

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 Vors	atz Hz	s^{-1}
10 ¹²	Т	10 ⁻¹ d	N N	kgms ⁻²
10^{9}	G	10^{-2} c	J	Nm = VAs
10^{6}	М	10^{-3} m		$VA = Js^{-1}$
10^{3}	k	10^{-6} μ	C	As
$\frac{10^2}{10^1}$	h	10 ⁻⁹ n	V	JC^{-1}
10^{1}	da	10 ⁻¹² p	F	CV^{-1}
	ı	10^{-15} f	Ω	VA^{-1}
		ı	H	VsA^{-1}

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

3 Polyadische Zahlensysteme

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

i=-n Z:Zahl, r:Basis, $d_i:$ Ziffer, p:#Ziffern vorne n:#Nachkommastellen Binäres Zahlensystem:

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

 $\begin{array}{c|c} & & & \\ i=-n & & & \\ \text{Octalsystem:} & & & \\ d_{i8} \in 0,1,2,3,4,5,6,7 & & \\ d_{i16} \in 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F \\ \end{array}$

Benötigte Bits: N:n Bit M:2n Bit N+M=2n+1 Bit

N + M = 2n + 1 $N \cdot M = 3n$ Bit

3.1 Umrechnung

	$Z \ge 1$	Z < 1
$r \rightarrow 10$		$Z_{10} = \sum r^{-i} \cdot d_{-i}$ 0.11 ₂ \rightarrow 1 \cdot 0.5 + 1 \cdot 0.25
$10 \rightarrow r$	$d_i = Z_{10}\%r^i$	
	$ \begin{vmatrix} d_i = Z_{10} \% r^i \\ 58/8 = 7 \text{Rest } 2(LSB) \\ 7/8 = 0 \text{ Rest } 7(MSB) \end{vmatrix} $	$0.4 \cdot 2 = 0.8$ Übertrag $0(MSB)$
	7/8 = 0 Rest 7(MSB)	$0.8 \cdot 2 = 1.6$ Übertrag 1

3.2 Zweierkomplement Wertebereich: $-2^{n-1} \le Z \le 2^{n-1} - 1$

Wandle 2 in -2 um:

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

3.3 Gleitkommadarstellung nach IEEE 754

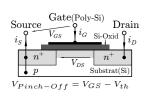
$$\begin{array}{c|c} Wert = (-1)^s \cdot 2^{e-127} \cdot 1.f & \text{Bsp:} -0.625 = -1 \cdot 2^{-1} \cdot 1.01_2 \\ s: \text{Vorzeichen, } e: \text{Exponent, } f: \text{Mantisse} & \Rightarrow s = 1, \ e = 126, \ f = 01_2 \\ \\ \text{Spezialwerte:} \ Wert = 0 \Leftrightarrow e = 0 & Wert = \infty \Leftrightarrow e = 255 \\ \\ \text{Bitverteilung(single/double):} \\ \hline \hline s(1) & e(8/11) & f(23/52) \\ \\ \hline \end{array}$$

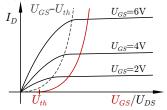
4 Halbleiter

	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

Metal Oxide Semiconductor Field Effekt Transistor





5.1 Bauteilparameter

 $\label{eq:Verstärkung:} \begin{array}{|c|c|c|} \hline \text{Verstärkung:} & \beta = K' \frac{W}{L} \text{ mit } K' = \frac{\mu \epsilon_{0x} \epsilon_{0}}{t_{0x}} \\ \hline & \text{Kanalweite} & \text{Kanallänge} & \text{L} \\ \hline & \text{Elektronenbeweglichkeit} & \mu & \mu_{n} \approx 250 \cdot 10^{-4} \frac{m^{2}}{Vs}, \, \mu_{p} \approx 100 \cdot 10^{-4} \frac{m^{2}}{Vs} \\ \hline \text{Dielektrizitätskonstante} & \epsilon_{ox} \approx 3,9 \\ \hline \text{Gateoxyddicke} & t_{ox} \\ \hline & \text{Verstärkung} & \epsilon_{0} = 8.8541878 \cdot 10^{-12} \frac{\text{A s}}{\text{V m}} \\ \hline \text{Kapazität} & \mu_{p} = \frac{\mu_{n} \varepsilon_{ox} \varepsilon_{0}}{t_{ox}} \cdot \frac{W}{L} = K' \frac{W}{L} = \frac{\mu_{n} C_{G}}{L^{2}} \\ \hline \text{C}_{G} = \varepsilon_{ox} \varepsilon_{0} \frac{WL}{t_{ox}} \\ \hline \text{Verzögerungszeit} & t_{pHL} \propto \frac{C_{L} t_{ox} L_{p}}{W_{p} \mu_{p} \varepsilon_{ox}} (V_{DD} - |V_{th}|) \\ \hline \end{array}$

