

1 Moore'sches Gesetz

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

2 Einheiten

Potenz	Vorsatz	Potenz	Vorsatz	$H z$	s^{-1}
10^{12}	T	10^{-1}	d	N	$kgms^{-2}$
10^9	G	10^{-2}	c	J	$Nm = VAs$
10^6	M	10^{-3}	m	W	$VA = Js^{-1}$
10^3	k	10^{-6}	μ	C	As
10^2	h	10^{-9}	n	V	JC^{-1}
10^1	da	10^{-12}	p	F	CV^{-1}
		10^{-15}	f	Ω	VA^{-1}
				H	VsA^{-1}

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

3 Polyadische Zahlensysteme

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

Z :Zahl, r :Basis, d_i :Ziffer, p :#Ziffern vorne n :#Nachkommastellen

Binäres Zahlensystem:

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

Octalsystem:	Hexadezimalsystem:
$d_{i8} \in \{0, 1, 2, 3, 4, 5, 6, 7\}$	$d_{i16} \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$

Benötigte Bits: $N : n$ Bit. $M : m$ Bit
 $N + M : \max\{n, m\} + 1$ Bit
 $N \cdot M : n + m$ Bit

3.1 Umrechnung

	$Z \geq 1$	$Z < 1$
$r \rightarrow 10$	$Z_{10} = \sum r^i \cdot d_i$ $101_2 \rightarrow 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1$	$Z_{10} = \sum r^{-i} \cdot d_{-i}$ $0.11_2 \rightarrow 1 \cdot 0.5 + 1 \cdot 0.25$
$10 \rightarrow r$	$d_i = Z_{10} \% r^i$ ($d_i = Z_{10} \bmod r^i$) $58/8 = 7$ Rest 2 (LSB) $7/8 = 0$ Rest 7 (MSB) (Ende wenn 0 erreicht) Auf Ende achten $1r3\%5 \rightarrow 0r1$	$0.4 \cdot 2 = 0.8$ Übertrag 0 (MSB) $0.8 \cdot 2 = 1.6$ Übertrag 1 (Wiederholen bis 1 oder Periodizität)

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

$Z \rightarrow -Z$ (Umkehrung gleich)	Bsp: 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

Bitverteilung(single/double):

$s(1)$	$e(8/11)$	$f(23/52)$
--------	-----------	------------

s : Vorzeichen, e : Exponent, f : Mantisse (**Nachkommastellen!** $2^{-1}2^{-2} \dots$)

Spezialwerte: $Z = 0 \Leftrightarrow e = 0$ $Z = +(-)\infty \Leftrightarrow e = 255, s = 0(1)$

IEEE \rightarrow Wert Z $Z = (-1)^s \cdot (1 + 0.f) \cdot 2^{e-127}$	Bsp: $s = 1, e = 126, f = 01_2$ $Z = -1 \cdot 2^{-1} \cdot 1.01_2 = -0.101_2 = -0.625$
Wert $Z \rightarrow$ IEEE (Binärdarstellung) $s = 0$ (positiv), $s = 1$ (negativ) $Z \rightarrow Z_2$ (beim Komma teilen) Z_2 n-mal shiften $\rightarrow 1.xxx \dots$ Exponent $e = n + 127 \rightarrow e_2$ Mantisse $f_2 = xxx \dots$	Bsp: $Z = 11.25$ $s = 0$ $Z = 1011.01_2$ $Z = 1.01101_2 \cdot 2^3$ $e = 3 + 127 = 130 = 10000010_2$ $f = 01101000 \dots_2$
Wert $Z \rightarrow$ IEEE (Formel) $s = 0$ (positiv), $s = 1$ (negativ) $E = \lfloor \log_2 Z \rfloor$ $e = E + 127 \rightarrow e_2$ $f = \left(\frac{\lfloor Z \rfloor}{2^E} - 1 \right) \cdot 2^{23} \rightarrow f_2$	Bsp: $Z = 11.25$ $s = 0$ $E = \lfloor \log_2 11.25 \rfloor = \lfloor 3,49 \dots \rfloor = 3$ $e = 3 + 127 = 130 = 10000010_2$ $f = \left(\frac{\lfloor 11.25 \rfloor}{2^3} - 1 \right) \cdot 2^{23} = 3407872 = 01101000 \dots_2$

