Test und Verlässlichkeit Foliensatz 4: Kontrollen

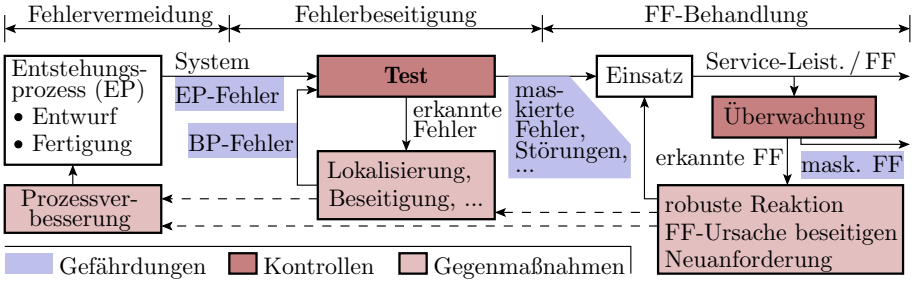
Prof. G. Kemnitz

November 6, 2022

Contents

Vorlesung	1	2	3	4
bis Abschn.	2.?? (??)	4.?? (??)	4.?? (??)	5.?? (??)

Die 3 Ebenen zur Sicherung der Verlässlichkeit



Jede der drei Ebenen zur Sicherung der Verlässlichkeit

1. Ergebnisüberwachung und FF-Behandlung.
2. Test und Fehlerbeseitigung.
3. Fehlervermeidung., Minderung und Beseitigung von Ursachen für die Fehlerentstehung.

umfasst Iterationen aus Kontrollen, Problembeseitigung und Erfolgskontrolle.

Kontrollen und Fehlerkultur

Die Vorlesung unterstellt eine für die Verlässlichkeit und die Modellierung idealisierte Fehlerkultur:

- alle erkannten relevanten Probleme beseitigen,
- Beseitigungserfolg kontrollieren und
- Rückbau nach erfolglosen Beseitigungsversuchen zur Minimierung der Entstehungsrate neuer Probleme.

Diese Fehlerkultur reduziert die

- Entstehungsursachen für Fehler,
- die nicht beseitigten Fehler und
- die FF, die Schaden verursachen

auf die nicht erkannten Entstehungsursachen, Fehler bzw. FF.

Die Tests und Kontrollen sind die Filter dafür, was unerkannt bleibt und damit wichtiger Ansatzpunkt zur Schaffung verlässlicher Systeme.

[reale Fehlerkultur: Kosten-Nutzen; menschliche Wünsche & Bedürfnisse; Tradition]

Was behandet der Foliensatz?

Mixed-Signal (Systeme mit analoger und digitaler Verarbeitung):

- Aufgabenteilung zwischen analoger und digitalter Verarbeitung.
- Prüfaufgaben und prüfgerechter Entwurf für analoge Verarbeitungsteile.

Baugruppentest:

- MDA, In-Circuit-Test, optische Inspektion, ...

Inspektion von Entwurfsergebnisse:

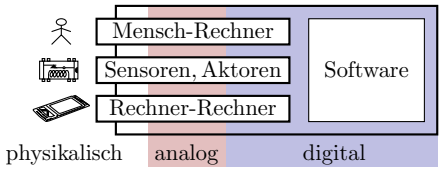
- Inspektionsziele und Technologien,
- Abschätzung der Fehlerüberdeckung und anderer Kenngrößen,
- Vergleich mit Zufallstest.

Kontrollen für digitale SL (Weiterführung von Foliensatz F1):

- Schnittstellen & Protokolle, Zeitüberwachung,
- Syntaxtest, Invarianten, WB,
- Fehlererkennende und -korrigierende Codes, Prüfkennzeichen,
- Wertekontrollen, ...

1 Mixed Signal

Mixed-Signal IT-System



Ein IT-System besteht überwiegend aus digitaler HW und der SW. Analoge Verarbeitung:

- Nur an den Schnittstellen zur physikalischen Umgebung.
- Überschaubare Funktionsvielfalt: Wandlung physikalisch ⇔ Elektrisch, Verstärkung, Bandbegrenzung, Wandlung analog ⇔ digital.
- Überschaubare messtechnische Aufgaben: Übertragungsfunktion, Frequenzgang, Linearität, ...

Kompliziertere Verarbeitungsfunktionen werden digital realisiert, noch kompliziertere Funktionen in SW.

1.1 Analog vs. digital

Analog vs. digital

[Trend analog ⇒ digital, Rundfunk, Fernsehen, Anlogrechner]

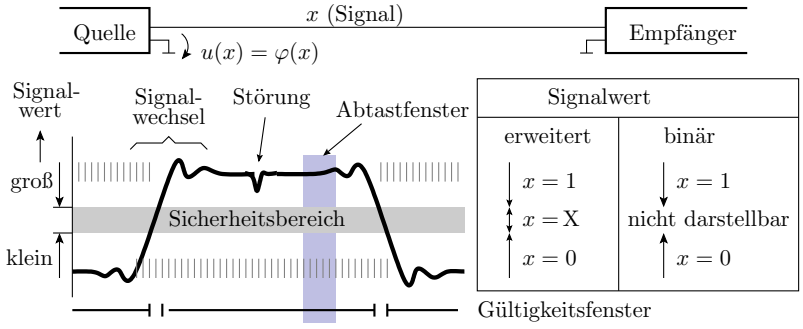
Probleme bei der Verarbeitung analogen Signale:

- Unterscheidung vieler Werte der signalführenden Größe (Spannung, ...); geringe Toleranz Verfälschungen durch Rauschen, Alterung, elektromagnetische Strörungen, ...
- Die Verfälschungen summieren sich für alle Verarbeitungsschritte.
- lange Verarbeitungssequenzen und Speichern von Zwischenwerten problematisch.

Probleme bei der Kontrolle analoger Signale:

- Addition von Messfehlern zu den Messwerten, Maskierungen und Phantom-FF.
- Forderung für die Prüftechnik: höhere Werte- und Zeitauflösung gegenüber den zu kontrollierenden Daten.
- Messfehlerrvermeidung auch beim Anschluss der Prüftechnik (Übersprechen, Refexionen, ...).

Digitalisierung



Digitalisierung: Informationstdarstellung durch Bits

- physikalische Werteunterscheidungt nur groß, klein und ungültig,
- Abtastung im Gültigkeitsfenster.
- Fehlertolerant gegen Verfälschungen durch Rauschen, induktives und kapazitives Übersprechen, Fertigungsstreuungen, Alterung, ...

Vorteile digitaler Verarbeitung

- Fehlertoleranz gegen Störungen, Fertigungsstreuungen, ...

Trotz der viel größeren Anzahl von Signalen für die Darstellung und Berechnungsschritten (z.B. für eine Addition von zwei Werten):

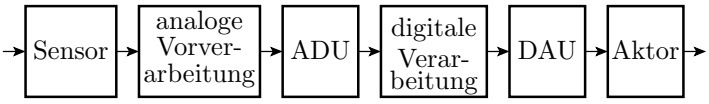
- geringere Fertigungskosten und Entwurfskosten[1],
- kleiner, schneller, genauer zuverlässiger.

Zusätzlicher funktionale Gestaltungsspielraum:

- Datenspeicherung,
- sequentielle und SW-gesteuerte Abarbeitung, ...
- Fehlerbeseitigung durch Programmänderung,
- Einschalttest,
- einprogrammierbare Testhilfen, Überwachungs- und Fehlerbehandlungsfunktionen.

1.2 Prüfgerechter Entwurf

Typische Testaufgaben



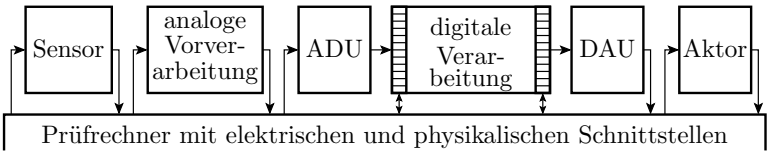
Typische Verarbeitungskette mit analogen Komponenten:

- Sensor, analoge Vorverarbeitung,
- analog-digital-Wandlung,
- digital-analog-Wandlung,
- analoge Nachbearbeitung, Aktor.

Weitere typische analoge Funktioneinheiten:

- Stromversorgung,
- Hochfrequenzbaugruppen, z.B. für WLAN.

Isolierter Test



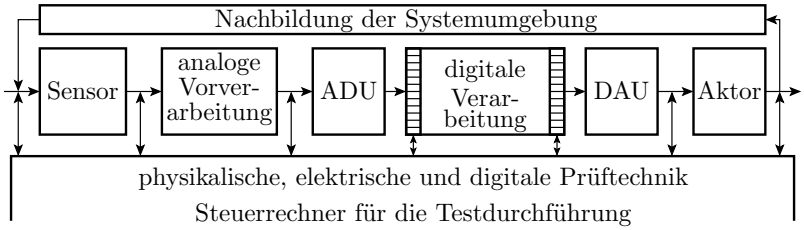
- Vorgabe und Kontrolle analoger Signalverläufe, zur Kontrolle Übertragungsfunktion, Frequenzgang, Verzerrung, ...
- Kontaktierung der Ein- und Ausgänge zum Anschluss geeigneter Prüftechnik,
- Deaktivierung der Eingabequellen aus der Funktionsumgebung,
- Ablaufsteuerung mit Steuerrechner.

Prüfgerechter Entwurf

Die Art der Testdurchführung für die HW muss schon in der Spezifikationsphase in den Entwurf einfließen. Ohne prüfgerechten Entwurf kein ausreichender Test, der Einsatzvoraussetzung ist.

[1] Komplexe Digitalschaltungen werden heute wie Software entworfen.

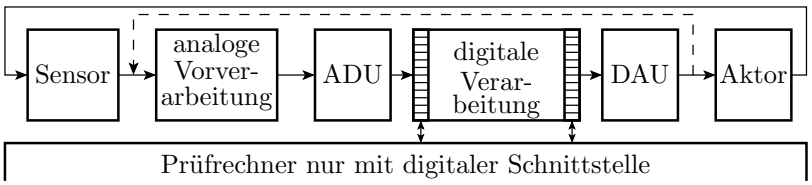
Test in der Systemumgebung



- physikalische Eingabewerte aus der Systemumgebung,
- Ausgabe über Aktoren,
- digitale Testein- und Ausgabe, in der Regel über einen Testbus[2].
- zusätzliche Beobachtung analoger Zwischenwerte ist hilfreich.

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).

Loop-Test



Nutzung der System-HW als Test-HW:

- Ansteuerung der Sensoren durch Aktoren oder
- Ansteuerung der analoger Eingaben durch analoge Ausgaben.
- Übergabe der analogen Testeingaben über System-DAU und
- Messen der analogen Testergebnisse über System-ADU.
- Der Steuerrechnen liefert und wertet nur noch digitale Daten aus.
- Erweiterung zum Selbsttest durch Abarbeitung des Testprogramms auf dem Systemrechner.

