Inhaltsverzeichnis

1 Logikgatter

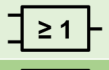
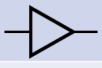
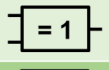
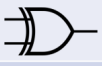
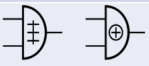
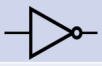
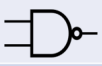
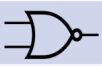
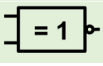
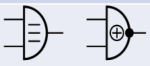
Function	Boolean Algebra (1)	IEC 60617-12 since 1997	US ANSI 91 1984	DIN 40700 until 1976
AND	A & B			
OR	A # B			
Buffer	A			
XOR	A $ B			
NOT	!A			
NAND	!(A & B)			
NOR	!(A # B)			
XNOR	!(A $ B)			

Abbildung 1: verschiedene Logikgatternotationen

2 Informationstheorie

Eine Information ist etwas, was vor seinem Eintreffen noch nicht bekannt war und kann viele Formen annehmen (Ton, Symbol, Text, Wert, ...). Information kann anhand der von ihr beseitigten Unsicherheit gemessen werden. Technisch gesehen ist die kleinste Masseinheit einer Information 1 Bit, sprich eine binäre Entscheidung.

2.1 Datenquellen

Eine Datenquelle wird als “diskret”bezeichnet, wenn sie abzählbare Messwerte liefert. Beispiele dafür sind digitale Sensoren, Ziehung der Lottozahlen, Wetterbericht im Radio. Datenqulle die “stetig” sind, liefern nicht abzählbare Messewerte. Beispiele dafür sind jegliche analoge Messewerte (regeln eines Potentiometers, ablesen eines analogen Thermometers).

Zudem kann eine Datenquelle “memoryless” sein, wenn die Messwerte statistisch unabhängig voneinander sind.

2.2 Formeln

Die Warscheinlichkeit einer Information wird berechnet, indem man die Vorkommnis der Information durch die gesamten Vorkommnisse teilt.

$$P(x) = \frac{k(x)}{K}$$

Der Informationsgehalt einer Information wird in Bit angegeben und wird mit

$$I(x) = \log_2 \frac{1}{P(x)}$$

berechnet

Die Entropie einer Datenquelle bezeichnet den durchschnittlichen Informationsgehalt aller Informationen die diese liefert. Die Masseinheit der Entropie ist $\frac{\text{Bit}}{\text{Symbol}}$

$$H(x) = \sum_{n=0}^{N-1} P(x_n) \log_2 \frac{1}{P(x_n)}$$

Die Mittlere Codelänge einer Datenquelle beschreibt die durchschnittliche Bitanzahl, die benötigt wird ein Zeichen der Quelle anzuzeigen.

$$L = \sum_{n=0}^{N-1} P(x_n) \cdot l_n$$

Die Redundanz einer Datenquelle bestimmt die Ineffizienz der Bitnutzung.

$$R = L(x) - H(x)$$

3 Quellencodierung

Das Ziel der Kompression ist, redundante und je nach Standard auch irrelevante Informationen zu minimieren um Ressourcen (Zeit, Energie, Bandbreite) zu sparen.

3.1 Huffman tree

Der Huffmantree bietet eine garantiert optimale Codierung. Das heisst eine Codierung mit der geringstmöglichen Redundanz. Dafür ordnet man alle Zeichen nach absteigender Probabilität und verbindet die kleinste Probabilität und die nächstgrössere zu einem Ast. Auf diesem addiert man die beiden Probabilitäten und ordnet den Ast neu ein. Diesen Vorgang wiederholt man, bis man an der Wurzel angekommen ist. Zur Überprüfung: die Summe aller Probabilitäten in der Wurzel muss 1 ergeben.

4 Lauflängenkodierung

Das Ziel der Lauflängenkodierung ist, Daten möglichst effizient zu speichern. Dabei kann z.B. “AAAAAABBCCCC” als “6A2B4C” dargestellt werden.

