Test und Verlässlichkeit Foliensatz 3: Ergebnisüberwachung

Prof. G. Kemnitz

Institut für Informatik, TU Clausthal (TV_F3)

June 7, 2018

Inhalt Foliensatz TV_F3: Ergebnisüberwachung und statische Tests

Überwachung und Test

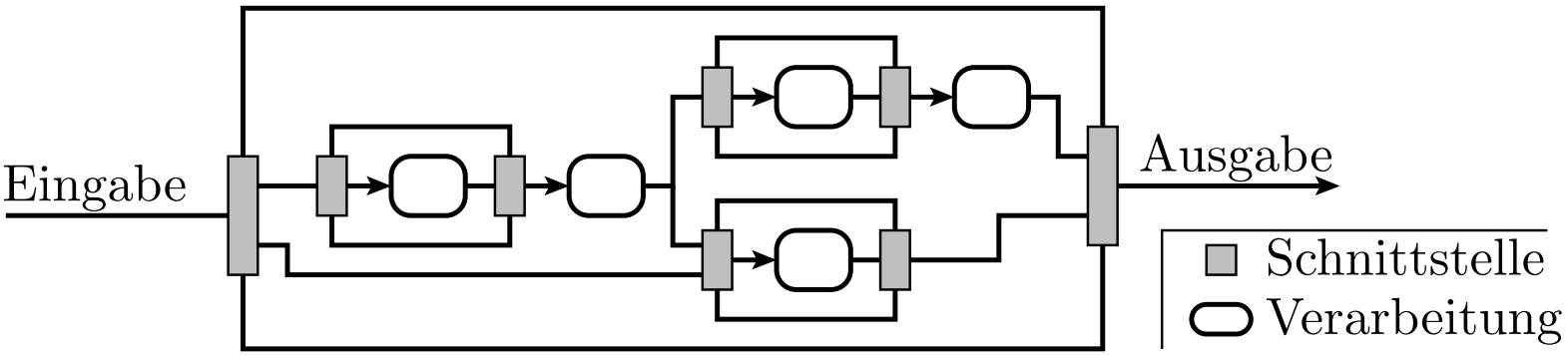
Ergebnisüberwachung und Test

Die Verlässlichkeit von IT-Systemen wird durch Fehlfunktionen beeinträchtigt, die durch Fehler, Störungen und Ausfälle verursacht werden. Die Gefährdungsminderung erfolgt auf drei Ebenen:

- geeignete Reaktionen auf FF: Problemerfassung, Wiederholung, Fehlerumgehung, ...
- Test und Fehlerbeseitigung und
- Fehlervermeidung durch Fehlerbeseitigung und Minderung der Störanfälligkeit.

Entscheident für die erzielbare Zuverlässigkeit und Sicherheit sind die Erkennungswahrscheinlichkeiten für FF bei der Überwachung und für Fehler beim Test.

Überwachung von Service-Leistungen



Im Service-Modell wird die Verarbeitung als Folge diskreter Schritte modelliert, die aus Eingaben Ausgaben erzeugen. Komplexe SL setzen sich aus Teil-SL zusammen. Service-Leister können sein:

- Digitale Schaltungen, Rechner, ...
- Programme, programmierte Rechern,
- Entstehungsprozesse oder Schritte davon.

Eine Überwachung erfolgt in der Regel an den Schnittstellen:

- Eingabe, Ausgabe,
- Schnittstellen zwischen Teil-SL.

Format- und Wertekontrolle

Kontrollier- und überwachbare Merkmale:

- Format: Konstante, immer erfüllte Merkmale (Zeitraster, ...).
- Daten: Variable Merkmale (Werte von Datenobjekten, ...).

Die Ein- und Ausgaben einer SL haben ein Format, das von der Art der SL abhängt, und einen Wert.

Definition Format

Menge aller kontrollierbaren datenunabhängigen Merkmale.

- Ausgaben mit Formatfehlern sind Fehlfunktionen.
- SL ohne Formatfehler können, aber müssen keine FF sein.
- Eingabefehler sind FF der SL, die diese erzeugt hat. ...

Statische und dynamische Tests

Test: Verfahren zum Ausspühren von Fehlern.

- statische Test: direkte Kontrolle von Merkmalen (z.B. Syntax),
- dynamischer Test: Ausprobieren der Funktion.

Ein Test ist gleichzeitig eine Kontrolle der Entstehungs-SL.

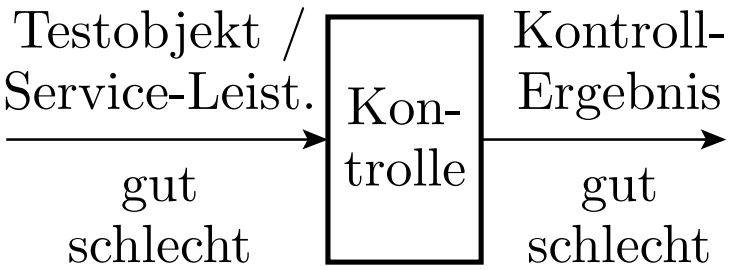
Während dynamische Tests erst am funktionsfähigen Produkt durchführbar sind, sind statische Tests bereits nach Teil-Entst.-SL, d.h. in früheren Phasen der Produktentstehung, möglich.

Prüf- und Reparatur-Technologien nutzen in der Regel eine Vielzahl statischer und dynamischer Tests, die auf unterschiedliche potentielle Fehler abzielen.

Die statischen Tests erfolgen in der Regel vor den dynamischen Tests. Dynamische Tests zielen auf potentielle Fehler, die statische Tests nur mit unzureichender Wahrscheinlichkeit erkennen.

FF von Überwachungs- und Test-SL

Überwachung und Test sind Kontrollen. Kontrollen sind SL mit potentiellen FF:

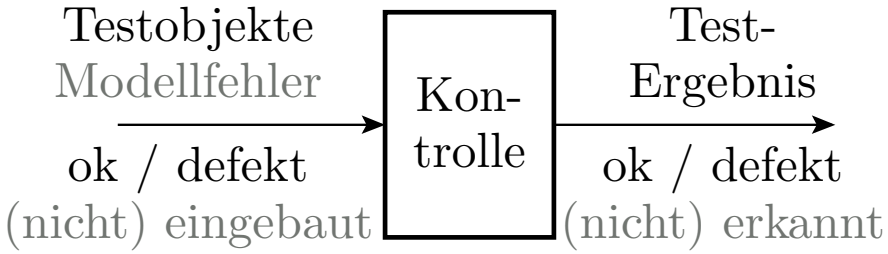


Service-Ergebnis	Kontroll-ergebnis	Fehlfunktion der Kontrolle
schlecht	schlecht	—
gut	gut	—
schlecht	gut	Maskierung
gut	schlecht	Phantom

Bei der binären Klassifikation in »richtig« und »falsch« gibt es zwei Arten von Kontroll-FF:

- Maskierung: Nichterkennen von Fehlern / FF.
- Phantom-Fehler / -FF: vermeindliches Erkennen nicht vorhandener Fehler / FF.

Kenngrößen für Tests



Test-objekt	Test-Ergebnis	Fehlfunktion des Tests
defekt	defekt	—
ok	ok	—
defekt	ok	Maskierung
ok	defekt	Phantomfehler

- Fehlerüberdeckung, Schätzer der Erkennungswahrscheinlichkeit:

$$p_{\mathrm{E}} \approx FC = \frac{\varphi_{\mathrm{Erk}}}{\varphi}$$

(φ – Fehleranzahl; φ_{Erk} – Anzahl erkannte Fehler).

- Wahrscheinlichkeit, dass Tests Phantomfehler[1] ausweisen:

$$p_{\mathrm{Phan}} \approx \frac{N_{\mathrm{Phan}}}{N_{\mathrm{Test}}}$$

(N_{Test} – Anzahl der Tests; N_{Phan} – Anzahl der Phantomfehler).

[1] Scheinbare Fehler, die in Wirklichkeit keine sind.

Kenngrößen für die Überwachung

- Erkennungs- und Maskierungswahrscheinlichkeit für FF:

$$p_{\mathrm{E}} \approx \frac{N_{\mathrm{EFF}}}{N_{\mathrm{FF}}}$$

$$p_{\mathrm{M}} = 1 - p_{\mathrm{E}} \approx \frac{N_{\mathrm{MFF}}}{N_{\mathrm{FF}}}$$

- Wahrscheinlichkeit für Phantomfehlfunktionen:

$$p_{\mathrm{Phan}} \approx \frac{N_{\mathrm{Phan}}}{N_{\mathrm{SL}}}$$

(N_{FF} – Anz. aller FF, N_{EFF} – Anz. erkennbare FF; N_{MFF} – Anz. maskierte FF; N_{SL} – Anz. aller SL; N_{Phan} – Anz. der Phantom-FF[2]).

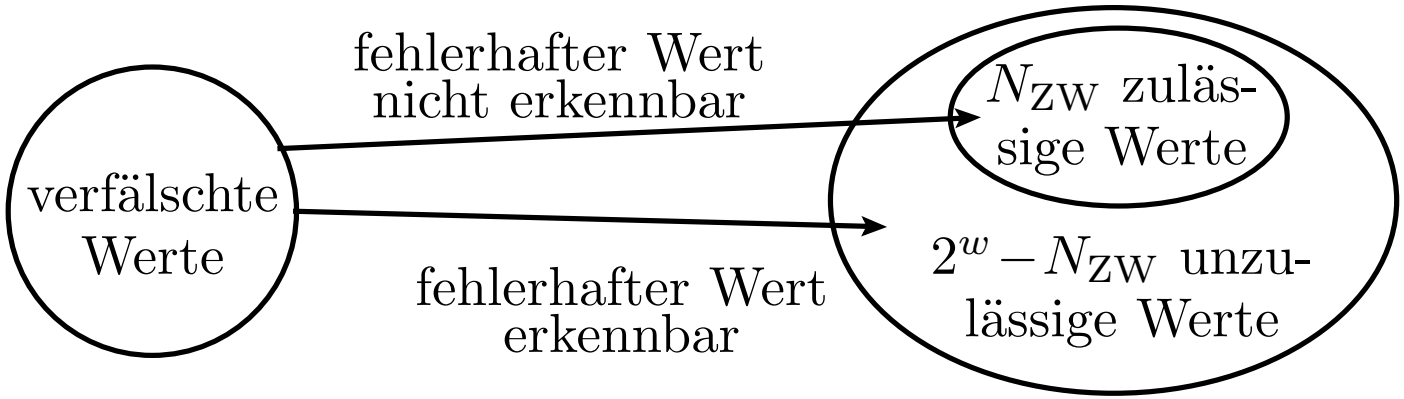
Die geringsten Maskierungs- und Phantom-FF-Wahrscheinlichkeiten haben Kontrollen auf Basis von Informationsredundanz.

[2] Vermeindlich erkannte FF, die in Wirklichkeit korrekte SL sind.

Informationsredundanz

Informationsredundanz

Die binäre Darstellung von N_{ZW} zu unterscheidenden zulässigen Werten verlangt mindestens $w \geq \log_2(N_{\mathrm{ZW}})$ Bits. Bei $2^w > N_{\mathrm{ZW}}$ (mehr darstellbare als darzustellende Werte) weisen unzulässige Werte auf Fehlfunktionen hin.



Eine Darstellung mit $w \gg \log_2(N_{\mathrm{ZW}})$ (die überwiegende Mehrheit der darstellbaren Werte ist unzulässig) erlaubt das Erkennen und bei geschickter Codierung sogar die Korrektur von Datenverfälschungen.

Erkennungswahrscheinlichkeit

- Wenn sich fehlerhafte Werte gleichmäßig auf zulässige und unzulässige Werte abbilden, Erkennungswahrscheinlichkeit:

$$p_{\mathrm{E}} \approx 1 - \frac{N_{\mathrm{ZW}}}{2^w} \tag{1}$$

  (N_{ZW} – Anzahl der zulässigen Werte; w – Bitanzahl zur Darstellung).
- Durch Wahl einer ausreichenden Bitanzahl w praktisch eins.
- Wenn fehlerhafte Werte unverhältnismäßig oft zulässig sind, kann p_{E} deutlich kleiner sein als nach Gl. 1.
- Wenn fehlerhafte Werte unverhältnismäßig oft unzulässig sind, ist p_{E} größer als nach Gl. 1.

Beispiel Rechtschreibtest

Wort im Wörterbuch enthalten?

- Maskierung: falsches Wort, das im Wörterbuch enthalten ist, z.B. »Maus« statt »Haus«.
- Phantom-FF: zulässiges Wort nicht im Wörterbuch.

- Anzahl der mit N_{Byte} darstellbaren Zeichenketten:

$$2^{8 \cdot N_{\mathrm{Byte}}}$$

  Anteil der gültigen Worte fast null. Nach Gl. 1 $p_{\mathrm{E}} \approx 1$.
- Tatsächlich $p_{\mathrm{E}} \approx 80\%$, weil Schreibfehler viel öfter als zufällige Bitverfälschungen gültige Worte sind.
- Es gibt auch relativ viele Worte, die im Wörterbuch fehlen. Phantom-FF-Wahrscheinlichkeit typisch $p_{\mathrm{Phan}} > 1\%$.

Einzelbitfehler bei Übertragung und Speicherung

Bei der Datenspeicherung und Übertragung sind Bitfehler äußerst selten.

Wenn die Daten so auf Datenobjekte aufgeteilt werden, dass jede Verfälschungsursache, z.B. ein Störimpuls, nur in jedem Datenobjekt ein Bit verfälscht, genügt der Nachweis aller Einzelbitfehler, um fast alle Datenverfälschungen zu erkennen.

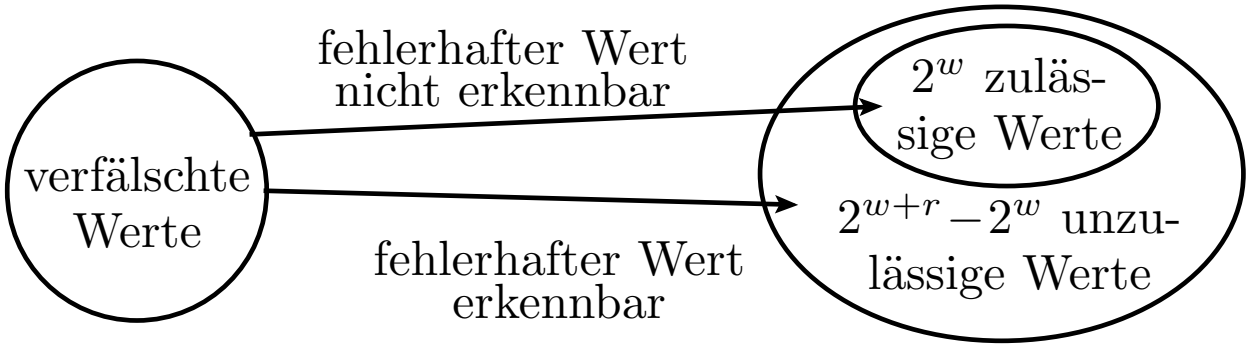
Für die Einzelbitfehlererkennung genügt ein Code, in dem nur Werte mit gerader (ungerader) Anzahl von Einsen gültig sind (siehe später Erweiterung um ein Paritätsbit).

Anteil gültiger Codeworte 50%, Erkennungswahrscheinlichkeit

$$p_{\mathrm{E}} \approx 1 \gg 1 - 50\%$$

Fehlererkennende Codes

Fehlererkennende Codes (FEC)



Bei einem fehlererkennenden Code werden die zulässigen Werte pseudo-zufällig auf eine viel größere Menge darstellbarer Werte verteilt. Pseudo-zufällig bedeutet hier, Zuordnung nach einem umkehrbaren Algorithmus so, dass Verfälschen weder bevorzugt auf zulässige noch auf unzulässige Datenworte abgebildet werden und Gl. 1 gilt. Mit 2^w zulässigen Werten und 2^{w+r} darstellbaren Werten beträgt die Erkennungswahrscheinlichkeit:

$$p_{\mathrm{E}} = 1 - \frac{2^w}{2^{w+r}} = 1 - 2^{-r} \qquad (2)$$

Arithmetische Codes

Arithmetische Codes werden durch eine Menge von arithmetischen Operationen gebildet. Beispiel Multiplikation der Datenworte mit einer ganzzahligen Konstanten:

$$s = 34562134 \cdot x$$

Von s sind nur die Vielfachen von 34562134 gültig. Die Anzahl der darstellbaren Werte ist 34562134 mal so groß wie die der gültigen Werte. Zu erwartende Verfälschungen werden nicht vorzugsweise auf Vielfache von 34562134 abgebildet. Erkennungswahrscheinlichkeit

$$p_{\mathrm{E}} \approx 1 - \frac{1}{34562134}$$

und damit fast eins. Bei sehr großen unbekannten Primzahlen als Multiplikatoren ist es selbst vorsätzlich kaum möglich, gültige Codeworte in andere gültige Codeworte zu verfälschen. Einsatz auch zur kryptographischen Verschlüsselung.

