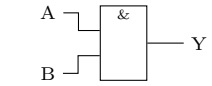
Digitaltechnik

Andrej Scheuer
ascheuer@student.ethz.ch
9. Januar 2021

Gates

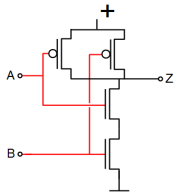
AND

$Y = A \wedge B \quad Y = A \cdot B$

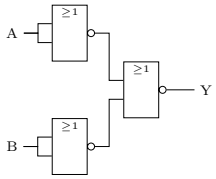


A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

NAND

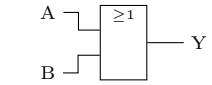


AND aus NOR



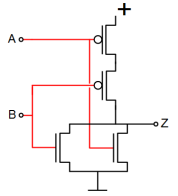
OR

$Y = A \vee B \quad Y = A + B$

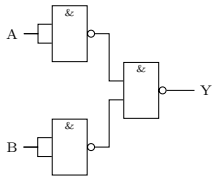


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

NOR

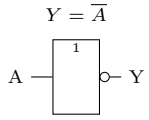


OR aus NAND

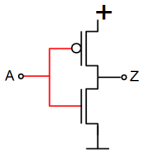


NOT

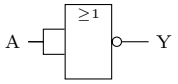
$Y = \overline{A}$



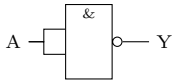
A	Y
0	1
1	0



NOT aus NOR



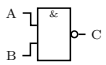
NOT aus NAND



Weitere Gates

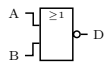
NAND

$C = \overline{A \wedge B}$



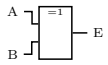
NOR

$D = \overline{A \vee B}$



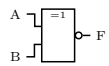
XOR

$E = A \oplus B$



XNOR

$F = \overline{A \oplus B}$

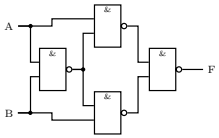


A	B	NAND C	NOR D	XOR E	XNOR F
0	0	1	1	0	1
0	1	1	0	1	0
1	0	1	0	1	0
1	1	0	0	0	1

$\text{XOR} = (A \wedge \overline{B}) \vee (\overline{A} \wedge B)$

$\text{XNOR} = (A \wedge B) \vee (\overline{A} \wedge \overline{B})$

XOR aus NAND



XOR aus NOR: Gleiches Schema wie NAND + 1 Inverter

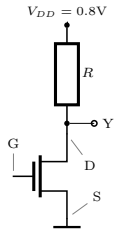
XNOR aus NAND: Gleiches Schema wie *XOR aus NOR*

XNOR aus NOR: Gleiches Schema wie *XOR aus NAND*

Es versteht sich natürlich, dass wenn von „Gleichem Schema wie…" gesprochen wird, die Gates trotzdem getauscht werden müssen.

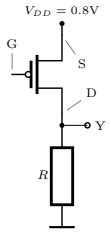
CMOS

NMOS



G	Schalter	Y
0	offen	1
1	zu	0

PMOS



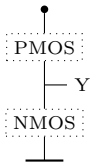
G	Schalter	Y
0	zu	1
1	offen	0

Konstruktion von CMOS-Gates

Regeln für CMOS-Schaltungen

1. CMOS-Gates bestehen aus gleich vielen NMOS und PMOS.
2. m Eingänge: m NMOS und m PMOS.
3. **NMOS in Serie → PMOS parallel**
4. **NMOS parallel → PMOS Serie**

Allg. Aufbau CMOS

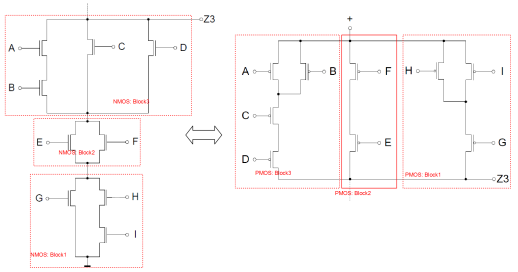


Pull-up: PMOS
Pull-down: NMOS

Pfade sind komplementär (**Serie ⇔ Parallel**)

Umwandlung Pull-up zu Pull-down

1. Teilbereiche (Blöcke) identifizieren.
2. Schritt 1 wiederholen, bis nur noch einzelne Transistoren vorkommen.
3. Falls Pull-down:
   - Von GND aus mit äusserstem Block beginnen.
   - PMOS → NMOS
4. Falls Pull-up:
   - Von V_{DD} aus mit äusserstem Block beginnen.
   - NMOS → PMOS.



Funktionsgleichung

parallel: $\vee$ Serie: $\wedge$	Pull-Up: $y = 1$ Pull-Down: $y = 0$	alle I : 0 → I invert. alle I : 1 → Gl. invert.

Boolsche Algebra

Grundregeln

Kommutativität

$$A \wedge B = B \wedge A$$
$$A \vee B = B \vee A$$

Assoziativität

$$A \wedge (B \wedge C) = (A \wedge B) \wedge C$$
$$A \vee (B \vee C) = (A \vee B) \vee C$$

Distributivität

$$(A \wedge B) \vee (A \wedge C) = A \wedge (B \vee C)$$
$$(A \vee B) \wedge (A \vee C) = A \vee (B \wedge C)$$

Nicht	$\overline{\overline{A}} = A$	
Null-Th.	$A \vee 0 = A$	$A \wedge 0 = 0$
Eins-Th.	$A \vee 1 = 1$	$A \wedge 1 = A$
Idempotenz	$A \vee A = A$	$A \wedge A = A$
V. Komp.	$A \vee \overline{A} = 1$	$A \wedge \overline{A} = 0$
Adsorp.	$A \vee (\overline{A} \wedge B) = A \vee B$	
	$A \wedge (\overline{A} \vee B) = A \wedge B$	
Adsorp.	$A \vee (A \wedge B) = A$	
	$A \wedge (A \vee B) = A$	
Nachbar.G.	$(A \wedge B) \vee (\overline{A} \wedge B) = B$	
	$(A \vee B) \wedge (\overline{A} \vee B) = B$	

De Morgan

1. Regel $\quad \overline{A \wedge B} = \overline{A} \vee \overline{B}$
2. Regel $\quad \overline{A \vee B} = \overline{A} \wedge \overline{B}$

Regeln gelten auch für n verknüpfte Terme.

