Digitaltechnik

Jirayu Ruh

Inhaltsverzeichnis

DISCLAIMER

Diese Notizen wurden verfasst auf Basis der Vorlesung Digitaltechnik (HS24) von M. Luisier, sowie dem Übungsskript von M. Vasylyev.

Ich übernehme keine Haftung über mögliche Fehler in den Notizen (Es hat sicherlich ein paar drinnen, da ich teils Sätze umformuliert habe und meine Persönliche Notizen beigefügt habe!).

Falls nicht anders deklariert wurden alle Grafiken eigenhändig mit Manim [The Manim Community Developers, 2024] oder CircuiTikz generiert.

Fehler können per Mail an jirruh@ethz.ch gemeldet werden.

Kapitel 1

Einführung in die digitale Welt

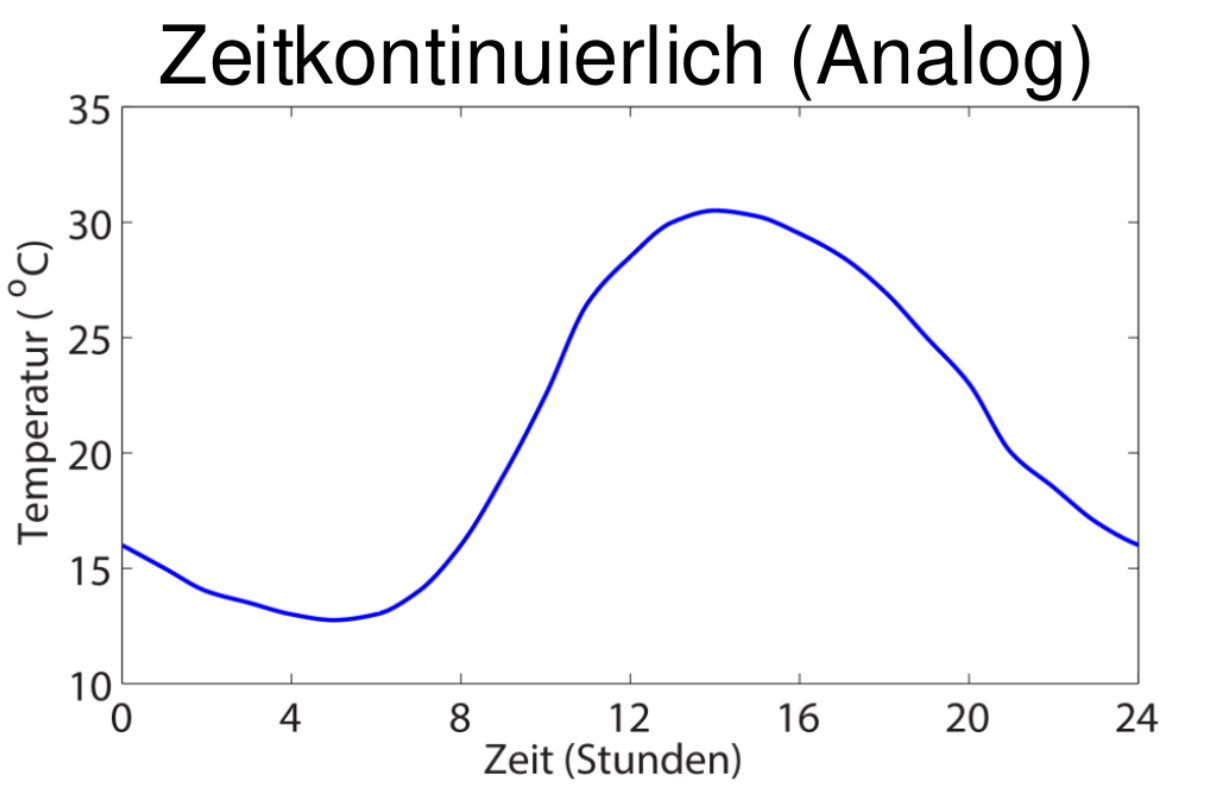
1.1 Grundlagen der Elektronik

1.1.1 Analog vs. Digital

Signale haben in der Elektrotechnik eine grosse Bedeutung. Wir unterscheiden zwischen zwei verschiedenen Signalen.

Definition 1.1.1: Das analoge Signal

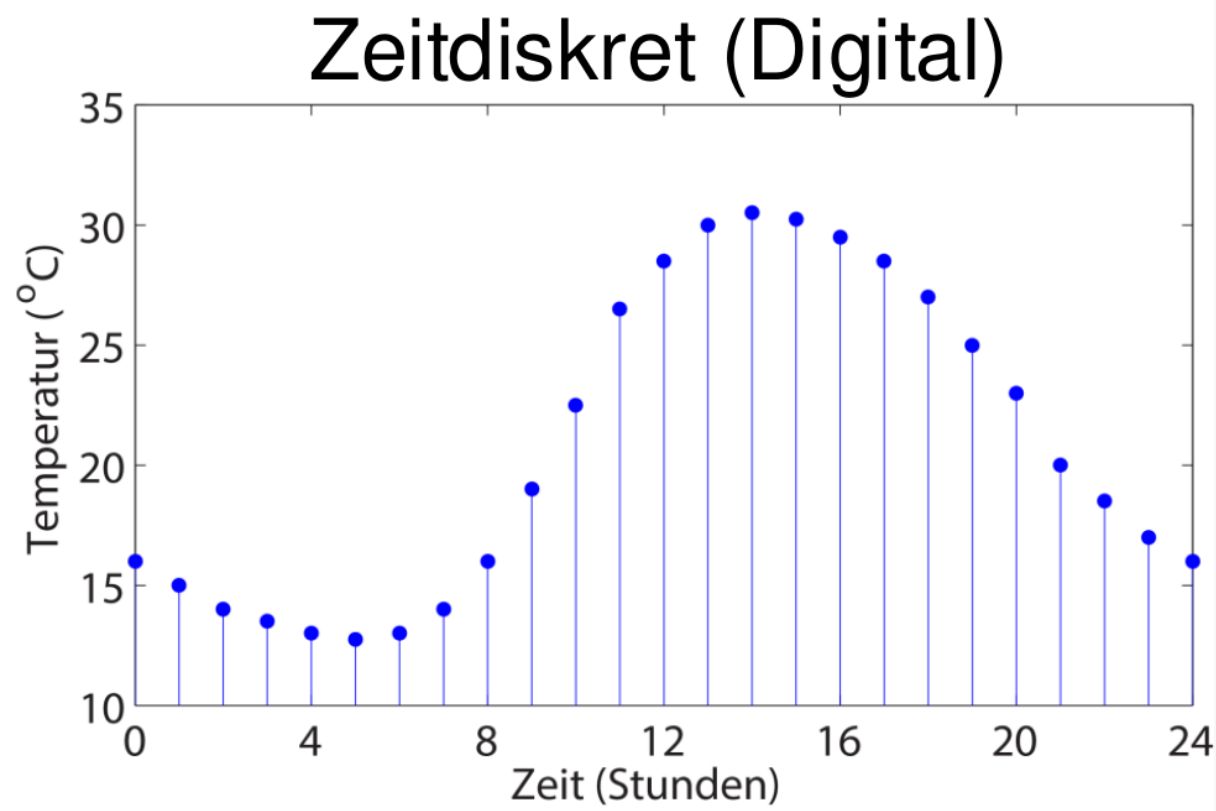
Analoge Signale haben kontinuierliche Werte und liefern deshalb unendlich genaue Informationen. [Luisier, 2024]



[Luisier, 2024]

Definition 1.1.2: Das digitale Signal

Digitale Signale bestehen aus diskreten Werten und hängen von der Betrachtungsart ab. [Luisier, 2024]



[Luisier, 2024]

In den meisten Fällen interessieren uns die Werte der digitalen Signale da diese ausreichend Daten liefern. Nun stellt sich die Frage welche Auflösung man haben möchte und wie sie dargestellt werden sollen.

Die Auflösung hängt nach Anwendungsbereich ab. Je nachdem braucht man eine höhere oder eine tiefere Auflösung. Für die Darstellung werden hauptsächlich Bits verwendet.

Definition 1.1.3: Bit

Ein Bit (binäre Stelle, englisch: Binary digit) kann verwendet werden, um Zustände darzustellen. [Luisier, 2024]

Bits können zwei Zustände haben: 1 und 0. Mehrere Bits hintereinander nennt man Codierung. Das erste Bit wird "Most Significant Bit"(MSB, höchstes Gewicht) genannt, das letzte "Least Significant Bit"(LSB, niedriges Gewicht). Wie tut man nun mit Bits Daten darstellen?

Bits können verschiedene Zustände annehmen. Diese Zustände können zu bestimmten Daten korrespondieren. Je mehr Bits man hat, desto mehr Daten kann man darstellen. Genauer gesagt kann man die Anzahl Zustände z mit den Anzahl Bits n wie folgt berechnen.

$$z = 2^n.$$

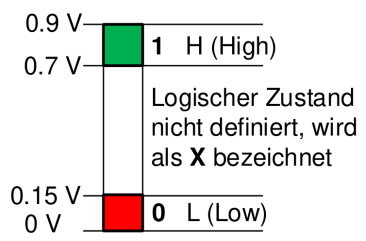
1.1.2 Strom und Spannung

In NUS I haben wir das Konzept von Strom und Spannung angeschaut. In der Digitaltechnik werden wir nun diese verwenden, um die Zustände von Bits darzustellen.

Die Digitaltechnik verwendet zwei verschiedene Methoden, um Bits darzustellen.

Definition 1.1.4: Darstellung der Bits durch Spannung

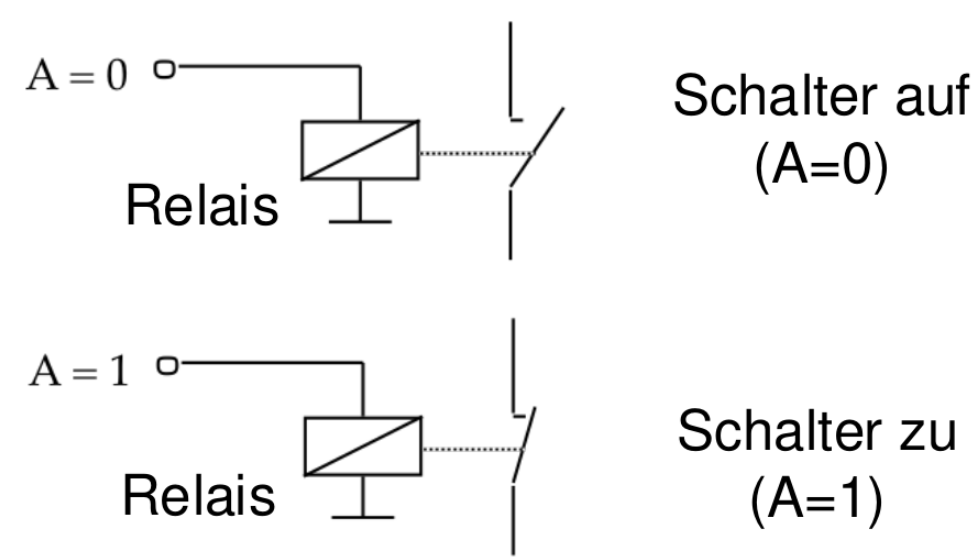
Der Zustand vom Bit kann durch die Menge der Spannung bestimmt werden. Ist die Spannung auf High (0.8 - 1 V), so hat der Bit den Zustand 1. Hat die Spannung einen Wert von 0 - 0.15 V (Low), so hat der Bit den Zustand 0.



[Luisier, 2024]

Definition 1.1.5: Darstellung der Bits durch Schalter

Bits können auch durch Schalter, genauer gesagt durch Relais dargestellt werden. Liegt beim Eingang vom Relais keine Spannung vor, so bleibt der Relais offen und der Zustand vom Bit ist 0. Im umgekehrten Fall schliesst sich der Relais und der Zustand vom Bit ist 1.



[Luisier, 2024]

Kapitel 2

Logische Verknüpfung

2.1 Basisfunktion

2.1.1 Nützliche Konzepte

Definition 2.1.1: Schaltfunktion

Eine Schaltfunktion f nimmt mehrere Variablen X_i, z.B. N, als Eingang und produziert eine einzige Variable Y als Ausgang.

$$Y = f(X_0, X_1, X_2, ..., X_{N-1}) \tag{2.1}$$

Der Informationsgehalt der Variablen beträgt ein Bit (1 oder 0). [Luisier, 2024]

Schaltfunktionen sind nichts anderes als eine Codierung. Damit man die Verschiedenen Codierungen zu bestimmten Ausgängen zuteilen kann, verwendet man Wahrheitstabellen.

Wahrheitstabellen nehmen in den linken Spalten die Wertekombinationen der Variablen X_i. In der rechten Spalte befindet sich das Ergebnis der Ausgangsvariable Y. Die grösse der Wahrheitstabelle hängt von der Anzahl der Variablen X_i ab. Grundsätzlich kann man sich folgendes Merken für die Anzahl Spalten S und Anzahl Zeilen Z.

$$S = i + 1.$$

$$Z = 2^i.$$

2.1.2 UND, ODER, NICHT Verknüpfung

In diesem Kapitel schauen wir uns verschiedene Verknüpfungen an. Diese sind relevant für das Verständnis von Schaltfunktionen.

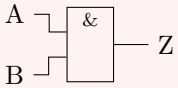
Definition 2.1.2: AND Verknüpfung

Wenn Aussage A (Eingang) wahr **und** Aussage B (Eingang) wahr sind, dann ist Aussage Y (Ausgang) wahr. [Luisier, 2024]

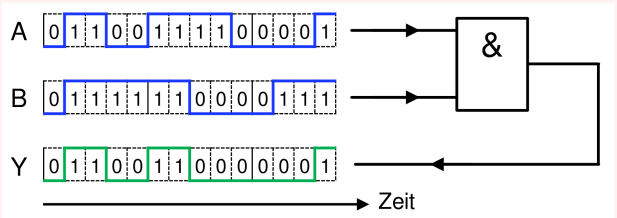
In einer logischen Gleichung wird das UND wie folgt gekennzeichnet.

$$A \wedge B = Y.$$

Das UND wird mit dem folgenden Schaltzeichen gekennzeichnet.



