Digitaltechnik

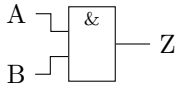
von Jirayu Ruh, jirruh@ethz.ch

Gatter

AND- und OR-Gatter

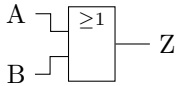
AND

$Z = A \wedge B$



OR

$Z = A \vee B$

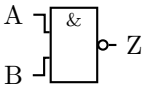


A	B	AND	OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

NAND- und NOR-Gatter

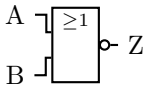
NAND

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



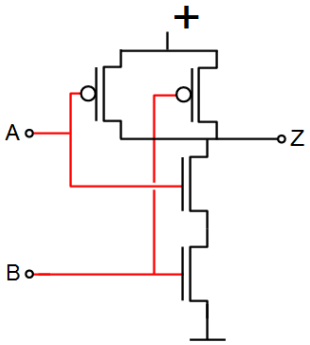
NOR

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

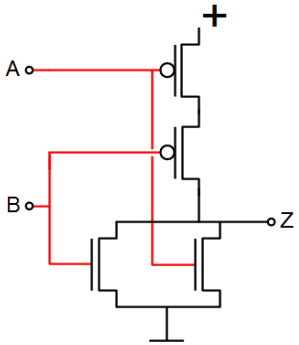


A	B	NAND	NOR
0	0	1	1
0	1	1	0
1	0	1	0
1	1	0	0

NAND

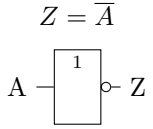


NOR

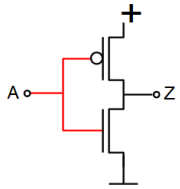


NOT

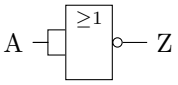
$Z = \overline{A}$



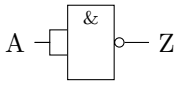
A	NOT
0	1
1	0



NOT aus NOR



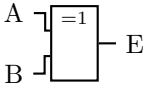
NOT aus NAND



XOR und XNOR

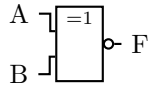
XOR

$$Z = A \oplus B = (A \wedge \overline{B}) \vee (\overline{A} \wedge B)$$



XNOR

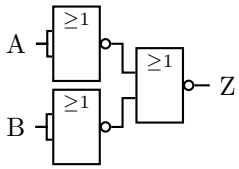
$$Z = \overline{A \oplus B} = (A \wedge B) \vee (\overline{A} \wedge \overline{B})$$



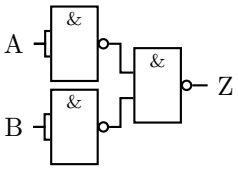
A	B	XOR	XNOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Gatter aus NAND- und NOR-Gatter

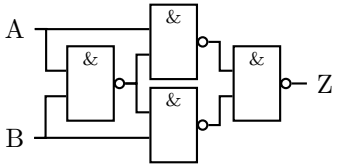
AND-Gatter aus NOR-Gatter



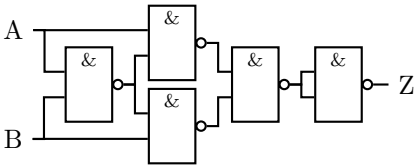
OR-Gatter aus NAND-Gatter



XOR Gatter



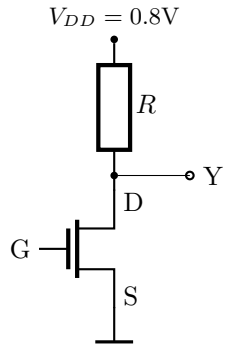
XNOR Gatter



CMOS Gatter

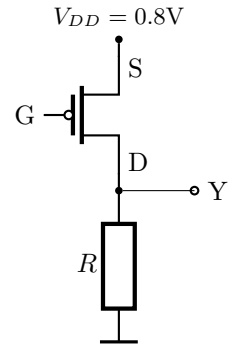
High Pegel "H": 0.9-0.7V Low Pegel "L": 0.15-0V

NMOS



G	Schalter	Y
0	offen	1
1	zu	0

PMOS



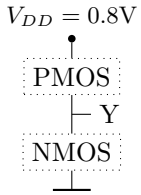
G	Schalter	Y
0	zu	1
1	offen	0

Zustand NN: Potential an Source unbestimmt, "free floating"

Konstruktion von CMOS-Gatter

CMOS-Gatter benötigen pro Eingang 1 NMOS + 1 PMOS.

Sie bestehen aus zwei ergänzenden Schaltungsteilen:

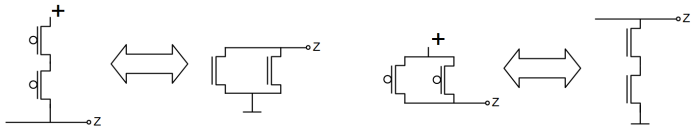


Pull-up Schaltung: PMOS
Pull-down Schaltung: NMOS

Funktionsgleichung CMOS-Gatter

Pull-Up: $Y_{pu} = 1$ Eingänge Invertiert
Pull-Down: $Y_{pd} = 0$ Eingänge nicht Invertiert

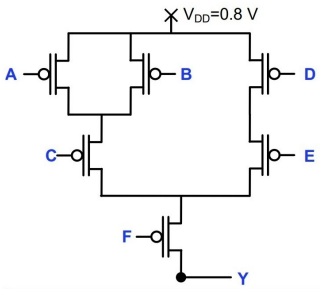
$$Y_{pu} = \overbrace{(\overline{A} \wedge \overline{B})}^{\text{Seriell}} \vee \overbrace{C}^{\text{Paralell}} \quad \Leftrightarrow \quad Y_{pd} = \overbrace{(A \vee B)}^{\text{Paralell}} \wedge \overbrace{C}^{\text{Seriell}}$$



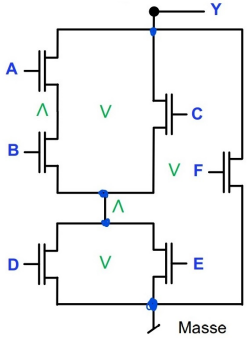
Umwandlung zwischen Pull-up und Pull-down

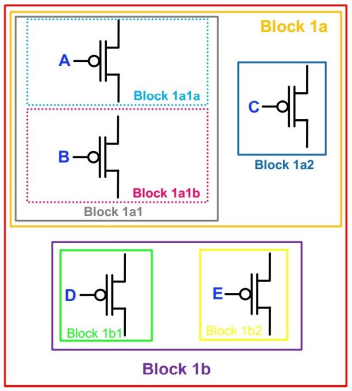
1. Schaltung in Parallele und Serielle Blöcke zerlegen
2. Umwandeln: Parallele $\rightleftharpoons$ Serielle Blöcke
3. 1),2) wiederholen bis einzelnen Transistoren übrig sind
4. Umwandeln: PMOS $\rightleftharpoons$ NMOS
5. Schaltungstyp entsprechend Y, GND bzw. VDD setzen

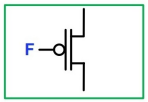
Pull-up Pfad einer CMOS Schaltung



Pull-down Pfad der CMOS Schaltung

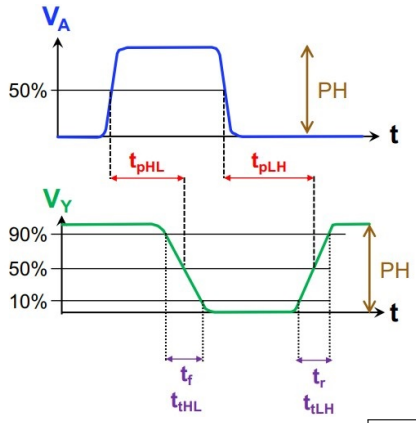


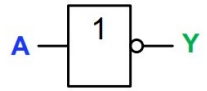




Block 1

Zeitverhalten CMOS Gatter





Zeitlaufdiagram eines CMOS Inverters im dynamischen (zeitabhängigen) Betrieb

PH: Pegelhub

Durchschnittliche Verzögerung: $t_d = \frac{t_{pHL} + t_{pLH}}{2}$

Boolsche Algebra

Es gelten folgende Grundgesetze, wie in der Algebra:

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

Aus diesem Grund gilt folgende für die Logikminimierung sehr nützliche Umwandlung:

$$\wedge \rightleftharpoons \cdot \qquad \vee \rightleftharpoons +$$

Zusätzlich zu den Grundregeln gelten folgende Regeln:

Nicht	$\overline{\overline{A}} = A$	
Null-Theorem	$A \vee 0 = A$	$A \wedge 0 = 0$
Eins-Theorem	$A \vee 1 = 1$	$A \wedge 1 = A$
Idempotenz	$A \vee A = A$	$A \wedge A = A$
Verknüpfung mit Komplement	$A \vee \overline{A} = 1$	$A \wedge \overline{A} = 0$
Adsorptionsgesetze	$A \vee (\overline{A} \wedge B) = A \vee B$	
	$A \wedge (\overline{A} \vee B) = A \wedge B$	
Absorptionsgesetze	$A \vee (A \wedge B) = A$	
	$A \wedge (A \vee B) = A$	
Nachbarschafts Gesetze	$(A \wedge B) \vee (\overline{A} \wedge B) = B$	
	$(A \vee B) \wedge (\overline{A} \vee B) = B$	

De Morgan'sche Gesetze

Beziehungen zwischen NAND/NOR und AND/OR:

Erstes Gesetz:	$\overline{A \wedge \cdots \wedge B} = \overline{A} \vee \cdots \vee \overline{B}$
Zweites Gesetz:	$\overline{A \vee \cdots \vee B} = \overline{A} \wedge \cdots \wedge \overline{B}$

Daraus folgen zwei sehr nützliche Umwandlungen:

$$A \wedge \cdots \wedge B = \overline{\overline{A} \vee \cdots \vee \overline{B}} \qquad A \vee \cdots \vee B = \overline{\overline{A} \wedge \cdots \wedge \overline{B}}$$

Normalformen

Der Minterm ist eine AND-Verknüpfung, welcher '1' ergibt für nur eine Kombination der Schaltungsvariablen.

Der Maxterm ist eine OR-Verknüpfung, welcher '0' ergibt für nur eine Kombination der Schaltungsvariablen. Bei der Bildung der Maxterme müssen die Variablen invertiert werden!

A	B	Z	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	$A \wedge B$	

Disjunktive Normalform DNF (ODER-Normalform)

Besteht aus einer ODER-Verknüpfung aller Minterme:

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

Konjunktive Normalform KNF (UND-Normalform)

Besteht aus einer UND-Verknüpfung aller Maxterme:

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

Kanonische Normalform

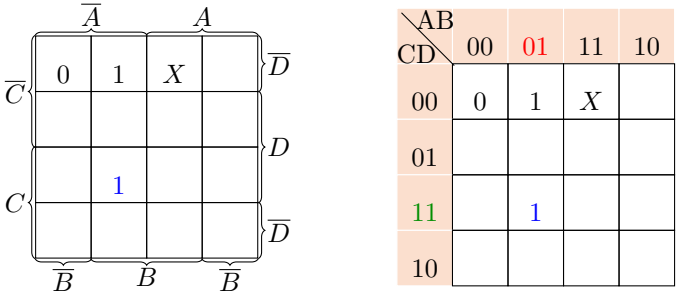
Die kanonische Normalform ist die unvereinfachte Normalform einer Wahrheitstabelle. Sie gibt also nicht notwendigerweise die einfachsten Funktionsgleichungen an.

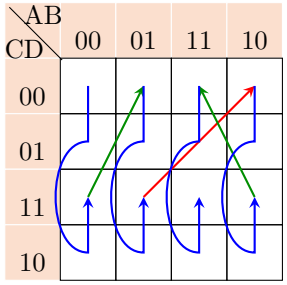
Karnaugh Diagramme (KVD)

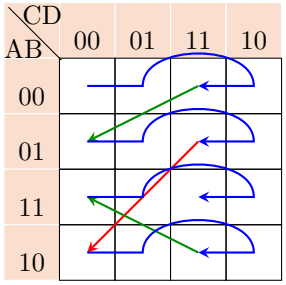
KV-Diagramme sind eine äquivalente Darstellungsform von Schaltfunktionen in Matrizenform. KVD ermöglichen die unkomplizierte Bildung der vereinfachten DNF/KNF.

Schema zum ausfüllen eines 4 Variablen KV-Diagramm:

A	B	C	D	f(A,B,C,D)
	⋮			⋮
0	1	1	1	f_{0111}
	⋮			⋮



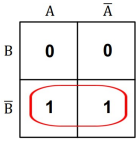




Bei 5 Schaltvariablen wird die 5te Schaltvariable mit einer ABCD Matrix für E und einer für $\overline{E}$ dargestellt (2D!).

Päckchen

Orthogonal benachbarte Minterme (bzw. Maxterme) werden zu einem "Päckchen" zusammengefasst.



Implizite Anwendung des Nachbarschaftsgesetz:

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

Es gelten folgende Regeln für Päckchen:

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

Don't Care Zustände

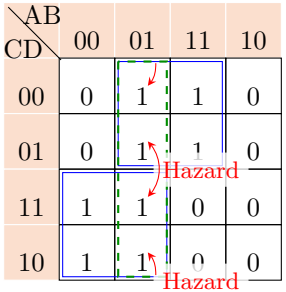
Redundante oder unmögliche Kombinationen der Eingangsvariablen werden mit einem 'X' im KVD markiert. Man darf solche Zustände benutzen um Päckchen zu bilden!

Hazard

Hazards sind kurzzeitige, unerwünschte Änderung der Signalwerte, die durch Zeitverzögerung der Gatter entstehen.

Strukturhazard (Statischer Hazard)

Statische Hazards sind im KVD Stellen, an denen sich Päckchen orthogonal berühren, aber nicht überlappen.



Lösung: Berührendes Päckchen mit zusätzlichem Päckchen (möglichst gross) verbinden.

Funktionale Hazards

Wenn sich Päckchen diagonal berühren, dann wechseln mindestens zwei Variablen. Dies ist ein funktionaler Hazard und kann nicht leicht behoben werden.

Zahlensysteme

Polyadische Systeme

Die Position einer Ziffer innerhalb einer Zahl gibt den Wert an, mit der die Basis des Zahlensystems an dieser Stelle potenziert wird. Umwandlung einer Zahl $D_{(R)}$ ins Dezimalsystem:

R Basis/Radix von D_R
b_i Koeffizienten ("Ziffern")

$$D_{(10)} = \sum_{i=-\infty}^{\infty} b_i \cdot R^i$$

Darstellung von $D_{(R)}$ 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 (0x)	16	$b_i \in \{0, 1, \ldots, 9, A, B, C, D, E, F\}$

Umwandlung Dezimal → Zahlensystem R

Für den Teil vor dem Komma:
Wiederholte ganzzahlige Division durch Basis R mit Rest r, der jeweilige Rest r entspricht der Ziffer im neuen Zahlensystem, beginnend mit dem **L**east **S**ignificant **D**igit (LSD).

$$\frac{D_{(10)}}{R} = Q_0 + r_0 \text{ und dann Rekursiv: } \frac{Q_i}{R} = Q_{i+1} + r_{i+1}$$

Für den 'Fraktionalteil' a_o hinter dem Komma:
Wiederholte Multiplikation von a_0 mit der Zahlenbasis R und ausführen der Abrundungsfunktion floor(x) zur Berechnung von K, welches der Ziffer im neuen Zahlensystem entspricht, beginnend mit dem **M**ost **S**ignificant **D**igit (MSD).

$$a_i \cdot R = P_i \quad \rightarrow \text{floor}(P_i) = K_{i-1},\ a_{i-1} = P_i - K_{i-1}$$

Ein Zahlenbeispiel:

$$(123)_{10} = (4 \cdot 5^2 + 4 \cdot 5^1 + 3 \cdot 5^0)_{10} = (443)_5$$
$$5 \cdot 0.614 = 3.07 \rightarrow \text{Ziffer3}$$
$$5 \cdot 0.07 = 0.35 \rightarrow \text{Ziffer0}$$
$$5 \cdot 0.35 = 1.75 \rightarrow \text{Ziffer1}$$
$$5 \cdot 0.75 = 3.75 \rightarrow \text{Ziffer3}$$
$$5 \cdot 0.75 = 3.75 \rightarrow \text{Ziffer3 } usw.$$
$$\Rightarrow \underline{\underline{(123.614)_{10} = (443.30\overline{13})_5}}$$

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

Zweierkomplement

Für negative Dualzahlen wird das Zweierkomplement verwendet. Das MSB (sign Bit) hat eine negative Wertigkeit.

