Test und Verlässlichkeit Foliensatz 5: Hardware-Test und Selbsttest.

Prof. G. Kemnitz

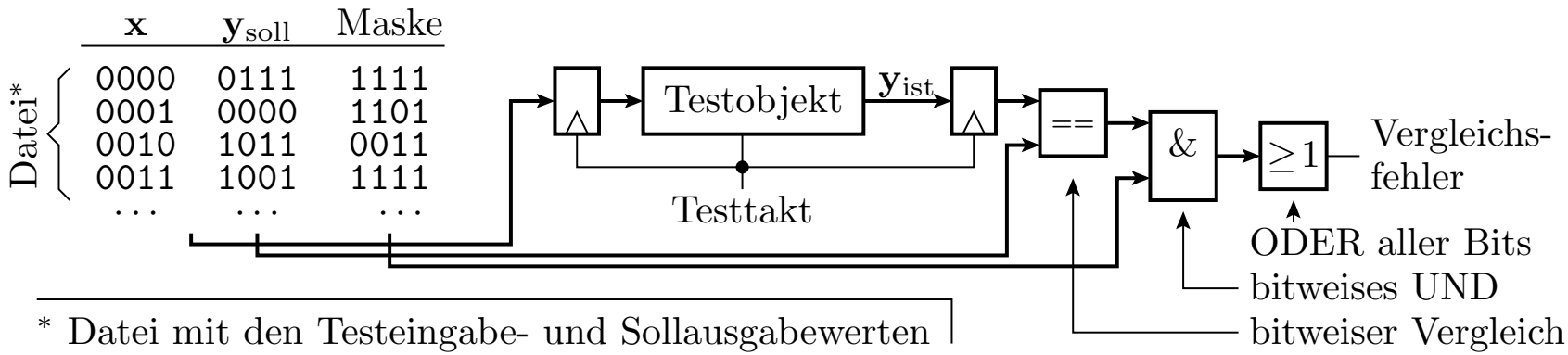
Institut für Informatik, TU Clausthal (TV_F5)

10. April 2022

Inhalt TV_F5: Hardware-Test und Selbsttest

Fehlermodellierung

Test digitaler Bausteine



Wiederhole für jeden Taktschritt des Tests:

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

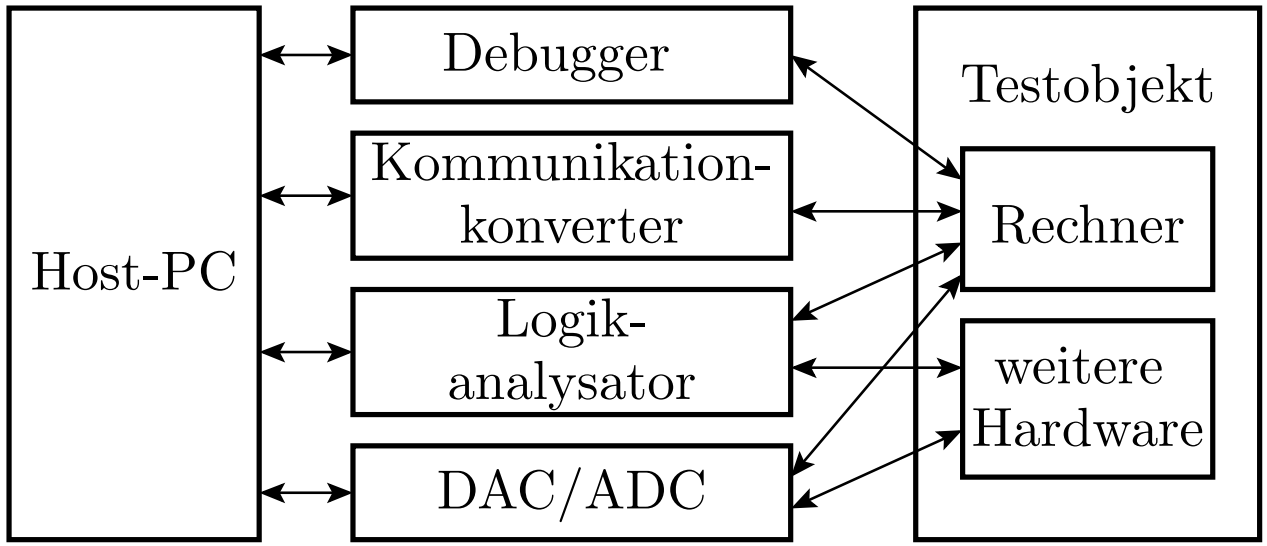
Auswertung vorzugsweise Vergleich mit Sollwerten unter Ausmaskierung von Ausgabebits ohne definierten Sollwert.

Testgüte: Fehlerüberdeckung, Anteil der nachweisbaren Fehler.

Initialisierung interner Speicherzellen zum Testbeginn.

Zur Verbesserung der Fehlerüberdeckung zusätzliche Steuerung und Beobachtung interner Speicherzustände.

Test von Rechnersystemen



Für Rechnersystem bieten sich software-gestützte (Selbst-) Tests an.
Dafür zusätzlich benötigte Funktionalität:

- zu Funktionseinheiten für die Bereitstellung der Testeingaben und Testausgaben (physikalisch, elektrisch, digital).
- Möglichkeiten zum Laden und Starten von Testprogrammen.
- Einen Debugger für Schrittbetrieb, Setzen von Haltepunkten, Trace-Aufzeichnung Lesen und Schreiben von Variablen und Speicherinhalten.

Güte der Tests

Bei einem Soll/Ist-Vergleich der Testausgaben mit korrekten Sollwerten sind Maskierungen bei der Kontrolle und Phantomfehler vernachlässigbar. Die Fehlerüberdeckung als Gütemaß hängt nur von der Anzahl und Auswahl der Testeingaben ab. Abschätzung mit Modellfehlern.

Fehlermodell: Algorithmus, der aus einer simulier- oder abarbeitbaren Beschreibung eine Modellfehlermenge berechnet.

Modellfehler: geringfügige Verhaltes- oder Beschreibungsänderung.

Haftfehler: Annahme von ständig eins oder ständig null für einen Gatteranschluss.

Geziele Suche: Suche von Testeingaben, für die der Modellfehler die Ausgabe verfälscht.

Zufallstest: Auswahl unabhängig von den Modellfehlern und den zu findenden Fehlern.

Test mit allen Eingabemöglichkeiten

Für den Nachweis, dass ein Service für alle Eingabedaten korrekte Ergebnisse liefert, müsste er mindestens mit allen Eingaben ausprobiert werden. Bereits ab wenigen Eingabebits unmöglich:

	m	2^m	t^*
Gatter, 4 Eingänge	4	16	16 µs
ALU, 68 Eingänge	68	$3 \cdot 10^{20}$	10^7 Jahre
vier Eingabevariablen vom Typ int32_t	128	$3 \cdot 10^{38}$	10^{25} Jahre

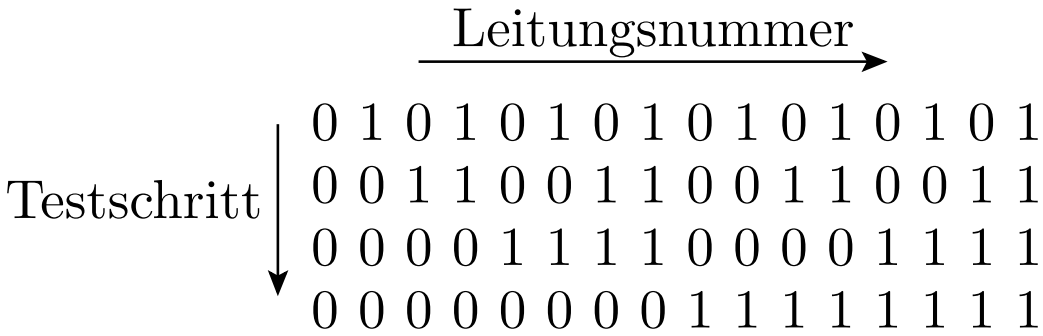
(m – Anzahl der Eingabebits; 2^m – Anzahl der Eingabemöglichkeiten; t^* – Testdauer bei einer Service-Ausführungszeit von 1µs.)

- Die meisten Systeme verarbeiten mehr als 128 Eingabebits.
- Hinzu kommen oft tausende oder mehr gespeicherte Bits, die auch mit variiert werden müssten.
- Geschätzte Zeit seit dem Urknall $14 \cdot 10^9$ Jahre.
- Es gibt auch Fehler, die verlangen Eingabefolgen für den Nachweis. Alle Variationen von 2, 3, ... Eingabevektoren ...

Nachweis aller unterstellten Fehler

Für regelmäßig strukturierte Schaltungen lassen sich oft Algorithmen für die Eingabegenerierung so formulieren, dass alle Fehler nach einem bestimmten Fehlemodell nachweisbar sind. Beispiel Nachweis alle Kurzschlüsse auf einer Leiterplatte, wenn auf allen Leitungen 0 oder 1 gesteuert werden kann und die Leitungen beobachtbar sind.

Hinreichende Nachweisbedingung für alle Kurzschlüsse ist, dass sich die gesteuerten Leitungspegel paarweise in mindestens einem Testschritt unterscheiden.



Logarithmische Zunahme der Testsatzlänge mit der Anzahl der potentiell kurzgeschlossenen Leitungen.

Ein Beispiel für einenen Speichertest folgt später in Abschn. 2.7.

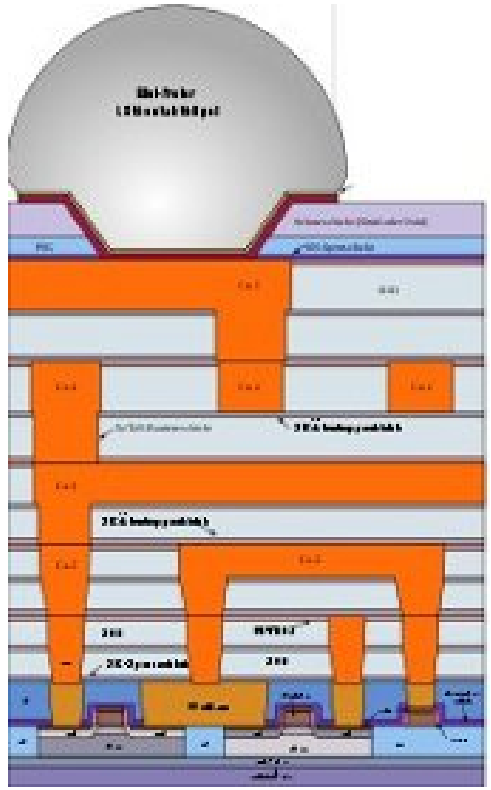
Schaltkreisfehler

Entstehung und Fehler integrierter Schaltkreise

Die Herausforderung für die Testauswahl sind hochintegrierte Schaltkreise mit ihren extrem vielen logischen Gattern und Verbindungen, die nicht von außen direkt steuerbar und beobachtbar sind.

Schaltkreise entstehen schichtenweise:

- 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
- überflüssiges Material (zu viel aufgetragen, zu wenige weggeätzt).

Einteilung in lokale und globale Fehler

Globale Fehler:

- Fehlerhafte Schichteigenschaften durch Prozesssteuerfehler. Betroffen sind alle Strukturelemente derselben Halbleiter-, Leitungs- oder Isolationsschicht.
- Großflächig überflüssiges oder fehlendes Material. Mehrfachkurzschlüsse oder Unterbrechungen.

Lokale Fehler:

- Unterbrechungen von Verbindungen,
- Kurzschlüsse zwischen benachbarten leitenden Gebieten.
- Transistoren, die nicht richtig ein- oder ausschalten,
- Leckströme ohne logische Fehlerwirkung.

Globale Fehler für Testauswahl uninteressant

Fehlerhafte Schichteigenschaften durch Prozesssteuerfehler:

- Überwachung auf Prozesssteuerfehler während der Fertigung.
- Kontrolle der Eigenschaften erzeugter Schichten nach Prozessschritten.
- Stichprobenkontrolle der Transistoreigenschaften, Leitwerte und Kapazitäten nach der Fertigung an speziellen Testschaltungen auf dem Wafer.
- Ausmessen der elektrischen Eigenschaften an den Anschlüssen incl. Versorgungsstrom.

Großflächig überflüssiges oder fehlenden Material:

- verursachen häufig FF oder komplette Funktionsunfähigkeit,
- erkennbar in der Regel beim Ausmessen der elektrische Anschlusseigenschaften oder vom sich anschließenden Grobtest.

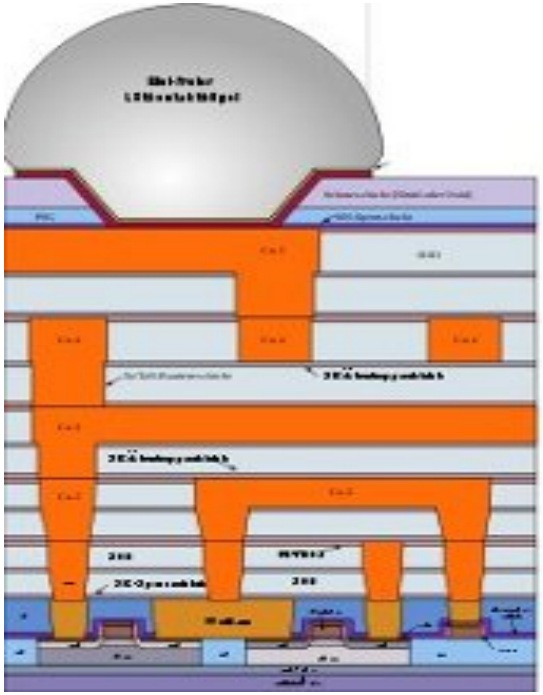
Anspruchsvoll ist die Suche nach kleinen Defekten, die überall sein können und nur selten FF verursachen.

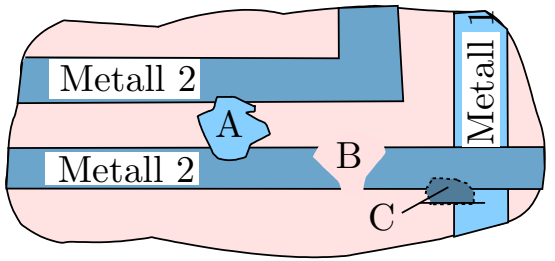
Lokale Fehler

Verbindungs- und Transistorfehler

Einzelfehler durch fehlendes und überflüssiges Material:

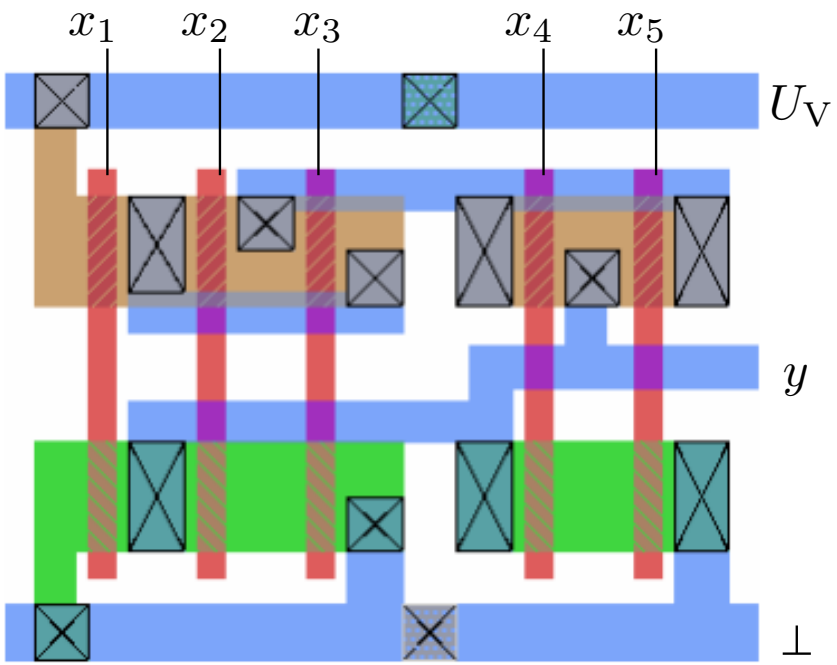
- kurzgeschlossene und unterbrochene Verbindung,
- nicht richtig ein- oder ausschaltende Transistoren,
- Leckströme ohne Beeinrächtigung der logischen Funktion, ...
- überhöhte Stromdichten oder Feldstärken, die zu Frühausfällen führen.

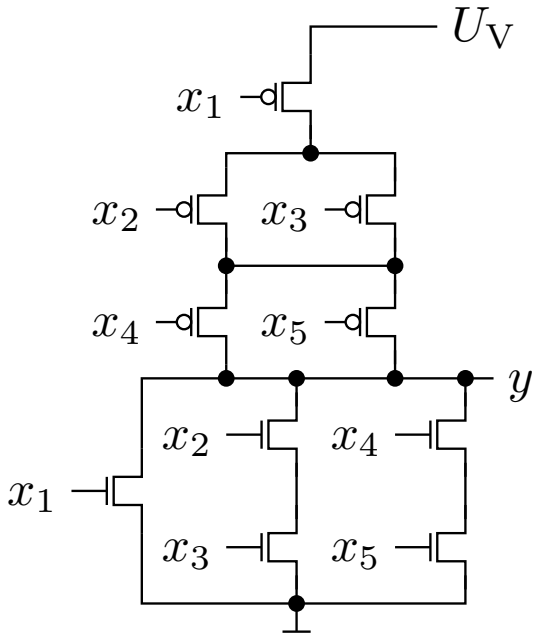




A zuätzliches Metall
B fehlendes Metall
C fehlende Isolation

Transistorebene



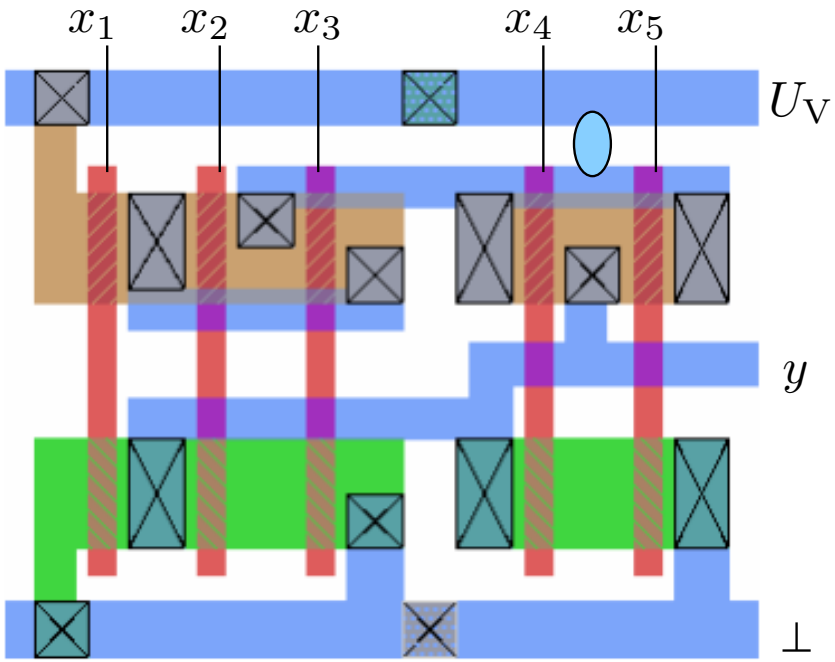


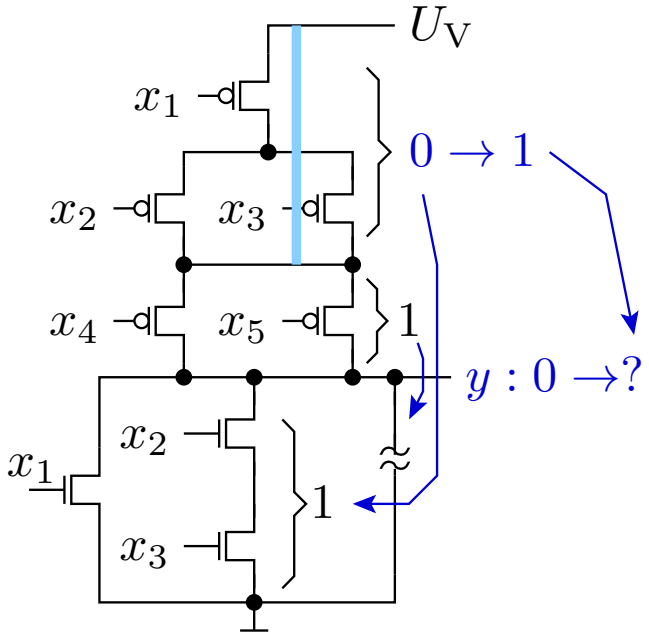
Durchkontaktierung
n-Gebiet
p-Gebiet
Polysilizium
Metall

$$y = \begin{cases} 1 & \text{wenn } \bar{x}_1 \wedge (\bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_4 \vee \bar{x}_5) \\ 0 & \text{wenn } x_1 \vee (x_2 \wedge x_3) \vee (x_4 \wedge x_5) \end{cases}$$

$$= \overline{x_1 \vee (x_2 \wedge x_3) \vee (x_4 \wedge x_5)}$$

Kurzschluss im Gatters





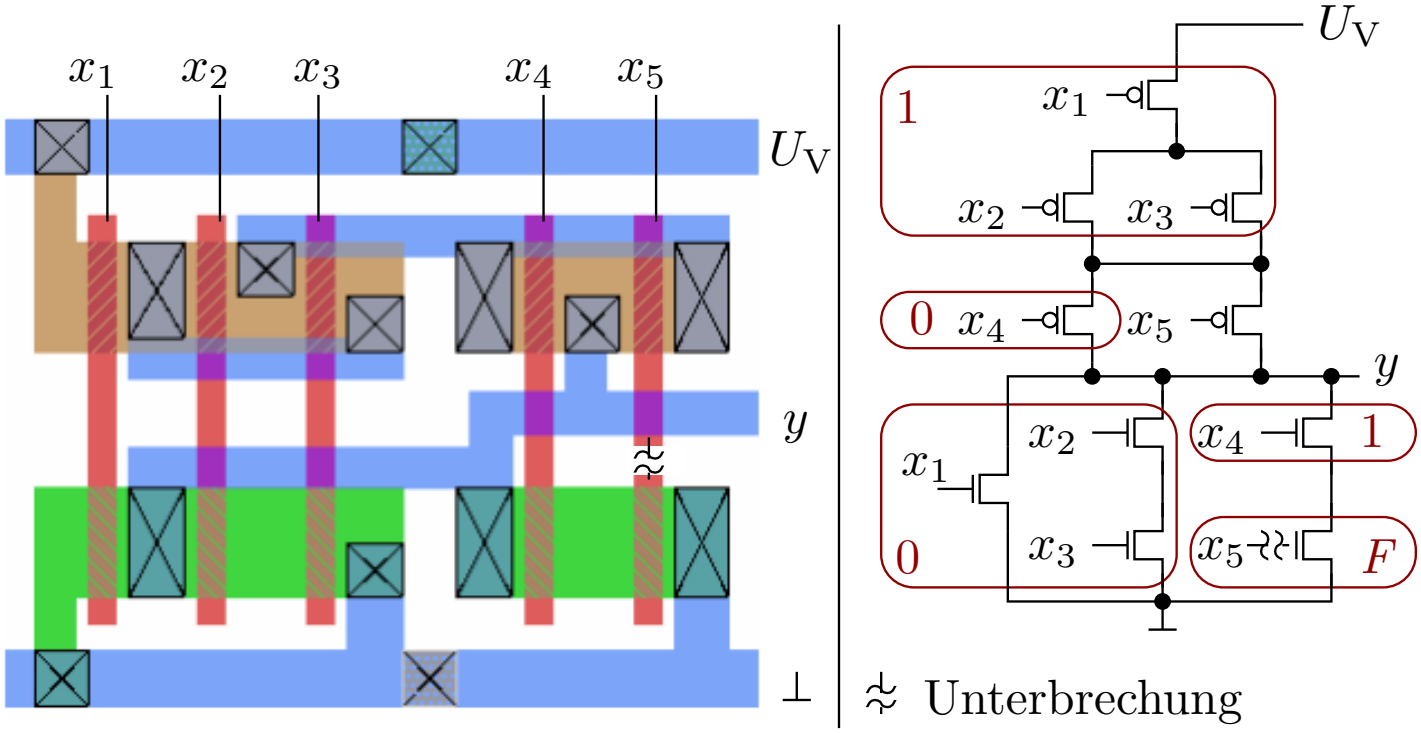
Kurzschluss

? erhöhter Ruhestrom
[+ Verfälschung $0 \to 1$]

Nachweisbar bei $\bar{x}_1 \wedge (\bar{x}_2 \vee \bar{x}_3) = 0$ UND $\bar{x}_4 \vee \bar{x}_5 = 1$ über

- statischen Ruhestrom (N- und PMOS-Netzwerk gleichzeitig ein),
- über eine logisch FF, wenn y beobachtbar ist und die Transistorbreiten so sind, dass der Kurzschluss y invertiert,
- möglichweise auch über längere Schaltverzögerungen.