Zyklische Codes

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

Codierung

$$
\begin{array}{rr}
 & 10010101101 \\
\odot & 10011 \\
\hline
\oplus & 10010101101 \\
\oplus & 10010101101 \\
\oplus & 00000000000 \\
\oplus & 00000000000 \\
\oplus & 10010101101 \\
\hline
 & 10001110100111
\end{array}
$$

Decodierung

$$
\begin{array}{l}
10001110100111 : 10011 = 100101101 \\
\oplus\ 10011 \\
\quad 10110 \\
\quad \oplus\ 10011 \\
\quad\quad 10101 \\
\quad\quad 10011 \\
\quad\quad\ 11000 \\
\quad\quad \oplus\ 10011 \\
\quad\quad\quad 10111 \\
\quad\quad\quad \oplus\ 10011 \\
\quad\quad\quad\quad 10011 \\
\quad\quad\quad\quad \oplus 10011 \\
\text{Rest:}\quad 00000 \qquad \text{(unverfälscht)}
\end{array}
$$

Mathematisch werden die zu multiplizierenden Faktoren als Polynome dargestellt:

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

Eine Multiplikation mit x 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

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

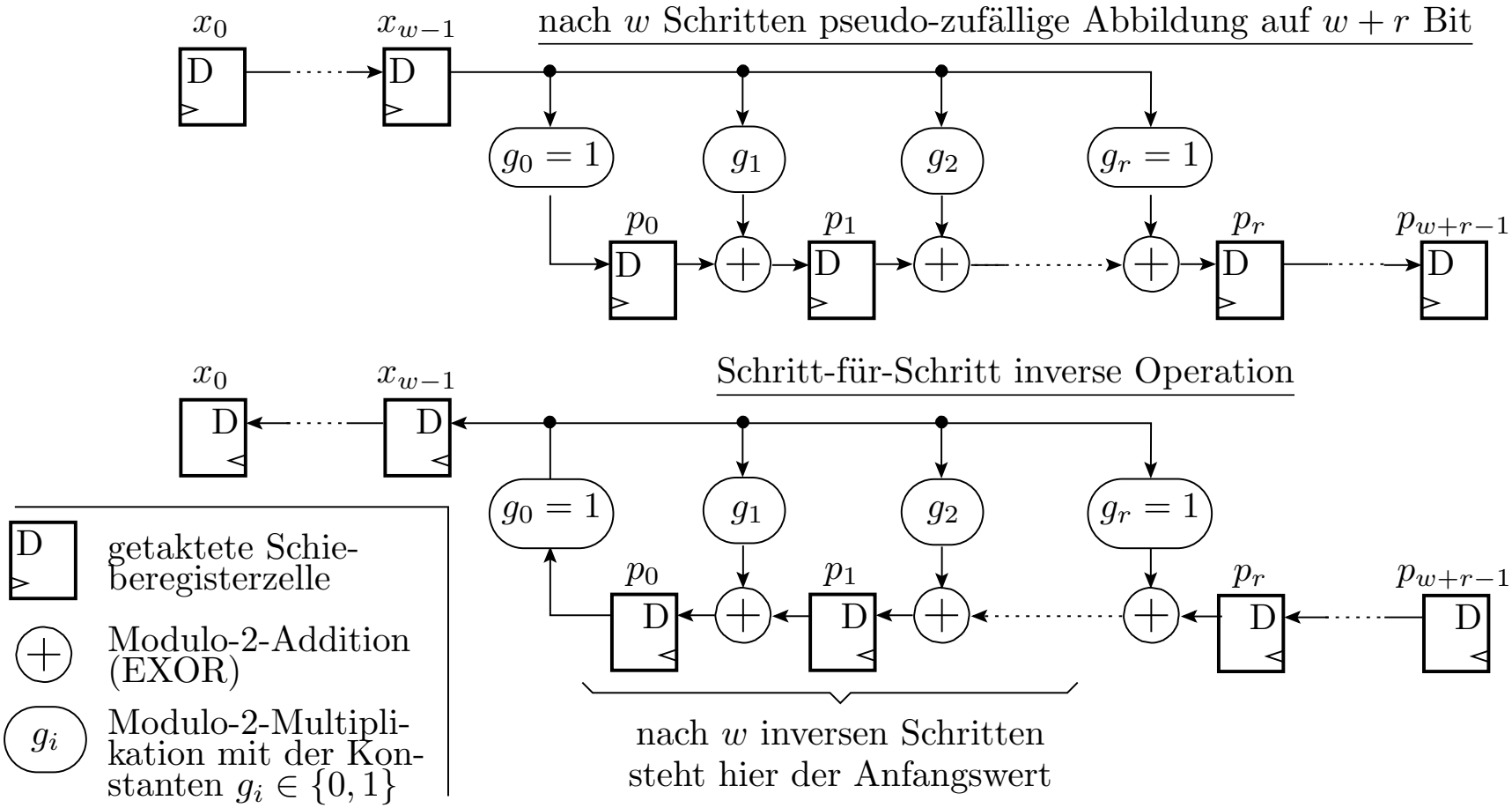
repräsentiert denselben Bitvektor, der für die Multiplikation der Folgen auf der Folie zuvor berechnet wurde. Die Polynomdivision:

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

liefert ohne Rest das Polynom der Originalfolge.

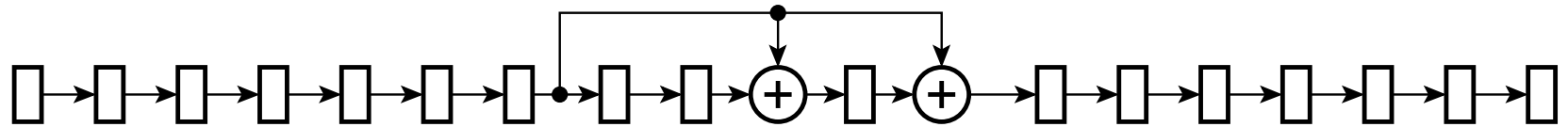
Linear rückgekoppelte Schieberegister

Die Codierung und Decodierung erfolgt mit Schieberegistern.



Erkennungswahrscheinlichkeit: $p_{\mathrm{E}} = 1 - \frac{2^w}{2^{w+r}} = 1 - 2^{-r}$

Beispiel für die Codierung



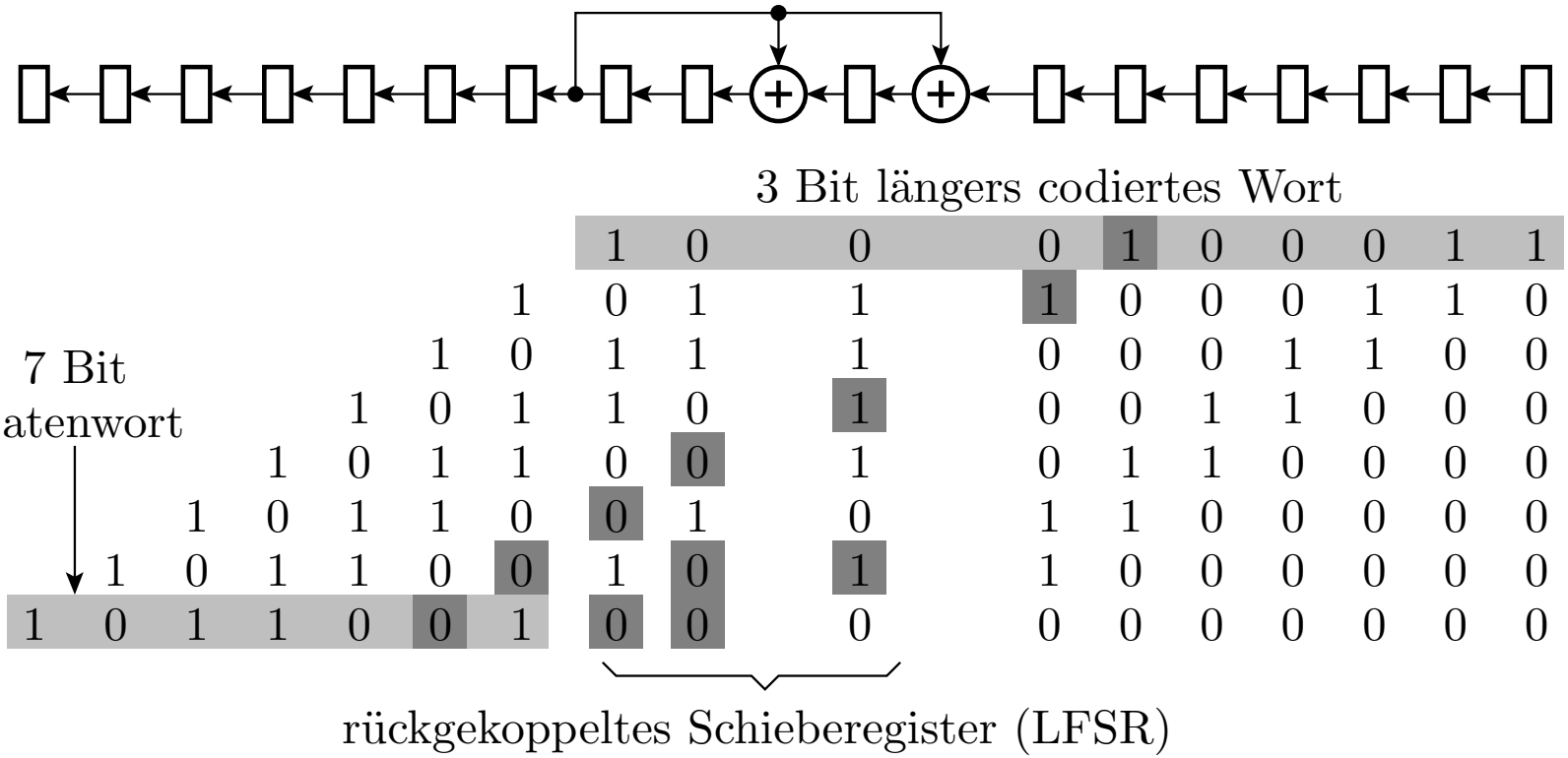
1	0	1	1	0	0	1	0	0		0		0	0	0	0	0	0	0
	1	0	1	1	0	0	1	0		1		1	0	0	0	0	0	0
		1	0	1	1	0	0	1		0		1	1	0	0	0	0	0
			1	0	1	1	0	0		1		0	1	1	0	0	0	0
				1	0	1	1	0		1		0	0	1	1	0	0	0
					1	0	1	1		1		0	0	0	1	1	0	0
						1	0	1		1		1	0	0	0	1	1	0
							1	0		0		0	1	0	0	0	1	1

Datenwort
7 Bit

3 Bit längeres codiertes Wort

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

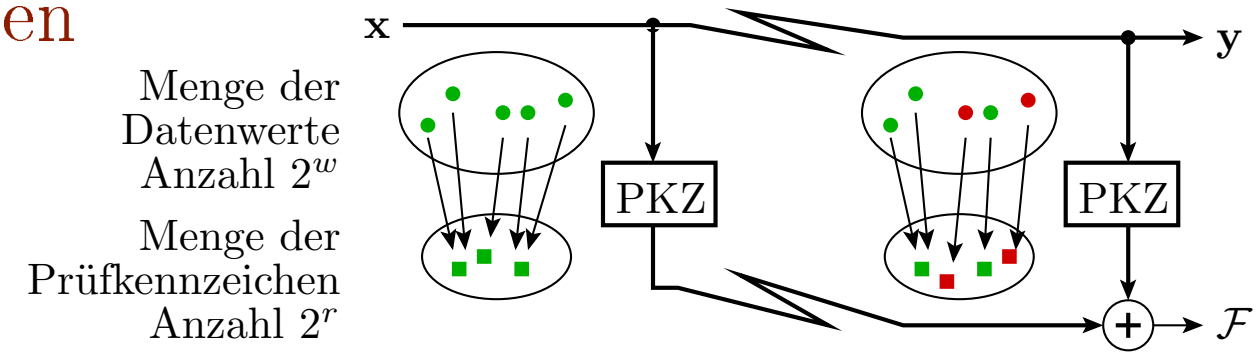
Rückgewinnung und Kontrolle



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

Prüfkennzeichen

Prüfkennzeichen



- Jedem w-Bit-Datenwort wird pseudo-zufällig genau eines der r-Bit-Prüfkennzeichen 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}. Erkennungswahrscheinlichkeit:

$$p_{\mathrm{E}} \approx 1 - \frac{2^w}{2^{w+r}} = 1 - 2^{-r} \tag{3}$$

Prüfsummen

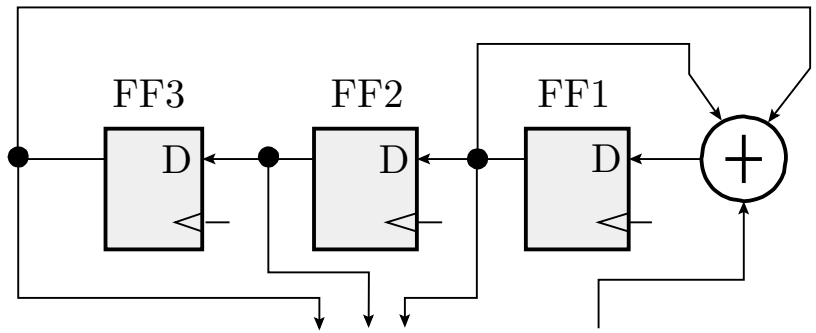
Prüfkennzeichenbildung durch Aufsummierung (arithmetisch, bitweises EXOR, ...).

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

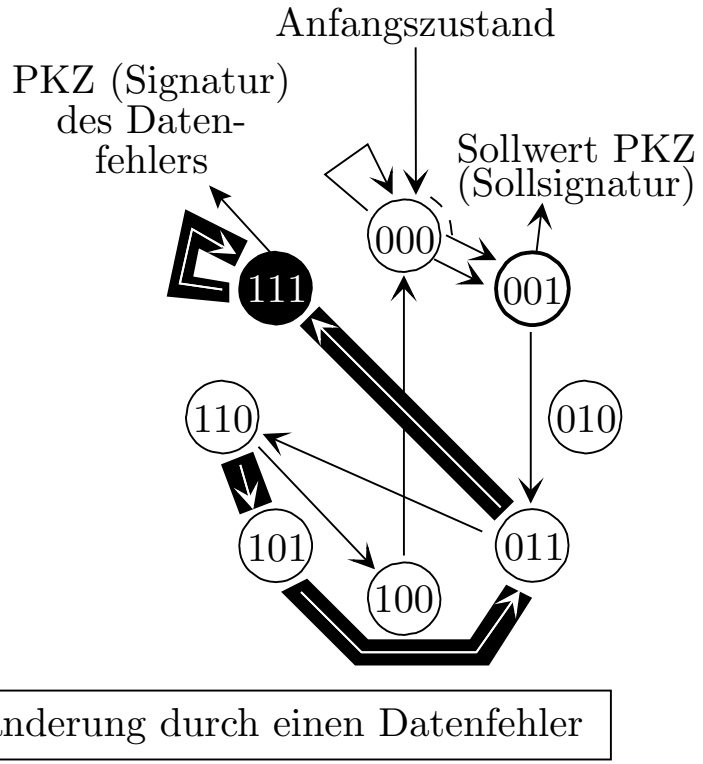
Bei »einfacher Genauigkeit« und »bitweisem EXOR« erscheint die Annahme »pseudo-zufällige Abbildung« gerechtfertigt[3]: $p_{\mathrm{E}} \approx 1 - 2^{-4}$. Bei »doppelter Genauigkeit« bilden sich Verfälschungen vorzugsweise auf die niederwertigen Bits ab. Maskierungswahrscheinlichkeit: $2^{-4} > 1 - p_{\mathrm{E}} \gg 2^{-8}$.

