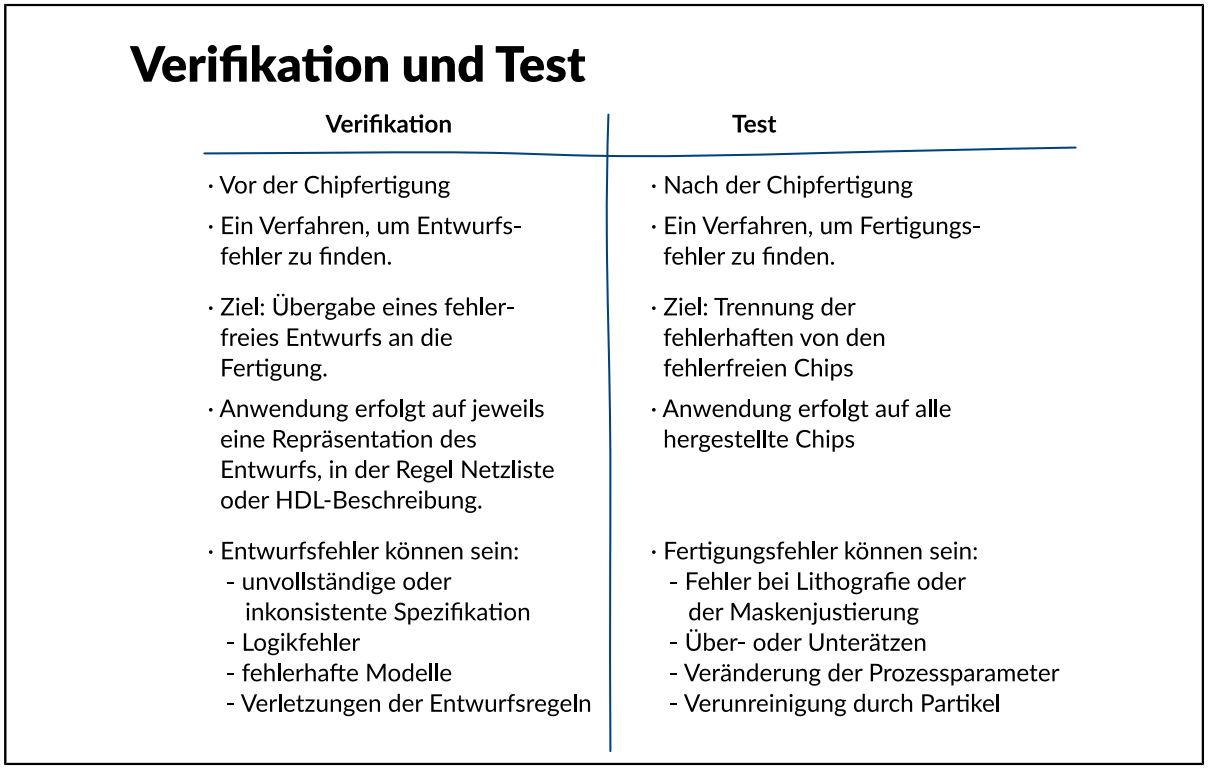
Electronic Design Automation (EDA)

Test

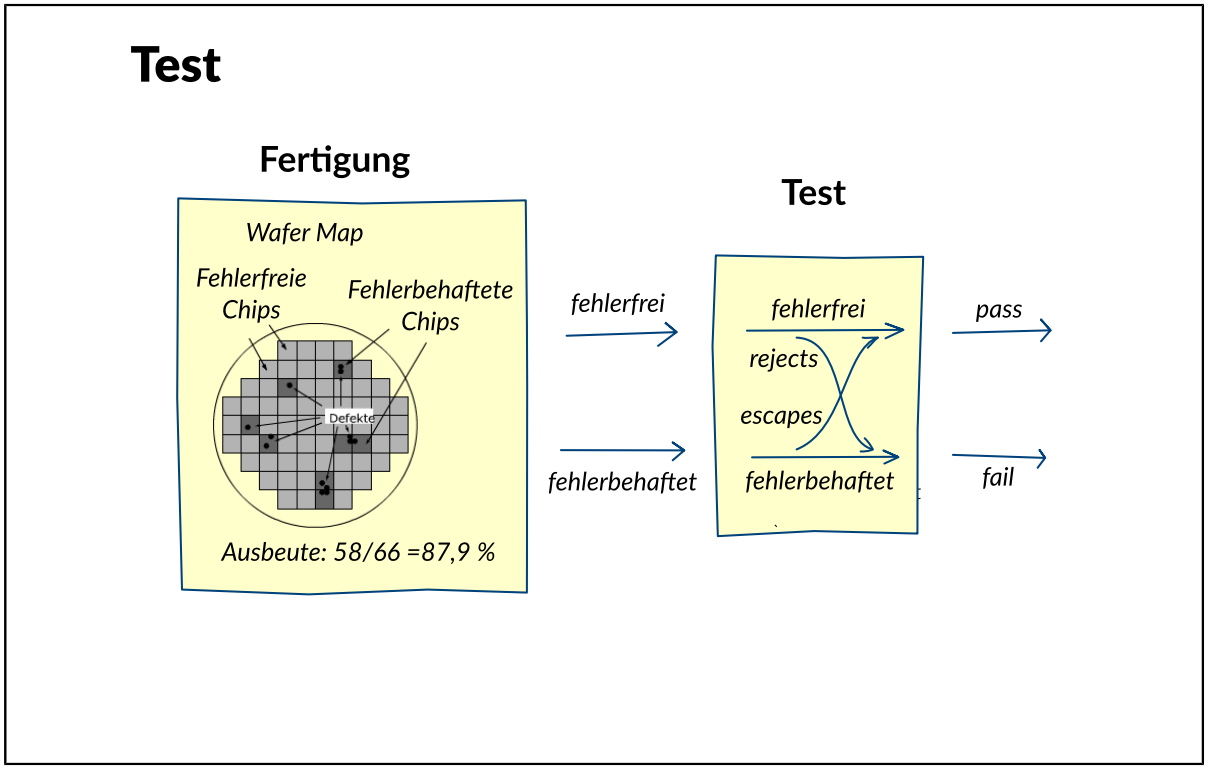
Date of generation: 7.6.2022

Test: Verifikation und Test



Ein großer Teil des Aufwands beim Entwurf einer integrierten Schaltung beinhaltet die Analyse der Entwurfsergebnisse zur Überprüfung auf Einhatung der Spezifikation. Je früher im Entwurfsprozess ein Entwurfsfehler entdeckt wird, desto leichter lässt es sich beheben. Die Überprüfung der Entwurfsschritte zu diesem Zweck wird als Verifikation bezeichnet. Nach der Fertigung des Chips muss die Spezifikation erneut überprüft werden, um festzustellen, ob möglicherweise während des Fertigungsprozesses defekte Chips entstanden sind. Verifiziert wird, um Entwurfsfehler zu beheben. Getestet wird, um defekte Chips auszusortieren.