3.4 Zahlenoperationen

- Festkomma (Vorzeichenlos)
 - Erweiterung: Null vorne anhängen
 - Addition: Bitweise
 - Subtraktion: Bitweise
 - Multiplikation: Add-Shift (Add für jede 1 im Multiplikant) (Resultat eins rechts Shiften)
Sonderfall: Multiplikation mit 2-er Potenz \rightarrow um Potenz mal shiften.
 - Division:
- Festkomma (Einser Komplement)
 - Erweiterung: Null an Stelle 2 einfügen.
 - Addition:
 1. Prüfe Beide Vorzeichen
 2. Gleiches Vorzeichen \rightarrow reguläre Addition
 3. Verschieden \rightarrow Subtraktion kleiner Operator von großem Operator. Übernahme Vorzeichen des großen Operators.

- Festkomma (Zweier Komplement)
 - Erweiterung: 1 vorne anhängen
 - Addition: Regulär (Gleiche Parameterlänge) (Overflow ignorieren)
 - Subtraktion: Addition mit komplementiertem Subtraktor (Gleiche Parameterlänge)(Overflow ignorieren)
 - Multiplikation:
 1. Zahlen auf Produktlänge erweitern.
 2. Zahlen mittels Add-Shift multiplizieren (Überflüssige Bits nach links rausschieben und ignorieren)

- Gleitkomma (IEEE Float)
 - Addition: Exponenten auf größeren angleichen, Mantissen addieren. Vorzeichen inspizieren.
 - Subtraktion:
 - Multiplikation: Exponenten auf größeren angleichen, Mantissen multiplizieren. Vorzeichen multiplizieren.
Sonderfall: Multiplikation mit 2-er Potenz \rightarrow Potenz zu Exponent addieren.
Achtung: bei addieren der Exponenten zweier Gleitkommazahlen muss von einem Exponenten der Bias abgezogen werden.
 - Division:

4 Zeichenkodierung

4.1 ASCII

American Standard Code for Information Exchange
Fixe Codewortlänge (7 Bit, 128 Zeichen)
 $0x00 - 0x7F$

4.2 UTF-8

Universal Character Set Transformation Format
Variable Codewortlänge (1-4 Byte) \rightarrow Effizient

Schema











- $MSB = 0 \rightarrow 8$ Bit (restliche Bit nach ASCII)
- $MSB = 1 \rightarrow 16, 24$ oder 32 Bit
 - Byte 1: Die ersten 3, 4, 5 Bit geben die Länge des Codewortes an (110, 1110, 11110)
 - Byte 2-4: Beginnen mit Bitfolge 10

4.3 Zahlensysteme

Base 10	Base 2	Base 8	Base 16
00	0000	0o00	0x0
01	0001	0o01	0x1
02	0010	0o02	0x2
03	0011	0o03	0x3
04	0100	0o04	0x4
05	0101	0o05	0x5
06	0110	0o06	0x6
07	0111	0o07	0x7
08	1000	0o10	0x8
09	1001	0o11	0x9
10	1010	0o12	0xA
11	1011	0o13	0xB
12	1100	0o14	0xC
13	1101	0o15	0xD
14	1110	0o16	0xE
15	1111	0o17	0xF

5 Boolsche Algebra

5.1 Boolsche Operatoren (Wahrheitstabelle WT)

		 A B out	 A B out	 A B out	 A B out	 A B out	 A B out
x	y	 A B Y	 A B Y	 A B Y	 A B Y	 A B Y	 A B 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: $f = c_1 + c_2 + c_3 \Rightarrow cov(f) = \{c_1, c_2, c_3\}$ $x \oplus y \equiv x\overline{y} + \overline{x}y$

5.2 Boolesche Funktionen

$$f : \{0, 1\}^n \rightarrow \{0, 1\} \quad f(\underline{x}) = f(x_1, x_2, \dots, x_n)$$

