Digitale Schaltungen

Zahlensysteme

Größte darstellbare Zahl nach Anzahl der Ziffern

b: Basis; p: Anz. Vorkommast.; n: Anz. Nachkommast.

$$Z_{max} = b^p - b^{-n}$$

Zahlenkonvertierung Dezimal nach Basis r

Sei z Zahl im Dezimalsystem.
Für $z \geq 1$

1) Teile z durch r. Rest ist letzte Ziffer (LSB).
2) Wiederhole mit Ergebnis von 1). Rest ist vorletzte Ziffer usw.
3) letzter Rest ist erste Ziffer (MSB).

Für $z < 1$

1) Multipliziere z mit r. Übertrag ist erste Ziffer (MSB).
2) Wiederhole mit Ergebnis von 1). Übertrag ist zweite Ziffer usw.
3) letzter Rest ist letzte Ziffer (LSB).

Anzahl der benötigten Ziffern n bezüglich der Basis r, um Z_{10} darstellen zu können:

$$n = \lfloor log_r(Z) \rfloor + 1$$

Negative Zahl bilden (2er Komplement, Radix)

$$K(Z) = r^n - Z$$

1) Betrag der Dezimalzahl in Binärsystem umwandeln
2) Binäre Zahl invertieren
3) Anschließend 1 addieren

Binäre Rechenoperationen

n_1: Bit-Anzahl von Zahl 1; n_2: Bit-Anzahl von Zahl 2
Maximale Anzahl an Bit, um Ergebnis darzustellen:

Addition: $n_E = max(n_1, n_2) + 1$
Multiplikation: $n_E = n_1 + n_2$

Gleitkommazahlen (IEEE 754)

1 bit Vorzeichen(v), 8 bit Exponent(e), 23 bit Mantisse(m)

Dezimalzahl → IEEE 754

Z positiv $\Rightarrow v = 0$; Z negativ $\Rightarrow v = 1$

$$e_{10} = \lfloor \log_2 |Z_{10}| \rfloor + 127 \qquad m_{10} = \lfloor (\frac{|Z_{10}|}{2^{e_{10}-127}} - 1) \cdot 2^{23} \rfloor$$

IEEE 754 → Dezimalzahl

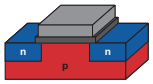
$$Z_{10} = (-1)^v \cdot (\frac{m_{10}}{2^{23}} + 1) \cdot 2^{e_{10}-127}$$

$$Z_2 = (-1)^v \cdot (1, m_2) \cdot 2^{e_{10}-127}$$

$e = 0$ ⇒Darstellung von „0"
$e = 255$ ⇒Darstellung von „∞"

MOSFET-Transistor

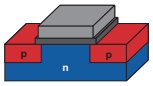
nMOS



Guter Pull-Down
Source am niedrigeren Potential ($U_{DS} > 0$)

$$I_D = \begin{cases} 0 & U_{GS} < U_t(aus) \\ & \wedge\, U_{DS} \geq 0 \\ \beta \left(U_{GS} - U_t - \frac{U_{DS}}{2}\right) U_{DS} & U_{GS} > U_t \text{ (linear)} \\ & \wedge\, 0 < U_{DS} < U_{GS} - U_t \\ \frac{\beta}{2} \left(U_{GS} - U_t\right)^2 & U_{GS} > U_t \text{ (Sättigung)} \\ & \wedge\, 0 < U_{GS} - U_t < U_{DS} \end{cases}$$

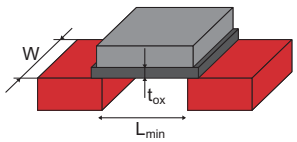
pMOS



Guter Pull-Up
Source am höheren Potential ($U_{DS} < 0$)

$$I_D = \begin{cases} 0 & U_{GS} > U_t(aus) \\ & \wedge\, U_{DS} \leq 0 \\ -\beta \left(U_{GS} - U_t - \frac{U_{DS}}{2}\right) U_{DS} & U_{GS} < U_t \text{ (linear)} \\ & \wedge\, 0 > U_{DS} > U_{GS} - U_t \\ \frac{-\beta}{2} \left(U_{GS} - U_t\right)^2 & U_{GS} < U_t \text{ (Sättigung)} \\ & \wedge\, 0 > U_{GS} - U_t > U_{DS} \end{cases}$$

Dimensionierung



$$\beta = K' \frac{W}{L} \text{ mit } K' = \frac{\mu \varepsilon_{Ox} \varepsilon_0}{t_{Ox}}$$

ε_{Ox}: rel. Dielektrizitätskonstante des Gate Oxyds
ε_0: Dielektrizitätskonstante
W: Kanalweite L: Kanallänge
μ: Beweglichkeit der Elektronen/Löcher
t_{Ox}: Gate Oxyd-Dicke

L immer L_{min}

$W \sim$ Drain Strom $\sim$ Schaltgeschwindkeit

$\mu_n = (1.5 \ldots 3.5) \cdot \mu_p \Rightarrow$ Kanalweite bei pMOS größer

$\Rightarrow$ nMOS schaltet prinzipiell schneller als pMOS

Verzögerungszeit (Propagation delay)

Zeit zwischen 50%-Pegeln von Eingang und Ausgang.

$$t_p = R_{on,p} \cdot C \cdot \ln(2) \quad , \quad R_{on,p} \approx \frac{1}{\beta(|U_{GSp}| - |U_{tp}|)}$$

$$t_{pLH} \sim \frac{C_{load} t_{Ox} L_p}{W_p \mu_p \varepsilon_{Ox} (U_{DD} - |U_{tp}|)}$$

Zunahme von:

C_{load} : Kapazitive Last	W_p : Kanalweite
t_{ox} : Oxiddicke	μ_p : Beweglichkeit der Ladungsträger
L_p : Kanallänge	ε_{ox} : Oxyd-Dielektrizität
U_{tp} : Schwellspannung (Betrag)	U_{DD} : Versorungsspannung
⇒ Verzögerungszeit steigt	⇒ Verzögerungszeit sinkt

