Test und Verlässlichkeit Foliensatz 4: Statische Tests

Prof. G. Kemnitz

Institut für Informatik, TU Clausthal (TV_F4)

7. Juni 2018

Inhalt TV_F4: Statische Tests

Statische und dynamische Tests

Test: Verfahren zum Ausspühren von Fehlern.

- statische Test: direkete Kontrolle von Merkmalen,
- dynamischer Test: Ausprobieren der Funktion.

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

Statische kontrollierbare Merkmale von Entwurfsbeschreibung:

- Syntax, Entwurfsregeln,
- Regeln für die Dokumentation,
- partielle und totale Korrektheit, ...

Elektronische Baugruppen:

- Bestückung, Verdrahtung,
- optisch und mit anderen Verfahren kontrollierbare Produktmerkmale, ...

Statische SW-Tests

Statische Tests für SW und Entwurfsbeschreibungen

Besonderheiten von SW und Enwurfsbeschreibungen für HW, ...:

- Textbeschreibung.
- Kontrollierbare Vorgaben für Syntax und Semantik.
- Entstehung in Phasen, Spezifikation, Pflichtenheft, Architekturentwurf, ... mit Zwischenkontrollen.
- Fehlerbeseitigung durch Editieren von Texten.
- Agile Entwürfe: Weiterentwicklung und Fehlerbeseitigung durch Updates während der gesamten Nutzungsdauer.

Statische Tests:

- Inspektion (Review),
- Statische Code-Analyse,
- Überprüfung der Korrektheit, ...

Inspektion

Inspektion (Review)

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

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

Einordnung, Merkmale und Besonderheiten:

- wenn manuell, arbeitsaufwändig,
- zufälliger Fehlernachweis mit subjektiv geprägter Güte,
- Nachweis nicht funktionaler Fehler (Standardverletzungen, unsichere Beschreibungsmittel, ...),
- für frühe Entwurfsphasen geeignet,
- Know-How-Weitergabe.

Automatisierung anstrebenswert.

Kenngrößen einer Inspektion

Inspektionsfehlerüberdeckung:

$$IFC = \frac{\varphi_{\text{Erk}}}{\varphi}$$

(φ_{Erk} – Anzahl der nachweisbaren; φ – Anzahl aller (entstandenen) Fehler). Insgesamt oder getrennt für funktionale und sonstige Fehler.

Abschätzmöglichkeiten:

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

Weitere Bewertungsgrößen für Inspektionen nach [4]:

- Effizienz: Gefundene Abweichungen pro Mitarbeiterstunde.
- Effektivität: Gefundene Abweichungen je 1000 NLOC[1].

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

Beispielaufgabe

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?

Lösung

Gegeben: Programmgröße 10.000 NLOC. Arbeitsaufwand 200 Stunden. 228 gefundene Fehler, davon 156 funktionale. Geschätzte Gesamtfehleranzahl 300, davon 200 funktionale.

Gesucht: Inspektionsfehlerüberdeckung IFC, Effizienz (gefundene Fehler pro Mitarbeiterstunde). Effektivität (Gefundene Abweichungen je 1000 NLOC).

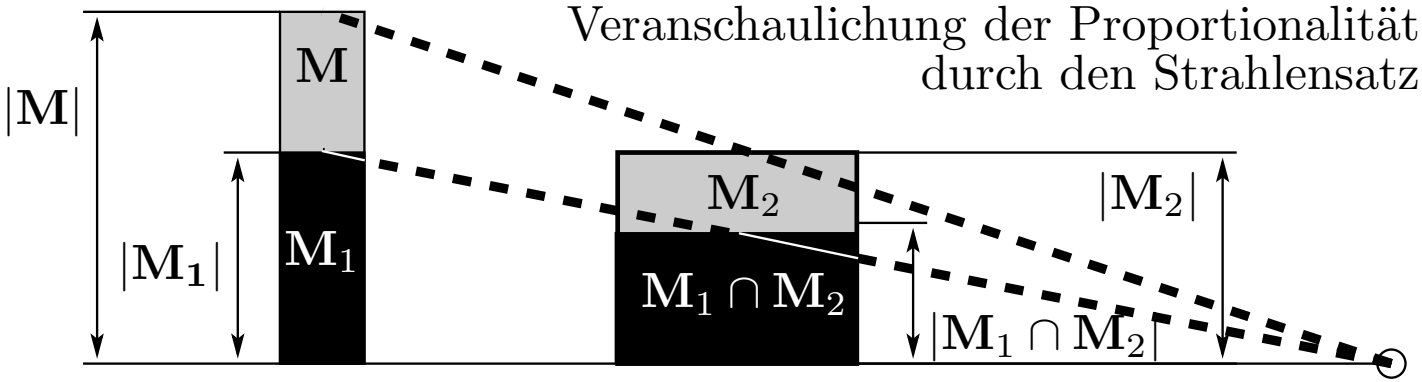
	gesamt	funktionale Fehler	sonstige Fehler
$IFC = \frac{\varphi_{\text{Erk}}}{\varphi}$	$\frac{228}{300}$	$\frac{156}{200}$	$\frac{72}{100}$
Effizienz	$\frac{228\text{ Fehler}}{200\text{ h}}$	$\frac{156\text{ Fehler}}{200\text{ h}}$	$\frac{72\text{ Fehler}}{200\text{ h}}$
Effektivität	$\frac{228\text{ Fehler}}{10.000\text{ NLOC}}$	$\frac{156\text{ Fehler}}{10.000\text{ NLOC}}$	$\frac{72\text{ Fehler}}{10.000\text{ NLOC}}$

Effizienz und Effektivität ergeben sich ausschließlich aus Zähl- und gemessenen Zeitwerten. In die IFC geht der schwer zu überrüfende Schätzwert für die Anzahl der nicht erkannten Fehler ein.

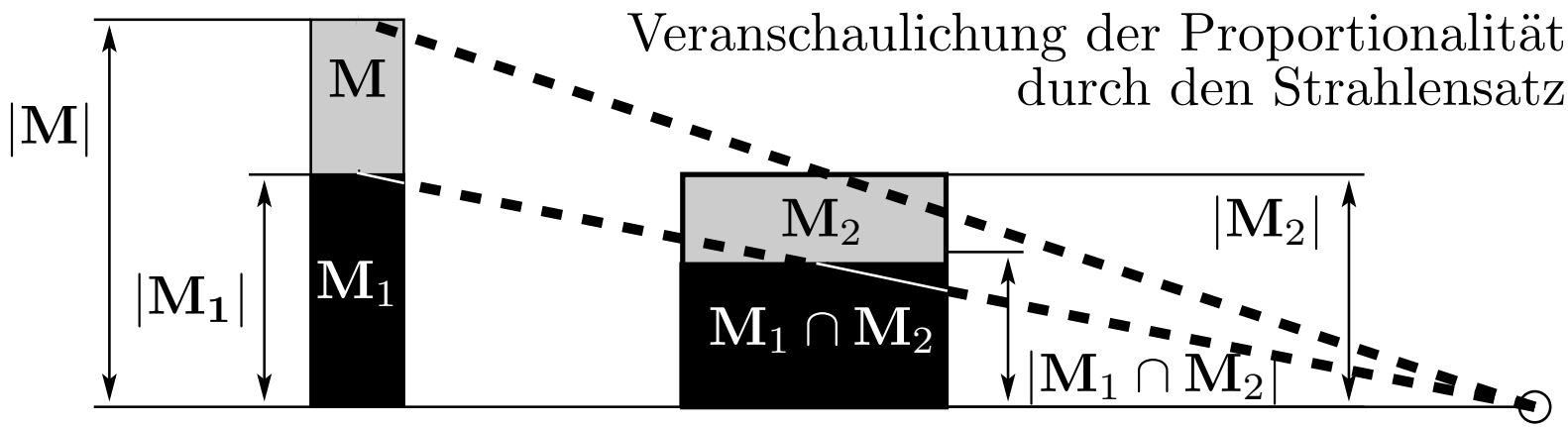
Capture-Recapture-Verfahren

Abgeleitet von einem Schätzer für die Größe von Tierpopulationen (z.B. von Vögeln in einem Gebiet) [2, 6, 5].

- Aus einer Menge $\mathbf{M}$ unbekannter Größe wird eine Menge $\mathbf{M}_1$ von Tieren eingefangen, gekennzeichnet und freigelassen.
- Nach Vermischung der Population Menge $\mathbf{M}_2$ von Tieren einfangen. Gekennzeichnete Tiere zählen.
- Bei tierunabhängiger Einfangwahrscheinlichkeit ist der Anteil der Tiere, die beim zweiten Einfangen gekennzeichnet sind...



(|...| – Größe der Mengen).



(beide Male eingefangen wurden) etwa gleich dem Anteil der gekennzeichneten Tiere:

$$\frac{|\mathbf{M}_1|}{|\mathbf{M}|} \approx \frac{|\mathbf{M}_1 \cap \mathbf{M}_2|}{|\mathbf{M}_2|}$$

($\mathbf{M}$ – Menge aller Tiere, $\mathbf{M}_1$, $\mathbf{M}_2$ – beim ersten bzw. zweiten mal eingefangene Tiere; $\mathbf{M}_1 \cap \mathbf{M}_2$ – Menge der beide Male eingefangenen Tiere). Geschätzte Größe der Tierpopulation:

$$|\mathbf{M}| \approx \frac{|\mathbf{M}_1| \cdot |\mathbf{M}_2|}{|\mathbf{M}_1 \cap \mathbf{M}_2|}$$

Fehler statt Tiere

Zwei Inspekteure i finden jeweils eine Menge von $\mathbf{M}_i$ Fehlern:

$$|\mathbf{M}| \approx \frac{|\mathbf{M}_1| \cdot |\mathbf{M}_2|}{|\mathbf{M}_1 \cap \mathbf{M}_2|}$$

($|\mathbf{M}_1 \cap \mathbf{M}_2|$ – Anzahl der von beiden Inspekteuren unabhängig voneinander gefundenen gleichen Fehler; $|\mathbf{M}|$ – geschätzte Anzahl der vorhandenen Fehler). Die geschätzte Fehlerüberdeckung ist das Verhältnis der Anzahl der insgesamt von beiden Inspekteuren gefundenen Fehler $|\mathbf{M}_1 \cup \mathbf{M}_2|$ zur geschätzten Geamtfehleranzahl $|\mathbf{M}|$:

$$IFC = \frac{\varphi_{\text{Erk}}}{\varphi} \approx \frac{|\mathbf{M}_1 \cup \mathbf{M}_2|}{|\mathbf{M}|} \approx \frac{|\mathbf{M}_1 \cap \mathbf{M}_2| \cdot |\mathbf{M}_1 \cup \mathbf{M}_2|}{|\mathbf{M}_1| \cdot |\mathbf{M}_2|}$$

Gebunden an die Annahmen:

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

Beispielaufgabe

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?

