# Digitale Schaltungen

## 1. Moore'sches Gesetz

- alle 18-24 Monate verdoppelt sich die Anzahl der Transistoren auf gleicher Fläche
- Exponentielles Wachstum der Transistorzahl, exponentieller Rückgange des Preises pro Transistor
- Herstellungskosten (Fixkosten, Variable Kosten, Technologiefaktor), Entwicklerproduktivität, Verlustleistungsdichte

## 2. Einheiten

| Potenz | Vorsatz | Potenz | Vorsatz |
|---|---|---|---|
| $10^{12}$ | T | $10^{-1}$ | d |
| $10^{9}$ | G | $10^{-2}$ | c |
| $10^{6}$ | M | $10^{-3}$ | m |
| $10^{3}$ | k | $10^{-6}$ | $\mu$ |
| $10^{2}$ | h | $10^{-9}$ | n |
| $10^{1}$ | da | $10^{-12}$ | p |
| | | $10^{-15}$ | f |

| | |
|---|---|
| $Hz$ | $s^{-1}$ |
| N | $kgms^{-2}$ |
| $J$ | $Nm = VAs$ |
| $W$ | $VA = Js^{-1}$ |
| $C$ | $As$ |
| $V$ | $JC^{-1}$ |
| $F$ | $CV^{-1}$ |
| $\Omega$ | $VA^{-1}$ |
| $H$ | $VsA^{-1}$ |

Bit $\xrightarrow{\cdot 8}$ Byte $\xrightarrow{\cdot 1024}$ kByte $\xrightarrow{\cdot 1024}$ MByte

## 3. Polyadische Zahlensysteme

| | | | | | |
|---|---|---|---|---|---|
| $Z$ | Zahl | $p$ | #Ziffern vorne | $d_i$ | Ziffer |
| $r$ | Basis | $n$ | #Nachkommastellen | | |

$$Z = \sum_{i=-n}^{p-1} r^i \cdot d_i = d_{p-1}...d_1 d_0 . d_{-1}...d_n$$

Binäres Zahlensystem:

$d_{i2} \in 0, 1 \qquad B = \sum_{i=-n}^{p-1} 2^i \cdot d_i \qquad d_{-n} : LSB; \quad d_{p-1} : MSB$

Octalsystem: $d_{i8} \in 0, 1, 2, 3, 4, 5, 6, 7$

Hexadezimalsystem: $d_{i16} \in 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F$

Benötigte Bits: $N : n$ Bit $M : 2n$ Bit $\quad N + M = 2n + 1$ Bit $\quad N \cdot M = 3n$ Bit

### 3.1. Umrechnung

| | $Z \geq 1$ | $Z < 1$ |
|---|---|---|
| $r \to 10$ | $Z_{10} = \sum r^i \cdot d_i$<br>$101_2 \to 1 \cdot 1 + 0 \cdot 2 + 1 \cdot 4$ | $Z_{10} = \sum r^{-i} \cdot d_{-i}$<br>$0.11_2 \to 1 \cdot 0.5 + 1 \cdot 0.25$ |
| $10 \to r$ | $d_i = Z_{10} \% r^i$<br>$58/8 = 7$Rest $2(LSB)$<br>$7/8 = 0$ Rest $7(MSB)$ | <br>$0.4 \cdot 2 = 0.8$ Übertrag $0(MSB)$<br>$0.8 \cdot 2 = 1.6$ Übertrag $1$ |

### 3.2. Zweierkomplement

**Wertebereich:** $-2^{n-1} \leq Z \leq 2^{n-1} - 1$

Wandle 2 in -2 um:

1. Invertieren aller Bits: $0010 \Rightarrow 1101$
2. Addition von 1: $1101 + 1 = 1110$
3. Ignoriere Überträge beim MSB: $\Rightarrow \; -2_{10} = 1110_2$

### 3.3. Gleitkommadarstellung nach IEEE 754

Wert $= (-1)^s \cdot 2^{e-127} \cdot 1.f$

$s$: Vorzeichen, $e$: Exponent, $f$: Mantisse

Bsp: $-0.625 = -1 \cdot 2^{-1} \cdot 1.01_2$

$\Rightarrow \; s = 1, e = 126, f = 01_2$

Spezialwerte: Wert $= 0 \Leftrightarrow e = 0 \qquad Wert = \infty \Leftrightarrow e = 255$

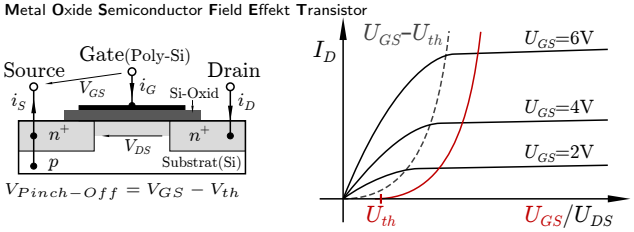
| Bitverteilung(single/double): | $s(1)$ | $e(8/11)$ | $f(23/52)$ |
|---|---|---|---|

## 4. Halbleiter

| | Isolator | Metall | undotiert | N-Typ | P-Typ |
|---|---|---|---|---|---|
| Ladungsträger | Keine | $e^-$ | $e^-/e^+$ | $e^-$ | $e^+$ |
| Leitfähigkeit | Keine | Sehr hoch | $\propto T$ | Hoch | Mittel |

## 5. MOS-FET's

**M**etal **O**xide **S**emiconductor **F**ield **E**ffekt **T**ransistor




$V_{Pinch-Off} = V_{GS} - V_{th}$

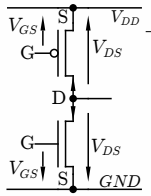
### 5.1. Bauteilparameter

Verstärkung: $\beta = K' \frac{W}{L}$ mit $K' = \frac{\mu \epsilon_{0x} \epsilon_0}{t_{0x}}$

| | |
|---|---|
| Kanalweite | W |
| Kanallänge | L |
| Elektronenbeweglichkeit | $\mu \quad \mu_n \approx 250 \cdot 10^{-4} \frac{m^2}{Vs}, \mu_p \approx 100 \cdot 10^{-4} \frac{m^2}{Vs}$ |
| rel. Dielektrizität des Gateoxyds | $\epsilon_{ox} \approx 3,9$ |
| Dielektrizitätskonstante | $\epsilon_0 = 8.8541878 \cdot 10^{-12} \frac{\mathrm{A\,s}}{\mathrm{V\,m}}$ |
| Gateoxyddicke | $t_{ox}$ |
| Verstärkung | $\beta = \frac{\mu_n \varepsilon_{ox} \varepsilon_0}{t_{ox}} \cdot \frac{W}{L} = K' \frac{W}{L} = \frac{\mu_n C_G}{L^2}$ |
| Kapazität | $C_G = \varepsilon_{ox} \varepsilon_0 \frac{WL}{t_{ox}}$ |
| Verzögerungszeit | $t_{pHL} \propto \frac{C_L t_{ox} L_p}{W_p \mu_p \varepsilon_{ox} (V_{DD} - \lvert V_{th} \rvert)}$ |

- große Kanalweite $\Rightarrow$ große Drain-Störme
  $\Rightarrow$ schnelle Schaltgeschwindigkeit (da $i_d \propto \beta \propto \frac{W}{L}$)
  Aber: große Fläche.
- nMos schaltet schneller als pMOS

### 5.2. pMos und nMos




| Transistor | Source liegt immer am | $V_{GS}, V_{DS}, I_D$ | Substrat |
|---|---|---|---|
| pMos<br>normally on | höheren Potential | $< 0$ | $+(V_{DD})$ |
| nMos<br>normally off | niedrigeren Potential | $> 0$ | $-(GND)$ |