Kontrollfragen

- Was sind aus Sicht der Verlässlichkeit wesentliche Vorteile der digitalen gegenüber der analogen Verarbeitung?
- Ein Spannungsmessgerät hat einen normalverteilten Messfehler mit einem Erwartungswert von 0,1 V und eine Standardabweichung von 0,2 V. Der zu überwachende Wert soll in einem Toleranzfenter von 0,99 V bis 1,01 V liegen. Wie groß sind die Wahrscheinlichkeiten für eine Fehlermaskierung bzw. einen Phantomfehler, wenn der gemessene Wert 0,991 V, 1,00 V und 1,011 V beträgt?
- Nennen Sie Maßnahmen des Prüfgerechten Entwurfs zur Sicherstellung einer ausreichenden Testbarkeit analoger Funktionseinheiten.
- Was bedeutet HIL (Hardware in the Loop) bzw. wozu braucht man einen HIL?

[2] Alternativ kann das Testprogramm auch auf dem Systemrechner laufen.

2 Baugruppentest

Hierarchie und Test

Rechnerhardware besteht aus tauschbaren Komponenten:

- tauschbare Teilsystem.
- tauschbare Steckeinheiten,
- austauschbare Bauteile.

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

Wiederholungsfragen

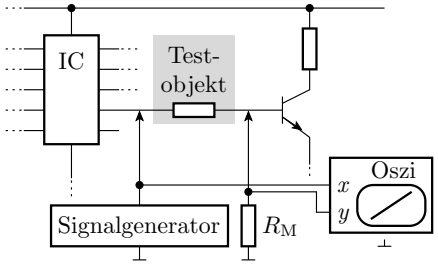
1. Warum tauschbare Komponenten bzw. unter welcher Bedingung nicht erforderlich (nicht zweckmäßig)?
2. Warum ist ein gründlicher Komponententest zu fordern?
3. Warum beginnt ein BG Test mit einem statischen Bestückungs- und Verbindungstest?
4. Warum ist ein Verzicht auf dynamische BG-Tests denkbar?

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 reine Relais-Matrix zur Verbindung mit den Prüfgeräten angeschlossen. Auch Bestückungsfehler lassen sich überwiegend mit Zweipunktmessungen von Strom-Spannungsbeziehungen erkennen. Fehlerlokalisierung im Vergleich zu der für einen dynamischen Test, der versagt, sehr einfach.

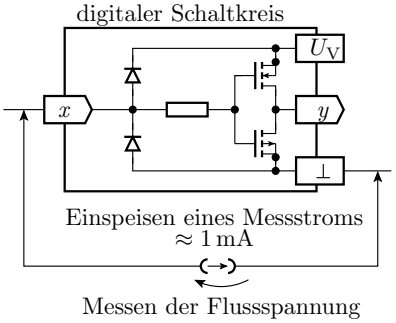
Bestückungstests



Kontrolle auf Bestückungsfehler durch Überprüfung ausgewählter Zweipunktmerkmale:

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

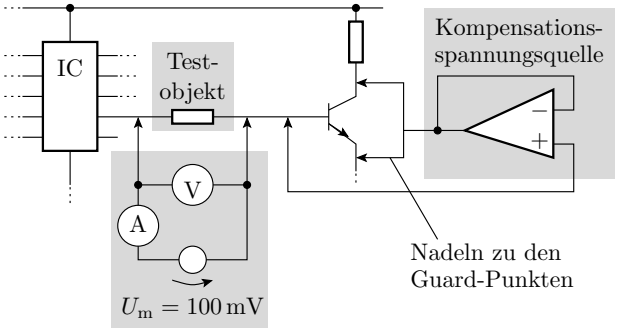
Zu Kontrolle, dass Schaltkreisanschlüsse mit dem Kontakt-Pad verbunden sind, werden die Schutzdioden zur Versorgungsspannung und Masse ausgemessen.



Analoger In-Circuit Test

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:

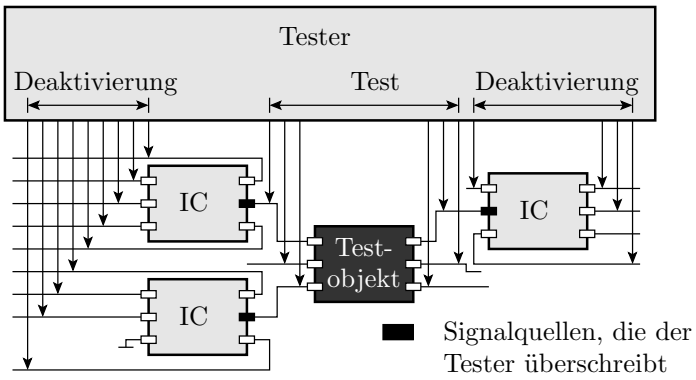
- die Toleranzbereiche der Sollwerte mit allen Bauteilstreuungen,
- die Erkennungssicherheit für Fehlbestückungen, z.B. bei sehr kleinen Kapazitäten.



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

- Erlaubt einen isolierten Zweipoltest.
- Vereinfacht die Testprogrammerstellung, insbesonderen die Sollwert- und Sollwerttoleranzfestlegung.
- Mindert die Häufigkeit von Fehlklassifikationen.

Digitaler In-Circuit-Test



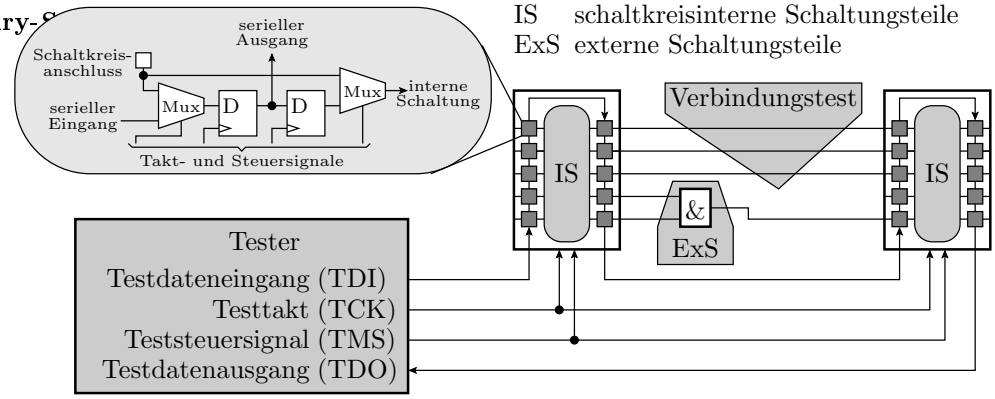
- Kontaktierung der Baugruppe über ein Nadelbetadapter.
- Isolierter Test der Schaltkreise durch Überschreiben der digitalen Schaltkreiseeingaben mit stromstarken Treibern.
- Im Gegensatz zum analogen ICT unter Spannung.
- Andere Schaltkreise werden möglichst deaktiviert (Anschlüsse hochohmig).

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. Besonderes Problem: Nach einem Ausfall der Baugruppe z.B. bei Vibration in einem Fahrzeug ist sofort erkennbar, dass es sich um einen Fertigungsfehler handelt, der (optisch) erkennbar gewesen wäre. Fall für Produkthaftung.

Boundary-S



Ersatz der mechanischen Nadeln durch »silicon nails« (seriell beschreibbare Register an den Schaltkreisanschlüssen, im Normalbetrieb überbrückt). Alternative zu den teuren, für jede Baugruppe speziell anzufertigenden Nadeladaptern.

Ablauf eines Testschritts für den Baugruppentest:

- BS-Register aller Schaltkreise auf der Baugruppe seriell beschreiben,
- Datenübergabe (Update),
- Datenübernahme (Capture),
- serielle Ausgabe der übernommen Wert und Laden des Eingabevektors für den nächsten Testschritt.

Fortsetzung Foliensatz F5, Abschn. 3.5 »Testbus«.

Kontrollfragen

- Zählen Sie Maßnahmen des prüfgerechten Entwurfs für den Baugruppentest auf.
- Was bedeutet isolierter Test von Komponenten und wie wird dieses Prinzip mit dem In-Circuit-Test umgesetzt?
- Warum ist für SMD-bestückte Baugruppen von Steuergeräten für KFZ eine optische Inspektion zwingend?
- Welche prüftechnische Problem des Baugruppentests löst Boundary-Scan?

3 Inspektion

Inspektion (Review)

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

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

Inspektion für Entwurfsergebnisse:

- Korrekturlesen der entandenen Dokumentationen,

- einsetzbar in frühen Entwurfsphasen[3],
- hohen manuellen Arbeitsaufwand.
- Es werden nicht nur Fehler erkannt, die nachweislich die Funktion beeinträchtigen, sondern auch Verstößen gegen Vereinbarungen und Standards und Antipattern[4].
- Know-How-Weitergabe als positiver Zusatzeffekt.

Kenngrößen einer Inspektion

Wichtigstes Gütemaß ist die Inspektionsfehlerüberdeckung:

$$I\hat{F}C = \frac{\#EF}{\#F}$$

($\#EF$ – Anzahl der nachweisbaren; $\#F$ – Anzahl aller (entstandenen) Fehler). Insgesamt oder getrennt für funktionale und sonstige Fehler.

Abschätzmöglichkeiten:

- Capture-Recapture-Verfahren (klassischer Ansatz).
- Modell Zufallstest.

Weitere Kenngrößen zur Bewertung von Inspektionsprozesse[5] [3]:

- Effizienz: Gefundene Abweichungen pro Mitarbeiterstunde.
- Effektivität: Gefundene Abweichungen je 1000 NLOC[6].

Beispiel 1. Zähl- und Zeitwerte zur Bewertung einer 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

1. die Inspektionsfehlerüberdeckung,
2. die Effizienz und
3. die Effektivität

der Inspektion?

	gesamt	funktionale Fehler	sonstige Fehler
$I\hat{F}C = \frac{\#EF}{\#F}$	$\frac{228}{300}$	$\frac{156}{200}$	$\frac{72}{100}$
Effizienz	$\frac{228\text{ Fehler}}{200\text{ h}}$	$\frac{156\text{ Fehler}}{200\text{ h}}$	$\frac{72\text{ Fehler}}{200\text{ h}}$
Effektivität	$\frac{228\text{ Fehler}}{10.000\text{ NLOC}}$	$\frac{156\text{ Fehler}}{10.000\text{ NLOC}}$	$\frac{72\text{ Fehler}}{10.000\text{ NLOC}}$

Effizienz und Effektivität ergeben sich ausschließlich aus Zähl- und gemessenen Zeitwerten. IFC basieren auf schlecht überprüfbaren Schätzwerten für die Gesamtfehleranzahl.

[3] Wichtig als Zwischenkontrollen im Stufenmodell vor der Cpdierung.