Lösung

1. Gesamtfehleranzahl:

$$\varphi = |\mathbf{M}| = \frac{|\mathbf{M}_1| \cdot |\mathbf{M}_2|}{|\mathbf{M}_1 \cap \mathbf{M}_2|}$$

2. Inspektionsfehlerüberdeckung:

$$IFC \approx \frac{|\mathbf{M}_1 \cap \mathbf{M}_2| \cdot |\mathbf{M}_1 \cup \mathbf{M}_2|}{|\mathbf{M}_1| \cdot |\mathbf{M}_2|} \quad (1)$$

Fehler	$\lvert\mathbf{M}_1\rvert$	$\lvert\mathbf{M}_2\rvert$	$\lvert\mathbf{M}_1 \cap \mathbf{M}_2\rvert$	$\varphi = \lvert\mathbf{M}\rvert$	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 (siehe später TV_F5). 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 numerischen Fehler. In Gl. 1 von vier Werten:

$$IFC \approx \frac{|\mathbf{M}_1 \cap \mathbf{M}_2| \cdot |\mathbf{M}_1 \cup \mathbf{M}_2|}{|\mathbf{M}_1| \cdot |\mathbf{M}_2|}$$

Größenordnung Zählwerte für vertrauenswürdige Schätzungen:

$$|\ldots| \gg 10^3$$

Systematische Fehler:

- ...

Systematische Fehler:

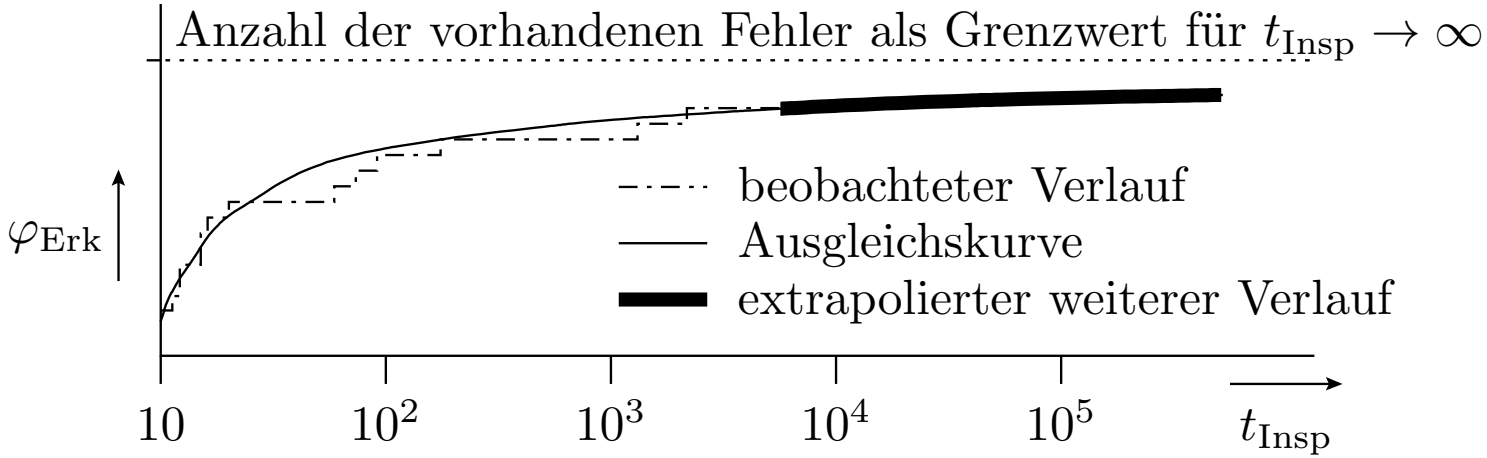
- Capture-Recaptur unterstellt für alle Fehler gleiche Erkennungswahrscheinlichkeit. Wenn gut erkennbare Fehler viel besser als schlecht erkennbare nachgewiesen werden, gelten die Schätzwerte nur für die guter kennbaren.
- Capture-Recaptur verbietet Informationsaustausch zwischen den Inspekteuren. Wenn doch, vergrößert das die Menge der gleichen gefundenen Fehler $\mathbf{M}_1 \cap \mathbf{M}_2$ gegenüber einer unabhängen Suche. Für $\mathbf{M}_1 = \mathbf{M}_2 \Rightarrow IFC = 1$.

Schätzwerte der Anzahl der nicht gefundenen Fehler und der Inspektionsfehlerüberdeckung nach Capture-Recapture sind mit erheblichen zufälligen und systematischen Schätzfehlern behaftet.

Inspektion als Zufallstest

Berücksichtigung, 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[2]:

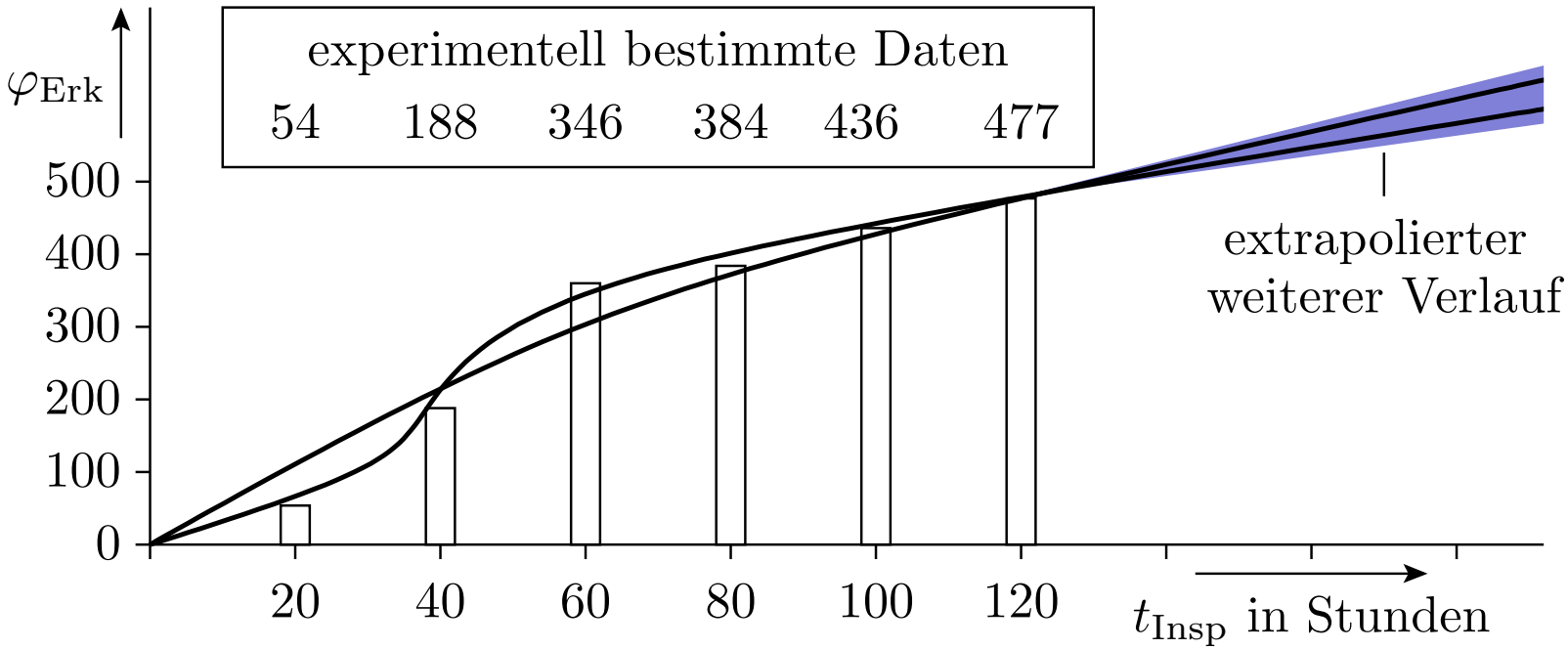


[2] Untersuchungen in dieser Richtung in der Literatur noch nicht gefunden.

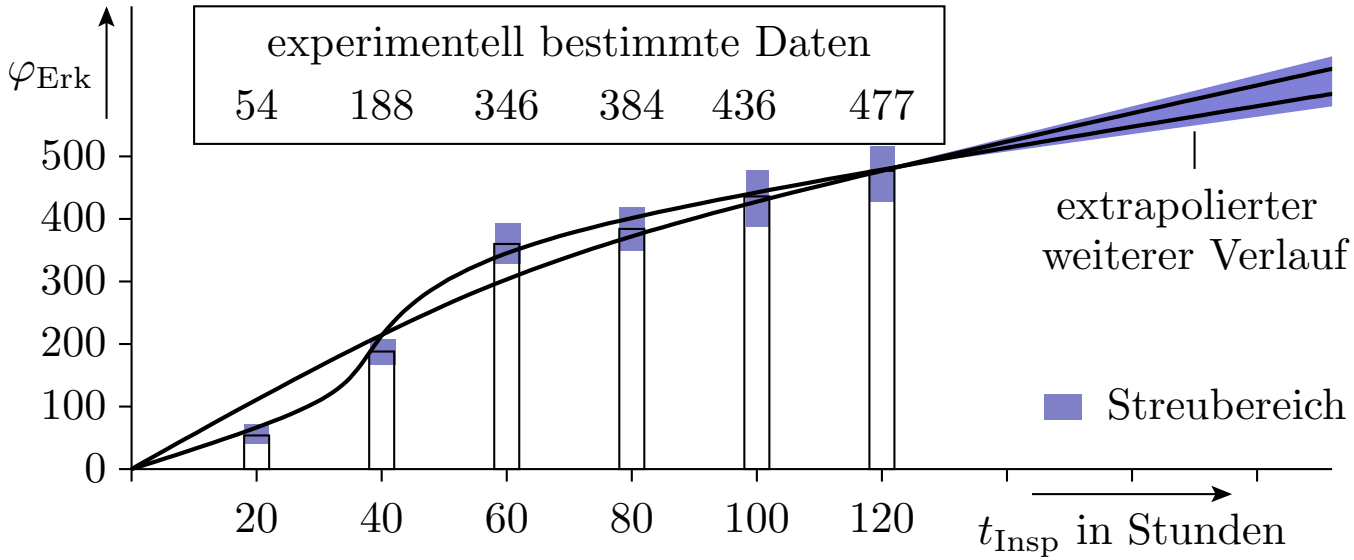
Experiment mit einem Inspekteur[3]

Inspektion des Buchmanuskripts [3] plus Beispielprogramme:

- Anzahl der gefunden Fehler in Abhängigkeit von der Inspektionsdauer.