[3] Kein Nachweis für vertauschte Summationsreihenfolge.

Prüfkennzeichenbildung mit LFSR[4]



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	

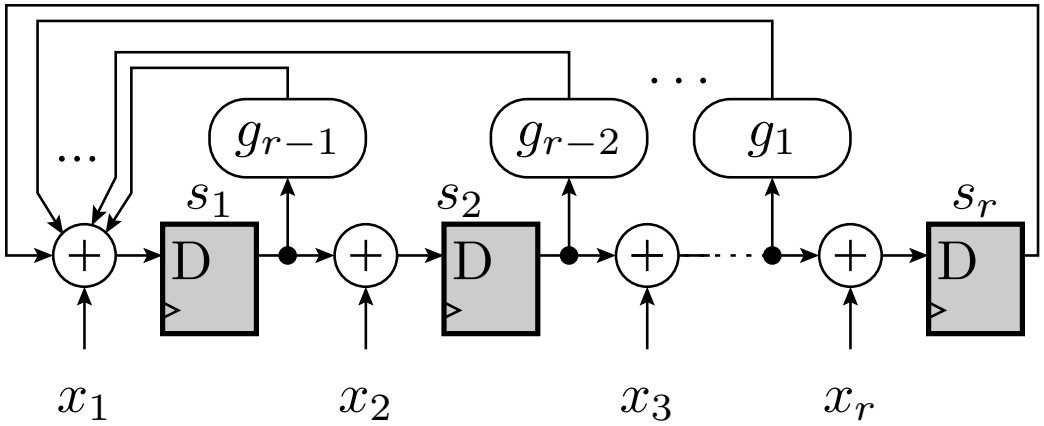


Das Prüfkennzeichen wird wie bei der CRC-Decodierung mit einem linear rückgekoppelten Schieberegister (LSFR) gebildet. Im Beispiel hat das LSFR im Gegensatz zur Polynomdivision Folie 23 eine zentrale Rückführung. Abbildung auch pseudo-zufällig.

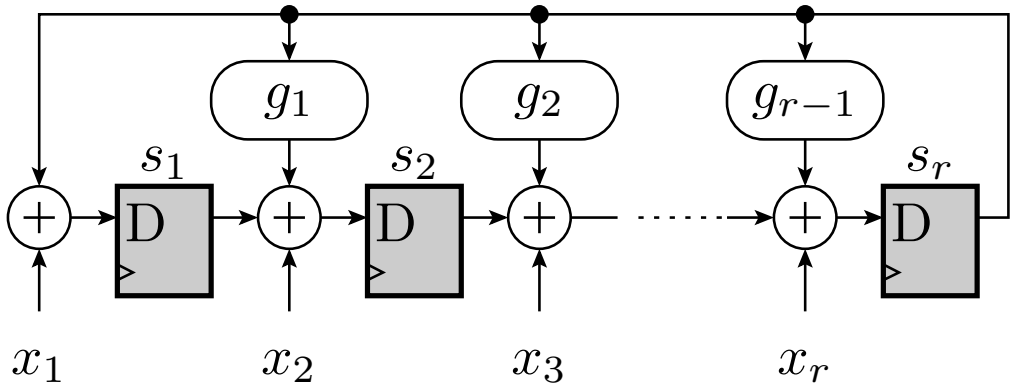
[4] LFSR – **L**inear **F**eeback **S**hift **R**egister.

LFSR für parallele Datenströme

Für die Bildung auf Prüfkennzeichen ist es nur wichtig, dass die Abbildung pseudo-zufällig hinsichtlich der zu erwartenden Verfälschungen erfolgt. Diese Eigenschaft hat auch ein rückgekoppeltes Schieberegister, bei dem die Daten modulo-2 als Bitvektoren zu den Registerzuständen addiert werden (paralleles Signaturregister).



Die Rückführung darf dabei auch wie bei der Polynom-Division dezentral sein.



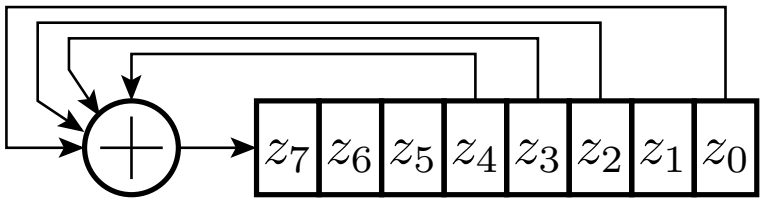
Die Koeffizienten g_i der Rückführung, bei der Polynom-Division das Divisor-Polynom, bestimmen die autonome Zyklusstruktur[5]. Die autonome Zyklusstruktur ist bei zentraler und dezentraler Rückführung mit denselben Rückführkoeffizienten gleich. Bevorzugt werden lange Zyklen, insbesondere sog. primitive Polynome, bei denen alle Zustände außer »alles null« einen $2^r - 1$ langen Maximalzyklus bilden. Gebräuchliche Rückführungen:

- USB (CRC-5): 5 Bit, nur $g_2 = 1$
- Ethernet (CRC-32, IEEE802.3): 32 Bit, $g_{26} = g_{23} = g_{22} = g_{16} = g_{12} = g_{11} = g_{10} = g_8 = g_7 = g_5 = g_4 = g_2 = g_1 = 1$

[5] Zyklusstruktur ohne Eingaben.

Autonome Zykluslänge und -struktur von LFSR

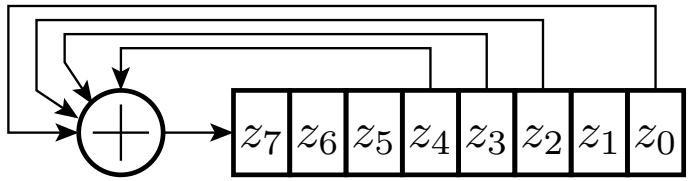
Autonom bedeutet Zyklusstrukur für Eingabewerte »alle 0« oder für LFSR ohne Eingänge. Wichtig für die Erzeugung wiederholungsfreier Pseudo-Zufallsfolgen. Beispiel 8-Bit autonomes LFSR:



0	0	1	1	0	0	0	1
0	0	0	1	1	0	0	0
0	0	0	0	1	1	0	0
0	0	0	0	0	1	1	0
1	0	0	0	0	0	1	1
1	1	0	0	0	0	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Übergangsfunktion:

$$\begin{aligned} z_7 &= z_4 \oplus z_3 \oplus z_2 \oplus z_0 \\ z_i &= z_{i+1} \text{ für } i \in \{0, 1, 2, \ldots 6\} \end{aligned}$$