4.1 LZ77

Der Lempel-Ziv 77 Algorithmus ist ein verlustfreier, tokenbasierter Textkomprimierungsalgorithmus. Er verwendet einen Vorschaufbuffer und einen Suchbuffer. Dabei werden Muster im Vorschaufbuffer im Suchbuffer gesucht, um festzustellen, ob einen bestimmte Zeichenfolge bereits einmal vorgekommen ist. Dabei wird der beste “Match” gesucht. Dieser kann entweder gar nicht gefunden werden (das erste Symbol im Vorschaubuffer wurde noch nie gespeichert) oder einen Treffer von 1 Symbol bis zum gesamten Vorschaubuffer (Falls der komplette Vorschaubuffer so 1:1 bereits abgespeichert worden ist) finden. Ein LZ77-Token besteht aus (Offset—Länge—Symbol). Der Offset besagt, vor wievielen Zeichen der längste Treffer gefunden wurde (0 wenn kein Treffer). Die Länge besagt, wie lang der Treffer war und das Symbol besagt, welches das nächste Symbol nach dem Treffer des Vorschaubuffers war. Die Buffers können variable Längen haben, der Vorschaubuffer muss allerdings 1 Zeichen grösser sein als der Suchbuffer, da ansonsten bei einem Treffer im gesamten Suchbuffer kein weiteres Zeichen aus dem Vorschaubuffer nachgereicht werden könnte.

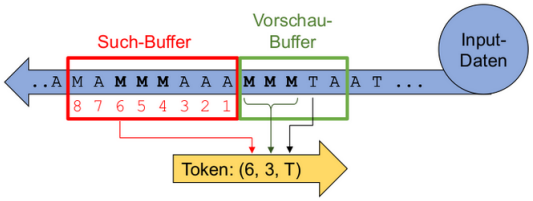


Abbildung 2: Illustration des LZ77 Algorithmus

4.2 LZW

Der Lempel-Ziv-Welch Algorithmus benutzt im Gegensatz zum LZ77 Algorithmus keine Buffers sondern arbeitet mit einem zur Laufzeit generierten Wörterbuch. Dafür wird jedes Zeichen beim ersten erscheinen mit dem dazugehörigen ASCII-Code (0-255) gespeichert, und als 256-ter Eintrag im Wörterbuch das Zeichen plus das nächstfolgende Zeichen gespeichert. Danach wird immer der beste Treffer gesucht, wobei das Wörterbuch mit jedem weitern Treffer um einen Eintrag wächst und die Einträge immer länger werden, womit bei zunehmender Textlänge und Wortwiederholungsrate die Kompression zunimmt. Um einen LZW-komprimierten Text wieder zu dekodieren wird der Algorithmus “rückwärts” ausgeführt. D.h. der erste Token wird aus dem ASCII-Table abgelesen und es

wird wieder ein neuer Eintrag an 256-ter Stell angelegt mit dem Zeichen und einem Platzhalter, der mit dem nächsten eingetragenen Zeichen gefüllt wird. So kann das Wörterbuch rekonstruiert werden, was uns erlaubt, es nicht abzuspeichern was noch mehr Speicherplatz spart.

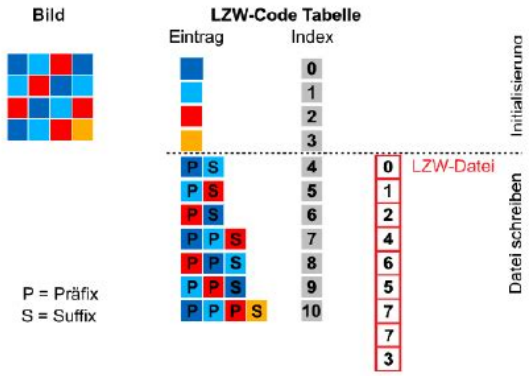


Abbildung 3: Illustration des LZW Algorithmus

5 Mediakompression

5.1 JPEG

Das JPEG-Kompressionsverfahren ist ein **verlustbehafteter** Kompressionsalgorithmus, welcher sich zur komprimierung natürlicher Bilder eignet. Der JPEG-Algorithmus umfasst 7 Schritte.