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



[Luisier, 2024]

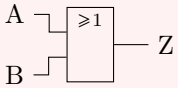
Definition 2.1.3: OR Verknüpfung

Wenn Aussage A (Eingang) wahr oder Aussage B (Eingang) wahr ist, dann ist Aussage Y (Ausgang) wahr. [Luisier, 2024]

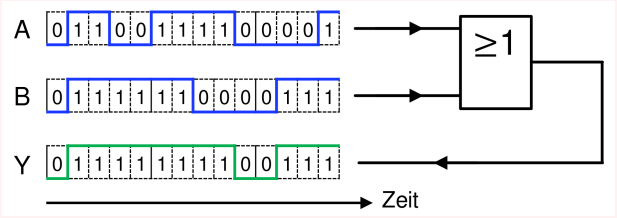
In einer logischen Gleichung wird das ODER wie folgt gekennzeichnet.

$$Y = A \vee B.$$

Das ODER wird mit dem folgenden Schaltzeichen gekennzeichnet.



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



[Luisier, 2024]

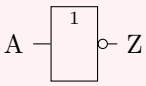
Definition 2.1.4: NICHT Verknüpfung

Wenn Aussage A (Eingang) wahr ist, dann ist Aussage Y (Ausgang) falsch. [Luisier, 2024]

In einer logischen Gleichung wird das NICHT wie folgt gekennzeichnet.

$$\bar{A} = Y.$$

Das NICHT wird mit dem folgenden Schaltzeichen gekennzeichnet.



A	Y
0	1
1	0

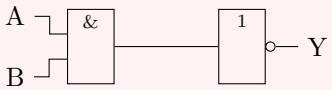
2.2 Schaltnetzanalyse

2.2.1 Zusammengesetzte Gatter

Die Darstellung von Schaltfunktionen kann mithilfe von Schaltnetzen realisiert werden. Die in Kapitel 2.1.2 erwähnten Schaltzeichen spielen in diesem Kapitel eine grosse Rolle. Wichtig ist vor allem die Kombination von Verknüpfungen.

Definition 2.2.1: NAND-Verknüpfung

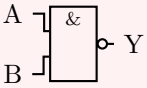
Invertierung der UND-Funktion



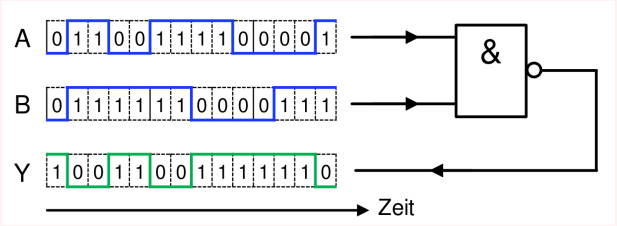
In einer logischen Gleichung wird das NAND wie folgt gekennzeichnet.

$$\overline{A \wedge B} = Y.$$

Das NAND wird mit den folgenden Schaltzeichen gekennzeichnet.



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



[Luisier, 2024]

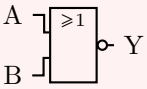
Definition 2.2.2: NOR-Verknüpfung

Invertierung der ODER-Funktion

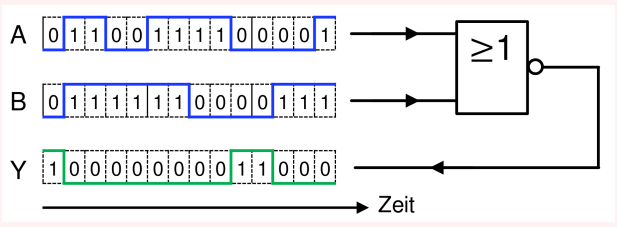
In einer logischen Gleichung wird das NOR wie folgt gekennzeichnet.

$$\overline{A \vee B} = Y.$$

Das NOR wird mit den folgenden Schaltzeichen gekennzeichnet.



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



[Luisier, 2024]

Definition 2.2.3: XNOR-Verknüpfung

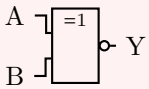
Ein Gatter, das eine logische 1 liefert, wenn beide Eingänge gleich sind, sonst 0. [Luisier, 2024]

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

In einer logischen Gleichung wird das XNOR wie folgt gekennzeichnet.

$$\overline{A \oplus B} = Y.$$

Das XNOR wird mit den folgenden Schaltzeichen gekennzeichnet.



Definition 2.2.4: XOR-Verknüpfung

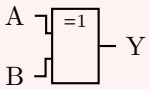
Ein Gatter, das eine logische 1 liefert, wenn beide Eingänge ungleich sind, sonst 0. [Luisier, 2024]

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

In einer logischen Gleichung wird das XOR wie folgt gekennzeichnet.

$$A \oplus B = Y.$$

Das XOR wird mit den folgenden Schaltzeichen gekennzeichnet.



Bemerkung:-

Das XOR-Gatter wird auch EXCLUSIV-OR Gatter genannt. [Luisier, 2024]

2.2.2 Schaltungen aus Grundgatter

Die Grundfunktionen und Schaltgatter sind nicht auf 2 Eingangsvariablen beschränkt. Grundsätzlich können Grundfunktionen und Schaltgatter N Eingänge haben.

Um die Komplexität von Schaltgattern mit mehreren Eingängen zu vereinfachen, kann man sie in mehreren Schaltgattern verwandeln, sodass die Logik der Schaltung lesbarer ist.

Kapitel 3

CMOS Schaltungen

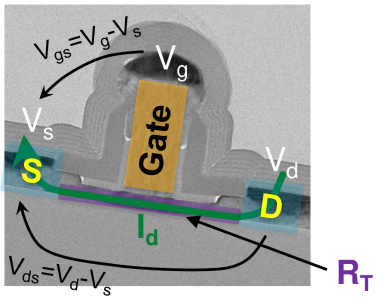
3.1 Wie funktionieren MOS-Transistoren?

3.1.1 Kurzer Überblick

In diesem Kapitel geht es um CMOS Transistoren. In Allgemeinen werden Transistoren verwendet, um Gatter zu bauen. Diese werden, wie in den letzten paar Kapitel besprochen in sehr vielen logischen Schaltungen verwendet, welche fundamental sind für viele elektronischen Geräte, welche uns begleiten.
Wie durch den Namen zu erkennen fungieren die Transistoren durch die CMOS Technologie. CMOS steht für "Complementary Metal-Oxide-Semiconductor". Die Transistoren sind also Metalloxid-Halbleiter.

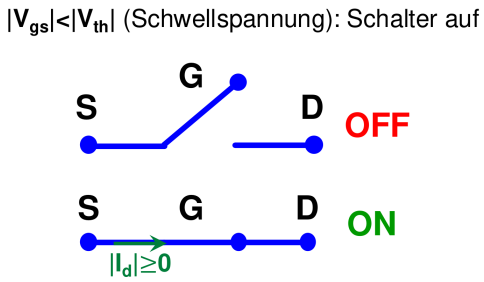
Definition 3.1.1: MOS Transistoren

MOS Transistoren sind steuerbare Widerstände bestehend aus 3 Kontakten, namentlich Source, Drain und Gate. Die Ladungsträger fliessen von Source zu Drain. Der Gate wird verwendet, um den Fluss von Ladungsträger zu beeinflussen.



Intel Single-Gate Transistor

R. Chau et al., *DRC*, (2003)



[Luisier, 2024]

MOS Transistoren verhalten sich wie Schalter. [Luisier, 2024] Ihr Widerstand wird durch die Source-Gate Spannung (V_{gs}) gesteuert. Dabei spielt die Schwellspannung (V_{th} th für Threshhold) eine grosse Rolle. Ist $|V_{gs}| < |V_{th}|$ ist der Transistor hochohmig und der Schalter ist zu. Im umgekehrten Fall ist der Transistor niederohmig. Somit ist der Schalter auf.

Bemerkung:-

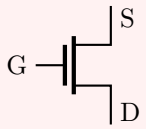
Die Schwellspannung V_{th} wird in diesem Semester keine grosse Rolle spielen. Es ist aber hilfreich fürs Verständnis.

Was sind jetzt aber CMOS Transistoren. Es gibt 2 CMOS Transistoren und beide sind ihr gegenseitiges Komplementär. Daher kommt auch ihr Name. Grundsätzlich unterscheidet man zwischen zwei verschiedene CMOS Transistoren.

Definition 3.1.2: NMOS-Transistor

NMOS-Transistoren haben einen n-dotierten Innenwiderstand. Dadurch erhält er die Eigenschaft, dass der Transistor leitet, sobald eine Spannung am Gate liegt.

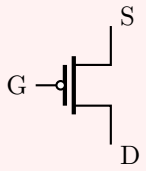
G	NMOS	Y
0	Leitet nicht	0
1	Leitet	1



Definition 3.1.3: PMOS-Transistor

PMOS-Transistoren haben einen p-dotierten Innenwiderstand. Dadurch erhält er die Eigenschaft, dass der Transistor leitet, sobald keine Spannung am Gate liegt.

G	PMOS	Y
0	Leitet	1
1	Leitet nicht	0



Bemerkung:-

Die Eigenschaft, dass NMOS- bzw. PMOS-Transistoren bei einer gewissen Gate Spannung leitet hat eine bestimmte Funktion bezüglich V_{gs}. NMOS Transistoren leiten, sobald eine positive Spannung am Eingang ist. Ist die Spannung negativ, leitet sie nicht. Umgekehrt ist es beim PMOS Transistor.

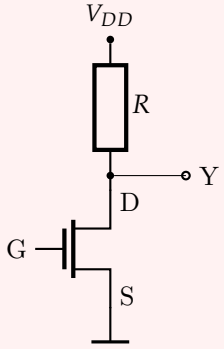
NMOS- und PMOS Transistoren haben vor allem eine bedeutende Verwendung bei Pull-up und Pull-Down Schaltungen.

Definition 3.1.4: Pull-down Schaltung

Eine Pull-down ist eine Schaltung, welcher die Spannung V_y von seinem Ausgang Y in Abhängigkeit von der Gate Spannung definiert. (V_{gs})

G	NMOS	Y
0	Leitet nicht	1
1	Leitet	0

$\Rightarrow$ Leitend falls Eingang $V_G = 1$, 0 liegt am Ausgang.

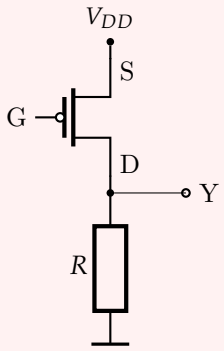


Definition 3.1.5: Pull-up Schaltung

Eine Pull-up ist eine Schaltung, welcher die Spannung V_y von seinem Ausgang Y in Abhängigkeit von der Gate Spannung definiert. (V_{gs})

G	NMOS	Y
0	Leitet	1
1	Leitet nicht	0

$\Rightarrow$ Leitend falls Eingang $V_G = 0, 1$ liegt am Ausgang.

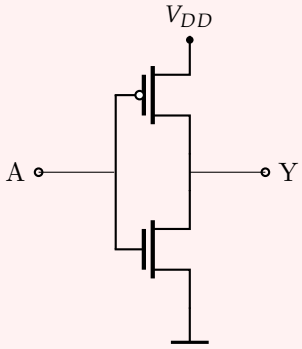


3.2 Pull-up und Pull-down Schaltungen

3.2.1 NOT, NAND, und NOR

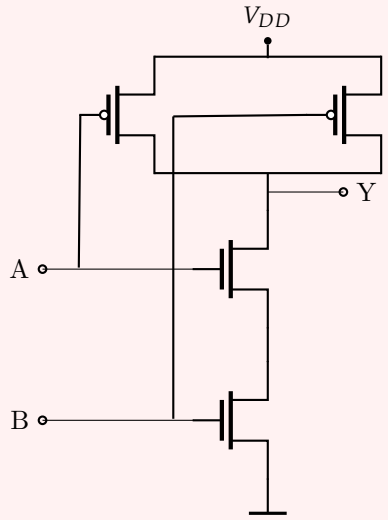
Definition 3.2.1: NICHT Gatter in CMOS Technik

Das NOT Gatter kann durch CMOS Gatter wie folgt dargestellt werden.



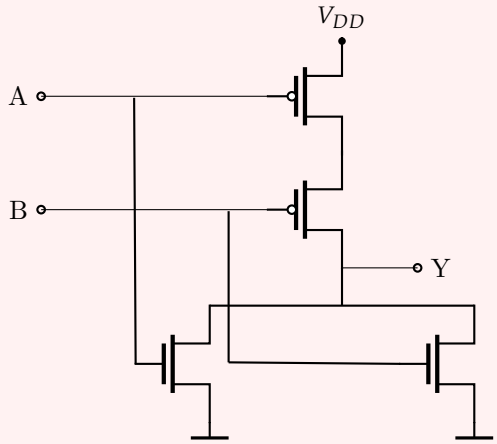
Definition 3.2.2: NAND Gatter in CMOS Technik

Das NAND Gatter kann durch CMOS Gatter wie folgt dargestellt werden.



Definition 3.2.3: NOR Gatter in CMOS Technik

Das NOR Gatter kann durch CMOS Gatter wie folgt dargestellt werden.



3.2.2 Komplexe Schaltungen

Jetzt wo wir wissen, wie logische Gatter durch CMOS Gatter dargestellt werden können, können wir komplexe logische Schaltungen durch CMOS Transistoren darstellen.