Offenes Gate

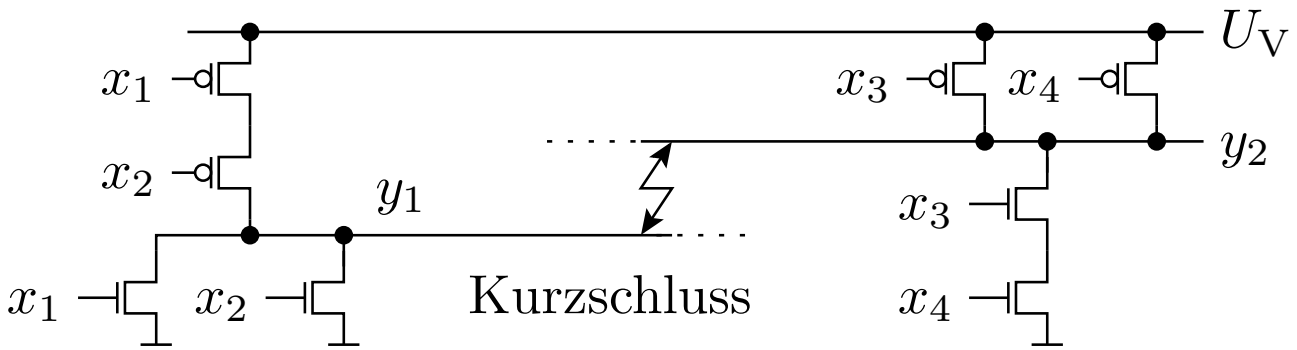


Nachweisvoraussetzungen: $x_1 \vee (x_2 \wedge x_3) = 0$ UND $x_4 = 1$

- wenn $F = 0$ zusätzlich $x_5 = 1$: kein Wechsel $y : \Downarrow \rightarrow 1$
- wenn $F = 1$ zusätzlich $x_5 = 0$: Ruhestrom [+ $y : 1 \rightarrow 0$]
- sicherer Nachweis: $y : \Uparrow\Downarrow$ durch $y \neq x_5$ nach 1. oder 2. Wechsel

(F– Schaltzustand des Transistors mit dem offenen Gate, kann während des Tests auch langsam wechseln).

Kurzschluss zweier Gatterausgänge



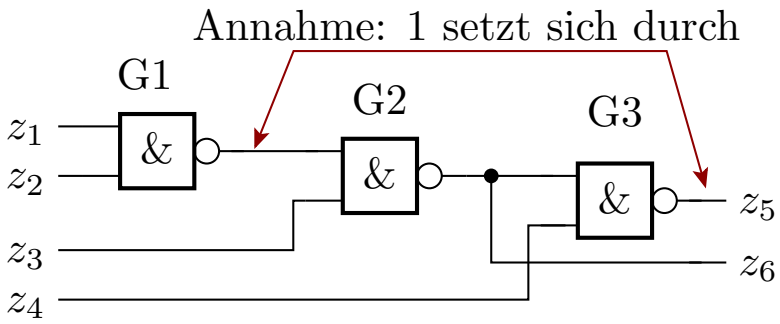
Mögliche Nachweisbedingungen:

1. $\bar{x}_1 \wedge \bar{x}_2 = 1$ und $x_3 \wedge x_4 = 1$ ($y_{1\text{Soll}} = 1$ und $y_{2\text{Soll}} = 0$)
2. $x_1 \vee x_2 = 1$ und $\bar{x}_3 \vee \bar{x}_4 = 1$ ($y_{1\text{Soll}} = 0$ und $y_{2\text{Soll}} = 1$)

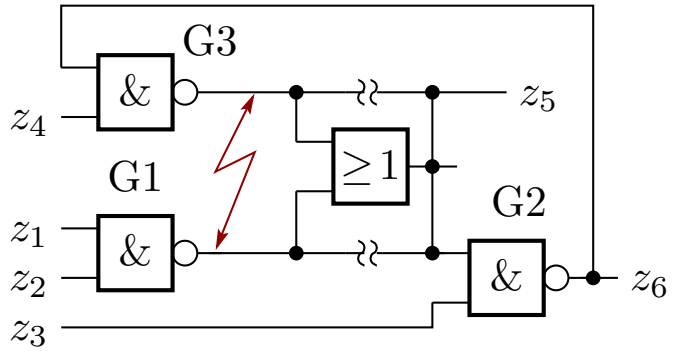
Ob sich dabei $y_1 = y_2 = 0$ oder $y_1 = y_2 = 1$ durchsetzt, hängt von den Transistorbreiten, bzw. Transistorsteilheiten ab. Der verfälschte y-Wert muss zusätzlich beobachtbar sein.

Zusätzliches Speicherverhalten durch Kurzschluss

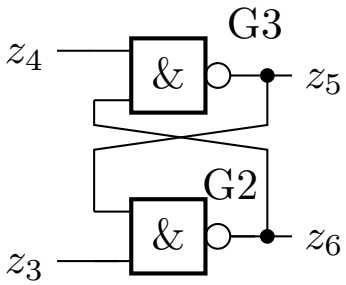
Schaltung mit Kurzschluss

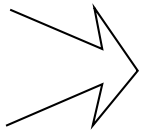


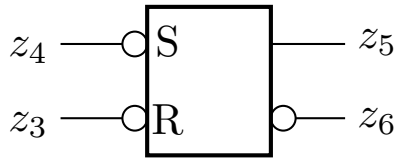
Kurzschlussnachbildung durch ein ODER



Ersatzschaltung für $z_1 = z_2 = 1$



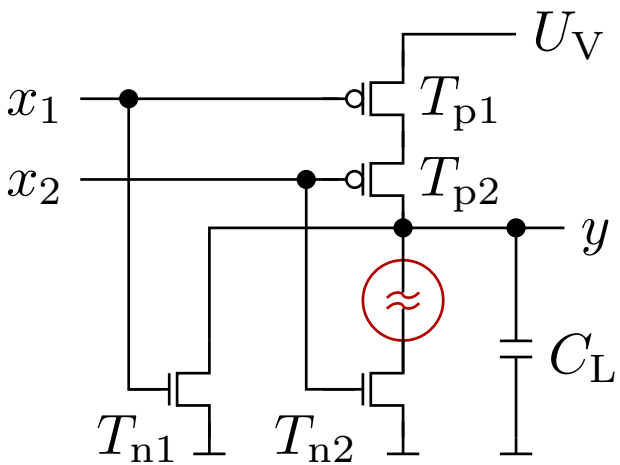




z_4	z_3	z_2	z_1	z_6	z_5	
0	1	1	1	0	1	(setzen)
1	0	1	1	1	0	(löschen)
1	1	1	1	speichern		

z_4	z_3	z_2	z_1	z_6	z_5	
0	0	1	1	1	1	
↓	↓			↓	↓	ändern nach
1	1	1	1	unbestimmt		

Stuck-Open-Fehler



x_2	x_1	y
0	0	1 (setzen)
0	1	0 (rücksetzen)
1	0	speichern
1	1	0 (rücksetzen)

x_2	x_1	y
0	0	1
0	1	0
0	0	1
1	0	0

Ausgewählte Unterbrechungen und Transistordefekte können bewirken, dass Gatterausgänge für bestimmte logische Eingaben isoliert sind oder nur zu langsame auf- oder entladen werden. Dieser Fehlertyp wird als Stuck-Open-Fehler bezeichnet und lässt sich nur über Schaltvorgänge am Gatterausgang zuverlässig nachweisen.

FM für DIC

Fehlermodelle (FM) für digitale Schaltkreise

Etablierte Fehlermodelle für digitale Schaltkreise:

- Haftfehler,
- Verzögerungsfehler,
- IDDQ-Fehler und
- Zellenfehler (regelmäßig strukturierte DIC, RAM, ...).

Nicht praxistauglich:

1. Toggle-Fehler,
2. Kurzschlüsse, Unterbrechungen,
3. Mehrfachfehler, Pfadverzögerungsfehler, ...

Warum[1]:

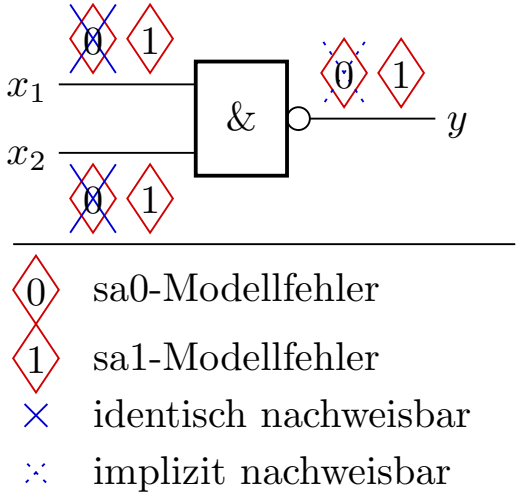
1. Wahrscheinlichkeit, dass Tests für Modellfehler tatsächliche Fehler nachweisen, gering.
2. Fehlersimulation und Testberechnung zu kompliziert.
3. Überproportionale Zunahme der Modellfehleranzahl mit der Testobjektgröße.

[1] Falls fehlerorientierte Testauswahl einmal für SW relevant wird, wird auch die Frage interessant, was für Fehlermodelle sich für Hardware nicht bewährt haben.

Haftfehler

Für jeden Gatteranschluss wird unterstellt:

- ein sa0 (stuck-at-0) Fehler
- ein sa1 (stuck-at-1) Fehler



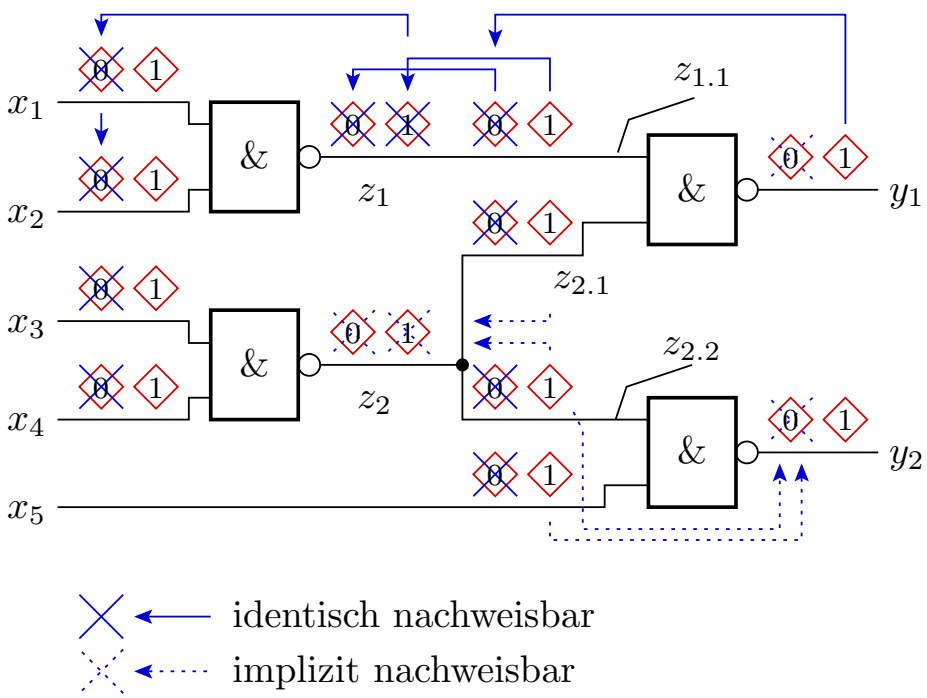
x_2	x_1	$\overline{x_2 \wedge x_1}$	sa0(x_1)	sa1(x_1)	sa0(x_2)	sa1(x_2)	sa0(y)	sa1(y)
0	0	1	1	1	1	1	0	1
0	1	1	1	1	1	0	0	1
1	0	1	1	0	1	1	0	1
1	1	0	1	0	1	0	0	1
			Nachweisidentität (gleiche Nachweismenge)					

Nachweisimplikation

zugehörige Eingabe ist Element der Nachweismenge

Zusammenfassung identisch nachweisbarer Fehler. Optionale Streichung redundanter und implizit nachweisbarer Modellfehler.

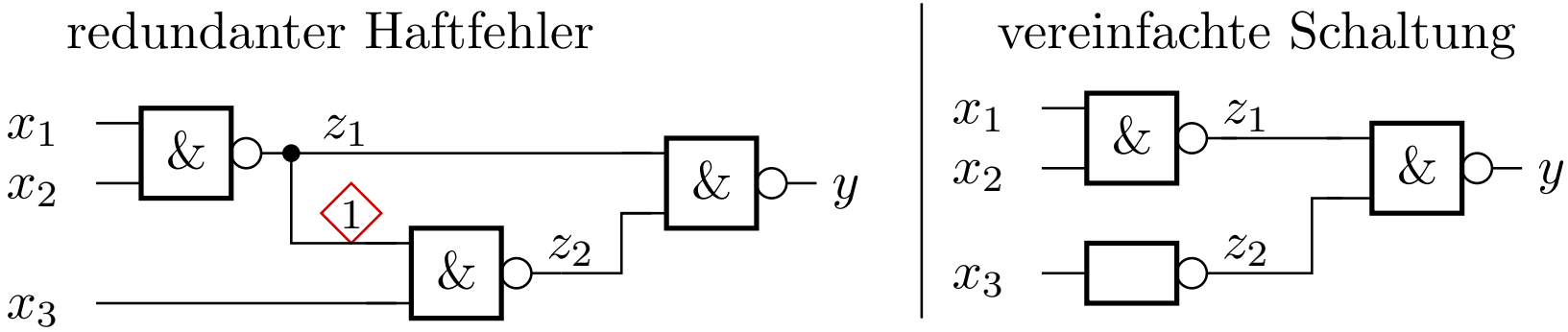
Identisch und implizit nachweisbare Fehler im Schaltungsverbund



Größe der Anfangsfehlermenge:	24
Anzahl der nicht identisch nachweisbaren Fehler:	14
ohne implizit nachgewiesene Fehler:	10

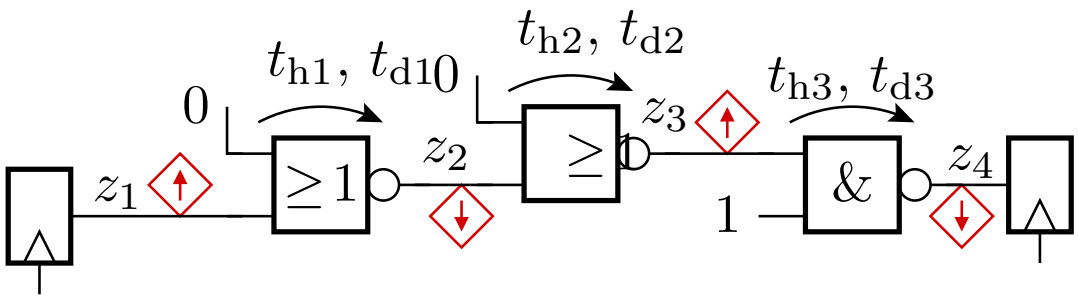
	Mengen von identisch nachweisbaren Fehlern	Nachweis impliziert durch
1	sa0(x_1), sa0(x_2), sa1(z_1), sa1($z_{1.1}$)	
2	sa1(x_1)	
3	sa1(x_2)	
4	sa0(x_3), sa0(x_4), sa1(z_2)	9, 12
5	sa1(x_3)	
6	sa1(x_4)	
7	sa0(z_2)	5, 6, 8, 11
8	sa0(z_1), sa0($z_{1.1}$), sa0($z_{2.1}$), sa1(y_1)	2, 3
9	sa1($z_{2.1}$)	
10	sa0(y_1)	1, 9
11	sa0($z_{2.2}$), sa0(x_5), sa1(y_2)	
12	sa1($z_{2.2}$)	
13	sa1(x_5)	
14	sa0(y_2)	12, 13

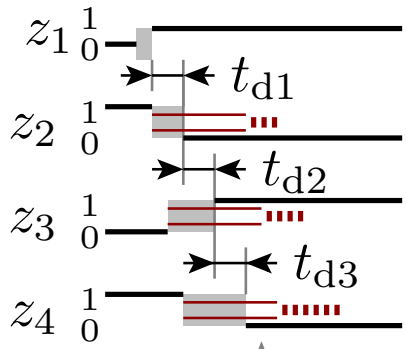
Redundante Fehler



- Bei $z_1 = 0$ ist der Fehler an y nicht beobachtbar und mit $z_1 = 1$ wird der Fehler nicht angeregt. Gatteranschluss mit sa1-Fehler kann mit »1« verbunden werden, ohne dass sich die Funktion ändert. Möglichkeit der Schaltungsvereinfachung.
- Der Nachweis der Redundanz kann schwieriger sein als die Suche eines Tests. Falls nicht erkannt, werden redundante Fehler als nicht erkannt gezählt und verringern die berechnete gegenüber der tatsächlichen Modellfehlerüberdeckung..
- Fehlermodelle, die viele redundante Fehler erzeugen sind zur Schätzung der tatsächlichen Fehlerüberdeckung ungeeignet.

Gatterverzögerungsfehler





verzögerter Anstieg

verzögerter Abfall

Abtastzeitpunkt

Fehler sicher nachweisbar

Fehler Nachweis möglich

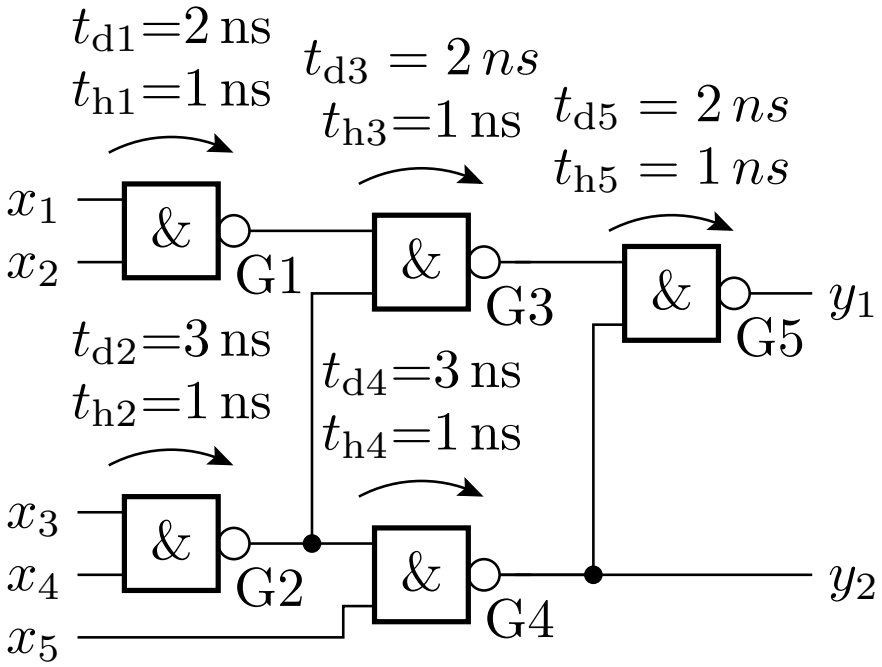
Fehler nicht nachweisbar

Statt der Haftfehler, Annahme an allen Gatteranschlüssen

- eines Slow-To-Raise- (verzögerter Signalanstieg) und
- eines Slow-To-Fall- (verzögerter Signalabfall)

Streichen identisch nachweisbarere, redundanter und optional implizit nachweisbarer Modellehler. Nachweis über einen Signalpfad durch die Schaltung, der an einem Abtastregister beginnt und endet durch eine Folge aus Initialisierungs- und Testeingabe.

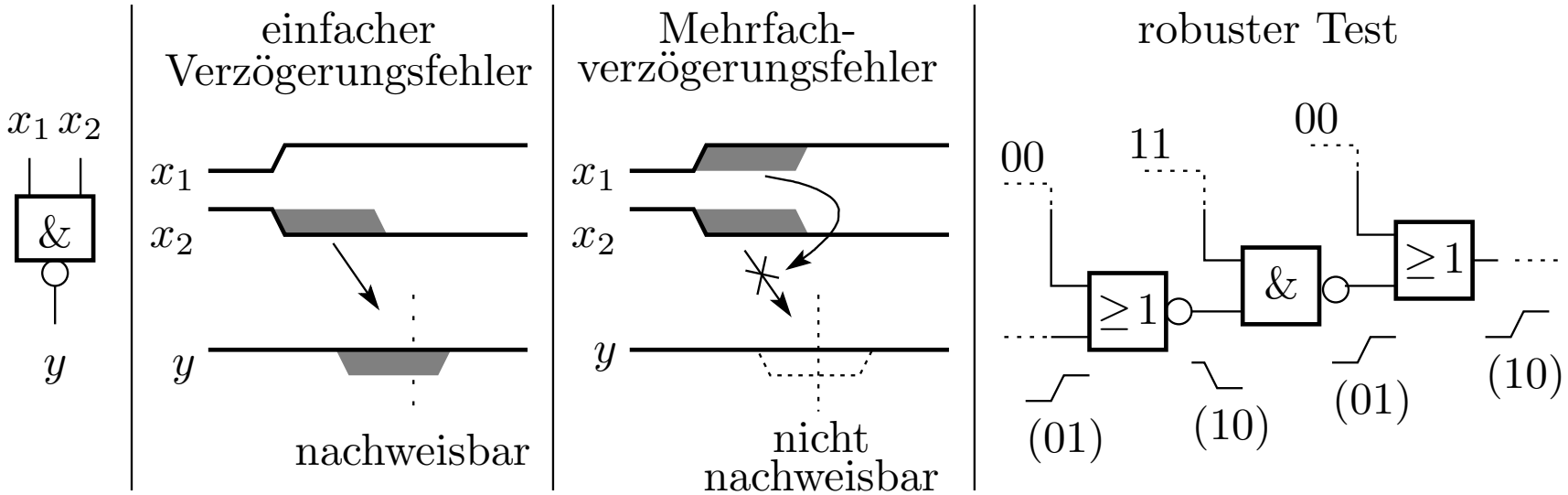
Nachweiswahrscheinlichkeit und Pfadlänge



Pfade	$\sum t_{h.i}$	$\sum t_{d.i}$
G1-G3-G5	3 ns	6 ns
G2-G3-G5	3 ns	7 ns
G2-G4-G5	3 ns	8 ns
G2-G4	2 ns	6 ns
G4-G5	2 ns	5 ns
G4	1 ns	3 ns

- Die minimal erkennbare Zusatzverzögerung ist die Differenz aus Taktperiode und Soll-Verzögerung.
- Je länger die Sollverzögerung, desto höher die Wahrscheinlichkeit, Fehlerverursachte Zusatzverzögerungen zu erkennen.
- Tests über die Pfade mit den längsten Sollverzögerungen erkennen alle Einzelverzögerungsfehler.

Mehrfachsfehler



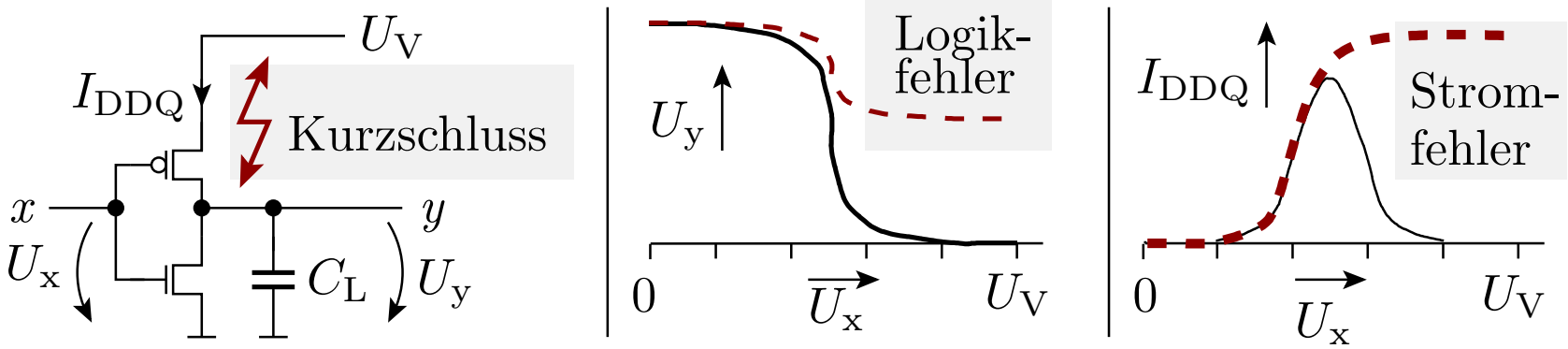
- Es sind Mehrfachfehler konstruierbar, die bei bestimmten Tests gegenseitig maskieren.
- Robuster Test: Ausschluss der gegenseitigen Maskierung, durch max. eine Signaländerung an den Eingängen jedes Gatters je Testschritt.
- Zusatzattribute wie »minimal erkennbare Zusatzverzögerung« und »Robustheit« bei der Fehlersimulation /Testberechnung mit berechnenbar bzw. berücksichtigbar.