Normalformen

Minterm	Maxterm
AND-Ausdruck	OR-Ausdruck
Output: 1	Output: 0
n Schaltvar. → 2^n mögl. Minterme.	n Schaltvar. → 2^n mögl. Maxterme.
nicht-invertierte Var: 1	nicht-invertierte Var: 0
invertierte Var: 0	invertierte Var: 0

Kanonisch Normalform: Alle Terme einer Schaltfunktion; nicht vereinfacht oder gekürzt.

Disjunktive Normalform

1. Identifiziere WT-Zeilen mit Output 1
2. **Minterme** für diese Zeilen aufstellen
3. Minterme mit **OR** verknüpfen

Konjunktive Normalform

1. Identifiziere WT-Zeilen mit Output 0
2. **Maxterme** für diese Zeilen aufstellen
3. Maxterme mit **AND** verknüpfen

A	B	Y	Minterme	Maxterme
0	0	1	$\overline{A} \wedge \overline{B}$	
0	1	0		$A \vee \overline{B}$
1	0	0		$\overline{A} \vee B$
1	1	1	$A \wedge B$	

DNF $\quad Y = (\overline{A} \wedge \overline{B}) \vee (A \wedge B)$ $\quad$ 1 Mint. erf. → 1

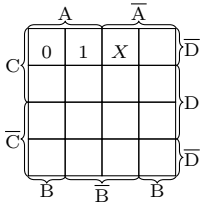
KNF $\quad Y = (A \vee \overline{B}) \wedge (\overline{A} \vee B)$ $\quad$ 1 Maxt. erf. → 0

NAND/NOR Schaltungen

Schaltung nur aus:

- NAND: DNF → 2× Negieren → 1× De Morgan
- NOR: KNF → 2× Negieren → 1× De Morgan oder: auf jeden Term der DNF De Morgan anwenden.

Karnaugh Diagramme (KVD)

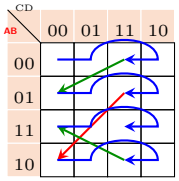


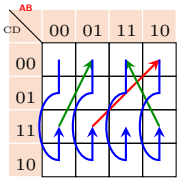
CD \ AB	00	01	11	10
00	0	1	X	
01				
11				
10				

Hat das Karnaugh Diagramm 5 Dimensionen, wird die 5te Dimension auf zwei Tabellen aufgeteilt.

Don't-Care-Zustände $X \in \{0, 1\}$ Redundante, überflüssige oder unmögliche Kombinationen der Eingangsvariablen werden mit einem X markiert.

Schema zum Ausfüllen





Päckchen

- Päckchen immer rechteckig (Ausnahme: über Ecken).
- Umfassen möglichst grosse Zweierpotenz.
- Dürfen über Ecken und Grenzen hinausgehen und sich überlappen.

DNF

1. KVD ausfüllen.
2. Päckchen mit **1** uo X.
3. Vereinfachte Minterme aufstellen.
4. Minterme mit OR verbinden.

KNF

1. KVD ausfüllen.
2. Päckchen mit **0** uo X.
3. Vereinfachte Maxterme aufstellen.
4. Maxterme mit AND verbinden.

Hazard

Kurzzeitige, unerwünschte Änderung der Signalwerte, die durch Zeitverzögerung der Gatter entstehen.

CD \ AB	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	1	0	0
10	1	1	0	0

Hazard Hazard

Statische Hazards Stellen im KVD, an denen sich Päckchen orthogonal berühren, aber nicht überlappen.
Lösung Berührende Päckchen mit zusätzlichen (möglichst grossen) Päckchen verbinden.

Zahlensysteme

D	zu berechnende positive Zahl
R	Basis/Radix von D
b_i	Koeffizient

$$D = \sum_{-\infty}^{\infty} b_i \cdot R^i$$

Darstellung D in Basis R: $\ldots b_2 b_1 b_0 . b_{-1} b_{-2} \ldots_R$

Dezimal	10	$b_i \in \{0, 1, \ldots, 9\}$
Dual/Binär	2	$b_i \in \{0, 1\}$
Oktal	8	$b_i \in \{0, 1, \ldots, 7\}$
Hexa	16	$b_i \in \{0, 1, \ldots, 9, A, B, C, D, E, F\}$

Umwandlung Zahlensysteme

1. Ganzzahlige Division mit R: $D/R = Q_0 + r_0$.
2. $$Q_i/R = Q_{i+1} + r_{i+1}$$
   bis $Q_i = 0$.
3. Erste Operation gibt MSB, letze Operation gibt LSB (aka. unten nach oben lesen.)

Für $1 > D \geq 0$

$$D \cdot R = P_0 \quad K_{-1} = \text{floor}\,(P_0) \quad a_{-1} = P_0 - K_{-1}$$
$$a_{-1} \cdot R = P_{-1} \ldots$$

K_i: Koeffizienten für Zahlensystem. Erste Operation gibt **MSB**, letze Operation gibt **LSB** (aka von oben nach unten lesen).

Binär zu Dezimal

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

2^{-1}	2^{-2}	2^{-3}	2^{-4}
0.5	0.25	0.125	0.0625

Binär zu Hex

0000	0	0100	4	1000	8	1100	C
0001	1	0101	5	1001	9	1101	D
0010	2	0110	6	1010	A	1110	E
0011	3	0111	7	1011	B	1111	F

Zweierkomplement

Sign Bit
0: positiv 1: negativ

Konstruktion

1. Zahl $|Z|$ in Binär B umwandeln.
2. B bitweise invertieren
3. 1 zu LSB addieren (! Übertrag)
4. Sign Bit hinzufügen (zuvorderst).