Einsmenge F von f : $F = \{\underline{x} \in \{0, 1\}^n | f(\underline{x}) = 1\}$
Nullmenge \overline{F} von f : $\overline{F} = \{\underline{x} \in \{0, 1\}^n | f(\underline{x}) = 0\}$

Kofaktor bezüglich

- $x_i : f_{x_i} = f|_{x_i=1} = f(x_1, \dots, 1, \dots, x_n)$
- $\overline{x}_i : f_{\overline{x}_i} = f|_{x_i=0} = f(x_1, \dots, 0, \dots, x_n)$

Eigenschaften von f(x)

- tautologisch ⇔ f(x) = 1 ∀x ∈ {0, 1}^n
- kontradiktorisch ⇔ f(x) = 0 ∀x ∈ {0, 1}^n
- unabhängig von xi ⇔ fxi = fxi
- abhängig von xi ⇔ fxi ≠ fxi

5.3 Multiplexer

f = x · a + x · b

(2 Eingänge a, b und 1 Steuereingang x)

f = x1x2a + x1x2b + x1x2c + x1x2d

(Eingänge: a, b, c, d Steuerung: x1, x2)

5.4 Wichtige Begriffe

Wichtige Begriffe:	Definition	Bemerkung
Signalvariable	x	x̂ ∈ {0, 1}
Literal	li = xi oder xi	i ∈ I0 = {1, ..., n}
Minterme,0-Kuben	MOC ∋ mj = ∏i∈I0 li	MOC = 2^n
d-Kuben	MC ∋ cj = ∏i∈Ij⊆I0 li	MC = 3^n
Distanz	δ(ci, cj) = {l l ∈ ci ∧ l̄ ∈ cj}	δij = δ(ci, cj)
Implikanten	MI = {c ∈ MC c ⊆ f} Terme, dessen Erfüllbarkeit identisch mit die der Formel sind	
Primimplikanten	MPI = {p ∈ MI p ⊄ c ∀c ∈ MI} Implikanten, die maximal freie Variablen besitzen	MPI ⊆ MI ⊆ MC
Kernprimimplikanten	Primimplikanten die für Überdeckung zwingend notwendig sind	Spalten mit 1 Eintrag in Überdeckungstabelle

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

5.5 Gesetze der boolschen Algebra

	Boolsche Algebra (0, 1; ·, +, x̄)	Mengenalgebra (P(G); ∩, ∪, Ā; ∅, G)
Kommutativ	x · y = y · x x + y = y + x	A ∩ B = B ∩ A A ∪ B = B ∪ A
Assoziativ	x · (y · z) = (x · y) · z x + (y + z) = (x + y) + z	(A ∩ B) ∩ C = A ∩ (B ∩ C) (A ∪ B) ∪ C = A ∪ (B ∪ C)
Distributiv	x · (y + z) = (x · y) + x · z x + (y · z) = (x + y) · (x + z)	A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C) A ∪ (B ∩ C) = (A ∪ B) ∩ (A ∪ C)
Idempotenz	x · x = x x + x = x	A ∩ A = A A ∪ A = A
Absorption	x · (x + y) = x x + (x · y) = x	A ∩ (A ∪ B) = A A ∪ (A ∩ B) = A
Neutral	x · 1 = x x + 0 = x	A ∩ G = A A ∪ ∅ = A
Dominant	x · 0 = 0 x + 1 = 1	A ∩ ∅ = ∅ A ∪ G = G
Komplement	x · x̄ = 0 x + x̄ = 1 x̄x̄ = x	A ∩ Ā = ∅ A ∪ Ā = G ĀĀ = A
De Morgan	x̄ · ȳ = x̄ + ȳ x̄ + ȳ = x̄ · ȳ	Ā ∩ B̄ = Ā ∪ B̄ Ā ∪ B̄ = Ā ∩ B̄

6 Beschreibungsformen

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

Eins-Zeilen als Implikanten (UND) schreiben und alle Implikanten mit ODER verknüpfen:
Z = A · B + C · D

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

Null-Zeilen negiert als Implikat (ODER) schreiben und alle Implikaten UND verknüpfen:
Z = (A + C) · (A + D) · (B + C) · (B + D)

6.3 Umwandlung in jeweils andere Form

1. Doppeltes Negieren der Funktion: Z = A · B + C · D
2. Umformung "untere" Negation (DeMorgan) : Z = A · B · C · D = (A + B) · (C + D)
3. Ausmultiplizieren: Z = (A + B) · (C + D) = A · C + A · D + B · C + B · D
4. Umformung "obere" Negation (DeMorgan) :
Z = A C · A D · B C · B D = (A + C) · (A + D) · (B + C) · (B + D)
Analog von KNF (KNF) nach DNF (DNF).