- 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

V_{GS} V_{DD}	Transistor	Source liegt immer am	V_{GS}, V_{DS}, I_D	Substrat
$G \rightarrow \bigcup_{A} V_{DS}$	pMos normally on	höheren Potential	< 0	$+(V_{DD})$
C V_{GS} C	nMos normally off	niedrigeren Potential	> 0	-(GND)

Drainstromformel

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

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

Drainstromformel:

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

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

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)

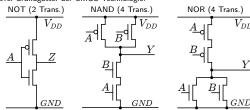
Null-Zeilen der Wertetabelle negieren und UND verknüpfen: $Z=(\overline{A}+\overline{C})\cdot(\overline{A}+\overline{D})\cdot(\overline{B}+\overline{C})\cdot(\overline{B}+D)$

5.5 Umwandlung in jeweils andere Form

- 1. Doppeltes Negieren der Funktion: $Z = \overline{\overline{A} \cdot \overline{B} + \overline{C} \cdot D}$
- 2. Umformung "untere" Negation (DeMorgan) : $Z = \overline{\overline{A} \cdot \overline{B}} \cdot \overline{\overline{C} \cdot D} = \overline{(A+B) \cdot (C+\overline{D})}$
- 3. Ausmultiplizieren: $Z = \overline{(A+B)\cdot(C+\overline{D})} = \overline{A\cdot C + A\cdot \overline{D} + B\cdot C + B\cdot \overline{D}}$
- 4. 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. ⇒ 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
 nMos
 pMos

 AND
 Serienschaltung
 Parallelschaltung

Serienschaltung

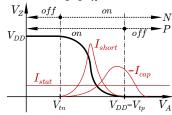
1. Möglichkeit: Direkt; ggf. Inverter vor die Eingänge und Ausgänge schalten

Parallelschaltung

2. Möglichkeit: Mit bullshit Algebra die Funktion nur mit NAND und NOR darstellen.

6.2 CMOS Verlustleistung

Inverterschaltvorgang $V_A: 0 \rightarrow 1$:



Dynamische Verlustleistung $P_{dyn} = P_{cap} + P_{short}$ Kapazitive Verluste $P_{cap} = \alpha_{01} f C_L V_{DD}^2$

Kurzschlussstrom $P_{short} = \alpha_{01} f \beta_n \tau (V_{DD} - 2V_{tn})^3$

 $\alpha_{0 \to 1} = \frac{\text{Schaltvorgänge(pos. Flanke)}}{\text{\# Betrachtete Takte}}$ Schalthäufigkeit

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_{stat} : Sub-Schwellströme, Leckströme, Gate-Ströme Abhängigkeit: $V_{DD} \uparrow: P_{stat} \uparrow V_{th} \uparrow: P_{stat} \downarrow \text{ (aber nicht proportional)}$

7 Sequentielle Logik

... Logik mit Gedächtnis. Bedingungen:

t_{Setup}	Stabilitätszeit vor der aktiven Taktflanke
t_{hold}	Stabilitätszeit nach der aktiven Taktflanke
t_{c2q}	Eingang wird spätestens nach t_{c2q} am Ausgang verfügbar
Max. Taktperiode	$t_{clk} \ge t_{1,c2q} + t_{logic,max} + t_{2,setup}$
Max. Taktfrequenz	$f_{max} = \left\lfloor rac{1}{t_{clk}} ight floor$ (Nicht aufrunden)
Holdzeitbedingung	$t_{hold} \leq t_{c2q} + t_{logic,min} ightarrow exttt{Dummy Gatter einbauen}$
Durchsatz	$\frac{1\text{Sample}}{t_{clk,pipe}} = f$
Latenz	t _{c1k} · #Pipelinestufen (das zwischen den FFs)

7.1 Pipelining

Nur bei synchronen(taktgesteuerten) Schaltungen möglich!

