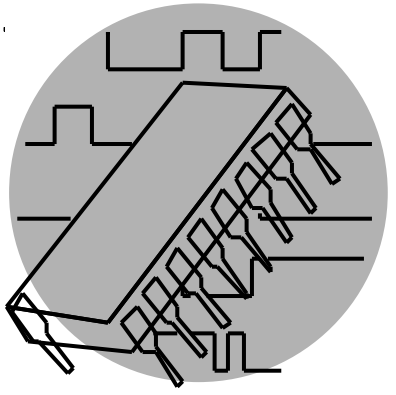
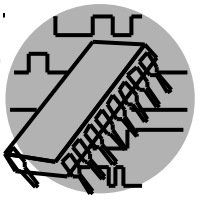


# Digital-Technik

Grundlagen

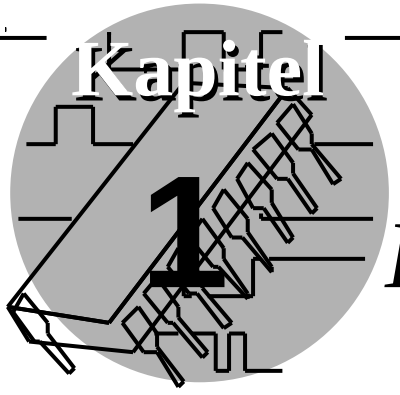
Ingenieurschule Biel, MicroLab - I3S, [v1.2]
Dr. M. Jacomet



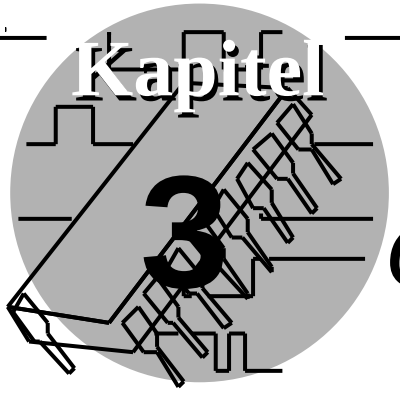
## Begleitende Literatur:

Es wird sehr empfohlen, eines der im folgenden aufgeführten Bücher neben dem Skript als Begleittext zu studieren:

- John P. Hayes, "Introduction to Digital Logic Design", Addison-Wesley Publishing Company, 1993, ISBN 0-201-15461-7
- L. Borucki, "Digitaltechnik", Verlag B. G. Teubner Stuttgart, 1989, ISBN 3-519-26415-3
- Mange, "Technique Numerique"




# Inhaltsverzeichnis




# Codierungen

## 3.1 Lernziele

Nach dem Studium des vorliegenden Kapitels sind die folgenden Aufgaben zu lösen oder die folgenden Fragestellungen präzise zu beantworten:

- Definition des Codes ist bekannt.
- Einsatzmöglichkeiten des Einschritt-Codes ist bekannt.
- Fehlererkennung und Fehlerkorrektur mittels Paritätsbit ist geläufig.
- Hamming-Code kann auf beliebige Wortbreite hergeleitet werden.

## 3.2 Einführung

Codes werden dazu verwendet, um Zahlen, Buchstaben und Zeichen in anderen Darstellungsformen zu verwenden. So repräsentieren unterschiedliche Codes die verschiedenen Darstellungsformen. Codes werden nach unterschiedlichen Gesichtspunkten entwickelt:

- geringer Speicherbedarf
- schelle, einfache Verarbeitung
- Vermeidung von Abtastfehlern, etc

Aufgrund der unterschiedlichen Zielsetzungen, existieren unmengen von Codes.

**Def:** Ein Code ist eine Vorschrift für die eindeutige Zuordung zwischen zwei Mengen.

## 3.3 Binär-Dezimal Codes

Binär-Dezimal Codes sind Codes, welche den Dezimal-Ziffern einen Code in im binären Zahlensystem zuweisen. Um die Ziffern 0 bis 9 binär zu codieren, werden 4 Bits benötigt. Hingegen können mit 4 Bits $2^4 = 16$ Zeichen dargestellt werden, was 6 mehr als benötigt ist. Um 10 Ziffern mit 4 Bits darzustellen, existieren prinzipiell die folgende Anzahl von Zuordnungsmöglichkeiten und damit auch unterschiedliche Code:

$$\frac{2^4!}{6!} = 2.9 \cdot 10^{10}$$

Die grosse Auswahlmöglichkeit erlaubt Codes nach verschiedenen Gesichtspunkten aufzustellen, wie:

- Bewertbarkeit: jeder Stelle des Binärzeichens ist eine feste Wertigkeit zugeordnet.
- Abtastsicherheit: Durch Fehler darf kein falscher Wert abgetastet werden.
- Übertragungssicherheit: Durch Fehler darf kein neues Zeichen entstehen.
- Rechenfähigkeit: Spezielle Codierungen erlauben gewisse Operationen rascher auszuühren.

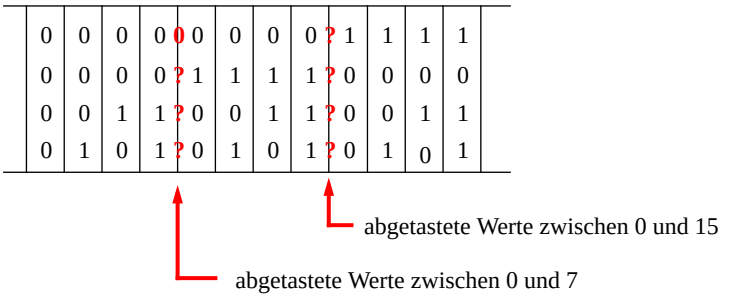
In Code-Tabellen finden sich diverse Codes. Die Nachfolgenden vier Code-Typen sind exemplarisch aufgeführt:

- Aiken, Stibitz Code: Diese Codes sind negationssymmetrisch (9er Komplement).
- 4-2-2-1 Code: Code verwendbar für schnelle Zähler.
- White Code: Dieser Code hat keine duale Wertigkeit.
- Gray, Glixon, O'Brien, Excess3: Dies sind Einschritt-Codes.

## 3.4 Einschritt-Codes

Mit Einschritt-Codes lassen sich Abtastfehler vermeiden. Sollen Strecken oder Winkel codiert werden, so benutzt man dazu Code-Stäbe oder Code-Scheiben. Beim Abtasten dieser Codiereinrichtungen kann es bei den Übergängen zu Fehlern kommen (siehe Figur 1).

In Figur 2 ist eine Code-Scheibe dargestellt, bei welcher mithilfe eines Einschritt-Codes (Gray Code) Abtastfehler vermieden werden können. Der Gray-Code ist für binär-dezimal Codierung über mehr als eine Dezimalstelle

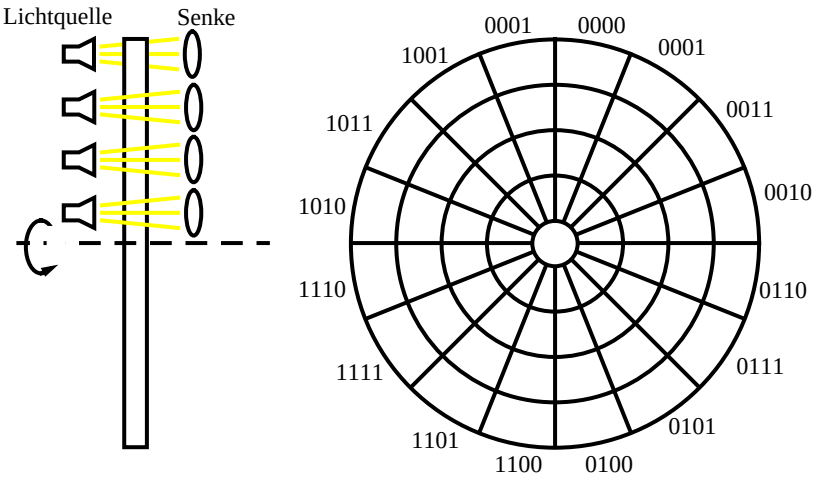



Figur 1: Abtastfehler

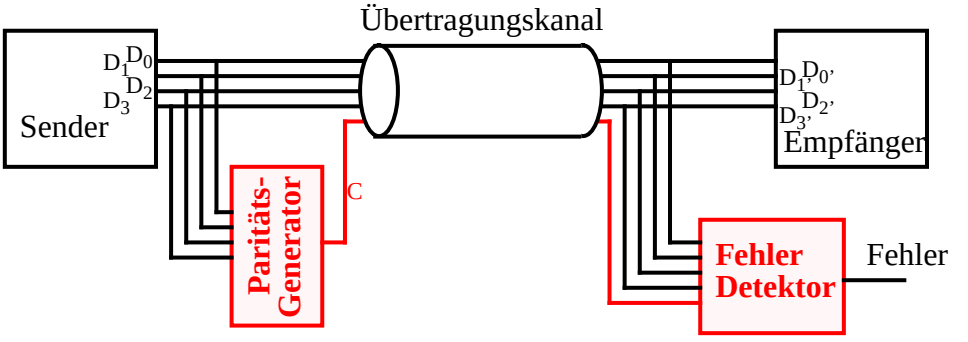
nicht geeignet, da sich die Codes für die Zeichen 0 und 9 in drei Stellen unterscheiden.

## 3.5 Fehlererkennung

Eine weiteres Kriterium für die Erzeugung eines Codes ist die Übertragungssicherheit, welche mit Redundanz erkauft wird. Redundanz im Code ermöglicht die Fehlererkennung oder sogar die Fehlerkorrektur. Fehlererkennbare Codes sind Codes, bei welchen immer ein “einfacher Fehler”, d.h. eine Verfälschung einer Bitposition von 0 auf 1 oder umgekehrt erkennbar ist.




Figur 2: Einschritt-Code




Figur 3: Paritätsbit für Fehlererkennung

### 3.5.1 Paritätskontrolle

Dei Paritätskontrolle ist die einfachste Art der Codesicherung. Die “1” eines Zeichens werden durch eine Zusatzstelle, dem sogenannten Paritätsbit, auf eine gerade (oder ungerade) Abzahl “1” ergänzt. Wird ein solches Paritätsbit einem Code-Wort zugefügt, so können bei diesem Verfahren Einzelfehler detektiert werden (siehe Figur 3).

Beim Paritätsbit können im betrachteten Fall die folgenden Fehler erkannt werden:

- Einzelfehler werden erkannt
- Doppelfehler werden nicht erkannt
- Dreifachfehler werden erkannt, etc

### 3.5.2 “m aus n” Code

Es stellt sich die Frage, inwiefern das Paritätsbit optimal für die Fehlererkennung bei einem BCD Code ist. Es wird deshalb eine neuer Code, der “m aus n” Code definiert. Bei diesem Code sind M von n Bits immer “1” gesetzt. Die Kontrolle erfolgt sehr einfach über die Quersumme der “1”, welche beim nachfolgend aufgestellten “2 aus 5” Code immer 2 sein muss (siehe Figur 4). Bei diesem Code werden die folgenden Fehler erkannt:

- Einzelfehler werden erkannt
- Doppelfehler werden erkannt, falls sie in die gleichen Fehler aufweisen.
- Dreifachfehler werden erkannt.

Wie beim Vergleich der beiden Codierungen Paritätsbit und “2 aus 5” Code ersichtlich ist, lassen sich mit dem letzteren noch mehr Übertragungsfehler detektieren.

# 3.6 Fehlerkorrektur

Sollen Fehler nicht nur erkannt, sondern auch korrigiert werden, so muss die Redundanz noch weiter erhöht werden.

Anhand eines 2-Bit breiten Informationswortes soll die Fehlerkorrektur mittels Paritätsbits ermittelt werden. Da ein einzelnes Paritätsbit dazu nicht ausreicht, ist ein Versuch mit 2 Paritätsbits vorzunehmen (Figur 5):

| Informationsbits | Paritätsbits | |
|---|---|---|
| $D_1$ $D_0$ | $C_1$ $C_0$ | |
| x | x | |
| x x | x | |
| 0 0 | 0 0 | |
| 0 1 | 1 1 | |
| 1 0 | 1 0 | |
| 1 1 | 0 1 | |
| 1 1 | 1 1 | Code mit Fehler<br>Wo ist der Fehler ? |

Figur 5: Versuch Fehlerkorrektur

Wie aus dem obigen Beispiel ersichtlich ist, braucht es mehr als 2 Paritätsbits um nur 2 Informationsbits zu überprüfen. Aus dem Hamming Code ist ersichtlich, wieviele Paritätsbits für eine bestimmte Anzahl Informationsbits notwending sind. In der nachfolgenden Figur 6 ist gezeigt, wie 3 Paritätsbits bei 4 Informationsbits Einfachfehler erkennen, lokalisieren und korrigieren können.

| | 7 4 2 1 0 |
|---|---|
| 0 | 1 1 0 0 0 |
| 1 | 0 0 0 1 1 |
| 2 | 0 0 1 0 1 |
| 3 | 0 0 1 1 0 |
| 4 | 0 1 0 0 1 |
| 5 | 0 1 0 1 0 |
| 6 | 0 1 1 0 0 |
| 7 | 1 0 0 0 1 |
| 8 | 1 0 0 1 0 |
| 9 | 1 0 1 0 0 |

Figur 4: “2 aus 5” Code

| $D_7$ | $D_6$ | $D_5$ | $C_4$ | $D_3$ | $C_2$ | $C_1$ |
|---|---|---|---|---|---|---|
| x | x | x | x | | | |
| x | x | | | x | x | |
| x | | x | | x | | x |

Figur 6: Hamming-Code

| $F_4$ | $F_2$ | $F_1$ | |
|---|---|---|---|
| 0 | 0 | 0 | kein Fehler |
| 0 | 0 | 1 | ($C_1$ ist falsch) |
| 0 | 1 | 1 | $D_3$ ist falsch |
| 1 | 0 | 0 | ($C_4$ ist falsch) |
| 1 | 0 | 1 | $D_5$ ist falsch |
| 1 | 1 | 0 | $D_6$ ist falsch |
| 1 | 1 | 1 | $D_7$ ist falsch |

Um Einfachfehler korrigieren zu können, sind die folgenden beiden Bedingungen beim Hamming-Code in Figur 6 notwendig:

- Jede Kolonne muss einen Eintrag haben.
- Kolonnen müssen unterscheidbar sein.

Mit den Variablen *k*, für Anzahl Paritätsbits und der Variablen *m*, der Anzahl Informationsbits lässt sich der folgende Zusammenhang finden:

$$m = 2^k - 1 - k$$

Somita lassen sich für eine beliebige Anzahl Informationsbits die entsprechende notwendige Anzahl Paritätsbits berechnen, damit eine Fehlerkorrektur von Einfachfehlern vorgenommen werden kann.

**Beispiel 1:** Bei den folgende fehlerbehafteten Hamming Codes (4 Informationsbits und 3 Paritätsbits, gerade Parität) sind die Einfachfehler zu lokalisieren und die korrekten Informationsworte zu erzeugen.

a) 0 0 1 0 1 0 0

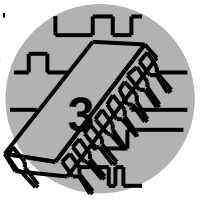
b) 1 0 0 0 1 0 0

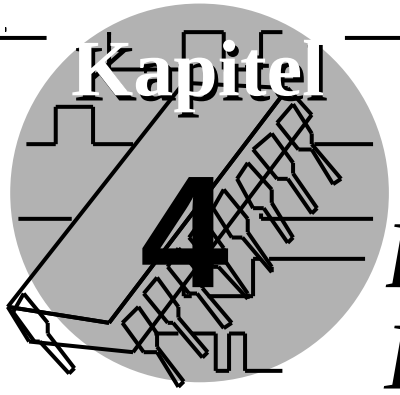
c) 0 0 0 1 1 0 0

## 3.7 Alphanumerische Codes

Alphanumerische Codes sind Codes für Buchstaben, Ziffern und Sonderzeichen. Ein Beispiel dafür ist der ASCII Code (american standard code of infor-

mation interchange), welcher speziell für den Datenaustausch zwischen Computern verwendet wird. Der ASCII Code weist 7 Bits auf. Das 8. Bit wird für länderspezifische Sonderzeichen oder als Paritätsbit verwendet.






# Boolesche Algebra und Logische Gatter

## 4.1 Lernziele

Nach dem Studium des vorliegenden Kapitels sind die folgenden Aufgaben zu lösen oder die folgenden Fragestellungen präzise zu beantworten:

- Unterschiede zwischen Boolescher- und klassischer Algebra.
- Welches sind Grund- und Spezial Funktionen der Booleschen Algebra.
- Rechenregeln der Booleschen Algebra können angewendet werden.
- Beliebige kombinatorische Schaltungen können analysiert werden.
- Ausgehend von Wahrheitstabellen können die disjunktiven oder konjunktiven Normalformen von Schaltungen hergeleitet werden.
- Schaltungen können mittels graphischer und algorithmischer Methode vereinfacht werden.

## 4.2 Boolesche Konstanten und Variablen

In der Digitaltechnik werden Bauelemente eingesetzt, welche ein binäres Verhalten aufweisen. Die Ausgangssignale dieser digitalen Bauelemente weisen somit nur zwei unterschiedliche Signalwerte auf. Der Mathematiker George Boole entwickelte 1847 auf der Basis dieser Zweiwertigkeit die nach ihm benannte Theorie der Booleschen Algebra. Später wurde diese Algebra von Shannon weiterentwickelt und zur Berechnung von logischen Schaltungen verwendet.

Die Boolesche Algebra unterscheidet sich somit prinzipiell von der gewöhnlichen Algebra durch ihre Konstanten und Variablen, welche nur zwei Werte annehmen können: “0” und “1”.

## 4.3 Grundfunktionen der Booleschen Algebra

Zusammenhänge zwischen den Variablen oder Konstanten bezeichnet man in der Booleschen Algebra analog zur gewöhnlichen Algebra ebenfalls als Funktionen. Allerdings existieren in der Booleschen Algebra keine kontinuierliche Funktionen, sondern es werden nur diskreten Wertepaaren Funktionswerte zugeordnet. Eine solche Zuordnung lässt sich einfach in einer Funktionstabelle oder *Wahrheitstabelle* definieren. In einer Wahrheitstabelle sind sämtliche Variablenkombinationen aufgeführt und jeder Kombination ist der entsprechende Funktionswert zugeordnet. In Figur 7 ist eine Wahrheitstabelle für zwei Eingangsvariablen $E_1$ und $E_2$ und einer Ausgangsvariablen $A$ dargestellt.

Die Wahrheitstabelle ist ein mögliches Mittel eine Funktion zu definieren. Eine weitere Variante ist die sogenannte *Boolesche Gleichung*, welche ähnlich wie die algebraische Gleichung für Berechnungen herangezogen werden kann.

Wie oben gezeigt, lassen sich mit Wahrheitstabellen und Booleschen Gleichungen Funktionen definieren. Es stellt sich nun die Frage was für Funktionen überhaupt in der Booleschen Logik existieren. Es kann einfach gezeigt werden, dass mit einer Variablen genau $2^{2^1}=4$, mit zwei Variablen $2^{2^2}=16$ Funktionen definiert werden können. Trotz dieser Vielfalt kann man mit nur wenigen Grundfunktionen sämtliche Funktionen definieren. Man unterscheidet zwischen drei wesentlichen Elementarfunktionen:

- Konjunktion: logische Multiplikation (logisches AND)
  Symbole $\cdot$ $\wedge$
- Disjunktion: logische Addition (logisches OR)
  Symbole $+$ $\vee$
- Komplement: logische Inversion (logisches NOT)
  Symbole $\overline{\phantom{x}}$

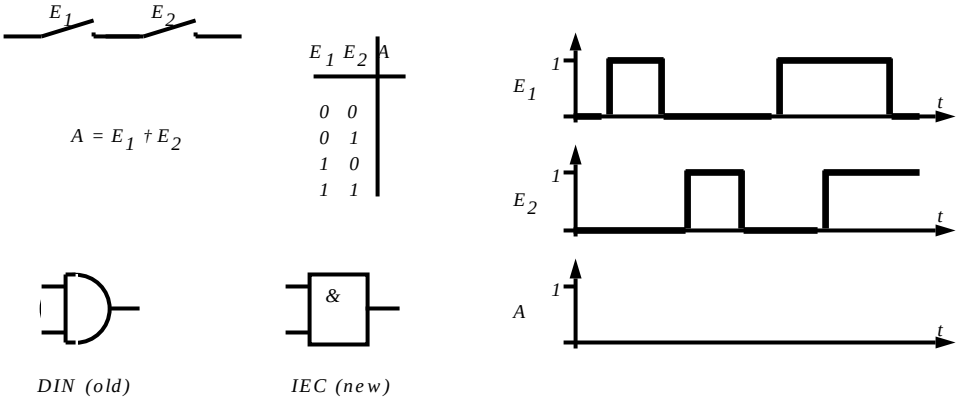
Prinzipiell reichen Komplement und Disjunktion oder Komplement und Konjunktion aus, um jede beliebige Boolesche Funktion zu beschreiben. Im

| $E_1$ | $E_2$ | $A$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

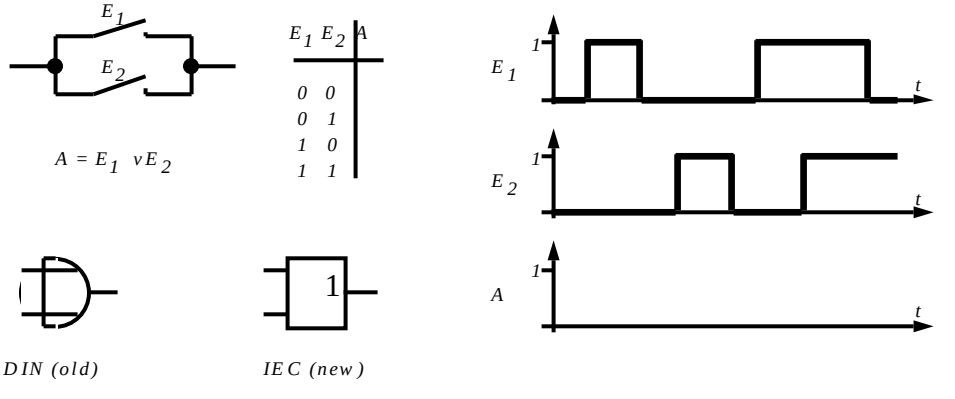
Figur 7: Wahrheitstabelle

nachfolgenden sind die drei Grundfunktionen in Form von Boleschen Gleichungen, Wahrheitstabellen, Schaltern und ihren Symbolen beschrieben.

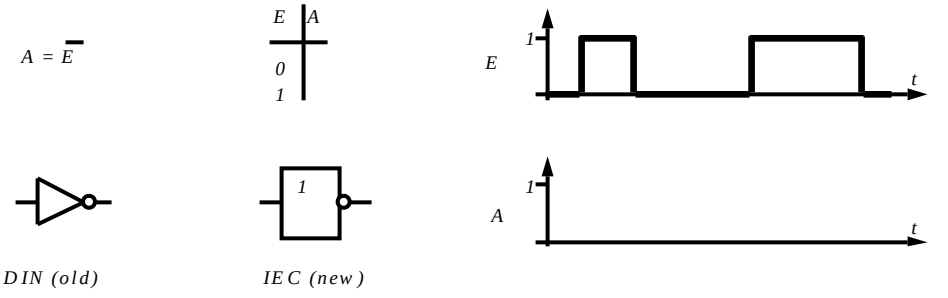
*Konjunktion*: (logisches AND)




*Disjunktion*: (logisches OR)



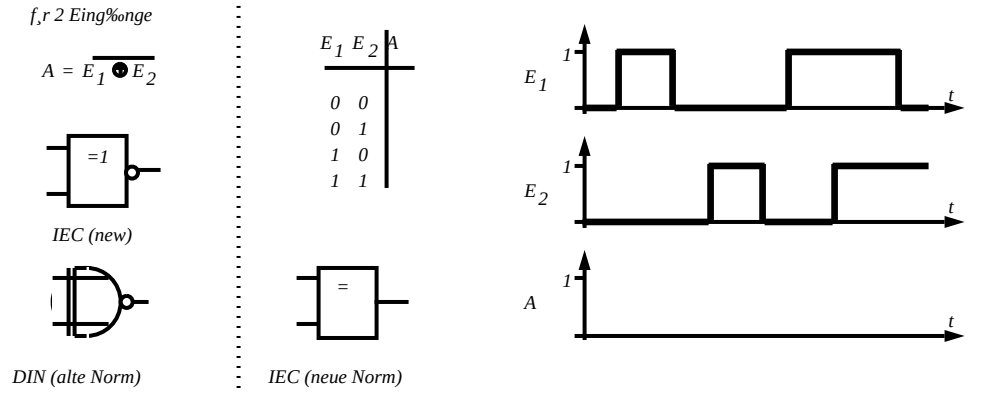
*Komplement*: (logisches NOT)

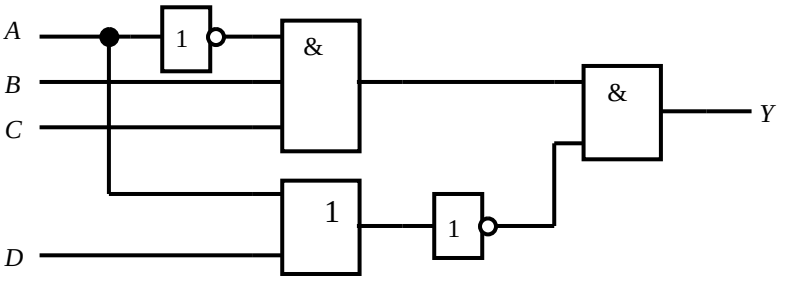



## 4.4 Spezielle Funktionen der Booleschen Algebra

Mit 2 Eingangsvariablen lassen sich in der Booleschen Algebra total 16 verschiedene Funktionen definieren. Neben den bereits erwähnten Grundfunktionen existieren deshalb noch einige weitere Booleschen Funktionen, wie zum Beispiel die Äquivalenz und Antivalenz Funktion.

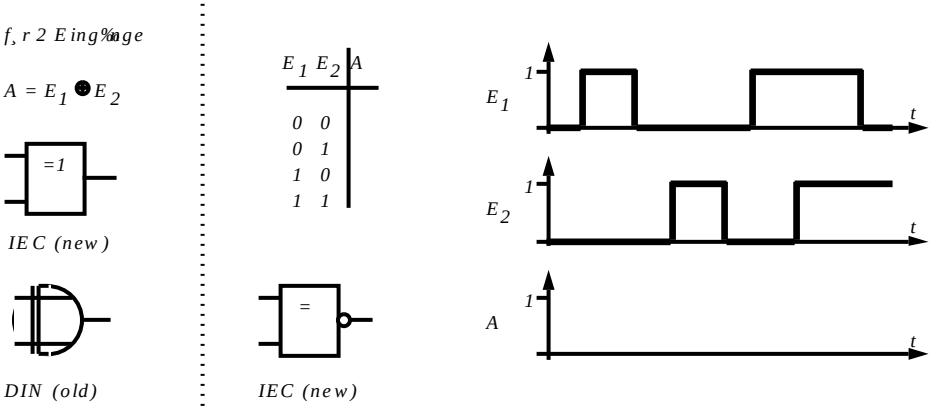
*Äquivalenz*: (Identität, bei zwei Eingängen: Exclusiv NOR, EXNOR)






Figur 8: Beispiel einer logischen Schaltung.

*Antivalenz*: (bei zwei Eingängen: exclusif OR, EXOR)



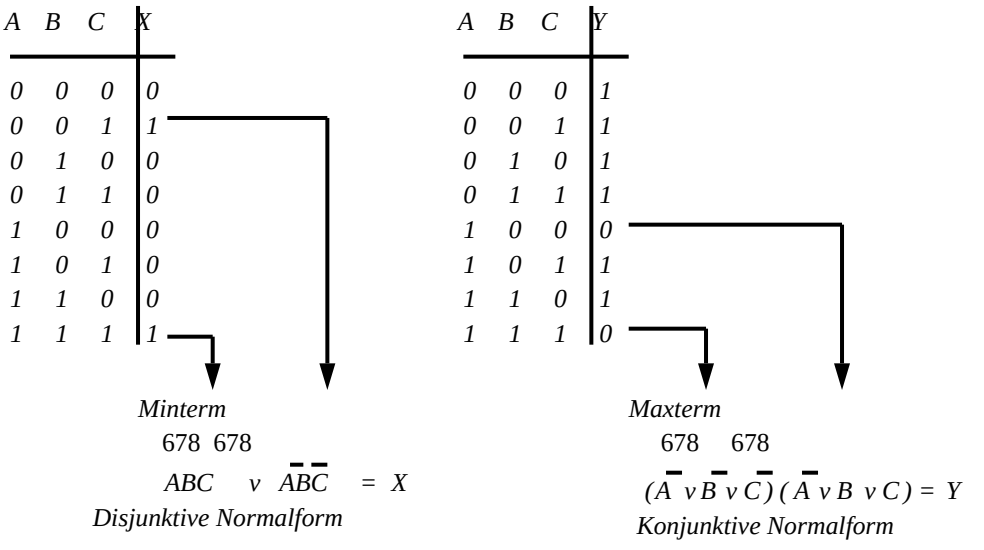

# 4.5 Analyse logischer Schaltungen

Kennt man das Logikschema oder auch die Boolesche Gleichung einer Schaltung, so lässt sich daraus der Wert des Ausganges in Abhängigkeit der Eingänge bestimmen. Als Beispiel sei das Schema der Schaltung in Figur 8 zu analysieren. Für einen gewählten Eingangsvektor (A,B,C,D)=(0,1,1,0) ist der Ausgangswert zu bestimmen. Ebenso lässt sich sehr einfach die Funktion der Schaltung in Form einer Booleschen Gleichung oder in Form einer Wahrheitstabelle finden.

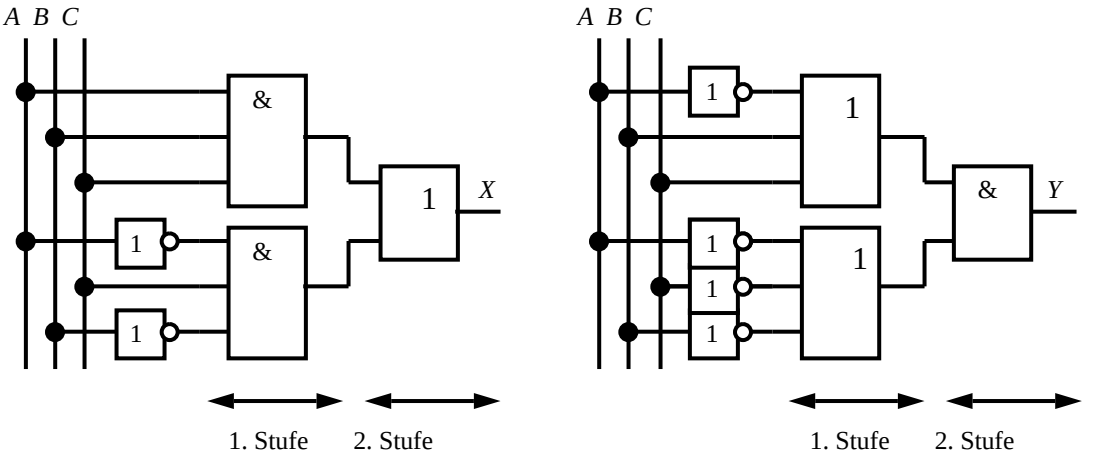
## 4.6 Disjunktive und Konjunktive Normalform

Aus der Algebra ist bekannt, dass Funktionen in unterschiedlichen Gleichungen dargestellt werden können. Gleichungen können erweitert, durch Klammerausdrücke verschachtelt werden etc. Ähnliches kann auch mit den Booleschen Gleichungen angestellt werden. Eine besonders nützliche und somit wichtige Darstellungsform der Booleschen Gleichungen sind die sogenannten disjunktive und konjunktive Normalformen. Hat man eine disjunktive Verknüpfung zwischen konjunktiven Termen, so spricht man von einer disjunktiven Normalform. Bei konjunktiver Verknüpfung zwischen disjunktiven Termen erhält man eine Konjunktive Normalform.

Ausgehend von der Wahrheitstabelle sollen anhand zweier unterschiedlicher Beispiele die beiden Normalformen der Booleschen Gleichungen systematisch hergeleitet werden.




Die so erhaltenen disjunktiven und konjunktiven Normalformen einer logischen Gleichung spielen eine besondere Rolle für das systematische vereinfachen von Verknüpfungsgleichungen. Belässt man die Booleschen




Gleichungen in diesen Normalformen, so erhält man daraus immer eine zweistufige Logik.

## 4.7 Rechenregeln der Booleschen Algebra

Die wichtigsten Rechenregeln der Booleschen Algebra seien ohne Kommentar in tabellarischer Form zusammengestellt.

| | | |
|---|---|---|
| Grundgesetzte: | (1) | $A \wedge A = A$ |
| | (2) | $A \vee A = A$ |
| Konstante: | (3) | $A \wedge 1 = A$ |
| | (4) | $A \wedge 0 = 0$ |
| | (5) | $A \vee 1 = 1$ |
| | (6) | $A \vee 0 = A$ |
| Komplemente: | (7) | $A \wedge \overline{A} = 0$ |
| | (8) | $A \vee \overline{A} = 1$ |
| | (9) | $\overline{\overline{A}} = A$ |
| De Morgan: | (10) | $\overline{A \wedge B} = \overline{A} \vee \overline{B}$ |
| | (11) | $\overline{A \vee B} = \overline{A} \wedge \overline{B}$ |

Kommutativ Gesetz: (12) $A \wedge B = B \wedge A$

(13) $A \vee B = B \vee A$

Distributiv Gesetz: (14) $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$

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

Assoziativ Gesetz: (16) $A \wedge (B \wedge C) = (A \wedge B) \wedge C$

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

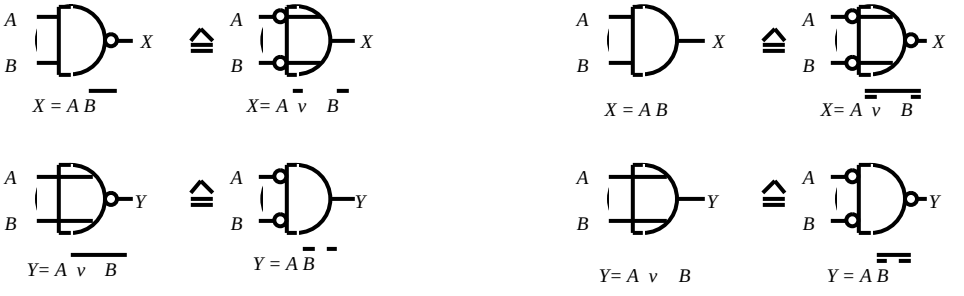
Absorption: (18) $A \wedge (A \vee B) = A$

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

Sehr häufig wird das Gesetz von DeMorgan verwendet, weil damit AND und OR Gatter ineinander umgewandelt werden können. Anhand der Gattersymbole der alten DIN Norm kann dies mit einer einfachen Regel sehr deutlich illustriert werden (Figur 9):

NAND: Der Inversionskreis des Ausganges des NAND Gatters ist als Tintenfleck aufzufassen. Will man diesen Tintenfleck auf die beiden Eingänge schieben, so entstehen zwei Tintenspuren (Striche) und an den Endpunkten wiederum zwei Tintenflecken.

