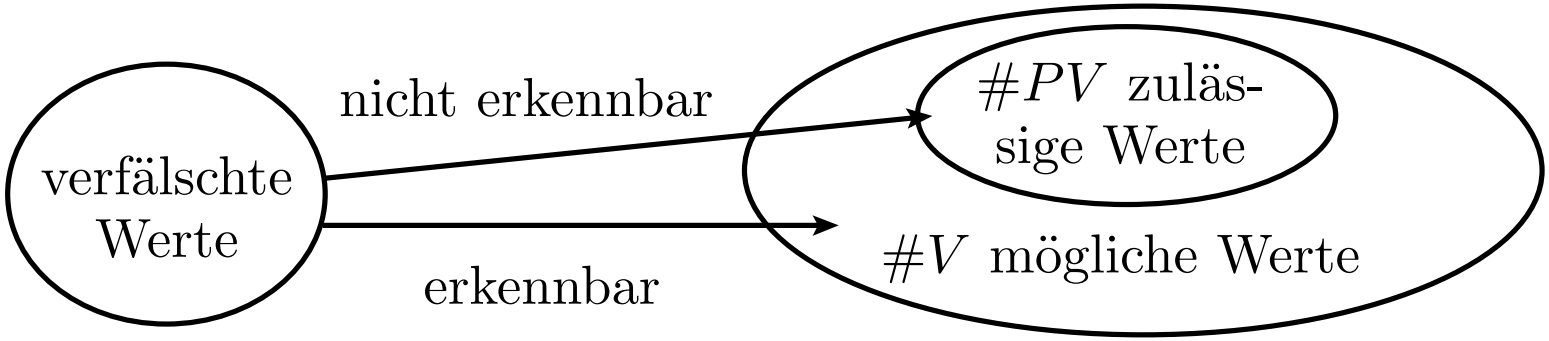
Die Maskierungswahrscheinlichkeit bestimmt der Fehler mit der keinsten nachzuweisenden Werteverfälschung $\varepsilon_{\Delta\mathrm{min}}$ und dessen MF-Raten $\zeta_{\Delta\,\mathrm{min}}$ ab:

$$p_{\mathrm{M}} = 1 - \zeta_{\Delta\,\mathrm{min}} \cdot \Phi\left(\frac{\varepsilon_{\Delta\mathrm{min}} - \varepsilon_{\mathrm{CW}}}{\sigma_{\mathrm{D}}}\right) \tag{1.8}$$

Δx_{min}	Fehlerverursachter Wert mit dem geringsten Abstand zum Sollwert.
p_{M}	Maskierungswahrscheinlichkeit.
$\zeta_{\Delta\,\mathrm{min}}$	MF-Rate der Fehler mit geringsten Wertabstand zum Sollwert.
$\varepsilon_{\Delta\mathrm{min}}$	Abstand der geringsten nachzuweisenen VerfÃ¤lschung vom Sollwert.
$\varepsilon_{\mathrm{CW}}$	Intervallradius des symmetrischen Vergleichsfensters.
σ_{D}	Standardabweichung der normalverteilten zufälligen Einflüsse.

Informationsredundanz

Wiederholung Informationsredundanz



Wenn Verfälschungen weder unzulässige noch zulässige Werte bevorzugen und alle unzulässigen Werte erkannt werden, zu erwartende MF-Überdeckung (vergl. Gl. 1.22 und 1.24):

$$\mu_{\mathrm{MC}} = 1 - \frac{\#PV}{\#V} \tag{1.9}$$

$$\mu_{\mathrm{MC}} = 1 - 2^{-r} \tag{1.10}$$

Phantom-MF-Rate null (siehe Gl. 1.23).

μ_{MC}	Zu erwartende Fehlfunktionsüberdeckung.
$\#PV$	Anzahl der zulässigen Werte (Number of permitted values).
$\#V$	Anzahl der möglichen Werte (Number of possible values).
r	Anzahl der redundanten Bits.

Klassischer Rechtschreibtest ist viel schlechter

Wort im Wörterbuch enthalten?

- Erkennung, wenn nicht falsches Wort nicht im Wörterbuch wie »Maus« statt »Haus«. Viel schlechter als nach (Gl. 4.9):

$$\mu_{\mathrm{MC}} = 60\% \ldots 90\% \ll 1 - \frac{\#PV}{\#V}$$

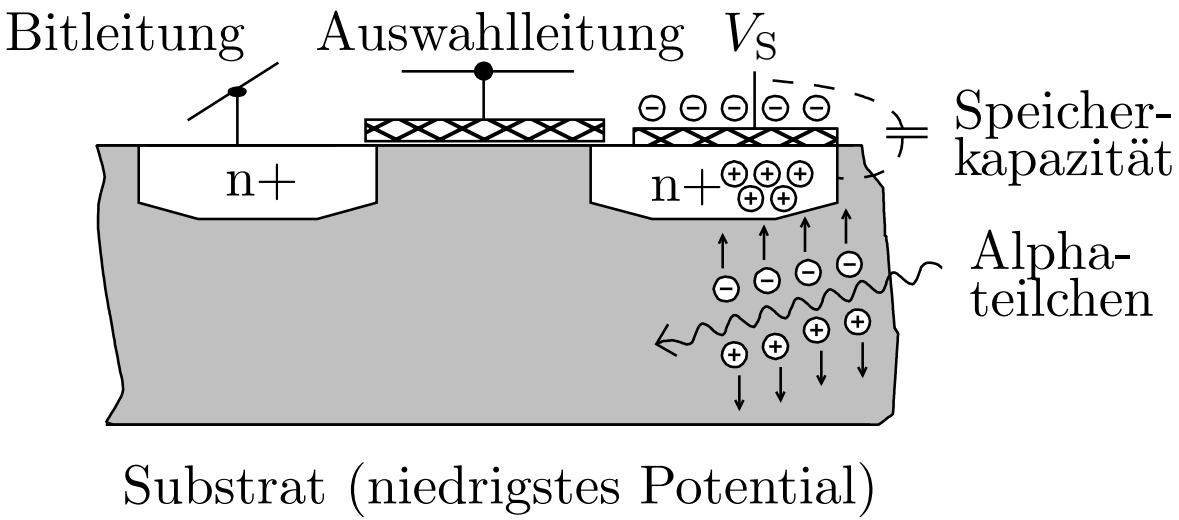
- Phantom-MF, zulässiges Wort nicht im Wörterbuch, typisch:

$$\zeta_{\mathrm{Phan}} = \frac{1 \ldots 10\ [\mathrm{PM}]}{1000\ [\mathrm{DS}]} \gg 0$$

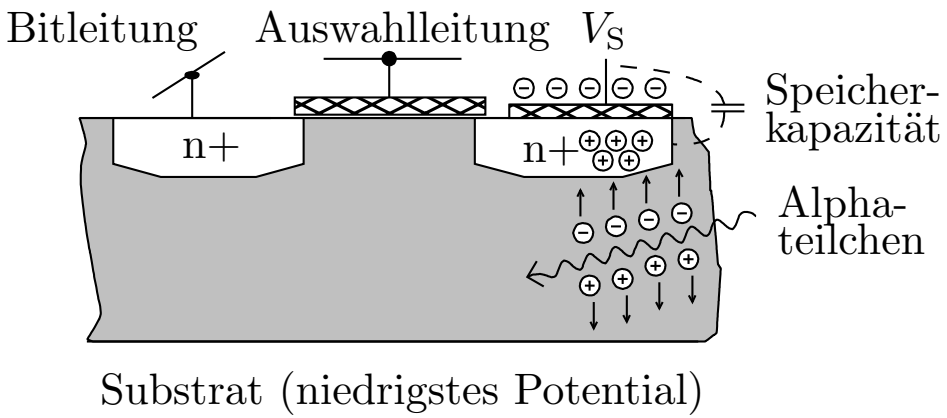
Anteil der zulässigen Worte an den darstellbaren Zeichenfolgen fast null, aber bei Schreibfehlern entstehen überdurchschnittlich oft zulässige Worte und nicht alle zulässigen Zusammensetzungen und Abänderungen von Worten stehen im Wörterbuch.

μ_{MC}	Zu erwartende Fehlfunktionsüberdeckung.
ζ_{Phan}	Phantom-Fehlfunktionsrate.
$[\mathrm{PM}]$	Zählwert in Phantom-Fehlfunktionen.
$[\mathrm{DS}]$	Zählwert in erbrachten Service-Leistungen.

Paritätstest für DRAMs und Speicherriegel



- Informationsspeicherung in winzigen Kapazitäten.
- Häufigste Ursache für Datenverfälschungen: Alphastrahlung.
- Deren Quellen radioaktiver Zerfall von Uran und Thorium, enthalten als Spurenelemente im Gehäuse und im Aluminium der Leiterbahnen oder Kernprozesse im Silizium durch Höhenstrahlung.
- Mittlerer Ereignisabstand: viele Stunden oder Tage.
- Paritätstest erkennt alle ungeradzahligen und im Mitte 50% der Verfälschungen.



- Energie eines Alphateilchen: $5\,\mathrm{MeV}$. Energieverlust bei der Generierung eines Elektronen-Loch-Paares $\approx 3{,}6\,\mathrm{eV} \Rightarrow$ Generierung von $\approx 10^6$ Ladungsträgerpaaren. Reichweite $\approx 89\,\mu\mathrm{m}$. gespeicherte Ladung $\approx 10^5$ Ladungsträger. Datenverlust einer oder mehrerer benachbarter Zellen möglich.
- Gleichzeitige Verfälschung durch zwei Alphateilchen unwahrscheinlich.
- Geometrische Trennung der Zellen eines Datenworts (getrennte Schaltkreise oder Speichermatrizen) $\Rightarrow$ Je Zerfall Einzelbitverfälschung je gelesenes Datenwort.

Auch bei der Übertragung sind Verfälschungen selten. Datenobjekte so verschränken, das auch Fehler-Bursts auf Einzelbitfehler je Datenobjekt abgebildet werden.

Einzelbitfehler bei Übertragung und Speicherung

Wenn die Daten so auf Datenobjekte aufgeteilt werden, dass jedes Verfälschungsereignis max. ein Bit je Datenobjekt verfälscht, genügt ein fehlererkennender Code für Einzelbitfehler, also ein Paritätstest.

Für seltene unabhängige Bitverfälschungen ist die Anzahl der verfälschten Bits je Datenobjekt X poisson-verteilt:

$$\mathbb{P}\left[X = k\right] = \mathrm{e}^{-\lambda} \cdot \frac{\lambda^k}{k!} \tag{3.43}$$

Paritätstest erkennt 50% und zwar alle ungeradzahligen Bitverf.:

$$\mu_{\mathrm{MC}} = \frac{\mathbb{P}\left[X = 1\right] + \mathbb{P}\left[X = 3\right] + \ldots}{\mathbb{P}\left[X \geq 1\right]} = \frac{\mathrm{e}^{-\lambda} \cdot \left(\lambda + \frac{\lambda^3}{6} + \ldots\right)}{1 - \mathrm{e}^{-\lambda}}$$

Für $\lambda \ll 1$ und Taylor-Reihe $\mathrm{e}^{-\lambda} = 1 - \lambda + \frac{\lambda^2}{2} - \ldots$:

$$\mu_{\mathrm{MC}} = \frac{(1-\lambda) \cdot \lambda}{1 - (1-\lambda)} = 1 - \lambda \gg 50\% \tag{1.11}$$

λ	Parameter der Poissonverteilung, hier zu erwartende Bitfehleranzahl je Datenwort.
μ_{MC}	Zu erwartende Fehlfunktionsüberdeckung.

Wertebereichskontrolle

Die Menge der zulässigen Werte eines Datenobjekts ist meist viel kleiner als die Menge der darstellbaren Werte. Überwachung zusammenhängender Wertebereiche:

```
u32 a;                  // Alter Angestellter WB:18...67
                        // WB(u32):0...0xFFFFFFFF
if (a < 18 || a > 67) <Fehlerbehandlung>;
```

Wenn bei Verfälschung alle darstellbaren Werte gleichwahrscheinlich wären, Erkennungswahrscheinlichkeit:

$$\mu_{\mathrm{MC}} = 1 - \frac{\overbrace{67-18+1}^{\text{Anz. zul. Werte}}}{\underbrace{2^{32}}_{\text{Anz. mögl. Werte}}} \approx 1 - 10^{-8}$$

Würde bedeuten, dass praktisch jede Verfälschung erkannt wird, aber:

- kleine Werte stehen viel öfter als große im Speicher,
- ...

- kleine Werte stehen viel öfter als große im Speicher„
- Verwechslung mit Alter andere Person, immer zulässig,
- Verwechslung mit der Hausnummer, oft zulässig, ...

Erkennungswahrscheinlichkeit viel schlechter als (Gl. 4.9):

$$\mu_{\mathrm{MC}} \ll 1 - \frac{\#PV}{\#V}$$

Die Leistungsfähigkeit von Wertebereichskontrollen liegt in der Vielzahl der Kontrollmöglichkeiten.

Auch gibt es Möglichkeiten, die MF-Überdeckung von WB-Kontrollen insbesondere in Programmen erheblich zu erhöhen:

- WB-Verschiebung in selten genutzte Zahlenbereiche durch Addition einer Konstanten,
- pseudozufällige Codierung der zulässigen Werte, ... (vergl. fehlererkennende Codes),
- zusätzliche Typüberwachung, ...

Fehlererkennende Codes

Prinzip der fehlererkennenden Codes

Umkehrbar eindeutige pseudo-zufällige Zuordnung der zulässigen Werte zu möglichen Werte.

Wenn die Verfälschungsursache »den Code-Schlüssel« nicht kennt, entstehen durch Fehlfunktionen weder bevorzugt zulässige noch unzulässige Werte. Damit gilt

$$\mu_{\mathrm{MC}} = 1 - \frac{\#PV}{\#V} \tag{4.9}$$

Beispiel: Multiplikation mit einer ganzzahligen großen Konstanten C

$$y = C \cdot x$$

Erkennungswahrscheinlichkeit

$$\mu_{\mathrm{MC}} = 1 - \frac{\#x}{\#x \cdot C} = 1 - \frac{1}{C} \tag{1.12}$$

Vor der Verarbeitung und Speicherung Multiplikation der zu überwachenden Werte mit C. Verfälschungen werden am Divisionsrest $y\%C \neq 0$ erkannt.

C Große ganzzahlige Konstante, bevorzugt eine Primzahl.

Polynom-Multiplikation und -division

Codierung durch die Multiplikation mit einer Konstanten, allerdings nicht arithmetisch, sondern modulo-2. In Hard- oder Software einfacher als arithmetische Multiplikation:

Codierung

$$
\begin{array}{rr}
 & 10010101101 \\
\odot & 10011 \\
\hline
\oplus & 10010101101 \\
\oplus & 10010101101 \\
\oplus & 00000000000 \\
\oplus & 00000000000 \\
\oplus & 10010101101 \\
\hline
 & 1000111\,0\,0100111
\end{array}
$$

Decodierung

$$
\begin{array}{rl}
 & 1000111\,0\,0100111 : 10011 = 10010101101 \\
\oplus & 10011 \\
\hline
 & 10110 \\
\oplus & 10011 \\
\hline
 & 10101 \\
 & 10011 \\
\hline
 & 11000 \\
\oplus & 10011 \\
\hline
 & 10111 \\
\oplus & 10011 \\
\hline
 & 10011 \\
\oplus & 10011 \\
\hline
\text{Rest:} & 0000 \qquad \text{(unverfälscht)}
\end{array}
$$

In der Literatur finden Sie noch eine andere Beschreibung für die gerade vorgeführte abgewandelte Form der Multiplikation und Division von Bitvektoren. Mathematisch werden die zu multiplizierenden Faktoren als Polynome dargestellt:

- $10011 \Rightarrow 1 \cdot z^4 \oplus 0 \cdot z^3 \oplus 0 \cdot z^2 \oplus 1 \cdot z^1 \oplus 1 \cdot z^0 = z^4 \oplus z \oplus 1$
- $10010101101 \Rightarrow z^{10} \oplus z^7 \oplus z^5 \oplus z^3 \oplus z^2 \oplus 1$

Eine Multiplikation mit z beschreibt eine Verschiebung um eine Bitstelle. Die Multiplikation mit null oder eins ist eine UND-Verknüpfung und $\oplus$ die modulo-2-Addition (EXOR). Das Produkt beider Polynome

$$\left(z^{10} \oplus z^7 \oplus z^5 \oplus z^3 \oplus z^2 \oplus 1\right) \cdot \left(z^4 \oplus z \oplus 1\right) \quad =$$
$$z^{14} \oplus z^{10} \oplus z^9 \oplus z^8 \oplus z^5 \oplus z^2 \oplus z \oplus 1$$

repräsentiert denselben Bitvektor, der für die Multiplikation der Folgen auf der Folie zuvor berechnet wurde. Die Polynomdivision:

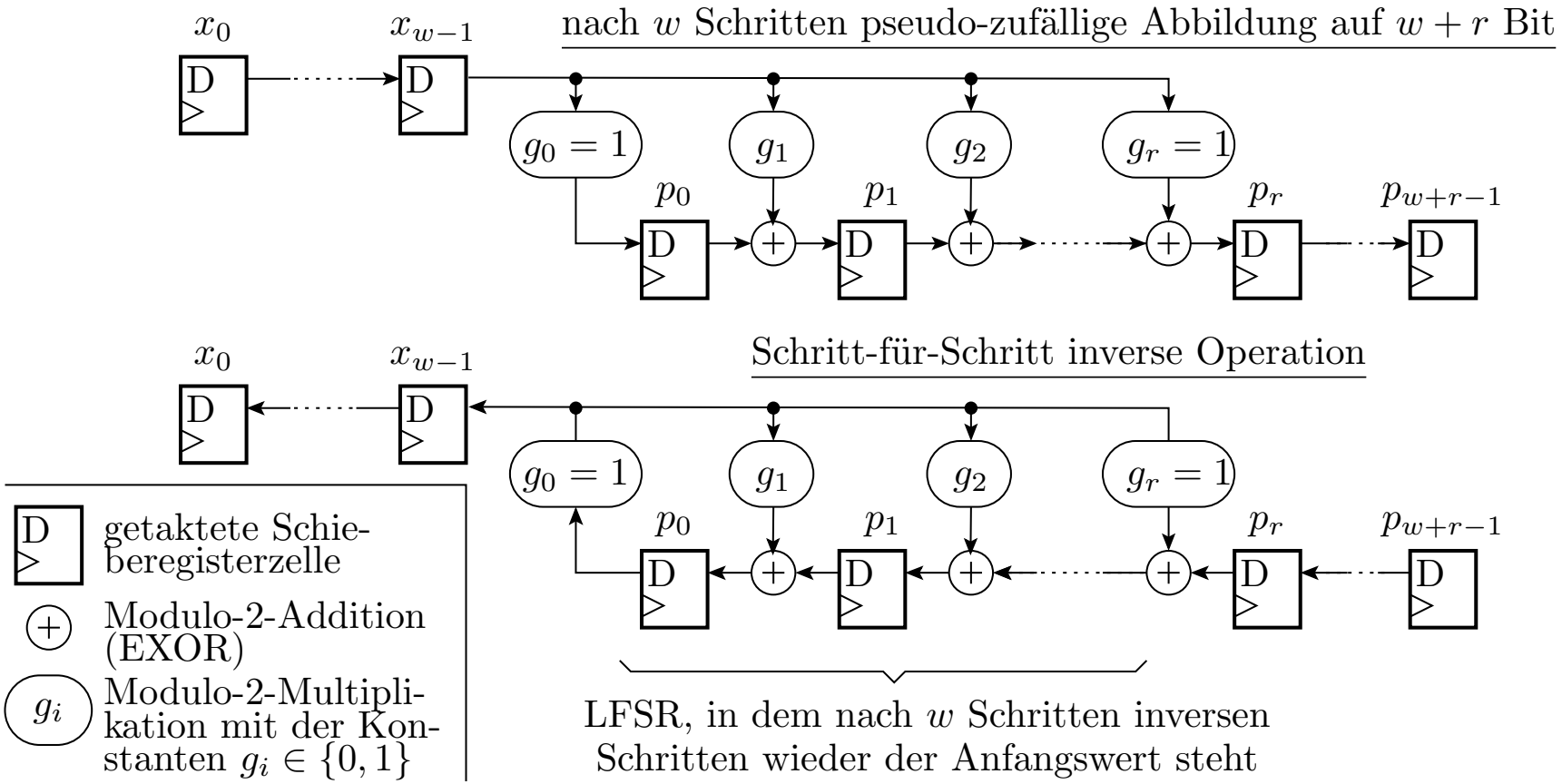
$$\left(z^{14} \oplus z^{10} \oplus z^9 \oplus z^8 \oplus z^5 \oplus z^2 \oplus z \oplus 1\right) : \left(z^4 \oplus z \oplus 1\right) \quad =$$
$$z^{10} \oplus z^7 \oplus z^5 \oplus z^3 \oplus z^2 \oplus 1$$

liefert ohne Rest das Polynom der Originalfolge.

---

z^i Logische 1 um i Bitstellen verschoben.

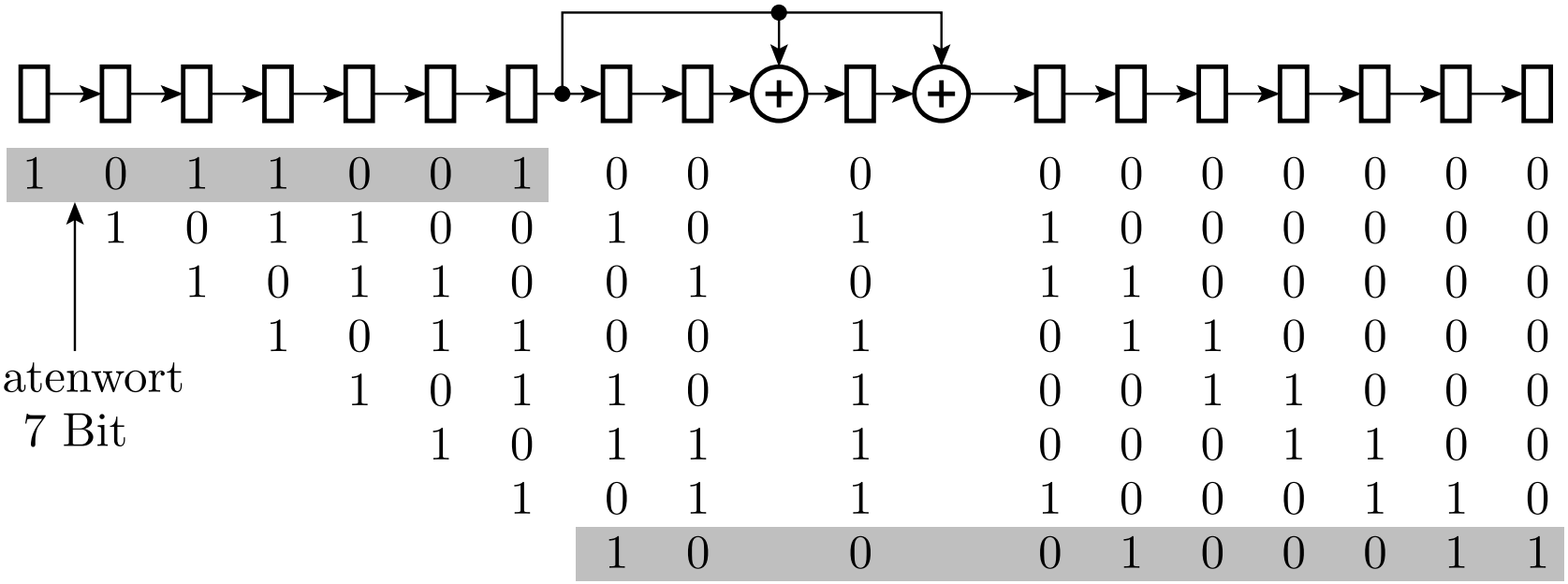
Polynommultiplikation und -division mit SR



$$\mu_{\mathrm{MC}} = 1 - 2^{-r} \tag{4.10}$$

SR	Schieberegister.
LFSR	Linear rückgekoppeltes Schieberegister (Linear feeback shift register).