6.4 Shannon Entwicklung

f = xi · fxi + xi · fxi = (xi + fxi) · (xi + fxi) = (fxi ⊕ fxi) · xi ⊕ fxi
f̄ = xi · f̄xi + xi · f̄xi

7 Logikminimierung

7.1 Nomenklatur

- mi Minterm: UND-Term in dem alle Variablen vorkommen (aus KDNF)
- Mi Maxterm: ODER-Term in dem alle Variablen vorkommen (aus KKNF)
- ci Implikant: UND-Term in dem freie Variablen vorkommen können
- Ci Implikat: ODER-Term in dem freie Variablen vorkommen können
- pi Primimplikant: UND-Term mit maximal freien Variablen
- Pi Primimplikat: ODER-Term mit maximal freien Variablen

7.2 Karnaugh-Diagramm

Zyklische Gray-Codierung:

2-dim	00 01 11 10
3-dim	000 001 011 010 110 111 101 100

≥xy | 00 | 01 | 11 | 10

0	1	0	0	0
1	X	1	1	0

Gleiche Zellen zusammenfassen: z.B. xy + y · z

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

7.3 Quine Methode

geg.: DNF/DNF oder Wertetabelle von f(x)
ges.: alle Primimplikanten pi (VollSOP)
Spezielles Resolutionsgesetz: x · a + x̄ · a = a
Absorptionsgesetz: a + a · b = a

1. KDNF/KDNF bestimmen (z.B. f(x, y, z) = xy = xyz + xyx̄)

2. Alle Minterme in Tabelle eintragen (Index von m ist (binär)Wert des Minterms)

3. 1-Kubus: Minterme die sich um eine Negation unterscheiden, zu einem Term verschmolzen (Resolutionsgesetz)

4. Der 1-Kubus muss zusammenhängend sein! (d.h. alle 1-Kubus Minterme müssen zusammenhängen)

5. Wenn möglich 2-Kubus bilden.

6. Wenn keine Kubenbildung mehr möglich → Primimplikanten

Beispiel (Quine Methode): y = a c̄ + abc + a b c̄ = a b c̄ + a b c̄ + abc + a b c̄

Anzahl pos.	Minterme	A	Implikanten mit 1 freien Variable	A	Implikanten mit 2 freien Variablen	A
0	a b c̄	c1	c1 = b c̄			
1	a b c̄	c1, c2	c2 = a c̄			
2	a b c̄	c2, c3	c3 = ab			
3	abc	c3				

	0-Kubus	A	1-Kubus	R	A	2-Kubus	A
m1	x1x2x3	✓	x2x3	m1&m5	p1		
m4	x1x2x3	✓	x1x2	m4&m5	✓	x1	p2
m5	x1x2x3	✓	x1x3	m4&m6	✓		
m6	x1x2x3	✓	x1x3	m5&m7	✓		
m7	x1x2x3	✓	x1x2	m6&m7	✓		

⇒ f(x1, x2, x3) = p1 + p2 = x2x3 + x1

7.4 Resolventenmethode

Ziel: alle Primimplikanten
Wende folgende Gesetze an:
Absorptionsgesetz: a + ab = a
allgemeines Resolutionsgesetz: x · a + x̄ · b = x · a + x̄ · 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(Absorption) 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. ⇒ alle nicht ausgestrichenen Terme bilden die VollSOP

f(x1, . . . , xn)

x · w + x̄ · w + x · y · w · z̄ + x̄ · y · w · z̄ + ȳ · w · z̄	0
+ w + y · w · z̄	1
+ w · z̄	2
+ w	3

