

5.5 Wichtige Begriffe

Wichtige Begriffe:	Definition	Bemerkung
Signalvariable	x	$\hat{x} \in \{0, 1\}$
Literal	$l_i = x_i$ oder $\overline{x_i}$	$i \in I_0 = \{1, \dots, n\}$
Minterme,0-Kuben	$M0C \ni m_j = \prod_{i \in I_0} l_i$	$ 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 \wedge \bar{l} \in c_j\} $	$\delta_{ij} = \delta(c_i, c_j)$
Implikanten	$MI = \{c \in MC \mid c \subseteq f\}$	
Primimplikanten	$MPI = \{p \in MI \mid p \not\subseteq c \ \forall c \in MI\}$	$MPI \subseteq MI \subseteq MC$
DNF (DNF)	eine Summe von Produkttermen	Terme sind ODER-verknüpft
KNF (KNF)	ein Produkt von Summentermen	Terme sind UND-verknüpft
KDNF (KDNF)	Summe aller Minterme	WT: 1-Zeilen sind Minterme
KKNF (KKNF)	Menge aller Maxterme	WT: 0-Zeilen negiert sind Maxterme
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

6 Beschreibungsformen

6.1 Disjunktive Normalform/Sum of products (DNF/DNF)

Eins-Zeilen als Implikanten (UND) schreiben und alle Implikanten mit ODER verknüpfen:
 $Z = \overline{A} \cdot \overline{B} + \overline{C} \cdot D$

6.2 Konjunktive Normalform/Product of sums (KNF/KNF)

Null-Zeilen negiert als Implikat (ODER) schreiben und alle Implikaten UND verknüpfen:
 $Z = (\overline{A} + \overline{C}) \cdot (\overline{A} + D) \cdot (\overline{B} + \overline{C}) \cdot (\overline{B} + D)$

6.3 Umwandlung in jeweils andere Form

- Doppeltes Negieren der Funktion: $Z = \overline{\overline{\overline{A} \cdot \overline{B} + \overline{C} \cdot D}}$
- Umformung "untere" Negation (DeMorgan): $Z = \overline{\overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D} = \overline{(A + B) \cdot (C + \overline{D})}$
- 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):
 $Z = \overline{A \cdot C \cdot A \cdot \overline{D} \cdot B \cdot C \cdot B \cdot \overline{D}} = \overline{(\overline{A} + \overline{C}) \cdot (\overline{A} + D) \cdot (\overline{B} + \overline{C}) \cdot (\overline{B} + D)}$

Analog von KNF (KNF) nach DNF (DNF).

6.4 Shannon Entwicklung

$f = x_i \cdot f_{x_i} + \overline{x_i} \cdot f_{\overline{x_i}} = (x_i + f_{\overline{x_i}}) \cdot (\overline{x_i} + f_{x_i}) = (f_{x_i} \oplus f_{\overline{x_i}}) \cdot x_i \oplus f_{\overline{x_i}}$
 $\overline{f} = x_i \cdot \overline{f_{x_i}} + \overline{x_i} \cdot \overline{f_{\overline{x_i}}}$

7 Logikminimierung

7.1 Nomenklatur

- m_i Minterm: UND-Term in dem alle Variablen vorkommen (aus KDNF)
- M_i Maxterm: ODER-Term in dem alle Variablen vorkommen (aus KKNF)
- c_i Implikant: UND-Term in dem freie Variablen vorkommen können
- C_i Implikat: ODER-Term in dem freie Variablen vorkommen können
- p_i Primimplikant: UND-Term mit maximal freien Variablen
- P_i Primimplikat: ODER-Term mit maximal freien Variablen

7.2 Karnaugh-Diagramm

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

$\overline{z}^x y$	00	01	11	10
0	1	0	0	
1	X	1	1	0

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

Don't Care Werte ausnutzen!
Achtung: Auf eventuelle Unterdefiniertheit überprüfen (Redundante Zeilen) (Kreiert Don't Cares)
Immer vollständig Nullen und Eines ausfüllen

7.3 Quine Methode

- geg.: DNF/DNF oder Wertetabelle von $f(x)$
ges.: alle Primimplikanten p_i (VollSOP)
- Spezielles Resoltuionsgesetz: $x \cdot a + \overline{x} \cdot a = a$
Absorptionsgesetz: $a + a \cdot b = a$
- KDNF/KDNF bestimmen (z.B. $f(x, y, z) = xy = xyz + xy\overline{z}$)
 - Alle Minterme in Tabelle eintragen (Index von m ist (binär)Wert des Minterms)
 - 1-Kubus: Minterme die sich um eine Negation unterscheiden, zu einem Term verschmolzen (Resolutionsgesetz)
 - Der 1-Kubus muss zusammenhängend sein! (d.h. alle 1-Kubus Minterme müssen zusammenhängen)
 - Wenn möglich 2-Kubus bilden.
 - Wenn keine Kubenbildung mehr möglich → Primimplikanten