[4] Beschreibungselemente, die die Übersichtlichkeit beeinträchtigen, die Kontrolle erschweren und die Fehlerentstehung begünstigen.

[5] Inspektionen sind arbeitsintensive technologischer Prozesse, die reifen. Effizienz und Effektivität bewerten das Aufwand-Nutzen-Verhältnis .

[6] NLOC: **N**etto **L**ines **o**f **C**ode. Anzahl der Code-Zeilen ohne Kommentar- und Leerzeilen.

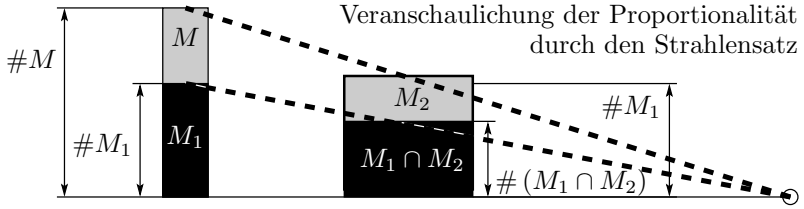
3.1 Capture-Recapture

Capture-Recapture-Verfahren

Abgeleitet von einem Schätzer für die Größe von Tierpopulationen (z.B. von Vögeln in einem Gebiet) [1, 5, 4].

- Aus einer Menge M unbekannter Größe wird eine Menge M_1 von Tieren eingefangen, gekennzeichnet und freigelassen.
- Nach Vermischung der Population Menge M_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:



$$\frac{\#M_1}{\#M} \approx \frac{\#(M_1 \cap M_2)}{\#M_2}$$

(#... – Größe der Mengen, hier Anzahl der Tiere; M – Menge aller Tiere, M_1, M_2 – beim ersten bzw. zweiten mal eingefangene Tiere; $M_1 \cap M_2$ – Menge der beide Male eingefangenen Tiere). Geschätzte Größe der Tierpopulation:

$$\#M \approx \frac{\#M_1 \cdot \#M_2}{\#(M_1 \cap M_2)}$$

Fehler statt Tiere

Zwei Inspekteure i finden jeweils eine Menge von M_i Fehlern:

$$\#M \approx \frac{\#M_1 \cdot \#M_2}{\#(M_1 \cap M_2)}$$

($\#(M_1 \cap M_2)$ – Anzahl der von beiden Inspekteuren unabhängig voneinander gefundenen gleichen Fehler; $\#M$ – geschätzte Anzahl der vorhandenen Fehler). Die geschätzte Fehlerüberdeckung ist das Verhältnis der Anzahl der insgesamt von beiden Inspekteuren gefundenen Fehler $\#(M_1 \cup M_2)$ zur geschätzten Gesamtfehleranzahl $\#M$:

$$IFC \approx \frac{\#(M_1 \cup M_2)}{\#M} \approx \frac{\#(M_1 \cap M_2) \cdot \#(M_1 \cup M_2)}{\#M_1 \cdot \#M_2}$$

Gebunden an die Annahmen:

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

Beispiel 2. Inspektionsergebnisse für ein Programm:

- Inspekteur 1: 228 gefundene Fehler, davon 156 funktionale.
- Inspekteur 2: 237 gefundene Fehler, davon 163 funktionale.

Schnittmenge: 105 Fehler, davon 73 funktionale.
Welche Schätzwerte ergeben sich nach dem Capture-Recapture-Verfahren für

1. die Gesamtfehleranzahl,
2. die Inspektionsfehlerüberdeckung?

1. Gesamtfehleranzahl:
$$\#M \approx \frac{\#M_1 \cdot \#M_2}{\#(M_1 \cap M_2)}$$
2. Inspektionsfehlerüberdeckung:
$$IFC \approx \frac{\#(M_1 \cap M_2) \cdot \#(M_1 \cup M_2)}{\#M_1 \cdot \#M_2} \tag{1}$$

Fehler	$\#M_1$	$\#M_2$	$\#(M_1 \cap M_2)$	$\#M$	IFC
alle	228	237	105	515	70%
funktional	156	163	73	348	71%
sonstige	72	74	32	166	68%

Vertrauenswürdigkeit der Schätzung

Zufälliger Fehler:

1. Als Richtwert ist die Intervallbreite, um die im Mittel Zählwerte von ihren Schätzwerten abweichen, das zwei bis fünffache der Wurzel aus dem Zählwert (FS3, Abschn. Bereichsschätzung). Für einen Schätzwert 100 nicht erkannte Fehler beträgt das Intervall für den Erwartungswert $[80, 120]$ bis $[50, 150]$.
2. Bei Multiplikationen und Divisionen addieren sich die relativen Schätzfehler. In Gl. 1
$$IFC \approx \frac{\#(M_1 \cap M_2) \cdot \#(M_1 \cup M_2)}{\#M_1 \cdot \#M_2}$$
von vier Zählwerten, d.h. man benötigt 4^2 mal so große Zählwerte um die IFC mit derselben Genauigkeit/Irrtumswahrscheinlichkeit zu schätzen.

Die Zählwerte für die Fehleranzahlen müssten bei 10^3 bis 10^4 liegen, was sie in der Praxis nicht tun werden. Hinzu kommen systematische Fehler ...

Systematische Fehler:

- Capture-Recaptur unterstellt für alle Fehler gleiche Erkennungswahrscheinlichkeit. Wenn gut erkennbare Fehler viel besser als schlecht erkennbare nachgewiesen werden, gelten die Schätzwerte nur für die gut erkennbaren.
- Capture-Recaptur verbietet Informationsaustausch zwischen den Inspekteuren. Falls es doch einen Informationsaustausch gibt, vergrößert der die Menge der gleichen gefundenen Fehler $M_1 \cap M_2$ gegenüber einer unabhängen Suche.
- Wenn die Inspekteure ihre Fehlerlisten voneinander abschreiben $M_1 = M_2$, ergibt sich als Schätzwert für die Inspektionsfehlerüberdeckung 100%.

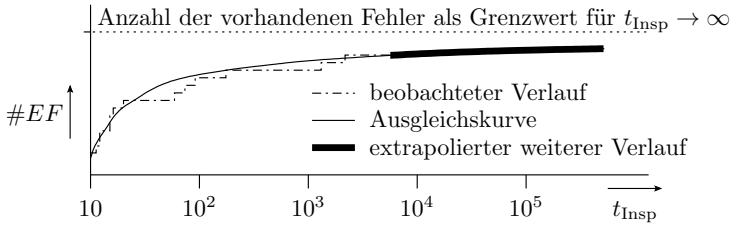
3.2 Inspektion als Zufallstest

Inspektion als Zufallstest

Einbeziehung, dass Fehlernachweiswahrscheinlichkeiten auch bei einer Inspektion um Größenordnungen variieren.

- Aufzeichnung der Anzahl der gefundenen Fehler in Abhängigkeit von der Inspektionsdauer.
- Abschätzen des weiteren Verlaufs.

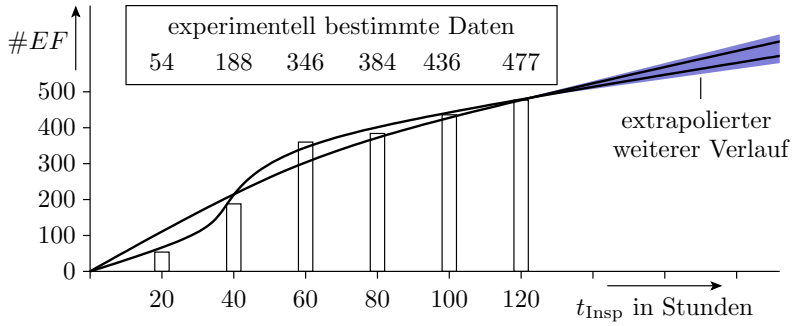
- Gesamtfehleranzahl ist der Grenzwert für eine unendliche Inspektionsdauer[7]:



Experiment mit einem Inspekteur[8]

Inspektion des Buchmanuskripts [2] plus Beispielprogramme:

- Anzahl der gefunden Fehler in Abhängigkeit von der Inspektionsdauer.



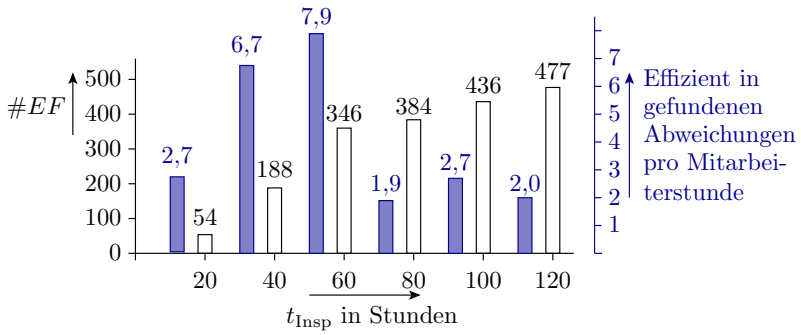
Unterschiedliche Approximationsmöglichkeiten für die weitere Abnahme der zu erwartenden Anzahl der nicht gefundenen Fehler, z.B. Abnahme der zu erwartenden Anzahl nicht gefundenen Fehler mit einem Exponenten $0 < k < 1$ (vergl. TV_F1):

$$\mathbb{E}\left[X\left(t_{\text{Insp}}\right)\right] = \mathbb{E}\left[X\left(t_0\right)\right] \cdot \left(\frac{t_{\text{Insp}}}{t_0}\right)^{-k}$$

(X – Anzahl der nicht nachweisbaren Fehler). Auch dieses Verfahren hat erhebliche Schätzfehler.

Untschiede zwischen Inspektion und Zufallstest

Bei einem Zufallstest nimmt die Effizienz (gefundene Abweichungen pro Mitarbeiterstunde) mit der Testdauer ab, weil nicht erkannte Fehler tendenziell schlechter als erkannte Fehler nachweisbar sind.



Die Beispielinspektion hatte offenbar eine »Anlernphase«, in der die Effizienz mit der Inspektionsdauer zugenommen hat.

[7] Untersuchungen in dieser Richtung in der Literatur noch nicht gefunden.
[8] Bachelor-Arbeit von Yu Hong.

- 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.

3.3 Inspektionstechniken

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]

Zertifizierung, Beispiel DO178A

Zertifizierungen: Verfahren zum Nachweis bestimmter Anforderungen

- für Produkte, Diensleistungen,
- Entstehungsprozesse, Informationssicherheit, ...

Inspektionstechnik mit speziellen Rollenverteilungen, Abläufen und Verantwortlichkeiten.

Der Standard DO-178A[9] benennt 71 Aspekte für die Zertifizierung von sicherheitskritischer Software für die Luftfahrt. Für die spezifizierten SW-Anforderungen ist bei der Zertifizierung zu kontrollieren, dass

- die Zielfunktionen in der SW-Spezifikation korrekt erfasst sind,
- die Zuordnung zwischen SW-Anforderungen und Zielfunktionen nachvollziebar ist.
- die SW-Anforderungen zur mit geplanten HW und Basis-SW (Betriebssystem) passen,
- die Algorithmen zur Anforderung passen.