NOR: Der Inversionskreis des Ausganges des NOR Gatters ist als Radiergummi aufzufassen. Verschiebt man diesen auf die beiden Eingänge, so werden die Striche wegradiert und es an den Endpunkten bleiben zwei Radiergummis übrig.

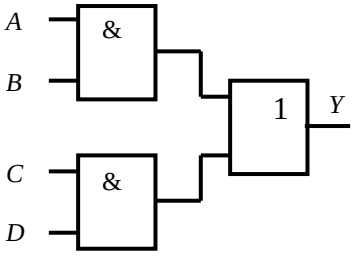



Figur 9: DeMorgans Gesetze.

DeMorgans Gesetze sind im Prinzip nur ein Speziallfall vom Shannonschem Theorem:

$$\overline{f(e_m, \bar{e}_n, \cdot, \vee, \oplus, \overline{\oplus}, \ldots)} = f(\bar{e}_m, e_n, \vee, \cdot, \overline{\oplus}, \oplus, \ldots)$$

**Beispiel 2:** Die folgende Logikschaltung ist so zu modifizieren, dass nur noch NAND Gatter benötigt werden.




## 4.8 Vereinfachung und Optimierung von Funktionen

Die aus einer Wahrheitstabelle gewonnene disjunktive oder konjunktive Normalform einer booleschen Funktion ist meist nicht die kürzeste und einfachste Form. Oft lassen sich Funktionen vereinfachen und zum Teil auch Variablen eliminieren. Die technische Realisierung der vereinfachten Funktion erfodert dann einen geringeren Aufwand als zum Beispiel eine der Normalformen.

Für die systematische Vereinfachung muss die boolesche Funktion in disjunktiver oder konjunktiver Normalform vorliegen. Es gibt verschiedene Vereinfachungsmethoden, von denen die folgenden beiden hier behandelt werden sollen:

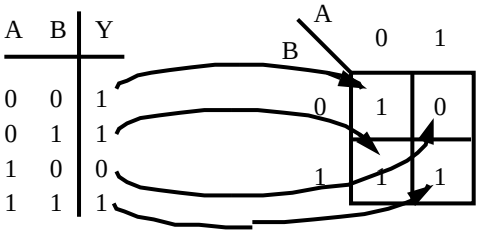
- graphisches Verfahren nach Karnaugh und Veitch
- rechnerisches Verfahren nach Quine und McCluskey

In beiden Verfahren können nur Funktionen mit einer Ausgangsvariablen vereinfacht werden. Verfahren, mit denen Funktionen mit mehreren Ausgangsvariablen optimiert werden können, basieren meist auf den obigen verfahren.

### 4.8.1 Vereinfachung mit dem Karnaugh-Diagramm

Um einfache Funktionen bis zu 5 oder maximal 6 Eingangsvariablen zu vereinfachen eignet sich die graphische Methode nach Karnaugh und Veitch sehr gut.

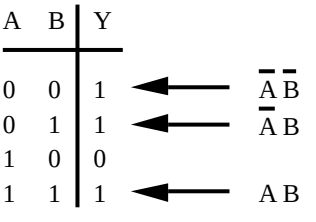
Jeder Zeile der Wahrheitstabelle wird eine “Zelle” im Karnaugh-Diagramm zugewiesen (siehe Figur 10).



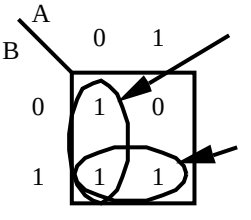

Figur 10: Karnaughtabelle

Nebeneinander liegende Minterme können in Gruppen von $2^n$ zusammengefasst werden. Es kommt das Gesetz $AB \vee A\bar{B} = A$ zur Anwendung.

Als Beispiel diene die folgende boolesche Funktion mit 2 Eingangsvariablen (siehe Figur 11):




$$Y = \bar{A}\bar{B} + \bar{A}B + AB$$




$\bar{A}$ (Elimination von B)

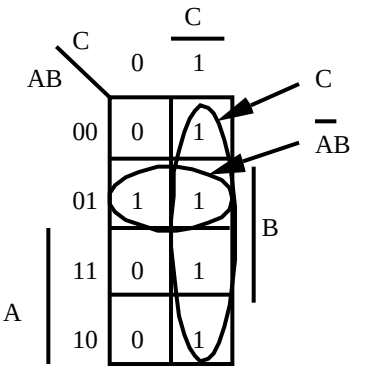
B (Elimination von A)

$$Y = \bar{A} \vee B$$

Figur 11: Vereinfachung bei 2 Eingangsvariablen.

Bei der Vereinfachung mit dem Karnaugh-Diagramm sind die folgenden Regeln zu befolgen:

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |



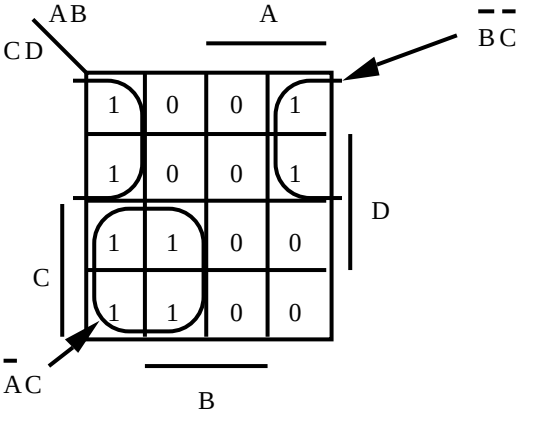

$$Y = \bar{A}B \vee C$$

Figur 12: Vereinfachung an Beispiel mit 3 Eingangsvariablen.

- Mit möglichst wenig Schlaufen sind alle “1” (oder “0”) zu erfassen.
- Es sind möglichst viele Zellen ($2^n$) mit einer Schlaufe zu erfassen.

Im Folgenden sind Beispiele für boolesche Funktionen mit mehrere Variablen zur Vereinfachung aufgeführt.

Die Graphiken in den Figuren 12, 13 und 14 zeigen Beispiele von Vereinfachungen mit 3, 4 und 5 Eingangsvariablen.

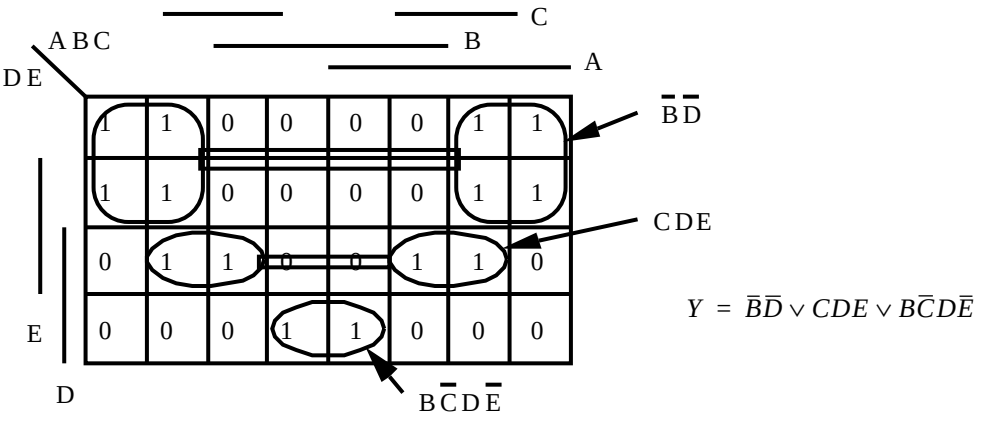



$$Y = \bar{A}C \vee \bar{B}\bar{C}$$

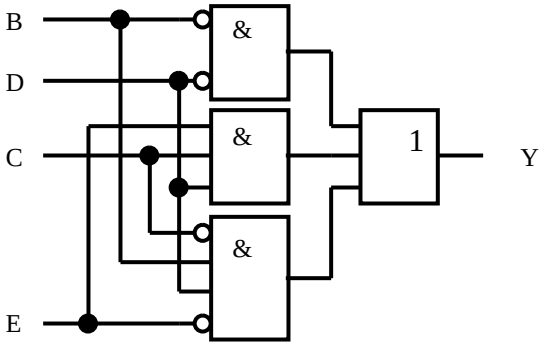
Figur 13: Vereinfachung an Beispiel mit 4 Eingangsvaria-

Bei diesem Beispiel mit den 5 Eingangsvariablen sei dazu noch die elektronische Schaltung dargestellt (siehe Figur 15):

Bei der Zusammenfassung von logischen “1” wurden immer möglichst grosse Schleifen gesucht. Vielfach treten aber auch Funktionen auf, bei denen die logischen “1” wie ein Schachbrett Muster verteilt sind. In diesen Fällen findet man praktisch keine Vereinfachungsmöglichkeiten. Nutzt man bei den Vereinfachungen nicht mehr die Gleichung $AB \vee A\bar{B} = A$, sondern die Bezie-




Figur 14: Vereinfachung an Beispiel mit 5 Eingangsvaria-




Figur 15: Schema des Beispiels mit 5 Eingangsvaria-

hung $A\bar{B} \vee \bar{A}B = A \oplus B$ aus, sollen lassen sich genau diese Schachbrett Muster zusammenfassen, wie dies am Beispiel in Figur 16 illustriert ist.

### 4.8.2 Unvollständig definierte Funktionen

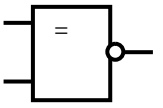
Sind in einer Wahrheitstabelle nicht immer alle Wertekombinationen der Variablen definiert (dont't care), so spricht man von einer unvollständig definierten Funktion. Sind solche "dont't cares" ("X") vorhanden, so lassen sie sich beliebig interpreteieren, was bei den Vereinfachungen mehr Möglichkeiten zur Optimierung offen lässt (siehe Figur 17).

| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| B \ A | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

$Y = A\bar{B} \vee \bar{A}B$

$Y = A \oplus B$




Figur 16: Schachbrettartiges Muster beim EXOR.

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | X |
| 1 | 1 | 0 | X |
| 1 | 1 | 1 | 1 |

| BC \ A | 0 | 1 |
|---|---|---|
| 00 | X | 0 |
| 01 | 1 | X |
| 11 | 0 | 1 |
| 10 | 0 | X |

$\bar{A}\bar{B}$, $A\bar{B}$, C, B

$Y = \bar{A}\bar{B} \vee AB$

Figur 17: Vereinfachung an Beispiel mit dont't cares.

### 4.8.3 Vereinfachung nach Quine und McCluskey

Das Vereinfachungsverfahren nach Quine und McCluskey ist ein rechnerisches Verfahren zur Vereinfachung von Booleschen Gleichungen mit einem Ausgang.

Bei diesem Verfahren werden die Minterme der disjunktiven Normalform systematisch miteinander verglichen, um Variablen der Beziehung $(A \vee \bar{A}) = 1$ zu eliminieren. Zwei Minterme können nur dann vereinfacht werden, wenn sie sich in nur einer Variablen unterscheiden. Um solche Beziehungen zu finden, werden die Minterme in Gruppen zusammengefasst, welche nach Anzahl nichtinvertierter Variablen geordnet werden. Um Vereinfachungen zu finden, sind dann aufeinanderfolgende Gruppen miteinander zu vergleichen.

Die folgende Boolesche Gleichung diene als Beispiel:

$$Y = \bar{A}\bar{B}C\bar{D} \vee \bar{A}\bar{B}CD \vee \bar{A}B\bar{C}\bar{D} \vee \bar{A}B\bar{C}D \vee A\bar{B}C\bar{D} \vee A\bar{B}CD \vee AB\bar{C}\bar{D} \vee AB\bar{C}D \vee ABCD$$

Die beiden Minterme der “Gruppe 1” sind mit denjenigen der “Gruppe 2”

**Tabelle 1: Vereinfachungstabelle nach Quine und McCluskey**

| Gruppe | disjunktive Normalform | ok | 1. Vereinfachung | ok | 2. Vereinfachung | ok |
|---|---|---|---|---|---|---|
| 1 | $\bar{A}\bar{B}C\bar{D}$ | ok | $\bar{A}\bar{B}C$ | ok | $\bar{B}C$ | p3 |
| | $\bar{A}B\bar{C}\bar{D}$ | ok | $\bar{B}C\bar{D}$ | ok | $B\bar{C}$ | p4 |
| 2 | $\bar{A}\bar{B}CD$ | ok | $\bar{A}B\bar{C}$ | ok | | |
| | $\bar{A}B\bar{C}D$ | ok | $B\bar{C}\bar{D}$ | ok | | |
| | $A\bar{B}C\bar{D}$ | ok | $\bar{B}CD$ | ok | | |
| | $AB\bar{C}\bar{D}$ | ok | $B\bar{C}D$ | ok | | |
| 3 | $A\bar{B}CD$ | ok | $A\bar{B}C$ | ok | | |
| | $AB\bar{C}D$ | ok | $AB\bar{C}$ | ok | | |
| 4 | $ABCD$ | ok | $ACD$ | p1 | | |
| | | | $ABD$ | p2 | | |

zu vergleichen (vgl. Tabelle 1). Sind zwei Minterme vorhanden, welche sich in nur einer Variablen unterscheiden, so sind die entsprechenden Minterm der “Gruppe 1” und “Gruppe 2” abzuhaken und der vereinfachte Term in die neue Rubrik “1. Vereinfachung” einzutragen. Dieses Verfahren wird nun solange fortgeführt, bis keine Vereinfachungen mehr möglich sind. Nicht abgehakte Minterme werden Primterme genannt.

Um nun eine minimale Funktion für die Boolesche Gleichung zu finden,

**Tabelle 2: Minterm-Primterm Tabelle**

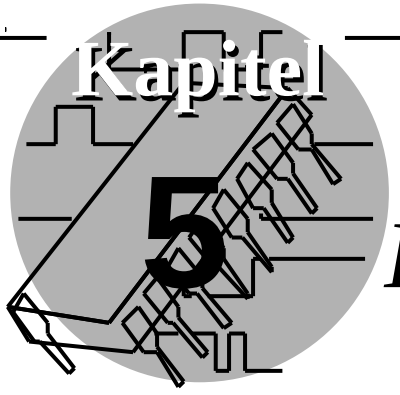
| Primterme | p1= | p2= | P3= | P4= |
|---|---|---|---|---|
| Minterme | $ACD$ | $ABD$ | $\bar{B}C$ | $B\bar{C}$ |
| $\bar{A}\bar{B}C\bar{D}$ | | | x | |
| $\bar{A}B\bar{C}\bar{D}$ | | | | x |
| $\bar{A}\bar{B}CD$ | | | x | |
| $\bar{A}B\bar{C}D$ | | | | x |
| $A\bar{B}C\bar{D}$ | | | x | |
| $AB\bar{C}\bar{D}$ | | | | x |
| $A\bar{B}CD$ | x | | x | |
| $AB\bar{C}D$ | | x | | x |
| $ABCD$ | x | x | | |

ist eine neue Tabelle (Tabelle 2) zu erstellen, in welcher die Abhängigkeit der Primterme von den Mintermen aufgeführt ist

Die Minimalform der Booleschen Gleichung ist nun die disjunktive Verknüpfung derjenigen Primterme, durch welche alle Minterme erfasst werden.

Im obigen Beispiel sind zwei Lösungen für die Minimalfrom möglich:

$$Y_1 = p_1 \vee p_3 \vee p_4 = ACD \vee \bar{B}C \vee B\bar{C}$$
$$Y_2 = p_2 \vee p_3 \vee p_4 = ABD \vee \bar{B}C \vee B\bar{C}$$




# *Kippschaltungen*

## 5.1 Lernziele

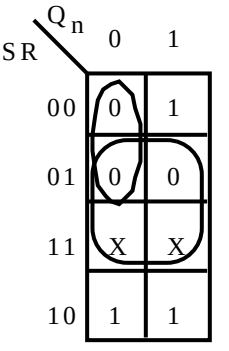
Nach dem Studium des vorliegenden Kapitels sind die folgenden Aufgaben zu lösen oder die folgenden Fragestellungen präzise zu beantworten:

- Verhalten der unterschiedlichen Implementationen von SR-Latchs sind bekannt.
- Unterschiede zwischen synchronen und asynchronen Eingängen, Puls- und Flankensteuerung, Latch und Flip-Flops sind bekannt.
- Charakteristisches Zeitverhalten von Speicherelementen, mit setup-, hold, Verzögerungszeit und Kontrollpuls-Breite sind bekannt.
- Unterschiede zwischen bistabilen, monostabilen und astabilen Kippstufen sind bekannt.
- Verhalten von RS-, D-, JK-, T-, etc Speichern ist bekannt.
- Master-Slave und Einspeicher Implementation des Flip-Flops können analysiert werden.
- Wie werden die Zeiten bei den Monoflops bestimmt und welche Monoflop Typen existieren.
- Wie entstehen Hazards, wie lassen sie sich vermeiden und wo müssen sie zwingend vermieden werden.

## 5.2 Einführung

Bei den bisher behandelten logischen Bauelementen waren die Ausgangssignale nur von den jeweils herrschenden Eingangssignalen abhängig. Im folgenden werden erstmals sogenannte Kippstufen behandelt, deren Ausgangssignale nicht nur von den aktuellen Eingangswerten abhängig sind, sondern auch von früher herrschenden Zustand abhängen. Kippstufen weisen somit ein Gedächtnis (Speicher) auf. Man unterscheidet prinzipiell drei Arten, bistabile, monostabile und astabile Kippstufen.

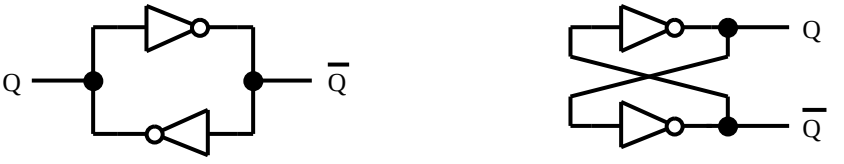
| S | R | $Q_{n+1}$ | |
|---|---|---|---|
| 0 | 0 | $Q_n$ | bisheriger Zustand gespeichert |
| 0 | 1 | 0 | Reset |
| 1 | 0 | 1 | Set |
| 1 | 1 | X | zuf%lliger Wert |




Figur 19: Wahrheitstabelle des SR-Latch's

## 5.3 Elementare bistabile Kippstufen (Latch)

Zwei Inverter ergeben zusammen ein Gebilde, welches von einem einmal eingenommenen logischen Zustand nicht mehr abweichen kann. Diese Anordnung kann 1 Bit speichern.

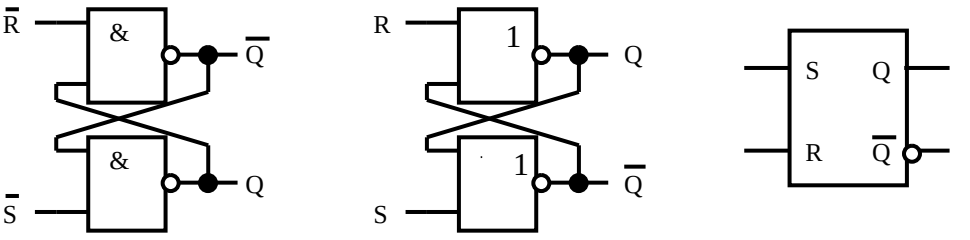



Figur 18: 1 Bit Speicher

### 5.3.1 Latch mit asynchronen Eingängen (Set-Reset Latch)

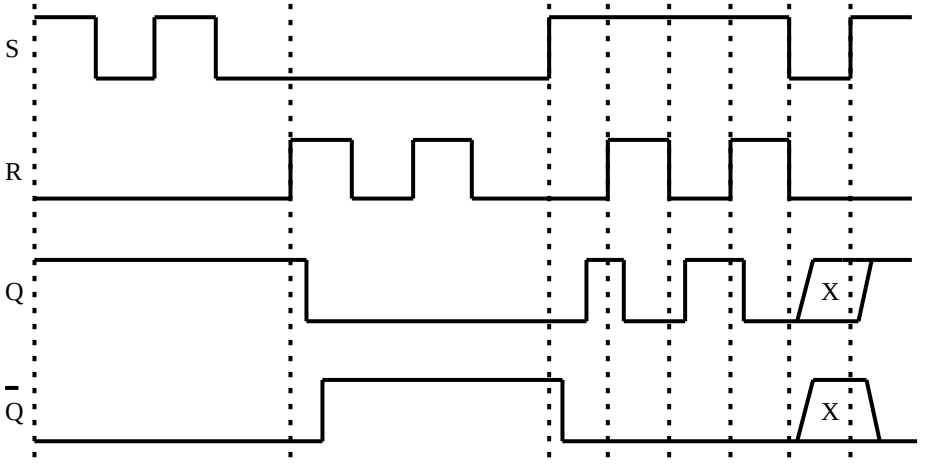
Ersetzt man die beiden Inverter durch NAND oder NOR Gatter, so erhält man ein steuerbares Speicherelement, das sogenannte Set-Reset Latch. Die Funktion eines solchen Set-Reset Latches lässt sich aber auch aus der Wahrheitstabelle mittels Karnaugh-Diagramm herleiten.

Die vereinfachte Boolesche Funktion für den invertierten Ausgang lautet: $\overline{Q}_{n+1} = R + \bar{S}\overline{Q}_n$. Dadurch lässt sich das SR-Latch mit zwei NOR Gatter realisieren.




Figur 20: Schema eines NAND und NOR SR-Latch's.

Typisch für ein Latch ist, dass sich Änderungen am Eingang sofort am Ausgang bemerkbar machen (transparentes Latch). Die Eingäge dieses Latch's werden deshalb auch als asynchron bezeichnet.

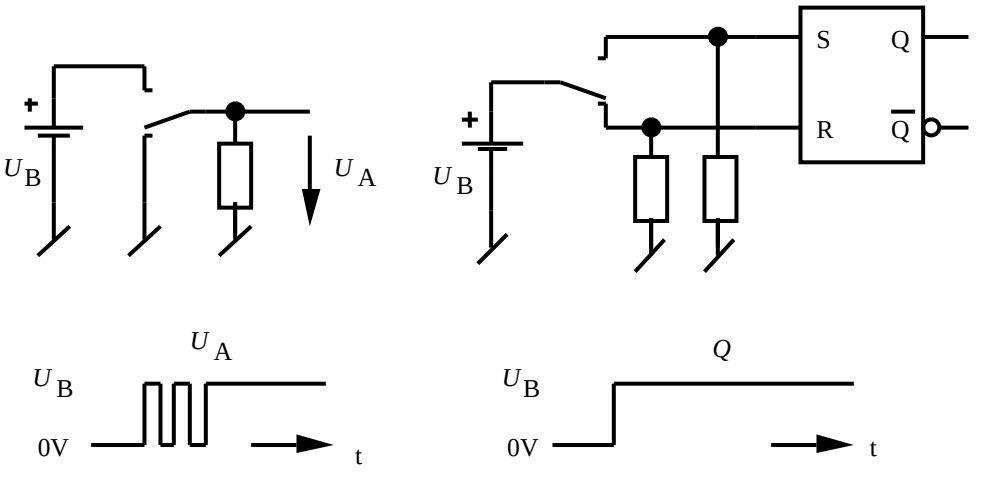



Figur 21: Zeitverhalten des (NOR-NOR) SR-Latch's

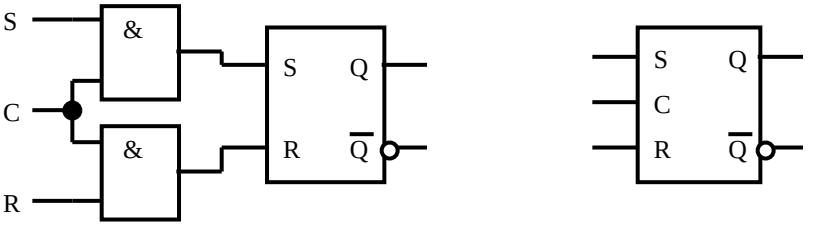
Das zeitliche Verhalten des SR-Latch's, aufgebaut aus NOR Gattern, ist in der obigen Abbildung dargestellt. Dabei ist die direkte Abhängigkeit der Ausgänge von den Eingängen zu sehen und das Auftreten eines zufälligen Ausgangswertes, wenn beide Eingänge gleichzeitig von "1" nach "0" wechseln.

SR-Latchs sind die Grundbausteine für komplexere Speicherbausteine.

**Anwendungsbeispiel:** mechanische Kontakte haben die unschöne Gewohnheit, beim Schliessen zu prellen. Mit einer geeigneten Beschaltung des RS-Latchs lässt sich dies verhindern.




Figur 22: Anwendungsbeispiel prellfreier Schalter.

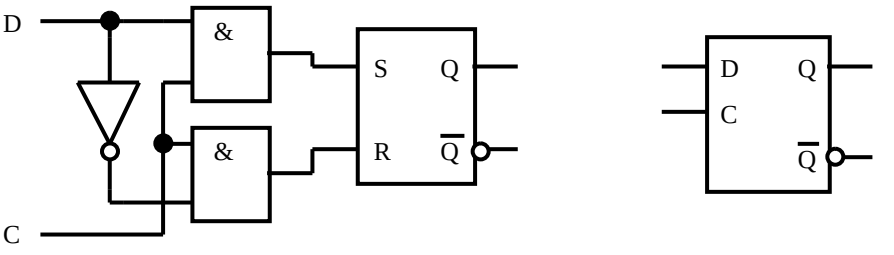



Figur 23: SR-Latch mit Kontroll Eingang.

### 5.3.2 Latch mit synchronen Eingängen

Es ist häufig von Nutzen mit Hilfe eines Aktivierungssignales (Gate, Strobe, Enable, Control Signal) den Zugang der Eingangssignale zum Latch steuern zu können. Dazu ist ein einfaches Netzwerk vor das SR-Latch zu schalten, welches die Set und Reset Eingänge Kontrolliert. Ist das Signal *C=0*, so vermögen die beiden Eingänge *S* und *R* den Zustand des Latch's nicht zu verändern.

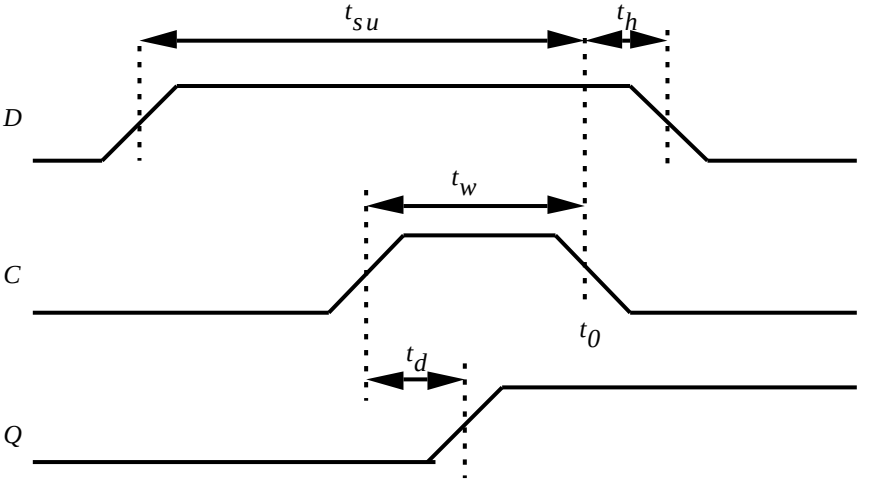
Ein weiteres Latch mit einem synchronen Eingang ist das sogennante D-Latch. Die am *D* Eingang anliegenden Daten erscheinen an den Ausgängen, wenn das Kontrollsignal *C=1* ist. Wird *C* logisch "0", so wird der letzte Wert gespeichert.



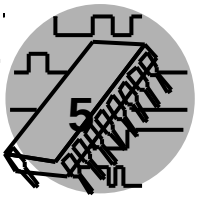

Figur 24: D-Latch

### 5.3.3 Zeitverhalten von Latch's mit synchronen Eingängen

Um ein korrektes Funktionnieren eines synchronen Latch's zu gewährleisten, müssen gewisse zeitlichen Bedingungen eingehalten werden:




Figur 25: Charakteristisches Zeitverhalten eines Latch's.



- **Setup-Zeit** ($t_{su}$): Minimale Zeit, welche die Eingangsdaten vor dem Speichervorgang ($t_0$) anliegen müssen, damit sie korrekt erkannt werden.
- **Hold-Zeit** ($t_h$): Minimale Zeit, welche die Eingangsdaten nach dem Speichervorgang ($t_0$) noch vorhanden sein müssen.
- **Kontrollpuls-Breite** ($t_w$): Minimale Zeit, welche das Kontrollsignal aktiv sein muss.
- **Verzögerungszeit** ($t_d$): Verzögerungszeit des Ausgangssignales.

Die Setup-Zeit ist nötig, um sicher zu sein, dass jede Änderung der Eingangsdaten die Schaltung durchlaufen hat, bevor das Kontrollsignal inaktiv wird. Die Setup-Zeit und die Kontrollpuls Breite sind dafür verantwortlich, dass die internen Signalwerte am Ausgang erscheinen, bevor die Eingangssignale verschwinden.

## 5.4 Pulssteuerung und Flankensteuerung

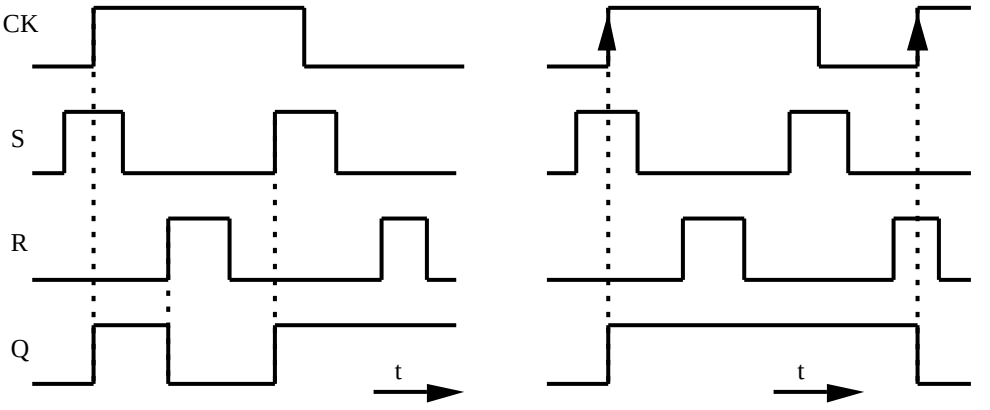
Bei den Speicherelementen existieren zwei grundsätzlich verschiedene Arten der Taktsteuerung.

- **Pulssteuerung** (Pulstriggerung). Bei der Pulssteuerung wirken die Informationseingänge solange auf den Speicherinhalt ein, wie der Pegel des Steuereinganges aktiv bleib.
- **Flankensteuerung** (Flankentriggerung). Bei der Flankensteuerung wirken die Informationseingänge nur während der aktiven Flanke des Steuereinganges auf den Speicherinhalt.

In Figur 26 ist der Unterschied dieser beiden Ansteuerungstypen in einem Zeitdiagramm dargestellt. Dabei ist deutlich zu sehen, dass bei der Pulssteuerung ein aktiver Pegel, bei der Flankensteuerung eine Pegeländerung den Speicherinhalt zu verändern vermag.

## 5.5 Flip-Flop's (bistabile Kippstufen)

Bei synchronen Digitalsystemen werden mit einem Steuersignal verschiedene Speicherelemente gleichzeitig angesteuert. Die Speicherelemente können dabei auch in Serie geschaltet werden, so dass die Ausgänge einer ersten Speicherstufe die Eingänge einer zweiten Speicherstufe bilden. Mit den früher behandelten Latch's lässt sich eine solche Schaltung nicht realisieren, da die Latch's im transparenten Zustand die Eingangsdaten durch die ganze Latch-Kette hindurch schieben würden. Es muss also ein neues Speicherelement eingeführt werden, das sogenannte Flip-Flop.

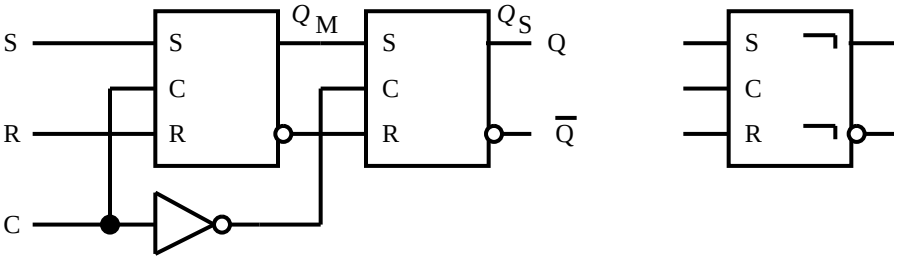



Figur 26: Gegenüberstellung von Pulssteuerung und Flankensteuerung.

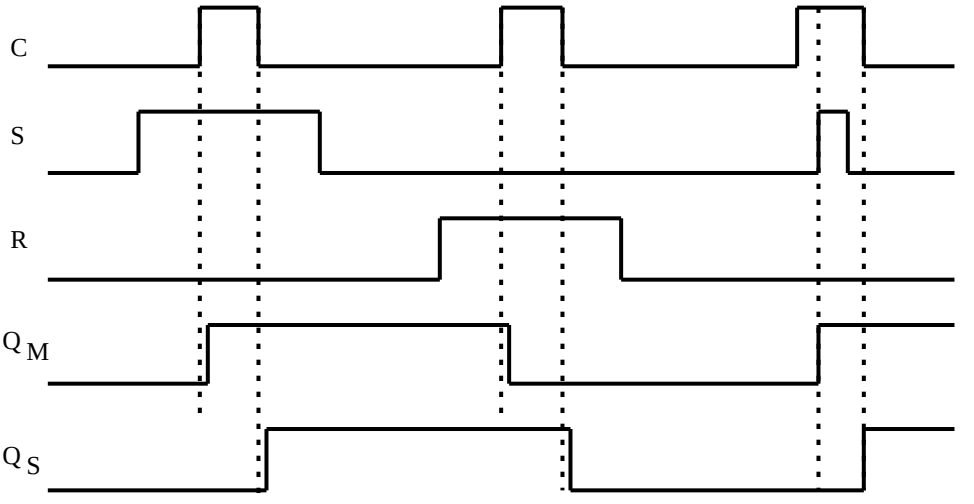
Unter einem Flip-Flop versteht man ein gesteuertes bistabiles Speicherelement, welches den transparenten Modus, wie es das Latch kennt, nicht besitzt. Das bedeutet, dass eine Änderung eines Flip-Flop Ausganges nie direkt das Resultat einer Änderung eines synchronen Einganges sein kann. Die Konsequenz davon ist, dass bei einem Flip-Flop eine Serieschaltung möglich ist, ohne dass dabei Daten verloren gehen, was bei einem Latch nicht möglich ist.

