

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: $y_i = f_i(x_0, x_2, \dots, x_n)$

Für N Eingänge gibt es 2^N Eingangskombinationen.

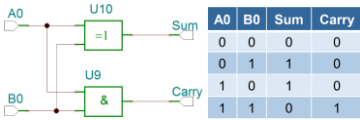
Logische Operatoren

Function	Boolean Algebra ⁽¹⁾	IEC 60617-12 since 1997
AND	$A \& B$	
OR	$A \# B$	
Buffer	A	
XOR	$A \$ B$	
NOT	$!A$	
NAND	$!(A \& B)$	
NOR	$!(A \# B)$	
XNOR	$!(A \$ B)$	

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

1-Bit Halb-Addierer (HA)

Bildet die Summe und Carry von zwei Bits.

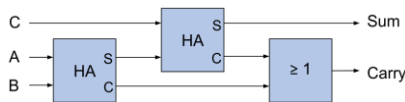


Sum = $A0 \$ B0$

Carry = $A0 \& B0$

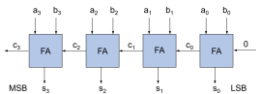
1-Bit Voll-Addierer (FA)

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



C	A	B	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

4-Bit Addierer



2. SEQUENTIELLE LOGIK

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

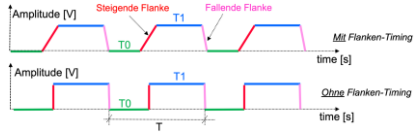
D-Flip-Flop

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



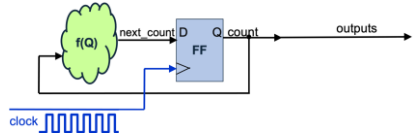
Periode $T = T_0 + T_1 [s]$

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

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

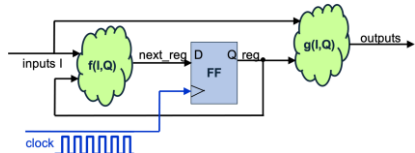
Zähler (Counter)

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



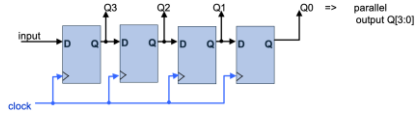
Zustandsautomaten (Finite State Machine / FSM)

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	Q1	Q0	red-on	yellow-on	green-on
00	0	0	1	0	0
01	0	1	0	1	0
10	1	0	0	0	1
11	1	1	0	0	0

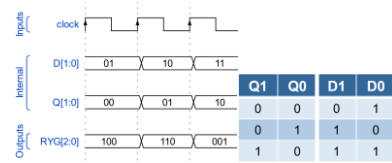
D0 = !D0, D1 = Q0 \$ Q1

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) wechselt den Wert (MSB-LSB)	
Vektor wechselt von unbekanntem in definierten Wert	
Vektor wechselt von bekanntem in undefinierten Wert	

Bsp.:



3. ZAHLENSYSTEME

4 Bit -> Nibble

8 Bit -> Byte (Octet)

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:

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

Addition

Erster Summand	1 1 0 1 0
Zweiter Summand	+ 1 0 1 1 1
Carry	1 1 1 1
Sum	1 1 0 0 0 1

Subtraktion

Minuend	1 1 0 1 0
Subtrahend	- 1 0 1 1 1
Carry	1 1 1
Differenz	0 0 0 1 1

Multiplikation

Faktoren	1 0 1 x 1 1 1 0
	1 1 1 0
	+ 0 0 0 0
	+ 1 1 1 0
Carry	1 1
Produkt	1 0 0 0 1 1 0

Division

Divisor + Divisor	1 1 0 1 1 0 + 1 0 1 0 = 1 0 1
	- 1 0 1 0
	0 0 1 1
	- 0 0 0 0
	0 1 1 1 0
	- 1 0 1 0
Rest	0 1 0 0

Darstellung negativer Zahlen

Binär	Dezimal	Sign+Magn.	Einerkomp.	Zweierkomp.	Exzess-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

2er-Komplementbildung (Vorzeichenwechsel)

Verfahren:	+2 -> -2	-2 -> +2
	0 0 0 0 0 1 0	1 1 1 1 1 1 0
* invertieren:	1 1 1 1 1 0 1	0 0 0 0 0 0 1
* 1 addieren:	0 0 0 0 0 0 1	0 0 0 0 0 0 1
	1 1 1 1 1 1 0	0 0 0 0 0 1 0
Spezialfälle:	0 -> 0	-128 -> Überlauf
	0 0 0 0 0 0 0	1 0 0 0 0 0 0
1er-Komplement	1 1 1 1 1 1 1	0 1 1 1 1 1 1
	0 0 0 0 0 0 1	0 0 0 0 0 0 1
	0 0 0 0 0 0 0	1 0 0 0 0 0 0