[9] DO-178: Software Considerations in Airborne Systems and Equipment Certification, a guideline for safety of safety-critical software used in certain airborne systems.

Zertifizierung (DO178A) – Fortsetzung

Bei die SW-Implemetierung ist der geplante Umgang mit folgenden potentiellen Problemquellen darzulegen (Vermeidung (wie?), statische/dynamische Tests (welche?), FF-Behandlung (wie?), ...:

- Speicher- und Stack-Benutzung
- WB-Überläufe bei arithmetischen Operationen (Festkomma, Gleitkomma, Fehlerbehandlung, ...
- Ressourcen-Beschränkung, Resourcen-Konflikte
- Worstcase Ausführungszeit, Cache-Verwaltung,
- Ausnahme- und FF-Behandlung,
- nicht initialisierte Variablen, unbenutzte Variablen,
- Datenintegrität und Wettläufe im Zusammenhang mit Tasks und Interrupts, ...

Systemtests müssen Anforderungen zugeordnet sein. Anderenfalls sind die Tests wegzulassen oder die Anforderungen zu ergänzen.
⇒ Wie verträgt sich das mit fehlerorientiert ausgewählten Tests, Modultests, langen Zufallstests, Robustheitstests, ... ?

4 Kontrolle digitaler SL

Format und Daten

Digitale Informationseinheiten, also auch die für die Ein- und Ausgabedaten von SL haben immer ein Format:

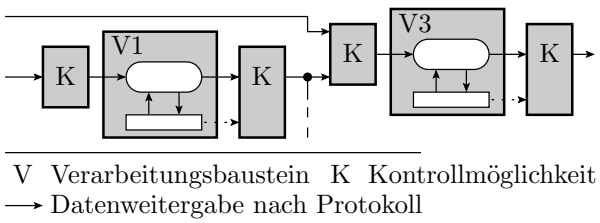
- Typ, Anzahl und Größe der Datenobjekte,
- Zeitschranken und andere Protokoll- und Signaleigenschaften,
- Wertebereiche, Invarianten, Syntax von Texteingaben, ...

mit überprüfbaren Merkmalen. Eine SL mit Formatfehler ist eine FF. Eine SL ohne Formatfehler kann, muss aber nicht ok. sein.

Formatkontrollen

- beschränken sich auf eingabeunabhängige Merkmale,
- sind damit nur Kontrollen auf Zulässigkeit, aber
- reagieren oft recht sensibel auf Fehler und FF.

4.1 Schnittstellen, 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-Protokoll, ...

Für Prozessoren, APIs, ... hunderte Seiten Doku

- 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 , ...

⇒ unzählige Möglichkeiten für Formatkontrollen.

Festlegung von Schnittstellenformaten

Der Entwurf einer IT-Funktionseinheit beginnt in der Regel mit den Schnittstellenfestlegungen:

- Anzahl und die Typen der Eingabeparameter und Rückgabewerte,
- Vorbedingungen, die die Eingabe erfüllen muss, ...
- Auf ausreichende Kontrollmöglichkeiten achten!

»Good Practice« zur Fehlervermeidung:

- verständlich: Vermeidung fehlerhafter Ansteuerung durch Fehlinterpretation der Doku, ...
- Berücksichtigung eingeplanter Erweiterung: Vermeidung von Fehlern durch Nachbesserungen.
- Standardisierte Formate, Protokolle, Sprachen, ...

Prüfgerechte Funktionsgrenzen:

- gut dokumentierbares EA-Verhalten, deterministisch,
- einfach zu findende Testbeispiel und Sollausgaben,
- einplanen von Kontrollen, ...

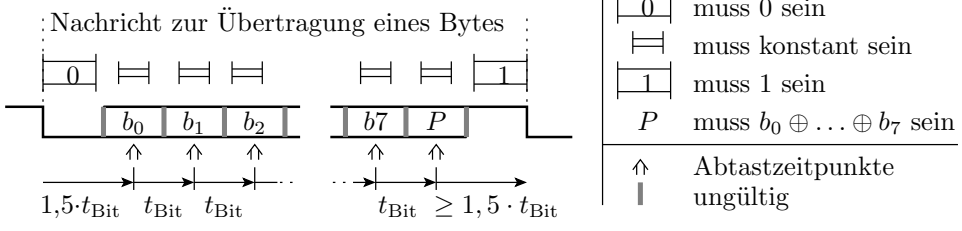
[Sprachenabhängige statische & einprogrammierte dynamisch Kontrollen, Python Typkontrolle Laufzeit, C statisch Zeigertyp, RUST Test/Release]

Busprotokoll, Beispiel UART

[HW-Schnittstelle, Bus, Bündel Leitungen + Stapel Doku]

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.



Kontrollierbare Formateigenschaften

Erkennbare Formatfehler

- Paritätsfehler: 1-Bit-CRC, $p_E \approx 50\%$ (siehe später Abschn. 3.75).
- 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 FF-Bits in Spezialregistern.

[USART Fehlerbits Atmel]

CAN-Bus: Vernetzung Fahrzeugsteuergeräte

ID	Nachrichtennummer	CRC	Prüfkennzeichen
DLC	Datenlängencode	EOF	Ende des Datenrahmens

Erkennbare Formatfehler:

- 15-Bit CRC als PKZ. Erkennungssicherheit (siehe später Abschn. 3.75)

$$p_E = 1 - 2^{-15} \approx 1 - 3 \cdot 10^{-5}$$

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

Fehlerbehandlung:

- Wiederholung bei Nachrichtenkollision,
- Notfallreaktion bei Ausfall anderer Steuergeräte,
- Einträge der FF-Nummern in den Fehlerspeicher, ...

[Kollisionserkennung]

Ethernet-Paket

Ethernet-Paket:

Sicherungsschicht			MAC-Empfänger	MAC-Absender	Protokolltyp	Nutzlast max. 1500 Bytes	Prüfkennz. CRC	
Bitübertragungsschicht	Präambel	Startbyte						Lücke zum nächsten Paket
Byteanzahl	7	1	6	6	2	46 bis 1500	4	12

Erkennbare Formatfehler:

- Prüfkennzeichenfehler. 4-Byte CRC, Erkennungssicherheit (siehe später Abschn. 3.75)

$$p_E = 1 - 2^{-32} \approx 1 - 2 \cdot 10^{-10}$$

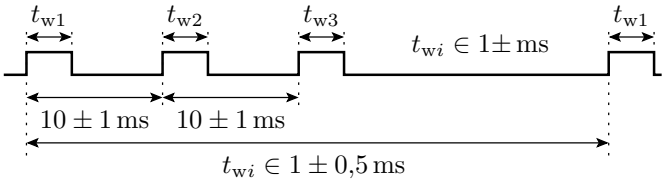
- Soll-/Ist-Abweichung konstante Bytes.
- Datenwechsel in unzul. Zeitfenstern,
- Unzul. Nutzdaten, ...

Überwachung durch die Prozessor-Hardware. Benachrichtigung der Software über Spezialbits. Fehlerbehandlung:

- Wiederholte Anforderung nicht erhaltener Pakete, ...

Signalintegrität, Beispiel PWM

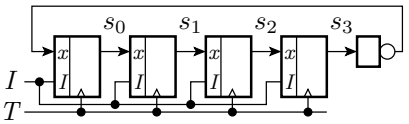
Pulsweitenmodulierte Signale (PWM) codieren die Information in den Pulsbreiten t_{wi} und werden u.a. zur Datenweitergabe von Sensoren an Rechner und von Rechnern an Aktoren genutzt.



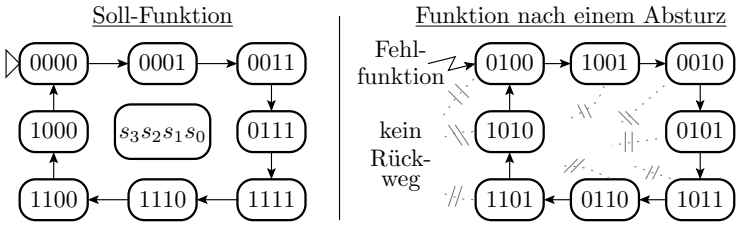
Überwachbare Formateigenschaften:

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

4.2 Zeitüberwachung

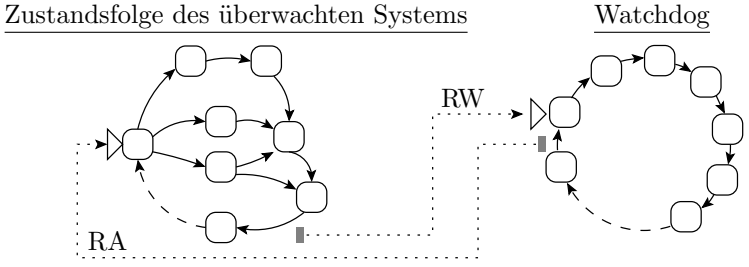


Redundante Zustände und Systemabsturz



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

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]

4.3 Syntaxtest

Syntaxtest

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

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

Statischer Test für manuelle erstellte oder bearbeitete Programme und Programmeingaben.

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.

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

EBNF als Beispielregelwerk für formale Sprachen

Beschreibungsmittel der EBNF (Eweiterte Backus-Naur-Form[10]) zur Definition von Sprachregeln:

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

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

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

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|...
```

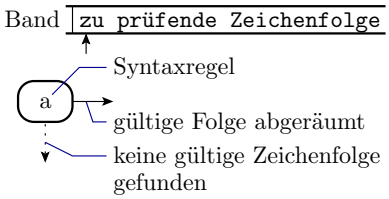
Beispielzeichenfolgen:

```
'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 dem Anfang.

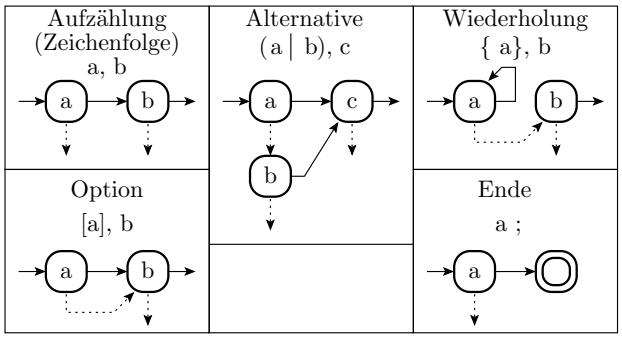
In jedem Automatenzustand wird versucht, eine Zeichenfolge nach der definierten Syntaxregel abzuräumen:



- 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.