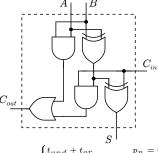
- Aufteilen langer kombinatorischer Pfade durch Einfügen zusätzlicher Registerstufen → Möglichst Halbierung des längsten Pfades
- · Zeitverhalten beachten (evtl. Dummy-Gatter einfügen)
- Durchsatz erhöht sich entsprechend der Steigerung der Taktfrequenz
- · Gesamtlatenz wird eher größer
- Taktfrequenz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

7.2 Parallel Processing

$$\mathsf{Durchsatz} = \frac{\#\mathsf{Modul}}{t_{Clk}, M_{Odul}} = f \qquad \qquad \mathsf{Latenz} = t_{clk}$$

- 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 $p_n = a_n \oplus b_n$ Summerbit $S_n = c_n \oplus p_n$

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

9 Speicherelemente

Flüchtig Speicherinhalt gehen verloren, wenn Versorgungsspannung V_{DD} wegfällt - Bsp: *RAM Nicht Flüchtig Speicherinhalt bleibt auch ohne $V_{D\,D}$ erhalten - Bsp: Flash Asynchron Daten werden sofort geschrieben/gelesen.

Synchron Daten werden erst mit $clk_{0\rightarrow 1}$ geschrieben.

 $\textbf{Dynamisch} \ \ \textbf{Ohne} \ \ \textbf{Refreshzyklen} \ \ \textbf{gehen} \ \ \textbf{auch} \ \ \textbf{bei} \ \ \textbf{angelegter} \ \ V_{DD} \ \ \textbf{Daten} \ \ \textbf{verloren} \ \ \textbf{-} \ \ \textbf{Bsp:} \ \ \textbf{DRAM}$ 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.

$${\sf Speicherkapazit\"{a}t} = {\sf Wortbreite} \cdot 2^{{\sf Adressbreite}}$$

9.1 Flip-Flop

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

9.2 Register

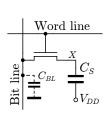
Ring aus zwei Invertern.

9.3 Latch

Set-Reset Latch:

0 an R/S schaltet. **Enable-Latch:** \ddot{a} and \ddot{a} Speicherzustand \ddot{a} auf \ddot{D}

9.4 DRAM Zelle (dynamisch)



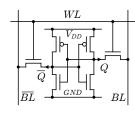
lange Bitlines ightarrow C_{BL} \uparrow , Laufzeit \uparrow quadratisch: $\frac{\text{Bit}}{\text{Zeile}} \stackrel{!}{=} \frac{\text{Bit}}{\text{Spalte}} = \frac{\text{Wort}}{\text{Zeile}} \cdot \frac{\text{Bit}}{\text{Wort}}$

- \bullet Wortleitung wird aktiviert, d.h. auf V_{DD} gelegt
- Bitleitung wird auf den gewünschten Wert (V_{DD} für 1, GND für 0) gelegt

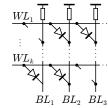
⇒ Kondensator wird auf entsprechendes Potential aufgeladen oder entladen, je nach vorherigem Wert

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

9.5 SRAM Zelle (statisch)

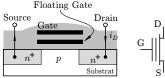


9.6 ROM - Read Only Memory



9.7 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

9.8 Organisation von Speichern

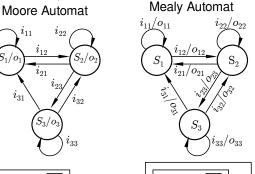
- 1 Byte besteht aus 8 Bit
- Ziel: möglichst quadratische Anordnung der Speicherzellen
- Wortbreite W berücksichtigen!

