— Informatik I —
Modul 3: Schaltnetze (Fortsetzung)

Universität Zürich[UZH]

CSG

Rechnerstrukturen und -organisation (1)

- Einführung (M1)
- Rechnerarithmetik 1 (M2)
  - Zahlensysteme
  - Zahlendarstellung
  - Zeichendarstellung

✓ 28.9. & 2.11.2011

- Schaltnetze (M3)
  - Einführung in die formalen Grundlagen logischer Beschreibungen
  - Voraussetzende technische Entwicklungen
  - Realisierung von Schaltnetzen auf Schalter und Gatterebene
  - Entwurf von Schaltnetzen und Laufzeiteffekte bei Schaltnetzen
- Schaltwerke (M4)
  - Formale Grundlagen (Endliche Automaten)
  - Asynchrone Schaltwerke und Flipflops
  - Synchrone Schaltwerke
  - Register-Transfer-Ebene
  - Spezielle Schaltwerke

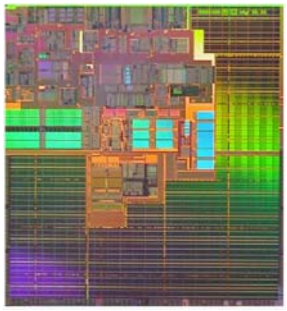
ab 16.11.2011

Modul 3: Schaltnetze

- Einführung in die formalen Grundlagen logischer Beschreibungen
  - Boolesche Algebra, Schaltalgebra
- Voraussetzende technische Entwicklungen
- Realisierung von Schaltnetzen auf Schalter- und Gatterebene
- Entwurf von Schaltnetzen
  - Logikminimierung, KV-Diagramme
  - Programmierung von Funktionen
- Laufzeiteffekte bei Schaltnetzen

Mikroprozessoren – Die Formel 1

- "Das Wettrennen um die schnellsten Mikroprozessoren könnte man als Formel 1 der Computertechnik bezeichnen." (Die ZEIT vom 18. März 1994)
- Die **Leistungssteigerung** bei Mikroprozessoren ist durch folgende Fortschritte erreicht worden:
  - durch Steigerung der Gatterzahl auf dem Chip,
  - durch Steigerung der Taktrate und
  - durch Fortschritte beim Hardware-Entwurf (Architektur, Mikroarchitektur und Entwurfswerkzeuge).

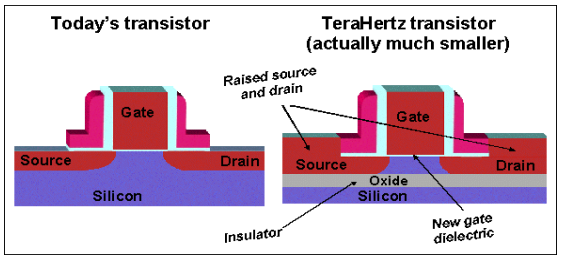


Intel Itanium 2 (Madison)

Beispiel: Terahertz-Transistor (Intel)

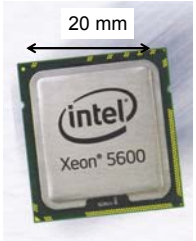
- Spitzen-Transitfrequenz: 2,63 THz, damit 0,38 ps Gatterverzögerung!
  - Also nur noch 0,000 000 000 000 38 s ...
  - Oder weniger als 76 µm Wegstrecke für elektromagnetische Wellen ...

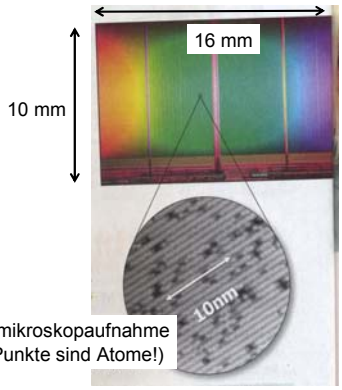
Tera: 10^{12}
pico: 10^{-12}
micro: 10^{-6}



Schaltkreise, CPUs – Größenordnungen

- 25 nm NAND Speicherchips
  - 8 GByte Speicher mit 167 mm² Fläche
  - Stapelbar – seit Mitte 2010 möglich!
- Intels XEON 5600 Serie
  - 32 nm Technologie
  - Bis zu 6 Cores





Rastertunnelmikroskopaufnahme (Die hellen Punkte sind Atome!)

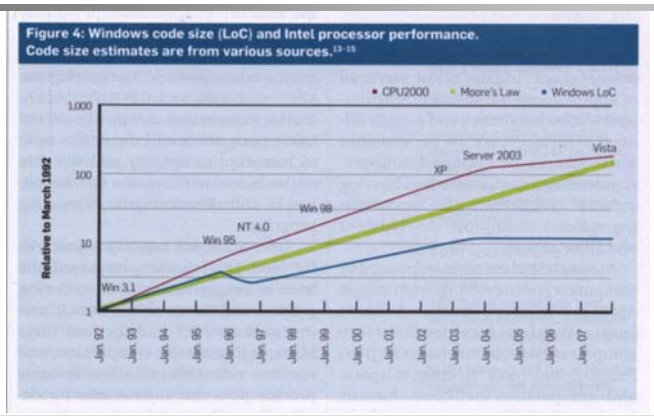
Immersionslithographie

2-12 “

Exponentialgesetz der Mikroelektronik: „Mooresches Gesetz“

- Die Anzahl der Transistoren pro (Prozessor-)Chip verdoppelt sich alle zwei Jahre.
- Die Verarbeitungsleistung der Hochleistungsprozessoren verdoppelt sich alle 18 Monate.
- Für den gleichen Preis liefert die Mikroelektronik die doppelte Leistung in weniger als zwei Jahren.
- Eine Chip-Fabrik stellt im Jahr 2002 die größte Einzelinvestition dar (10 Milliarden US-Dollar).
- Die Kooperation großer Firmen ist notwendig:
  - EUVLLC (extrem ultraviolet limited liability company) von AMD, Motorola und Intel.

Beispielanwendung „Moore‘s Law“: Code



Immer mehr Transistoren auf einem VLSI-Chip

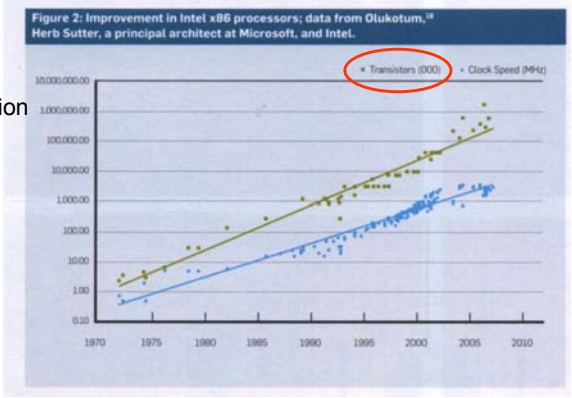
- SIA 1997 Roadmap für Prozessoren:
  - SIA = American Semiconductor Industry
  - http://public.itrs.net/
  - http://www.sematech.org/public/home.htm

Year of 1st shipment	1997	1999	2001	2003	2006	2009	2012
Local clock (GHz)	0.75	1.25	1.5	2.1	3.5	6	10
Across chip (GHz)	0.75	1.2	1.4	1.6	2	2.5	3
Chip size (mm^2)	300	340	385	430	520	620	750
Feature size (nm)	250	180	150	130	100	70	50
Number of chip I/O	1450	2000	2400	3000	4000	5400	7300
Transistors/chip	11M	21M	40M	76M	200M	520M	1.4G

…wurde schnell von der Realität überholt!