Verarbeitungsbreite

Register	Bezeichnung	Max uint	Max int
4 Bit	Nibble	0...15	-8...7
8 Bit	Byte	0...255	-128...127
16 Bit	Word	0...65535	-32768...32767
32 Bit	Double Word	0...4.29 · 10 ⁹	-2.15 · 10 ⁹ ... 2.15 · 10 ⁹
64 Bit	Logn Word	0...1.84 · 10 ¹⁸	-9.22 · 10 ¹⁷ ... 9.22 · 10 ¹⁷

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

$$p(s_i) = \frac{N_i}{N}$$

bestimmt, wobei N_i die Anzahl der Vorkommen des Symbols s_i und N die Gesamtanzahl der Symbole in der Nachricht ist.

Informationsgehalt

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

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

definiert. Er gibt an, wie viel Information das Symbol liefert, wobei seltene 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

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

definiert. Er gibt den durchschnittlichen Informationsgehalt pro Symbol an.

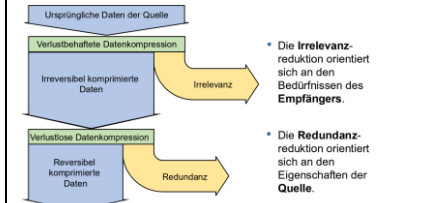
Wenn alle Symbole die gleiche Wahrscheinlichkeit haben, vereinfacht sich die Formel zu:

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

5. QUELLENCODIERUNG

Ziel:

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



Irrelevanzreduktion: Für den Empfänger Bedeutungslose Date weglassen (Verlustbehaftet)

Redundanzreduktion: Redundante Informationen weglassen (Verlustfrei). Es entsteht, wenn die Codierung mehr Bit umfasst als dessen Informationsgehalt (z.B. Morse Code: Häufige 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	$c_2 = (1110)$	$\ell_2 = 4 \text{ Bit}$

Redundanz von Codes

Der durchschnittliche Codewortlänge L eines Codes wird durch die Formel

$$L = \sum_{i=1}^N p(s_i) \cdot \ell_i \left[\frac{\text{Bit}}{\text{Symbol}} \right]$$

bestimmt, ℓ_i die Länge des entsprechenden Codeworts ist.

Die Redundanz R wird definiert durch die Formel

$$R = L - H(X) \left[\frac{\text{Bit}}{\text{Symbol}} \right]$$

- $R > 0$ Code kann noch verlustfrei komprimiert werden
- $R = 0$ Code kann nicht mehr verlustfrei komprimiert werden
- $R < 0$ Code wird verlustbehaftet komprimiert

Kompressionsrate

Die Kompressionsrate CR eines Codierungsverfahrens wird definiert als das Verhältnis der Grösse der ursprünglichen Daten D_{orig} zur Grösse der komprimierten Daten D_{comp} :

$$CR = \frac{D_{orig}}{D_{comp}}$$

Eine höhere Kompressionsrate bedeutet eine stärkere Reduktion der Datenmenge.

Run Length Encoding (RLE)

Runs werden mit Tokens codiert: (Marker, Anzahl, Code). Als Marker wird ein wenig genutzter Code verwendet. Dieses muss dafür immer codiert werden.

Bsp.: . . . TERRRRRRRRMAUGGQCSSSSSSSSSL . . .

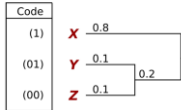
Komprimiert: . . . TER**A08RMA01AU02GWC**A10SL . . .