Bildung Zweierkomplement für eine negative Zahl:

i) Absolutbetrag der Zahl in Dualzahl umwandeln
ii) Dualzahl bitweise invertieren und beim LSB '+1'
iii) Zuvorderst das sign Bit hinzufügen

Darstellung Rationaler Zahlen als Dualzahl (Q-Format)

Vorzeichenbehaftete Zahlen mit rationalem Anteil werden im Q-Format dargestellt. Bei m Vorkomma- und n Nachkommabits sind insgesamt $k = 1 + m + n$ Binärstellen erforderlich.

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

Binäre Rechenoperationen

Addition

Schriftliche bitweise Addition mit Übertrag.

	1	1	1		1	1					← Überträge
		0	1	1	0	0	1.	0	1	1	$25.375_{(10)}$
+		1	1	1	0	1	1.	1	0	0	$59.5_{(10)}$
=	1	0	1	0	1	0	0.	1	1	1	$84.875_{(10)}$

Subtraktion

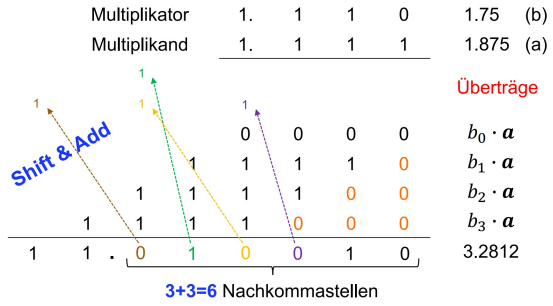
Addition, aber in 2er-Komplement Darstellung.

	1	1	1			1	1	Übertrag
		0	1	1	1	0	1	1	$59_{(10)}$
+		1	1	1	0	0	0	1	$-15_{(10)}$
=		0	1	0	1	1	0	0	$44_{(10)}$

Minuend und Subtrahend müssen über gleiche Stellenanzahl verfügen, sonst kürzere Zahl linksbündig mit Vorzeichenbit erweitern. Überträge nach dem Vorzeichenbit ignorieren!

Multiplikation

Multiplikation zweier vorzeichenloser Dualzahlen:



Codes

Tetraden-Codes

- **BCD**: Häufig benutzt, keine Rundungsfehler!
- **Excess-3 und Aiken**: Ziffern liegen symmetrisch im Binärfeld, günstige Verteilung für dezimale Rechenwerke
- **4-2-2-1**: Interessante Gewichtung für A/D Wandler
- **Gray und O'Brien**: Einschrittige Codes (Schwächere Auswirkung von Übertragungsfehlern), keine Fehlinformation bei Übergängen (Winkelkodierung)

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

Parity-Bits

Redundante Kodierung, welche Übertragungsfehler erkennen kann, solange höchstens ein Fehler pro Bitgruppe geschieht.

Bitgruppe wird durch ein Parity-Bit ergänzt, welches die Bitgruppe auf geradzähligkeit (P_E) oder ungeradzähligkeit (P_O) überprüft. Der Datenempfänger kann so die Richtigkeit der Datenübertragung überprüfen.

Korrekt mit 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 mit P_E

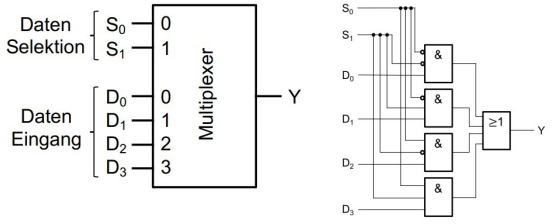
0	1	0	1	0
1	1	1	1	1
1	0	1	1	1
0	0	1	0	1
0	0	0	1	1

Für eine Fehlerkorrektur muss zusätzlich ein Prüfwort übertragen werden, welches Spaltenweise ein Parity-Bit bildet. Der Empfänger kann so das fehlerbehaftete Bit in der Matrixdarstellung (siehe Bsp.) erkennen und korrigieren.

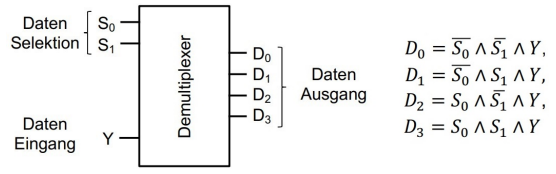
Einfache Hardwarekomponente

Multiplexer und Demultiplexer

Multiplexer ermöglichen das durch Steuersignale gewählte Aufschalten eines Eingangssignal aus mehreren möglichen:



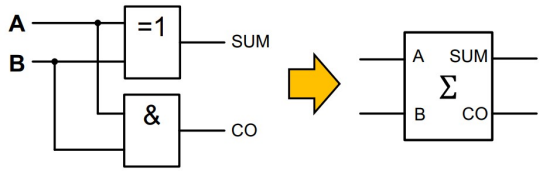
Demultiplexer nehmen Daten aus einem einzigen Kanal und verteilen es auf einen beliebigen Ausgang.



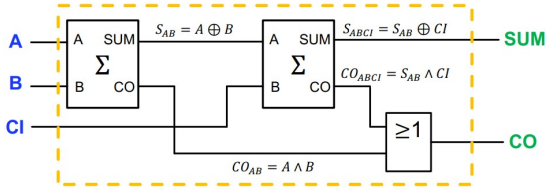
$$D_0 = \overline{S_0} \wedge \overline{S_1} \wedge Y,$$
$$D_1 = \overline{S_0} \wedge S_1 \wedge Y,$$
$$D_2 = S_0 \wedge \overline{S_1} \wedge Y,$$
$$D_3 = S_0 \wedge S_1 \wedge Y$$

Halb- und Volladdierer

Halbaddierer sind Rechenschaltungen, die zwei Dualzahlen addieren. Ausgänge: SUM (Summe), CO (Carry, Übertrag)



Volladdierer haben einen zusätzlichen Eingang CI (Carry in), dieser ermöglicht das Bilden von Mehrbit-Addierer.



Mehrbit-Addierer (Paralleladdierer)

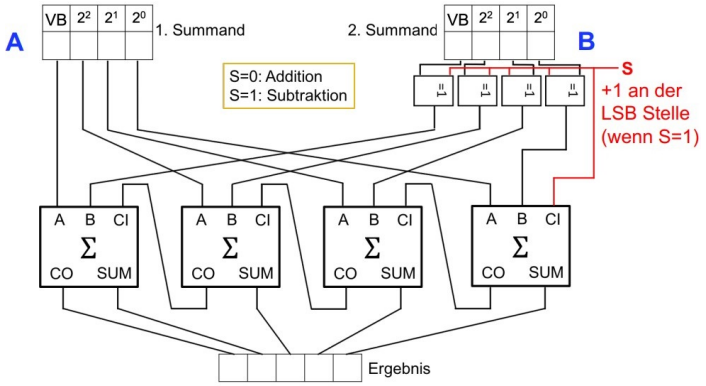
Ein Paralleladdierer in Normalform ist sehr aufwendig zu realisieren, da $\sim n \cdot 2^{2n-1}$ Min-/Maxterme verknüpft werden müssen. Vorteil: Laufzeit unabhängig von Stellenanzahl

Ein Ripple-Carry Addierer ist eine Kaskadierung von Volladdierer. Einfach skalierbar, leidet aber am 'ripple' Effekt, d.h. Laufzeiten addieren sich auf.

Der Carry-Look-Ahead Addierer kombiniert die Vorteil der beiden, d.h. man kaskadiert die Addierer, aber berechnet die Überträge parallel zur Summenbildung. (Berechnungsaufwand linear zur Stellenanzahl, aber Laufzeit konstant)

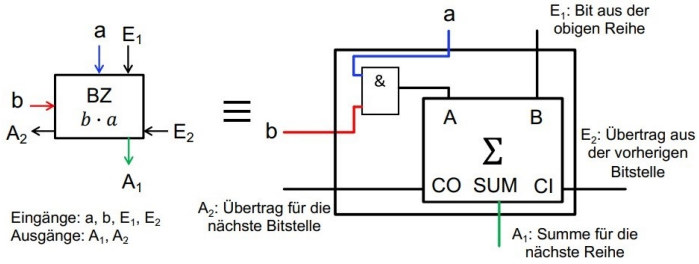
Ripple-Carry Addierer mit Subtraktion

Die Subtraktion erfolgt durch Bildung des 2er-Komplement:



Hardware Multiplizierer

Folgt dem Prinzip der Bitweisen Multiplikation. Besteht aus folgender Basiszelle:



Die Multiplikation mit negativen Zahlen im 2er-Komplement ist eher schwierig. Eine Möglichkeit ist der iterative Booth Algorithmus.