Prozessorentwicklungen — Beispiel: x86

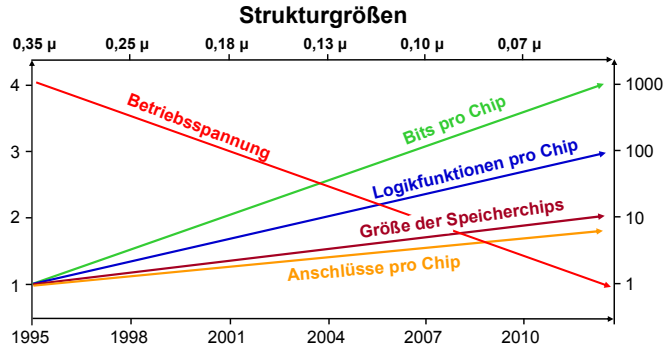
1 Million



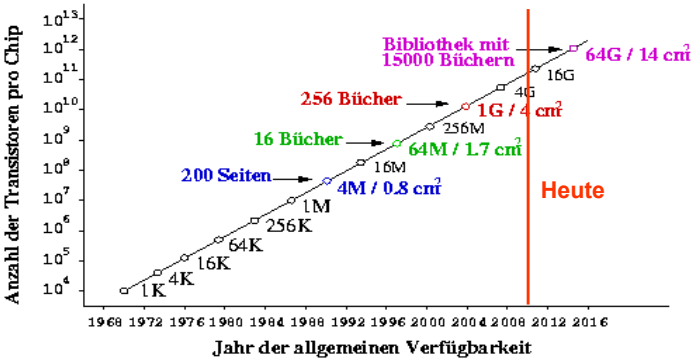
Beispiele bei Prozessoren

- AMD K6 II: 9,3 Mio Transistoren
- AMD K6 III: 21,3 Mio (inkl. 256 KB Level 2 Cache)
- AMD Athlon : 22 Mio
- Intel Pentium III: 9,5 Mio
- Intel Pentium III E: 28,1 Mio (inkl. 256 KB Level 2 Cache)
- Sun Ultra-Sparc III: 29 Mio, 900 MHz
- 2000: Intel Pentium 4: 55 Mio (inkl. L2)
- 2002: Intel Itanium 2: 410 Mio (inkl. L3)
- 2008: AMD X3 8450: 758 Mio Kosten: ca. 50 US$ pro Stück!
- 2010: Intel Xeon 5600: 2,3 Mrd Kosten: ca. 1500 US$ pro Stück!

Mehr Leistung bei weniger Stromverbrauch

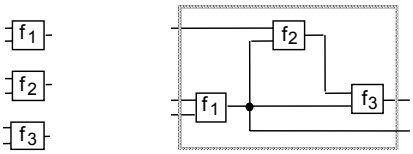


Verfügbarkeit von Speicherchips



Aufbau informationsverarbeitender Systeme (1)

- Meist sehr komplexe Systeme:
  - Vielzahl von Komponenten
  - Komponenten sind untereinander verbunden (Struktur)
  - Anwendung eines alten römischen Prinzips: „Teile und herrsche“!
  -
    ➔ Gewünschtes Verhalten



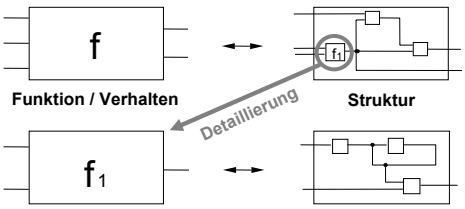
Komponenten + Struktur = Gewünschtes Verhalten

Aufbau informationsverarbeitender Systeme (2)

- Zwei wesentliche Aufgaben in der Informationstechnologie (IT) eines Informatikers/Ingenieurs/Systemarchitekten:
  - Entwurf (Synthese) und Analyse
- Hauptaufgabe des Entwurfs:
  - Bestimmte Komponenten mit bekanntem Verhalten in einer Struktur so zu verbinden, daß das Gewünschte erhalten bleibt bzw. resultiert und die Kosten (interne und externe) möglichst gering sind
    - Intern: Chip-Fläche (Quartz), Anzahl der E/A-Ports, ...
    - Extern: Energieeffizienz, CO2 Fußabdruck, ...
- Hauptaufgabe der Analyse:
  - Vorhersage und Simulation des Verhaltens einer Struktur aus bekannten Komponenten

Aufbau informationsverarbeitender Systeme (3)

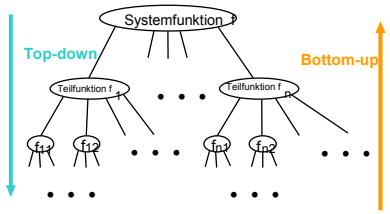
- Um die Komplexität beherrschen zu können, ist es notwendig, verschiedene Abstraktionsebenen einzuführen.



- Diese Hierarchisierung erleichtert sowohl den Entwurf als auch die Analyse

Vorgehensweisen beim Entwurf

- Zwei Extremstrategien:
- Top-down Entwurf: Rekursive Zerlegung der Gesamtfunktion, bis alle Teilfunktionen durch bekannte Komponenten ausgeführt werden
- Bottom-up Entwurf sukzessive Kombination von bekannten Elementen, bis das gewünschte Systemverhalten erreicht ist



Modul 3: Schaltnetze

- Einführung in die formalen Grundlagen logischer Beschreibungen
  - Boolesche Algebra, Schaltalgebra
- Voraussetzende technische Entwicklungen
- Realisierung von Schaltnetzen auf Schalter- und Gatterebene
- Entwurf von Schaltnetzen
  - Logikminimierung, KV-Diagramme
  - Programmierung von Funktionen
- Laufzeiteffekte bei Schaltnetzen

Realisierung von Schaltnetzen

Hierarchie:

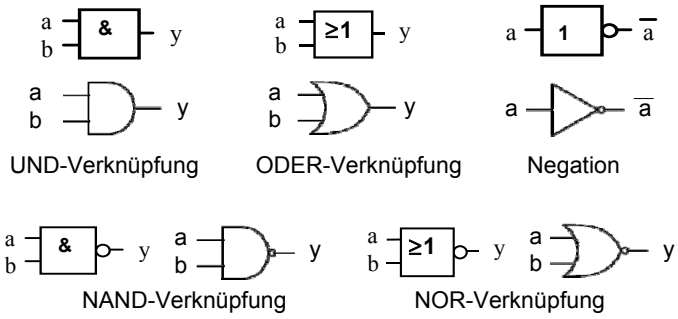
- Register-Transfer-Ebene: logische Bausteine als Grundelemente (Grundlage der Programmierung)
  - Multiplexer, Schieberegister, Addierer, ... **In M3/M4/M5 diskutiert**
- Gatterebene: logische Gatter **In M3 oben theoretisch gezeigt, jetzt in Gatter übersetzt**
  - UND, ODER, NAND,...
- Schalterebene: Transistoren als Schalter
  - Elektrotechnische Grundlagen **Beispiel in M1 skizziert**
- Layoutebene: Transistortechnologie
  - Elektrotechnische Grundlagen **Beispiel in M1 skizziert**

Bottom-up

Gatterebene

- Abstrahierung von der internen Realisierung der Verknüpfungsbausteine
- Beschränkung auf das logische Verhalten
  - Abbildung logischer Funktionen auf Schaltungen ohne tiefergehende elektrotechnische Kenntnisse
- Verknüpfungsbausteine werden durch Schaltsymbole dargestellt

Schaltsymbole (DIN 40900 Teil 12, ANSI/IEEE-Standard 91-1984, IEC-Standard & US-Symbole)



Verknüpfungsbausteine dieser Art werden **Gatter** genannt.