Beeinflussbare Parameter: $C_{load}, W_p, U_{DD}, U_{tp}$

CMOS-Inverter und Logik

Verlustleistung

Dynamisch

Abhängig von den Signalflanken und der Schaltfunktion

Kapazitiver Anteil:
Eingangskapazitäten nachfolgender Gatter, Leitungskapazitäten, parasitäre Kapazitäten

Kurzschlussanteil:
Querstrom (aufgrund endlicher Flankensteilheit)

Statisch

Sub-Schwellstrom, Leckstrom (Diodensperrstrom), Gate-Strom
⇒ abhängig von Versorgungs- und Schwellspannung

$P_{dyn} \approx P_{stat}$
$P_{Cap} = \alpha_{01} f C U_{dd}^2$
$P_{Short} = \alpha_{01} f \beta_n \tau (U_{dd} - 2U_{tn})^3$

$\alpha = \frac{Anzahl\ der\ steigenden\ Flanken\ des\ Signals\ Z}{Anzahl\ der\ steigenden\ Flanken\ des\ Taktsignals}$

Falls Schaltwahrscheinlichkeit gegeben ist:

$\alpha = P(Z=0) \cdot P(Z=1)$

Singlecore vs. Multicore: dyn. Verlustleistung

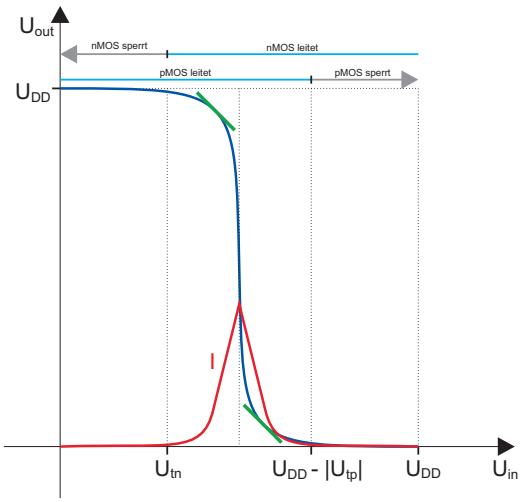
n: Anzahl der Kerne
k: um diesen Faktor verkürzte Laufzeit

Singlecore: $P_{dyn} \sim f \cdot U_{dd}^2$

Annahme: Programm kann perfekt parallelisiert werden.

Multicore: $P_{dyn} \sim \frac{k^3 \cdot f \cdot U_{dd}^2}{n^2}$

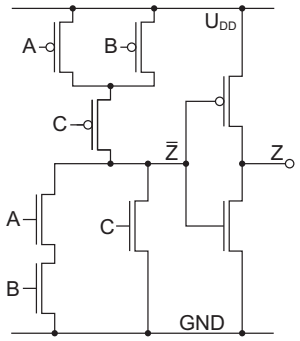
Kennlinie von CMOS-Inverter



v_{out} hat definierten Logikpegel, wenn gilt: $|\frac{\partial v_{out}}{\partial v_{in}}| < 1$

Kombinatorische Logik

Realisierung von CMOS-Schaltungen



Beispiel: $Z = AB + C$

CMOS ist grundsätzlich invertierend!

DNF und KNF

DNF (= CSOP):
1-Zeilen mit ODER verknüpfen

$Z = \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot C$

KNF (= POS):
Werte der 0-Zeilen invertieren und mit UND verknüpfen

$Z = (A + B + C) \cdot (A + \overline{B} + \overline{C}) \cdot$
$\cdot (\overline{A} + B + \overline{C}) \cdot (\overline{A} + \overline{B} + C)$

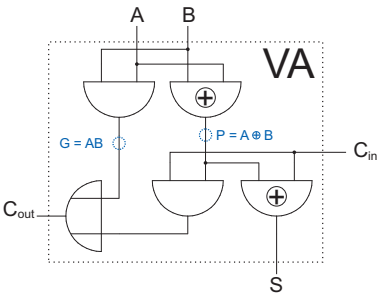
A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Umwandlung von SOP nach POS

1) Doppeltes Negieren der Funktion
2) Umformung der unteren Negation (2x DeMorgan)
3) Ausmultiplizieren
4) Umformung der oberen Negation (2x DeMorgan)

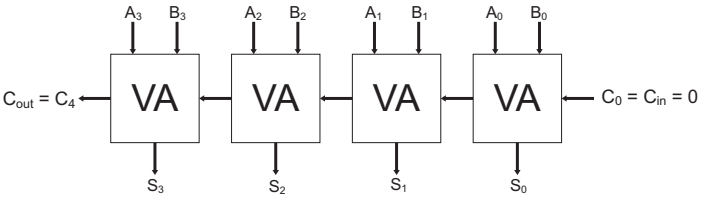
Volladdierer

A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$S = P \oplus C_{in}$, $C_{out} = G + P \cdot C_{in}$

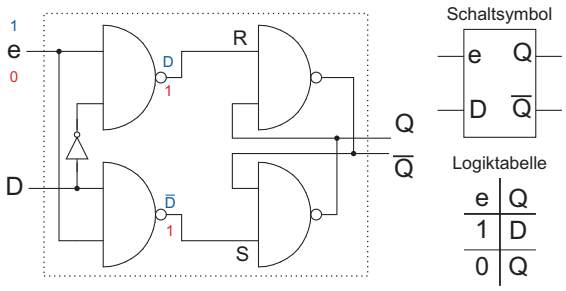
Ripple-Carry-Adder



Verzögerungszeit wird vom Carry-Übertrag dominiert!
Maximale Verzögerungszeit, wenn beim LSB das Signal von G wechselt und bei allen anderen Gattern gilt: $P = 1$.