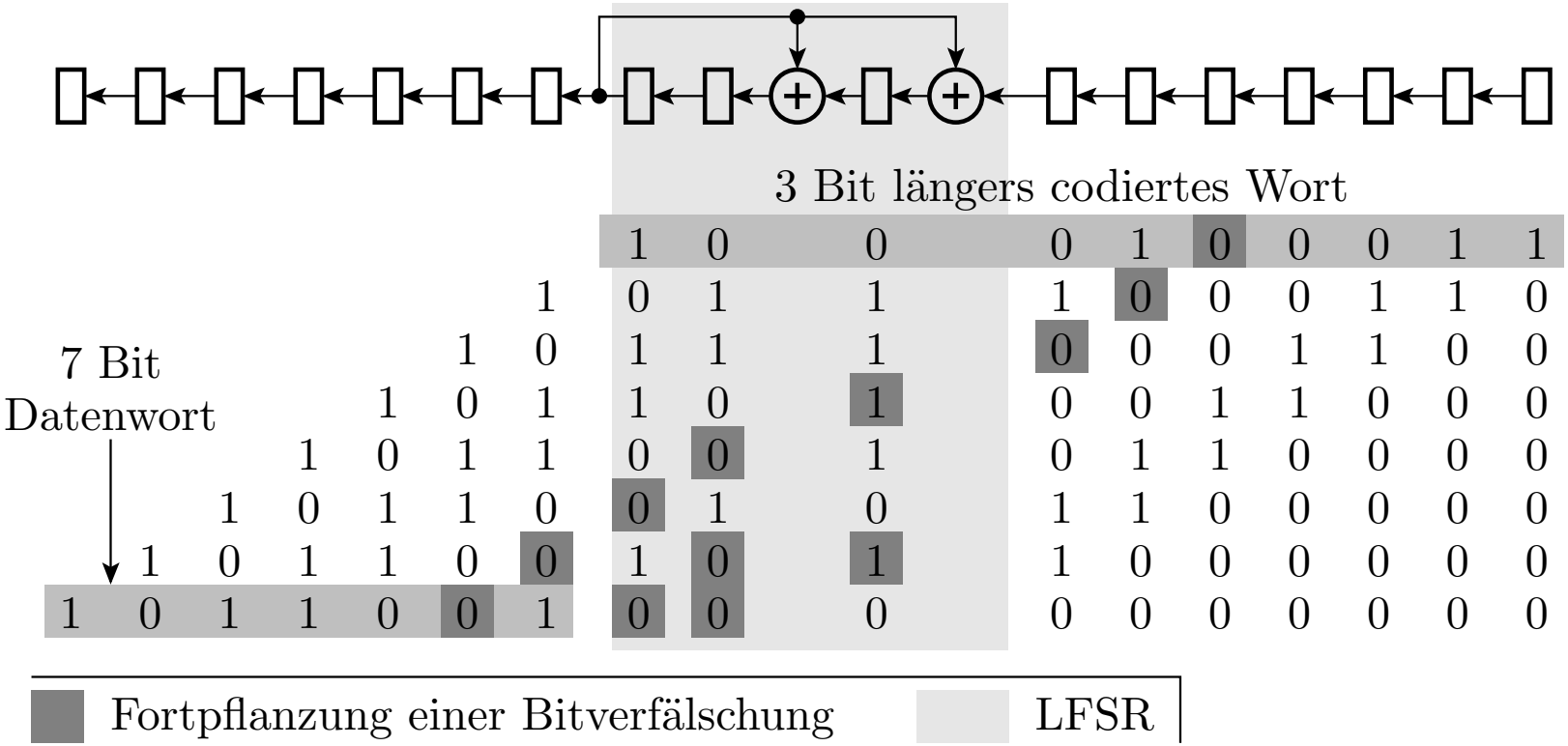
Codierung durch SR-Polynommultiplikation



- Das Ergebnis ist $r = 3$ Bit länger und pseudo-zufällig umcodiert.
- Anzahl der zulässigen Codeworte bleibt 2^7.
- Anzahl der möglichen Codeworte vergrößert sich auf 2^{10}.

r Anzahl der redundanten Bits.

Rückgewinnung durch Polynomdivision



							1	0	0	0	1	0	0	0	1	1		
						1	0	1	1	1	0	0	0	1	1	0		
					1	0	1	1	1	0	0	0	1	1	0	0		
				1	0	1	1	0	1	0	0	1	1	0	0	0		
			1	0	1	1	0	0	1	0	1	1	0	0	0	0		
		1	0	1	1	0	0	1	0	1	1	0	0	0	0	0		
	1	0	1	1	0	0	1	0	1	1	0	0	0	0	0	0		
1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0		

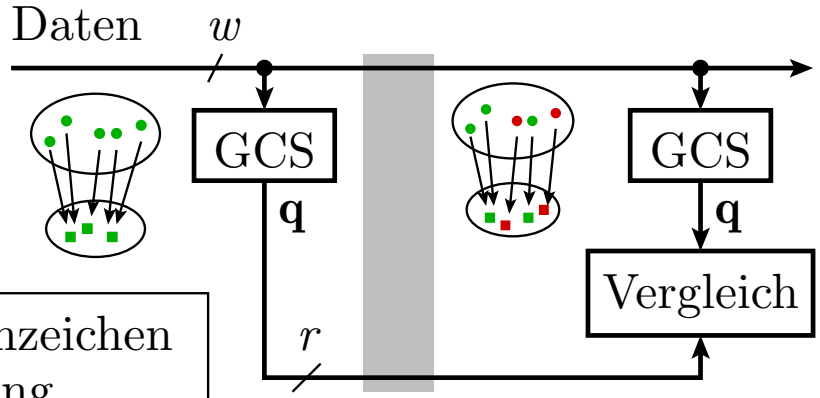
Eine Bitverfälschung verursacht mit der Wahrscheinlichkeit $p_{\mathrm{D}} = 1 - 2^{-3}$ einen von null abweichenden Endwert im LFSR.

LFSR Linear rückgekoppeltes Schieberegister (Linear feeback shift register).

p_{D} Nachweiswahrscheinlichkeit (Detection probability).

Prüfkennzeichen

Prüfkennzeichen



- Jedem w-Bit-Datenwort wird pseudo-zufällig genau eines der r-Bit-Prüfkennzeichen **q** zugeordnet $(w \gg r)$.
- Nach der Übertragung oder Speicherung wird das Prüfkennzeichen ein zweites mal gebildet.
- Wenn weder die Daten noch das Prüfkennzeichen verfälscht sind, stimmen beide Prüfkennzeichen überein.

Für pseudo-zufällig gebildete Prüfkennzeichen gilt:

- Anzahl der zulässigen Prüfkennzeichen-Werte-Paare 2^w,
- Anzahl darstellbarer Paare 2^{w+r}:

$$\mu_{\mathrm{MC}} = 1 - 2^{-r} \tag{4.10}$$

r Anzahl der redundanten Bits.
w Anzahl der Datenbits (Number of data bits).

Prüfsummen

Prüfzeichbildung durch byteweise Addition oder EXOR-Verknüpfung.

einfache Genauigkeit	doppelte Genauigkeit	bitweises EXOR
1011 11	1011 11	1011
0010 2	0010 2	0110
1101 13	1101 13	1101
0100 4	0100 4	1100
(1) 1110 14 (+16)	0001 1110 30	1100

Bei »einfacher Genauigkeit« und »bitweisem EXOR« erscheint die Annahme »pseudo-zufällige Abbildung« gerechtfertigt*:

$$\mu_{\mathrm{MC}} \approx 1 - 2^{-4}$$

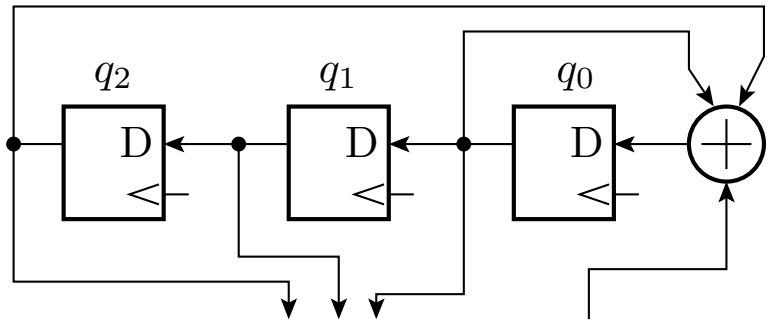
Bei »doppelter Genauigkeit« bilden sich Verfälschungen vorzugsweise auf die niederwertigen Bits ab:

$$1 - 2^{-4} \leq \mu_{\mathrm{MC}} < 1 - 2^{-8}$$

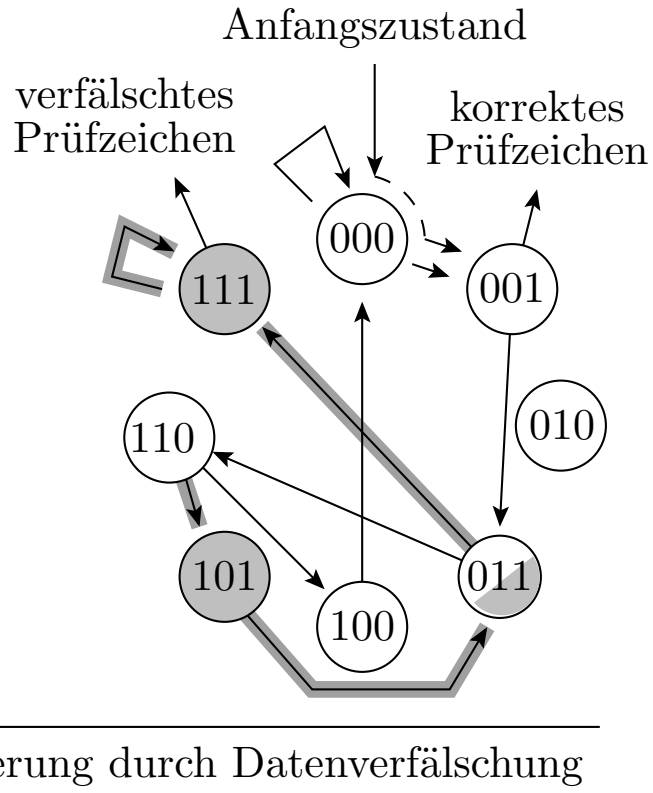
μ_{MC} Zu erwartende Fehlfunktionsüberdeckung.

* Kein Nachweis für vertauschte Summationsreihenfolge.

Prüfkennzeichenbildung mit LFSR



Initialwert	0	0	0	1
Schritt 1:	0	0	1	0
Schritt 2:	0	1	1	1
Schritt 3:	1	1	0	1
Schritt 4:	1	0	0	1
Schritt 5:	0	0	0	0
Schritt 6:	0	0	0	1
Schritt 7:	0	0	1	

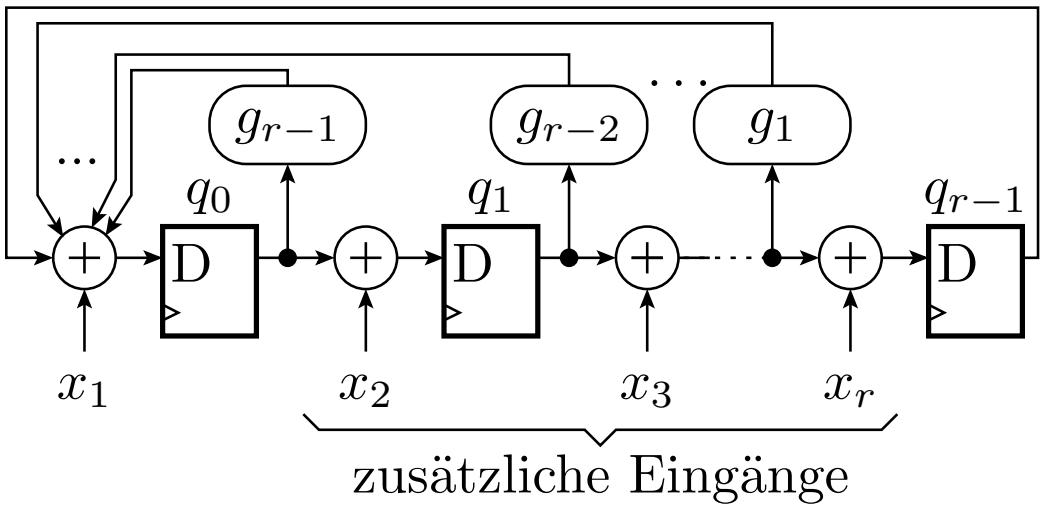


Im Beispiel hat das LSFR abweichend von Polynomdivision (siehe Folie 4.105) eine zentrale Rückführung. Abbildung auch pseudo-zufällig.

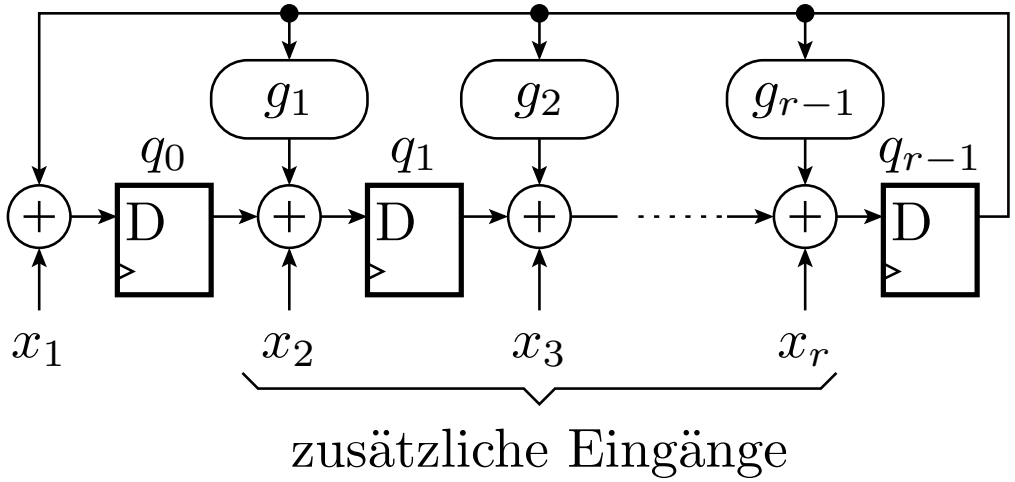
q_i	Registerbit i des linear rückgekoppelten Schieberegisters.
LFSR	Linear rückgekoppeltes Schieberegister (Linear feeback shift register).

Allgemeine Struktur von LFSR

Für die Bildung auf Prüfzeichen ist es nur wichtig, dass die Abbildung pseudo-zufällig hinsichtlich der zu erwartenden Verfälschungen erfolgt. Diese Eigenschaft hat auch ein rückgekoppeltes Schieberegister, bei dem in jedem Zeitschritt mehrere Eingabebits modulo-2 zu den Registerzuständen addiert werden.



x_i	Bitfolge an Testobjektausgang i.
g_i	Rückführstellen und gleichzeitig Bits des Generatorpolynoms.
q_i	Registerbit i des linear rückgekoppelten Schieberegisters.
r	Bitanzahl des linear rückgekoppelten Schieberegisters.



Die Rückführung darf dabei auch wie bei der Polynom-Division dezentral sein. Die Koeffizienten g_i der Rückführung, bei der Polynom-Division das Divisor-Polynom, bestimmen die autonome Zyklusstruktur*. Die autonome Zyklusstruktur ist bei zentraler und dezentraler Rückführung mit denselben Rückführkoeffizienten gleich.

Bevorzugt werden lange Zyklen, insbesondere sog. primitive Polynome, bei denen alle Zustände außer »alles null« einen $2^r - 1$ langen Maximalzyklus bilden.

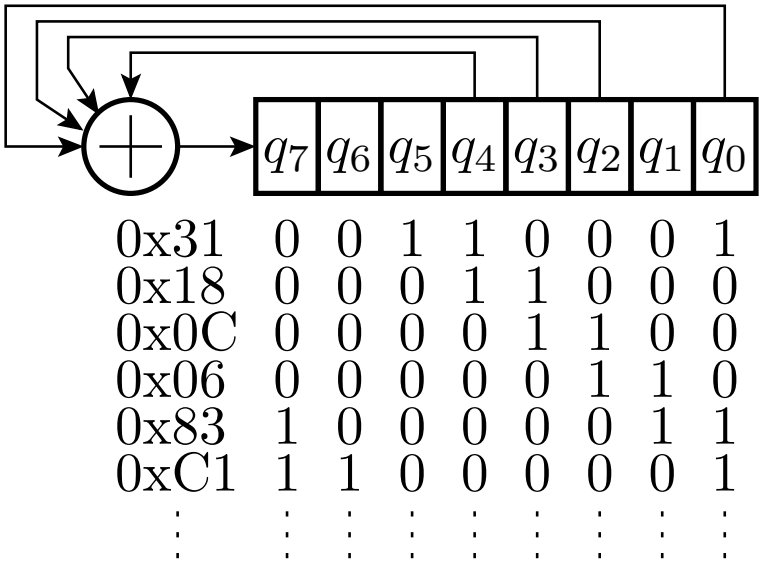
---

r Bitanzahl des linear rückgekoppelten Schieberegisters.

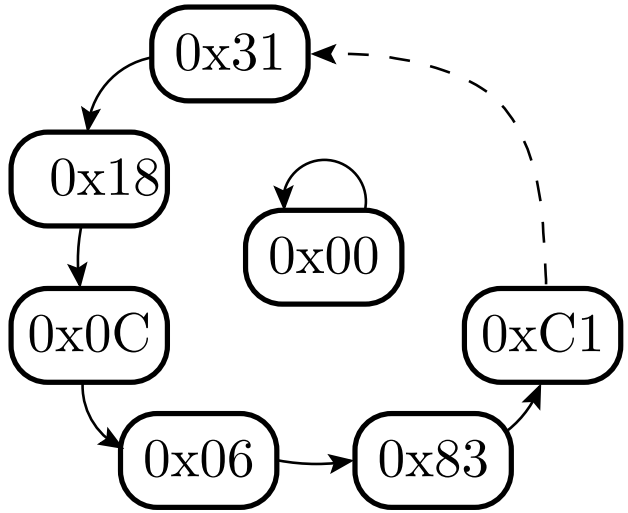
* Zyklusstruktur ohne Eingaben.

Autonome Zykluslänge und -struktur von LFSR

Autonom bedeutet Zyklusstrukur für Eingabewerte »alle 0« oder für LFSR ohne Eingänge. Beispiel 8-Bit autonomes LFSR:



0x31	0	0	1	1	0	0	0	1
0x18	0	0	0	1	1	0	0	0
0x0C	0	0	0	0	1	1	0	0
0x06	0	0	0	0	0	1	1	0
0x83	1	0	0	0	0	0	1	1
0xC1	1	1	0	0	0	0	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

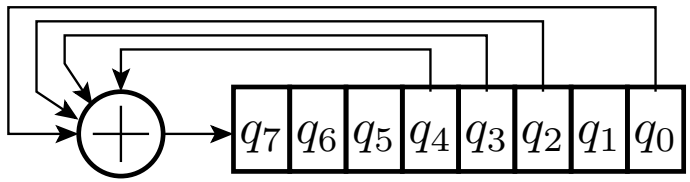


Übergangsfunktion:

$$\begin{aligned} q_7 &= q_4 \oplus q_3 \oplus q_2 \oplus q_0 \\ q_i &= q_{i+1} \text{ für } i \in \{0, 1, 2, \ldots, 6\} \end{aligned}$$

q_i Bitwerte des linear rückgekoppelten Schieberegisters.

Bestimmung der Zykluslänge durch Simulation

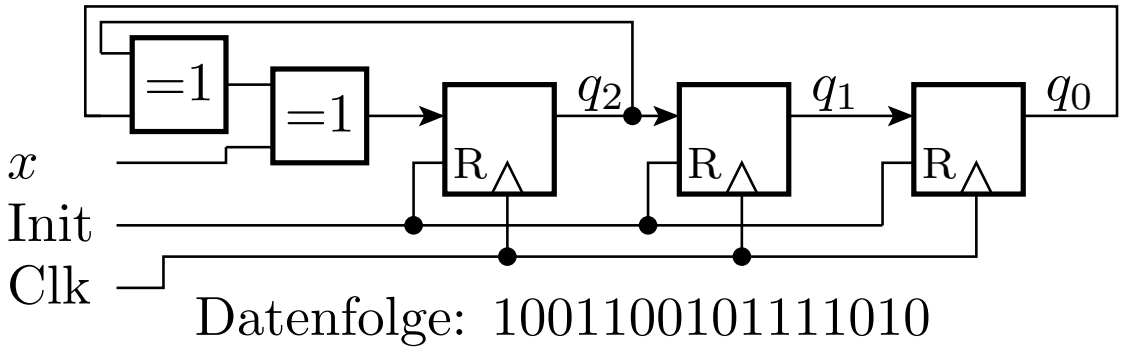



```
...
#define SA 0x31
uint8_t q = SA;          // Startwert setzen
while(1){
  q = (q>>1) ^ (q<<7) ^ ((q<<5)&0x80)
      ^ ((q<<4)&0x80) ^ ((q<<3)&0x80);
  <Ausgabe von s>
  if (q==SA) break;  // bis wieder Anfangswert
  ...                // weiter mit nicht enthal-
}                    // tenem Zustand als Startw.
```

- 0x00 geht in sich selbst über.
- Alle anderes 255 Zustände gehen zyklisch ineinander über.
- max. Zykluslänge $2^r - 1$: primitive Rückkopplung.

Beispiel 4.3: Prüfkennzeichen mit LFSR

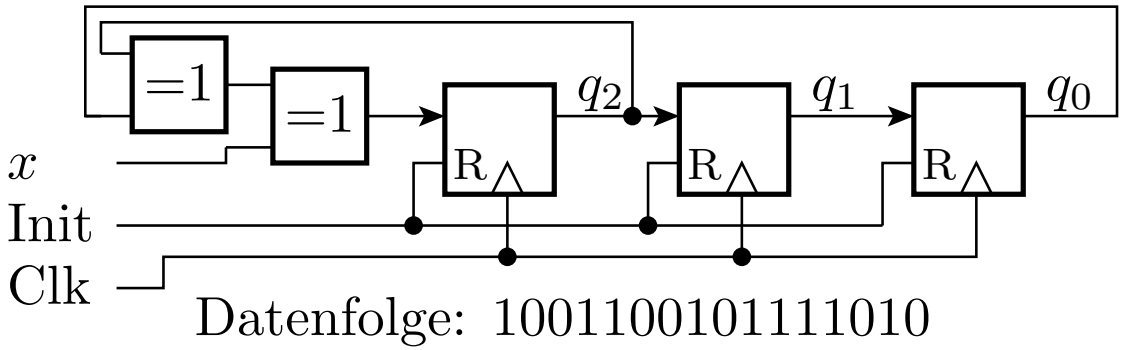
Gegeben ist folgendes linear rückgekoppelte Schieberegister:



a) *Welches Prüfkennzeichen* $\mathbf{q} = q_2 q_1 q_0$ *hat die Datenfolge bei Abbildung beginnend mit dem linken Bit und Startwert 000?*

b) *Wie hoch ist die zu erwartende Fehlfunktionsübertdeckung?*

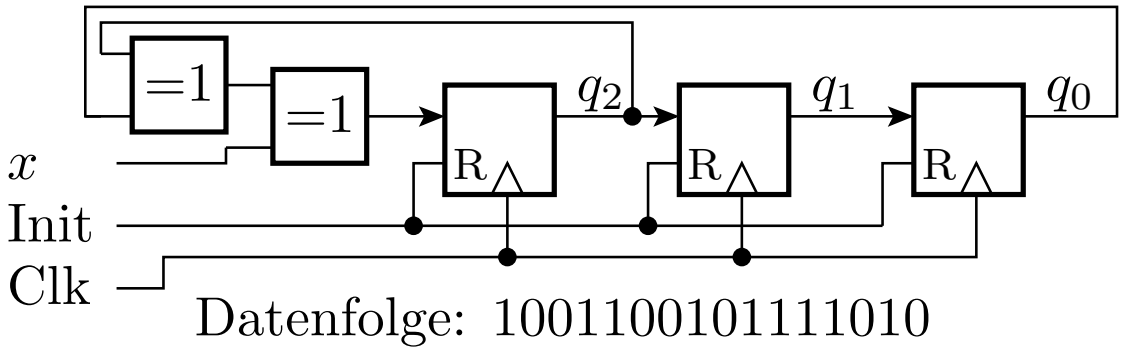
q_i	Bitwerte des linear rückgekoppelten Schieberegisters.
x	Eingang zur Abtastung der Bitfolge an Testobjektausgang.
R	Rücksetzeingang.
Init	Initialisierungssignal.
Clk	Taktsignal (Clock signal).



a) *Welches Prüfkennzeichen* $\mathbf{q} = q_2 q_1 q_0$ *hat die Datenfolge bei Abbildung beginnend mit dem linken Bit und Startwert 000?*

	x	q_2	q_1	q_0
0	1	0	0	0
1	0	1	0	0
2	0	1	1	0
3	1	1	1	1
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	1	1	0	1

	x	q_2	q_1	q_0
8	0	1	1	0
9	1	1	1	1
10	1	1	1	1
11	1	1	1	1
12	1	1	1	1
13	0	1	1	1
14	1	0	1	1
15	0	0	0	1
Prüfkennzeichen: 16		1	0	0

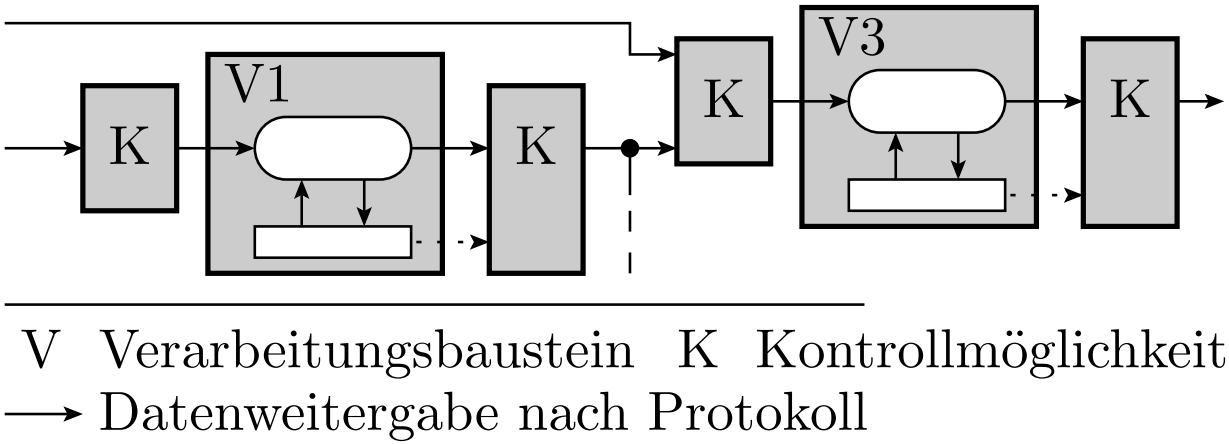


b) *Wie hoch ist die zu erwartende Fehlfunktionsübertdeckung?*

$$\mu_{\mathrm{MC}} = 1 - 2^{-3} = 87{,}5\%$$

μ_{MC} Zu erwartende Fehlfunktionsüberdeckung.