Bedeutung der Zeichen

& : andere Schreibweise für $\wedge$

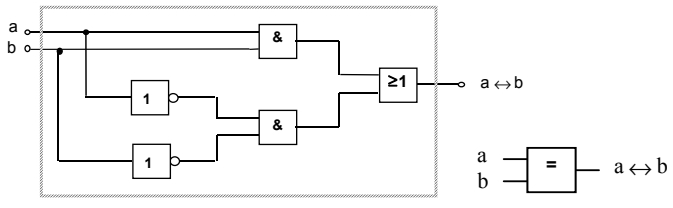
≥ 1 : der Ausgang ist genau dann 1, wenn an ≥ 1 Eingängen eine 1 liegt

° : Negation

Weitere Symbole, alte Darstellungen und die Logik hinter den Symbolen finden sich im Web!

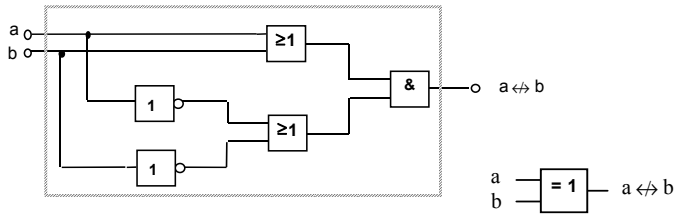
Äquivalenz-Gatter

Aus einfacheren Gattern lassen sich hierarchisch komplexere Gatter aufbauen, die teilweise eigene Symbole besitzen.



Äquivalenz als Komposition einfacherer Schaltglieder.

Antivalenz-Gatter



Antivalenz als Komposition einfacherer Schaltglieder.

Modul 3: Schaltnetze

- Einführung in die formalen Grundlagen logischer Beschreibungen
  - Boolesche Algebra, Schaltalgebra
- Voraussetzende technische Entwicklungen
- Realisierung von Schaltnetzen auf Schalter- und Gatterebene
- Entwurf von Schaltnetzen
  - Logikminimierung, KV-Diagramme
  - Programmierung von Funktionen
- Laufzeiteffekte bei Schaltnetzen

Entwurf von Schaltnetzen (1)

- Praktischer Entwurf von Schaltnetzen muß beachten, daß
  - reale Gatter keine idealen Verknüpfungen sind, sondern z.B. Wärme abgeben,
  - reale Gatter Platz benötigen (Quadratmicrometer),
  - reale Gatter Verzögerungszeiten besitzen (Microsekunden),
  - ...
- Teil 1: Technische Kriterien:
  - Leistungsaufnahme, Schaltzeit, Platzbedarf, Material, Lebensdauer
  - Korrekte Realisierung unter Beachtung des statischen und dynamischen Verhaltens der verwendeten Bauelemente (Hasards und Wettläufe)
  - Berücksichtigung technischer Beschränkungen, z.B. begrenzte Anzahl von Eingängen der Logikelemente, begrenzte Belastbarkeit von Elementausgängen

Entwurf von Schaltnetzen (2)

- Teil 2: Ökonomische Kriterien
  - Geringe Kosten für den Entwurf (Entwurfsaufwand), z.B. Lohnkosten, Kosten für Rechnerbenutzung zur Entwurfsunterstützung
  - Geringe Kosten für die Realisierung (Realisierungsaufwand), z.B. Kosten für die eingesetzten Bauelemente (bei der Herstellung integrierter Schaltkreise wird die Realisierung auf einer möglichst kleinen Chipfläche gefordert)
  - Geringe Kosten für Inbetriebnahme, laufenden Betrieb, z.B. Test, Wartung

Entwurf von Schaltnetzen (3)

- Grenzen zwischen technischen und ökonomischen Kriterien sind teilweise fließend.
- Einzelne Kriterien stehen auch im Widerspruch zueinander.
  - z.B. Erhöhung der Zuverlässigkeit ⇒ Erhöhung der Kosten
    geringere Realisierungskosten ⇒ höhere Entwurfskosten

➔ Aufgabe des Entwerfers besteht darin, für eine bestimmte Problemstellung den günstigsten Kompromiss zu finden.

- Ziel des Entwurfs: Unter Einhaltung bestimmter technischer Kriterien vor allem einen günstigen Kompromiss bezüglich der ökonomischen Kriterien anzustreben, um so zu einem Minimum an Gesamtkosten zu gelangen.

Minimierungsverfahren

- Grundlage:
  - Realisierung der Schaltnetze in zweistufiger Form
- Darstellungsform, deren Realisierung die geringsten Kosten verursacht, bezeichnet man als
  - Minimalform
- Der Vorgang der Erzeugung einer Minimalform wird als
  - Minimierung bezeichnet.
- Es gibt drei Arten von Minimierungsverfahren:
  - Algebraische Verfahren
  - Graphische Verfahren
  - Tabellarische Verfahren
- Algebraische und graphische Verfahren eignen sich nur für Funktionen mit bis zu 5/6 Variablen, danach werden sie zu unübersichtlich. Danach wendet man tabellarische Verfahren an.

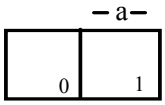
Praktische Einordnung

Aufgabenstellung	Primterme	Auswahl
Variablenanzahl ≤ 6	KV-Diagramm	KV-Diagramm Überdeckungstabelle
Geg. DF(KF) Ges. DMF(KMF) [Disj./Konj. Minimalform]	Consensus Quine-McCluskey	Überdeckungstabelle
Geg. DF(KF) Ges. KMF(DMF) [Disj./Konj. Form]	Nelson	Überdeckungstabelle

Für erweiterte/leistungsfähigere Verfahren wird auf die Literatur verwiesen!

Graphische Verfahren

- Das KV-Diagramm (nach Karnaugh und Veith)
  - Auch oft nur Karnaugh map, Karnaugh-Diagramm
- Ausgangspunkt ist ein Rechteck, dessen rechte Hälfte der Variablen a und dessen linke Hälfte $\bar{a}$ zugeordnet wird.