7.5 Überlagerung Bestimmung der MinSOP

(Bestimmung der Kernprimimplikanten) Geg: KDNF/KDNF (∑ mi) und VollSOP (∑ pi)
Ges: MinSOP (Minimalform)
Überdeckung: C = (m0 ≤ p1) · (m2 ≤ p1 + m2 ≤ p2) = 1
C = τ1 · (τ1 + τ2) = τ1 + τ1 τ2 = τ1
Alternativ: Mit Überdeckungstabelle bestimmen. Bsp:

	Minterme				
Primterme	m1	m2	...	mN	L(pi)
p1	✓				L(p1)
p2	✓			✓	L(p2)
.					.
pK		✓			L(pK)

Algorithmus:

1. Suche Spalten mit nur einem Minterm.

2. Streiche andere Spalten des zugehörigen Primterms.

3. Streiche Primterme, dessen Minterme alle gestrichen sind.

4. Dominierte Zeilen streichen.

Homepage: [www.latex4ei.de](#) - Fehler bitte **sofort** melden.

Emanuel Regnath ([Emanuel.Regnath@tum.de](#)), Martin Zellner ([martin.zellner@mytum.de](#)), Hendrik Böttcher ([hendrik.boettcher@tum.de](#))

(Gekürzt, überarbeitet - [lukas.kompatscher@tum.de](#)) Stand: 16.2.2017 2

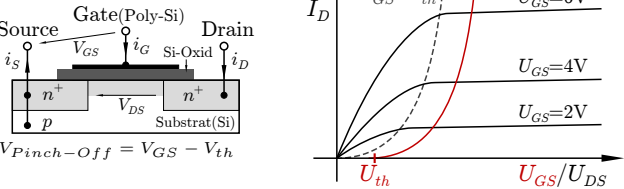
K : Anzahl der Primterme
 N : Anzahl der Minterme
 $L(p_i)$: Kosten/Länge der Primimplikanten
 $L(z)$: Länge des Terms z = Summe der Literale in Teiltermen + Anzahl der Teilterme
Primimplikanten von Tabelle ausrechnen: Minterme für jeden Primterm ablesen und reduzieren.
Länge Primimplikanten: anhand Anzahl von Kreuzen ablesen.

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



9.1 Bauteilparameter

Verstärkung: $\beta = K' \frac{W}{L}$ mit $K' = \frac{\mu \epsilon_{ox} \epsilon_0}{t_{ox}}$ $[\beta] = \frac{A}{V^2}$

Kanalweite	W
Kanallänge	L
Elektronenbeweglichkeit	$\mu_{\mu n} \approx 250 \cdot 10^{-4} \frac{m^2}{Vs}, \mu_p \approx 100 \cdot 10^{-4} \frac{m^2}{Vs}$
rel. Dielektrizität des Gateoxyds	$\epsilon_{ox} \approx 3,9$
Dielektrizitätskonstante	$\epsilon_0 = 8.8541878 \cdot 10^{-12} \frac{As}{Vm}$
Gateoxyddicke	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 (2 Signale)	Zeit zwischen $S_1 = 50\%$ und $S_2 = 50\%$ LH/HL bezieht sich auf Ausgang t_r : Zeit zwischen 10% und 90% t_f : Zeit zwischen 90% und 10%
Anstiegszeit (Selbes Signal)	
Abfallzeit (Selbes Signal)	

- 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

9.2 Drainstrom

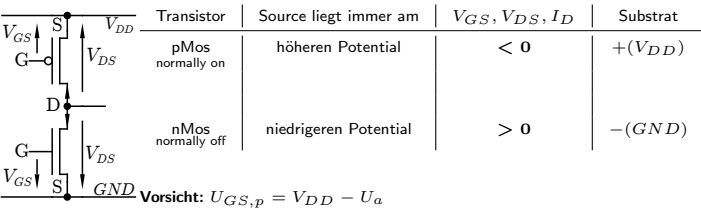
nMos (p-dotiertes Substrat, n-dotierte Drain/Source), schlechter pull up (Pegeldegenerierung)