Bestimmen der Zyklusstruktur durch Simulation (z.B. C-Prograann):

```
...
#define ZA 0x31
uint8_t z = ZA;          // Startwert setzen
while(1){
  z = (z>>1) ^ (z<<7) ^ ((z<<5)&0x80)
      ^ ((z<<4)&0x80) ^ ((z<<3)&0x80);
  <Ausgabe von z>
  if (z==ZA) 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.

Experiment Fehlererkennungssicherheit von LFSR

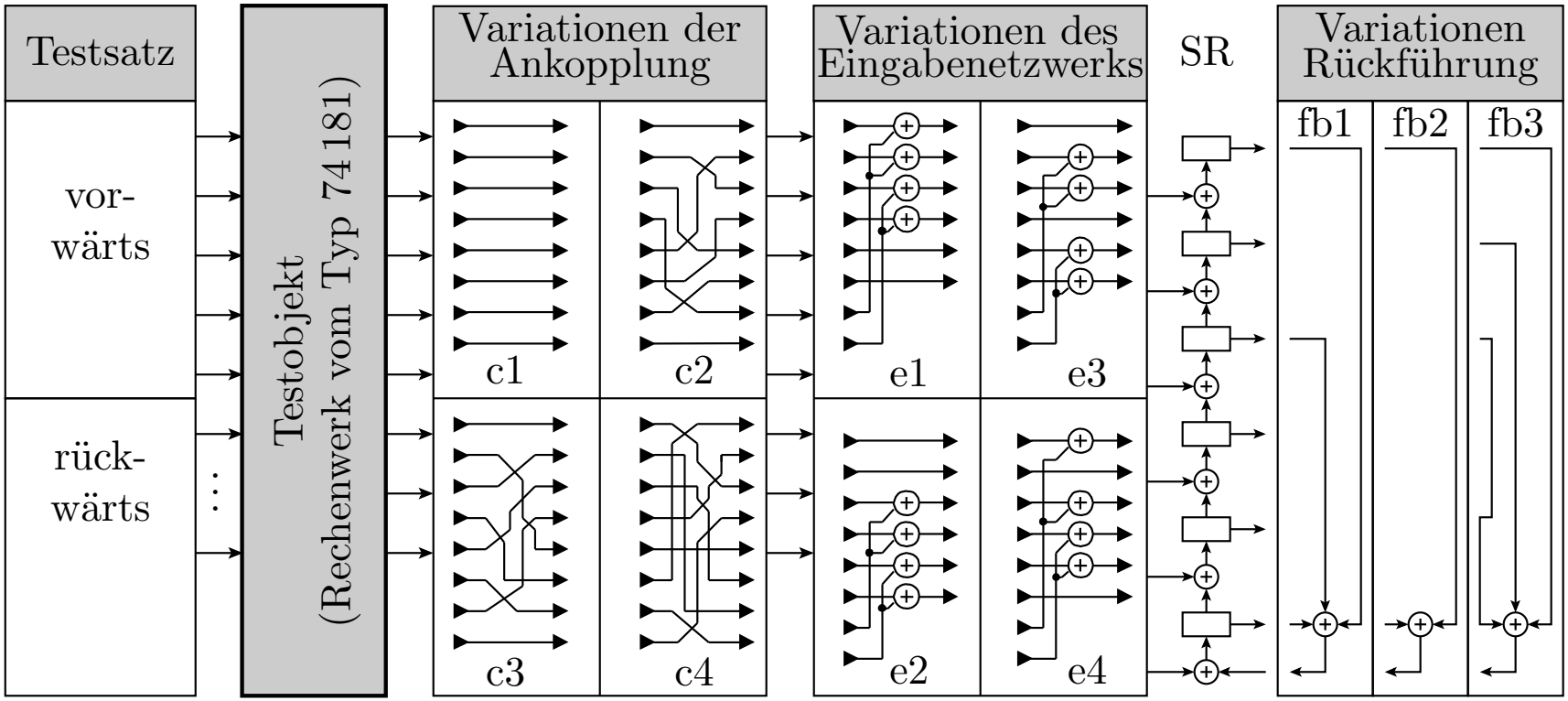
Es ist schwer zu glauben, dass

- mit r-Bit Prüfkennzeichen beliebige Verfälschung mit einer Wahrscheinlichkeit $p_{\mathrm{E}} = 1 - 2^{-r}$ erkannt werden und
- die Schaltungsstruktur, die Rückführung etc. kaum Einfluss auf die Erkennungswahrscheinlichkeit haben sollen.

Deshalb ein Experiment:

- Simulation einer Schaltung (4-Bit-Rechenwerk) mit einem Testsatz und 250 verschiedenen Haftfehlern. Berechnung des Prüfkennzeichens für jeden Fehler.
- Variation der Testsatzreihenfolge,
- Variation der Ankopplung an das LFSR und
- Variation der Rückführung.

Zählen der nachweisbaren Fehler für jede Konfiguration.



Aus $r = 6$ bit folgt, dass jeder Fehler mit einer Wahrscheinlichkeit $p_{\mathrm{E}} = 1 - 2^{-6} = 98{,}44\%$ erkenn- und mit einer Wahrscheinlichkeit $p_{\mathrm{F}} = 2^{-6} = 1{,}36\%$ nicht erkennbar sein müsste. Definition einer Zufallsgröße X_i zum Zählen der nicht erkennbaren Fehler:

$$\begin{aligned} P(X_i = 0) &= 1 - 2^{-6} \text{ Fehler } i \text{ nachweisbar} \\ P(X_i = 1) &= 2^{-6} \text{ Fehler } i \text{ nicht nachweisbar} \end{aligned}$$

Wenn die Theorie stimmt, müsste die Anzahl der maskierten Fehler

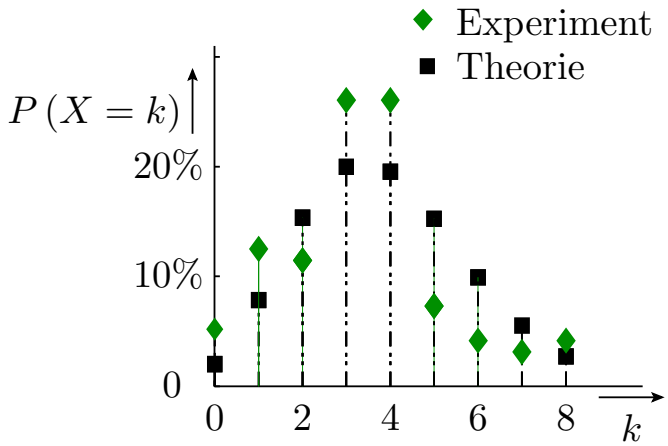
$$X = \sum_{i=1}^{250} X_i$$

binomialverteilt sein (siehe später Foliensatz 6):

$$P(X = k) = \left(\begin{array}{c} N \\ k \end{array} \right) \cdot 2^{-r \cdot k} \cdot (1 - 2^r)^{N-k}$$

Anzahl der maskierten Fehler

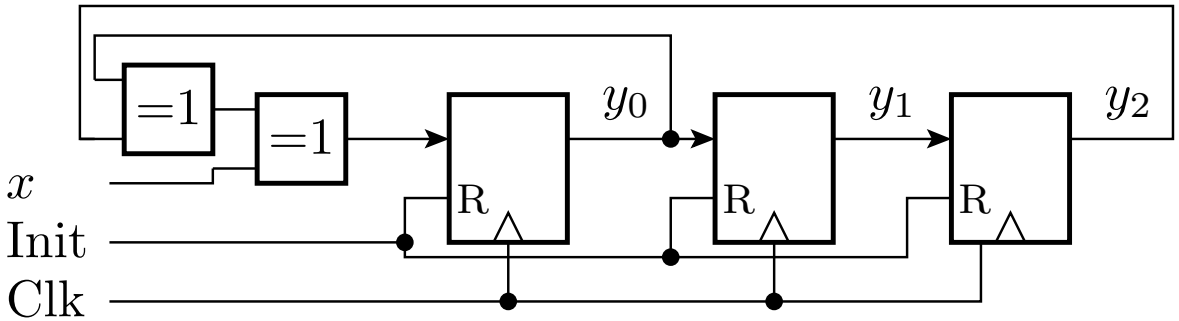
		e1				e2				e3				e4			
		c1	c2	c3	c4	c1	c2	c3	c4	c1	c2	c3	c4	c1	c2	c3	c4
vor-wärts	fb1	3	4	1	2	3	4	3	3	4	2	4	3	4	3	4	6
	fb2	3	4	1	7	2	2	1	4	2	1	1	3	2	5	3	7
	fb3	5	2	2	8	4	5	3	4	3	6	3	7	5	3	3	4
rück-wärts	fb1	6	4	4	2	3	4	3	4	3	4	3	4	4	8	4	5
	fb2	2	0	0	1	4	1	4	1	0	0	0	1	1	1	4	1
	fb3	2	4	3	4	4	8	5	8	3	3	3	6	3	3	4	3



Abweichungen zwischen Vorhersage und Experiment nicht signifikant. Hypothese $p_F = 2^{-6}$ bestätigt.

Beispielaufgabe

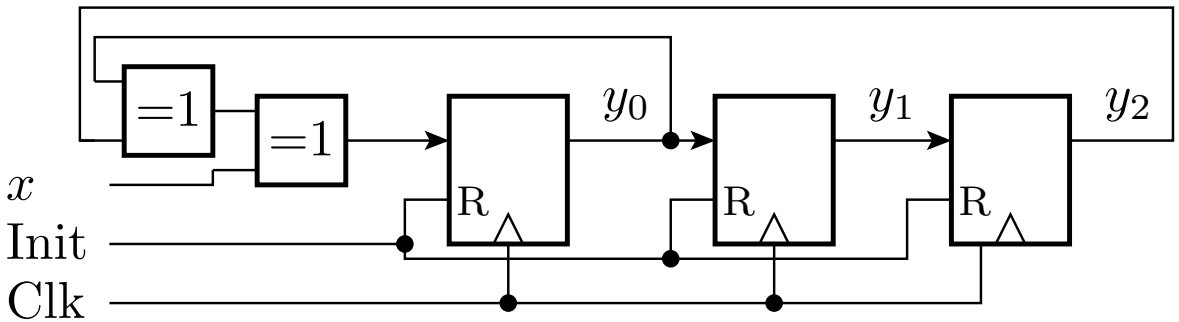
Gegeben ist folgendes linear rückgekoppelte Schieberegister:



	x	y_2	y_1	y_0
0	1	0	0	0
1	0			
2	0			
3	1			
4	1			
5	0			
6	0			
7	1			
8	0			
9	1			
10	1			
11	1			
12	1			
13	0			
14	1			
15	0			
PKZ:				

1. Welches Prüfkennzeichen $\mathbf{y} = y_2 y_1 y_0$ hat die Datenfolge »1001100101111010« bei Abbildung beginnend mit dem höchstwertigen Bit. Startwert 000.
2. Wie hoch ist Fehlererkennungswahrscheinlichkeit?

Lösung



	x	y_2	y_1	y_0
0	1	0	0	0
1	0	0	0	1
2	0	0	1	1
3	1	1	1	1
4	1	1	1	1
5	0	1	1	1
6	0	1	1	0
7	1	1	0	1
8	0	0	1	1
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	1	1	0
15	0	1	0	0
PKZ:		0	0	1

Erkennungswahrscheinlichkeit:

$$p_{\mathrm{E}} \approx 1 - 2^{-3} = 87{,}5\%$$

Zusammenfassung

Datensicherung mit fehlererkennenden Codes / Prüfkennzeichen:

- Geringer Berechnungsaufwand.
- Geringer Zusatzaufwand für Datenübertragung und -speicherung (r zusätzlich gespeicherte / übertragene Bits für eine Datenobjekt beliebiger Größe).
- Maskierungswahrscheinlichkeit 2^{-r}. Mit ausreichendem r immer vernachlässigbar klein.

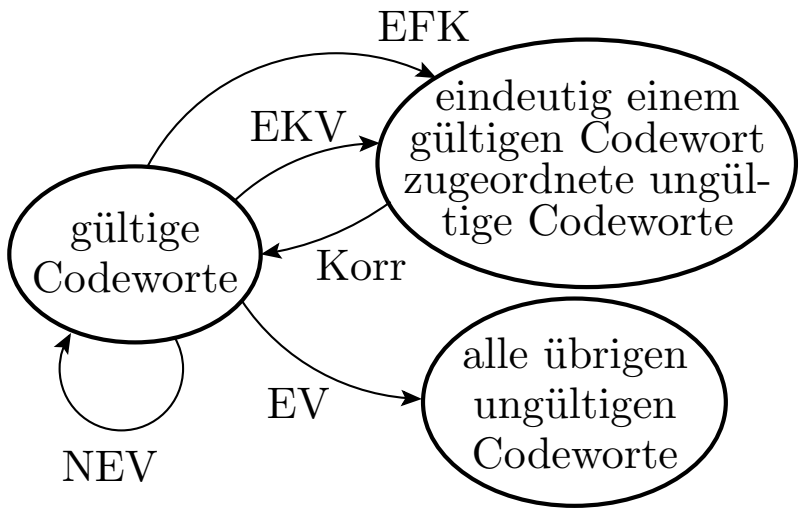
Dateien, Nachrichten etc. werden sehr oft mit einem Prüfkennzeichen übertragen und gespeichert. Programmierte PKZ-Bildung und -Kontrolle bevorzugt »Prüfsumme«.
Hardware-PKZ-Bildung und -Kontrolle bevorzugt »LFSR«.

Fehlererkennende Codes sind weniger gebräuchlich.

Fehlerkorr. 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:

$$2^{N_{\mathrm{Bit}}} \geq N_{\mathrm{CWG}} + N_{\mathrm{CWG}} \cdot N_{\mathrm{CWK/CWG}}$$

(N_{CWG} – Anzahl gültige Codeworte; $N_{\mathrm{CWK/CWG}}$ – Anzahl korrigierbare Codeworte je gültiges Codewort). Die Erkennungswahrscheinlichkeit als Anteil der übrigen ungültigen Codeworte verringert sich durch Korrekturmöglichkeiten.

Beispiel: Korrektur von Einzelbitfehler

Anzahl korrigierbare Codeworte je gültiges Codewort gleich Bitanzahl:

$$N_{\mathrm{CWK/CWG}} = N_{\mathrm{Bit}}$$

Mindestbitanzahl:

$$2^{\mathrm{N_{Bit}}} \geq N_{\mathrm{CWG}} + N_{\mathrm{CWG}} \cdot N_{\mathrm{Bit}}$$

Für $\mathrm{N_{CWG}} = 256$ gültige Codeworte:

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

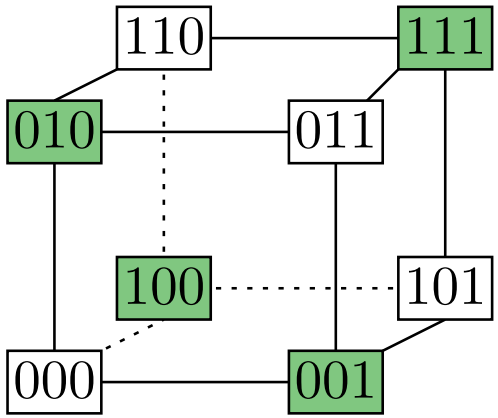
Probe:

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

Hamming-Codes

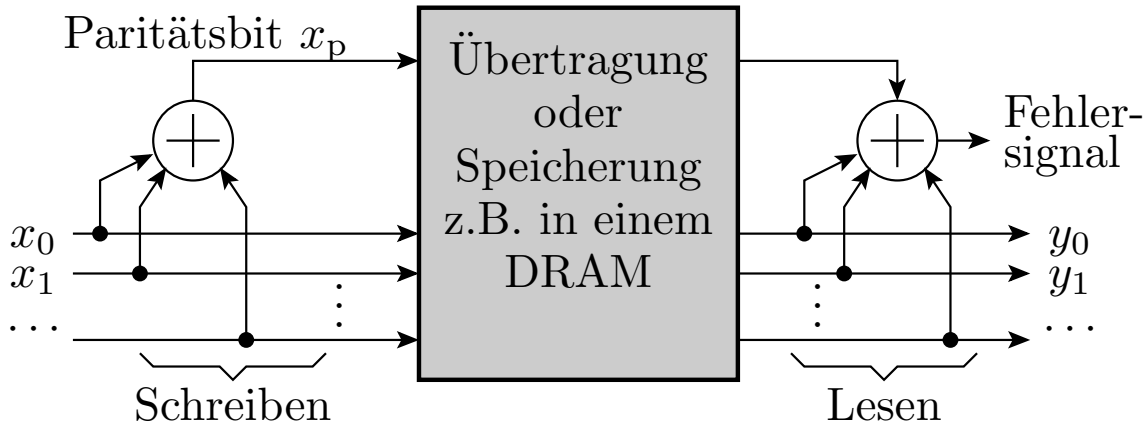
Hamming-Distanz

Die Hamming-Distanz N_{Ham} ist die Anzahl der Bitpositionen, in denen sich zwei Codeworte unterscheiden. Distanz von 2 oder mehr garantiert, dass ein 1-Bit Fehler nicht zu einem anderen gültigen Codewort führt.



- Erkennen von k-Bit Fehlern verlangt eine Hamming-Distanz von mindestens $N_{\text{Ham}} \geq k + 1$.
- Um k-Bit-Fehler korrigieren zu können, ist eine Hamming-Distanz von $N_{\text{Ham}} \geq 2k + 1$ erforderlich.

Parität als Prüfkennzeichen (Hamming-Distanz 2)



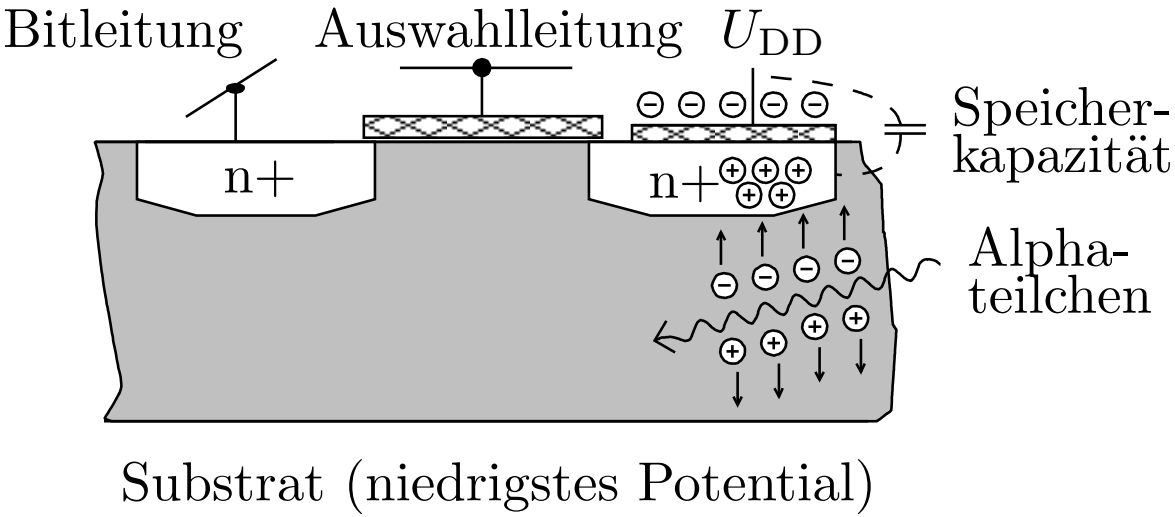
Einzelprüfbit, modulo-2 Summe (EXOR-Verknüpfung):

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

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

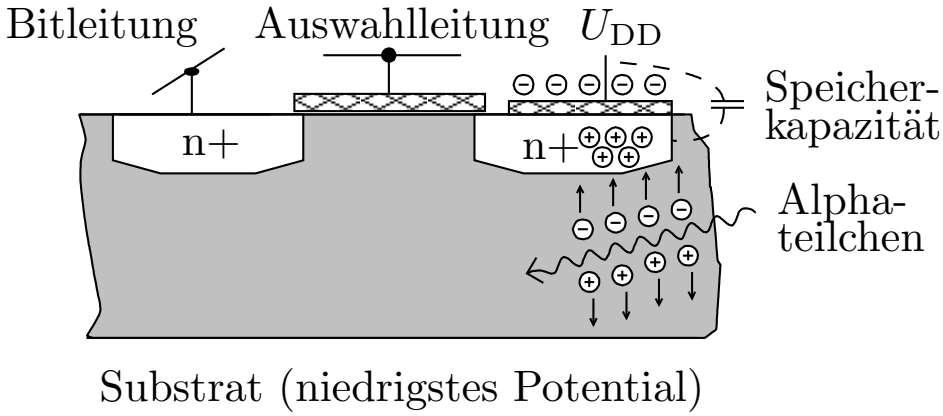
- Erkennt jede ungeradzahlige Anzahl von Bitverfälschungen.
- Wenn geradzahlige und ungeradzahlige Bitfehler gleichhäufig auftreten: $p_\mathrm{E} \approx 50\%$
- Überwiegend Einzelbitfehler: $p_\mathrm{E} \gg 50\%$

Paritätstest für DRAMs und Speicherriegel



- Informationsspeicherung in winzigen Kapazitäten.
- Häufigste Ursache für Datenverfälschungen: Alphastrahlung.
- Deren Quellen radioaktiver Zerfall von Uran und Thorium, die als Spurenelemente im Gehäuse und im Aluminium der Leiterbahnen enthalten sind, und Kernprozesse im Silizium durch Höhenstrahlung. Seltene Ereignisse.

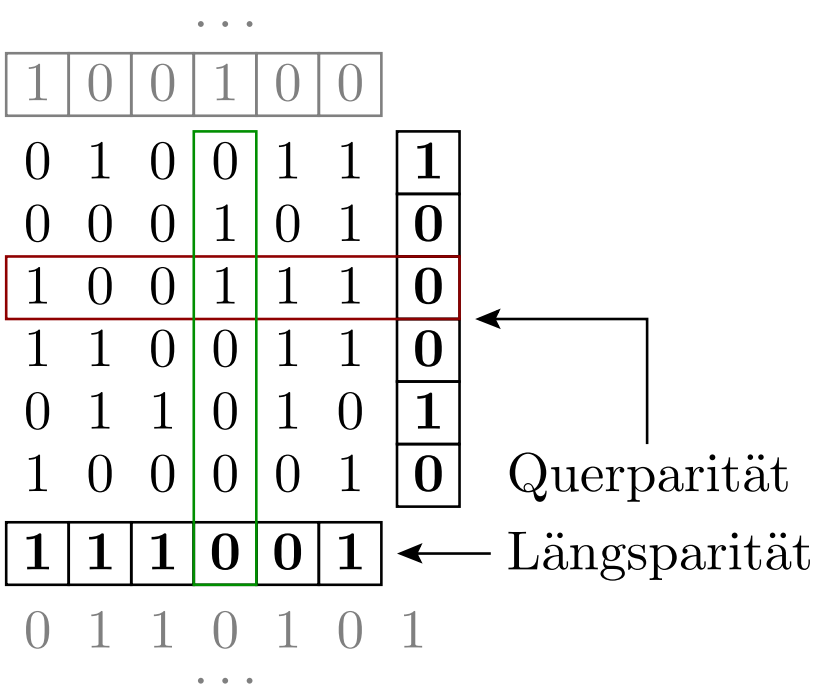
- Energie eines Alphateilchen: $5\,\mathrm{MeV}$. Energieverlust bei der Generierung eines Elektronen-Loch-Paares $\approx 3{,}6\,\mathrm{eV} \Rightarrow$ Generierung von $\approx 10^6$ Ladungsträgerpaaren. Reichweite $\approx 89\,\mu\mathrm{m}$. gespeicherte Ladung $\approx 10^5$ Ladungsträger. Datenverlust einer oder mehrerer benachbarter Zellen möglich.



- Mittlerer Zeitabstand zwischen zwei Datenverfälschungen Stunden. Gleichzeitige Verfälschung durch zwei Alphateilchen unwahrscheinlich.
- Geometrische Trennung der Zellen eines Datenworts (getrennte Schaltkreise oder Speichermatritzen) $\Rightarrow$ Einzelbitverfälschung je gelesenes Datenwort.
- 100%iger Nachweis durch Paritätskontrolle.

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).



Beispielaufgabe

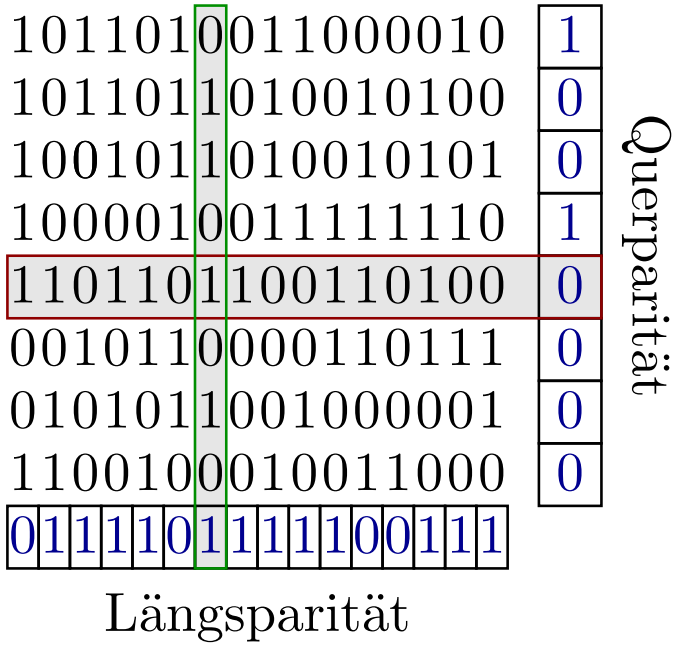
Kontrollieren Sie für die nachfolgenden Bitfelder mit Kreuzparität, ob eine erkennbare oder eine erkenn- und korrigierbare Verfälschung vorliegt und führen Sie, wenn möglich, die Korrektur durch.

1011010011000010	1
1011011010010100	0
1001011010010101	0
1000010011111110	1
1101101100110100	0
0010110000110111	0
0101011001000001	0
1100100010011000	0
0 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1	

Querparität

Längsparität

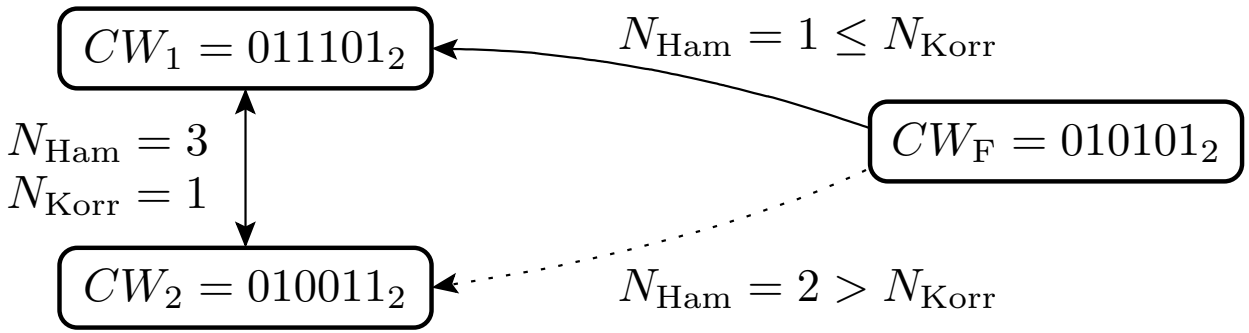
Lösung



Korrektur: Bit in Zeile 5, Spalte 7 invertieren (null setzen).

1-Bit fehlerkorrigierende Hamming-Codes

Ab einem Hamming-Abstand $N_{\text{Ham}} \geq 3$ ist jeder 1-Bit-Verfälschung eindeutig einem gültiges Codewort zugeordnet.



CW_i	gültige Codewörter
CW_{F}	verfälschtes Codewort
N_{Ham}	Hamming-Abstand

Korrektur durch Ersatz des verfälschten Codeworts durch das mit Hamming-Distanz $N_{\text{Ham}} = 1$. Bei Hamming-Distanz $N_{\text{Ham}} = 3$ werden 2-Bit-Verfälschungen bei der Korrektur falschen gültigen Codeworten zugeordnet.

Konstruktion eines 1-Bit fehlerkorrigierenden Codes

Zusammensetzen des Gesamtcodeworts aus

- einem Datenwort mit minimaler Bitanzahl

$$w \geq \log_2 (N_{\mathrm{CWG}})$$

und einem Prüfkennzeichen der Bitanzahl

$$r = N_{\mathrm{Bit}} - w$$

das aus dem Datenwort mit mod-2 Summen berechnet wird.

- Wahl der mod-2 Summen so, dass bei einer Bitverfälschung die mod-2 Summe des gesendeten und empfangenen Prüfzeichens die binärcodierte Bitnummer der Verfälschung ist:

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

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

Beispiel: $w = 8$, $r = 4$, $\mathbf{q} = q_3 q_2 q_1 q_0$

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

Das erste Bit jeder Summe sei das Prüfbit q_i. Die restlichen sind Datenbits. Ohne Verfälschung ist die Differenz null.

Beispielzuordnung:

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

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

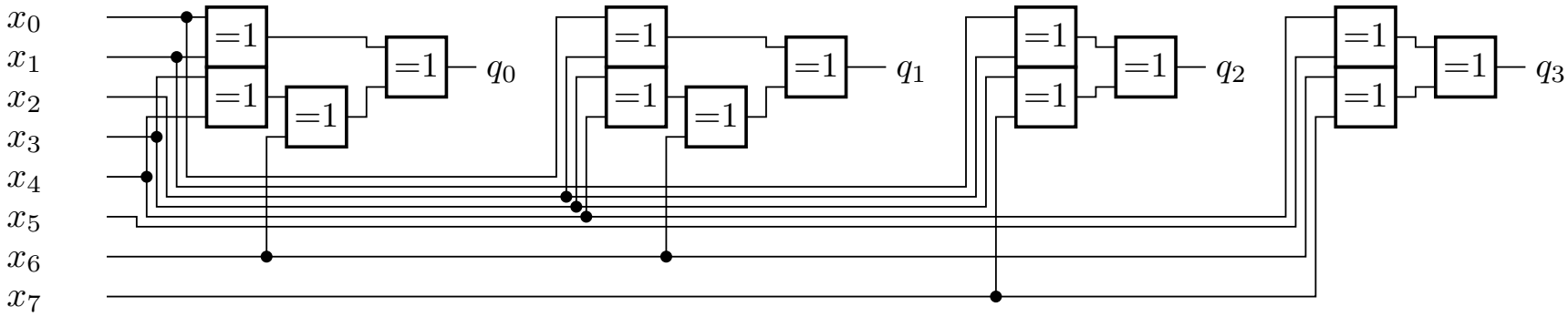
Für die erste Summe gilt:

$$\begin{aligned} \Delta q_0 &= b_1 \oplus b_3 \oplus b_5 \oplus b_7 \oplus b_9 \oplus b_{11} \\ 0 &= q_0 \oplus x_0 \oplus x_1 \oplus x_3 \oplus x_4 \oplus x_6 \\ q_0 &= x_0 \oplus x_1 \oplus x_3 \oplus x_4 \oplus x_6 \end{aligned}$$

Wie lauten die Bildungsregeln für q_1 bis q_3? (an der Tafel anhand der Folie zuvor herleiten)

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

Codierschaltung:



Die Korrekturschaltung besteht aus demselben Coder wie zur Bildung von $\mathbf{q} = q_3 \ldots q_0$. Vergleich durch bitweises EXOR des empfangenen und des im Empfänger gebildeten Prüfzeichens. Invertierung des verfälschten Bits.

Bitfehlerkorrektur als VHDL-Case-Anweisung:

```
case (q_Empf xor q_berechnet) is
 when "0011" => x(0) <= not x(0);
 when "0101" => x(1) <= not x(1);
  ...
 when others => null;
end case;
```

Beispielaufgabe

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

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

1. Bilden Sie für den Werte $w_1 = 0x8B$ das Codewort.
2. Bestimmen Sie für das Codewort $c_2 = 0xA9B$ den Wert.

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													
$w_1 = 0x8B$													$c_1 =$
$c_2 = 0xA9B$													$d\,q_2 =$

Lösung

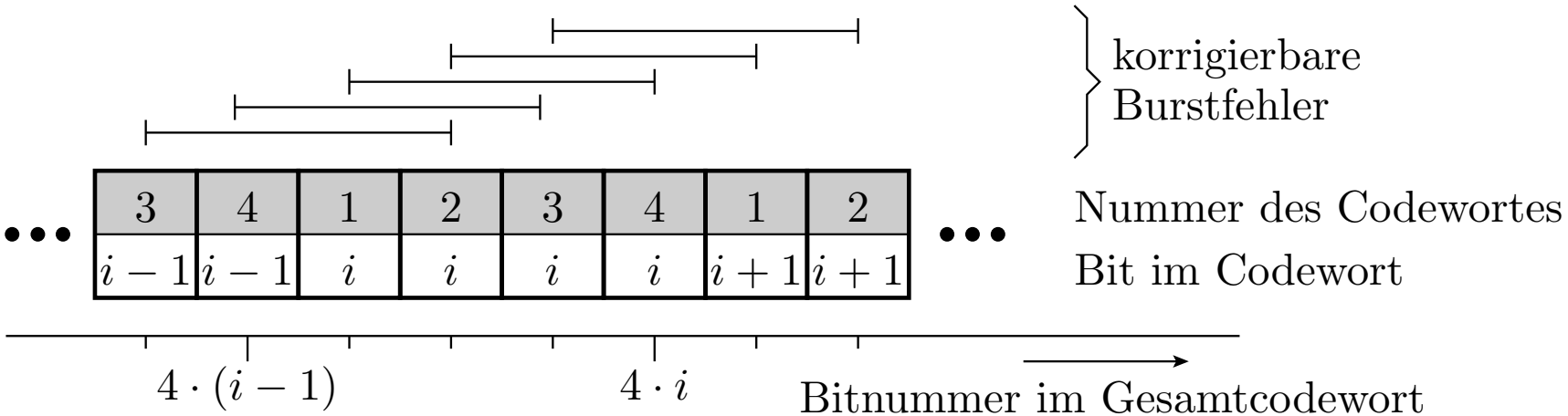
Bitnummer	12	11	10	9	8	7	6	5	4	3	2	1	
Zuordnung	x_7	x_6	x_5	x_4	q_3	x_3	x_2	x_1	q_2	x_0	q_1	q_0	
Kontrollbits													
w_1= 0x8B	1	0	0	0	1	1	0	1	1	1	0	1	c_1= 0x8DD
c_2= 0xA9B	1	0	1	0	1	0	0	1	1	0	1	1	$d\,q_2 = 12_{10}$

w_2 : Wert 0xA2 mit verfälschtem $x_7 \Rightarrow w_2$ =0x22

Korrektur von Burstfehlern

- Bei der Datenübertragung, beim Lesen von CDs, ... ist oft eine Folge aufeinanderfolgender Bits verfälscht.
- Burst-Fehler: In einer Bitfolge sind an einer Stelle bis zu m aufeinanderfolgende Bits verfälscht.

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



Beispielaufgabe

1. Codierung Datenfolge 0x8B, 0x22, 0x9C so, dass bis zu 3-Bit lange Burstfehler korrigierbar sind, durch Verschränkung von je drei aufeinanderfolgenden H8-12-Codeworten.
2. 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												
w_1= 0x8B												
w_2= 0x22									1			
w_3=0x9C												

Bits 30 bis 32

Lösung

1

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												
w_1= 0x8B	1	0	0	0	1	1	0	1	1	1	0	1
w_2= 0x22	0	0	1	0	1	0	0	1	1	0	1	1
w_3=0x9C	1	0	0	1	0	1	1	0	1	0	0	0

2

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													
w_1= 0x8B	1	0	0	0	1	1	0	1	1	1	0	1	$d\,q_1 = 11_{10}$
w_2= 0x22	0	0	1	0	1	0	0	1	1	0	1	1	$d\,q_2 = 11_{10}$
w_3=0x9C	1	0	0	1	0	1	1	0	1	0	0	0	$d\,q_3 = 10_{10}$

Durch Burstfehler invertierte Bits 30 bis 32,

RAID Systeme

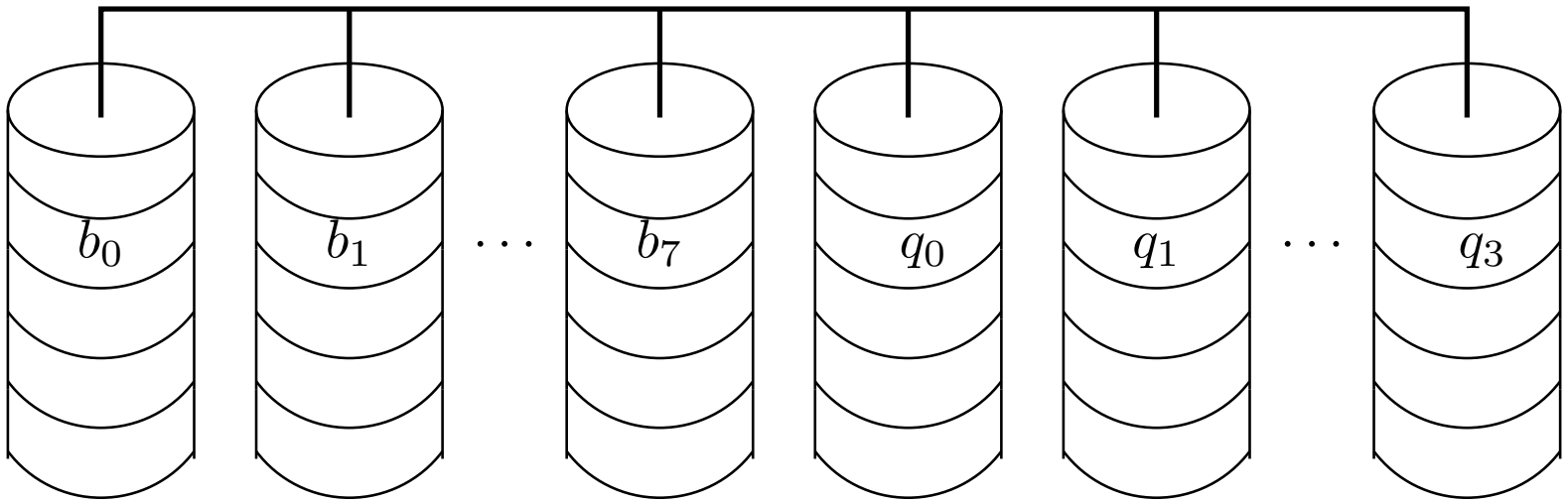
RAID, RAID Level 1

RAID – **R**edundant **A**rray of **I**ndependent **D**isks. Anwendung der behandelten Codes zur Korrektur bei Datenspeicherung auf Festplatten.

RAID Level 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 Level 2

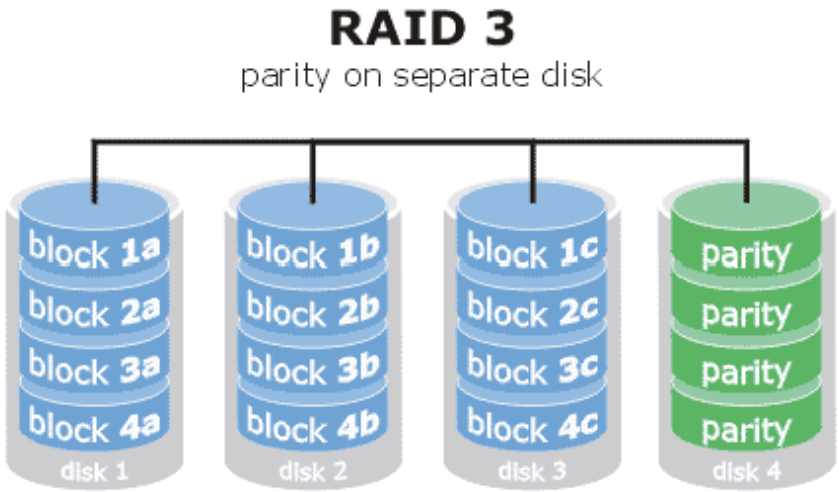
Bei RAID Level 2 werden die Daten in einem 1-Bit-korrigierenden Hamming-Code gespeichert, und zwar jedes der w Daten- und der r Kontrollbits auf einer anderen Platte, z.B. $w = 8$ Datenbit- und $r = 4$ Kontrollbitplatten. Im Vergleich zu RAID 1 werden statt der doppelten Plattenanzahl nur 50% mehr Platten benötigt.



Gilt als aufwändig und ungebräuchlich.

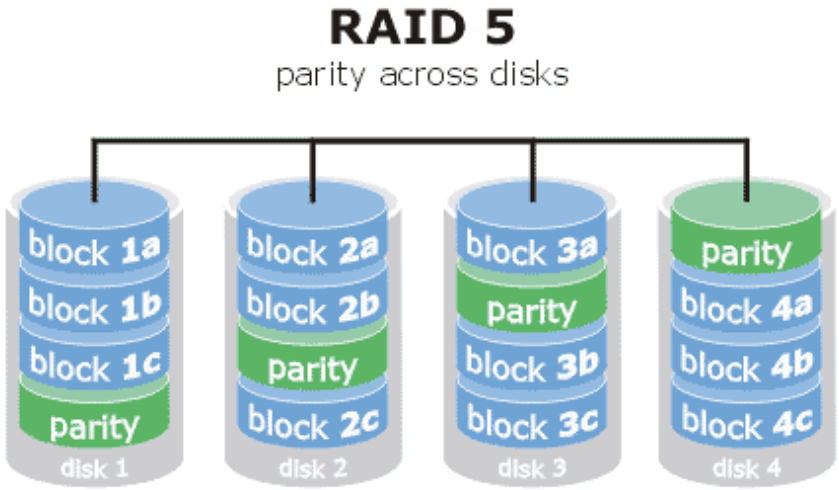
RAID Level 3

Auf einer Extra-Platte wird bitweise die Querparitätsbit der anderen Platten gebildet. Zusätzlich werden auf jeder Platte die Längsparitätsbits (oder Prüfkennzeichen) gespeichert.
1-Bit-Fehlerkorrektur nach dem Prinzip der Kreuzparität. Erlaubt die Tolerierung eines einzelnen Plattenausfalls.



RAID Level 5

Fehlertoleranz ähnlich wie Level 3, nur dass Datenzugriffe durch unabhängige Lese- und Schreiboperationen (statt ausschließlich parallel) erlaubt sind. Größere schreibbare Datenblöcke. Die Paritätsinformation verteilt sich auf alle Platten. Gleichfalls tolerant gegenüber einem einzelnen Plattenausfall. Am häufigsten genutzte RAID-Struktur.



RAID ist kein Backup-Ersatz

Backup: Sicherungskopien von (wichtigen / aufwändig neu zu erzeugenden) Daten. Typisch:

- Tägliche automatische Erstellung durch das Rechenzentrum.
- Nur Änderungen zum letzten Backup.
- Aufbewahrung mehrerer Versionen an einem getrennten Ort.

Wird benötigt zur Datenwiederherstellung nach

- gleichzeitiger Zerstörung aller Platten z.B. durch Überspannungsspitzen, Feuer, ...
- Diebstahl von Datenträgern,
- einem versehentlichen Löschen, das erst nach Stunden oder Wochen bemerkt wird.

Formatkontrollen

Formatkontrollen

Kontrollier- und überwachbare Merkmale von Service-Ein-und Ausgaben:

- Format: Konstante, immer erfüllte Merkmale (Zeitraster, ...).
- Daten: Variable Merkmale (Werte von Datenobjekten, ...).

Prüfkennzeichen und fehlererkenne Codes gehören zu den Formaten. Erlauben Überwachung mit sehr hohen Erkennungsrate. Leider nicht für alle SL geeignet. Weitere kontrollierbare Formateigenschaften:

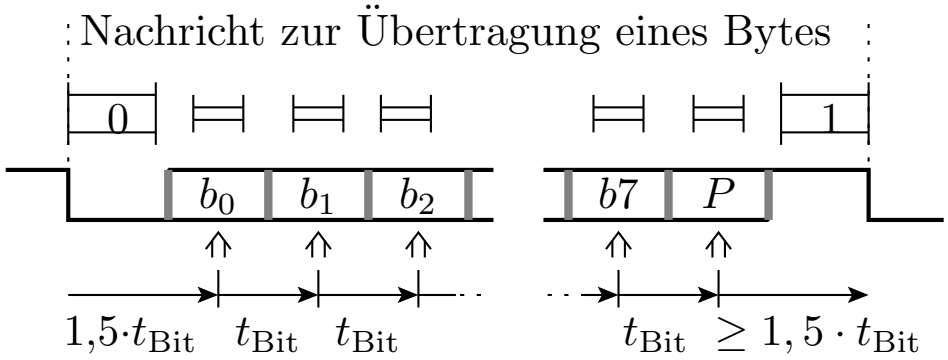
- Syntax, Wertebereiche,
- Signalmerkmale,
- Toleranzbereiche für Werte und Zeiten, ...

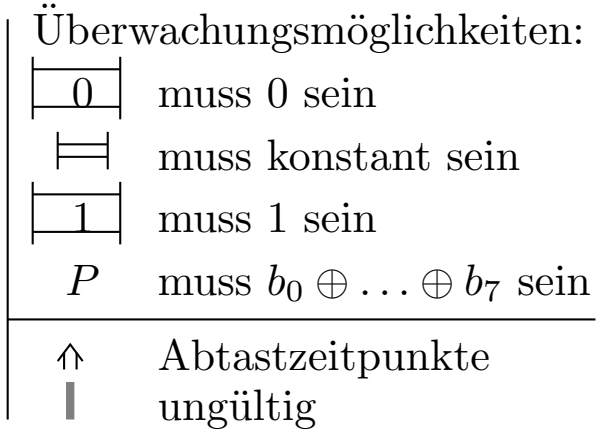
Übertragungsprotokolle

Übertragungsprotokoll, UART

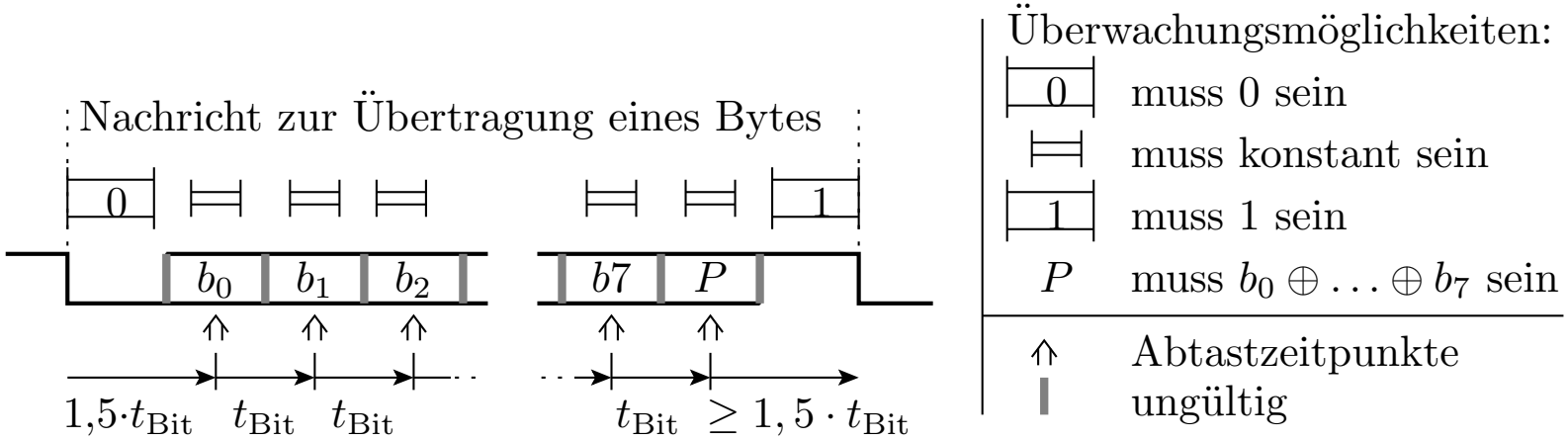
Bei der Datenübertragung wird das Format durch das Protokoll festgelegt. UART-Protokoll zur bytweisen Datenübertragung:

- 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.





Kontrollierbare Formateigenschaften



Erkennbare Formatfehler

- Paritätsfehler: 1-Bit-CRC, $p_{\text{E}} \approx 50\%$
- Datenrahmenfehler: Datenwechsel in Zeitfenstern, in denen keine Änderung erlaubt. Zu kurze Start- oder Stoppbitzeit.
- Datenüberlauf: mehr ankommende Bytes als der Empfänger verarbeiten kann.
- Datenspezifische Formatmerkmale (Startbyte, Prüfsummen, ...)

Ethernet-Datenpakt

Ethernet-Paket:

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

Erkennbare Formatfehler:

- Prüfkennzeichenfehler. 4-Byte CRC, Erkennungssicherheit

$$p_{\mathrm{E}} = 1 - 2^{-32} \approx 1 - 2 \cdot 10^{-10}$$

- Soll-/Ist-Abweichung konstante Bytes.
- Datenwechsel in unzul. Zeitfenstern,
- Unzul. Nutzdaten, ...

Reaktionen auf erkannte FF:

- Wiederholtes Senden bei Nachrichtenkollision.
- Wiederholte Anforderung nicht erhaltener Pakete.

CAN-Bus[6] Nachricht

ID	Nachrichtennummer	CRC	Prüfkennzeichen
DLC	Datenlängencode	EOF	Ende des Datenrahmens

Erkennbare Formatfehler:

- PKZ-Fehler. 15-Bit CRC, Erkennungssicherheit

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

- Soll-/Ist-Abweichung konstante Bits, unzul. Datenwechsel,
- Überwachung auf »Lebenszeichen« durch alle anderen Steuergeräte.

Reaktionen auf erkannte FF:

- Wiederholtes Senden bei Nachrichtenkollision.
- Notfallreaktion bei Ausfall anderer Steuergeräte.

Syntaxtest

Syntaxtest

Ein Syntaxtest kontrolliert für eine Zeichenfolge, dass sie »ein Wort einer formalen Sprache« ist.

Die Kontrolle und Datenextraktion erfolgt mit einem Automtaten.

Ein spracherkennenden 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.

Anwendung: manuelle Dateneingabe.

Formale Sprachen

Eine formale Sprache definiert zulässige Worte durch Syntaxregeln.
Beispiel EBNF (Eweiterte Backus-Naur-Form[7]):

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

(NTS – zu ersetzendes Symbol, Nicht-Terminalsymbol).

[7] Siehe http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf.

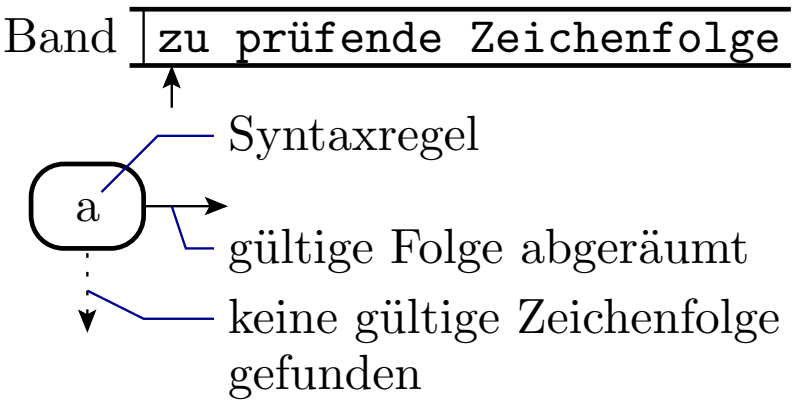
Beispiele für EBNF-Syntaxregeln