Drainstromformel:
nMos (p-dotiertes Substrat, n-dotierte Drain/Source), schlechter pull up (Pegeldegenerierung)

$$I_d = \begin{cases} 0, & \text{für } U_{gs} - U_{th} \leq 0 \quad \text{(Sperrber.)} \\ \beta[(u_{gs} - U_{th}) \cdot u_{ds} - \frac{1}{2} u_{ds}^2], & \text{für } 0 \leq U_{gs} - U_{th} \geq u_{ds} \quad \text{(linearer Ber.)} \\ \frac{1}{2} \beta \cdot (u_{gs} - U_{th})^2, & \text{für } 0 \leq U_{gs} - U_{th} \leq u_{ds} \quad \text{(Sättigungsber.)} \end{cases}$$

Drainstromformel:
pMos (n-dotiertes Substrat, p-dotierte Drain/Source), schlechter pull down (Pegeldegenerierung)

$$I_d = \begin{cases} 0, & \text{für } U_{gs} - U_{th} \geq 0 \quad \text{(Sperrber.)} \\ -\beta[(u_{gs} - U_{th}) \cdot u_{ds} - \frac{1}{2} u_{ds}^2], & \text{für } 0 \geq U_{gs} - U_{th} \leq u_{ds} \quad \text{(linearer Ber.)} \\ -\frac{1}{2} \beta \cdot (u_{gs} - U_{th})^2, & \text{für } 0 \geq U_{gs} - U_{th} \geq u_{ds} \quad \text{(Sättigungsber.)} \end{cases}$$

### 5.3. Disjunktive Normalform (DNF/SOP)

Eins-Zeilen der Wertetabelle ODER verknüpfen:
$Z = \overline{A} \cdot \overline{B} + \overline{C} \cdot D$

### 5.4. Konjunktive Normalform (KNF/POS)

Null-Zeilen der Wertetabelle negieren und UND verknüpfen:
$Z = (\overline{A} + \overline{C}) \cdot (\overline{A} + \overline{D}) \cdot (\overline{B} + \overline{C}) \cdot (\overline{B} + D)$

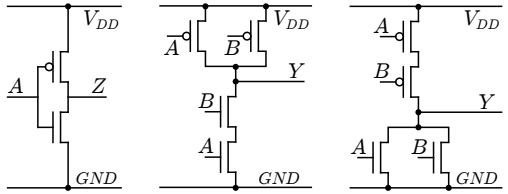
### 5.5. Umwandlung in jeweils andere Form

**Vorgehen**

- Doppeltes Negieren der Funktion: $Z = \overline{\overline{\overline{A} \cdot \overline{B} + \overline{C} \cdot D}}$
- Umformung "untere" Negation (DeMorgan) : $Z = \overline{\overline{\overline{A} \cdot \overline{B}} \cdot \overline{\overline{C} \cdot D}} = \overline{(A + B) \cdot (C + \overline{D})}$
- Ausmultiplizieren: $Z = \overline{(A + B) \cdot (C + \overline{D})} = \overline{A \cdot C + A \cdot \overline{D} + B \cdot C + B \cdot \overline{D}}$
- Umformung "obere" Negation (DeMorgan) :
  $Z = \overline{AC} \cdot \overline{A\overline{D}} \cdot \overline{BC} \cdot \overline{B\overline{D}} = (\overline{A} + \overline{C}) \cdot (\overline{A} + D) \cdot (\overline{B} + \overline{C}) \cdot (\overline{B} + D)$
- Analog von KNF nach DNF.

## 6. CMOS - Logik

Komplementäre Logik liefert grundsätzlich negierte Ausgänge. $\Rightarrow$ NAND einfacher als AND.
Drei Grundgatter der CMOS-Technologie:




Falls $GND$ und $V_{DD}$ vertauscht würden, dann $NAND \to AND$ und $NOR \to OR$
Allerdings schlechte Pegelgenerierung.

### 6.1. Gatterdesign

**Vorteil:** (Fast) nur bei Schaltvorgängen Verlustleistung - wenig statische Verluste

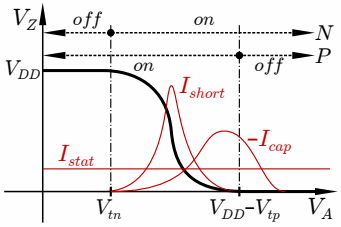
| Netzwerk | Pull-Down | Pull-Up |
|---|---|---|
| Transistoren | **n**Mos | **p**Mos |
| AND | Serienschaltung | Parallelschaltung |
| OR | Parallelschaltung | Serienschaltung |

1. Möglichkeit: Direkt; ggf. Inverter vor die Eingänge und Ausgänge schalten.
2. Möglichkeit: Mit bullshit Algebra die Funktion nur mit NAND und NOR darstellen.


von Emanuel Regnath (Emanuel.Regnath@tum.de), Martin Zellner (martin.zellner@mytum.de) und Hendrik Böttcher (hendrik.boettcher@tum.de)

### 6.2. CMOS Verlustleistung

Inverterschaltvorgang $V_A : 0 \rightarrow 1$:




| | |
|---|---|
| **Dynamische Verlustleistung** | $P_{\text{dyn}} = P_{\text{cap}} + P_{\text{short}}$ |
| Kapazitive Verluste | $P_{\text{cap}} = \alpha_{01} f C_L V_{DD}^2$ |
| Kurzschlussstrom | $P_{\text{short}} = \alpha_{01} f \beta_n \tau (V_{DD} - 2V_{tn})^3$ |
| Schalthäufigkeit | $\alpha_{0\rightarrow1} = \frac{\text{Schaltvorgänge(pos. Flanke)}}{\#\text{ Betrachtete Takte}}$ |

Abhängig von den Signalflanken, mit Schaltfunktionen verknüpft

$\approx V_{DD}1/ \propto$ Schaltzeit: $\frac{V_{DD2}}{V_{DD1}} = \frac{t_{D1}}{t_{D2}}$ (bei Schaltnetzen $t_{log}$)

Verzögerungszeit $\propto \frac{1}{V_{DD}-V_{th}}$

**Statische Verlustleistung** $P_{\text{stat}}$: Sub-Schwellströme, Leckströme, Gate-Ströme

Abhängigkeit: $V_{DD}$ ↑: $P_{\text{stat}}$ ↑ $\quad$ $V_{\text{th}}$ ↑: $P_{\text{stat}}$ ↓ $\quad$ (aber nicht proportional)

## 7. Sequentielle Logik

... Logik mit Gedächtnis.

### 7.1. Bedingungen

| | |
|---|---|
| $t_{\text{Setup}}$ | Stabilitätszeit vor der aktiven Taktflanke |
| $t_{\text{hold}}$ | Stabilitätszeit nach der aktiven Taktflanke |
| $t_{\text{c2q}}$ | Eingang wird spätestens nach $t_{\text{c2q}}$ am Ausgang verfügbar |
| Max. Taktperiode | $t_{\text{clk}} \geq t_{1,c2q} + t_{\text{logic,max}} + t_{2,\text{setup}}$ |
| Max. Taktfrequenz | $f_{\text{max}} = \left\lfloor \frac{1}{t_{\text{clk}}} \right\rfloor$ (Nicht aufrunden) |
| Holdzeitbedingung | $t_{\text{hold}} \leq t_{\text{c2q}} + t_{\text{logic,min}} \rightarrow$ Dummy Gatter einbauen |
| Durchsatz | $\frac{1\text{Sample}}{t_{\text{clk,pipe}}} = f$ |
| Latenz | $t_{\text{clk}} \cdot$ #Pipelinestufen (das zwischen den FFs) |

### 7.2. Pipelining

Nur bei synchronen(taktgesteuerten) Schaltungen möglich!

