

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 | $VS A^{-1}$ |

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

3 Polyadische Zahlensysteme

$$Z = \sum_{i=-n}^{p-1} r^i \cdot d_i = d_{p-1} \dots d_1 d_0 . d_{-1} \dots d_n$$
$$Z: \text{Zahl}, \quad r: \text{Basis}, \quad d_i: \text{Ziffer}, \quad p: \# \text{Ziffern vorne} \quad n: \# \text{Nachkommastellen}$$

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

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

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

3.1 Umrechnung

| | $Z \geq 1$ | $Z < 1$ |
|--------------------|--|--|
| $r \rightarrow 10$ | $Z_{10} = \sum r^i \cdot d_i$ $101_2 \rightarrow 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1$ | $Z_{10} = \sum r^{-i} \cdot d_{-i}$ $0.11_2 \rightarrow 1 \cdot 0.5 + 1 \cdot 0.25$ |
| $10 \rightarrow r$ | $d_i = Z_{10} \% r^i$ $58/8 = 7$ Rest $2(LSB)$ $7/8 = 0$ Rest $7(MSB)$ (Ende wenn 0 erreicht) | $0.4 \cdot 2 = 0.8$ 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 | | |
| Spezialwerte: $Z = 0 \Leftrightarrow e = 0$ $Z = +(-)\infty \Leftrightarrow e = 255, s = 0(1)$ | | |
| IEEE \rightarrow Wert Z $Z = (-1)^s \cdot 1.f \cdot 2^{e-127}$ | 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) $s = 0(\text{positiv}), s = 1(\text{negativ})$ $Z \rightarrow Z_2$ (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) $s = 0(\text{positiv}), s = 1(\text{negativ})$ $E = \lfloor \log_2 Z \rfloor$ $e = E + 127 \rightarrow e_2$ $f = \left(\frac{ Z }{2^E} - 1\right) \cdot 2^{23} \rightarrow f_2$ | Bsp: $Z = 11.25$ $s = 0$ $E = \lfloor \log_2 11.25 \rfloor = \lfloor 3,49\dots \rfloor = 3$ $e = 3 + 127 = 130 = 10000010_2$ $f = \left(\frac{ 11.25 }{2^3} - 1\right) \cdot 2^{23} = 3407872 = 01101000\dots_2$ | |

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

5 Boolsche Algebra

5.1 Boolesche Operatoren (Wahrheitstabelle WT)

| | |  |  |  |  |  |  |
|---|---|---|--|---|---|---|---|
| x | y | AND | OR | XOR | NAND | NOR | EQV |
| 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: $f = c_1 + c_2 + c_3 \Rightarrow cov(f) = \{c_1, c_2, c_3\}$

5.2 Gesetze der booleschen Algebra

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

5.3 Boolesche Funktionen

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

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

Kofaktor bezüglich

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

Eigenschaften von $f(\underline{x})$

- tautologisch $\Leftrightarrow f(\underline{x}) = 1 \quad \forall \underline{x} \in \{0, 1\}^n$
- tautologisch $\Leftrightarrow f(\underline{x}) = 1 \quad \forall \underline{x} \in \{0, 1\}^n$
- kontradiktorisch $\Leftrightarrow f(\underline{x}) = 0 \quad \forall \underline{x} \in \{0, 1\}^n$
- unabhängig von $x_i \Leftrightarrow f_{x_i} = f_{\bar{x}_i}$
- abhängig von $x_i \Leftrightarrow f_{x_i} \neq f_{\bar{x}_i}$

5.4 Multiplexer

$f = x \cdot a + \bar{x} \cdot b$ (2 Eingänge a, b und 1 Steuereingang x)
 $f = \bar{x}_1 \bar{x}_2 a + \bar{x}_1 x_2 b + x_1 \bar{x}_2 c + x_1 x_2 d$ (Eingänge: a, b, c, d Steuerung: x_1, x_2)

5.5 Wichtige Begriffe

| Wichtige Begriffe: | Definition | Bemerkung |
|--------------------|---|----------------------------------|
| Signalvariable | x | $\hat{x} \in \{0, 1\}$ |
| Literal | $l_i = x_i$ oder \bar{x}_i | $i \in I_0 = \{1, \dots, n\}$ |
| Minterme, 0-Kuben | $MOC \ni m_j = \prod_{i \in I_0} l_i$ | $ MOC = 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 l \in c_i \wedge \bar{l} \in c_j\} $ | $\delta_{ij} = \delta(c_i, c_j)$ |
| Implikanten | $MI = \{c \in MC c \subseteq f\}$ | |
| Primimplikanten | $MPI = \{p \in MI p \not\subseteq c \forall c \in MI\}$ | $MPI \subseteq MI \subseteq MC$ |

| | | |
|-----------------|-----------------------------------|--|
| SOP (DNF) | eine Summe von Produkttermen | Terme sind ODER-verknüpft |
| POS (KNF) | ein Produkt von Summentermen | Terme sind UND-verknüpft |
| CSOP (KDNF) | Summe aller Minterme | WT: 1-Zeilen sind Minterme |
| CPOS (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/SOP)