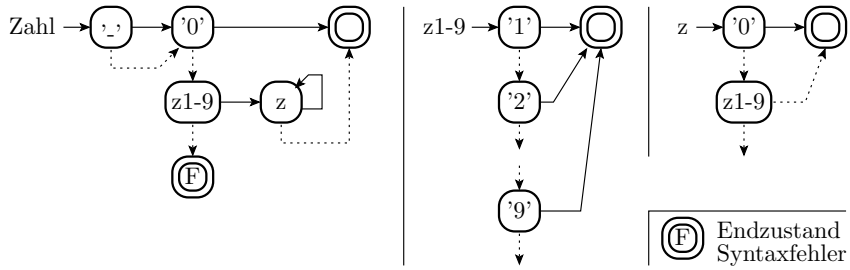
$\downarrow$-Übergänge ohne dargestellten Zielknoten enden im Fehlerzustand.

Von der EBNF zum Automaten



Beispiel:

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

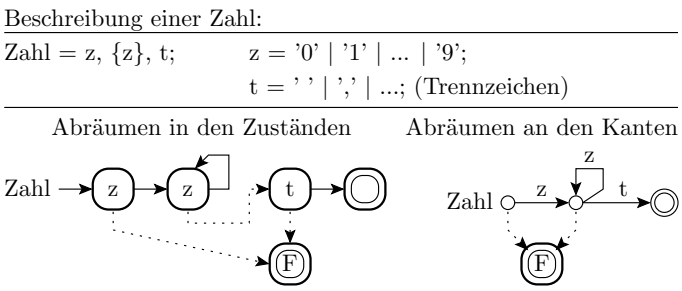


Welche Zustände durchläuft der Automat. Was erkennt er?

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

Abräumen an den Kanten

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



Zusammenfassung

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.

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 Programmier- und Datenbeschreibungssprachen, Eingabemasken]

4.4 Invarianten, WB

Invarianten

Invariante: die unabhängig von den zu verarbeitenden Daten immer erfüllt sein müssen. Überprüfbar

- durch Beweis, bei der Compilierung,

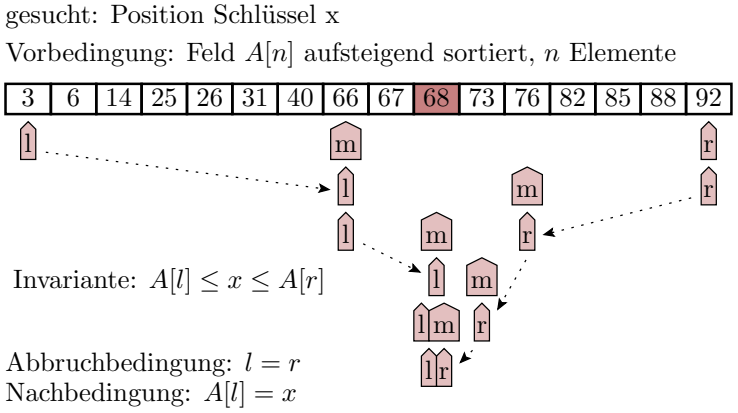
- durch Überwachung während der Laufzeit.

  [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.
- Geschlosses physikalisches System: Energie, Impuls, ...
- Iteration: Vorbedingungen, Schleifeninvarianten, Nachbedingungen.
- ...

Beispiel mit einer Schleifeninvarianten



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.

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:

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

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, ...

Tatsächliche Erkennungswahrscheinlichkeit der WB-Überwachung:

$$p_{\mathrm{E}} \ll 1 - 10^{-8}$$

Eine einzelne Wertebereichskontrolle hat in der Regel nur eine geringe Erkennungswahrscheinlichkeit, aber es sind sehr viele Kontrollen (auch für Überläufe) möglich, z.T. schon zur Compile-Zeit; Laufzeit-WB-Überwachung vom Compiler automatisch einfügbar.

Die Leistungsfähigkeit von Wertebereichskontrollen liegt in der Vielzahl der Kontrollmöglichkeiten.

Erhöhung der Erkennungswahrscheinlichkeit von WB-Kontrollen:

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

4.5 Fehlererk. Codes

Wiederholung Informationsredundanz

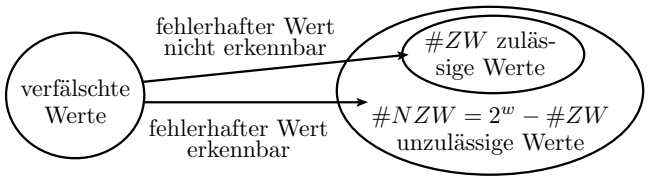
Die Unterscheidung von $\#ZW$ zulässigen Werten verlangt

$$w \geq \log_2(\#ZW)$$

Bits, mit denen 2^w Werte darstellbar sind. Davon sind

$$\#NZW = 2^w - \#ZW$$

unzulässig. Eine FF verfälscht einen zulässigen Wert in entweder einen anderen zulässigen oder einen unzulässigen Wert:



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

$$p_{\mathrm{E}} = \mathbb{E}[FFC] \geq 1 - \frac{\#ZW}{2^w}$$

$$p_{\mathrm{E}} = \mathbb{E}[FFC] \geq 1 - \frac{\#ZW}{2^w} \tag{2}$$

($\#ZW$ – Anzahl der zulässigen Werte; w – Bitanzahl zur Darstellung). Mit einer Informationsredundanz von r zusätzlichen (redundanten) Bits:

$$r = w - w_{\min}$$

$$p_{\mathrm{E}} = 1 - \underbrace{\frac{\#ZW}{2^{w_{\min}}}}_{\leq 1} \cdot 2^{-r} \geq 1 - 2^{-r}$$

Für eine Erkennswahrscheinlichkeit von praktisch eins genügen $r \approx 10 \ldots 20$ redundante Bits. Aber die Voraussetzungen:

- Erkennen aller unzulässiger Werte und
- Abbildung fehlerhafte Werte *gleichmäßig auf zulässige und unzulässige* Werte.

sind nicht immer erfüllt.

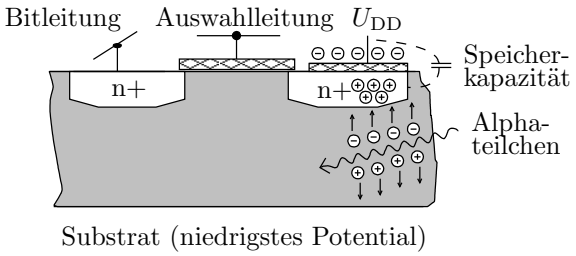
Beispiel Rechtschreibtest

Wort im Wörterbuch enthalten?

- Maskierung: falsches Wort, das im Wörterbuch enthalten ist, z.B. »Maus« statt »Haus«. Größenordnung der Erkennungswahrsch.
$$p_{\mathrm{E}} \approx 80\%$$
- Phantom-FF: zulässiges Wort nicht im Wörterbuch. Größenordnung der Phantom-FF-Rate:
$$\zeta_{\mathrm{Phan}} \approx \frac{1\ \mathrm{Phantom-FF}}{100\ \mathrm{kontrollierte\ Worte}}$$

Bei Abbildung fehlerhafte Werte gleichmäßig auf zulässige und unzulässige Werte Anzahl der mit #Bytes darstellbaren Zeichenketten:
$$2^{8\cdot\#\mathrm{Bytes}}$$
z.B. 2^{80} für 10 Zeichen große Worte, zulässig davon angenommen 10^6:
$$p_{\mathrm{E}} \approx 1 - \frac{10^6}{2^{80}} \approx 1 - \frac{10^6}{10^{24}} = 1 - 10^{-18} \gg 80\%$$

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: vieler Stunden oder Tage.
- Energie eines Alphateilchen: 5 MeV. Energieverlust bei der Generierung eines Elektronen-Loch-Paares $\approx 3{,}6\,\mathrm{eV} \Rightarrow$ Generierung von $\approx 10^6$ Ladungsträgerpaaren. Reichweite $\approx 89\,\mu$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 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 nur ein Bit je Datenobjekt verfälscht, genügt ein fehlererkennender Code für Einzelbitfehler (Paritätsbit).

Für seltene unabhängige Bitverfälschungen ist die Anzahl der verfälschten Bits je Datenobjekt X poissonverteilt:

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

(λ – Erwartungswert). Erkennungswahrscheinlichkeit Paritätstest:

$$p_\mathrm{E} = \frac{\mathbb{P}[X=1]}{\mathbb{P}[X\geq 1]} = \frac{\mathrm{e}^{-\lambda}\cdot\lambda}{1-\mathrm{e}^{-\lambda}}$$

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

$$p_\mathrm{E} = \frac{(1-\lambda)\cdot\lambda}{1-(1-\lambda)} = 1-\lambda$$

Gleichm. Abb. auf zul. und unzul. Werte und $r=1$ Paritätsbit $p_\mathrm{E} = 50\%$

Prinzip der fehlererkennenden Codes

Die Abschätzung der Maskierungswahrscheinlichkeit

$$p_\mathrm{M} = 1 - p_\mathrm{E} \approx 2^{-r}$$

kann um Zehnerpotenzen

- zu optimistisch sein, siehe Rechtschreibtest, oder
- zu pessimistisch sein, sie Paritätstest Datenspeicherung und Übertragung.

Fehlererkennde Codes

Sicherstellung, dass fehlerhafte Werte gleichmäßig auf zulässige und unzulässige Werte abgebildet werden durch pseudo-zufällige Zuordnung der zulässigen auf die $\geq 2^r$mal so große Menge der möglichen Werte.

Arithmetische Codes

Bildung durch lineare arithmetische Operationen, z.B. Multiplikation mit Kontante $K = 34562134$:

$$s = K \cdot x$$

x – Wert; s – Codewort. Kontrolle auf Zulässigkeit:

```
if (s%K){FFF-Behandlung}; // s%K Divisionsrest
```

Wegen Linarität besteht für Verfälschungen $\Delta s \neq 0$ unabhängig vom Wert x bzw. vom Sollwert s Erkennungswahrscheinlichkeit

$$p_\mathrm{E} \approx 1 - K^{-1}$$

Bei sehr großen unbekannten Primzahlen als Faktor K ist es selbst vorsätzlich kaum möglich, gültige Codeworte in andere gültige Codeworte zu verfälschen. Einsatz auch zur kryptographischen Verschlüsselung.