KV-Diagramm für eine Variable

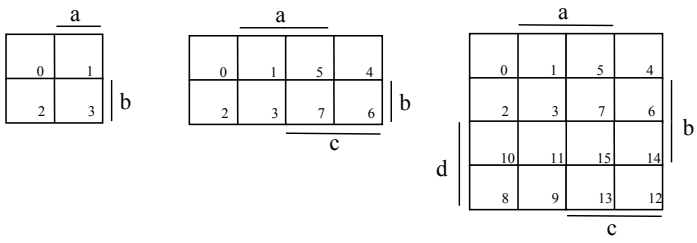
- Alternative Verfahren:
  - Quine McClusky über Indexmengen (maschinelle Unterstützung machbar)
  - Kasakow mittels Heuristiken für große Variablenmengen

KV-Diagramm

- Die Zahl in den Feldern gibt den Index der Variablenbelegung an:
  - Index des Minterms, der dort den Wert 1 annimmt.
- Durch Eintragen der Wahrheitswerte 0 oder 1 in die Felder des KV-Diagramms wird eine Boolesche Funktion charakterisiert.
- Das KV-Diagramm ist eine weitere Darstellungsform Boolescher Funktionen (Alternative zur Funktionstabelle).

KV-Diagramme

- KV-Diagramme für mehrere Variable erhält man durch Spiegelung (für jede neue Variable verdoppelt sich die Anzahl der möglichen Belegungen)



KV-Diagramme

- Jedes Feld hat eine eindeutige Variablenzuordnung, die an den Rändern abgelesen werden kann:
  - Feld 11 in Bild c hat die Zuordnung a b $\bar{c}$ d.
- Der Index in den Feldern gibt den Index der zum Feld gehörenden Variablenbelegung an (Variablen in umgekehrter alphabetischer Reihenfolge angetragen).
  - Feld 11 = 1011_2 = d $\bar{c}$ b a

KV-Diagramme (Erstellung)

- Funktion sei in Tabellenform gegeben:
  - Jede Zeile der Funktionstabelle entspricht einem Feld im KV-Diagramm.
  - ⇒ Für jede Zeile der Funktionstabelle sucht man das zugehörige Feld im KV-Diagramm und trägt den Funktionswert ein.
- Trick, um das Auffinden der Felder im KV-Diagramm zu erleichtern:
  - Man schreibt die Eingangsvariablen in „umgekehrter alphabetischer Reihenfolge“ in die Tabelle.
  - Dann kann das KV-Diagramm gemäß der Indizierung seiner Felder mit den Werten ausgefüllt werden, welche die Tabelle beim Durchlaufen von oben nach unten liefert.

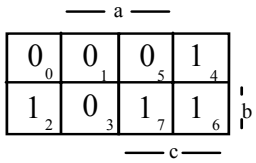
KV-Diagramme

Beispiel: $y = \bar{a}\, b \vee b\, c \vee \bar{a}\, \bar{b}\, c$

Funktionstabelle:

Index	c	b	a	y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

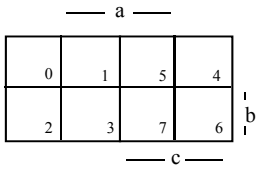
Damit ergibt sich das folgende KV-Diagramm:



Eigenschaften der KV-Diagramme (1)

- Wesentliche Eigenschaft:
  - Symmetrisch zu einer Achse liegende Minterme unterscheiden sich lediglich in einer Variablen.

- Beispiel:



Minterm 0 = $\bar{c}\, \bar{b}\, \bar{a}$
Minterm 1 = $\bar{c}\, \bar{b}\, a$

oder Minterm 0 = $\bar{c}\, \bar{b}\, \bar{a}$
Minterm 4 = $c\, \bar{b}\, \bar{a}$

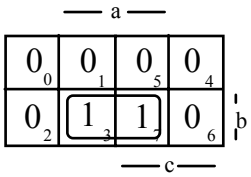
oder Minterm 1 = $\bar{c}\, \bar{b}\, a$
Minterm 3 = $\bar{c}\, b\, a$

Eigenschaften der KV-Diagramme (2)

- Nach den Regeln der Booleschen Algebra lassen sich Terme, die sich nur in einer Variablen unterscheiden, zusammenfassen:
- Beispiel:

$$a\, b\, c \vee a\, b\, \bar{c} = a\, b\, (c \vee \bar{c}) = a\, b$$

- Es entsteht ein Term ohne diese Variable.
- Symmetrisch zu den Achsen des KV-Diagramms liegende Minterme lassen sich zu einem einfacheren Term zusammenfassen.



Definition: Primimplikant

- Ein Implikant p ist Primimplikant, falls es keinen Implikanten $q \neq p$ gibt, der von p impliziert wird

$$\forall q:\ q \neq p \Leftrightarrow (p \rightarrow q)$$

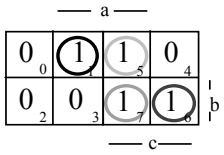
- *d.h.*, p ist von größtmöglicher Ordnung (p umfasst einen maximal großen Einsblock).
- Es gilt:
  Jede Funktion ist als Disjunktion ihrer Primimplikanten darstellbar.

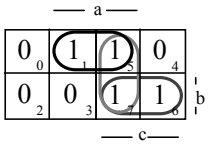
Herauslesen der Primimplikanten

- Herauslesen der Primimplikanten aus dem KV-Diagramm:
  - Man versucht, möglichst große Blöcke von Einsen im Diagramm zu finden, wobei jeder Einsblock 2^k Felder umfassen muß.

Beispiel:
$f = \bar{a}\, b\, c \vee a\, c \vee a\, \bar{b}\, \bar{c}$

Beispiel





Vier Minterme:
($a\, \bar{b}\, \bar{c}$, $a\, \bar{b}\, c$, $a\, b\, c$, $\bar{a}\, b\, c$)

Drei Primimplikanten:
Implikanten erster Ordnung
($a\, \bar{b}$, $a\, c$, $b\, c$)

Aber: ($a\, \bar{b}$, $b\, c$) genügen eigentlich!

- $f = a\, \bar{b}\, \bar{c} \vee a\, \bar{b}\, c \vee a\, b\, c \vee \bar{a}\, b\, c$
  $= a\, \bar{b} \vee a\, c \vee b\, c = a\, \bar{b} \vee b\, c$

Herauslesen einer Funktion

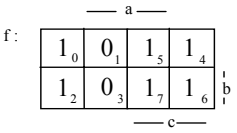
Auffrischung:

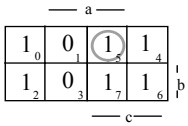
- Definition:
  Es sei $D(x_1,...,x_m)$ eine Disjunktion von Literalen
  $\bigvee L_i = L_1 \vee \ldots \vee L_m$ oder die Konstante „0“ oder „1“
- Der Term $D(x_1,...,x_m)$ heißt Implikat einer Booleschen Funktion $y(x_1,...,x_m)$ [f Funktion von x_i],
  wenn $D \rightarrow y$ [y Produktterm aus Literalen dieser x_i]
- Das heißt für jede Belegung $B \in \{0,1\}^n$ gilt:
  Wenn D(B) = 0, dann ist auch y(B) = 0.

- y ist Implikant von y, wenn y die Funktion y impliziert, d.h. wenn die Menge aller Einsstellen von y in der Menge aller Einsstellen von y enthalten ist.