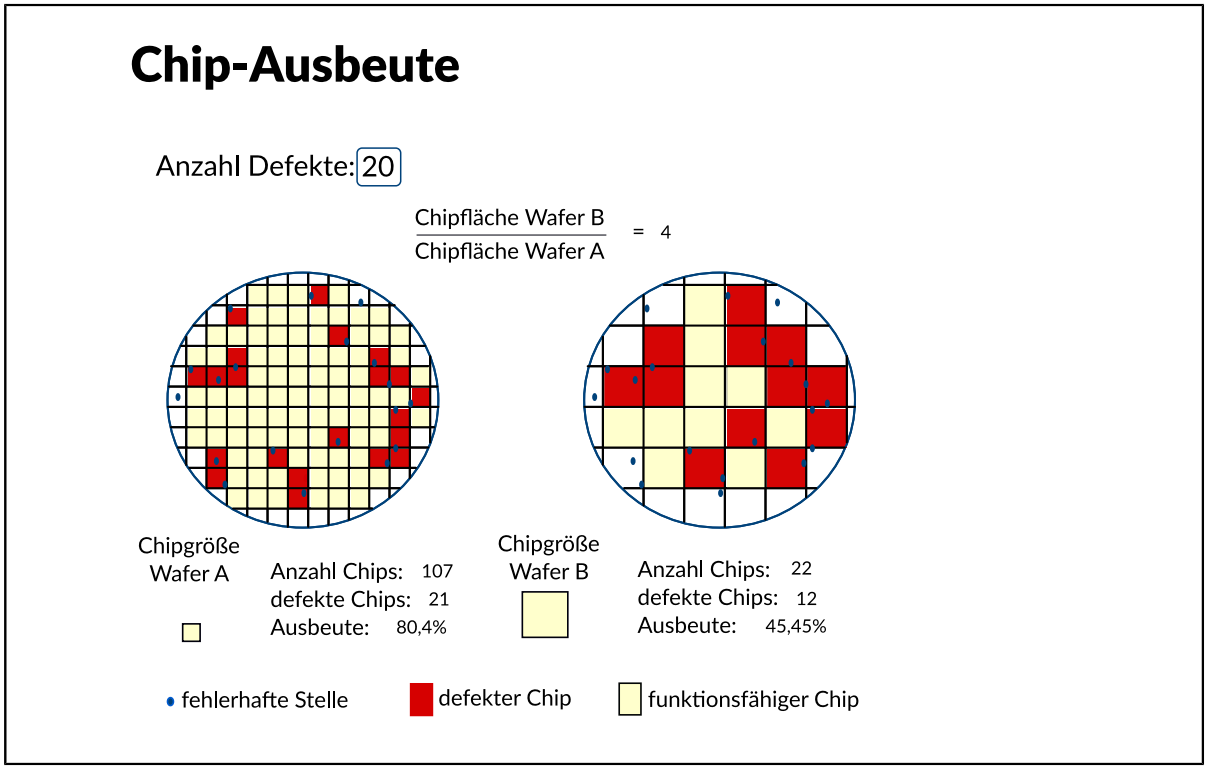
Test: Test



Wie bereits erwähnt soll der Test die fehlerhaften und fehlerfreien Chips voneinander trennen. Der Test soll dabei ausschließlich fehlerfreie Chips als solche markieren. Auf Grund der Komplexität der integrierter Schaltungen ist es nicht möglich die Schaltungen vollständig zu testen, so dass diese Anforderungen nicht vollständig erfüllt werden kann und fehlerbehaftete Chips als fehlerfrei "erkannt" werden. Es entstehen "escapes". Auf der anderen Seite kann auf Grund der Unvollkommenheit der Testumgebung (z.B. mangelnde Kontaktierung zwischen Testspitzen und Testpunkten) zum Aussortieren fehlerfreie Chips kommen, die als fehlerbehaftet erkannt werden. Es entstehen "rejects".

Die "escapes" führen zu den Ausfällen in den Zielsystemen und verursachen zusätzliche Kosten, die durch den Austausch der Chips bzw. der Systeme entstehen. Ein "reject" senkt künstlich die Ausbeute und trägt ebenfalls zur Steigung der Kosten bei.

Test: Chip-Ausbeute



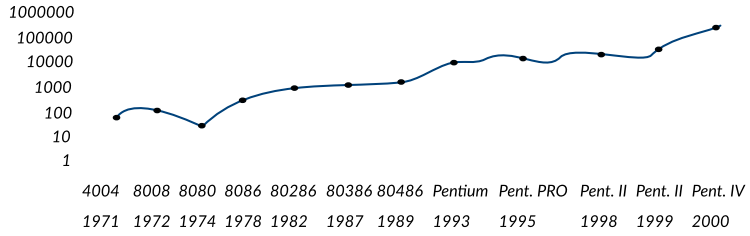
Test: Testbarkeitsindex

Testbarkeitsindex

Das Testbarkeitsindex ist ein Maß für die Erreichbarkeit schaltungsinterner Knoten

$$\text{Testbarkeitsindex} = \frac{\text{Anzahl Transistoren}}{\text{Anzahl externer Signalanschlüsse}}$$

Testbarkeitsindex von Intel-Mikroprozessoren:



4004	8008	8080	8086	80286	80386	80486	Pentium	Pent. PRO	Pent. II	Pent. II	Pent. IV
1971	1972	1974	1978	1982	1987	1989	1993	1995	1998	1999	2000

Die Testkosten sind bei der Herstellung integrierter Schaltungen ein wesentlicher Faktor. Sie steigen mit wachsender Komplexität. Ein wesentlicher Grund dafür ist, dass die Anzahl externer Anschlüsse nicht im gleichem Maße steigt, wie die Anzahl integrierter Transistoren. Hieraus folgt, dass die Zahl interner Zustände, die beim Test zu durchlaufen ist, zunimmt, wodurch die Testzeit und damit die Testkosten steigen.

Test: Testqualität

Testqualität

$$\text{Testqualität} = \frac{\text{Anzahl korrekter Bausteine}}{\text{Anzahl ausgelieferter Bausteine}}$$

Wenn 99,9 % gut genug sind, dann werden:

- heute 17 Kinder den falschen Eltern zugeordnet.
- jedes Jahr 268.000 defekte Autoreifen ausgeliefert.
- jedes Jahr 14.000 defekte PCs verkauft.
- jedes Jahr 2,5 Mio. Bücher mit einem falschen Schutzumschlag geliefert.
- täglich zwei unsichere Landungen auf dem Flughafen von Los Angeles stattfinden.
- in der nächsten Stunde 18.000 E-Mails ihren Adressaten verfehlen.

Quelle: K. - T. Cheng: "Lecture on VLSI Testing Techniques", University of California, Santa Barbara.

Die Qualität der ausgelieferten Chips ist ein wichtiges Verkaufsargument. Heute geforderte (und erreichte) Testqualitäten liegen im Bereich ppm (parts per million). 99,9 % fehlerfreie Chips sind also bei weitem zu wenig.

Test: Vollständiger Test kombinatorischer Schaltungen

Vollständiger Test kombinatorischer Schaltungen

Annahme: Ein Mikroprozessor mit 300 Signal-Eingängen sei rein kombinatorisch aufgebaut.