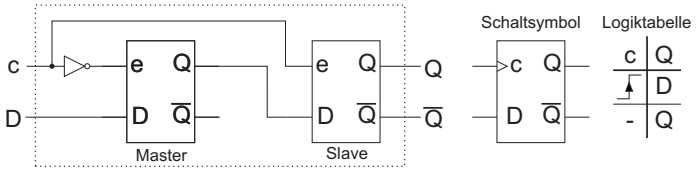
Sequentielle Logik

Set-Reset Latch / Enable Latch



Übernimmt Eingangswert, solange e = 1 (Level-controlled)

Flip-Flop



Timing

t_{setup}: Zeit, in der der Eingangswert vor aktiver Taktflanke stabil sein muss

t_{hold}: Zeit, in der der Eingangswert nach aktiver Taktflanke stabil bleiben muss

t_{c2q}: Zeit, nach der der Eingangswert nach der Taktflanke stabil an Q anliegt (=Ausgangslatenz des Registers)

t_{clk}: Clock

$t_{längsterPfad}$: (=kritischer Pfad) Längste Verzögerungszeit zwischen zwei Registerstufen

$$\boxed{t_{clk} \geq t_{c2q} + t_{längsterPfad} + t_{setup}}$$

$$\boxed{t_{hold} \leq t_{c2q} + t_{kürzesterPfad}}$$

$$f_{max} = \frac{1}{t_{setup} + t_{c2q} + t_{längsterPfad}}$$

(Für maximalen Durchsatz die Einheit „op/s“ verwenden!)

$Gesamtlatenz$ = (Maximale Anzahl hintereinander geschalteter Register -1) $\cdot\, t_{clk}$

Für eine dauerhaft korrekte Datenübergabe müssen alle Register mit der selben Frequenz arbeiten (ansonsten: Verletzung der Hold-/Setup-Zeit usw.).

Pipelining

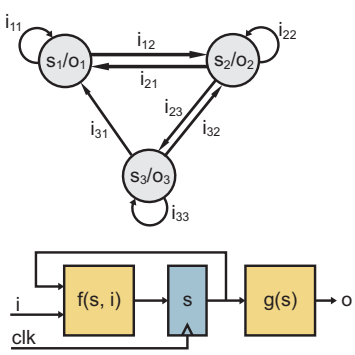
Aufteilen langer kombinatorischer Pfade durch Einfügen zusätzlicher Registerstufen, um die Taktfrequenz erhöhen zu können (Gesamtlatenz wird allerdings nicht kleiner).
⇒ Möglichst Halbierung des längsten Pfades!
⇒ Evtl. müssen sog. „Dummy-Gatter“ eingefügt werden!

Parallele Verarbeitungseinheiten

- 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

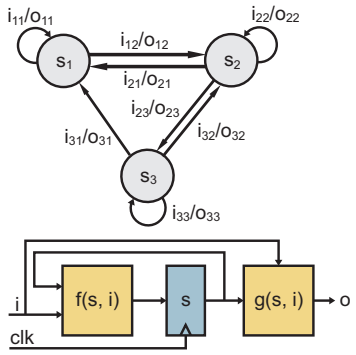
Finite state machine

Moore



Ausgang nur vom Zustand abhängig.

Mealy



Ausgang vom Zustand und von Eingabe abhängig.

Vorteile:

Kein Kombinatorischer Pfad von Eingängen zu Ausgängen ⇒ Begrenzung der Logik-Tiefe	Weniger Zustände, Übersichtlicher, Allgemeiner

Nachteile:

Hohe Anzahl von Zuständen	Lange kombinatorische Pfade bei Vekettung mehrerer FSMs

Speicher

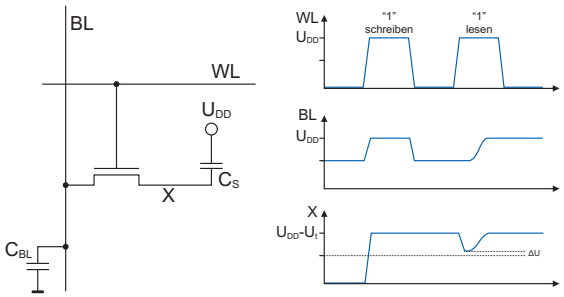
Definitionen

Bandbreite: Datenmenge pro Zeiteinheit zum Schreiben oder Lesen [bit/sec]
Latenz: Zeitdifferenz zwischen Anforderung und Ausgabe von Daten [sec]
Zykluszeit: Zeitdifferenz zwischen auffeinander folgenden Schreib/Lese Zyklen [sec]
Asynchron: Lese-/Schreibvorgang beginnt direkt
⇒ neues Datenwort kann unmittelbar am Ausgang anliegen
Synchron: Fester Systemtakt (→ Fließbandverarbeitung)
⇒ Datenwort ist frühestens mit nächstem Takt nach Anlegen einer neuen Adresse zu erwarten

Bei der Anordnung von Speicherzellen in Reihen und Spalten wird darauf geachtet, dass es möglichst quadratisch ist.

Leseverstärker zwischen Speicherzelle und Decoder beschleunigen den Speicherzugriff.

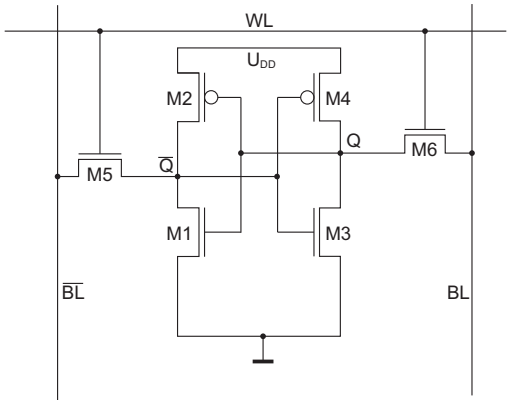
1-Transistor DRAM-Zelle



$$\Delta U = (U_X - \frac{U_{DD}}{2}) \cdot \frac{C_S}{C_S + C_{BL}}$$

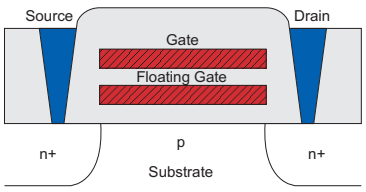
Signalverstärkung und Refresh-Zyklen notwendig.