Beispiel (Quine Methode):

	0-Kubus	A	1-Kubus	R	A	2-Kubus	A
m_1	$\overline{x}_1 \overline{x}_2 x_3$	✓	$\overline{x}_2 x_3$	$m_1 \& m_5$	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	$x_1 \overline{x}_2 x_3$	✓	$x_1 \overline{x}_3$	$m_4 \& m_6$	✓		
m_6	$x_1 x_2 \overline{x}_3$	✓	$x_1 x_3$	$m_5 \& m_7$	✓		
m_7	$x_1 x_2 x_3$	✓	$x_1 x_2$	$m_6 \& m_7$	✓		

⇒ $f(x_1, x_2, x_3) = p_1 + p_2 = \overline{x}_2 x_3 + x_1$

7.4 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

- 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")
- überprüfe ob Resolventen aus der 1. Schicht Kuben aus Schicht 0 überdecken(Absorbtion) und streiche diese Kuben aus Schicht 0
- Schicht i besteht aus den möglichen Resolventen von Schicht 0 bis $(i - 1)$. Abgestrichene Kuben aus vorherigen Schichten brauchen **nicht** mehr beachtet werden.
- Sobald in der i-ten Schicht +1 steht oder keine weiteren Resolventen gebildet werden können, ist man fertig. ⇒ alle nicht ausgestrichenen Terme bilden die VollSOP

$f(x_1, \dots, x_n)$	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

7.5 Überlagerung Bestimmung der MinSOP

Geg: KDNF/KDNF ($\sum m_i$) und VollSOP ($\sum p_i$) Ges: MinSOP (Minimalform)

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. Bsp:

	Minterme				
Primterme	m_1	m_2	\dots	m_N	$L(p_i)$
p_1	✓				$L(p_1)$
p_2	✓			✓	$L(p_2)$
\vdots					\vdots
p_K		✓			$L(p_K)$

Algorithmus:

- Suche Spalten mit nur einem Minterm.
- Streiche andere Spalten des zugehörigen Primterms.
- Streiche Primterme, dessen Minterme alle gestrichen sind.

K : Anzahl der Primterme
 N : Anzahl der Minterme
 $L(p_i)$: Kosten/Länge der Primimplikanten

8 Halbleiter

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

9 MOS-FET's

Metal Oxide Semiconductor Field Effekt Transistor

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

9.1 Bauteilparameter

Verstärkung: $\beta = K' \frac{W}{L}$ mit $K' = \frac{\mu \epsilon_{ox} \epsilon_0}{t_{ox}}$ $[\beta] = \frac{A}{V \cdot s}$

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 Gateoxys	$\epsilon_{ox} \approx 3,9$
Dielektrizitätskonstante	$\epsilon_0 = 8.8541878 \cdot 10^{-12} \frac{As}{Vm}$
Gateoxydicke	t_{ox}
Verstärkung	$\beta = \frac{\mu_n \epsilon_{ox} \epsilon_0}{t_{ox}} \cdot \frac{W}{L} = K' \frac{W}{L} = \frac{\mu_n C_G}{L^2}$
Kapazität	$C_G = \epsilon_{ox} \epsilon_0 \frac{WL}{t_{ox}}$
Verzögerungszeit	$t_{pHL} \propto \frac{C_L t_{ox} L p}{W_p \mu_p \epsilon_{ox} (V_{DD} - V_{th})}$
Verzögerungszeit	t_{pHL} Propagation delay von 90% auf 10%
Verzögerungszeit	t_{pLH} Propagation delay von 10% auf 90%

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

9.2 Drainstrom

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

Id = { 0, für Ugs - Uth ≤ 0 (Sperrber.)
β[(ug_s - Uth) · uds - 1/2 uds^2], für 0 ≤ Ugs - Uth ≤ uds (linearer Ber.)
1/2 β · (ug_s - Uth)^2, für 0 ≤ Ugs - Uth ≤ uds (Sättigungsber.) }

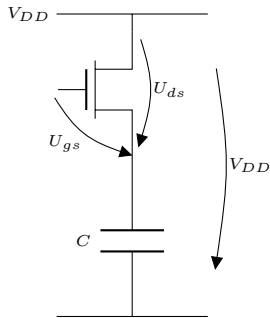
pMos (n-dotiertes Substrat, p-dotierte Drain/Source), schlechter pull down (Pegeldegenerierung)