Dann ist die Anzahl möglicher Eingangskombinationen:

$$2^{300} = \text{ca. } 2 * 10^{90}$$

Nehmen wir weiter an, der Test wird mit einer Frequenz von 200 MHz ausgeführt.

Folgerung: Dann dauert der Test

$$\frac{2 * 10^{90}}{200 * 10^{6} s^{-1}} = \text{ca. } 3 * 10^{74} \text{ Jahre}$$

Ein vollständiger Test ist bereits bei kombinatorischen Schaltungen mit einer großen Anzahl von Eingängen unmöglich.

Test: Vollständiger Test sequentieller Schaltungen

Vollständiger Test sequentieller Schaltungen

Bei sequentiellen Schaltungen kommt zur Variation der Eingangsmuster die der internen Zustände hinzu.

Zahl der erforderlichen Tests:

2^{n+m}

n = Anzahl der Eingänge
m = Anzahl der Zustandbits

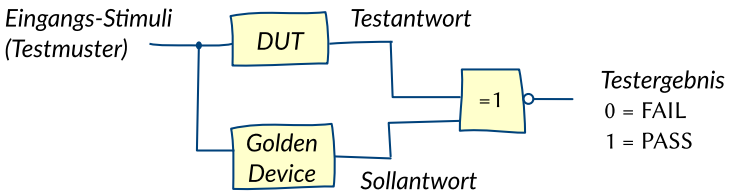
Beispiele:

D-Flipflop (data, reset, clock, enable) n = 4, m = 1	$2^5 = 32$ Tests
30-bit-Zähler (clock, reset) n = 2, m = 30	$2^{32} = 4 * 10^9$ Tests

Bei sequentiellen Schaltungen vervielfacht sich die Problemgröße durch die Anzahl der Zustände. Die Anzahl der notwendigen Test steigt auf 2n+m. n entspricht dabei der Anzahl der Eingänge und m der Breite der Zustandsvariable, die für die Kodierung der Zustände der sequentiellen Schaltung verwendet wird.

Test: Testdurchführung

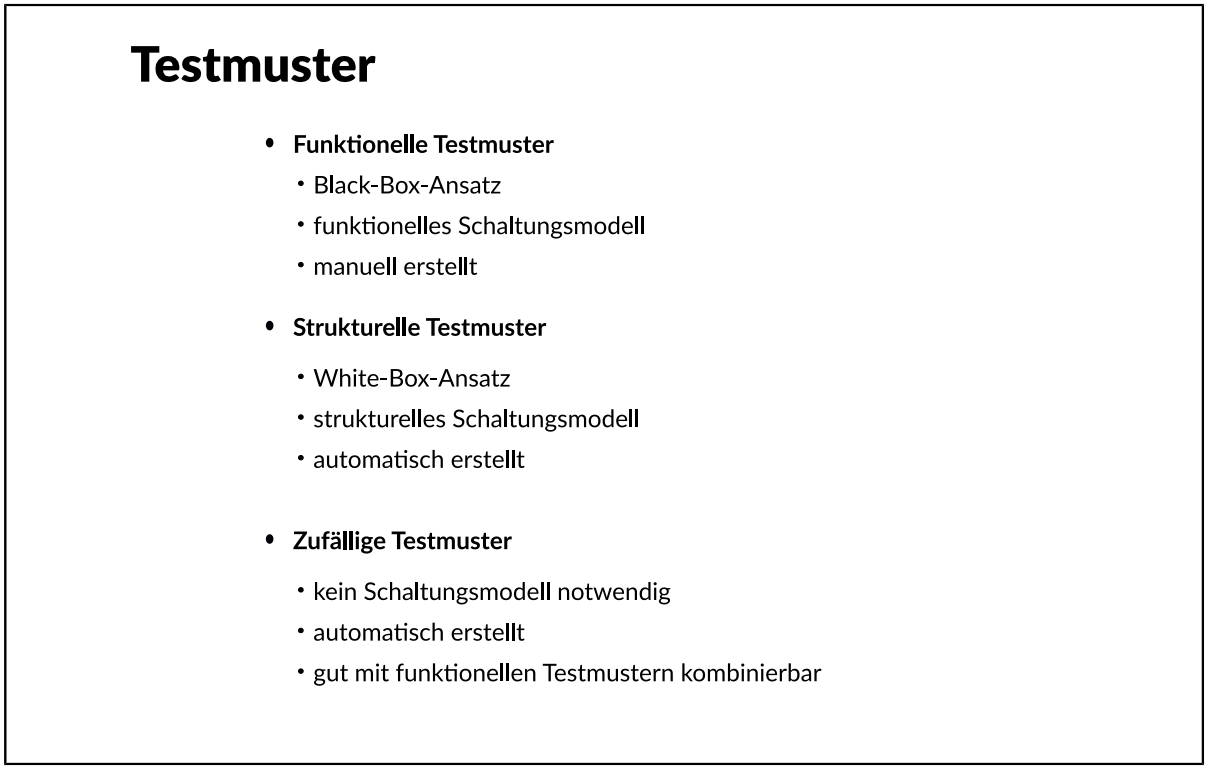
Testdurchführung



DUT: Device Under Test

Golden Device: Simulationsmodell oder bekannt fehlerfreie Schaltung

Test: Testmuster



Man unterscheidet drei Arten von Eingangsstimuli:

- **Funktionelle Testmuster**
  Der Entwickler erstellt, ausgehend von der Funktion des Prüflings, manuell Eingangsstimuli. Jede spezifizierte Funktion ist durch sie einmal anzusprechen. Je komplexer der Baustein ist, desto schwieriger gestaltet sich die manuelle Erstellung. Automatische Verfahren können nur in einfachen Fällen verwendet werden.
- **Strukturelle Testmuster**
  Die Eingangsstimuli werden entsprechend der logischen Schaltungsstruktur durch Stimuligeneratoren automatisch berechnet. Eine Reihe von Algorithmen (z.B. D-Algorithmus, PODEM-Algorithmus, FAN-Algorithmus) wurde dazu entwickelt. Die Schaltungskomplexität, besonders die sequentielle Tiefe, verschlechtert die Effizienz der Algorithmen. Durch geeignete Strukturmaßnahmen in der Schaltung oder durch Zusatzschaltungen kann die Effizienz jedoch wieder verbessert werden.
- **Zufällige Testmuster**
  Zufallstestmuster können auf einfache Weise generiert werden. Sie werden häufig als Ergänzung funktioneller Testmuster eingesetzt.