### 5.5.1 RS-Flip-Flop

Aus Figur 27 ist ersichtlich, dass sich ein Master-Slave Flip-Flop prinzipiell aus zwei einfachen gesteuerten Latch's aufbaut. Verwendet man dazu zwei gesteuerte RS-Latch's, so erhält man das entsprechende RS-Master-Slave Flip-Flop.




Figur 27: Prinzipschema und Symbol des RS-Master-Slave Flip-Flop's.



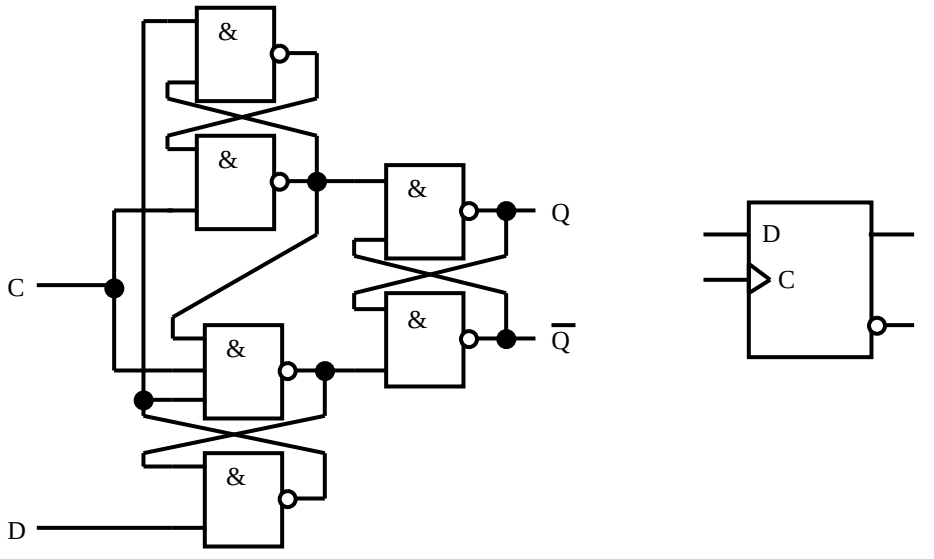

Figur 28: Impulsdiagramm des RS-Master-Slave Flip-Flop's.

Im Impulsdiagramm ist das zeitliche Verhalten dieses Master-Slave Flip-Flop Types dargestellt. Bei einem hohen *C*-Signal werden die Daten in das erste Latch, dem Master, eingelesen. Die Ausgänge des Flip-Flop's behalten den alten Zustand gespeichert. Erst wenn das *C*-Signal logisch "0" ist, werden die Daten vom zweiten Latch, dem Slave, übernommen und zum Ausgang des Flip-Flop's gebracht. Das Master-Latch ist in dieser Phase im gespeicherten Zustand und lässt keine neuen Daten passieren. Das Signal $Q_M$ ist der *Q* Ausgang des Masters, das Signal $Q_S$ der *Q* Ausgang des Slaves und damit des gesamten Flip-Flop's.

Der zweite "1" Puls des Set Einganges fällt innerhalb der "1" Phase des *C*-Takteinganges. Da diese Eingangsänderung vom Master-Latch übernommen wird und bei tiefem *C*-Eingang auf die Flip-Flop Ausgänge übertragen wird, spricht man in diesem Fall von einem **pulsgetriggerten** Flip-Flop. Das Ausgangssignal des Flip-Flop's reagiert also nicht auf eine Flanke des *C*-Eingangs sondern nur auf dessen Pegel.

### 5.5.2 D- Flip-Flop

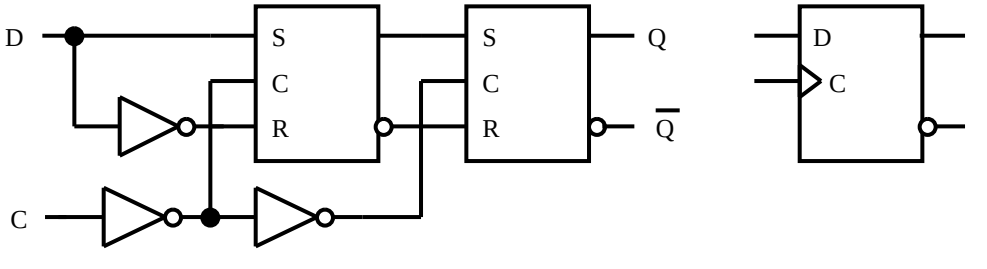
Ähnlich den verschiedenen Varianten des einfachen Latch's, lassen sich nun ebenfalls mehrere Flip-Flop Varianten realisieren. Verwendet man das Master-Slave Prinzip auf das D-Latch an, so erhält man die ältere Implementationsvariante des D-Flip-Flop's. Ändert sich der *D* Eingang während einem tiefen *C* Signal vorübergehend, so übernimmt das Master-Latch diese Änderung ebenfalls nur vorübergehend. Für die Flip-Flop Ausgangssignale ist aber



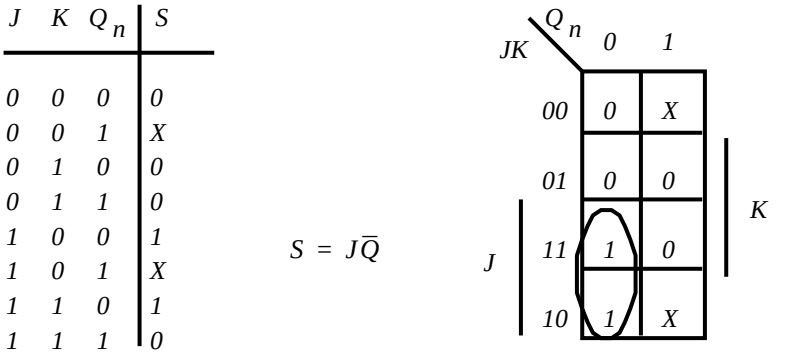

Figur 30: Einspeicher Implementation des D-Flip-Flop's.

nur der aktuelle Eingangswert bei steigender Taktflanke von Bedeutung, weshalb man hier von einem **flankengetriggerten** D-Flip-Flop spricht.

Eine andere, aktuellere Implementationsvarinate des flankengetriggerten D-Flip-Flop's stellt das Schema in Figur 30 dar. Hier handelt es sich nicht mehr um eine Master-Slave Struktur, sondern um einen Einfachspeicher mit vorgeschalteter Logik. Analysiert man die Funktionsweise dieser Struktur etwas genauer, so stellt man fest, dass bei tiefem Taktsignal *C* die erste Kippstufe nicht wie beim Master-Slave Flip-Flop als Speicher fungiert, da die Ausgänge dieser Kippstufe unabhängig vom Eingang *D* logisch "1" sind. Erst




Figur 29: Master-Slave Implementation des flankengesteuerten D-Flip-Flop's.



| J | K | $Q_n$ | S |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | X |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | X |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

$S = J\overline{Q}$

| JK \ $Q_n$ | 0 | 1 |
|---|---|---|
| 00 | 0 | X |
| 01 | 0 | 0 |
| 11 | 1 | 0 |
| 10 | 1 | X |

J, K

Figur 32: Herleitung der Bedingung für den *S* Eingang.

wenn das Taktsignal von “0” auf “1” wechselt, übernimmt zuerst die erste, dann die zweite Kippstufe den entsprechenden Wert des Einganges.

### 5.5.3 JK-Flip-Flop

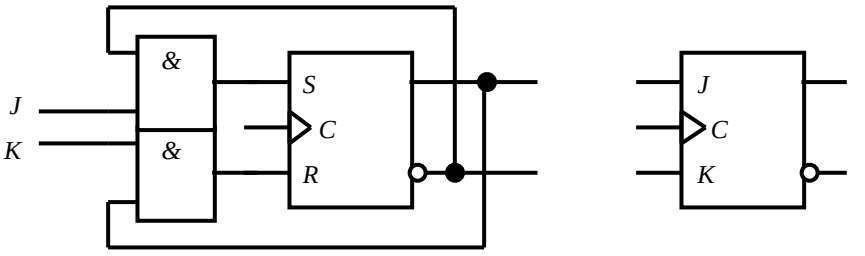
Neben einer grossen Funktionsvielfalt verschiedener Flip-Flop’s wird vor allem das JK-Flip-Flop sehr häufig eingesetzt. Ähnlich dem RS-Flip-Flop weist es ebenfalls zwei Dateneingänge auf, welche das Flip-Flop setzten, beziehungsweise rückstellen können. Der einzige, aber wesentliche Unterschied zum RS-Flip-Flop besteht aber darin, dass beim JK-Flip-Flop beide Dateneingänge gleichzeitig logisch “1” sein dürfen. Die Bedeutung der beiden *J* und *K* Eingänge ist in der Wahrheitstabelle in Figur 31 dargestellt:

Erweitert man diese Wahrheitstabelle um den Eingang $Q_n$, so findet man eine Beziehung für den *S* Eingang eines SR-Flip-Flop’s (Figur 32). Ähnlich kann für den *R* Eingang vorgegangen werden.

In Figur 33 ist das Schema des JK-Flip-Flop’s dargestellt. Daraus ist ersichtlich, wie aus dem RS- ein JK-Flip-Flop hergeleitet werden kann.

| J | K | $Q_{n+1}$ |
|---|---|---|
| 0 | 0 | $Q_n$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $\overline{Q}_n$ |

Figur 31: Wahrheitstabelle des JK-Flip-Flop’s.




Figur 33: JK-Flip-Flop aufgebaut aus RS-Flip-Flop.

## 5.6 Asynchrone Eingänge von Flip-Flop's

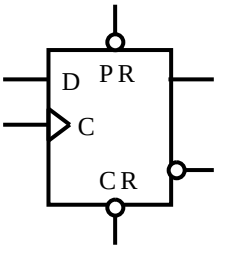
Neben den synchronen Dateneingängen können die im vorausgegangenen Abschnitt besprochenen Flip-Flop's noch asynchrone Eingänge aufweisen. Die zusätzlichen asynchronen Eingänge werden normalerweise für das Setzen oder Rückstellen der Flip-Flop's benötigt:

- asynchroner Preset Eingang (PR)
- asynchroner Clear Eingang (CR)

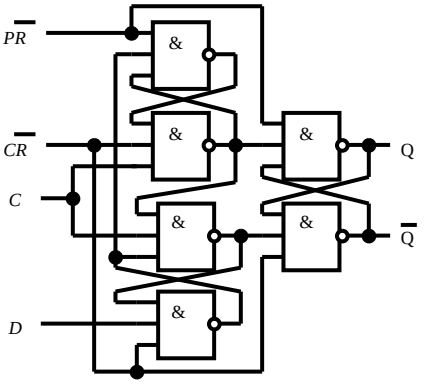
Der Preset Eingang setzt den Ausgang *Q* des Flip-Flop's, der Clear Eingang löscht den Ausgang *Q*. In Figur 34 ist das Symbol eines flankengetriggerten D-Flip-Flop's mit asynchronen $\overline{PR}$ und $\overline{CR}$ Eingängen dargestellt. Die Implementation, wie sie beim SN 7474 Schaltkreis dafür verwendet wird ist in Figur 35 dargestellt.

Die beiden Signale sind aktiv tief, was bedeutet, dass sie nicht gleichzeitig logisch "0" sein dürfen, weil sonst der Ausgangswert *Q* undefiniert ist.

Beim Entwurf von Digitalschaltungen sollte darauf geachtet werden, dass die darin verwendeten Flip-Flop's zumindest beim Systemstart in einen definierten zustand gebracht werden können. Dies bedingt, dass nur Flip-Flop's




Figur 34: Symbol eines D-Flip-Flop's mit asynchronen Setzt- und Rücksetzeingängen.




Figur 35: Implementation des SN7474 D-Flip-Flop's.

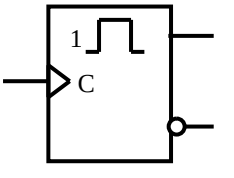
definierter Zustand zu jedem Zeitpunkt erreicht werden kann, was nach einem generellen Initialisierungs-Signal verlangt (meist $\overline{RESET}$ ).

## 5.7 Monoflops (monostabile Kippstufen)

Ein Monoflop ist eine rückgekoppelte binäre Schaltung, die wie ein Flip-Flop zwei verschiedene innere Zustände annehmen kann. Der Unterschied zum Flip-Flop besteht darin, dass beim Monoflop nur einer der beiden inneren Zustände stabil ist. Der andere Zustand behält seinen Wert nur für eine bestimmte Zeit und kippt danach wieder in seinen stabilen Zustand zurück.

Monoflops lassen sich aus RS-Latch's oder Flip-Flop's mit vorgeschalteten Differenzier- oder Verzögerungsgliedern aufbauen. Um die Zeitkonstante einstellen zu können, wird ein externes RC-Glied verwendet.

Figur 36: Symbole einer flankengetriggerten monostabilen Kippstufe.



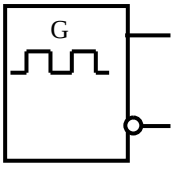

In Figur 36 ist das Symbol einer monostabiler Kippschaltungen dargestellt. Bei aktiver Triggerung erzeugt es einen positiven Ausgangspuls definierter Länge. Bei den Monoflops unterscheidet man zwei verschiedene Typen:

- nachtriggerbare Monoflops
- nicht nachtriggerbare Monoflops

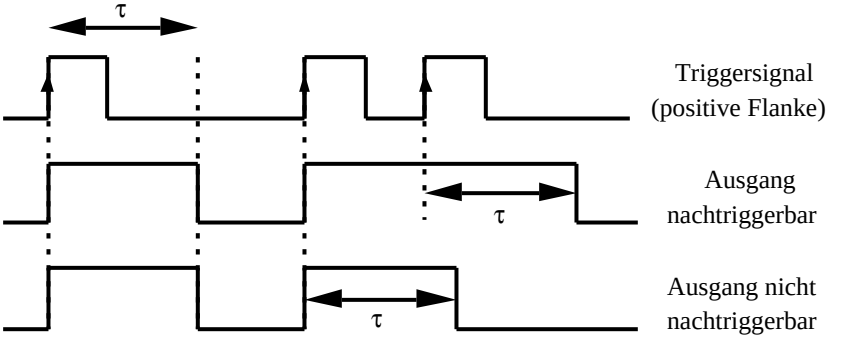
Der Unterschied dieser beiden Typen ist im Zeitdiagramm in Figur 37 illustriert. Beim nicht nachtriggerbaren Monoflop haben aktive Triggersignale währendem gerade ein Puls am Ausgang erzeugt wird keinen Einfluss auf das Ausgangssignal. Ein Vertreter eines Monoflops ist zum Beispiel der Baustein SN74123, welcher je nach Beschaltung auf die steigende oder fallende Flanke reagieret und nachtriggerbar oder nichtnachtriggerbar arbeiten kann.

## 5.8 Multivibrator (astabile Kippstufe)

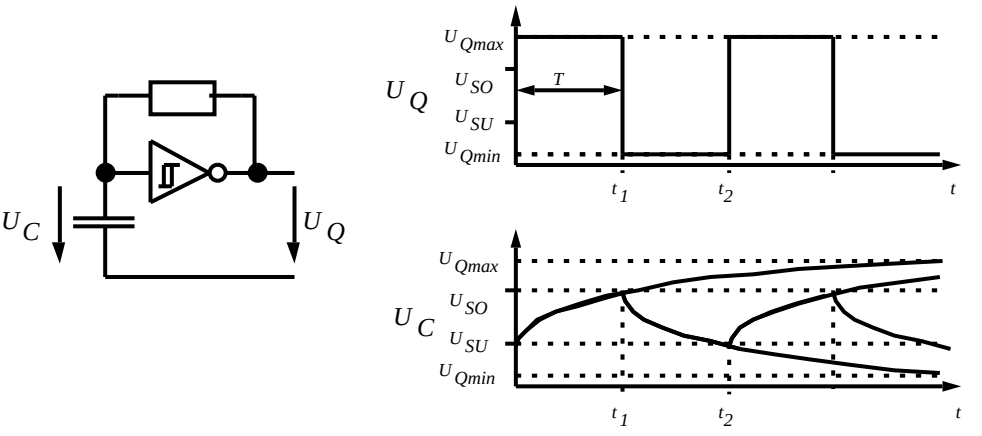
Ein Multivibrator ist eine rückggekoppelte binäre Schaltung, die zwei verschiedene innere Zustände hat, zwischen denen sie dauernd hin und her kippt. Die Schaltung wird deshalb auch astabile Kippstufe oder Rechteckgenerator genannt. Das astabile Verhalten wird wie beim Monoflop durch eine zeitabhängige Rückkopplung mit Differenzier- oder Verzögerungsgliedern erreicht.




Figur 38: Symbol eines Multivibrators.



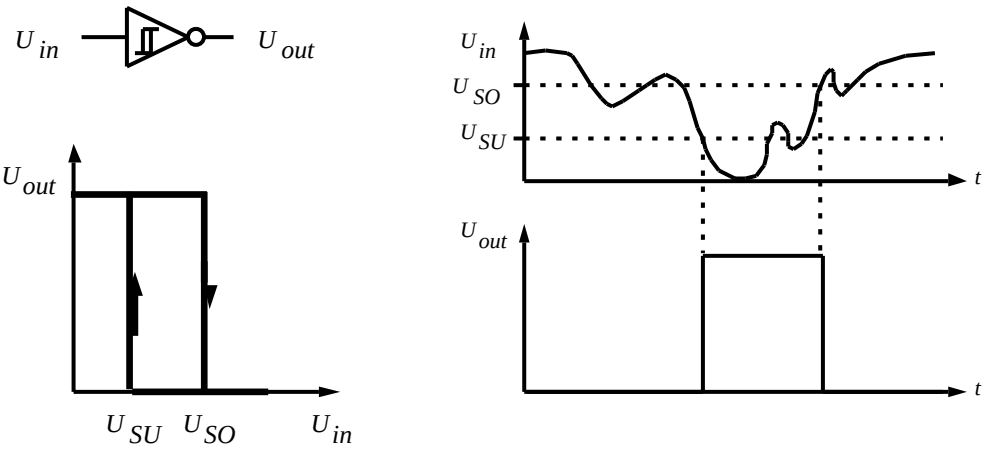

Figur 37: Zeitdiagramm von nachtriggerbaren und nicht nachtriggerbaren Monoflops.



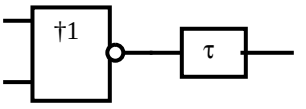

Figur 40: Zeitdiagramm eines einfachen Multivibrators.

Eine sehr einfache Schaltung eines Multivibrators besteht aus einem Schmitt-Trigger und einem RC-Verzögerungsglied, wie es in Figur 40 dargestellt ist. Ein Schmitt-Trigger verhält sich ähnlich einem Inverter, mit dem Unterschied, dass am Eingang nicht eine, sondern je eine tiefe und eine hohe Schwellspannung existiert. Er wird vielfach dazu verwendet, um Störungen besser unterdrücken zu können.

Die Spannung $U_C$ über dem Kondensator wird in der Aufladephase von $U_{SU}$ gegen $U_{Qmax}$ aufgeladen. Sobald der Wert der oberen Schwellspannung $U_{SO}$ erreicht wird, wird der Aufladevorgang durch einen Entladevorgang




Figur 39: Spannungsverläufe des Schmitt-Triggers.




Figur 41: Reales logisches Gatter mit Verzögerung.

ersetzt. Die Spannung $U_C$ wird dabei von $U_{SO}$ gegen $U_{Qmin}$ entladen, bis $U_{SU}$ erreicht wird. In der Zeitphase von 0 bis $t_1$ gilt die folgende Beziehung:

$$u_C = (U_{Qmax} - U_{SU})\left(1 - e^{\frac{t}{RC}}\right) + U_{SU}$$

Zum Zeitpunkt $t=t_1$ ist die Kondensatorspannung gleich der oberen Schwellspannung $U_{SO}$. Damit erhält man für die halbe Periode $T$ die folgende Beziehung:

$$T = t_1 = RC\ln\frac{U_{Qmax} - U_{SU}}{U_{Qmax} - U_{SO}}$$

Aus dieser Beziehung ist ersichtlich, wie die Periode mit einem RC-Glied festgelegt werden kann.

# 5.9 Hazards

Bei den bisherigen Betrachtungen wurde immer angenommen, dass ein logisches Gatter keine interne Verzögerungszeit aufweist. Dadurch war auch die Problematik von Einschwingvorgängen beiseite geschoben. Bringt man bei logischen Elementen nicht nur ihre logische Funktion ins Spiel, sondern berücksichtigt auch ihr zeitliches Verhalten, so wird man ungewohnte Phänomene beobachten, welche die korrekte Funktionsweise einer Schaltung zunichte machen können.

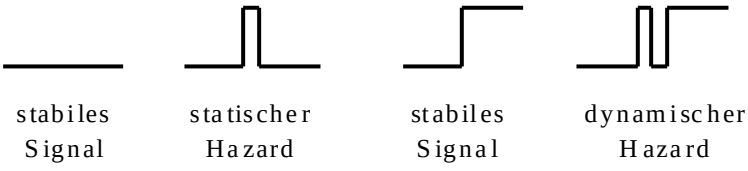
In Figur 41 ist ein reales logisches Gatter dargestellt, welches aus einer logischen Booleschen Funktion und einem Verzögerungselement besteht. Berücksichtigt man solche endliche Gatterlaufzeiten, so können unliebsame Einschwingvorgänge beobachtet werden.

## 5.9.1 Entstehung von Hazards

Hängt das Ausgangssignal einer kombinatorischen Schaltung neben den Eingangssignalen auch von Verzögerungszeiten ab, dann weist die Schaltung einen Hazard auf. Bei einer kombinatorischen Schaltung können “Ein-

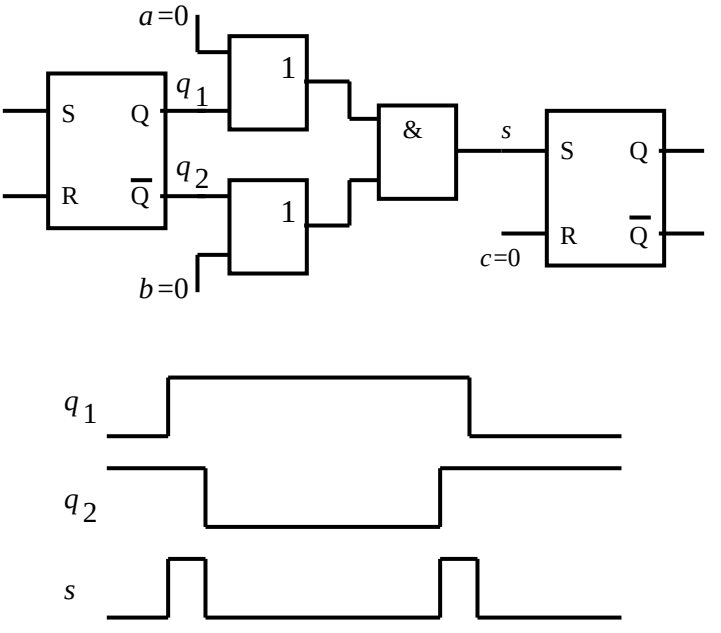
schwingvorgänge” von Hazards betroffen werden. Hazards lassen sich prinzipiell in zwei Klassen unterteilen:

- statische Hazards
- dynamische Hazards

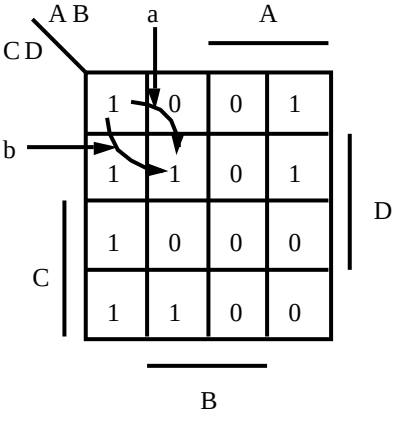



Figur 42: Gegenüberstellung statischer und dynamischer Hazards.

Es ist ein statischer Hazard anwesend, wenn nur eine vorübergehende Änderung des Ausgangssignales hervorgerufen wird und der statische Ausgangswert nicht tangiert wird. Bei einem dynamischen Hazard ändert sich das Ausgangssignal mehrmals, befor es auf seinen stabilen Wert wechselt.




Figur 43: Schaltung mit statischem Hazard an Signal *s*.




Figur 44: Enstehung eines Hazards.

In Figur 43 ist ein Schaltungsbeispiel wiedergegeben, welches aufgrund eines Hazards nicht funktionstüchtig ist. Das erste SR-Latch weise am *Q* Ausgang den logischen Wert “0” auf. Wird es darauf gesetzt, so wird zuerst das Signal $q_1$ auf “1” wechseln, befor $q_2$ auf “0” fällt. Da somit kurzzeitig beide Eingänge des UND-Gatters “1” sind, wird am Signal *s* ein statischer Hazard auftreten. Dieser Hazard setzt nun fälschlicherweise das zweite Latch.

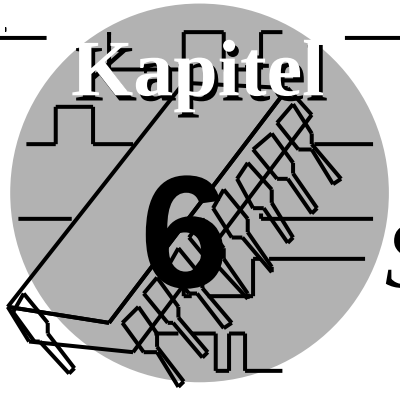
Hazards können nur dann auftreten, wenn sich mehr als eines der Eingangs- oder der internen Signale fast gleichzeitig wechseln. Ob in bestimmten Situationen ein Hazard auftreten kann, lässt sich durch das Betrachten des Karnaugh-Diagrammes (Figur 44) feststellen. Die Signale *ABCD* wecheln von “0000” nach “0101”. Da die beiden Signale *B* und *D* in der Realität nicht exakt gleichzeitig von “0” nach “1” wechseln können, wird das eine Signal zuerst den Wechsel vornehmen. Wie aus dem Pfeil *a* ersichtlich ist, ensteht ein statischer Hazard, wenn zuerst das Signal *B* von “0” nach “1” und erst danach *D* wechselt. Ist die Reihenfolge umgekehrt, so führt der Weg von Pfeil *b* nur über logische “1”, das heisst das Ausgangssignal bleibt stabil und weist keine Hazards auf.

### 5.9.2 Vermeidung von Hazards

Hazards lassen sich leider in kombinatorischen Schaltungen nicht immer unterdrücken. Vermeiden lassen sich gegebenenfalls Situationen, welche Hazards erzeugen können. Es ist deshalb wichtig, besonders kritische Signale auf die Möglichkeit von Hazards zu untersuchen und entsprechende Massnahmen zu ergreifen. Generell darf man feststellen, dass Clock, Setzt- und Rücksetzsignale von Flip-Flop’s kritische Signale sind. Ein statischer Hazard auf eine solche Leitung ruft immer ein Fehlverhalten der Schaltung hervor. Befolgt man zum Beispiel die folgende Entwurfsregel, so lassen sich schon viele kritische Hazards eliminieren:

- Keine kombinatorische Verknüpfungen zur Erzeugung von Clocksignalen und Setzt- beziehungsweise Rücksetztsignalen verwenden.
- Hazardempfindliche Signale korrekt einsynchronisieren.

Es darf generell festgestellt werden, dass asynchrone Schaltung sehr kritisch bezüglich dem Entstehen von Hazards sind. Dies ist ein Grund, weshalb soweit wie möglich synchrone Schaltungen entworfen werden sollen. Die Sicherheit, welche man sich mit synchronen Schaltungen erkauft, geht auf Kosten des Synchronisierungs-Mechanismus, was erhöhte Verzögerungszeiten bedeutet.




# Sequenzer

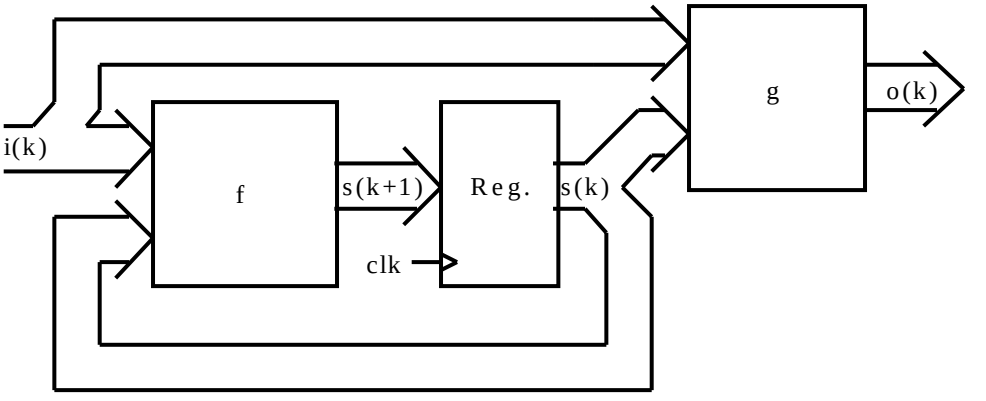
## 6.1 Lernziele

Nach dem Studium des vorliegenden Kapitels sind die folgenden Aufgaben zu lösen oder die folgenden Fragestellungen präzise zu beantworten:

- Wodurch unterscheiden sich Mealy-, Moore und Medwedjew Automaten. Welches sind die Vor-, respektive Nachteile derselben.
- Ausgehend von der beschreibung eines Algorithmus kann das entsprechende Zustandsdiagramm aufgestellt werden.
- Vollständigkeit und Konsistenz werden beim Aufstellen des Zustandsdiagrammes berücksichtigt.
- Ausgehend vom Zustandsdiagramm lassen sich Übergangstabelle und Ausgangstabelle generieren und daraus die minimisierte Form des Automaten synthetisieren.
- Die Problematik der parasitären Zustände und die Initialisierung des Automaten ist bekannt.
- Die Problematik der Hazards in Sequenzern und bei der Ansteuerung externer Datenpfade ist bekannt.
- Die Problematik der Einsynchronisation asynchroner Signale in Sequenzern ist bekannt.
- Das Prozess Modell Zustandsmaschine und Datenpfad (FSMD) von D. Gajski ist bekannt.
- Ein Beispiel wie der Black-Jack Dealer kann selbständig gelöst werden.

## 6.2 Grundlegende Sequenzer Strukturen

Sequenzer sind Schaltungen, welche aus einer gegebenen Sequenz von Eingangssignalen eine vordefinierte Sequenz von Ausgangssignalen erzeugen. Sequenzer, auch endliche Automaten, Folgeschaltungen oder Zustandsmaschinen genannt (engl. “finit state machine”), sind in der Digitaltechnik und vor allem in VLSI Schaltungen sehr häufig anzutreffen. Die Gründe dafür sind



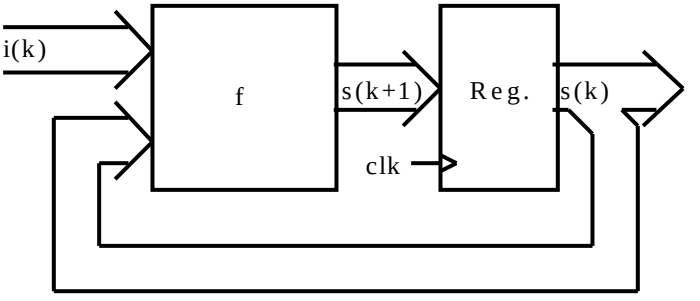

Figur 45: Struktur des Mealy-Automaten.

naheliegend. Ablaufsteuerungen lassen sich einfach und verständlich in Form von Zustandsdiagrammen beschreiben. Die Synthetisierung von Schaltungen aus den Zustandsdiagrammen erfolgt völlig automatisch und damit zeiteffizient und fehlerfrei, was für den Entwurf von Digital- oder VLSI Schaltungen höchst willkommen ist.

Zwei bekannte Implementationsvarianten endlicher Automaten sind der Mealy- und der Moore-Automat. Beide Automaten sind je aus zwei kombinatorischen Blöcken (mit den Funktionen *f* und *g*) und einem Register für die Zustandsspeicherung aufgebaut. In Figur 45 ist die Struktur eines Mealy-Automaten dargestellt. Daraus ist ersichtlich, dass die Ausgänge *o* sowohl von den internen Zuständen *s*, als auch direkt von den Eingängen *i* abhängig sind. Der Mealy-Automat erfüllt die folgenden Funktionen:

$$o(k) = g(i(k), s(k))$$
$$s(k+1) = f(i(k), s(k))$$

Etwas anders ist die Struktur des Moore-Automaten (siehe Figur 46), bei welchem keine direkte Abhängigkeit von Ein- und Ausgangssignalen mehr

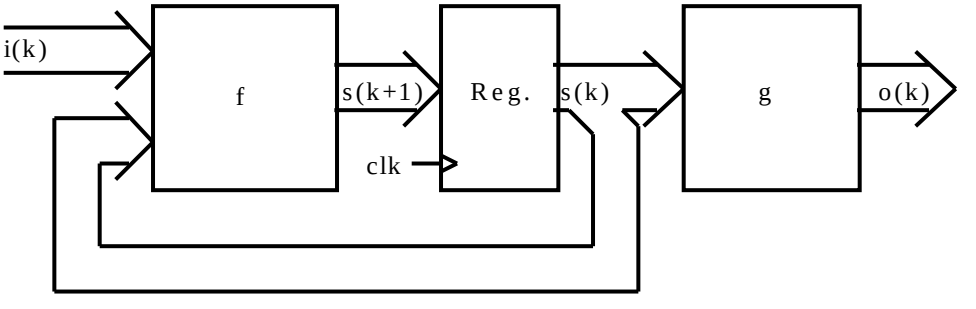



Figur 47: Der Medwedjew-Automat ist ein Spezialfall des Moore-Automaten.

besteht. Diese Tatsache wiederspiegelt sich direkt in seinen charakteristischen Gleichungen:

$$o(k) = g(s(k))$$
$$s(k+1) = f(i(k), s(k))$$

Ein recht häufig gesehener Spezialfall des Moore-Automaten ist der sogenannte Medwedjew-Automat, bei welchem die Zustandsvariablen direkt als Ausgänge dienen (siehe Figur 47). Durch den Verzicht auf einen kombinatorischen Block nach den Zustandsvariablen ist gewährleistet, dass die Aus-




Figur 46: Struktur des Moore-Automaten.

gänge Hazard-frei sind. Die charakteristischen Gleichungen des Medwedjew-Automaten vereinfachen sich dadurch wie folgt:

$$o(k) = s(k)$$
$$s(k+1) = f(i(k), s(k))$$

Bei allen drei Automaten wird der kombinatorische Block *f* meist in Form einer PLA Struktur (AND-OR-Ebene) aufgebaut.