5.1.1 Farbraumtransformation

Unser Auge ist empflindlicher auf Helligkeitsunterschiede als auf Farbunterschiede. Das ermöglicht eine stärkere verlustbehaftete Kompression der Farben bevor wir markante Unterschiede feststellen können. Deshalb trennen wir die 3 Farbelemente aus dem RGB Farbraum auf in eine Helligkeitskomponente, einen Grün-Rot-Wert und einen Grün-Blau-Wert. Das das grüne Farbspektrum bleibt so stäker erhalten, da wir evolutionär bedingt empfindlicher auf Grünabstufungen sind als für andere Farben. Deshalb trennen wir die 3 Farbelemente aus dem RGB Farbraum auf in eine Helligkeitskomponente, einen Grün-Rot-Wert und einen Grün-Blau-Wert. Das das grüne Farbspektrum bleibt so stäker erhalten, da wir evolutionär bedingt empfindlicher auf Grünabstufungen sind als für andere Farben. Die Y-Komponente entspricht dem Graustufenbild des Ursprungbildes.

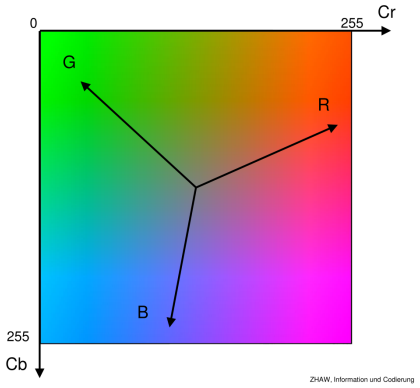


Abbildung 4: Verhältnis Cb zu Cr

5.1.2 Chrominanz Downsampling

Unter dem Chrominanz Downsampling versteht man das Zusammenfassen mehrere Pixel der Farbebenen. Hier werden das erste Mal irrelevante Informationen "eliminiert".

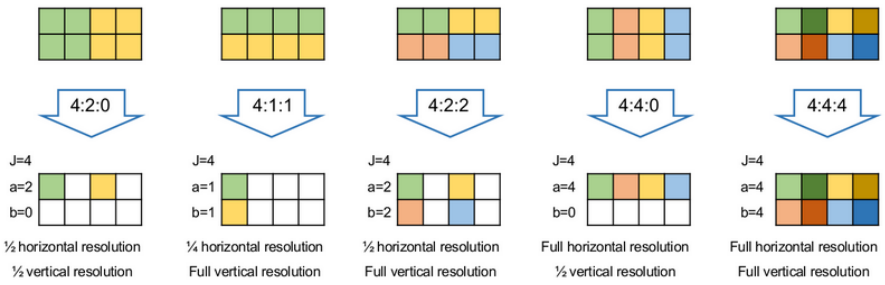


Abbildung 5: Chrominanz Downsampling

5.1.3 Pixel-Gruppierung

Die Idee der nächsten Schritte ist, kleine Pixelgruppierungen möglichst effizient abzuspeichern. Dafür wird das jeder Informationskanal in 8x8 Pixelblöcke aufgeteilt, die dann jeweils gemeinsam komprimiert werden. Falls ein Kanal nicht sauber in 8x8 Pixelblöcke unterteilt werden kann, wird bei den inkompleten Blöcken am Rand entweder die letzte Spalte oder Zeile dupliziert, bis der Block voll ist oder den fehlenden Blöcken wird ein fixer Wert 0 zugewiesen. Beide Optionen führen zu kleinen, kaum spürbaren Artefakten an den Rändern.

5.1.4 Diskrete Cosinustransformation

Dieser Schritt dient zur Übersetzung der Informationskanäle von Werten von 0-255 zu einer Kombination aus Cosinusfunktionen mit variiernder Frequenz und ist komplett verlustfrei. Die Pixelblöcke werden einer nach dem anderen zu einer gewichteten Addition der Cosinusfunktionen übersetzt, da die Koeffizienten weniger Speicherplatz benötigen als die Werte der einzelnen Pixel selbst.