Protokolle



IT-Systeme bestehen in der Regel aus einer Vielzahl komplexer Bausteine, die über wohldefinierte Schnittstellen und Protokolle miteinander kommunizieren:

- Software-Schnittstellen, Busprotokoll,
- Netzwerkprotokolle, Signaldefinitionen, Internet-Protokolle, ...

Für Prozessoren, APIs, ... hunderte Seiten Dokumentation

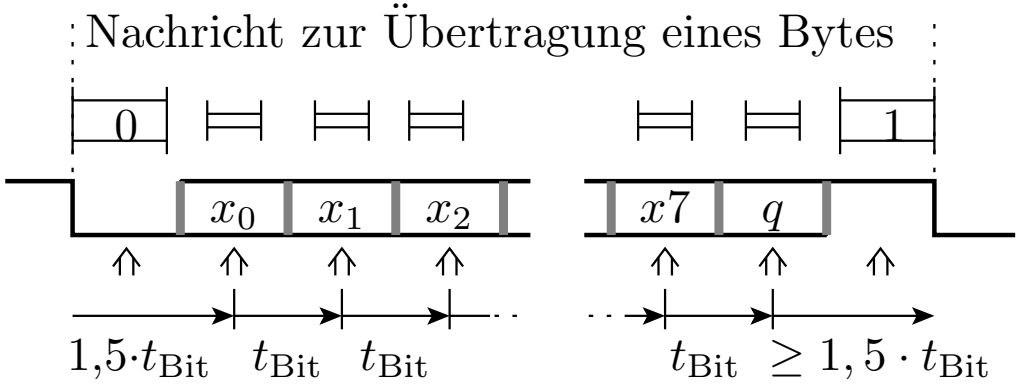
- welche Daten wie, mit welchen Befehlen in welcher Reihenfolge bei der Anforderung welcher SL zu übergeben sind,
- welche Vorbedingungen die Daten zu erfüllen haben,
- welche überprüfbaren Bedingungen korrekte Ergebnisse erfüllen,
- Antwortzeitschranken , ...

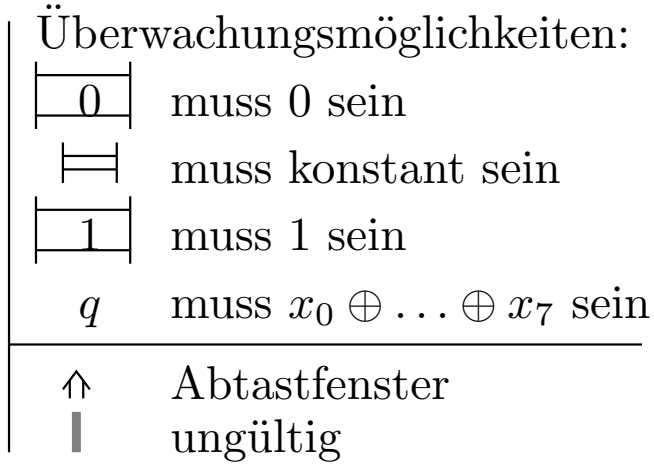
$\Rightarrow$ unzählige Möglichkeiten für Formatkontrollen.

Beispiel UART

Das UART-Protokoll ist der Klassiker zur bytweisen Datenübertragung zwischen Mikrorechner über eine Leitung je Übertragungsrichtung:

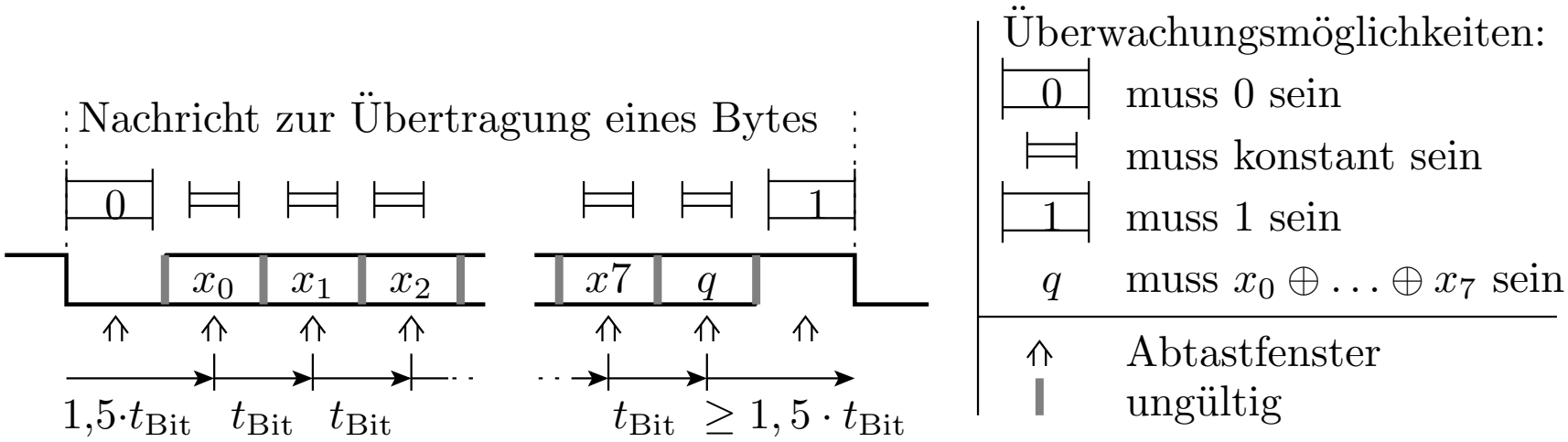
- Baud-Rate (Bitzeiten pro s): 4800, 9600, ...
- 1 Startbit mit Wert 0,
- 7 / 8 /9 Datenbits,
- kein /gerades / ungerades Paritätsbit und
- 1 / 2 Stoppbit(s) mit Wert 1.





UART	Universal Asynchronous Receiver Transmitter.
x_i	Datenbit i.
q	Partitätsbit.
t_{Bit}	Bitzeit.

Kontrollierbare Formateigenschaften

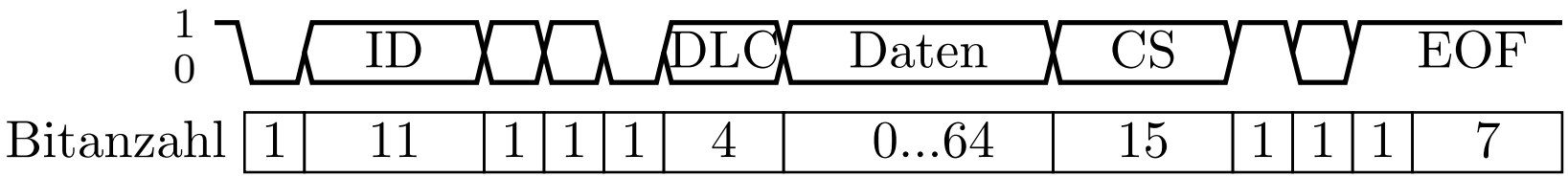


Erkennbare Formatfehler

- Paritätsfehler: Invertierung von 1, 3, 5, oder 7 Bit.
- Datenrahmenfehler: Datenwechsel in Zeitfenstern, in denen keine Änderung erlaubt. Zu kurze Start- oder Stoppbitzeit.
- Datenüberlauf: mehr ankommende Bytes als der Empfänger verarbeiten kann.

Überwachung durch die Prozessor-Hardware. Benachrichtigung der Software durch Setzen von durch die SW abfragbare Bits in Spezialregistern der HW.

CAN-Bus



ID	Nachrichtennummer	CS	15-Bit Prüfkennzeichen
DLC	Datenlängencode	EOF	Ende des Datenrahmens

Eingesetzt z.B. zur Vernetzung von Fahrzeugsteuergeräten.

Erkennbare Formatfehler:

- 15-Bit Prüfzeichen, Erkennungssicherheit:

$$p_{\mathrm{D}} = 1 - 2^{-15} \approx 1 - 3 \cdot 10^{-5}$$

- Soll-/Ist-Abweichung konstante Bits, unzul. Datenänderungszeiten.

Fehlerfunktionsbehandlung:

- Wiederholung bei Nachrichtenkollision,
- Notfallreaktion bei Ausfall anderer Steuergeräte,
- Fehlerspeicher für aufgetretene Fehlfunktionen, ...

Ethernet-Protokoll

Sicherungs-schicht			MAC-Empfänger	MAC-Absender	Protokoll-typ	Nutzlast max. 1500 Bytes	Prüfkennzeichen 4 Byte	
Bitübertra-gungsschicht	Präambel	Start-byte						Lücke zum nächsten Paket
Byteanzahl	7	1	6	6	2	46 bis 1500	4	12

Erkennbare Formatfehler:

- 4-Byte-Prüfzeichen:

$$\mu_{\mathrm{FC}} = 1 - 2^{-32} \approx 1 - 2 \cdot 10^{-10}$$

- korrekte Präambel, korrektes Startbyte, ...
- Wiederholanforderung bei fehlerhaft empfangenen oder nicht erhaltenen Datenpaketen, ...

Mittlere Zeit zwischen dem Empfang nicht erkannter verfälschter Datenpakete bei 10^5 Datenpaketen pro s und 10^{-4} MF je Datenpaket:

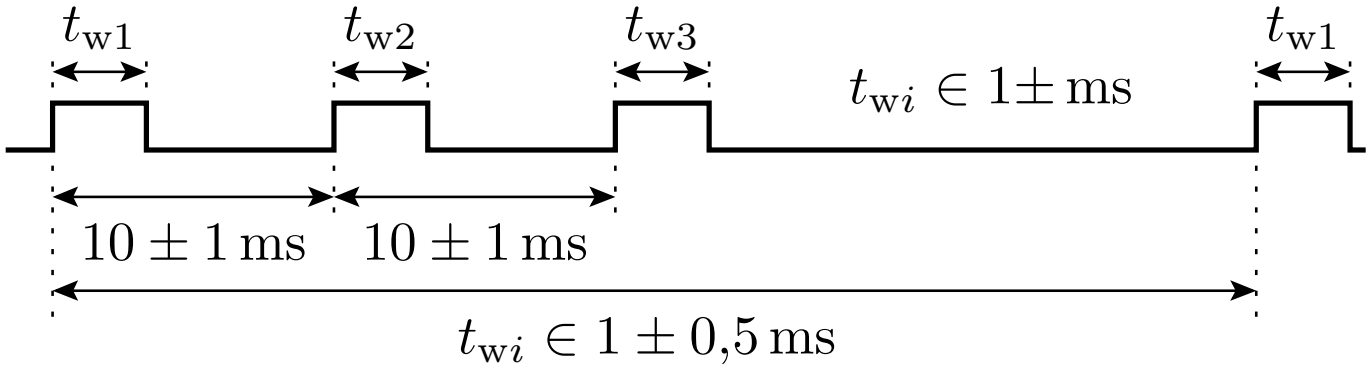
$$\frac{1}{10^5 \ \left[\frac{\mathrm{DP}}{\mathrm{s}}\right] \cdot 10^{-4} \ \left[\frac{\mathrm{MF}}{\mathrm{DP}}\right] \cdot 2^{-32}} > 13{,}5\,\mathrm{Jahre}$$

$\left[\frac{\mathrm{DP}}{\mathrm{s}}\right]$ Anzahl der Datenpakete pro Sekunde.

$\left[\frac{\mathrm{MF}}{\mathrm{DP}}\right]$ Zählwertverhältnis in Fehlfunktion je Datenpaket.

Signalintegrität, Beispiel PWM

Pulsweitenmodulierte Signale (PWM) codieren die Information in den Pulsbreiten $t_{\mathrm{w}i}$ und werden u.a. zur Datenweitergabe von Sensoren an Rechner und von Rechnern an Aktoren genutzt.

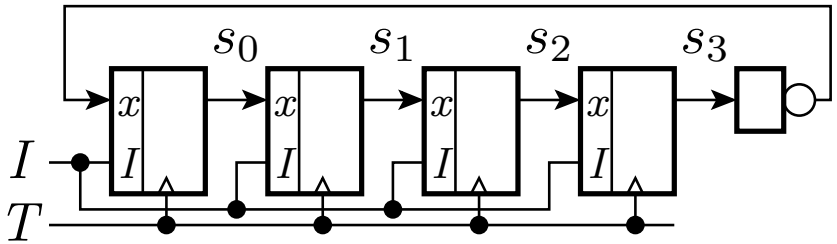


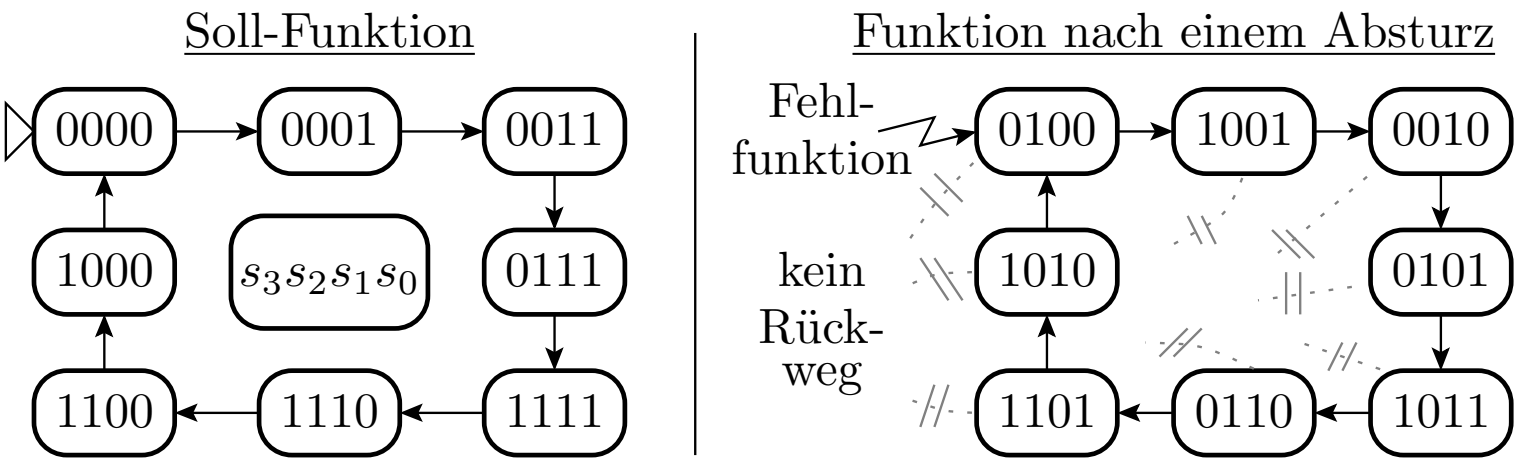
Überwachbare Formateigenschaften:

- Periodendauer,
- minimale und maximale Pulsbreite,
- Amplitude der Pulse.

Zeitüberwachung

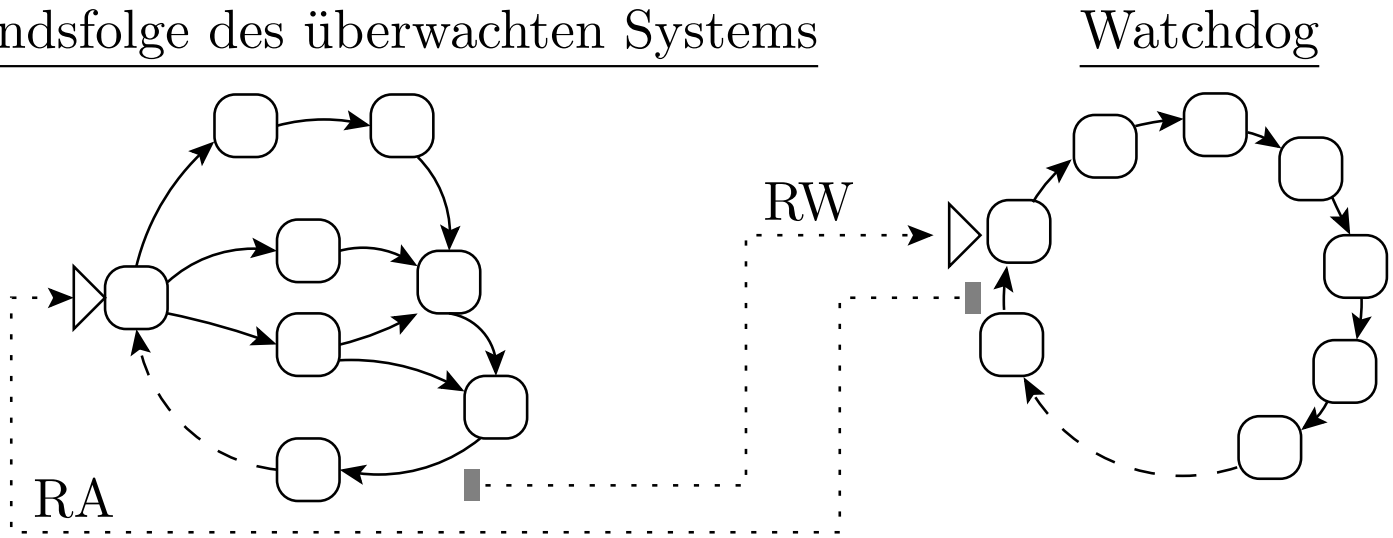
Redundante Zustände und Systemabsturz





- Automaten und Programme nutzen nur einen kleinen Teil der $2^{\#SB}$ möglichen Zustände ($\#SB$ – Anzahl Zustandsbits).
- Wenn eine MF den Zustand so verfälscht, dass nie wieder ein zulässiger Zustand ereicht wird, Absturz.
- Bei komplexer HW und SW ist $\#SB$ um Zehnerpotenzen größer 4. Unüberschaubar viele Absturzmöglichkeiten.
- Abstürze sind lässtige, gut erkennbare, häufig beobachtete MF.

Watchdog



RA Ein Überlauf des Watchdogs initialisiert den Automaten neu.

RW Bei einem bestimmten, regelmäßig stattfindenden Zustandsübergang wird der Watchdog neu initialisiert.

Das überwachte System setzt in periodisch zu erreichenden Sollzuständen einen Zeitzähler zurück, der bei Überlauf das System neu startet und dabei auch wieder einen zulässigen Zustand herstellt.

Das Watchdog-Prinzip wird angewendet für

- einzelne Aufgaben,
- einzelne Programme in Multi-Task-Systemen,
- komplette Rechner, ...

Prozessoren haben in der Regel einen Hardware-Watchdog

- mit programmierbarer Zeit bis zum Überlauf,
- der, wenn eingeschaltet, nicht vom Programm, d.h. auch nicht durch Fehlfunktionen, ausschaltbar ist, sondern nur durch Neustart,
- bei Überlauf einen Betriebssystemaufruf zur Fehlerbehandlung und/oder einen Rechnerneustart auslöst.

[Debugger & Watchdog]

Invarianten

Invarianten

Invarianten: Bedingungen, die unabhängig von den zu verarbeitenden Daten immer erfüllt sein müssen. Überprüfbar

- durch Beweis, bei der Compilierung,
- durch Überwachung während der Laufzeit.

  [RUST Test Fail Fast, Release Fail Slow]

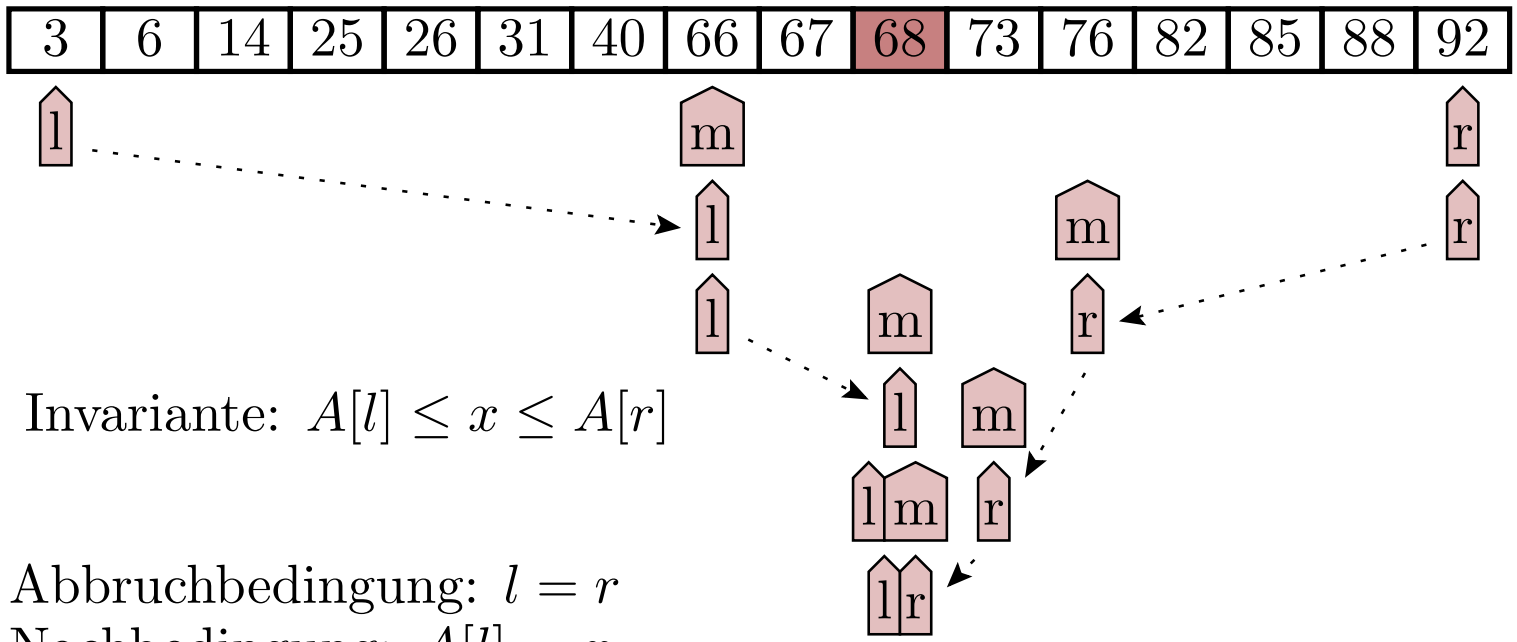
Beispiele für Invarianten:

- Sortieren einer Liste: Anzahl und Summe der Elemente.
- Task-Liste: die max. Anzahl der aktiven Tasks.
- Variable: Wertebereich,
- Zeiger: null oder Adresse eines Objekts mit zulässigem Typ.
- Geschlossenes physikalisches System: Energie, Impuls, ...
- Iteration: Vorbedingungen, Schleifeninvarianten, Nachbedingungen.
- ...