Ist die Blocklänge länger als Zahl, vorangehende 0(-en) miteinbeziehen.

2^{er}Komplement zu Dezimal

$$D_{(10)} = -b_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} b_i \cdot 2^i$$

Wertebereich 2^{er}-Komp. $\left[-2^{n-1}, 2^{n-1} - 1\right]$

mQn

$$D_{(10)} = -b_m \cdot 2^m + \sum_{i=0}^{m-1} b_i \cdot 2^i + \sum_{i=1}^{n} b_i \cdot 2^{-i}$$

m: Vorkommabits, n: Nachkommabits
Sign-Bit muss nur einmal vor dem m codiert werden.

Binäre Rechenoperationen

Addition

Bitweise Addition der Binärzahlen. Leere Slots werden mit 0 aufgefüllt.

Subtraktion

Addition via 2^{er}Komp. Übertrag von MSB ignorieren.

Multiplikation

1. Bitweise Multiplikation des Multiplikanden a mit b_i des Multiplikator.
2. Sukzessive Multiplikationen werden um ein Bit (0) nach links verschoben.
3. Anzahl Nachkommabits ergibt sich aus der Summe der Anzahl Nachk.bits der Operatoren.

$$\begin{array}{r} b_0 \cdot a \\ +b_1 \cdot a\ 0 \\ +b_2 \cdot a\ 0\ 0 \\ +b_3 \cdot a\ 0\ 0\ 0 \\ \hline = \text{Sum} \end{array}$$

Division

1. Identifiziere Teil des Divident > Divisor (Unterblock). Für jede Stelle, sodass Divident < Divisor, 0 in Quotient.
2. Unterblock − Divisor, 1 an Quotient anhängen, Rest behalten.
3. An das Resultat der Subtraktion Bits des Dividenten anhängen. Wiederholen bis Subtraktion 0 ergibt.

$$\begin{array}{r|l} 11100 : 0111 = 0100 & \\ \hline 1\ \%\ 111 = 0 & R = 1 \\ 11\ \%\ 111 = 0 & R = 11 \\ 111\ \%\ 111 = 1 & R = 000 \\ 0000\ \%\ 111 = 0 & R = 0000 \\ 00000\ \%\ 111 = 0 & R = 00000 \end{array}$$

Parity-Bits

Hilft Bit-Fehler zu finden.
Bitsequenz wird in 4 Bits unterteilt. Pro Nibble wird ein Parity-Bit angefügt. Nach 4 Blöcken folgt ein Prüfwort.

Parity-Bit	Anz. 1	PB	Nibble + PB
Even P_E	ungerade	1	
	gerade	0	gerade
Odd P_O	ungerade	0	
	gerade	1	ungerade

01010 11011 10111 00101 00011

Korrekt P_E

0	1	0	1	0
1	1	0	1	1
1	0	1	1	1
0	0	1	0	1
0	0	0	1	1

Fehler P_E

0	1	0	1	0
1	1	1	1	1
1	0	1	1	1
0	0	1	0	1
0	0	0	1	1

Latches und FlipFlops

Kombinatorische Schaltung
Output hängt von Inputs und Verknüpfungen ab.

Sequentielle Schaltung
Enthält Rückkopplungen, Outputs hängen von vorherigen Werten ab.

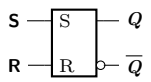
Latch
(Takt)zustandgesteurte Schaltung → Änderungen am Eingang können während der **ganzen aktiven Taktphase** den Output beeinflussen.

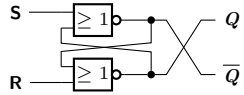
FlipFlops
Taktflankengesteuerte Schaltung → Input zum Zeitpunkt der Taktwechsels wird wirksam.

Latches

Alle taktzustandgesteurte Schalt. sind gegenüber Störimpulsen empfindlich. (T = 1 übernimmt jede Änderung)

SR-Latch



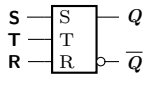


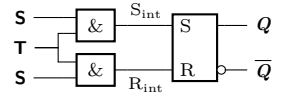
S	Set	→	setzt Q auf 1
R	Reset	→	setzt Q auf 0

$$Q_{n+1} = S \vee \left(Q_n \wedge \overline{R}\right)$$

Fall	**S**	**R**	Q_{n+1}	
1	0	0	Q_n	speichern
2	0	1	0	zurücksetzten
3	1	0	1	setzen
4	1	1	-	unzulässig

SRT-Latch

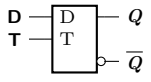


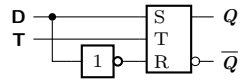


T		S_{int}	R_{int}	
0	→	0	0	→ Datenspeicherung
1	→	S	R	→ Normales SR-Latch

Änderungen werden nur übernommen, wenn T/CLK aktiv ist.

D-Latch

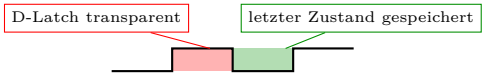




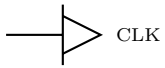
Bauelement, das Daten für die Periodendauer eines Taktes speichern kann.

$$Q_{n+1} = \left(Q_n \wedge \overline{\mathrm{T}}\right) \vee (\mathrm{D} \wedge \mathrm{T})$$

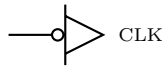
T	Q_{n+1}		
0	Q_n	→	alter Ausgang gespeichert
1	D	→	Input übernommen



FlipFlops



Input beim Übergang von **0** → **1** von CLK wirksam.