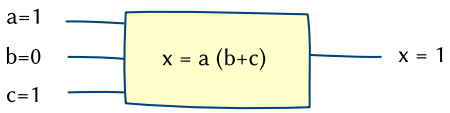
Wichtig ist es, eine Aussage zu treffen, welche Qualität die ausgewählten Testmuster besitzen, d.h. die Frage zu beantworten, welche Fehler erkannt werden können. Für ein internes Netz wird dabei berechnet, ob mit den vorhandenen Testmustern der Fehlerfall beobachtet werden kann. Zusätzlich wird ein Fehlerabdeckungsgrad als Maß für die Vollständigkeit der Fehlererkennung angegeben. Es errechnet sich aus der Anzahl der erkannten Fehler bezogen auf die Anzahl aller möglichen Fehler entsprechend dem zugrunde gelegten Modell. Zur Berechnung des Fehlerabdeckungsgrads werden so genannte Fehlersimulatoren eingesetzt.

Test: Funktionelle Testmuster

Funktionelle Testmuster

Funktionelle Testmuster basieren auf einer funktionellen Schaltungsbeschreibung.

Black-Box-Ansatz: Beschreibung des Ausgangsverhaltens in Abhängigkeit von der Eingangsbelegung.



Bei funktionellen Testmustern besteht die Gefahr, dass die Schaltung nur bezüglich einer beabsichtigten bzw. vorhersehbaren Funktion getestet wird. So bleiben leicht Fehler für "nicht vorhergesehene" Eingangsbelegungen unentdeckt.

Größter Nachteil funktioneller Testmuster ist, dass wegen eines fehlenden Fehlermodells keine Aussage über die Qualität einer Menge von Testmustern (Testmustersatz) gemacht werden kann.

Test: Strukturelle Testmuster

Strukturelle Testmuster

Strukturelle Testmuster basieren auf einer Strukturbeschreibung der Schaltung (White-Box-Ansatz).

Annahme von konkreten Fehlern in der Schaltung → Fehlermodelle erforderlich!

Qualität der strukturellen Testmuster ist messbar:

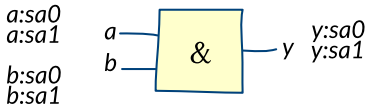
$$\text{Fehlerabdeckungsgrad} = \frac{\text{Anzahl der erkennbaren Fehler}}{\text{Anzahl aller möglichen Fehler gemäß des Fehlermodells}}$$

Strukturelle Testmuster werden erzeugt, indem entsprechend einem geeigneten Fehlermodell ein Fehler in der Schaltung an einem konkreten Ort angenommen wird und eine Eingangsbelegung berechnet wird, mit der dieser Fehler einstellbar und beobachtbar ist. Da hier tatsächlich einzeln Schaltungsfehler modelliert werden, spricht man auch von einem "defektorientierten" Test.

Die Qualität eines strukturellen Testmustersatzes ist über seinen Fehlerabdeckungsgrad messbar.

Test: Fehlermodelle

Fehlermodelle



Haftfehlermodell
2 mögliche Haftfehler pro Netz

a | y:sa0 = Knoten ist ständig logisch 0

a | y:sa1= Knoten ist ständig logisch 1

Das Haftfehlermodell ist das "klassische" Fehlermodell.
"Nicht klassische" Fehler:

- Stuck-Open-Fehler (offener Schaltungsknoten)
- Verzögerungsfehler (verzögerte Signalflanke)
- Übergangsfehler (ausbleibende Signalflanke)
- Stuck-On-Fehler (ständig eingeschalteter Transistor)
- Brückenfehler (Kurzschluss zwischen zwei Knoten)
- Crosstalk-Fehler (Übersprechen)

Vorteile des Haftfehlermodells:

- Einfaches Modell -> gut für automatische Testmustergenerierung geeignet
- Modelliert eine große Menge physikalischer Defekte
- Technologieunabhängig (Gatterebene)

Ein Fehlermodell stellt eine Abstraktion physikalischer Fehlermechanismen dar. Ein spezifisches Modell deckt jeweils nur eine Untermenge von Fehlermechanismen ab, es gibt deshalb zahlreiche unterschiedliche Modelle, die sich teilweise auch überlappen.

Wegen der Komplexität des Testproblems benutzt man in der Regel sehr einfache Fehlermodelle, die sich jedoch in der Praxis bewährt haben. Das bekannteste ist das so genannte Haftfehlermodell (Stuck-at-0/1), das von folgenden Voraussetzungen ausgeht:

- Fehler treten nur in Verbindungsleitungen (Netzen) auf.
- Ein Fehler äußert sich stets in der Art, dass ein Netz ständig auf logisch 0 (stuck-at-0, sa 0) oder logisch 1 (stuck-at-1, sa 1) festgehalten wird.
- Die Schaltung enthält nur einen Fehler (Einzelhaftfehlermodell).

Das Haftfehlermodell ist das klassische Fehlermodell. Alle anderen Modelle beschreiben so genannte nicht-klassische Fehler, dazu gehören:

- Stuck-open-Fehler ("offener" Schaltungsknoten)
- Verzögerungsfehler (verzögerte Signalflanke)
- Übergangsfehler (ausbleibende Signalflanke)
- Stuck-on-Fehler (ständig eingeschalteter Transistor)
- Brückenfehler (Kurzschluss zwischen zwei Knoten)
- Crosstalk - Fehler (Fehler durch Übersprechen)

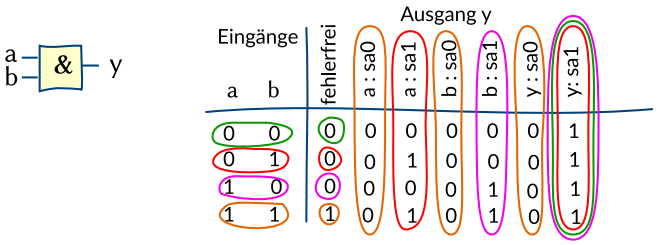
Haftfehler sind physikalisch gesehen Verbindungen zwischen einem Knoten und einer Versorgungsspannungsleitung (Masse, VDD). Das Haftfehlermodell hat folgende Vorteile:

- Wegen seiner Einfachheit lässt es sich leicht in Algorithmen zur automatischen Testmustergenerierung einbauen.

- Es modelliert eine relativ große Menge physikalischer Defekte.
- Es ist technologieunabhängig, da auf der Gatterebene angesiedelt.

Test: Fehlererkennung und Testmuster

Fehlererkennung und Testmuster



(a,b) = (0,1) erkennt die Fehler a:sa1 und y:sa1

(a,b) = (0,0) erkennt den Fehler y:sa1

(a,b) = (1,0) erkennt die Fehler b:sa1 und y:sa1

(a,b) = (1,1) erkennt die Fehler a:sa0, b:sa0 und y:sa0

a:sa0, b:sa0 und y:sa0 sind **äquivalente** Fehler, da jeder Test für einen von ihnen auch die anderen findet.

y:sa1 ist ein **dominanter** Fehler, da er auch von allen Testmustern für a:sa1 oder b:sa1 entdeckt wird.