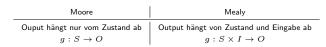
Speicherkapazität = $2^M \cdot 2^N$ Bester Fall für M = N# Reihen = N $2^M = W \cdot 2^K$ # Spalten = K

10 Automaten

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

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





10.1 Vorgehensweise

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



1 Boolsche Algebra

	Mengenalgebra	Boolesche Algebra
	$(P(G);\cap,\cup,\overline{A};G,\emptyset)$	$(0,1;\cdot,+,\overline{x})$
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	$A\cap (B\cup C)=(A\cap B)\cup (A\cap C)$	$x\cdot (y+z) = x\cdot y + x\cdot z$
	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	$x + (y \cdot z) = (x + y) \cdot (x + y)$
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

$$\begin{array}{ll} f=x\cdot a+\overline{x}\cdot b & \text{(2 Eingänge a,b und 1 Steuereingang x)} \\ f=\overline{x}_1\overline{x}_2a+\overline{x}_1x_2b+x_1\overline{x}_2c+x_1x_2d & \text{(Eingänge: a,b,c,d Steuerung: x_1,x_2)} \end{array}$$

1.2 Wichtige Begriffe

LUT: Look Up Table

Wichtige Begriffe:	Definition	Bemerkung				
Signalvariable	x	$\hat{x} \in \{0, 1\}$				
Literal	$l_i = x_i$ oder $\overline{x_i}$	$i \in I_0 = \{1,, n\}$				
Minterme,0-Kuben	$MOC ightarrow m_j = \prod_{i \in I_0} l_i$	$ M0C = 2^n$				
d-Kuben	$MC\ni c_j=\prod_{i\in I_j\subseteq I_0}l_i$	$ MC = 3^n$				
Distanz	$\delta(c_i, c_j) = \{l \mid l \in c_i \land \overline{l} \in c_j\} $	$\delta_{ij} = \delta(c_i, c_j)$				
Implikanten	$MI = \{c \in MC \mid c \subseteq f\}$					
Primimplikanten	$MPI = \{ p \in MI \mid p \not\subset c \ \forall c \in M$	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 Metho				
` ′		oder Schichtenalgorithmus				
MinSOP (min. 1)	Minimale Summe v. Primimplikanten durch Überdeckungstabelle					
FPGA: Field Programm	nable Gate Array					

1.3 Boolesche Operatoren (Wahrheitstabelle WT)

×	у	AND	OR	XOR	NAND	NOR	EQV
		$x \cdot y$	x + y	$x \oplus y$	$\overline{x \cdot y}$	x + y	$x \oplus y$
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

$$\begin{array}{l} f = x_i \cdot f_{x_i} + \overline{x}_i \cdot f_{\overline{x}_i} = (x_i + f_{\overline{x}_i}) \cdot (\overline{x}_i + f_{x_i}) = (f_{x_i} \oplus f_{\overline{x}_i}) \cdot x_i \oplus f_{\overline{x}_i} \\ \overline{f} = x_i \cdot \overline{f}_{x_i} + \overline{x}_i \cdot \overline{f}_{\overline{x}_i} \end{array}$$

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) : $f=\overline{\overline{\overline{x}\cdot\overline{y}}\cdot\overline{\overline{z}\cdot w}}=\overline{(x+y)\cdot(z+\overline{w})}$
- 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) :

 $f = \overline{xz} \cdot \overline{x\overline{w}} \cdot \overline{yz} \cdot \overline{y\overline{w}} = (\overline{x} + \overline{z}) \cdot (\overline{x} + w) \cdot (\overline{y} + \overline{z}) \cdot (\overline{y} + w)$

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)
- ullet 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	Α	1-Kubus	R	A	2-Kubus	Α	
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	$x_1x_2\overline{x}_3$	√	$x_{1}x_{3}$	$m_5 \& m_7$	√			
m_7	$x_1x_2x_3$	🗸	$x_{1}x_{2}$	$m_6 \& m_7$	√			

le 2.6 Quine's und McCluskey's Bestimmung der MinSOP

Geg: CSOP $(\sum m_i)$ und VollSOP $(\sum p_i)$ Ges: MinSOP

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_i 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^{Kubendimension}

2.8 (R)OBDD

(Reduced) Ordered Binary Decision Diagram



ROBDD → SOP: Alle Pfade zur 1 verodern: $f = x + \overline{x}y$

ROBDD -> POS: Alle Pfade zu 0 verodern, kompletten Term negieren, DeMorgan anwenden