Pfadverzögerungsfehler

Annahme eines Slow-To-Rise- und eines Slow-To-Fall-Fehlers für alle Schaltungspfade. Garantiert, dass für jeden Gatterverzögerungsfehler die bestmöglichen Tests gesucht werden. Nicht zielführend weil:

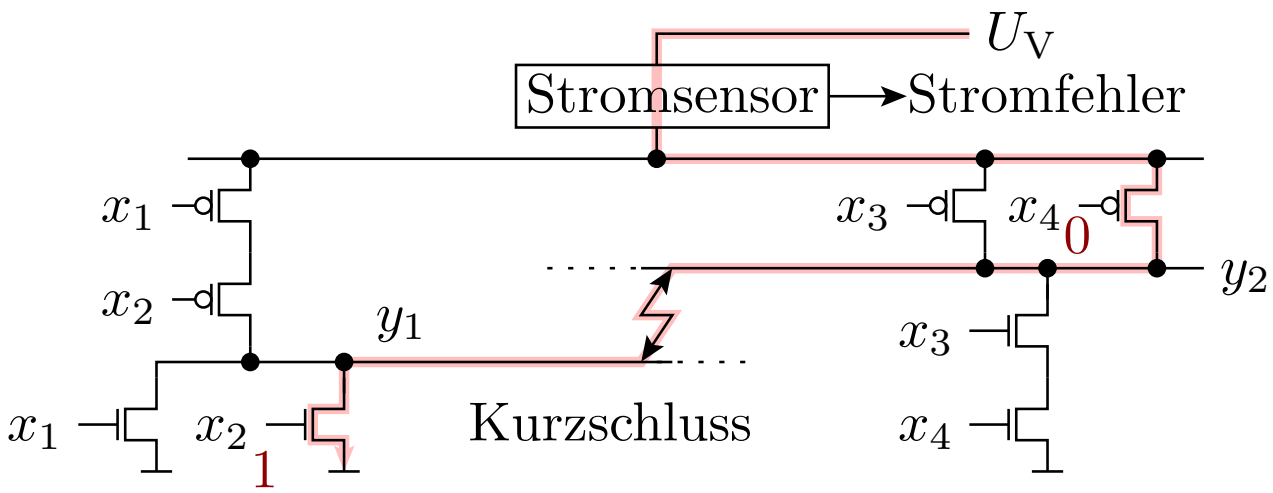
- Überproportionales (im ungünstigsten Fall exponentielles) Wachstum der Modellfehleranzahl mit der Systemgröße.
- Im selben Maße wächst der Aufwand für Fehlersimulation / Testsuche.
- Mehr Modellfehler je Systemgröße lassen vermuten, dass der Anteil der redundanten Fehler und der untereinander ähnlich nachweisbaren Fehler zunimmt.
- Es ist kaum belegbar bzw. untersucht, dass der Zusatzaufwand für Testsuche / Fehlersimulation zu besseren Testsätzen oder einer besseren Vorhersagegenauigkeit der tatsächlichen Fehlerüberdeckung führt.

Ruhestromüberwachung



In einer CMOS-Schaltung ist der Gatterausgang nur entweder über NMOS-Transistoren mit 0 (Masse) oder über PMOS-Transistoren mit 1 (Versorgungsspannung) verbunden. Nach jedem Schaltvorgang klingt der Strom auf einen sehr kleinen Wert ab. Die Hälfte der zu erwartenden lokalen Schaltkreisfehler sind Kurzschlüssen, nicht richtig ausschaltende Transistoren, ... die, wenn sie zur Wirkung kommen, ein messbaren Ruhestrom I_{DDQ} verursachen. I_{DDQ}-Überwachung erkennt Defekte auch ohne Beobachtungspfad zu einem Ausgang.

Kurzschlussnachweis über den Ruhestrom



Nachweis über statische Stromaufnahme, wenn $y_1 \neq y_2$

- Vorteile I_{DDQ}–Test: Einfachere logische Nachweisbedingungen, einfachere fehlerorientierte Testsuche, kürzere Zufallstetest bei gleicher Fehlerüberdeckung.
- Probleme I_{DDQ}–Test: Unterscheidung von zulässigem und überhöhtem Ruhestrom funktioniert nur bis einige tausend Gatter. Integrierte Stromsensoren, ...

Toggle-Test und Zellenfehler

Toggle-Test (alle Signale sind während des Test mindestens einmal auf »0« und einmal auf »1« zu steuern): bedingte Wahrscheinlichkeiten, dass Tests, die lokal »0« oder »1« einstellen, zufällig einen lokalen Fehler anregen und beobachtbar machen, ist gering.

Zellenfehlermodell Annahme, dass die logische Funktion eines kleinen Logikbausteins fehlerhaft ist. Für die Konstruktion algorithmischer Tests für regelmäßig strukturierte Schaltungen wie Speicher nützlich.
Bei freistrukturierten Schaltungen könnte man theoretisch jedes Gatter als Zelle betrachten und versuchen, es mit der kompletten Wertetabelle zu testen. Führt zu vielen redundanten im Schaltungsverbund nicht nachweisbaren Fehlerannahmen.

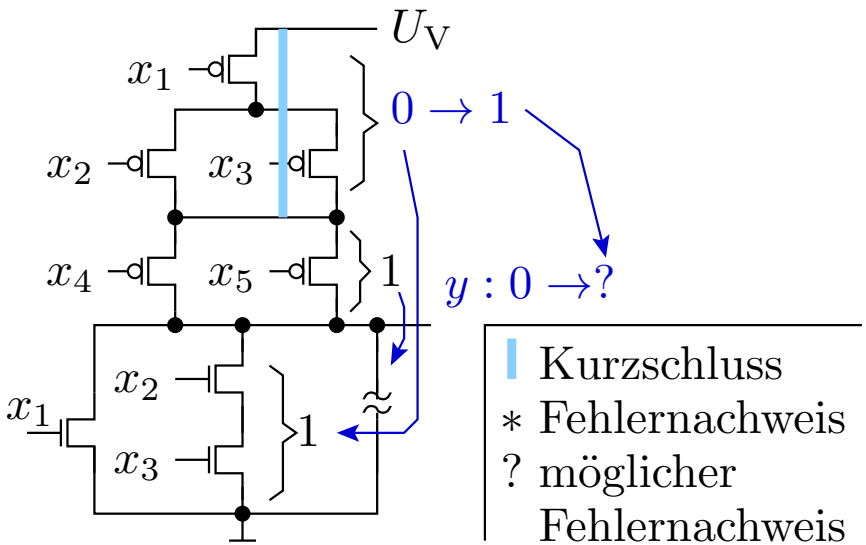
Ein guten Fehlermodell ist ein Kompromiss zwischen Rechenaufwand und Aussagewert für die Fehlerüberdeckung für tatsächliche Fehler.

Nachweisbeziehungen

Nachweisbeziehungen zwischen Modellfehlern und tatsächlichen Fehlern

Die etablierten Fehlermodelle für Schaltkreise, modellieren nicht das Verhalten von zu erwartenden Fehlern, sondern generieren eine große Menge von Modellfehlern, in der für möglichst alle zu erwartenden Fehler ähnlich nachweisbare Fehler enthalten sind, deren Nachweis mit einer hinreichenden Wahrscheinlichkeit den Nachweis der tatsächlichen Fehler impliziert.

Kurzschlussnachweis mit Haftfehlertests



$x_3\,x_2\,x_1$		$\mathrm{sa0}(x_3)$	$\mathrm{sa0}(x_2)$	$\mathrm{sa0}(x_1)$
0 0 0				
0 0 1	?			∗
0 1 0				
0 1 1	?			∗
1 0 0				
1 0 1	?			∗
1 1 0	?	∗	∗	
1 1 1	?			

Statischer Ruhestrom und eventuell $y : 0 \to 1$ für $y_{\mathrm{soll}} = 0$ UND $\bar{x}_4 \vee \bar{x}_5 = 1$. Haftfehler mit ähnlichen Nachweisbedingungen:

Haftfehler	zusätzliche Nachweisbedingung
$\mathrm{sa0}\,(x_1)$	$x_1 = 1$ und ($x_2 = 0$ oder $x_3 = 0$)
$\mathrm{sa0}\,(x_2)$	$x_2 = 1$ und $x_1 = 0$ und $x_3 = 1$
$\mathrm{sa0}\,(x_3)$	$x_3 = 1$ und $x_1 = 0$ und $x_2 = 1$

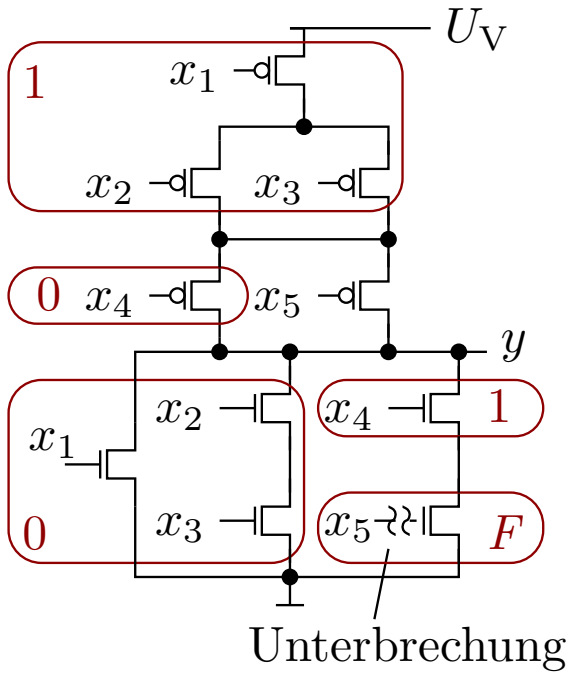
»Offenes Gate« mit Haftfehlertests

Wenn $a = \bar{x}_1 \wedge (\bar{x}_2 \vee \bar{x}_3) \wedge x_4$ und

- $y = 0$ und $x_5 = \Uparrow$: kein $y = \Downarrow$
- abgetrennter Transistor dauerhaft an und $x_5 = 0$: eventuell kein $y = \Uparrow$

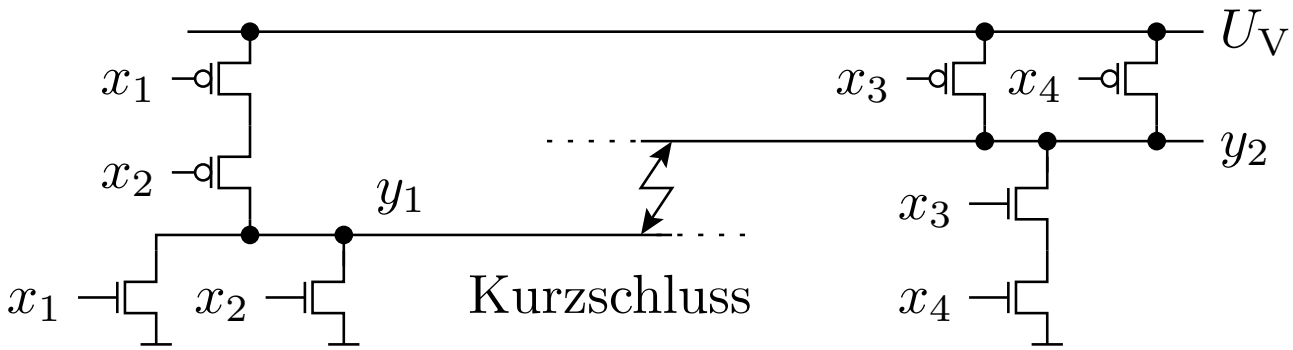
Ähnlich nachweisbare Haftfehler:

- sa0 (x_5) wenn $x_5 = \Uparrow$
- sa1 (x_5) wenn Reihenschaltung NMOS x_5, x_4 niederohmiger als Reihenschaltung PMOS x_5, x_2 [$\|$ x_3], x_1



(a – gemeinsame Anregungsbedingung; $\Uparrow$ – Wechsel 0 nach 1; $\Downarrow$ – Wechsel von 1 nach 0). Zu beiden Nachweismöglichkeiten gibt es einen Haftfehler, bei derem Nachweis die Unterbrechung am Gate unter je einer Zusatzbedingung auch nachgewiesen wird.

Kurzschluss zweier Gatterausgänge



Nachweis über statische Stromaufnahme: $y_1 \neq y_2$

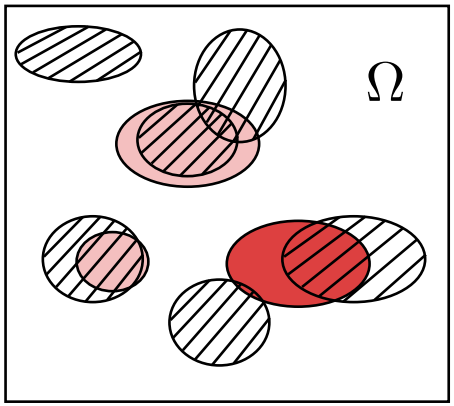
Bedingungen für einen möglichen logischen Nachweis:

- y_1 und y_2 müssen sich im fehlerfreien Fall unterscheiden.
- Je nach Fehlerwirkung müssen y_1 oder y_2 dabei beobachtbar sein.

Ähnlich nachweisbare Haftfehler:

- $\mathrm{sa0}\,(x_1)\,,\ \mathrm{sa0}\,(x_2)$ wenn $y_2 = 1$ ist und sich durchsetzt
- $\mathrm{sa1}\,(x_1)\,,\ \mathrm{sa1}\,(x_2)$ wenn $y_2 = 0$ ist und sich durchsetzt
- $\mathrm{sa0}\,(x_3)\,,\ \mathrm{sa0}\,(x_4)$ wenn $y_1 = 1$ ist und sich durchsetzt
- $\mathrm{sa1}\,(x_3)\,,\ \mathrm{sa3}\,(x_4)$ wenn $y_1 = 0$ ist und sich durchsetzt

Fehler und Modellfehler



Ω	Menge aller Testeingabewerte
	Eingabewerte, die den Fehler nachweisen
	Eingabewerte, die den Fehler eventuell (unter Zusatzbedingungen) nachweisen
	Nachweismenge ähnlich nachweisbarer Modellfehler

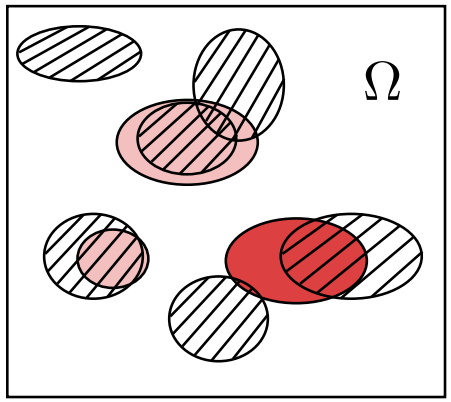
Für alle logisch nachweisbaren lokalen Fehler enthält die Modellfehlermenge ähnlich nachweisbare Haftfehler:

- ähnliche lokale Nachweisbedingung,
- übereinstimmende Anregungsbedingung,
- übereinstimmende Beobachtungsbedingung.

Jeder Test für einen Modellfehler i weist mit einer Wahrscheinlichkeit p_{ij} jeden der ähnlich nachweisbaren tatsächlichen Fehler $j$nach.

Fehlerüberdeckung bei fehlerorientierter Suche

Für jeden Modellfehler j wird ein Test gesucht und bei Erfolg $a_j \geq 1$ Tests gefunden. Jeder der a_j Tests weist Fehler i mit Wahrscheinlichkeit p_{ij} nach:



$$p_i = 1 - \prod_{j=1}^{n_i} (1 - p_{ij})^{a_j}$$

(n_i – Anzahl der ähnlich nachweisbaren Modellfehler zu Fehler i). Fehlerüberdeckung

$$FC \approx \frac{1}{\#F} \cdot \sum_{i=1}^{\#F} p_i$$

($\#F$ – Anzahl der tatsächlichen Fehler).

Modellrechnungen

Annahme derselben bedingten Wahrscheinlichkeit $p_{ij} = p$, dass ein Test für Modellfehler j den tatsächlichen Fehler i nachweist für alle Paare (»tatsächliche Fehler«, »ähnlich nachweisbarer Modellfehler«) und unterschiedliche Verteilungen $\mathbb{P}[X = k]$ für die Anzahl der gefundenen Tests k, die mit Wahrscheinlichkeit p den tatsächlichen Fehler i nachweisen.

k	0	1	2	3	4	5	6	7	8	9	10
A	1%	2%	30%	40%	7%	0					
B	1%	1%	2%	5%	7%	12%	15%	17%	17%	15%	8%

Fehlerüberdeckung:

$$FC = 1 - \sum_{k=1}^{k_{\max}} \mathbb{P}[X = k] \cdot (1-p)^k$$

	$p = 10\%$	$p = 25\%$	$p = 50\%$	$p = 75\%$
A:	23,0%	50,3%	80,1%	94,7%
B:	49,1%	81,7%	96,1%	98,5%

$$FC = 1 - \sum_{k=1}^{k_{\max}} \mathbb{P}\left[X = k\right] \cdot (1-p)^k$$

	$\mathbb{P}\left[X > 0\right]$	$p = 10\%$	$p = 25\%$	$p = 50\%$	$p = 75\%$
A:	99%	23,0%	50,3%	80,1%	94,7%
B:	99%	49,1%	81,7%	96,1%	98,5%

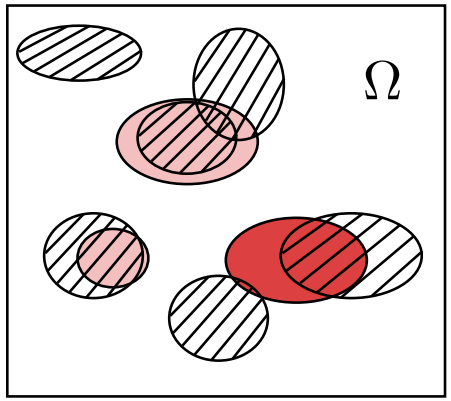
Ergebnisdiskussion:

- Die Wahrscheinlichkeit $\mathbb{P}\left[X = 0\right]$ (kein Test für ähnlich nachweisbaren Modellfehler), begrenzt FC nach oben.
- Ursache $k = 0$ kann sein, dass es keine ähnlich nachweisbaren Modellfehler gibt oder für diese keine Tests gefunden werden, mögliche, aber nicht sichere Korrelation mit FC_{M}.
- Die bedingte Wahrscheinlichkeit p und die Anzahl der Tests je Modellfehler haben offenbar auch erheblichen Einfluss.

Bei fehlerorientierter Testsuche ist es offenbar zielführender, für jeden Modellfehler mehrere unterschiedliche Tests zu suchen, als für jeden Modellfehler mindestens einen Test zu finden.

Fehlerüberdeckung bei zufälliger Auswahl

Die Nachweismengen lokaler Fehler sind wegen der geteilten Anregungs- und Beobachtungsbedingungen ähnlich groß, wie im Mittel die der ähnlich nachweisbaren Modellfehler. Berücksichtigung tendenzieller Unterschiede durch eine Skalierungsfaktor c:



$$h_{\mathrm{M}}(\zeta) \approx h(c \cdot \zeta)$$

Wenn die ähnlich nachweisbaren Modellfehler tendentiell mehr FF verursachen als die tatsächlichen Fehler $c < 1$ sonst $c \geq 1$. Nach Foliensatz 2, Abschn. 2.3 »Fehler- und Modellfehler« ist die Fehlerüberdeckung für Testsatzlänge n etwa die der Modellfehlerüberdeckung $c \cdot n$–fachen Testsatzlänge:

$$FC(n) \approx FC_{\mathrm{M}}(c \cdot n)$$

Bei zufälliger Testauswahl ist die Fehlerüberdeckung einfacher und genauer aus der Modellfehlerüberdeckung vorhersagbar als bei gezielter Suche.

Testsuche

Testsuche

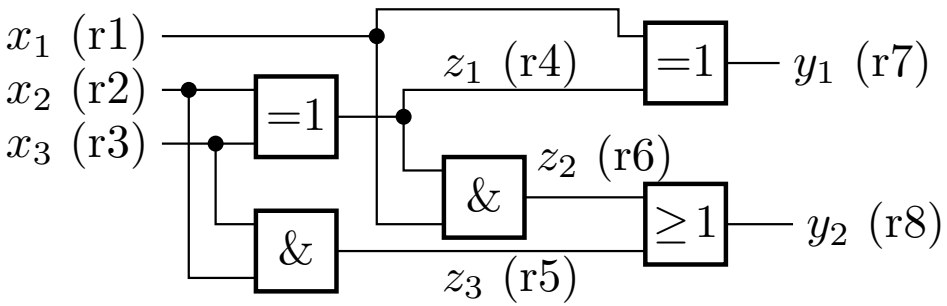
Zusammenstellen der Modellfehlermenge
Wiederhole, bis genug Modellfehler nachgewiesen werden
geziele, manuelle, zufällige Auswahl weiterer Testeingaben
Fehlersimulation, abhaken der nachweisbaren Modellfehler

- Zu Testsuche gehört eine Fehlersimulation.
- Für Modellfehler mit hoher FF-Rate genügt »Zufallsauswahl« der Testeingaben, bei Bedarf mit Aussortieren ausprobierter Testeingaben, die keine oder nur Modellfehler nachweisen, für die es bereits genug Testeingaben gibt.
- Für Modellfehler mit geringer FF-Rate gibt es Suchalgorithmen, die ausgehend vom Fehlerort Steuer- und Beobachtungspfade sensibiliesieren.
- Problematisch sind redundante Fehler, genauer der Nachweis, dass es für einen Modellfehler keine Nachweismöglichkeit gibt und er folglich nicht als »nicht nachweisbar« zu zählen ist.

Fehlersimulation

Simulation von Haftfehlern

Schaltung eines Volladdierers



r1 bis r8 Prozessorregister

Programm für die Gutsimulation

lade x_1 in Register r1
lade x_2 in Register r2
lade x_3 in Register r3
r4 = r2 xor r3
speichere Inhalt r4 in z_1
r5 = r2 and r3
speichere Inhalt r5 in z_3
r6 = r1 and r4
speichere Inhalt r6 in z_2
r7 = r1 xor r4
speichere Inhalt r7 in y_1
r8 = r5 or r6
speichere Inhalt r8 in y_2

- Jede zweistellige Logikoperation ist ein Maschinenbefehl.
- In jeder der 8, 16, 32 oder 64 Bits der Operanden kann ein anderer Testfall oder ein anderer Fehler simuliert werden.

Aufwandsabschätzung am Beispiel

- Schaltungsgröße: 10^4 Gatter
- Anzahl der Testschritte / Testeingaben: 10^4
- Anzahl der Modellfehler: 10^4
- Simulationsaufwand je Gatter: 10 ns

Rechenaufwand:

- wenn jeder Fehler mit allen Testeingaben simuliert wird ohne bitparallele Simulation: 10^4 s, ca. 3 h.
- Wenn mit jedem der 32 bzw. 64 Bits ein anderer Fehler simuliert wird, nur 6 bzw. 3 Minuten.
- Wenn bereits nachgewiesene Modellfehler nicht weiter mit simuliert werden, unter 1 Minute.