Beispiel mit einer Schleifeninvarianten

gesucht: Position Schlüssel x

Vorbedingung: Feld $A[n]$ aufsteigend sortiert, n Elemente



Innerhalb der binären Suche nach dem Element mit Schlüsse x darf $A[l]$ nie größer und $A[r]$ nie kleiner als der gesuchte Schlüssel sein.

Syntax

Syntaxtest

Kontrolle, ob eine Zeichenfolge ein Wort einer formalen Sprache ist.

Formale Sprache: Definition zulässiger Zeichenfolgen durch Syntaxregeln, deren Einhaltung sich durch einen spracherkennenden Automaten kontrollieren lässt.

Statischer Test für manuell erstellte oder bearbeitete Programme und Programmeingaben.

Ein spracherkennender Automat ist zwar selbst kein einfaches, schnell zu schreibendes Programm, aber das Programm für einen spracherkennenden Automaten lässt sich nach bekannten Algorithmen aus den Syntaxregeln der Sprache generieren.

Syntaxtests erkennen viele menschengemachter Fehler. Für maschinell erzeugte Daten, die nicht manuell zu bearbeiten sind, eignet sich die Weitergabe mit Prüfzeichen oder verschlüsselt mit fehlererkennenden oder korrigierenden Codes besser.

EBNF als Beispielregelwerk für formale Sprachen

Beschreibungsmittel der EBNF zur Definition von Sprachregeln:

Beschreibungsmittel	EBNF-Darstellung
Definition	NTS = Ersetzungsregel
Aufzählung	..., ...
Endezeichen	...;
Alternative	...|...
Option	[...]
Wiederholung	{...}
Gruppierung	(...)
Zeichenkette (Terminalsymbolfolge)	"..." oder '...'

EBNF Erweiterete Backus-Naur-Form.
NTS Nicht-Terminalsymbol, zu ersetzendes Symbol.

Beispiele für EBNF-Syntaxregeln

```
Zahl = ['-'],((ZiffernAusserNull,{Ziffer})|'0');
ZifferAusserNull = '1'|'2'|'3'| ... |'9';
Ziffer = ZifferAusserNull | '0';
```

Beispielzeichenfolgen:

```
'-1001': zulässig
'3,23' : nur bis Komma zulässig
'010'  : nur vor 1 zulässig
```

```
Bezeichner = Buchstabe,{Buchstabe|Ziffer},TZ;
TZ = ','|';'|Leerzeichen|...
```

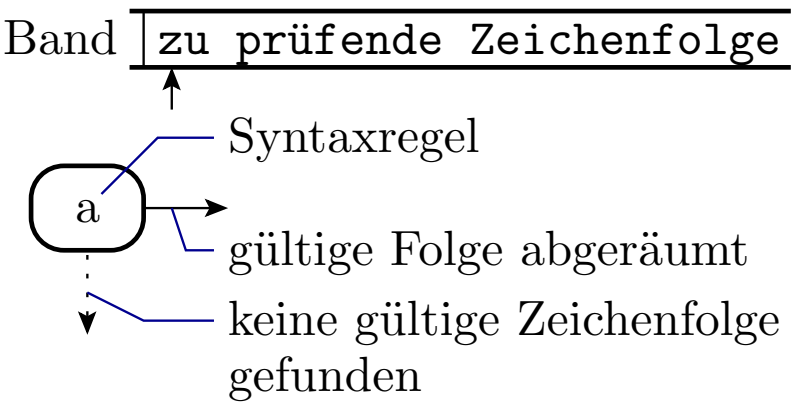
Beispielzeichenfolgen:

```
'a100;': zulässig,
'aa_3,': unzulässig wegen '_'
'1A'   : unzulässig wegen führender Ziffer
```

Spracherkennende Automaten

Die zu prüfende Zeichenfolge liegt auf einem Band mit einem Zeiger auf den Anfang.

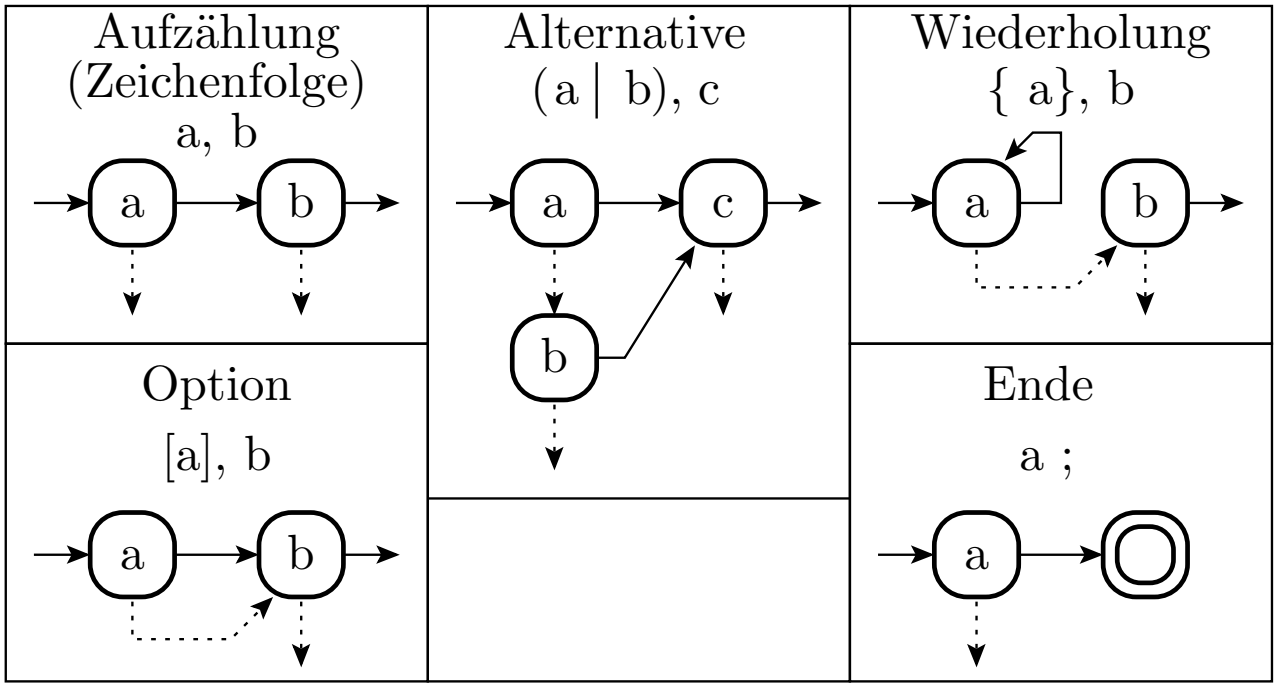
In jedem Automatenzustand wird versucht, eine Zeichenfolge nach definierter Syntaxregel abzuräumen:



- Wenn möglich, wandert der Zeiger zum ersten Zeichen nach der abgeräumten Folge und der Knoten wird über $\longrightarrow$ verlassen.
- Sonst bleibt der Zeiger und der Knoten wird über $\downarrow$ verlassen.

$\downarrow$-Übergänge ohne dargestellten Zielknoten enden im Fehlerzustand.

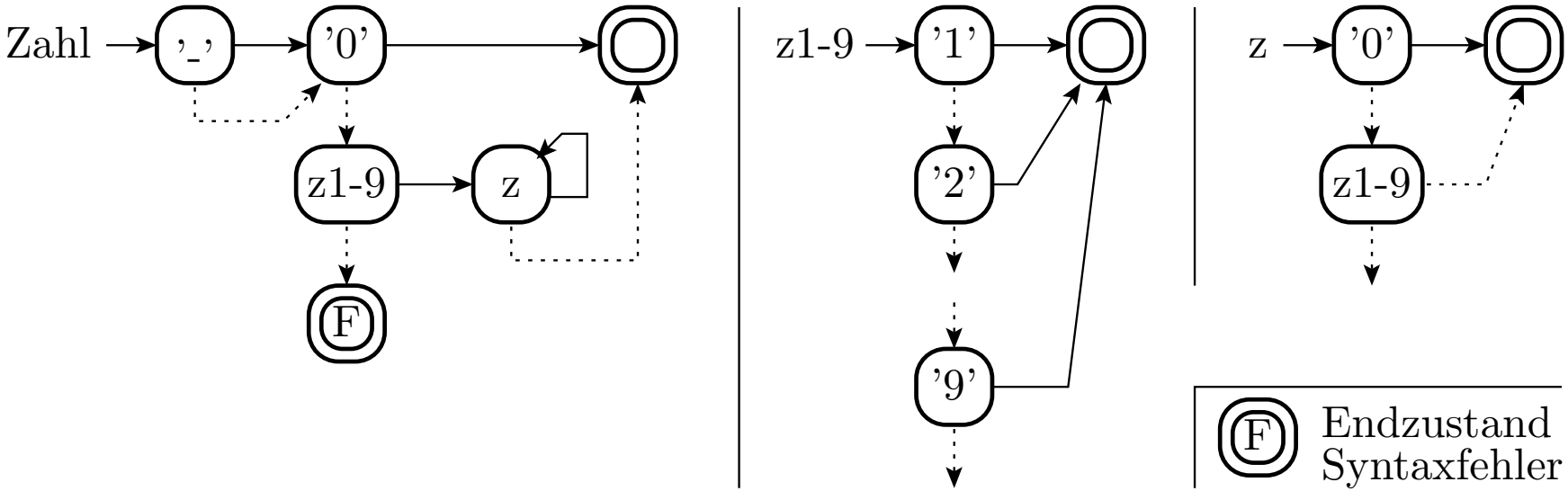
Von der EBNF zum Automaten



Beispiel:

```
Zahl = ['-'],((z1-9, {z}) | '0');
    z1-9 = '1' | '2' | ... | '9';
    z    = z1-9 | '0';
```

```
Zahl = ['-'],((z1-9, {z}) | '0');
z1-9 = '1' | '2' | ... | '9';
z    = z1-9 | '0';
```



Welche Zustände durchläuft der Automat. Was erkennt er?

- "125_": '-'↓, '0'↓, z1-9→, z→, z→, z↓, Endzustand: $125\underline{\surd}^{\downarrow}$
- "k89": '-'↓, '0'↓, z1-9↓, Endzustand: Syntaxfehler, $\overset{\downarrow}{k}$
- "-0701" : '-'→, '0' →, Endzustand: $0\overset{\downarrow}{\surd}7$

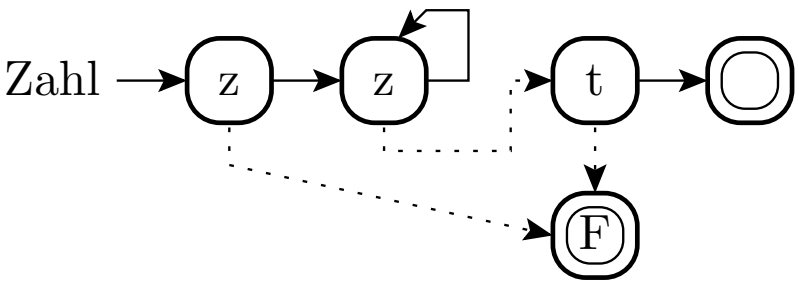
Abräumen an den Kanten

- Abräumen der Zeichen an den Kanten.
- Beschreibung derselben Syntaxregeln mit weniger Zuständen.
- Die »Sonst-Kanten« zum Fehlerzustand können weggelassen werden.

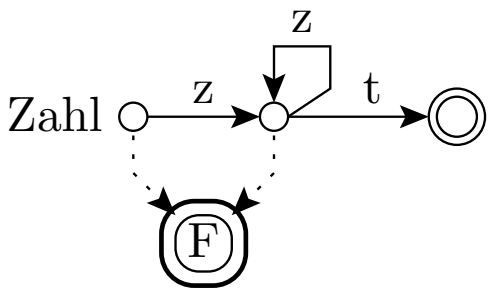
Beschreibung einer Zahl:

Zahl = z, {z}, t; z = '0' | '1' | ... | '9';
t = ' ' | ',' | ...; (Trennzeichen)

Abräumen in den Zuständen

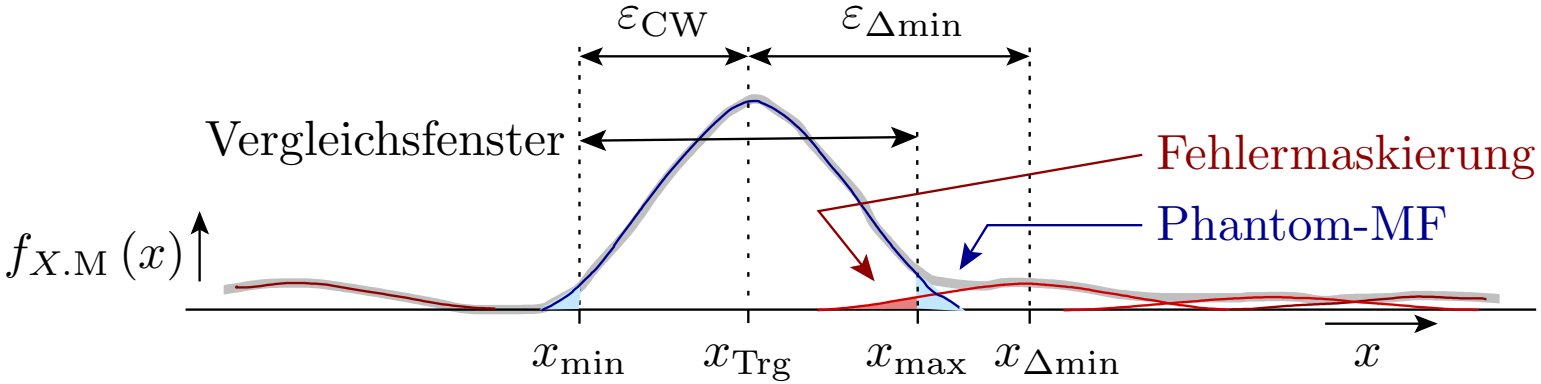


Abräumen an den Kanten



Zusammenfassung

Vergleich



- Bitweiser Vergleich ist robust gegen Störungen. Zum Ausschluss von Fehlern im Vergleicher und falschen Sollwerten sind Vergleiche und Sollwerte zu testen.
- Durch Bits dargestellte Information kann zufällige Einflüssen unterliegen (Rauschen, Quantisierungsfehler, Rundungsfehler, ...). Fenstervergleich. Phantom-MF-Rate:

$$\zeta_{\mathrm{Phan}} = 2 \cdot \left(1 - \Phi\left(\frac{\varepsilon_{\mathrm{CW}}}{\sigma_{\mathrm{D}}}\right)\right) \tag{4.7}$$

- Maskierungswahrscheinlichkeit:

$$p_{\mathrm{M}} = 1 - \zeta_{\Delta\,\mathrm{min}} \cdot \Phi\left(\frac{\varepsilon_{\Delta\mathrm{min}} - \varepsilon_{\mathrm{CW}}}{\sigma_{\mathrm{D}}}\right) \tag{4.8}$$

.

Informationsredundanz

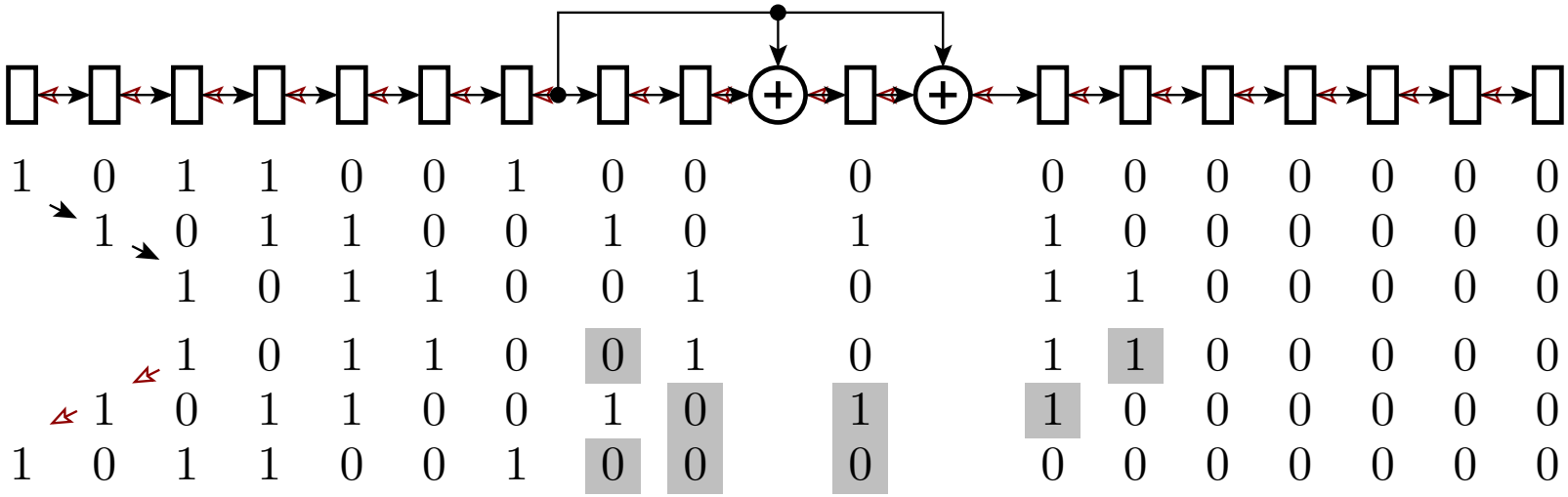
- Bei gleichmäßiger pseudozufälliger Abbildung von Verfälschungen auf zulässige und erkennbar unzulässige Werte:
$$\mu_{\mathrm{MC}} = 1 - \frac{\#PV}{\#V} \tag{4.9}$$
und keine Phantom-MF.
- Rechtschreibtest durch Kontrolle ob Wörter im Wörterbuch:
$$\mu_{\mathrm{MC}} \ll 1 - \frac{\#PV}{\#V}; \quad \zeta_{\mathrm{Phan}} > 0$$
- Paritätstest und poisson-verteilte Anzahl der verfälschten Bits bei Speicherung und Übertragung:
$$\mu_{\mathrm{MC}} = \frac{(1-\lambda)\cdot\lambda}{1-(1-\lambda)} = 1 - \lambda \tag{4.11}$$
nach (Gl. 4.9) nur 50%, also für kleine λ viel besser.
- Wertebereichstests im allgemeinen viel schlechter als Gl. 4.9, aber viele Kontrollmöglichkeiten.

Fehlererkennende Codes

- Pseudo-zufällige umkehrbar eindeutige Abbildung von $\#PV$ zulässigen auf $\#V \gg \#PV$ mögliche Datenworte, z.B. durch Multiplikation mit einer großen Konstanten. Rückgewinnung durch Division und Kontrolle Divisionsrest:

$$\mu_{\mathrm{MC}} = 1 - \frac{1}{C} \tag{4.12}$$

- Bevorzugt für HW und SW: Codierung Polynommultiplikation, Rückgewinnung Polynomdivision mit r-Bit Schieberegistern.



Prüfkennzeichen

Pseudo-zufällige Abbbildung großer Datenmassive auf ein r-Bit Prüfzeichen, das mit übertragen und gespeichert wird. Kontrolle durch nochmalige Bildung des Prüfkeinnzeichen für die gelesenen oder empfangenen Daten und Vergleich mit dem übertragenen Prüfzeichen. Erkennungswahrscheinlichkeit und Phantom-MF-Rate gleichfalls:

$$\mu_{\mathrm{MC}} = 1 - 2^{-r} \tag{4.10}$$

$$\zeta_{\mathrm{Phan}} = 0 \tag{1.23}$$

Bevorzugte Bildungsalgorithmen:

- arithmetische Aufsummierung zu einer Prüfsumme,
- EXOR-Summierung zu einer Prüfsumme,
- Prüfkennzeichenbildung mit LFSR: seriell (1 Bit pro Schritt), parallel (Wort pro Schritt), mit dezentraler oder zentraler Rückführung, ...

Als LFSR werden solche mit primitiver Rückführung bevorzugt, d.h. r-Bit-LFSR, die im autonomen Betrieb alle $2^r - 1$ Zustände ungleich null in einem Maximalzyklus durchlaufen.

Protokolle, Zeitüberwachung

Protokolle beschreiben für Schnittstellen zwischen IT-Systemen und oder Funktionsbausteinen:

- welche Daten wie übergeben werden,
- einzuhaltende Vorbedingungen der Daten,
- Nutzungsbedingungen der Schnittstelle,
- zu übergebende Prüfzeichen,
- weitere kontrollierbare Kriterien wie Zeitschranken.

Als Beispiele wurden die Busprotokolle: USART, CAN und Ethernet und als Beispiel für einen einfachen Signalverlauf für die Informationsübergabe »PWM« erörtert.

Zeitüberwachung: Watchdog, vom Prinzip her ein freilaufener Zähler, der mit Lebenszeichen vom System rückgesetzt wird. Bei Ausbleiben der Lebenszeichen Fehlerbehandlung (Abbruch, Neuinitialisierung, ...)

Syntaxkontrolle

Syntax-Kontrollen basieren auf formalen Sprachen. Mit den einfachen Ersetzungsregeln »darf sein«, »darf n-mal vorkommen«, ... lassen sich viele Datenformate und auch Formate für Programme beschreiben.

Ein Programm für die Spracherkennung und Datenextraktion, dass auch noch verständliche Fehlermeldungen generiert, wird zwar schnell groß und kompliziert, lässt sich aber automatisch aus der Sprachbeschreibung generieren.

Syntaxtests mit spracherkennenden Automaten werden sowohl für statische Kontrollen von Programmbeschreibungen als auch zur Kontrolle manueller Eingaben genutzt. Respektable Erkennungswahrscheinlichkeit für manuelle Programmier- und Eingabefehler.

Für maschinell erzeugte Daten, die nicht manuell zu bearbeiten sind, eignet sich die Weitergabe mit Prüfzeichen oder verschlüsselt in fehlererkennende oder korrigierende Codes besser.

Fehlertoleranz

Fehlertoleranz

(von lateinisch tolerare »erleiden, erdulden«)

Fehlertoleranz in der Technik: Aufrechterhalten der Funktion auch nach Auftreten eines Ausfalls. Ziele:

- Erhöhung der Verfügbarkeit und Lebensdauer durch Tolerierung von Ausfällen.
- Minderung der Rate der nach außen dringenden Fehlfunktionen und nicht erbringbaren Service-Leistungen durch Tolerierung von MF insbesondere durch Fehler.
- Erhöhung der Sicherheit gegenüber Datenverlusten durch redundante Datenspeicherung.
- Erhöhung der Betriebssicherheit durch einen Notbetrieb bei Ausfällen und internen MF.

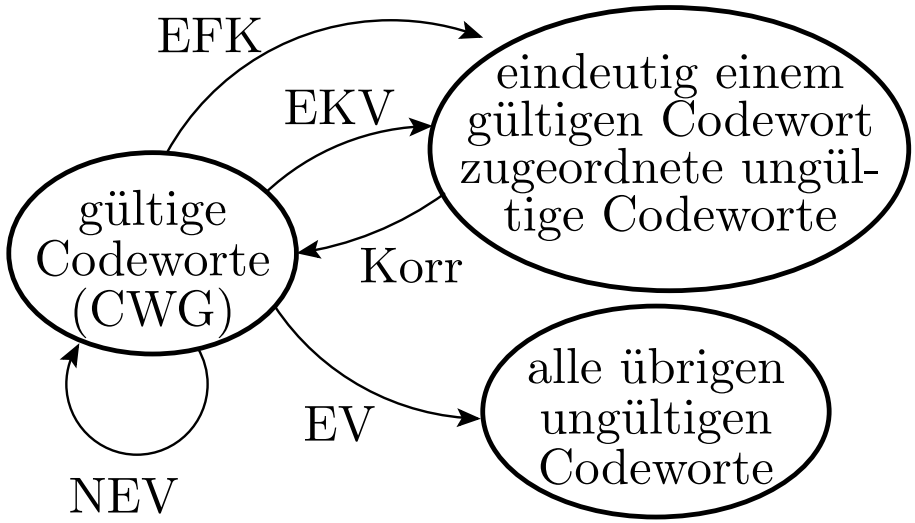
Massnahmen:

- RAID und Backup-Speicher,
- unterbrechungsfreie Stromversorgung (USV),
- Reserveeinheiten (kalt, warm, heiß, siehe Absch. 3.6.3), ...