Definition 3.2.4: Pull-up / Pull-down Prinzip

Für die Darstellung von logischen Schaltungen durch CMOS Gattern gilt:

- CMOS Gatter müssen aus genau so vielen NMOS und PMOS Transistoren bestehen.
- Bei m Eingängen gibt es m NMOS und m PMOS Transistoren

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

[Luisier, 2024]

In den vorherigen Definitionen haben wir uns Pull-up und Pull-down Schaltungen von einzelnen logischen Gattern angeschaut. Wie wir aber in Kapitel 2.2.1 gesehen haben, können logische Gatter miteinander verknüpft werden. Dies ist auch der Fall für CMOS Gatter. Um die verschiedenen logischen Gattern zu erkennen, kann man beim Pull-up bzw. beim Pull-down Pfad die Transistoren in Blöcke unterteilen. Dabei gelten die folgenden Regeln.

Pull-up

- 2 parallelgeschaltene PMOS Transistoren → ODER
- 2 seriegeschaltene PMOS Transistoren → UND

Pull-down

- 2 parallelgeschaltene NMOS Transistoren → UND
- 2 seriegeschaltene NMOS Transistoren → ODER

Diese Regeln gelten nicht nur für einzelne Transistoren, sondern auch für Blöcke von Transistoren.

3.2.3 Zeitverzögerung

CMOS Gatter können nicht unendlich schnell schalten. Ladungen (Elektronen und Löcher) müssen bewegt werden, was Laufzeit Verzögerungen verursacht. [Luisier, 2024] In unserem Fall spielen der Propagation Delay von Low to High bzw. High to Low eine grosse Rolle. Aus der Tabelle kann man erkennen, dass es mehrere Faktoren hat, welche die Verzögerungszeit beeinflussen können. Wir fokussieren uns aber hauptsächlich auf die Propagation Delay.

t_{pHL}	Verzögerungszeit (Propagation Delay High Low)	Beim Übergang H → L Gemessen bei 50% des Pegelhubs
t_{pLH}	Verzögerungszeit (Propagation delay Low High)	Beim Übergang L → H Gemessen bei 50% des Pegelhubs
r_r	Anstieg- (Rise-) Zeit Transition Low High	Gemessen zwischen 10% und 90% des Pegelhubs
t_{tLH}	Transition Low High	
t_f	Abfall- (Fall-) Zeit	Gemessen zwischen 90% und 10% des Pegelhubs
t_{tHL}	Transition High Low	

In diesem Semester arbeiten wir hauptsächlich mit t_d (Delay Time). Diese kann wie folgt bestimmt werden.

$$t_d = \frac{t_{pLH} + t_{pHL}}{2}.$$

Kapitel 4

Schaltalgebra (Bool'sche Algebra)

4.1 Verknüpfungsgesetze

4.1.1 Basis- und Vereinfachungsregeln

Logische Funktionen können sehr komplex werden. Glücklicherweise können sie vereinfacht werden durch Rechenoperationen, welche uns schon bekannt sind.

- Die Reihenfolge der Variablen in UND-Verknüpfungen und in ODER-Verknüpfungen beeinflusst das Ergebnis nicht.
- Variablen können zu Gruppen zusammengefasst werden.
- Gemeinsame Variablen in UND- und ODER-Verknüpfungen können verteilt (ausmultipliziert, ausgeklammert) werden.
- $\bar{\bar{A}} = A$ (Nicht)
- $A \vee 0 = A \ / \ A \wedge 0 = 0$ (Null-Theorem)
- $A \vee 1 = 1 \ / \ A \wedge 1 = A$ (Eins-Theorem)
- $A \vee A = A \ / \ A \wedge A = A$ (Idempotenz)
- $A \vee \bar{A} = 1 \ / \ A \wedge \bar{A} = 0$ (Verknüpfung mit Komplement)
- $A \vee (\bar{A} \wedge B) = A \vee B \ / \ A \wedge (\bar{A} \vee B) = A \wedge B$ (Adsorptionsgesetz)
- $A \vee (A \wedge B) = A \ / \ A \wedge (A \vee B) = A$ (Absorptionsgesetz)
- $(A \wedge B) \vee (\bar{A} \wedge B) = B \ / \ (A \vee B) \wedge (\bar{A} \vee B) = B$ (Nachbarschaftsgesetz)

Für die Reihenfolge der Rechnungen gilt:

1. Klammern
2. Negation
3. AND, NAND, OR, NOR vor XOR, XNOR

Bemerkung:-

AND, NAND, OR, NOR, XOR und XNOR sind untereinander gleichwertig.

4.2 Die De Morgan'schen Regeln

4.2.1 Beziehung zwischen UND und OR

Es existieren Beziehungen zwischen den NOR/UND und zwischen den NAND/ODER Verknüpfungen, sodass ein gegenseitiger Austausch möglich ist: De Morgan'sche Regeln. [Luisier, 2024]

De Morgan'sche Regeln

1. Die NAND-Funktion kann durch eine ODER-Funktion mit invertierten Eingängen ersetzt werden. ($\overline{A \wedge B} = \bar{A} \vee \bar{B}$)

2. Die NOR-Funktion kann durch eine UND-Funktion mit invertierten Eingängen ersetzt werden. ($\overline{A \vee B} = \bar{A} \wedge \bar{B}$)

[Luisier, 2024]
Mit den Grundgattern UND, ODER, NICHT kann jede logische Verknüpfung realisiert werden. Die De Morgan'schen Regeln ermöglichen den Ersatz von Grundgattern. Jedes "Universalgatter„ NAND oder NOR allein ist ausreichend, um logische Verknüpfungen zu realisieren. [Luisier, 2024]

Bemerkung:-

Die Beziehung zwischen der Pull-up und Pull-down Schaltung ist durch die De Morgan'schen Regeln gegeben. [Luisier, 2024]

4.3 Normalformen

4.3.1 Definition von Min- und Maxterm

Definition 4.3.1: Minterm

Ein Minterm ist eine AND-Verknüpfte logische Schaltung. Bei n Eingangsvariablen gibt es 2^n mögliche Minterme.

Der Minterm kann bestimmt werden, indem man die AND-Verknüpfung aller vorhandenen Variablen bildet und die Variablen, die 0 sind, invertiert.

Bemerkung:-

Beim Minterm haben die nicht-invertierten Variablen den Wert 1, die invertierten Variablen den Wert 0.

Definition 4.3.2: Maxterm

Ein Maxterm ist eine OR-Verknüpfte logische Schaltung. Bei n Eingangsvariablen gibt es 2^n mögliche Maxterme.

Wie beim Minterm kann der Maxterm bestimmt werdem, indem man die OR-Verknüpfung aller vorhandenen Variablen bildet und die Variablen, die 1 sind, invertiert.

Bemerkung:-

Beim Maxterm haben die nicht-invertierten Variablen haben den Wert 0, die invertierten Variablen den Wert 1.

4.3.2 DNF und KNF

Ist die Wahrheitstabelle einer Schaltfunktion f gegeben, so können zwei verschiedene Normalformen gebildet werden. Diese werden im späteren Teil des Semesters wichtig sein.

Definition 4.3.3: Disjunktive Normalform (DNF)

Die Disjunktive Normalform ist die OR-Verknüpfung der Minterme.

Definition 4.3.4: Konjunktive Normalform (KNF)

Die Konjunktive Normalform ist die AND-Verknüpfung der Maxterme.

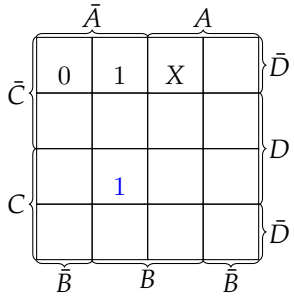
Kapitel 5

Schaltungssynthese

5.1 Karnaugh-Diagramme

5.1.1 Einführung und Regeln

Karnaugh Diagramme werden zur Vereinfachung von logischen Gleichungen verwendet. Wir haben zwar in Kapitel 4.1.1 und 4.2 eine Methode gelernt, logische Gleichungen zu vereinfachen. Die Karnaugh Diagramme sind eine andere Methode, welche in manchen Fällen schneller ist.



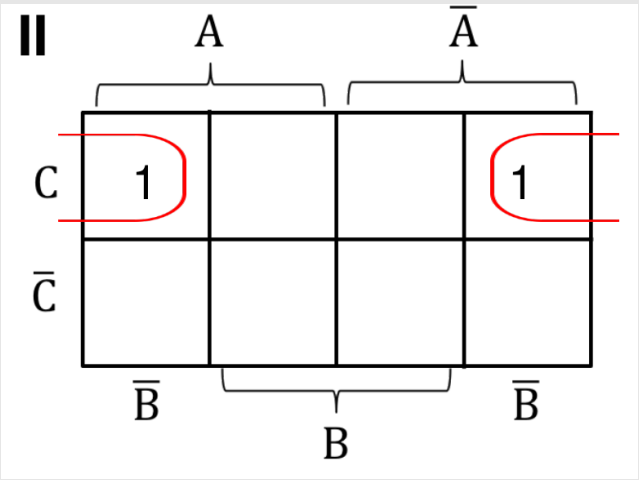
Karnaugh Diagramme können für logische Gleichungen mit mehreren Eingangsvariablen gebildet werden. Grundsätzlich gilt für n Eingangsvariablen hat das Karnaugh Diagram 2^n Felder. Jedes Feld im Karnaugh Diagramm ist für ein Minterm oder ein Maxterm gedacht. Dabei gilt wie bei der Wahrheitstablle wenn die Ausgangsvariable 1 ist handelt es sich um ein Minterm. Ist die Ausgangsvariable 0, so ist es ein Maxterm.
Das interessante an Karnaugh Diagrammen ist, dass man mit den Min- bzw. Maxtermen "Päckchen"bilden kann. Päckchen können wie folgt gebildet werden.

1. Päckchen haben eine Grösse von 2^n
2. Die Päckchen bestehen entweder nur aus Min- bzw. Maxtermen.
3. Es soll immer das grösstmögliche Päckchen gebildet werden.
4. Päckchen können sich schneiden.

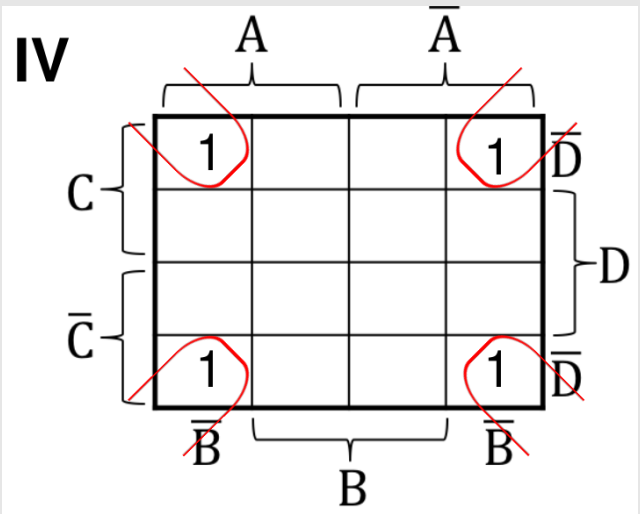
Durch das Bilden von Päckchen können neue Min- bzw. Maxterme gebildet werden, da in Päckchen Variablen in ihrer negierten und nicht negierten Form vorkommen. Diese gleichen sich aus und verschwinden.

Bemerkung:-

Karnaugh Diagramme bestehend aus mehreren Eingangsvariablen können ëigenartige"Päckchen haben.



[Luisier, 2024]

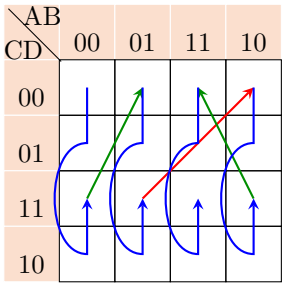


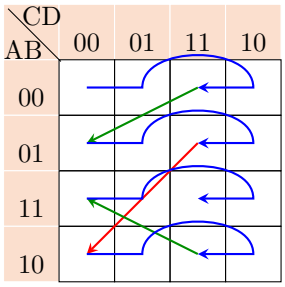
[Luisier, 2024]

Bemerkung:-

Karnaugh Diagramme können unendlich viele Eingangsvariablen haben, jedoch beschränken wir uns auf maximal fünf Eingangsvariablen, da Karnaugh Diagramme mit mehr als fünf Eingangsvariablen unübersichtlich wird.

Um das Karnaugh Diagramm mit einer Wahrheitstablle auszufüllen, gibt es eine bestimmte Vorgehensweise.





5.2 Don‘t Care Zustände (X)

5.2.1 Nicht benutzte Wertekombinationen

Ab und zu bekommen wir mehr Codierungen als wir nötig haben. Aus diesem Grund verwenden wir Don’t Care Zustände.

Definition 5.2.1: Don’t Care Zustände

Don’t Care Zustände kennzeichnen Codierungen, welche nicht relevant sind für die logische Funktion. Die Codierung wird mit einem X gekennzeichnet. Don’t Care Zustände können den Wert 0 oder 1 annehmen.

Don’t Care Zustände haben vorallem eine grosse Bedeutung für die Bildung von Päckchen. Da Don’t Care Zustände den Wert 1 und 0 annehmen können, können grössere Päckchen gebildet werden.

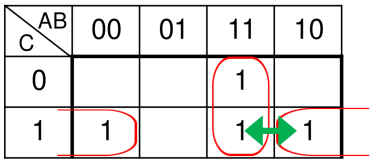
5.3 Was sind Hazards?

5.3.1 Identifizierung und Behandlung

Definition 5.3.1: Hazard

Ein Hazard ist eine kurzzeitige und unerwünschte Änderung von Signalwerten. [Luisier, 2024] Diese tritt meistens aufgrund der Zeitverzögerung von Gattern auf.

Hazards können aber frühzeitig erkannt und eliminiert werden mit Hilfe des Karnaugh Diagramms. Für Hazards gilt wenn sich Päckchen orthogonal berühren, markieren diese kritische Werteänderungen (statische Hazards). [Luisier, 2024] Durch ein zusätzliches Päckchen, welches die zwei orthogonal berührenden Min- bzw. Maxtermen in den Päckchen umschliesst kann der Hazard eliminiert werden.