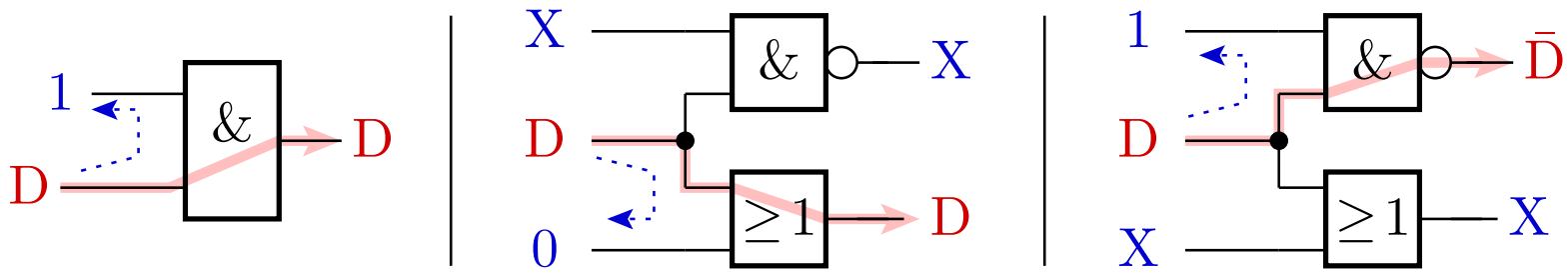
D-Algorithmus

D (Discrepancy)-Kalkül von Roth

Erweiterung der Logikwerte um 3 Pseudo-Werte[2]:

D	0 wenn unverfälscht, 1 wenn verfälscht.
$\bar{D}$	1 wenn unverfälscht, 0 wenn verfälscht.
X	Signalwert ist ungültig oder für den Fehlernachweis ohne Bedeutung.

Regeln für die Sensibilisierung eines Beobachtungspfades:



[2]W. Daehn: Testverfahren in der Mikroelektronik: Methoden und Werkzeuge. Springer 1997.

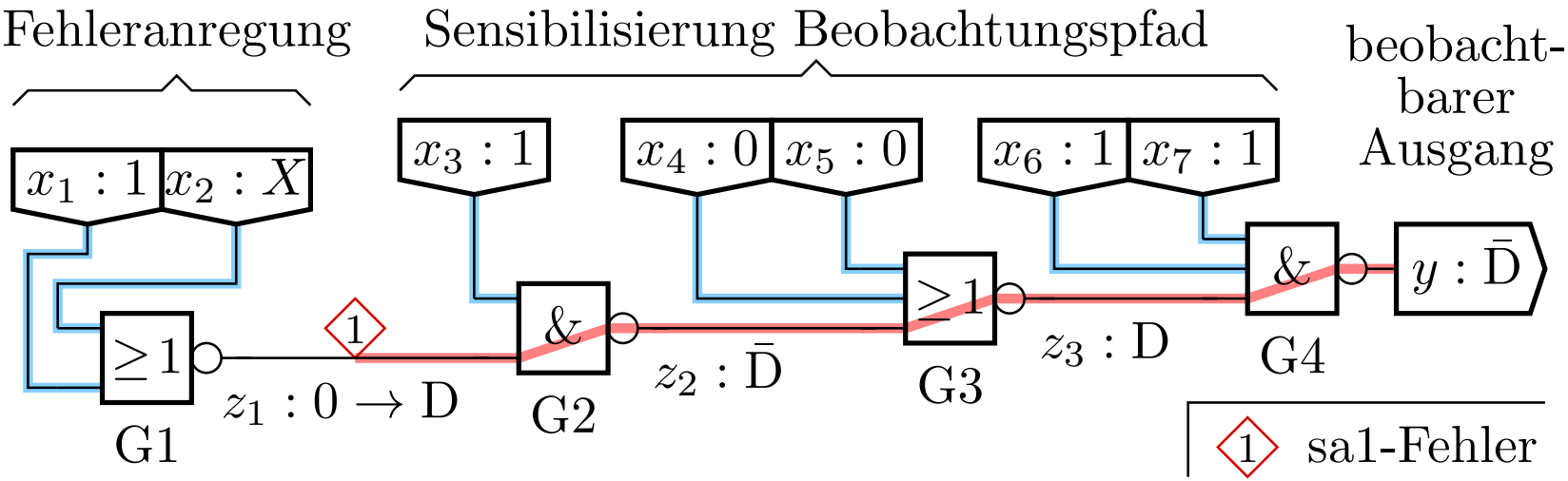
Testsuche für Haftfehler

Ein Haftfehler unterstellt für den Fehlerort, dass der Wert

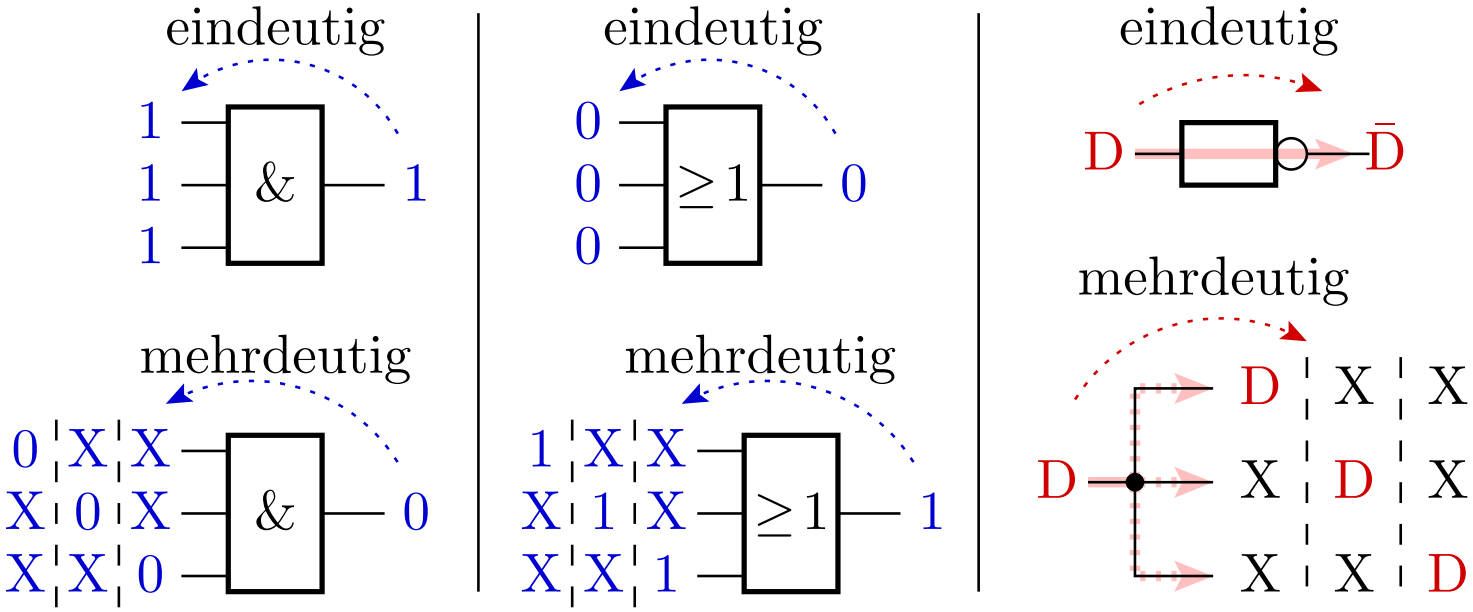
- entweder ständig 0 (sa0) oder
- ständig 1 ist (sa1) ist.

Ausgehend vom Fehlerort werden Eingaben gesucht,

- die den Wert am Fehlerort invertieren und
- bei denen die Invertierung am Fehlerort an einem Ausgang beobachtbar ist.

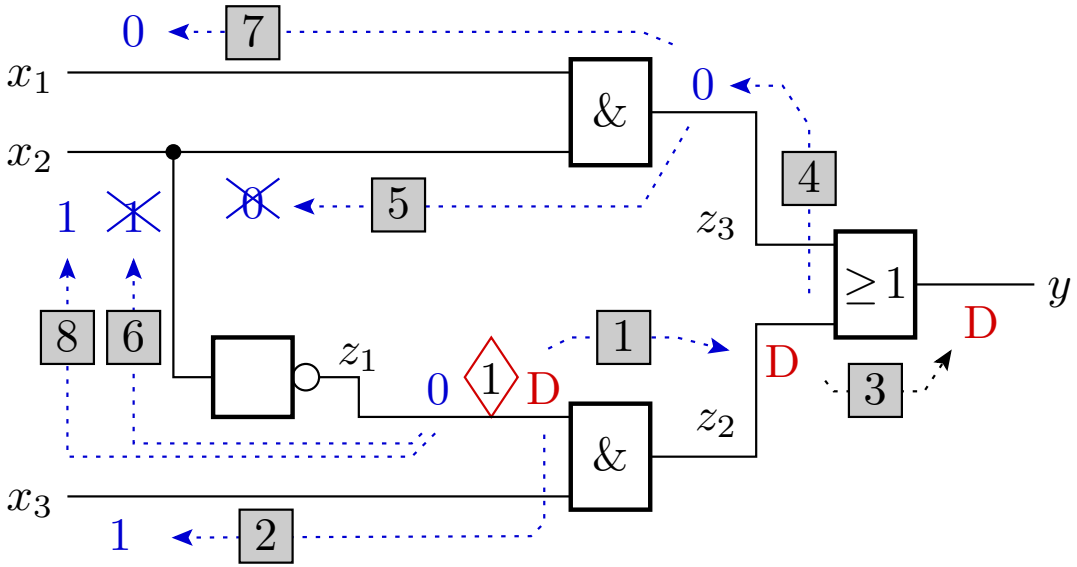


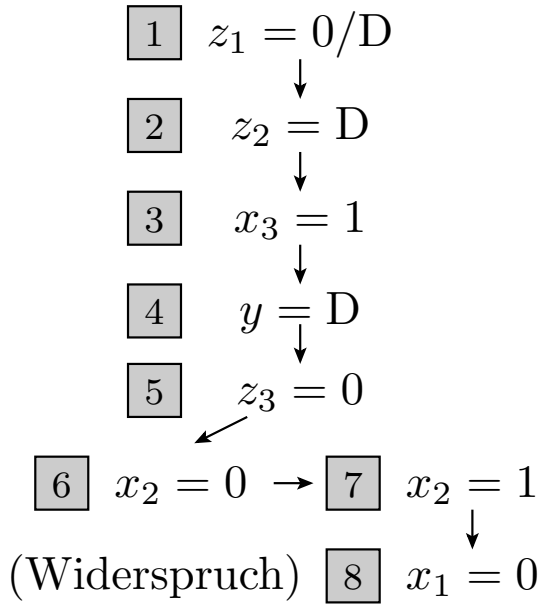
Ein- und mehrdeutige Pfade



Ausgehend vom Fehlerort:

- Festlegen von Werten zur Weiterführung des Beobachtungs- oder eines Steuerpfads.
- Bei Widersprüchen zurück zu letzten Möglichkeit einer Alternativentscheidung, ... $\Rightarrow$ Baumsuche





	x_3	x_2	x_1	z_3	z_2	z_1	y
1	X	X	X	X	X	0D	X
2	X	X	X	X	D	0D	X
3	1	X	X	X	D	0D	X
4	1	X	X	X	D	0D	D
5	1	X	X	0	D	0D	D
6	1	0	X	0	D	0D	D
7	1	~~0~~1	X	0	D	0D	D
8	1	1	0	0	D	0D	D

Baumsuche:

- Bei der Wertefestlegung können Widerspüche auftreten.
- Zurück zur letzten mehrdeutigen Entscheidung.
- Keine Lösung nach Durchmusterung des gesamten Baums. $\Rightarrow$ Fehler nicht nachweisbar

Erfolgsrate der Testberechnung:

- Anteil der Fehler, für die ein Test gefunden oder für die der Beweis »nicht nachweisbar« erbracht wird.

- Die Testsuche für einen Fehler kann hunderte von Wertefestlegungen beinhalten.
- Der Suchraum wächst exponentiell mit der Anzahl der mehrdeutigen Festlegungen. Suchräume der Größen $> 2^{30\ldots40}$ nicht mehr vollständig durchsuchbar.
- Abbruch der Suche nach einer bestimmten Rechenzeit.

Heuristiken:

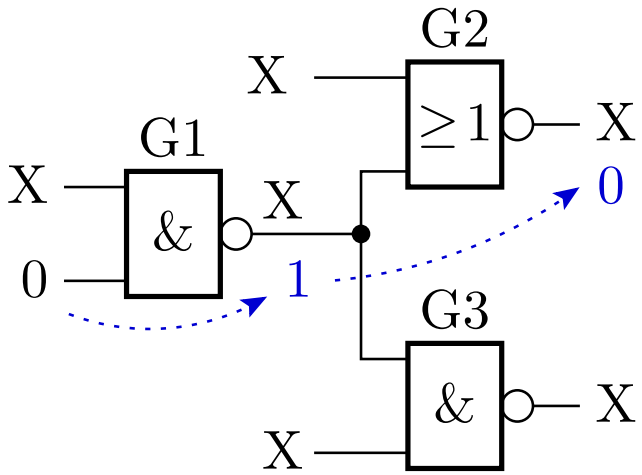
- Frühe Erkennung von Widersprüchen,
- Suchraumbegrenzung und
- gute Suchraumstrukturierung.

Implikationstest

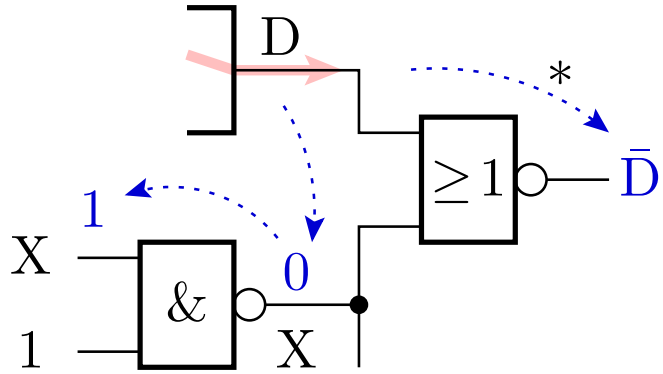
Implikationstest (Widerspruchsfrüherkennung)

- Aus den berechneten Wertefestlegungen alle eindeutig folgenden Werte berechnen.

Implikation in Signalflussrichtung



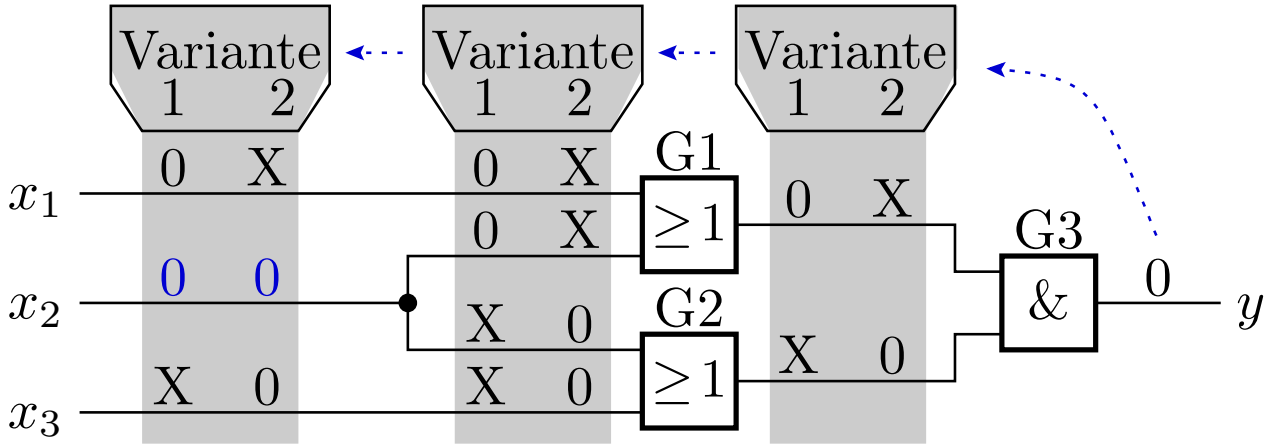
D-Pfad- und Rückwärtsimplikation



* eindeutige Weiterführung

- Mindert die Entscheidungsbaumtiefe.

- Rückwärtsimplikation über mehrere Gatterebenen:

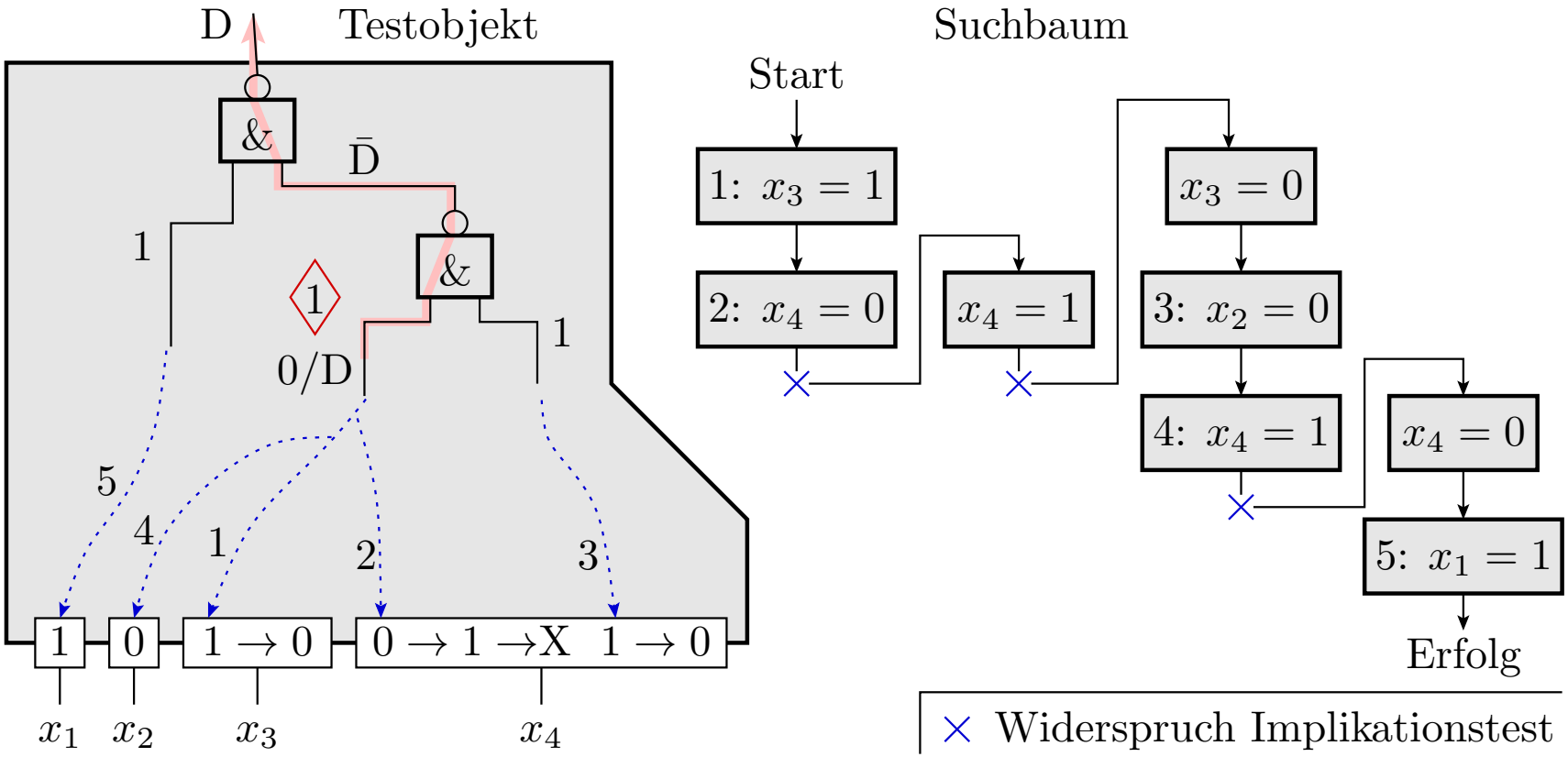


- Für $y = 0$ gibt es zwei Einstellmöglichkeiten.
- Für beide Möglichkeiten muss $x_2 = 0$ sein.
- Das Erkennen von Implikationen dieser Art mindert die Backtracking-Häufigkeit um bis zu 80 %.

Suchraumstrukturierung

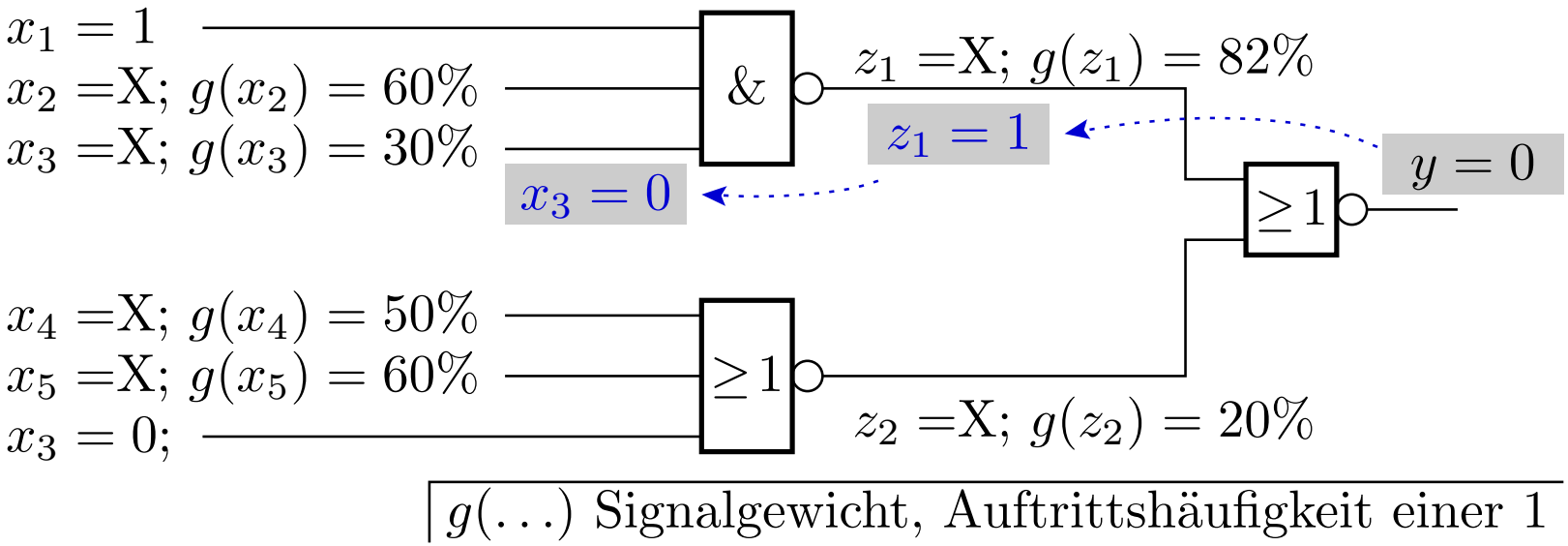
Suchraumbegrenzung

- Der D-Algorithmus baut den Suchbaum über alle mehrdeutigen Wertefestlegungen auf.
- Nur die Schaltungseingänge können unabhängig voneinander alle Wertevariationen annehmen.
- Es genügt, den Suchbaum mit den Eingabewertefestlegungen aufzubauen.
- Begrenzt Suchraum auf $2^{\#E}$ ($\#E$ – Eingangsanzahl). Verringert Rechenaufwand um Zehnerpotenzen.



- Lange Steuerpfade vom Fehlerort und vom D-Pfad zu Eingängen.
- Aufbau des Suchbaums über Eingangssignale.
- Wenn Implikationstest-Widerspruch, letzte Eingabefestlegung invertieren.

Geschätzte Erfolgswahrscheinlichkeiten



- Schätzen der Signalwichtungen[3] über eine kurze Simulation mit Zufallswerten oder analytisch.
- Wahl der Steuerwerte / Beobachtungspfade, die mit größerer Wahrscheinlichkeit aktivierbar / sensibilisierbar sind.

[3] Die Wichtung eines Signals ist die Auftrittshäufigkeit einer »1«.