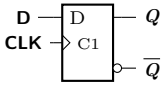


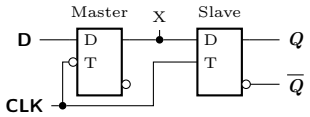
Input beim Übergang von **1** → **0** von CLK wirksam.

Positive/steigende Taktflanke

Negative/fallende Taktflanke

D-FlipFlop

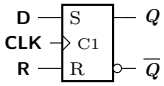


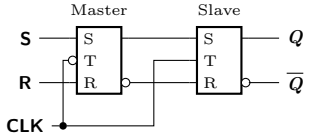


$Q_{n+1} = \mathrm{D}$ wenn CLK $0 \to 1$

Master	low-active	CLK = 0
Slave	high-active	CLK = 1

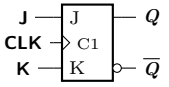
SR-FlipFlop

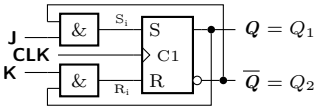




$Q_{n+1} = \mathrm{S} \vee \left(\overline{\mathrm{R}} \wedge Q_n\right)$ wenn CLK $0 \to 1$

JK-FlipFlop





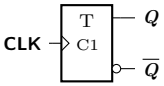
$Q_{n+1} = \left(J \wedge \overline{Q_n}\right) \vee \left(\overline{K} \wedge Q_n\right)$ wenn CLK $0 \to 1$

Fall	J	K	Q_{1n+1}	Q_{2n+1}	
1	0	0	Q_{1n}	Q_{2n}	speichern
2	0	1	0	1	zurücksetzten
3	1	0	1	0	setzen
4	1	1	$\overline{Q_{1n}}$	$\overline{Q_{2n}}$	wechseln

Bei J = K = 1 wechselt Output. (toggel)

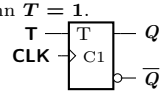
T-FlipFlop

V1 Ausgang wechselt bei jeder aktiven Taktflanke.



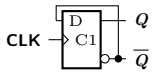
$Q_{n+1} = \overline{Q_n}$
wenn CLK $0 \to 1$

V2 Ausgang wechselt bei aktiver Taktflanke nur wenn $T = 1$.

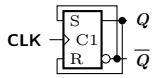


$Q_{n+1} = \overline{Q_n}$
wenn CLK $0 \to 1 \wedge T = 1$

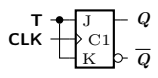
BS V1



BS V1

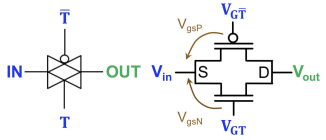


BS V2



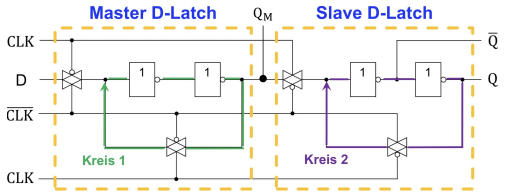
D-FlipFlop in CMOS-Technik

Transmission Gates

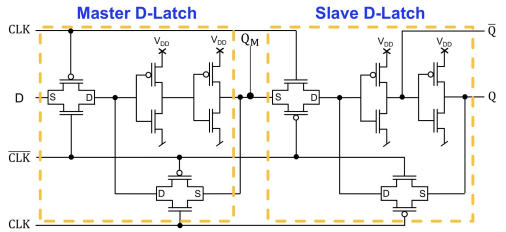


IN	T	Widerstand	OUT
0	0	hochohm.	-
0	1	niederohm.	0
1	0	hochohm.	-
1	1	niederohm.	1

TG sperrt wenn Widerstand hochohmig ist. (T = 0)



CLK
0 Input ins erste Latch übertragen
1 Latch verriegelt, Wert im Kreis gefangen

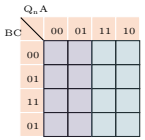


D-FlipFlop ⇔ JK-FlipFlop

1. JK-FF kann immer durch D-FF ersetzt werden.
   D-FF: $D_n = \left(J \wedge \overline{Q_n}\right) \vee \left(\overline{K} \wedge Q_n\right)$:JK-FF
2. Ein D-FF kann nur durch JK-FF ersetzt werden wenn:
   a) Schaltung eine Rückkopplung enthält.
   b) Input D als $\left(F_1 \wedge \overline{Q_n}\right) \vee \left(F_2 \wedge Q_n\right)$ geschrieben werden kann.

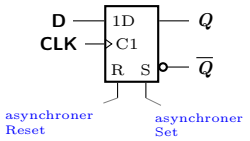
Gleichung für D-FF → JK-FF

1. Wahrheitstabelle mit Einängen und Rückkopplung.
2. Wahrheitstabelle in Q_n und $\overline{Q_n}$.
3. Separat Päckchen in Q_n und $\overline{Q_n}$ machen.
4. Päckchen mit OR verbinden. Ggf. Q_n und $\overline{Q_n}$ ausklammern.



Asynchroner Set/Reset Input

Können gespeicherte Zustände asynchron zu CLK überschreiben.



Verzögerungszeiten

t_s	Setup-Zeit	Solange muss Signal vor aktiver Taktflanke stabil anliegen.
t_h	Hold-Zeit	Solange muss Signal nach aktiver Taktflanke stabil anliegen.
t_{pd}	Verzögerungszeit	Durchlaufzeit

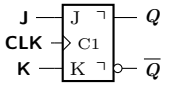
$$T_{\min} \geq t_{\mathrm{pd1}} + t_{\mathrm{pd,ks}} + t_{\mathrm{s2}} \qquad f_{\max} = \frac{1}{T_{\min}}$$

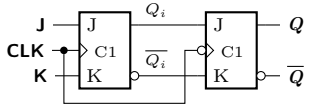
t_h kann bei der Berechnung von $f_{\max}$ vernachlässigt werden.
Es wird der längste Pfad zwischen zwei FlipFlops betrachtet.