Ein Fehler wird erkannt, wenn am Ausgang ein Wert festgestellt wird, der vom Wert der fehlerfreien Schaltung abweicht. Eine Eingangsbelegung, die die Erkennung eines Fehlers ermöglicht, heißt Testmuster. Ein Fehler kann durch mehrere Testmuster erkannt werden. Ein Testmuster kann mehrere Fehler erkennen. Dabei sind folgende Unterscheidungen wichtig:

"Fehler f1 dominiert Fehler f2, wenn jeder Test für Fehler f2 auch Fehler f1 entdeckt."

entsprechend:

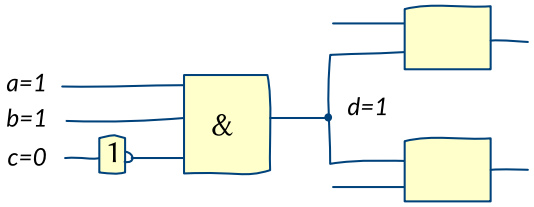
"Zwei Fehler f1 und f2 sind äquivalent, wenn jeder Test für Fehler f1 auch Fehler f2 findet, und umgekehrt."

Mit Hilfe dieser Fehlertypen kann die Liste aller zu testenden Fehler deutlich reduziert werden. Im Beispiel reicht es , für einen vollständigen Test, die Fehler a:sa0, a:sa1 und b:sa1 zu berücksichtigen. Ein Testmustersatz, der alle Fehler erkennt, wäre dann (a, b)={(0,1),(1,0),(1,1)}.

Test: Einstellbarkeit (Steuerbarkeit)

Einstellbarkeit (Steuerbarkeit)

Einstellbarkeit bezeichnet die Möglichkeit, ein Netz auf einen definierten Wert (0/1) setzen zu können:



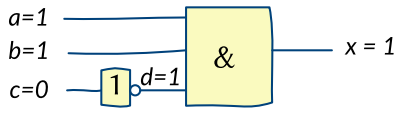
Der Testvektor x_T=(a,b,c)=(1,1,0) zwingt d auf logisch 1.
d=1 ist also einstellbar.
d=1 braucht man z.B. für einen Fehler d:sa0.

Wenn ein Fehler testbar sein soll, muss er einstellbar und beobachtbar sein. Einstellbarkeit (Steuerbarkeit) heißt, es gelingt durch eine Eingangsbelegung ein Netz auf einen definierten Wert einzustellen. Insbesondere gelingt es damit, am Fehlerort einen vom Fehler abweichenden Wert einzustellen. Im Beispiel wäre dies für den Fehler sa0 am Ausgang d des UND-Gatters der Fall.

Test: Beobachtbarkeit

Beobachtbarkeit

Beobachtbarkeit bezeichnet die Möglichkeit, den Wert eines Netzes am Ausgang einer Schaltung beobachten zu können.



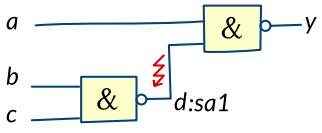
Der Eingang c=0 stellt hinter dem Inverter den Wert d=1 ein.
Dieser Wert ist durch den Testvektor $\mathbf{x}_T$=(a,b,c)=(1,1,0) am Ausgang y beobachtbar.

Beobachtbarkeit heißt, dass durch eine Eingangsbelegung ein Netz am Ausgang beobachtet werden kann. Insbesondere gelingt es damit, einen fehlerhaften Wert eines Netzes zum Ausgang zu propagieren.

Bei redundanzfreier kombinatorischer Logik ist grundsätzlich jeder Fehler steuer- und beobachtbar.

Test: Beispiel: Fehlererkennung

Beispiel: Fehlererkennung



Eingänge			d		Ausgang y	
a	b	c	fehlerfrei	d:sa1	fehlerfrei	d:sa1
0	0	0	1	1	1	1
0	0	1	1	1	1	1
0	1	0	1	1	1	1
0	1	1	0	1	1	1
1	0	0	1	1	0	0
1	0	1	1	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	1	0

- 8 mögliche Eingangskombinationen
- Einstellbarkeit am Fehlerort erfordert b = c = 1
- Beobachtbarkeit am Ausgang ist nur für die Eingangsbelegung (a, b, c) = (1, 1, 1) gegeben.

Die Schaltung ermöglicht 8 verschiedene Eingangskombinationen.

Einstellbarkeit am Fehlerort erfordert b = c = 1.

Beobachtbarkeit am Ausgang ist nur für die Eingangsbelegung (a, b, c) = (1, 1, 1) gegeben (gültiges Testmuster).

Der Fehler sa0 am gleichen Ort wäre mit mehreren Testmustern testbar.

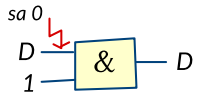
Test: D-Algorithmus

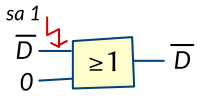
D-Algorithmus

D-Notation:

Logikwerte (fehlerfrei/fehlerhaft):

D = 1/0; $\overline{D}$ = 0/1; 0 = 0/0; 1 = 1/1; X = X/X;



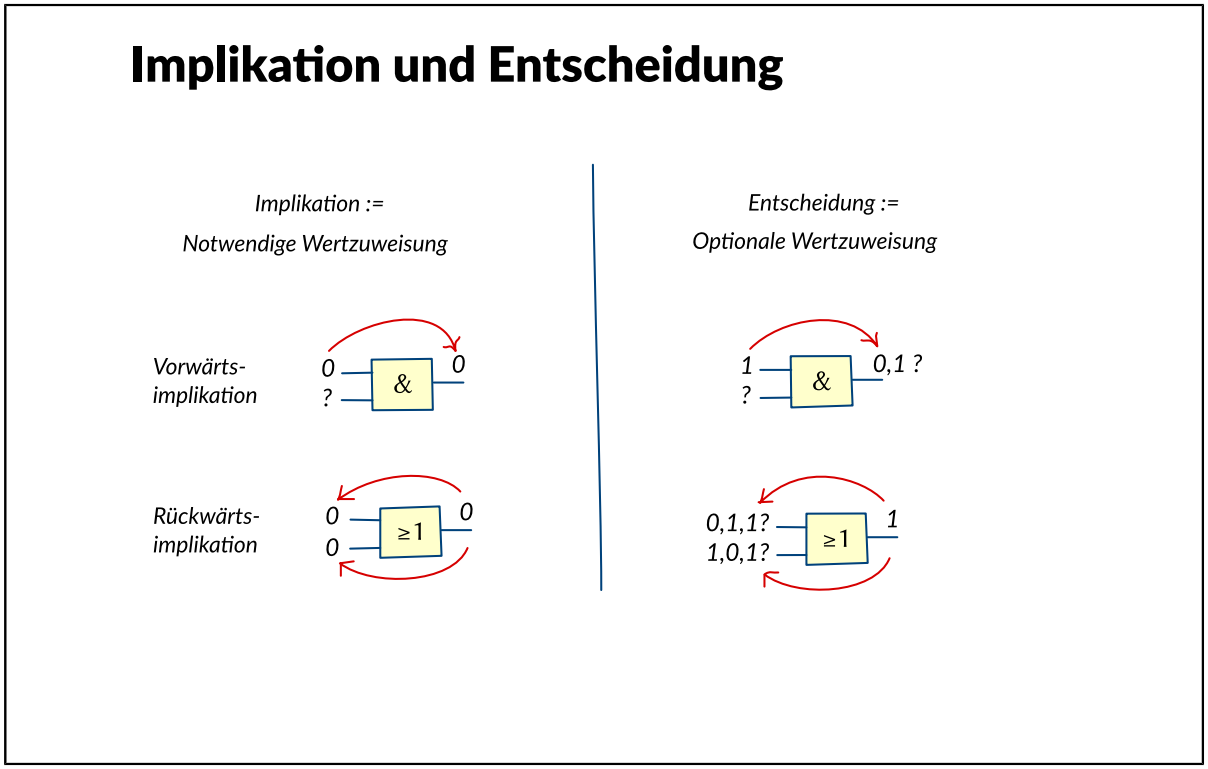


AND	0	1	X	D	$\overline{D}$
0	0	0	0	0	0
1	0	1	X	D	$\overline{D}$
X	0	X	X	X	X
D	0	D	X	D	0
$\overline{D}$	0	$\overline{D}$	X	0	$\overline{D}$