Komplexe Funktionsbausteine

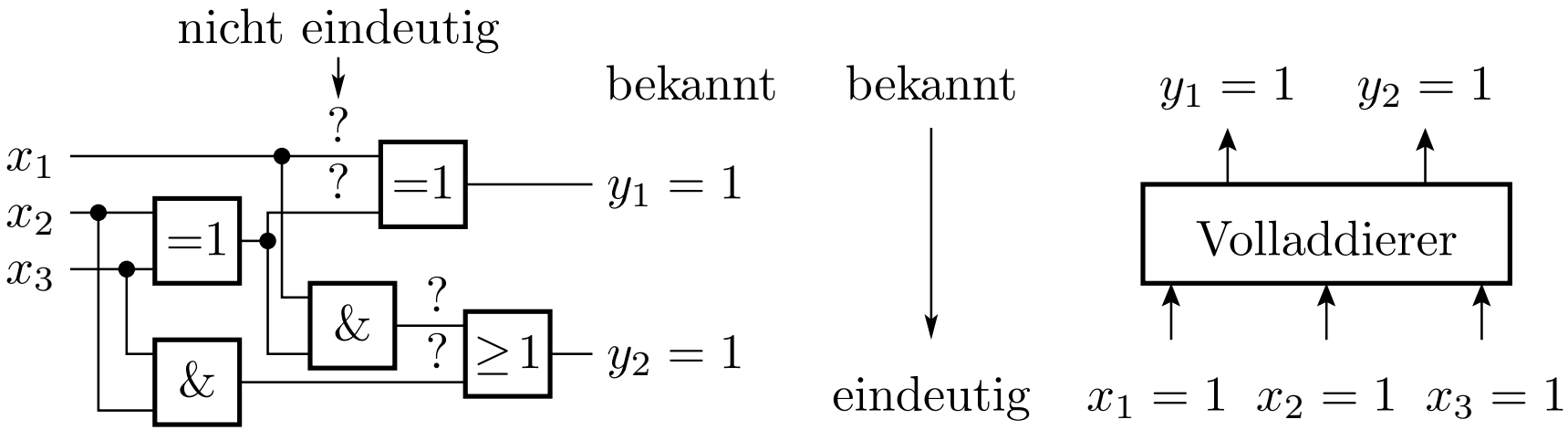
Komplexe Funktionsbausteine

- Beschreibung durch Tabellenfunktion (Bsp. Volladdierer):

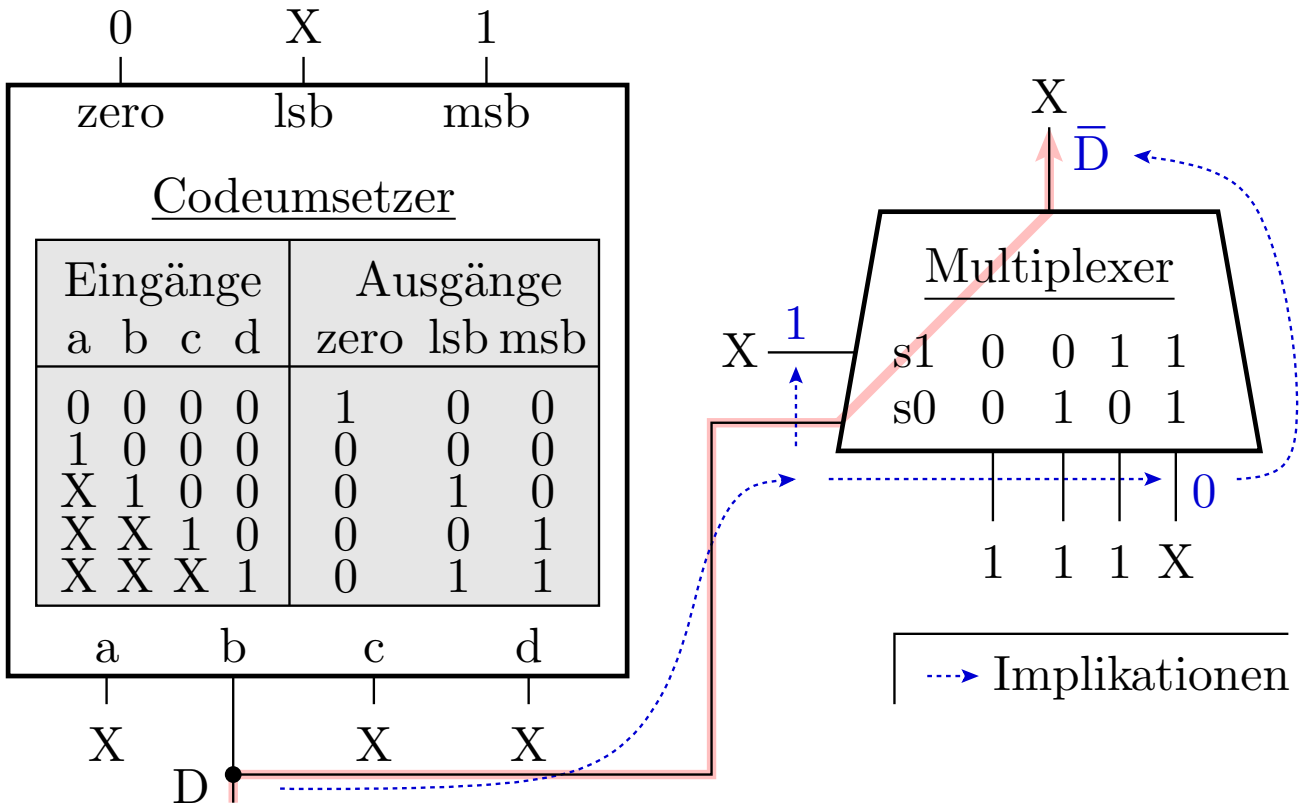
x_2	x_1	x_0	s	c	gegeben		Lösungsmenge
0	0	0	0	0	XXX00	$\Rightarrow$	00000
0	0	1	1	0			
0	1	0	1	0	01DXX	$\Rightarrow$	01D$\bar{\text{D}}$D
0	1	1	0	1			
1	0	0	1	0	1XXXD	$\Rightarrow$	10D$\bar{\text{D}}$D, 1D0$\bar{\text{D}}$D
1	0	1	0	1			
1	1	0	0	1	11XX1	$\Rightarrow$	11111, 111001
1	1	1	1	1			

- Vervollständigung des Vektors der gegebenen Anschlusswerte durch Vergleich mit allen Tabellenzeilen:
  - »1« und »0« passen nur auf »1« und »0«.
  - »X« passt immer.
  - »D« muss für »D=0« und für »D=1« passen.

Implikationstest an einem Volladdierer



- An der Gatterbeschreibung eines Volladdierers ist die Implikation $y_1 = y_2 = 1 \Rightarrow x_1 = x_2 = x_3 = 1$ nicht zu erkennen. Lösungsfindung über Baumsuche.
- Bei Zusammenfassung zu einer Tabellenfunktion wird die Lösung bereits bei der Anschlusswertevervollständigung erkannt.

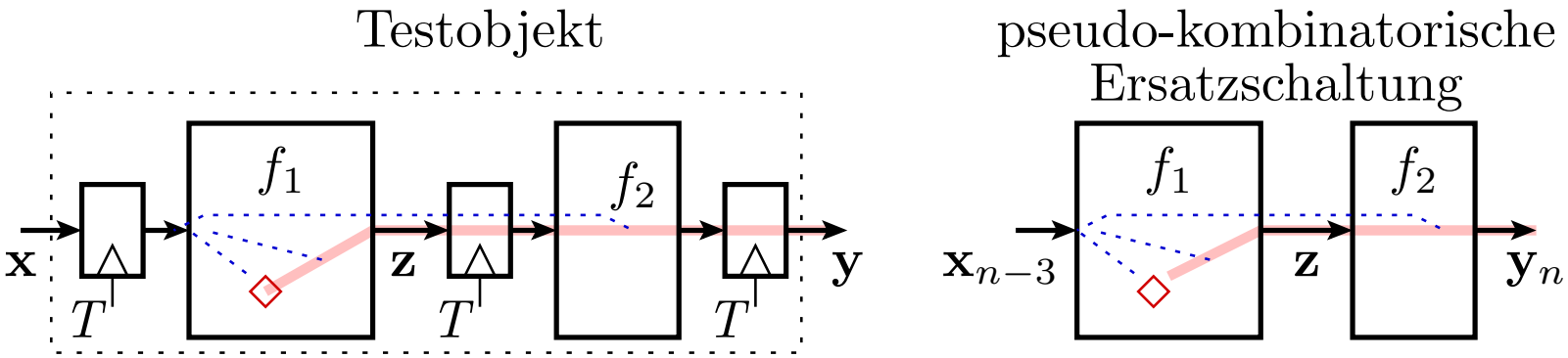


- »lsb« hängt bei »zero=0« und »msb=1« nicht von »b« ab. Eindeutiger D-Pfad über Multiplexer.
- Tabelleneingabewerte »X« (Eingang beeinflusst nicht die Ausgabe) führt zu Tabellen mit $\ll 2^{N_\mathrm{E}}$ Tabellenzeilen (N_E – Anzahl der Eingänge).

Sequentielle Schaltungen

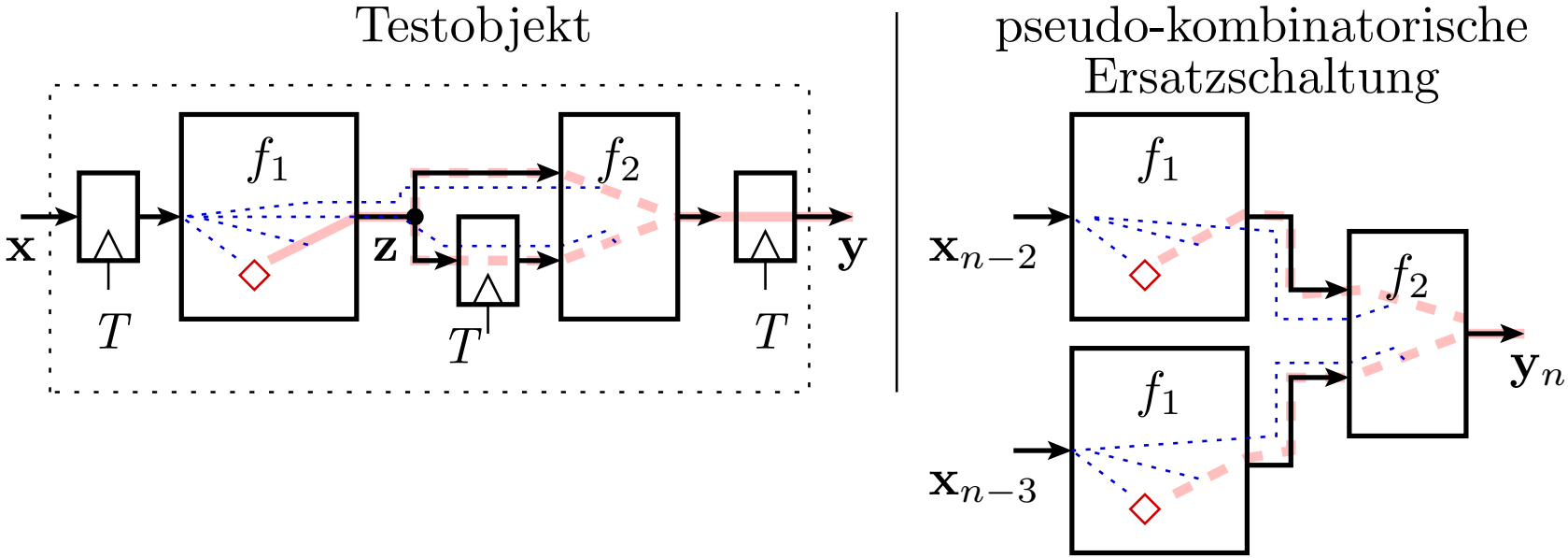
Pseudo-kombinatorische Ersatzschaltung

Schaltungen mit Speicherelementen werden für die Testsuche zu einer pseudo-kombinatorischen Ersatzschaltung aufgerollt. Abtastregister in einem geradlinigen Berechnungsfluss werden weggelassen:



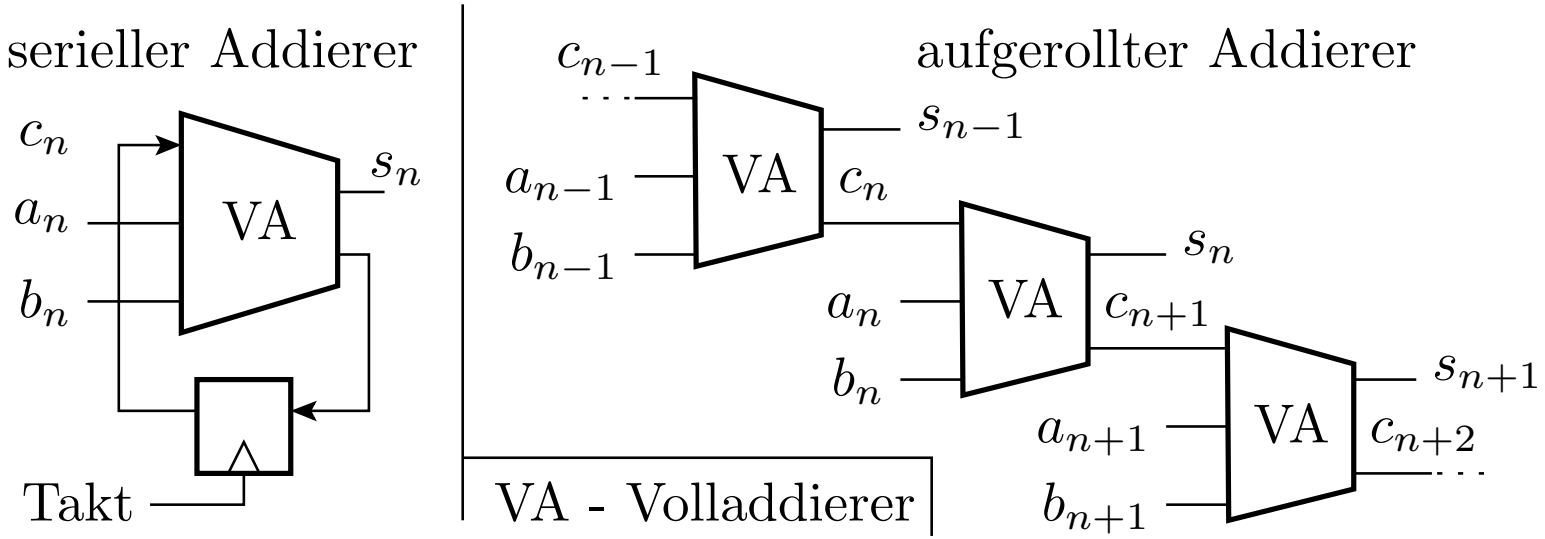
- Testberechnung wie für eine kombinatorische Schaltung.
- Die Verzögerung der Ausgabe gegenüber der Eingabe wird erst bei der Testdurchführung berücksichtigt.

Verarbeitung in mehreren Zeitebenen



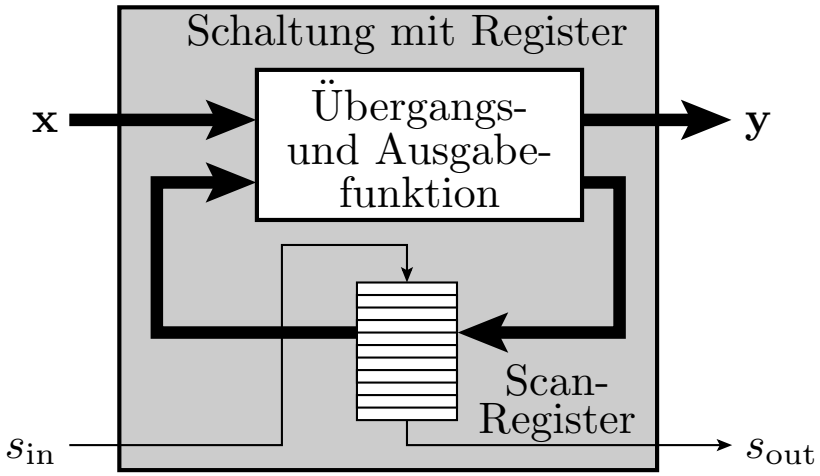
- Mehrere Kopien gleicher Schaltungsteile in der pseudo-kombinatorischen Ersatzschaltung.
- Der eingebaute Haftfehler ist in jeder Kopie der Teilschaltung.
- Berechnet wird eine Folge von Testeingaben für mehrere Zeitschritte (Mehr-Pattern-Test).

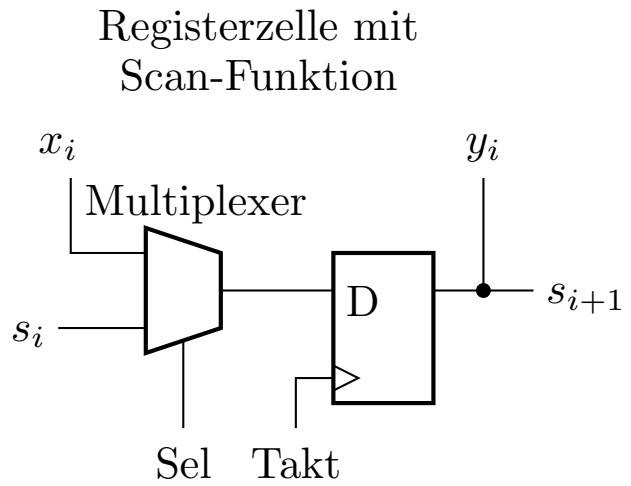
Schaltungen mit Rückführung



- Pseudo-kombinatorischen Ersatzschaltung mit endlos vielen Kopien der Übergangsfunktion.
- Längenbegrenzung der Steuer- und Beobachtungspfade.
- Alternative: Lese- und Schreibzugriff auf Zwischenergebnisse und -zustände, z.B. durch Verschalten der internen Speicherzeichen zu einem Scan-Register (vergl. Folie 69).

Scan-Verfahren





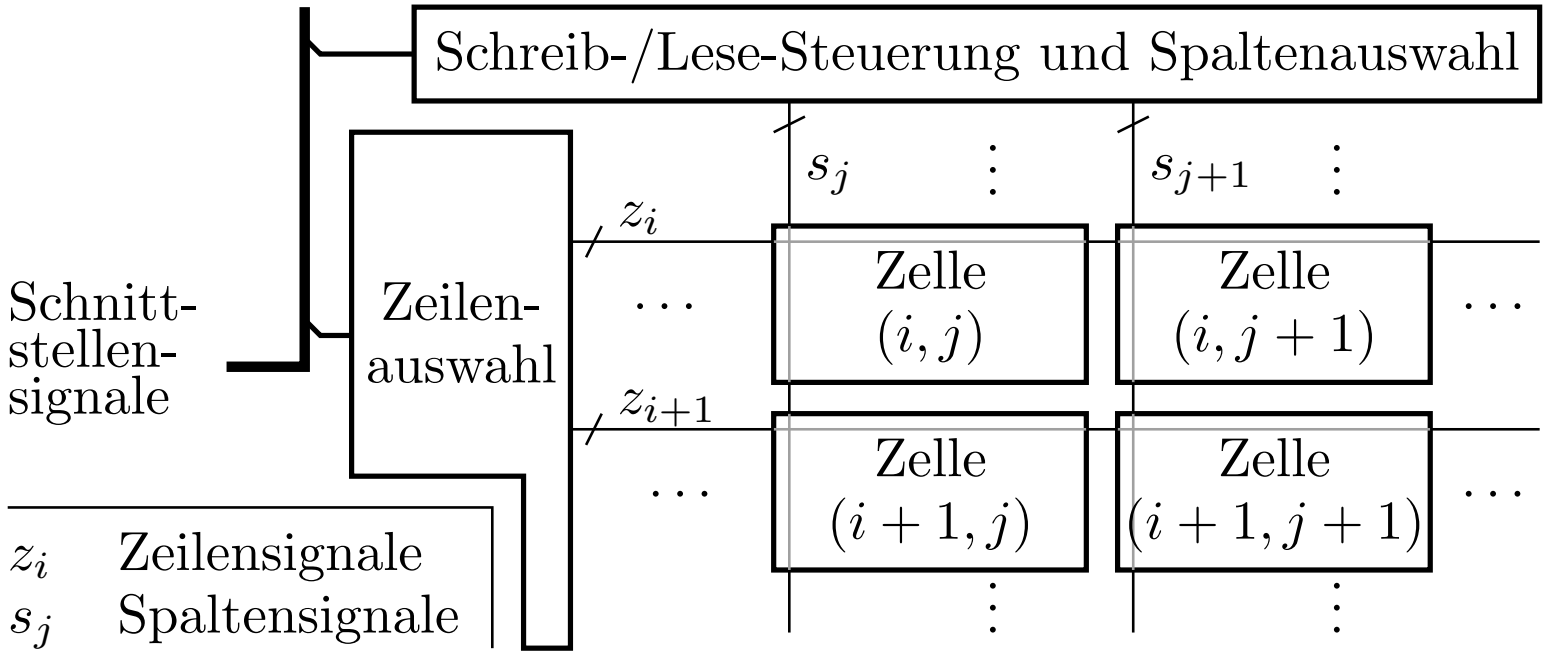
Lese- und Schreibzugriff während des Tests durch Umschalten des Zustandsspeicher in ein Schieberegister.

- Mindestschaltungsaufwand ein Multiplexer je Speicherzelle.
- Ablauf eines Testschritts: r Schiebeschritte zum Beschreiben des Zustandsspeichers, Testschritt, r Schiebeschritte zum Lesen (und Überschreiben) des Zustandsspeichers.

Speichertest

Blockspeicher

Große Speicher besteht im Kern aus einer regelmäßigen 2D-Anordnung von flächenminimierten Speicherzellen um geben von der Zeilen- und Spaltenauswahl. Die Grundfunktionen (nur lesbar, beschreib- und lesbar, ...) hängen von der Zellenfunktion ab.



Typische Fehlerannahmen für einen SRAM siehe nächste Folie.

beteiligte Zellen	Name	Definition	Fälle	Testfolge für den Nachweis
1	Haftfehler	Wert der Speicherzelle ist nicht setzbar	stuck-at-0 stuck-at-1	$W(i)1$, $R(i)1$ $W(i)0$, $R(i)0$
	Übergangsfehler	Wert der Speicherzelle i ist nur in einer Richtung änderbar	kein Übergang $1 \to 0$ $0 \to 1$	 $W(i)1$, $R(i)1$, $W(i)0$, $R(i)0$ $W(i)0$, $R(i)0$, $W(i)1$, $R(i)1$
	Stuck-open-Fehler	kein Zugriff auf Speicherzelle i (Ausgabe des Wertes der vorherigen Leseoperation)		$W(i)0$, $R_1(j)$, $R(i)0$, $W(i)1$, $R_0(j)$, $R(i)1$
	zerstörendes Lesen	Inhalt von Speicherzelle i wird beim Lesen verändert	$R(i) \Rightarrow C(i) = \overline{C(i)}$	$W(i)0$, $R(i)0$, $R(i)0$ $W(i)1$, $R(i)1$, $R(i)1$
2	Kopplung Typ 1	Veränderung des Inhalts von Zelle i bestimmt Zustand in Zelle j	$W(i)0 \Rightarrow C(j) = 0$ $W(i)0 \Rightarrow C(j) = 1$ $W(i)1 \Rightarrow C(j) = 0$ $W(i)1 \Rightarrow C(j) = 1$	$W(j)0$, $W(i)0$, $R(j)0$, $W(i)1$, $R(j)0$ $W(j)1$, $W(i)0$, $R(j)1$, $W(i)1$, $R(j)1$
	Kopplung Typ 2	Veränderung des Inhalts von Zelle i bewirkt eine Änderung in Zelle j	$C(i) = \overline{C(i)} \Rightarrow C(j) = \overline{C(j)}$	$W(j)0$, $W(i)0$, $R(j)0$, $W(i)1$, $R(j)0$, $W(i)0$, $R(j)0$ $W(j)1$, $W(i)0$, $R(j)1$, $W(i)1$, $R(j)1$, $W(i)0$, $R(j)1$

$W(i)0$ Schreibe in Zelle i eine 0
$W(i)1$ Schreibe in Zelle i eine 1
$R(j)$ Lese eine beliebige andere Zelle
$C(\ldots)$ Inhalt Zelle ...