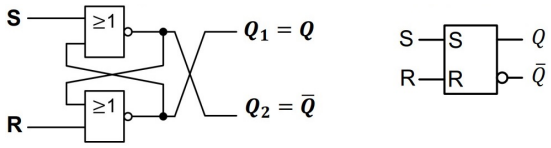
Latches und Flipflops

Sequentielle Schaltungen welche Rückkopplungen enthalten.

Zustandgesteurte Latches

Zustandsgesteuert: Das Verhalten eines Latches hängt nicht nur von den aktuellen Eingangsvariablen ab, sondern auch von den intern gespeicherten Zuständen.

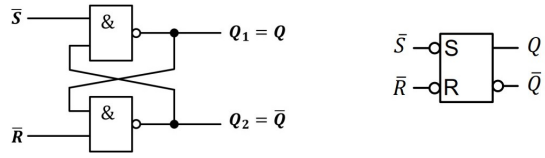
SR-Latch



$$Q_{n+1} = S \vee (Q_n \wedge \overline{R}) \quad \text{Bedingung: } R \wedge S = 0$$

S	R	Q_{n+1}	
0	0	Q_n	speichern
0	1	0	reset
1	0	1	set
1	1	-	unzulässig

$\overline{SR}$-Latch



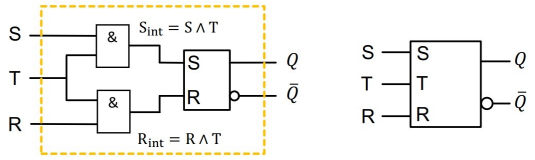
$\overline{S}$	$\overline{R}$	Q_{n+1}	
0	0	-	unzulässig
0	1	1	set
1	0	0	reset
1	1	Q_n	speichern

Taktzustandgesteuerte Latches

Taktzustandgesteuert: Änderungen am Eingang werden nur wahrgenommen, wenn das Taktsignal $T = 1$ ist.

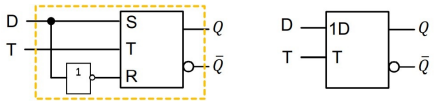
Taktzustandgesteuerte Latches sind gegenüber Störimpulsen empfindlich, da bei $T = 1$ jede Änderung am Eingang übernommen wird.

SRT-Latch



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

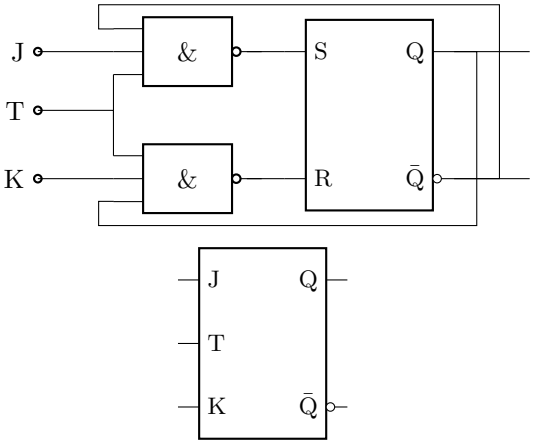
D-Latch



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

T	Q_{n+1}		
0	Q_n	$\rightarrow$	alter Ausgang gespeichert
1	D	$\rightarrow$	Input übernommen

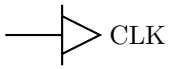
JK-Latch



T	J	K	Q_{n+1}	
0	X	X	Q_n	speichern
1	0	0	Q_n	speichern
1	0	1	0	reset
1	1	0	1	set
1	1	1	$\bar{Q_n}$	kippen

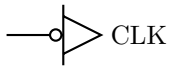
Flipflops

Taktflankensteuerung: Serieschaltung von zwei mit gegenphasigem Takt gesteuerten Latches(Master-Slave Aufbau).



Input beim Übergang von $0 \rightarrow 1$ von CLK wirksam.

Positive/steigende Taktflanke

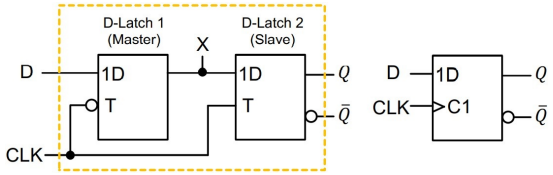


Input beim Übergang von $1 \rightarrow 0$ von CLK wirksam.

Negative/fallende Taktflanke

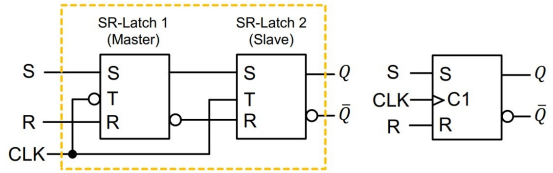
Vorteil: Sehr robust gegenüber Störimpulsen

D-Flipflop



$$Q_{n+1} = D_n \quad \text{Wenn CLK } 0 \rightarrow 1$$

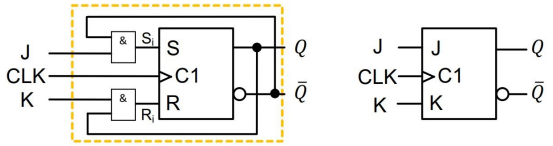
SR-Flipflop



$$Q_{n+1} = [S \vee (Q_1 \wedge \overline{R})]_n \quad \text{Reqs: } R \wedge S = 0, \text{CLK } 0 \rightarrow 1$$

JK-Flipflop

Beim JK-Flipflop gibt es keinen unzulässigen Zustand mehr, dieser wurde durch eine Toggle Funktionalität ersetzt.



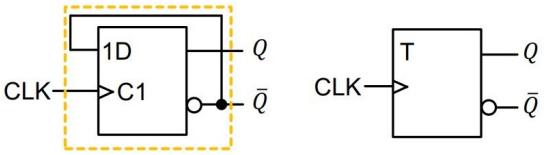
$$Q_{n+1} = (J \wedge \overline{Q_n}) \vee (\overline{K} \wedge Q_n) \quad \text{Wenn CLK } 0 \rightarrow 1$$

J	K	Q_{n+1}	$\overline{Q}_{n+1}$	
0	0	Q_n	$\overline{Q}_n$	speichern
0	1	0	1	reset
1	0	1	0	set
1	1	$\overline{Q}_n$	Q_n	toggle

Es gibt natürlich auch (takt)zustandsgesteuerte JK-Latches!

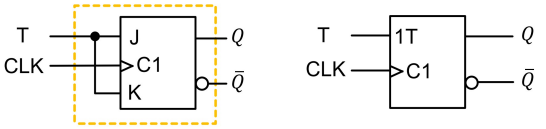
Toggle-Flipflop

Schaltung welche bei jeder aktiven Taktflanke kippt.



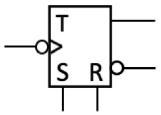
$$Q_{n+1} = \overline{Q_n} \quad \text{Wenn CLK } 0 \rightarrow 1$$

Folgende Schaltung kippt nur bei Taktflanken wenn T=1:



Asynchroner Set/Reset Input

Können gespeicherte Zustände asynchron zu CLK überschreiben, d.h. jederzeit auch ohne ein Taktflankensignal.

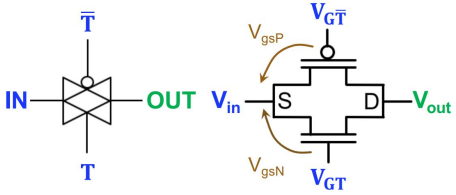


D-Flipflop in CMOS-Technik

In den meisten Anwendungen werden D-Flipflops verwendet, da sie mit CMOS Technik effizient realisierbar sind.