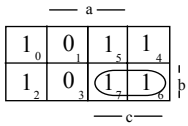
Beispiel (1)

Beispiel: f = Minterme (0,2,4,5,6,7)

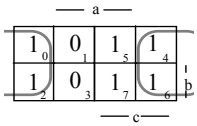




$g = a\,\bar{b}\,c$
ist Implikant



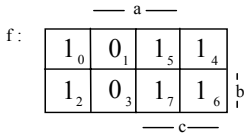
$g = b\,c$
ist Implikant

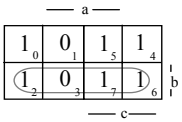


$g = \bar{a}$
ist Implikant

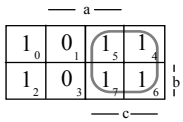
Beispiel (2)

Beispiel: f = Minterme (0,2,4,5,6,7)

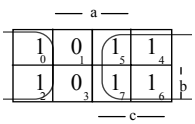




g = b
ist KEIN Implikant



g = c
ist Implikant



$g = \bar{a} \vee c$
ist KEIN Implikant

Implementation von Funktionen

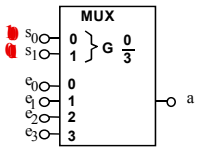
- Implementation regelmäßig wiederkehrender Funktionen.

- Ein Multiplexer (Abkürzung: MUX) ist ein Baustein mit mehreren Eingängen und einem Ausgang, wobei über n Steuerleitungen einer der 2^n Eingänge auf den Ausgang geschaltet wird.

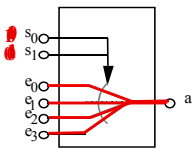
- Multiplexer werden nach ihrer Größe als 2^n:1 - Multiplexer (alternativ als 1-aus-2^n- Multiplexer) klassifiziert.

1-aus-4-Multiplexer

- Schaltbild und logisches Verhalten:



s_1	s_0	a
0	0	e_0
0	1	e_1
1	0	e_2
1	1	e_3



Logische Funktionen mit Multiplexern

- Ein Multiplexer kann nicht nur zur Steuerung von Datenflüssen sondern auch zur Realisierung logischer Funktionen verwendet werden.

- Man kann mit einem 2^n: 1 - Multiplexer eine logische Funktion mit n+1 Variablen implementieren.

- Hierzu wird die sog. Implementierungstabelle verwendet.

Implementierungstabelle

- Die Tabelle besteht aus:
  - 2^n Spalten für die möglichen Belegungen der n Steuereingänge
  - 2 Zeilen für die negierte und nicht negierte (n+1)-te Variable
- In die Tabelle werden die Funktionswerte in Abhängigkeit von den Variablen eingetragen.
- Anschließend betrachtet man jede Spalte für sich und ordnet ihr eine einstellige Funktion g ∈ {0, 1, a, $\bar{a}$} zu, mit der dann der Eingang belegt wird, der zu der entsprechenden Steuervariablenkombination gehört.

Beispiel

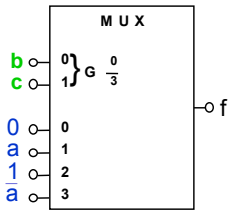
Realisierung einer Funktion mit Multiplexer:

$$f = \bar{a}\,c \vee \bar{b}\,c \vee a\,b\,\bar{c}$$

Implementierungstabelle bei Wahl von b und c als Steuereingänge:

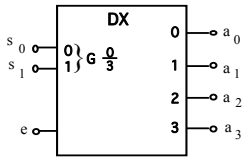
c b	0 0	0 1	1 0	1 1
a = 0	0	0	1	1
a = 1	0	1	1	0
f =	0	a	1	$\bar{a}$

Realisierung



Demultiplexer/Dekoder (1)

- Der zum Multiplexer korrespondierende Baustein, der einen Eingang abhängig von n Steuerleitungen auf einen von 2^n Ausgängen schaltet, heißt Demultiplexer.
- Beispiel:



s_1	s_0	a_0	a_1	a_2	a_3
0	0	e	0	0	0
0	1	0	e	0	0
1	0	0	0	e	0
1	1	0	0	0	e

Schaltbild und logisches Verhalten eines 1-auf-4-Demultiplexers

Demultiplexer/Dekoder (2)

- Man beachte:
  - der Demultiplexer hat einen Enable-Eingang e sowie n Eingänge s_i für eine Dualzahl, die an den 2^n Ausgängen a_j dekodiert bereitgestellt wird.
- Enable-Eingang e = 0, dann liegen alle Ausgänge auf 0 ansonsten wird eine 2-bit-Zahl dekodiert, z.B. wird bei Anlegen der Zahl 2 (s_1= 1, s_0= 0) der Ausgang a_2= 1 und alle anderen Ausgänge bleiben 0.
- Der Demultiplexer wird deshalb auch Dekoder genannt.

Realisierung mittels Speicherbausteinen

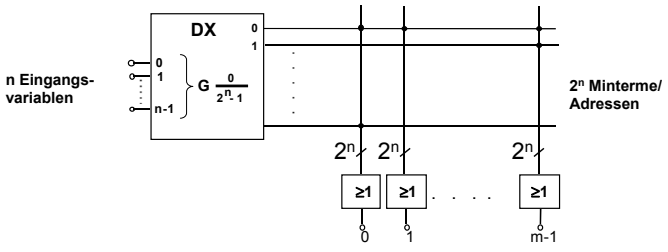
- Bei den bisher behandelten Bausteinen (Gatter, Multiplexer, Dekoder) war die Funktion fest vorgegeben.

  ➔ festverdrahtete Logik
- Höherintegrierte Verknüpfungsbausteine müssen die Flexibilität bieten, an viele verschiedene Anwendungen anpaßbar zu sein. Diese Anpassung wird als Personalisierung oder als Programmierung bezeichnet.

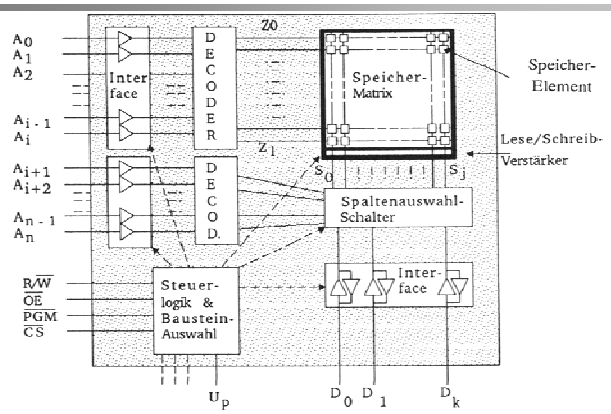
  ➔ mikroprogrammierte Logik
  (siehe M6 für entsprechende CPUs)

Schematischer Aufbau eines Speicherbausteins

Speicheranordnung, in der beliebige Funktionstabellen abgelegt werden.



Organisation von Speicherbausteinen



Erläuterung zum Speicherbaustein

- Durch das Anlegen von Eingangssignalen wird eine Speicherzelle ausgewählt (adressiert) und der dort gespeicherte Funktionswert an den Ausgängen zur Verfügung gestellt.
- Die Leitungen, die den Dekoder verlassen, entsprechen also den Mintermen von n Eingangsvariablen, also den Zeilen der Funktionstabelle.
- Das Speichern einer 1 für eine bestimmte Ausgangsvariable i bedeutet, daß dieser Minterm in die ODER-Verknüpfung am i-ten Ausgang einbezogen wird, eine 0 heißt, daß der Minterm nicht benutzt wird.