[Kryptographie: Angreifer darf auch Schlüssel nicht finden oder erraten können]

Zyklische Codes, Polynommultiplikation

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

Codierung

```
        10010101101
⊙           10011
-----------------------
⊕     100110101101
⊕      10010101101
⊕       00000000000
⊕      00000000000
⊕ 10010101101
-----------------------
  10001110100111
```

Decodierung

```
100011100100111 : 10011 = 10010101101
⊕10011
-------
  10110
 ⊕10011
 -------
   10101
   10011
   -----
    11000
   ⊕10011
   -------
    10111
   ⊕10011
   -------
     10011
    ⊕10011
    -------
Rest: 0000        (unverfälscht)
```

Polynom-Multiplikation und Division

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:

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

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

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

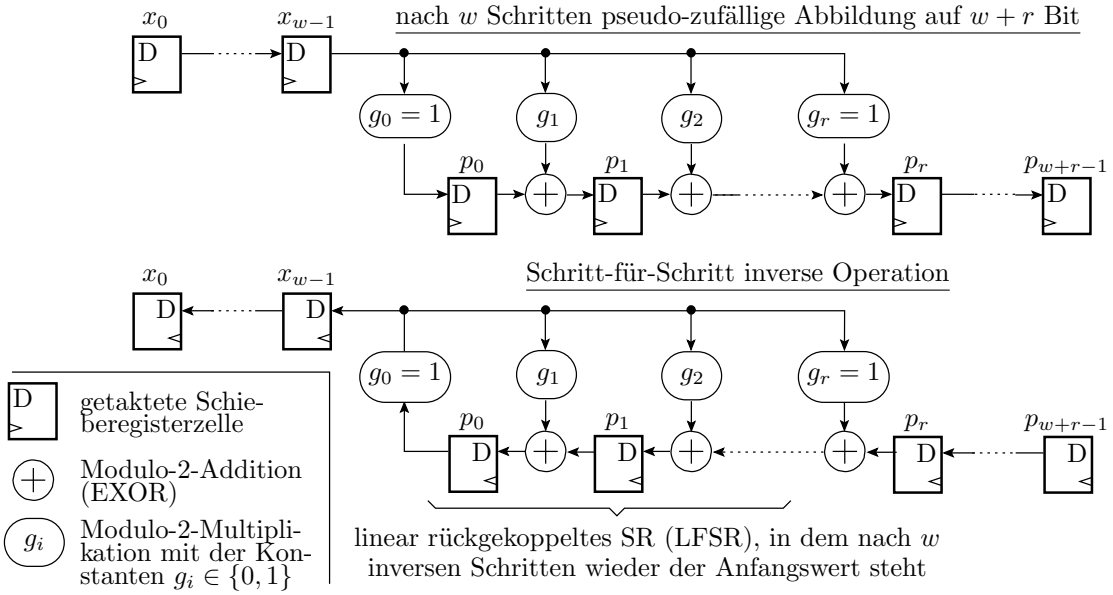
repräsentiert denselben Bitvektor, der für die Multiplikation der Folgen auf der Folie zuvor berechnet wurde. Die Polynomdivision:

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

liefert ohne Rest das Polynom der Originalfolge.

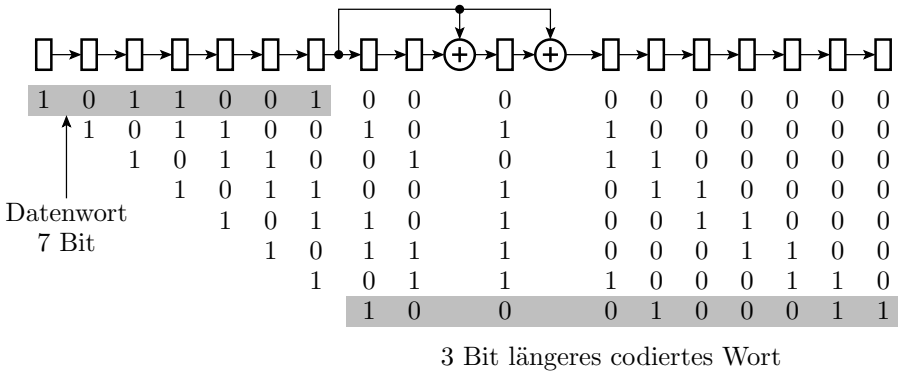
Polynommultiplikation und -Division mit SR

Codierung mit SR (**S**hift **R**egister), Decodierung mit LFSR (**L**inear **F**eedbeack **S**hift **R**egisters).



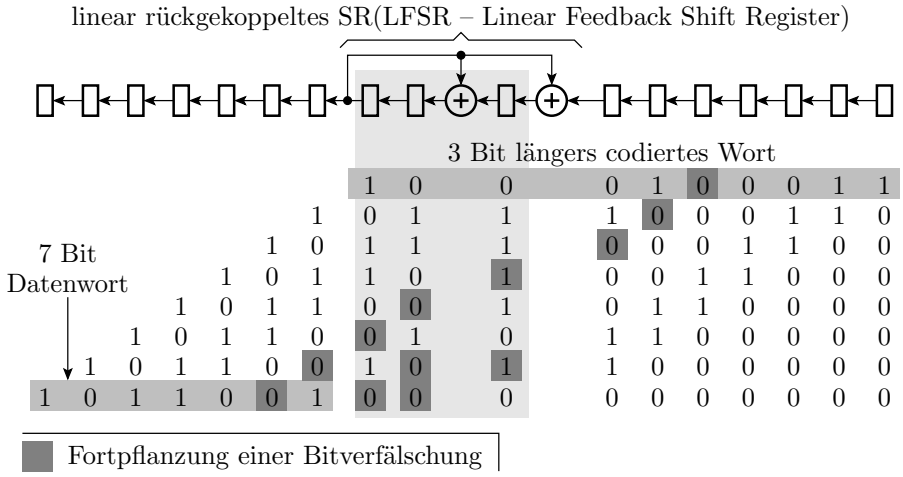
Erkennungswahrscheinlichkeit: $p_E = 1 - \frac{2^w}{2^{w+r}} = 1 - 2^{-r}$

Beispiel für die Codierung



- 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}.

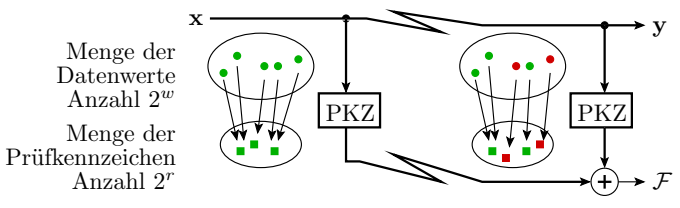
Rückgewinnung und Kontrolle



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

4.6 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:

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

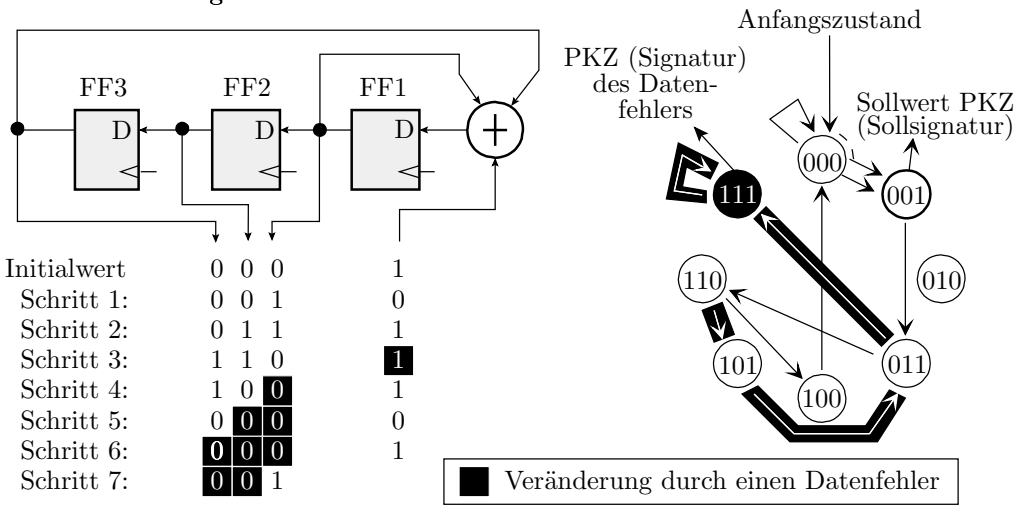
Prüfsummen

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[11]: $p_{\mathrm{E}} \approx 1 - 2^{-4}$. Bei »doppelter Genauigkeit« bilden sich Verfälschungen vorzugsweise auf die niederwertigen Bits ab. Maskierungswahrscheinlichkeit: $2^{-4} > 1 - p_{\mathrm{E}} \gg 2^{-8}$.

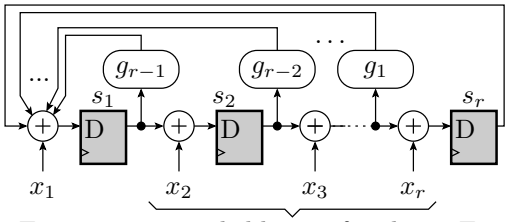
Prüfkennzeichenbildung mit LFSR[12]



Das Prüfkennzeichen wird wie bei der CRC-Decodierung mit einem linear rückgekoppelten Schieberegister (LSFR) gebildet. Im Beispiel hat das LSFR im Gegensatz zur Polynomdivision Seite 74 eine zentrale Rückführung. Abbildung auch pseudo-zufällig.

Allgemeine Struktur von LFSR

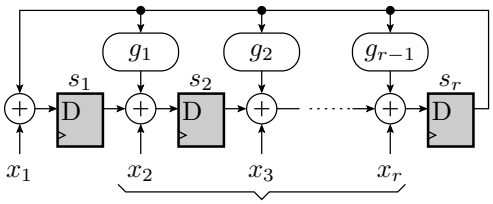
Für die Bildung auf Prüfkennzeichen ist es nur wichtig, dass die Abbildung pseudo-zufällig hinsichtlich der zu erwartenden Verfälschungen erfolgt. Diese Eigenschaft hat auch ein rückgekoppeltes Schieberegister, bei dem in jedem Zeitschritt mehrere Mehrere Eingabebits modulo-2 zu den Registerzuständen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



Erweiterungsmöglichkeit auf mehrere Eingänge

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

[11] Kein Nachweis für vertauschte Summationsreihenfolge.
[12] LFSR – **L**inear **F**eeback **S**hift **R**egister.



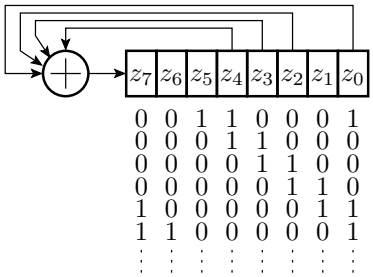
Erweiterungsmöglichkeit auf mehrere Eingänge

Die Koeffizienten g_i der Rückführung, bei der Polynom-Division das Divisor-Polynom, bestimmen die autonome Zyklusstruktur[13]. 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.