$R(i)0$ Lese Inhalt Zelle i und vergleiche mit Sollwert 0
$R(i)1$ Lese Inhalt Zelle i und vergleiche mit Sollwert 1
$R_0(j)$ Lese eine andere Zelle, in der 0 steht
$R_1(j)$ Lese eine andere Zelle, in der 1 steht

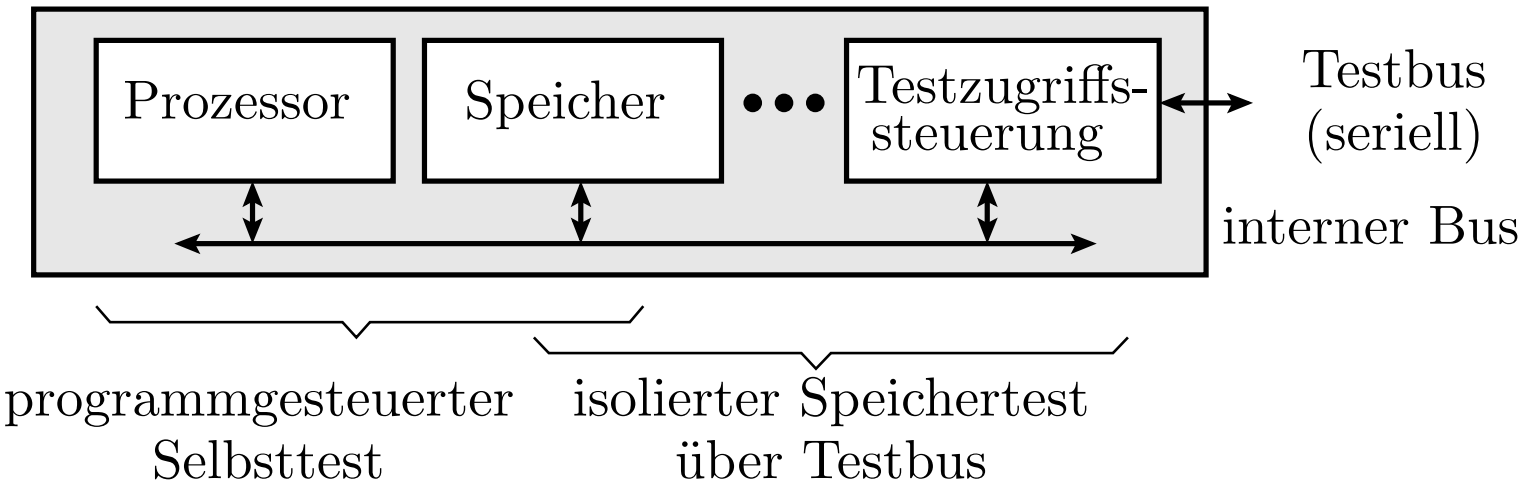
Beispiel Marching Test

Adresse i	Initialisierung	March 1	March 2	March 3
0	$W(i)0$	$R(i)0, W(i)1$	$R(i)1, W(i)0$	$R(i)0, W(i)1$
1	$W(i)0$	$R(i)0, W(i)1$	$R(i)1, W(i)0$	$R(i)0, W(i)1$
2	$W(i)0$	$R(i)0, W(i)1$	$R(i)1, W(i)0$	$R(i)0, W(i)1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$N-1$	$W(i)0$	$R(i)0, W(i)1$	$R(i)1, W(i)0$	$R(i)0, W(i)1$

	March 4		March 1a		March 2a
0	$R(i)1, W(i)0$	Wartezeit	$R(i)0, W(i)1$	Wartezeit	$R(i)1$
1	$R(i)1, W(i)0$		$R(i)0, W(i)1$		$R(i)1$
2	$R(i)1, W(i)0$		$R(i)0, W(i)1$		$R(i)1$
$\vdots$	$\vdots$		$\vdots$		$\vdots$
$N-1$	$R(i)1, W(i)0$		$R(i)0, W(i)1$		$R(i)1$

Mehrfaches Durchwandern des Speichers in unterschiedlicher Reihenfolge mit der Operationsfolge Zelle Lesen, Wert kontrollieren und inversen Wert zurückschreiben.

Test eingebetteter Blockspeicher



Eingebettete Blockspeicher werden vorzugsweise isoliert von ihrer Schaltungsumgebung getestet:

- über herausgezogenen Bussignale,
- über den Testbus oder
- programmgesteuert vom Prozessor als eingebauter Selbsttest (BIST – Built-In Self-Test).

Selbsttest

Selbsttest

Einbau der Testfunktionseinheiten mit in den Schaltkreis:

- Testmustergenerator: Pseudo-Zufallsgenerator, Zähler, Schieberegister.
- Testablaufsteuerung, in der Regel über Testbus.
- Ausgabekontrolle, vorzugsweise durch Bildung eines Prüfkennzeichens mit LFSR (Linear Feedback Shift Register).

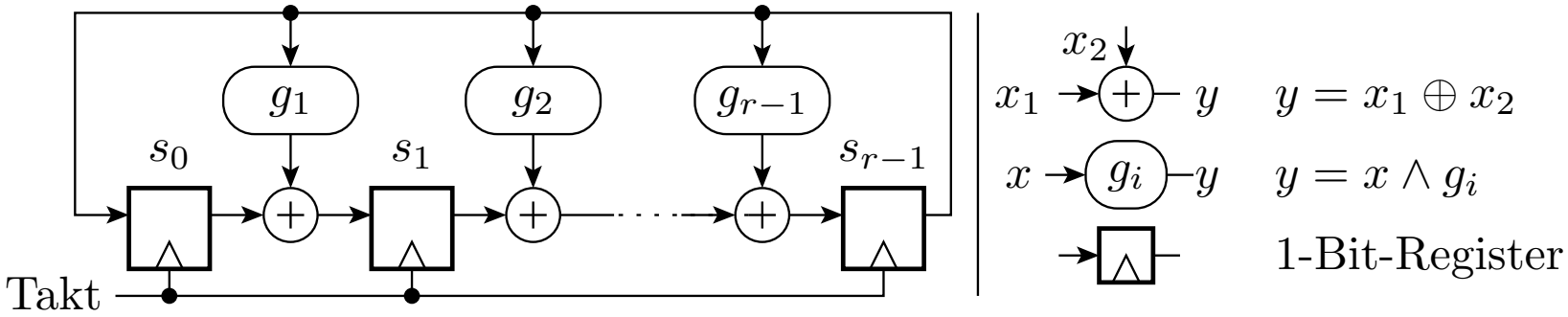
Vorteile:

- Erlaubt sehr große Testsatzlängen, Test mit voller Geschwindigkeit,
- nutzbar auch später im Zielsystem für den Einschalttest.

Pseudo-Zufallsregister

Linear rückgekoppelte Schieberegister

Ein linear rückgekoppelte Schieberegister (LFSR **L**inear **F**eeback **S**hift **R**egister) in einer ersten Ausführung verschiebt seinen r-Bit-Zustand $\mathbf{s} = (s_{r-1}, s_{r-2}, \ldots, s_0)$ um eine Stelle nach links und addiert, wenn das herausgeschobene Bit s_{r-1} gleich »1« ist, eine Bitvektorkonstante $\mathbf{g} = (g_{r-1}, g_{r-1}, \ldots, g_1, 1)$ zum Zustand **s**:



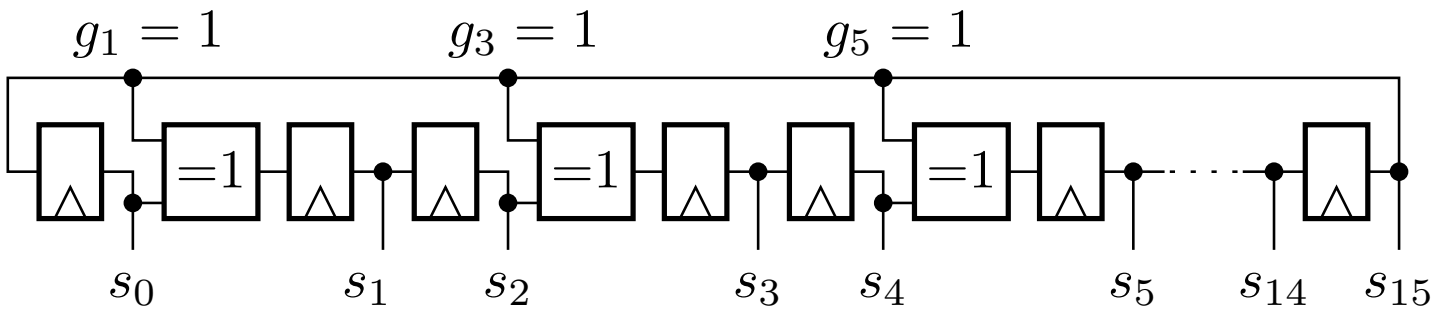
Für jede Bitanzahl r des Zustandsvektors gibt es Konstanten **g**, sog. »primitive Polynome«, bei denen alle Zustände außer 000...0 ineinander übergehen. Nur solche Konstanten **g** werden verwendet.

Primitiven Polynome und die Konstante g

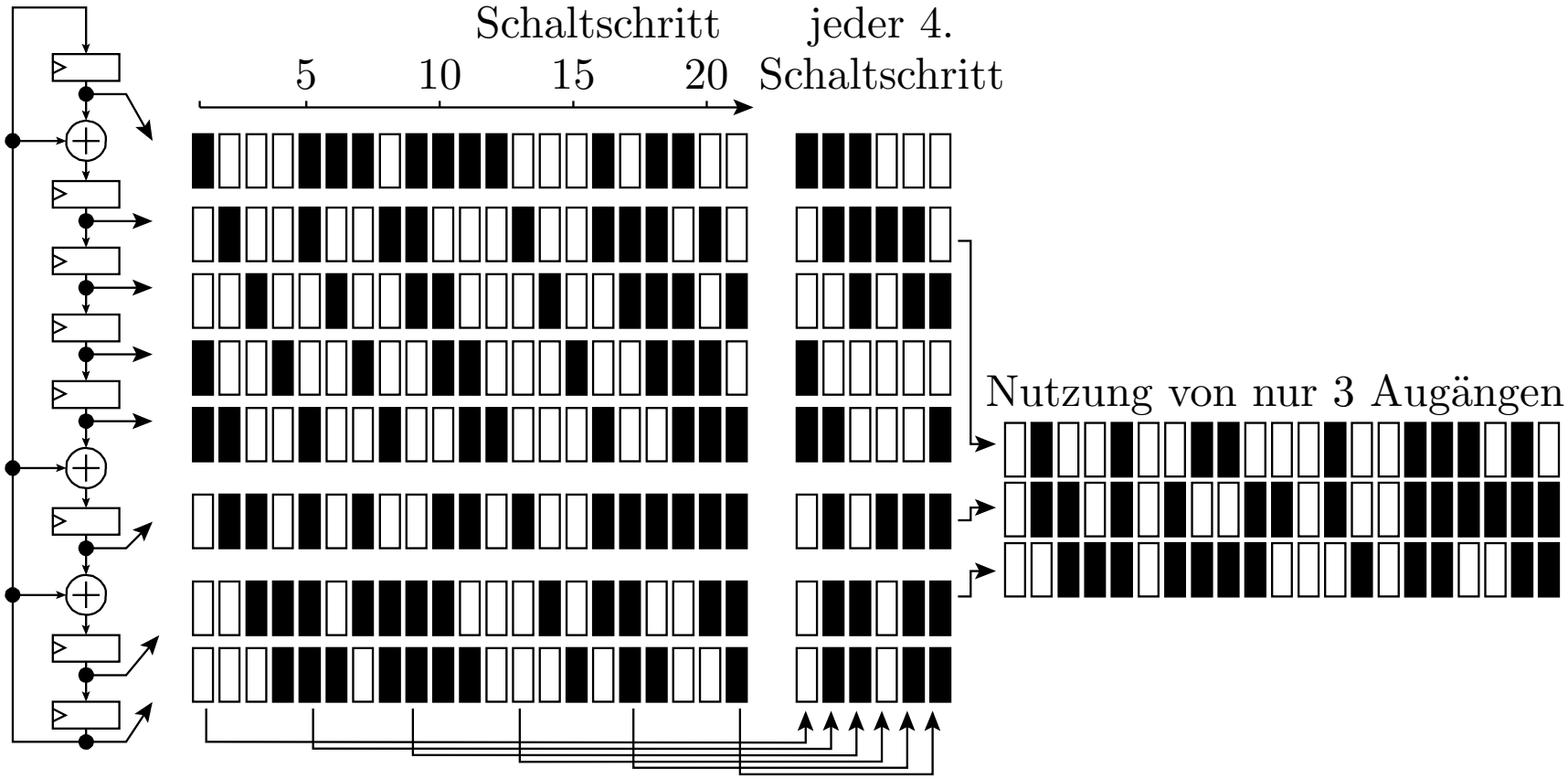
Mit dem Internet-Suchbegriff »Primitive Polynome« findet man z.B. für 16-Bit LFSR:

$$x^{16} \oplus x^5 \oplus x^3 \oplus x \oplus 1$$

Das bedeutet $g_1 = g_3 = g_5 = 1$ und alle anderen $g_i|_{i \notin \{1,3,5\}} = 0$. In Realisierung als Digitalschaltung für $g_i = 1$ EXOR-Gatter einfügen und für $g_i = 0$ EXOR-Gatter weglassen.

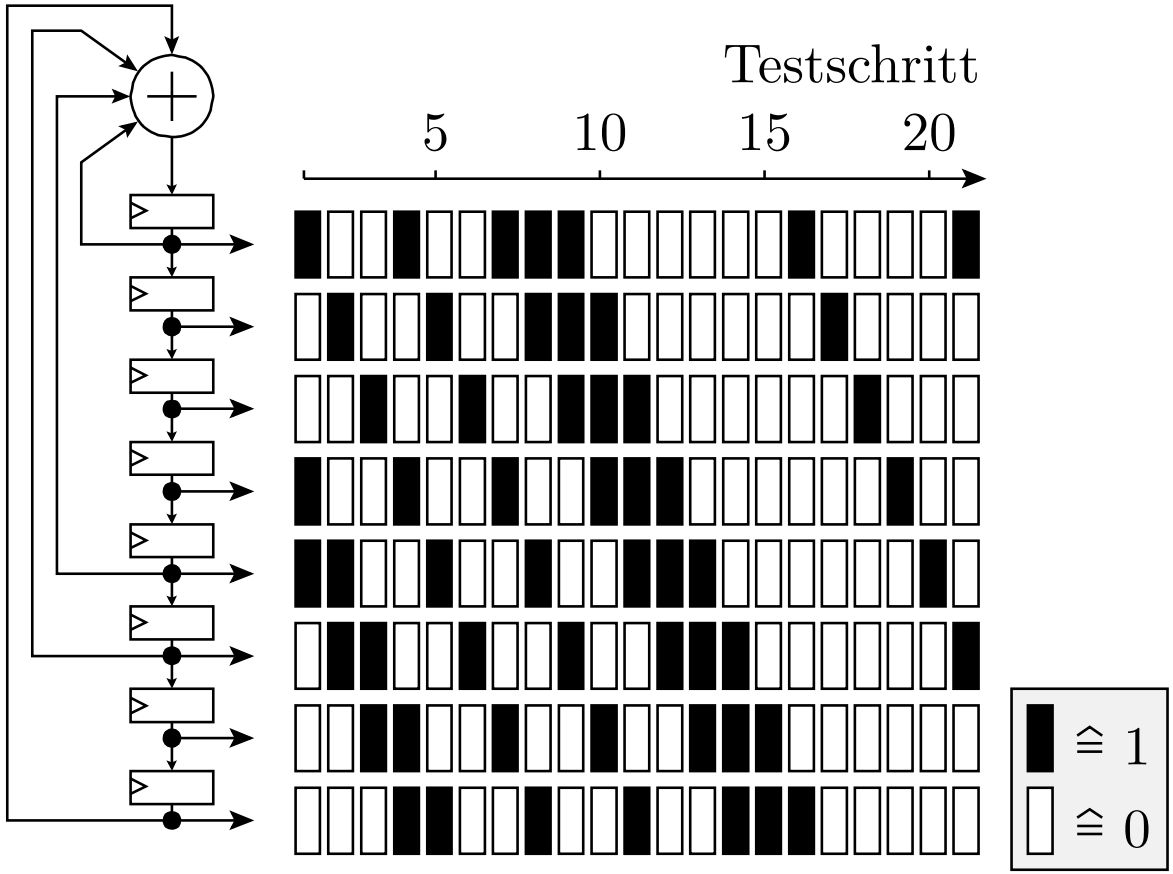


Pseudo-Zufallsfolge eines 8-Bit-LFSR



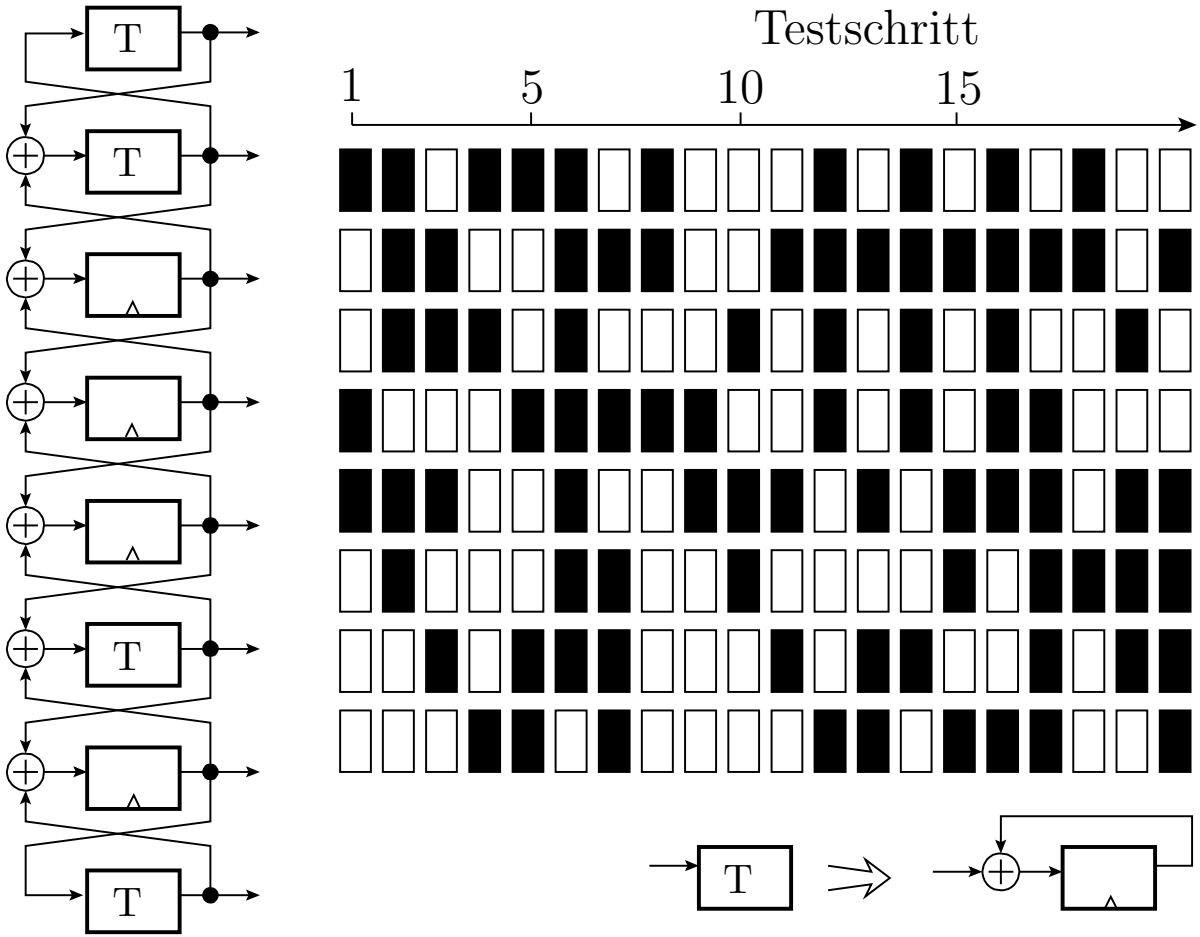
Falls die »Streifenmuster« durch die Schiebeoperationen stören, nur einen Teil der Ausgaben nutzen.

Bei »Umkehrung« der Signalflussrichtung wird aus den verteilten EXOR-Gattern ein zentrales EXOR-Netzwerk am Eingang.



Gleiche Zyklusstruktur bei gleichen Rückführstellen. Bitfolgen mit Phasenverschiebung größer 1 auch durch EXOR mehrerer Bitströme.

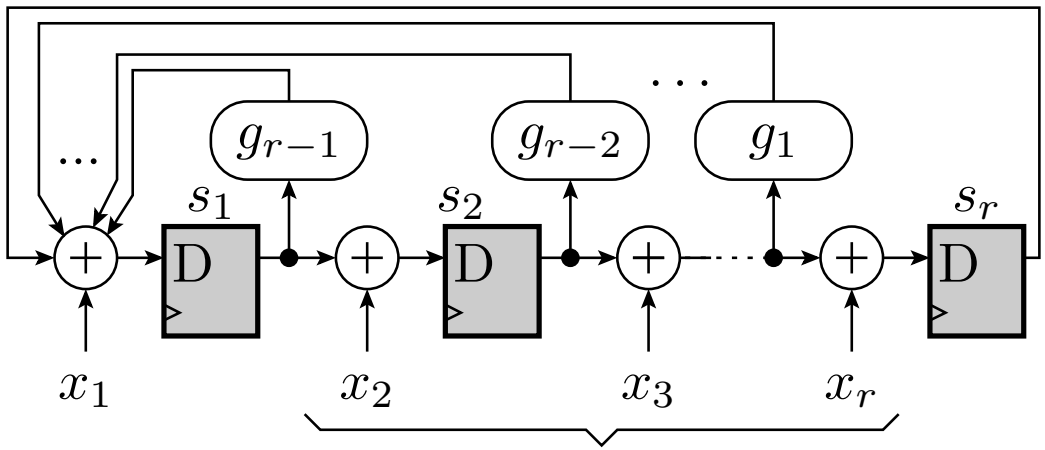
Es gibt viele weitere lineare Automaten, die auch zyklisch Bitfolgen in zufälliger Reihenfolge erzeugen. Beispiel Zellenautomaten, bei denen jedes Folgebit aus dem eigenen und den Zuständen der Nachbarbits gebildet wird:



Signaturregister

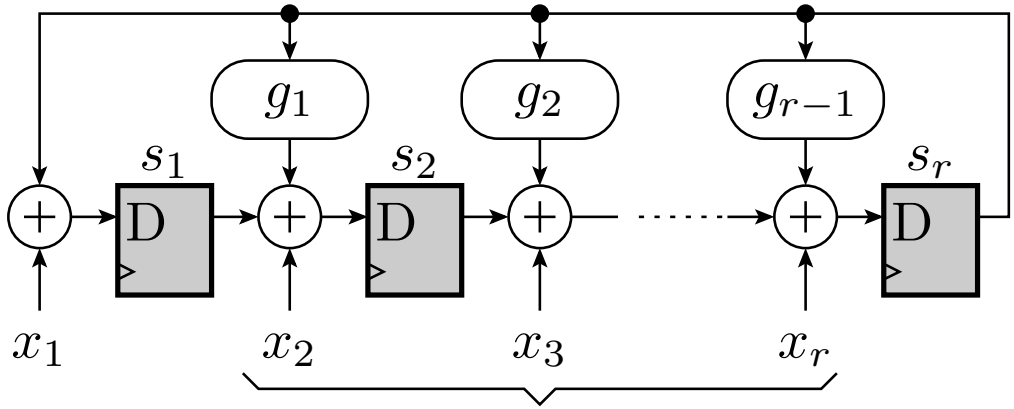
LFSR für parallele Datenströme

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



Erweiterungsmöglichkeit auf mehrere Eingänge

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



Erweiterungsmöglichkeit auf mehrere Eingänge

Die Koeffizienten g_i der Rückführung, bei der Polynom-Division das Divisor-Polynom, bestimmen die autonome Zyklusstruktur[4]. 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.

[4] Zyklusstruktur ohne Eingaben.

Experiment Fehlererkennungssicherheit von LFSR

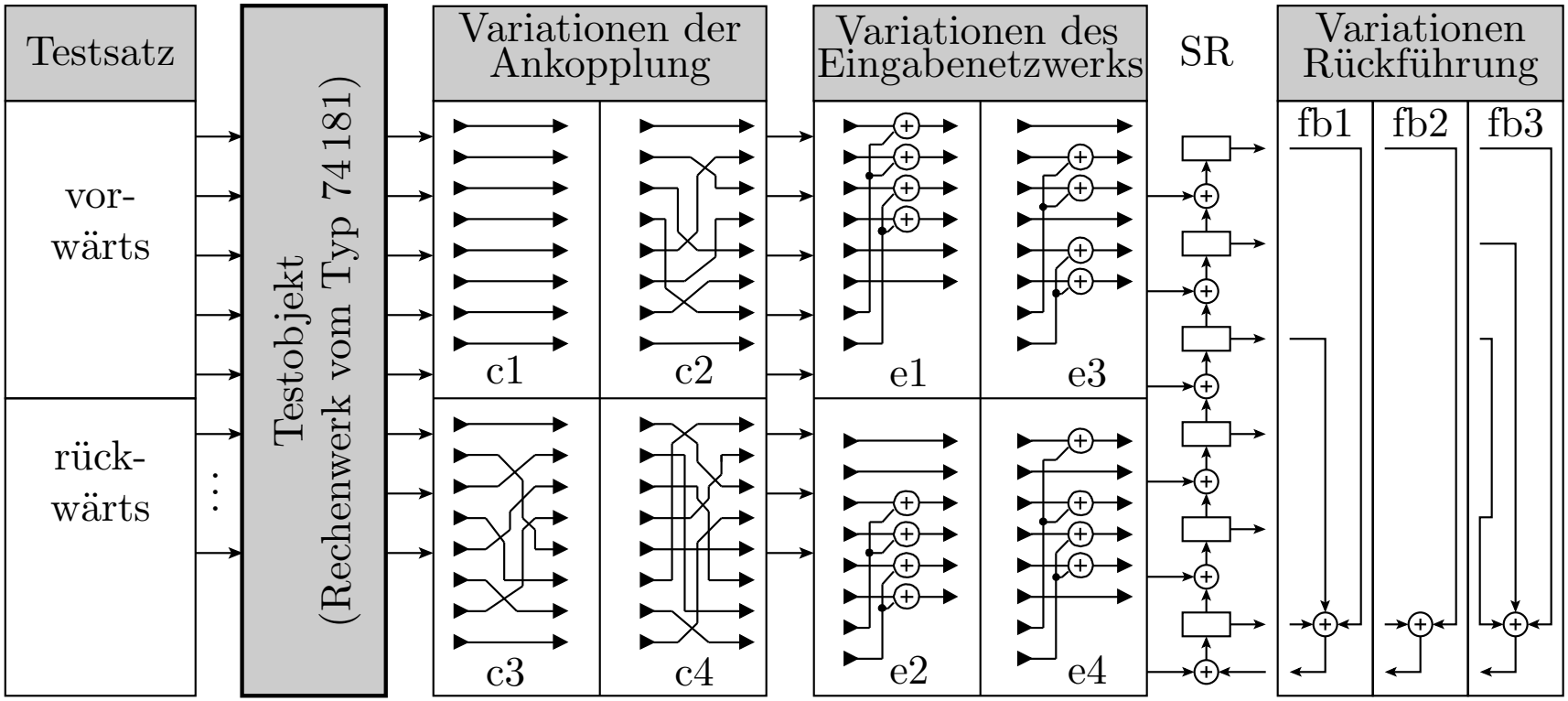
Es ist schwer zu glauben, dass

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

Deshalb ein Experiment:

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

Zählen der nachweisbaren Fehler für jede Konfiguration.