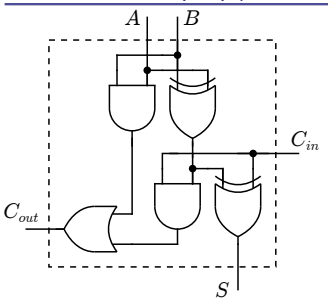
- Aufteilen langer kombinatorischer Pfade durch Einfügen zusätzlicher Registerstufen
  → Möglichst Halbierung des längsten Pfades
- Zeitverhalten beachten (evtl. Dummy-Gatter einfügen)
- Durchsatz erhöht sich entsprechend der Steigerung der Taktfrequenz
- Gesamtlatenz wird eher größer
- Taktfrequenz erhöht sich

### 7.3. Parallel Processing

Durchsatz $= \frac{\#\text{Modul}}{t_{\text{clk,Modul}}} = f$ $\qquad$ Latenz $= t_{\text{clk}}$

- Paralleles, gleichzeitiges Verwenden mehrere identischer Schaltnetze
- Zusätzliche Kontrolllogik nötig (Multiplexer)
- Taktfrequenz und Latenz bleiben konstant
- Durchsatz steigt mit der Zahl der Verarbeitungseinheiten
  ABER: deutlich höherer Ressourcenverbrauch

## 8. Volladdierer (VA) / Ripple-C(u)arry-Adder




Generate $g_n = a_n \cdot b_n$
Propergate $p_n = a_n \oplus b_n$
Summenbit $S_n = c_n \oplus p_n$
Carry-out: $c_{n+1} = c_n \cdot p_n + g_n$

Laufzeiten:

$$t_{sn} = \begin{cases} t_{cn} + t_{xor} & t_{cn} > t_{xor} \\ 2t_{xor} & sonst \end{cases}$$

$$t_{cn+1} = \begin{cases} t_{and} + t_{or} & p_n = 0, g_n = 1 \\ t_{xor} + t_{and} + t_{or} & p_n = 0, g_n = 0 \\ t_{cn} + t_{and} + t_{or} & p_n = 1 \end{cases}$$

## 9. Speicherelemente

### 9.1. Definitionen

**Flüchtig** Speicherinhalt gehen verloren, wenn Versorgungsspannung $V_{DD}$ wegfällt - Bsp: *RAM
**Nicht Flüchtig** Speicherinhalt bleibt auch ohne $V_{DD}$ erhalten - Bsp: Flash
**Asynchron** Daten werden sofort geschrieben/gelesen.
**Synchron** Daten werden erst mit $clk_{0\rightarrow1}$ geschrieben.
**Dynamisch** Ohne Refreshzyklen gehen auch bei angelegter $V_{DD}$ Daten verloren - Bsp: DRAM
**Statisch** Behält den Zustand bei solange $V_{DD}$ anliegt (keine Refreshzyklen nötig) - Bsp: SRAM
**Bandbreite:** Bitanzahl, die gleichzeitig gelesen/geschrieben werden kann.
**Latenz:** Zeitverzögerung zwischen Anforderung und Ausgabe von Daten.
**Zykluszeit:** Minimale Zeitdifferenz zweier Schreib/Lesezugriffe.

Speicherkapazität = Wortbreite · $2^{\text{Adressbreite}}$

### 9.2. Flip-Flop

2 besteht aus zwei enable-Latches
**Flip-Flop:** ändert Zustand bei steigender / (fallender) Taktflanke.

| $clk$ | $Q$ | $\overline{Q}$ |
|---|---|---|
| $0 \rightarrow 1$ | $D$ | $\overline{D}$ |
| sonst | $Q$ | $\overline{Q}$ |

### 9.3. Register

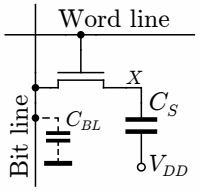
Ring aus zwei Invertern.

### 9.4. Latch

**Set-Reset Latch:**
Zwei gegenseitig rückgekoppelte NAND-Gatter. 0 an R/S schaltet.
**Enable-Latch:** ändert Speicherzustand auf $D$ nur wenn $e = 1$:

| e | Q |
|---|---|
| 0 | Q |
| 1 | D |

### 9.5. DRAM Zelle (dynamisch)




lange Bitlines → $C_{BL}$ ↑, Laufzeit↑
quadratisch: $\frac{\text{Bit}}{\text{Zeile}} \stackrel{!}{=} \frac{\text{Bit}}{\text{Spalte}} = \frac{\text{Wort}}{\text{Zeile}} \cdot \frac{\text{Bit}}{\text{Wort}}$

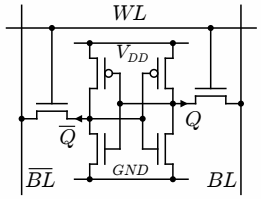
**Schreiben**

- Wortleitung wird aktiviert, d.h. auf $V_{DD}$ gelegt
- Bitleitung wird auf den gewünschten Wert ($V_{DD}$ für 1, GND für 0) gelegt
  ⇒ Kondensator wird auf entsprechendes Potential aufgeladen oder entladen, je nach vorherigem Wert

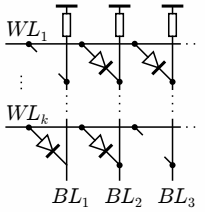
**Lesen**

- Wortleitung wird aktiviert, d.h. auf $V_{DD}$ gelegt.
- Bitleitung wird auf $V_{DD}/2$ vorgeladen.
- Adresstransistor wird geöffnet
  → Ladungsaustausch zwischen $C_S$ und $C_{BL}$
  → Potential der BL wird um $\Delta V$ erhöht (1 lesen) oder erniedrigt (0 lesen)
  → $\Delta V = \left(V_X - \frac{V_{DD}}{2}\right) \cdot \frac{C_S}{C_S + C_{BL}}$ i.d.R $C_{BL} >> C_S \rightarrow \Delta V$ sehr klein
  → Leseverstärker nötig!

### 9.6. SRAM Zelle (statisch)

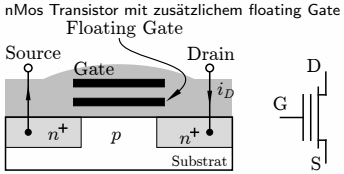



### 9.7. ROM - Read Only Memory




### 9.8. Flash (nicht flüchtig)

nMos Transistor mit zusätzlichem floating Gate in der Oxidschicht.




‚0' speichern: $V_{GS} = V_{DS} = 4 \cdot V_{DD}$, S an GND
‚0' löschen: S von GND trennen, G an GND und D an 4mal VDD

### 9.9. Organisation von Speichern

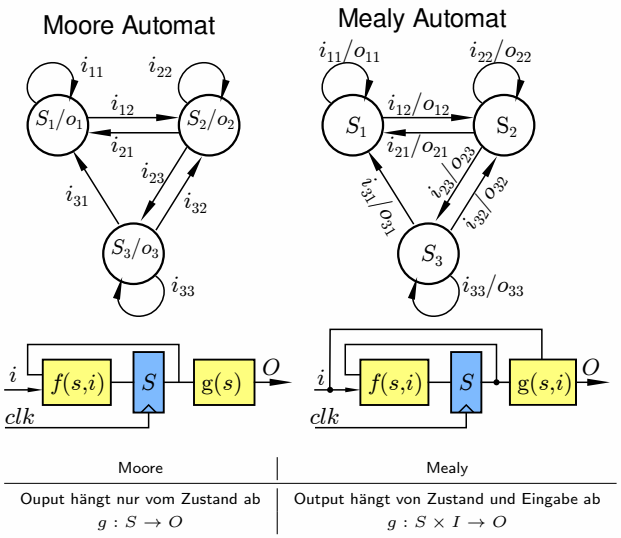
- 1 Byte besteht aus 8 Bit
- Ziel: möglichst quadratische Anordnung der Speicherzellen
- Wortbreite W berücksichtigen!