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

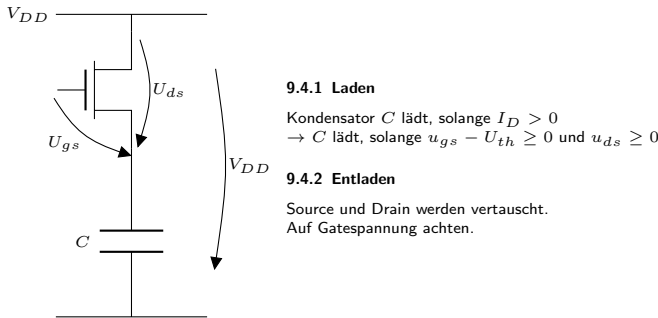
pMos (n-dotiertes Substrat, p-dotierte Drain/Source), schlechter pull down (Pegeldegenerierung)

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

9.3 pMos und nMos

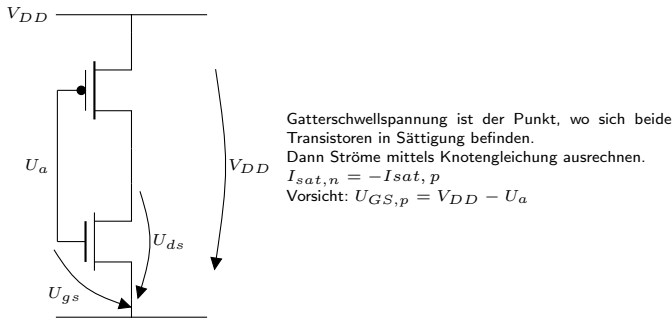


9.4 Kondensatoraufgaben



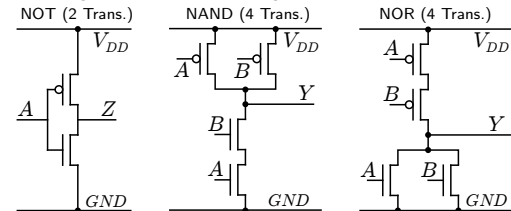
- 9.4.1 Laden
Kondensator C lädt, solange $I_D > 0$
 $\rightarrow C$ lädt, solange $u_{gs} - U_{th} \geq 0$ und $u_{ds} \geq 0$
- 9.4.2 Entladen
Source und Drain werden vertauscht.
Auf Gatespannung achten.

9.5 Gatterschwellspannungsaufgaben



10 CMOS - Logik

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



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

10.1 Gatterdesign

Netzwerk	Pull-Down nMos	Pull-Up pMos
Transistoren		
AND	Serienschaltung	Parallelschaltung
OR	Parallelschaltung	Serienschaltung

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

10.2 Umwandlung in Nand und Nor

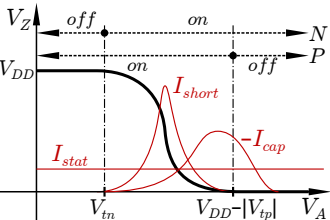
Gatter	Funktion	NAND Form	NOR Form
NOT	\bar{A}	$\overline{A \cdot A}$	$\overline{A + A}$
AND	$A \cdot B$	$\overline{\overline{A \cdot B \cdot A \cdot B}}$	$\overline{\overline{A + A + B + B}}$
OR	$A + B$	$\overline{\overline{A \cdot A \cdot B \cdot B}}$	$\overline{\overline{A + B + A + B}}$
NAND	$\overline{A \cdot B}$	$\overline{A \cdot B}$	$\overline{\overline{A + A + B + B + A + A + B + B}}$
NOR	$\overline{A + B}$	$\overline{\overline{A \cdot A \cdot B \cdot B \cdot A \cdot A \cdot B \cdot B}}$	$\overline{A + B}$

10.3 Anzahl Gatter aus Netzwerk berechnen

Jede Unterbrechung in der Mittellinie (Mittellinie \rightarrow Eingang CMOS Transistor) ist die Grenze zwischen zwei Gattern.