6-Transistor SRAM-Zelle



Die Transitoren M1 - M4 realisieren ein CMOS-Latch zur Speicherung einer "0"bzw. "1". Über die Transistoren M5 und M6 wird die Speicherzelle zum Lesen oder Schreiben ausgewählt.

Flash



Lesen

Information sitzt auf dem Floating Gate. Je nach Zustand (1/0) sitzen dort Ladungsträger oder nicht. Evtl. vorhandene Ladungsträger bewirken eine Änderung der Schwellspannung des Transistors, der bei Anlegen einer Gatespannung je nach Speicherzustand durchschaltet oder nicht.

Schreiben

An Gate und Drain wird eine hohe Programmierspannung angelegt (Source auf GND). Dadurch werden die Elektronen stärker beschleunigt als normal und tunneln zum Floating Gate.

Löschen

An Drain wird eine hohe Spannung und an Gate GND angelegt (Source von GND abgetrennt). Die Elektronen werden dadurch von Drain abgezogen (tunneln durch das Oxyd).

EDS

Algebra

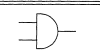
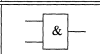
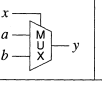
Boolesche Algebra

1) $x \cdot y = y \cdot x, x + y = y + x$
2) $(x \cdot y) \cdot z = x \cdot (y \cdot z)\ ,\ \ (x + y) + z = x + (y + z)$
3) $x \cdot (y + z) = x \cdot y + x \cdot z$
4) $x \cdot x = x\ ,\ \ x + x = x$
5) $x \cdot (x + y) = x\ ,\ \ x + x \cdot y = x$
6) $1 \cdot x = x\ ,\ \ 0 + x = x$
7) $0 \cdot x = 0\ ,\ \ 1 + x = 1$
8) $x \cdot \overline{x} = 0\ ,\ \ x + \overline{x} = 1$
9) $\overline{\overline{x}} = x$
10) $\overline{x \cdot y} = \overline{x} + \overline{y}\ ,\ \ \overline{x + y} = \overline{x} \cdot \overline{y}$

XOR Algebra

1) $x \oplus y = y \oplus x$
2) $(x \oplus y) \oplus z = x \oplus (y \oplus z)$
3) $x \cdot (y \oplus z) = x \cdot y \oplus x \cdot z$
4) $x \oplus 0 = x$
5) $\overline{x} = x \oplus 1\ ,\ \ \overline{x \oplus y} = x \oplus y \oplus 1$
6) $x \oplus \overline{x} = 1\ ,\ \ x \oplus x = 0$
7) $x \oplus x \oplus x = x$
8) $x \oplus y = \overline{x} \oplus \overline{y}$
9) $x \oplus y = x \cdot \overline{y} + \overline{x} \cdot y$
10) $x \oplus y = (x + y) \cdot (\overline{x} + \overline{y})$
11) $x + y = x \oplus y \oplus x \cdot y$
12) $x \cdot y = x \oplus y \oplus (x + y)$

Schaltsymbole

	Schaltungssymbol amerikanisch	Schaltungssymbol deutsch	Schaltungssymbol DIN-Norm 40900	Verknüpfung, Abbildungsvorschrift	Funktion, Eins-Menge
UND AND				$y = f(x_1,x_2) = x_1 \cdot x_2$	$f = \{11\}$
ODER OR				$y = f(x_1,x_2) = x_1 + x_2$	$f = \{01,10,11\}$
NICHT NOT				$y = f(x) = \overline{x}$	$f = \{0\}$
NAND				$y = f(x_1,x_2) = \overline{x_1 \cdot x_2}$	$f = \{00,01,10\}$
NOR				$y = f(x_1,x_2) = \overline{x_1 + x_2}$	$f = \{00\}$
XOR (exkl. ODER)				$y = f(x_1,x_2) = x_1 \oplus x_2$	$f = \{01,10\}$
XNOR (Äquivalenz)				$y = f(x_1,x_2) = \overline{x_1 \oplus x_2}$	$f = \{00,11\}$
Subjunktion (Implikation)				$y = f(x_1,x_2) = x_1 \rightarrow x_2 = \overline{x}_1 + x_2$	$f = \{00,01,11\}$
MUX		x: Selektoreingang a,b: Dateneingänge		$y = \beta(x,a,b) = x \cdot a + \overline{x} \cdot b$	$\beta = \{001,011,110,111\} = \{0*1,11*\}$

Binäre Boolsche Funktionen

Kofaktor

- $f|_{x_i=1} = f_{x_i}$, $f|_{x_i=0} = f_{\overline{x}_i}$
- $(f_{x_i})_{x_j} = (f_{x_j})_{x_i} = f_{x_i x_j}$
- $x_i \cdot f = x_i \cdot f_{x_j}$, $\overline{x}_i \cdot f = \overline{x}_i \cdot f_{\overline{x}_j}$
- $x_i + f = x_i + f_{\overline{x}_j}$, $\overline{x}_i + f = \overline{x}_i + f_{x_j}$