```
Zahl = ["-"], (ZiffernAusserNull, {Ziffer}) | "0";

ZiffernAusserNull = "1"|"2"|"3"| ... |"9";

Ziffer = ZiffernAusserNull | "0";
```

```
Bezeichner = Buchstabe, {Buchstabe | Ziffer};
```

Spracherkennende Moore-Automaten

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

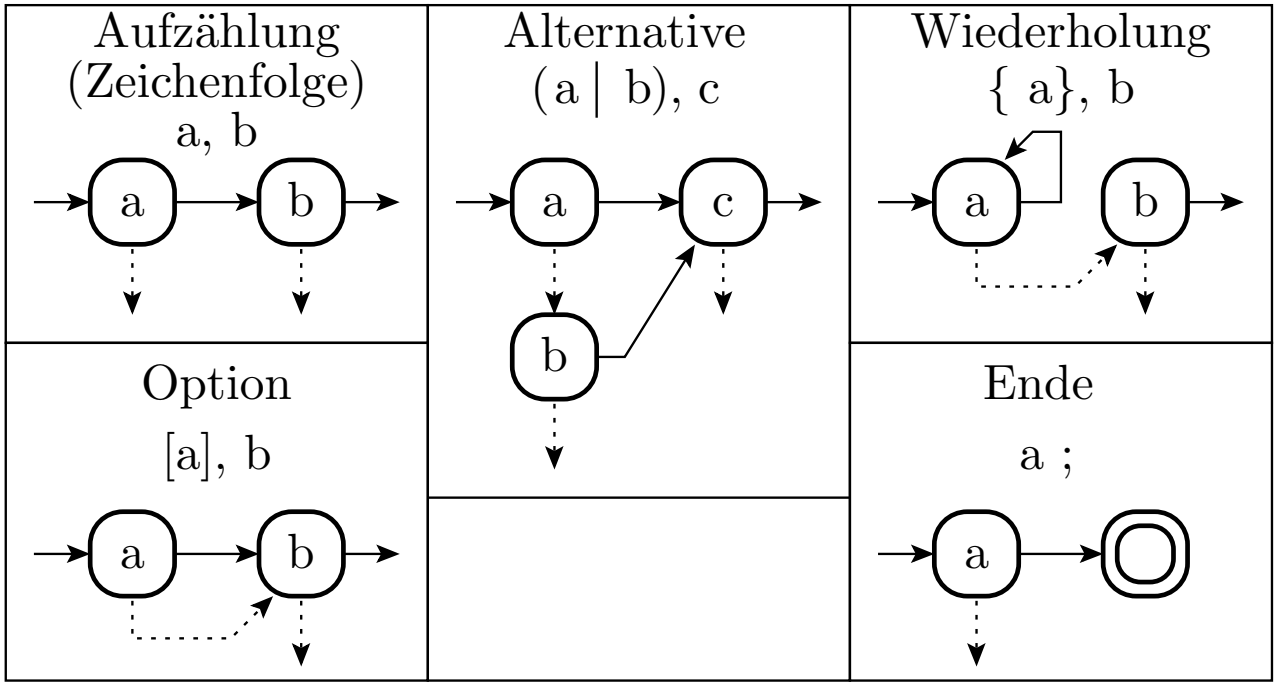


In jedem Automatenzustand wird versucht, eine Zeichenfolge nach der definierten Syntaxregel abzuräumen:

- 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 Alternative enden im Fehlerzustand.

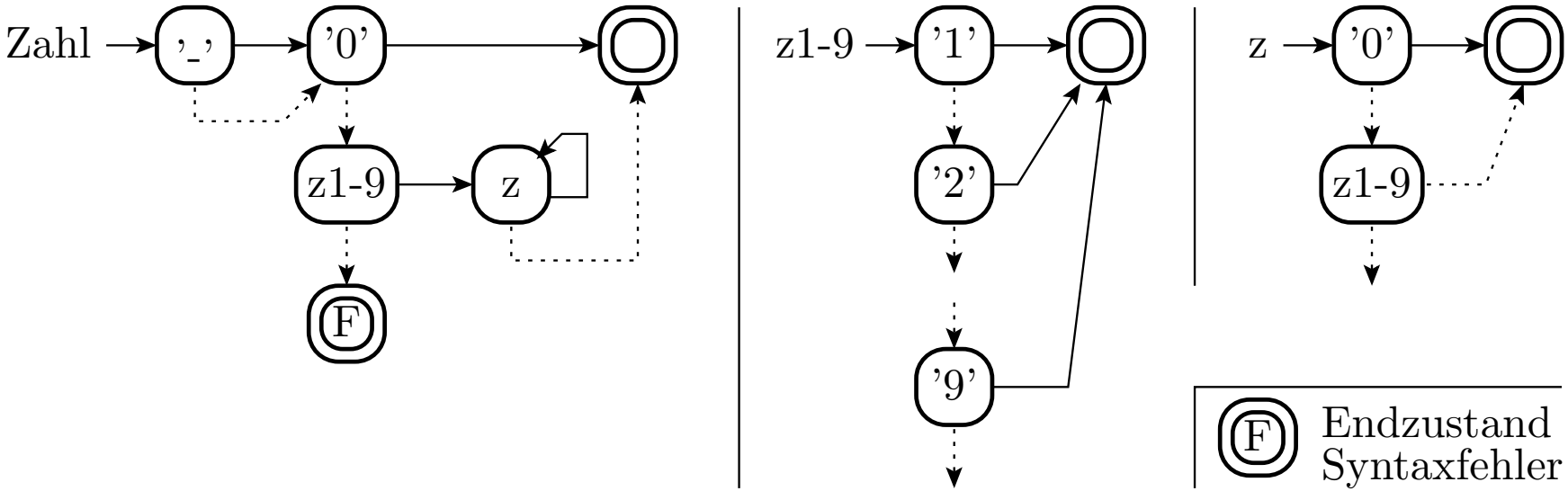
Von der EBNF zum Automaten



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?

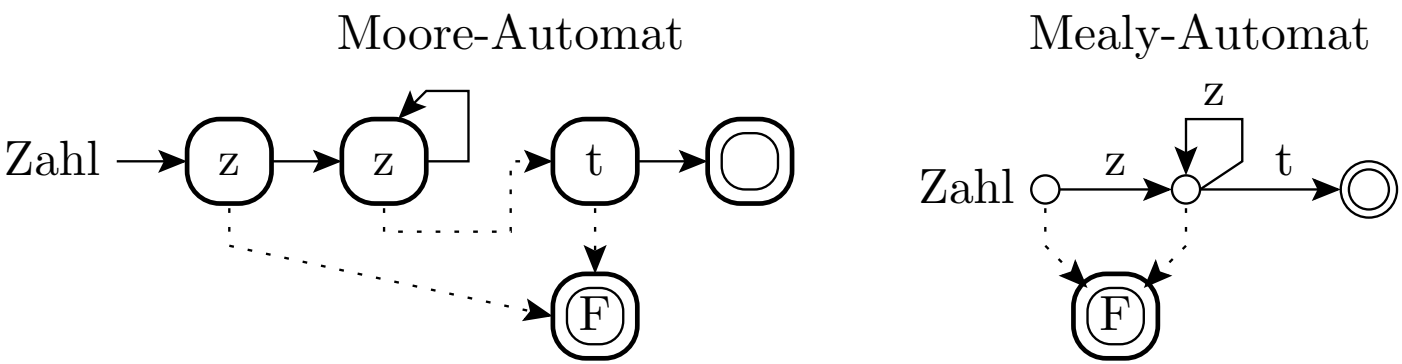
- "125_": '-'↓, '0'↓, z1-9→, z→, z→, z↓, Endzustand »ok«
- "k89": '-'↓, '0'↓, z1-9↓, Endzustand Syntaxfehler
- "-0701" : '-'→, '0' →, Endzustand »ok« (»-0« gefunden)

Spracherkennender Mealy-Automat

- 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)



Mit den einfachen Ersetzungsregeln »darf sein«, »darf n-mal vorkommen« lassen sich viele Datenformate und auch Formate für Programme, die in Entstehungsprozessen als Daten eingegeben werden, 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.

Umgekehrt lässt sich eine Sprache auch so definieren, dass die Erkennung sehr einfach ist.

Ein Syntaxtest erkennt alle Verletzungen der Syntaxregeln. Idealerweise sind alle syntaktisch korrekten Daten weiterverarbeitbar, aber nicht unbedingt richtig.

Ein Syntaxtest erkennt grob geschätzt die Hälfte der Eingabefehler durch Menschen bei der Datenerfassung.

Standardisierte Sprachen

Sprachen für die manuelle Datenerfassung:

- Programmiersprachen (C, Java, ...)
- XML (**E**xtensible **M**arkup **L**anguage) zur Darstellung hierarchischen Darstellung strukturierter Daten in Form von Textdateien.
- CVS: Beschreibung tabellarischer Daten, ...

Für automatisch generierte Daten, die nur aufzubewahren oder weiterzugeben, aber nicht für eine manuelle Bearbeitung vorgesehen sind, eignen sind fehlererkennende Codes und Prüfkennzeichen besser. Weniger Programmier- und Rechenaufwand. Höhere Fehlererkennungssicherheit.

Wertebereichskontrollen

Wertebereichskontrolle

Daten in einem Programm / einer Schaltungsbeschreibung haben in der Regel viel kleinere Wertebereiche als Darstellungsbereiche:

- Altersangabe für eine Person mit einer 32-Bit-Integer-Zahl (≈ 120 zulässige und 2^{32} darstellbare Werte)
- Zeiger auf ein n Elemente großes Feld in einem 32-Bit-System; zulässig sind nur die Werte

$$w = a + g \cdot i \quad \text{mit} \quad i \in \{0, \} 1, \ldots, n-1$$

(a – Feldanfang; g – Elementegröße); darstellbar 2^{32} Werte.

Programmierung von Wertebereichskontrollen:

- Fallunterscheidung mit einprogrammierten Reaktionen für erwartete FF und spezielle Fehlerbehandlung.
- Assertion-Anweisung (vergl. TV_F2«, Abschn. 4) mit Betriebssystemaufruf für unerwartete FF und allgemeine Fehlerbehandlung.

Erkennungswahrscheinlichkeit

Beispiel: Darstellung des Alters einer Person, $WB \in [0, 119]$ mit einer 32-Bit-Integer-Variablen:

Fehlerannahmen:

- Verwechslung mit dem Alter einer anderen Person: meist falsch, immer zulässig, nie nachweisbar.
- Verwechslung mit der Hausnummer: meist falsch, oft zulässig[8], selten nachweisbar.
- Verwechslung mit anderem gespeicherten Wert. Kleine positive Datenwerte treten überproportional häufig auf.
- Grob geschätzt:

$$p_{\mathrm{E}} < 50\% \ll 1 - \frac{120}{2^{32}}$$

Die Erkennungswahrscheinlichkeit einer Wertebereichskontrolle ist schwer abschätzbar und oft nicht sehr hoch. Die Leistungsfähigkeit des Verfahrens liegt in der Vielzahl der Kontrollmöglichkeiten.

[8] Hausnummern sind immer größer null und meist keiner 119.

Erhöhung der Erkennungswahrscheinlichkeit

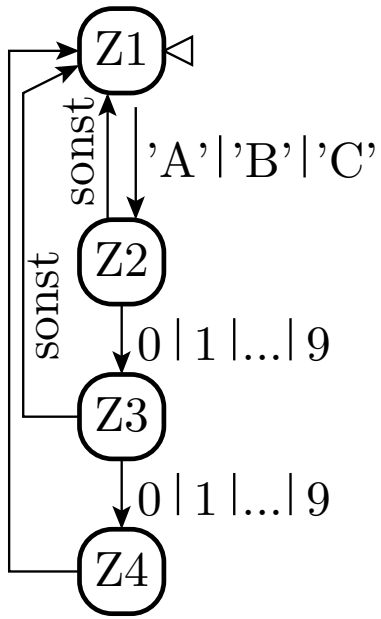
Zur Erhöhung der Erkennungswahrscheinlichkeit von Wertebereichskontrollen sind die zulässigen Werte ähnlich wie bei einem fehlererkennenden Code »zufällig« in der Menge der darstellbaren Werte zu verteilen:

- Verschiebung der Wertebereiche um einen konstanten Wert,
- Multiplikation mit einer Konstanten,
- Statt der üblichen Zustandscodierung