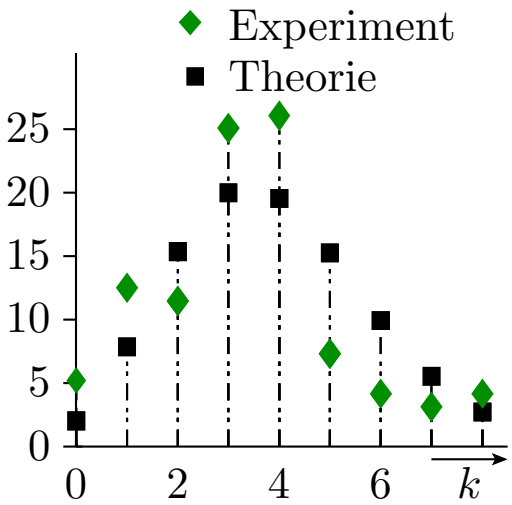
Aus $r = 6$ bit folgt, dass jeder Fehler mit einer Wahrscheinlichkeit $p_{\mathrm{E}} = 1 - 2^{-6} = 98{,}44\%$ erkenn- und mit einer Wahrscheinlichkeit $p_{\mathrm{F}} = 2^{-6} = 1{,}36\%$ nicht erkennbar sein müsste. Wenn das stimmt, müsste die Anzahl der maskierten Fehler poisson-verteilt sein:

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

mit Erwartungswert $\lambda = 250 \cdot p_{\mathrm{F}} = 3{,}9$ (vergl. Foliensatz 3, Aschn. 2.2).

Experimentell bestimme Anzahl der Maskierungen:

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



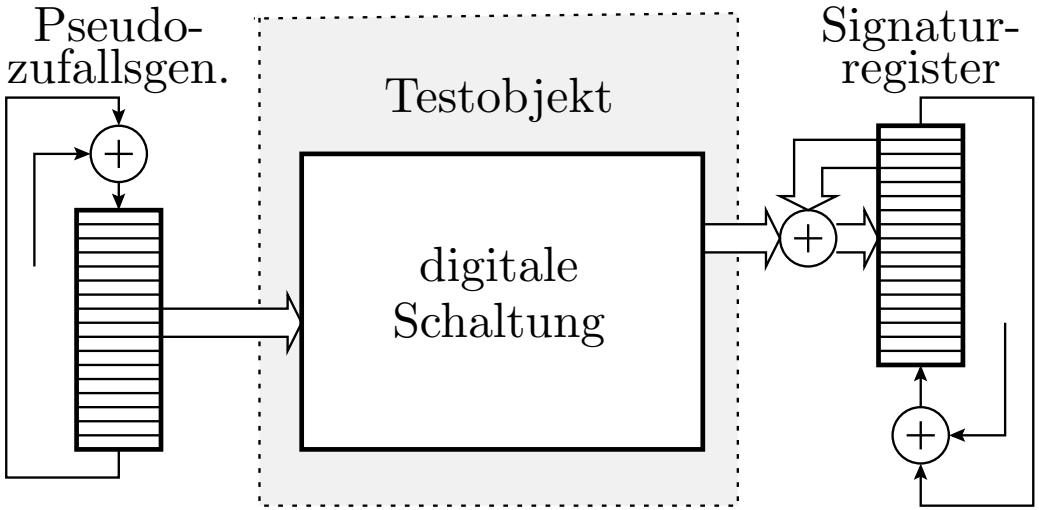
Vergleich experimentelle Ergebnisse mit zu erwartenden Zählwerten:

k	0	1	2	3	4	5	6	7	8
$96 \cdot e^{-\lambda} \cdot \frac{\lambda^k}{k!}$	1,3	7,5	14,7	19,2	18,73	14,6	9,5	5,3	2,6
Experiment	5	12	11	25	26	6	4	3	4

- Keine signifikante Abweichung von erwarteter Possonverteilung.
- Kein erkennbarer dominanter Einfluss der Rückkopplung auf die Anzahl der Maskierungenen.

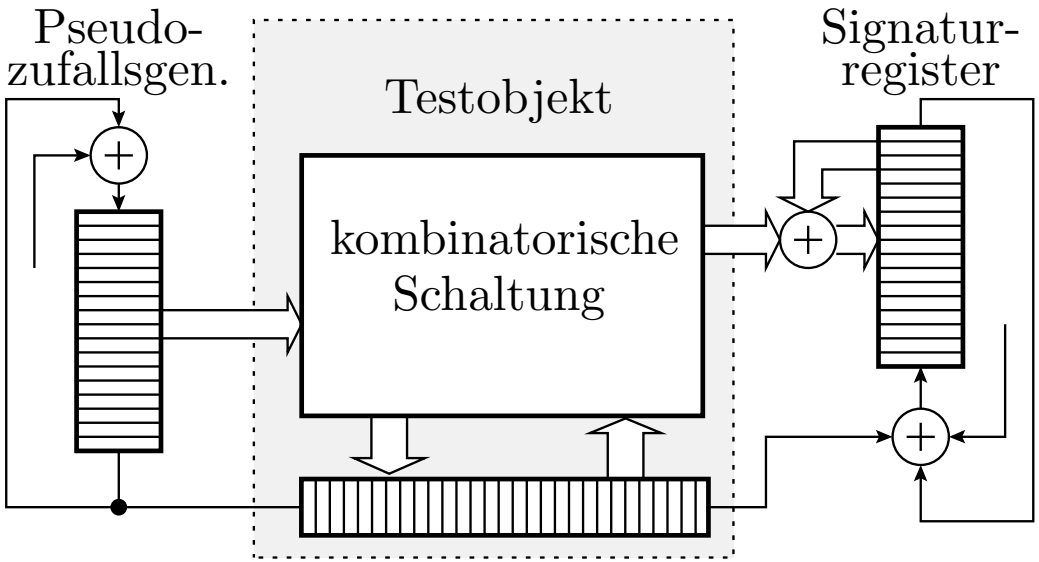
Selbsttest mit LFSR

Selbsttest (BIST **B**uilt-in **S**elf **T**est)



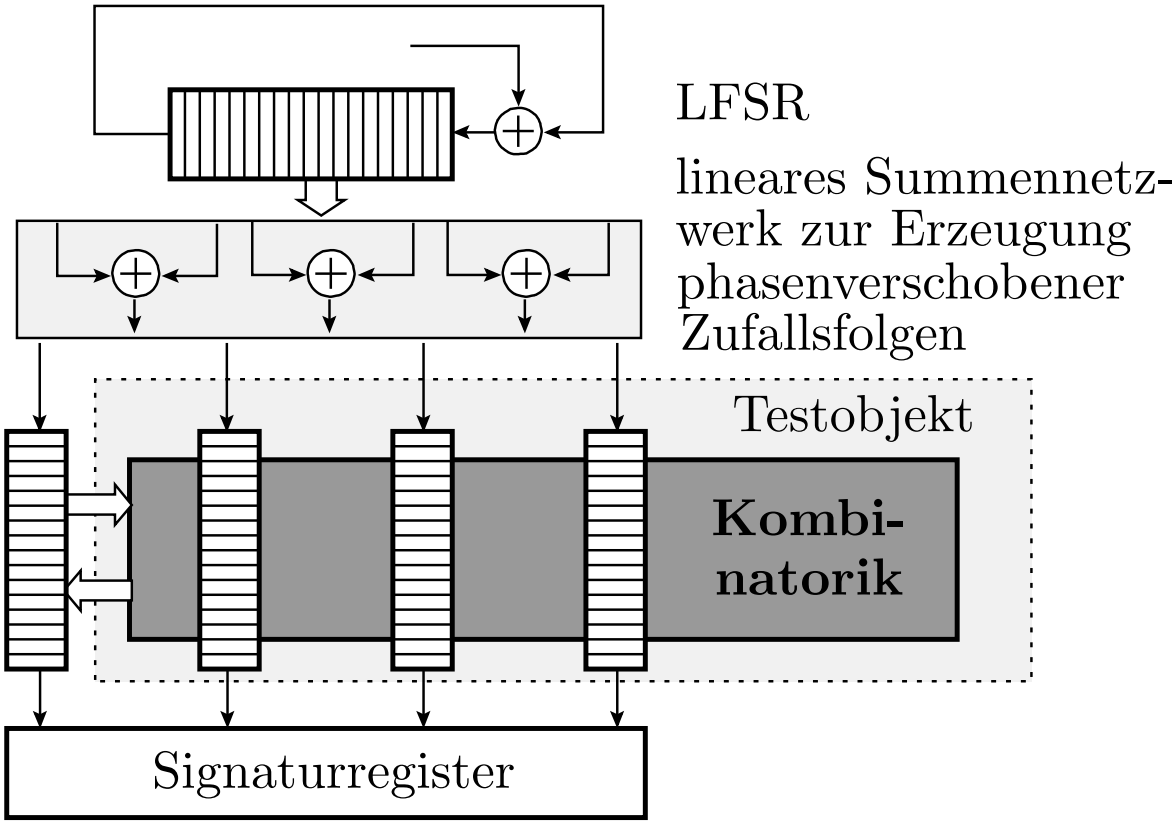
- Einrahmen der Schaltung mit Schieberegistern und Ergänzung einiger EXOR-Gatter.
- Wenn als Schieberegister vorhandene Ein- und Ausgaberegister verwendet werden, besonders niedriger Zusatzaufwand.
- Test mit voller Schaltungsgeschwindigkeit von Millionen bis Milliarden Test pro Sekunde.

BIST plus Scan



- Zwischen den Testschritten Zustandsregister seriell in das Signaturregister auslesen und neu beschreiben.
- Der isolierte Test der Übergangsfunktion reduziert in die Regel die erforderliche Testzeit viel mehr, als sie sich durch die zusätzlichen Schiebeschritte erhöht.

Für sehr große Systeme, z.B. Multi-Chip-Module mit mehreren Scan-Registern, die zwischen den Testschritten parallel gelesen und mit neuen Zufallswerten beschrieben werden.



Weiterführende Literatur [G. Kemnitz: Test und Verlässlichkeit von Rechnern. Springer 2007.]

Fehlerorientierte Wichtung

Fehlerorientierte Wichtung

Fehlerorientiert gesuchte Tests verlangen einen entsprechend großen Speicher, für Selbsttest ungeeignet. Alternative fehlerorientierte Wichtung.

Die Wichtung $g(x)$ eines binären Signal x ist die Auftrittswahrscheinlichkeit einer eins:

$$g(x) = \mathbb{P}[x = 1]$$

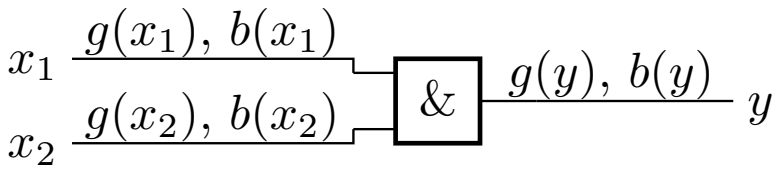
In einem System, in dem logische Werte UND, ODER, ... verknüpft werden, sind die Wahrscheinlichkeiten

- der Steuer- und Beobachtbarkeit und
- des Fehlernachweises

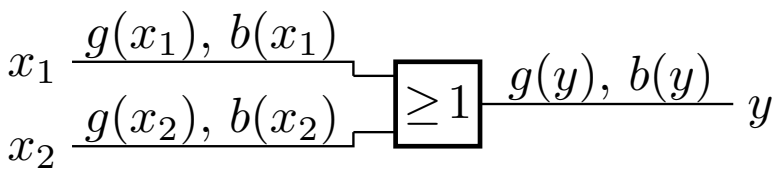
Funktionen der Wichtungen an den Eingängen und damit über Eingabewichtungen einstellbar.

Berechnung der Wichtungen und Beobachtbarkeiten

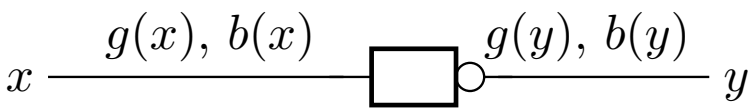
Die Wichtung einer UND-Verknüpfung ist das Produkt der Wichtungen der Operanden. ... Die Eingabe einer UND-Operation ist beobachtbar, wenn die andere Eingabe eins, bei einer ODER-Operation, wenn die andere Eingabe null ist. ...



$$b(x_2) = b(y) \cdot g(x_1)$$
$$b(x_1) = b(y) \cdot g(x_2)$$
$$g(y) = g(x_1) \cdot g(x_2)$$

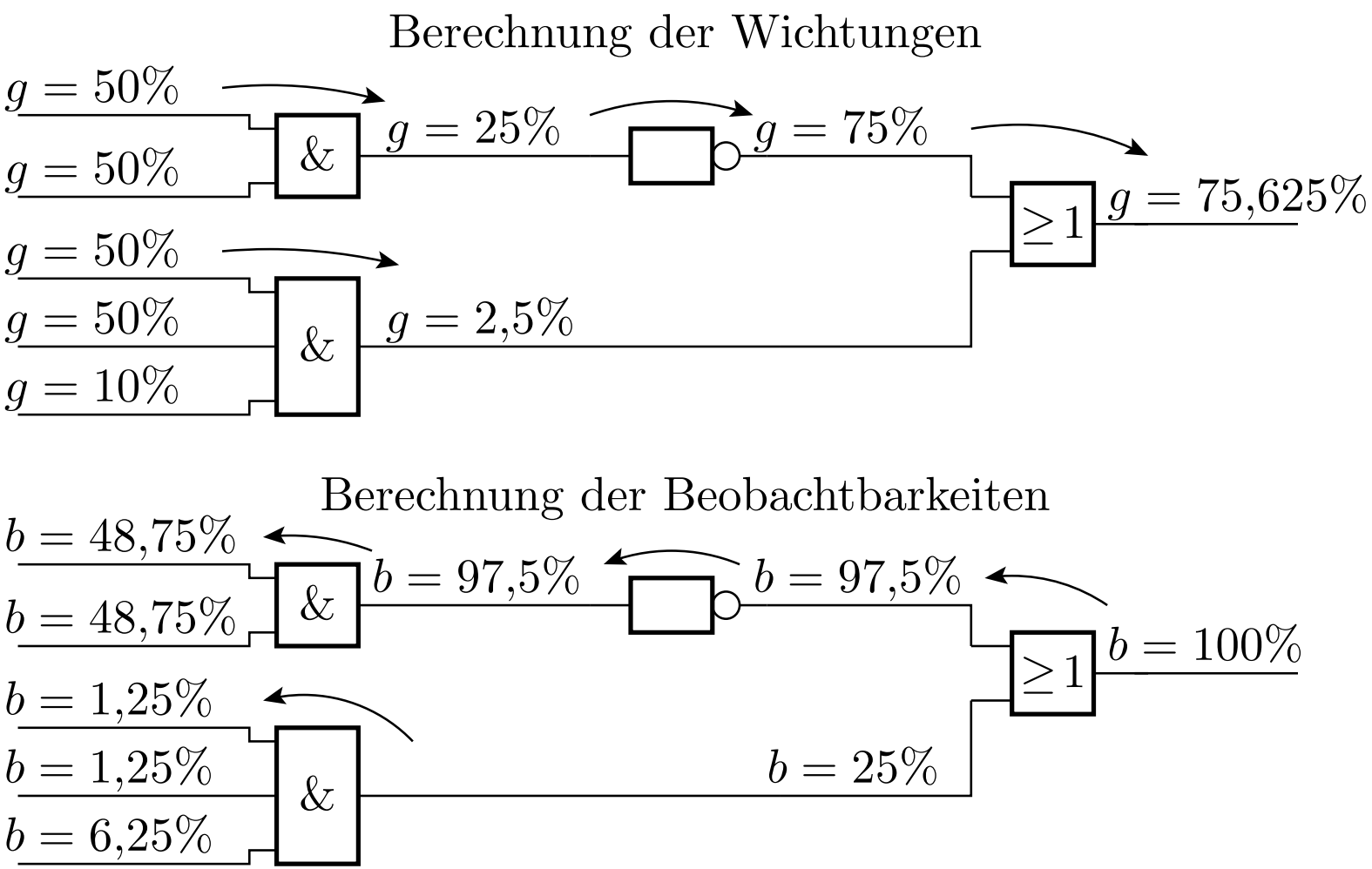


$$b(x_1) = b(y) \cdot (1 - g(x_2))$$
$$b(x_2) = b(y) \cdot (1 - g(x_1))$$
$$g(y) = 1 - (1 - g(x_1)) \cdot (1 - g(x_2))$$



$$b(x) = b(y)$$
$$g(y) = (1 - g(x))$$

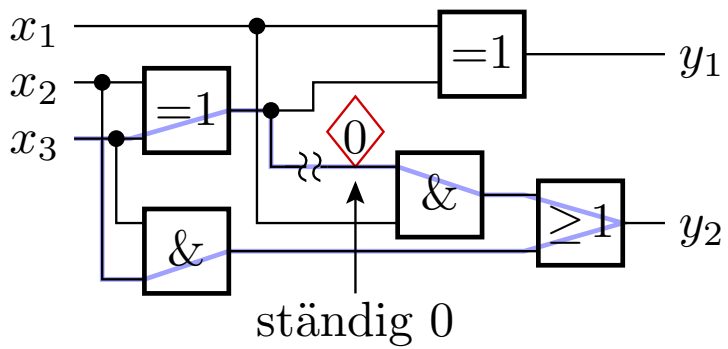
Wichtungen werden in Richtung und Beobachtbarkeiten entgegen der Richtung des Berechnungsflusses bestimmt.



Die Anregungswahrscheinlichkeit eines sa0-Fehlers ist die Wichtung $g(...)$ und die eines sa1-Fehler die Gegenwahrscheinlichkeit $1 - g(...)$ am Fehlerort. Die Nachweiswahrscheinlichkeit ist das Produkt aus Anregungs- und Beobachtungswahrscheinlichkeit.

Rekonvergente Auffächerung

Bei rekonvergenter Auffächerung werden gleiche Zufallswerte nach unterschiedlicher Verknüpfung mit anderen Werten logisch verknüpft. Für verknüpfte Werte, die von derselben Zufallsgröße abhängen, gelten die einfachen Regeln auf Folie 95 nicht. Berechnung aus den Auftrittshäufigkeiten der Eingaben für den Fehlernachweis:

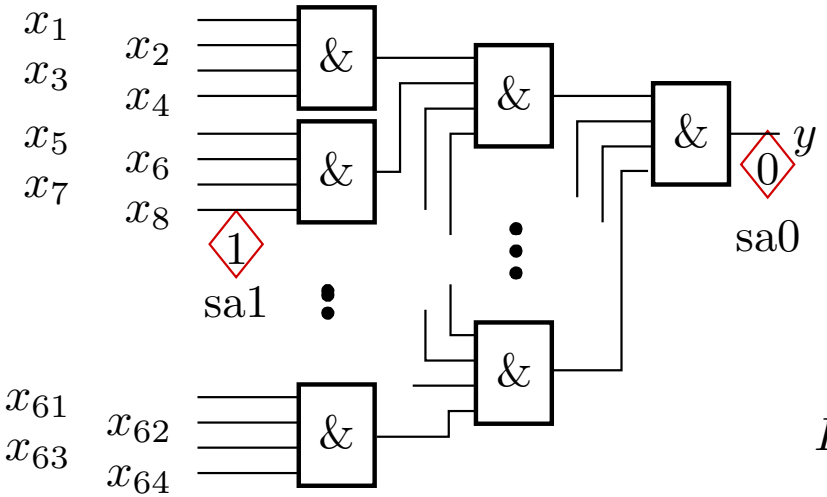


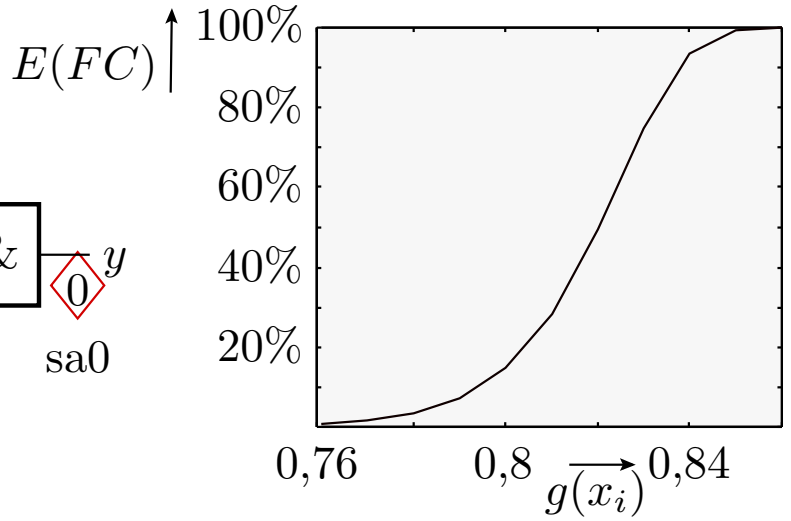
rekonvergente Auffächerung

Eingaben die den Fehler nachweisen

Eingabe x_3	x_2	x_1	Ausgabe y_2	y_1	Auftrittshäufigkeit		
0	0	0	0	0	0,125	0,1	0,1
0	0	1	0	1	0,125	0,05	0,1
0	1	0	0	1	0,125	0,15	0,2
0	1	1	1	0	0,125	0,2	0,05
1	0	0	0	1	0,125	0,05	0,2
1	0	1	1	0	0,125	0,2	0,05
1	1	0	1	0	0,125	0,05	0,2
1	1	1	1	1	0,125	0,2	0,1
Nachweiswahrscheinlichkeit:					0,25	0,4	0,1

Fehlerüberdeckung und Wichtung





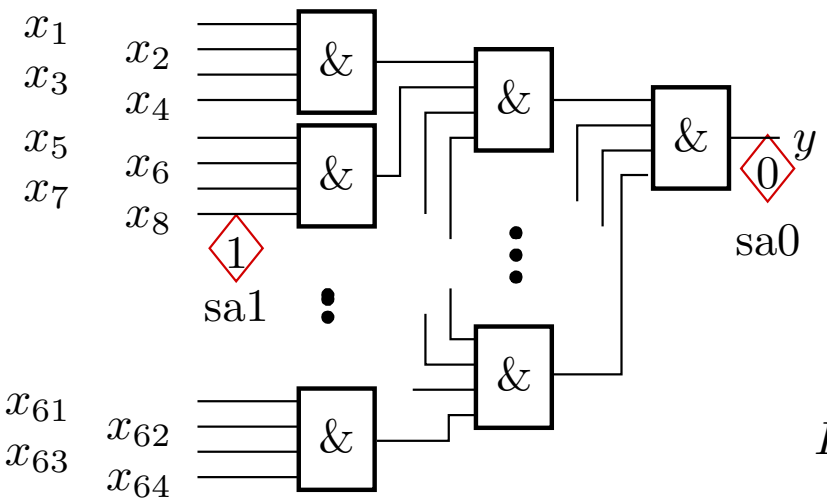
$E(FC)$ Fehlerüberdeckung für $n = 10^6$ Testschritte

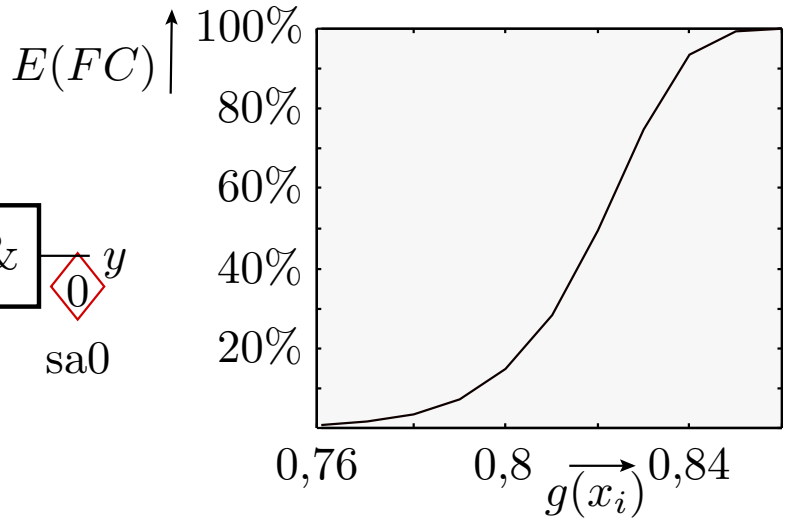
Angenommene Fehler: Für je einen der 64 Eingänge ständig 1:

$$p_{\mathrm{sa1}} = g^{63} \cdot (1 - g)$$

Für den Ausgang ständig 0:

$$p_{\mathrm{sa0}} = g^{64}$$





$E(FC)$ Fehlerüberdeckung für $n = 10^6$ Testschritte

Zu erwartende Fehlerüberdeckung als Mittelwert der Nachweiswahrscheinlichkeit:

$$FC = 1 - \frac{64 \cdot \mathrm{e}^{-g^{63} \cdot (1-g) \cdot n} + \mathrm{e}^{-g^{64} \cdot (1-g) \cdot n}}{65}$$

Eine Wichtung von $g = 86\%$ verringert die erforderliche Testsatzlänge für $FC \geq 99\%$ von $n \gg 2^{64}$ auf $n \approx 10^6$.