## 6.3 Beschreibung von Sequenzern

Für die Beschreibung des Verhaltens eines Sequenzers stehen prinzipiell zwei Mittel zur Verfügung:

- Übergangstabelle und Ausgangstabelle
- Zustandsdiagramm

### 6.3.1 Übergangs- und Ausgangstabelle

Durch die Übergangstabelle, zusammen mit der Ausgangstabelle wird ein endlicher Automat oder Sequenzer vollständig beschrieben. Diese Beschreibungsart lehnt sich sehr stark an die bereits bekannten Wahrheitstabellen an.

Die Übergangstabelle, auch Zustandstabelle genannt, definiert den Übergang von einem alten in einen neuen Zustand, weshalb diese Tabelle häufig auch Übergangstabelle genannt wird. Im Gegensatz zu den Wahrheitstabellen aus der rein kombinatorischen Logik, bei welchen die Ausgänge nur in Abhängigkeit der Eingänge beschrieben wurden, sind in der Übergangstabelle die Ausgänge in Abhängigkeit der Eingänge und der alten Zuständen zu beschrieben. Betrachtet man die in den Figuren 45, 46 und 47 dargestellten Sequenzer Strukturen mit den dazugehörigen Gleichungen, so ist ersichtlich, dass jeweils die Funktion $f(i(k), s(k))$ diese Übergangsbedingungen beschreibt. Die Übergangstabelle stellt somit die Funktion $f(i(k), s(k))$ tabellarisch dar.

Ein Beispiel einer einfachen Übergangstabelle ist in Figur 48 dargestellt. Ist der Automat im Zustand_1, so springt er in der nächsten Taktphase in den Zustand_2, sofern der Eingang *a* logisch “1” ist, andernfalls springt er in den Zustand_4. Auf diese einfache Art lässt sich jede Zeile als eine Übergangsbedingung lesen. Bei grösseren Übergangstabelle n bleibt die Interpretation zwar eindeutig, die Übersichtlichkeit geht aber schnell verloren.

| alte Zustände | | Eingänge | neue Zustände | |
|---|---|---|---|---|
| Name | s(k) | i(k)=ab | Name | s(k+1) |
| Zustand_1 | 00 | 1X | Zustand_2 | 01 |
| Zustand_1 | 00 | 0X | Zustand_4 | 11 |
| Zustand_2 | 01 | 0X | Zustand_2 | 01 |
| Zustand_2 | 01 | 1X | Zustand_3 | 10 |
| Zustand_3 | 10 | X1 | Zustand_1 | 00 |
| Zustand_3 | 10 | 00 | Zustand_3 | 10 |
| Zustand_3 | 10 | 10 | Zustand_4 | 11 |
| Zustand_4 | 11 | XX | Zustand_1 | 00 |

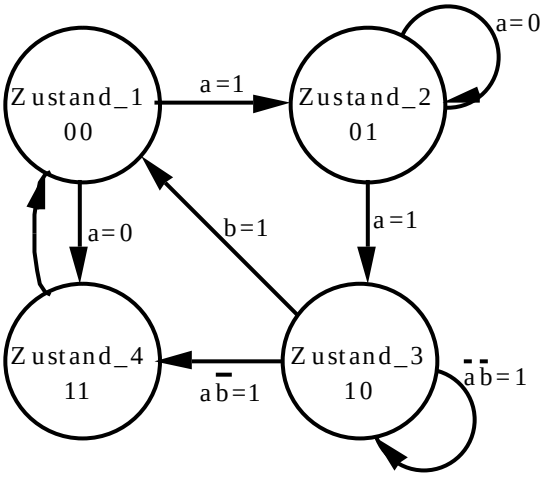
Figur 48: Beispiel einer Übergangstabelle.

Beim Moore und beim Mealy-Automaten (siehe Figur 45 und 46) erzeugen erst die Funktionen $g(s(k))$, respektiv $g(i(k), s(k))$ die eigentlichen Ausgänge. Bei diesen beiden Automatentypen ist somit eine weitere Tabelle, die Ausgangstabelle für eine vollständige Beschreibung des Sequenzers notwendig. Diese Ausgangstabellen stellen aber nichts anderes als eine einfache kombinatorische Verknüpfung der betrachteten Eingänge dar und kann somit durch eine gewöhnliche Wahrheitstabelle dargestellt werden.

### 6.3.2 Zustandsdiagramm

Das Zustandsdiagramm erlaubt die Funktion eines Sequenzers auf eine graphische Art und Weise zu beschreiben. Die Funktionalität kann damit sehr übersichtlich dargestellt werden. Durch das Einführen von Namen und Begriffen, welche Zustände und Übergangsbedingungen beschreiben, kann auf einer höheren Abstraktionsebene gearbeitet werden. Bei sehr grossen und komplexen Sequenzern wird durch eine hierarchische Verschachtelung der Zustandsdiagramme die Übersichtlichkeit im einzelnen Diagramm gewahrt.

In Figur 49 ist dasselbe Beispiel, wie es in der Übergangstabelle von Figur 48 beschrieben ist, als Zustandsdiagramm dargestellt. In diesem Diagramm wird ein Zustand durch einen Knoten (Kreis) dargestellt. Die Pfeile stellen die Übergänge von einem Zustand (Knoten) in einen anderen dar. Unbeschriftete Pfeile weisen auf einen unbedingten Zustandswechsel hin, beschriftete Pfeile stellen bedingte Übergänge dar. Der Übergang von Zustand_3 nach Zustand_4 erfolgt in diesem Beispiel nur, wenn die beiden Eingangsvariablen *a* und *b* die Bedingung $a\bar{b} = 1$ erfüllt ist.




Figur 49: Beispiel eines Zustandsdiagramms.

## 6.4 Analyse von Sequenzerschaltungen

Bei der Analyse eines Sequenzers sind aus dem Schema die Zustands- und eventuell die Ausgangstabelle herzuleiten. Mithilfe der erarbeiteten Tabellen lässt sich das Zustandsdiagramm aufstellen, welches eine übersichtlichere Darstellung des Verhaltens vermittelt.

Die Übergangstabelle und das Zustandsdiagramm sind aus der Schaltung in Figur 50 herzuleiten. Die Frage nach der Funktion der Schaltung kann danach einfach beantwortet werden. Dazu werden zuerst die algebraischen Gleichungen aufgestellt, nach welchen die Flip-Flop Eingänge gehorchen müssen.

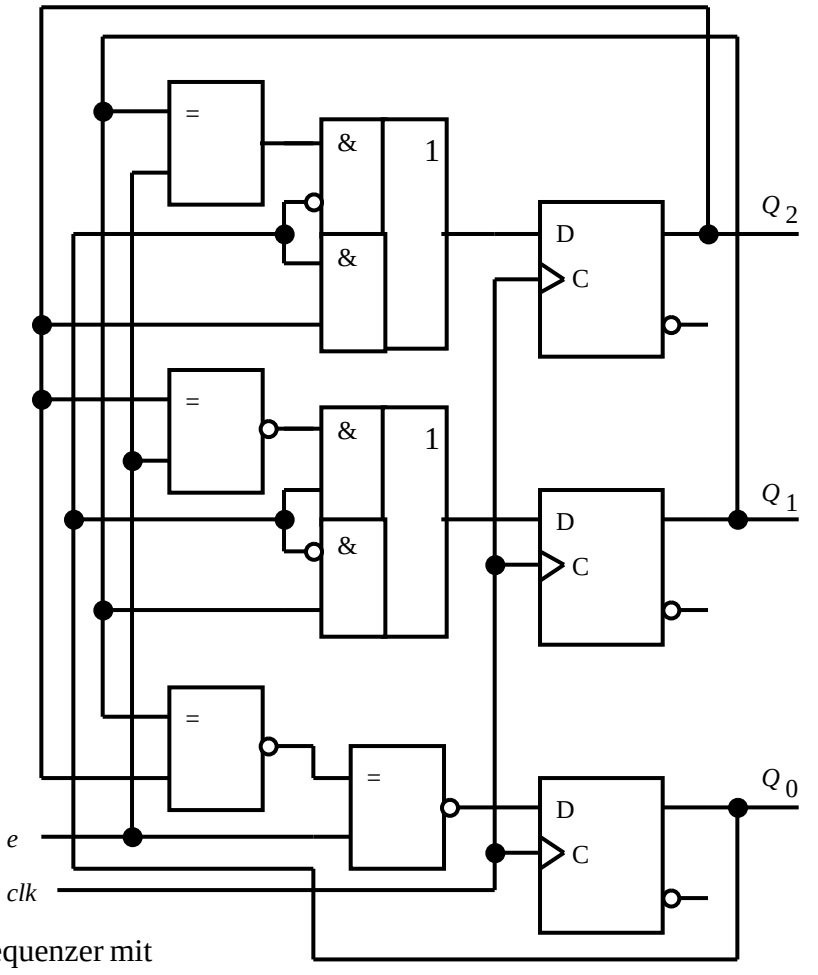
$$Q_2^{n+1} = \overline{(Q_1^n \oplus e)} \cdot \overline{Q}_0^n + Q_0^n \cdot Q_2^n$$

$$Q_1^{n+1} = (Q_2^n \oplus e) \cdot Q_0^n + \overline{Q}_0^n \cdot Q_1^n$$

$$Q_0^{n+1} = (Q_1^n \oplus Q_2^n) \oplus e^n$$

Anhand dieser Gleichungen lässt sich die Übergangstabelle aufstellen, welche aus den logischen Werten des Eingangs *e* und den alten Zuständen der Speicherelemente $Q_i^n$ die neuen Zustände für die Speicher $Q_i^{n+1}$ definiert (siehe Figur 51).

Mit den Informationen der Übergangstabelle kann das Zustandsdiagramm gezeichnet werden. In der Tabelle sind acht verschiedene Zustände codiert, was zu den entsprechenden acht Zuständen (Knoten) im Zustandsdiagramm




Figur 50: Sequenzer mit D-Flip-Flop's.

im Diagramm übertragen, so ist ein vollständige Funktionsbeschreibung des Sequenzers in einer graphischen Form dargestellt (siehe Figur 52). Aus dem Diagramm ist nun gut ersichtlich, dass für den Eingang *e*=1 der Sequenzer im Uhrzeigersinn in einem zyklischen Code vorwärts zählt, bei *e*=0 im selben zyklischen Code rückwärts zählt.

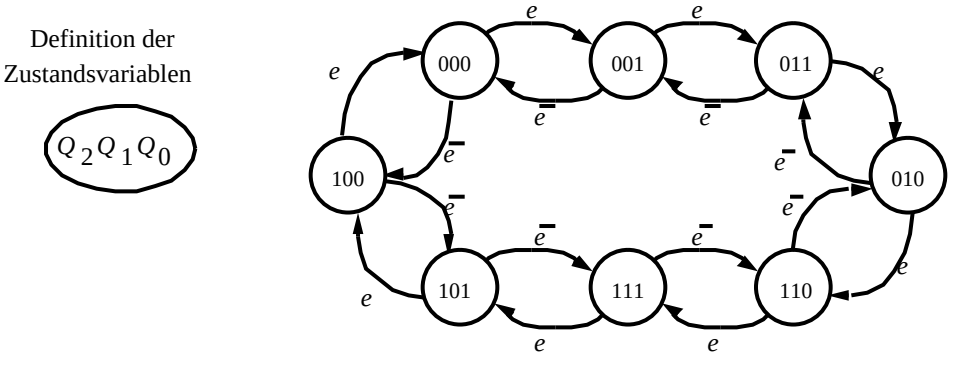
Da bei der Schaltung in Figur 47 die Speicherzustände direkt als Ausgänge verwendet werden, spricht man von einem Medwedjew Automaten.

## 6.5 Synthese von Sequenzerschaltungen

Der Entwurf eines Sequenzers oder Automaten erfolgt über das Zustandsdiagramm, wie es in Figur 49 (siehe auch Figur 52) dargestellt ist. Den Zustän-

| $e$ | $Q_2^n$ | $Q_1^n$ | $Q_0^n$ | $Q_2^{n+1}$ | $Q_1^{n+1}$ | $Q_0^{n+1}$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 |

Figur 51: Übergangstabelle des Automaten in Figur 50.




Figur 52: Zustandsdiagramm des Automaten in Figur 50.

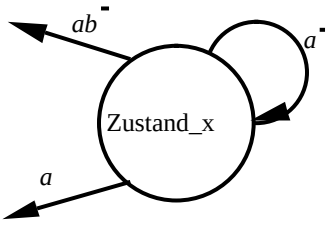
den, namentlich Zustand_1 ... Zustand_4, werden Werte zu ihrer Identifikation zugewiesen. Bei den Mealy- und Moore-Automaten können durch geschickte Zuweisungen von Werten zu den einzelnen Zustandsvariablen optimierte kombinatorische Blöcke entstehen. Den Ablauf im Zustandsdiagramm und damit die Ausgangswerte werden durch die Eingangsvariablen

*a* und *b* bestimmt, welche in diesem Beispiel gerade mit den Ausgängen identisch sein sollen (Medwedjew-Automat). Um vom Zustandsdiagramm zu einer Implementation des Automaten zu gelangen, ist die Übergangstabelle herzuleiten (siehe Figur 48). Jede Zeile der Tabelle repräsentiert einen Übergang (Pfeil) eines alten in einen neuen Zustand unter der erfüllten Übergangsbedingung.

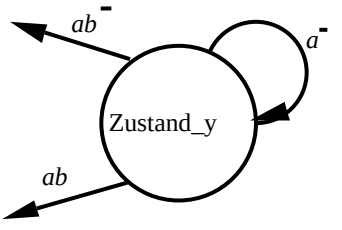
### 6.5.1 Vollständigkeit und Konsistenz

Beim Aufstellen des Zustandsdiagramms sind die folgenden Regeln strikt einzuhalten:

- In keinem der Zustände dürfen gleichzeitig zwei Übergangsbedingungen erfüllt werden können. Dies bedeutet, dass die UND-Verknüpfung (Schnittmenge) von zwei beliebigen Übergangsbedingungen (Pfeile), welche einen Zustand (Knoten) verlassen, immer “0” sein muss.



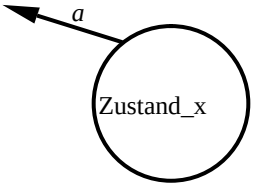

inkonsistent



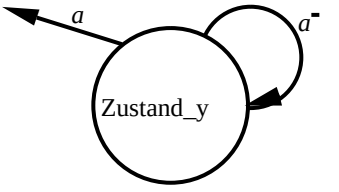

korrekt

- Ebenso dürfen keine Eingangskombinationen auftreten, welche nicht einem Übergang zugewiesen sind. Dies bedeutet, dass die

ODER-Verknüpfung aller Übergangsbedingungen (wegführende Pfeile) immer “1” sein muss.




unvollst‰ndig

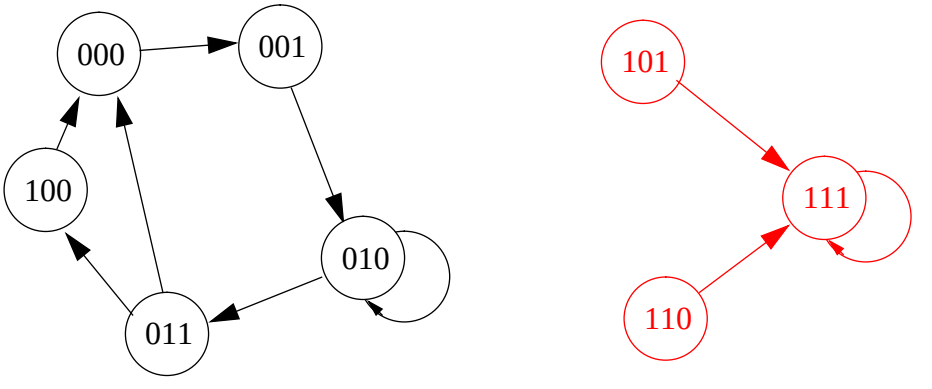



korrekt

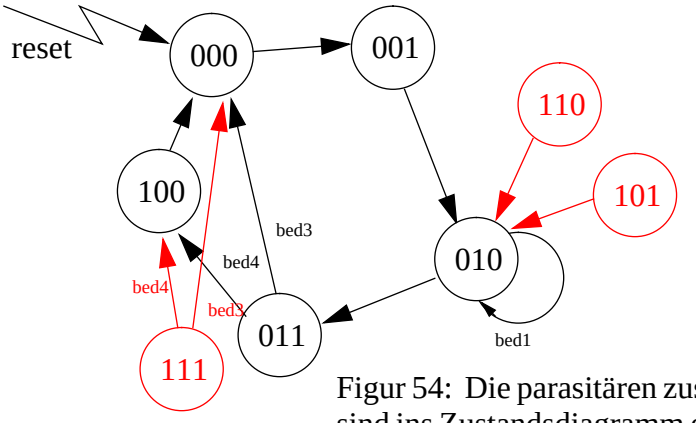
Ist eine dieser Bedingungen nicht erfüllt, so ist das Zustandsdiagramm inkonsistent oder unvollständig und somit nicht realisierbar.

### 6.5.2 Parasitäre Zustände

Vorsicht ist mit den sogenannten parasitären Zuständen geboten (siehe Figur 53). Dies sind Zustände, welche nicht in die Ablaufsteuerung eingebunden sind. Sie treten auf, wenn nicht alle der möglichen $2^n$ Zustandskodierungen (bei $n$ Zustandsvariablen) verwendet werden. Es ist dafür zu sorgen, dass man nicht in einem parasitären Zustand gefangen bleibt und das auch keine Schatten-Zustandsdiagramme ausgeführt werden, welche aus parasitären Zuständen bestehen. Ziel jeder Entwicklung ist es, eine möglichst robuste Schaltung zu erhalten. Die Auswirkung von temporären Fehlern sind auf




Figur 53: Zustandsdiagramm mit Schatten-Zustands-diagramm aus parasitären Zuständen.




Figur 54: Die parasitären zustände aus Figur 53 sind ins Zustandsdiagramm eingebettet.

einen möglichst kurzes Zeitinterval zu begrenzen. Bei einem Sequenzer gibt es Regeln, wie mit den parasitären Zuständen umgegangen werden soll:

- So weit es möglich ist sind parasitäre Zustände zur Deckung mit den echten Zuständen zu bringen (siehe 111 mit 011)
- Alle parasitären Zustände sind auf kürzestem Weg ins Zustandsdiagramm zu führen (110 und 101).
- Aus jedem Zustand kann auf Wunsch in einen Start-Zustand gesprungen werden.

In Figur 54 sind zwei der drei parasitären Zustände aus Figur 53 mit den echten Zuständen zur Deckung gebracht worden. Der dritte parasitäre Zustand führt direkt ins Zustandsdiagramm. Zusätzlich kann durch ein Reset-Signal jederzeit in den Start-Zustand gesprungen werden.

Grosse Sequenzer benötigen meistens eine unverhälnismässig grosse AND-OR-Ebene. Ein Unterteilen in mehrere kleinere Sequenzer reduziert nicht nur die benötigte Siliziumfläche, sondern reduziert auch gleichzeitig die Verzögerungszeiten im kombinatorischen Block.

Es ist nicht immer ratsam, den kombinatorischen Teil der Sequenzer als AND-OR-Ebene aufzubauen. Moderne Synthese-Werkzeug, welche die vielen Freiheitsgrade der Logiksynthese (AND, OR, EXOR, komplexe Gatter) nutzen, sind häufig imstande kompaktere oder schnellere Sequenzer zu erzeugen, als dies mit den klassischen PLA-basierten Sequenzerstrukturen der Fall ist.

### 6.5.3 Hazards in Sequenzern

Kombinatorische Schaltungen sind potentielle Quellen für die Produktion von Hazards. Da in einem Sequenzer kombinatorische Schaltungsblöcke vorhanden sind, besteht auch hier die Gefahr von unerwünschten Hazards. Betrachtet man die in den Figuren 45 bis 47 skizzierten Sequenzer-Struktu-

ren, so stellt man fest, dass nur der kombinatorische Block *g* Hazards am Ausgang zu produzieren vermag. Nur bei Medwedjew-Automaten entstehen keine Hazards. Bei den anderen beiden Automaten-Typen können mit einem zusätzlichen nachgeschalteten Register die Hazards unterdrückt werden, wobei jedoch eine zusätzliche Verzögerung um eine Taktperiode in Kauf genommen werden muss.

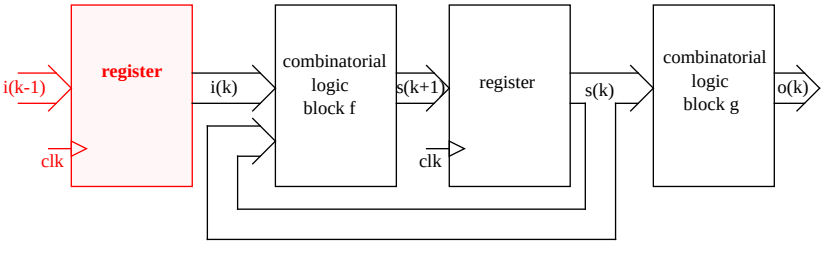
### 6.5.4 Einsynchronisation

Betrachtet man die Implementationsstrukturen eines endlichen Automaten (Figuren 45, 46 und 47), so ist ersichtlich, dass durch den kombinatorischen Block *g* sich die Setup-Zeiten der Eingänge erhöhen. Solange sich die Eingänge des Automaten synchron zum Takt ändern ergeben sich daraus keine Setup-Zeit Verletzungen. Probleme können dann entstehen, wenn sich die Eingänge völlig asynchron zum Takt ändern können. Durch die unterschiedlichen Laufzeiten durch den kombinatorischen Block, können an den Speichereingängen kurzzeitig unerwartete Eingangskombinationen auftreten, welche zu einem falschen Zustand (eventuell parasitären Zustand) führen können.

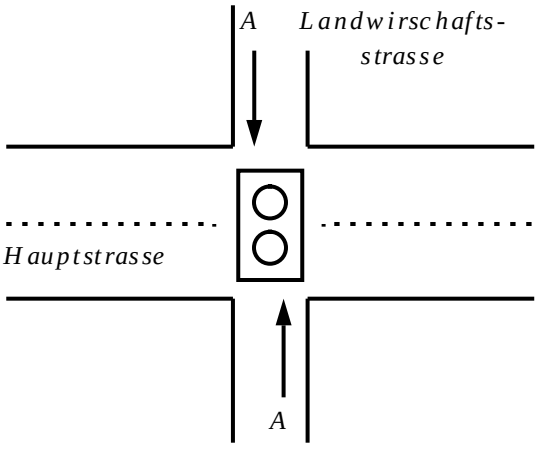
Eine Möglichkeit, die Problematik asynchroner Eingänge bei Sequenzern etwas zu entschärfen, ist die Einsynchronisation solcher Signale. In Figur 55 ist gezeigt, wie bei einem Mealy-Automaten durch ein vorgeschaltetes Register eine Einsynchronisation vorgenommen werden kann. Die erhöhte Sicherheit erkauft man sich aber mit einer grösseren Verzögerungszeit.

### 6.5.5 Entwurfsbeispiel: Lichtsignalsteuerung

Anhand einer einfachen Lichtsignalsteuerung soll ein Moore Automat synthetisiert werden. Die Funktionsweise des Lichtsignals sei wie folgt beschrieben.



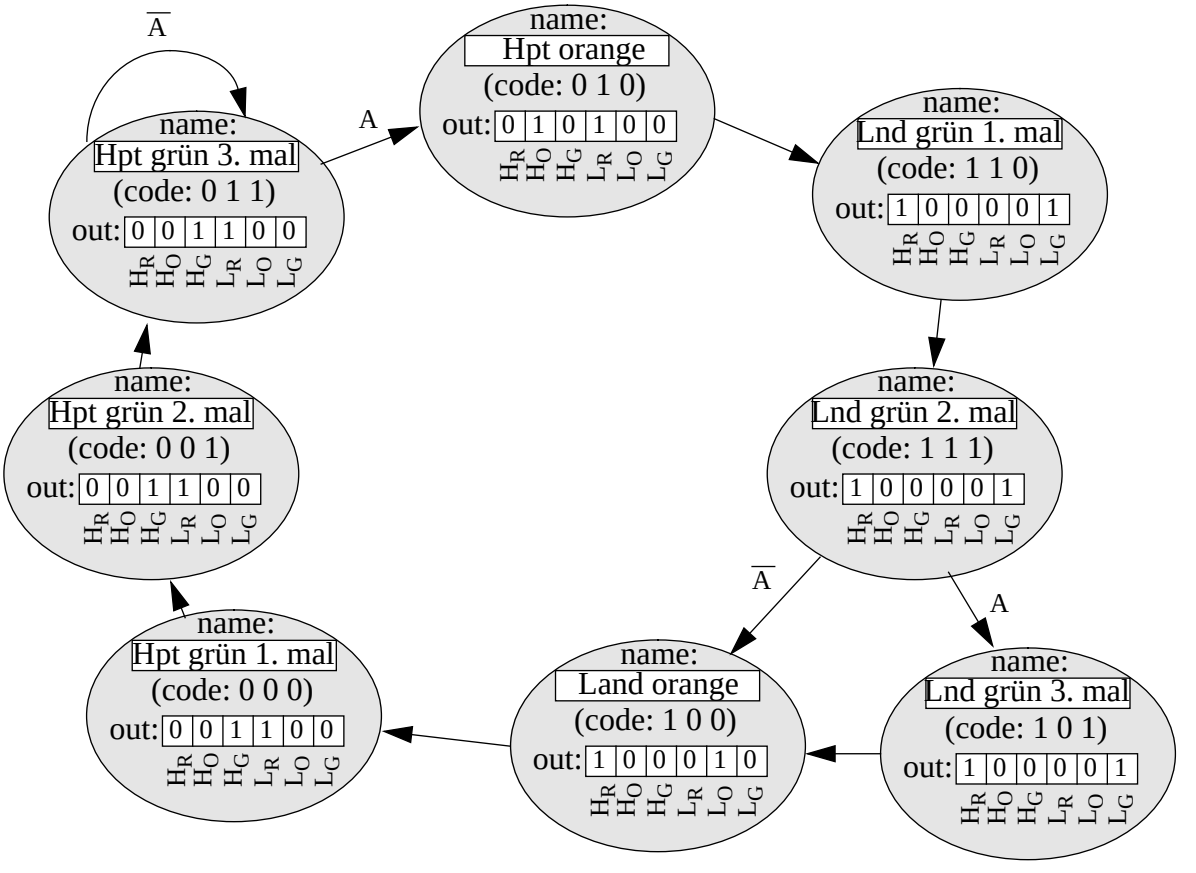

Figur 55: Einsynchronisation in Moore-Automaten.




Figur 56: Lichtsignalsteuerung.

**Problemstellung:** Eine stark befahrene Hauptstrasse weist eine Kreuzung mit einer selten befahrenen Landwirtschaftsstrasse auf. Die Landwirtschaftsstrasse weist vor der Kreuzung einen Sensor auf, welcher die Anwesenheit von Landwirtschaftsfahrzeugen detektieren kann ($A$=1, Fahrzeug vorhanden). Warten keine Fahrzeuge auf der Landwirtschaftsstrasse, so soll das Signal auf der Hauptstrasse grün ($H_G$=1) und dasjenige der Landwirtschaftsstrasse rot ($L_R$=1) sein. Wartet ein Fahrzeug auf der Landwirtschaftsstrasse, so wechselt das Signal auf der Hauptstrasse zuerst auf orange ($H_O$=1) und eine Zeiteinheit später auf rot ($H_R$=1). Gleichzeitig wird das Signal der Landwirtschaftsstrasse auf grün ($L_G$=1) gestellt. Nach zwei Zeiteinheiten wechselt das Signal auf der Landwirtschaftsstrasse auf orange ($L_O$=1) und danach auf rot, sofern keine weiteren Landwirtschaftsfahrzeuge erscheinen. Andernfalls bleibt das Signal der Landwirtschaftsstrasse im gesamten maximal 3 Zeiteinheiten auf grün geschaltet. Das Signal der Landwirtschaftsstrasse kann frühestens nach 4 rot-Zyklen wieder auf grün geschaltet werden.

In einem ersten Schritt zur Entwicklung eines Automaten ist das Zustandsdiagramm zu erstellen. Gemäss den Regeln aus der Problemstellung werden die Zustände und deren Übergänge graphisch dargestellt. Da in diesem Beispiel nur acht verschiedene Zustände auftreten, wurden sie mit drei Bits codiert, welche die Ausgänge der Speicher darstellen werden. Die geforderten Signale für die beiden Lichtsignale mit den drei Lampen müssen somit aus den Zuständen erzeugt werden. Für die Realisierung dieser Schaltung wird somit ein Moore-Automat eingesetzt werden. Aus dem Zustandsdiagramm lässt sich nun sehr einfach die Übergangstabelle erstellen, indem für jede Übergangsbedingung eine Zeile in der Tabelle aufgestellt wird. In Figur 58 ist die entsprechende Übergangstabelle dargestellt.




Figur 57: Zustandsdiagramm der Lichtsignalsteuerung.

| $A$ | $Q_2^n$ | $Q_1^n$ | $Q_0^n$ | $Q_2^{n+1}$ | $Q_1^{n+1}$ | $Q_0^{n+1}$ |
|---|---|---|---|---|---|---|
| *X* | *0* | *0* | *0* | *0* | *0* | *1* |
| *X* | *0* | *1* | *0* | *1* | *1* | *0* |
| *X* | *1* | *0* | *0* | *0* | *0* | *0* |
| *X* | *1* | *0* | *1* | *1* | *0* | *0* |
| *X* | *1* | *1* | *0* | *1* | *1* | *1* |
| *X* | *0* | *0* | *1* | *0* | *1* | *1* |
| *0* | *0* | *1* | *1* | *0* | *1* | *1* |
| *1* | *0* | *1* | *1* | *0* | *1* | *0* |
| *0* | *1* | *1* | *1* | *1* | *0* | *0* |
| *1* | *1* | *1* | *1* | *1* | *0* | *1* |

Figur 58: Übergangstabelle für die Lichtsignalsteuerung.

| $Q_2$ | $Q_1$ | $Q_0$ | $H_G$ | $H_O$ | $H_R$ | $L_G$ | $L_O$ | $L_R$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

Figur 59: Ausgangstabelle der Lichtsignalsteuerung.

Aus den codierten Zustandsbits sind nun noch die Ansteuerungen für die beiden Lichtsignalanlagen mit den drei Signalfarben zu generieren. Dazu ist eine sogenannte Ausgangstabelle zu erstellen. In Figur 59 ist eine solche Ausgangstabelle dargestellt. Die drei Zustandsbits $Q_i$ bilden die Eingänge in der Wahrheitstabelle, die Stellgrössen der Lichtsignale die Ausgänge.

Mit der Ausgangstabelle und dem Zustandsdiagramm oder der Übergangstabelle ist die Funktion des Moore-Automaten eindeutig bestimmt. Durch Vereinfachungen mittels Karnaugh-Diagrammen der beiden Tabellen lässt sich nun eine logische Schaltung für die Lichtsignalanlage aufbauen. Aus der Übergangstabelle erhält man die Beziehungen:

$$Q_2^{n+1} = \overline{Q}_0 Q_1 + Q_0 Q_2$$
$$Q_1^{n+1} = \overline{Q}_0 Q_1 + Q_0 \overline{Q}_2$$
$$Q_0^{n+1} = \overline{Q}_1 \overline{Q}_2 + \overline{A} Q_0 \overline{Q}_2 + \overline{Q}_0 Q_1 Q_2 + A Q_1 Q_2$$

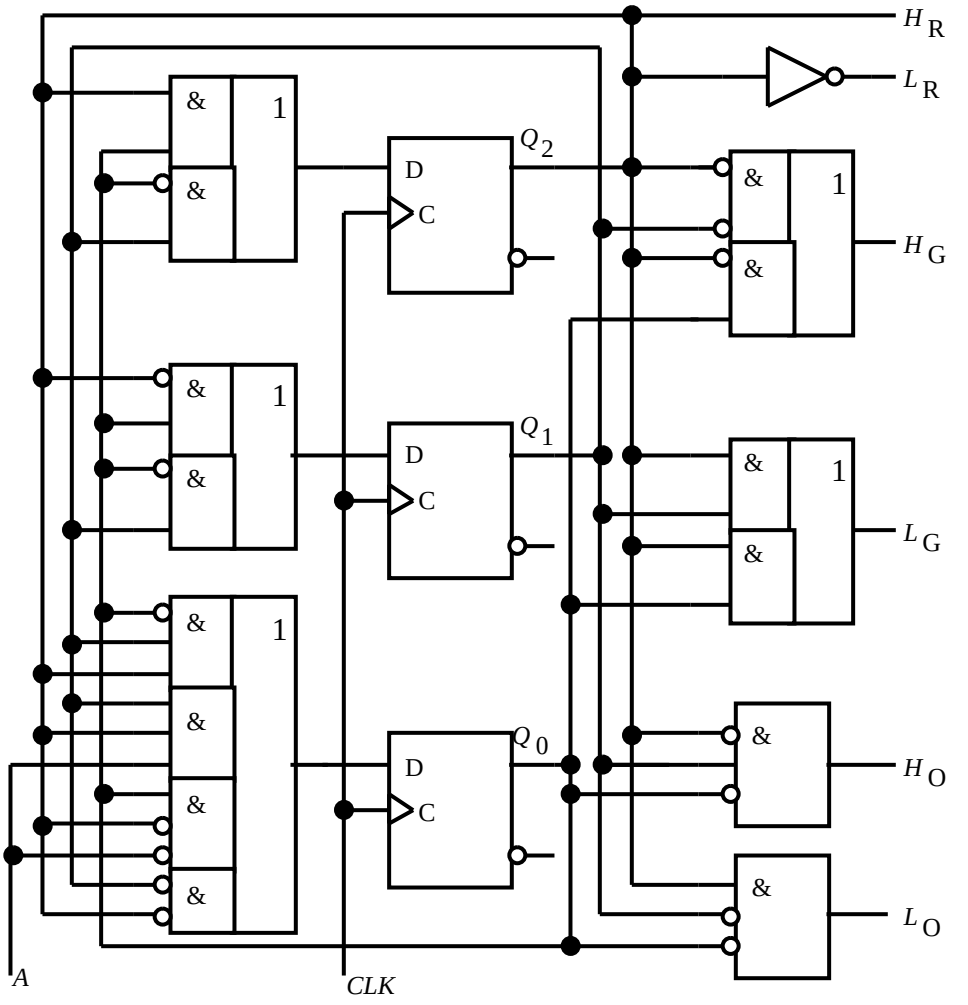
Aus der Ausgangstabelle erhält man die folgenden einfachen booleschen Gleichungen:

$$H_G = Q_0 \overline{Q}_2 + \overline{Q}_1 \overline{Q}_2 \qquad L_G = Q_0 Q_2 + Q_1 Q_2$$
$$H_O = \overline{Q}_0 Q_1 \overline{Q}_2 \qquad L_O = \overline{Q}_0 \overline{Q}_1 Q_2$$
$$H_R = Q_2 \qquad L_R = \overline{Q}_2$$

Aus diesen vereinfachten Gleichungen lässt sich nun direkt das Logikschema der Lichtsignalsteuerung in Form eines Moore-Automaten erstellen, wie es in Figur 60 abgebildet ist.