3 Funktionale Dekomposition



Verfahren

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

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

Variablen Variablen

→ Meist kann man eine günstige Aufteilung per BDD fin-

- ullet Auswerten von $f(\hat{oldsymbol{x}},oldsymbol{y})$ und Bilder der Dekompositionsmatrix: $f = \overline{x}_1 \overline{x}_2 y_1 + \overline{x}_1 \overline{x}_2 x_3 y_1 + \overline{x}_1 x_2 \overline{x}_3 \overline{y}_1 \dots$
- Trage die Funktionswerte in die Matrix ein
- ullet Suche Spalten, die die selben Werte je $oldsymbol{y}$ haben
- Codiere gleiche Spalten mit gleichem z

freie Variablen		g	ebunden	e Variab	len $(x_1$	$, x_2, x_3$)	
(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
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:

$\hat{oldsymbol{x}_i}$	\hat{z}_i	$\underline{z} = h(x)$	$g(\underline{\boldsymbol{z}}_i,\underline{\boldsymbol{y}})$
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	$\overline{z}_1 z_2 = \overline{x}_1 x_2 \overline{x}_3 + x_1 x_2 x_3$	\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)
- → müssen in der Kompositionsfunktion auftreten
- (2) $\hat{z}_i \cdot g(z, y)$ stellen die Komposistionsfunktion dar

 $g(\underline{z}, y) = \overline{z}_1 \overline{z}_2 y_1 + \overline{z}_1 z_2 \overline{y}_1 + z_1 \overline{z}_2 \overline{y}_1 + z_1 \overline{z}_2 y_1 y_2 = \text{Kompositionsfunktion}$

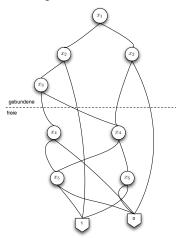
• Notationen: |x| = Zahl der Eingangsvariablen (gebunden)

|z| = Zahl der Dekompositionsvariablen $|X|=2^{\left|\underline{x}
ight|}=$ Zahl der möglichen Zustände aller gebundenen Variablen

 $|Z| = 2^{|\underline{z}|} = \text{Zahl aller Dekompositions variablen}$

3.1 Funktionale Dekomposition mit ROBDD

Ermitteln der gebundenen bzw. freien Variablen mittels BDD:



 $\mbox{ Dekompositions bedingung } \quad |\underline{\pmb{z}}| \leq |\underline{\pmb{x}}| - 1 \ \mbox{bzw.} \ |Z| \leq \frac{1}{2} \ |X|$

- Nehme immer eine Ebene an:
 Oberhalb = gebundene Variablen
 Unterhalt = freie Variablen
- Zähle die Knoten, die durch die kreuzenden Äste erreicht werden (hier mit Δ bezeichnet)
- ullet Auch wenn ein Knoten durch zwei oder mehrere Äste erreicht wird, darf er nur **einmal** gezählt werden (im Beispiel: $\Delta=4$)
- ullet Wenn $|\underline{oldsymbol{z}}| = \lceil \log_2 \Delta \rceil \leq |\underline{oldsymbol{x}}| 1 o \mathsf{DK} ext{-Bed.}$ erfüllt
- ullet Pfade zur 1 ergeben Dekompositionsfunktion (gebundene Variablen ofreie Variablen o1)

Zuordnungstabelle:

geb. Variablen	z_1	z_2	freie Variablen
111	0	0	x4x5
110, 010, 011	0	1	$x_4x_5 + \overline{x}_4\overline{x}_5$
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 $h = \left\{f \setminus c_l \cdot l\right\}$ z.B. $f = \overline{x}y + xyz + \overline{x} \cdot \overline{y} \cdot \overline{z}$ (entferne x: d.h. l = x und $c_l = yz$) $h = \overline{x}u + \overline{x} \cdot \overline{y} \cdot \overline{z}$
- $\begin{array}{l} \bullet \ \ \text{Wenn} \ h_{c_{\overline{l}} \cdot \overline{l}} = 1 \ \text{ist das Literal entfernbar} \\ \text{z.B.} \ h_{c_{\overline{l}} \cdot \overline{l}} = (\overline{x}y + \overline{x} \cdot \overline{y} \cdot \overline{z})_{\overline{x}yz} = 1 \end{array}$