Aufteilung:
Speicherkapazität $= 2^M \cdot 2^N$ Bester Fall für $M = N$
# Reihen $= N$
$2^M = W \cdot 2^K$
# Spalten $= K$

# 10. Automaten

| | |
|---|---|
| $I$ | Eingabealphabet |
| $O$ | Ausgabealphabet |
| $S$ | Menge von Zuständen |
| $R \subseteq S$ | Menge der Anfangszustände |
| $f : S \times I \to S$ | Übergangsrelation |
| $g$ | Ausgaberelation |

DFA 6-Tupel $\{I, O, S, R, f, g\}$







| Moore | Mealy |
|---|---|
| Ouput hängt nur vom Zustand ab $g : S \to O$ | Output hängt von Zustand und Eingabe ab $g : S \times I \to O$ |

## 10.1. Vorgehensweise

- $I, O$ bestimmen
- $S$ festlegen
- $R$ bestimmen
- $f, g$ bestimmen

# Entwurfsverfahren

## 1. Boolsche Algebra

| | Mengenalgebra $(P(G); \cap, \cup, \overline{A}; G, \emptyset)$ | Boolesche Algebra $(0, 1; \cdot, +, \overline{x})$ |
|---|---|---|
| Kommutativ | $A \cap B = B \cap A$ | $x \cdot y = y \cdot x$ |
| | $A \cup B = B \cup A$ | $x + y = y + x$ |
| Assoziativ | $(A \cap B) \cap C = A \cap (B \cap C)$ | $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ |
| | $(A \cup B) \cup C = A \cap (B \cup C)$ | $x + (y + z) = (x + y) + z$ |
| Distributiv | $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ | $x \cdot (y + z) = x \cdot y + x \cdot z$ |
| | $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ | $x + (y \cdot z) = (x + y) \cdot (x + z)$ |
| Indempotenz | $A \cap A = A$ | $x \cdot x = x$ |
| | $A \cup A = A$ | $x + x = x$ |
| Absorbtion | $A \cap (A \cup B) = A$ | $x \cdot (x + y) = x$ |
| | $A \cup (A \cap B) = A$ | $x + (x \cdot y) = x$ |
| Neutral | $A \cap G = A$ | $x \cdot 1 = x$ |
| | $A \cup \emptyset = A$ | $x + 0 = x$ |
| Dominant | $A \cap \emptyset = \emptyset$ | $x \cdot 0 = 0$ |
| | $A \cup G = G$ | $x + 1 = 1$ |
| Komplement | $A \cap \overline{A} = \emptyset$ | $x \cdot \overline{x} = 0$ |
| | $A \cup \overline{A} = G$ | $x + \overline{x} = 1$ |
| | $\overline{\overline{A}} = A$ | $\overline{\overline{x}} = x$ |
| De Morgan | $\overline{A \cap B} = \overline{A} \cup \overline{B}$ | $\overline{x \cdot y} = \overline{x} + \overline{y}$ |
| | $\overline{A \cup B} = \overline{A} \cap \overline{B}$ | $\overline{x + y} = \overline{x} \cdot \overline{y}$ |

### 1.1. Multiplexer

$f = x \cdot a + \overline{x} \cdot b$ (2 Eingänge $a, b$ und 1 Steuereingang $x$)

$f = \overline{x}_1\overline{x}_2 a + \overline{x}_1 x_2 b + x_1\overline{x}_2 c + x_1 x_2 d$ (Eingänge: $a, b, c, d$ Steuerung: $x_1$, $x_2$)

### 1.2. Wichtige Begriffe

| Wichtige Begriffe: | Definition | Bemerkung |
|---|---|---|
| Signalvariable | $x$ | $\hat{x} \in \{0, 1\}$ |
| Literal | $l_i = x_i$ oder $\overline{x_i}$ | $i \in I_0 = \{1, \ldots, n\}$ |
| Minterme,0-Kuben | M0C $\ni m_j = \prod_{i \in I_0} l_i$ | $\lvert M0C \rvert = 2^n$ |
| d-Kuben | MC $\ni c_j = \prod_{i \in I_j \subseteq I_0} l_i$ | $\lvert MC \rvert = 3^n$ |
| Distanz | $\delta(c_i, c_j) = \lvert \{l \mid l \in c_i \wedge \overline{l} \in c_j\} \rvert$ | $\delta_{ij} = \delta(c_i, c_j)$ |
| Implikanten | $MI = \{c \in MC \mid c \subseteq f\}$ | |
| Primimplikanten | $MPI = \{p \in MI \mid p \not\subset c \; \forall c \in MI\}$ | $MPI \subseteq MI \subseteq MC$ |

| | | |
|---|---|---|
| SOP (DNF) | eine Summe von Produkttermen | Terme sind ODER-verknüpft |
| POS (KNF) | ein Produkt von Summentermen | Terme sind UND-verknüpft |
| CSOP (nur 1) | Menge aller Minterme | analog CPOS |
| VollSOP (nur 1) | Menge aller Primimplikanten | Bestimmung siehe Quine Methode oder Schichtenalgorithmus |
| MinSOP (min. 1) | Minimale Summe v. Primimplikanten | durch Überdeckungstabelle |

FPGA: Field Programmable Gate Array
LUT: Look Up Table

### 1.3. Boolesche Operatoren (Wahrheitstabelle WT)

| x | y | AND $x \cdot y$ | OR $x + y$ | XOR $x \oplus y$ | NAND $\overline{x \cdot y}$ | NOR $\overline{x + y}$ | EQV $\overline{x \oplus y}$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

Konfiguration: $f = c_1 + c_2 + c_3 \Rightarrow cov(f) = \{c_1, c_2, c_3\}$

## 2. Beschreibungsformen

### 2.1. Sum of products (SOP/DNF)
Eins-Zeilen der Wertetabelle ODER verknüpfen:
$f = \overline{x} \cdot \overline{y} + \overline{z} \cdot w$

### 2.2. Product of sums (POS/KNF)
Null-Zeilen der Wertetabelle negieren und UND verknüpfen:
$f = (\overline{x} + \overline{z}) \cdot (\overline{x} + \overline{w}) \cdot (\overline{y} + \overline{z}) \cdot (\overline{y} + w)$

### 2.3. Shannon Entwicklung
$f = x_i \cdot f_{x_i} + \overline{x}_i \cdot f_{\overline{x}_i} = (x_i + f_{\overline{x}_i}) \cdot (\overline{x}_i + f_{x_i}) = (f_{x_i} \oplus f_{\overline{x}_i}) \cdot x_i \oplus f_{\overline{x}_i}$
$\overline{f} = x_i \cdot \overline{f}_{x_i} + \overline{x}_i \cdot \overline{f}_{\overline{x}_i}$

### 2.4. Umwandlung in jeweils andere Form
1. Doppeltes Negieren der Funktion: $f = \overline{\overline{x} \cdot \overline{y} + \overline{z} \cdot w}$
2. Umformung "untere" Negation (DeMorgan) : $f = \overline{\overline{x} \cdot \overline{y}} \cdot \overline{\overline{z} \cdot w} = \overline{(x + y) \cdot (z + \overline{w})}$
3. Ausmultiplizieren: $f = \overline{(x + y) \cdot (z + \overline{w})} = \overline{x} \cdot z + x \cdot \overline{w} + y \cdot z + y \cdot \overline{w}$
4. Umformung "obere" Negation (DeMorgan) :
$f = \overline{\overline{x}z} \cdot \overline{x\overline{w}} \cdot \overline{yz} \cdot \overline{y\overline{w}} = (\overline{x} + \overline{z}) \cdot (\overline{x} + w) \cdot (\overline{y} + \overline{z}) \cdot (\overline{y} + w)$
Analog von POS nach SOP.