Endliche Automaten können für verschiedene Aufgaben herangezogen werden. Mit ihnen können Zähler verschiedenster Art aufgebaut werden, wie dies in Figur 50 angedeutet wurde. Das Hauptanwendungsgebiet für endliche Automaten sind vor allem Ablaufsteuerungen, wie sie fast ausnahmslos in jeder komplexeren Digitalschaltung anzutreffen sind. Dabei wird der Ablauf und das Zusammenspiel von unterschiedlichen Hardwareeinheiten über Steuersignale, welche der Automat generiert, kontrolliert. Ein etwas simples Beispiel dazu stellt die behandelte Lichtsignalsteuerung dar, da hier nur das Kommando für die Signallampen erzeugt wurde.

Ein anspruchvolleres Beispiel sei dem interessierten Leser als Aufgabe kurz umrissen.




Figur 60: Schema der Lichtsignalsteuerung.

### 6.5.6 Entwurfsbeispiel: Black-Jack Dealer

Es ist eine logische Schaltung zu entwerfen, welche das Verhalten des Gebers beim Black-Jack Kartenspiel nachsimuliert.

**Spielbeschreibung:** Beim Black-Jack Spiel ist ein Kartengeber mit einem oder mehreren Spielern beteiligt. Ziel des Spieles ist es, durch das Verlangen von einer Karte nach der anderen, möglichst nahe an 21 Kartenpunkte zu gelangen, diese aber nicht zu überschreiten. Die Jasskarten weisen Punktzahlen zwischen 1 und 11 auf, wobei das Ass mit 11 oder mit 1 Punkt verrechnet werden darf.

**Spielregeln:** Die Spieler bestimmen ihr Verhalten beim Spiel selbst, der Kartengeber als Spielkasinoangestellter darf nicht auf Risiko spielen, sondern muss, wenn er selbst Karten nimmt, dies nach ganz bestimmten Regeln tun. Der Kartengeber fordert für sich eine Karte nach der anderen an, bis seine Punktzahl 16 überschritten hat. Dabei verrechnet er für das Ass 11 Punkte, sofern er die Maximalpunktzahl 21 nicht überschritten hat, andernfalls verrechnet er für das Ass 1 Punkt und spielt gemäss seinen Regeln weiter.

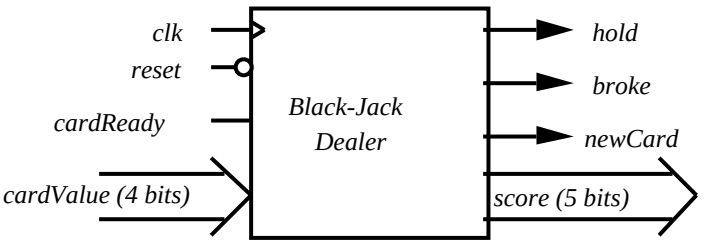
**Aufgabenstellung:** Es ist eine logische Schaltung zu entwerfen, welche das Verhalten des Black-Jack Dealers nachempfindet. Zur Kommunikation mit der Umwelt weise die Schaltung die folgenden Einsignale

- Neue Karte bereit (card ready: *cardReady*)
- Punktzahl der neuen Karte (card value: *cardValue*, 4 Bits)
- Spielstrat (reset aktiv tief: *reset*)
- Systemtakt (clock: *clk*)

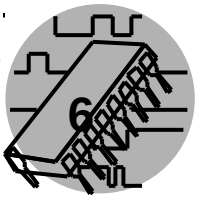
und die folgenden Ausgänge auf:

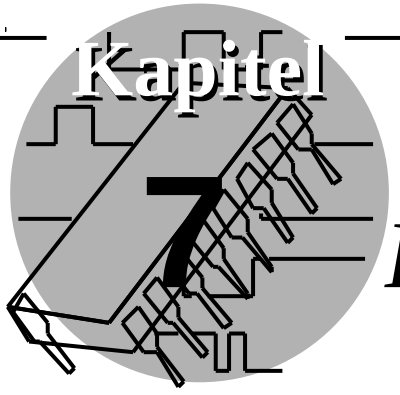
- Punktzahl überschritten (broke: *broke*)
- Halt, genügend Punkte gesammelt (hold: *hold*)
- Anforderung für eine weitere Karte (new card: *newCard*)
- Gesamtpunktzahl am Spielende angeben (score: *score*, 5 Bits)

Die Schaltung ist in Figur 61 als Black-Box dargestellt.




Figur 61: Black-Jack Dealer Schnittstelle mit dem Bediener.






# *Fuzzy Logik*

## 7.1 Lernziele

Nach dem Studium des vorliegenden Kapitels sind die folgenden Aufgaben zu lösen oder die folgenden Fragestellungen präzise zu beantworten:

- Welche Idee liegt der Fuzzy Logik zugrunde.
- Was sind die Vor- und Nachteile eines Fuzzy Logik Systems gegenüber einer konventionellen Implementation (z.B. Regelung).
- Begriffe wie unscharfe Menge, Zugehörigkeitgrad, linguistische Variable, Fuzzy Operatoren sind bekannt.
- Die arbeistweise eines Fuzzy Logik Systems ist bekannt.
- Die unterschiedlichen Inferenz- und Defuzzifizier-Methoden sind bekannt.
- Anwendungsgebiete der Fuzzy Logik sind bekannt.
- Ein einfaches Fuzzy Logik System kann selbständig aufgebaut werden.

## 7.2 Geschichtlicher Hintergrund

Aufgeschreckt durch japanische Produkte im Consumer-Bereich, welche sich auf elegante Art und Weise die Fuzzy-Set Theorie zunutze machen, wurde die westliche Industrie-Welt erst in den letzten beiden Jahren auf das Potential der Fuzzy-Logik aufmerksam. Erfahrungen mit industriellen Produkten sind somit vor allem in Japan vorhanden. Durch die Angst, den Japanern wieder einmal auf einem zukunftsträchtigen Gebiet die Führung überlassen zu müssen, sind Ingenieure mit praktischen Erfahrungen in Europa über Nacht zu den gefragtesten Experten avanciert. Postuliert wurde die Fuzzy-Set Theorie aber nicht im fernen Osten, sondern bereits vor über einem Viertel Jahrhundert vom amerikanischen Professor Lotif A. Zadeh an der Universität Berkeley. In der Zwischenzeit wurde die Theorie von ihm verfeinert, durch andere Forschungsteams aufgegriffen und erweitert.

Die Fuzzy-Set Theorie widersprach dem damaligen Geist der Ingenieurwissenschaften mit ihrem Hang zur technischen Perfektion durch immer genaueres Messen, Rechnen, Steuern und Regeln. Prof. Zadeh wollte mit seiner neuen Theorie den Ingenieuren ein Werkzeug schaffen, mit welcher sie die Natur und ihre Gesetzmässigkeiten besser beschreiben konnten. Dabei distanzierte er sich von der allgemein beachteten Meinung, dass nur exakte Angaben zu einem brauchbaren Resultat führen können. Es ist kaum erstaunlich, dass er mit diesem Ansatz gerade im Zeitalter der sich rasch entwickelnden Computertechnik kaum verstanden wurde. Das Zitat von Henri Matisse (1947)

*"Genauigkeit ist nicht Wahrheit"*

trifft genau den Kerngedanken der Fuzzy-Set Theorie. Mit seiner Theorie führt Prof. Zadeh den Blick des Ingenieurs weg von übertriebenen Genauigkeitsbetrachtungen, hin zu einem besseren Verständnis der Gesetzmässigkeiten der Natur und somit zu einem besseren Systemverständnis.

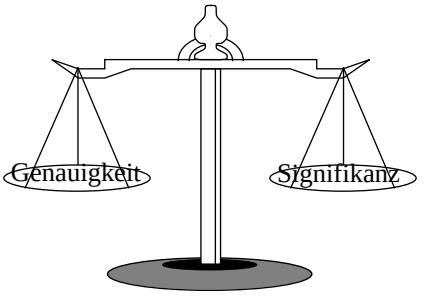
## 7.3 Philosophie der Fuzzy-Logik

Die Ingenieurwissenschaften sind sogenannte exakte Wissenschaften. Das heisst, die Ingenieure rühmen sich, alles genau und bis auf die letzte Stelle hinter dem Komma ausrechnen zu können. Es stellt sich die berechtigte Frage, was nützt einem diese Präzision, wenn sich die realen Vorgänge in der Natur selten genau beschreiben lassen. Diese manchmal übertriebene Präzision verteuert einerseits die Produkte, andererseits lenken sie die Entwicklungsingenieure von den wichtigeren, übergeordneten Zusammenhängen ab. Das heisst, grosse Genauigkeit und hohe Komplexität lassen sich nicht immer unter einen Hut bringen, wie Professor Zadeh in seinem Prinzip der Unvereinbarkeit feststellte:

> *"In dem Masse, in welchem die Komplexität eines Systems ansteigt, nimmt unsere Fähigkeit präzise und damit signifikante Aussagen über sein Verhalten zu machen ab. Präzision und Signifikanz schliessen sich ab einem gewissen Komplexitätsgrad gegenseitig aus."*

Liest man dieses Prinzip der Unvereinbarkeit, so könnte man fast dazu verleitet werden, in Analogie zu gewissen Gesetzen in der Elektronik, wie zum Beispiel dem konstanten Verstärkungs-Bandbreiteprodukt, ein Präzisions-Signifikanzprodukt zu definieren, welches als charakteristische Grösse für die Lösung einer Aufgabe als konstant anzusehen wäre. Der Grundgedanke der Fuzzy-Set Theorie wird sich zwar kaum formal festhalten, sondern eher als Leitgedanke formulieren lassen: Durch Abstraktion und ohne unnötige

Genauigkeit gelingt es, Wichtiges von Unwichtigem zu trennen und komplexe Probleme zu vereinfachen. Es gilt somit, das Gleichgewicht zwischen übertriebener Genauigkeit und Signifikanz neu zu überdenken.




Die Fuzzy-Logik verzichtet auf übertriebene Genauigkeit, bietet dafür aber die Möglichkeit, Systeme auf einfache Art und Weise verbal zu beschreiben. Die Fuzzy-Set Theorie (Lehre der unscharfen Mengen) kann somit als mathematisches Konstrukt verstanden werden, welches von der Erkenntnis bestimmt ist, dass Präzision nicht zum Selbstzweck werden darf. Das Postulat, überall dort auf übertriebene Genauigkeit zu verzichten, wo sie für die Lösung eines Problems nicht lebensnotwendig ist, stellt das traditionelle Weltbild vieler Ingenieure auf den Kopf. Fuzzy-Logik ist also eine präzis definierte, quantitative mathematische Theorie für das Modellieren und Räsonieren mittels linguistischer Ungewissheit.

## 7.4 Wozu eine neue Theorie einführen?

Es stellt sich mit Recht die Frage, warum in klassischen Gebieten, wie zum Beispiel der Mustererkennung, der Expertensysteme oder der Regelungstechnik, wo zum Teil jahrzehntelange Erfahrungen vorhanden sind, plötzlich nach einer neuen technischen Grundlage gearbeitet werden soll. Wenn zum Beispiel sowohl mit der konventionellen Regelungstechnik, als auch mit der neuen Fuzzy-Set Theorie in der gleichen Zeit und mit dem gleichen personellen und materiellen Aufwand vergleichbar gute Lösungen gefunden werden, so bleibt die Frage nach dem “wozu” sicher unbeantwortet im Raume stehen.

Soll ein System nach der klassischen Regelungstechnik entworfen werden, so fährt man am besten, wenn zuerst die Regelstrecke mathematisch modelliert wird. Alltägliche Beispiele aus der Regelungstechnik zeigen aber, dass immer wieder Fälle auftreten, in welchen Ingenieure nicht, oder nur mit grossem Aufwand und Spezialwissen, in der Lage sind, die Realität überhaupt zu modellieren. Das Bilden von aussagekräftigen Modellen, wie sie für die Pro-

blemlösung mit der Regelungstechnik nötig sind, ist nicht immer trivial. Neben Schwierigkeiten beim Auffinden der physikalischen Differentialgleichungen ist meist auch ein Abschätzen der unterschiedlichen Einflussgrössen nötig, um wesentliche von vernachlässigbaren Einflüssen zu trennen.

In der Datenanalyse und Mustererkennung werden technische Lösungen nicht selten sehr komplex. Der Grund ist vielfach in der schwierigen mathematischen Modellierung und der aufwendigen technischen Implementation von charakteristischen Eigenschaften zu finden. Da bei der Fuzzy-Set Theorie die Vorgehensweise völlig anders und damit äusserst unkonventionell ist, resultieren manchmal vor allem in diesen beiden Anwendungsgebieten sehr elegante Lösungsansätze. Der Entwicklungsingenieur muss sein System nicht mit mathematischen Gleichungen modellieren, um es handhaben zu können, sondern er muss das gewünschte Verhalten des Systems verstehen und beschreiben können, um es auf eine verbale Art und Weise zu charakterisieren, zu regeln oder um Schlussfolgerungen zu ziehen. Der Ingenieur sucht also nicht primär nach strengen mathematischen Gleichungen, sondern er stellt Regeln auf, nach denen sich das System zu verhalten hat.

Es ist und darf nicht das Ziel der Fuzzy-Set Theorie sein, die klassische Datenanalyse, Mustererkennung oder Regelungstechnik zu ersetzen, sondern nur, sie in Beispielen von schwer modellierbaren Problemstellungen durch eine geeignete neue Methode zu ergänzen. Auch sollten Ingenieure nicht dazu verleitet werden, bestehende Lösungen einfach durch Fuzzy-Logik Ansätze zu ersetzt. Vorsicht beim Einsatz der Fuzzy-Set Theorie ist vor allem in der Regelungstechnik walten zu lassen, denn es wäre zum Beispiel unsinnig, die Klasse der linearen Regelstrecken mit den höchst nichtlinearen Ansätzen der Fuzzy-Set Theorie regeln zu wollen.

## 7.5 Eigenschaften von Produkten basierend auf der Fuzzy-Set Theorie

Durch die bestehende Euphorie wird der Eindruck erweckt, dass viele Probleme in der Informatik und Elektronik mit der Anwendung der Fuzzy-Set Theorie besser und schneller gelöst werden könnten als dies auf den klassischen Wegen der Fall ist. Inwieweit dies den Tatsachen entspricht, ist schwer nachvollziehbar und nur durch eigene Erfahrungen abzuschätzen. Einige Eigenschaften, mit welchen die Fuzzy-Logik und die Fuzzy-Control Systeme immer wieder charakterisiert werden, seien hier bewusst kommentarlos aufgeführt und entsprechend mit Vorsicht zu geniessen:

- Fuzzy-Logik wird in den verschiedensten Anwendungsgebieten auftauchen und alle möglichen Produkte benutzerfreundlicher gestalten.
- Maschinen treffen blitzschnelle Entscheidungen aufgrund einer Kombination verschiedenster Messwerte, wobei diese nicht präziser eingegeben werden müssen als Schätzungen eines erfahrenen Menschen.
- Der Einsatz unscharfer Logik führt zur Senkung des Entwicklungsaufwandes, zur Verbesserung der Wartbarkeit und Robustheit der Systeme.
- Relativ einfache und billige Geräte können Aufgaben übernehmen, die einst grosse Rechenleistungen verlangten.

Die eine oder andere der obigen Aussagen mag zutreffen. Was aber ganz bestimmt zutrifft, ist der kreative Freiraum, den die Fuzzy-Set Theorie beinhaltet. Um diesen Freiraum auszuschöpfen, ist ein solides und breites Fachwissen vonnöten, damit tatsächlich neue und bessere Lösungen erarbeitet werden können und nicht einfach nur Bestehendes mit einer neuen Technologie nachgebaut wird. Natürlich gilt es bei den heutigen Produkten auch vermehrt den Marktfaktoren Rechnung zu tragen. Dabei sind Produktionskosten und nicht zuletzt Werbeeffekte wesentliche Kriterien für den Erfolg eines Produktes. Ein extremes, nicht zur Nachahmung empfohlenes Beispiel, ist ein Toilettenpapier in Japan, welches sich anscheinend mit dem Aufdruck Fuzzy-Logik besser vermarkten lässt.

## 7.6 Die Fuzzy-Set Theorie

Was sind nun diese sogenannten Fuzzy-Sets, und was lässt sich damit anfangen. In der klassischen Mengenlehre werden Elemente einer bestimmten Menge zugeordnet. Diese Zuweisungen sind immer eindeutig und klar definiert ($x \in X$). Die Zahl 164 wird zum Beispiel der Menge der Zahlen kleiner 165 ($K = \{x \mid x < 165\}$) zugeordnet, währenddem die Zahl 166 nicht in dieser Menge enthalten ist.

### 7.6.1 Unscharfe Mengen

Die Fuzzy-Set Theorie ist eine Verallgemeinerung oder Erweiterung der klassischen Mengenlehre. Der Begriff Fuzzy-Set wird hier am besten mit unscharfer Menge übersetzt. Im Gegensatz zur klassischen Mengenlehre sind in der Lehre der unscharfen Menge Elemente nicht immer zu 100% einer Menge zuzuordnen. Dies bedeutet, dass Elemente nur zu einem gewissen Grad einer unscharfen Menge zugeordnet werden. Dieser Grad wird durch

einen Zugehörigkeitsgrad oder eine Zugehörigkeitsfunktion festgelegt. Eine unscharfe Menge $\tilde{A}$ besteht somit immer aus Wertepaaren, den Elementen $x$ und ihren entsprechenden Zugehörigkeitsgraden $\mu$.

$$\tilde{A} = \{(x,\mu_{\tilde{A}}(x)) \mid x \in X\}$$

Ein Zugehörigkeitsgrad 1 bedeutet, dass das entsprechende Element zu 100% einer Menge angehört, der Zugehörigkeitsgrad 0 deutet an, dass das Element nicht in der Menge vorhanden ist. Zugehörigkeitsgrade zwischen diesen beiden Extremwerten bedeuten, dass das Element nur zu einem entsprechenden Grad der unscharfen Menge angehört. Die klassischen Mengen sind somit ein Spezialfall der unscharfen Mengen, in denen die Zugehörigkeitsgrade nur die beiden diskreten Werte 0 und 1 annehmen können.

Als Beispiel sei die unscharfe Menge $\tilde{A}$ beschrieben, welche die Menge aller ganzen Zahlen nahe bei 10 enthalten soll. Eine mögliche Definition dieser Menge sieht wie folgt aus:

$$\tilde{A} = \left\{ x, \mu_{\tilde{A}}(x) \mid \mu_{\tilde{A}}(x) = (1 + (x-10)^2)^{-1} \right\}$$

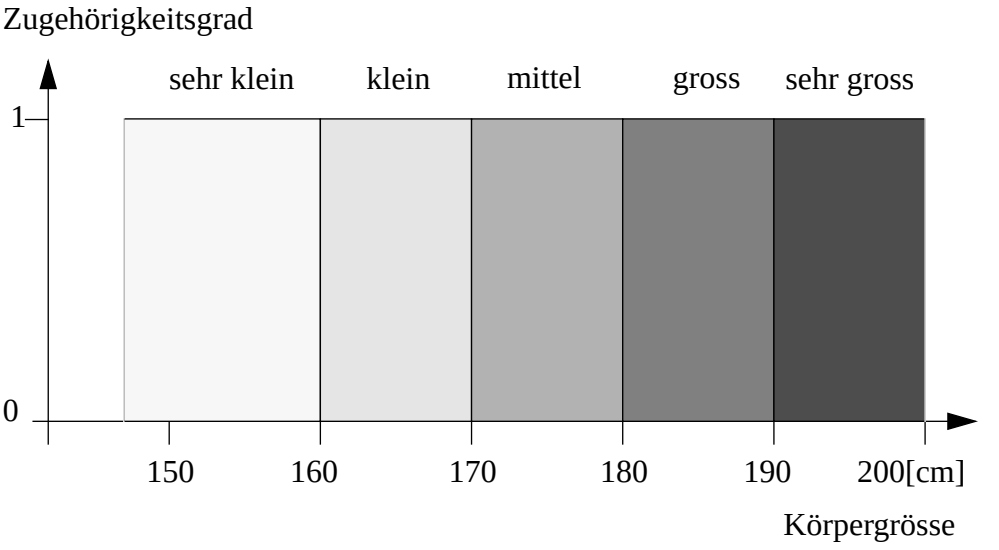
Berechnet man die Elemente, welche dieser unscharfen Menge angehören, so erhält man die Menge $\tilde{A}$ ={.., (7,0.1), (8,0.2), (9,0.5), (10,1.0), (11,0.5), (12,0.2), (13,0.1), ...}.

### 7.6.2 Linguistische Variablen

Die Fuzzy-Sets sind ein geeignetes Mittel, spezielle Begriffe des menschlichen Denkens darzustellen. Betrachtet man die sprachlichen Ausdrucksformen, so stellt man fest, dass die weitaus häufigsten Aussagen einen unscharfen Charakter haben und sich selten durch präzise Zahlen ausdrücken lassen.

- Gesucht ist ein junger, grosser, schlanker Mann mit mittellangen braunen Haaren.
- Leicht bewölkt bis bedeckt, am Abend ist mit Regenschauer zu rechnen.

All diese beschreibenden Ausdrücke geben uns zwar ein genügend klares Bild, sind aber für eine Verarbeitung durch einen Computer kaum geeignet. Mit Hilfe der sogenannten linguistischen Variablen (Fuzzy-Variable) lassen



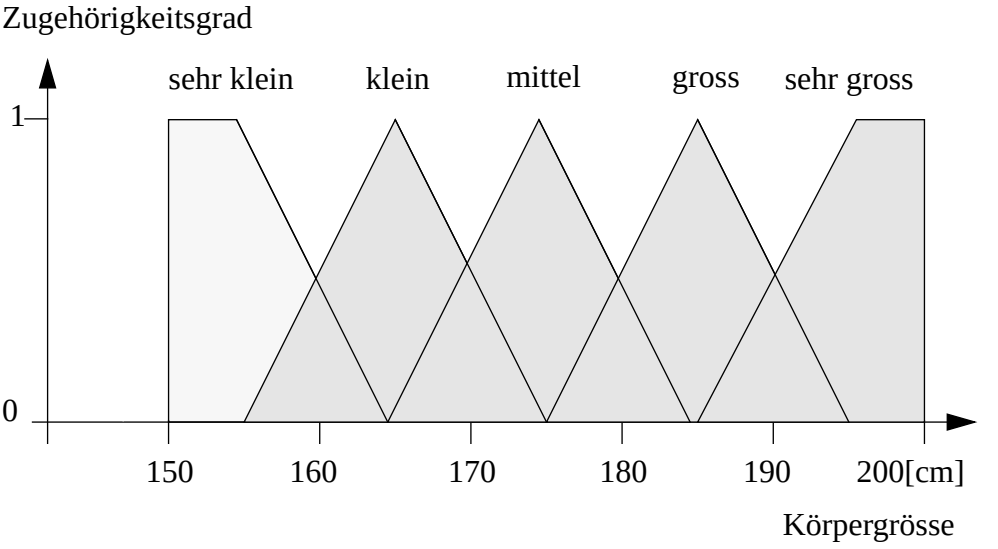

Figur 62: Diskrete Darstellungsform der mehrwertigen Variablen Körpergrösse.

sich diese beschreibenden Ausdrücke erfassen, ohne dass sie ihren unscharfen Charakter verlieren aber trotzdem mit einem Computer verarbeitet werden können. Linguistische Variablen sind keine Zahlen, sondern Worte oder Sätze. Sie setzen sich zusammen aus einer definierten Anzahl unterschiedlicher Fuzzy-Sets.

Als Beispiel wird die Körpergrösse eines Mannes in Form einer diskreten und einer linguistischen Variablen definiert. Die linguistische Variable “Körpergrösse” setzt sich zusammen aus den fünf unscharfen Mengen “sehr-klein”, “klein”, “mittel”, “gross” und “sehr-gross”, welche häufig auch nur als Terme bezeichnet werden. Aus Figur 63 ist ersichtlich, dass ein Mann bei einer Grösse von 179cm als mittel, bei 181cm bereits eindeutig als gross eingestuft wird. Diese “digitale” Trennung zwischen mittel und gross entspricht ganz und gar nicht der menschlichen Denk- und Ausdrucksweise. Bei der linguistischen Variablen in Figur 63 hingegen existiert ein fliessender Übergang zwischen den unscharfen Mengen. Ein Mann der Grösse 179cm wird zum Grad 0.6 zur Menge der Männer mittlerer Grösse und zum Grad 0.4 zur Menge der grossen Männer gezählt.

### 7.6.3 Fuzzy-Operatoren

Auf den Fuzzy-Sets sind zahlreiche Operatoren, wie Addition, Subtraktion, algebraische und kartesische Produkte usw. definiert. In diesem Text wird nur auf diejenigen Operatoren eingegangen, welche speziell in der Fuzzy-Logik Theorie benötig werden. Zieht man Quervergleiche mit der bool’schen Logik,




Figur 63: Linguistische-Variable für die Körpergrösse eines Mannes. Die Fuzzy-Variable “Körpergrösse” setzt sich zusammen aus fünf unterschiedlichen Fuzzy-Sets (Terme).

so stösst man auf die UND, ODER und NICHT Operatoren. Sehr ähnliche Operatoren existieren in der Fuzzy-Set Theorie, welche als UND, ODER und NICHT Funktionen herbeigezogen werden können:

**Tabelle 3:**

| Operatoren | Fuzzy Zugehörigkeitsfunktion |
|---|---|
| UND Operator | $\mu_{\tilde{C}} = \min\{\mu_{\tilde{A}}, \mu_{\tilde{B}}\}$ |
| ODER Operator | $\mu_{\tilde{C}} = \max\{\mu_{\tilde{A}}, \mu_{\tilde{B}}\}$ |
| NICHT Operator | $\mu_{\tilde{C}} = 1 - \mu_{\tilde{A}}$ |

Aus der Tabelle ist ersichtlich, dass die UND Verknüpfung zweier Fuzzy-Sets als das Minimum der beiden unscharfen Mengen definiert werden kann. Auch der bool’sche UND Operator erzeugt im Prinzip das Minimum der beteiligten Variablen.

Die oben beschriebenen Operatoren können sehr recheneffizient implementiert werden. Sie widerspiegeln aber nicht immer die Bedeutung der Verknüpfungen in der menschlichen Sprache. Zum Beispiel:

- Das Wetter ist schön, wenn es warm UND sonnig ist.

Die Verknüpfung UND kann zwar in einer ersten Näherung als das Minimum aus den Zugehörigkeitsgraden von warm und sonnig interpretiert werden. Trifft zum Beispiel sonnig zum Grad 0.4 und warm zum Grad 0.35 zu, so ist das Wetter zum Grad 0.35 schön. Ist es hingegen zum Grad 0.8 sonnig und trotzdem nur zum Grad 0.3 warm, so wäre das Wetter nur zum Grad 0.3 als schönes Wetter zu bezeichnen. Als Mensch empfindet man aber das Wetter bestimmt schöner als im ersten Fall. Der UND Operator darf somit nicht immer stur als Minimum definiert werden, sondern ist differenzierter zu interpretieren. Trifft nämlich die eine Variable sehr viel stärker zu als die andere, so kompensiert sie zum Teil den niedrigen Zugehörigkeitsgrad der anderen. Um diesem Umstand Rechnung zu tragen, wurden die sogenannten kompensatorischen Operatoren eingeführt. Ein Vertreter solcher Operatoren ist der Gamma Operator, welcher wie folgt definiert ist:

$$\mu_C = (\mu_A \cdot \mu_B)^{1-\gamma} \cdot (1-(1-\mu_A)\cdot(1-\mu_B))^{\gamma}$$

Es muss kaum erwähnt werden, dass dieser Operator sehr rechenintensiv ist und deshalb in vielen industriellen Anwendungen nicht verwendet wird. Werden hingegen Fuzzy-Expertensysteme aufgebaut, welche ein differenziertes und dem menschlichen Sprachgebrauch angepasstes Bewerten unterschiedlicher Aussagen vornehmen muss, so ist der Gamma Operator dazu am besten geeignet. Mit dem Wert von $\gamma$ wird ein fliessender Übergang zwischen einem logischen UND (keine Kompensation) und einem logischen ODER (völlige Kompensation) geschaffen, falls diese nicht mit den Min- und Max-Funktionen, sondern wie in der nachfolgenden Tabelle definiert werden.

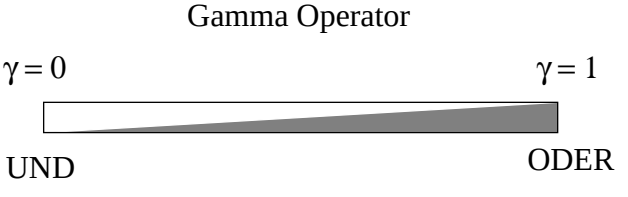
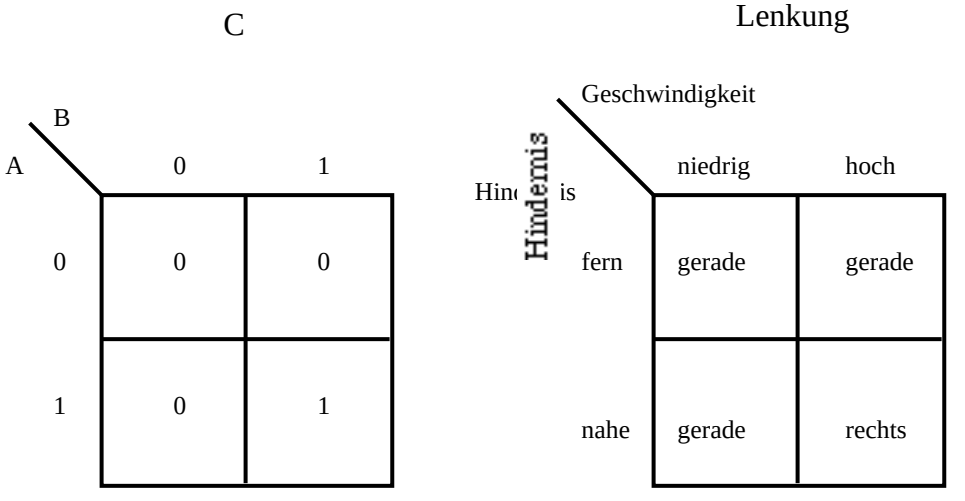



**Tabelle 4:**

| Operator | Fuzzy Zugehörigkeitsfunktion | Gamma $\gamma$ |
|---|---|---|
| UND Operator | $\mu_{\tilde{C}} = \mu_{\tilde{A}} \cdot \mu_{\tilde{B}}$ | $\gamma = 0$ |

**Tabelle 4:**

| | |
|---|---|
| ODER Operator | $\mu_{\tilde{C}} = \mu_{\tilde{A}} + \mu_{\tilde{B}} - \mu_{\tilde{A}} \cdot \mu_{\tilde{B}} \quad \gamma = 1$ |




Figur 64: Gegenüberstellung von Wahrheitstabellen in Binär- und Fuzzy-Logik.

## 7.7 Fuzzy-Logik Theorie

Anwendung findet die Fuzzy-Set Theorie in der Fuzzy-Logik. Die Fuzzy-Logik ist eine Erweiterung der Mehrwerte-Logik, in welcher die Wahrheitswerte durch linguistische Variablen ersetzt werden.

Verknüpfungen werden in der klassischen Binär- oder Mehrwertelogik entweder in Form von Wahrheitstabellen oder in Form von Gleichungen dargestellt. In der Fuzzy-Logik können ähnliche Darstellungsformen verwendet werden. Es existieren aber unendlich viele Wahrheitswerte in der Fuzzy-Logik, was eine tabellarische Darstellungsform im Prinzip verunmöglicht. Trotzdem gibt es aber die Möglichkeit, Fuzzy-Verknüpfungen in einer Fuzzy-Wahrheitstabelle zu erfassen, indem nicht die Wahrheitswerte selbst, sondern die einzelnen Fuzzy-Sets (Terme) tabelliert werden. In Figur 64 sind zwei Beispiele von Wahrheitstabellen der klassischen Binärlogik und der Fuzzy-Logik einander gegenübergestellt. In der Fuzzy-Logik sind nicht wie in der Binärlogik die Wahrheitswerte “0” und “1”, sondern die Terme der linguistischen Variablen tabellarisch erfasst.

Die zur klassischen Binärlogik äquivalente Darstellungsform der Gleichungen in der Fuzzy-Logik sind die sogenannten Fuzzy-Regeln:

- Binärlogik Gleichung: $A \cap B = C$
- Fuzzy-Logik Regel: WENN Geschwindigkeit hoch UND Hindernis nahe,

  DANN Fahrzeug rechts abbiegen.