[3] Bachelor-Arbeit von Yu Hong.



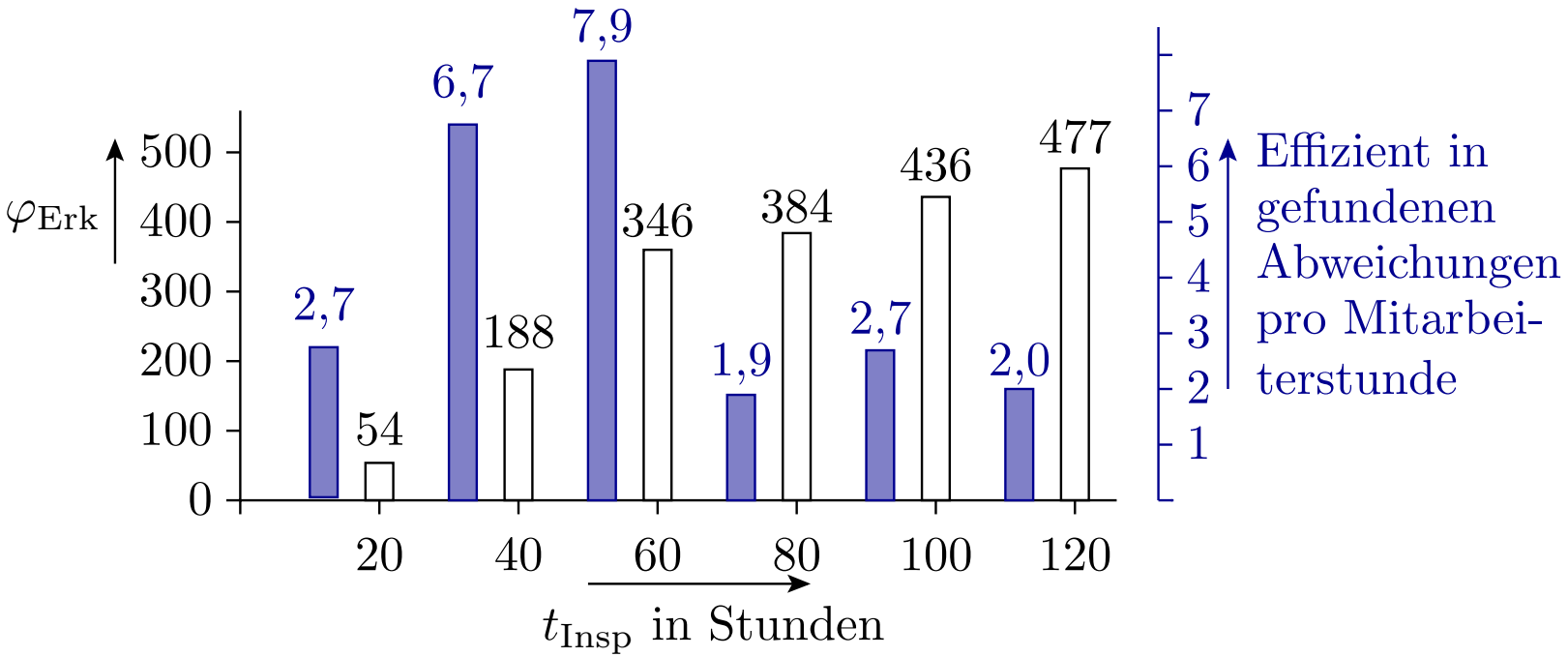
Unterschiedliche Approximationsmöglichkeiten für die weitere Abnahme der zu erwartenden Anzahl der nicht gefundenen Fehler, z.B. Abnahme der zu erwartenden Anzahl nicht gefundenen Fehler mit einem Exponenten $0 < k < 1$ (vergl. TV_F1):

$$E\left(\varphi_{\text{NErk}}\left(t_{\text{Insp}}\right)\right) = E\left(\varphi_{\text{NErk}}\left(t_0\right)\right) \cdot \left(\frac{t_{\text{Insp}}}{t_0}\right)^{-k}$$

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.

- 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, ...

Ein gute Inspektionstechnologie vermeidet die uneffizienten Einarbeitungs- und Ermüdungsphasen.

Inspektionstechniken

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

Einteilung der Inspektionstechniken

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

Syntax, Korrektheit

Syntax-, Typ-, WB- und Korrektheitstests

Kontrollverfahren aus der theoretischen Informatik:

1. Syntaxtest: Kontrolle, dass eine Zeichenfolge ein Wort einer Sprache ist (vergl. TV_F3, Abschn. 3.2).
2. Typ- und WB-Kontrollen. Wertekontrollen für Konstanten.
3. Kontrollen auf unsichere Programmkonstrukte.
4. Test der partiellen Korrektheit: Nachweis, dass jede Eingabe einer Berechnung, die eine Vorbedingung P erfüllt, auch eine Nachbedingung Q erfüllt, falls die Berechnung terminiert.
5. Test der totalen Korrektheit: zusätzlicher Beweis, dass die Berechnung terminiert.

(1) bis (3) sind die üblichen Kontrollen bei der Eingabe von Programmen und Entwurfsbeschreibungen in den Rechner. Die Fehlererkennungsmöglichkeiten hängen von der Programmiersprache, Zusatzregeln, ... ab.

VHDL[4] – Sprache mit strenger Typkontrolle

VHDL definiert keine Datentypen, sondern Beschreibungsmittel, um Datentypen zu definieren. Beispiel seien die Zahlentypen.

- Ein Zahlentyp ist ein zusammenhängender Zahlenbereich:

```
type <Zahlentyp> is range <Bereich>;
```

- Bereiche können auf- oder absteigend geordnet sein:

```
type tWochentag is range 1 to 7;
type tBitnummer is range 3 downto 0;
```

  $\text{Wochentag} \in \{1,2,3,4,5,6,7\}$; $\text{Bitnummer} \in \{3,2,1,0\}$

- ganzzahlige Bereichsgrenzen $\Rightarrow$ diskreter Zahlentyp
- Bereichsgrenzen mit Dezimalpunkt $\Rightarrow$ reeller Zahlentyp

```
type tWahrscheinlichkeit is range 0.0 to 1.0;
```

[4] Hardware-Beschreibungssprache mit für Kontrollen besonders gut geeignetem Typenkonzept.

Kontrollmöglichkeiten

- Kontrollen bei einer Variablenzuweisung

  Variablenname := *Ausdruck*;

  Typenübereinstimmung. Zulässiger Wert des Ausdrucks.

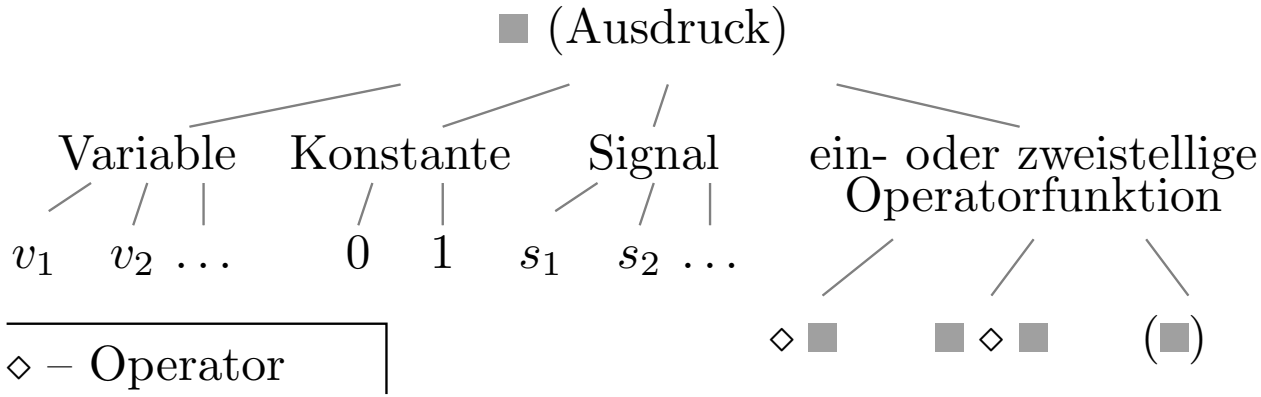
```
variable Wochentag: tWochentag;
variable Bitnummer: tBitnummer;
...
Wochentag := Bitnummer; -- Typ unzulaessig
Wochentag := 9;         -- Wert unzulaessig
```

- Kontrollen bei Signalzuweisungen

  Signalname <= W **[after** t_d**]{,** W **after** t_d**};**

  Gleicher Typ des Ausdruck W und des Signal. Ausdruck t_d muss Typ TIME haben; $t_\mathrm{d} \geq 0$.

Kontrollen in Ausdrücken: Ein Ausdruck ist eine rekursive Beschreibung einer Funktion mit ein- und zweistelligen Operatorfunktionen.



- Jeder Operator ist nur für bestimmte Operandentypen definiert.
- Jeder Kombination aus Operator und Operandentypen ist ein Ergebnistyp zugeordnet[5].
- Eine strenge Typenprüfung erkennt nicht nur, sondern vermeidet auch Fehler, indem sie den Programmierer zwingt, genauer über die beabsichtigte Zielfunktion nachzudenken.

[5] In VHDL sind die arithmetischen Operatoren (+, -, *, /) nur identische Operandentypen, z.B tWochentag, definiert.