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

Abbildung 6: 8x8 Pixelarray

1260	-1	-12	-5	2	-2	-3	1
-23	-18	-6	-3	-3	0	0	-1
-11	-9	-2	2	0	-1	-1	0
-7	-2	0	2	1	0	0	0
-1	-1	2	2	0	-1	1	1
2	0	2	0	-1	2	1	-1
-1	0	0	-2	-1	2	1	-1
-3	2	-4	-2	2	1	-1	0

Abbildung 7: 8x8 Frequenzarray

Es findet also bereits eine verlustfreie Kompression statt. Für die Kompression verwendet man die Forward DCT über dem 2 dimensionalen 8x8-Pixelblock Array und erhält daraus für jeden Pixelblock ein 2 dimensionales Frequenzarray indem die jeweiligen Koeffizienten festgehalten werden. Die Forward DCT sieht wie folgt aus:

$$F_{vu} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} B_{yx} \cos\left(\frac{(2x+1)u\pi}{16}\right) \cos\left(\frac{(2y+1)v\pi}{16}\right)$$

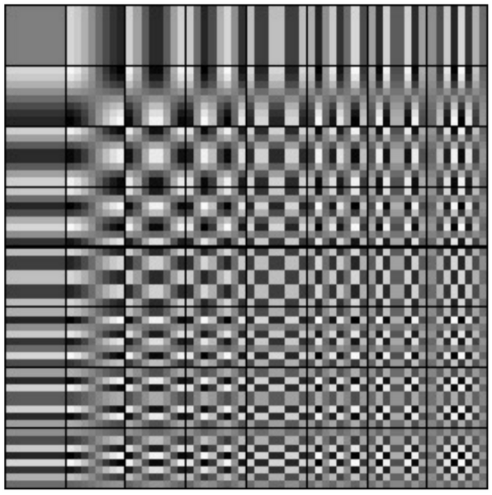


Abbildung 8: 8x8 Cosinusfunktionsmatrix

5.1.5 Quantisierung

Die Quantisierung ist der letzte verlustbehaftete Schritt im ganzen Kompressionsverfahren. Hierfür verwendet man je nach gewünschter Kompression eine tolerantere oder eine aggressivere Quantisierungstabelle. Eine Quantisierungstabelle ist eine Tabelle die analog zu den Pixelblöcken 8x8 Werte beinhaltet, wobei die Werte oben links tiefer sind und gegen unten und gegen rechts höher werden. Indem man alle Werte des Frequenzarrays durch den entsprechenden Quantisierungskoeffizienten teilt und das Resultat auf die nächste ganze Zahl rundet erhält man die Quantisierte Koeffiziententabelle. Dabei gehen feine Unterschiede, die in der ursprünglichen Frequenztabelle unten rechts waren verloren, da sie durch höhere Werte geteilt werden, welche dann auf 0 abgerundet werden. Die massgebenden Frequenzen bleiben dabei besser erhalten.