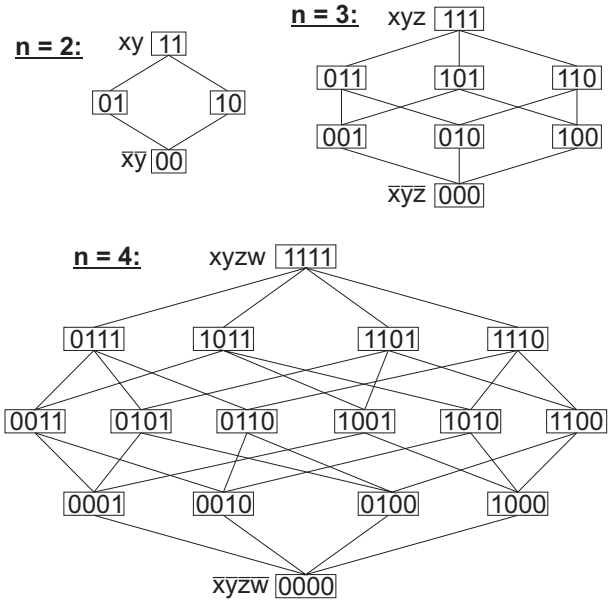
Entwicklungssätze

- $f = x_i \cdot f_{x_i} + \overline{x}_i \cdot f_{\overline{x}_i} = \beta(x_i, f_{x_i}, f_{\overline{x}_i})$
- $\overline{f} = x_i \cdot \overline{f_{x_i}} + \overline{x}_i \cdot \overline{f_{\overline{x}_i}}$

Sonstiges

- f unabhängig von $x_i \Leftrightarrow f_{x_i} = f_{\overline{x}_i} \Leftrightarrow f_{x_i} \oplus f_{\overline{x}_i} = 0$
- f abhängig von $x_i \Leftrightarrow f_{x_i} \neq f_{\overline{x}_i} \Leftrightarrow f_{x_i} \oplus f_{\overline{x}_i} = 1$
- f positiv symmetrisch in x_i und $x_j \Leftrightarrow f_{x_i \overline{x}_j} = f_{\overline{x}_i x_j}$
- f negativ symmetrisch in x_i und $x_j \Leftrightarrow f_{x_i x_j} = f_{\overline{x}_i \overline{x}_j}$

Kubengraphen



Menge aller Knoten (0-Kuben) $= 2^n$
Menge aller Kuben $= 3^n$
Menge aller Kanten $= \frac{n}{2} \cdot 2^n$

Kubenabstand $\delta(c_1, c_2)$:
Anzahl(#) an Literalen, die in c_1 negiert und in c_2 nicht negiert vorkommen und umgekehrt.
#Kubusliterale = #Raumdimensionen - #Kubusdimension
#Überdeckte Minterme $= 2^{Kubusdimension}$

Begriffe

Cover

z.B.: $f = y \cdot z + \overline{x} \cdot y + \overline{x} \cdot \overline{y}$

$\Rightarrow cov(f) = \{y \cdot z, \overline{x} \cdot y, \overline{x} \cdot \overline{y}\}$

MinSOP nach Quine/McCluskey

Nachteile

1) Benötigt CSOP als Eingangsform

2) Länge wächst exponentiell mit Dimension

Gesetze

Spezielles Resolutionsgesetz

$x \cdot a + \overline{x} \cdot a = a$

Absorptionsgesetz

$a + a \cdot b = a$

Minimierungstabelle

m_i	0-Kubus	A	1-Kubus	A	...
m_0	$\overline{x} \cdot \overline{y} \cdot \overline{z}$	✓	$\overline{x} \cdot \overline{z}$		
m_2	$\overline{x} \cdot y \cdot \overline{z}$	✓	$\overline{x} \cdot y$		
m_3	$\overline{x} \cdot y \cdot z$	✓	$y \cdot z$		
m_7	$x \cdot y \cdot z$	✓			

$\Rightarrow$ VollSOP : $f = \overline{x} \cdot \overline{z} + \overline{x} \cdot y + y \cdot z = p_1 + p_2 + p_3$

Minimierung der Primimplikanten

Algebraisch

$C = (m_0 \subset p_1) \cdot (m_2 \subset p_1 + m_2 \subset p_2) \cdot (m_3 \subset p_2 + m_3 \subset p_3) \cdot (m_7 \subset p_3)$
$C = \tau_1 \cdot (\tau_1 + \tau_2) \cdot (\tau_2 + \tau_3) \cdot \tau_3 \stackrel{!}{=} 1$
$C = \tau_1 \cdot \tau_3 + \tau_1 \cdot \tau_2 \cdot \tau_3$
$C = \tau_1 \cdot \tau_3$

Überdeckungstabelle

$p \backslash m$	m_0	m_2	m_3	m_7
p_1	1	1	0	0
p_2	0	1	1	0
p_3	0	0	1	1

$\Rightarrow$ MinSOP : $f = p_1 + p_3 = \overline{x} \cdot \overline{z} + y \cdot z$

Resolventenmethode

Gesetze

Allgemeines Resolutionsgesetz

$x \cdot a + \overline{x} \cdot b = x \cdot a + \overline{x} \cdot b + a \cdot b$

Absorptionsgesetz

$a + a \cdot b = a$

f	Schicht
$\overline{x} \cdot y + \cancel{x \cdot y \cdot z} + \cancel{\overline{x} \cdot \overline{y} \cdot z}$	0
$+y \cdot z + \overline{x} \cdot \overline{z}$	1

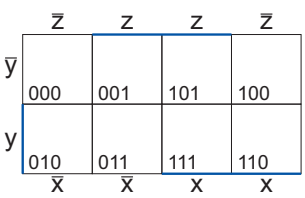
Mehrfachimplikanten

Implikanten, die in mehreren Funktionen vorhanden sind, können gemeinsam genutzt werden.
⇒ Nur sinnvoll, wenn dadurch die Gesamtliteralzahl sinkt.