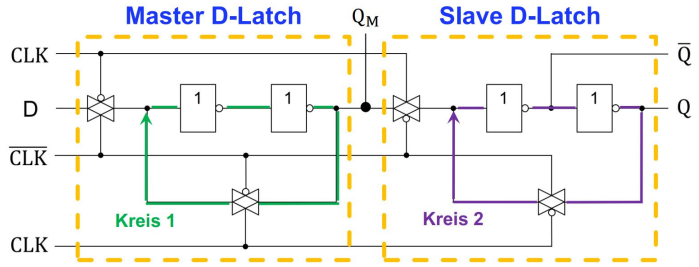
Transmission Gates (TG)

TGs bestehen aus 2 Transistoren, einem NMOS und einem PMOS.

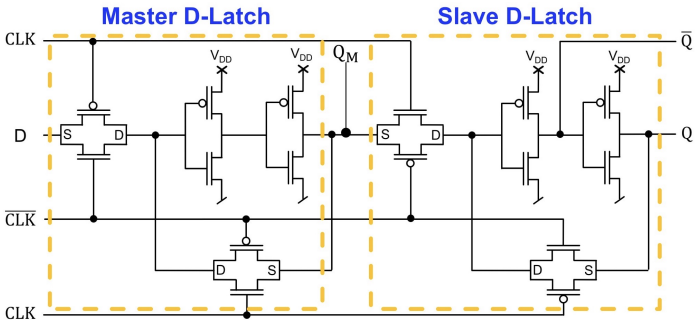


IN	T	Widerstand	OUT
0	0	hochohmig	-
0	1	niederohmig	0
1	0	hochohmig	-
1	1	niederohmig	1

Pro D-Latch sind 2 TG und 2 Inverter notwendig.



Insgesamt sind also 8NMOS und 8PMOS notwendig.

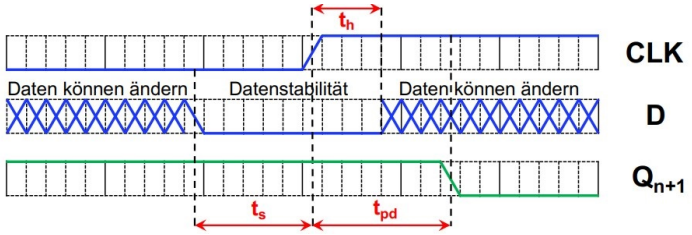


D-Flipflop ⇒ JK-Flipflop

Ein JK-FF kann nur mit einem D-FF realisiert werden, wenn:

$$D_n = (J_n \wedge \overline{Q}_n) \vee (\overline{K}_n \wedge Q_n)$$

Verzögerungszeiten

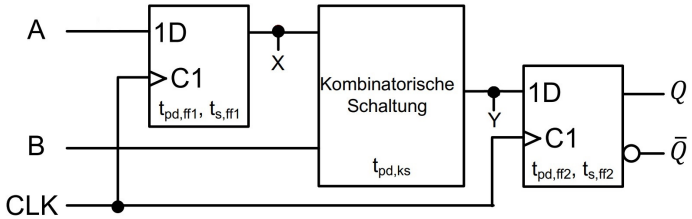


Setup-Zeit (t_s)	Solange muss Signal an FF vor aktiver Taktflanke stabil anliegen.
Hold-Zeit (t_h)	Solange muss Signal an FF nach aktiver Taktflanke stabil anliegen.
Verzögerungszeit (t_{pd})	Durchlaufzeit

Bei Verletzung der Zusatzbedingungen t_s, t_h kann der Zustand des FF unbestimmt oder metastabil werden.

Maximale Taktfrequenz bei seriellen FFs

In einem Schaltnetz mit mindestens zwei Flipflops in Serie (kann auch der gleiche FF sein), ist die maximale Taktfrequenz des Clocks begrenzt.



$$T_{\min} \geq t_{pd,ff1} + t_{pd,ks} + t_{s,ff2} \qquad f_{max} = \frac{1}{T_{min}}$$

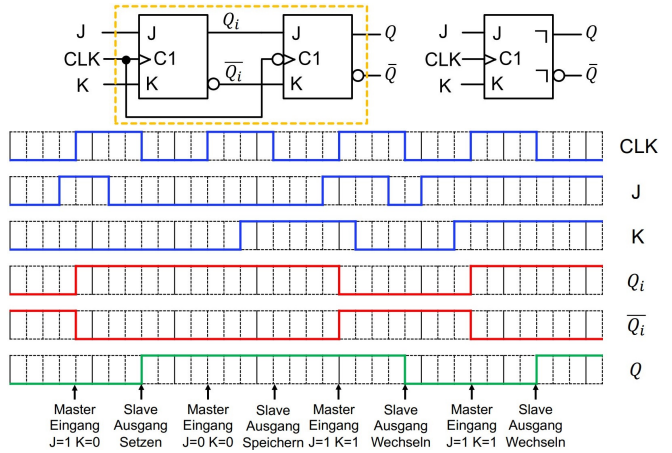
wobei $t_{pd,ks}$ Verzögerungszeit kombinatorische Schaltung

Bei komplizierten kombinatorischen Schaltungen begrenzt der Pfad mit der längsten Zeitverzögerung die Taktfrequenz.

Master-Slave Flipflops (Zwischenspeicher FFs)

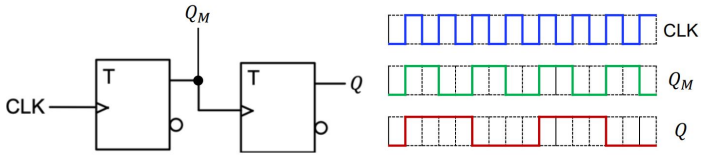
- Übernehmen Eingangssignal mit der steigenden (bzw. fallenden) Taktflanke.
- Geben das Ausgangssignal mit der nächsten fallenden (bzw. steigenden) Taktflanke aus.

JK-Master-Slave FF (mit SR-,D-,T-FF auch möglich):

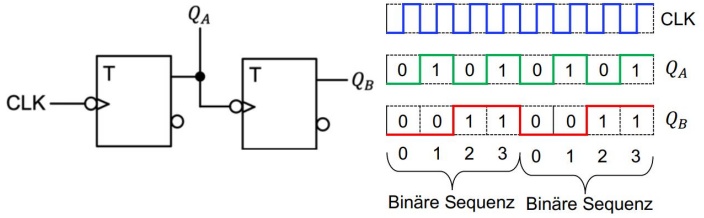


Frequenzteiler und Zähler

T-FF können gut verwendet werden, um die Periode T eines periodischen Signals zu verlängern. Mit n-T-FFs kann die Frequenz durch den Faktor 2^n geteilt werden.



Eine andere wichtige Anwendung von FFs ist als digitale Dualzähler. Dafür müssen T-FFs mit Rückflankensteuerung ($1 \rightarrow 0$) verwendet werden. Mit n-T-FFs kann man von 0 bis 2^{n-1} zählen.



Automaten

Ein Automat beschreibt ein System, welches auf seine Eingänge reagiert und ein Ausgangsignal produziert, dass vom Eingangssignal und momentanen Zustand abhängt.

Endliche Automaten, bzw. Finite state machines (FSM), können nur vorprogrammierte Lagen passieren. Schaltungen mit Rückkopplungen sind typische Beispiele für FSM.

Bei synchronen Automaten sind alle FF gleich getaktet, d.h. Zustandsänderungen sind synchron mit dem Takt.

Mit n-D-FF kann eine FSM 2^n innere Zustände speichern.

Beschreibung von Automaten

Formelle Beschreibung von Automaten

$X_n = (x_1, \ldots, x_e)$	Eingangsalphabet
$Y_n = (y_1, \ldots, y_b)$	Ausgangsalphabet
$Z_n = (z_1, \ldots, z_m)$	Zustandsmenge
$Z_0 \in Z$	Anfangszustand
$f_{c1} : (X_n, Z_n) \to Z_{n+1}$	Folgezustandsfunktion
$f_{c2} : (X_n, Z_n) \to Y_n$	Ausgangsfunktion

Zustandsfolgetabellen (Folgezustandstabellen)

Listet in Form einer Wahrheitstabelle alle Kombinationen des aktuellen Zustandvektors Z_n und Eingangsvektor X_n, aus welchen der Ausgangsvektor Y_n und Folgezustandvektor Z_{n+1} entsteht.