3.2.3 Literal hinzufügen (reduce)

Kann man l zu c hinzufügen ohne f zu verändern?

$$\begin{array}{l} f=c+h \stackrel{?}{=} c \cdot l + h \\ \text{Zulässigkeitsbedingung: } c \cdot \overline{l} \subseteq h \\ \text{z.B. } f=xy+\overline{x}yz+xz \text{ (füge } l=\overline{z} \text{ hinzu)} \\ \text{Ist } xyz\subseteq \overline{x}yz+xz? \\ \text{Ja} \Rightarrow f^*=xy\overline{z}+\overline{x}yz+xz \end{array}$$

Gemeinsame Literalentfernung: Prüfe 2. Literal nachdem 1. Literal entfernt wurde

3.3 Strukturanalyse

Tautologie:
$$f_{x_i}=1 \land f_{\overline{x_i}}=1 \Rightarrow f=1$$
 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:
$$f=yz+xz+\overline{yz}+\overline{yz}+y\overline{z} \Rightarrow \text{monoton steigend in } x$$

$$f(x,y,z)=x \ \varphi(z)+h(y,z) \Rightarrow \text{Prüfe } h \text{ auf Tautologie}$$

4 Nützliches Wissen

4.1 Mehrfachimplikanten

Sind gleiche Implikanten in mehreren verschiedenen Funktionen vorhanden? Prüfe $f_1 \cap f_2$ auf Mehrfachimplikanten: $f_1 \cdot f_2 = 0$. Nutzung von Mehrfachimplikanten ist sinnvoll wenn die Gesamtliteralzahl heider SC

Nutzung von Mehrfachimplikanten ist sinnvoll wenn die Gesamtliteralzahl beider SOPs kleiner ist als ohne Verwendung von Mehrfachimplikanten.

4.2 VolISOP 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 $S^0 \in S$

 $\begin{array}{ll} {\sf Zustandsfkt.} & \delta: S \times I \to S, \ S_k \mapsto \delta(S_i, X_j) \\ {\sf Ausgangsfkt.} & \lambda: S \times I \to O, \ Y_l \mapsto \lambda(S_i, X_j) \\ {\sf ZA-fkt.} & \mu: S \times I \to S \times O, \ (S_k, Y_l) \mapsto \mu(S_i, X_j) \end{array}$

 $k\text{-}\ddot{\mathbf{A}}$ quivalenz $S_i\overset{k}{\sim}S_j$ 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 Einganssequenzen die Ausgaben und die Zustandsübergänge äquivalent sind.

→ Zeige: es lassen sich keine weiteren Äquivalenzklassen bilden.

6 Karnaugh- Diagramm

 $\mathsf{Zyklische} \ \mathsf{Gray-Codierung:} \ \mathsf{2dim:} 00, 01, 11, 10 \ \mathsf{3dim:} 000, 001, 011, 010, 110, 111, 101, 100$

	00				_		
0	1	0	0	0	Gleiche Zellen zusammenfassen: z.B. $\overline{xy} + y \cdot z$		
1	X	1	1	0			
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
- 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

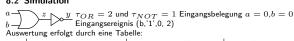
$f(x_1,\ldots,x_n)$	Schicht
$x\cdot w + \overline{x}\cdot w + x\cdot y\cdot w\cdot \overline{z} + \overline{x}\cdot y\cdot w\cdot \overline{z} + \overline{y}\cdot w\cdot \overline{z}$	0
$+w+y\cdot w\cdot \overline{z}$	1
$+w\cdot \overline{z}$	2
+w	3

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



t	a	b	Z	У	ausgewertete Elemente	neue Ereignisse
0	'0'	'0'	'1'	'0'	init	(b,'1',0, 2)
2		'1'			OR	(b,'1',0, 2) (z,'1',2,4)
4			'1'		NOT	(y,'0',4,5)
5				'0'		
- · ˈ	. /1		c.		· C · · · · · · ·	

Ereignis: (betroffenes Signal, neuer Signalwert, t, t + au)