10.4 CMOS Verlustleistung

Inverterschaltvorgang $V_A : 0 \rightarrow 1$:



Achtung: Logikpegel sind über die Steigung der $|VTC| \leq 1$ des Inverters definiert.
Zusammensetzung I_{short} :

Transistor	$(0, V_{tn})$	$(V_{tn}, V_{DD}/2)$	Um $V_{DD}/2$	$(V_{DD}/2, V_{DD} - V_{tp})$	$(V_{DD} - V_{tp} , V_{DD})$
n-MOS	Sperrt	Sättigung	Sättigung	Linear	Linear
p-MOS	Linear	Linear	Sättigung	Sättigung	Sperrt

Dynamische Verlustleistung $P_{dyn} = P_{cap} + P_{short} \Rightarrow P_{dyn} \propto V_{DD}^2$
Kapazitive Verluste $P_{cap} = \alpha_{01} f C_L V_{DD}^2$
Kurzschlussstrom $P_{short} = \alpha_{01} f \beta_n \tau (V_{DD} - 2V_{tn})^3$

Schaltheufigkeit $\alpha_{0 \rightarrow 1} = \frac{\text{Schaltvorgänge (pos. Flanke)}}{\# \text{ Betrachtete Takte}} \quad (\text{max } 0.5)$
Schaltheufigkeit (periodisch) $\alpha = \frac{f_{switch}}{f_{clk}}$

Abhängig von den Signalfanken, mit Schaltfunktionen verknüpft
 $\approx V_{DD} \cdot 1 / \alpha$ Schaltzeit: $\frac{V_{DD} \cdot 2}{t_{D1}} = \frac{t_{D1}}{t_{D2}}$ (bei Schaltnetzen t_{log})

Verzögerungszeit $t_{pd} \propto \frac{C_L t_{ox} L_p}{W_p \mu_p \epsilon (V_{DD} - V_{th})}$

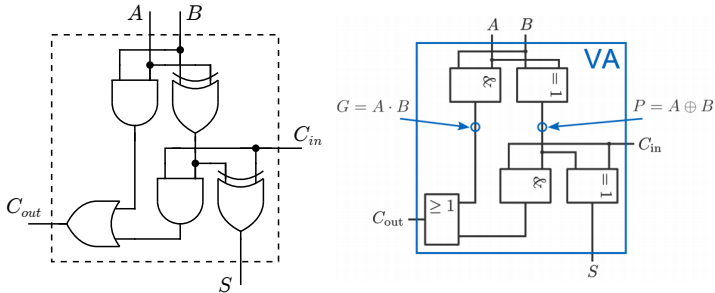
t_{pd} ist Zeit zwischen crossover 50% von Eingang zu crossover 50% am Ausgang.

Steigend mit: Kapazitiver Last, Oxiddicke, Kanallänge, Schwellspannung

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

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

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

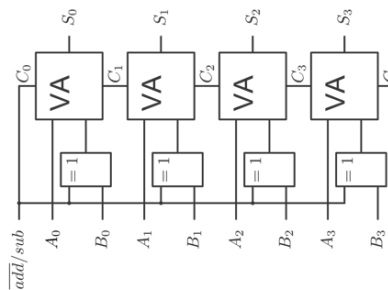


Generate $g_n = a_n \cdot b_n$
Propagate $p_n = a_n \oplus b_n$
Summenbit $S_n = c_n \oplus p_n = a_n \oplus b_n \oplus c_n$
 $S_n = a_n b_n c_n + a_n b_n \bar{c}_n + \bar{a}_n b_n c_n + \bar{a}_n b_n \bar{c}_n$ (Ungerade Anzahl von Eingängen 1)
 genau ein Eingang high alle Eingänge high
Carry-out $c_{n+1} = c_n \cdot p_n + g_n$
 $c_{n+1} = a_n b_n \bar{c}_n + a_n \bar{b}_n c_n + \bar{a}_n b_n c_n + \bar{a}_n \bar{b}_n \bar{c}_n$ (Mindestens zwei Eingänge 1)
 zwei Eingänge 1 drei Eingänge 1

Laufzeiten
 $t_{sn} = \begin{cases} t_{cn} + t_{xor} & t_{cn} > t_{xor} \\ 2t_{xor} & \text{sonst} \end{cases}$
 $t_{cn+1} = \begin{cases} t_{and} + t_{or} & a_n = b_n = 1 \\ t_{xor} + t_{and} + t_{or} & a_n = b_n = 0 \\ t_{cn} + t_{and} + t_{or} & a_n \neq b_n \end{cases}$ ($g_n = 1$) ($p_n = 0, g_n = 0$) ($p_n = 1$)