No	Eingang X_n	Momentaner Zustand Z_n	Folgezustand Z_{n+1}	Ausgang Y_n
1	$x_1, x_2, \cdots, x_e$	$z_{1n}, z_{2n}, \cdots, z_{mn}$	$z_{1n+1}, z_{2n+1}, \cdots, z_{mn+1}$	$y_1, y_2, \cdots, y_b$
⋮				
v				

Man kann die verschiedenen Vektoren abstrahieren durch Dualzahlen oder ihnen sinnvolle Namen geben, welche den Zustand/Eingang/Ausgang erklären.
Um die Anzahl Zeilen und Spalten einer Zustandsfolgetabelle zu bestimmen kann man die folgenden Formeln verwenden.

$$Z = 2^{eb+zb}$$

$$S = eb + (2 \cdot zb) + ab$$

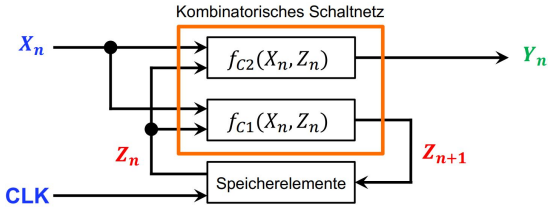
eb ist die Anzahl Eingangsbits, zb die Anzahl Zustandsbits und ab die Anzahl Ausgangsbits.

Zustandsdiagramme (Zustandsgraphen)

Äquivalente graphische Darstellung der Folgezustandstabelle. Zustandsdiagramme bestehen aus Kanten und Knoten, siehe spätere Abbildungen bei den verschiedenen Automatentypen.

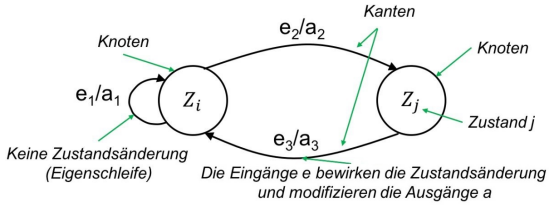
Mealy-Automat

Bei einem Mealy-Automat beinflussen Eingangsveränderungen das Ausgangsignal jederzeit, d.h. das Ausgangssignal ist Störungsanfällig. Ein Mealy-Automat hat folgende Struktur:



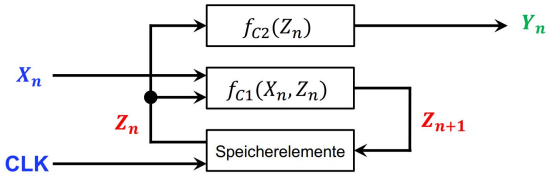
$$Y_n = f_{C2}(X_n, Z_n) \qquad Z_{n+1} = f_{C1}(X_n, Z_n)$$

Erst bei einem Taktwechsel $t_n \to t_{n+1}$ schalten die D-FF, d.h. $Z_{n+1} \to Z_n$.

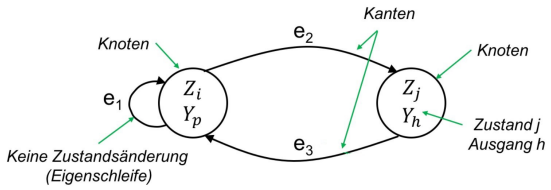


Moore-Automat

Sonderfall des Mealy-Automat, wo Y_n nur von getaktetem Z_n abhängt. Aus diesem Grund ist er weniger Störungsanfällig.



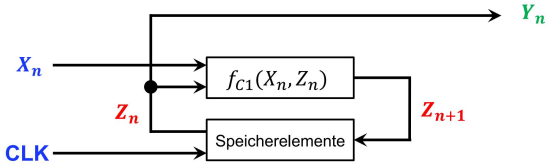
$$Y_n = f_{C2}(Z_n) \qquad Z_{n+1} = f_{C1}(X_n, Z_n)$$



Eine komplett störungssichere Automatenfunktion ist jedoch nur möglich, wenn die Eingangswerte X_n mit gleich getakteten D-FF synchronisiert werden.

Medwedjew-Automat

Ausgangsvektor Y_n ist identisch mit dem Zustandsvektor Z_n.



$$Y_n = Z_n \qquad Z_{n+1} = f_{C1}(X_n, Z_n)$$

Entwurf eines Automaten

Die Normale Entwurfsabfolge ist, wie folgt:

1. Auftrag lesen und analysieren
2. Zustandsmenge bestimmen, daraus folgt Anzahl Zustandsvariablen und D-FF.
3. Kodierung: Eingangs- und Ausgangsvariablen definieren
4. Darstellung der Zustandsfolge in einem Zustandsdiagram
5. Zustandsfolgetabelle aufstellen.
6. Minimierung der Ausgangs- und Übergangsfunktionen mit KV-Diagrammen
7. Einfluss unbenutzter Zustände (Don't Cares) überprüfen!
8. Schaltplan anhand der Schaltfunktion konstruieren

Wenn das Schaltwerk eines Automaten vorgegeben ist, wird die Reihenfolge vertauscht.

Umwandlung Mealy ⇔ Moore

Achtung: Bei der Umwandlung verändert sich das Zeitverhalten der Ausgänge.

Moore → Mealy

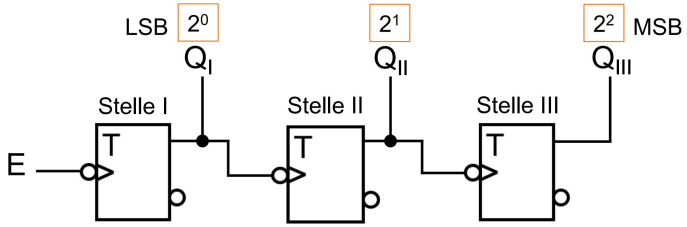
Die Moore zu Mealy Umwandlung ist einfach, da einfach die Ausgänge von den Knoten der Folgezustände auf die entsprechenden Kanten gewechselt werden müssen.

Mealy → Moore

Ein Mealy-Automat lässt sich immer in einen Moore-Automat mit gleicher Funktion umwandeln, dieser besitzt aber in der Regel mehr Zustände.
Eine direkte Moore Implementierung ist nur möglich, wenn jeder Zustand Z_n, unabhängig von X_n, immer das gleiche Ausgangssignal Y_n produziert. Ist dies nicht der Fall, dann muss man neue Zustände definieren.

Asynchronzähler

Ein einfacher Asynchron Vorwärtszähler aus T-FF sieht, wie folgt, aus:



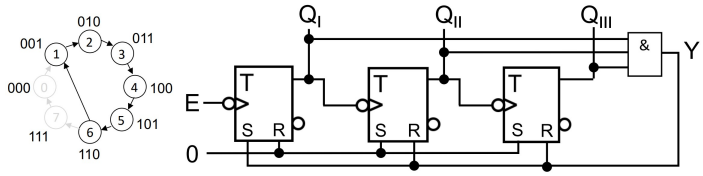
Will man einen Rückwärtszähler realisieren, dann muss man auf die positive Flanke gesteuerte FFs verwenden $(0 \rightarrow 1)$ und statt Q_i die invertierten Ausgänge $\overline{Q}_i$ benutzen.

Die Problematik vom Asynchronzähler ist, dass die Zustandsänderungen einen 'ripple' Effekt aufzeigen, d.h. die FF-Verzögerungszeiten kumulieren sich entlang der Schaltung. Die maximale Taktfrequenz, für welche ein Asynchronzähler 'theoretisch' noch funktioniert, ist:

$$f_{max} = \frac{1}{\sum t_{pd,FFs}}$$

Modulo-n Zähler

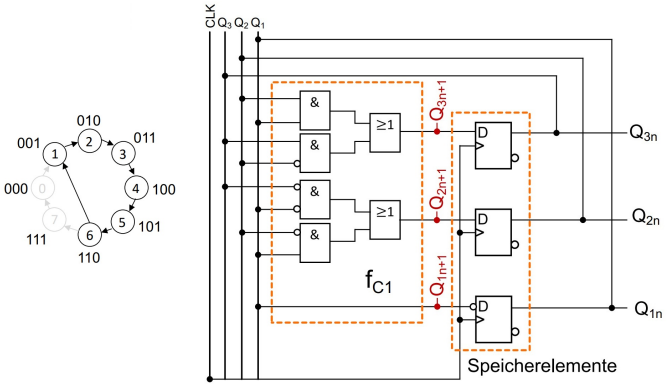
Ein Modulo-n Zähler zählt bis zu einem bestimmten Zustand n und springt dann zum vorgegebenen Anfangszustand zurück. Dafür benötigt man FFs mit Asychronen Reset und Set Eingängen, welche benutzt werden um den Rücksprung zum Anfangszustand auszuführen.



