

## 1. KOMBINATORISCHE LOGIK

Einfache logische Operationen ohne Speicher.

Die Ausgänge ändern sich nur in Abhängigkeit von den Eingängen. Jeder Ausgang  $y_i$  lässt sich durch eine boolesche Funktion  $f_i$  aller Eingänge beschreiben:  $y_i = f_i(x_0, x_1, \dots, x_n)$

Für  $N$  Eingänge gibt es  $2^N$  Eingangskombinationen.

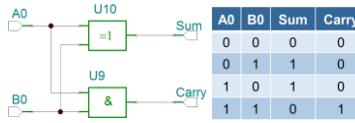
### Logische Operatoren

Function	Boolean Algebra	IEC 60617-12 since 1997
AND	$A \& B$	
OR	$A \# B$	
Buffer	$A$	
XOR	$A \$ B$	
NOT	$\text{!}A$	
NAND	$\text{!}(A \& B)$	
NOR	$\text{!}(A \# B)$	
XNOR	$\text{!}(A \$ B)$	

A	B	!A	A&b	A#B	A\\$B
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

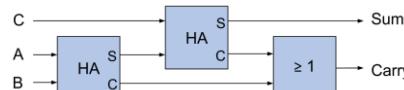
### 1-Bit Halb-Addierer (HA)

Bildet die Summe und Carry von zwei Bits.

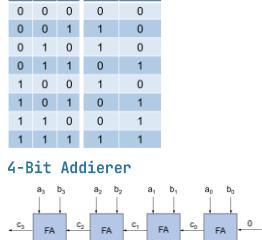


### 1-Bit Voll-Addierer (FA)

Bildet die Summe und Carry von zwei Bits und vorherigem Carry.



### 4-Bit Addierer

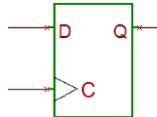


## 2. SEQUENTIELLE LOGIK

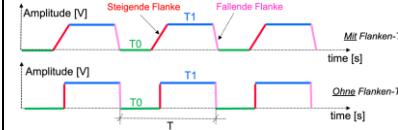
Sequentielle Logik hat gegenüber der Kombinatorischen Logik mehrere Zustände und enthält Speicher. Grundelement dafür sind D-Flip-Flops.

### D-Flip-Flop

Wert am Eingang  $D$  wird gespeichert und an den Ausgang  $Q$  übertragen, wenn  $C$  von 0 auf 1 wechselt.  
Ein Flip-Flop hat 2 Zustände.  
 $N$  Flip-Flops haben  $2^N$  Zustände.



### Clock Signal



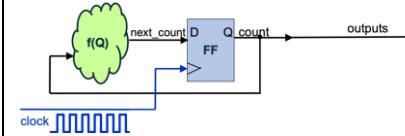
Periode  $T = T_0 + T_1[s]$

Frequent  $f = \frac{1}{T}[\text{Hz}]$

Duty Cycle  $\frac{T_1}{T}$

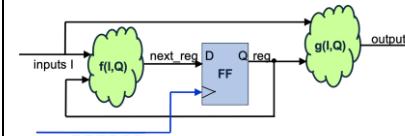
### Zähler (Counter)

Reihenfolge der Zustände und Zustand vom Ausgang hängt vom internen Zustand/Logik ab.



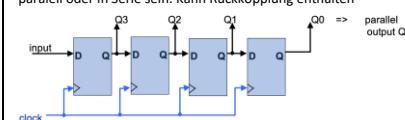
### Zustandsautomaten (Finite State Machine / FSM)

Der Ausgang ist abhängig vom Input und dem Status des Speichers. Der FF-Ausgangswert  $Q$  entspricht dem Zustand des Automaten.



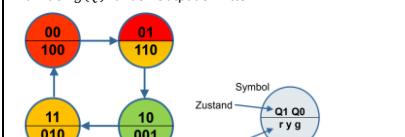
### Schieberegister

Jedes FF verzögert den Input um einen Takt. Schieberegister können parallel oder in Serie sein. Kann Rückkopplung enthalten.



### Zustandsdiagramm (Bsp. Ampel)

Um ein Zustandsautomat von einem Zustandsdiagramm zu bauen, muss man die Funktion  $f(Q)$  für den nächsten Zustand und die Funktion  $g(Q)$  für den Output ermitteln.