Speichertypen (1)

- Je nach Personalisierung des Speicherbausteins unterscheidet man verschiedene Speichertypen.
- ROM (Read Only Memory):
- Speicherbausteine, auf die nur lesend zugegriffen werden kann. Programmierung beim Hersteller (maskenprogrammierbare ROMs), wird während der ganzen Lebenszeit des Bausteins beibehalten.

Speichertypen (2)

- PROM (Programmable Read Only Memory):
- Programmierbare ROMs, die erst vom Benutzer programmiert werden.
- Programmierung:
  - Durchbrennen von mikroskopisch kleinen Verbindungen (engl.: fusible link)
  - Auf den Baustein aufgebrachte Ladungen, die über Jahre hinweg durch physikalische Prozesse festgehalten werden (engl.: stored charge).

Speichertypen (3)

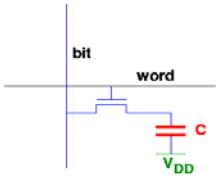
- EPROM (Erasable Programmable Read Only Memory):
- Ein benutzerprogrammierbarer Speicherbaustein, durch UV-Licht wieder löschbar und dann neu programmierbar.
- Quarzfenster auf der Bausteinoberfläche.
- Es gibt auch Speicherbausteine, die elektrisch (durch das Anlegen höherer Spannungen) gelöscht werden können
  - EEPROM: Electrically Erasable PROM
  - EAROM: Electrically Alternable ROM.
  - Anwendung z.B. Speicher bei Digitalkameras (Compact Flash, Memory Stick)

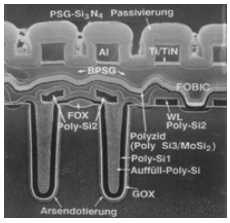
Speichertypen (4)

- RAM (Random Access Memory):
- Speicher, auf die während des Normalbetriebs lesend und schreibend zugegriffen werden kann.
- Ein RAM-Baustein verliert seine Programmierung, wenn er von der Spannungsversorgung abgetrennt wird.
- Man unterscheidet :
  - Dynamische RAM-Bausteine (DRAM)
  - Statische RAM-Bausteine (SRAM).

Dynamische RAM-Bausteine (DRAM)

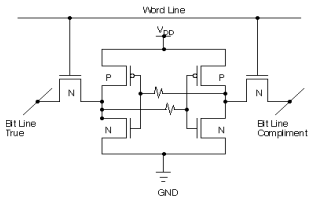
- Ein Kondensator dient als Ladungsspeicher, und ein Transistor wird zum Ankoppeln an diesen Ladungsspeicher benötigt.
- Der Speicherinhalt muß in regelmäßigen Zeitabständen „aufgefrischt" werden (memory refresh).

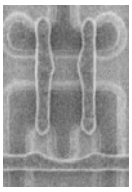


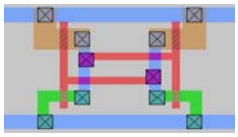


Statische RAM-Bausteine (SRAM)

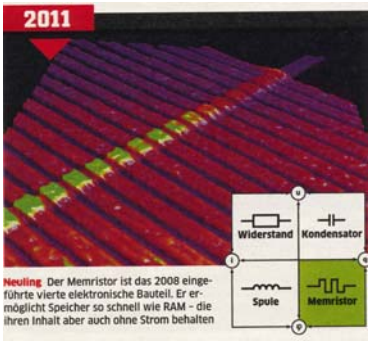
- Als Speicherzelle wird ein Flipflop verwendet (Erklärung folgt in Modul 4).
  - Die Speicherzelle hält ihre Programmierung auch ohne Regeneration.
  - Die Zugriffszeit bei einem statischen RAM ist wesentlich kürzer als bei einem dynamischen RAM.
  - Eine statische Speicherzelle benötigt etwa 6 bis 8 Transistoren, eine dynamische dagegen deutlich weniger (z.B. ein Transistor).







Memresistor



Gegenüberstellung von RAMs und ROMs

- Unterschiede zwischen RAMs und ROMs betreffen vor allem:
  - Lese-/Schreib-Möglichkeiten (RW; *read-write*)
  - Zugriffszeiten (ZZ; für R/W)
  - Speicherpermanenz ohne Spannungsversorgung (SP)
  - Realisierbare Speichergröße (SG)

	ROM	PROM	EPROM	Flash-PROM	DRAM	SRAM
RW	R	R	R ((W))	R (W)	RW	RW
ZZ	+	+	+ / –	+ / –	++	+++
SP	+	+	+	+	–	–
SG	+	+	+	+	+	+

PLA (Programmable Logic Array)

- Bisher wurde die gesamte Funktionstabelle in einem Speicherbausteinen abgespeichert und die Funktion durch ihre DNF (Disjunktive Normalform) realisiert.
- Verwendet man stattdessen die DMF (Disjunktive Minimalform), lassen sich Funktionen oft sehr viel kompakter darstellen.
- PLA (Programmable Logic Array):
- Im Unterschied zum ROM werden bei PLA eingangsseitig nicht Minterme, sondern Primimplikanten der Minimalüberdeckung erzeugt.
- Dazu wird der Dekoder durch eine UND-Matrix ersetzt.

FPLA und PAL

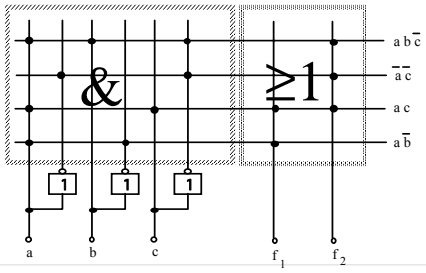
- PLAs werden ähnlich wie ROMs bereits bei der Herstellung personalisiert.
- Ein vom Benutzer zu programmierendes PLA mit fest vorgegebener Anzahl von Eingangsvariablen n, Produkttermen k und Ausgangsvariablen m wird FPLA (field programmable logic array) genannt.
- Alternativ dazu werden PAL-Bausteine (programmable array logic) angeboten, bei denen die UND- bzw. ODER-Matrix bereits in der Herstellung personalisiert wurde.

Beispiel: PAL-Realisierung

Die (schon minimierten) Funktionen

$$f_1 = a\,\bar{b} \vee a\,c \quad \text{und} \quad f_2 = \bar{a}\,\bar{c} \vee a\,c \vee a\,b\,\bar{c}$$

sollen mit einem PAL realisiert werden:



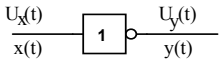
Modul 3: Schaltnetze

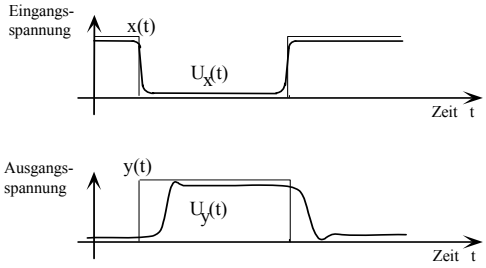
- Einführung in die formalen Grundlagen logischer Beschreibungen
  - Boolesche Algebra, Schaltalgebra
- Voraussetzende technische Entwicklungen
- Realisierung von Schaltnetzen auf Schalter- und Gatterebene
- Entwurf von Schaltnetzen
  - Logikminimierung, KV-Diagramme
  - Programmierung von Funktionen
- Laufzeiteffekte bei Schaltnetzen

Laufzeiteffekte

- Auf der Gatterebene wurden die Gatter bisher als ideale logische Verknüpfungen betrachtet.
- In der Realität werden Gatter jedoch z.B. mittels Transistoren, Widerstände, Kapazitäten realisiert (Layoutebene).
- Der zeitliche Signal-Verlauf eines realen Gatters weicht vom Verlauf der idealen booleschen Größen ab.