Konzept der physikalischen Typen

Möglichkeit, für jede physikalische Größe einen ganzzahligen Typ mit Maßeinheit zu verarbeiten, z.B. für

$$I_\mathrm{D} = K \cdot \begin{cases} 0 & U_\mathrm{GS} < U_\mathrm{th} \\ \frac{(U_\mathrm{GS}-U_\mathrm{th})^2}{2} & \text{sonst wenn } (U_\mathrm{GS} - U_\mathrm{DS} < U_\mathrm{th}) \\ (U_\mathrm{GS} - U_\mathrm{th}) \cdot U_\mathrm{DS} - \frac{U_\mathrm{DS}^2}{2} & \text{sonst} \end{cases}$$

```
type tSpg   is -1E9 to 1E9; -- Spannung in 10nV
type tSteil is -1E9 to 1E9; -- Steilheit in 10nA/V^2
type tStrom is -1E9 to 1E9; -- Strom in 10nA

variable UGS, UDS, UTH: tSpg;
variable k: tSteil;
variable ID: tStrom;
```

Wertebereiche: Spannung: $\mp 10\,\mathrm{V}$, Steilheit: $\mp 10\frac{\mathrm{A}}{\mathrm{V}^2}$, Strom: $\mp 10\,\mathrm{A}$.
Erlaubt Zusatzkontrollen, dass keine fremden physikalischen Größen addiert werden, Wertebereiche überschritten werden, ...

$$I_\mathrm{D} = K \cdot \begin{cases} 0 & U_\mathrm{GS} < U_\mathrm{th} \\ \frac{(U_\mathrm{GS}-U_\mathrm{th})^2}{2} & \text{sonst wenn } (U_\mathrm{GS} - U_\mathrm{DS} < U_\mathrm{th}) \\ (U_\mathrm{GS} - U_\mathrm{th}) \cdot U_\mathrm{DS} - \frac{U_\mathrm{DS}^2}{2} & \text{sonst} \end{cases}$$

In der nachfolgenden Implementierung

```
A1: if UGS<UTH then ID:=0;
A2: elsif UGS-UDS<UTH then
A3:   ID:=k/2*((UGS-UTH)**2/1E8)/1E8;
A4: else
A5:   ID:=k*((UGS-UTH)*UDS -(UDS**2)/2E8)/1E8;
A6: end if;
```

erkennt der Compiler / könnte der Compiler erkennen:

- fehlen die Typkonvertierungen für die Multiplikation »tSteil« mit »tSpg« und vor der Zuweisung an »tStrom«.
- Wertebereichsüberlauf bei Produkten größer 10^9. Nach den Divisionen durch 10^8 max. eine gültige Dezimalstelle. ...

$$I_\mathrm{D} = K \cdot \begin{cases} 0 & U_\mathrm{GS} < U_\mathrm{th} \\ \frac{(U_\mathrm{GS}-U_\mathrm{th})^2}{2} & \text{sonst wenn } (U_\mathrm{GS} - U_\mathrm{DS} < U_\mathrm{th}) \\ (U_\mathrm{GS} - U_\mathrm{th}) \cdot U_\mathrm{DS} - \frac{U_\mathrm{DS}^2}{2} & \text{sonst} \end{cases}$$

Mit Typcasts und ausreichend großen WBs für Zwischenprodukte:

```
type tProd is -2E18 to 2E18;
variable tmp: tProd;
...
if UDS<UTH then ID:=0;
elsif UGS-UDS<UTH then
 tmp := ((tProd(UGS)-tProd(UTH))**2)/2E8;
else
 tmp := ((tProd(UGS)-tProd(UTH))*tProd(UDS)/2)/1E8;
 tmp := tmp - ((tProd(UDS))**2)/2E8;
end if;
ID := tStrom(tmp * tProd(k))/1E8);
```

Probleme umgehen, statt lösen

Physikalischen Typen haben sich nicht durchgesetzt. Statt dessen

- Gleitkommazahlen
  - bestehen aus Vorzeichen, Exponent und Mantisse,
  - können Werte über viele Zehnerpotenzen und die Sonderwerte $\pm\infty$ und NaN (not a Number) darstellen.
  - Vermeidung WB-Verletzungen und zu großer Rundungsfehler.
- Programmgeneratoren zur Konvertierung aus Gleitkommaberechnungen in Festkommarechnung:
  - Automatische Einfügung erforderlicher WB-Anpassungen.
  - Vermeidet gleichfalls WB-Verl., zu große Rundungsfehler.

  Beispielanwendung: Entwicklung und Test von Steuer- und Regelungsalgorithmen für Motorsteuergeräte mit unter Matlab/Simulink mit anschließender Code-Generierung ...

Statische Code-Analyse

Statische Code-Analyse

Untersuchung des Quellcodes auf Problemquellen:

- fehlerbegünstigend, Verständnis erschwerend,
- Uneindeutigkeiten,
- Programmkonstrukte, die Speicherlecks, Deadlocks, ... begünstigen,
- potentielle Sicherheitsrisiken,
- falsche Benutzung von Betriebssystemschnittstellen, ...

MISRA

Regeln für C-Programme:

- Bezeichnerlänge max. 31 Zeichen (längere Bezeichner werden von manchen Compilern nach 31 Zeichen abgeschnitten, Risiko, dass Compiler unterschiedliche Variablen zu einer zusammenfasst.
- Unterschiedliche Bezeichner für unterschiedliche Objekte:

```
int16_t i; {
 int16_t i; // Hier zwei Variablen i definiert.
            // Nach MISRA-Standard unzulässig.
 i = 3;     // Denn, welche ist hier gemeint?
}
```

- Jeder Variablen ist vor ihrer Nutzung ein Wert zuzuweisen, ...

Insgesamt über 100 Regeln, zum Teil verpflichtend, zum Teil Empfehlungen.

API-Benutzungsregeln

Beispielregeln für die Benutzung der Windows-API aus [1]:

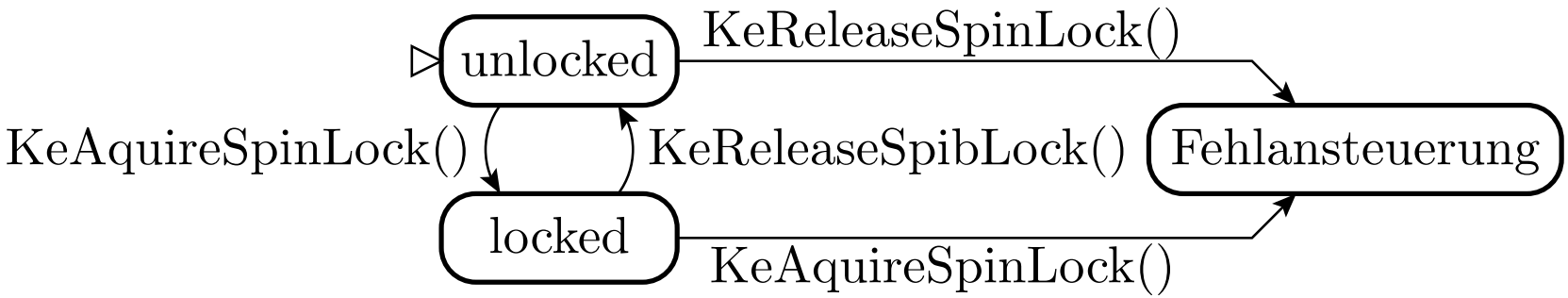
spinlock Spinlocks müssen alternierend reserviert und freigegeben werden.

spinlocksafe Vermeidung von Deadlocks mit Spinlocks.

criticalregions Problemvermeidung im Zusammenhang mit der Nutzung kritischer Regionen.

...

Kontrollautomat für Regel »spinlock«:



Eine zu testende Treiberfunktion

Eine Treiberfunktion ruft »KeAquire..« und »KeRelease...« u.U. mehrfach auf, in Fall-unterscheidungen, Schleifen, ... Für jeden Kontrollpfad muss der Spinlock alternierend bedient werden.

Fehlerausschluss erfordert Kontrolle für alle Pfade.

Reale Treiberfunktionen haben hunderte von Code-Zeilen. Kontrolle selbst so einfacher Regeln nicht trivial.

```
void example() {
 do {
   KeAcquireSpinLock();
   nPacketsOld = nPackets;
   req = devExt->WLHV;
   if(req && req->status){
    devExt->WLHV = req->Next;
    KeReleaseSpinLock();
    irp = req->irp;
    if(req->status > 0){
     irp->IoS.Status = SUCCESS;
     irp->IoS.Info = req->Status;
    } else {
     irp->IoS.Status = FAIL;
     irp->IoS.Info = req->Status;
    }
    SmartDevFreeBlock(req);
    IoCompleteRequest(irp);
    nPackets++;
   }
 } while(nPackets!=nPacketsOld);
 KeReleaseSpinLock();
}
```

Sicherheitslücken

Die bekannteste Funktion, die Sicherheitslücken in C-Programmen verursacht, ist

```
char * strcpy(char *dest, char *src);
```

für Eingabezeichenketten ohne Längenkontrolle. Zu lange Zeichenketten überschreiben nachfolgende Variablen ...

Problemvermeidung durch statische Code-Analyse:

- Suche alle Aufrufe von strcpy (die Eingabedaten in Puffer kopieren).
- Ersatz durch

  ```
  char * strncpy(char *dest, char *src, int n);
  ```

  mit der zusätzlichen Übergabe der Puffergröße n.

Wie lässt sich durch Überschreiben von Daten hinter einem Zeichenkettenpuffer der Programmablauf manipulieren?

Baugruppen

Statische Tests für Baugruppen und CPS

Besonderheiten beim Test im Vergleich zu Software:

- Kontrolle physikalischer Größen.
- Beim Test defekter Systeme ist Schaden am System oder der Umwelt zu vermeiden.

Im weiteren behandelte statische Tests:

- MDA (Manufacturing Defect Analysis),
- Optische Inspektion,
- ICT (In Circuit Test) und
- dessen Nachbildung mit Boundary-Scan.

Inbetriebnahme

Inbetriebnahme elektronischer Baugruppen

Zur Vermeidung der Zerstörung von Bauteilen gelten bei uns im Labor und in den Praktika folgende Inbetriebnahmeregeln:

1. Sichtkontrolle im spannungsfreien Zustand.
2. Elektrische Verbindungskontrolle mit einem Durchgangsprüfer, Multimeter oder Tester ohne Betriebsspannung.
3. Rauchtest: Test mit Strombegrenzung und ständiger Kontrolle auf Erwärmung und Rauchentwicklung.
4. Funktionstests mit einem Labornetzteil mit eingestellter Strombegrenzung.

Während der Änderung / Fehlerbeseitigung an Schaltungen ist immer die Versorgungsspannung auszuschalten.