### Zustandslogik / Ausgangslogik

Q1	Q0	D1	D0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

state  
red\_on yellow\_on green\_on

$D0 = !Q0$ ,  $D1 = Q0 \oplus Q1$

red =  $!Q1$ , yellow =  $Q0$ , green =  $!Q0 \& Q1$

### Zeitverlaufsdiagramme / Vektoren

Signal wechselt von 0 nach 1	
Signal wechselt von 1 nach 0	
Vektor (= Bus oder Signalgruppe)	
Vektor wechselt von unbekanntem in definierten Wert	
Vektor wechselt von bekanntem in undefinierten Wert	

Bsp.:

Inputs	clock	D[1:0]	Q[1:0]	Q[1:0]	Objects
		01	01	01	RYG[2:0]
		10	00	00	100
		11	01	01	110
		00	10	10	001
					1
					0
					1

### Multiplication

$$\begin{array}{r} 1\ 0\ 1 \times 1\ 1\ 1\ 0 \\ \hline 1\ 1\ 1\ 0 \\ + 0\ 0\ 0\ 0 \\ + 1\ 1\ 1\ 0 \\ \hline 1\ 1 \\ 1\ 0\ 0\ 0\ 1\ 1\ 0 \end{array}$$

### Division

$$\begin{array}{r} 1\ 1\ 0\ 1\ 1\ 0 \div 1\ 0\ 1\ 0 = 1\ 0\ 1 \\ - 1\ 0\ 1\ 0 \\ \hline 0\ 0\ 1\ 1\ 1 \\ - 0\ 0\ 0\ 0 \\ \hline 0\ 1\ 1\ 1\ 0 \\ - 1\ 0\ 1\ 0 \\ \hline 0\ 1\ 0\ 0 \end{array}$$

### Darstellung negativer Zahlen

Binär	Dezimal	Sign+Magna.	Einerkompl.	Zweierkompl.	Excess-8
1111	15	-7	+0	-1	+7
1110	14	-6	-1	-2	+6
1101	13	-5	-2	-3	+5
1100	12	-4	-3	-4	+4
1011	11	-3	-4	-5	+3
1010	10	-2	-5	-6	+2
1001	9	-1	-6	-7	+1
1000	8	0	-7	-8	0
0111	7	+7	+7	+7	-1
0110	6	+6	+6	+6	-2
0101	5	+5	+5	+5	-3
0100	4	+4	+4	+4	-4
0011	3	+3	+3	+3	-5
0010	2	+2	+2	+2	-6
0001	1	+1	+1	+1	-7
0000	0	+0	+0	0	-8

### 3. ZAHLENSYSTEME

4 Bit -> Nibble

8 Bit -> Byte (Octet)

Higher Nibble Lower Nibble

Dec Bin Hex Oct

Dec	Bin	Hex	Oct
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	8	10
9	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17

### Stellenwertsysteme

Um in Zahlen in dezimal umzurechnen, kann folgende formel verwendet werden, wobei  $i$  die Stelle (begonnen bei 0),  $b$  die Basis und  $a_i$  die Ziffer an Stelle  $i$ :

$$Z = \sum a_i \cdot b^i$$

Das Schieben um  $s$  Stellen wird wie folgt berechnet:

$$Z_{\text{neu}} = Z \cdot b^s$$

### Addition

$$\begin{array}{r} \text{Erster Summand} & 1\ 1\ 0\ 1\ 1\ 0 \\ \text{Zweiter Summand} & + 1\ 0\ 1\ 1\ 1\ 1 \\ \text{Carry} & \text{Sum} \\ \hline & 1\ 1\ 1\ 0\ 0\ 1\ 1 \end{array}$$

### Subtraktion

$$\begin{array}{r} \text{Minuend} & 1\ 1\ 0\ 1\ 1\ 0 \\ \text{Subtrahend} & - 1\ 0\ 1\ 1\ 1\ 1 \\ \text{Carry} & \text{Sum} \\ \hline & 1\ 1\ 1\ 0\ 0\ 1 \end{array}$$

$$\begin{array}{r} \text{Differenz} & 0\ 0\ 0\ 1\ 1\ 1 \\ \hline & 0\ 0\ 0\ 1\ 1 \end{array}$$

### 4. INFORMATIONSTHEORIE

#### Discrete Memoryless Source (DMS)

Eine diskrete gedächtnislose Quelle (DMS) ist ein Modell, das eine Informationsquelle beschreibt, die eine endliche Menge von Symbolen  $S = \{s_1, s_2, \dots, s_m\}$  mit bestimmten Wahrscheinlichkeiten  $P = \{p(s_1), p(s_2), \dots, p(s_m)\}$  erzeugt, wobei jedes Symbol unabhängig von den vorherigen Symbolen ausgewählt wird.