Huffman Code

Die Huffman-Codierung ist ein Verfahren zur optimalen Quellencodierung, das darauf abzielt, die durchschnittliche Codewortlänge zu minimieren. Der Algorithmus funktioniert wie folgt:

- Liste aller Symbole mit ihren Wahrscheinlichkeiten erstellen.
- Die beiden Symbole mit den kleinsten Wahrscheinlichkeiten auswählen und zu einem neuen Symbol zusammenfassen, dessen Wahrscheinlichkeit die Summe der beiden ist.
- Diesen Vorgang wiederholen, bis nur noch ein Symbol übrig ist.
- Den Baum von unten nach oben durchgehen und jedem Symbol einen Code zuweisen, wobei links eine 0 und rechts eine 1 hinzugefügt wird

$$P(X) = 0.80 \quad P(Y) = 0.10 \quad P(Z) = 0.10$$

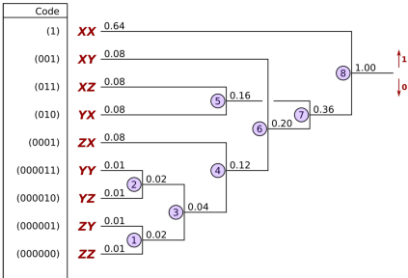


Um die Redundanz weiter zu reduzieren, kann die Huffman-Codierung auf Symbolgruppen (z.B. Paare oder Tripel von Symbolen) angewendet werden, anstatt auf einzelne Symbole. Dafür gilt:

$$p(x_1, x_2, \dots, x_n) = p(x_1) \cdot p(x_2) \cdot \dots \cdot p(x_n)$$

$$H_n(X) = n \cdot H(X) \left[\frac{\text{Bit}}{\text{n Symbole}} \right]$$

Hierbei ist $H_n(X)$ die Entropie der Symbolgruppe der Länge n .

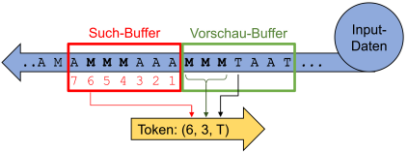


H, L, R sind in diesem Bsp. nun Bit/2 Symbole.

LZ77 (Lempel Ziv)

Die LZ77-Codierung ist ein verlustfreies Kompressionsverfahren, das auf der Ersetzung von wiederkehrenden Mustern durch Verweise basiert:

- Suche im Suchbuffer nach der längsten Übereinstimmung mit dem Anfang des Lookahead-Buffers.
- Erstelle ein Tripel (Offset, Länge, Nächstes Symbol), wobei:
 - Offset:** die Position der Übereinstimmung im Suchbuffer ist,
 - Länge:** die Länge der Übereinstimmung ist,
 - Nächstes Symbol:** das erste Symbol im Lookahead-Buffer nach der Übereinstimmung ist.
- Bei keiner Übereinstimmung: (0, 0, Nächstes Symbol)
- Wiederhole den Vorgang, bis alle Daten codiert sind.



Bsp:

Such-Buffer	Vorschau-Buffer	Offset	Länge	Symbol
10 9 8 7 6 5 4 3 2 1	A M A M M	0	0	A
10 9 8 7 6 5 4 3 2 1	A M A M M	0	0	M
10 9 8 7 6 5 4 3 2 1	A M A M M	2	2	M
10 9 8 7 6 5 4 3 2 1	A M A M M	4	2	A
10 9 8 7 6 5 4 3 2 1	A M A M M	6	4	T

Bit ganz rechts im Vorschau-Buffer dürfen nicht codiert werden.

Für den Abschluss des Codes muss eine Lösung definiert werden (Token / Grösse im Header)

LZW (Lempel Ziv Welch)

Die LZW-Codierung (Lempel-Ziv-Welch) ist ein weiteres verlustfreies Kompressionsverfahren, das auf der Erstellung eines Wörterbuchs basiert:

- Initialisiere das Wörterbuch mit allen möglichen Symbolen der Eingabedaten oder andere Charsets (ASCII).
- Lege die Eingabedaten und finde die längste Zeichenkette, die im Wörterbuch vorhanden ist.
- Gib den Index dieser Zeichenkette im Wörterbuch aus.
- Füge die längste Zeichenkette plus das nächste Symbol zur Eingabe zum Wörterbuch hinzu.
- Wiederhole den Vorgang, bis alle Daten codiert sind.

Bsp: ANAMMAAAMMTAAAT

Index	Eintrag	Token
0...255	ASCII	
256	AM	65 (A)
257	MA	77 (M)
258	AMM	256 (AM)
259	MM	77 (M)
260	MAA	257 (MA)
261	AA	65 (A)
262	AMMM	258 (AMM)
...