Id = { 0, für Ugs - Uth ≥ 0 (Sperrber.)
-β[(ug_s - Uth) · uds - 1/2 uds^2], für 0 ≥ Ugs - Uth ≤ uds (linearer Ber.)
-1/2 β · (ug_s - Uth)^2, für 0 ≥ Ugs - Uth ≤ uds (Sättigungsber.) }

9.3 pMos und nMos

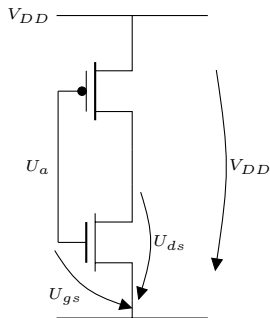
Table with 5 columns: VGS, Transistor, Source liegt immer am, VGS, VDS, ID, Substrat. It compares pMos and nMos characteristics.

9.4 Kondensatoraufgaben



- 9.4.1 Laden: Kondensator C lädt, solange ID > 0 -> C lädt, solange ugs - Uth ≥ uds ≥ 0
- 9.4.2 Entladen: Source und Drain werden vertauscht. Auf Gatespannung achten.

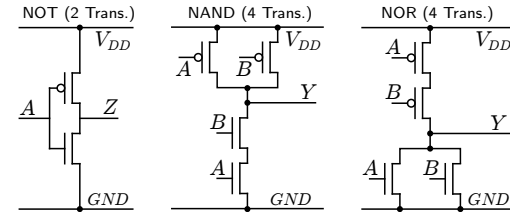
9.5 Gatterschwellspannungsaufgaben



Gatterschwellspannung ist der Punkt, wo sich beide Transistoren in Sättigung befinden. Dann Ströme mittels Knotengleichung ausrechnen. Isat,n = -Isat,p Vorsicht: UGS,p = VDD - Ua

10 CMOS - Logik

Vorteil: (Fast) nur bei Schaltvorgängen Verlustleistung - wenig statische Verluste
Drei Grundgatter der CMOS-Technologie:



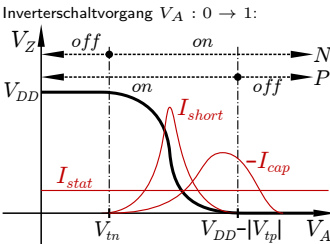
Falls GND und VDD vertauscht würden, dann NAND -> AND und NOR -> OR
Allerdings schlechte Pegelgenerierung.

10.1 Gatterdesign

Table with 3 columns: Netzwerk, Pull-Down Transistoren, Pull-Up Transistoren. It lists AND, OR, and their implementations using nMos and pMos.

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

10.2 CMOS Verlustleistung



Achtung: Logikpegel sind über die Steigung der |VTC| ≤ 1 des Inverters definiert.
Zusammensetzung Ishort:

Table with 6 columns: Transistor, (0, Vtn), (Vtn, VDD/2), Um VDD/2, (VDD/2, VDD - |Vtp|), (VDD - Vtp, VDD). It categorizes n-MOS and p-MOS into different operating regions.

Dynamische Verlustleistung: Pdyn = Pcap + Pshort
Kapazitive Verluste: Pcap = α01 fCL VDD^2
Kurzschlussstrom: Pshort = α01 fβn τ(VDD - 2Vtn)^3
Schalthäufigkeit: α0-1 = Schaltvorgänge(pos. Flanke) / # Betrachtete Takte (max 0.5)
Schalthäufigkeit (periodisch): α = fswitch / fclk
Abhängig von den Signalfanken, mit Schaltfunktionen verknüpft
≈ VDD1 / ∝ Schaltzeit: VDD2 = tD1 / tD2 (bei Schaltnetzen tlog)
Verzögerungszeit tpd ∝ CL * tox * Lp / (Wp * μp * ε * (VDD - Vth))

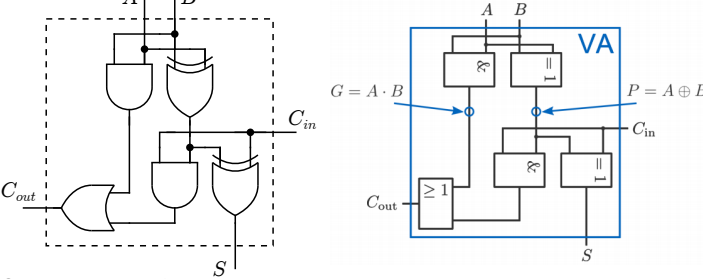
tpd ist Zeit zwischen crossover 50% von Eingang zu crossover 50% am Ausgang.