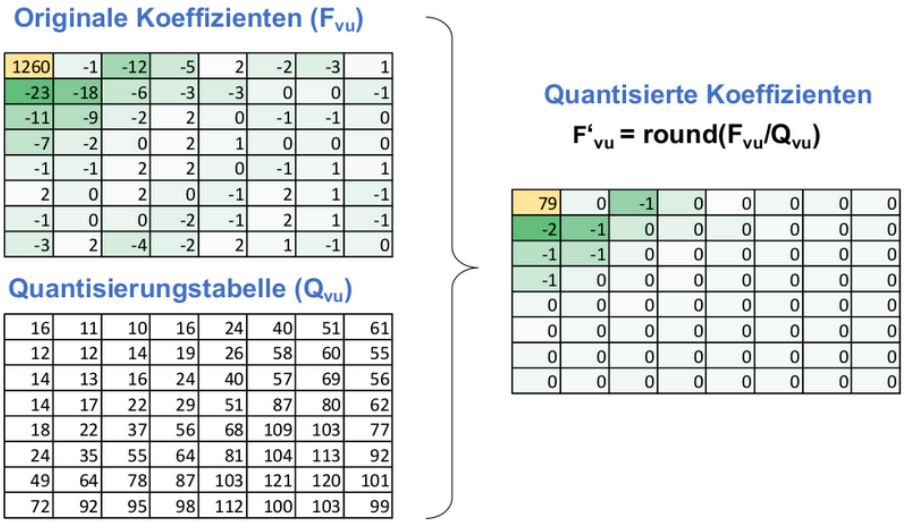


Abbildung 9: Quantisierung

5.1.6 Entropy-Coding

Die Entropiecodierung dient dem effizienten speichern der quantifizierten Koeffizententabelle. Da oben rechts tendenziell höhere Werte sind und unten linke mehr 0 Werte, wird die Matrix in einem Zick-Zack-Muster durchlaufen und diese Tokens mithilfe eines RLE-Verfahrens gespeichert. Das RLE-Verfahren ist im JPEG-Algorithmus nicht genau vorgegeben.

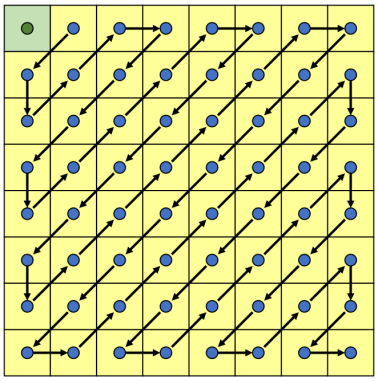


Abbildung 10: Zick-Zack durchlauf

5.1.7 In Datei verpacken

Alle Werte die benötigt werden, um das Bild wieder herzustellen werden nun in eine .jpg Datei gespeichert. Das beinhaltet die komprimierte Koeffizentenmatrix sowie die genutzte Quantisierungstabelle und die entsprechenden Anhaltspunkte zum genutzten RLE-Verfahren.

5.2 Audiokompression

Schallwellen sind analoge Werte, welche zur Speicherung oder Bearbeitung in digitale Werte umgewandelt werden müssen. Dafür "tastet" man die analoge Frequenz ab und arbeitet mit diesen Werten weiter.

5.2.1 Abtasten

Das Shannon (oder Shannon-Nyquist Theorem) besagt, dass die Abtastfrequenz für ein Signal mit höchster Frequenzkomponente ω grösser als 2ω sein muss,um das Signal garantiert verlustfrei abzubilden. Das menschliche Gehör kann Frequenzen bis zu ca. 22kHz verzeichnen, was jedoch von Gehör zu Gehör unterschiedlich ist. Deshalb wird in der Unterhaltung oft mit einer Abtastrate von 44.1kHz oder höher gearbeitet, da wir Frequenzen über 22kHz, die mit einer solchen Abtastrate verloren gehen, sowieso nicht hören würden.

5.2.2 Quantisierung

Mit der Quantisierung wird die Amplitude der Abtastpunkte festgehalten. Da das eingehende Signal stetig ist, wir jedoch nur mit diskreten Werten arbeiten können, geht hier unweigerlich Information verloren. Die Auflösung der Quantisierung bestimmt, wie viele Höhenabstufungen für die Amplitude der Frequenz gespeichert werden. Die Werte, welche nicht genau auf einer Höhenabstufung liegen, werden zur nächsten Abstufung auf- beziehungsweise abgerundet.