Fehlfunktionen beim Wechsel des Eingagnssignals

$\frac{\text{vor}}{\text{nach}}$ aktiver Taktfl: $\frac{\text{längster}}{\text{kürzester}}$ Pfad zw. FF

Zwischenspeicher-FF

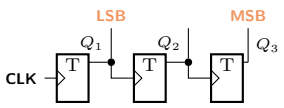


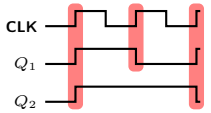


FlipFlop, dass Input bei steigender Taktflanke übernimmt und bei der nächsten fallenden Taktflanke ausgibt. (oder umgekehrte Flanken)

⌝ Ausgabe bei fallender Flanke
⌟ Ausgabe bei steigender Flanke

Frequenzteiler und Zähler





Kaskadieren von T-Flipflops führt zu einer Frequenzreduktion von CLK um Faktor 2. Kann als Bitzähler verwendet werden (ohne CLK). MSB ist längste Frequenz.
$n_{T,ff} \to 0 \ldots (2^n - 1)$ $f_R = \frac{f_C}{2^n}$

Automaten

Ein System, das auf seine Eingänge reagiert und einen Ausgang produziert, der vom Eingangssignal und momentanen Zustand abhängt.
Bei synchronen Automaten besitzen alle Speicherelemente (FlipFlops) den gleichen Takteingang.

Formale Beschreibung

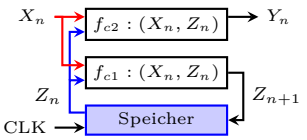
$X = (x_1, \ldots, x_e)$	Eingangsalphabet mit **e** Eingängen
$Y = (y_1, \ldots, y_b)$	Ausgangsalphabet mit **b** Ausgängen
$Z = (z_1, \ldots, z_m)$	Zustandsmenge mit **m** internen Zuständen
$Z_0 \in Z$	Anfangszustand
$f_{c1} : (X_n, Z_n) \to Z_{n+1}$	Übergangsfunktion
$f_{c2} : (X_n, Z_n) \to Y_n$	Ausgangsfunktion

Automatentypen

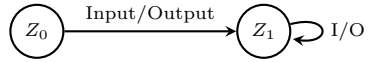
Mealy-Automat

Ausgänge von inneren Zuständen und Eingängen abhängig.

$Y_n = f_{c2}(X_n, Z_n)$



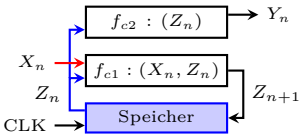
Zustandsdiagramm



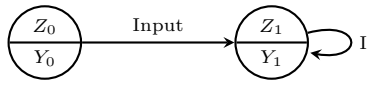
Moore-Automat

Ausgänge nur von inneren Zuständen abhängig (keine Verbindung zwischen Input und Output).

$Y_n = f_{c2}(Z_n)$



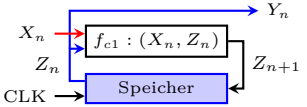
Zustandsdiagramm



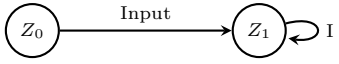
Medwedjew-Automat

Ausgänge entsprechen inneren Zuständen.

$Y_n = Z_n$



Zustandsdiagramm



Nachtrag Zustandsdiagramm

Knoten interne Zustände
Kanten Übergänge zwischen Zuständen

Wichtig Von jedem Knoten aus muss es für jeden Eingang eine Kante geben, diese können aber zusammengefasst werden.

Zustandsfolgetabelle

Auflistung aller möglichen Kombinationen der aktuellen inneren Zuständen sowie den Eingängen mit den dazugehörigen Folgezuständen und Ausgängen.

$x_1 \dots x_e$	$z_{1n} \dots z_{mn}$	$z_{1(n+1)} \dots z_{m(n+1)}$	$y_1 \dots y_b$

$e + 2m + b$ Spalten

2^{e+m} Zeilen

Wichtig: für e, m, b Anzahl Bits verwenden, nicht Anzahl Zustände.

Entwurf eines Automaten

1. Auftrag lesen und analysieren → Automatentyp bestimmen.
2. Zustandsmenge bestimmen → Anzahl erforderlich D-FlipFlops $\lceil \log_2(\text{Anzahl Zustände}) \rceil$.
3. Eingangs- und Ausgangsvariablen definieren, Kodierung.
4. Darstellung der Zustandsfolge in einem Zustandsdiagram.
5. Zustandsfolgetabelle aufstellen.
6. Minimierte Ausgangs- und Übergangsfunktion bestimmen mit KV-Diagrammen bestimmen.
7. Unbenutzte Zustände überprüfen.
8. Schaltplan anhand Schaltfunktion konstruieren.

Umwandlung Mealy ⇔ Moore

Moore → Mealy

1. Ausgänge von Folgezuständen auf Kanten schreiben.
2. Ausgänge bei Zuständen entfernen.

Mealy → Moore

1. Ausgänge in Knoten schreiben, an denen Kante endet.
2. Knoten mit mehr als einem Ausgang multiplizieren → neu kodieren.
3. Eingehende Kanten entsprechend der Ausgänge auf neue Knoten umhängen.
4. Ausgehende Kanten für alle neue Knoten kopieren.

Diese Umwandlung ist immer möglich, aber meistens werden mehr Zustände benötigt.

Wichtig: Das Zeitverhalten der Ausgänge verändert sich bei der Umwandlung.

Mealy	Eingangsveränderungen beeinflussen den Ausgang sofort.
Moore	Eingangsveränderungen haben erst bei Taktflanke Einfluss (weniger Störungsanfällig)

Asynchronzähler

Dualzähler	Kaskadierung von T-FlipFlops
Vorwärtszähler	negativ flankengesteuerte FlipFlops
Rückwärtszähler	$\overline{Q_i}$ benutzen oder positive flankengesteuerte FlipFlops.