6. JPEG

JPEG ist ein Verlustbehaftetes Kompressionsverfahren für Bilder. Es umfasst folgende Schritte

- Farbraumkonvertierung:** RGB \rightarrow YCbCr (Y: Helligkeit, Cb/Cr: Farbe)
- Subsampling:** Chrominanz-Reduktion (z.B. 4:2:0)
- Blockbildung:** Aufteilung in 8x8-Pixel-Blöcke

- Diskrete Cosinus Transformation (DCT):** Transformation in Frequenzbereich
- Quantisierung:** Verlustbehaftete Koeffizientenreduktion
- Entropiekodierung:** Verlustlose RLE + Huffman-Kodierung

Farbraumkonvertierung

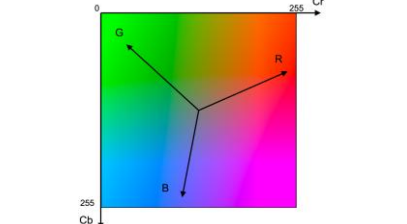
RGB wird in YCbCr umgewandelt. Berücksichtigt die Farbempfindlichkeit des Auges:

Y Luminanz (Graustufenintensität / Helligkeit)
Cb Chrominanz (Blauanteil)
Cr Chrominanz (Rotanteil)

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

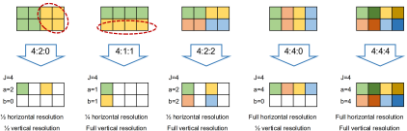
$$C_b = 0.564 \cdot (B - Y)$$

$$C_r = 0.713 \cdot (R - Y)$$



Supsampling

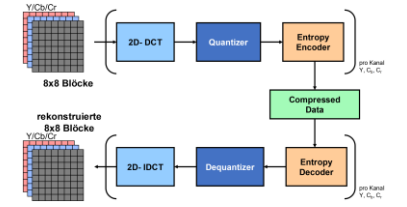
Das Auge hat weniger Rezeptoren für Farbe, darum können die Farbenen Cb und Cr mit geringerer Auflösung dargestellt werden. Der Schema-Indikator (J:a:b) gibt an, wie viele Chrominanzwerte pro J Luminanzwerte gespeichert werden:



Blockverarbeitung

Die einzelnen Kanäle werden in 8x8 Blöcke zusammengefasst und unabhängig voneinander komprimiert.

Abhängig vom gewählten Subsampling verändert sich die der Blöcke in den Farbkännen.



Diskrete Cosinus Transformation (DCT)

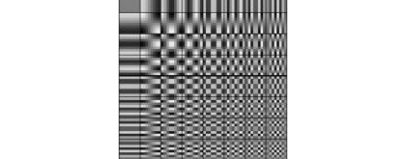
Transformation eines 8x8-Blocks $f(x, y)$ in den Frequenzbereich:

$$C(u, v) = \frac{1}{4} a(u) a(v) \sum_{x=0}^7 \sum_{y=0}^7 f(x, y) \cos \left(\frac{(2x+1)u\pi}{16} \right) \cos \left(\frac{(2y+1)v\pi}{16} \right)$$

$$f(x, y) = \frac{1}{4} \sum_{u=0}^7 \sum_{v=0}^7 a(u) a(v) C(u, v) \cos \left(\frac{(2x+1)u\pi}{16} \right) \cos \left(\frac{(2y+1)v\pi}{16} \right)$$

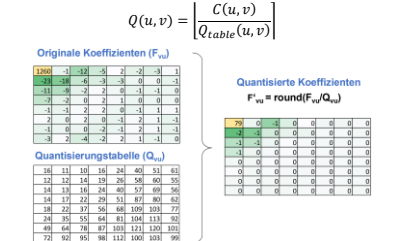
wobei

$$a(x) = \begin{cases} \frac{1}{\sqrt{2}} & \text{wenn } x = 0 \\ 1 & \text{sonst} \end{cases}$$



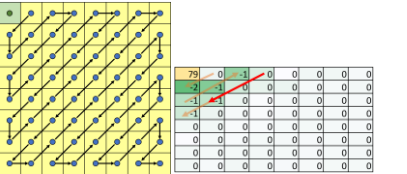
Quantisierung

Beim Quantisieren werden die DCT-Koeffizienten anhand einer Quantisierungstabelle dividiert und gerundet. So fallen unwichtige Koeffizienten weg.



Entropiekodierung

Run Length Encoding des quantisierten DCT-Musters mit Zick-Zack-Scanning von nacheinander folgenden Nullen plus nachfolgendem Koeffizienten. Wenn nur noch Nullen folgen, wird es mit einem End of Block Symbol (EOB) angegeben.