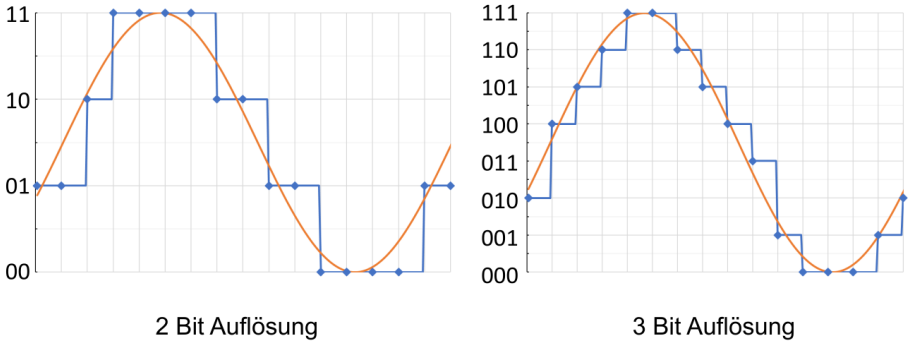


Abbildung 11: Quantisierung eines analogen Signals

Bei einer Umwandlung von stetigen zu diskreten Werten entsteht ein Rauschen, so auch bei der Konvertierung von analogen zu digitalen Signalen. Dieses Quantisierungsrauschen ist die Differenz des Rohsignals zu dem digitalen Signal. Daraus folgt, je feiner die diskreten Werte granuliert sind, desto kleiner ist auch das Rauschen. Die Lautstärke des Rauschens beträgt $6\text{dB} \cdot \text{Anzahl Bit}$

5.2.3 Pulsecode Modulierung

Die Pulsecode Modulierung fasst die abgetasteten und quantisierten Menge zusammen, mit dem Ziel, diese möglichst effizient zu speichern/transportieren. Dafür gibt es verschiedene Herangehensweisen. Gemessen wird die Effektivität der Modulierung durch die Bitrate (Bits/Sekunde). Dafür wird die Anzahl Abtastpunkte mit der Quantisierungstiefe multipliziert. Beispiel anhand des europischen Telefonstandards ITU_T G.711 (A-Law):

$$8000\text{Hz} \cdot 8\text{Bit} = 64\text{KBit/s}$$

Es gibt viele PCM Standards, von denen ein paar hier beschrieben werden.

Linear PCM Es wird für jeden Abtastpunkt die Amplitude numerisch festgehalten.

Differential PCM Es wird jeweils die Veränderung der Amplitude relativ zum letzten Abtastpunkt numerisch festgehalten.

Adaptive Differential PCM Es wird jeweils die Veränderung zur Veränderung der Amplitude relativ zum letzten Abtastpunkt numerisch festgehalten.

5.2.4 Verlustfreie Audiokompression

Wave und Flac bieten die Möglichkeit, PCM-Signale verlustfrei abzuspeichern. Wichtig ist dabei, dass nur bei der Speicherung des PCM-Signals keine weitern Information verloren gehen, das gespeicherte digitale PCM-Signal beinhaltet allerdings bereits einen gewissen Verlust gegenüber dem aufgenommenen Analogen Signal. Eine Wave Datei komprimiert das PCM-Signal nicht, wohingegen der Flac-Standard ein Verfahren ähnlich dem LZ-Algorithmus, wodurch eine verlustfreie Kompression möglich ist.

5.2.5 Verlustbehaftete Audiokompression

Die Verlustbehaftete Audiokompression 2 Faktoren aus. Zum einen ist unser Gehör nicht gleich empfindlich auf alle Frequenzen. Das erlaubt es uns, nicht/kaum hörbare Töne zu verwerfen und nur die relevanten Frequenzen abzuspeichern.