- Anzahl Bits = Anzahl T-FlipFlop
- LSB nach 1. FlipFlop, MSB ganz rechts

Probleme von Asynchronzählern

- Verzögerungen der Zustandsänderungen kumulieren sich entlang der Schaltung.
- Zeitverzögerung ist bei jedem Zustand anders.

Damit jeder mögliche Zustand bei n FlipFlops (kurz) auftritt:

$$f_{max} = \frac{1}{\sum_{i=1}^{n} t_{pd,i}}$$

Modulo-n Zähler

Zählt bis zu einem bestimmten Zustand und springt dann auf einen definierten Zustand zurück. Es werden n Zustände durchlaufen.
Kombinatorische Schaltung (AND-Gates) registrieren den Endzustand und setzen den definierten Zustand mittels der asynchronen Set- und Reset-Eingänge.

Def. Z (Bit)	0	R	mit komb. Schalt. verbinden
	1	S	

Anderer asynchroner Eingang an GND.

Synchronzähler

Alle FlipFlops haben das selbe Taktsignal.
Meistens **Medwedjew**-Automaten.

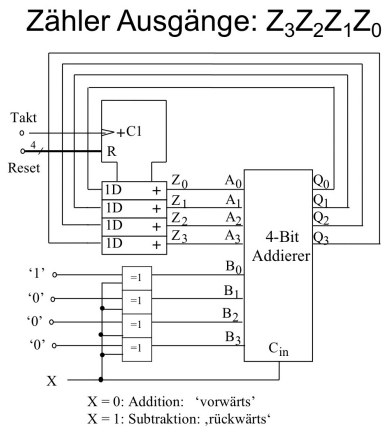
Entwurf

1. Zustandsgraph zeichnen.
2. Folgezustandstabelle aufstellen.
3. Für alle Folgezustände KV-Diagramme erstellen → Gleichung Folgezustand.
4. Zeichnen (Ausgänge = interne Zustände)

Vorwärts-Rückwärtszähler

Zusätzlicher Eingang bestimmt Zählrichtung → wie Synchronzähler entwerfen.

Alternative

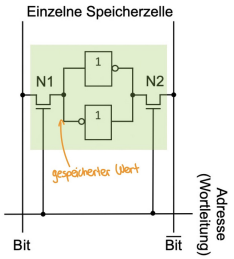


D-FlipFlops mit Addierer kombinieren.

X	$B_3B_2B_1B_0$			
0	0001	→	+1	addiert
1	1111	→	−1	

Schieberegister

SRAM



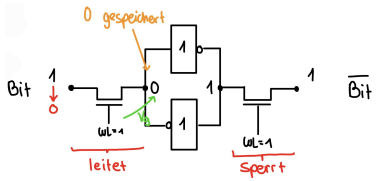
Wortleitung: Anwählen Speicherzelle
Bitleitung: Speicherinhalt lesen oder setzten

Bit	$\overline{Bit}$	
1	0	1 schreiben
0	1	0 schreiben
1	1	lesen

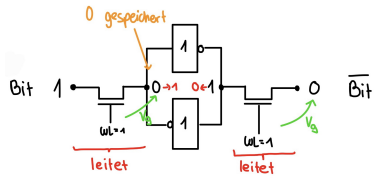
Wortleitung bei allen 1.

- Gespeicherte Wert steht immer auf linker Seite (Bit)
- Beim Lesen gibt Bit den Wert zurück; $\overline{Bit}$ den Invertierten.
- Beim Schreiben muss Bit auf den gewünschten Wert und $\overline{Bit}$ auf den Invertierten gesetzt werden.

Lesen

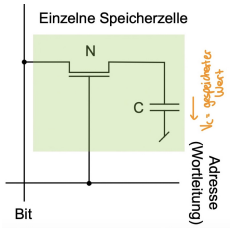


1 Schreiben



0 Schreiben Gleich wie Schreiben einer 1, aber Bit = 0.

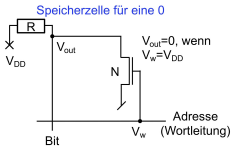
DRAM

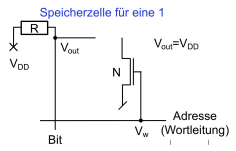


Wortleitung: Anwählen Speicherzelle
Bitleitung: Speicherinhalt lesen oder setzten
Schreiben: Bitleitung wird auf gewünschten Wert gesetzt → Kondensator: lädt, entlädt oder bleibt gleich.
Lesen: Parasitäre Kapazität wird ausgenutzt, um aus Veränderung von V_{out} gespeicherten Wert zu ermitteln.

ROM

Read-Only-Memory wird zur Herstellungszeit als 0 oder 1 programmiert.





Diverses

Physikalische Zuordnung logischer Zustände

0	Low 0 V	Ground
1	High 0.8 V	VDD

Toleranzen:
- GND: 0 V. . . 0.15 V
- VDD 0.7 V. . . 0.9 V

Schaltelemente

Multiplexer

Sendet eines von 2^n Eingangssignalen an den Ausgang. Hat n Auswahlbits.

Demultiplexer

Sendet 1 Eingangssignal an einen von 2^n Ausgänge. n Auswahlbits.

Halbaddierer

Addiert 2 Binärzahlen A und B. Produziert Summe und Carry-Out.

$$\text{SUM} = A \oplus B \qquad \text{CO} = A \wedge B$$

Volladdierer

Nimmt einen zusätzlichen Input CI entgegen.

$$\text{SUM} = (A \oplus B) \oplus CI \qquad \text{CO} = (A \wedge B) \vee (S_{AB} \wedge CI)$$

Serienaddierer

Addition einer Stelle pro Taktschritt.

Paralleladdierer (Normalform)

Addition aller Stellen pro Taktschritt.