Steigend mit: Kapazitiver Last, Oxiddicke, Kanallänge, Schwellspannung

Sinkend mit: Kanalweite, Ladungsträger Beweglichkeit, Oxyd Dielektrizität, Versorgungsspannung

Statische Verlustleistung Pstat: Sub-Schwellströme, Leckströme, Gate-Ströme Abhängigkeit:
VDD ↑: Pstat ↑ Vth ↑: Pstat ↓ (aber nicht proportional)

11 Volladdierer (VA)/Ripple-C(u)arry-Adder



Generate gn = an · bn
Propagate pn = an ⊕ bn
Summenbit Sn = cn ⊕ pn = an ⊕ bn ⊕ cn
Sn = anbn cn + an bn cn + an bn cn + an bn cn (Ungerade Anzahl von Eingängen 1)
Carry-out cn+1 = cn ⊕ pn + gn
cn+1 = anbn cn + an bn cn + an bn cn + an bn cn (Mindesten zwei Eingänge 1)

Laufzeiten: tsn = { tcn + txor tcn > txor, 2txor sonst
tcn+1 = { tand + tor an = bn = 1 (gn = 1), txor + tand + tor an = bn = 0 (pn = 0, gn = 0), tcn + tand + tor an ≠ bn (pn = 1)

12 Sequentielle Logik

Logik mit Gedächtnis (Speicher).

12.1 Begriffe/Bedingungen

Table with 2 columns: Parameter (tSetup, thold, tc2q, Min. Taktperiode, Max. Taktfrequenz, Holdzeitbedingung, Durchsatz, Latenz) and Description/Formula.

12.2 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

12.3 Parallel Processing

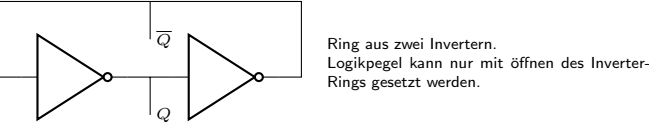
- Durchsatz = #Modul / tclk, Modul = f Latenz = tclk
- Paralleles, gleichzeitiges Verwenden mehrere identischer Schaltnetze
 - Zusätzliche Kontrolllogik nötig (Multiplexer)
 - Taktfrequenz und Latenz bleiben konstant
 - Durchsatz steigt mit der Zahl der Verarbeitungseinheiten ABER: deutlich höherer Ressourcenverbrauch

13 Speicherelemente

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 $clk_{0 \rightarrow 1}$ 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^{Adressbreite}

13.1 Speicherzelle/Register



13.2 Latch

Set-Reset Latch:
Zwei gegenseitig rückgekoppelte NAND-Gatter.

Active Low Logik:
 $\overline{S} = 0 \Rightarrow Q = 1, \overline{R} = 0 \Rightarrow Q = 0$

\overline{R}	\overline{S}	Q
1	1	Q
0	1	0
1	0	1
0	0	$Q = \overline{Q}$

Enable-Latch: ändert Speicherzustand auf D nur wenn e = 1.
Level-Controlled ⇔ Latch.

e	Q
0	Q
1	D

13.3 Flip-Flop

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

14 Automaten

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

I	Eingabealphabet
O	Ausgabealphabet
S	Menge von Zuständen
$R \subseteq S$	Menge der Anfangszustände
$f : S \times I \rightarrow S$	Übergangsrelation
g	Ausgaberation

Moore Automat

Mealy Automat

Zustandsnummerierung immer einfügen.

Moore	Mealy
Output hängt nur vom Zustand ab	Output hängt von Zustand und Eingabe ab
Kein direkter kombinatorischer Pfad Eingang⇒Ausgang	Generell weniger Zustände als Moore.
$s' = f(s, i), o = g(s)$	$s' = f(s, i), o = g(s, i)$
$g : S \rightarrow O$	$g : S \times I \rightarrow O$

14.1 Wahrheitstabelle einer FSM

i	$S = S_0 \dots S_n$	o	$S' = S'_1 \dots S'_n$
0	0...0	o0,0...0	$S'_{0,0...0}$
⋮	⋮	⋮	⋮
1	1...1	o1,1...1	$S'_{1,1...1}$

Moore: o ist $f(S)$, nächster Zustand $S' = f(i, S)$
Mealy: o ist $f(i, S)$, nächster Zustand $S' = f(i, S)$