[Luisier, 2024]

Bemerkung:-

Wenn sich Päckchen diagonal berühren entsteht ein funktionaler Hazard. Dieser kann nicht leicht behoben werden.

Kapitel 6

Zahlen und Codes

6.1 Zahlensysteme

6.1.1 Was ist ein Zahlensystem?

Definition 6.1.1: Zahlensystem

Ein Zahlensystem ist eine Methode, um Zahlen darzustellen. Wir Menschen sind gewohnt, das Dezimalsystem zu verwenden. In der Digitaltechnik sind jedoch andere Zahlensysteme von grosser Bedeutung.

Zahlensysteme

- Dezimalsystem stellt Zahlen in Zehnerpotenzen dar. (Basis = 10)
- Oktalsystem stellt Zahlen in Achterpotenzen dar. (Basis = 8)
- Dualsystem (Binärsystem) stellt Zahlen in Zweierpotenzen dar. (Basis = 2)
- Hexadezimalsystem stellt Zahlen in sechszehnerpotenz dar. (Basis = 16)

Bemerkung:-

Da das Hexadezimalsystem über die Zahl 9 geht werden Buchstaben für die darauffolgenden Zahlen verwendet.

Bemerkung:-

Das Oktal- und Hexadezimalsystem wird verwendet um das Binärsystem kompakter zu machen. Sie können päckchenweise ineinander umgerechnet werden. [Luisier, 2024]

6.1.2 Umwandlung zwischen Zahlensystemen

Wie vorher schon erwähnt können binär Zahlen sehr einfach in Oktalzahlen und Hexadezimalzahlen umgewandelt werden. Die Umwandlung von Dezimalzahlen in ein anderes Zahlensystem ist ein wenig komplizierter.
Grundsätzlich gilt, dass wenn man vom Dezimalsystem in ein anderes System umwandelt, dass man die Zahl durch die korrespondierende Basis dividiert. Der Rest entspricht jeweils die Ziffer in der Codierung. Bei der Division von Zahlen > 1 entsteht bei der ersten Division der MSB und bei der letzten Division der LSB. Bei Zahlen < 1 ist es umgekehrt.

6.1.3 Negative Dualzahlen (Zweierkomplement)

Für negative Zahlen ist es am einfachsten, die negative Zahl ins Binärsystem umzuwandeln und diese, wenn nötig in das gefragte Zahlensystem umzuwandeln. Dabei geht man wie folgt vor.

Umwandlung von negativen Zahlen

1. Den Betrag der negativen Zahl ins Binärsystem umwandeln.
2. Die umgewandelte negative Zahl invertieren. ($0 \rightarrow 1$ und $1 \rightarrow 0$)
3. Addiere 1 zum LSB.
4. Vorzeichenbit als MSB einfügen.

Bemerkung:-

Die negative Zahl im Binärsystem wird Zweierkomplement genannt. Ob das Binärsystem oder das Zweierkomplementsystem verwendet wird, wird meistens in der Aufgabe gesagt.

Negative rationale Zahlen bestehen aus einem Vorzeichenbit, sowie m Vorkomma- und n Nachkommabits. Somit sind $1 + m + n$ Bits erforderlich. Das Komma wird vor den Vorkommabits eingefügt.

6.1.4 Rechenoperationen mit Dualzahlen

Die Addition von zwei Dualzahlen erfolgt bitweise, beginnend mit den LSB, von rechts nach links [Luisier, 2024]. Falls die Codierungen nicht die gleiche Anzahl an Bits haben werden Nullen hinzugefügt.
Für die Subtraktion ist es am einfachsten, die Zahl, welche subtrahiert wird ins Zweierkomplement umzuwandeln und anschliessend zur anderen Codierung zu addieren.

6.2 Codes

6.2.1 Was ist ein Code?

In Kapitel 1.1 haben wir gelernt was eine Codierung ist. Codes kommen hauptsächlich in vierer Gruppen vor (Tetraden) und neben den Binär Codierung gibt es viele weitere Codes.

Codes mit Tetraden

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

Codes werden hauptsächlich verwendet, um Informationen zu speichern, Daten zu komprimieren und Fehlerkorrekturen und Verschlüsselungen von Nachrichtenübertragungen zu machen.

6.2.2 Fehlererkennung und Fehlerkorrektur

Bei der Übertragung von Codes können Fehler auftauchen. Deswegen werden neben den Codes auch Parity Bits übertragen. Parity kennzeichnen ob die Codierung gerade oder ungerade ist. Falls bei der Übertragung ein Fehler entstanden ist und der Parity Bit nicht übereinstimmt, so wird ein Fehler gemeldet.

Eine weitere Stufe zur Fehlererkennung ist das Verwenden von einem Prüfwort. Das Prüfwort ist eine Codierung mit einem Parity Bit. Das Prüfwort wird gebildet durch die Addition von mehreren Codierungen, welche auf Fehler untersucht werden sollen. Dabei wird der Übertrag ignoriert. Hat das Prüfwort ein Fehler, so stimmt der Parity Bit nicht mehr überein und es wird nach der Falschen Codierung gesucht.

Kapitel 7

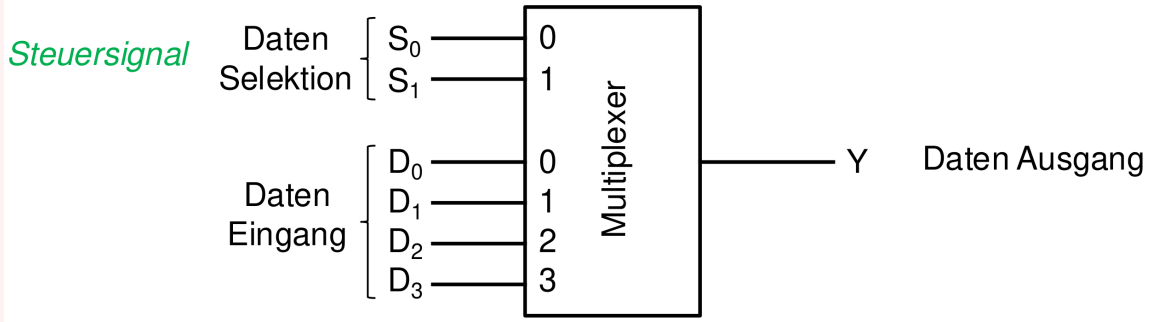
Rechenschaltungen und Datenpfadkomponenten

7.1 Was sind Datenpfadkomponenten?

7.1.1 Multiplexer und Code-Umsetzer

Definition 7.1.1: Multiplexer

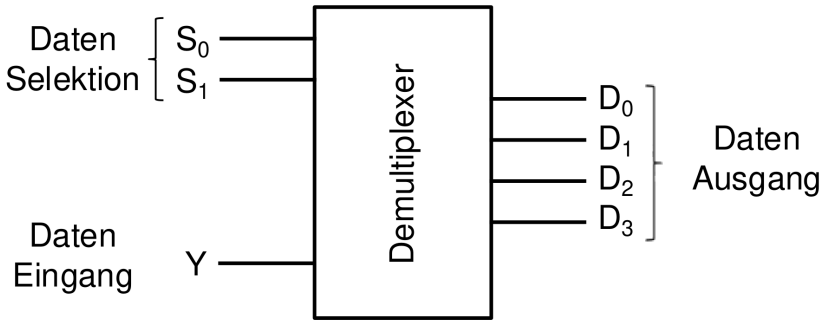
Multiplexer sind Bauelemente, welche durch eine Codierung als Eingang Daten als Ausgang schaltet. Multiplexer können auch erweitert werden indem man mehrere Kanäle einführt für den Dateneingang. Diese werden Daten Selektion genannt.



[Luisier, 2024]

Definition 7.1.2: Demultiplexer

Demultiplexer haben die umgekehrte Funktion von Multiplexer. Demultiplexer sind Bauelemente, welche Daten als Eingang haben und eine Codierung als Ausgang haben. Wie beim Multiplexer kann auch eine Daten Selektion gebraucht werden für verschiedene Kanäle.



[Luisier, 2024]

Definition 7.1.3: Code-Umsetzer

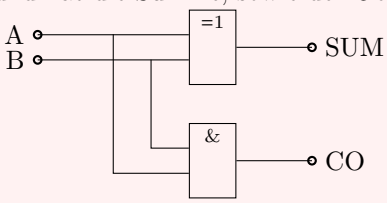
Code-Umsetzer können eine Codierung als Eingang nehmen und diese dann umwandeln in eine andere Codierung umwandeln (Kapitel 6.2).

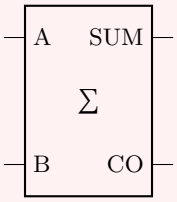
7.2 Addierer

7.2.1 Halb- und Volladdierer

Definition 7.2.1: Halbaddierer

Ein Halbaddierer ist ein Bauelement, welches in der Lage ist zwei Bits zu addieren. Es nimmt 2 Bits als Eingang und hat die Summe, sowie der Übertrag als Ausgang.

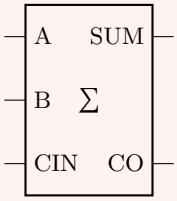




Für Zahlen im Binärsystem reichen Halbaddierer nicht aus, da sie den Übertrag nicht beachten. Deswegen kommt der Volladdierer ins Spiel.

Definition 7.2.2: Volladdierer

Volladdierer sind Bauelemente, dessen Funktion es ist bei der Summe von 2 Bits den Übertrag mit einzubeziehen.



7.2.2 Mehrbit-Addierer und Subtrahierer

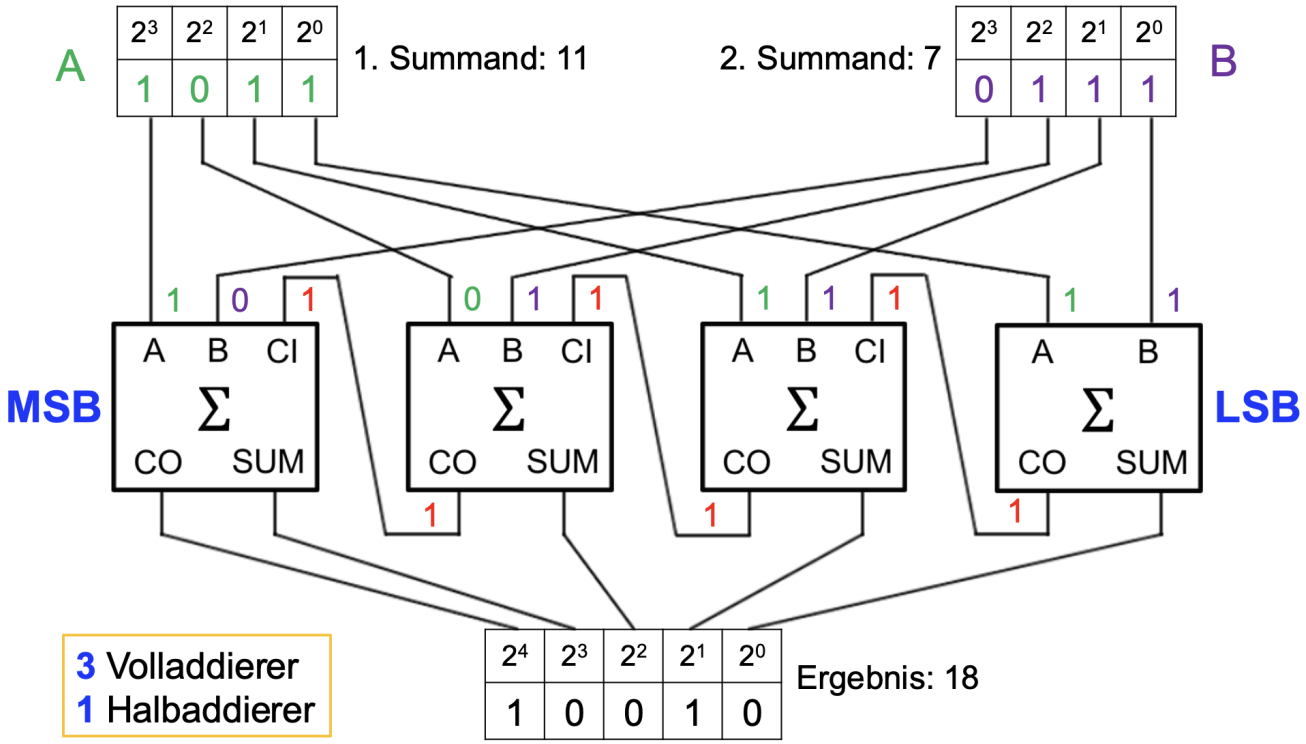
Wir haben nun die Basis zur Erstellung einer logischen Schaltung zum addieren von Zahlen im Binärsystem. Der Prozess vom Addieren von Dualzahlen kann entweder bitseriell oder bitparallel erfolgen.

- Bitseriell: Pro Taktperiode wird nur ein Bit addiert
- Bitparallel: In einer Taktperiode werden alle Bits addiert.

Wird die Dualzahl seriell bzw. parallel addiert, so wird das Bauelement Serienaddierer bzw. Paralleladdierer. Bei den Paralleladdierer unterscheidet man zwischen den folgenden Methoden.

- Paralleladdierer in der Normalform
- Ripple-Carry Addierer
- Carry-Look-Ahead Addierer

Jedes dieser Paralleladdierer haben ihre Vor- und Nachteile.
Beim Paralleladdierer in der Normalform wird die Normalform für jedes Ausgangsbit verwendet. Der Vorteil davon ist, dass durch die Normalform das Bauelement eine sehr geringe Propagation Delay hat. Die Normalform bringt aber auch Nachteile, da diese eine sehr komplizierte kombinatorische Schaltung ist.
Im Vergleich zum Paralleladdierer in der Normalform ist der Ripple-Carry Addierer um einiges langsamer, da es die Bits nacheinander durch ein Halb- und mehrere Volladdierer addiert. Der Vorteil ist, dass die kombinatorische Schaltung einfacher zu realisieren ist.