MDA

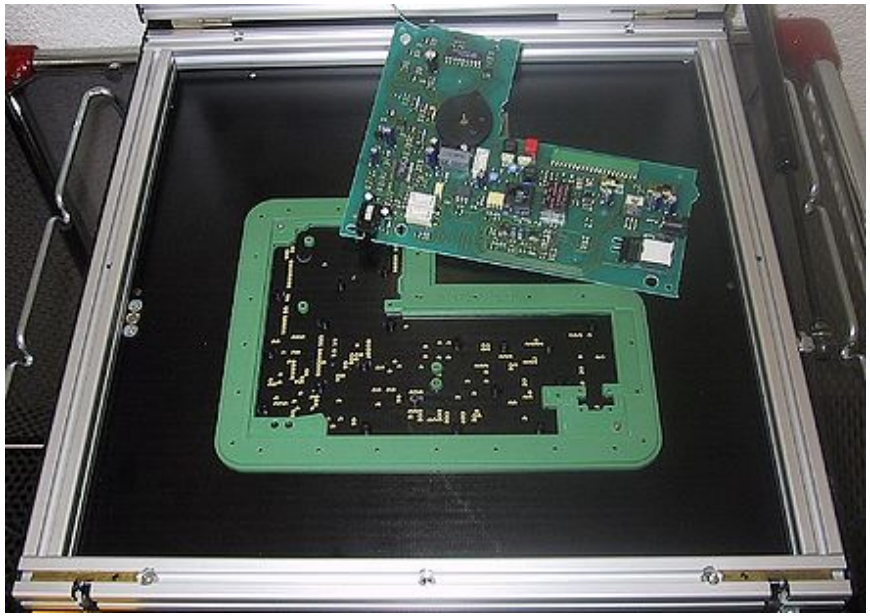
MDA (Manufacturing Defect Analyzer)

Prüfsysteme, die im spannungsfreien Zustand nach:

- Unterbrechungen,
- Kurzschlüssen und
- Fehlbestückungen

suchen.

Nadelbettadapter



In der Serienfertigung erfolgt die Kontaktierung für den Test auf Kurzschlüsse, Unterbrechungen und Bestückungsfehler mit einem mit Unterdruck angesaugten Nadeladapter.
Über die Nadeln werden Prüfgeräte angeschlossen.

Weitere Unterteilung der Prüfverfahren:

- MDA (Manufacturing Defect Analyzer),
- ICT (In-Circuit Test).

Weiterentwicklung im Zuge der zunehmenden Miniaturisierung:

- Boundary-Scan: »Silicon Nails« statt Nadeln.
- Selbsttest (BIST): Prüftechnikeinbau in die Schaltung.

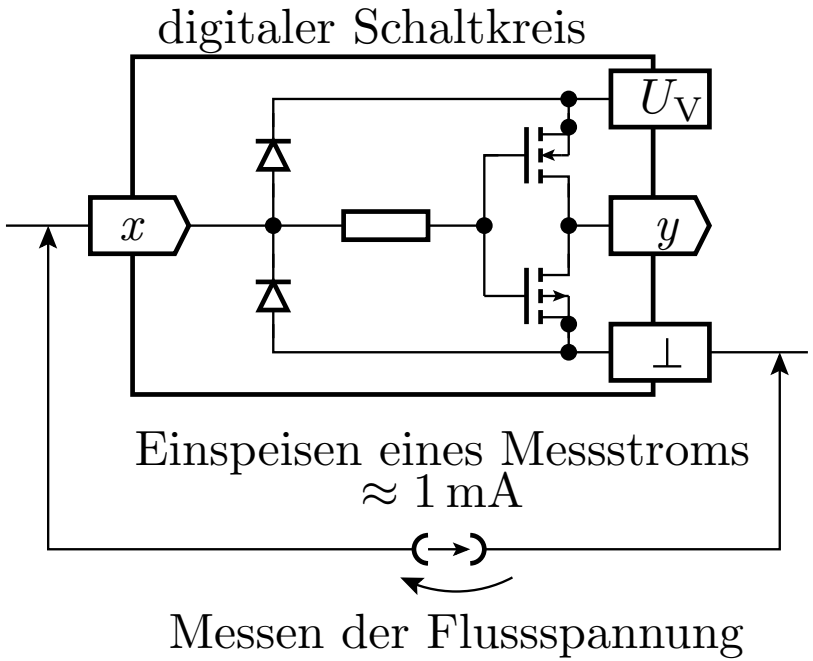
MDA

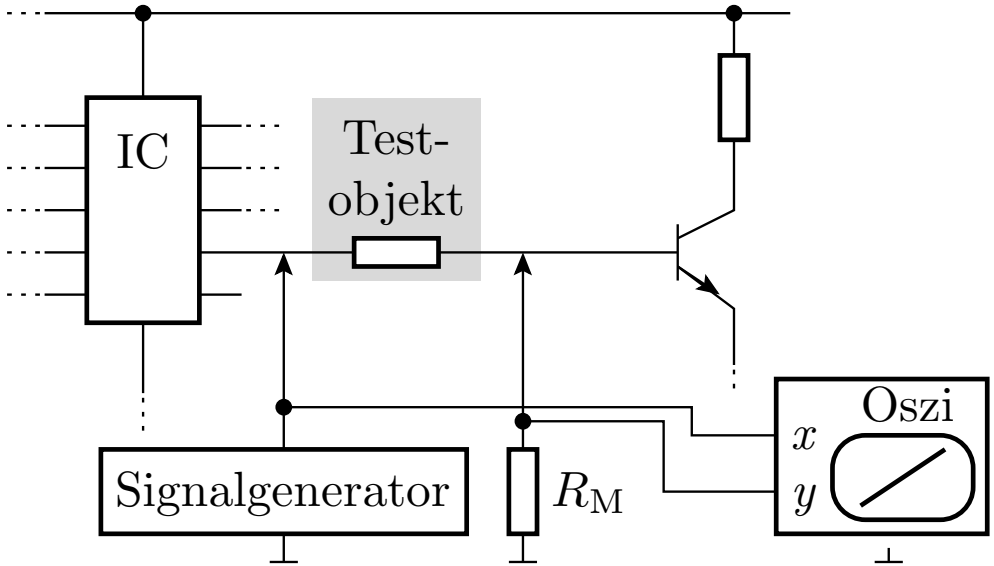
Suche potentieller Bestückungs- und Verdrahtungsfehler mit elektrischen Zweipunktmessungen:

- Stromeinspeisung und Messung der Spannung oder
- Spannungseinspeisung und Strommessung.

Bauteiltypische Strom-Spannungsbeziehungen für Sinuseingabe:

- Widerstand: Gerade,
- Kondensator: Elypse,
- Diode: Kennlinie mit Knick,
- Schaltkreise: Ausmessen der Schutzdioden,
- Unterbrechung: kein Strom,
- Kurzschluss: kein Spannungsabfall,
- ...