Fehlerkorrigierende Codes

Fehlerkorrigierende Codes

EFK	erkennbar, aber falsch korrigierte Datenverfälschung
EKV	erkennbare und korrigierbare Datenverfälschung
EV	erkennbare, nicht korrigierbare Datenverfälschung
NEV	nicht erkennbare Datenverfälschung
Korr	Korrektur



Erweiterung der Menge der darstellbaren Codeworte um eine viel größere Menge korrigierbarer Codeworte und optional um unzulässige nicht korrigierbare Codeworte. Mindestbitanzahl:

$$2^{\#\mathrm{Bit}} \geq \#VCW + \#VCW \cdot \#CVC \tag{1.13}$$

Die Erkennungswahrscheinlichkeit als Anteil der übrigen ungültigen Codeworte verringert sich durch Korrekturmöglichkeiten.

$\#Bit$	Bitanzahl des Codeworts.
$\#VCW$	Anzahl der gültigen Codeworte.
$\#CVC$	Anzahl korrigierbare Codeworte je gültiges Codewort.

Beispiel: Korrektur von Einzelbitfehler

Anzahl korrigierbare Codeworte je gültiges Codewort gleich Bitanzahl:

$$\#CVC = \#\text{Bit}$$

Mindestbitanzahl:

$$2^{\#\text{Bit}} \geq \#VCW + \#\text{Bit} \cdot \#VCW = (\#\text{Bit} + 1) \cdot \#VCW$$

Für $\#\text{VCW} = 256$ gültige Codeworte:

$$\begin{aligned} 2^{\#\text{Bit}} &\geq 256 \cdot (1 + \#\text{Bit}) \\ \#\text{Bit} &\geq 12 \end{aligned}$$

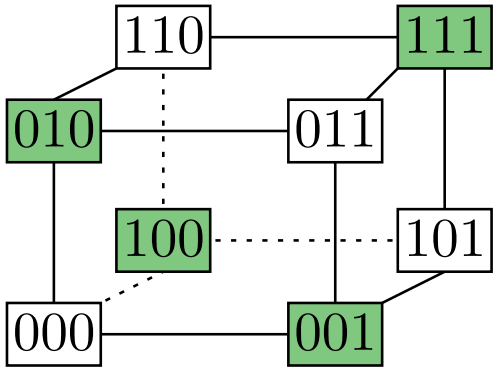
Probe:

$$2^{12} > 2^8 \cdot (1 + 12)$$

$\#Bit$	Bitanzahl des Codeworts.
$\#VCW$	Anzahl der gültigen Codeworte.
$\#CVC$	Anzahl korrigierbare Codeworte je gültiges Codewort.

Hamming-Codes

Hamming Codes werden durch die Hamming-Distanz $\#HD$ beschrieben. Das ist die Anzahl der Bitpositionen, in denen sich zwei Codeworte mindestens unterscheiden. Distanz von 2 oder mehr garantiert, dass eine 1-Bit Verfälschung nicht zu einem anderen gültigen Codewort führt.



- Erkennen alle Codeworte mit bis zu $\#DB$ verfälschten Bits:

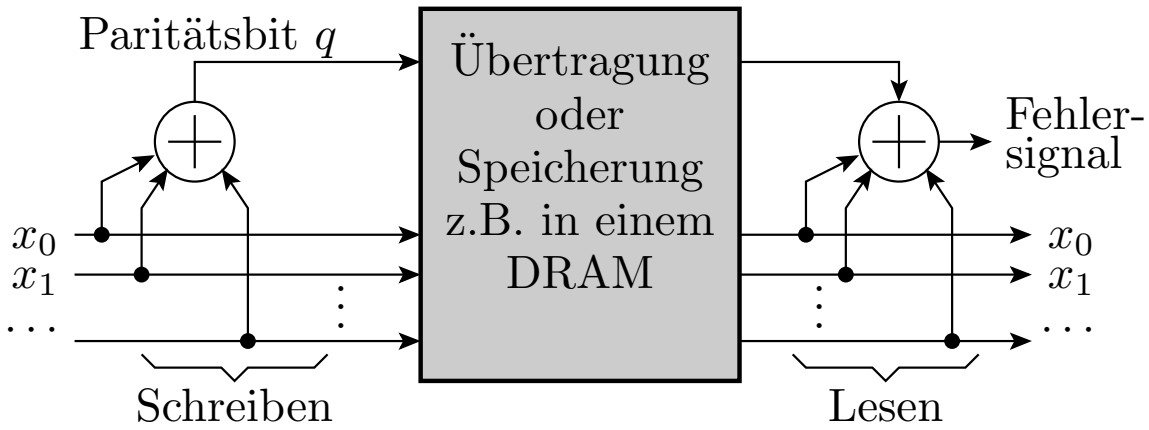
$$\#HD \geq \#DB + 1 \tag{1.14}$$

- Korrektur alle Codeworte mit bis zu $\#CB$ verfälschten Bits:

$$\#HD \geq 2 \cdot \#CB + 1 \tag{1.15}$$

$\#HD$ Hamming-Distanz.

Paritätsbit als Prüfkennzeichen ($\#HD = 2$)



Paritätsbit q gleich modulo-2 Summe (EXOR) der Datenbits:

$$q = x_{n-1} \oplus x_{n-2} \oplus \ldots \oplus x_1 \oplus x_0$$

bei gerader Anzahl von Einsen »0« sonst »1«.

Erkennt ungeradzahlige und insgesamt 50% der Bitverfälschungen.
Bei pseudozufälliger gleichmäßiger Abbildung nach

$$\mu_{\mathrm{MC}} = 1 - 2^{-r} \tag{4.10}$$

Die zu erwartende MF-Überdeckung für $r = 1$ Prüfbit ist $\mu_{\mathrm{MC}} = 50\%$.

μ_{MC} Zu erwartende Fehlfunktionsüberdeckung.

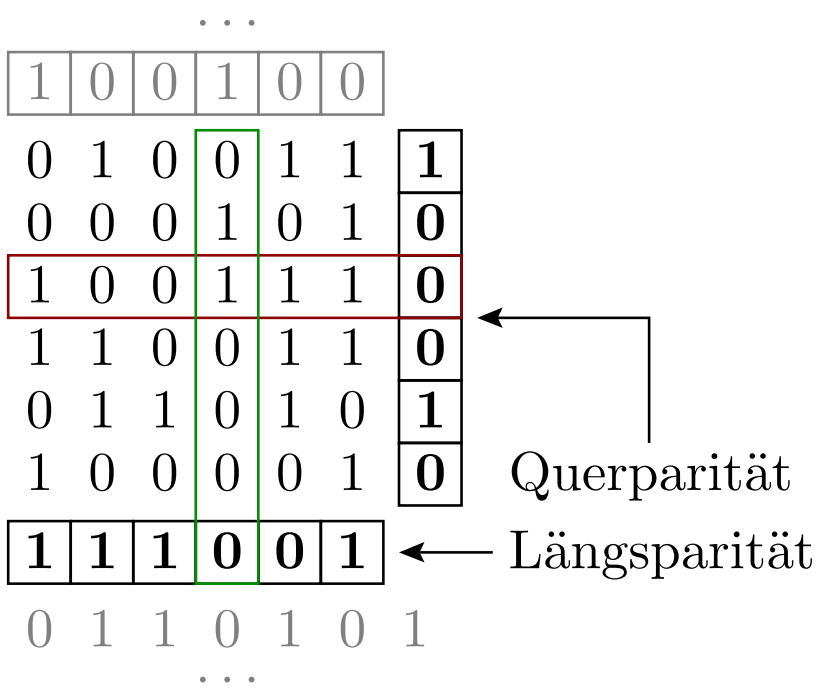
- Beim Lesen gespeicherter und Empfang übertragender Daten unter Ausschluss von Verfälschungen mehrerer Bits im selben Datenwort durch dieselbe Ursache

$$\mu_{\mathrm{MC}} = \frac{(1-\lambda)\cdot\lambda}{1-(1-\lambda)} = 1-\lambda \tag{4.11}$$

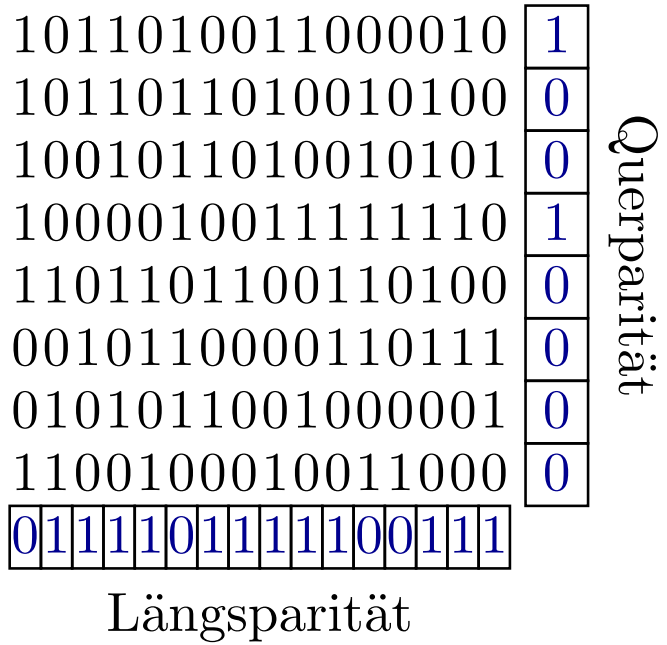
μ_{MC}	Zu erwartende Fehlfunktionsüberdeckung.
λ	Parameter der Poissonverteilung, hier zu erwartende Bitfehleranzahl je Datenwort.

Kreuzparität (Fehlerkorrigierender Paritätscode)

Daten sind in einem 2-dimensionalen Array organisiert. Paritätsbildung für alle Zeilen und Spalten. Erlaubt Lokalisierung und Korrektur von 1-Bit Fehlern. Einsatz in redundanten Festplatten-Arrays (RAID 3 und RAID 5).



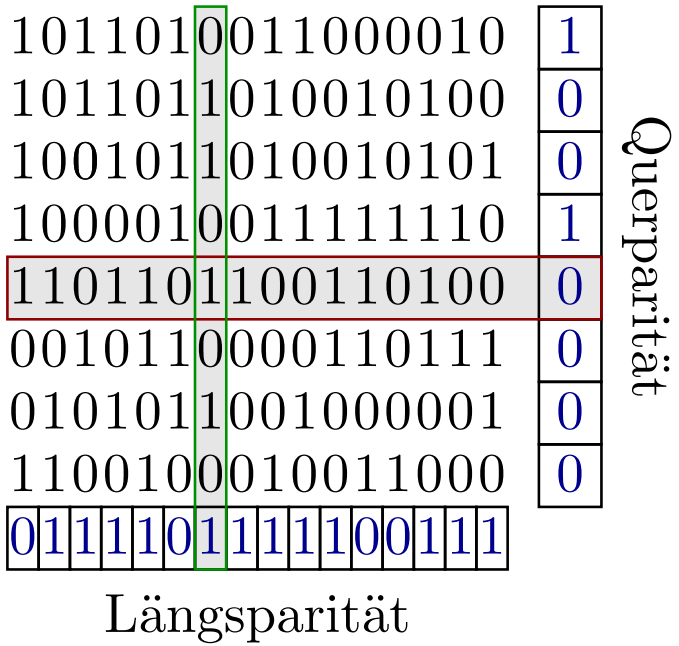
Beispiel 4.4: Kreuzparität



a) *Kontrolle der Zeilen- und Spaltenparität?*

b) *Liegt keine, eine erkennbare nicht korrigierbare oder eine erkenn- und korrigierbare Verfälschung vor?*

a) *Kontrolle der Zeilen- und Spaltenparität?*

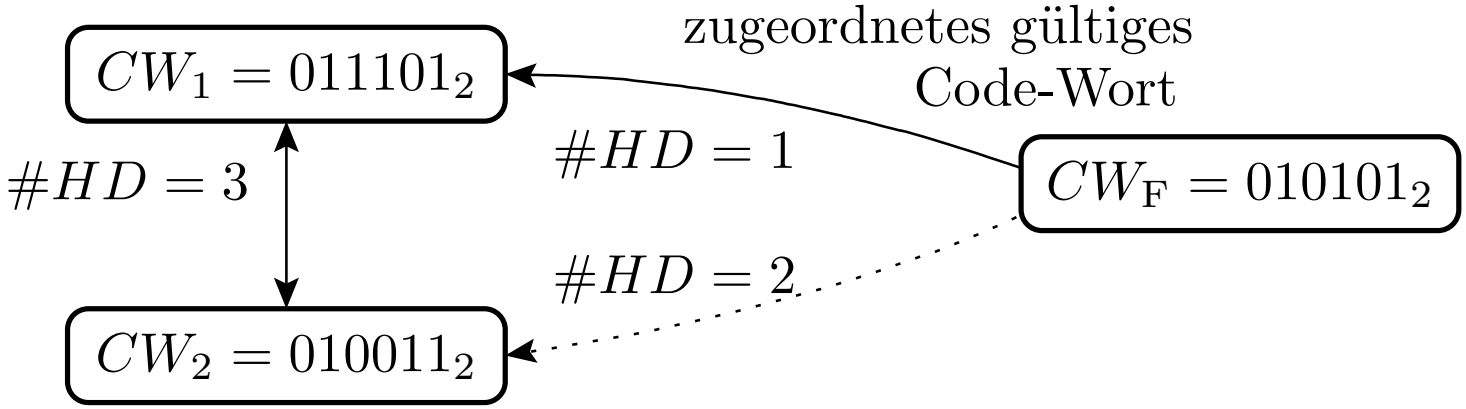


b) *Liegt keine, eine erkennbare nicht korrigierbare oder eine erkenn- und korrigierbare Verfälschung vor?*

Korrigierbar durch Invertierung des Bits in Zeile 5, Spalte 7 (null setzen).

1-Bit fehlerkorrigierende Hamming-Codes

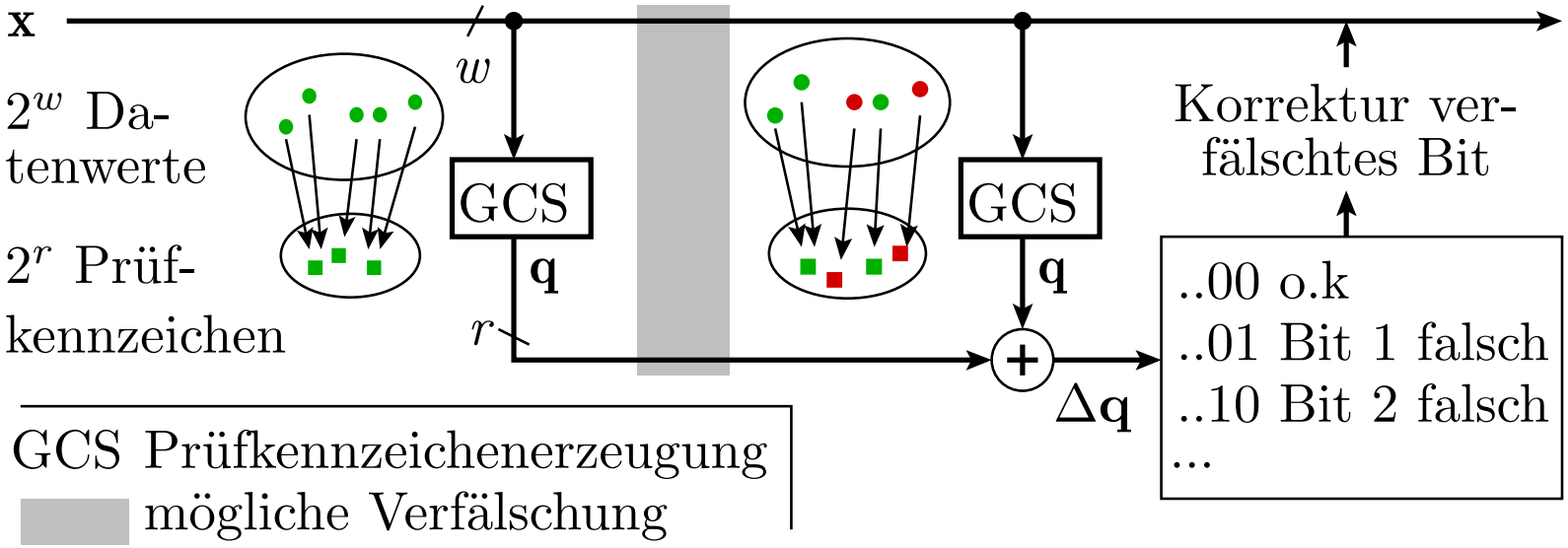
Ab einem Hamming-Abstand $\#HD \geq 3$ ist jede 1-Bit-Verfälschung eindeutig einem gültigen Codewort zugeordnet.



Korrektur durch Ersatz des verfälschten Codeworts durch das mit Hamming-Distanz $\#HD = 1$. Bei Hamming-Distanz $\#HD = 3$ werden Codeworte mit zwei oder mehr verfälschten Bits falschen gültigen Codeworten zugeordnet.

$\#HD$	Hamming-Distanz.
CW_i	Code-Wort i.
CW_F	verfälschtes Code-Wort.

Konstruktion 1-Bit fehlerkorrigierender Code



Erzeugung des r-Bit Prüfkennzeichens **q** so aus dem w-Bit Datenwort **x**,: dass die mod-2 Summen des übertragenen und des danach gebildeten Prüfkennzeichens die Nummer des verfälschten Bits ist

verfälschtes Bit	1	2	3	4	5	...
Prüfzeichendifferenz $\Delta\mathbf{q}$	..001	..010	..011	..100	..101	...

q r-Bit Prüfkennzeichen.
$\Delta\mathbf{q}$ EXOR-Differenz der erhaltenen und der neu berechneten Prüfbits.

Prüfzeichenbildung für 1 Byte Einzelbitkorrektur

Anzahl der Prüfbits r nach Gl. 4.13 für $w = 8$:

$$2^{w+r} \geq \left(1 + \underbrace{w + r}_{\text{Anz. Einzelbitfehler}}\right) \cdot \underbrace{2^w}_{\#VCW} = (1 + 12) \cdot 2^8;\ r = 4$$

verfälschtes Bit	1	2	3	4	5	...
Prüfzeichendifferenz $\Delta\mathbf{q}$	0001	0010	0011	0100	0101	...

$$\begin{aligned}
\Delta q_0 &= \underline{b_1} \oplus b_3 \oplus b_5 \oplus b_7 \oplus b_9 \oplus b_{11} \\
\Delta q_1 &= \underline{b_2} \oplus b_3 \oplus b_6 \oplus b_7 \oplus b_{10} \oplus b_{11} \\
\Delta q_2 &= \underline{b_4} \oplus b_5 \oplus b_6 \oplus b_7 \oplus b_{12} \\
\Delta q_3 &= \underline{b_8} \oplus b_9 \oplus b_{10} \oplus b_{11} \oplus b_{12}
\end{aligned}$$

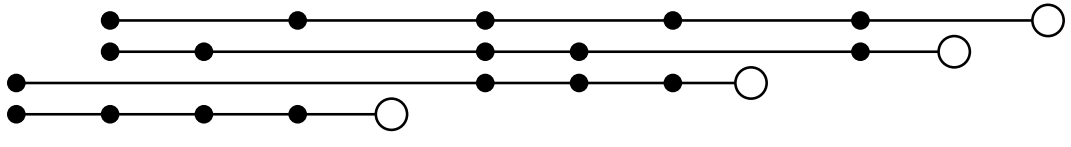
Die unterstrichenen Bits jeder Prüfsumme seinen die Prüfbits q_i.

w	Anzahl der Datenbits.
r	Anzahl der Prüfbits.
b_i	Bit i des Gesamtcodeworts.
Δq_i	Prüfkennzeichendifferenz Bit i.

Bitzuordnung, Prüfbit-Berechnung

Bitnummer	12	11	10	9	8	7	6	5	4	3	2	1
Zuordnung	x_7	x_6	x_5	x_4	q_3	x_3	x_2	x_1	q_2	x_0	q_1	q_0

○ Prüfbit
• Datenbit



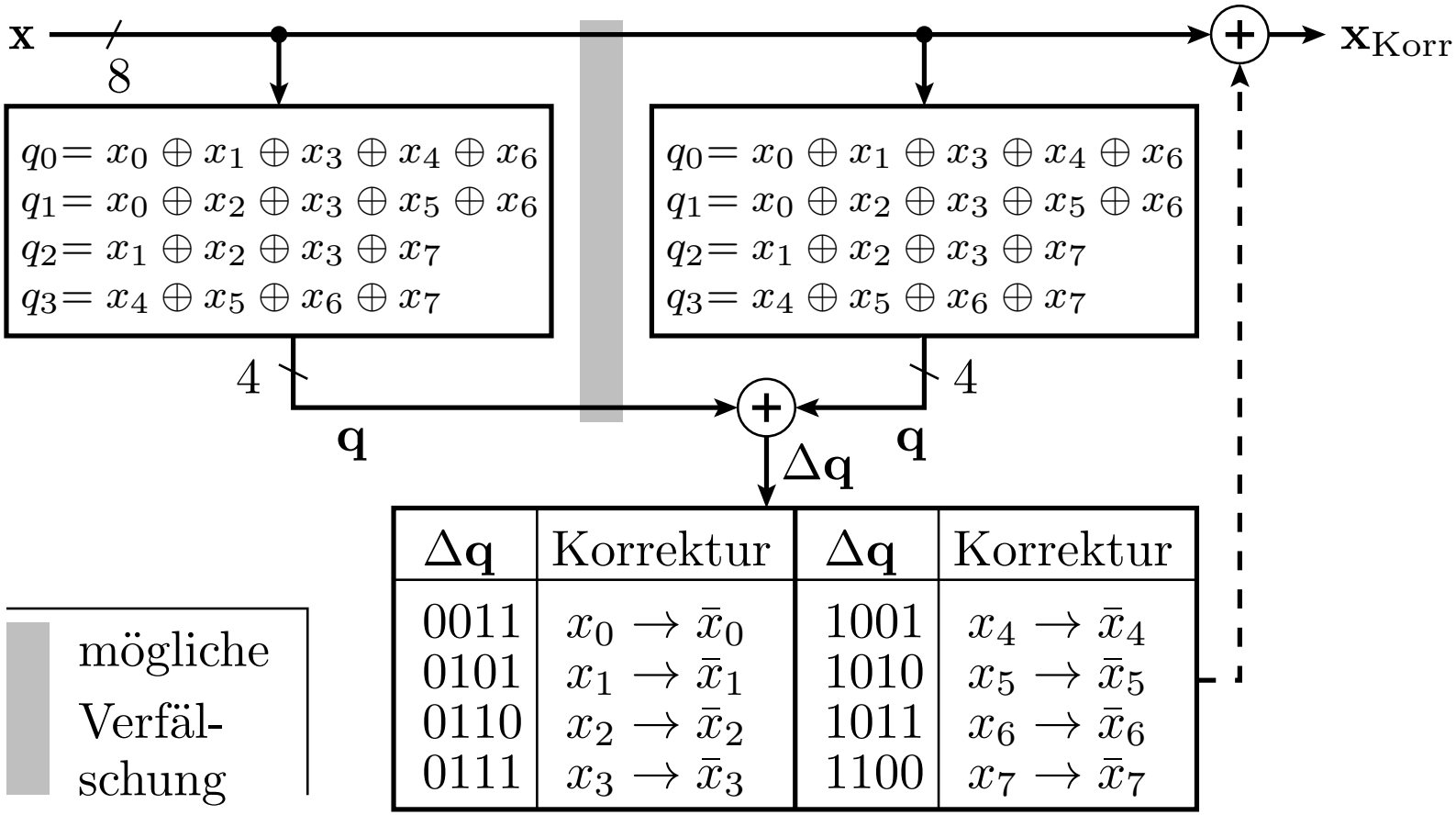
$$\begin{aligned}
\Delta q_0 &= b_1 \oplus b_3 \oplus b_5 \oplus b_7 \oplus b_9 \oplus b_{11} &&= q_0 \oplus x_0 \oplus x_1 \oplus x_3 \oplus x_4 \oplus x_6 \\
\Delta q_1 &= b_2 \oplus b_3 \oplus b_6 \oplus b_7 \oplus b_{10} \oplus b_{11} &&= q_1 \oplus x_0 \oplus x_2 \oplus x_3 \oplus x_5 \oplus x_6 \\
\Delta q_2 &= b_4 \oplus b_5 \oplus b_6 \oplus b_7 \oplus b_{12} &&= q_2 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_7 \\
\Delta q_3 &= b_8 \oplus b_9 \oplus b_{10} \oplus b_{11} \oplus b_{12} &&= q_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7
\end{aligned}$$

Auflösung nach q_i für $\Delta\mathbf{q} = 0$:

$$\begin{aligned}
q_0 &= x_0 \oplus x_1 \oplus x_3 \oplus x_4 \oplus x_6 \\
q_1 &= x_0 \oplus x_2 \oplus x_3 \oplus x_5 \oplus x_6 \\
q_2 &= x_1 \oplus x_2 \oplus x_3 \oplus x_7 \\
q_3 &= x_4 \oplus x_5 \oplus x_6 \oplus x_7
\end{aligned}$$

q_i	Prüfbit i.
x_i	Datenbit i.