[Luisier, 2024]
Der Carry-Look-Ahead Addierer ist ähnlich wie eine Ripple-Carry Addierer mit dem Unterschied, dass der parallel, während die Bits addiert werden der Carry-Out addiert wird und somit der Carry-Look-Ahead Addierer ein wenig schneller ist als der Ripple-Carry Addierer. Der Nachteil davon ist, dass die kombinatorische Schaltung immer komplizierter wird, wenn die Dualzahl grösser wird.

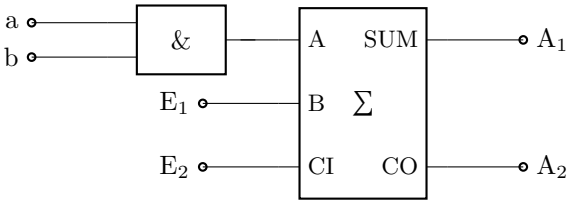
7.3 Hardware Multiplizierer

7.3.1 Rechenregeln und Grundprinzipien

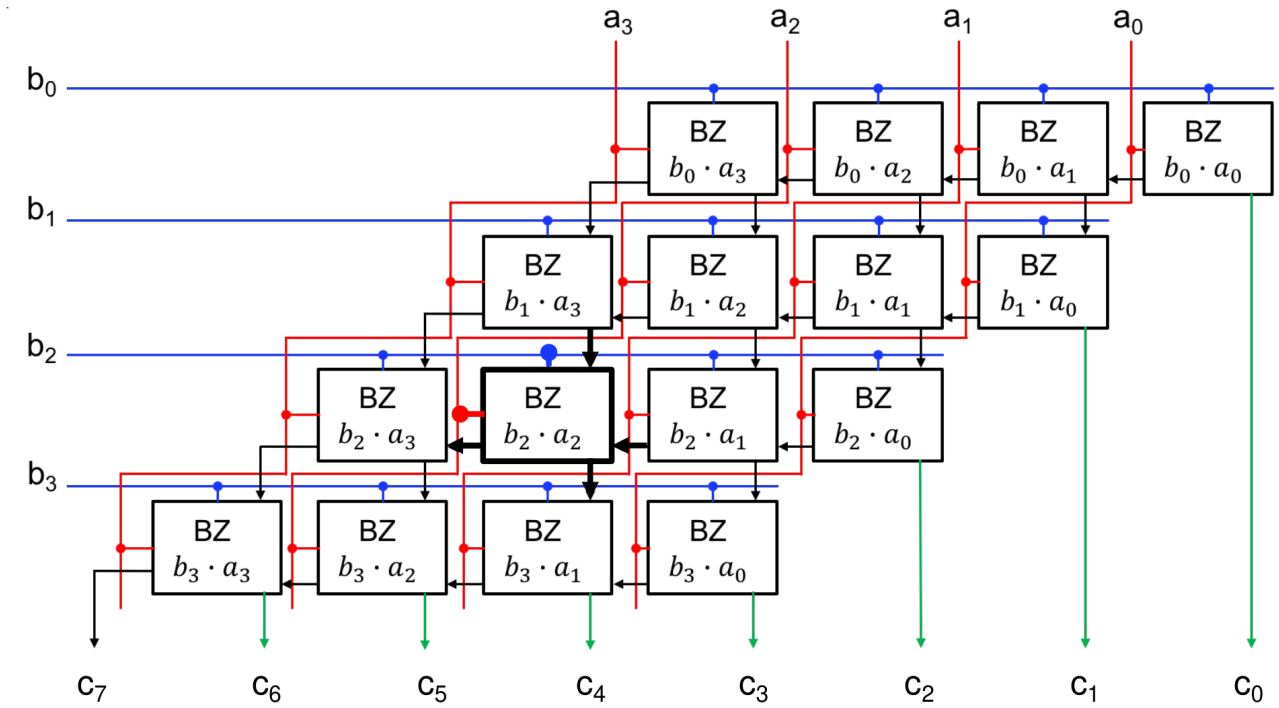
Die Multiplikation von Bits entspricht das logische AND. Doch wie geht man vor? Grundsätzlich geht man vor wie wenn man Zahlen im Dezimalsystem schriftlich multipliziert.

1. Den Multiplikand mit dem LSB vom Multiplikator multiplizieren und aufschreiben.
2. Für die Codierung darunter eine Null als LSB einfügen. Für jede weitere Zeile eine Null mehr.
3. Den ersten Schritt mit den nächsten Bit wiederholen.

Ein Multiplikator Bauelement besteht aus mehreren Basiszellen. Diese bestehen aus einem AND und einem Volladdierer.



Ein 4-Bit Multiplikator sieht zusammen mit allen Basiszellen wie folgt aus.



[Luisier, 2024]

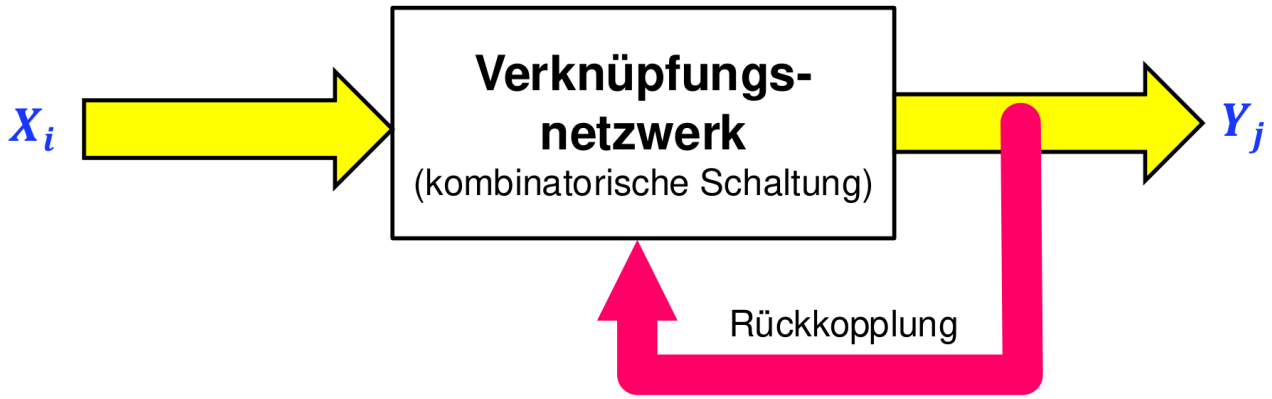
Kapitel 8

Latches und Flipflops

8.1 Grundlagen der Latches

8.1.1 Rückkopplungsprinzip

In den letzten paar Kapitel haben wir hauptsächlich kombinatorische Schaltungen untersucht. Kombinatorische Schaltungen nehmen mehrere Eingänge und kombinieren sie durch logische Bauelemente zu einem Ausgang. Im Gegensatz zu kombinatorische Schaltungen können sequentielle Schaltungen ihren Ausgang als Eingang wiederverwenden. Dieser wird auch Rückkopplung genannt.

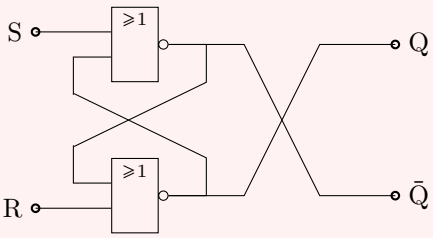


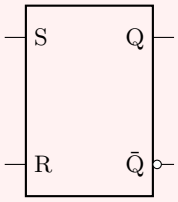
[Luisier, 2024]
Dieses Prinzip der Rückkopplung ist eine essenzielle Basis für Bauelemente, deren Aufgabe es ist Informationen zu speichern. Diese fungieren meistens als Kippschaltungen (Latches). Wir unterscheiden zwischen zwei Verschiedene Latches.

1. SR-Latch (Set/Reset Latch)
2. D-Latch (Data Latch)

Definition 8.1.1: SR-Latch

Der SR-Latch ist ein Kippschalter, welcher 2 Eingänge hat und 2 Ausgänge, wobei einer der Ausgänge die Inversion des anderen ist. Eines der Eingänge setzt den Zustand des Ausgangs auf 1 während der andere Eingang den Ausgang auf 0 setzt (zurücksetzt). Werden beide Eingänge nicht beschaltet, so bleiben die Eingänge in ihren vorherigen Zustand.



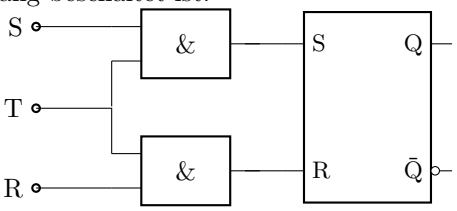


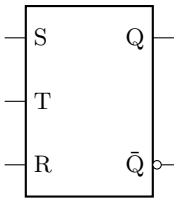
S	R	Q_{n+1}	$\bar{Q}_{n+1}$
0	0	Q_n	$\bar{Q}_n$
0	1	0	1
1	0	1	0
1	1	X	X

Bemerkung:-

Bei einem SR-Latch dürfen die Eingänge nicht gleichzeitig beschaltet werden, da dies zu einem Fehler führt.

SR-Latches können auch taktgesteuert sein. Dies bedeutet, dass der Ausgang nur verändert wird, sobald der Takteingang beschaltet ist.





8.1.2 D-Latch (Data-Latch)

Definition 8.1.2: D-Latch

Der D-Latch ist ein Kippschalter, welcher 1 Eingang hat und 2 Ausgänge. Der Eingang ist in Vergleich zum SR-Latch allein zuständig für das Setzen und Rücksetzen des Ausgangs bzw. des invertierten Ausgangs. Wie beim SR-Latch können D-Latches auch Taktgesteuert sein.

Taktgesteuerte Latches haben jedoch einen Nachteil. Sie sind sehr empfindlich zu Störungen. Deswegen eignet es sich eher, Flip-Flops zu verwenden, welche Taktflankengesteuert sind.

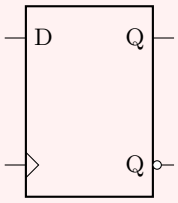
8.2 Einführung in die Flipflops

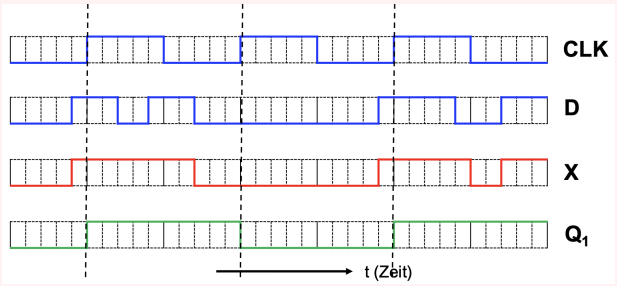
8.2.1 SR- und D-Flipflops

In Vergleich zu taktgesteuerten Latches ändern taktflankengesteuerte Latches (Flipflops) den Zustand ihres Ausgangs, nur wenn der Eingang beschaltet wird und wenn das Taktsignal seinen Zustand ändert (entweder $T = 0 \longrightarrow 1$ oder $T = 1 \longrightarrow 0$). Ändert sich das Taktsignal nicht, so ändert sich der Zustand des Ausgangs nicht.

Definition 8.2.1: D-Flipflops

D-Flipflops haben die gleiche Funktion wie D-Latches. Der einzige Unterschied ist, dass D-Flipflops taktflankengesteuert sind.

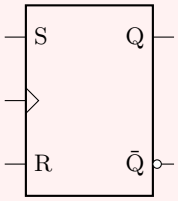


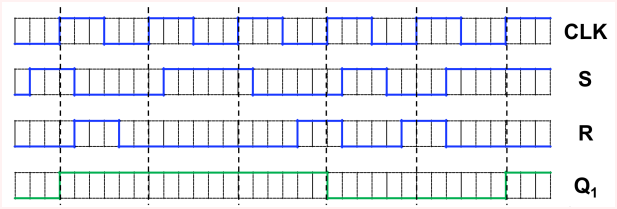


[Luisier, 2024]

Definition 8.2.2: SR-Flipflops

Wie beim D-Flipflop haben SR-Flipflops die gleiche Funktion wie SR-Latches mit dem Unterschied, dass SR-Flipflops taktflankengesteuert sind.





[Luisier, 2024]

8.2.2 Dynamik von Flipflops

Wie bei logischen Gattern sind Flipflops limitiert durch Verzögerungszeiten.

- t_{pd}: Verzögerungszeit → misst die Verzögerungszeit zwischen einer aktiven Taktflanke am Clock und ihrer Reaktion am Ausgang des Flipflops.
- t_s: Setup-Zeit → bestimmt wie lange ein Daten-Signal (hier D) vor der aktiven Taktflanke unverändert anliegen muss, um sicher in das Flipflop übernommen zu werden.
- t_h: Haltezeit → gibt an, wie lange ein Daten-Signal nach der aktiven Taktflanke unverändert anliegen muss, um sicher in das Flipflop übernommen zu werden.

[Luisier, 2024]
Die Verzögerungszeiten von Flipflops beeinflusst die Frequenz vom Taktsignal. Ist die Frequenz zu hoch, so kann der Flipflop nicht rechtzeitig schalten. Die maximale Taktfrequenz eines Flipflops wird meistens durch den Hersteller gegeben.

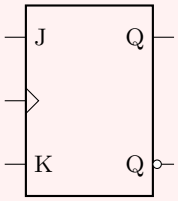
8.3 Mehr über Flipflop Grundlagen

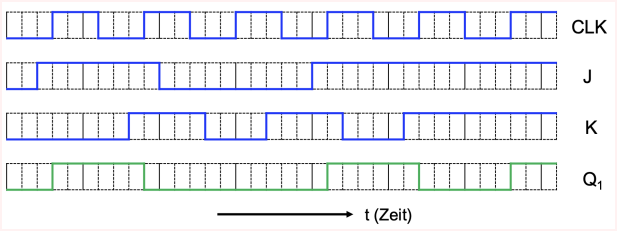
8.3.1 JK und T-Flipflops

Definition 8.3.1: JK-Flipflops (Jump-Kill Flipflops)

Der JK-Flipflop ist eine Erweiterung des SR-Flipflops. Beim SR-Flipflop war das Problem, dass beide Eingänge nicht gleichzeitig beschaltet werden dürfen, da dies zu einem Fehler führt. Dieses Problem existiert nicht mehr, da JK-Flipflops einfach die Ausgänge invertiert (Toggeln).