Der Zustand $n + 1$ ist kurzzeitig für die Durchlaufzeit des AND-Gatters vorhanden an den Ausgängen, bis der Rücksprung zum Anfangszustand durchgeführt wurde.

Synchronzähler

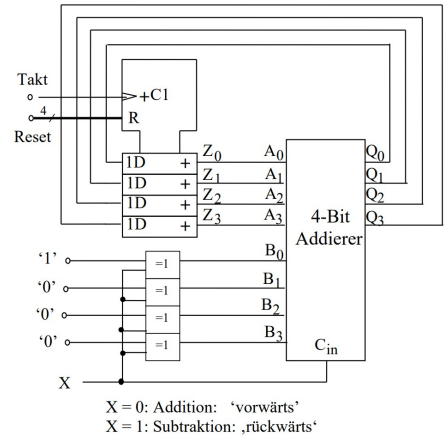
Bei einem Synchronzähler haben alle FF den gleichen Takt. Synchronzähler sind meist Medwedjew-Automanten, wo der Steuereingang benutzt wird um zwischen Vorwärts- und Rückwartszählen zu wechseln. Beispiel anhand eines Modulo-6 Zählers mit Anfangszustand '1':



Falls JK-FF verwendet werden müssen, dann muss man bei der Minimierung mit dem KV-Diagramm aufpassen, dass man die charakteristische Gleichung des JK-FF einhält.

Vorwärts-Rückwärtszähler (Reversible Zähler)

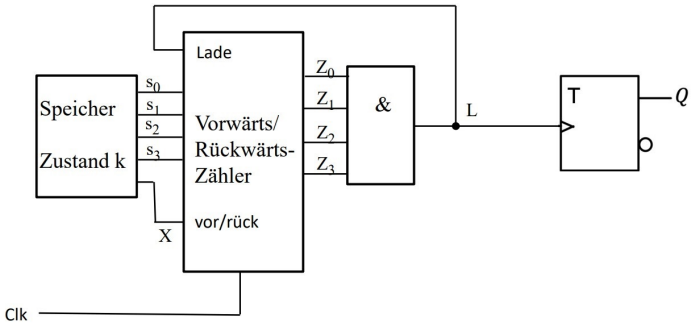
Ein grosser reversibler Zähler ist relativ aufwendig zu realisieren als Automat. Einfacher ist es D-FFs geschickt mit einem Addierer zu kombinieren.



Für das Wechseln zwischen Vorwärts- und Rückwärtszählen wechselt man beim Addierer einfach zwischen Addition und Subtraktion (2er Komplement).

Frequenzteiler mit einem Zähler

Gegeben ist ein Zähler mit N Zuständen und einem Anfangszustand k. Wenn der Zustand N erreicht wird, dann wird bei der nächsten Taktflanke der Anfangszustand k geladen und der T-FF wechselt seinen Zustand.



Mit einer solchen Schaltung kann man einen Frequenzteiler realisieren, welcher flexibler ist in seinem Teilungsverhältnis als ein simpler Frequenzteiler aus nur T-FFs.

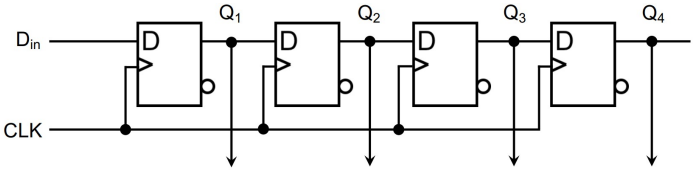
Der Ausgang vom T-FF hat die Frequenz:

$$f_{out} = \frac{f_{in}}{2(N-k+1)}$$

Speicher

Schieberegister

Ein Schieberegister ist ein Seriell-Parallel Wandler. Es ist eine Kette von D-FFs, durch welche die Seriellen Daten getaktet 'durchgeschoben' werden.



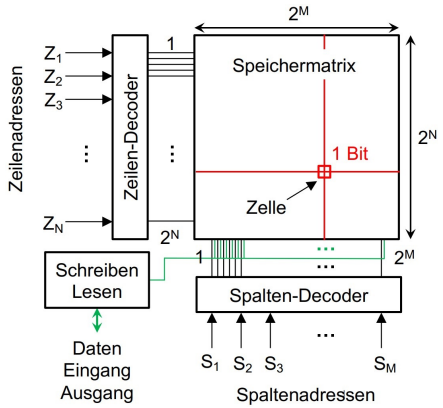
Schieberegister finden unter anderem Verwendung als Speicherregister um Daten zwischen 2 'Orten' zu speichern.

Halbleiterspeicher

Bei Halbleiterspeicher muss unterschieden werden zwischen:

ROM:	'Read only memory' (Nur lesen)
RAM:	'Random access memory' (Schreiben & Lesen, Wahlfreier Zugang)
Flüchtig:	Abhängig von Versorgungsspannung
Nicht flüchtig:	Unabhängig von Versorgungsspannung

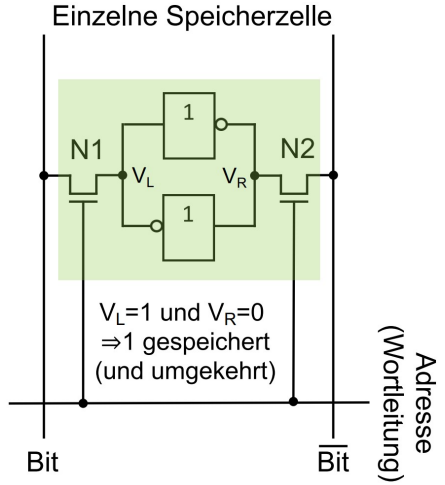
Halbleiterspeicher werden oft als Zellenarray oder Matrizen organisiert, die von einem Zeilen-Decoder und einem Spalten-Decoder angesteuert werden.



Dank den Decoder ist jede Speicherzelle einzeln ansprechbar mit nur linearem Wachstum in Adressleitungen bei höherer Zeilen-/Spaltenanzahl. Das Konstruktionsprinzip ist ein 'AND-Baum', wo jedes AND ein Minterm bildet. So besitzt jede Speicherzelle eine eindeutige Adresse.

SRAM (Statische-RAM)

SRAMs sind flüchtige Speicher, welche überall Anwendung finden, wo schneller Datenzugriff notwendig ist. Eine SRAM Zelle besteht aus 6 Transistoren. Mit der Adressleitung selektiert man die Speicherzelle.

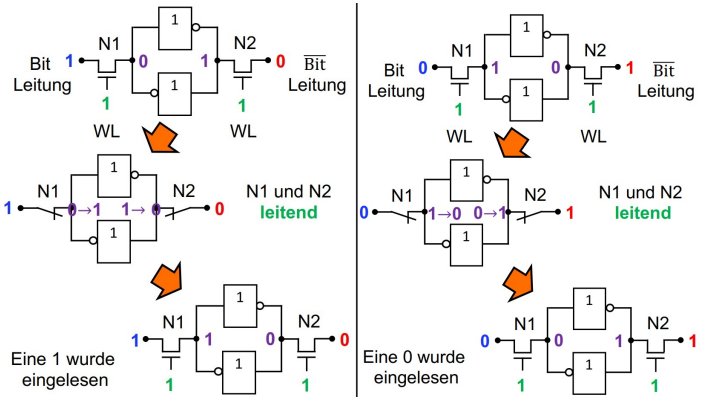


An V_L liegt der gespeicherte Wert der Speicherzelle und an V_R der invertierte Wert.

CMOS sind symmetrische Bauteile, Source und Drain hängen vom Elektronenstrom ab. Dementsprechend schaltet der CMOS, wenn vom Gate zu einem der beiden symmetrischen Anschlüssen ein Spannungsabfall vorhanden ist.