Beispiel: (79) (1,-2) (0,-1) (0,-1) (0,-1) (2,-1) (0,-1) (EOB)

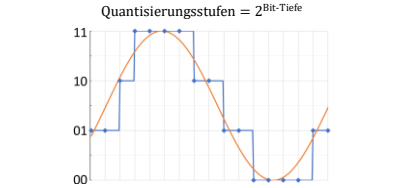
7. AUDIODEKODIERUNG
Abtasttheorem von Shannon

Ein Signal kann vollständig rekonstruiert werden, wenn es mit einer Frequenz abgetastet wird, die mindestens doppelt so hoch ist wie die höchste Frequenzkomponente des Signals. Sonst treten Alias-Effekte auf.

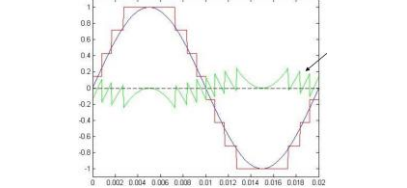
$$f_s \geq 2f_{max}$$

Quantisierung

Die kontinuierlichen Amplitudenwerte werden in diskrete Stufen unterteilt. Die Anzahl der Stufen bestimmt die Bit-Tiefe.

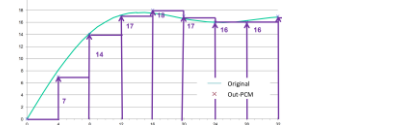


Das Quantisierungsrauschen ist die Differenz zwischen dem Originalsignal und dem quantisierten Signal.



Puls-Code-Modulation (PCM)

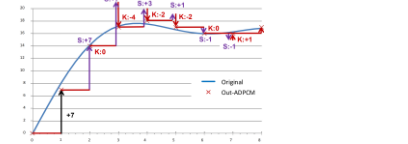
- Lineare PCM:** Die Amplitudenwerte werden linear in diskrete Stufen unterteilt und mit einer festen Bit-Tiefe kodiert.



- Differential PCM (DPCM):** Die Differenz zwischen aufeinanderfolgenden Abtastwerten wird kodiert.

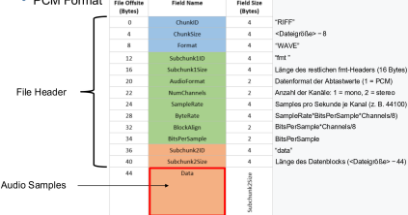


- Adaptive DPCM (ADPCM):** Die Quantisierungsstufen werden dynamisch angepasst basierend auf dem Signalverlauf. Kodiert wird nur der Korrekturwert K.

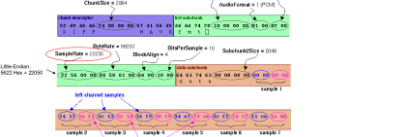


Wave File Format

Containerformat für Audiodaten, das PCM-kodierte Rohdaten speichert (keine Kompression). Es besteht aus einem Header und den eigentlichen Audiodaten.



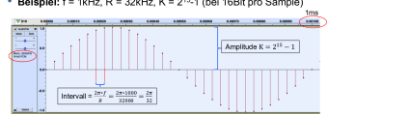
Beispiel:



Formeln***

Tonerzeugung eines reinen Sinustones
Die einzelnen Samples S_i für eine gewünschte Frequenz f kann in Abhängigkeit der Abtastrate R , und dem Skalierungsfaktor K berechnet werden.

$$S_i = K \cdot \sin \left(\frac{i \cdot 2\pi \cdot f}{R} \right)$$



Free Lossless Audio Codec (FLAC)

FLAC ist ein verlustfreies Audioformat, das Musik um 30-50% komprimiert. Es nutzt lineare Vorhersage und Golomb-Rice-Kodierung, um Audiodaten effizient zu speichern.

MPEG

MPEG nutzt die Menschliche Hörschwelle und den Maskierungseffekt zur verlustbehafteten Kompression von Audiodaten.

Schalldruckpegel (SPL)

Der Schalldruckpegel in Dezibel (dB) wird berechnet als:

$$L_p = 20 \cdot \log_{10} \left(\frac{p}{p_0} \right) \text{ [dB]}$$

wobei p der gemessene Schalldruck und p_0 der Referenzschalldruck (20 μ Pa) ist.

Eine Verdopplung der wahrgenommenen Lautstärke entspricht einer Erhöhung des Schalldruckpegels um etwa 6 dB.

$$20 \cdot \log_{10}(2) \approx 6.02 \text{ dB}$$

8. KANALCODIERUNG

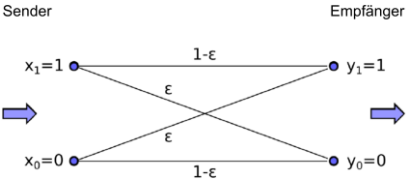
Bit Error Rate (BER)

Die Bitfehlerwahrscheinlichkeit ε gibt die Wahrscheinlichkeit an, dass ein einzelnes Bit während der Übertragung fehlerhaft empfangen wird.