```
#define EmpfAut_Z1 1
#define EmpfAut_Z2 2
...
```

Zufallscodierung für Automaten

```
#define EmpfAut_Z1 0x35
#define EmpfAut_Z2 0x58
...
```



- Objekte mit Typkennung, »Objektgröße«, Wertebereich, Prüfsumme, ...

```
struct tFeld {
 uint8_t typ;   //Konstante für Typ, z.B. 0x4D
 uint8_t len;   //Wertebereich 0 bis 100
 int16_t *feld;//Feld Indexbereich 0 bis len-1
}               //WB Elemente -10000 bis 10000
```

  und Typ-, Bereichskontrollen, ... in den Bearbeitungsmethoden:

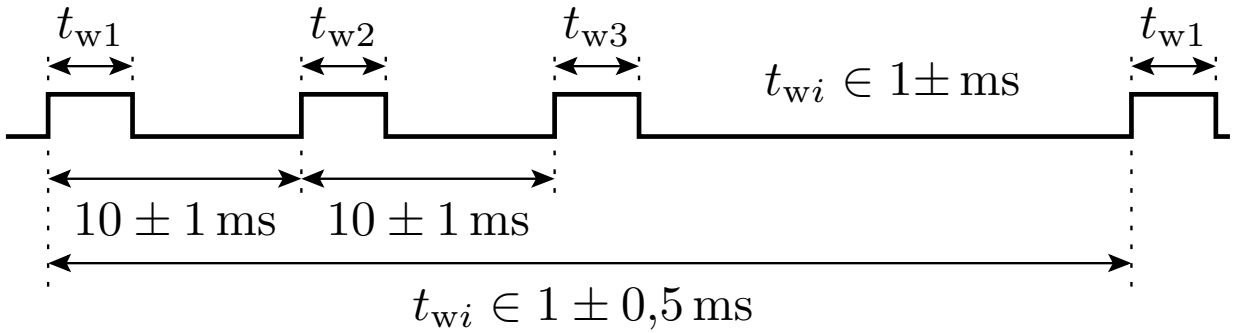
```
uint8_t check_tFeld(struct tFeld f){
 if (f.typ/=0x4D) return 1;
 if (f.len>100) return 2;
 for (i=0;i<100;i++)
  if ((f.feld(i)<-10000) || (f.feld(i)>10000))
   return i+3;
 return 0;}
```

- ...

Signalüberwachung

Auf der Schaltungsebene kommunizieren IT-Systeme über Signale. Ein Signal ist ein zeitlicher Werteverlauf einer physikalischen Größe. Die zu übergebende Information steckt in Merkmalen: Zeitverlauf der Amplitude, Frequenz, ... Auch hier teilt sich die Darstellung ein in

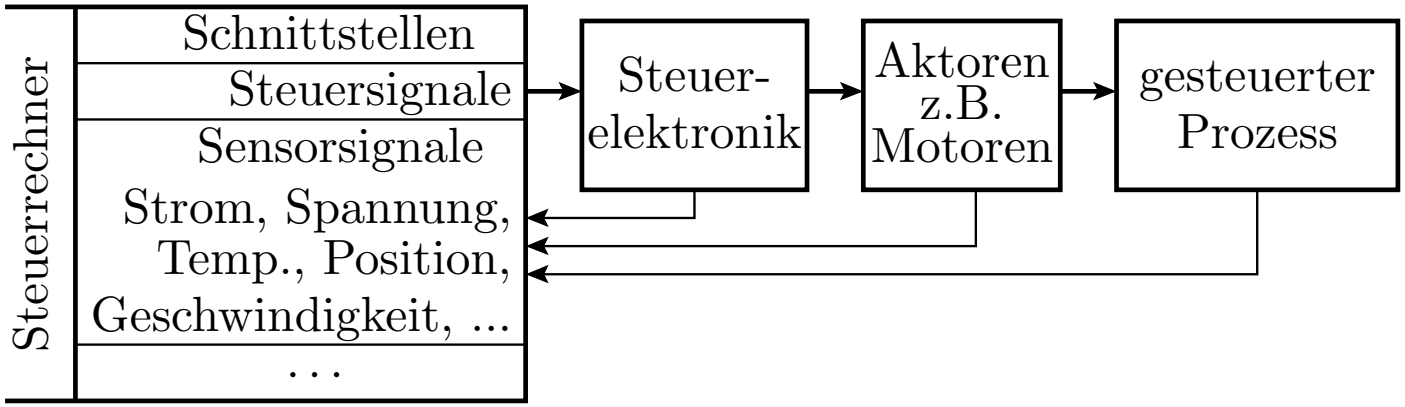
- Format (Gültigkeitsmerkmale) und
- Merkmale, die Werte repräsentieren.



Das dargestellte PWM-Signal besteht aus einer Pulsfolge, bei der der Low-Pegel, der High-Pegel, der Pulsabstand und die Pulsbreite alle in einem bestimmten und damit kontrollierbaren Toleranzbereich liegen müssen. Die Information steckt typisch in der Pulsbreite, dem Abstand oder beidem.

Überwachung analoger Signale

Analoge Signale hat man im Wesentlichen als Sensoreingaben.

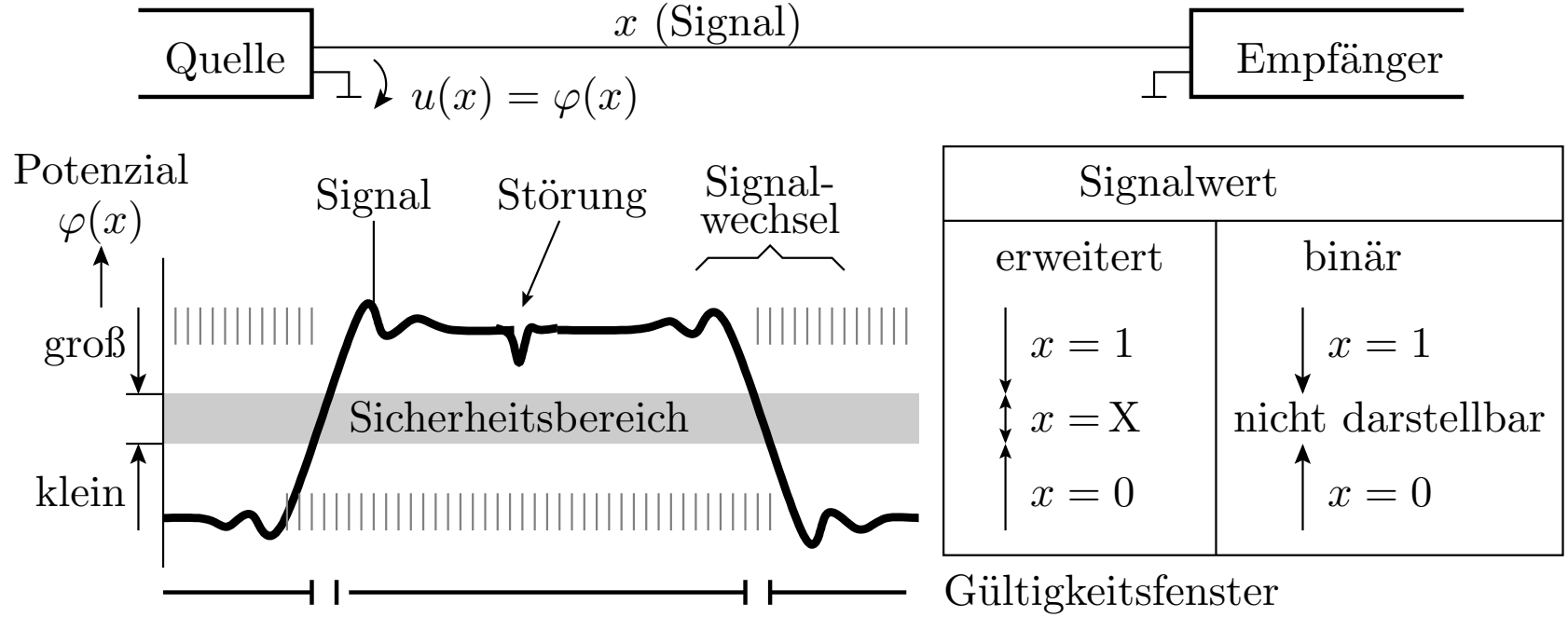


- Bereichsüberwachungen, Überwachung von Pulsbreiten, ...
- Überwachung komplexer Merkmale wie Stromsignaturen an Schrittmotoren auf Schrittfehler[9].

[9] Die Kontrolle komplexer Signaleigenschaften erfolgt erst nach der Digitalisierung, ist z.T. sehr rechenintensi und ein Großteil der Gesamtfunktionalität eines Steuergeräts.

Digitale vs. analoge Signale

Die Digitalisierung schafft einen Zwischenbereich zwischen den gültigen Werten. Jedes Gatter im Signalfluss korrigiert alle tolerierbaren Eingabeverfälschungen, die z.B. durch Rauschen und andere Störungen entstehen. Erweiterbar um die Erkennung unzulässiger Signalwerte und Glitches in den Gültigkeitsfenstern.



Wertekontrollen

Überwachung der Ergebniswerte von SL

Die bisher behandelten Überwachungsverfahren überprüfen nur Formatmerkmale, d.h. die Zulässigkeit einer SL.

Eine unzulässige SL ist eine FF. Eine zulässige SL kann, aber muss keine FF sein.

Eine Kontrolle auf Richtigkeit verlangt zusätzliche Kontrollen der Werte. Mögliche Verfahren:

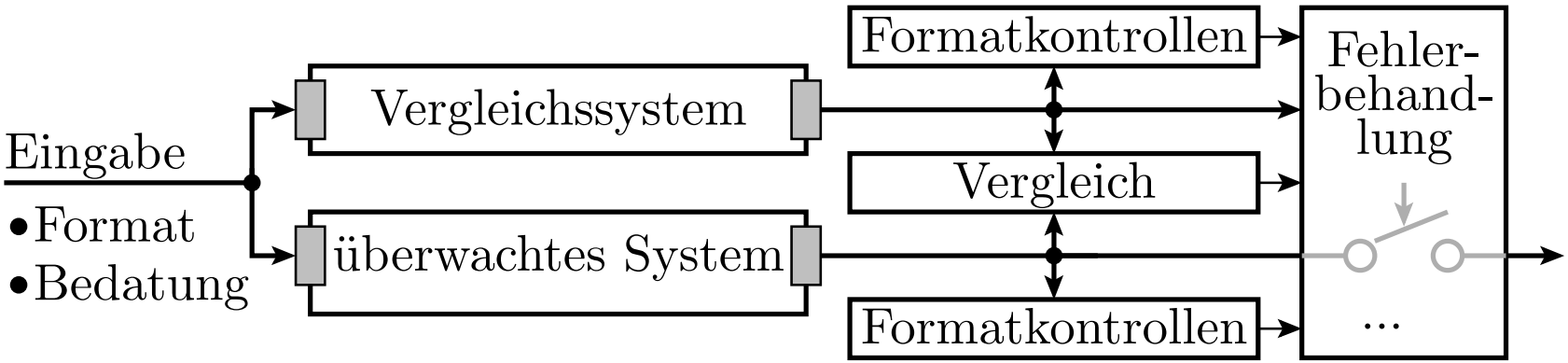
1. Vergleich mit Sollwerten,
2. Mehrfachberechnung und Vergleich,
3. Rückrechnung der Eingabe aus Ausgaben + Vergleich und
4. die Überwachung von hinreichenden Kriterien für die Richtigkeit.

Keines der Verfahren ist perfekt. (1) setzt Testbedingungen, (2) Diversität, (3) eine umkehrbar eindeutige Abbildung der Eingaben auf Ausgaben und (4) überprüfbare Richtigkeitskriterien voraus. ...

Mehrfachber. & Vergleich

Mehrfachberechnung und Vergleich

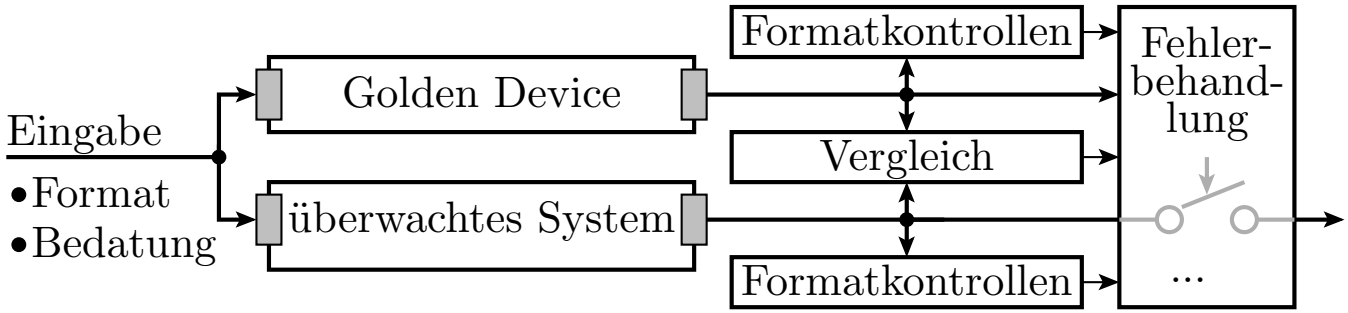
Die Ergebnisse werden mehrfach berechnet. Übereinstimmende Ergebnisse gelten als fehlerfrei. Bei Abweichung kann bei mindestens 3 Berechnungen und Mehrheitsentscheid oder Zusatzkontrollen ein Ergebnis als richtig gewertet und weitergereicht werden.



Das Vergleichssystem kann sein:

- ein »Golden Device« (Ausgaben per Definition richtig) oder
- ein reales System, dessen Ausgaben genau wie die des überwachten Systems fehlerhaft sein können.

Vergleich mit einem »Golden Device«

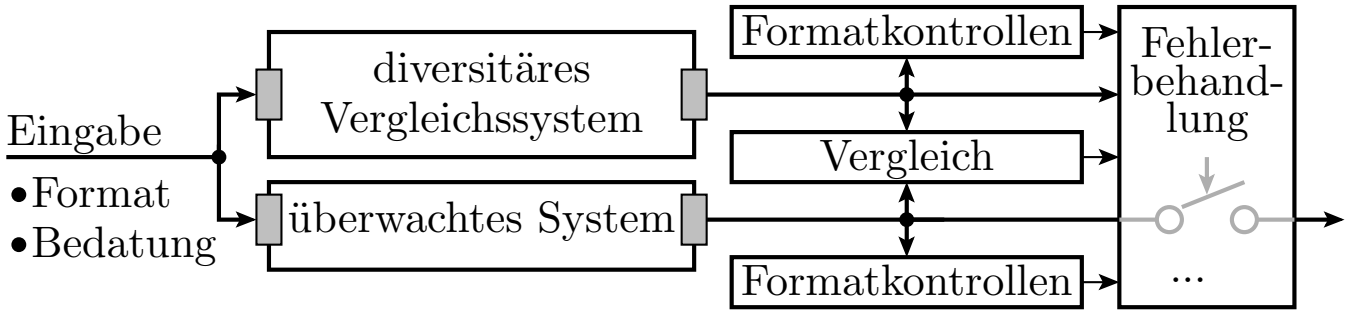


Ein »Golden Device« ist ein gründlich getestetes System, ein Mustergerät oder bei einem Regressionstest die Vorgängerversion. Seine Ausgaben gelten per Definition als richtig. Mit einem »Golden Device« und einem idealen Vergleicher[10] werden alle Werteabweichungen erkannt.

Ein wirklich immer korrekt arbeitendes System gibt es nicht und Vergleich verlangt korrekte Formate. Deshalb Zusatzkontrollen.

[10] Vergleicher, der übereinstimmende Werte immer als übereinstimmend und abweichende Werte immer als abweichend klassifiziert. In der Realität sind Vergleichsfehler nicht vollständig ausschließbar.

Vergleich diversitärer Systeme



Ein reales System als Vergleichssystem liefert ähnlich häufig falsche Ergebnisse wie das überwachte System. Phantom-FF-Wahrsch.:

$$p_{\mathrm{Phan}} = \frac{1}{Z_{\mathrm{V}}}$$

(Z_{V} – Zuverlässigkeit Vergleichssystems). Problematisch sind gleiche Fehler, die die Ausgaben übereinstimmend verfälschen. Gleiche FF werden beim Vergleich immer maskiert. Maskierungswahrscheinl.:

$$p_{\mathrm{M}} = \frac{1}{Div} \quad 1 \leq Div \leq Z_{\mathrm{V}}$$

(Div – Diversität, vergl. TV_F2, Abschn. 4.4).

Exakter Vergleich und Fenstervergleich

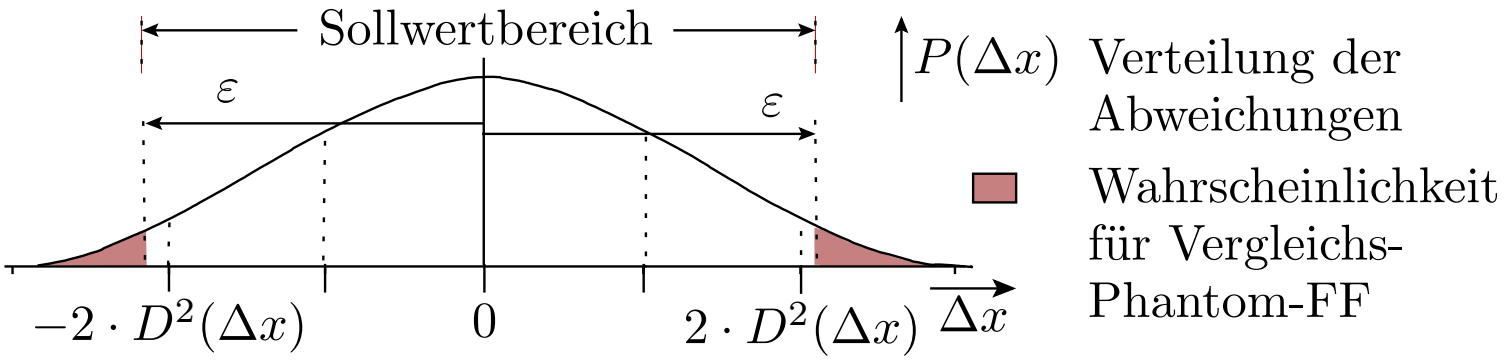
Vergleich mit exakten Sollwerten (ganzzahlig, Aufzählungstypen, ...):