Fehlerorientierte Wichtungsauswahl

Statt einheitlicher Wichtung aller Eingabesignale

- Festlegung einer individuellen Wichtung für jeden Eingang,
- Wichtungsumschaltung jeweils nach einem längeren Zufallstest.

Ein pragmatischer Ansatz dazu aus[5]:

1. Festlegung einer größeren Menge von Modellfehlern.
2. Längerer Test mit ungewichteten Zufallswerten und Abhaken aller damit nachweisbaren Modellfehler.
3. Suche für die restlichen Modellfehler eine Eingabewichtung, die deren Nachweiswahrscheinlichkeiten erheblich erhöht.
4. Längerer Test mit den so gewichteten Zufallswerten und Abhaken aller damit nachweisbaren Modellfehler.
5. Wenn erforderlich, Wiederholung von Schritt 3 und 4.

[5] J. Hartmann, G. Kemnitz: How to do weighted random testing for BIST? ICCAD 1993.

Auswahl der Wichtungswerte

- Für alle Modellfehler, die der ungewichtete Zufallstest nicht nachweist, gezielte Berechnung einer Eingabe (D-Algorithmus) mit möglichst vielen »X« (Don't Care) Eingabewerten.
- Begrenzung der Wichtung auf 5 Werte:

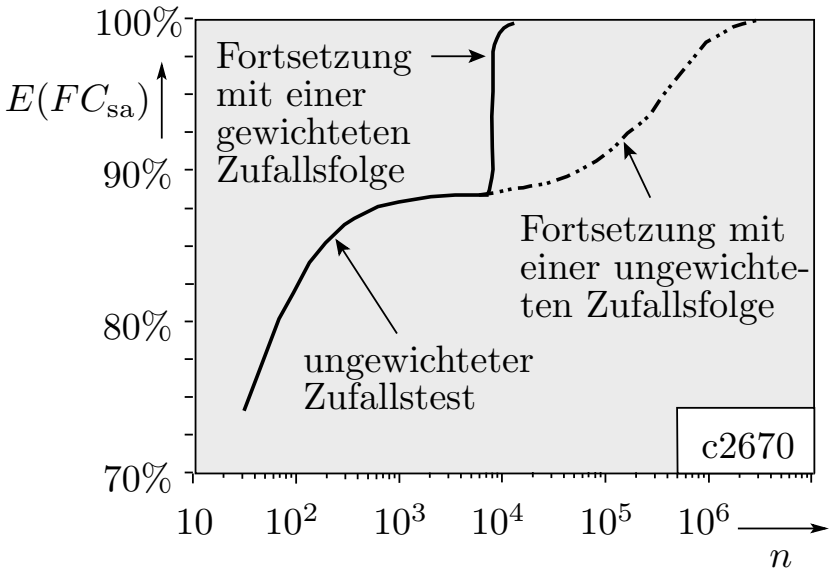
$$g(x_i) = \begin{cases} 0 & x_i \in \{0, \mathrm{X}\} \\ 2^{-\#E_{\mathrm{AND}}} & \#N \gg \#E \\ 0{,}5 & \text{sonst} \\ 1 - 2^{-\#E_{\mathrm{AND}}} & \#N \ll \#E \\ 1 & x_i \in \{1, \mathrm{X}\} \end{cases}$$

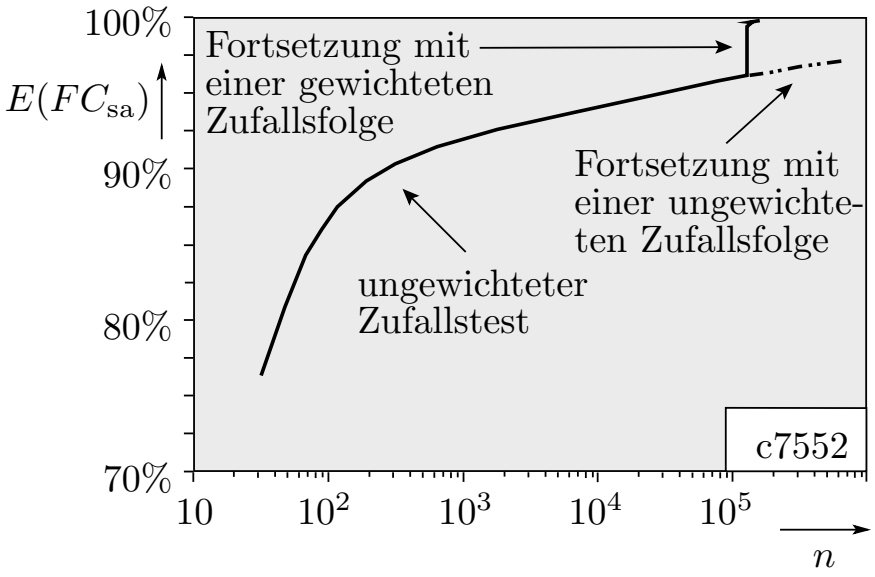
($\#N$ – Anzahl der Nullen; $\#E$ – Anzahl der Einsen für Eingang x_i in den Testeingaben; $\#E_{\mathrm{AND}} \in \{2, 3, 4, \ldots\}$ Anzahl der [N]AND-verknüpften Zufallsfolgen zur Wichtungserzeugung).

- Die zweite und weitere Wichtungsberechnungen berücksichtigen nur noch die Testeingaben bis dahin nicht nachgewieser Fehler.

Experiment mit den Schaltungen c2670 und c7552[6]

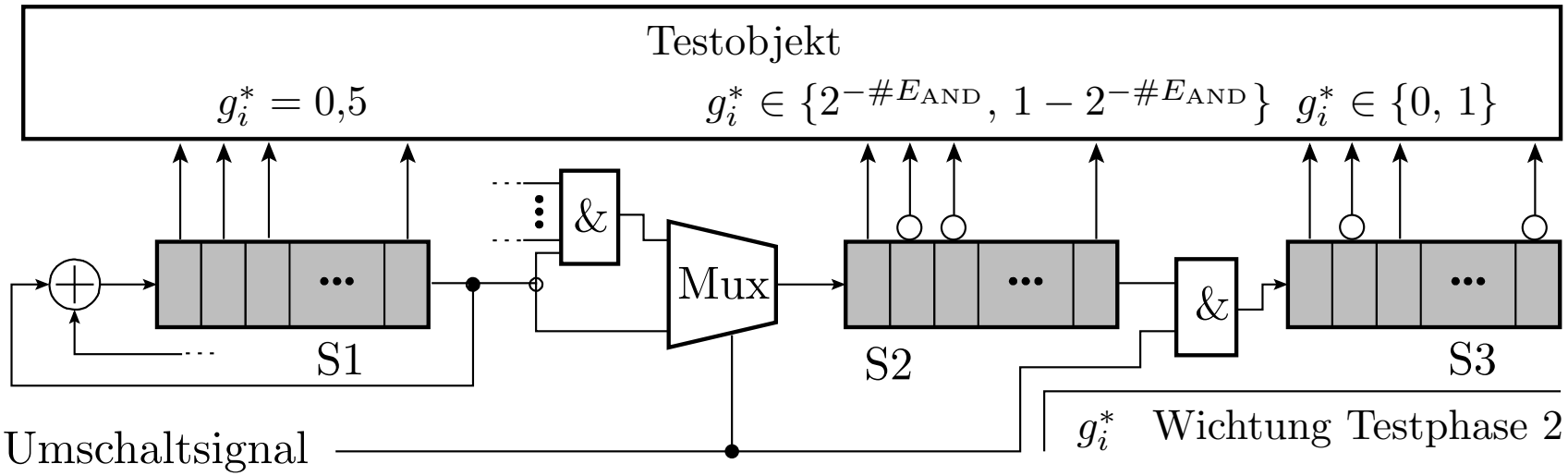
- Test mit 10^4 bzw. 10^5 ungewichteten Zufallsmustern, die 90% bzw. 95% der Haftfehler nachweisen.
- Gezielte Testberechnung für die restlichen Haftfehler.
- Individuelle Wichtung aller Eingabebits zur Maximierung der mittleren Auftrittshäufigkeit der berechneten Testeingaben.





[6] Kombinatorische Benchmarkschaltungen zum Vergleich von Testlösungen. Die Zahl hinter dem »c« ist die Anzahl der Signalleitungen.

Implementierung als Selbsttest

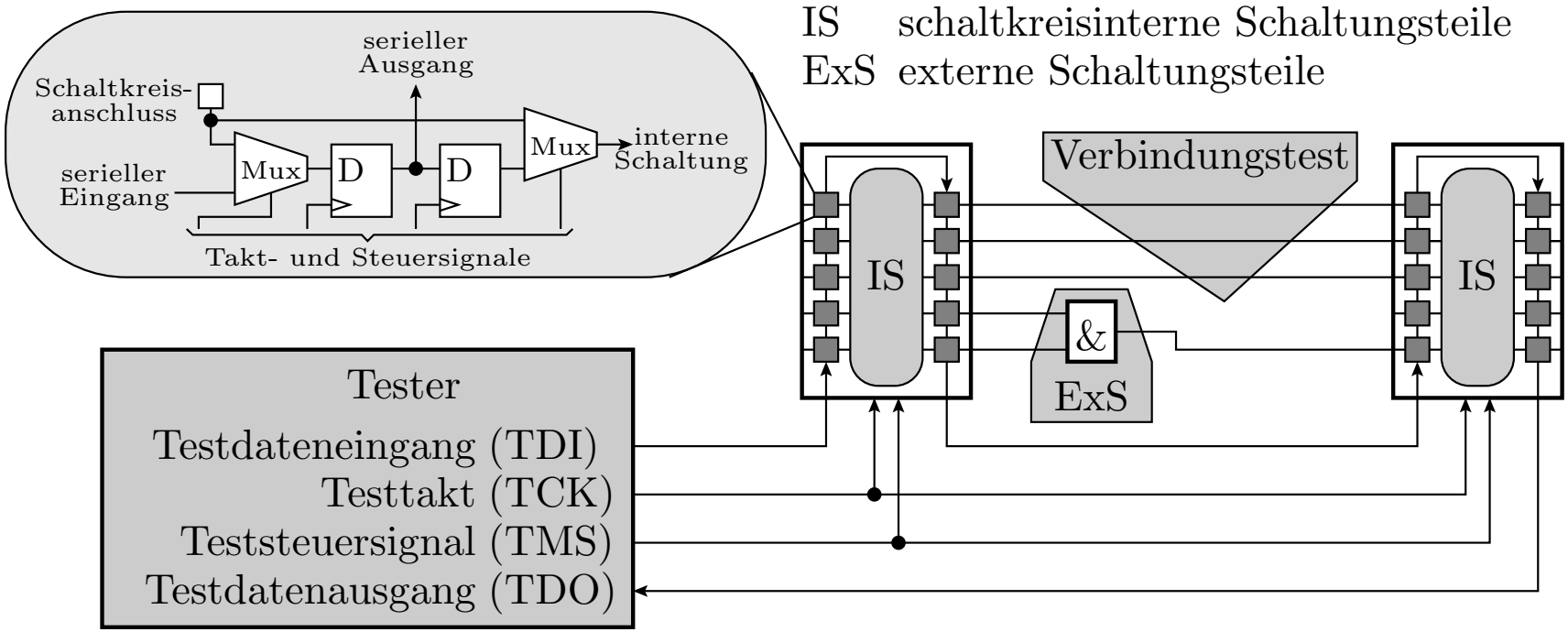


- Testphase 1: Erzeugung ungewichteter Pseudo-Zufallseingaben mit LFSR S1. Serielle Weitergabe an die Schiebereg. S2 und S3.
- Testphase 2: Verringerung der Wichtung in S2 durch UND-Verknüpfung von $\#E_{\text{AND}}$ Ausgabefolgen von S1 und für S2 durch »UND 0«. Erzeugung der Wichtungen $1 - 2^{-\#E_{\text{AND}}}$ und »1« durch Inverierung.

Nicht nennenswert aufwändiger als ohne Wichtung.

Testbus

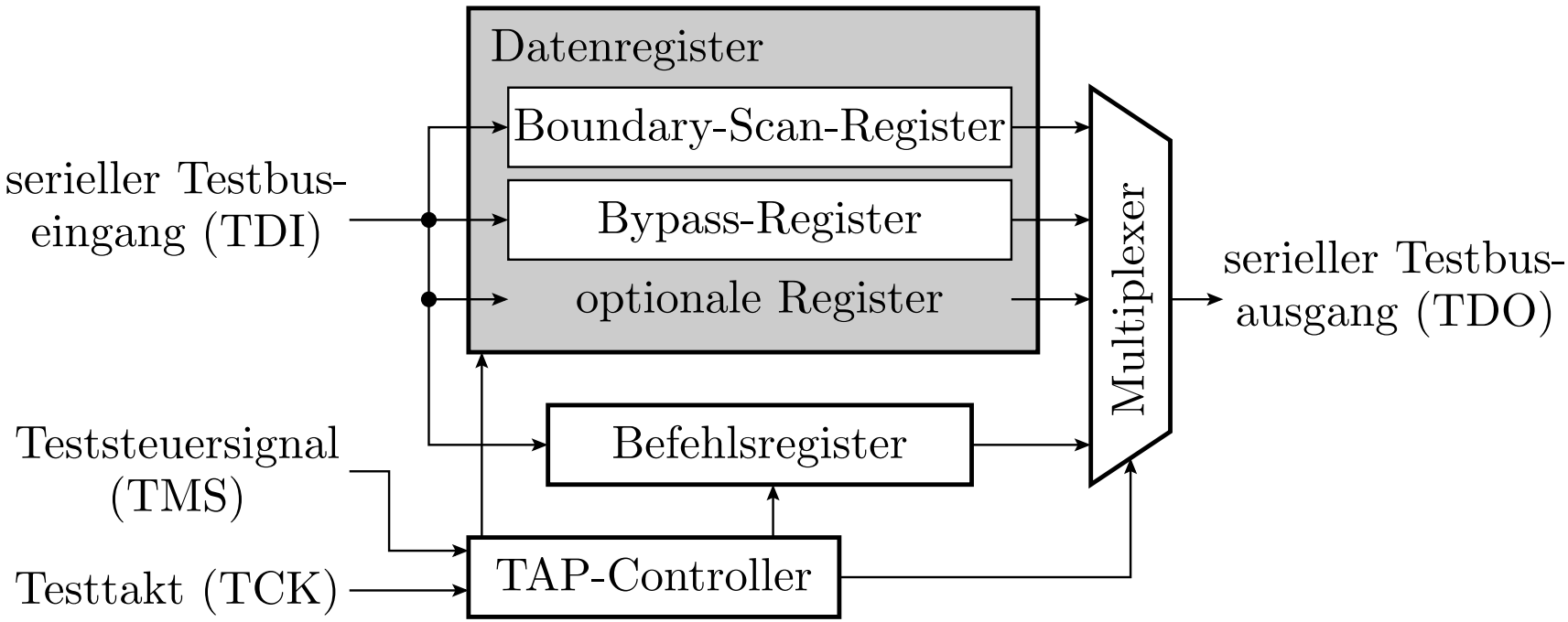
Testbus – JTAG (IEEE 1149.1)



Steuerung integrierter Testhilfen in Schaltkreisen und auf Baugruppen über einen seriellen Bus (minimale Anzahl von Verbindungsleitungen):

- Ersatz der mechanischen Nadeln für den Verbindungs-, Bestückungs- und isolierten Test integrierte Teststrukturen.
- Laden und Lesen von Programm- und Konfigurationsdaten, ...

JTAG-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).

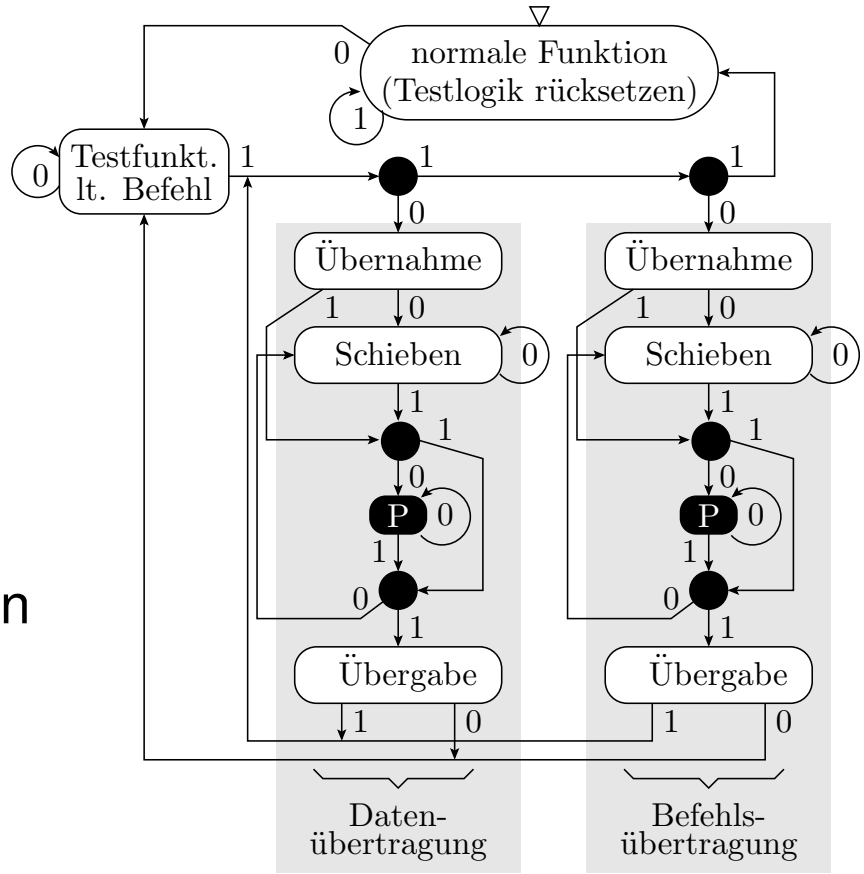
Erlaubt Lese- und Schreibzugriff auf LFSR für den Selbsttest.

Lesen und Schreiben der Testregister

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

Typischer Testablauf:

- Befehlsregister lesen (Bestückungskontrolle[7]),
- Bauteilnummern lesen (Bestückungskontrolle),
- Einen Teil der Schaltkreise auf Bypass setzen. Für die anderen Datenregister auswählen.
- Verbindungstest.
- Adressregister Auswählen,
- Adressregister Schreiben,
- Adressierten Speicherplatz lesen, Datenregister auswählen, ...



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

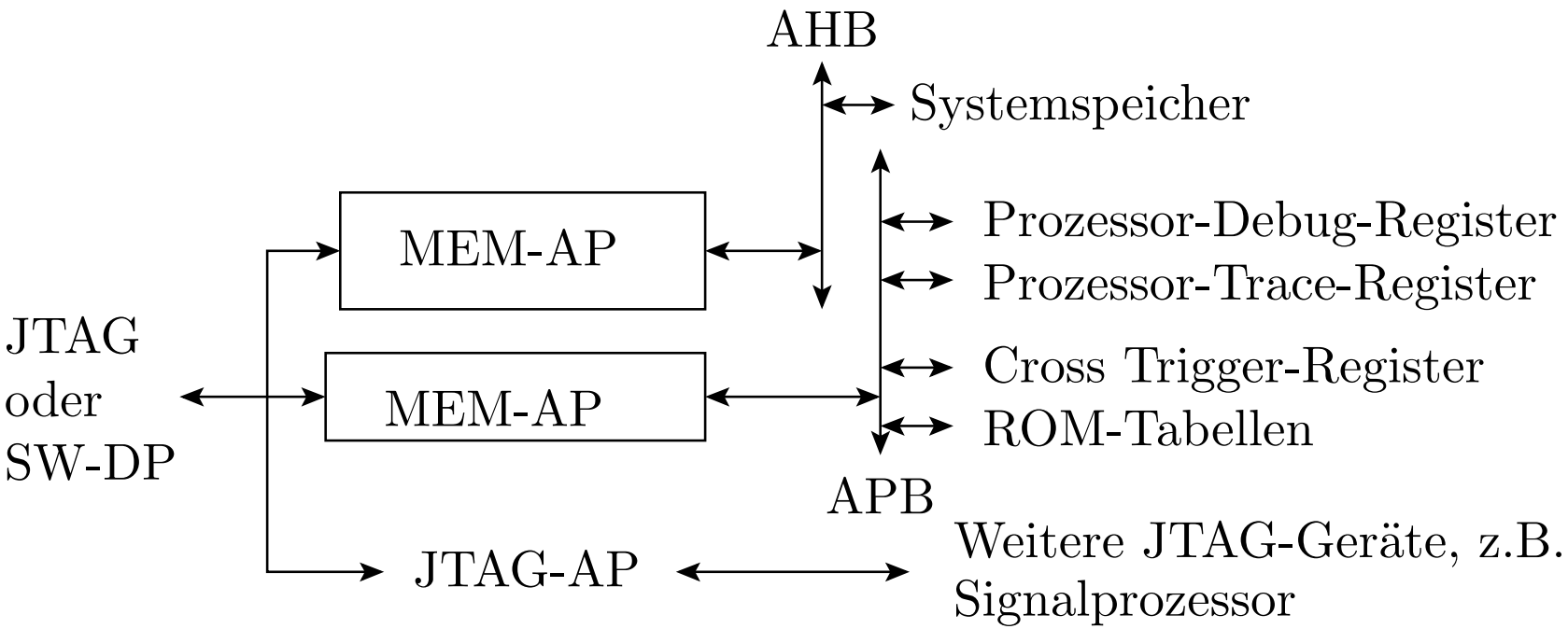
Über das Befehlsregister lässt sich eine

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

Typische unterstützte Testfunktionen:

- lesbare Hersteller- und Bauteilidentifikationsregister für den Bestückungstests.
- Steuerung von Steuerung 0, 1, hochohmig und Lesen der Logikwerte der IC-Anschlüsse für den Verbindungstest.
- Laden und Lesen von Programm- und Konfigurationsdaten.
- Steuerung integrierter Debugger-Funktionen,
- Steuerung von Selbsttests, ...

ARM-Testbusarchitektur



AHB	Advanced High Performance Bus
APB	Advanced Peripherial Bus
MEM-AP	Memory Acces Port
SW-DP	Serial-Wire Debug Port

Auch MR-Debug-Hilfen lassen sich über den Testbus steuern.