Ein asymmetrischer Kanal hat unterschiedliche Fehlerwahrscheinlichkeiten für $\varepsilon_{0 \rightarrow 1}$ und $\varepsilon_{1 \rightarrow 0}$, abhängig davon, ob ein 0-Bit oder ein 1-Bit fehlerhaft übertragen wird.

Binary Symmetric Channel (BSC)

Die Bitfehlerwahrscheinlichkeit ε ist für beide Bitwerte gleich.



Fehlerwahrscheinlichkeit eines Blocks

Mit der BER ε kann man die Wahrscheinlichkeit $P_{0,N}$ ausrechnen, mit der eine Sequenz von N Datenbits korrekt (d.h. mit 0 Bitfehlern) übertragen wird:

Erfolgswahrscheinlichkeit: $P_{0,N} = (1 - \varepsilon)^N$

Fehlerwahrscheinlichkeit: $P_{fehler,N} = 1 - P_{0,N} = 1 - (1 - \varepsilon)^N$

Mehr-**Bit Fehlerwahrscheinlichkeit einer Sequenz**

Die Wahrscheinlichkeit, dass in einer Sequenz von N Bits **genau F Bitfehler** auftreten, ist gegeben durch:

$$P_{F,N} = \binom{N}{F} \varepsilon^F (1 - \varepsilon)^{N-F}$$

Für die Wahrscheinlichkeit, dass **höchstens F Bitfehler** auftreten, gilt:

$$P_{\leq F,N} = \sum_{i=0}^F \binom{N}{i} \varepsilon^i (1 - \varepsilon)^{N-i}$$

Für die Wahrscheinlichkeit, dass **mehr als F Bitfehler** auftreten, gilt:

$$P_{>F,N} = 1 - P_{\leq F,N} = \sum_{i=F+1}^N \binom{N}{i} \varepsilon^i (1 - \varepsilon)^{N-i}$$

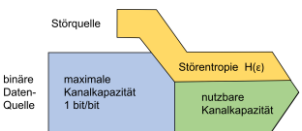
Kanalkapazität

Die Kanalkapazität C eines Binary Symmetric Channels (BSC) ist definiert als die maximale Übertragungsrate, bei der eine fehlerfreie Kommunikation möglich ist. Sie wird berechnet als:

$$C_{BSC} = 1 - H_b(\varepsilon)$$

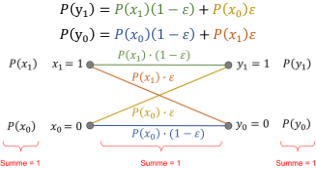
wobei $H_b(\varepsilon)$ die binäre Entropie-Funktion ist:

$$H_b(\varepsilon) = -\varepsilon \log_2(\varepsilon) - (1 - \varepsilon) \log_2(1 - \varepsilon)$$



Ein- und Ausgangswahrscheinlichkeiten

Die Ein- und Ausgangswahrscheinlichkeiten eines BSC sind wie folgt definiert:



Ein- und Ausgangsentropie***

Hamming-Distanz

Die **Hamming-Distanz** d zwischen zwei Codewörtern ist die Anzahl der Positionen, an denen sich die Codewörter unterscheiden.

Die **minimale Hamming-Distanz** d_{min} eines Codes ist die kleinste Hamming-Distanz zwischen allen möglichen Paaren von Codewörtern.

Ein Code heißt perfekt, wenn alle Codewörter die gleiche Hamming Distanz aufweisen.

Korrigierbare Bitfehler: $t = \left\lfloor \frac{d_{min}-1}{2} \right\rfloor$

Erkennbare Bitfehler: $d_{min} - 1$

Hamming-Gewicht

Das Hamming-Gewicht w eines Codeworts ist die Anzahl der Einsen in diesem Codewort.

Das minimale Hamming-Gewicht w_{min} eines Codes ist das kleinste Hamming-Gewicht aller möglichen Codewörter.

Die minimale Hamming-Distanz d_{min} zweier Codewörter ist gleich dem Hamming-Gewicht des XORs der beiden Codewörter:

$$d_{min}(c_1, c_2) = w(c_1 \oplus c_2)$$