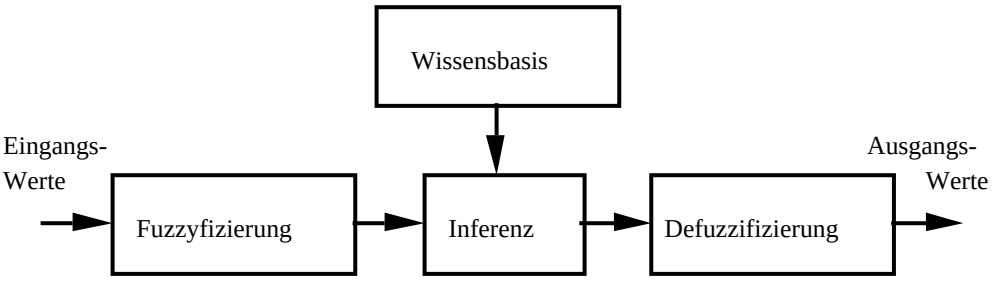
Vergleicht man diese beiden Aussagen, so stellt man fest, dass im einen Fall die bool'sche Variable "C", im anderen Fall der Term "rechts" der linguistischen Variablen "Fahrrichtung" das Resultat, respektiv die Schlussfolgerung ist. In der Fuzzy-Logik werden die klassischen Gleichungen der binären Logik zu Fuzzy-Gleichungen, welche als Aussagen oder Regeln betrachtet werden.

Beim Einsatz der Fuzzy-Logik wird das Räsonieren und die daraus getroffenen Schlussfolgerungen mit den Konstrukten aus der Fuzzy-Set Theorie geführt. Dazu wird Know-How oder Expertenwissen in Form von Regeln zusammengestellt und in der Anwendung ausgeführt. Das "Know-How" entspricht in der Binärlogik dem Entwurf von logischen Gleichungen (kombinatorisches Schaltschema). Als Denkanstoss kann sich der geneigte Leser überlegen, wie sich entsprechend der sequentiellen Binärlogik eine sequentielle Fuzzy-Logik definieren und einsetzen liesse (Fuzzy Zustandsmaschine).

## 7.8 Arbeitsweise eines Fuzzy-Logik Systems

Fuzzy-Logik Systeme lassen sich prinzipiell auf zwei Arten einsetzen. Im einen Fall arbeiten Fuzzy-Systeme als Regel-Systeme, wo sie elektronische, physikalische oder chemische Prozesse regeln. Im anderen Fall werden Fuzzy-Systeme als "Mini-Expertensysteme" verwendet, wo sie steuern, kontrollieren oder die gezogenen Schlussfolgerungen in Form von Ratschlägen oder Hinweisen zur Verfügung stellen.

Der Aufbau des Fuzzy-Systems selbst ist in beiden Fällen identisch. Ein Fuzzy-System lässt sich prinzipiell in vier Blöcke unterteilen, wie dies in Figur 65 dargestellt ist. Es spielt überhaupt keine Rolle, wie die einzelnen Blöcke implementiert werden. Man kann sich eine reine Software-Lösung vorstellen oder auch speziell für Fuzzy-Systeme zugeschnittene Hardwareeinheiten, welche in digitaler oder analoger Logik implementiert sein können. In den nachstehenden Abschnitten wird auf die einzelnen Blöcke eingegangen.

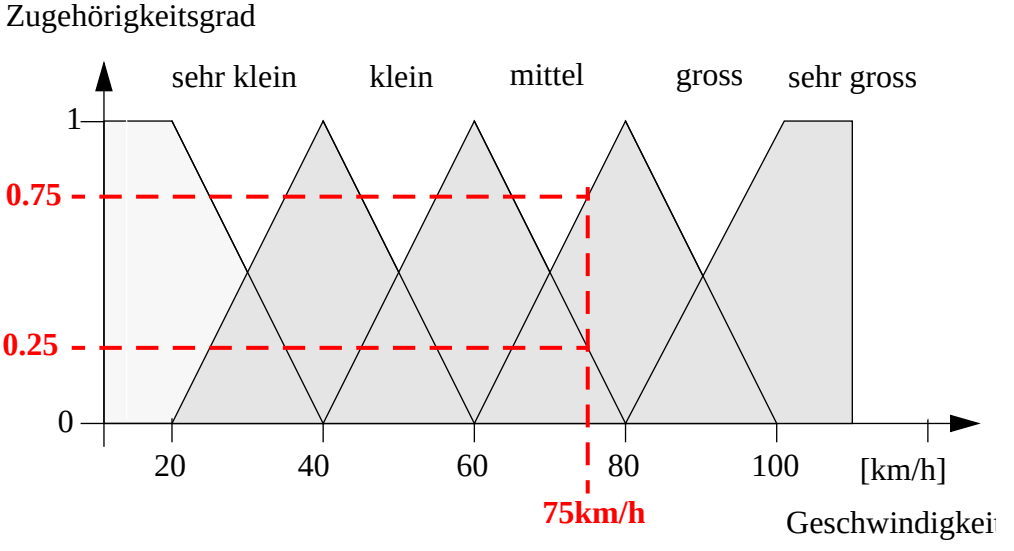



Figur 65: Aufbauprinzip eines Fuzzy-Systems.

### 7.8.1 Fuzzyfizierung

In der technischen Welt, speziell in den Ingenieurdisziplinen, arbeitet man immer mit quantifizierbaren Grössen, wie Zahlen und Werten. Solche Werte können durch Messungen über Sensoren oder durch interne Berechnungen anfallen. Es gilt nun diese “scharfen” Werte (‘crisp values’) auf die unscharfen Fuzzy-Werte abzubilden. Dieser Prozess wird Fuzzyfizierung genannt.

Am Beispiel der Fuzzy-Variablen Geschwindigkeit in Figur 66 wird der Wert 75km/h fuzzyfiziert. Wie aus der Abbildung ersichtlich ist, erhält man




Figur 66: Fuzzyfizierung eines ‘scharfen’ Geschwindigkeitswertes.

dabei zwei Wertepaare, da die Geschwindigkeit von 75km/h sowohl als mittel, wie auch als hoch gilt. Die Fuzzyfizierung ergibt (mittel, 0.25) und (hoch, 0.75). Das heisst, dass 75km/h zum Grad 0.25 als mittlere und zum Grad 0.75 als hohe Geschwindigkeit anzusehen ist. Diese Wertepaare dienen der Inferenzmaschine als Eingangswerte.

### 7.8.2 Inferenzmaschine und Wissensbasis

In der Inferenzmaschine werden die fuzzyfizierten Werte verarbeitet. Wie diese Werte verarbeitet und miteinander verknüpft werden, wird durch den Inhalt der Wissensbasis festgelegt. In der Wissensbasis ist das Expertenwissen in Form von Regeln festgehalten.

Um die Arbeitsweise einer Inferenzmaschine aufzuzeigen, dienen die beiden Eingangsvariablen Hindernisabstand *A* und Geschwindigkeit *V* eines Fahrzeuges mit den fuzzyfizierten Werten *A*={(nahe, 0.4)} und *V*={(mittel, 0.25), (hoch, 0.75)}. Angenommen die folgenden beiden Regeln seien in der Wissensbasis, so ergeben sie einen Beitrag zur Ausgangsvariable Lenkung:

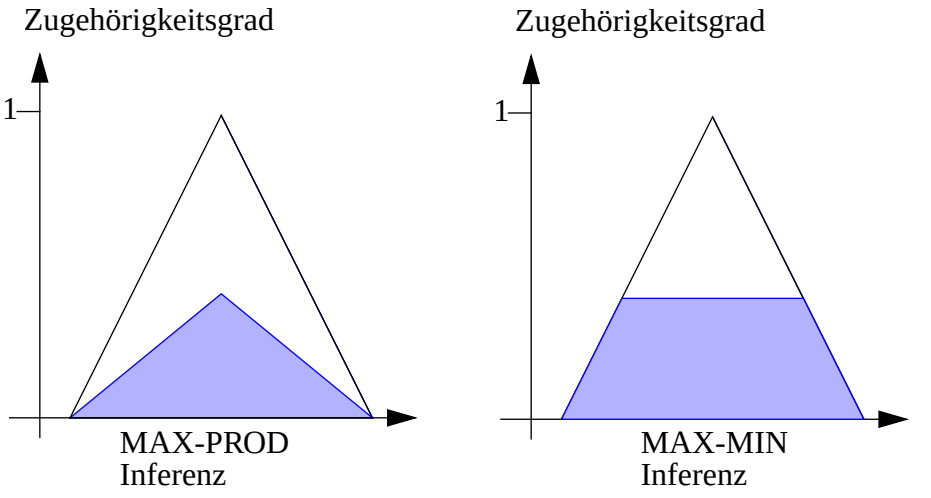
- FALLS Hindernis nahe UND Geschwindigkeit hoch, DANN Lenkung rechts

- FALLS Hindernis nahe UND Geschwindigkeit mittel, DANN Lenkung geradeaus

Die erste Regel ergibt 0.4 = MIN(0.4, 0.75) für die unscharfe Menge Lenkung rechts. Lenkung geradeaus wird gemäss der zweiten Regel zu 0.25 = MIN(0.4, 0.25). Die linguistische Ausgangsvariable Lenkung wird somit *L* = {(geradeaus, 0.25), (rechts, 0.4)}. Die Inferenzmaschine verknüpft also unscharfe Mengen unterschiedlicher Variablen miteinander und erzeugt wiederum eine Anzahl unscharfer Mengen einer oder mehrerer Ausgangsvariablen. Diese Ausgangsvariablen können nun je nach Inferenzmethode unterschiedlich interpretiert werden. Man unterscheidet zwischen der MAX-MIN und MAX-PROD Methode (siehe Figur 67). Vergleicht man die Ausgangswerte nach der Defuzzyfizierung, so kann man feststellen, dass die Wahl der Inferenzmethode praktisch keinen Einfluss auf die Resultate hat.

### 7.8.3 Defuzzyfizierung

Die Fuzzy-Wertepaare am Ausgang der Inferenzmaschine müssen in reelle Werte umgewandelt werden. Diese Umwandlung wird als Defuzzyfizierung bezeichnet. Wie aus dem obigen Beispiel ersichtlich ist, können verschiedene Regeln zu unterschiedlichen Schlussfolgerungen führen. Die eine Regel will dass Fahrzeug geradeaus lenken, die andere will das Fahrzeug nach rechts abbiegen lassen. Die Defuzzyfizierung beinhaltet somit ein Abwägen ver-



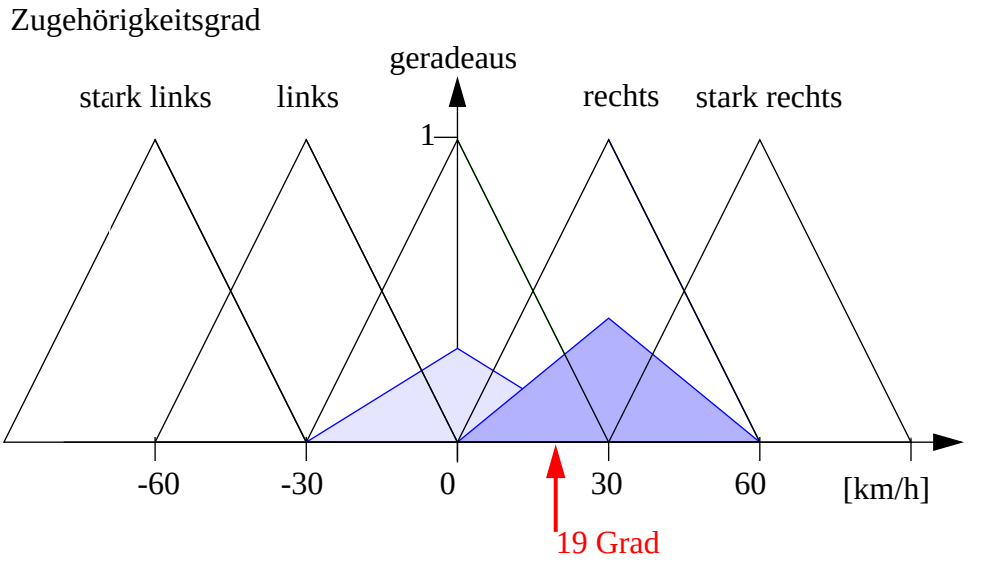

Figur 67: Gegenüberstellung der MAX-PROD und der MAX-MIN Inferenzmethoden.

schiedener Regeln oder eben ein Räsonieren. Es existieren verschiedene mathematische Verfahren, wie diese Defuzzyfizierung vorgenommen werden kann, die drei am häufigsten angetroffenen Verfahren sind:

- Schwerpunktsmethode (rechenintensiv)
- Höhenmethode (schnell, dafür nur grobe Abschätzung)
- Gewichtete Höhenmethode (Kompromiss)

Am Beispiel der Schwerpunktsmethode sei die Defuzzyfizierung der obigen Resultate für die Fahrzeuglenkung vorgerechnet. Wie es der Name schon vermuten lässt, wird bei der Schwerpunktsmethode der Flächenschwerpunkt der unscharfen Mengen bestimmt und daraus ein reeller Wert für die Ausgangsvariable Lenkung bestimmt. Aus der Definition der linguistischen Variablen Lenkung in Figur 68 ist ersichtlich, dass nach der Schwerpunktsmethode eine leichte Auslenkung nach rechts von 19 Grad nach der Defuzzyfizierung resultiert. Die Formel für die Defuzzyfizierung nach der Schwerpunktsmethode lautet wie folgt, wobei $F_i$ die Flächen der einzelnen unscharfen Mengen und $x_i$ die x-Koordinaten der Schwerpunkte bezeichnen:

$$x = \frac{\sum_i F_i x_i}{\sum_i F_i}$$




Figur 68: Defuzzyfizierung nach Schwerpunkts-Methode.

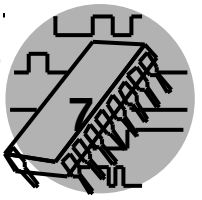
Die Berechnung nach der Schwerpunktsmethode ist rechenintensiv, dafür ist aber das Resultat sehr sensitiv auf jede Änderung der Zugehörigkeitsfunktion der einzelnen Fuzzy-Mengen.

## 7.9 Anwendungsgebiete

Die Fuzzy-Set Theorie hat schon in zahlreichen Disziplinen Einzug gehalten, wie den Computer-Wissenschaften, der künstlichen Intelligenz, den Expertensystemen, der Mustererkennung, der Robotik, der Steuerungs- und Regelungstechnik usw.

Vor etwa 4 Jahren wurde in der japanischen Stadt Sendai eine Untergrundbahn mit einer Fuzzy-Steuerung ausgerüstet. Anfahren, Beschleunigen und Bremsen werden unabhängig von der Passagierzahl so sanft gesteuert, dass sich die Fahrgäste auf den Stehplätzen inzwischen gar nicht mehr festhalten. Der Mann im Führerstand, dessen vergleichsweise grobes Fingerspitzengefühl nicht mehr gefragt ist, wird fürs Nichtstun bezahlt.

Wie winzige Expertensysteme verhalten sich neuartige Prozessorchips, die in Japan jetzt in vielen Haushaltgeräten, Consumer-Produkten und zunehmend auch in Industriesteuerungen eingebaut werden. Waschmaschinen wählen je nach Verschmutzungsgrad der Wäsche die effektivsten



Waschprogramme, bei Videorecordern werden falsch belichtete Aufnahmen verhindert, Fokus und Blende narrensicher gesteuert, Gegenlicht, Schatten und sogar Zitterbewegungen kompensiert, ein Taschencomputer erkennt 3000 verschiedene handgeschriebene japanische Schriftzeichen, bei Fahrzeugen werden die Ruckbewegungen beim Gangwechsel eliminiert, die Temperaturführung bei chemischen Prozessen geregelt, die Wartezeiten bei Liften verkleinert, etc.

Diese Aufzählung beschreibt nur ein paar wenige der bereits existierenden Anwendungen. Der Phantasie sind im Prinzip keine Grenzen gesetzt. Bekannte Produkte aus der Fuzzy-Logik lassen sich in drei Anwendungsgebiete einteilen:

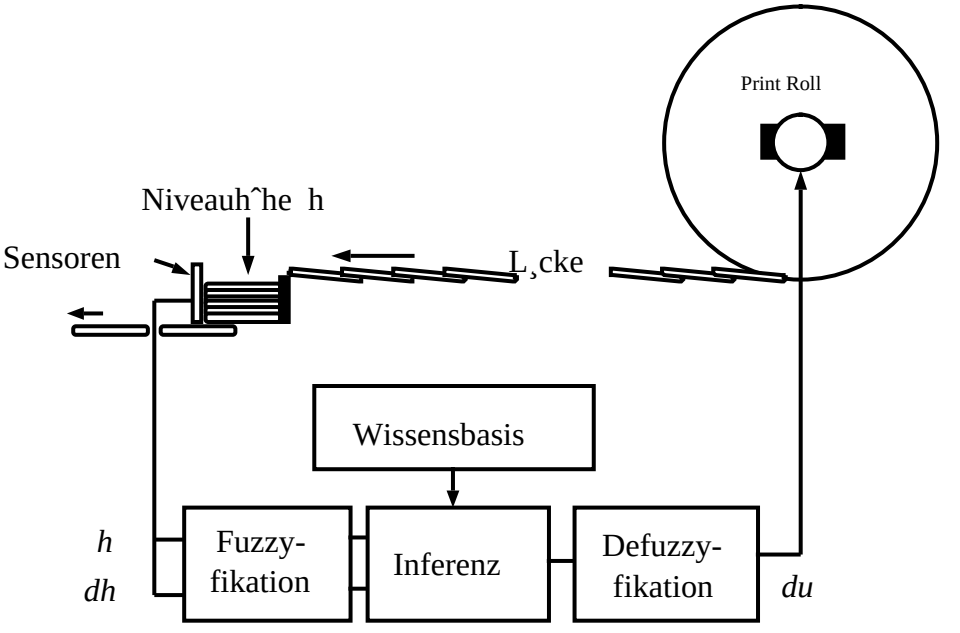
- **Fuzzy Expertensysteme:** Medizinische Diagnosesysteme, Expertensysteme für Syntaxkorrekturen, Wissensbasierte Sprachsynthese, Wissensbasierte Steuerung flexibler Fertigungsanlagen, Expertensysteme zur Bestimmung der Kreditwürdigkeit, etc.
- **Fuzzy Datenanalyse:** Schrifterkennung, optische Muster- und Objekterkennung, Datenkompression, etc
- **Fuzzy Control:** Waschmaschinen, Kamerarecorder, Fotoapparate, Lift- und Zugsteuerungen, ABS und Anti-Schleuder Systeme, Anwendungen im geschlossenen Regelkreis, Echtzeitanwendungen, etc.

## 7.10 Ein Fuzzy-Logik Produkt aus der Schweizer Maschinenindustrie

An einem ersten konkreten Beispiel, der Print-Roll Maschine, wurde eine Fallstudie über die Industrietauglichkeit und die Einsatzmöglichkeiten der Fuzzy-Logik durchgeführt. Der unerwartet rasche Erfolg führte dazu, dass aus dem ersten Anwendungsbeispiel direkt ein Serie-Produkt entstand.

### 7.10.1 Funktionsprinzip des Print-Roll Systems

Das Print-Roll System ist eine zentrale Einheit in der Müller Martini Automatisierungstechnik der Zeitschriften- und auch Zeitungsproduktion. Es dient als Zwischenspeicher für gedruckte Zeitschriften-Bogen. Die verschiedenen Bogen einer Zeitschrift werden von der Druckmaschine als Schuppenstrom direkt auf Rollen aufgewickelt. Diese Rollen werden bis zur Zeitschriftenproduktion in grossen Lagern aufbewahrt. Auf den Zeitpunkt der Fertigstellung hin, d.h. Zusammentragen und Heften der einzelnen Bogen zu einer Zeitschrift, werden die Rollen mit den gedruckten Bogen mit Robotern aus den Lagern geholt und automatisch an die jeweiligen Anleger der Sammelheft-




Figur 69: Prinzipschaltung der Niveaukontrolle mit einfachem Fuzzy-Regler.

Maschinen angedockt. Die als Schuppenstrom aufgewickelten Bogen werden nun via Anleger von den Rollen abgewickelt und gelangen über eine Taktsteuerung auf die Sammelkette, wo sie im Falz übereinandergelegt, geheftet und dann geschnitten werden. Für die Anpassung der Abrollgeschwindigkeit der einzelnen Rollen an den zentralen Produktionstakt (Sammelkette), ist an jeder Abrollstation eine dezentrale Regelung vorhanden. Um eine exakte Synchronisation auf den zentralen Produktionstakt zu erreichen und um Unregelmässigkeiten der zuführenden Schuppenströme auszugleichen, sind an den Anlegern der Sammelheft-Maschine kleine Zwischenspeicher (Bogen-Stapel) vorhanden. Es gilt nun, die Höhe dieses Stapels an jedem Anleger individuell auf einem möglichst konstanten Niveau zu regeln, um eine zuverlässige und fehlerfreie Produktion zu erhalten. Diese Regelungen der Abrollgeschwindigkeiten wurden bis heute mit klassischen PI-Reglern ausgeführt.

### 7.10.2 Ein Lösungsansatz mittels Fuzzy-Logik

Aus zwei Gründen wollte man ein bestehendes System mit einem neuen Ansatz überarbeiten. Ein Grund war der zu häufige Ausfall der Antriebsmotoren der Rollen vor dem Ablauf der Garantiezeit. Messwertaufnehmer zeig-

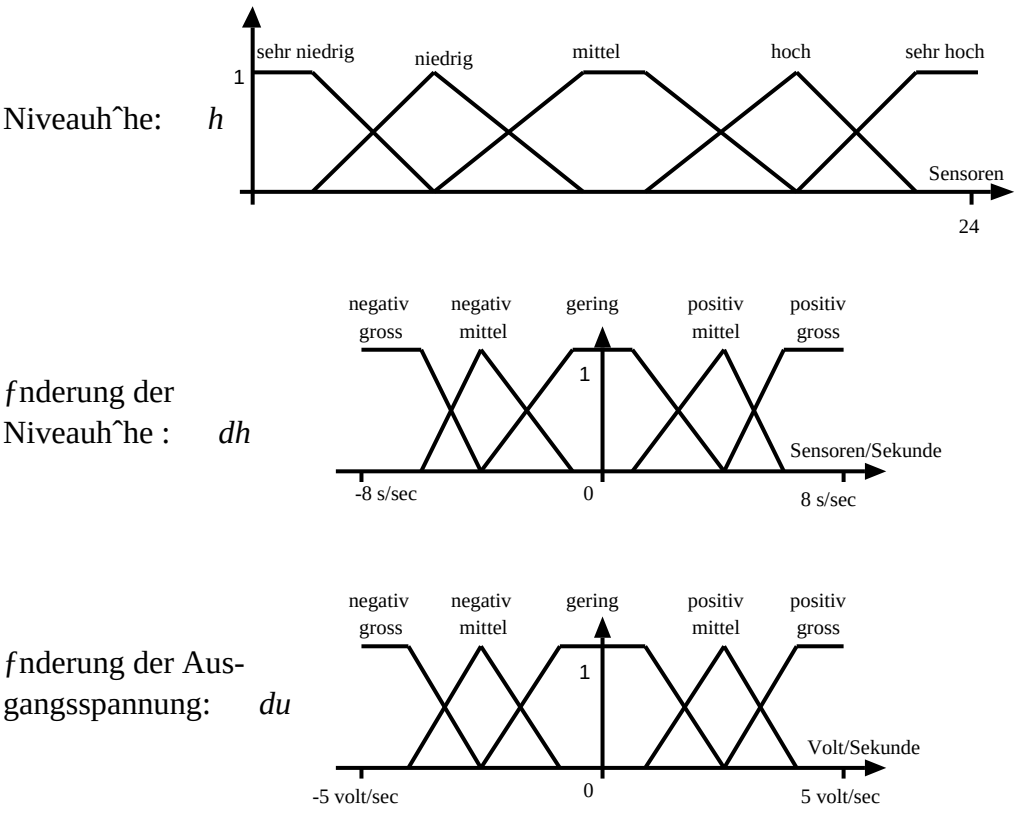
ten in der rauhen Betriebsumgebung grosse Schwankungen auf, was zu einem dauernden und unruhigen Nachregeln führte und so die Lebensdauer des Antriebsmotors reduzierte. Weiter wollte man aus wirtschaftlichen Überlegungen die Produktionsgeschwindigkeit erhöhen. Dabei handelte man sich natürlich Probleme aus den verschiedensten technischen Bereichen ein. Unter anderem zeigte sich, dass die vorhandene Niveaukontrolle den neu gesteckten Zielen nicht mehr gewachsen war. Es musste somit nach neuen Lösungsansätzen gesucht werden. Mit der Anwendung der Fuzzy-Set Theorie erhoffte man sich mehr Freiheitsgrade beim Entwurf des Reglers. Erste Tests in der Praxis haben gezeigt, dass die Anlage heute ruhiger, zuverlässiger und erst noch schneller produziert.

### 7.10.3 Implementation der Fuzzy-Logik

Zwei verschiedene Fuzzy-Regler wurden für die Niveau-Kontrolle entworfen. In einem ersten Ansatz wurde ein reiner Fuzzy-Regler implementiert, welcher nachfolgend beschrieben ist (Figur 69). In einem zweiten Entwurf entschied man sich für eine Hybrid-Lösung, in welchem einem konventionellen digitalen PI-Regler eine übergeordnete Fuzzy-Steuerung die P- und I-Parameter den äusseren Bedingungen entsprechend laufend verändert (siehe Figur 72 auf Seite 87).

Die Eingangs- und Ausgangsgrössen der Regelstrecke sind für den Fuzzy-Regler in linguistische Variablen überzuführen. Ein Reihe von 24 Sensoren misst die Niveauhöhe *h*. Die Ableitung, d.h. die Änderung der Niveauhöhe *dh* kann sehr einfach aus den vorhergehenden Messwerten gebildet werden. Der Fuzzy-Regler liefert am Ausgang die Änderung der Ausgangsspannung *du*, welche als Stellgrösse für die Ansteuerung der Rolle verwendet wird (Figur 69). Die konkrete Abbildung dieser drei linguistischen Variablen ist in Figur 70 dargestellt:

Der Wissensbasis-Kasten in Figur 69 beinhaltet die sogenannte Inferenzstrategie, welche durch einen Satz von Regeln festgelegt ist. Im Beispiel der Niveau-Kontrolle können beispielsweise die folgenden Regeln aus dieser Wissensbasis zur Anwendung gelangen:

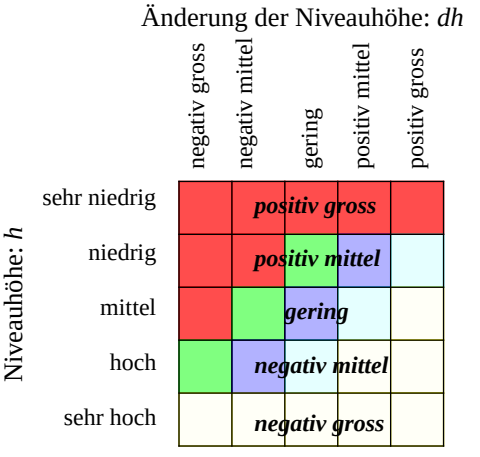



Figur 70: Definition der linguistischen Variablen.

- WENN die Niveauhöhe sehr tief ist, DANN soll Änderung der Ausgangsspannung positiv-gross sein.
- WENN die Niveauhöhe mittel ist UND die Änderung der Niveauhöhe negativ mittel ist, DANN soll die Änderung der Ausgangsspannung positiv mittel sein.
- WENN Niveauhöhe tief UND die Änderung der Niveauhöhe positiv mittel ist, DANN soll die Änderung der Ausgangsspannung gering sein.

Ein vollständiger Satz von Regeln, wie sie in der Wissensbasis der Niveaukontrolle vorhanden ist, ist in Form einer zwei-dimensionalen Wahrheitstabelle in Figur 71 dargestellt.

Mit den eingangs definierten linguistischen Variablen und den oben aufgeführten Regeln wurde ein gleichmässiges Abrollen der Zeitschriften- und Zei-

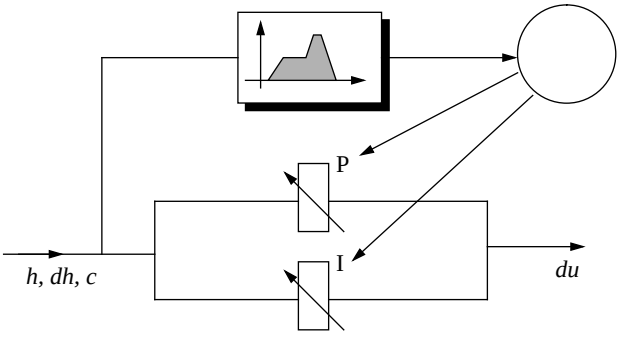



Figur 71: Fuzzy-Regeln zusammengefasst in Fuzzy-Wahrheitstabelle. Die Symmetrie in den Regeln ist deutlich zu erkennen.

tungsbogen auch bei stark erhöhter Produktionsgeschwindigkeit erreicht. Ein erster Testlauf lief erfolgreich, obwohl später festgestellt wurde, dass mehrere der 24 Höhenmessensoren nicht korrekt funktionierten. Dieser Versuch untermauerte die Fehlertoleranz und Robustheit des Fuzzy-Reglers zumindest für dieses Beispiel.

Um weitere Erfahrungen in der Anwendung der Fuzzy-Set Theorie zu erlangen, wurde zusätzlich ein Hybrid-Regler entwickelt. Dabei wurde einem klassischen PI-Regler eine Fuzzy-Steuerung überlagert, welche die P- und I-Parameter des klassischen Reglers den sich dauernd ändernden Verhältnissen anpasst. Das Blockschaltbild des Hybrid-Reglers ist in Figur 72 dargestellt.

Mit dem hybriden PI/Fuzzy-Regler in Figur 72 wurde das Systemverhalten weiter verbessert. Simulationsresultate zeigen den Vergleich zwischen einem klassischen PI-Regler und dem implementierten Hybrid-Regler (Figur 73). Der Hybrid-Regler zeigt im gesamten Regelbereich deutlich verminderte Über- und Unterschwingungen, was auf die adaptiven Eigenschaften zurückzuführen ist. Weiter hat sich gezeigt, dass die Schwankungen des Messwertaufnehmers sich nicht mehr durch ein nervöses Nachregeln des Antriebsmotores bemerkbar machen und auch nicht jede kleine Abweichung zum Sollwert sofort nachgeregelt wird.




Figur 72: Prinzip eines hybriden PI/Fuzzy-Reglers.

Figur 73: Vergleich zwischen klassischem PI- und hybridem PI/Fuzzy-Regler.

## 7.11 Einführungsstrategie der Fuzzy-Set Theorie

Sind in einem Unternehmen Anwendungsgebiete für die Fuzzy-Logik ersichtlich oder will man sich einfach den Möglichkeiten dieser neuen Technologie nicht verschliessen, so sollte sie nicht mit Fanfaren sondern mit Bedacht eingeführt werden. Mit der Anwendung der Fuzzy-Set Theorie betritt man in solchen Fällen Neuland, welches sich bei einem falschen Vorgehen leicht als Glatteis entpuppen kann. Mit einer bereits bewährten vier-Punkte Strategie können unliebsame Überraschungen vermieden werden:

- In einer ersten Phase ist ein guter und umfassender Kenntnisstand über das Gebiet der Fuzzy-Set Theorie zu gewinnen. Eine Einarbeitung in die Theorie ist zum heutigen Zeitpunkt am sinnvollsten durch entsprechende Literatur, eventuell in Ergänzung mit einschlägigen Kursen zu bewerkstelligen.
- Um sich erste Praxiserfahrungen anzueignen, sollte anhand bekannter klassischer Übungsbeispiele der Umgang mit den neu gewonnenen Freiheiten durch die Fuzzy-Set Theorie gelernt werden. Zwei Gründe rechtfertigten diesen zusätzlichen Aufwand. Einerseits förderte dieses Vorgehen das vertiefte Verständnis der Fuzzy-Set Theorie. Andererseits gibt es den Ingenieuren ein Gefühl dafür, wie gut sie mit dieser unkonventionellen Technologie umzugehen vermögen und wie komplexe Aufgaben sie bereits damit zu lösen wagen dürfen.
- In einer dritten Phase ist eine erste eigene Fuzzy-Logik Anwendung im Unternehmen anzupacken. Es ist darauf zu achten, dass diese erste Anwendung noch nicht allzu komplex ist, damit das vorprogrammierte positive Resultat die Motivation und den Ideenreichtum der beteiligten Ingenieure anstachelt.
- Eine Verbreitung der Fuzzy-Set Theorie und der gewonnenen Erfahrungen sind im Unternehmen in Form eines kurzen Intensivkursen den interessierten Entwicklungsingenieuren anzubieten. In einer Diskussionsrunde kann nach weiteren firmeninternen Anwendungen oder Anwendungspotentialen Ausschau gehalten werden.

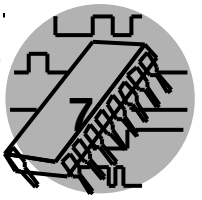
## 7.12 Zukunftsperspektiven

Der Begriff Fuzzy-Logik darf nicht als Gespenst der japanischen Industrierevolution gehandelt werden, sondern soll in der Ausbildung der Ingenieure,

neben den klassischen Methoden seinen Platz erhalten. In der Firma Müller Martini zum Beispiel wurden sämtliche Elektronik-Entwicklungsingenieure in Theorie und Anwendung der Fuzzy-Logik geschult. Dadurch haben sie die Möglichkeiten, aber auch Grenzen dieser neuen Technik erlernt und sind nun fähig zu entscheiden, wann und wo die Fuzzy-Logik am besten eingesetzt werden kann. Wichtig ist auch, dass in den Ingenieurschulen selbst die Grundausbildung entsprechend ergänzt wird, wie dies zum Beispiel an der Ingenieurschule Biel der Fall ist.

Literatur

[1] L. A. Zadeh, Fuzzy Sets, Informations and Control, 1965, S 338
[2] L. A. Zadeh, Fuzzy Sets and Applications, John Wiley & Sons, 1987
[3] H. J. Zimmermann, Fuzzy Sets Theory - and its Applications, Kluwer- Nijhoff Publishing, 1990
[4] C. v. Altrock, über den Daumen gepeilt, c't 1991, S 188-206



# Digitaltechnik Übungssammlung: Teil 1

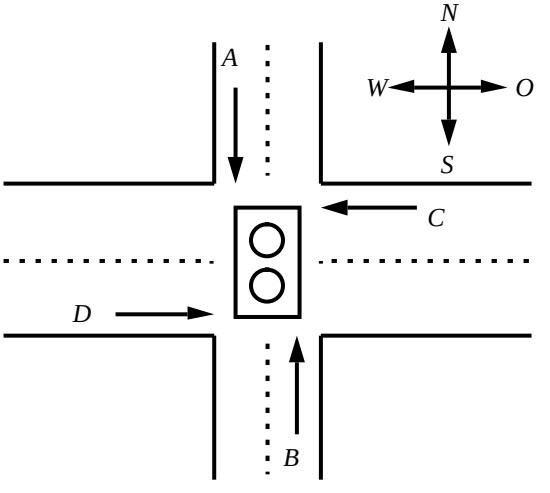
*M. Jacomet*
*Ingenieurschule Biel*

**Aufgabe 1:** In Figur 74 ist die Kreuzung von einer Haupt- und einer Nebenstrasse dargestellt. Sensoren vor dem Kreuzungspunkt stellen fest, ob ein Auto sich auf der Hauptstrasse *C* und *D* oder sich auf der Nebenstrasse *A* und *B* befindet. Die Ausgangswerte dieser Sensoren sind logisch “0”, wenn kein Auto vorhanden ist und “1”, wenn sie ein Auto detektieren. An der Kreuzung befindet sich eine Lichtsignalanlage, welche durch diese Sensoren entsprechend den folgenden Regeln gesteuert wird.