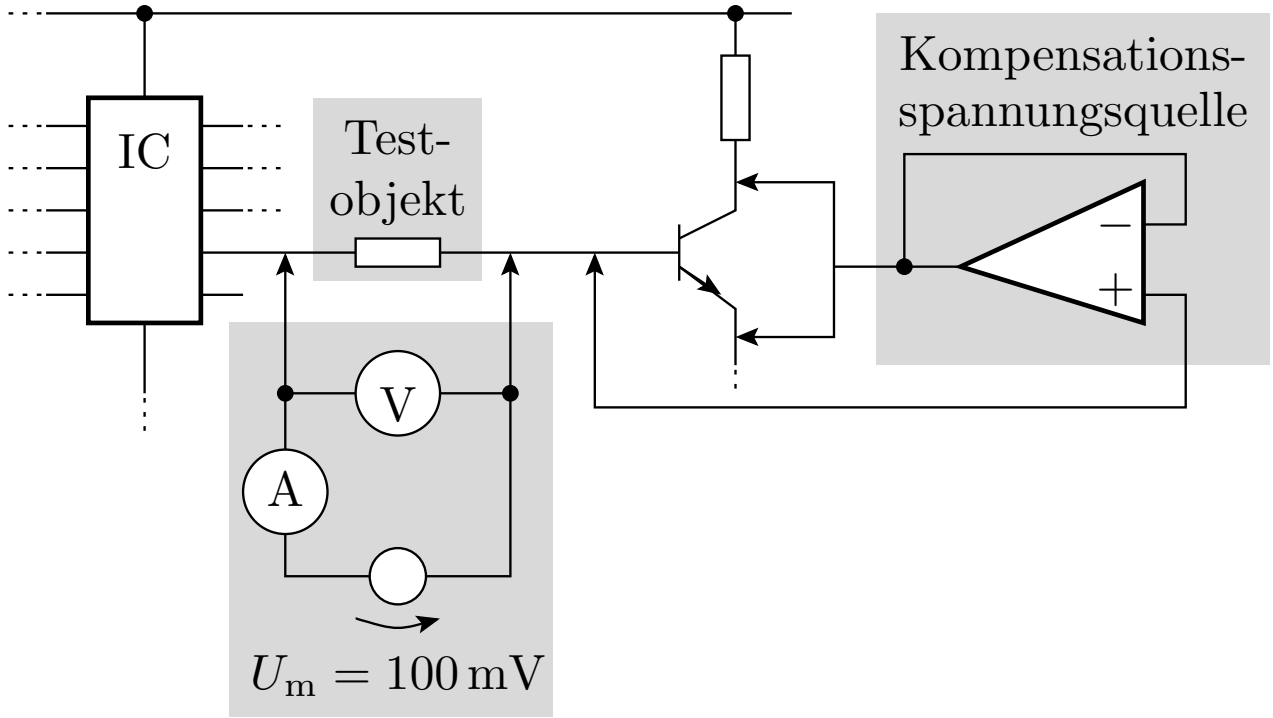




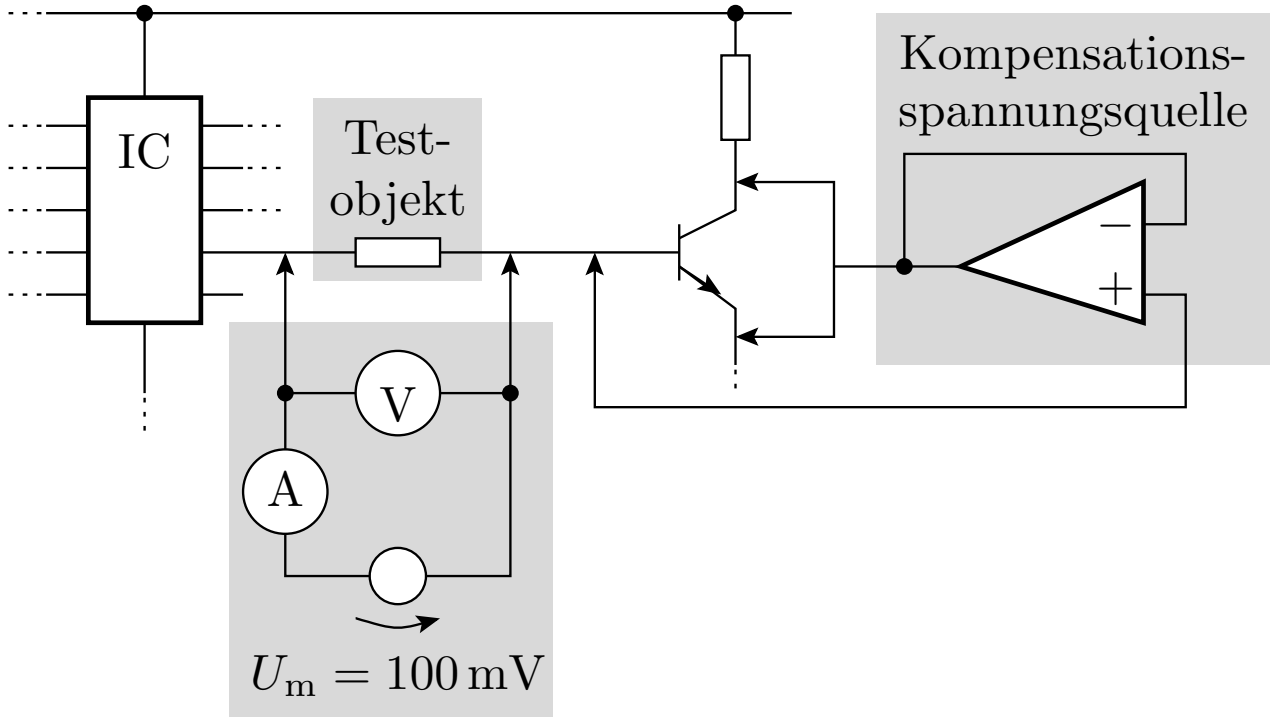
- Die Strom-Spannungs-Beziehung zwischen zwei Punkten in einer Schaltung hängt von der kompletten Schaltung, nicht nur einem einzelnen Bauteil ab.
- Bestimmbar durch Ausprobieren an einem »Golden Device«.
- Problematisch können sein
  - die Toleranzbereiche der Sollwerte unter Berücksichtigung der Bauteilstreuungen,
  - die Erkennungssicherheit für Fehlbestückungen, z.B. bei sehr kleinen Kapazitäten.

In-Circuit-Test

Analoger In-Circuit Test

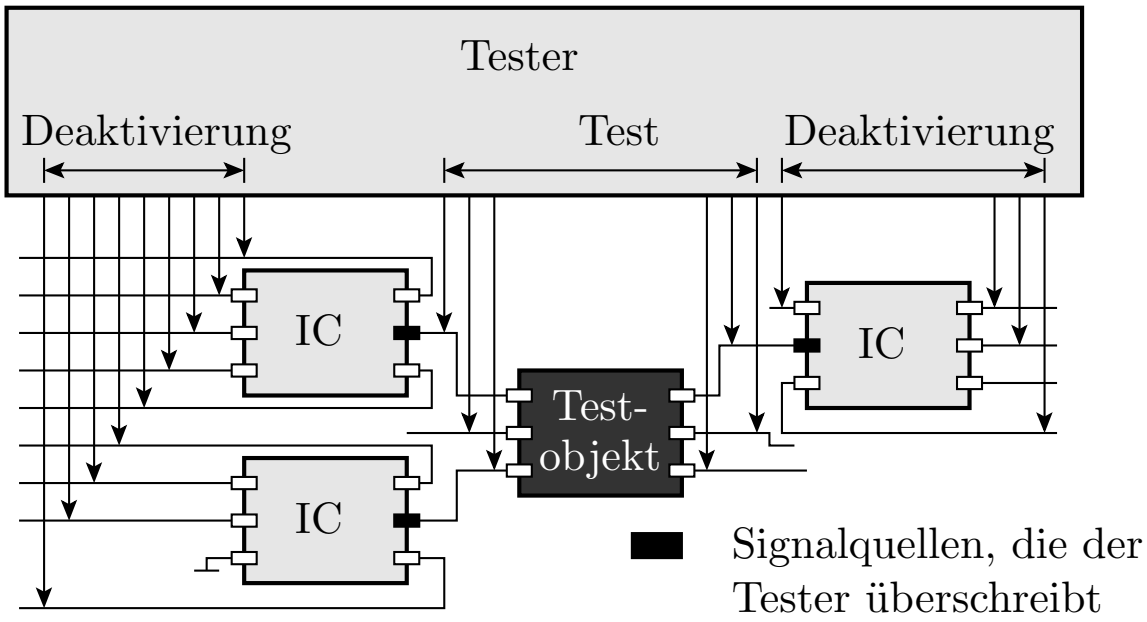


Unterdrückung von Parallelströmen zum Testobjekt durch Kompensation der Spannungsabfälle über den wegführenden Bauteilen auf einer Testobjektseite auf null. Erlaubt einen isolierten Zweipoltest.



- Vereinfacht die Testauswahl, Sollwertfestlegung, ...
- Mindert die Häufigkeit von Fehlklassifikationen.
- Für digitale Schaltkreise wenig geeignet.

Digitaler In-Circuit-Test



- 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).

Voraussetzung für ICT-Einsatz »prüfgerechter Entwurf«:

- Bei Vakuumansaugen luftdichter Rand, keine Löcher.
- Geeignete Kontaktflächen.
- Deaktivierungsmöglichkeit der Schaltkreise, ...

Automatische Generierung der Testvorschrift möglich:

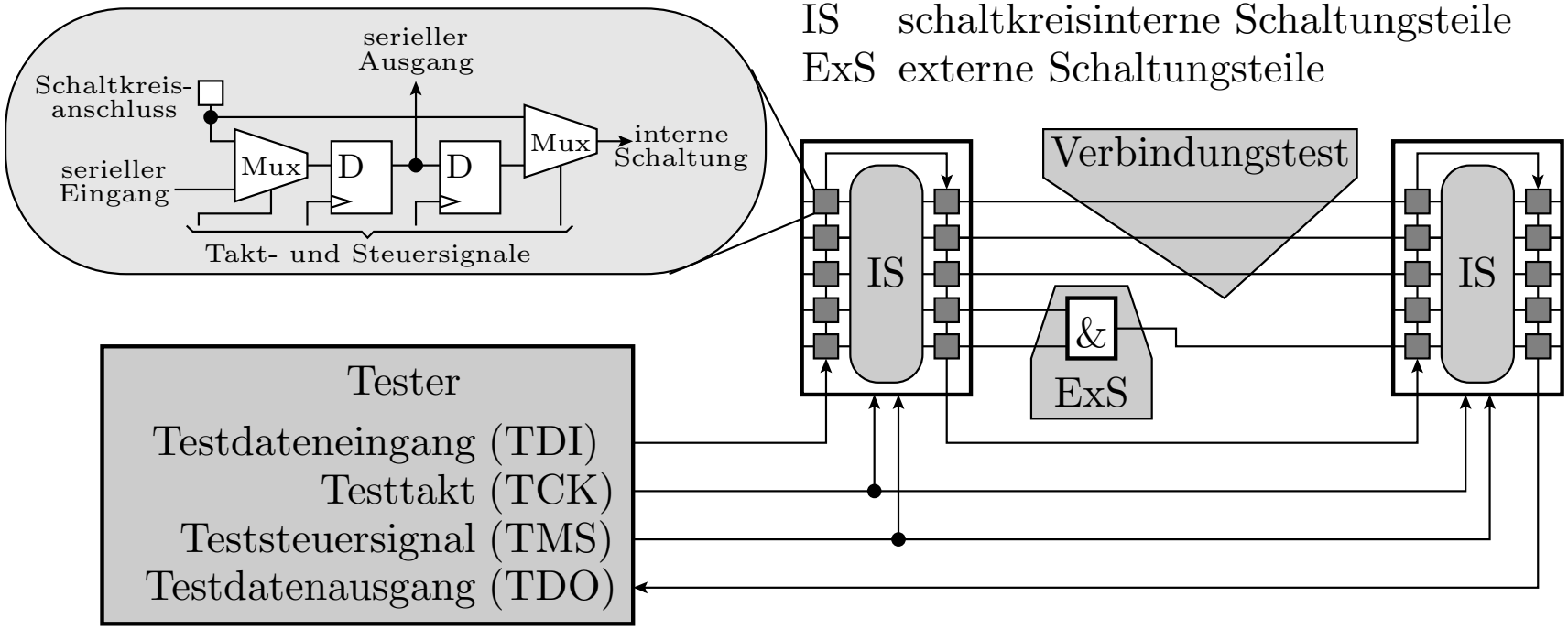
- Zusammensetzen aus Test- und Deaktivierungsvorschriften für alle Bauteile (gut gestelltes Problem).

Fehlererkennung und Lokalisierung:

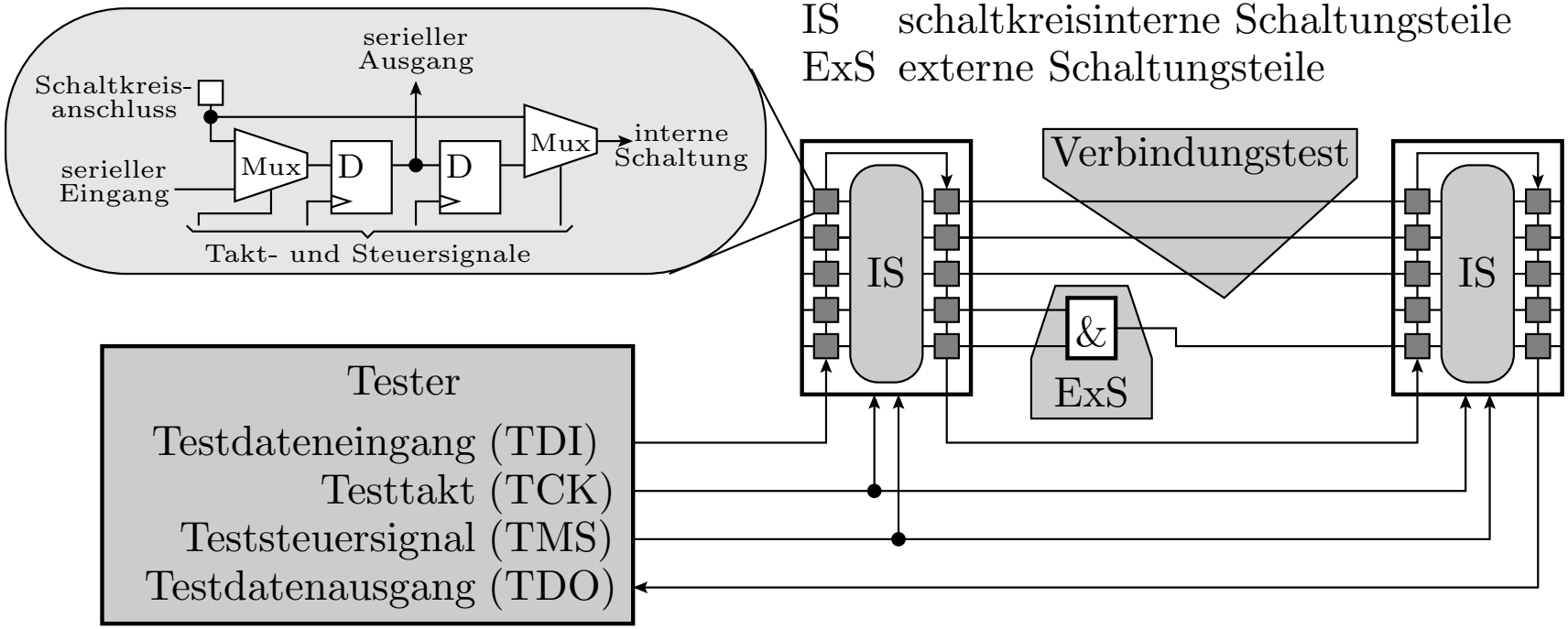
- Erkennt fast alle Kurzschlüsse, Unterbrechungen und Fehlbestückungen und gibt den genauen Fehlerort an.