### 2.5. Quine Methode
geg.: SOP oder Wertetabelle
ges.: alle Primimplikanten (VollSOP)

spezielles Resoltuionsgesetz: $x \cdot a + \overline{x} \cdot a = a$
Absorptionsgesetz: $a + a \cdot b = a$

- CSOP bestimmen (z.B. $f(x, y, z, w) = xy\overline{z} + x\overline{y}z + xyz$)
- alle Minterme in Tabelle eintragen (Index von m ist (binär)Wert des Minterms)
- Wenn Kubenabstand = 1 (ein "don't care") in 1-Kubus aufnehmen und $A$ abhaken. Wenn nicht ist dieser Minterm bereits ein Primimplikant.
- der 1-Kubus muss zusammenhängend sein! (d.h. alle 1-Kubus Minterme müssen zusammenhängen)
- Wenn möglich 2-Kubus bilden.
- Wenn keine Kubenbildung mehr möglich → VollSOP

Beispiel (Quine Methode):

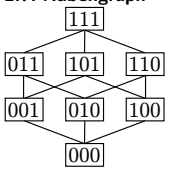
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $m_0$ | 0-Kubus | A | 1-Kubus | R | A | 2-Kubus | A |
| $m_1$ | $\overline{x}_1\overline{x}_2x_3$ | ✓ | $\overline{x}_2x_3$ | $m_1\&m_5$ | $p_1$ | | |
| $m_4$ | $x_1\overline{x}_2\overline{x}_3$ | ✓ | $x_1\overline{x}_2$ | $m_4\&m_5$ | ✓ | $x_1$ | $p_2$ |
| $m_5$ | $x_1\overline{x}_2x_3$ | ✓ | $x_1\overline{x}_3$ | $m_4\&m_6$ | ✓ | | |
| $m_6$ | $x_1x_2\overline{x}_3$ | ✓ | $x_1x_3$ | $m_5\&m_7$ | ✓ | | |
| $m_7$ | $x_1x_2x_3$ | ✓ | $x_1x_2$ | $m_6\&m_7$ | ✓ | | |

### 2.6. Quine's und McCluskey's Bestimmung der MinSOP
Geg: CSOP ($\sum m_i$) und VollSOP ($\sum p_i$) Ges: MinSOP

Überdeckung:
$C = (m_0 \subseteq p_1) \cdot (m_2 \subseteq p_1 + m_2 \subseteq p_2) \stackrel{!}{=} 1$
$C = \tau_1 \cdot (\tau_1 + \tau_2) = \tau_1 + \tau_1\tau_2 = \tau_1$

Alternativ: Mit Überdeckungstabelle bestimmen.

### 2.7. Kubengraph



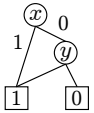

Kubenabstand $\delta(c_1, c_2)$: Kleinste Anzahl an Kanten, die nötig sind, um $c_i$ und $c_j$ zu verbinden bzw. Anzahl an Literalen die in $c_1$ negiert und in $c_2$ nicht negiert vorkommen.

#Literale = #Raumdimensionen - #Kubusdimensionen

Max. Kubenabstand: #Dimensionen - #Kubusdimensionen(größter Kubus)

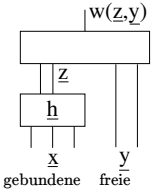
überdeckte Minterme: $2^{\text{Kubendimension}}$

### 2.8. (R)OBDD
(**R**educed) **O**rdered **B**inary **D**ecision **D**iagram




ROBDD → SOP: Alle Pfade zur 1 verodern:
$f = x + \overline{x}y$
ROBDD → POS: Alle Pfade zu 0 verodern, komplett-ten Term negieren, DeMorgan anwenden

## 3. Funktionale Dekomposition




gebundene Variablen — freie Variablen

Bei einer Funktion $f(\underline{\boldsymbol{v}})$ mit $n$ Eingängen und einer möglichen Aufteilung von $\underline{\boldsymbol{v}}$ in $\underline{\boldsymbol{v}} = \underline{\boldsymbol{x}}$ und $\underline{\boldsymbol{y}}$ (wobei die Aufteilung disjunkt ist), kann $f(\underline{\boldsymbol{v}}) = f(\underline{\boldsymbol{x}}, \underline{\boldsymbol{y}})$ in $g(h(\underline{\boldsymbol{x}}), \underline{\boldsymbol{y}})$ zerlegt werden.

Zerlegung sinnvol, wenn $|\underline{\boldsymbol{z}}| \leq |\underline{\boldsymbol{x}}| - 1$ oder $|Z| \leq \frac{1}{2}|X|$
Kompositionsfunktion: $w = g(\underline{\boldsymbol{z}}, \underline{\boldsymbol{y}})$
Dekompositionsfunktion: $\underline{\boldsymbol{z}} = h(\underline{\boldsymbol{x}})$

→ Meist kann man eine günstige Aufteilung per BDD finden.

Verfahren:

- Auswerten von $f(\underline{\hat{\boldsymbol{x}}}, \underline{\boldsymbol{y}})$ und Bilder der Dekompositionsmatrix:
  $f = \overline{x}_1\overline{x}_2y_1 + \overline{x}_1x_2x_3y_1 + \overline{x}_1x_2\overline{x}_3\overline{y}_1 \ldots$
- Trage die Funktionswerte in die Matrix ein
- Suche Spalten, die die selben Werte je $\underline{\boldsymbol{y}}$ haben
- Codiere gleiche Spalten mit gleichem $z$

| freie Variablen | gebundene Variablen $(x_1, x_2, x_3)$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $(y_1, y_2)$ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| 00 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 01 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 10 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 11 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| $\underline{\boldsymbol{z}} = h(x)$ | 00 | 00 | 01 | 00 | 10 | 10 | 10 | 01 |

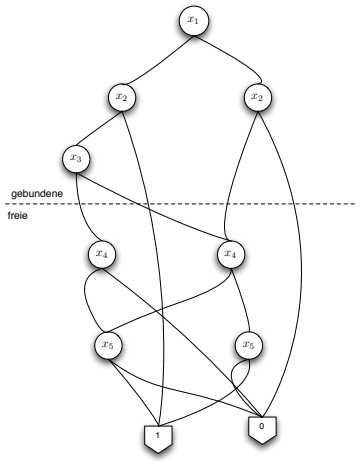
- Konstruiere die Dekompositionsfunktion
  $|\underline{\boldsymbol{z}}| \leq |\underline{\boldsymbol{x}}| - 1$ bzw. $|Z| \leq \frac{1}{2}|X|$
- Stelle Zuordnungstabelle auf:

| $\underline{\hat{\boldsymbol{x}}}_{\boldsymbol{i}}$ | $\underline{\hat{\boldsymbol{z}}}_{\boldsymbol{i}}$ | $\underline{\boldsymbol{z}} = h(x)$ | $g(\underline{\boldsymbol{z}}_i, \underline{\boldsymbol{y}})$ |
|---|---|---|---|
| 000, 001, 011 | 00 | $\overline{z}_1\overline{z}_2 = \overline{x}_1\overline{x}_2 + \overline{x}_1x_2x_3$ | $y_1$ |
| 010, 111 | 01 | $\overline{z}_1z_2 = \overline{x}_1x_2\overline{x}_3 + x_1x_2x_3$ | $\overline{y}_1$ |
| 100, 101, 110 | 10 | $z_1\overline{z}_2 = x_1\overline{x}_2 + x_1x_2\overline{x}_3$ | $\overline{y}_1 + y_2$ |
| | 11 | | 0 |