Vorteile
- Maximal 3 Grundgatter zwischen Input und Output.
- Laufzeit ist unabhängig von Stellenzahl der Summanden.

Nachteile Bei Addition von n-stelligen Summanden müssen $\sim n \cdot 2^{2n-1}$ Min-/Maxterme verknüpft werden.

→ Schnell aber Schaltungsaufwendig

Ripple-Carry Addierer (Paralleladdierer)

Vorteile

- Durch Kaskadierung einfach skalierbar.
- Schaltungsaufwand linear zur Stellenzahl.

Nachteile

- SUM und CO für die i-te Stelle können erst nach der Berechnung der $(i-1)$-ten Stelle gebildet werden.
- Addierzeit linear zu Stellenzahl

Langsamer als Normalformaddierer aber einfacher zu realisieren.

Carry-Look-Ahead Addierer (Paralleladdierer)

Kombination der Vorteile des Normalform- und Ripple-Carry-Addierer → schnelle Schaltung mit begrenztem Aufwand.

Praktische Realisierung Addierer werden kaskadiert, Berechnung der Überträge erfolgt parallel zur Summenbildung.
Berechnungsaufwand ist linear zur Stellenzahl, Laufzeit bleibt konstant.

Booth-Algorithmus

Dient der Multiplikation von Binärzahlen (A & B).
Berechnung über Zwischenprodukte P_i.
Division durch 2 bedeutet: Verschiebung des Kommas nach links (shift), mit Vorzeichenverdoppelung falls nötig.

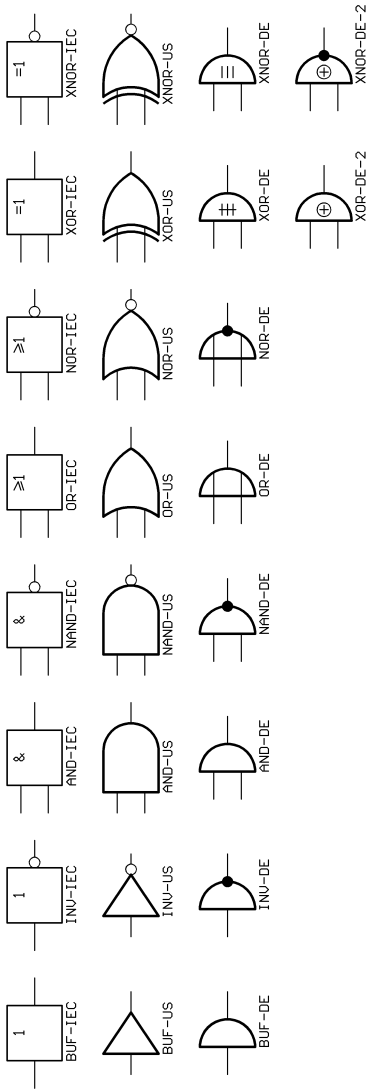
a_i	a_{i-1}	Operation
0	0	$P_i = P_{i-1}/2$
0	1	$P_i = (P_{i-1} + B)/2$
1	0	$P_i = (P_{i-1} - B)/2$
1	1	$P_i = P_{i-1}/2$

Anfangswerte: $P_{-1} = 0$, $a_{-1} = 0$
Beim letzten Schritt entfällt die Division durch 2.

Zahlencodes

Binär	BCD	Excess-3	Aiken	4-2-2-1	Gray	O'Brien
0000	0		0	0	0	
0001	1		1	1	1	
0010	2		2	2	3	0
0011	3	0	3	3	2	
0100	4	1	4		7	4
0101	5	2			6	3
0110	6	3		4	4	1
0111	7	4		5	5	2
1000	8	5				
1001	9	6				
1010		7				9
1011		8	5			
1100		9	6	6	8	5
1101			7	7	9	6
1110			8	8		7
1111			9	9		8

Gate Varianten



Zehnerpotenzen

Potenz	Präfix	Symbol
10^{-15}	Femto	f
10^{-12}	Piko	p
10^{-9}	Nano	n
10^{-6}	Mikro	μ
10^{-3}	Milli	m
10^{-2}	Zenti	c
10^{-1}	Dezi	d
10^{1}	Deka	da
10^{2}	Hekto	h
10^{3}	Kilo	k
10^{6}	Mega	M
10^{9}	Giga	G
10^{12}	Tera	T
10^{15}	Peta	P