8.3 Delay

- ullet transport delay: Verzögerung um au_{pd}
- $\bullet\,$ inertial delay: Verzögerung um τ_{pd} und Impulse die kleiner als τ_{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 $F_{\nu}=\{f_{\mu}\in F\,|\,t_{\nu}Rf_{\mu}\}$: Menge aller Fehler die vom Test t_{ν} erkannt werden. Fehleranzahl = $2\cdot$ Signalanzahl = $2\cdot$ Eingänge + Interne Signale + Ausgänge) Testgruppe $T_{\mu}=\{t_{\nu}\in T\,|\,t_{\nu}Rf_{\mu}\}$ ist die Menge aller Tests die den Fehler f_{μ} 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(x))

$$y_z = y(z, \underline{\boldsymbol{x}}) \oplus y(\overline{z}, \underline{\boldsymbol{x}}) \mid \hat{=} y(z=1) \oplus y(z=0)$$

9.2.1 Rechenregeln

$$\begin{array}{ll} y_x = 0 \ \ \text{falls} \ y \neq f(x) & (z \cdot w)_x = z \cdot w_x \oplus z_x \cdot w \oplus z_x \cdot w_x \\ y_y = 1 & (z + w)_x = \overline{z} \cdot w_x \oplus z_x \cdot \overline{w} \oplus z_x \cdot w_x \\ (\overline{y})_x = y_x & y_x = y_z \cdot z_x \ \ \text{falls} \ y = y(z(x)) \\ (z \oplus w)_x = z_x \oplus w_x & (y_z)_w = (y_w)_y \end{array}$$

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 $x + y = x \oplus y \oplus xy$

$$\text{Test f\"{u}r} \begin{cases} a/0: a \cdot y_a \overset{!}{=} 1 \\ a/1: \overline{a} \cdot y_a \overset{!}{=} 1 \end{cases} \quad \Rightarrow \text{Testmuster finden}.$$

9.2.3 XOR-Regeln

$$\begin{array}{ll} x \oplus y = \overline{x} \cdot y + x \cdot \overline{y} & x + y = x \cdot y \oplus x \oplus y \\ x \oplus y = (x + y) \cdot (\overline{x} + \overline{y}) & x \cdot y = x \oplus y \oplus (x + y) \\ x \cdot (y \oplus z) = x \cdot y \oplus x \cdot z & \overline{x} = x \oplus 1 \\ x \oplus x = 0 & x \oplus 0 = x \\ (x + y) \oplus y = x \cdot \overline{y} & x \overline{y} + yz = x \overline{y} \oplus yz \end{array}$$

9.2.4 Schaltnetze



$$\begin{bmatrix} y_x = y_z z_x \overline{w}_x + y_w w_x \overline{z}_x + z_x w_x \cdot [\overline{z} \circ \overline{w} \oplus z \circ w] \\ \circ = \mathsf{XOR}/\mathsf{XNOR} \Rightarrow [\overline{z} \circ \overline{w} \oplus z \circ w] = 0 \\ \circ = \mathsf{AND}/\mathsf{NAND}/\mathsf{OR}/\mathsf{NOR} \Rightarrow [\overline{z} \circ \overline{w} \oplus z \circ w] = \overline{z} \oplus w \end{bmatrix}$$

Schaltnetz mit Rekonvergenzmasche (4 Fälle):

- \Rightarrow Einfachpfadsensibilisierung: x-z-y1. $y_z z_x \overline{w}_x = 1$
- 2. $y_w w_x \overline{z}_x = 1$ \Rightarrow Einfachpfadsensibilisierung: x-w-y
- $z_x w_x \cdot [\overline{z} \circ \overline{w} \oplus z \circ w] = 1 \implies \mathsf{Mehrfachpfadsensibilisierung}$
- $z_x w_x \cdot \overline{[\overline{z} \circ \overline{w} \oplus z \circ w]} = 1 \implies \text{Selbstmaskierung}$

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:

- Boolsche 0
- Boolsche 1 1
- Xundefined
- D1 falls fehlerfrei, 0 falls fehlerhaft (teste stuck-at 0)
- \overline{D} 0 falls fehlerfrei, 1 falls fehlerhaft (teste stuck-at 1)

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

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

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

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

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	\boldsymbol{x}
Testpunkte	Y
nächste Schaltwerkszustände	\boldsymbol{z}
Schaltwerkszustände	${m s}$
4 4 1	

Es gilt: $s^t = z^{t-1}$ 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