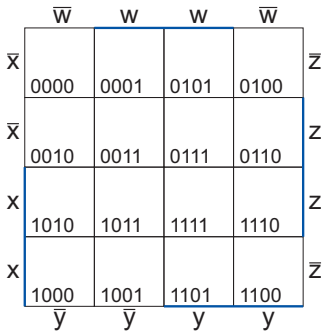
1) Zur Vereinfachung MinSOP von $f_1, f_2, ...$ bestimmen
2) Mehrfachimplikant $= f_1 \cdot f_2 \cdot ...$

Karnaugh-Diagramm

n = 3:



n = 4:



Heuristische Minimierung

Literalentfernung (expand)

z.B.: Dürfen $\overline{y}$ und $\overline{z}$ aus $\overline{y} \cdot \overline{z} \cdot \overline{w}$ einzeln bzw. gemeinsam entfernt werden?

- $\overline{y}$ darf entfernt werden, falls $f_{y \cdot \overline{z} \cdot \overline{w}} = 1$
- $\overline{z}$ darf entfernt werden, falls $f_{\overline{y} \cdot z \cdot \overline{w}} = 1$
- $\overline{y}$ und $\overline{z}$ dürfen entfernt werden, falls $f_{y \cdot z \cdot \overline{w}} = 1$

Kubenentfernung (remove)

z.B.: Darf der Kubus $\overline{x}w$ von f entfernt werden?

1) $\Rightarrow \overline{x}w$ von f entfernen $\rightarrow h$
2) $\overline{x}w$ darf entfernt werden, falls $h_{\overline{x}w} = 1$

Literalzahlerhöhung (reduce)

→ Zum Verlassen lokaler Minima.

z.B.: Darf man zum Kubus xy von $f = xy + \overline{x}yz + xz$ das Literal $\overline{z}$ hinzufügen?

1) $\Rightarrow xy$ von f entfernen $\rightarrow h$
2) Darf hinzugefügt werden, wenn $xyz \subseteq h$ (In diesem Fall → Ja)

Strukturanalyse

Monoton steigende Funktion

f monoton steigend in $x_i \Leftrightarrow f_{\overline{x}_i} \subset f_{x_i} \Leftrightarrow f_{x_i} = f_{x_i} + f_{\overline{x}_i}$
$\Leftrightarrow \overline{x}_i$ kommt in Funktion nicht vor!

Falls f monoton steigend, kann f aufgeteilt werden:

$$f = x_i \cdot \underbrace{f_{x_i}}_{\varphi} + \underbrace{f_{\overline{x}_i}}_{g}$$

Monoton fallende Funktion

f monoton fallend in $x_i \Leftrightarrow f_{x_i} \subset f_{\overline{x}_i} \Leftrightarrow f_{\overline{x}_i} = f_{\overline{x}_i} + f_{x_i}$
$\Leftrightarrow x_i$ kommt in Funktion nicht vor!

Falls f monoton fallend, kann f aufgeteilt werden:

$$f = \overline{x}_i \cdot \underbrace{f_{\overline{x}_i}}_{\varphi} + \underbrace{f_{x_i}}_{g}$$

Tautologie

$f = 1 \Leftrightarrow f_{x_i} = 1 \wedge f_{\overline{x}_i} = 1$

Falls f monoton steigend oder fallend: $f = 1 \Leftrightarrow g = 1$

Funktionale Dekomposition

Notationen

- $|x|$: Anzahl der gebundenen Eingangsvariablen
- $|X|$: Anzahl aller Zustände der geb. Eingangsvariablen
- $|z|$: Anzahl der Dekompositionsvariablen
- $|Z|$: Anzahl aller Zustände der Dekompositionsvar.

Schritt 1

Auswerten von $f(\underline{x}, \underline{y})$ (Wahrheitswertetabelle)

$\underline{x}$: gebundene Variablen, $\underline{y}$: freie Variablen

$w = \overline{x}_1 \cdot \overline{x}_3 \cdot \overline{y}_1 + x_3 \cdot \overline{y}_2 + \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \cdot y_1 +$
$\overline{x}_2 \cdot \overline{x}_3 \cdot \overline{y}_1 + x_1 \cdot x_2 \cdot \overline{y}_2 + x_1 \cdot x_2 \cdot x_3 \cdot y_1$

$\hat{y}$ \ $\hat{x}$	000	001	010	011	100	101	110	111
00	1	1	1	1	1	1	1	1
01	1	0	1	0	1	0	0	0
10	0	1	0	1	0	1	1	1
11	0	1	0	0	0	0	0	1

Anzahl der benötigten Variablen z: $|z| = \lceil log_2(|Z|) \rceil$

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

Schritt 2

Konstruktion der Dekompositionsfunktion

$\underline{z} = \underline{h}(\underline{x})$ Dekompositionsfunktion

(Willkürliche) Zuordnung von Belegungen $(\hat{z}_1, \hat{z}_2, ...)$:

$\hat{\underline{x}} \in X$	$f(\hat{\underline{x}}, \underline{y})$	$\hat{\underline{z}}$
000, 010, 100	$\overline{y}_1$	00
001, 111	$y_1 + \overline{y}_2$	10
011, 101, 110	$\overline{y}_2$	11

$z_1 = \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + \overline{x}_1 \cdot x_2 \cdot x_3 + x_1 \cdot \overline{x}_2 \cdot x_3 +$
$x_1 \cdot x_2 \cdot \overline{x}_3 = ... = x_1 \cdot x_2 + x_3$
$z_2 = \overline{x}_1 \cdot x_2 \cdot x_3 + x_1 \cdot \overline{x}_2 \cdot x_3 + x_1 \cdot x_2 \cdot \overline{x}_3$