```
if (x!=Sollwert) <Fehlerbehandlung>;
```

Fenstervergleich (Werte mit Nachkommastellen, Analogwerten, ...):

```
if (abs(x-Sollwert) > eps) <Fehlerbehandlung>;
```

Die Phantom-FF- und Maskierungswahrscheinlichkeit eines Fenstervergleichs ergeben sich aus der Verteilung der Abweichungen vom Sollwert und dem Fensterradius ε (siehe später TV_F5):



Mess-, Quantisierungs- und Rundungsfehler

Bei Quantisierung und Rundung ist ein Fehler von $\pm 0{,}5\,\mathrm{LSB}$[11] unvermeidbar. Die Standardabweichung als die Wurzel aus der mittleren quadratischen Abweichung ist die reichliche Hälfte davon:

$$\sqrt{D^2\left(X\right)} = \sqrt{\int_0^{0{,}5} x^2 \cdot dx} \cdot \mathrm{LSB} = \frac{1}{\sqrt{12}} \cdot \mathrm{LSB} = 0{,}29 \cdot \mathrm{LSB}$$

Hinzu kommen Linearitätsfehler, oft $> 1\,\mathrm{LSB}$, Rundungsfehler.

Rundung von drei Werten auf 8 Nachkommabits:

Wert	nächster darstellbarer Wert	Rundungsfehler
125,4380	0x7D,70=125,4375	−0,05%
2,7130	0x2,B7=2,7148	+0,18%
28,2000	0x= 28,1992	−0,08%

[11] LSB (least significant bit) Wert einer Quantisierungsstufe.

Fehlerfortpflanzung

Wert	nächster darstellbarer Wert	Rundungsfehler
125,4380	0x7D,70=125,4375	−0,05%
2,7130	0x2,B7=2,7148	+0,18%
28,2000	0x= 28,1992	−0,08%

Multiplikation der drei auf 8 Nachkommabits gerundeten Werte:

$$\begin{aligned} 125{,}438 \cdot 2{,}713 \cdot 28{,}2 &= 9596{,}8349 \\ \text{0x7D,70} \cdot \text{0x2,B7} \cdot \text{0x1C,33} &= \text{0x2569,C7} = 9577{,}7773 \\ \text{Abweichung}: &\quad -19{,}0575 \end{aligned}$$

Einige Regeln der Fortpflanzung der Fehlergrenzen:

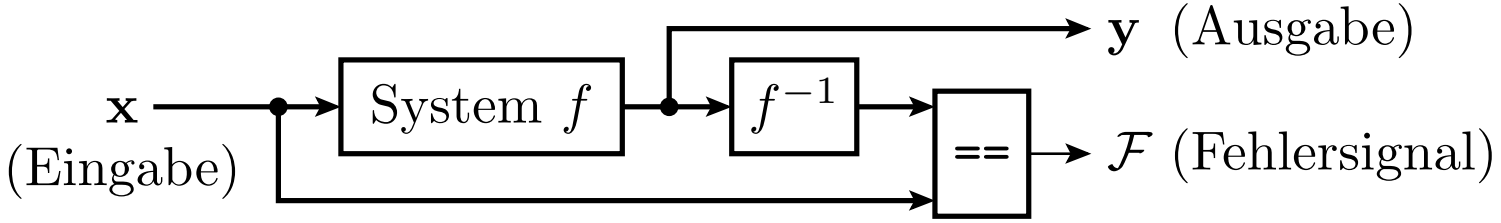
- Add., Sub.: Addition der absoluten Fehlergrenzen.
- Mult., Div.: Addition der relativen Fehlergrenzen.

Problematisch sind Differenzen, bei denen sich die Werte gegenseitig aufheben, aber die absoluten Fehler addieren.

Loop-Back Test

Loop-Back Test (Rückrechnung der Eingabe + Vgl.)

Für umkehrbare Funktion $f(x)$ mit $f^{-1}(f(x)) = x$ lässt sich das Ergebnis auch dadurch kontrollieren, dass aus dem Ergebnis die Eingabe zurückberechnet und mit den ursprünglichen Service-Eingaben verglichen wird:



Beispiele für Funktionen mit Umkehrfunktion:

- Quadrierung $\leftrightarrow$ Wurzelberechnung,
- Addition $\leftrightarrow$ Subtraktion,
- Multiplikation $\leftrightarrow$ Division,
- Analog/Digital-Wandlung $\leftrightarrow$ Digital/Analog-Wandlung,
- Daten versenden $\leftrightarrow$ Daten empfangen, ...

- Im Vergleich zu »Verdopplung und Vergleich« haben eine Funktion und ihre Umkehrfunktionen einen anderen Algorithmus und dadurch tendentiell eine höhere Diversität und Erkennungswahrscheinlichkeit.
- Für Service-Leistungen, für die ein Service mit Umkehrfunktion ohnehin vorhanden ist, z.B. außer dem seriellen Sender auch der passende serielle Empfänger, ist ein Loop-Back Test die naheliegenste und einfachste Lösung.
- Bei zu erwartenden Mess-, Quantisierungs- und numerischen Fehlern Fenstervergleich.

Kontrollkriterien für Richtigkeit

Kontrolle auf Richtigkeit

Es gibt Aufgaben, die sind durch Kontrollkriterien für die Richtigkeit der Ergebnisse spezifiziert:

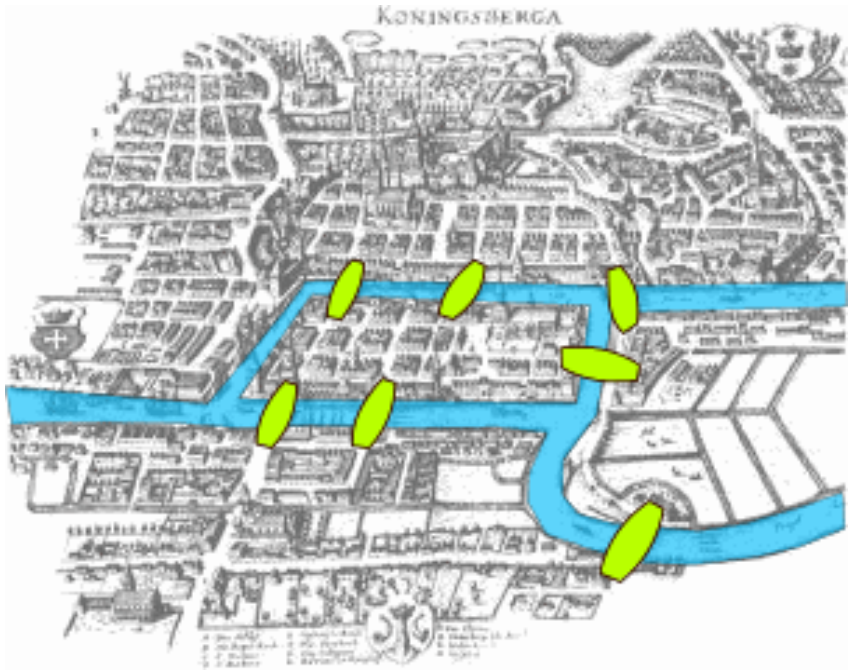
- Suche eine Funktion, die eine Differenzialgleichung erfüllt. Kontrolle durch Einsetzen der Funktion in die DGL.
- Suche einen Test, der einen Fehler nachweist. Kontrolle durch Simulation des Systems mit und ohne Fehler und Vergleich der berechneten Ausgaben.
- Suche einen Weg, der alle Knoten eines Graphen verbindet.

Die vorgegebenen Kontrollfunktionen sind oft einfacher und dadurch fehlerärmer zu realisieren als die Suchalgorithmen. Im Entwurfsprozess ist es zweckmäßig, diese zuerst zu programmieren, eventuell sogar diversitär, und sehr gründlich zu testen, um sie später gleichermaßen für die Suche und die Kontrolle des Suchergebnisses zu verwenden.

Königsberger Brückenproblem

Aufgabe: Suche einen Weg, der alle sieben Brücken über den Pregel genau einmal überquert.

Ergebnis sei eine verkettete Liste der Nummern der Brücken, die zu überqueren sind.



Probe:

```
Setze für alle Brücken die Zahl der Überquerungen null
Wiederhole für jedes Listenelement
  Anzahl der überquerten Brücke +1
Wiederhole für jede Brücke
  Kontrolle, dass die Anzahl eins ist.
```