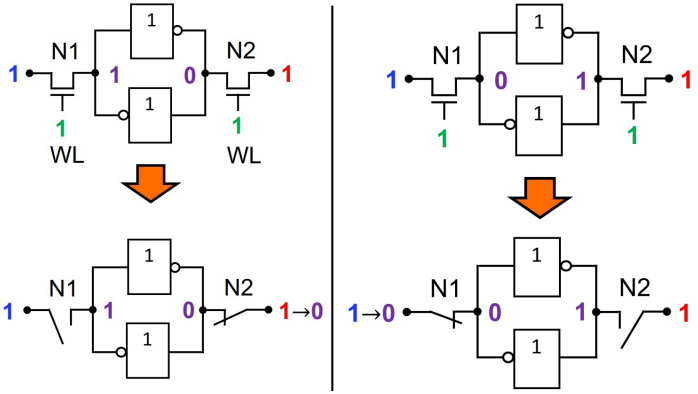
Schreibevorgang

Die Bitleitung wird beim Schreibevorgang fest auf GND/VCC geschaltet.



Lesevorgang

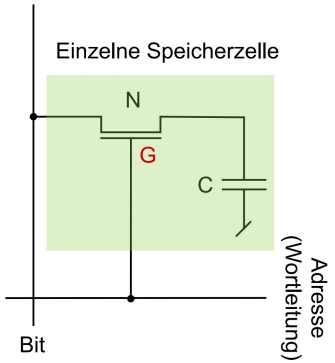
Die Bitleitung ist nicht fest auf GND/VCC geschaltet und verhält sich wie ein Kondensator. Sie übernimmt deswegen den Wert der Speicherzelle.



Links wird eine 1 ausgelesen und Rechts eine 0.

DRAM (Dynamische-RAM)

DRAMs sind flüchtige Speicher, welche periodisch (20ms) wieder aufgefrischt werden müssen aufgrund des Leckstroms vom Kondensator. DRAMs besitzen eine höhere Dichte als SRAMs aber langsämere Zugriffszeiten.



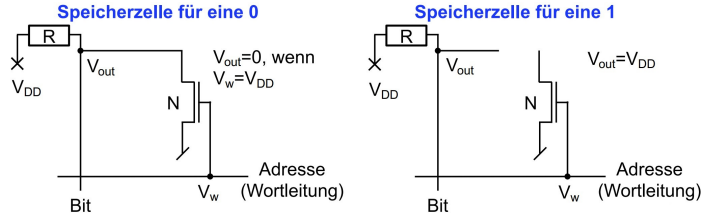
Der Kondensator agiert als Speicher. Beim Schreibevorgang wird die Bitleitung fest auf GND/VCC geschaltet und der Kondensator aufgeladen (1) bzw. entladen (0).

Beim Lesevorgang ist die Bitleitung nicht fest auf GND/VCC geschaltet und verhält sich wie ein Kondensator. Sie übernimmt deswegen den in der Zelle gespeicherten Wert.

MROMs (Maskable ROM)

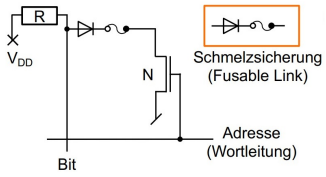
ROMs sind nichtflüchtige Speicher. Sie werden überall dort benötigt, wo ein System unveränderliche Daten-/Programmstrukturen benötigt.

MROM Speicherzellen werden durch spezielle 'Masken' bei der Herstellung als 0 oder 1 programmiert (Ideal für Massenproduktion).



PROMs (Programmable ROM)

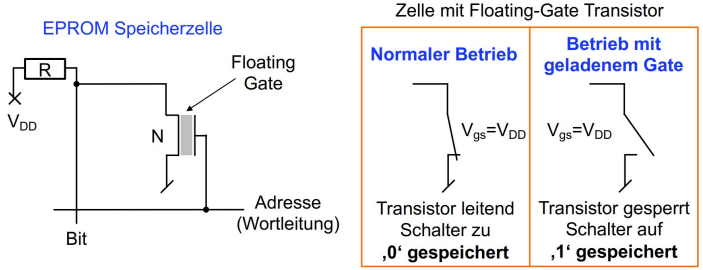
PROMs sind für kleinere Produktionsvolumen geeignet, weil sie durch den Benutzer einmalig programmiert werden können.



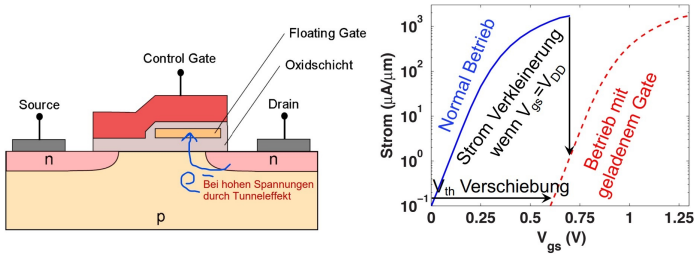
Die Programmierung erfolgt durch die Schmelzsicherung. Für eine '0' muss man nichts machen und für eine '1' wird die Schmelzsicherung durch eine hohe Spannung durchgebrannt.

EPROM (Erasable PROM)

EPROMs lassen sich durch UV-Bestrahlung optisch löschen und werden durch Anlegen einer hohen Spannung programmiert. Sie sind also mehrfach beschreibbar (≈1'000-Mal).



Floating-Gate Transistoren können negative Ladungen speichern, was die Kennline der für das Schalten notwendigen Gate-Source Spannung nach rechts verschiebt, als Folge reicht die $V_{DD} \approx 0.8V$ Spannung nicht mehr aus für das Schalten des Transistors.

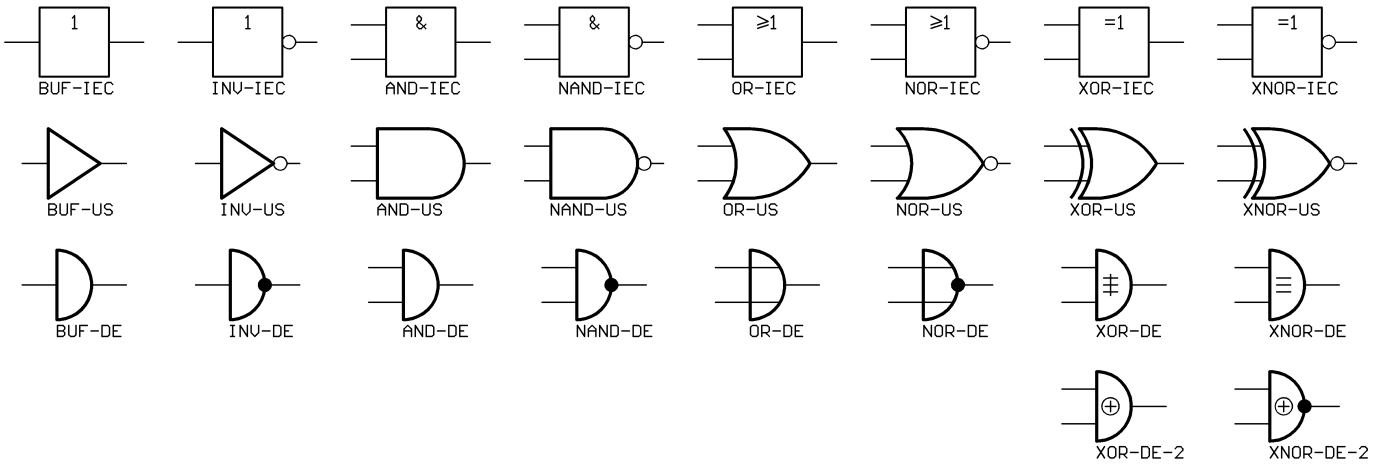


Nach etwa 20 minütiger Bestrahlung durch UV-Licht sind die negativen Ladungen wieder aus dem Floating-Gate entfernt und der Transistor zeigt erneut normales Verhalten.

EEPROM (Electrically erasable PROM)

Mit einer dünneren Oxidschicht beim Floating-Gate Transistor können die Gate Ladungen auch elektrisch gelöscht werden. Solche EEPROMs können ohne Probleme 10^6-Mal neubeschrieben werden.

EEPROMs sind, unter anderem, die Grundbausteine von den heutzutage weit verbreiteten FLASH Speicher.


1
BUF-IEC
1
INV-IEC
&
AND-IEC
&
NAND-IEC
≥1
OR-IEC
≥1
NOR-IEC
=1
XOR-IEC
=1
XNOR-IEC
BUF-US
INV-US
AND-US
NAND-US
OR-US
NOR-US
XOR-US
XNOR-US
BUF-DE
INV-DE
AND-DE
NAND-DE
OR-DE
NOR-DE
XOR-DE
XNOR-DE
XOR-DE-2
XNOR-DE-2