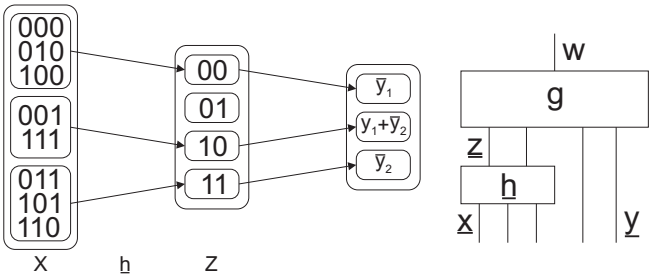
Schritt 3

Konstruktion der Kompositionsfunktion

$w = g(\underline{z}, \underline{y})$ Kompositionsfunktion

$w = \overline{z}_1 \cdot \overline{z}_2 \cdot \overline{y}_1 + z_1 \cdot \overline{z}_2 \cdot (y_1 + \overline{y}_2 \cdot) + z_1 \cdot z_2 \cdot \overline{y}_2$
$w = ... = \overline{z}_1 \cdot \overline{z}_2 \cdot \overline{y}_1 + z_1 \cdot \overline{z}_2 \cdot y_1 + z_1 \cdot \overline{y}_2$

Graphische Darstellung



Finite state machine (FSM)

FSM= $(S, I, O, \delta, \lambda, S^0)$ S endliche Zustandsmenge, I Menge der Eingangsmuster, O Menge der Ausgangsmuster, δ Zustandsübergangsfunktion, λ Ausgangsfunktion, $S^0 \in S$ Anfangszustand

Zustandsminimierung

1) Eleminieren nicht erreichbarer Zustände

2) Zusammenfassen äquivalenter Zustände

1-äquivalente Zustände

	00	01	10	11	
S1	S4/0	S4/1	S2/1	S1/1	A
S2	S4/0	S5/1	S1/1	S2/1	
S3	S1/1	S4/0	S2/0	S3/1	B
S4	S1/1	S2/0	S3/0	S5/1	
S5	S1/1	S2/0	S3/0	S4/1	

2-äquivalente Zustände

	00	01	10	11	
S1	S4/0 B	S4/1 B	S2/1 A	S1/1 A	A
S2	S4/0 B	S5/1 B	S1/1 A	S2/1 A	
S3	S1/1 A	S4/0 B	S2/0 A	S3/1 B	B
S4	S1/1 A	S2/0 A	S3/0 B	S5/1 B	
S5	S1/1 A	S2/0 A	S3/0 B	S4/1 B	C

Da 2- und 3-äqivalente Zustände identisch, sind diese Zustände (absolut) äquivalent.

3-äquivalente Zustände

	00	01	10	11	
S1	S4/0 C	S4/1 C	S2/1 A	S1/1 A	A
S2	S4/0 C	S5/1 C	S1/1 A	S2/1 A	
S3	S1/1 A	S4/0 C	S2/0 A	S3/1 B	B
S4	S1/1 A	S2/0 A	S3/0 B	S5/1 C	
S5	S1/1 A	S2/0 A	S3/0 B	S4/1 C	C

Realisierung

	x = 0	x = 1
S_1	(S_1, 0)	(S_2, 1)
S_2	(S_1, 0)	(S_3, 0)
S_3	(S_3, 0)	(S_1, 1)

Zustandscodierung:
$S_1 \triangleq 00 \triangleq \overline{s}_1\overline{s}_2$
$S_2 \triangleq 01 \triangleq \overline{s}_1 s_2$
$S_3 \triangleq 10 \triangleq s_1\overline{s}_2$

x	s_1s_2	z_1z_2	y
0	00	00	0
1	00	01	1
0	01	00	0
1	01	10	0
0	10	10	0
1	10	00	1

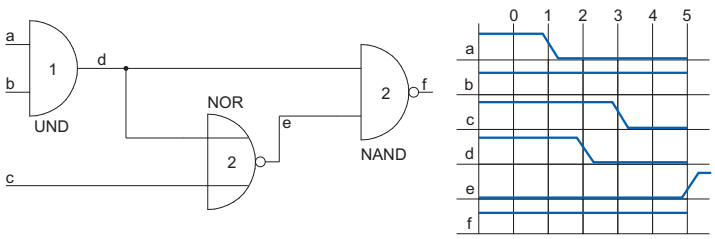
$y = x \cdot \overline{s}_1 \cdot \overline{s}_2 + x \cdot s_1 \cdot \overline{s}_2 = x \cdot \overline{s}_2$
$z_1 = x \cdot \overline{s}_1 \cdot s_2 + \overline{x} \cdot s_1 \cdot \overline{s}_2$
$z_2 = x \cdot \overline{s}_1 \cdot \overline{s}_2$

Logiksimulation

Ereignis: Wertänderung eines Signals
Ereignis: $E = (z, \text{val}(z, t_{exe}), t_{gen}, t_{exe})$
Laufzeitabhängige Effekte:

1) Race: "Wettlauf" von Signalwertänderungen auf Leitungen vor einem gemeinsamen Knoten (Gatter)

2) Hazard: (Spike, Glitch) Signalwertverlauf, der der reinen Logikfunktion der Schaltung widerspricht und nur durch die Verzögerungszeiten entsteht.



t	a	b	c	d	e	f	ausgewertete Elemente	neue Events*
0	1	1	1	1	0	1	init	(a,'0',0,1), (c,'0',0,3)
1	0						AND	(d,'0',1,2)
2				0			NAND, NOR	
3			0				NOR	(e,'1',3,5)
5					1		NAND	

*(signal, value, t_{gen},t_{exe})

VHDL

ENTITY Bausteinname IS
Definiert die Schnittstelle einer Logik

PORT (Schnittstelle)
Definiert Ein- und Ausgänge

ARCHITECTURE ... OF ... IS
Beschreibt den internen Aufbau

PROCESS (Signalliste)
Alle Prozesse laufen nebeneinander ab

COMPONENT Gatternname
Beschreibt eine interne Komponente

Testverfahren

Begriffe

Fehlerhafte Schaltung: y_μ
Fehlergruppe: Menge aller erkannten Fehler von Test t_v
Fehleranzahl = 2· Signalanzahl
Testgruppe: Menge aller Tests die Fehler f_μ erkennen
Testmenge: Enthaltene Tests erkennen alle angenommenen Fehler
Fehlerüberdeckung: Vereinigung der Fehlergruppen enthält alle angenommenen Fehler