11.1 Multibit Addierer / Subtrahierer

Subtraktion entspricht Addition mit negativem Subtrahenden
 Zweierkomplement zur Bildung des negativen Subtrahenden
 → Invertieren aller Bits des Subtrahenden und Addition von 1
 $XOR: X \oplus 0 = X, X \oplus 1 = \bar{X}$



12 Sequentielle Logik

Logik mit Gedächtnis (Speicher).

12.1 Begriffe/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
Min. Taktperiode	$t_{clk} \geq t_1, c2q + t_{logic, max} + t_2, setup$
Max. Taktfrequenz	$f_{max} = \left\lfloor \frac{1}{t_{clk}} \right\rfloor$ (Nicht aufrunden)
Holdzeitbedingung	$t_{hold} \leq t_{c2q} + t_{logic, min} \rightarrow$ Dummy Gatter einbauen
Durchsatz	$\frac{1}{t_{sample}} = f$
Latenz	$t_{clk} \cdot \# \text{Pipelinstufen}$ (das zwischen den FFs)

12.2 Pipelining

Nur bei synchronen(taktgesteuerten) Schaltungen möglich!

- Aufteilen langer kombinatorischer Pfade 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

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

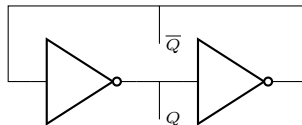
- Paralleles, gleichzeitiges Verwenden mehrere identischer Schaltnetze
- Zusätzliche Kontrolllogik nötig (Multiplexer)
- Taktfrequenz und Latenz bleiben konstant
- Durchsatz steigt mit der Zahl der Verarbeitungseinheiten
ABER: deutlich höherer Ressourcenverbrauch

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.

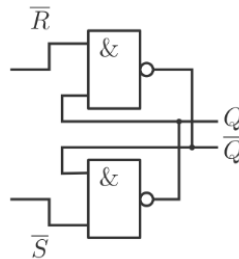
$$\text{Speicherkapazität} = \text{Wortbreite} \cdot 2^{\text{Adressbreite}}$$

13.1 Speicherzelle/Register



Ring aus zwei Invertiern.
 Logikpegel kann nur mit öffnen des Inverter-Rings gesetzt werden.

13.2 Latch



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

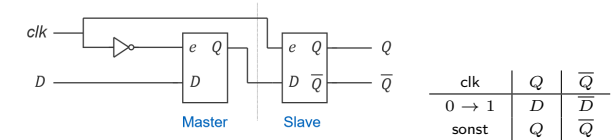
Active Low Logik:
 $\bar{S} = 0 \Rightarrow Q = 1, \bar{R} = 0 \Rightarrow Q = 0$

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

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

e	Q
0	Q
1	D

13.3 Flip-Flop



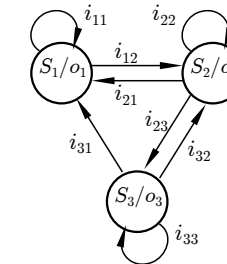
Besteht aus zwei enable-Latches
Flip-Flop: Ändert Zustand bei steigender/(fallender) Taktflanke.

14 Automaten

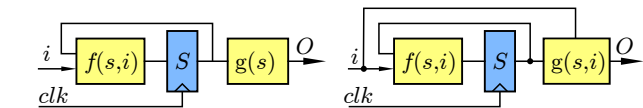
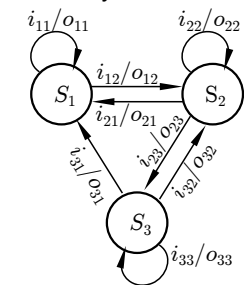
DFA 6-Tupel $\{I, O, S, R, f, g\}$

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 \Rightarrow 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	$o_{0,0 \dots 0}$	$S'_{0,0 \dots 0}$
...
1	1...1	$o_{1,1 \dots 1}$	$S'_{1,1 \dots 1}$

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