J	K	Q_{1n+1}	Q_{2n+1}
0	0	Q_{1n}	Q_{2n}
0	1	0	1
1	0	1	0
1	1	$\overline{Q_{1n}}$	$\overline{Q_{2n}}$





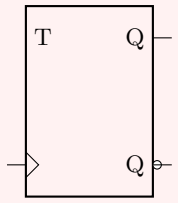
[Luisier, 2024]

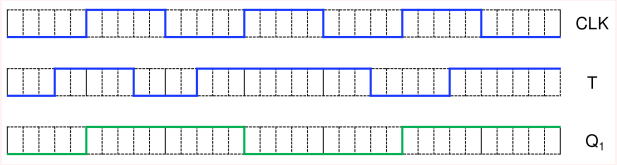
Bemerkung:-

JK-Latches existieren auch.

Definition 8.3.2: T-Flipflops (Toggle Flipflops)

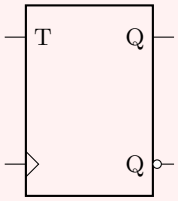
Beim T-Flipflop gibt es zwei Varianten.
Bei der ersten Variante hat der T-Flipflop nur den Takteingang. Die Aufgabe dieses Flipflops ist es, den Zustand vom Ausgang zu invertieren, sobald das Taktsignal seinen Zustand ändert.

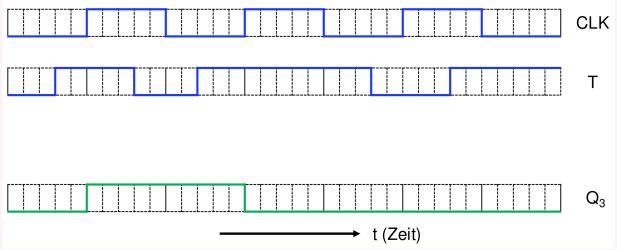




[Luisier, 2024]

Bei der zweiten Variante hat der T-Flipflop 2 Eingänge. Bei diesem Flipflop wird der Ausgang invertiert, sobald der Toggle Eingang beschaltet ist und das Taktsignal seinen Zustand ändert.





[Luisier, 2024]

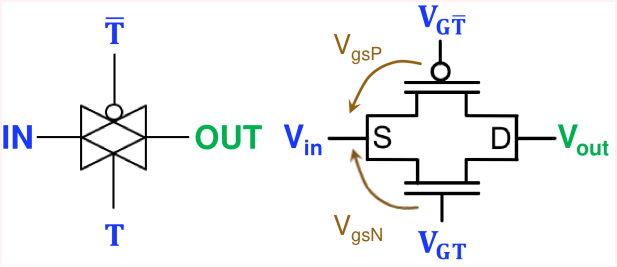
8.3.2 D-Flipflops in CMOS Technik

Um D-Flipflops durch Transistoren realisieren zu können, benötigen wir Transmission Gates.

Definition 8.3.3: Transmission Gates

Transmission Gates sind Bauelemente, welche für den Aufbau von Logikgattern benötigt wird. Sie bestehen aus einem PMOS Transistor und einem NMOS Transistor. Dadurch können sie High- als auch Low-Pegel übertragen. [Vasylyev, 2024]

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

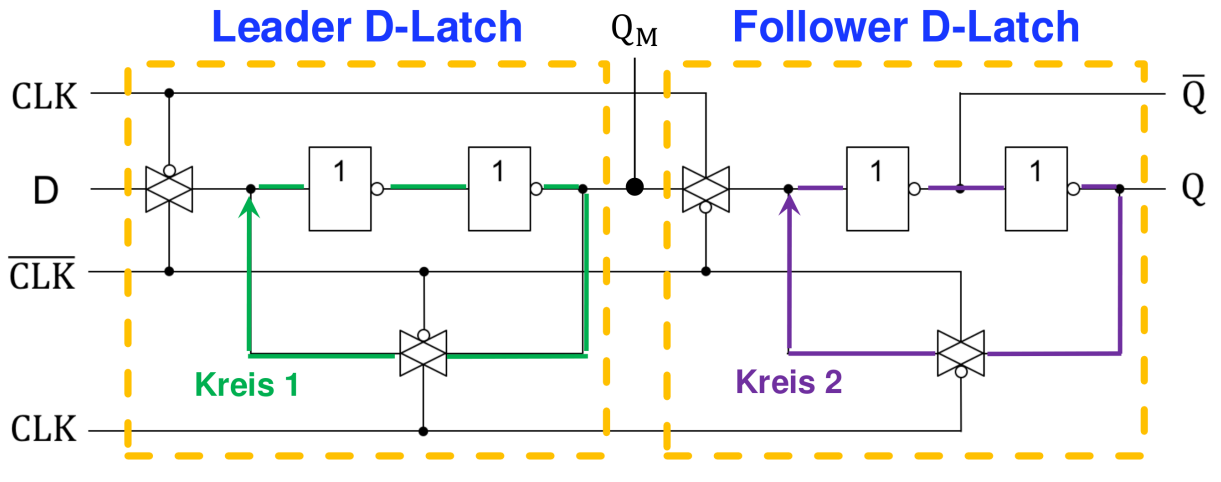


[Luisier, 2024]

Das Verhalten von Transmission Gates hängt von der Eingangsspannung V_{in} ab. Ist das Taktsignal beschaltet und die Eingangsspannung < 0, so ist die Gate-Source Spannung V_{gs} des PMOS Transistors $-V_{\text{in}}$ und der PMOS Transistor leitet. Dies ist anders beim NMOS Trasistor, welcher gesperrt bleibt. Wenn $V_{\text{in}} > 0$ dann ist es die umgekehrte Situation und der NMOS Transistor leitet.

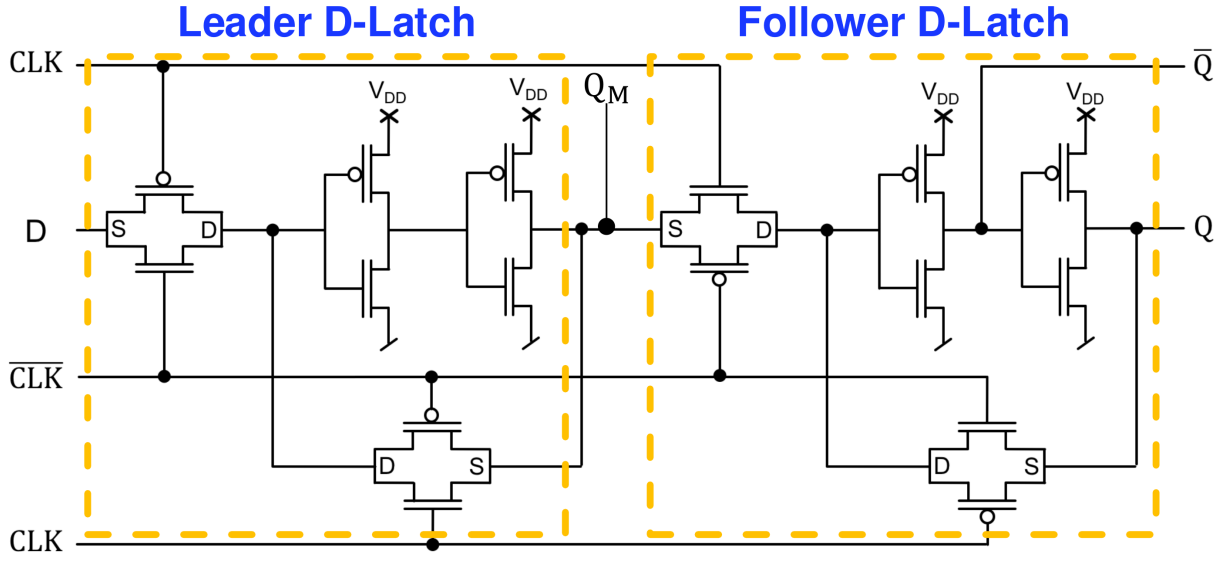
Definition 8.3.4: D-FlipFlops in CMOS Technik

Um ein D-Flipflop mit Transmission Gates zu realisieren, benötigt man vier Transmission Gates und vier Inverter bzw. zwei Transmission Gates und zwei Inverter pro Latch.



[Luisier, 2024]

Mit CMOS Transistoren würde die vorherige Schaltung wie folgt aussehen.

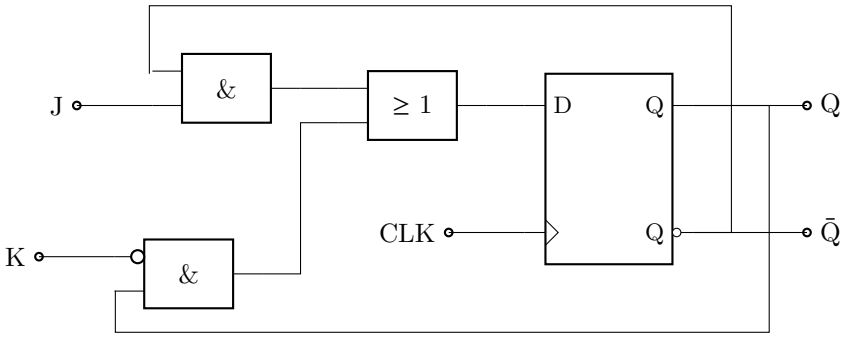


[Luisier, 2024]

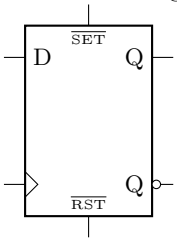
⇒ Für ein D-Flipflop werden acht PMOS- und acht NMOS-Transistoren benötigt.

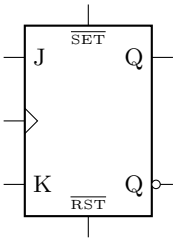
8.3.3 Weitere Flipflop Eigenschaften

Ein JK-Flipflop kann immer durch ein D-Flipflop ersetzt werden. Neben den D-Flipflop selbst werden zwei AND und ein OR zusätzlich benötigt.



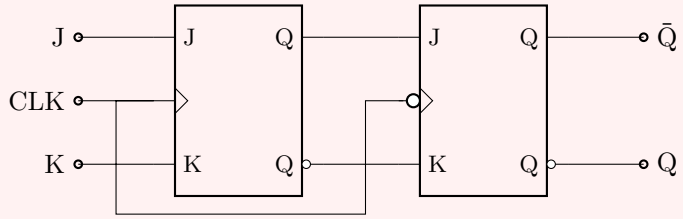
Flipflops können zusätzlich noch einen zusätzlichen Setz- bzw. Rücksetzkontakt haben. Dieser tut den Ausgang, egal in welchen Zustand das Taktsignal oder die Eingänge sind, setzen bzw. rücksetzen.

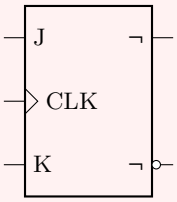




Definition 8.3.5: JK-LF-Flipflops (Jump-Kill-Leader-Follower Flipflops)

JK-LF-Flipflops (Zwischenspeicher Flipflops) sind in der Lage Signale zwischenzuspeichern und erst bei der nächsten fallenden Taktflanke zum Ausgang weiterzuleiten.

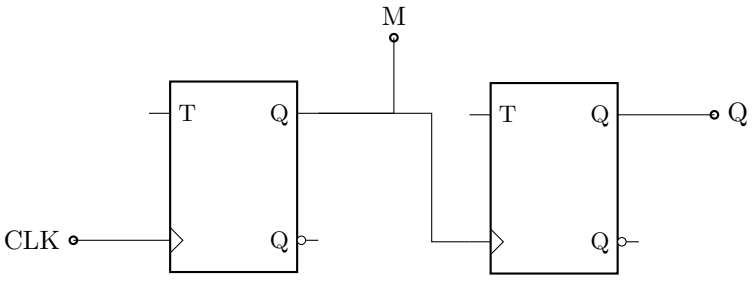




8.4 Schaltungen mit Flipflops

8.4.1 Frequenzteiler und Zähler

Mithilfe von T-Flipflops kann die Frequenz vom Taktsignal verringert werden. Jeder T-Flipflop verringert die Frequenz um den Faktor 2. Dies bedeutet mit n Flipflops wird die Frequenz um den Faktor 2^n verringert.



Mit zwei T-Flipflops kann auch ein zwei Bit Zähler erstellt werden. Dazu wird der Ausgang vom ersten Flipflop für den LSB verwendet und der Ausgang vom zweiten Flipflop als MSB verwendet. Die höchste Zahl hängt von der Anzahl Flipflops ab. Grundsätzlich kann man sagen, dass n Flipflops von 0 bis $2^n - 1$ zählen kann.

Kapitel 9

Automaten

9.1 Was sind Automaten?

9.1.1 Definition und Grundlagen

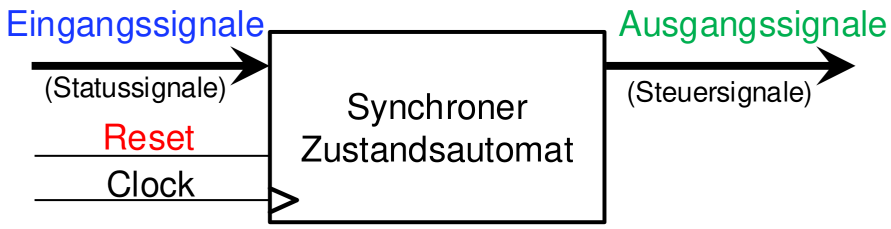
Definition 9.1.1: Automat

Ein Automat beschreibt ein System, welches auf sein Eingang reagiert und ein Ausgang produziert, der von dem Eingangssignal und von dem momentanen Zustand des Systems abhängt. [Luisier, 2024] Ein typisches Beispiel für ein Automaten sind Schaltwerke.
Wenn wir von Automaten sprechen, reden wir hauptsächlich von endlichen Automaten. Endliche Automaten haben nur die möglichkeit endlich viele Eingänge, Ausgänge und gespeicherte Zustände zu haben. Unendliche Automaten gibt es nicht.