OR	0	1	X	D	$\overline{D}$
0	0	1	X	D	$\overline{D}$
1	1	1	1	1	1
X	X	1	X	X	X
D	D	1	X	D	1
$\overline{D}$	$\overline{D}$	1	X	1	$\overline{D}$

Für die automatische Generierung von Testmustern hat sich ein heute in zahlreichen Varianten verfügbarer Ansatz durchgesetzt, der auf das so genannte D (Discrepancy)-Kalkül von Roth zurückgeht. Dort wird für ein Signal, welches im fehlerfreien Fall den Wert '1' und im fehlerhaften Fall den Wert '0' besitzt (sa 0), der logische Wert 'D' notiert. Verhält es sich umgekehrt, d.h. ist der Wert im fehlerfreien Fall '0' und im fehlerhaften Fall '1', so wird dem Signal der logische Wert 'not D' (gesprochen "D quer") zugewiesen (sa 1). Ist der logische Wert für den fehlerfreien und fehlerhaften Fall gleich, nämlich '1' oder '0', so hat das Signal den Wert '1' bzw. '0'. Man erhält auf diese Weise eine fünfwertige Logik mit den Werten 0, 1, X, D und not D.

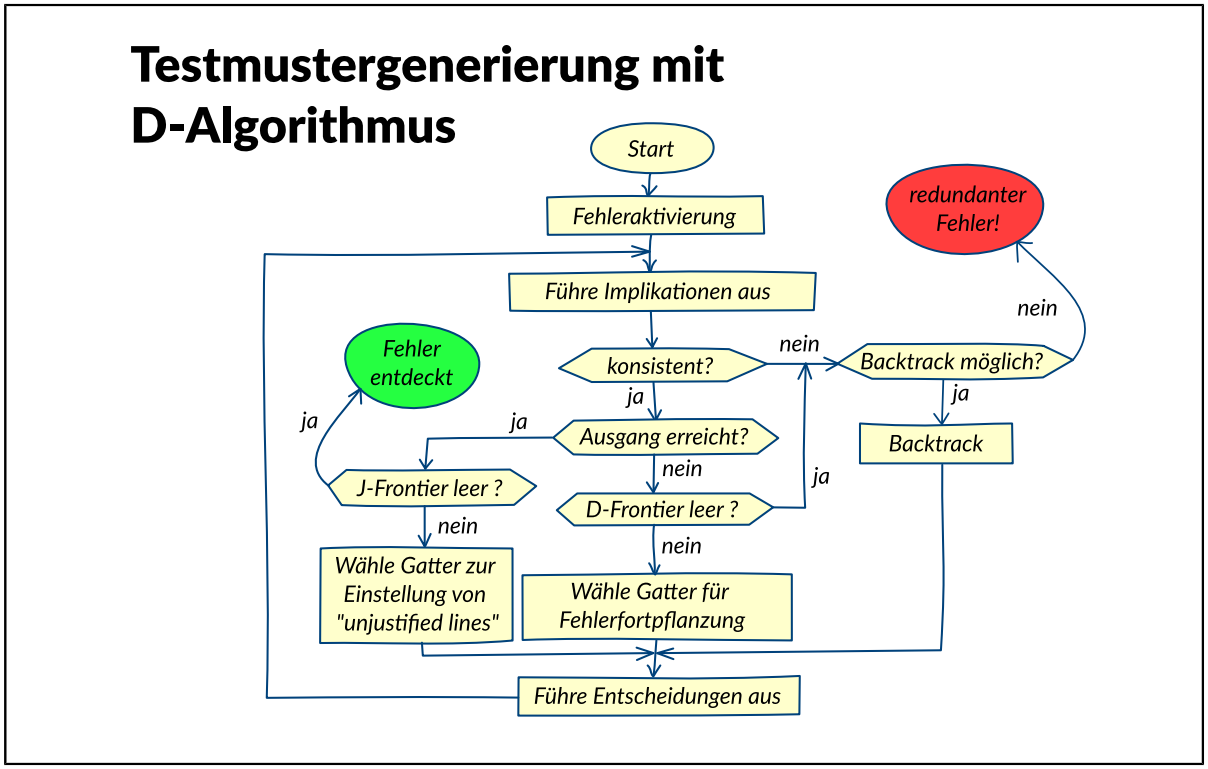
Test: Implikation und Entscheidung



Bei der Analyse logischer Schaltungen für die Testmustererzeugung erfolgen Wertzuweisungen. Es gibt zwei unterschiedliche Varianten:

- Implikationen sind notwendige Wertzuweisungen, die sich zwangsläufig und eindeutig ergeben.
- Entscheidungen sind optionale Wertzuweisungen, die Werte zuweisen, welche falsch sein können, d.h. die einen Konflikt herbeiführen können.

Test: Testmustergenerierung mit D-Algorithmus



Die Testmustergenerierung beginnt mit dem Rücksetzen aller Signale auf X (unbekannt) und dem Setzen des Signals am Fehlerort auf D (sa0) bzw. not D (sa1), der so genannten Fehleraktivierung. Nachdem das Fehlersignal gesetzt worden ist, werden alle möglichen logischen Implikationen ausgeführt, d.h. Wertzuweisungen, die sich aus dem Setzen des Fehlersignals eindeutig ergeben. Daraus ergibt sich eine Wertebelegung, die Ausgangspunkt für alle weiteren Schritte der Testmustergenerierung bildet. Bei den im Verlaufe der Testmustergenerierung entstehenden Wertebelegungen interessieren uns stets zwei Mengen von Netzen:

Die **D-frontier** (D-drive) besteht aus denjenigen Gattern, deren Wert am Ausgang (noch) unbestimmt ist, die aber ein oder mehrere Fehlersignale (D oder not D) an den Eingängen besitzen. Die D-frontier gibt an, wie weit sich fehlerhafte Signale in Richtung der primären Ausgänge ausgebreitet haben.