### Binary Memoryless Source (BMS)

Eine binäre gedächtnislose Quelle (BMS) ist ein Spezialfall der DMS, bei dem die Symbolmenge auf zwei Symbole  $S = \{0, 1\}$  beschränkt ist. Die Wahrscheinlichkeiten für die beiden Symbole sind  $P = \{p(0), p(1)\}$ , wobei  $p(0) + p(1) = 1$  gilt.

#### Auftretenswahrscheinlichkeit

Die Auftretenswahrscheinlichkeit eines Symbols  $s_i$  in einer Nachricht wird durch die relative Häufigkeit bestimmt, wobei  $N_i$  die Anzahl der Vorkommen des Symbols  $s_i$  und  $N$  die Gesamtzahl der Symbole in der Nachricht ist.

#### Informationsgehalt

Der Informationsgehalt  $I(s_i)$  eines Symbols  $s_i$  wird durch die Formel definiert.

$$I(s_i) = -\log_2(p(s_i)) \text{ [Bit]}$$

definiert. Er gibt an, wie viel Information das Symbol liefert, wobei seltener Symbole mehr Information enthalten.

#### Mittlere Informationsgehalt (Entropie)

Der mittlere Informationsgehalt  $H(X)$  einer diskreten Zufallsvariable  $X$  mit den möglichen Symbolen  $S = \{s_1, s_2, \dots, s_m\}$  und den Wahrscheinlichkeiten  $P = \{p(s_1), p(s_2), \dots, p(s_m)\}$  wird durch die Formel

$$H(X) = -\sum_{i=1}^N p(s_i) \cdot \log_2(p(s_i)) \text{ [Bit]} \text{ / Symbol}$$

definiert. Er gibt den durchschnittlichen Informationsgehalt pro Symbol an.

Wenn alle Symbole die gleiche Wahrscheinlichkeit haben, vereinfacht sich die Formel zu:

$$H(X) = \log_2(N) \text{ [Bit]} \text{ / Symbol}$$

### 5. QUELLENCODIERUNG

Ziel:

- Speicherplatz sparen
- Bandbreite reduzieren
- Übertragungszeit reduzieren
- Energie sparen



Irrelevanzreduktion: Für den Empfänger bedeutungslose Datei weglassen (Verlustbehaftet)

Redundanzreduktion: Redundante Informationen weglassen (Verlustfrei). Es entsteht, wenn die Codierung mehr Bit umfasst als dessen Informationsgehalt (z.B. Morse Code: Heufige Buchstaben -> kurzer Codes)

#### Codes unterschiedlicher Länge

Codes unterschiedlicher Länge sind möglich, solange sie Präfixfrei (kein Code bildet den Anfang eines anderen Codes) sind. Bsp:

Symbol	Code	Codewortlänge
$x_0$	$c_0 = (10)$	$\ell_0 = 2 \text{ Bit}$
$x_1$	$c_1 = (110)$	$\ell_1 = 3 \text{ Bit}$
$x_2$	$c_2 = (1110)$	$\ell_2 = 4 \text{ Bit}$





## Faltungscode Coderate

Die Coderate  $R$  eines Faltungscodes ist definiert als

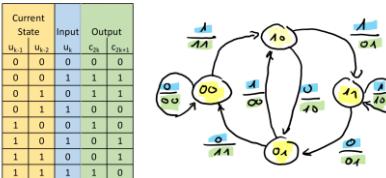
$$R = \frac{K}{2 \cdot (K + m)}$$

wobei  $K$  die Anzahl der Eingangsbits pro Zeiteinheit und  $m$  die Anzahl der Speicherzustände des Faltungscoders ist.

Im Grenzfall, wenn  $K$  gegen unendlich geht gilt:

$$\lim_{K \rightarrow \infty} R = \frac{1}{2}$$

## Zustandsdiagramm



## Freie Distanz

Die freie Distanz  $d_{free}$  eines Faltungscodes ist die minimale Hamming-Distanz zwischen zwei unterschiedlichen Codewörtern. Da Faltungscodes linear sind, gilt  $d_{free} = w_{min}$ .