Boundary-Scan

Boundary-Scan



- Ersatz der mechanischen Nadeln durch »silicon nails« (seriell beschreibbare Register an den Schaltkreisanschlüssen, im Normalbetrieb überbrückt).
- Verbindungen, Innenschaltung und »ExS's« separat testbar.



Ablauf eines Testschritts für den Baugruppentest:

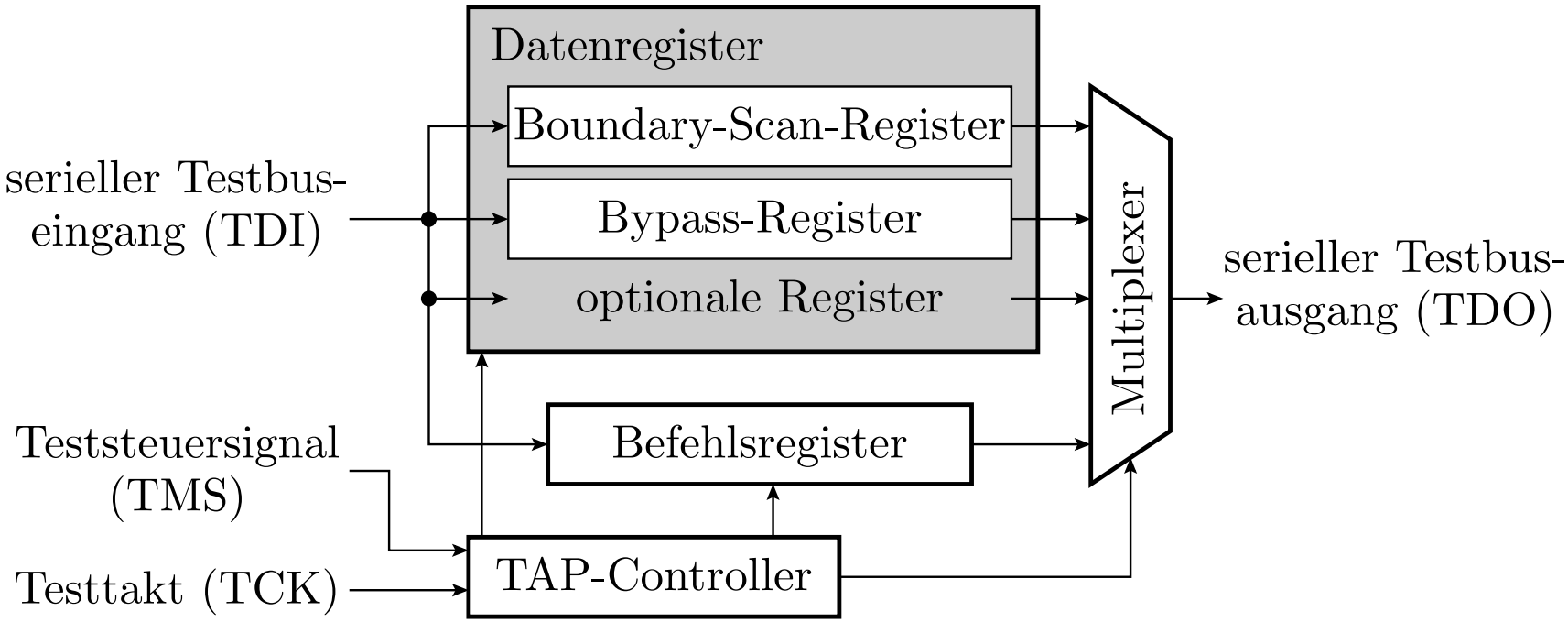
- BS-Register aller Schaltkreise auf der Baugruppe seriell beschreiben,
- einen Arbeitsschritt ausführen,
- die im Arbeitsschritt in den BS übernommenen Werte seriell an den Tester ausgeben und zeitgleich nächsten Eingabevektor laden.

- Ursprungsidee: Alternative zu den teuren, für jede Baugruppe speziell anzufertigenden Nadeladaptern.
- Als IEEE 1149.1 standardisierter serieller Testbus.

Nutzbar für weitere Test-, Diagnose- und Rekonfigurationsaufgaben, z.B. in den Laborübungen:

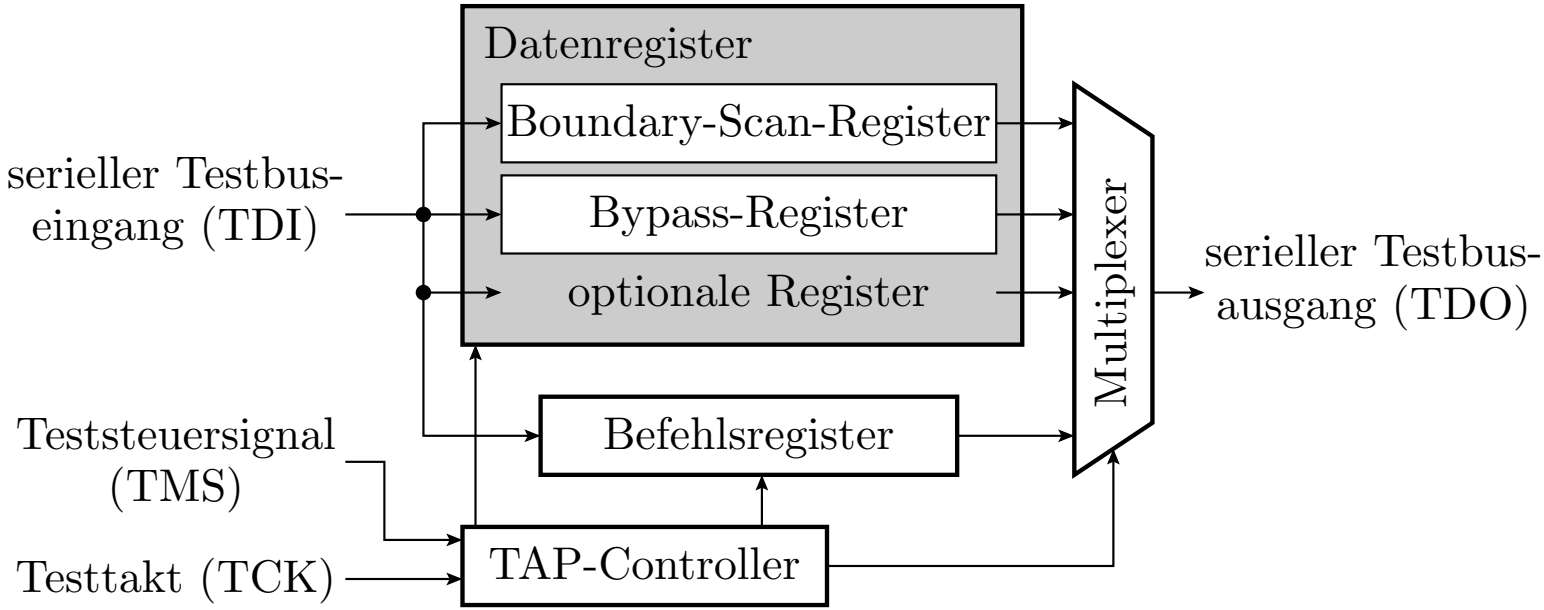
- Programmierung der Mikroprozessoren und FPGAs.
- Steuerung der In-Circuit-Debugger unter der AVR- und Microblaze-Entwicklungsumgebung.
- Steuerung des integrierten Logikanalysators Chip-Scope.

Die Testbusarchitektur der Schaltkreise



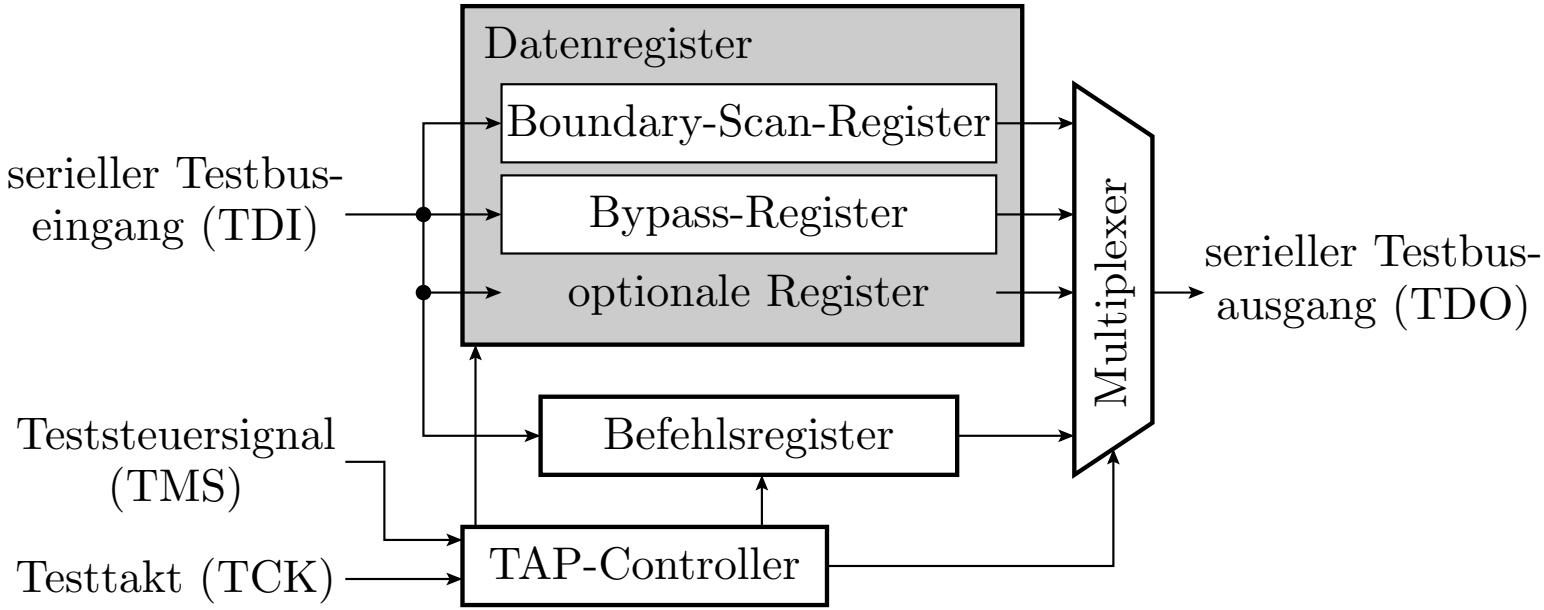
Eine Boundary-Scan-Implementierung umfasst:

- den TAP- (Test Access Port) Controller
- ein Befehlsregister
- mehrere Testdatenregister (mindestens das Boundary-Scan- und das Bypass-Register).