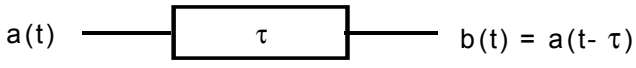
Realer und idealer Signalverlauf (Inverter)





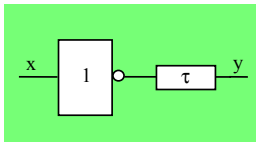
Realistischere Beschreibung von Gattern

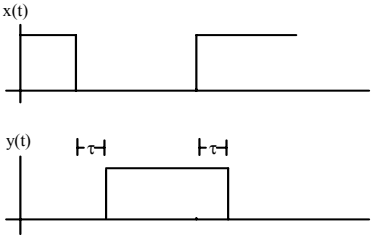
- Um die Effekte auf der Gatterebene annähernd zu beschreiben, gibt es eine Reihe verschieden komplexer Modelle.
- Einfaches Modell: Totzeitmodell
  - Es werden lediglich die durch Gatter und Leitungen entstehenden Totzeiten berücksichtigt.
  - Ein reales Verküpfungsglied (Gatter) wird modelliert durch:
    - Ein ideales Verknüpfungsglied ohne Verzögerungsanteil und
    - Ein Totzeitglied als reines Verzögerungsglied (steht für die Schaltzeit des Gatters und ggf. für Leitungsverzögerungen).
- Das zeitliche Verhalten einer binären Größe hinter einem Totzeitglied ist dasselbe wie dasjenige vor dem Totzeitglied, aber um die Zeit τ versetzt:



Beispiel: Totzeitmodell eines Inverters

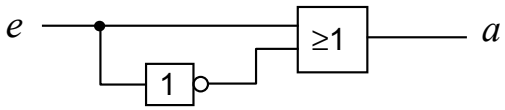
- Mit Hilfe dieses einfachen Modells lassen sich Laufzeiteffekte bereits sehr gut modellieren (auch wenn dieses Modell noch sehr idealisierend ist!).





Beispiel: Inverteranwendung

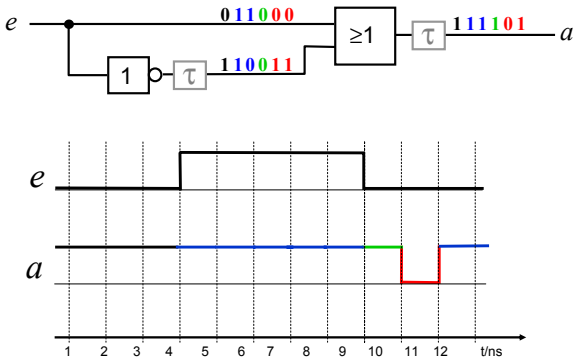
Gegeben:



$a = e \vee \overline{e} = 1$

Beide Gatter haben eine Verzögerungszeit von 1 ns.

Zeit-Diagramm



Verhalten eines Schaltnetzes bei Änderung der Eingabebelegung (1)

- Ideales Schaltnetz:
- Das Ausgangssignal ändert sich nicht, wenn alte und neue Belegung denselben logischen Verknüpfungswert liefern.
- Das Ausgangssignal ändert sich genau einmal, wenn alte und neue Belegung verschiedene logische Verknüpfungswerte liefern.

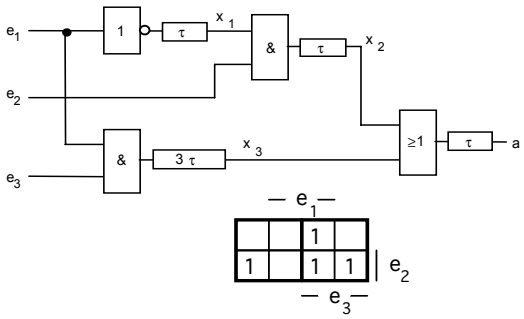
Verhalten eines Schaltnetzes bei Änderung der Eingabebelegung (2)

- Reales Schaltnetz:
- Die Änderung läuft auf verschieden langen Wegen mit verschiedenen Verzögerungen durch das Schaltnetz.
- Mehrfache Änderungen des Ausgangssignals sind möglich, bis sich der stabile Endwert einstellt

  ➔ Hasardfehler

Beispiel

Funktion: $a = \overline{e}_1 e_2 \vee e_1 e_3$



Eingabewechsel

- Es sollen die folgenden Eingabewechsel betrachtet werden:

a) Die Eingänge e_2 und e_3 seien konstant 1, der Eingang e_1 wechsle von 0 auf 1

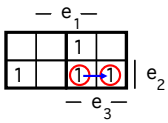
b) Die Eingänge e_2 und e_3 seien konstant 1, der Eingang e_1 wechsle von 1 auf 0

Funktion: $a = \bar{e}_1 e_2 \vee e_1 e_3$

Funktionswerte bei den Übergängen:

$(e_3,e_2,e_1) = (1,1,0)$ $\Rightarrow$ a = 1

$(e_3,e_2,e_1) = (1,1,1)$ $\Rightarrow$ a = 1

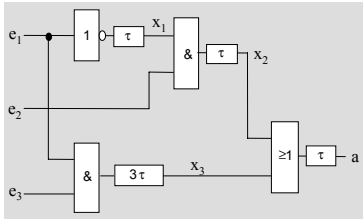


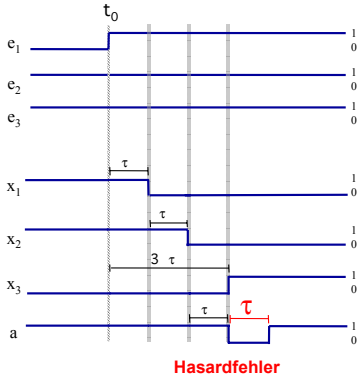
$\Rightarrow$ korrektes Verhalten bei den Übergängen.

Bei beiden Übergängen darf sich der Wert von a nicht ändern.
Er muß **konstant 1** bleiben.

Genau dieses Verhalten kann jedoch nicht garantiert werden !

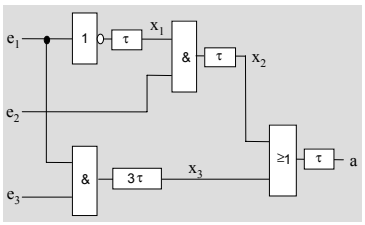
Das Verhalten anhand des Totzeitmodells

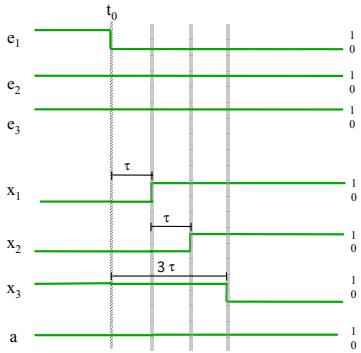




Hasardfehler

Das Verhalten anhand des Totzeitmodells





kein Hasardfehler

Ergebnis

- Beim Wechsel e_1 von 0 auf 1 liefert das Ausgangssignal nicht ständig den korrekten Funktionswert

➔ Hasardfehler

- Beim Wechsel e_1 von 1 auf 0 ist das Ausgangssignal hingegen korrekt

Begriffe: Eingabewechsel, Übergang

- Definition:
  Ein Eingabewechsel ist die Änderung einer oder mehrerer Eingangsvariablen zu einem bestimmten Zeitpunkt.
  - Falls sich mehrere Eingangsvariablen ändern sollen, so müssen sie dies gleichzeitig tun.