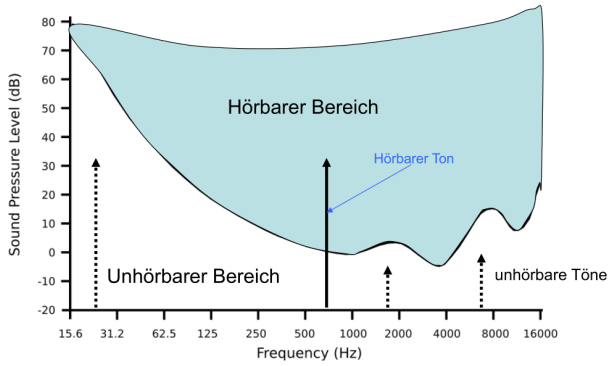


Abbildung 12: menschliche Hörschwelle über dem Frequenzband

Weiterer Sparbedarf besteht bei kleinen Amplituden in der Nähe von grösseren Amplituden. Unser Gehör passt sich lauten Schallereignissen an und blendet kurz vor und ein bisschen länger nach einem lauten Ton leisere Töne aus. Je höher die Amplitude eines lauten Ereignisses, desto länger bleibt unser Gehör auf diese Amplitude angepasst. Durch diesen Maskierungseffekt den hohe Amplituden auf kleinere Amplituden ausüben können wir weitere Informationen verwerfen, da diese von unserem Gehör schlechter, wenn überhaupt, wahrgenommen werden können. Dieser Maskierungseffekt wird genutzt, indem das Audiosignal in unterschiedliche Frequenzbänder unterteilt wird, die mit verschiedenen Bittiefen abgetastet werden. Dabei wird die Bittiefe möglichst klein gehalten, wobei das dadurch steigende Quantisierungsrauschen immernoch durch den Maskierungseffekt unter der Höhrschwelle gehalten wird.

5.2.6 Schalldruckpegel

Der Schalldruckpegel ist eine logarithmische Grösse in Dezibel zur Beschreibung der Stärke eines Schallereignises. Berechnet wird er Schalldruckpegel mit folgender Formel:

$$L = 20 \cdot \log_2(\frac{p}{p_0})$$

, wobei L den Schalldruck in Dezibel repräsentiert, p für den effektiven Schalldruck und p_0 den Bezugsschalldruck (die Hörschwelle beträgt $2 \cdot 10^{-5}$Pa). Eine Verdopplung des Schalldruckpegels entspricht einer Zunahme von ca. 6dB.

6 Kanalcodierung

Da Datenkanäle (=Datenschnittstellen) per se fehlerbehaftet sind, bedarf es einer Codierung der übertragenen Daten, welche den Empfänger erkennen lässt, ob die Übertragung fehlerfrei erfolgte (BEC) oder sogar eine Korrektur der entstandenen Fehler ermöglicht (FEC). Bei allen möglichen Verfahren wird die Rednundanz erhöht, da nicht mehr nur die Daten sondern auch ein Kontrollmechanismus übertragen werden muss. Beispiele für Kanalcodierungen sind positive oder Negative Quittungen, ein erweiterung der Nachricht durch eine Prüfsumme oder simple mehrfache Übertragung.

6.1 Bit Error Ratio

Die Bit Error Ratio, kurz BER, beschreibt die Warscheinlichkeit, dass in einem binären Kanal ein Bit während der Übertragung fälschlicherweise von 1 auf 0 oder von 0 auf 1 gewechselt wird.

$$\text{BER} = \varepsilon = \frac{\text{fehlerhafte Bits}}{\text{übertragene Bits}}$$

Bei einem symmetrischen Kanal gilt $\varepsilon_{1\to0} = \varepsilon_{0\to1}$, wohingegen die BER für einen asymmetrischen Kanal ungleich sein können. Die Erfolgswarscheinlichkeit

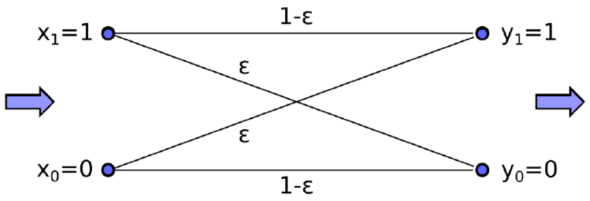


Abbildung 13: Fehlerwarscheinlichkeiten eines BSC

der Übertragung eines Frames A mit einer Anzahl Bit N beträgt

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

. Im Umkehrschluss beträgt die Fehlerwarscheinlichkeit $1-(1-\varepsilon)^N$. Die Warscheinlichkeit, dass in einer übertragenen Datensquenz mit N Bits genau F Fehler auftreten beträgt

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

Für die Warscheinlichtkeit bis zu einer bestimmten Anzahl Bitfehler n, summiert man diesen Term mit $F = 0, F = 1, ..., F = n$.

7 Fehlererkennung

8 Fehlerkorrektur