- Konstruktion der Kompositionsfunktion
  → via Dekompositionsmatrix (1) / Zuordnungstabelle (2)
  (1) alle eingetragenen 1en $\hat{=}$ 1 am Ausgang
  → müssen in der Kompositionsfunktion auftreten
  (2) $\underline{\hat{\boldsymbol{z}}}_{\boldsymbol{i}} \cdot g(z, y)$ stellen die Komposistionsfunktion dar

  $g(\underline{\boldsymbol{z}}, \underline{\boldsymbol{y}}) = \overline{z}_1\overline{z}_2y_1 + \overline{z}_1z_2\overline{y}_1 + z_1\overline{z}_2\overline{y}_1 + z_1\overline{z}_2y_1y_2$ = Kompositionsfunktion
- Notationen: $|\underline{\boldsymbol{x}}|$ = Zahl der Eingangsvariablen (gebunden)
  $|\underline{\boldsymbol{z}}|$ = Zahl der Dekompositionsvariablen
  $|X| = 2^{|\underline{\boldsymbol{x}}|}$ = Zahl der möglichen Zustände aller gebundenen Variablen
  $|Z| = 2^{|\underline{\boldsymbol{z}}|}$ = Zahl aller Dekompositionsvariablen

### 3.1. Funktionale Dekomposition mit ROBDD
Ermitteln der gebundenen bzw. freien Variablen mittels BDD:




**Dekompositionsbedingung** $|\underline{z}| \leq |\underline{x}| - 1$ bzw. $|Z| \leq \frac{1}{2}\,|X|$

- Nehme immer eine Ebene an:
  Oberhalb = gebundene Variablen
  Unterhalt = freie Variablen
- Zähle die Knoten, die durch die kreuzenden Äste erreicht werden (hier mit $\Delta$ bezeichnet)
- Auch wenn ein Knoten durch zwei oder mehrere Äste erreicht wird, darf er nur **einmal** gezählt werden (im Beispiel: $\Delta = 4$)
- Wenn $|\underline{z}| = \lceil \log_2 \Delta \rceil \leq |\underline{x}| - 1 \rightarrow$ DK-Bed. erfüllt
- Pfade zur 1 ergeben Dekompositionsfunktion ( gebundene Variablen $\rightarrow$freie Variablen $\rightarrow$1)

Zuordnungstabelle:

| geb. Variablen | $z_1$ | $z_2$ | freie Variablen |
|---|---|---|---|
| 111 | 0 | 0 | $x_4 x_5$ |
| 110, 010, 011 | 0 | 1 | $x_4 x_5 + \overline{x}_4 \overline{x}_5$ |
| 101, 100 | 1 | 0 | 1 |
| | 1 | 1 | |

### 3.2. Heuristische Minimierung
Kofaktorbildung: Setze alle $x_i = 1$ und alle $\overline{x}_i = 0$
z.B. $f = x\overline{y} + \overline{x}yw + xw \Rightarrow f_x = \overline{y} + w$

#### 3.2.1 Kubenentfernung (remove)
- $h = f \setminus c$ ($f$ ohne den zu entfernenden Kubus)
  z.B. $f = \overline{x}yz + xyz + \overline{x}yz \Rightarrow$ für $c = \overline{x}yz \Rightarrow h = xyz + \overline{x}yz$
  $h_{\overline{x}yz} = 1 + 0 = 1 \Rightarrow$ entfernbar
- Bildung des Kofaktors $h_c$
- Wenn $h_c = 1 \Rightarrow c$ ist entfernbar

2 Kuben gemeinsam entfernen: teste 2. Kubus **nachdem** der 1. entfernt wurde.

#### 3.2.2 Literalentfernung (expand)
- Aufstellen von $h = \{f \setminus c_l \cdot l\}$
  z.B. $f = \overline{x}y + xyz + \overline{x} \cdot \overline{y} \cdot \overline{z}$ (entferne $x$: d.h. $l = x$ und $c_l = yz$)
  $h = \overline{x}y + \overline{x} \cdot \overline{y} \cdot \overline{z}$
- Wenn $h_{c_l \cdot \overline{l}} = 1$ ist das Literal entfernbar
  z.B. $h_{c_l \cdot \overline{l}} = (\overline{x}y + \overline{x} \cdot \overline{y} \cdot \overline{z})_{\overline{x}yz} = 1$

#### 3.2.3 Literal hinzufügen (reduce)
Kann man $l$ zu $c$ hinzufügen ohne $f$ zu verändern?
$f = c + h \stackrel{?}{=} c \cdot l + h$
Zulässigkeitsbedingung: $c \cdot \overline{l} \subseteq h$
z.B. $f = xy + \overline{x}yz + xz$ (füge $l = \overline{z}$ hinzu)
Ist $xyz \subseteq \overline{x}yz + xz$?
Ja $\Rightarrow f^* = xy\overline{z} + \overline{x}yz + xz$
Gemeinsame Literalentfernung: Prüfe 2. Literal nachdem 1. Literal entfernt wurde.

### 3.3. Strukturanalyse
Tautologie: $f_{x_i} = 1 \ \wedge \ f_{\overline{x}_i} = 1 \quad \Rightarrow f = 1$

Monoton steigend in $x_i$: $\boxed{f_{\overline{x}_i} \subseteq f_{x_i}}$ dann gilt auch $f_{\overline{x}_i} = 1 \ \Rightarrow \ f = 1$

Monoton fallend in $x_i$: $\boxed{f_{x_i} \subseteq f_{\overline{x}_i}}$ dann gilt auch $f_{x_i} = 1 \ \Rightarrow \ f = 1$

Beispiel:
$f = yz + xz + \overline{y}z + \overline{yz} + y\overline{z} \Rightarrow$ monoton steigend in $x$
$f(x, y, z) = x\,\varphi(z) + h(y, z) \Rightarrow$ Prüfe $h$ auf Tautologie

## 4. Nützliches Wissen
### 4.1. Mehrfachimplikanten
Sind gleiche Implikanten in mehreren verschiedenen Funktionen vorhanden?
Prüfe $f_1 \cap f_2$ auf Mehrfachimplikanten: $f_1 \cdot f_2 =?$
Nutzung von Mehrfachimplikanten ist sinnvoll wenn die Gesamtliteralzahl beider SOPs kleiner ist als ohne Verwendung von Mehrfachimplikanten.

### 4.2. VollSOP erstellen
Benutze die Resolventenmethode um alle Resolventen zu erzeugen und so aus der MinSOP eine VollSOP zu erstellen.