Über TMS und TAP-Controller steuerbare Funktionen:

- Capture: Übernahme von Daten aus der Schaltung in das Befehlsregister.
- Shift: Werte im Befehlsregister eine Position weiter schieben.
- Update: eingeschobenes Bitmuster in das Befehlsregister übernehmen.
- Dieselben drei Funktion für ein über das Befehlswort ausgewähltes Datenregister, ...



Datenregister:

- Boundary-Scan: Register am Schaltkreisrand
- Bypass: 1-Bit-Register zur Überbrückung des Schaltkreises in der Schieberegisterkette der Baugruppe

Optionale Erweiterungen:

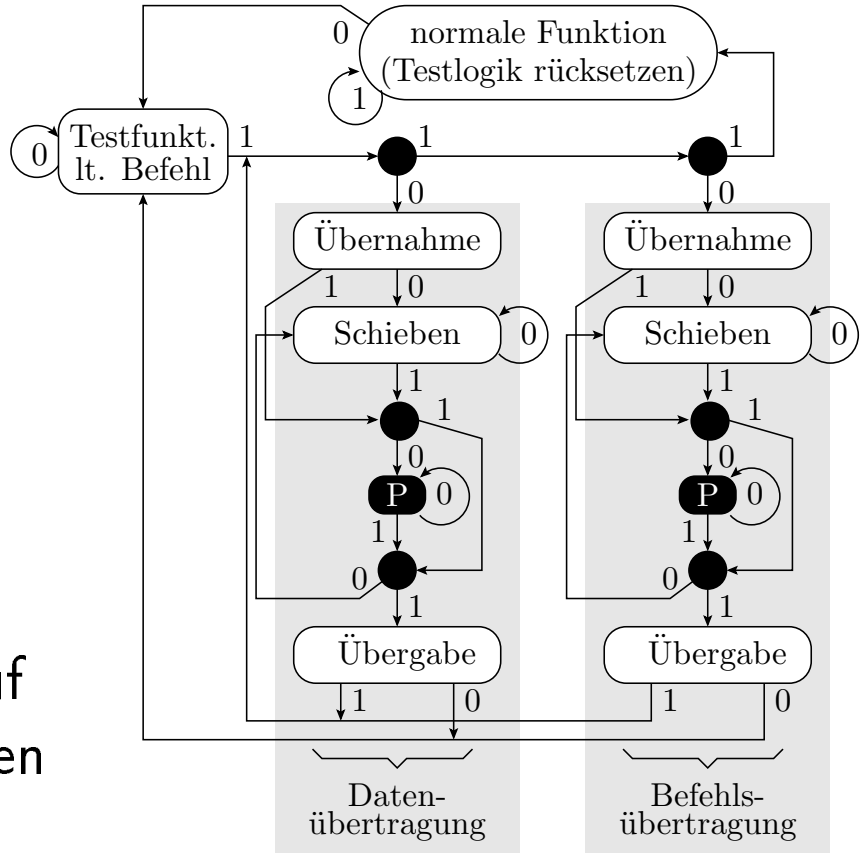
- Hersteller- und Bauteilidentifikationsregister
- weitere Test-, Programmier- oder Debug-Register

TAP-Controller, Busprotokoll

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

Typischer Testablauf:

- Befehlsregister lesen (Bestückungskontrolle[6]),
- Bauteilnummern lesen (Bestückungskontrolle),
- Einen Teil der Schaltkreise auf Bypass setzen. Für die anderen Datenregister auswählen.
- Verbindungstest. ...

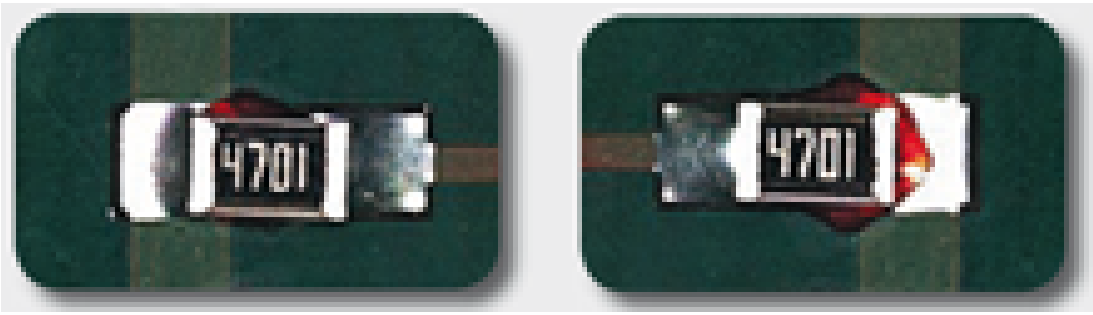


[6] Das Befehlsregister übergibt beim Lesen ein Muster zur Erkennung von Unterbrechungen der Schieberegisterkette auf der Baugruppe.

Optische Inspektion

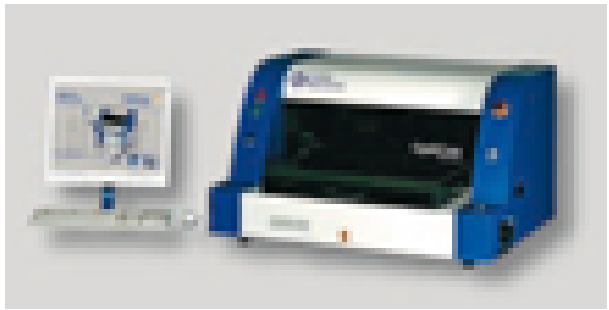
Optische Inspektion

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 (siehe Produkthaftung, TV_F2, Abschn. 4.4).

Automatisierte Baugruppeninspektion



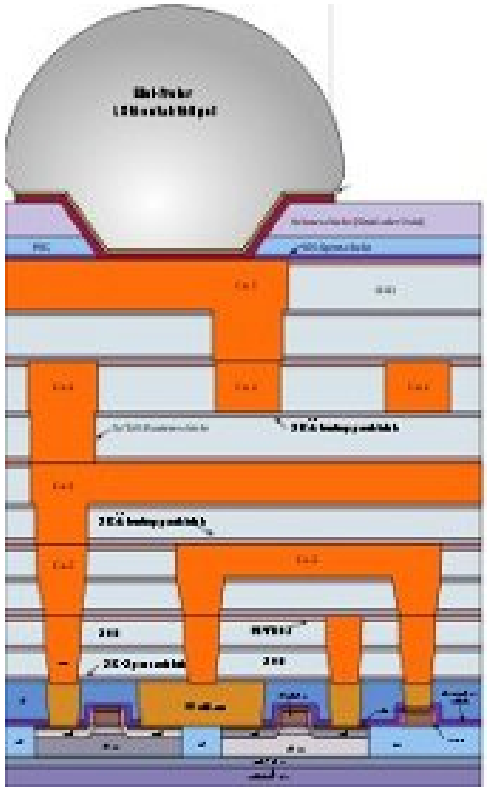
- Bildverarbeitungssystem mit Beleuchtung, Kamera, Verarbeitung, Monitor.
- Lernen von Bildern mit Fehlern und korrekten Bauteilen.
- Generierung des Prüfprogramms aus einer geometrischen Beschreibung und einer Bilddatenbank.
- Pflicht für sicherheitskritische Baugruppen (Automotive).

Statische Tests für Schaltkreise

Fertigungsfehler integrierter Schaltkreise

Schaltkreise werden schichtenweise hergestellt:

- Auftragen von Schichten (z.B. Fotolack oder Metall),
- Belichten des Fotolacks durch eine Maske, die die Geometrie der zu erzeugenden Schichtelemente festlegt,
- Entfernen der belichteten (unbelichteten) Bereiche des Fotolacks,
- Fortätzen der freiliegenden Schichten neben dem Fotolack und entfernen des Fotolacks.



Typische Herstellungsfehler:

- fehlendes (zu wenig aufgetragenes zu viel weggeätztes) und
- überflüsiges (zu viel aufgetragenes, zu wenige weggeätztes)

Leitungs- oder Isolationsmaterial in einer Schicht.

Schaltkreisfehler und statische Tests

Einteilung der Schaltkreisfehler:

- Globale Fehler: Funktion großflächig beeinträchtigt. Prozesssteuerfehler, ...
- Lokale Fehler: defekte Einzelstrukturen.

Statische Schaltkreistests zielen vorrangig auf globale Fehler.

- Messen der Stomaufnahme (Versorgungsleitungen, Datenanschlüsse).
- Ausmessen spezieller hierzu gefertigter Teststrukturen.
- Stichprobenweises Messen von Schichteigenschaften (Dicke, Widerstand, ...).

Statische Tests erfolgen zum Teil zwischen den einzelnen Fertigungsschritten. Die herstellungsbegleitenden Tests erlauben:

- frühzeitiges Aussortieren fehlerhafter Zwischenprodukte,
- Erkennung / Lokalisierung Prozessfehler; Fehlervermeidung.

Literatur

4. Literatur

[1] T. Ball.
Thorough static analysis of device drivers.
In *EuroSys*, pages 73–85, 2006.

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

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

[4] Peter Liggesmeyer.
Software-Qualit"at: Testen, Analysieren und Verifizieren von Software.
Spectrum, 2002.

[5] 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.

[6] 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.