Die **J-frontier** besteht aus allen Gattern, deren Ausgänge einen festen logischen Wert besitzen, der sich jedoch (noch) nicht aus den Werten an den Gatter-Eingängen implizieren lässt. Die Ausgänge der J-frontier bezeichnet man oft auch als unjustified lines. Sie stellen Netze dar, an denen sich durch Wertzuweisungen im Inneren der Schaltung feste logische Werte ergeben haben, die aber noch nicht durch geeignete Wahl von Wertzuweisungen an den primären Eingängen eingestellt sind.

Die Aufgabe eines deterministischen Testmustergenerators lässt sich damit auch so beschreiben: Finde eine binäre Wertebelegung für die primären Eingänge der Schaltung, so dass

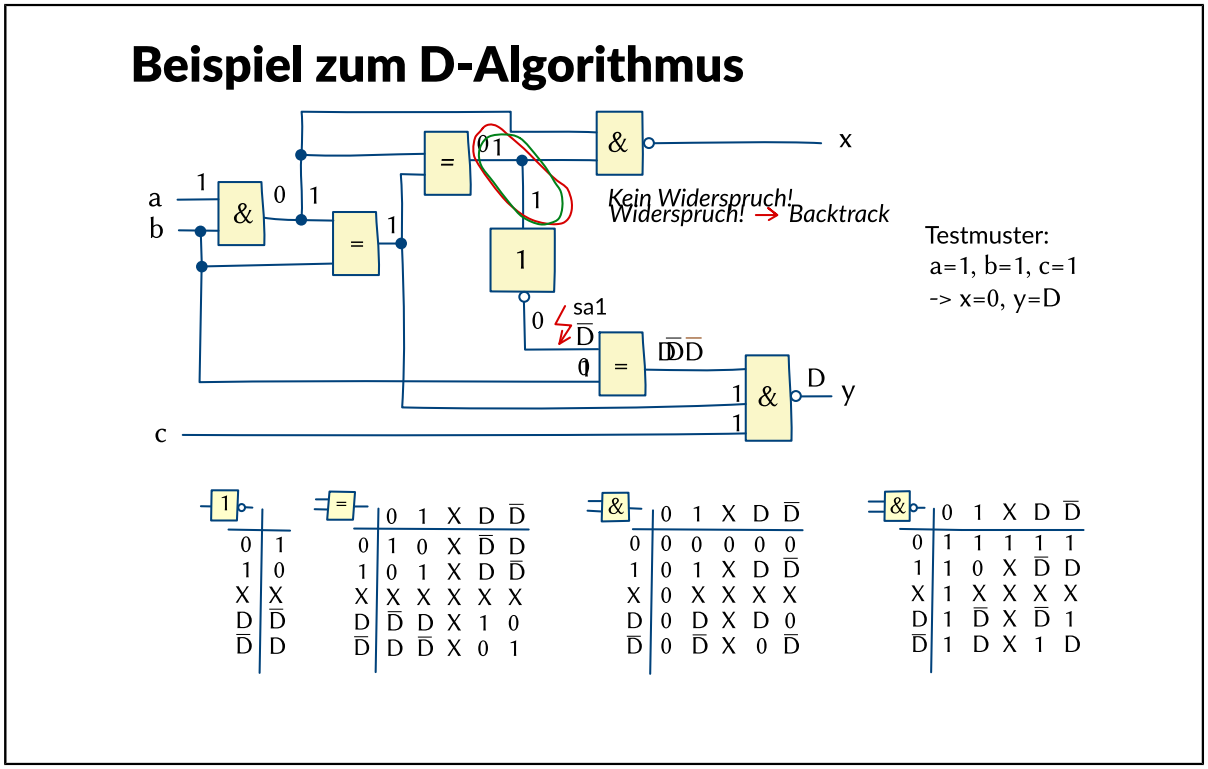
- die D-frontier einen primären Ausgang erreicht, d.h. wenigstens an einem primären Ausgang ein Fehlersignal anliegt (Beobachtbarkeit) und
- die J-frontier die primären Eingänge erreicht, d.h. im Inneren der Schaltung keine unjustified lines mehr existieren (Einstellbarkeit).

Der vorgestellte Algorithmus zur deterministischen Testmustergenerierung geht dabei so vor, dass er Schritt für Schritt durch geeignete Wertzuweisungen an geschickt ausgewählten Netzen die D-frontier in Richtung der primären Ausgänge und die J-frontier in Richtung der primären Eingänge verschiebt. Der Vorgang der Testmustererzeugung kann als eine Abfolge von optionalen und notwendigen Wertzuweisungen verstanden werden. Wie die notwendigen Wertzuweisungen bestimmt und wie die

optionalen Wertzuweisungen geschickt ausgewählt werden, variiert von Algorithmus zu Algorithmus und bestimmt die Leistungsfähigkeit der verschiedenen Tools. Dennoch ist der grundsätzlicher Ablauf bei allen herkömmlichen Verfahren weitgehend gleich.

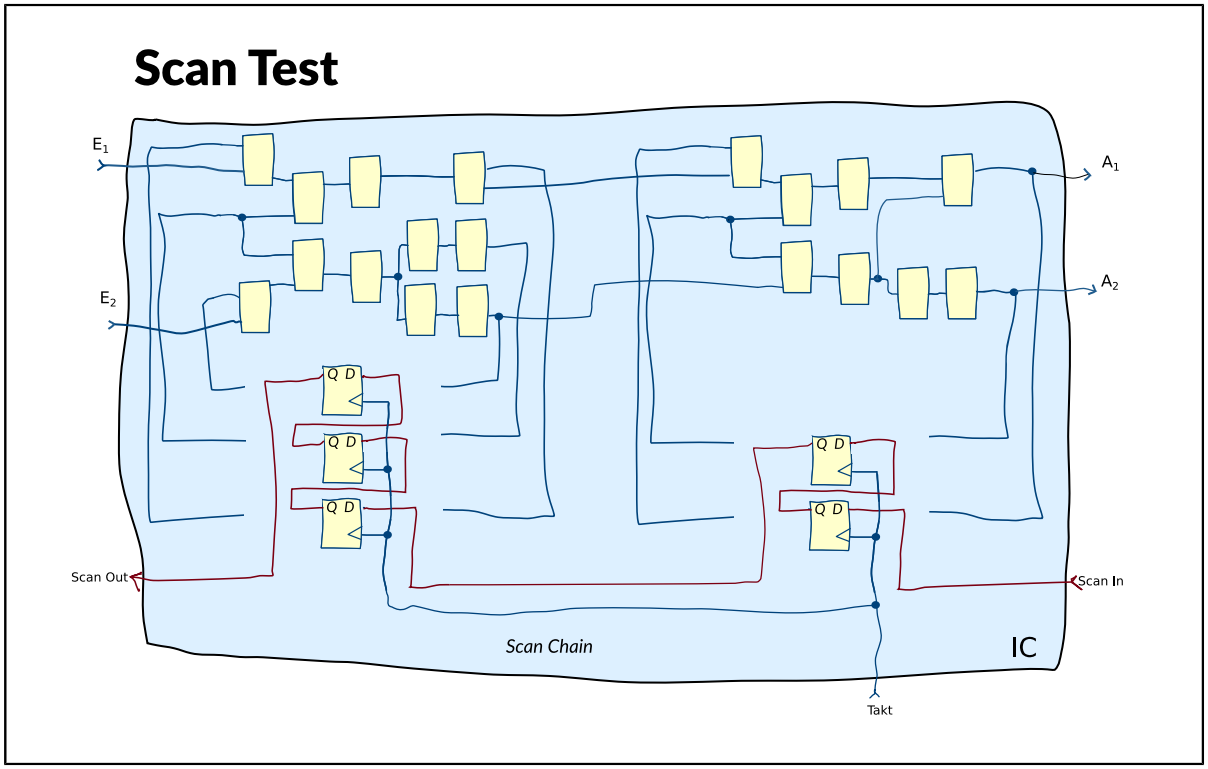
Führt der Algorithmus zu einem inneren Widerspruch, der sich nicht mehr durch einen Backtrack (also eine optionale Wertzuweisung) auflösen lässt, hat man einen **redundanten Fehler** gefunden. Dies sollte gar nicht oder nur sehr selten vorkommen, da durch den Algorithmus bewiesen ist, dass es keine Belegung an den Eingängen geben kann, so dass der Fehler am Ausgang sichtbar ist. Solch redundante Fehler werden bei Auftreten oft durch geeignete Redesign-Maßnahmen entfernt.