- Definition:
  Ein Übergang ist der Vorgang im Schaltnetz, der vom Eingabewechsel ausgelöst wird. Er beginnt mit dem Eingabewechsel und endet mit dem Eintreten des neuen Ruhezustandes.

Begriffe: Hasardfehler - Hasard

- Definition:
  Ein Hasardfehler ist eine mehrmalige Änderung der Ausgangsvariablen während eines Übergangs.

- Definition:
  Ein Hasard ist die durch das Schaltnetz gegebene logisch-strukturelle Vorbedingung für einen Hasardfehler, ohne Berücksichtigung der konkreten Verzögerungswerte.

Hasardbehaftete Übergänge (1)

- Jeder Hasard ist eine Eigenschaft eines bestimmten Überganges im Schaltnetz.
- Zur Betrachtung, ob ein bestimmter Übergang hasardbehaftet ist oder nicht, interessiert nur:
  - Die logische Funktion, die durch das Schaltnetz realisiert wird.
  - Die Struktur des Schaltnetzes, d.h. die Anzahl, die Verknüpfungsfunktionen und die genaue Anordnung der Gatter zur Realisierung der Funktion, nicht jedoch die tatsächlichen Verzögerungswerte der verwendeten Gatter.

Hasardbehaftete Übergänge (2)

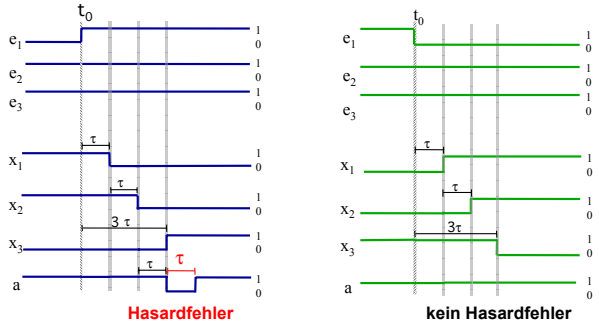
- Tritt in einem konkreten Schaltnetz bei einem bestimmten Übergang ein Hasardfehler auf, so ist dieser Übergang hasardbehaftet, also:

  Hasardfehler ➔ Hasard

- Die Umkehrung gilt jedoch nicht: Ist ein Übergang hasardbehaftet, so folgt hieraus nicht notwendigerweise das Eintreten eines Hasardfehlers.

  Hasard ∧ ungünstige Verzögerungswerte ➔ Hasardfehler

Beispiel 1



Der Übergang (e_3,e_2,e_1) : (1,1,0) → (1,1,1) ist hasardbehaftet, da es die Möglichkeit zu einem Hasardfehler gibt.

Funktionshasard

- Definition:
  Ein Funktionshasard ist ein Hasard, dessen Ursache in der zu realisierenden Funktion liegt.
- Er tritt in jedem möglichen Schaltnetz für diese Funktion auf. Er kann nicht behoben werden.

⇒ Für ein konkretes Schaltnetz mit Funktionshasard kann zwar der Funktionshasardfehler durch günstige Wahl der Verzögerungswerte behoben werden, nicht jedoch der Hasard selbst.

Strukturhasard

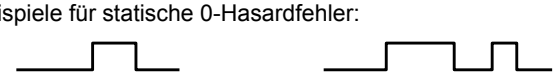
- Definition:
  Ein Strukturhasard ist ein Hasard, dessen Ursache in der Struktur des realisierten Schaltnetzes liegt.
- Ein Strukturhasard kann deshalb immer durch Änderung der Schaltnetzstruktur bei gleicher Schaltnetzfunktion behoben werden.

⇒ Es ist grundsätzlich möglich, ein anderes Schaltnetz zu entwerfen, welches dieselbe Funktion realisiert und den Strukturhasard beseitigt.

Statischer 0-Hasard

- Analog zu den Übergängen werden die Hasards als statisch bzw. dynamisch bezeichnet, je nachdem, bei welcher Art von Übergang sie auftreten.
- Ein Hasard in einem statischen 0-Übergang heißt statischer 0-Hasard.

Beispiele für statische 0-Hasardfehler:



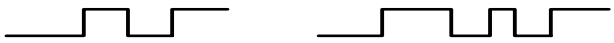
Statischer 1-Hasard

- Ein Hasard in einem statischen 1-Übergang heißt statischer 1-Hasard.
- Beispiele für statische 1-Hasardfehler:

Der Übergang (1,1,0) → (1,1,1) im Beispiel enthält also einen statischen 1-Hasard.

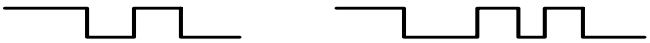
Dynamischer 01-Hasard

- Ein Hasard in einem dynamischen 01-Übergang heißt dynamischer 01-Hasard.
- Beispiele für dynamische 01-Hasardfehler:

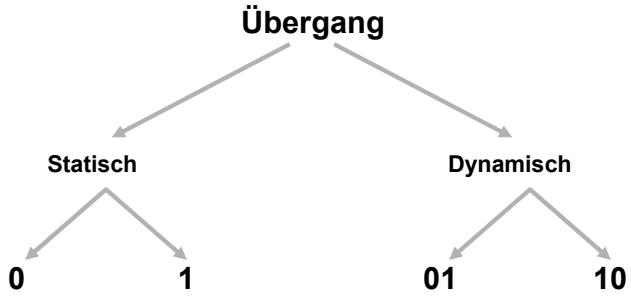


Dynamischer 10-Hasard

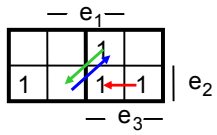
- Ein Hasard in einem dynamischen 10-Übergang heißt dynamischer 10-Hasard.
- Beispiele für dynamische 10-Hasardfehler:



Klassifizierung von Hasards



Übergangsbeispiele



Statischer 1-Übergang:

Übergang (e_3,e_2,e_1): (1,1,0) → (1,1,1)

Dynamischer 01-Übergang:

Übergang (e_3,e_2,e_1): (0,1,1) → (1,0,1)

Dynamischer 10-Übergang

Übergang in umgekehrter Richtung: (1,0,1) → (0,1,1)

Klassifizierung von Laufzeiteffekten

Zum Erkennen von / benötigt man	Funktion des Schaltnetzes	Struktur des Schaltnetzes (daher auch Funktion)	konkrete Verzögerungen der Gatter zur gegebenen Struktur des Schaltnetzes
Funktions-hasards			
Struktur-hasards			
Funktions-hasardfehler			
Struktur-hasardfehler			

Die dunkelgrauen Felder markieren die notwendigen Informationen.
Die hellgrauen Felder markieren die daraus folgenden Informationen.