Codier-, Erkennungs- und Korrekturschaltung



Δq	Korrektur	Δq	Korrektur
0011	$x_0 \to \bar{x}_0$	1001	$x_4 \to \bar{x}_4$
0101	$x_1 \to \bar{x}_1$	1010	$x_5 \to \bar{x}_5$
0110	$x_2 \to \bar{x}_2$	1011	$x_6 \to \bar{x}_6$
0111	$x_3 \to \bar{x}_3$	1100	$x_7 \to \bar{x}_7$

- Gleiches Prüfbit-Bildung vor und nach Speichern/ Übertragung.
- Differenzbildung durch bitweises EXOR.
- Wenn $\Delta\mathbf{q} \neq 0$ und Datenbit verfälscht, Invertieren.

Beispiel 4.5: Fehlerkorrigierender Code

b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1
x_7	x_6	x_5	x_4	q_3	x_3	x_2	x_1	q_2	x_0	q_1	q_0

$$q_0 = x_0 \oplus x_1 \oplus x_3 \oplus x_4 \oplus x_6 \qquad q_2 = x_1 \oplus x_2 \oplus x_3 \oplus x_7$$
$$q_1 = x_0 \oplus x_2 \oplus x_3 \oplus x_5 \oplus x_6 \qquad q_3 = x_4 \oplus x_5 \oplus x_6 \oplus x_7$$

a) *Bestimmung des Codeworts* $\mathbf{b}$ *für den Datenwert* $\mathbf{x} = 0x8B$.

Bitnummer	12	11	10	9	8	7	6	5	4	3	2	1	
Zuordnung	x_7	x_6	x_5	x_4	q_3	x_3	x_2	x_1	q_2	x_0	q_1	q_0	
Kontrollbits	=	=	=	=	=	=	=	=	=	=	=	=	
$\mathbf{x} = 0x8B$													$\mathbf{b} =$
$\mathbf{b} = 0xA9B$													$\Delta\mathbf{q} =$
korrigiert													$\mathbf{x} =$

b) *Extrahieren des Werts* $\mathbf{x}$ *aus dem Codewort* $\mathbf{b} = 0xA9B$*?*

a) *Bestimmung des Codeworts* $\mathbf{b}$ *für den Datenwert* $\mathbf{x} = 0x8B$.

Bitnummer	12	11	10	9	8	7	6	5	4	3	2	1	
Zuordnung	x_7	x_6	x_5	x_4	q_3	x_3	x_2	x_1	q_2	x_0	q_1	q_0	
Kontrollbits													
$\mathbf{x} = 0x8B$	1	0	0	0	1	1	0	1	1	1	0	1	$\mathbf{b} = 0x8DD$

Der Wert 0x8B hat das Codewort 0x8DD.

b) *Extrahieren des Werts* $\mathbf{x}$ *aus dem Codewort* $\mathbf{b} = 0xA9B$*?*

Bitnummer	12	11	10	9	8	7	6	5	4	3	2	1	
Zuordnung	x_7	x_6	x_5	x_4	q_3	x_3	x_2	x_1	q_2	x_0	q_1	q_0	
Kontrollbits	=	=	–	–		=	=	=		=	–	–	
$\mathbf{b} = 0$xA9B	1	0	1	0	1	0	0	1	1	0	1	1	$\Delta\mathbf{q} = 12_{10}$
korrigiert	0	0	1	0		0	0	1		0			$\mathbf{x} = 0$x22

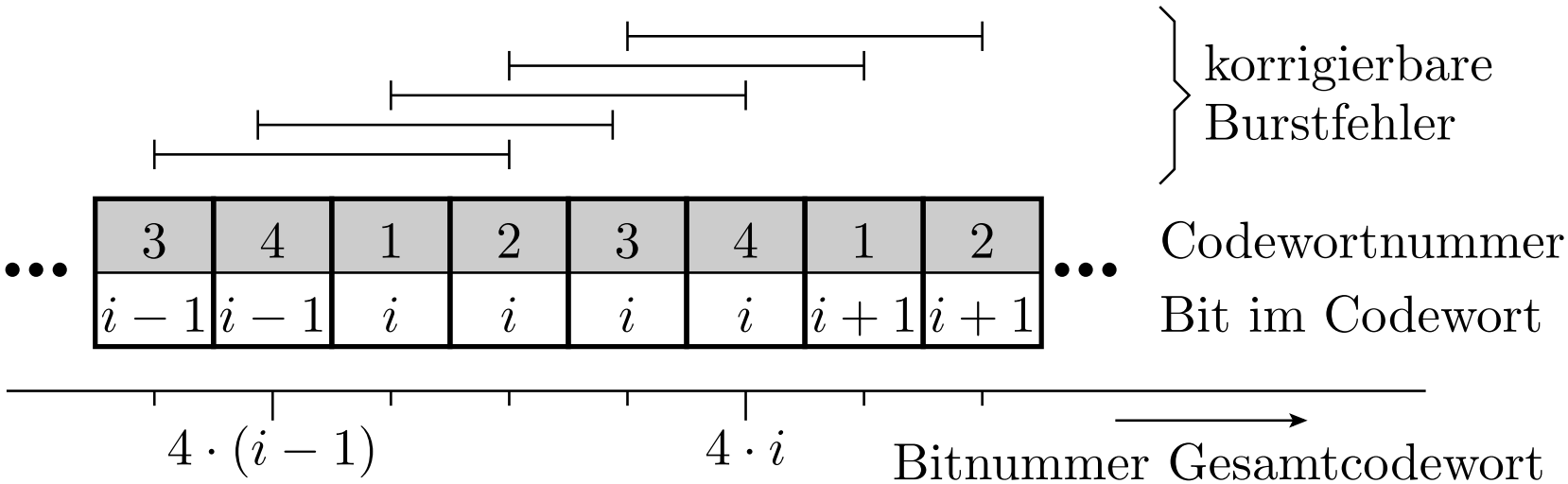
Im Codewort 0x49B steckt der Wert 0xA2 mit $\Delta\mathbf{q} = 0\text{b}1100 = 12$. Das verfälschte Bit 12 im Codewort ist Datenbit x_7. Invertierung von x_7 ergibt den Datenwert 0x22.

Korrigierbare Burstfehler durch Verschränkung

Burstfehler

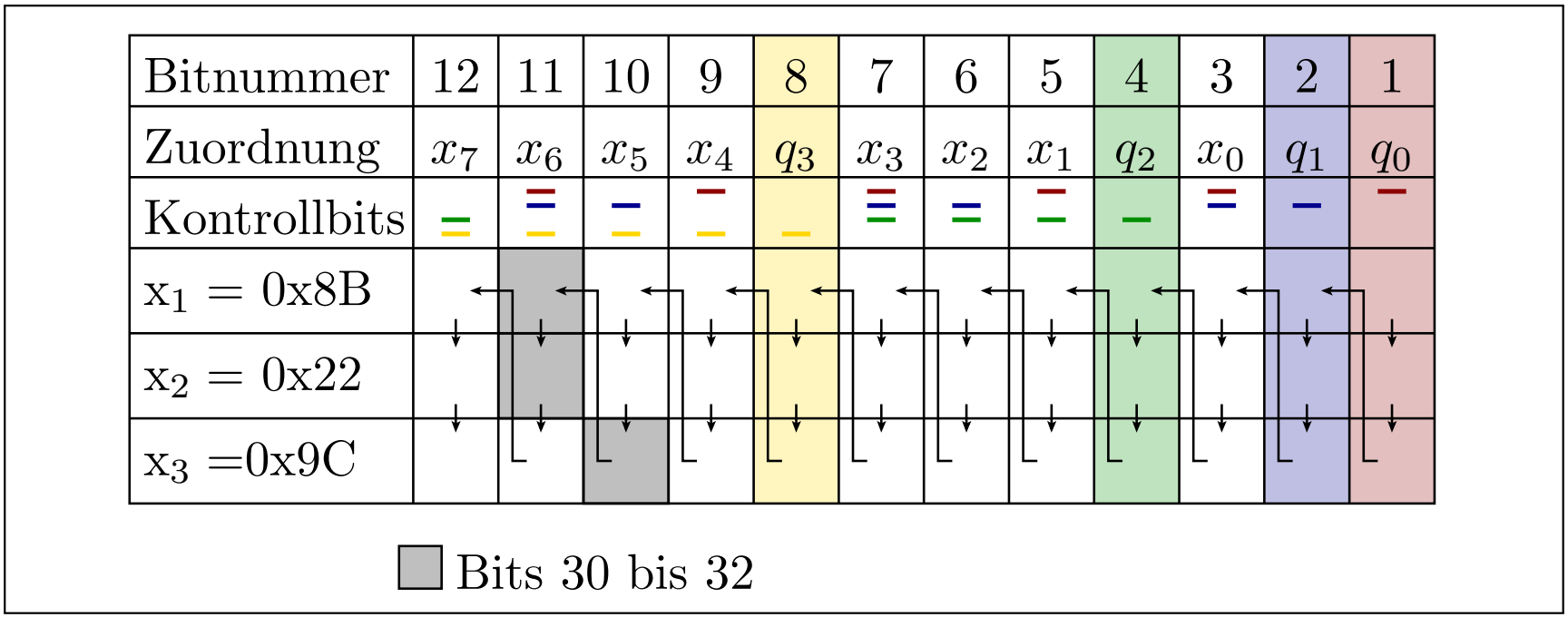
Verfälschung von bis zu m aufeinanderfolgende Bits. Typisch für Lesefehler von CDs und Übertragungsfehler.

Zusammensetzen eines fehlerkorrigierenden Codes für m-Bit Burstfehler für eine $m \cdot n$ Bit lange Folgen aus m fehlerkorrigierenden Codeworten für 1-Bit-Fehler für n Bit lange Folgen durch Verschränkung.



Beispiel 4.6: Burstfehler

a) *Codierung Datenfolge 0x8B, 0x22, 0x9C so, dass bis zu 3-Bit lange Burstfehler korrigierbar sind, durch Verschränkung von je drei aufeinanderfolgenden H8-12-Codeworten?*



b) *Zeigen Sie, dass eine Invertierung der Bits 30 bis 32 korrigiert wird?*

a) *Codierung Datenfolge 0x8B, 0x22, 0x9C so, dass bis zu 3-Bit lange Burstfehler korrigierbar sind, durch Verschränkung von je drei aufeinanderfolgenden H8-12-Codeworten?*

Bitnummer	12	11	10	9	8	7	6	5	4	3	2	1
Zuordnung	x_7	x_6	x_5	x_4	q_3	x_3	x_2	x_1	q_2	x_0	q_1	q_0
Kontrollbits	=	=	=	=	=	=	=	=	=	=	=	=
x_1 = 0x8B	1	0	0	0	1	1	0	1	1	1	0	1
x_2 = 0x22	0	0	1	0	1	0	0	1	1	0	1	1
x_3 =0x9C	1	0	0	1	0	1	1	0	1	0	0	0

b) *Zeigen Sie, dass eine Invertierung der Bits 30 bis 32 korrigiert wird?*

Bitnummer	12	11	10	9	8	7	6	5	4	3	2	1	
Zuordnung	x_7	x_6	x_5	x_4	q_3	x_3	x_2	x_1	q_2	x_0	q_1	q_0	
Kontrollbits													
$x_1 = 0x8B$	1	0	0	0	1	1	0	1	1	1	0	1	$\Delta\mathbf{q} = 1011_2$
$x_2 = 0x22$	0	0	1	0	1	0	0	1	1	0	1	1	$\Delta\mathbf{q} = 1011_2$
$x_3 = 0x9C$	1	0	0	1	0	1	1	0	1	0	0	0	$\Delta\mathbf{q} = 1010_2$

Durch Burstfehler invertierte Bits 30 bis 32,

RAID und Backup

RAID 0 und 1

Organisation mehrerer physischer Massenspeicher (Festplatten oder SSD) zu einem logischen Laufwerk. Mit RAID können Systeme Ausfälle einer oder mehrerer Festplatten oder SSD ohne Datenverlust und in vielen Fällen sogar ohne Ausfallzeiten überstehen.

RAID 0: Verteilung der Daten blockweise auf mehrere Festplatte so, dass sich der Datendurchsatz erhöht. Die Speicherkapazitäten summieren sich, aber keine Tolerierung von Plattenausfällen.

RAID 1: Zwei gespiegelte Festplatten. Die Daten werden versetzt geschrieben, so dass das Schreiben etwas länger dauert, aber mit nahe doppelter Geschwindigkeit gelesen werden kann. Bei Ausfall einer Platte existieren alle Daten noch auf der zweiten Festplatte. Die Lesegeschwindigkeit reduziert sich, aber das System bleibt funktionsfähig.

RAID Redundantes Array unabhängiger Festplatten.
SSD Solid State Drive, Festplattennachbildung mit Halbleiterspeichern.

RAID 0+1: Verschaltung der Hälfte der Festplatten zu einem RAID 0 und Spiegelung auf die andere Hälfte. Tolerierung von Plattenschäden in einem der beiden RAID 0, solange das andere RAID 0 intakt ist.

RAID 10: Verschaltung paarweise gespiegelter Platten zu einem RAID 0. Tolerierung von einem Plattenschaden je gespiegeltes Plattenpaar.

RAID mit Paritätsblöcken (RAID 2 bis 7)

	Paritätsbildung	Ausfall Disc 2	Ausfall Disc 4
Disc 1: Daten	0001 0011 0010	0001 0011 0010	0001 0011 0010
Disc 2: Daten	1101 0101 0000	xxxx xxxx xxxx	1101 0101 0000
Disc 3: Daten	1101 1111 1100	1101 1111 1100	1101 1111 1100
Disc 4: Parität	**0001 1001 1110**	0001 1001 1110	xxxx xxxx xxxx
	Wiederherstellung:	**1101 0101 0000**	**0001 1001 1110**

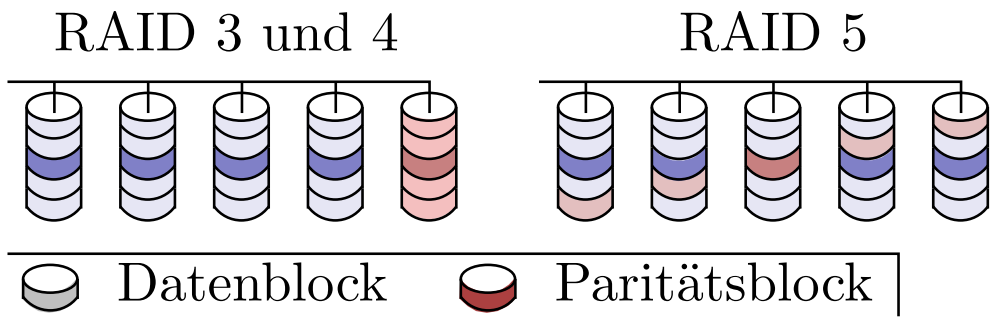
Datenverteilung wie bei RAID 0 blockweise über mehrere Platten.
Bildung und Speicherung von Paritätsblöcken so, dass

- nur Blöcke unterscheidlicher Platten zusammengefasst werden,
- der Paritätsblock auf noch einer anderen Platte gespeichert wird.

Nach Ausfall einer Platte lassen sich alle Blöcke auf der ausgefallenen Platte aus denen auf anderen Platten gespeicherten Blöcken durch bitweise EXOR rekonstruieren, vorausgesetzt es ist bekannt

- welche Platte ausgefallen ist und
- und wo die zugehörigen ver-EXOR-ten Blöcke stehen.

Verteilung der Paritätsblöcke auf Platten



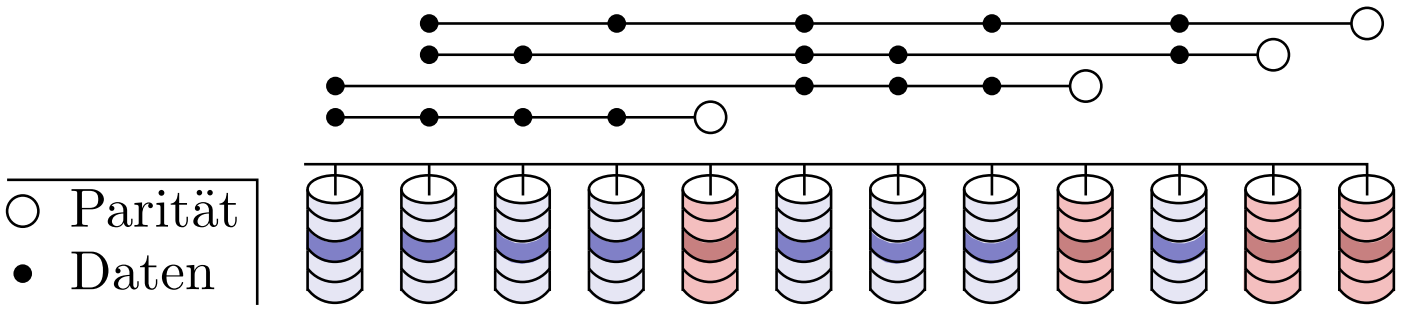
Schreiben eines Blocks:

- Block lesen, Paritätsänderung bestimmen, schreiben,
- Paritätsblock lesen, ändern, schreiben.

Wenn die Paritätsblöcke alle auf derselben Festplatte stehen (RAID 3 und 4), viel mehr Schreib-Lese-Zugriffe auf diese. Verkürzte Lebensdauer. Alternative ist gleichmäßige Verteilung der Paritätsblock auf alle Platten (RAID 5).

Erweiterte Fehlertoleranz

Festplatte	12	11	10	9	8	7	6	5	4	3	2	1
	x_7	x_6	x_5	x_4	q_3	x_3	x_2	x_1	q_2	x_0	q_1	q_0



Statt einem Paritätsblock, mehrere nach dem Prinzip der 1-Bit-Fehlerkorrektur. Toleriert nicht nur komplette Plattenausfälle, sondern auch einzelne Bitverfälschungen (RAID 2).

Weitere Ansätze:

- Zusätzliche Längsparitätsblöcke zur Wiederherstellung einzelner verfälschter Blöcke nicht komplett ausgefallener Platten nach dem Prinzip der Kreuzparität.
- Fehlertoleranz für mehr als einen zeitgleichen Plattenausfall, ...

Wiederherstellung, weitere Schutzmaßnahmen

Ersatz der ausgefallenen Platte vor dem nächsten Ausfall:

- Laufwerkwechsel in der Regel im ausgeschalteten Zustand. Rebuilt (Rekonstruktion der verlorenen Daten) bei Systemstart.
- Hot-Fix: Reserveplatte, die bei Ausfall für die ausgefallene Platte in das RAID eingebunden wird.
- Hot-Swap: Plattenaustausch und Rebuilt im Betrieb.

Neben HW-Ausfällen gibt es weitere Ursachen für den Verlust schwer wiederzubeschaffende Daten. Auftrittshäufigkeiten:

- 59% Hardwareprobleme
- 26% Anwenderfehler
- 9% Softwarefehler
- 2% Schadware (Vieren, ...)
- 2% Naturkatstrophen
- 2% sonstiges.

Selbst ausgefeilte RAIDs tolerieren nur HW-Probleme. Den einzig wirklich zuverlässigen Schutz gegen Datenverluste bieteten konsequent geplante und durchgeführte Backups.

Redundanz

Redundanz

(von lateinisch redundare, überlaufen, sich reichlich ergießen).

In der Technik sind Redundanzen zusätzliche Ressourcen, die im fehler- und störungsfreien Betrieb nicht benötigt werden:

- Motoren, Baugruppen, komplette Geräte, Steuerleitungen,
- Stromversorgung, Leistungsreserve, ...

zur Erhöhung der Verfügbarkeit, Zuverlässigkeit und/oder Sicherheit.

Arten von Redundanz:

- Standby-Redundanz: Service-Übernahme nach Ausfällen z.B. durch eine USV nach Stromausfall.
- räumlich verteilte Redundanz: Service-Übernahme nach örtlich begrenzten Zerstörungen z.B. eines zerstörten Rechenzentrums.
- diversitäte Redundanz: Service-Übernahme bei Fehlfunktionen durch Fehler.

Bei Standby-Redundanz wird zwischen heißer und kalter Redundanz unterschieden (siehe Abschn. 3.6.3 *Reserveeinheiten*)

KooN (k out of n) Systeme

Systeme aus n gleichartigen Komponenten (Rechnerknoten, Servern, Verbindungen, ...), die insgesamt verfügbar sind, solange k der n Komponenten verfügbar sind. Die Anzahl der verfügbaren Komponenten ist binomial-verteilt:

$$\mathbb{P}[X = k] = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k} \tag{3.33}$$

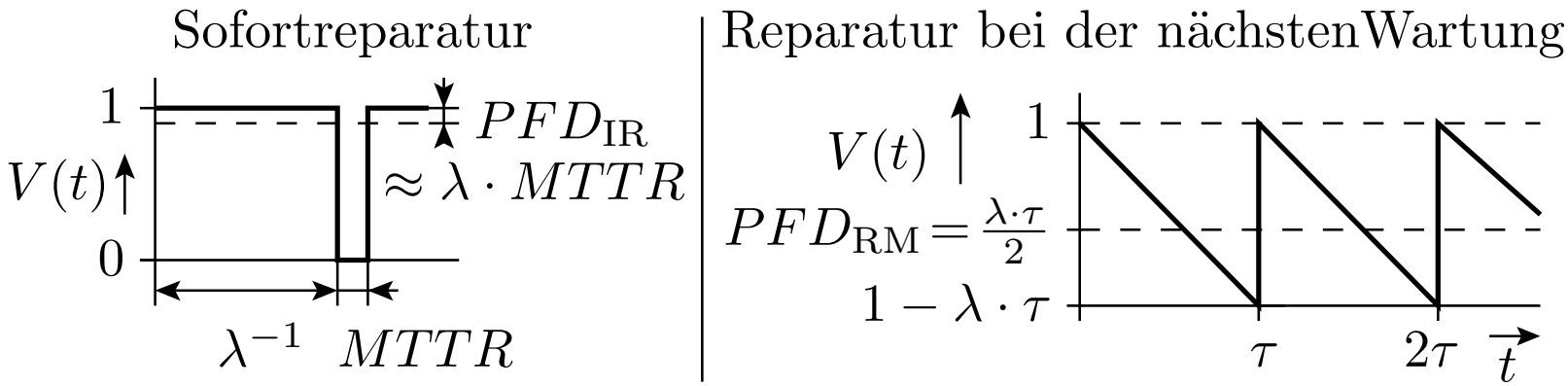
Bei Ausfall von weniger als $n - k$ redundanten Komponenten möglicher Leistungsabfall:

- Aufgabenumverteilung auf andere Systembestandteile.
- Bei Notstromversorgung Abschalten von Systemteilen.
- Transmission Control Protocol (TCP): auch dann noch eine sichere Punkt-zu-Punkt-Verbindung, wenn einzelne Knoten im Netzwerk überlastet, falsch eingestellt sind oder Daten verfälschen.
- Bis zum Ausfall der letzten Einheit ist auch auch bei $k > 1$ oft noch ein Notbetrieb möglich.

n Anzahl der identischen Teilsysteme.
k Anzahl der Teilsysteme, die verfügbar sein müssen von den n vorhandenen.
p Wahrscheinlichkeit der Verfügbarkeit jedes einzelnen Teilsystems auf Anforderung.

Verfügbarkeit 1oo1 mit Wartung

Gewartetet Systeme ohne Redundanz (siehe Abschn. 3.6.4 *Wartung*).