- Das Signal O-W ist grün, wenn sich gleichzeitig Autos auf den beiden Fahrspuren *C* und *D* befinden.
- Das Signal O-W ist grün, wenn sich Autos auf den Spuren *C* oder *D* befinden und auf *A* oder auf *B* aber nicht auf beiden gleichzeitig.
- Das Signal N-S ist grün, wenn es Autos auf den Spuren *A* und *B* und auch auf der Spur *C* oder *D* aber nicht auf beiden gleichzeitig hat.
- Das Signal N-S ist ebenfalls grün, wenn es Autos auf einer der beiden Spuren *A* oder *B* hat, aber keines auf *C* oder *D*.
- Das Signal O-W ist grün, wenn überhaupt keine Autos vorhanden sind.

Die Schaltung muss die zwei Ausgänge O-W und N-S aufweisen, welche den logischen Werte “0” für ein grünes und “1” für ein rotes Signal annehmen soll.

a) Falls Situationen auftreten können, bei welchen die Stellung des Lichtsignales nicht definiert ist, so sind die ergänzenden Regeln so zu finden, dass sich die Anlage logisch verhält. Geben Sie fehlende Regeln explizit an. Stellen Sie die Wahrheitstabelle auf

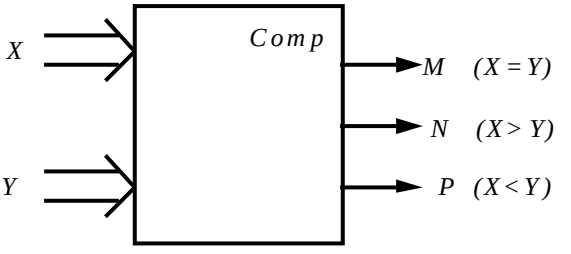



Figur 74:

b) Die Ausgangssignale der vier Sensoren *A*, *B*, *C* und *D* können als logische Eingänge für eine Steuerung verwendet werden. Entwerfen Sie eine logische Schaltung, welche als Lichtsignal-Steueranlage verwendet werden kann. Vereinfachen Sie die Schaltung so weit wie möglich und illustrieren Sie jeden einzelnen Entwicklungsschritt.

**Aufgabe 2:** In Figur 75 ist eine Vergleicherschaltung (**Komparator**) dargestellt, welche zwei binäre Werte von je 2 bits **x** ($x_1$, $x_0$) und **y** ($y_1$, $y_0$) miteinander vergleicht. Die Schaltung stellt fest, ob die beiden Binärzahlen gleich sind oder ob die eine oder die andere grösser ist. Der Komparator weist drei Ausgänge auf, welche wie folgt definiert sind:

- $M$ = 1 falls beide Werte gleich gross sind.
- $N$ = 1 falls **x** grösser ist als **y**.
- $P$ = 1 falls **y** grösser ist als **x**.




Figur 75:

a) Entwerfen Sie die Karnaugh-Diagramme des Komparators für alle drei Ausgänge.

b) Entwickeln und minimisieren Sie die logische Schaltung für die Ausgänge *N* und *P*. Zeichne Sie das entsprechende Schema der Schaltung.

c) Leiten Sie die boolesche Gleichung für den Ausgang *M* her. Verwenden Sie die Regeln der Booleschen Algebra, um eine Struktur zu finden, welche direkt durch Exclusiv ODER realisiert werden kann.

**Aufgabe 3:** Zeigen Sie die Gültigkeit des folgenden Booleschen Ausdruckes auf:

$$A + (B \cdot C \cdot D) = (A + B) \cdot (A + C) \cdot (A + D)$$

**Aufgabe 4:** Vereinfachen Sie die Funktionen, welche durch die Karnaugh-Diagramme in Figur 76, Figur 77 und Figur 78 dargestellt ist und geben Sie die entsprechende Booleschen Gleichungen an.

Figur 76:

| DE \ ABC | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 01 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

Figur 77:

| *AB* \ *CDE* | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 01 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 11 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 10 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

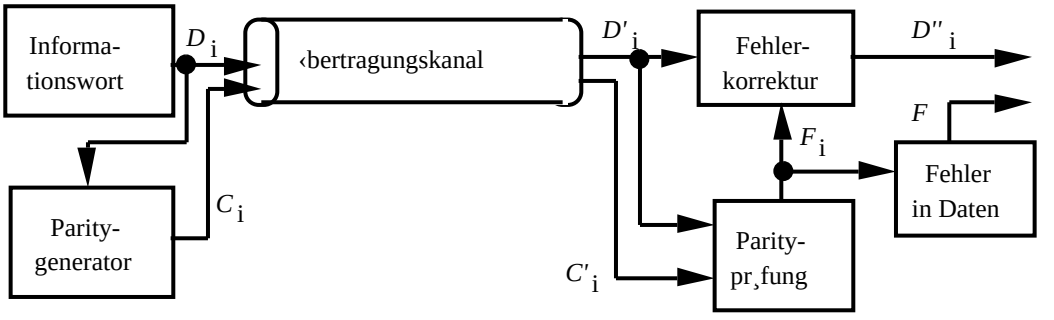
Figur 78:

| DE \ ABC | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 1 | 1 | X | 0 | 0 | 0 | X | 1 |
| 01 | 0 | 1 | X | 1 | 1 | X | 0 | 0 |
| 11 | 0 | 1 | 1 | 1 | 1 | X | X | 0 |
| 10 | 1 | X | X | 0 | 0 | X | X | 1 |

Figur 79:

| *AB* \ *CDE* | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 01 | X | 0 | 0 | 1 | X | 1 | 1 | 0 |
| 11 | 0 | 0 | 0 | 1 | X | 1 | 1 | 0 |
| 10 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**Aufgabe 5:** Für die Übertragung von einem 4 Bit Wort $D_i$ wird ein Hamming-Code verwendet, welcher zur Fehlererkennung und Fehlerkorrektur 3 zusätzliche Paritätsbits $C_i$ verwendet (siehe Figur 80). Die drei Paritätsbits komplettieren die geprüften Bits auf gerade Parität. Das Paritätsbit $C_1$ überprüft die Informationsbits $D_7$, $D_5$, $D_3$, das Paritybit $C_2$ überprüft $D_7$, $D_6$, $D_3$ und $C_4$ überprüft $D_7$, $D_6$, $D_5$.




Figur80:

a) Stellen Sie in einer Tabelle alle gültigen 7 Bit breiten Codeworte zusammen.

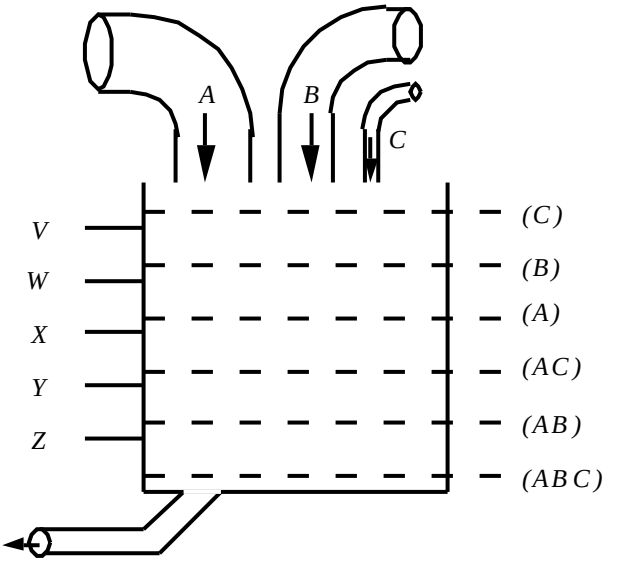
b) Die Signale $F_4$, $F_2$, $F_1$ werden logisch “1”, wenn sie einen Paritätsfehler in den Paritätsbits $C'_4$, $C'_2$, $C'_1$ erkannt haben. Stellen Sie die Karnaugh Diagramme für alle drei $F_i$ Signale (siehe Figur 80 Block Parityprüfung) auf.

c) Stellen Sie die boolesche Gleichung für $F_1$ auf und versuchen Sie den Ausdruck so weit wie möglich auf Exclusiv-ODER Funktionen zurückzuführen. Zeichnen Sie das dazugehörige Logikschema.

d) Die Logikschema für die anderen beiden Paritätsfehlersignale $F_4$, $F_2$ sind anzugeben.

e) Entwerfen Sie eine logische Schaltung mit dem Ausgang *F*, welcher logisch “1” wird, wenn ein Informationsbit fehlerhaft übertragen wurde und entsprechend eine Korrektur des Informationswortes nötig ist (Wahrheitstabelle, Karnaugh-Diagramm, Schema).

**Aufgabe 6:** Ein Wasserreservoir wird aus drei Zuleitungen gespeist, deren Ergiebigkeit von *A* nach *C* abnimmt. Die Sensoren *V, W, X, Y* und *Z* geben den Wasserstand an. Die Sensoren nehmen den logischen Wert “1” an, wenn sie sich im Wasser befinden. Die Zuleitungen sollen gemäss der Figur 81 geöffnet werden (logisch “1” bedeutet offen). Beim höchsten Stand deckt die Zuleitung *C* gerade die Verluste durch Verdunstung ab. Bestimmen Sie über das minimisierte Karnaugh-Diagramm die Steuerfunktion für die Zuleitungen *A, B, C*.

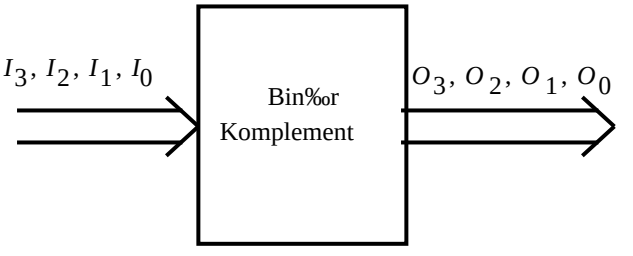



Figur81:

**Aufgabe 7:** Entwerfen Sie einen Binär-Komplementierer für eine 4-stellige Binärzahl mit der Gewichtung 8-4-2-1, der zu jeder Binärzahl das Komplement zu 16 liefert. Die Eingangsvariablen seien mit $I_3$, $I_2$, $I_1$, $I_0$ und die Ausgangsvariablen mit $O_3$, $O_2$, $O_1$, $O_0$ bezeichnet ($I_3$ und $O_3$ sind die MSB’s).

a) Stellen Sie die Wahrheitstabelle für den Binär-Komplementierer auf.

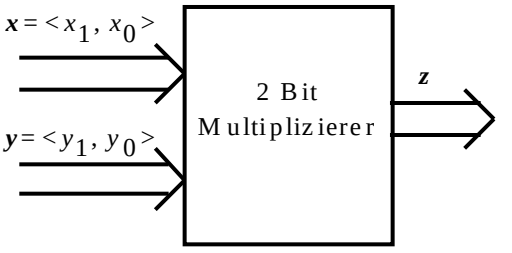
b) Ausgehend von der Wahrheitstabelle sind Karnaugh-Diagramme aufzustellen, Vereinfachungen vorzunehmen und Logikschema zu entwerfen.




Figur82:

**Aufgabe 8:** Es ist eine Multiplizierschaltung zu entwerfen (siehe Figur 83), welche zwei binäre Zahlen **x** ($x_1$, $x_0$) und **y** ($y_1$, $y_0$) von je zwei Bits Breite multiplizieren kann. Bestim-

men Sie die vereinfachten booleschen Ausdrücke für die Ausgänge **z** des Multiplizierers.

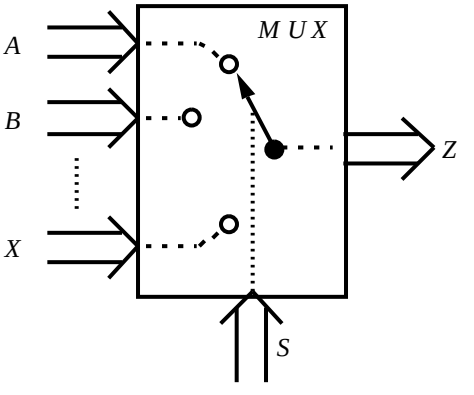



Figur 83:

**Aufgabe 9:** Dividieren Sie schriftlich im binären System die beiden Zahlen $148_{dez}$ und $7_{dez}$ durcheinander. Geben Sie das exakte Endresultat an:

$148_{dez}:7_{dez}=?$

**Problèm 10:** Ein **Multiplexer** hat mehrere Eingangsports, aber nur einen Ausgangsport. Die sogenannten SELECT (S) Leitungen wählen nun denjenigen Eingangsport aus, dessen Eingangswerte am Ausgangsport erscheinen sollen. In Figur 84 ist das Symbol eines allgemeinen Multiplexers dargestellt. Ein Multiplexer verhält sich wie ein Umschalter, welcher durch einen Code an den SELECT Eingängen den Ausgangsport mit einem der Eingangsports verbindet.




Figur 84:

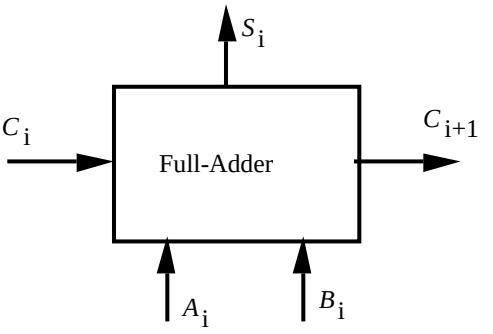
Es ist ein Multiplexer mit zwei Eingangsports à je zwei Bits (Port *A*: $A_1$, $A_0$; Port *B*: $B_1$, $B_0$), dem Ausgangsport *Z* ($Z_1$, $Z_0$) und dem Signal *S* (SELECT Eingang) zu entwerfen. Wenn *S* logisch "1" wird, so wird der Port *B* auf den Ausgang geschalten.

a) Entwerfen Sie die Wahrheitstabelle.

b) Entwickeln Sie die minimisierten Booleschen Gleichungen für alle Ausgänge mit Hilfe der Karnaugh-Tabelle und zeichnen Sie das Schema.

**Aufgabe 11:** Um zwei binäre Variablen *A* und *B* zu addieren, können sogenannte Volladdierer eingesetzt werden, welche in Serie geschaltet werden. Dabei wird pro Bit eine solche Volladdiererstufe eingesetzt, welche jeweils ein Bit des Resultates *S* berechnen. Die Überträge werden durch die sogenannte Carry Leitungen $C_i$ einer Stufe mit einem höherwertigen Bit zugeführt.

Figur85:




a) Stellen Sie die Wahrheitstabelle für den Volladdierer dar.

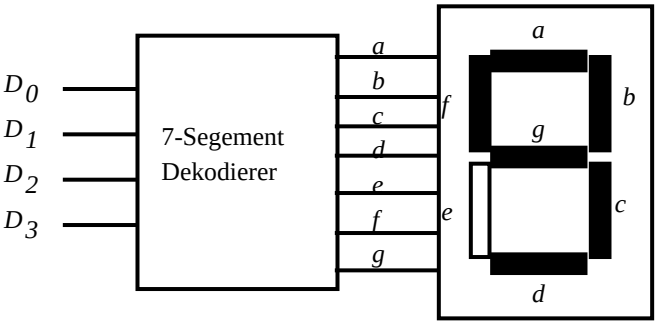
b) Entwickeln Sie ein Schaltung für den Volladdierer. (Karnaugh-Diagramm, Minimisierung, Boolesche Gleichung, Schaltung).

**Aufgabe 12:** Gesucht ist eine logische Schaltung, welche beim Auftreten einer Primzahl (1, 2, 3, 5, 7, ...) zwischen 0 und 31 ein logisches “1”-Signal erzeugt. Eingangsvariable **x** ($x_4$, $x_3$, $x_2$, $x_1$, $x_0$). (Wahrheitstabelle, Karnaugh-Diagramm, Boolesche Gleichung, Schema).

**Aufgabe 13:** Vereinfachen Sie den folgenden Booleschen Ausdruck nach dem Prinzip von Quine and McCluskey.

Y=A’BCD’E’F’ + A’BCD’E’F + A’BCD’EF’ + A’BCD’EF + A’BCDE’F’ + A’BCDE’F + A’BCDEF’ + A’BCDEF + ABC’DEF + ABC’DE’F + ABC’D’EF + ABC’D’E’F + AB’C’DEF +AB’C’DE’F + AB’C’D’EF + AB’C’D’E’F + ABCDE’F + ABCDE’F’ + ABC’DE’F’

**Aufgabe 14:** Es ist eine Dekodierschaltung für eine 7-Segment Anzeige zu entwerfen (siehe Figur 86). Die Schaltung besitze einen 4-bit BCD Eingang (binär decodierten Dezimaleingang) und die Augänge *a* bis *g* für die Ansteuerung der 7 Segmente. Zu dekodieren sind die Ziffern *0* bis *9* und die Buchstaben *A, b, C, d, E* und *F.*




Figur 86:

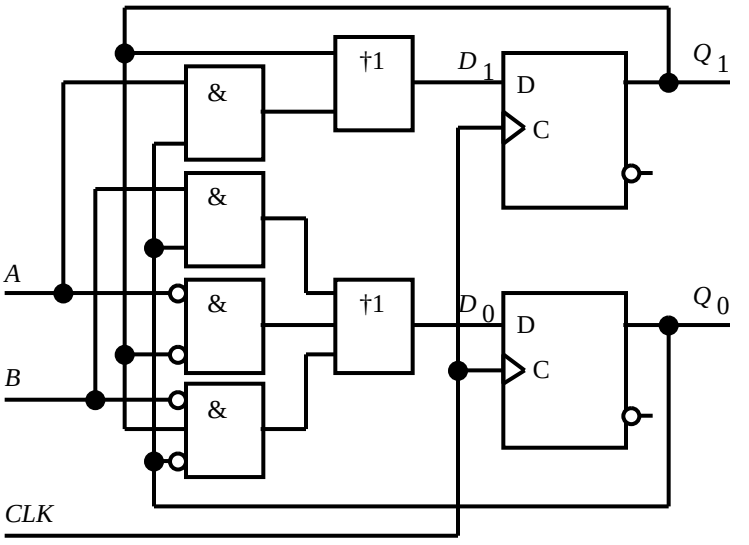
a) Stellen Sie die Wahrheitstabelle für die Dekodierschaltung auf.

b) Leiten Sie die minimisierte boolesche Gleichung und die logische Schaltung für die verschiedenen Segment her.

# Digitaltechnik Übungssammlung: Teil 2

*M. Jacomet*
*Ingenieurschule Biel*

**Aufgabe 1:** In Figur 87 ist die Schaltung eines sehr einfachen Sequenzers dargestellt. Die Schaltung ist zu studieren und zu analysieren.



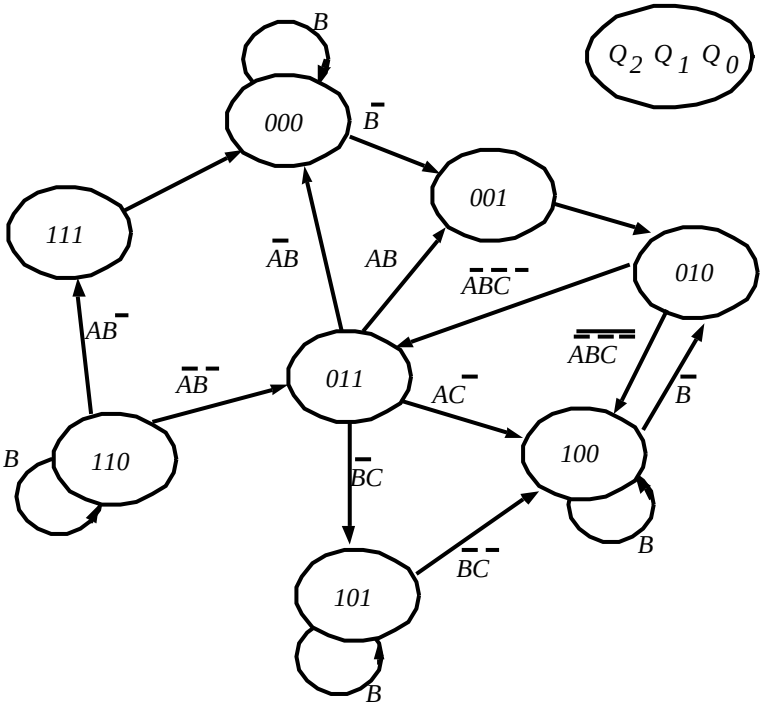

Figur 87:

a) Bestimmen Sie die booleschen Gleichungen für die Eingänge der Flip-Flop's ($D_1$, $D_0$).

b) Stellen die komplette Wahrheitstabelle auf.

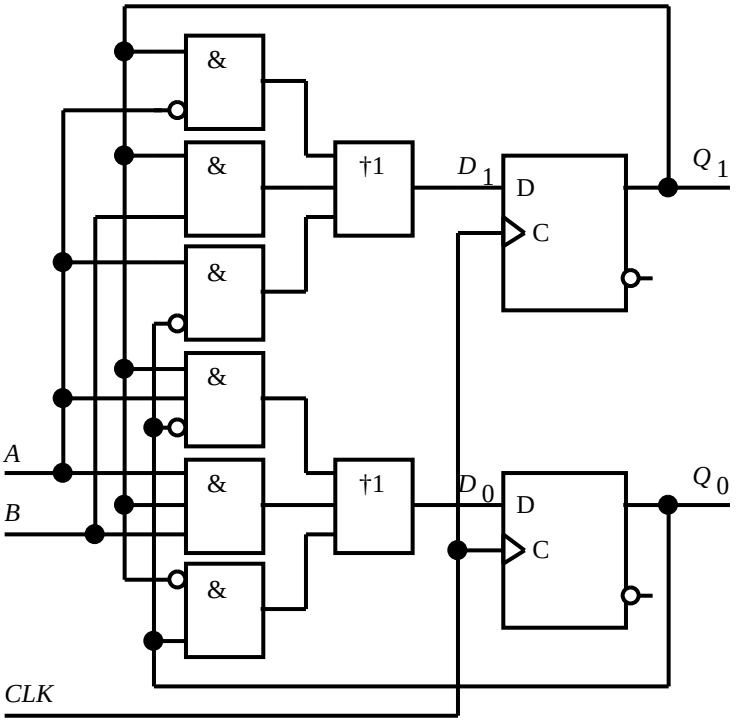
c) Erstellen Sie das Zustandsdiagramm. Die Übergangsbedingungen sind wenn nötig zu vereinfachen.

**Aufgabe 2:** In Figur 88 ist ein Zustandsdiagramm gegeben. Das Zustandsdiagramm ist auf Vollständigkeit und Korrektheit hin zu überprüfen. Geben Sie bei vorhandenen Fehlern genau an, wo sie sich befinden und womit sie in Widerspruch stehen.




Figur88:

**Aufgabe 3:** In Figur 87 ist die Schaltung eines einfachen Sequenzers dargestellt. Die Schaltung ist zu studieren und zu analysieren.



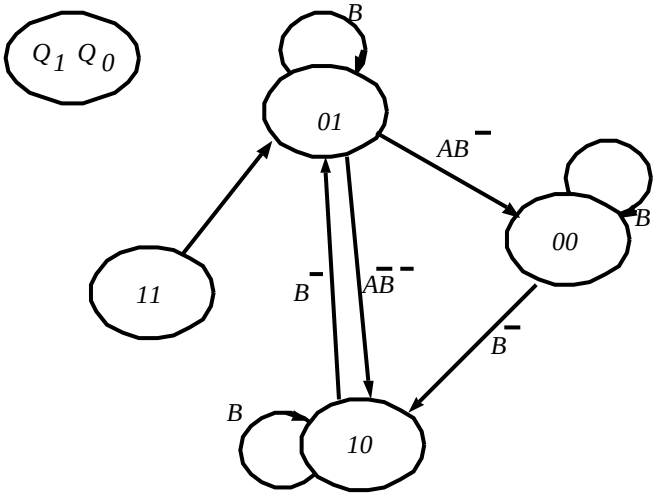

Figur 89:

a) Bestimmen Sie die Booleschen Gleichungen für die Eingänge der Flip-Flop's ($D_1$, $D_0$).

b) Stellen die komplette Wahrheitstabelle auf.

c) Erstellen Sie das Zustandsdiagramm. Die Übergangsbedingungen sind wenn nötig zu vereinfachen.

**Aufgabe 4:** In Figur 90 ist ein Zustandsdiagramm dargestellt. Entwerfen Sie einen Medwedjew Automaten, welcher den skizzierten Ablauf ausführt.



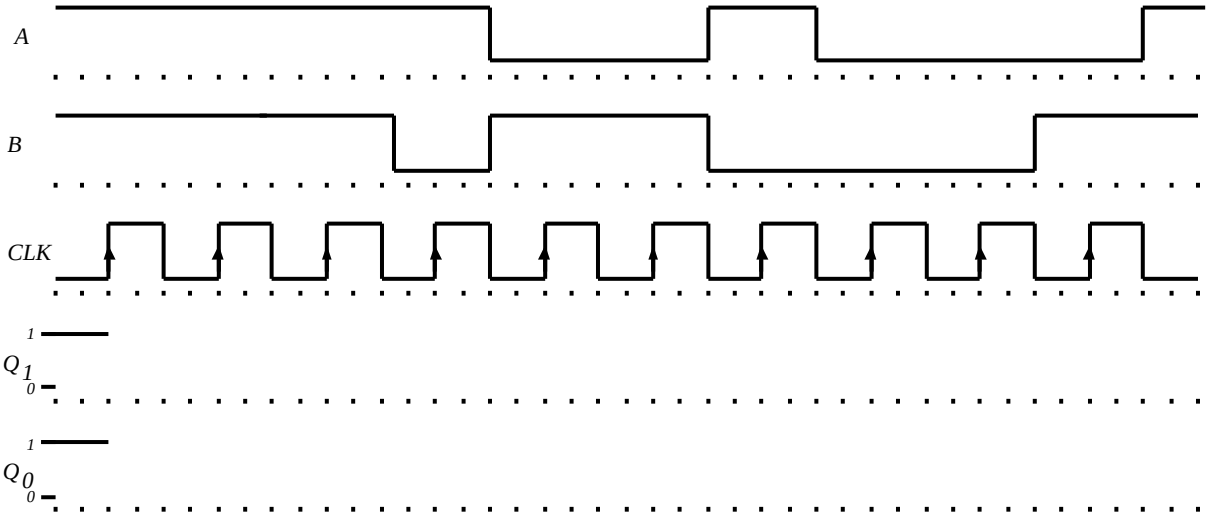

Figur 90:

a) Stellen Sie die Übergangstabelle für die Funktion *f* des Automaten dar.

b) Entwickeln Sie die minimisierten Booleschen Gleichungen für die Erzeugung der neuen Zustandswerte.

c) Entwerfen Sie das Schema des kompletten Medwedjew Automaten.

**Aufgabe 5:** Die Funktionsweise der Zustandsmaschine aus Figur 90 werde anhand einer Eingangssequenz analysiert. Zu Beginn seien sämtliche Flip-Flops an den nichtinvertierenden Ausgängen auf “1” gesetzt. Geben Sie den Verlauf der Ausgangssignale der Zustandsmaschine an, wenn die Eingangssignale den Verlauf gemäss Figur 91 aufweisen. Die steigende Flanke des *CLK* Signales ist die aktive Flanke des Zustandsregisters.




Figur91:

**Aufgabe 6:** Entwurfsbeispiel: Black-Jack Dealer. Es ist eine logische Schaltung zu entwerfen, welche das Verhalten des Gebers beim Black-Jack Kartenspiel nachahmt.

**Spielbeschreibung:** Beim Black-Jack Spiel ist ein Kartengeber mit einem oder mehreren Spielern beteiligt. Ziel des Spieles ist es, durch das Verlangen von einer Karte nach der anderen, möglichst nahe an 21 Kartenpunkte zu gelangen, diese aber nicht zu überschreiten. Die Jasskarten weisen Punktzahlen zwischen 1 und 11 auf, wobei das Ass mit 11 oder mit 1 Punkt verrechnet werden darf.

**Spielregeln:** Die Spieler bestimmen ihr Verhalten beim Spiel selbst, der Kartengeber als Spielkasinoangestellter darf nicht auf Risiko spielen, sondern muss, wenn er selbst Karten nimmt, dies nach ganz bestimmten Regeln tun. Der Kartengeber fordert für sich eine Karte nach der anderen an, bis seine Punktzahl 16 überschritten hat. Dabei verrechnet er für das Ass 11 Punkte, sofern er die Maximalpunktzahl 21 nicht überschritten hat, andernfalls verrechnet er für das Ass 1 Punkt und spielt gemäss seinen Regeln weiter.

**Aufgabenstellung:** Es ist eine logische Schaltung zu entwerfen, welche das Verhalten des Black-Jack Dealers nachempfindet. Zur Kommunikation mit der Umwelt weise die Schaltung die folgenden Einsignale

- Neue Karte bereit (card ready aktiv hoch: *cardReady*)
- Punktzahl der neuen Karte (card value: *cardValue*, 4 Bits)
- Spielstart (reset aktiv tief: *reset*)
- Systemtakt (clock: *clk*)

und die folgenden Ausgänge auf:

- Punktzahl überschritten (broke aktiv hoch: *broke*)
- Halt, genügend Punkte gesammelt (hold aktiv hoch: *hold*)
- Anforderung für eine weitere Karte (new card aktiv hoch: *newCard*)
- Gesamtpunktzahl am Spielende angeben (score: *score*, 5 Bits)

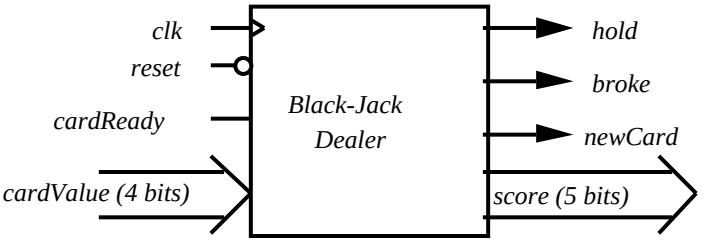
Die Schaltung ist in Figur 92 als Black-Box dargestellt.

a) Studieren Sie den Einsatz der Hardwarestrukturen aus Figur 93 und Figur 94 für den Black-Jack Dealer.

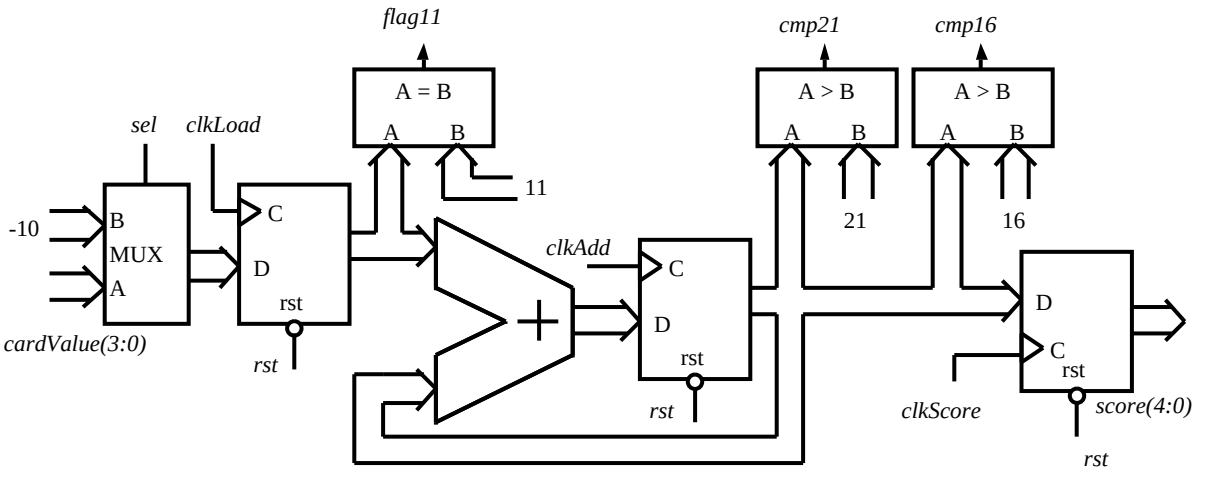
b) Legen Sie die Eingänge und Ausgänge der Zustandsmaschine fest. Welche Ausgänge dürfen keine Hazards aufweisen.

c) Entwickeln Sie ein Zustandsdiagramm, welches die Spielregeln ohne berücksichtigung der Regeln der Asse bearbeiten kann.

d) Entwickeln Sie ein Zustandsdiagramm, welches die kompletten Spielregeln bearbeiten kann.