Test: Beispiel zum D-Algorithmus



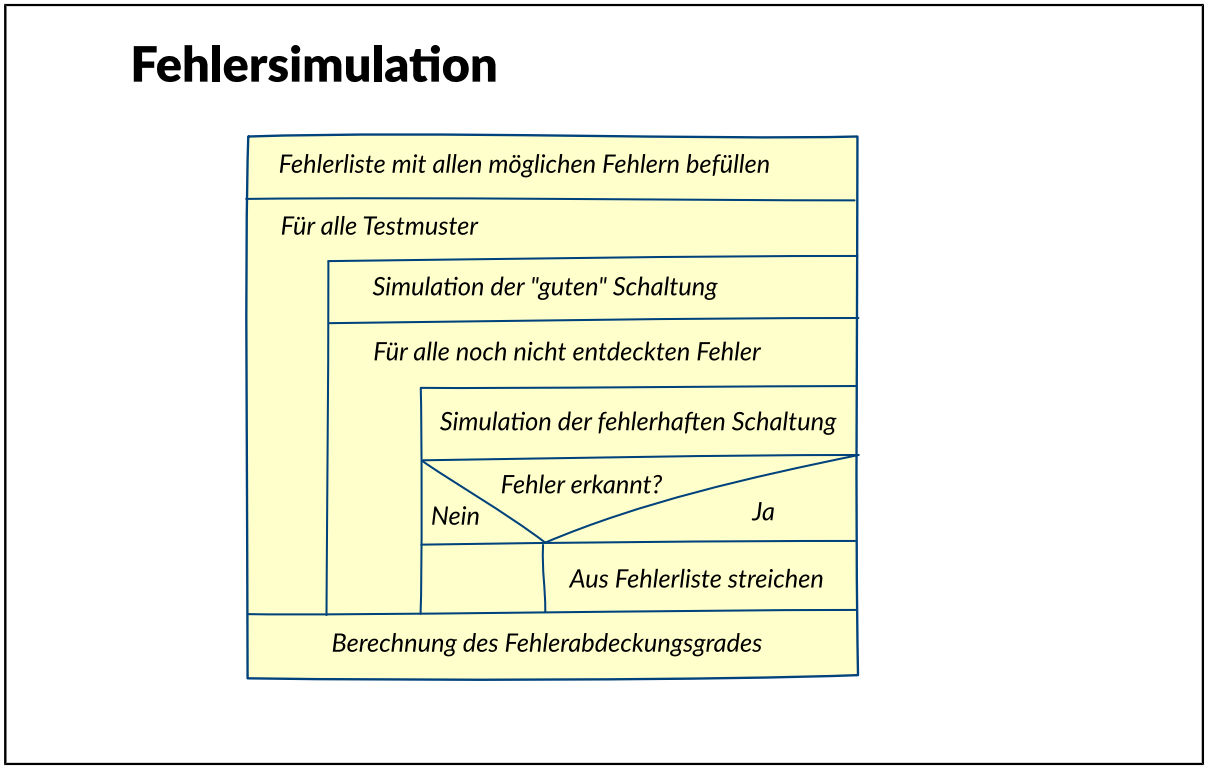
Im hier dargestellten Beispiel nehmen wir einen sa1-Fehler an und weisen dementsprechend das Fehlersignal not D zu. Die Animation zeigt, wie Wertzuweisungen innerhalb der Schaltung und an den Eingängen erfolgen und schließlich ein Testmuster erzeugt wird, mit dem der Fehler steuerbar (0 am Fehlerort) und beobachtbar (D an y) ist.

Test: Scan Test



Der Scan-Test dient dazu, während des Tests alle Werte von internen Flip-Flops nach außen zu führen. Die Flips-Flops sind dazu untereinander verkettet (Scan Chain) und das Ende Kette über einen Pin nach außen geführt. Dadurch können die Werte seriell nach außen zum Test geführt werden. Die Flip-Flops müssen dazu in einen sogenannten Shift-Betrieb geschaltet werden. Das Verketten der Scan Chain wird von speziellen EDA-Werkzeugen automatisiert durchgeführt.

Test: Fehlersimulation



Fehlersimulation dient dazu, die Qualität von Testmustersätzen zu bewerten. Dazu muss festgestellt werden, ob bei Annahme eines bestimmten Fehlers in der Schaltung ein Eingangsmuster geeignet ist, an einem Ausgang eine Veränderung gegenüber der fehlerfreien Schaltung hervorzurufen. Dies kann durch eine logische Simulation festgestellt werden.

Der Fehlerabdeckungsgrad ist der Quotient aus Anzahl gefundener Fehler und Anzahl aller Fehler gemäß Fehlermodells.

Ein Problem besteht in der Vielzahl möglicher Fehler (und Eingangsmuster), die zur Erreichung tragbarer Simulationszeiten parallele Methoden erforderlich macht. Parallele Methoden können mit einem Rennen verglichen werden, in dem eine beliebige Zahl von Teilnehmern (fehlerhafte Schaltungen) gegeneinander laufen. Mit einem normalen Logiksimulator lässt sich exakt dasselbe Ergebnis erzielen, aber jeder Teilnehmer muss einzeln laufen. Es müssen für N Fehler also N+1 Simulationsläufe durchgeführt werden, je einen für jeden Fehler und einen zusätzlichen für die fehlerfreie Schaltung. Neben der Zeiteinsparung haben parallele Methoden auch den Vorteil, dass sich der Vergleich zwischen fehlerhafter und korrekter Schaltung stark vereinfachen lässt.

Die bekannten parallelen Algorithmen werden auch als simultane, deduktive und konkurrierende Verfahren bezeichnet. Dabei nutzen alle die Eigenschaft, dass im Normalfall die fehlerhafte Schaltung nur geringfügig von der korrekten abweicht.