Mittlere Wahrscheinlichkeit der Nichtverfügbarkeit:

$$PFD = \eta_{\mathrm{IR}} \cdot \lambda \cdot MTTR + (1 - \eta_{\mathrm{IR}}) \cdot \frac{\lambda \cdot \tau}{2} \tag{3.105}$$

$V\,(t)$	Überlebenswahrscheinlichkeit als Funktion der Lebensdauer.
PFD	Wahrscheinlichkeit der Nichtverfügbarkeit bei Anforderung.
η_{IR}	Anteil der Ausfälle, die durch Sofortreparatur beseitigt werden.
λ	Ausfallrate (Failure rate).
$MTTR$	Mittlere Reparaturzeit (Mean time to repair).
τ	Wartungsintervall.

Beispiel 4.7: PFD und Verfügbarkeit eines Autos

Fahrdauer innerhalb eines Wartungsintervalls $\frac{10.000\text{km}}{50\,\text{km/h}} = 200\text{h}$, Ausfallrate $\lambda = 10^{-3}\,\text{h}^{-1}$, Anteil der sofort bemerkten und beseitigten Ausfälle $\eta_{\text{IR}} = 80\%$, mittlere Reparaturzeit $MTTR = 2\,\text{h}$.

Wartungsintervall $\tau = 200\text{h}$, $\lambda = 10^{-3}\,\text{h}^{-1}$, $\eta_{\text{IR}} = 80\%$, $MTTR = 2\,\text{h}$

a) *Wahrscheinlichkeit der Nichtverfügbarkeit?*

b) *Verfügbarkeit?*

τ	Wartungsintervall.
λ	Ausfallrate (Failure rate).
η_{IR}	Anteil der Ausfälle, die durch Sofortreparatur beseitigt werden.
$MTTR$	Mittlere Reparaturzeit (Mean time to repair).

Wartungsintervall $\tau = 200\text{h}$, $\lambda = 10^{-3}\,\text{h}^{-1}$, $\eta_{\text{IR}} = 80\%$, $MTTR = 2\,\text{h}$

a) *Wahrscheinlichkeit der Nichtverfügbarkeit?*

$$PFD = \eta_{\text{IR}} \cdot \lambda \cdot MTTR + (1 - \eta_{\text{IR}}) \cdot \frac{\lambda \cdot \tau}{2} \tag{3.105}$$

$$PFD = 0{,}8 \cdot 10^{-3}\,\text{h}^{-1} \cdot 2\,\text{h} + \frac{0{,}2 \cdot 10^{-3}\,\text{h}^{-1} \cdot 200\text{h}}{2} = 2{,}16\%$$

Wartungsintervall $\tau = 200\text{h}$, $\lambda = 10^{-3}\,\text{h}^{-1}$, $\eta_{\text{IR}} = 80\%$, $MTTR = 2\,\text{h}$

b) *Verfügbarkeit?*

$$A = 1 - PFD \tag{1.2}$$

$$A = 97{,}84\%$$

Verfügbarkeit von 1oo2 Systemen

Wahrscheinlichkeit, dass mindestens eines von zwei unabhängig ausfallenden identischen Teilsystemen mit Überlebenswahrscheinlichkeit

$$V(t) = \mathrm{e}^{-\lambda \cdot t} \tag{3.97}$$

überlebt:

$$V_{1\mathrm{oo}2.\mathrm{IF}}(t) = 1 - \left(1 - \mathrm{e}^{\lambda t}\right)^2 = 2 \cdot \mathrm{e}^{\lambda t} - \mathrm{e}^{2\lambda t}$$

Wenn Reparatur erst bei der Wartung, ist die Verfügbarkeit die mittlere Überlebenswahrscheinlichkeit in einem Wartungsintervall τ:

$$\begin{aligned} A_{1\mathrm{oo}2.\mathrm{IFRM}} &= \frac{1}{\tau} \int_0^{\tau} \left(2 \cdot \mathrm{e}^{\lambda t} - \mathrm{e}^{2\lambda t}\right) \cdot \mathrm{d}\tau \\ &= \frac{2}{\lambda\tau} \cdot \left(1 - \mathrm{e}^{\lambda\tau}\right) - \frac{1}{2\lambda\tau} \cdot \left(1 - \mathrm{e}^{2\lambda\tau}\right) \end{aligned}$$

$V(t)$	Überlebenswahrscheinlichkeit eines einzelnen Teilsystems als Funktion der Lebensdauer.
$V_{1\mathrm{oo}2.\mathrm{IF}}$	Überlebensw. von mindestens einem Teilsystem bei unabhängiger Ausfallursache.
$A_{1\mathrm{oo}2}$	Verfügbarkeit von mindestens einem der beiden Teilsysteme.
$*_{*.\mathrm{IFRM}}$	Unabhängige Ausfallursache, Reparatur erst bei nächster Wartung.

$$A_{1oo2.IFRM} = \frac{2}{\lambda\tau} \cdot \left(1 - e^{\lambda\tau}\right) - \frac{1}{2\lambda\tau} \cdot \left(1 - e^{2\lambda\tau}\right)$$

Mit der Näherung $e^{-x} = 1 - x + \frac{x^2}{2} - \frac{x^3}{6}$

$$\frac{2}{\lambda\tau} \cdot \left(1 - e^{\lambda\tau}\right) = \frac{2}{\lambda\tau} \cdot \left(1 - \left(1 - \lambda\tau + \frac{(\lambda\tau)^2}{2} - \frac{(\lambda\tau)^3}{6}\right)\right) = 2 - \lambda\tau + \frac{(\lambda\tau)^2}{3}$$

$$\frac{1}{2\lambda\tau} \cdot \left(1 - e^{2\lambda\tau}\right) = \frac{1}{2\lambda\tau} \cdot \left(1 - \left(1 - 2\lambda\tau + \frac{(2\lambda\tau)^2}{2} - \frac{(2\lambda\tau)^3}{6}\right)\right) = 1 - \lambda\tau + \frac{2 \cdot (\lambda\tau)^2}{3}$$

$$A_{1oo2.IFRM} = 2 - \lambda\tau + \frac{(\lambda\tau)^2}{3} - 1 - \lambda\tau + \frac{2 \cdot (\lambda\tau)^2}{3} = 1 - \frac{(\lambda\tau)^2}{3}$$

Wenn beide Teilsysteme unabhängig voneinander ausfallen, nimmt die PFD mit dem Quadrat von $\lambda\tau$ ab:

$$PFD_{1oo2.IFRN} = 1 - A_{1oo2.IFRM} = \frac{(\lambda\tau)^2}{3}$$

A_{1oo2}	Verfügbarkeit von mindestens einem der beiden Teilsysteme.
$*_{*.IFRM}$	Unabhängige Ausfallursache, Reparatur erst bei nächster Wartung.
λ	Ausfallrate (Failure rate).
τ	Wartungsintervall.
PFD_{1oo2}	Wahrscheinlichkeit, dass beide Teilsysteme ausgefallen sind bei Anforderung.

Gleiche Ausfallursache und Sofortreparatur

Für zwei unabhängig ausfallende Systeme und Sofortreparatur ist die Wahrscheinlichkeit, dass nicht beide zufällig gleichzeitg ausgefallen sind, das Quadrat der Wahrscheinlichkeiten, dass nach Gl. 3.104 eines ausgefallen ist:

$$PFD_{1oo2.\mathrm{IFIR}} = (\lambda \cdot MTTR)^2$$

Wenn beide Komponenten zeitgleich durch dieselbe Ursache ausfallen (CC-Ausfälle), ist die PFD die eines Einzelsystems.

- Reparatur erst bei der nächten Wartung (Gl. 3.103a):

$$PFD_{1oo2.\mathrm{CCRM}} = \frac{\lambda\tau}{2}$$

- Sofortreparatur (Gl. 3.104):

$$PFD_{1oo2.\mathrm{CCIR}} = \lambda \cdot MTTR$$

PFD_{1oo2}	Wahrscheinlichkeit, dass beide Teilsysteme ausgefallen sind bei Anforderung.
$*_{*.\mathrm{IFIR}}$	Unabhängigen Ausfallursache, Sofortreparatur.
$*_{*.\mathrm{CCRM}}$	Gleiche Ausfallursache, Reparatur erst bei nächster Wartung.
$*_{*.\mathrm{CCIR}}$	Gleiche Ausfallursache, Sofortreparatur.

Gesamte PFD als gewichteter Mittelwert:

$$\begin{aligned} PFD_{1oo2} &= \eta_{CC} \cdot (\eta_{IR} \cdot PFD_{1oo2.CCIR} + (1-\eta_{IR}) \cdot PFD_{1oo2.CCRM}) \\ &+ (1-\eta_{CC}) \cdot (\eta_{IR} \cdot PFD_{1oo2.IFIR} + (1-\eta_{IR}) \cdot PFD_{1oo2.IFRM}) \\ &= \eta_{CC} \cdot \left(\eta_{IR} \cdot \lambda \cdot MTTR + (1-\eta_{IR}) \cdot \tfrac{\lambda\tau}{2}\right) \\ &+ (1-\eta_{CC}) \cdot \left(\eta_{IR} \cdot (\lambda \cdot MTTR)^2 + (1-\eta_{IR}) \cdot \tfrac{(\lambda\tau)^2}{3}\right) \end{aligned}$$

Im Normalfall $MTTR \ll \tau$ (Reparaturzeit viel kürzer als Wartungsintervall):

$$\begin{aligned} PFD_{1oo2} &= (1-\eta_{IR}) \cdot \frac{\lambda\tau}{2} + (1-\eta_{CC}) \cdot (1-\eta_{IR}) \cdot \frac{(\lambda\tau)^2}{3} \\ &= (1-\eta_{IR}) \cdot \left(\frac{\eta_{CC} \cdot \lambda\tau}{2} + \frac{(1-\eta_{CC}) \cdot (\lambda\tau)^2}{3}\right) \end{aligned} \tag{1.16}$$

1oo2-Redundanz verringert bei einem hohen Anteil von CC-Ausfällen $\eta_{CC} \gg \lambda\tau$ die PFD auf den Anteil der CC-Ausfälle und für seltene CC-Ausfälle $\eta_{CC} \ll \lambda\tau$ nimmt die PFD mit dem Quadrat der PFD eines Einzelsystems ab.

η_{CC}	Rate der Ausfälle durch eine gemeinsame Ursache.
η_{IR}	Anteil der Ausfälle, die durch Sofortreparatur beseitigt werden.
PFD_{1oo2}	Wahrscheinlichkeit, dass beide Teilsysteme ausgefallen sind bei Anforderung.

Weitere k aus n Systeme

Systemen mit größerem n oder k verhalten sich bei CC-Ausfällen auch wie ein Einzelsystem, d.h. die n Einzelsysteme fallen gemeinsam aus:

$$PFD_{koon.\mathrm{CC}} = PFD_{1oo2.\mathrm{CC}} = \frac{(1-\eta_{\mathrm{IR}})\cdot\lambda\tau}{2} + \eta_{\mathrm{IR}}\cdot\lambda\cdot MTTR$$

Für unabhängige Ausfälle und Reparatur erst bei der nächsten Wartung beträgt die PFD nach [6]:

k	1	2	3	4
$PFD_{koo2.\mathrm{IFRM}}$	$\frac{(\lambda\cdot\tau)^2}{3}$	$\lambda\cdot\tau$	-	-
$PFD_{koo3.\mathrm{IFRM}}$	$\frac{(\lambda\cdot\tau)^3}{4}$	$(\lambda\cdot\tau)^2$	$\frac{3\cdot\lambda\cdot\tau}{2}$	-
$PFD_{koo4.\mathrm{IFRM}}$	$\frac{(\lambda\cdot\tau)^4}{5}$	$(\lambda\cdot\tau)^3$	$2\cdot(\lambda\cdot\tau)^2$	$2\cdot\lambda\cdot\tau$

In Anlehnung an Gl. 4.16 ergibt sich als Gesamt-PFD für den Normalfall $MTTR \ll \tau$ (Reparaturzeit viel kürzer als Wartungsintervall):

$$PFD_{koon} = (1-\eta_{\mathrm{IR}})\cdot\left(\frac{\eta_{\mathrm{CC}}\cdot\lambda\tau}{2} + (1-\eta_{\mathrm{CC}})\cdot PFD_{koon.\mathrm{IFRM}}\right) \qquad (1.17)$$

k	1	2	3	4
$PFD_{koo2.\mathrm{IFRM}}$	$\frac{(\lambda\cdot\tau)^2}{3}$	$\lambda\cdot\tau$	-	-
$PFD_{koo3.\mathrm{IFRM}}$	$\frac{(\lambda\cdot\tau)^3}{4}$	$(\lambda\cdot\tau)^2$	$\frac{3\cdot\lambda\cdot\tau}{2}$	-
$PFD_{koo4.\mathrm{IFRM}}$	$\frac{(\lambda\cdot\tau)^4}{5}$	$(\lambda\cdot\tau)^3$	$2\cdot(\lambda\cdot\tau)^2$	$2\cdot\lambda\cdot\tau$

$$PFD_{koon} = (1-\eta_{\mathrm{IR}})\cdot\left(\frac{\eta_{\mathrm{CC}}\cdot\lambda\tau}{2} + (1-\eta_{\mathrm{CC}})\cdot PFD_{koon.\mathrm{IFRM}}\right) \quad (4.17)$$

Anmerkungen:

- Hohe Verfügbarkeit mit und ohne Redundanzen erfordert geringe Ausfallraten und angepasste Wartungsintervalle.
- Einen wirksame Verfügbarkeitserhöhung durch redundante Einheiten verlangt, dass CC-Ausfälle sehr unwahrscheinlich sind: kalte Reserve, räumliche Trennung, ...

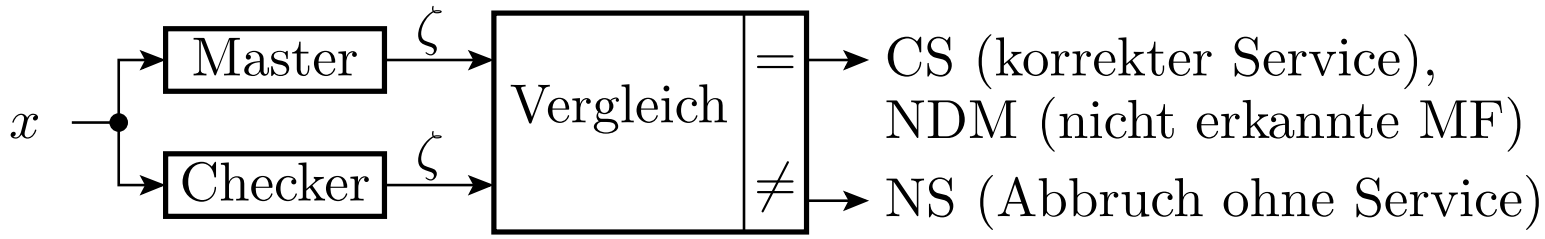
PFD_{koon}	Wahrscheinlichkeit, dass weniger als k der n Teilsysteme bei Anforderung verfügbar sind.
$*_{*.\mathrm{IFRM}}$	Unabhängige Ausfallursache, Reparatur erst bei nächster Wartung.
λ	Ausfallrate (Failure rate).
τ	Wartungsintervall.
η_{CC}	Rate der Ausfälle durch eine gemeinsame Ursache.
η_{IR}	Anteil der Ausfälle, die durch Sofortreparatur beseitigt werden.

Nutzung von Redundanzen in der Praxis

- Regelmäßig wartbare Systeme ohne extreme Anforderungen an die Verfügbarkeit und Sicherheit sind in der Regel 1oo1 (keine Redundanz).
- Im Maschinenbau je nach Sicherheitsstufe: auch 1oo2, um Reparaturzeiten zu vermeiden.
- Nur Systeme ohne einen in kurzer Zeit erreichbaren sicheren Zustand, z. B.: Flugzeugsteuerungen, Atomkraftwerke, Chemiereaktoren, auch 2oo2, 2oo3 oder 2oo4 mit Master-Checker-Betrieb oder Mehrversionsentscheid.
- Große KooN-Redundanzen: fehlertolerante Server-Cluster, Verbindungsnetzwerke, ..., also Systeme, die ohnehin aus vielen gleichen Teilsystemen bestehen mit hohen Anforderungen an die Verfügbarkeit.

Systemlösungen

Master-Checker-System mit Notbetrieb

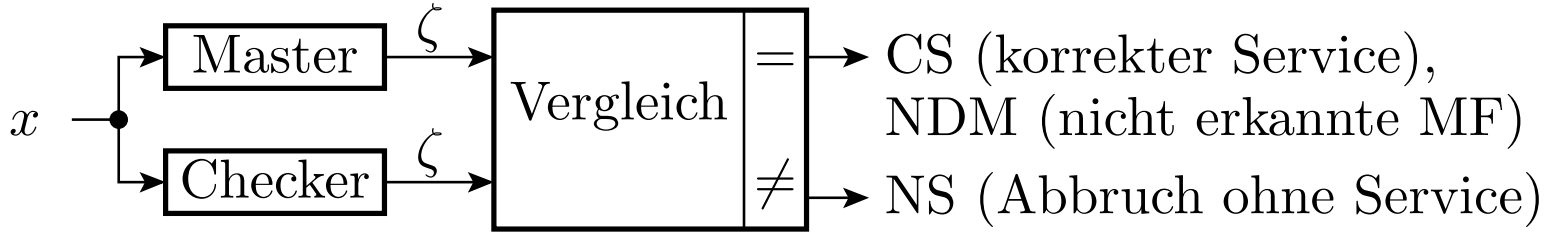


Für ein Master-Checker-System mit Abbruch nach Vergleichsfehler und ohne Berücksichtigung von Ausfällen wurde hergeleitet (siehe Folie 1.81 *Master-Checker-Überwachung mit STMF*) :

$$A_{\mathrm{MT}} = \frac{MTBM}{MTBM + 2 \cdot \eta_{\mathrm{Div}} \cdot MTB} \tag{1.34}$$

$$R_{\mathrm{MT}} = \frac{R_{\mathrm{MS}} - 1}{(1 - \eta_{\mathrm{Div}})} \tag{1.36}$$

A_{MT}	MT-Verfügbarkeit, Zeitanteil, den das System nicht mit MT beschäftigt ist.
ζ_{MS}	Übereinstimmende Fehlfunktionsrate von Master und Checker.
$MTTR$	Mittlere Reparaturzeit (Mean time to repair).
MTS	Mittlere Service-Dauer (Mean time to service).
R_{MT}	Zuverlässigkeit mit Fehlfunktionsbehandlung (Reliability with malfunction treatment).
R	Zuverlässigkeit (reliability) ohne Fehlfunktionsbehandlung.
η_{Div}	Diversitätsrate, Anteil der nicht übereinstimmenden MF bei Mehrfachberechung.



Zweifachberechnung und Vergleich ist ein 2oo2-System mit eine ausfallbezogenen Verfügbarkeit nach Gl. 4.17 von:

$$A_{2oo2} = 1 - \eta_{CC} \cdot (1 - \eta_{IR}) \cdot \frac{\lambda\tau}{2} + (1 - \eta_{CC}) \cdot (1 - \eta_{IR}) \cdot \lambda \cdot \tau$$
$$= 1 - (1 - \eta_{IR}) \cdot \left(1 - \tfrac{\eta_{CC}}{2}\right) \cdot \lambda \cdot \tau$$

Insgesamt verfügbar als 2oo2 **UND** als MCS (Master-Checker-System):

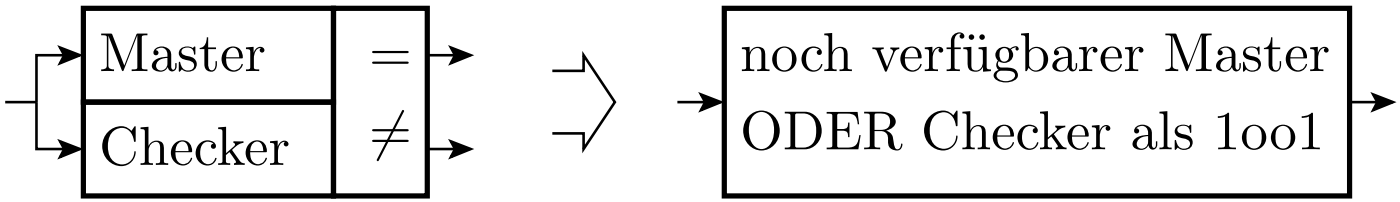
$$A_{2oo2.MCS} = A_{2oo2} \cdot A_{MT}$$

Mit beiden Verfügbarkeiten nahe 100%:

$$A_{2oo2.MCS} = 1 - (1 - \eta_{IR}) \cdot \left(1 - \tfrac{\eta_{CC}}{2}\right) \cdot \lambda \cdot \tau - \zeta \cdot \tfrac{MTTR}{MTS} \qquad (1.18)$$

A_{2oo2}	Verfügbarkeit als 2oo2 Master-Check-System.
η_{CC}	Rate der Ausfälle durch eine gemeinsame Ursache.
η_{IR}	Anteil der Ausfälle, die durch Sofortreparatur beseitigt werden.
λ	Ausfallrate (Failure rate).
τ	Wartungsintervall.

Verfügbarkeitserhöhung durch Notbetrieb



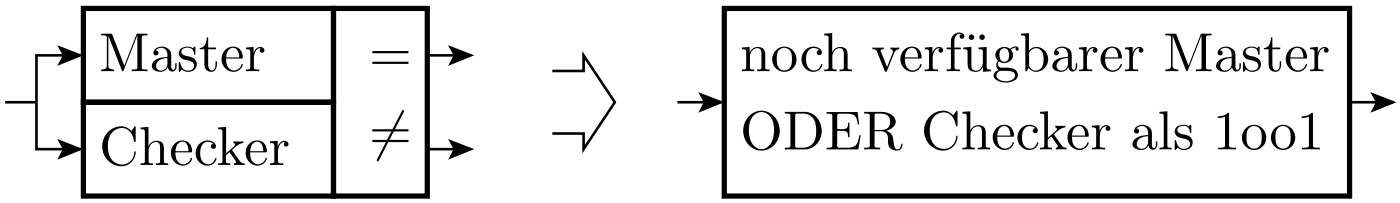
Ohne Vergleich und Aussortieren abweichender Ergebnisse ist ein Master-Checker-System ein 1oo1-System mit der Verfügbarkeit eins abzüglich PFD nach (Gl. 3.105):

$$A_{1oo1} = 1 - \eta_{IR} \cdot \lambda \cdot MTTR - (1 - \eta_{IR}) \cdot \frac{\lambda \cdot \tau}{2}$$

Gesamtverfügbarkeit des Master-Checker-Systems mit Notbetrieb:

$$A_{MCN} = A_{2oo2} + (1 - A_{2oo2}) \cdot A_{1oo1} \tag{1.19}$$

A_{1oo1}	Verfügbarkeit als nichtüberwachtes Einzelsystem.
η_{IR}	Anteil der Ausfälle, die durch Sofortreparatur beseitigt werden.
λ	Ausfallrate (Failure rate).
$MTTR$	Mittlere Reparaturzeit (Mean time to repair).
τ	Wartungsintervall.

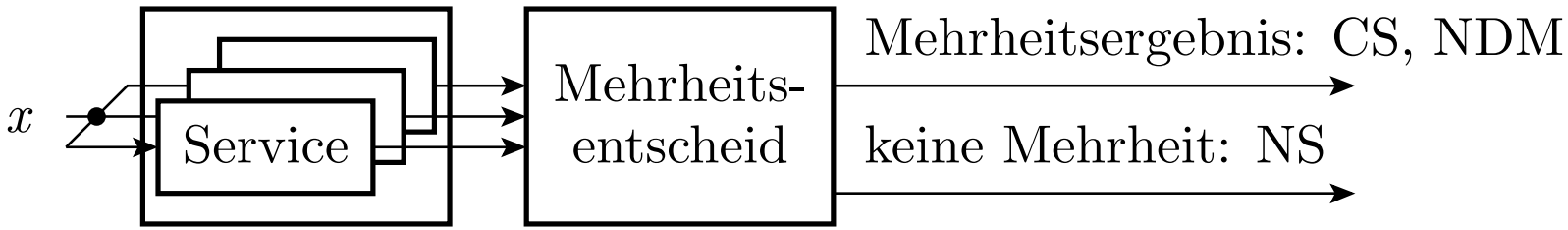


Die Zuverlässigkeit im Notbetrieb ist nur R statt $(R-1)\,/\,(1-\eta_{\mathrm{Div}})$ und insgesamt im Mittel:

$$R_{\mathrm{MCN}} = \frac{\frac{R-1}{(1-\eta_{\mathrm{Div}})} \cdot A_{2\mathrm{oo}2} + R \cdot (1 - A_{2\mathrm{oo}2}) \cdot A_{1\mathrm{oo}1}}{A_{\mathrm{MCN}}} \tag{1.20}$$

R_{MCB}	Zuverlässigkeit eines Master-Checker-Systems mit Notbetrieb.
R	Zuverlässigkeit (reliability) ohne Fehlfunktionsbehandlung.
η_{Div}	Diversitätsrate, Anteil der nicht übereinstimmenden MF bei Mehrfachberechung.
$A_{2\mathrm{oo}2}$	Verfügbarkeit als 2oo2 Master-Check-System.
$A_{1\mathrm{oo}1}$	Verfügbarkeit als nichtüberwachtes Einzelsystem.
A_{MCB}	Verfügbarkeit eines Master-Checker-Systems mit Notbetrieb.