Figur 92: Black-Jack Dealer Schnittstelle mit dem Be-

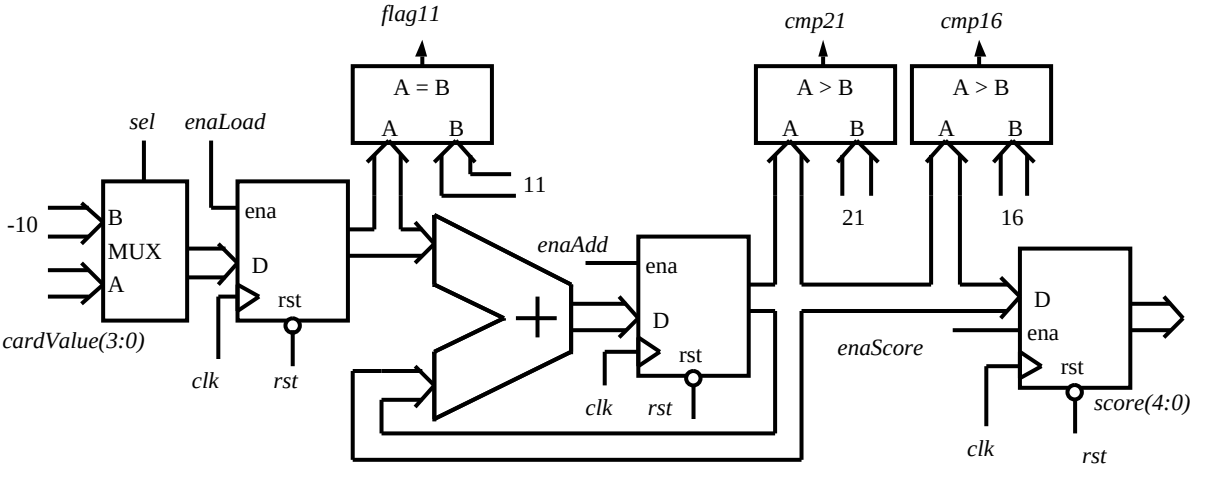



*clk* :xxx positive edge triggerd
*sel* : sel=1, input A selected
comparator out = "1" if comparison achieved

FSM output definition

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *hold* | *broke* | *newCard* | *sel* | *clkLoad* | *clkAdd* | *clkScore* | *rst* |

Figur 93 Daten-Pfad des Black-Jack Dealers, halbsynchroner Design




*clk* : positive edge triggerd
*sel* : sel=1, input A selected
comparator out = "1" if comparison achieved

FSM output definition

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *hold* | *broke* | *newCard* | *sel* | *enaLoad* | *enaAdd* | *enaScore* | *rst* |

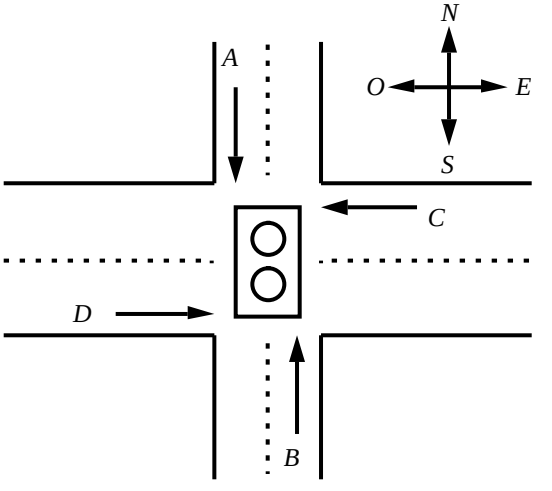
Figur 94 Daten-Pfad des Black-Jack Dealers, synchroner Design

# Technique Numérique Exercices: Serie 1

*M. Jacomet*
*Ecole d'Ingénieurs Bienne*

**Problèm 1:** La Figur 95 nous montre l'intersection entre une route principale et une route secondaire. Des capteurs de voitures ont été placés le long des voies *C* et *D* (route principale) et des voies *A* et *B* (route secondaire). Les sorties de ces capteurs sont à "0" quand il n'y a pas de voitures et à "1" quand il y en a. Le feu de circulation se trouvant à cette intersection est commandé par les règles de décision suivantes:

- Le feu E-O est vert quand il y a en mêmes temps des voitures dans les deux voies *C* et *D*.
- Le feu E-O est vert quand il y a des voitures dans *C* ou *D* et quand il y en a dans *A* ou dans *B* mais pas dans les deux.
- Le feu N-S est vert quand il y a des voitures dans les voies *A* et *B* et qu'il y en a dans *C* ou dans *D* mais pas dans tous les deux.
- Le feu N-S est aussi vert quand il y a des voitures dans *A* ou *B* et qu'il y a pas des voitures dans *C* ou *D*.
- Le feu E-O est vert quand il n'y a pas de voitures du tout.




Figur 95:

a) Peut être il y a des situations quand le feu n'est pas encore définit. Cherchez si nécessaire les règles qui manquent pour obtenir un feu de circulation qui se comporte logiquement.

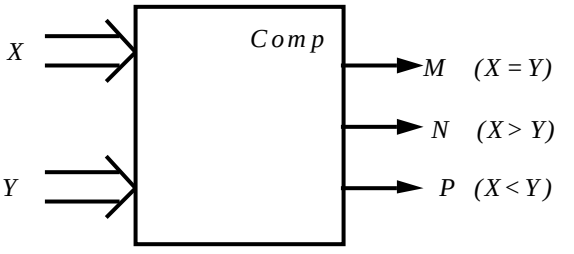
b) En utilisant le capteurs *A, B, C* et *D* comme entrées numériques, concevez un circuit logique qui commande le feu de circulation. Ce circuit a deux sorties soit E-O et N-S, qui prennent la valeur "0" quand le feu doit être vert et la valeur "1" quand la

feu doit être rouge. Simplifiez le plus possible ce circuit et illustrez toutes les étapes du développement.

**Problèm 2:** La Figur 96 représente un détecteur de grandeur relative (**comparateur**) qui accepte deux nombres binaires de 2 bits **x** ($x_1$, $x_0$) et **y** ($y_1$, $y_0$) et qui établit si ces nombres sont égaux et, dans la négative, celui est le plus grand. Il possède trois sorties définies ainsi:

- M=1 si les deux nombres sont égaux.
- N=1 si **x** est supérieure à **y**.
- P=1 si **y** est supérieure à **x**

Figur 96:




a) Concevez les diagrammes de Karnaugh du comparateur pour tous les trois sorties.

b) Concevez et minimisez le circuit logique du comparateur qui comprend quatre entrées et les deux sorties *N* et *P*. Dessinez le schéma.

c) Concevez la fonction booléenne pour la sortie *M* et utilisez les règles de l'algèbre booléenne pour une fonction qui peut être réalisé par des portes OU exclusif.

**Problèm 3:** Prouvez la suivante relation booléenne:

$$A + (B \cdot C \cdot D) = (A + B) \cdot (A + C) \cdot (A + D)$$

**Problèm 4:** Simplifiez les fonctions définîtes par les diagrammes de Karnaugh aux Figur 97, Figur 98 et Figur 99 et cherchez les fonctions booléennes.

Figur 97:

| DE \ ABC | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 01 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

Figur 98:

| *AB* \ *CDE* | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 01 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 11 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 10 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

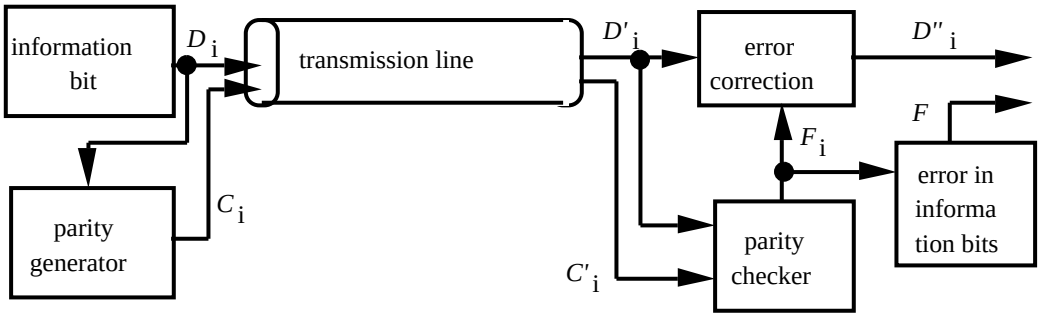
Figur 99:

| DE \ ABC | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 1 | 1 | X | 0 | 0 | 0 | X | 1 |
| 01 | 0 | 1 | X | 1 | 1 | X | 0 | 0 |
| 11 | 0 | 1 | 1 | 1 | 1 | X | X | 0 |
| 10 | 1 | X | X | 0 | 0 | X | X | 1 |

Figur100:

| *AB* \ *CDE* | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 01 | X | 0 | 0 | 1 | X | 1 | 1 | 0 |
| 11 | 0 | 0 | 0 | 0 | X | 1 | 1 | 0 |
| 10 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

**Problèm 5:** La transmission de la donné $D_i$ (4 bit) est effectué en utilisant un code Hamming, avec 3 bits de parités $C_i$ pour détecter et localiser une faute simple (voir Figur 101). Les 3 bits de parités gênèrent une parité pair de tous les bits contrôlés. Le bit de parité $C_1$ vérifie les bits $D_7$, $D_5$, $D_3$, le bit de parité $C_2$ vérifie $D_7$, $D_6$, $D_3$ et $C_4$ vérifie $D_7$, $D_6$, $D_5$.




Figur101:

a) Dessinez la tableau contenant tous les codes permises (7 bits).

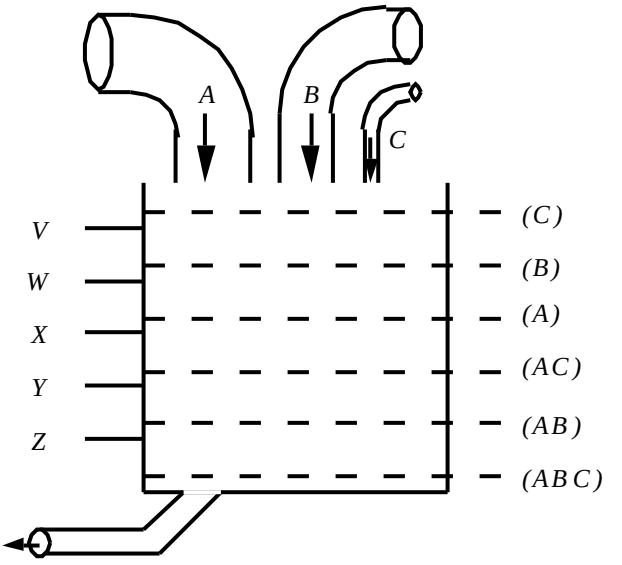
b) Les signaux $F_4$, $F_2$, $F_1$ prennent la valeur “1”, si une erreur de parité est détéctée parmis les bits de parités $C'_4$, $C'_2$, $C'_1$. Dessinez les diagrammes de Karnaugh pour les trois signaux $F_i$ (voir Figur 101 bloc “test parité”).

c) Trouvez l’équation booléenne de $F_1$ et essayez d’écrire l’éxpression par des opérations “exclusive ou”, aussi loin que c’est possible. Dessinez le schéma logique du circuit.

d) Dessinez les schémas logiques pour les autres deux bits “erreurs de parité” $F_4$, $F_2$.

e) Développez un circuit logique avec la sortie *F*, qui prend la valeur “1” lorsqu’un bit de donné est transmise fausse et la correction d’un bit de donné devient alors nécessaire (table de vérité, diagramme Karnaugh, schéma).

**Problèm 6:** Un réservoir d'eau est alimenté par 3 conduits d'apport, dont le rendement diminue de *A* vers *C*. Les capteurs *V, W, X, Y* et *Z* servent à mesurer le niveau du liquide. La



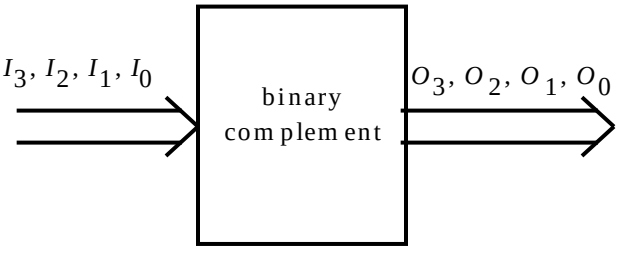

Figur102:

Figur 102 indique les conditions pour l'actionnement des conduits (ouverture = logique "1"). Quand le niveau dépasse *V,* le conduit *C* doit compenser les pertes d'évaporation . Déterminez *A, B, C*, c'est-à-dire les fonctions pour commander les conduits.

**Problèm 7:** Développez un montage pour fournir le complément par rapport à 16 des nombres binaires à 4 chiffres. Dessinez les variables d'entrée par $I_3$, $I_2$, $I_1$, $I_0$ et de sortie par $O_3$, $O_2$, $O_1$, $O_0$ ($I_3$ et $O_3$ sont les MSB's).

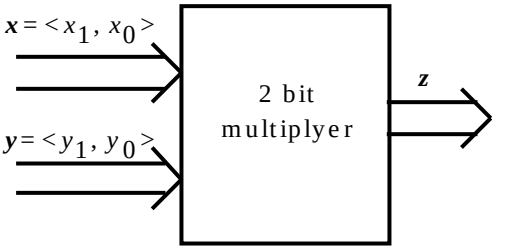
a) Déterminez la table de vérité pour le circuit.

b) Développez le diagramm de Karnaugh, minimisez les equations et dessinez le schema.




Figur103:

**Problèm 8:** Développez un montage logique qui permet de multiplier deux nombres binaires de deux bits, **x** ($x_1$, $x_0$) et **y** ($y_1$, $y_0$). Le produit est de la forme **z**($z_3$, $z_2$,$z_1$, $z_0$). Déterminez et simplifiez les équations booléennes qui définissent la sortie **z**.

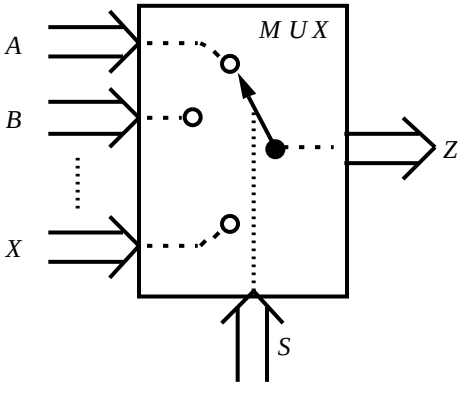



Figur104:

**Problèm 9:** Divisez par éctrit en système binaire les deux nombres $146_{dec}$ et $6_{dec}$. Calculez la valeur exacte:

$146_{dec}$:$6_{dec}$=?

**Problèm 10:** Un **multiplexeur** ou sélecteur de données est un circuit logique ayant plusieurs groupes d'entrées de données mais seulement un groupe de sortie qui communique les données. L'aiguillage du groupe des entrées de données qui nous intéresse en sortie est commandé par les entrées SELECT (S). La Figur 105 illustre le symbole d'un multiplexeur général (MUX). Un multiplexeur se comporte comme un commutateur dans lequel un code numérique appliqué aux entrées SELECT commande les entrées de données qui sont raccordées à la sortie.




Figur105:

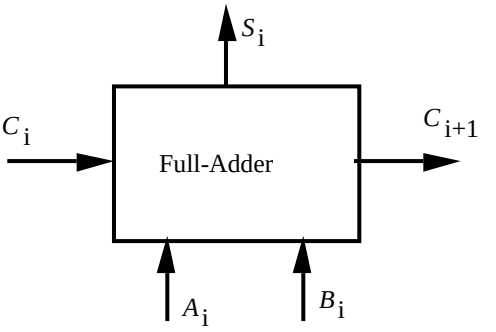
Concevez un multiplexeur avec deux groupes d'entrées à deux bits (groupe *A*: $A_1$, $A_0$, groupe *B*: $B_1$, $B_0$), la sortie *Z* ($Z_1$, $Z_0$) et le signal *S* (l'entrée SELECT). Si on applique *S*=1 au sélecteur, le groupe *B* des données est raccordé à la sortie.

a) Développez la table de vérité.

b) Cherchez la fonction booléenne minimale avec la méthode de Karnaugh pour toutes les sorties et dessinez le circuit numérique.

**Problèm 11:** Pour l'addition de deux variables binaires *A* et *B*, on peut utiliser des éléments additionneur complet branchés en série. Pour chaque bit de l'addition on a besoin d'un additionneur complet, qui calcule un bit du résultat *S*. Les reports sont transmis par le signal "carry" $C_i$ à l'étage suivant avec une valence plus élevée.

Figur106:




a) Déterminez la table de vérité de l'additionneur complet.

b) Développez un circuit numérique pour un additionneur complet. (Diagramme Karnaugh, minimisation, équation booléenne, circuit logique).

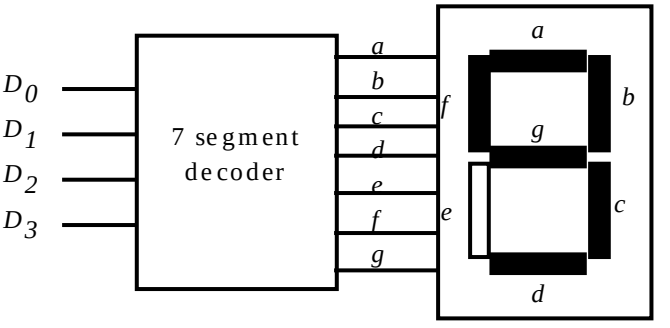
**Problèm 12:** Développez un montage logique qui est capable de trouver tous les nombres premiers entre 0 et 31 en générant un signal "1". La variable d'entrée **x** ($x_4$, $x_3$, $x_2$, $x_1$, $x_0$). (table de vérité, diagramme de Karnaugh, équation booléenne, schéma).

**Problèm 13:** Simplifiez l'équation suivante d'après l'algorithme de Quine et McCluskey.

Y=A'BCD'E'F' + A'BCD'E'F + A'BCD'EF' + A'BCD'EF + A'BCDE'F' + A'BCDE'F + A'BCDEF' + A'BCDEF + ABC'DEF + ABC'DE'F + ABC'D'EF + ABC'D'E'F + AB'C'DEF +AB'C'DE'F + AB'C'D'EF + AB'C'D'E'F + ABCDE'F + ABCDE'F' + ABC'DE'F'

**Problèm 14:** Vous devez développer le décodeur pour un affichage 7 segments. A l'entrée le décodeur reçoit 4 bits BCD (binaire codé décimale) et à la sortie on a les signaux *a* jusque'a *g* pour allumer les 7 segments. Le décodeur doit pouvoir affiches les chiffres de *0* à *9* et les lettres *A, b, C, d, E* et *F.*

Figur107:



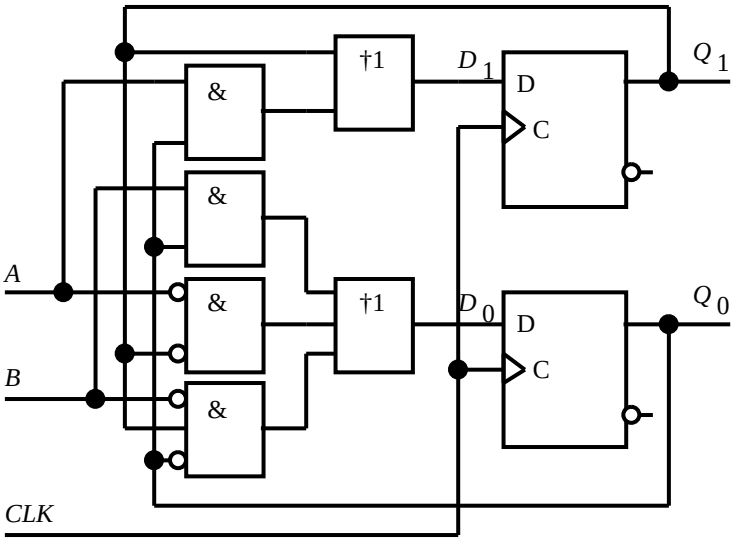

a) Donnez la table de vérité du décodeur.

b) Pour tous les segment d'affichage, cherchez les équations booléennes minimums et donnez ses équivalents au niveau du circuit logique.

# Technique Numérique Exercices: Serie 2

*M. Jacomet*
*Ecole d'Ingénieurs Bienne*

**Problèm 1:** La Figur 108 nous montre le circuit d'un séquenceur très simple. Etudiez et analysez le circuit.



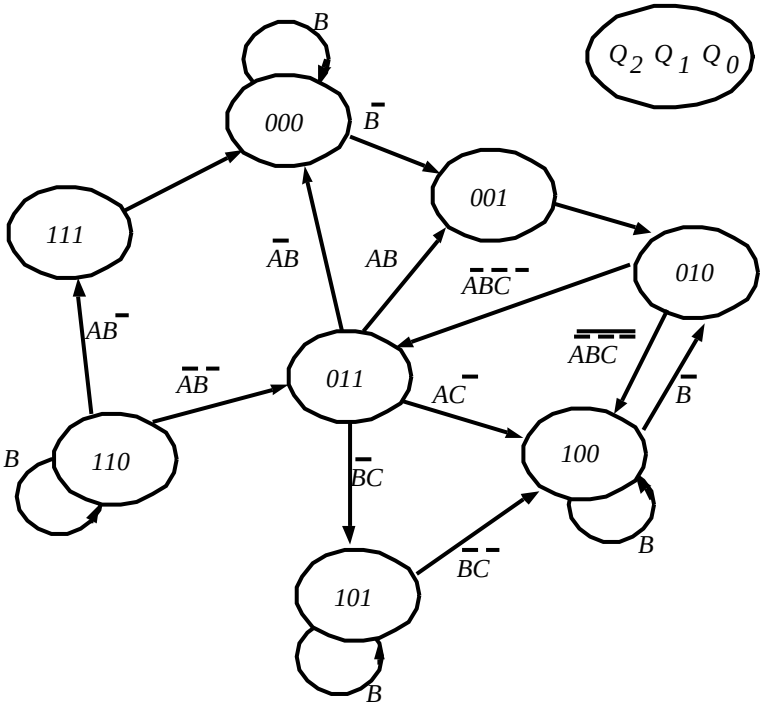

Figur108:

a) Déterminez les fonctions booléennes pour tous les entrés deux ($D_1$, $D_0$) des bascules.

b) Déterminez la table de transition.

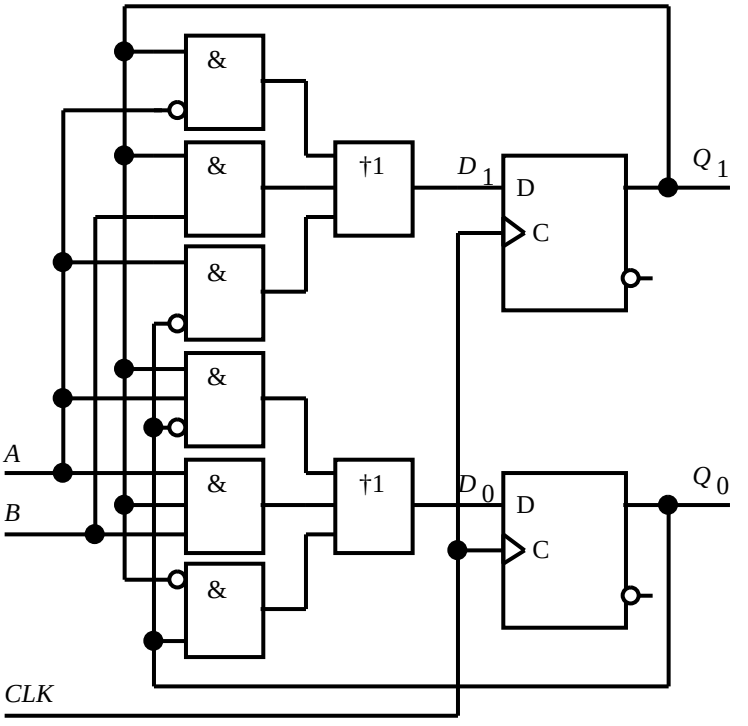
c) Dessinez le diagramme d'état. Simplifiez les conditions de transition si possible.

**Problème 2:** La Figur 109 représente un diagramme d'états. Il faut contrôler le diagramme, vérifier s'il est incomplet ou inconsistant. S'il y a des fautes, il faut spécifier les fautes exactement.




Figur109:

**Problème 3:** La Figur 108 nous montre le circuit d'un séquenceur très simple. Etudiez et analysez le circuit.



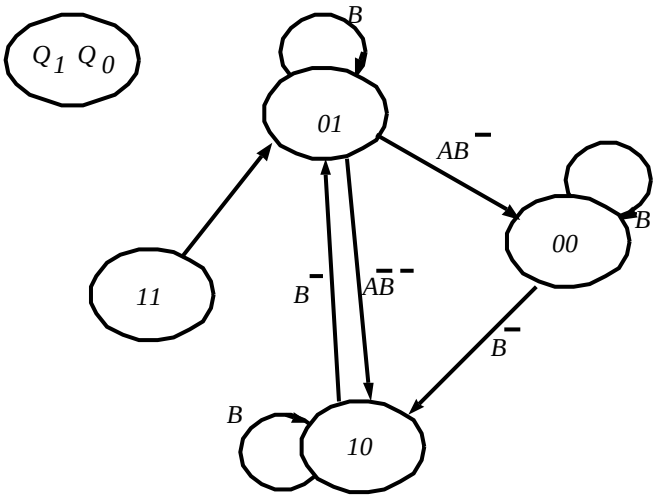

Figur110:

a) Déterminez les fonctions booléennes pour tous les deux entrés ($D_1$, $D_0$) des bascules.

b) Déterminez la table de transition.

c) Dessinez le diagramme d'état. Simplifiez, si possible, les conditions de transition.

**Problème 4:** Un diagramme d'états est illustré en Figur 111. Développez une machine d'états finis du type Medwedjew qui satisfait le fonctionnement décrit dans la figure.



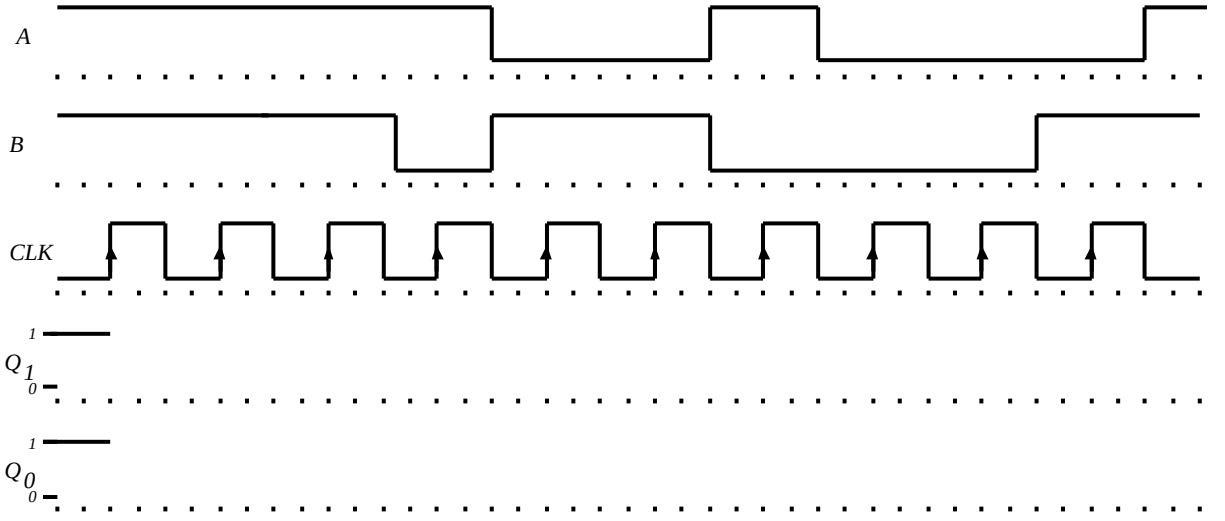

Figur111:

a) Donnez la table de transition de la fonction *f* de la machine d'états finis..

b) Développez les équations booléennes minimisées qui génèrent les nouveaux états.

c) Concevez le schéma complet de la machine d'états finis.

**Problème 5:** Le comportement de la machine d'états finis de la Figur 111 est analysé avec une séquence des entrés. Aux début tous les flip-flops sont initialisés aux valeurs logiques "1". Dessinez les valeurs des signaux de sorties dans le temps, si les entrés de la Figur 112 attaquent la machine d'états finis. Les flancs montants du signal *CLK* représentent le flanc actif du registre d'états finis.




Figur112:

**Problèm 6:** Circuit d'exemple: Black-Jack Dealer. On doit développer un circuit logique, qui est capable de le joueur du casino de la banque qui distribue les cartes pour jouer au black-jack.

**Description du jeu:** Dans le jeu black-jack il y a un joueur qui distribue les cartes (membre du casino) qui joue contre un où plusieurs joueurs. Le but du jeu est de reprendre une carte après l'autre est d'approcher le plus possible un total de 21 points sans les dispasser. Les cartes de jass donnent des points entre 1 et 11, c'est au joueur de décider si son ass compte 1 point ou 11 points.

**Règles du jeu:** Le Black-Jack Dealer, ayant employé comme du casino ne peut pas jouer avec des risques. S'il reprend des cartes par lui-même, il doit obéir à des règles biens définies. Il commande une carte après l'autre, jusqu'à ce qu'il a un total de plus que 16 points. Il compte un ass comme 11 points, sauf sile total dépasse 21 points. Dans ce cas il compte l'ass avec 1 point et continue à jouer d'après ses règles.

**Description du travail:** Il faut développer un circuit logique, qui remplace le Black-Jack Dealer (joueur du casino). Pour communiquer avec l'entourage le circuit possède les signaux suivants:

- une nouvelle carte est disponible (carte prête active haut: *cardReady*)
- points de la nouvelle carte (valeur de la carte: *cardValue*, 4 bits)
- début du jeu (reset asynchrone active bas: *reset*)
- horloge globale (horloge: *clk*)

et les sortie suivantes:

- trop de points, perdue (perdue active haut: *broke*)
- stop, suffisamment de points (stop active haut: *hold*)
- commander une nouvelle carte (nouvelle carte active haut: *newCard*)
- à la fin du jeu afficher les points total (points total: *score*, 5 bits)

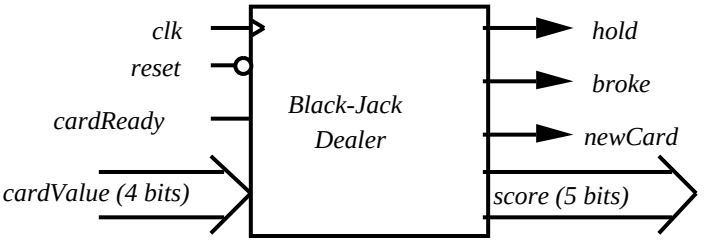
Le schéma bloc du Black-Jack Dealer, comme il est décrit dans le cours technique numérique, est dessinéz dans la Figur 113. .

a) Étudiez l'utilisation du circuit "data-path" de la Figur 114 et la Figur 115 pour le Black-Jack Dealer.

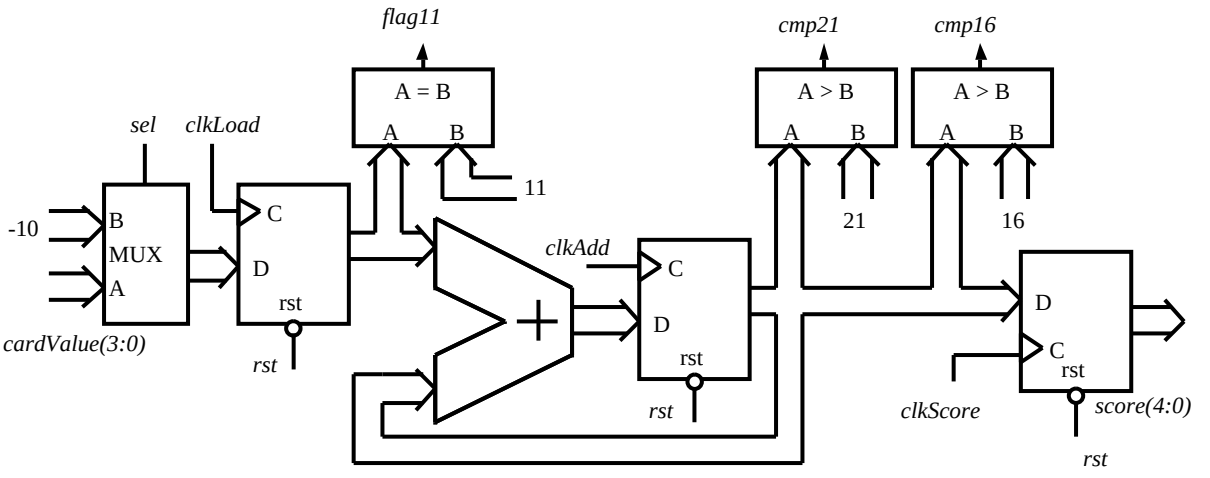
b) Définissez les entrées et sorties de la machine à états finis. Quelles sont les sorties sensibles qui ne doivent pas prendre des valeurs aléatoires (no hazards).

c) Développez une machine à état finis, qui est capable de jouer, sans pourtant respecter les règles concernant l'ass.

d) Développer une machine à état finis, qui est capable de jouer en respectant toutes les règles du jeu.




Figur 113: : Interface du Black-Jack Dealer avec l'entourage.

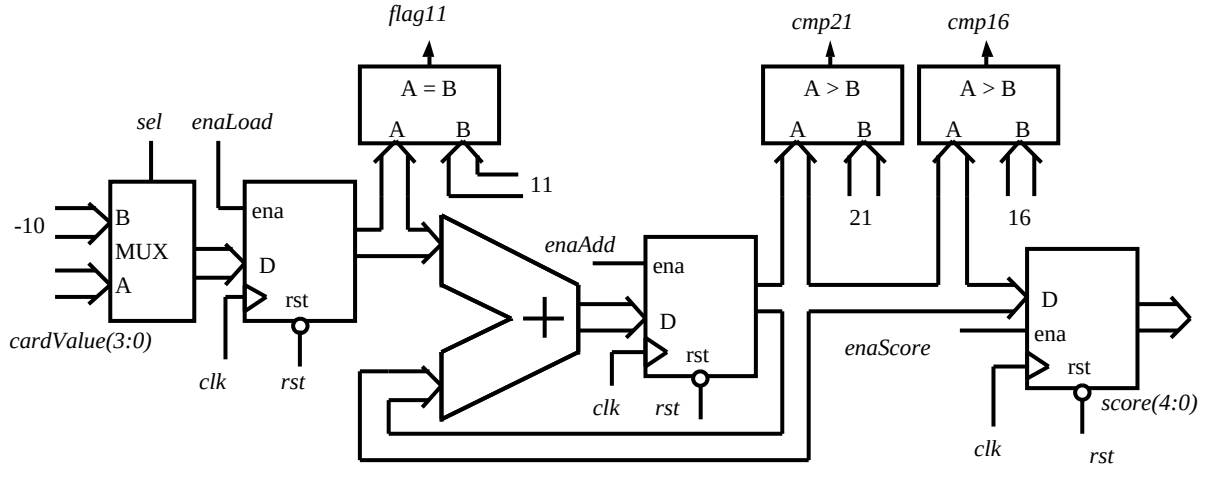



*clk* :xxx positive edge triggerd
*sel* : sel=1, input A selected
comparator out = "1" if comparison achieved

FSM output definition

| *hold* | *broke* | *newCard* | *sel* | *clkLoad* | *clkAdd* | *clkScore* | *rst* |
|---|---|---|---|---|---|---|---|

Figur 114 ”data-path” du circuit Black-Jack Dealer, circuit semi-synchron.




*clk* : positive edge triggerd
*sel* : sel=1, input A selected
comparator out = "1" if comparison achieved

FSM output definition

| *hold* | *broke* | *newCard* | *sel* | *enaLoad* | *enaAdd* | *enaScore* | *rst* |
|---|---|---|---|---|---|---|---|

Figur 115 ”data-path” du circuit Black-Jack Dealer, circuit synchron.