Bei den Automaten unterscheiden wir zwischen synchrone und asynchrone Automaten.

Synchrone Automaten

Bei synchrone Automaten haben alle Speicherelemente (Flipflops) das gleiche Taktsignal. Daraus folgt, dass interne Zustandsänderungen synchron mit dem Taktsignal laufen.



[Luisier, 2024]

Asynchrone Automaten

In vergleich zu synchrone Automaten haben die Speicherelemente in asynchrone Automaten nicht das gleiche Taktsignal. Eine Zustandsänderung wird durch die Eingangssignale initiiert.

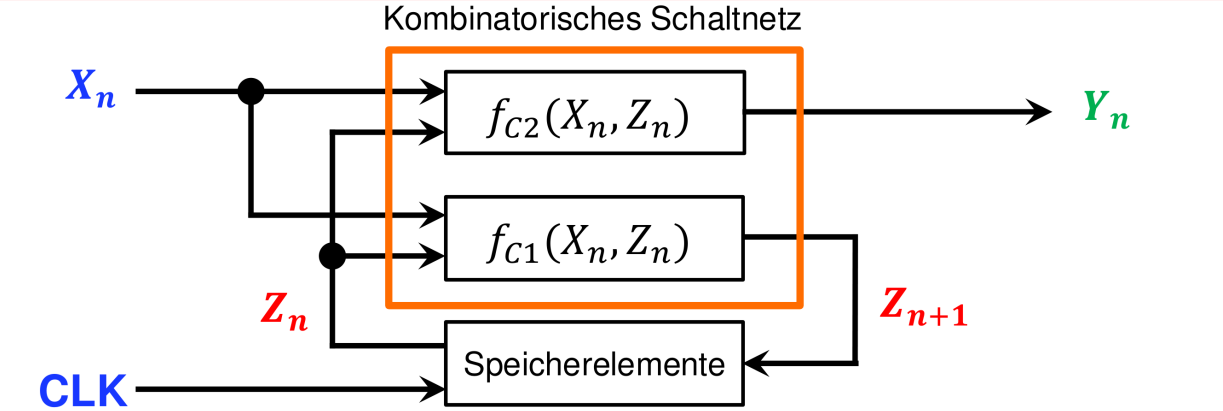
Ein endlicher Automat wird durch ein 6-Tupel charakterisiert:

$X = (x_1, x_2, ..., x_e)$	Eingabealphabet mit e Eingängen x_i, die durch binäre Eingangsvariablen $\{0, 1\}$ repräsentiert werden.
$Y = (y_1, y_2, ..., y_b)$	Ausgabealphabet mit b Ausgängen y_i, die als Bits mit Wert $\{0, 1\}$ dargestellt werden.
$Z = (z_1, z_2, ..., z_m)$	Zustandsmenge mit m inneren Zustansvariablen z_i, die einen Wert z_i, die einen Wert $\{0, 1\}$ haben können. Insfesamt gibt es $n_m = 2^m$ Zustände.
$Z_0 \in Z$	Anfangszustand
$f_{c1} : (X_n, Z_n) \rightarrow Z_{n+1}$	Übergangs-, Überführungsfunktion
$f_{c1} : (X_n, Z_n) \rightarrow Y_n$	Ausgangs-, Ausgabefunktion

9.1.2 Automatentypen: Mealy vs. Moore

Definition 9.1.2: Mealy Automat

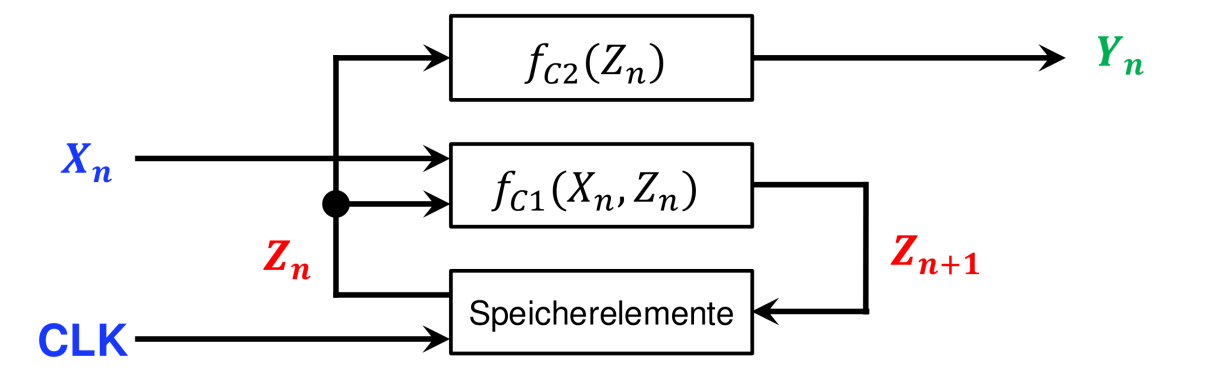
Ein Mealy Automat ist ein endlicher Automat, dessen Ausgang abhängig ist vom Eingang X_n, sowie den momentanen Zustand Z_n. Der Folgezustand Z_{n+1} ist abhängig vom momentanen Zustand sowie dem Eingang.



[Luisier, 2024]

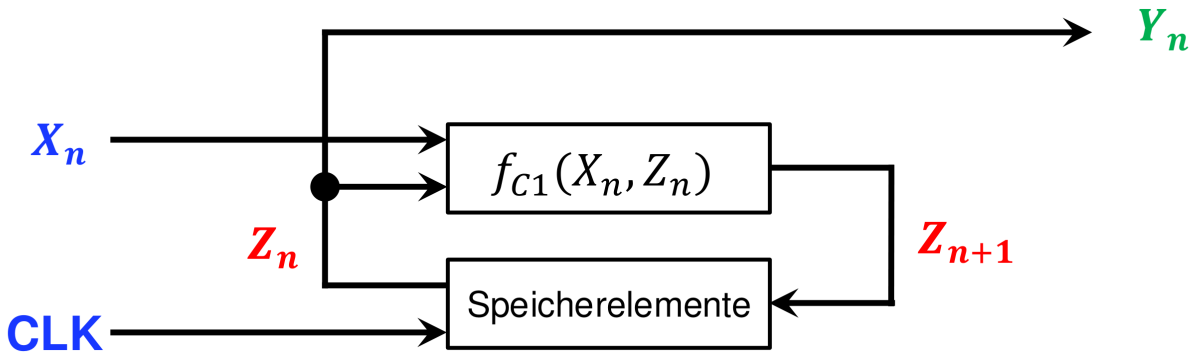
Definition 9.1.3: Moore Automat

Im Vergleich zum Mealy Automat ist der Ausgang vom Moore Automat nur abhängig vom momentanen Zustand Z_n. Der Folgezustand Z_{n+1} ist wie beim Mealy Automat abhängig vom momentanen Zustand, sowie dem Eingang.



[Luisier, 2024]

Eine weitere Art von Automat ist der Medwedjew Automat. Dieser unterscheidet sich vom Moore Automat dadurch, dass sein Ausgang gleich dem momentanen Zustand ist. Der Folgezustand ist wie beim Mealy Automat abhängig vom momentanen Zustand und dem Eingang.



[Luisier, 2024]

Bemerkung:-

Medwedjew Automaten haben eine grosse Relevanz für Zähler.

9.2 Entwurf und Analyse von Automaten

9.2.1 Zustandsdiagramm, Folgezustandstabelle

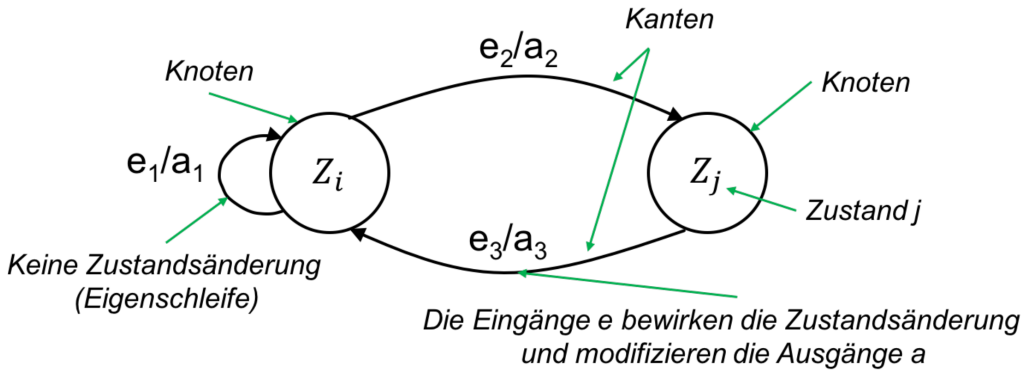
Automaten können auf verschiedenen Weisen beschrieben werden.

- Ausgangs- und Übergangsfunktionen
- Zustandsfolgetabellen (Folgezustandstabelle)
- Zustandsdiagramme oder Zustandsgraphen
- Karnaugh Diagramme

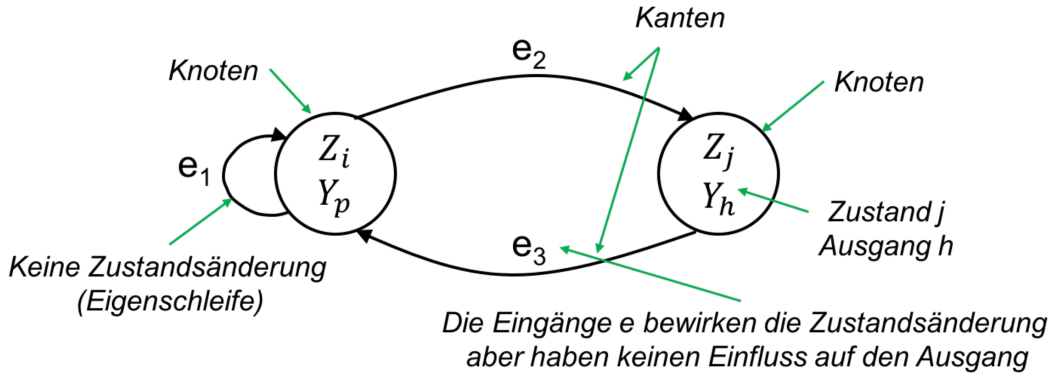
In den meisten Fällen fängt man mit dem Zustandsdiagramm, da dieser sehr schnell generiert werden kann. Aus dem Zustandsdiagramm kann die Folgezustandstabelle konstruiert werden. Wenn diese vorhanden ist, kann ein Karnaugh-Diagramm für jede innere Zustandsvariable und für jeden Ausgang aufgesetzt werden. Damit werden die Gleichungen der Ausgangs- und Übergangsfunktionen minimiert. Am Ende wird ein Schaltwerk erzeugt. [Luisier, 2024] Es ist aber auch möglich, dass der umgekehrte Fall auftritt.

Definition 9.2.1: Zustandsdiagramm

Ein Zustandsdiagramm ist eine graphische Darstellung der Funktion von Automaten, die aus Knoten und gericheteten Kanten besteht.



[Luisier, 2024]
Die Knoten bezeichnen die internen Zustände, die Kanten den Übergang zwischen zwei Zuständen. Die Eingangskombination e, die die Zustandsänderung bewirkt, und der Ausgang a werden an der jeweiligen Kante vermerkt. [Luisier, 2024]



[Luisier, 2024]
Wichtig zu erwähnen ist, dass bei Moore Automaten die Knoten sowohl die internen Zustände als auch die Ausgänge bezeichnen.

Definition 9.2.2: Folgezustandstabelle

Folgezustandstabellen zeigen alle möglichen Kombinationen eines Automaten an. Die Anzahl Reihen einer Folgezustandstabelle ist 2^{e+m}, wobei e die Anzahl Eingangsbits und m die Anzahl Zustandsbits ist. Die Anzahl Spalten beträgt $e + 2m + b$, wobei b die Anzahl Ausgangsbits ist.

9.2.2 Eingangs-, Ausgangs-, Zustandsvariablen

Um einen Automaten zu entwerfen kann man das folgende Kochrezept befolgen.

1. Zustandsmenge bestimmen, daraus folgt die Anzahl der Zustandsvariablen und der erforderlichen D-Flipflops
2. Definition der Ein- und Ausgangsvariablen, Kodierung
3. Darstellung der Zustandsfolge in einem Zustandsdiagramm
4. Aufstellung der Zustandsfolgetabelle
5. Bestimmung der minimierten Ausgangs- und Übergangsfunktionen mit der Hilfe von KV-Diagrammen

6. Prüfung auf unbenutzte Zustände
7. Konstruktion des Schaltplans anhand der Schaltfunktion

[Luisier, 2024]

9.3 Garagenautomat: Vertiefung

9.3.1 Mealy ↔ Moore Umwandlung

Mealy und Moore Automaten lassen sich **immer** ineinander umwanden. [Vasylyev, 2024] Wichtig dabei zu erwähnen ist, dass sich das zeitliche Verhalten der Eingangs- und Ausgangssignale sich ändert aufgrund von ihren verschiedenen Eigenschaften (Kapitel 9.1). Mealy Automaten sind sehr anfällig für Störungen, da ihr Ausgang abhängig ist vom Eingang und momentanen Zustand. Der Eingang ist nicht abhängig vom Taktsignal. Bei einer Störung am Eingang besteht die Möglichkeit, dass ein Ausgang beschaltet wird aber der Zustand nicht aktualisiert wird.