## 5. Automaten
sind abstrakte Maschinen mit $r$ Zuständen $S_i \in S$ , die auf sequentielle Eingangssignale $X_j \in I = \mathbb{B}^n$ mit Ausgangssignalen $Y_l \in O = \mathbb{B}^m$ und Zustandsänderungen reagiern.

| | |
|---|---|
| Startzustand | $S^0 \in S$ |
| Zustandsfkt. | $\delta : S \times I \rightarrow S, \ S_k \mapsto \delta(S_i, X_j)$ |
| Ausgangsfkt. | $\lambda : S \times I \rightarrow O, \ Y_l \mapsto \lambda(S_i, X_j)$ |
| ZA-fkt. | $\mu : S \times I \rightarrow S \times O, \ (S_k, Y_l) \mapsto \mu(S_i, X_j)$ |

$k$-**Äquivalenz** $S_i \stackrel{k}{\sim} S_j$ Für eine Eingangssequenz der Länge $k$ sind bei $S_i$ und $S_j$ die Ausgaben gleich und die Zustandsübergänge gleich bzw. $k-1$ Äquivalent.

**Totale Äquivalenz** $S_i \sim S_j$ falls für alle Eingangssequenzen die Ausgaben und die Zustandsübergänge äquivalent sind.
$\rightarrow$ Zeige: es lassen sich keine weiteren Äquivalenzklassen bilden.

## 6. Karnaugh- Diagramm
Zyklische Gray-Codierung: 2dim:00, 01, 11, 10 3dim:000, 001, 011, 010, 110, 111, 101, 100

| $z \diagdown xy$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | X | 1 | 1 | 0 |

Gleiche Zellen zusammenfassen: z.B. $\overline{xy} + y \cdot z$

Don't Care Werte ausnutzen!

## 7. Resolventenmethode
Ziel: alle Primimplikanten

Wende folgende Gesetze an:
Absorptionsgesetz: $a + ab = a$
allgemeines Resolutionsgesetz: $x \cdot a + \overline{x} \cdot b = x \cdot a + \overline{x} \cdot b + ab$

Anwendung mit Schichtenalgorithmus
1. schreibe die Funktion $f$ in die 0. Schicht
2. bilde **alle möglichen** Resolventen aus der 0. Schicht und schreibe sie in die nächste Schicht als ODER Verknüpfungen (Resolventen zu $f$ "hinzufügen")
3. überprüfe ob Resolventen aus der 1. Schicht Kuben aus Schicht 0 überdecken(Absorbtion) und streiche diese Kuben aus Schicht 0
4. Schicht i besteht aus den möglichen Resolventen von Schicht 0 bis $(i-1)$. Abgestrichene Kuben aus vorherigen Schichten brauchen **nicht** mehr beachtet werden.
5. Sobald in der i-ten Schicht $+1$ steht oder keine weiteren Resolventen gebildet werden können, ist man fertig. $\Rightarrow$ alle nicht ausgestrichenen Terme bilden die VollSOP

| $f(x_1, \ldots, x_n)$ | Schicht |
|---|---|
| $x \cdot w + \overline{x} \cdot w + x \cdot y \cdot w \cdot \overline{z} + \overline{x} \cdot y \cdot w \cdot \overline{z} + \overline{y} \cdot w \cdot \overline{z}$ | 0 |
| $+w + y \cdot w \cdot \overline{z}$ | 1 |
| $+w \cdot \overline{z}$ | 2 |
| $+w$ | 3 |

## 8. Laufzeit
### 8.1. Laufzeitabhängige Effekte
- **Race**, "Wettlauf" zweier Signalwertänderungen vor einem gemeinsamen Gatter
- **Hazard / Spike / Glitch**, Stelle des Signalwertverlaufes, die wegen der Laufzeitverzögrung nicht den Erwartungen entspricht

### 8.2. Simulation


$\tau_{OR} = 2$ und $\tau_{NOT} = 1$ Eingangsbelegung $a = 0, b = 0$
Eingangsereignis (b,'1',0, 2)
Auswertung erfolgt durch eine Tabelle:

| t | a | b | z | y | ausgewertete Elemente | neue Ereignisse |
|---|---|---|---|---|---|---|
| 0 | '0' | '0' | '1' | '0' | init | (b,'1',0, 2) |
| 2 | | '1' | | | OR | (z,'1',2,4) |
| 4 | | | '1' | | NOT | (y,'0',4,5) |
| 5 | | | | '0' | | |

Ereignis: (betroffenes Signal, neuer Signalwert, $t$, $t + \tau$)

### 8.3. Delay
- transport delay: Verzögerung um $\tau_{pd}$
- inertial delay: Verzögerung um $\tau_{pd}$ und Impulse die kleiner als $\tau_{pd}$ sind werden ignoriert

### 8.4. VHDL- VHSIC Hardware Description Language
```
ENTITY Bausteinname IS                    //Definiert die Schnittstelle einer Logik
PORT (Schnittstellenliste)                //Definiert Ein- und Ausgänge

ARCHITECTURE Rumpfname OF Bausteinname IS //Beschreibt den internen Aufbau
PROCESS (Signalliste)                     // Alle Processes laufen nebeneinander ab
COMPONENT Gattername                      // Beschreibt eine interne Komponente
```

## 9. Testverfahren
Mit wenig Fragen viel Information erhalten. Signal muss beobachtbar und einstellbar sein!

### 9.1. Begriffe
Fehlergruppe $F_\nu = \{f_\mu \in F \mid t_\nu R f_\mu\}$: Menge aller Fehler die vom Test $t_\nu$ erkannt werden.
Fehleranzahl = 2· Signalanzahl = 2(Eingänge + Interne Signale + Ausgänge)
Testgruppe $T_\mu = \{t_\nu \in T \mid t_\nu R f_\mu\}$ ist die Menge aller Tests die den Fehler $f_\mu$ erkennen.

Zwei einzelne Fehler sind nicht unterscheidbar, wenn sie immer gemeinsam von einem Test entdeckt werden.

### 9.2. Bullshit-Differenz $y_z$
Ziel: schnelles finden von Testbedingungen für $f = y(z(\underline{x}))$

$\boxed{y_z = y(z, \underline{x}) \oplus y(\overline{z}, \underline{x})} \ \widehat{=}\ y(z = 1) \oplus y(z = 0)$

#### 9.2.1 Rechenregeln
| | |
|---|---|
| $y_x = 0$ falls $y \neq f(x)$ | $(z \cdot w)_x = z \cdot w_x \oplus z_x \cdot w \oplus z_x \cdot w_x$ |
| $y_y = 1$ | $(z + w)_x = \overline{z} \cdot w_x \oplus z_x \cdot \overline{w} \oplus z_x \cdot w_x$ |
| $(\overline{y})_x = y_x$ | $y_x = y_z \cdot z_x$ falls $y = y(z(x))$ |
| $(z \oplus w)_x = z_x \oplus w_x$ | $(y_z)_w = (y_w)_y$ |

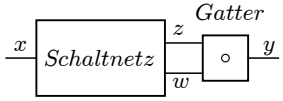
#### 9.2.2 AND $\rightarrow$ XOR $(+, \overline{x})$
Vorgehen:
1. Negation über konjunkte Terme entfernen: DeMorgan $\overline{xy} = \overline{x} + \overline{y}$)
2. Negation über disjunkte Terme entfernen: $\overline{x} = x \oplus 1$)
3. "+" entfernen $x + y = x \oplus y \oplus xy$

Test für $\begin{cases} a/0 : a \cdot y_a \stackrel{!}{=} 1 \\ a/1 : \overline{a} \cdot y_a \stackrel{!}{=} 1 \end{cases}$ $\Rightarrow$ Testmuster finden.