0	0	0000 0000	64	64	0100 0000	128	128	1000 0000	192	−64	1100 0000
1	1	0000 0001	65	65	0100 0001	129	−127	1000 0001	193	−63	1100 0001
2	2	0000 0010	66	66	0100 0010	130	−126	1000 0010	194	−62	1100 0010
3	3	0000 0011	67	67	0100 0011	131	−125	1000 0011	195	−61	1100 0011
4	4	0000 0100	68	68	0100 0100	132	−124	1000 0100	196	−60	1100 0100
5	5	0000 0101	69	69	0100 0101	133	−123	1000 0101	197	−59	1100 0101
6	6	0000 0110	70	70	0100 0110	134	−122	1000 0110	198	−58	1100 0110
7	7	0000 0111	71	71	0100 0111	135	−121	1000 0111	199	−57	1100 0111
8	8	0000 1000	72	72	0100 1000	136	−120	1000 1000	200	−56	1100 1000
9	9	0000 1001	73	73	0100 1001	137	−119	1000 1001	201	−55	1100 1001
10	10	0000 1010	74	74	0100 1010	138	−118	1000 1010	202	−54	1100 1010
11	11	0000 1011	75	75	0100 1011	139	−117	1000 1011	203	−53	1100 1011
12	12	0000 1100	76	76	0100 1100	140	−116	1000 1100	204	−52	1100 1100
13	13	0000 1101	77	77	0100 1101	141	−115	1000 1101	205	−51	1100 1101
14	14	0000 1110	78	78	0100 1110	142	−114	1000 1110	206	−50	1100 1110
15	15	0000 1111	79	79	0100 1111	143	−113	1000 1111	207	−49	1100 1111
16	16	0001 0000	80	80	0101 0000	144	−112	1001 0000	208	−48	1101 0000
17	17	0001 0001	81	81	0101 0001	145	−111	1001 0001	209	−47	1101 0001
18	18	0001 0010	82	82	0101 0010	146	−110	1001 0010	210	−46	1101 0010
19	19	0001 0011	83	83	0101 0011	147	−109	1001 0011	211	−45	1101 0011
20	20	0001 0100	84	84	0101 0100	148	−108	1001 0100	212	−44	1101 0100
21	21	0001 0101	85	85	0101 0101	149	−107	1001 0101	213	−43	1101 0101
22	22	0001 0110	86	86	0101 0110	150	−106	1001 0110	214	−42	1101 0110
23	23	0001 0111	87	87	0101 0111	151	−105	1001 0111	215	−41	1101 0111
24	24	0001 1000	88	88	0101 1000	152	−104	1001 1000	216	−40	1101 1000
25	25	0001 1001	89	89	0101 1001	153	−103	1001 1001	217	−39	1101 1001
26	26	0001 1010	90	90	0101 1010	154	−102	1001 1010	218	−38	1101 1010
27	27	0001 1011	91	91	0101 1011	155	−101	1001 1011	219	−37	1101 1011
28	28	0001 1100	92	92	0101 1100	156	−100	1001 1100	220	−36	1101 1100
29	29	0001 1101	93	93	0101 1101	157	−99	1001 1101	221	−35	1101 1101
30	30	0001 1110	94	94	0101 1110	158	−98	1001 1110	222	−34	1101 1110
31	31	0001 1111	95	95	0101 1111	159	−97	1001 1111	223	−33	1101 1111
32	32	0010 0000	96	96	0110 0000	160	−96	1010 0000	224	−32	1110 0000
33	33	0010 0001	97	97	0110 0001	161	−95	1010 0001	225	−31	1110 0001
34	34	0010 0010	98	98	0110 0010	162	−94	1010 0010	226	−30	1110 0010
35	35	0010 0011	99	99	0110 0011	163	−93	1010 0011	227	−29	1110 0011
36	36	0010 0100	100	100	0110 0100	164	−92	1010 0100	228	−28	1110 0100
37	37	0010 0101	101	101	0110 0101	165	−91	1010 0101	229	−27	1110 0101
38	38	0010 0110	102	102	0110 0110	166	−90	1010 0110	230	−26	1110 0110
39	39	0010 0111	103	103	0110 0111	167	−89	1010 0111	231	−25	1110 0111
40	40	0010 1000	104	104	0110 1000	168	−88	1010 1000	232	−24	1110 1000
41	41	0010 1001	105	105	0110 1001	169	−87	1010 1001	233	−23	1110 1001
42	42	0010 1010	106	106	0110 1010	170	−86	1010 1010	234	−22	1110 1010
43	43	0010 1011	107	107	0110 1011	171	−85	1010 1011	235	−21	1110 1011
44	44	0010 1100	108	108	0110 1100	172	−84	1010 1100	236	−20	1110 1100
45	45	0010 1101	109	109	0110 1101	173	−83	1010 1101	237	−19	1110 1101
46	46	0010 1110	110	110	0110 1110	174	−82	1010 1110	238	−18	1110 1110
47	47	0010 1111	111	111	0110 1111	175	−81	1010 1111	239	−17	1110 1111
48	48	0011 0000	112	112	0111 0000	176	−80	1011 0000	240	−16	1111 0000
49	49	0011 0001	113	113	0111 0001	177	−79	1011 0001	241	−15	1111 0001
50	50	0011 0010	114	114	0111 0010	178	−78	1011 0010	242	−14	1111 0010
51	51	0011 0011	115	115	0111 0011	179	−77	1011 0011	243	−13	1111 0011
52	52	0011 0100	116	116	0111 0100	180	−76	1011 0100	244	−12	1111 0100
53	53	0011 0101	117	117	0111 0101	181	−75	1011 0101	245	−11	1111 0101
54	54	0011 0110	118	118	0111 0110	182	−74	1011 0110	246	−10	1111 0110
55	55	0011 0111	119	119	0111 0111	183	−73	1011 0111	247	−9	1111 0111
56	56	0011 1000	120	120	0111 1000	184	−72	1011 1000	248	−8	1111 1000
57	57	0011 1001	121	121	0111 1001	185	−71	1011 1001	249	−7	1111 1001
58	58	0011 1010	122	122	0111 1010	186	−70	1011 1010	250	−6	1111 1010
59	59	0011 1011	123	123	0111 1011	187	−69	1011 1011	251	−5	1111 1011
60	60	0011 1100	124	124	0111 1100	188	−68	1011 1100	252	−4	1111 1100
61	61	0011 1101	125	125	0111 1101	189	−67	1011 1101	253	−3	1111 1101
62	62	0011 1110	126	126	0111 1110	190	−66	1011 1110	254	−2	1111 1110
63	63	0011 1111	127	127	0111 1111	191	−65	1011 1111	255	−1	1111 1111

2^0	1.0	2^{-0}	1.0
2^1	2.0	2^{-1}	0.5
2^2	4.0	2^{-2}	0.25
2^3	8.0	2^{-3}	0.125
2^4	16.0	2^{-4}	0.0625
2^5	32.0	2^{-5}	0.03125
2^6	64.0	2^{-6}	0.015625
2^7	128.0	2^{-7}	0.0078125
2^8	256.0	2^{-8}	0.00390625
2^9	512.0	2^{-9}	0.001953125
2^{10}	1024.0	2^{-10}	9.765625^{-4}
2^{11}	2048.0	2^{-11}	4.8828125^{-4}
2^{12}	4096.0	2^{-12}	2.44140625^{-4}