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/POS)

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

6.3 Umwandlung in jeweils andere Form

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

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

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

| \sum^{xy} | 00 | 01 | 11 | 10 |
|-------------|----|----|----|----|
|-------------|----|----|----|----|

0 1 0 0 0

1 X 1 1 0

Don't Care Werte ausnutzen!

7.3 Quine Methode

geg.: DNF/SOP oder Wertetabelle von $f(x)$

ges.: alle Primimplikanten p_i (VollSOP)

Spezielles Resoluionsgesetz: $x \cdot a + \overline{x} \cdot a = a$
Absorptionsgesetz: $a + a \cdot b = a$

- KDNF/CSOP 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 | $\overline{x}_1\overline{x}_2x_3$ | ✓ | \overline{x}_2x_3 | $m_1\&m_5$ | p_1 | | |
| m_4 | $x_1\overline{x}_2\overline{x}_3$ | ✓ | $x_1\overline{x}_2$ | $m_4\&m_5$ | ✓ | | |
| m_5 | $x_1\overline{x}_2x_3$ | ✓ | $x_1\overline{x}_3$ | $m_4\&m_6$ | ✓ | x_1 | p_2 |
| m_6 | $x_1x_2\overline{x}_3$ | ✓ | x_1x_3 | $m_5\&m_7$ | ✓ | | |
| m_7 | $x_1x_2x_3$ | ✓ | x_1x_2 | $m_6\&m_7$ | ✓ | | |

$$\Rightarrow f(x_1,x_2,x_3) = p_1 + p_2 = \overline{x}_2x_3 + x_1$$

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

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

Alternativ: Mit Überdeckungstabelle bestimmen. Bsp:

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

K : Anzahl der Primterme

N : Anzahl der Minterme

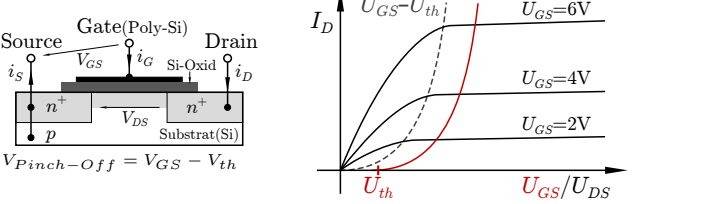
$L(p_i)$: Kosten/Länge der Primimplikanten

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 Effect Transistor



9.1 Bauteilparameter

| | |
|----------------------------------|--|
| Verstärkung: | $\beta = K' \frac{W}{L}$ mit $K' = \frac{\mu \varepsilon_{ox} \varepsilon_0}{t_{ox}}$ $[\beta] = \frac{A}{V^2}$ |
| Kanalweite | W |
| Kanallänge | L |
| Elektronenbeweglichkeit | μ $\mu_n \approx 250 \cdot 10^{-4} \frac{m^2}{Vs}$, $\mu_p \approx 100 \cdot 10^{-4} \frac{m^2}{Vs}$ |
| rel. Dielektrizität des Gateoxys | $\varepsilon_{ox} \approx 3,9$ |
| Dielektrizitätskonstante | $\varepsilon_0 = 8.8541878 \cdot 10^{-12} \frac{As}{Vm}$ |
| Gateoxydicke | t_{ox} |
| Verstärkung | $\beta = \frac{\mu_n \varepsilon_{ox} \varepsilon_0}{t_{ox}} \cdot \frac{W}{L} = K' \frac{W}{L} = \frac{\mu_n C_G}{L^2}$ |
| Kapazität | $C_G = \varepsilon_{ox} \varepsilon_0 \frac{WL}{t_{ox}}$ |
| Verzögerungszeit | $t_{pHL} \propto \frac{C_L t_{ox} L p}{W_p \mu_p \varepsilon_{ox} (V_{DD} - V_{th})}$ |

- 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)

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

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

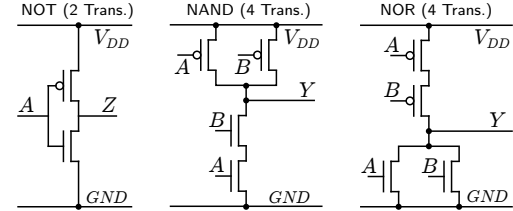
$$I_d = \begin{cases} 0, & \text{für } U_{gs} - U_{th} \geq 0 \quad (\text{Sperrber.}) \\ -\beta[(u_{gs} - U_{th}) \cdot u_{ds} - \frac{1}{2} u_{ds}^2], & \text{für } 0 \geq U_{gs} - U_{th} \leq u_{ds} \quad (\text{linearer Ber.}) \\ -\frac{1}{2} \beta \cdot (u_{gs} - U_{th})^2, & \text{für } 0 \geq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

9.3 pMos und nMos

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

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.