#### 9.2.3 XOR-Regeln

| | |
|---|---|
| $x \oplus y = \overline{x} \cdot y + x \cdot \overline{y}$ | $x + y = x \cdot y \oplus x \oplus y$ |
| $x \oplus y = (x + y) \cdot (\overline{x} + \overline{y})$ | $x \cdot y = x \oplus y \oplus (x + y)$ |
| $x \cdot (y \oplus z) = x \cdot y \oplus x \cdot z$ | $\overline{x} = x \oplus 1$ |
| $x \oplus x = 0$ | $x \oplus 0 = x$ |
| $(x + y) \oplus y = x \cdot \overline{y}$ | $x\overline{y} + yz = x\overline{y} \oplus yz$ |

#### 9.2.4 Schaltnetze




Lokal: $y = z \circ w$
$y_z = (\overline{z} \circ w) \oplus (z \circ w)$
$y_w = (z \circ \overline{w}) \oplus (z \circ w)$

Global: $y_x = [(z \oplus z_x) \circ (w \oplus w_x)] \oplus [z \circ w]$

$\boxed{y_x = y_z z_x \overline{w}_x + y_w w_x \overline{z}_x + z_x w_x \cdot [\overline{z} \circ \overline{w} \oplus z \circ w]}$

$\circ$ = XOR/XNOR $\Rightarrow$ $[\overline{z} \circ \overline{w} \oplus z \circ w] = 0$
$\circ$ = AND/NAND/OR/NOR $\Rightarrow$ $[\overline{z} \circ \overline{w} \oplus z \circ w] = \overline{z \oplus w}$

Schaltnetz mit Rekonvergenzmasche (4 Fälle):

1. $y_z z_x \overline{w}_x = 1$ $\Rightarrow$ Einfachpfadsensibilisierung: $x$—$z$—$y$
2. $y_w w_x \overline{z}_x = 1$ $\Rightarrow$ Einfachpfadsensibilisierung: $x$—$w$—$y$
3. $z_x w_x \cdot [\overline{z} \circ \overline{w} \oplus z \circ w] = 1$ $\Rightarrow$ Mehrfachpfadsensibilisierung
4. $z_x w_x \cdot [\overline{z} \circ \overline{w} \oplus z \circ w] = 1$ $\Rightarrow$ Selbstmaskierung

Schaltnetz mit Baumstruktur:
keine Mehrfachpfadsensibilisierung oder Selbstmaskierung!
$y_x = y_z \cdot z_x$ (Kettenregel für $x$—$u$—$z$—$y$)

### 9.3. Fehlersimulation

Gegeben: Testmuster Gesucht: getestete Fehler.

**Achtung** Teste immer nur für eine spezielle Belegung

**Begriffe** Fanout-Stamm: Verzweigungspunkt Vereinigungspunkt: Rekonvergenzpunkt

#### 9.3.1 Fehlerbaumkonstruktion für gegebenes Testmuster

1. Alle Signalwerte der Schaltung für gegebenes Testmuster bestimmen
2. Durch die Simmulation die Beobachtbarkeit der Fanout-Stämme bestimmen.
3. Schaltung an Fanout-Stämmen gedanklich auftrennen. (In Fanout-Freie Zonen zerlegen)
4. Fehlerbaumkonstruktion in den Bäumen(FF-Zonen) vom Ausgang in Richtung Eingänge auf Basis der lokalen Sensitivitäten (mit • markieren).
5. Aus Fehlerbaum Menge der beobachtbaren Signale $S^0$ (sensitiver Pfad zum Ausgang) und Menge der getesteten Fehler $F_t$ durch Einstellbarkeit (sensitiver Pfad zum Eingang) angeben.
   z.B. $S^\circ = \{a, b, b_2, y\}$ $F_t = \{a/1, b/0, b_2/0y/1\}$

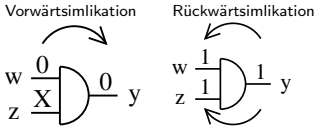
### 9.4. Deterministische Testmustergenerierung. D-Algorithmus

Zahl aller Fehlerpfade: $2^n - 1$ für $n$ = Zahl der Einfachfehlerpfade

Findet für jeden stuck-at Fehler einen Test, falls möglich.
5-wertige Logik:

| | |
|---|---|
| 0 | Boolsche 0 |
| 1 | Boolsche 1 |
| $X$ | undefined |
| $D$ | 1 falls fehlerfrei, 0 falls fehlerhaft (teste stuck-at 0) |
| $\overline{D}$ | 0 falls fehlerfrei, 1 falls fehlerhaft (teste stuck-at 1) |

Lokale Implikation:




globale Implikation: mehr als 1 Gatter zwischen Testsignal und implizierten Signal:

$\boxed{A \Rightarrow B \;\Leftrightarrow\; \overline{B} \Rightarrow \overline{A}}$

Lernkriterium (für $y = z(x)$): $y_{z_1} \cdot y_{z_2} \stackrel{!}{=} 1$

Vorgehen (z.B für Test $x_1/0$):
$F$ : Fehlertestsignal ($x_1 = D$)
$S$ : Sensibilisierung ($x_2 = 1$)
$I$ : Impliaktion ($y = \overline{D}$)
$O$ : Optionale Pfade (z.B bei XOR)

Ist Ausgang 0 bzw. 1 $\Rightarrow$ Fehler nicht testbar.
Ist Ausgang $D$ bzw. $\overline{D}$ $\Rightarrow$ Fehler testbar.

### 9.5. Einstellbarkeit

$\boxed{C_0 + C_1 = 1}$

$C_0$: Nulleinstellbarkeit $0 \leq C_0 \leq 1$
Wahrscheinlichkeit das ein Testvektor zur 0 führt
$C_1$: Einseinstellbarkeit $0 \leq C_1 \leq 1$

ohne Vorgabe für Eingangsvariable x: $C_0(x) = 0,5$ und $C_1(x) = 0,5$
Beachte: Je nach Gatter sind besimmte Einstellbarkeiten schneller zu berechnen! (siehe Tabelle)

| Gatter | Einstellbarkeit (Ausgang) | Berechnung |
|---|---|---|
| AND | $C_1$ | $C_1(x_1) \cdot C_1(x_2)$ |
| NAND | $C_0$ | $C_1(x_1) \cdot C_1(x_2)$ |
| OR | $C_0$ | $C_0(x_1) \cdot C_0(x_2)$ |
| NOR | $C_1$ | $C_0(x_1) \cdot C_0(x_2)$ |
| XOR | $C_1$ | $C_0(x_1) \cdot C_1(x_2) + C_1(x_1) \cdot C_0(x_2)$ |
| NOT | $C_1$ | $C_0(x)$ |

Wichtig: Nur bei Baumstruktur exakt.

### 9.6. Schaltwerke

| | |
|---|---|
| Eingangsvariable | $\underline{\boldsymbol{X}}$ |
| Testpunkte | $\underline{\boldsymbol{Y}}$ |
| nächste Schaltwerkszustände | $\underline{\boldsymbol{Z}}$ |
| Schaltwerkszustände | $\underline{\boldsymbol{S}}$ |

Es gilt: $\underline{\boldsymbol{s}}^t = \underline{\boldsymbol{z}}^{t-1}$ $t$ : $t$-te Taktperiode.

Verbesserung der Einstellbarkeit und Beobachtbarkeit durch Zusatzlogik:
Zusätzlicher Testeingang: Mehr Platzbedarf, mehr Leistungsaufnahmen.
Zusätzlicher Ausgang: Zusätzlicher Pin, langsameres Signal.

## 10. Auch wichtig

**Schrödingers Katze**