Um  $d_{free}$  zu bestimmen, müssen alle möglichen Pfade im Zustandsdiagramm betrachtet werden, die vom Nullzustand ausgehen und wieder zu diesem zurückkehren.

## Optimum Free Distance (OFD)

Die OFD ist die maximale freie Distanz, die ein Faltungscode mit gegebenem Constraint Speicher  $m$  und anzahl an Generatoren  $\gamma$  erreichen kann.

$m$	$\gamma = 2$ Generatoren	$d_{free}$
2	(10) <sub>b</sub> , 111 <sub>b</sub>	5
3	(110) <sub>b</sub> , 1111 <sub>b</sub>	6
4	(1001) <sub>b</sub> , 11101 <sub>b</sub>	7
5	(10101) <sub>b</sub> , 111101 <sub>b</sub>	8
6	(1010101) <sub>b</sub> , 1100101 <sub>b</sub>	10
7	(1010011) <sub>b</sub> , 1111001 <sub>b</sub>	10
8	(10110001) <sub>b</sub> , 11110101 <sub>b</sub>	12

$m$	$\gamma = 3$ Generatoren	$d_{free}$
2	(104) <sub>b</sub> , 1111 <sub>b</sub>	8
3	(1101) <sub>b</sub> , 11011 <sub>b</sub> , 1111 <sub>b</sub>	10
4	(10101) <sub>b</sub> , 11011 <sub>b</sub> , 11111 <sub>b</sub>	12
5	(10011) <sub>b</sub> , 10101 <sub>b</sub> , 111101 <sub>b</sub>	13
6	(1011011) <sub>b</sub> , 1100101 <sub>b</sub> , 1111011 <sub>b</sub>	15
7	(10010101) <sub>b</sub> , 11011001 <sub>b</sub> , 11100101 <sub>b</sub>	18
8	(10110111) <sub>b</sub> , 11011001 <sub>b</sub> , 111001001 <sub>b</sub>	18

## Impulsantwort des Generators

Die Impulsantwort des Generators  $g_i$  ist die Ausgabe des Faltungscoders, wenn ein einzelnes 1-Bit in den Eingang gegeben wird, gefolgt von unendlich vielen 0-Bits. Für einen Faltungscode mit zwei Generatoren  $g_1$  und  $g_2$  sind die Impulsantworten:

$$g_1 = [g_{11}, g_{12}, \dots, g_{1(m+1)}]$$

$$g_2 = [g_{21}, g_{22}, \dots, g_{2(m+1)}]$$

## Mathematische Interpretation

Die Operation des Faltungscoders kann als Polynom-Multiplikation modulo 2 interpretiert werden. Das Eingangsbstrom  $u(x)$  wird mit den Generatorpolynomen  $g_1(x)$  und  $g_2(x)$  multipliziert, um die Ausgangspolynome  $v_1(x)$  und  $v_2(x)$  zu erhalten:

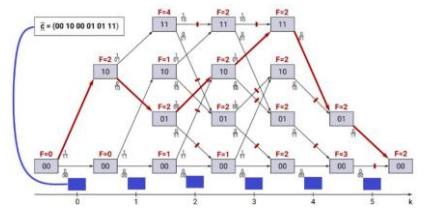
$$v_1(x) = u(x) \cdot g_1(x) \mod 2$$

$$v_2(x) = u(x) \cdot g_2(x) \mod 2$$

## Viterbi-Decoder

Der Viterbi-Decoder ist ein Algorithmus zur Maximum-Likelihood-Decodierung von Faltungscodes. Er verwendet das Zustandsdiagramm des Faltungscoders, um den wahrscheinlichsten Pfad zu bestimmen, der zu dem empfangenen Codewort führt. Der Algorithmus arbeitet in zwei Hauptschritten:

- **Vorwärtsdurchlauf:** Berechnung der Pfadmetriken für alle möglichen Zustände zu jedem Zeitpunkt.
- **Rückwärtssuchlauf:** Bestimmung des optimalen Pfads durch Rückverfolgung der Zustände mit den besten Pfadmetriken.



1. Alle möglichen Pfade einzeichnen und die Fehler notieren.
2. Fehler werden summiert
3. Am Schluss Pfad mit wenigen Fehlern zurück