Mehrfachberechung mit Mehrheitsentscheid



Mindestens 3 identische System führen dieselbe Berechnung aus und geben das Mehrheitsergebnis weiter. DS-Rate und Zuverlässigkeit ohne Berücksichtigung von Ausfällen:

$$A_{\mathrm{MT}} = \frac{MTBM}{MTBM+2\cdot\eta_{\mathrm{Div}}\cdot(MTB+MTS)} \tag{1.42}$$

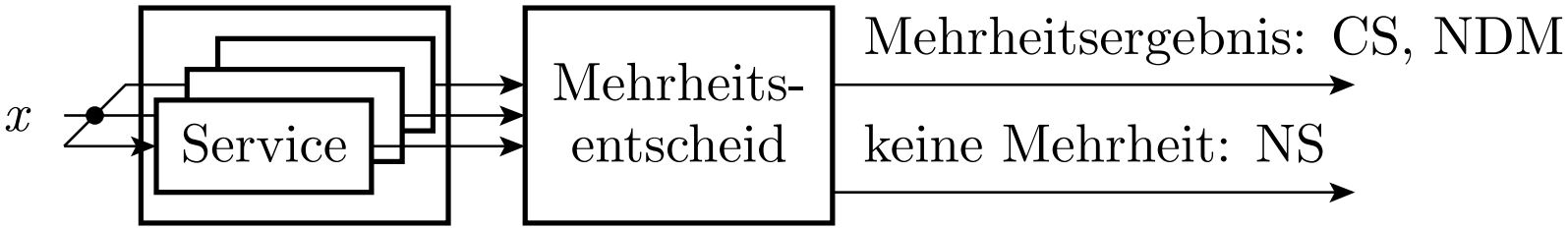
$$R_{\mathrm{MT}} = \frac{R_{\mathrm{MS}}}{(1-\eta_{\mathrm{Div}})} - 1 \tag{1.44}$$

Nach einem Ausfall reduziert sich die Zuverlässigkeit bis zum Abschluss der Reparatur auf die eines Master-Checker-Systems

$$R_{\mathrm{MT}} = \frac{R_{\mathrm{MS}}-1}{(1-\eta_{\mathrm{Div}})} \tag{1.36}$$

und wenn vor Abschluss der Reparatur ein zweites System ausfällt, auf die ohne MF-Behandlung R.

Ausfallbezogene Verfügbarkeit



Wenn nur das Ausfallrisiko betrachtet wird, ist das System verfügbar:

- mit 3-Versionen Mehheitsentscheid:

$$A_{3oo3} = 1 - \eta_{CC} \cdot (1 - \eta_{IR}) \cdot \frac{\lambda\tau}{2} + (1 - \eta_{CC}) \cdot (1 - \eta_{IR}) \cdot \frac{3 \cdot \lambda \cdot \tau}{2}$$

- als Master-Checker-System mit Notbetrieb:

$$A_{MCB} = A_{2oo2} + (1 - A_{2oo2}) \cdot A_{1oo1} \tag{4.19}$$

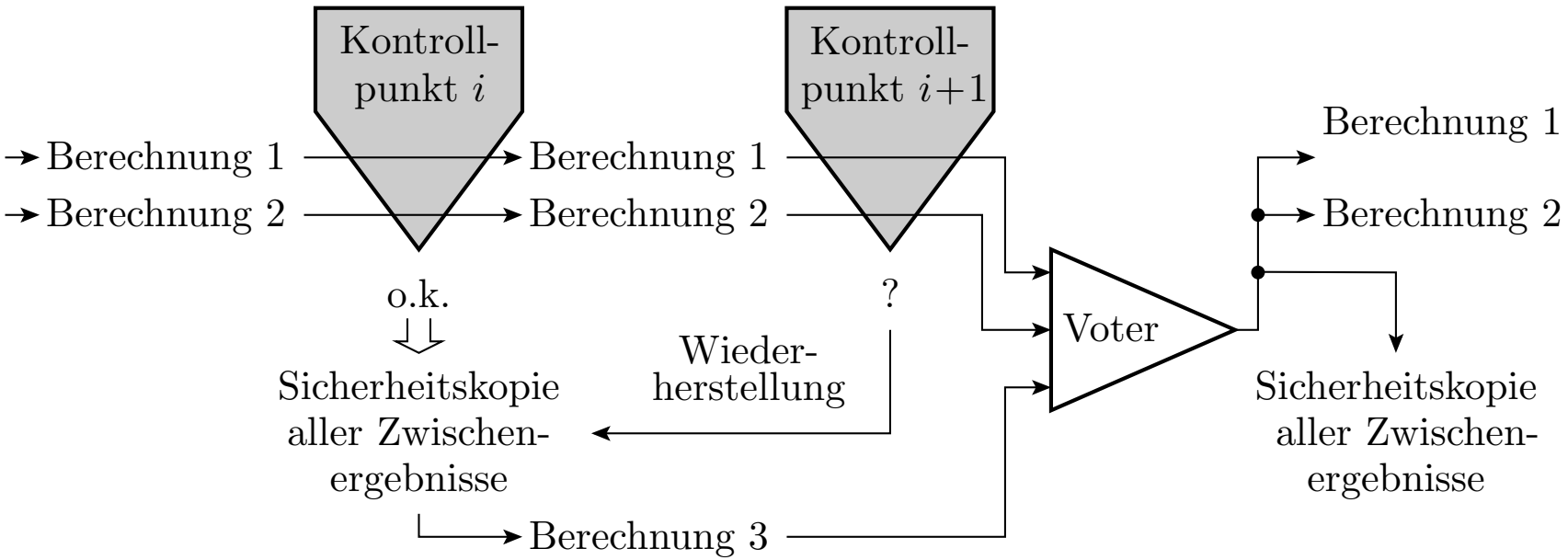
- Gesamtverfügbarkeit:

$$A_{V3B} = A_{3oo3} + (1 - A_{3oo3}) \cdot A_{MCN}$$

A_{3oo3} Verfügbarkeit als 3-Versionssystem mit Mehrheitsentscheid.
A_{V3B} Verfügbarkeit eines 3-Versionssystems mit Notbetrieb.
A_{MCB} Verfügbarkeit eines Master-Checker-Systems mit Notbetrieb.

Ein 2oo3-System mit Mehrfachberechung und Vergleich wurde bereits 1956 von John von Neumann für die damaligen Röhrenrechner vorgeschlagen, in denen alle paar Stunden eine Röhre ausgefallen ist.

Check-Point-Roll-Back-Recovery [5]



- Zwei parallele möglichst unabhängig verfügbare Berechnungen.
- An einprogrammierten Kontrollpunkten im Programm werden die Bearbeitungszustände (Variablen, Register, ...) verglichen.
- Bei Übereinstimmung Speicherung des Bearbeitungszustands in einem geschützten Speicher.
- Bei Abweichung, Laden der letzten Sicherheitskopie und Berechnungswiederholung (Roll-Back Recovery).

- Nach Roll-Back Recovery am nächsten Kontrollpunkt wieder Vergleich.
- Wenn Übereinstimmung, diesen als gesicherten Zustand speichern, sonst Abbruch.

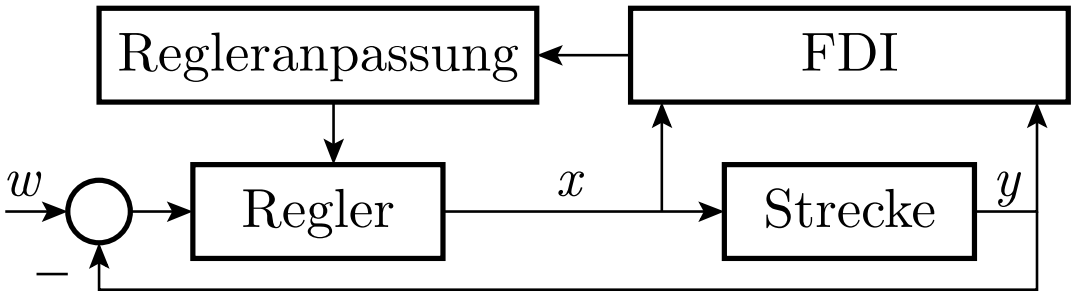
Vergleich mit 3-Versionen Mehrheitsentscheid:

- nur 2 Systeme und für die meisten Service-Leistungen nur zwei Berechnungen erforderlich.
- ausfallbezogenen Teilverfügbarkeit: Mehheitsentscheid 2oo2, mindestens ein Einzelsyystem 1oo2.

Beispiel: Sequoia-System [1]:

- Berechnung auf zwei Prozessoren mit eigenem Write-Back-Cache.
- Vergleich in jedem Takt.
- Zustands-Backup bei Ereignissen wie Stack-Überlauf und Prozesswechsel.
- Hauptspeicher hat die Funktion des stabilen Speichers.

Fehlertolerantes Regelungssystem



In einem Reglersystem wird vom Sollwert w der zu regelnde Ist-Wert y abgezogen. Aus der Differenz bildet der Regler den Stellwert x für die Regelstrecke (z.B. eine Heizung, wenn y eine Temperatur ist).

Hinzufügen einer zusätzlichen Überwachungs- und Fehlerbehandlungsschicht (FDI) mit den Aufgaben:

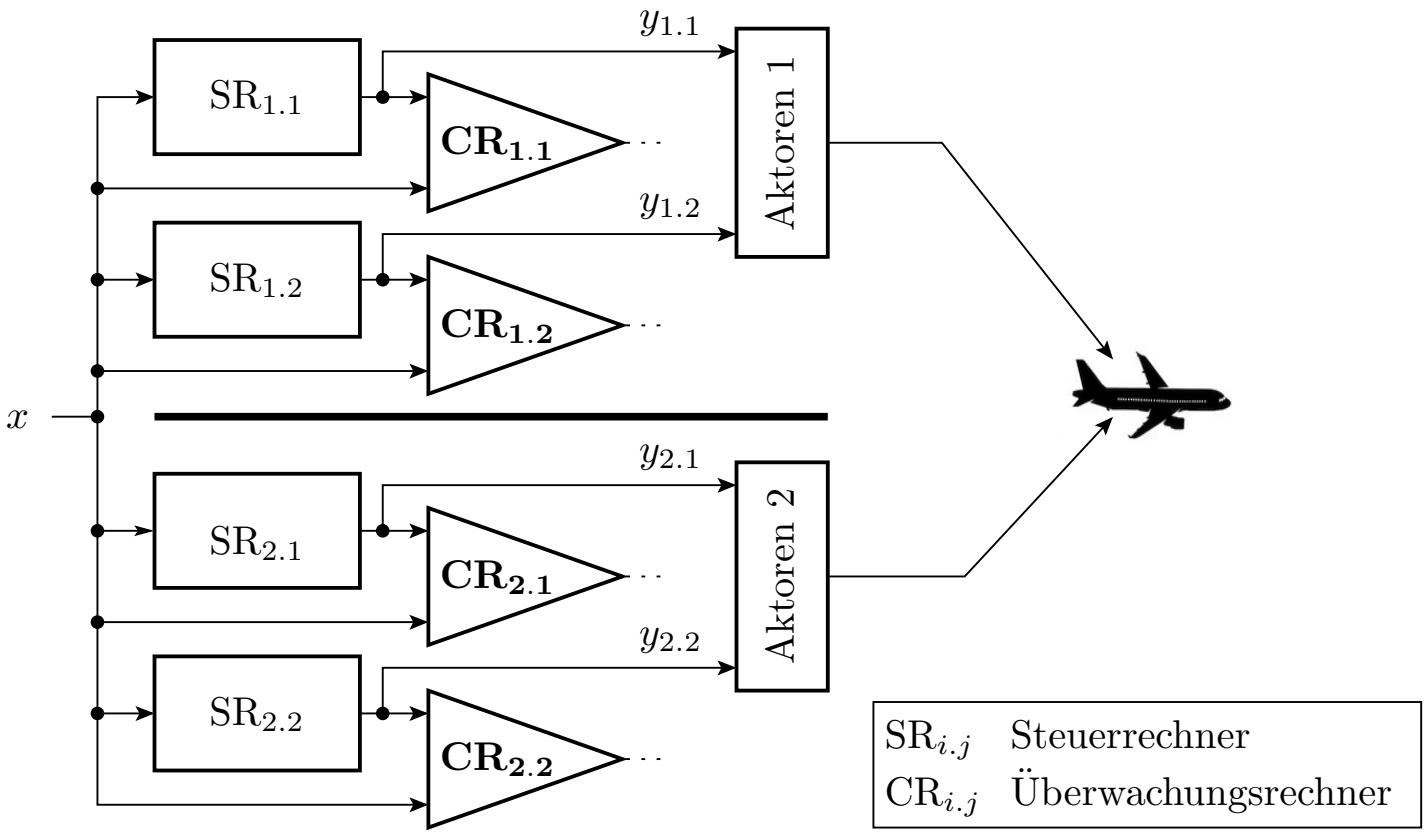
- Fehlerdiagnose (Abschätzung von Fehlerursache und -ort) und
- Anpassung der Regelung an den aktuellen Fehlerzustand so, dass eine Mindestfunktionalität gewährleistet bleibt.

Zur Erzielung von Fehlertoleranz gegenüber Fehlfunktionen von Regler und Strecke durch Störungen, Fehler und Ausfälle.

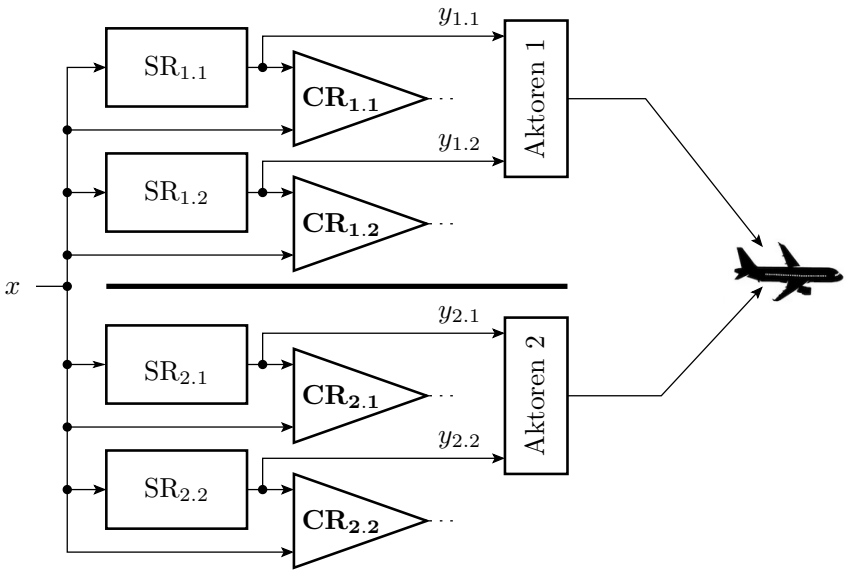
FDI Fehlerdetektion, -isolation und -identifikation.

Flugsteuersystem Airbus A3XX [8]

Hochsicherheitskritische Anwendungen müssen möglichst alle Fehlfunktionen, auch solche durch nicht erkannte Entwurfsfehler, nicht erkannte Fertigungsfehler und Ausfälle tolerieren.



- Zwei identische Systeme mit allen Sensoren, Aktoren und zwei Rechnerpaaren.
- Jedes Rechnerpaar besteht aus einem Steuerrechner $SR_{i.j}$, der die Aktoren ansteuert, und einem Überwachungsrechner $CR_{i.j}$.



- Normalzustand Rechner $SR_{1.1}$ steuert und $CR_{1.1}$ überwacht. Zweites Rechnerpaar Stand-By. System 2 abgeschaltet.
- Bei Ausfall übernimmt Rechnerpaar 1 von Rechnerpaar 2. Bei Komplett-, Sensor- oder Aktorausfällen übernimmt System 2 von System 1.

Diversität: Rechner unterschiedlicher Hersteller, getrennte Software-Entwicklung nach Spezifikationen, die unabhängig von einer gemeinsamen Basisspezifikation abgeleitet wurden.

Zusammenfassung

Fehlerkorrigierende Codes

Erweiterung der Menge der darstellbaren Codeworte um eine viel größere Menge korrigierbarer Codeworte und optional um unzulässige nicht korrigierbare Codeworte. Mindestbitanzahl:

$$2^{\#\mathrm{Bit}} \geq \#VCW + \#VCW \cdot \#CVC \tag{4.13}$$

Hamming-Distanz:

- zum Erkennen von bis zu $\#DB$ verfälschten Bits:

$$\#HD \geq \#DB + 1 \tag{4.14}$$

- zur Korrektor von bis zu $\#CB$ verfälschten Bits:

$$\#HD \geq 2 \cdot \#CB + 1 \tag{4.15}$$

Hamming-Codes

- Paritätsbit: modulo-2 Summe (EXOR-Verknüpfung) aller Datenbits. Nachweis aller ungeradzahligen Verfälschungen.
- Kreuzparität: Zeilen- und Spaltenparität für ein 2D-Datenfeld. Korrektur bis zu einem verfälschten Bit.
- 1-Bit fehlerkorrigierender Hamming-Code. Rechenweg zur Festlegung der EXOR-Summen für die Kontrollbits so, dass die Kontrollbitdifferenz die binärcodierte Nummer des verfälschten Bits ist.
- Konstruktion von Codes für die Burst-Fehler Korrektur durch Verschränkung von Code-Worten für die Korrektur von Einzelbitverfälschungen.

RAID und Backup

Ein RAID ist ein logisches Laufwerk aus mehreren physischen Festplatten oder SSD, optional mit redundanter Datenspeicherung:

- RAID0: redundanzfreie Speicherung auf mehrere Festplatten.
- RAID1: paarweise gespiegelte Platten oder Plattengruppen. Tolerierung von mindestens einem Plattenausfall.
- RAID mit Paritätsblöcken: bei n Platten speichern auf $n-1$ Platten Datenblöcke und auf eine ein Paritätsblock. Erlaubt nach einem Plattenausfall die Rekonstruktion aller Daten aus denen der noch verfügbaren Platten.

Erweiterungen:

- Techniken für Bitfehlerkorrektur , ...
- Ho-Fix: Reserveplatte, die sofort nach Plattenausfall deren Funktion übernimmt.
- Rebuilt: Rekonstruktion der verlorenen Daten für die Ersatzplatte.

Selbst ausgefeilte RAIDs tolerieren nur HW-Probleme. Wirklich zuverlässigen Schutz gegen Datenverluste bieteten Backups.

Redundanzen, KooN-Systeme

Wahrscheinlichkeit der Nicht-Verfügbarkeit:

- 1oo2-System:

$$PFD_{1oo2} = (1 - \eta_{IR}) \cdot \left(\frac{\eta_{CC} \cdot \lambda\tau}{2} + \frac{(1-\eta_{CC}) \cdot (\lambda\tau)^2}{3} \right) \quad (4.16)$$

- Weitere KooN-Systeme:

k	1	2	3	4
$PFD_{koo2.IFRM}$	$\frac{(\lambda \cdot \tau)^2}{3}$	$\lambda \cdot \tau$	-	-
$PFD_{koo3.IFRM}$	$\frac{(\lambda \cdot \tau)^3}{4}$	$(\lambda \cdot \tau)^2$	$\frac{3 \cdot \lambda \cdot \tau}{2}$	-
$PFD_{koo4.IFRM}$	$\frac{(\lambda \cdot \tau)^4}{5}$	$(\lambda \cdot \tau)^3$	$2 \cdot (\lambda \cdot \tau)^2$	$2 \cdot \lambda \cdot \tau$

$$PFD_{koon} = (1 - \eta_{IR}) \cdot \left(\frac{\eta_{CC} \cdot \lambda\tau}{2} + (1 - \eta_{CC}) \cdot PFD_{koon.IFRM} \right) \quad (4.17)$$

Praktische Nutzung:

- Maschinen bis 1oo2
- Flugzeugsteuerungen, Atomkraftwerke, Chemiereaktoren, .. auch 2oo2 bis 2oo4.

Systemlösungen

Master-Checker-System mit Notbetrieb:

- Verfügbarkeit als Master-Checker-System:

$$A_{2oo2.\mathrm{MCS}} = 1 - (1 - \eta_{\mathrm{IR}}) \cdot \left(1 - \frac{\eta_{\mathrm{CC}}}{2}\right) \cdot \lambda \cdot \tau - \zeta \cdot \frac{MTTR}{MTS} \tag{4.18}$$

- Verfügbarkeit einschließlich Notbetrieb ohne Checker und Vergleich:

$$A_{\mathrm{MCB}} = A_{2oo2} + (1 - A_{2oo2}) \cdot A_{1oo1} \tag{4.19}$$

- Zuverlässigkeit mit Umschaltung in den Notbetrieb:

$$R_{\mathrm{MCB}} = \frac{\frac{R-1}{(1-\eta_{\mathrm{Div}})} \cdot A_{2oo2} + R \cdot (1 - A_{2oo2}) \cdot A_{1oo1}}{A_{\mathrm{MCN}}} \tag{4.20}$$

3-Versions System mit den Notbetriebsarten:

- solange alle Teilsysteme verfügbar, Mehrheitsentscheid,
- wenn nur noch zwei Teilsysteme verfügbar, Master-Checker-Betrieb,
- solange nur noch ein Teilsystem verfügbar, Betrieb als nicht überwachtes Einzelsystem.

Check-Point-Roll-Back-Recovery: Master-Checker-Paar mit Drittberechnung und Mehrheitsentscheid nach Vergleichsfehlern. Spart das dritte System, braucht aber ein Backup für die Zwischenergebnisse an den Kontrollpunkten für die möglicherweise erforderliche dritte Berechnung.

Fehlertolerantes Regelungssystem: zusätzlichen Überwachungs- und Fehlerbehandlungsschicht (FDI) zur Anpassung der Regelung an den aktuellen Fehlerzustand. Die den Fehlerzuständen zugeordneten Regler bilden eine funktionale Redundanz.

Flugsteuersystem Airbus: Beispiel für ein komplexes System mit

- Paaren aus Master-Rechnen und Kontrollrechnern zur Überwachung und MF-Behandlung,
- diversitären Redundanzen für Fehlertoleranz gegenüber SW-Fehlern und
- Komplett-Redundanz incl. aller Sensoren und Aktoren zur Minimierung der Risiken gegenüber allen denkbaren CC-Ausfällen.

Literatur

5. Literatur

[1] P.A. Bernstein.
Sequoia: a fault-tolerant tightly coupled multiprocessor for transaction processing.
Computer, 21(2):37–45, 1988.

[2] Nader B. Ebrahimi.
On the statistical analysis of the number of errors remaining in a software design document after inspection.
IEEE Transactions on Software Engineering, 23(8):529–532, 1997.

[3] Peter Liggesmeyer.
Software-Qualität: Testen, Analysieren und Verifizieren von Software.
Spectrum, 2002.

[4] Frank Padberg, Thomas Ragg, and Ralf Schoknecht.
Using machine learning for estimating the defect content after an inspection.
IEEE Transactions on Software Engineering, 30(1):17–28, 2004.

[5] D. K. Pradhan, D. D. Sharma, and N. H Vaidya.
Roll-forward checkpointing schemes.
In *Lecture Notes in Computer Science 744*, pages 93–116. Springer Verlag, 1994.

[6] Marvin Rausand and Arnljot Hsyland.
Systems Reliability Theory, Models, Statistical Methods, and Applications.
Wiley-Interscience, 2004.

[7] Qinbao Song, Martin Shepperd, Michelle Cartwright, and Carolyn Mair.
Software defect association mining and defect correction effort prediction.
IEEE Transactions on Software Engineering, 32(2):69–82, 2006.

[8] Pascal Traverse.
Dependability of digital computers on board airplanes.
Dependable Computing for critical applications, 4:134–152, 1991.