Autonome Zykluslänge und -struktur von LFSR

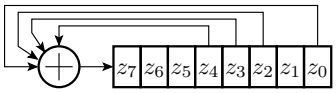
Autonom bedeutet Zyklusstrukur für Eingabewerte »alle 0« oder für LFSR ohne Eingänge. Beispiel 8-Bit autonomes LFSR:



Übergangsfunktion:

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

Bestimmung der Zykluslänge durch Simulation



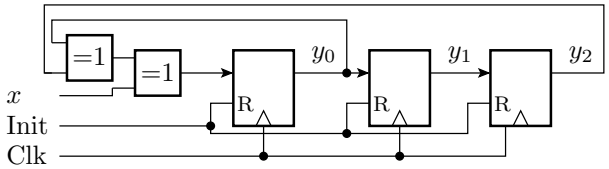

```
...
#define ZA 0x31
uint8_t z = ZA;      // Startwert setzen
while(1){
  z = (z>>1) ^ (z<<7) ^ ((z<<5)&0x80)
       ^ ((z<<4)&0x80) ^ ((z<<3)&0x80);
  <Ausgabe von z>
  if (z==ZA) break; // bis wieder Anfangswert
  ...               // weiter mit nicht enthal-
}                   // tenem Zustand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

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

[13] Zyklusstruktur ohne Eingaben.

Beispielaufgabe

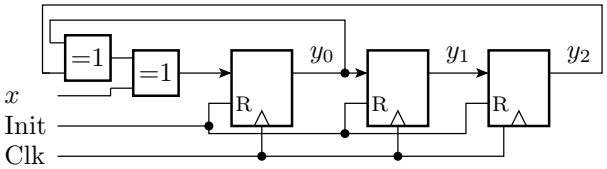
Gegeben ist folgendes linear rückgekoppelte Schieberegister:



	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 $\mathbf{y} = y_2y_1y_0$ hat die Datenfolge »1001100101111010« bei Abbildung beginnend mit dem höchstwertigen Bit. Startwert 000.
2. Wie hoch ist Fehlererkennungswahrscheinlichkeit?

Lösung

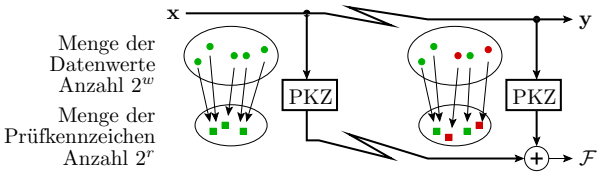


	x	y_2	y_1	y_0
0	1	0	0	0
1	0	0	0	1
2	0	0	1	1
3	1	1	1	1
4	1	1	1	1
5	0	1	1	1
6	0	1	1	0
7	1	1	0	1
8	0	0	1	1
9	1	1	1	1
10	1	1	1	1
11	1	1	1	1
12	1	1	1	1
13	0	1	1	1
14	1	1	1	0
15	0	1	0	0
PKZ:		0	0	1

Erkennungswahrscheinlichkeit:

$$p_E \approx 1 - 2^{-3} = 87{,}5\%$$

Zusammenfassung



Datensicherung mit Prüfkennzeichen:

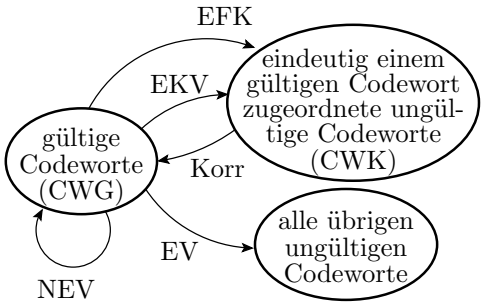
- Geringer Berechnungsaufwand.
- Geringer Zusatzaufwand für Datenübertragung und -speicherung (r zusätzlich gespeicherte / übertragene Bits für eine Datenobjekt beliebiger Größe).
- Maskierungswahrscheinlichkeit 2^{-r}. Mit ausreichendem r immer vernachlässigbar klein.

Dateien, Nachrichten etc. werden mit Prüfkennzeichen übertragen und gespeichert. In Software sind die PKZ bevorzugt Prüfsummen, in Hardware LFSR. Fehlererkennende Codes nur, wenn die übertragenen und gespeicherten Daten nicht lesbar sein sollen.

4.7 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:

$$2^{\#\text{Bit}} \geq \#CWG + \#CWG \cdot \#CWK/CWG$$

(#Bit $= w + r$ – Bitanzahl; $\#CWG$ – Anzahl gültige Codeworte; $\#CWK/CWG$ – 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:

$$\#CWK/CWG = \#\text{Bit}$$

Mindestbitanzahl:

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

Für #CWG = 256 gültige Codeworte:

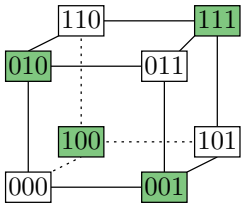
$$\begin{aligned} 2^{\#\text{Bit}} &\geq 256 \cdot (1 + \#\text{Bit}) \\ \#\text{Bit} &\geq 12 \end{aligned}$$

Probe:

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

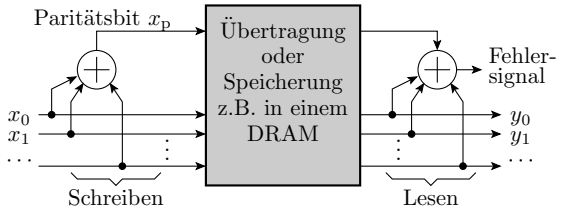
Hamming-Distanz

Die Hamming-Distanz 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 $Ham \geq k + 1$.
- Um k-Bit-Fehler korrigieren zu können, ist eine Hamming-Distanz von $Ham \geq 2 \cdot k + 1$ erforderlich.

Parität als Prüfkennzeichen ($Ham = 2$)



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

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

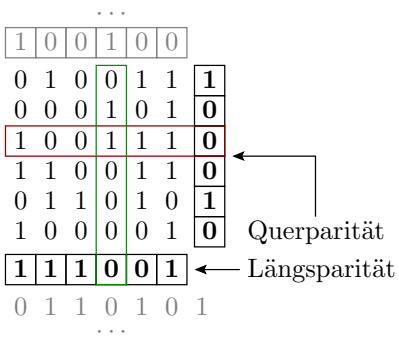
bei gerader Anzahl von Einsen »0« sonst »1«.

- Erkennt jede ungeradzahlige Anzahl von Bitverfälschungen.
- Wenn geradzahlige und ungeradzahlige Bitfehler gleichhäufig auftreten: $p_\mathrm{E} \approx 50\%$
- Bei unabhängiger Verfälschung mit Erwartungswert $\lambda \ll 1$ (typ. für DRAMs, vergl. Seite 67):

$$p_\mathrm{E} = 1 - \lambda$$

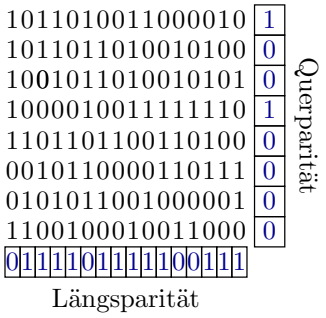
Kreuzparität (Fehlerkorrigierender Paritätscode)

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).

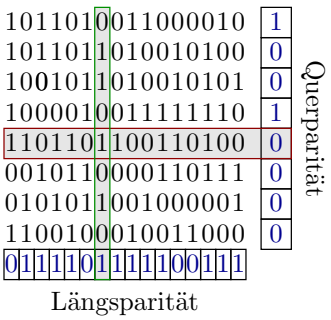


Beispielaufgabe

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.



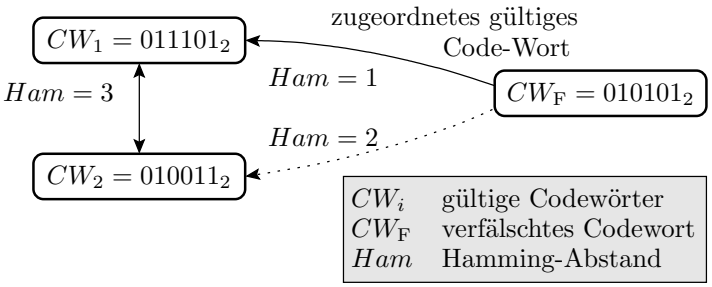
Lösung



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

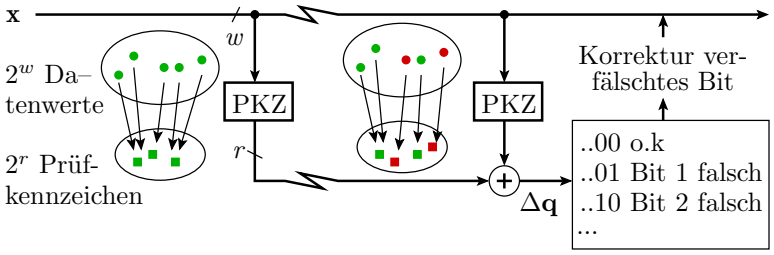
1-Bit fehlerkorrigierende Hamming-Codes

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



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

Konstruktion 1-Bit fehlerkorrigierender Code



Ergänzung des w-Bit Datenworts $\mathbf{x}$ um ein r-Bit Prüfkennzeichen so, dass die mod-2 Summen des übertragenen und des nach der Übertragung gebildeten Prüfkennzeichens die Bitnummer des verfälschten Bits ist[14]:

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

Prüfkennzeichen für 1 Byte Einzelbitkorrektur

Anzahl der Datenbits $w = 8$, minimale Prüfbitanzahl r:

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

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

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

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

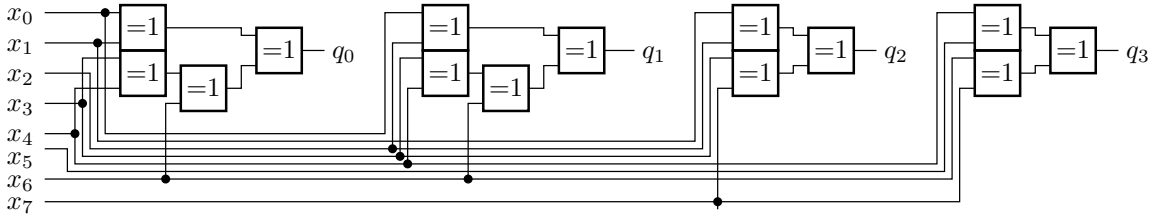
[14] Die Zuordung von Datenbit- und Prüfbitnummern zu Gesamtbitnummern muss dafür geeignet festgelegt werden.

Ersatz Bit- durch Daten- und Kontrollbitnr.

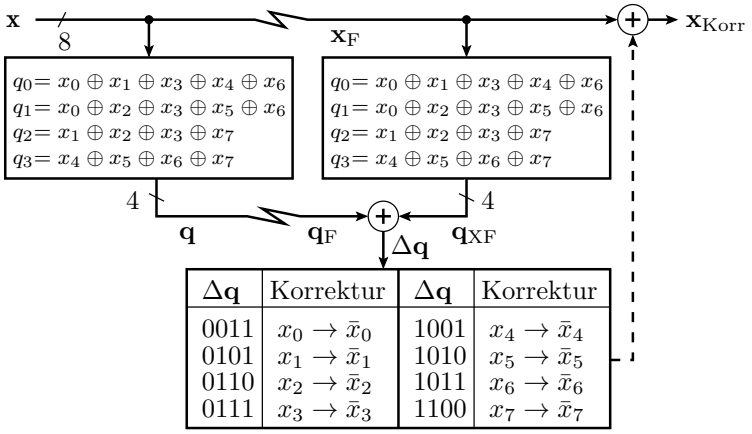
$$\begin{array}{lll} \Delta q_0 & = b_1 \oplus b_3 \oplus b_5 \oplus b_7 \oplus b_9 \oplus b_{11} & = q_0 \oplus x_0 \oplus x_1 \oplus x_3 \oplus x_4 \oplus x_6 \\ \Delta q_1 & = b_2 \oplus b_3 \oplus b_6 \oplus b_7 \oplus b_{10} \oplus b_{11} & = q_1 \oplus x_0 \oplus x_2 \oplus x_3 \oplus x_5 \oplus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

Bildung unverfälschtes PKZ aus unverfälschten Daten ($\Delta\mathbf{q} = 0$):

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



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 PKZ-Bildung im Sender und Empfänger.
- Differenzbildung durch bitweises EXOR.
- Wenn Datenbit verfälscht, Korrektur durch Invertierung.

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

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

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													
x = 0x8B													**b** =
b = 0xA9B													$\Delta\mathbf{q}$ =

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													
$\mathbf{x}$ = 0x8B	1	0	0	0	1	1	0	1	1	1	0	1	$\mathbf{b}$ = 0x8DD
$\mathbf{b}$ = 0xA9B	1	0	1	0	1	0	0	1	1	0	1	1	$\Delta\mathbf{q} = 12_{10}$

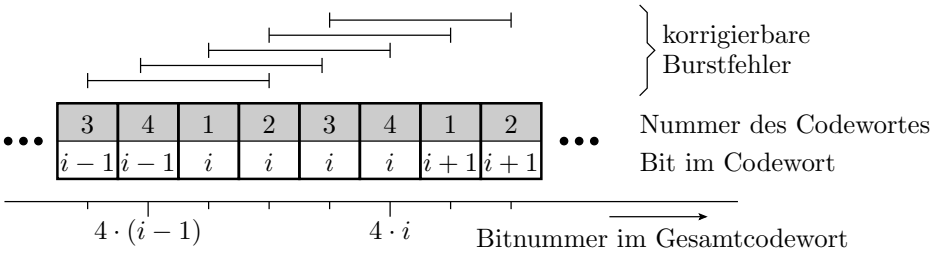
- Wert: 0x8B: Code-Wort 0x8DD
- Code-Wort 0x49B: Wert 0xA2 mit $\Delta\mathbf{q}$ = 0b1100; Invertierung von $b_{12} = x_7 \Rightarrow$ Wert: 0x22

4.8 Burstkorrektur

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												