Um dieses Problem zu bekämpfen könnte man den Eingang mit dem Taktsignal Synchronisieren.

9.4 Fortgeschrittene Automateneigenschaften

9.4.1 Gekoppelte Automaten

In einigen Fällen ist es geeigneter komplexe Automaten aus mehreren Teilautomaten aufzubauen. Diese sind oft übersichtlicher.

[Luisier, 2024]

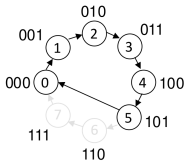
9.5 Zähler

9.5.1 Asynchronzähler

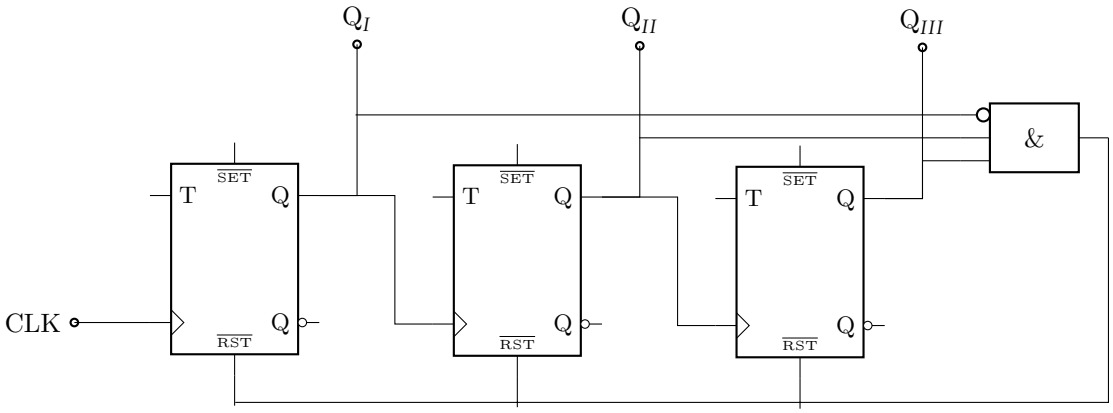
In Kapitel 8.4 haben wir gelernt, dass man T-Flipfops verketten kann, um n-Bit Dualzähler zu erstellen. Das Problem mit diesen sogenannten Asynchronzähler ist die Zeitverzögerung der T-Flipflops. Je mehr Bits der Dualzähler hat, desto mehr T-Flipflops haben wir und desto länger braucht der letzte T-Flipflop zum Schalten für den MSB. Deshalb müssen bei Asynchronzähler die maximale Taktfrequenz des Taktsignals berücksichtigt werden. Dies limitiert die Geschwindigkeit des Dualzählers.

9.5.2 Modulo-n (mod-n) Zähler

In manchen Fällen möchte man nicht das volle Pottential von Asynchronzähler verwenden. Man möchte nur bis zu einer bestimmten Dualzahl hochzählen. In diesem Fall kommen Modulo-n Zähler ins Spiel. Wie die n-Bit Dualzähler bestehen Modulo-n Zähler aus T-Flipfops mit dem Unterschied, dass die T-Flipflops einen Reset Eingang haben. Die Ausgänge der T-Flipflops sind mit einem AND Gatter verbunden. Sobald die Zahl $n+1$ am Ausgang liegt, werden die Reset-Eingänge von den Flipflops beschaltet und der Zähler fängt wieder bei 0 an. Soll der Zähler bei einer anderen Zahl anfangen zu zählen, so muss bei den Flipflops mit einem Set-Eingang gearbeitet werden.

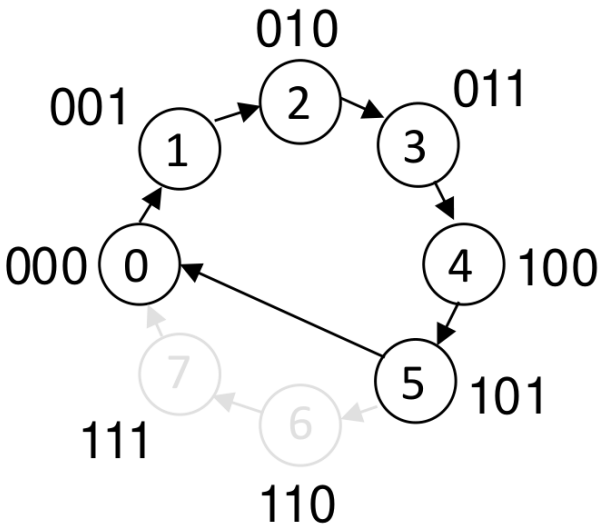


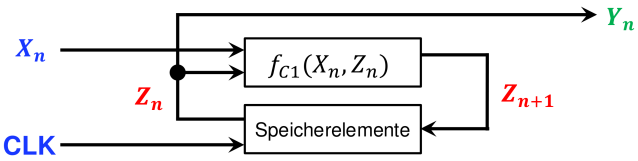
[Luisier, 2024]



9.5.3 Synchronzähler

Beim Asynchronzähler ist das Taktsignal der Flipflops immer vom vorherigen Ausgang des Flipflops abhängig (ausser beim ersten Flipflop). Im Gegensatz zu dem haben die Flipflops von Synchronzähler das gleiche Taktsignal und schalten gleichzeitig. Synchronzähler werden meistens als Medwedjew-Automaten implementiert.



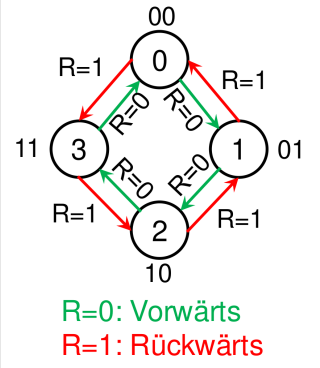


[Luisier, 2024]

[Luisier, 2024]

Bemerkung:-

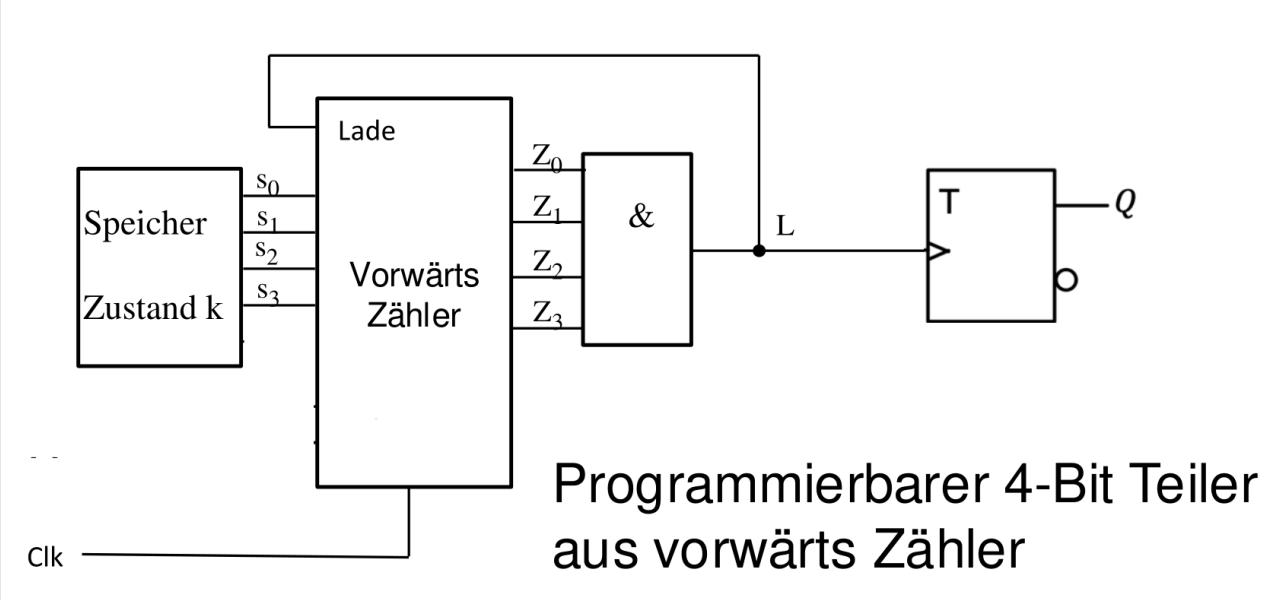
Rückwärtszähler können auch durch Medwedjew-Automaten realisiert werden. Es muss zusätlich ein weiterer Eingang hinzugefügt werden, welcher die Zählrichtung bestimmt.



[Luisier, 2024]

Bemerkung:-

In Kapitel 8.4 haben wir gesehen, dass T-Flipflops als Freqenzteiler fungieren können. Dies kann auch mit einem Medwedjew-Automaten und ein AND Gatter realisiert werden. Das AND Gatter ist mit einem T-Flipflop verbunden, welcher beschaltet wird, sobald die Zahl erreicht wird.



[Luisier, 2024]

Kapitel 10

Mikroprozessoren

10.1 Mikroprozessoren

10.1.1 Aufbau eines einfachen Rechners

Ein funktionierender Computer soll:

- logische Verknüpfungen von binären Zahlen durchführen können.
- Grundrechenarten beherschen können.
- Abläufe steuern können.
- sein Zustand andhand der Eingaben vom Benutzer ändern können.

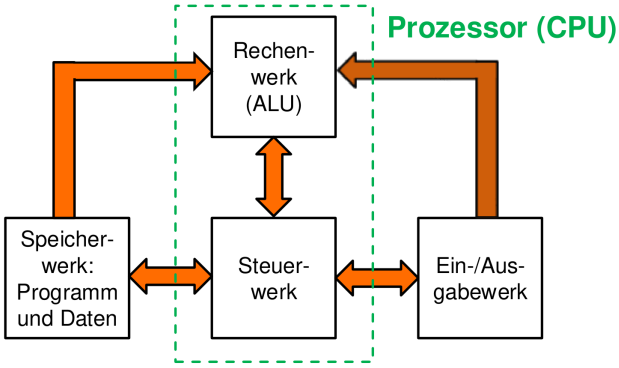
Aus diesem Grund benötigt ein Rechner

1. Ein Speicherwerk, um das auszuführende Programm sowie Daten (z.B. Zwischenergebnisse) ablegen zu können
2. Ein Rechenwerk (Arithmetic Logic Unit, ALU), um die arithmetischen und logischen Operationen durchzuführen
3. Ein Steuerwerk, um den Programmablauf und die Kommunikation zwischen den verschiedenen Blöcken zu regeln
4. Ein Ein-/Ausgabewerk, um mit der Aussenwelt kommunizieren zu können.

Die meisten universellen Rechner haben eine Grundstruktur nach der "von NeumannÄrchitektur. Der universelle Rechner durchläuft ein Zyklus bestehend aus fünf Teilschritten.

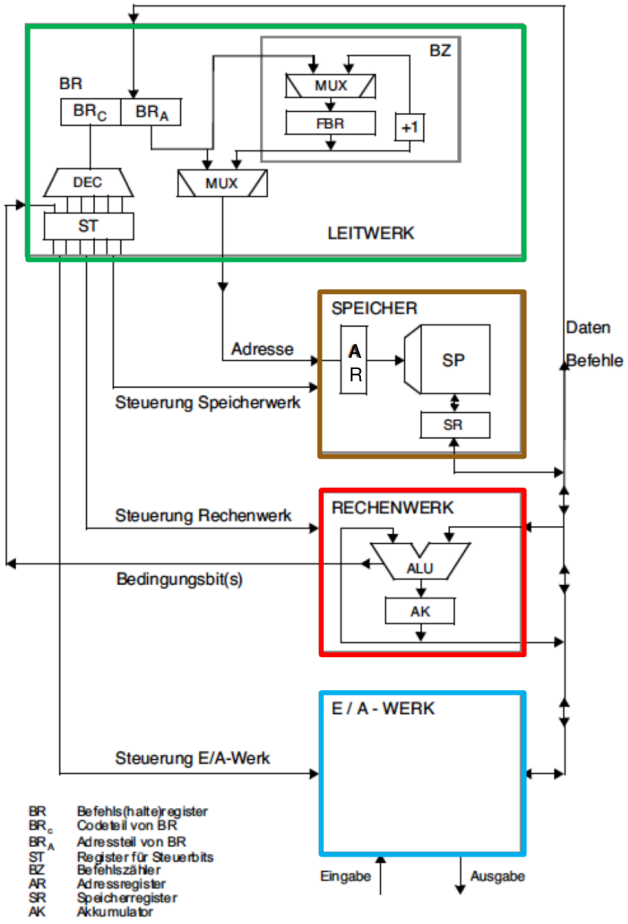
- FETCH (Befehlsabruf): das Steuerwerk holt aus dem Programmspeicher den nächsten Befehl.
- DECODE (Dekodierung): ein Befehlswort, das aus einem Instruktionsteil (‚was soll getan werden') und aus einer Datenadresse (‚welche Daten sollen manipuliert werden') wird durch das Steuerwerk in Schaltinstruktionen aufgelöst.
- FETCH OPERANDS (Operandenabruf): aus dem Speicher werden die Operanden geholt, die durch den Befehl verändert werden sollen oder als Parameter verwendet werden.
- EXECUTE (Befehlsausführung): eine arithmetische oder logische Operation wird vom ALU ausgeführt.
- WRITE BACK (Rückschreiben): das Ergebnis der Berechnung wird in Register oder Speicher zurückgeschrieben.

[Luisier, 2024]



[Luisier, 2024]

[



[Luisier, 2024]

[Luisier, 2024]

Literaturverzeichnis

[Luisier, 2024] Luisier, D. M. (2024). Vorlesungsfolien.

[The Manim Community Developers, 2024] The Manim Community Developers (2024). Manim – Mathematical Animation Framework.

[Vasylyev, 2024] Vasylyev, M. (2024). Übungen.