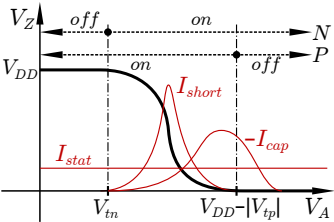
10.1 Gatterdesign

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

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

10.2 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 $P_{short} = \alpha_{01} f \beta_n \tau (V_{DD} - 2V_{tn})^3$
Schalthäufigkeit $\alpha_{0 \rightarrow 1} = \frac{\text{Schaltvorgänge(pos. Flanke)}}{\# \text{ Betrachtete Takte}}$
Schalthäufigkeit (periodisch) $\alpha = \frac{f_{switch}}{f_{clk}}$

Abhängig von den Signalfanken, mit Schaltfunktionen verknüpft
 $\approx V_{DD1} / \propto$ Schaltzeit: $\frac{V_{DD2}}{V_{DD1}} = \frac{t_{D1}}{t_{D2}}$ (bei Schaltnetzen t_{log})

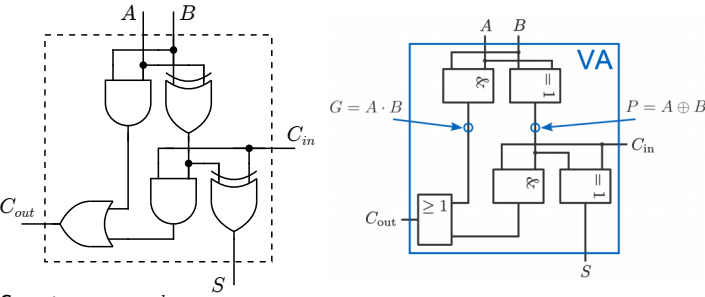
Verzögerungszeit $\propto \frac{C_L t_{ox} L_p}{W_{pmp} \epsilon (V_{DD} - V_{th})}$

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

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

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 | $f_{max} = \lfloor \frac{1}{t_{clk}} \rfloor$ (Nicht aufrunden) |
| Holdzeitbedingung | $t_{hold} \leq t_{c2q} + t_{logic,min} \rightarrow$ Dummy Gatter einbauen |
| Durchsatz | $\frac{1 \text{ Sample}}{t_{clk,pipe}} = f$ |
| Latenz | $t_{clk} \cdot \# \text{ Pipelinestufen (das zwischen den FFs)}$ |

12.2 Pipelining

Nur bei synchronen(taktgesteuerten) Schaltungen möglich!

- Aufteilen langer kombinatorischer Pfade durch Einfügen zusätzlicher Registerstufen
 \rightarrow 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 = $\frac{\# \text{ Modul}}{t_{clk,Modul}} = f$ 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.

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

13.1 Speicherzelle/Register

Ring aus zwei Invertern.

13.2 Latch

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

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

13.3 Flip-Flop

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

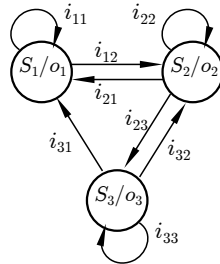
| | | |
|-------------------|---|-----------|
| clk | Q | \bar{Q} |
| 0 \rightarrow 1 | D | \bar{D} |
| sonst | Q | \bar{Q} |

14 Automaten

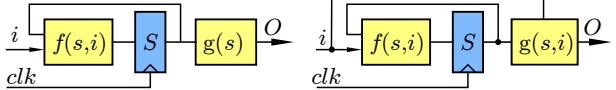
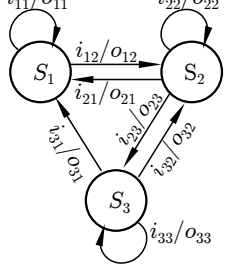
DFA 6-Tupel $\{I, O, S, R, f, g\}$

| | |
|--------------------------------|---------------------------|
| 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



| Moore | Mealy |
|---|--|
| Opout hängt nur vom Zustand ab $s' = f(s, i), o = g(s)$ $g : S \rightarrow O$ | Output hängt von Zustand und Eingabe ab $s' = f(s, i), o = g(s, i)$ $g : S \times I \rightarrow O$ |