x_1 = 0x8B												
x_2 = 0x22												
x_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	–	–	–	–	_	–	–	–	–	–	–	–
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

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	–	–	–	–	_	–	–	–	–	–	–	–	
x_1 = 0x8B	1	0	0	0	1	1	0	1	1	1	0	1	$\Delta\mathbf{q}_1 = 11_{10}$
x_2 = 0x22	0	0	1	0	1	0	0	1	1	0	1	1	$\Delta\mathbf{q}_2 = 11_{10}$
x_3 =0x9C	1	0	0	1	0	1	1	0	1	0	0	0	$\Delta\mathbf{q}_3 = 10_{10}$

Durch Burstfehler invertierte Bits 30 bis 32,

Zusammenfassung Formatkontrollen

Eine Service-Leistung umfasst

- Format (konstante, immer erfüllte Merkmale), und
- Daten (variable von der Eingabe und von internen Zuständen abhängige Merkmale).

Formatmerkmale:

- Anzahl und Typ der Datenobjekte, Wertebereiche, Zeitschranken,
- bei manueller Eingabe der Syntax und
- bei zu speichernden und zu übertragenden Daten Prüfkennzeichen.

Vor allem Prüfkennzeichen und die Syntaxkontrolle haben sehr gute fehlererkennende Eigenschaften.

WB und Typ-Kontrollen haben einzeln vielleicht nicht die größte Nachweiswahrscheinlichkeit für FF, aber es gibt sehr viele Kontrollmöglichkeit und wenn man die ausnutzt, entsteht insgesamt doch eine respektable FFC. Eine Überwachung von Abarbeitungszeitschranken erkennt alle Programmabstürze.

4.9 Wertekontrolle

Wertekontrolle

Die Kontrolle der Ergebniswerte einer SL ist keine Alternative zu den Formatkontrolle, sondern eine Ergänzung.

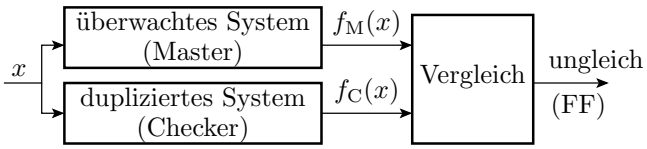
Möglichkeiten zur Wertekontrolle unter Testbedingungen:

- Vergleich mit vorab berechneten Sollwerten, exakt oder Bereichskontrolle.
- Vergleich mit »Golden Device« (Referenzsystem, dessen Ausgaben per Definition als korrekt gelten).
- Regressionstest (Sollwertberechnung mit der Vorgängerversion als Golden Device).
- Alles, was auch unter Betriebbdingungen funktioniert.

Möglichkeiten für Wertekontrollen unter Betriebsbedingungen:

- Mehrfachberechnung und Vergleich.
- Rückgewinnung der Eingaben und Vergleich,
- Funktionsspezifische Kontrollen.

Verdopplung und Vergleich



Geeignet für alle deterministischen Systeme, die aus Eingaben Ausgaben erzeugen. FF-Überdeckung gleich Diversität (vergl. F1, Absch. 3.2 Fehlerbehandlung. Überwachungsverfahren):

$$FFC = Div = \frac{\#DFF}{\#FF}$$

(Div– Diversität; $\#DFF$ – Anzahl der Master-FF, bei denen eine Wiederholung zum korrekten Ergebnis oder zu einer geänderten FF führt). Phantom-FF-Rate:

$$\zeta_{\text{Phan}} \approx \zeta_{\text{VS}} \cdot (1 - Div)$$

ζ_{Phan} – FF-Rate Vergleichssystem, ζ_{VS} – FF-Rate Vergleichssystem.

Nicht erkennbare, d.h. übereinstimmende FF haben praktisch immer eine gemeinsame Ursache. Warum?

[Abschätzung der Maskierungswahrsch. ähnlich wie bei PKZ]

Mögliche Ursachen für übereinstimmende FF:

- gleicher Entwurfs- oder Fertigungsfehler,
- dieselbe Störung, Eingabe- oder Zustandsverfälschung,
- Fehler in oder bei der Interpretation der Spezifikation,
- Fehler in der Entwurfssoftware (Compiler, ...), in genutzten Bibliotheken, Dokumentationen, ...

Mehrfachberechung und Vergleich ist brauchbar:

- wenn FF überwiegend durch Störungen oder Ausfälle entstehen,
- die Berechnungen diversitär erfolgen.

[Rückblick FS1 Korrektur FF durch Wiedeholung]

Schaffung von Diversität

Eine FFC größer als die natürliche Diversität erfordert zusätzliche konstruktive Aufwändungen:

1. Hardware-Diversität: Unterschiedliche HW.
2. Hardware-Entwurfsdiversität: Unabhängig entworfene HW.
3. Syntaktische Diversität: Unterschiedlich übersetzte Software.
4. Software-Diversität: Unabhängig entworfene SW.
5. Diversitärer Service-Anforderung: Ungeeignet für Verdopplung und Vergleich, da abweichende Sollwerte[15].

FF-Klassen, erkennbar durch die zusätzliche Diversifizierung:

1. zusätzlich Fertigungsfehler und Ausfälle der HW.
2. zusätzlich HW-Entwurfsfehler.
3. zusätzlich FF des Compilers.
4. zusätzlich SW-Entwurfsfehler.
5. zusätzlich Spezifikationsfehler.

[15] Standardverfahren zur manuellen Korrektur von FF durch Fehler.

Anmerkungen zur Software-Diversität

Software-Fehler als Hauptquelle für FF verlangen SW-Diversität:

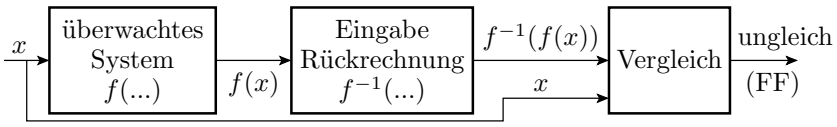
- Komplette Entwicklung mindestens zweimal.
- durch getrennte Teams, keine Kommunikation,
- aus einer nicht diversitären Spezifikation, ...

Ursprüngliche euphorische Meinung, dass so Diversität gegenüber allen Fehlern, außer denen in der Spezifikation erzielbar sei, nicht bestätigt. Die direkte oder indirekte Kommunikation der Entwicklungsteams über die Interpretation der Spezifikation, während des Test etc. trägt Gemeinsamkeiten in die Entwürfe. Neigung von Menschen, gewisse Fehler zu wiederholen, ... Erzielbare Divisität laut[16]

$$Div \leq 90\%$$

Eingaberückgewinnung und Vergleich

Für umkehrbare Funktion $f(x)$ mit $f^{-1}(f(x)) = x$ lassen sich die Ergebniswerte 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,
- Analog/Digital-Wandlung $\leftrightarrow$ Digital/Analog-Wandlung,
- Daten versenden $\leftrightarrow$ Daten empfangen, ...

Es ist nicht ausschließbar, dass sich FF von f und f^{-1} sich gegenseitig aufheben, aber wegen $f \neq f^{-1}$ sind Fehlern mit dieser Wirkung unwahrscheinlicher als bei Verdopplung und Vergleich.

Funktionsspezifische Kontrollen

Eine Reihe von Zielfunktionen bieten weitere Kontrollmöglichkeiten für das Ergebnis auf Richtigkeit:

- Sortieren: Sortierte Menge enthält alle Elemente der Originalmenge in der richtigen Reihenfolge.
- Suche einen Weg durch einen Graphen: Graph unverändert und Weg erfüllt die Zielvorgaben.
- Suche einen Test für den Nachweis eines Fehlers: Test weist den Fehler nach.

Solche funktionsspezifischen Kontrollen umgehen oft das Diversitätsproblem durch Verschiedenartigkeit von Berechnung und Kontrolle. Hohe Erkennungswahrscheinlichkeiten erzielbar. Leider gibt es für die meisten Zielfunktion keine derartige Kontrollmöglichkeit für die SL auf Richtigkeit.

5 Literatur

Literatur

[1] 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.

[2] Günter Kemnitz. *Technische Informatik 2: Entwurf digitaler Schaltungen*. Springer, 2011.

[16] U. Voges, Software-Diversität und ihre Modellierung - Software-Fehlertoleranz und ihre Bewertung durch Fehler- und Kostenmodelle, Springer (1989)

[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] 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.