Fehlererkennung: $t_v Rf_\mu = y(\underline{x}_v) \oplus y_\mu(\underline{x}_v) = 1$

Fehlerunterscheidung: $y_\mu(\underline{x}_v) \oplus y_\kappa(\underline{x}_v) = 1$

Zwei einzelne Fehler sind nicht unterscheidbar, wenn deren Testgruppen gleich sind

Boolesche Differenz

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

Rechenregeln

1) $y_x = 0$ falls $y \neq f(x)$
2) $y_y = 1$
3) $(\overline{y})_x = y_x$
4) $(z \oplus w)_x = z_x \oplus w_x$
5) $(z \cdot w)_x = z \cdot w_x \oplus z_x \cdot w \oplus z_x \cdot w_x$
6) $(z + w)_x = \overline{z} \cdot w_x \oplus z_x \cdot \overline{w} \oplus z_x \cdot w_x$
7) Falls $y = y(z(x)) : y_x = y_z \cdot z_x$
8) $(y_z)_w = (y_w)_z$

Test

$z/0 = z \cdot y_z = 1$
$z/1 = \overline{z} \cdot y_z = 1$

Fehlerbelegung (Einstellbarkeit)

$z/0 : z(\underline{x}) = 1$
$z/1 : z(\underline{x}) = 0$

Sensibilisierungsbelegung (Beobachtbarkeit)

$y_z(\underline{x}) = 1$

Strukturbezogene Berechnung der BD

Redeweisen für y_x:

- BD von y nach x (Boolean Difference)
- Beobachtbarkeit von x an y (Observability)
- Empfindlichkeit von y gegen x (Sensitivity)

Redeweisen für $y_x = 1$:

- y ist von x funktionell abhängig
- Fehlbelegung an x ⇒ Fehlbelegung an y
- Einfachfehler an x an y beobachtbar

$y_x = y(x) \oplus y(\overline{x})$
$y \oplus y_x = y(\overline{x}) = z(\overline{x}) \circ w(\overline{x})$
$\boxed{y \oplus y_x = (z \oplus z_x) \circ (w \oplus w_x)}$
$y_x = [(z \oplus z_x) \circ (w \oplus w_x)] \oplus [z \circ w]$
$\boxed{y_x = y_z \cdot z_x \cdot \overline{w_x} + y_w \cdot w_x \cdot \overline{z_x} + z_x \cdot w_x \cdot [\overline{z} \circ \overline{w} \oplus z \circ w]}$
$y_x = [(z \oplus z_x) \circ (w \oplus w_x) \circ (v \oplus v_x)] \oplus [z \circ w \circ v]$

Pfadsensibilisierung

1) $y_z \cdot z_x \cdot \overline{w_x} = 1$: Einfachfehlerpfad x-z-y
2) $y_w \cdot w_x \cdot \overline{z_x} = 1$: Einfachfehlerpfad x-w-y
3) $z_x \cdot w_x \cdot [\overline{z} \circ \overline{w} \oplus z \circ w] = 1$: Mehrfachpfadsensibilisierung
4) $z_x \cdot w_x \cdot \overline{[\overline{z} \circ \overline{w} \oplus z \circ w]} = 1$: Selbstmaskierung

Fehlerbaumkonstruktion

1) Gut-Simulation der Eingangsbelegung
2) Fehler x/v mit $v = \overline{\text{val}(x)}$ einstellbar.
3) Welche Eingänge wirken sich auf das jeweilige Gatter aus? ⇒ Mit Punkt und dicker Linie markieren
4) Fan-outs auftrennen. Die relevanten Fan-outs sind diejenigen, die sich bei verändertem Signalwert auf den Ausgang auswirken.
5) Menge S^0: Alle Signale, die im Fehlerbaum eine „Verbindung zum Ausgang" haben.
6) Menge F^t: Alle Fehler, die sich auf den Ausgang auswirken.

D-Algorithmus

D: '1' im fehlerfreien Fall, '0' im fehlerhaften Fall $\overline{D}$: '0' im fehlerfreien Fall, '1' im fehlerhaften Fall D-Kette V_D: Einfach-/Mehrfachfehlerpfad von x zum Ausgang y

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

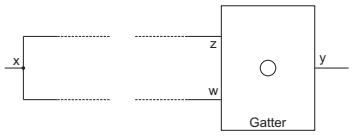
$+$	0	1	X	D	$\overline{D}$
0	0	1	X	D	$\overline{D}$
1	1	1	1	1	1
X	X	1	X	X	X
D	D	1	X	D	1
$\overline{D}$	$\overline{D}$	1	X	1	$\overline{D}$

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

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

F: Fehlerbelegung; S: Sensibilisierung I: Implikation; O: Optionale Wertzuweisung

Globale Implikation



Lernprozedur (Kontrapositionsgesetz)

$\widehat{x} \Rightarrow \widehat{y}$ genau dann, wenn $\overline{\widehat{y}} \Rightarrow \overline{\widehat{x}}$
Es sollen nur globale Implikationen gelernt werden.

Nur lernenswert falls bei **Vorwärtsimplikation** gilt: $y_z \cdot y_w = 1$

Einstellbarkeitsmaße(Controllability)

C_0: Nulleinstellbarkeit; C_1: Einseinstellbarkeit x Eingangsvariable ⇒ $C_0(x) = C_1(x) = 0.5$
Bei mehreren Alternativen zur Sicherstellung eines Signalwertes wird derjenige mit dem größten Einstellbarkeitsmaß ausgewählt (Nur bei Baumstrukturen exakt!).