1 Zahlensysteme

Umrechnen von Dezimalzahlen in andere Zahlensysteme

Die Dezimalzahl 338 wird ins 5er-System umgewandelt:

- 338 : 5 = 67 Rest 3
- 67 : 5 = 13 Rest 2
- 13 : 5 = 2 Rest 3
- 2 : 5 = 0 Rest 2
- Rückwärts gelesen: 2323

Umrechnen von anderen Zahlensystemen in Dezimalzahlen

Die Zahl 20022 (3er-System) wird ins Dezimalsystem umgewandelt:

- $2 * 3^0 = 2$
- $2 * 3^1 = 6$
- $0 * 3^2 = 0$
- $0 * 3^3 = 0$
- $2 * 3^4 = 162$
- $2 + 6 + 0 + 0 + 162 = 170$

1.1 Negative Zahlen

1.1.1 Einerkomplement

- 1. Die Zahl -6 wird ins Dualsystem umgewandelt: $6 = 0110$
- 2. Das Einerkomplement wird gebildet, indem alle Bits invertiert werden: 1001
- 3. Das Ergebnis ist -6 im Einerkomplement: 1001

1.1.2 Zweierkomplement

Wertebereich z.B. 8 Bit: +127 bis -128, Asymentrie aufgrund der 0.

- 1. Subtraktion ist auch eine Addition mit einer negativen Zahl $2 - 6 = 2 + (-6) = -4$
- 2. Die Addition $2 + (-6)$ aufschreiben
- 3. Zahlen aus dem Dezimal- ins Dualsystem umschreiben. $2 = 0010; 6 = 0110$
- 4. Da wir mit einer negativen Zahl rechnen -6, müssen wir das Komplement (1001) bilden und mit 1 (0001) addieren, damit wir das sogenannte Zweierkomplement erhalten.
- 5. Addition vom Komplement und 1:

$$\begin{array}{r} 1001 \\ +0001 \\ \hline 1010 \end{array}$$

- 6. Addition mit der 2 und -6
  $2 + (-6)$:

$$\begin{array}{r} 0010 \\ 1010 \\ \hline 1100 \\ = -4 \end{array}$$

Kurzgesagt: Um ein Zweierkomplement zu bilden muss man invertieren und mit1 (0001) addieren.

2 Digitaltechnik

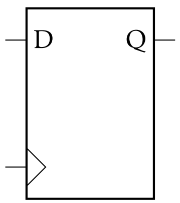
2.1 Operatoren

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

2.2 Flip-Flops

Flip-Flops sind Speicherelemente, die den Zustand speichern und bei einem Taktimpuls den Zustand ändern. Ein normaler "D-Flip-Flop"kann genau ein Bit Information speichern.

- Clock C
- Data D
- Ausgang Q



3 Informationstheorie

3.1 Typen von Datenquellen

3.1.1 Discrete Memoryless Source (DMS)

- Discrete heisst, dass die Quelle (zeitlich) einzelne Ereignisse liefert.
- Memoryless bedeutet, die Quelle erinnert sich beim Produzieren eines Ereignisses nicht an die Vorgeschichte. → Die Ereignisse sind (statistisch) unabhängig voneinander

3.1.2 Binary Memoryless Source (BMS)

- Bei dieser Quelle handelt es sich um eine DMS, die aber nur zwei verschiedene Ereignisse erzeugt.
- Ausgabe ist eine Folge von 0 und 1

3.2 Zweier-Logarithmus

$$x = log_2(K) = \frac{log_{10}(K)}{log_{10}(2)}$$

3.3 Gleiche Wahrscheinlichkeit $P(x_n)$

- Je mehr Fälle es gibt, desto seltener tritt ein bestimmtes Ereignis ein.
- Je seltener ein Ereignis ist, desto höher ist sein Informationsgehalt.
- N sei wieder die Anzahl der möglichen Ereignisse. Wenn alle Ereigniswerte x_n die Gleiche Auftretungswahrscheinlichkeit $P(x_n)$ haben, gilt:

$$P(x_n) = \frac{1}{N} \rightarrow N = \frac{1}{P(x_n)}$$

3.4 Informationsgehalt von Ereignissen $I(x_n)$

- Je seltener ein Ereignis eintritt, desto grösser ist der Informationsgehalt (Überraschungseffekt)
- Die folgende Formel gilt allgemein:

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

3.5 Entropie $H(X)$

Den mittleren Informationsgehalt von Quellen nennt man Entropie:

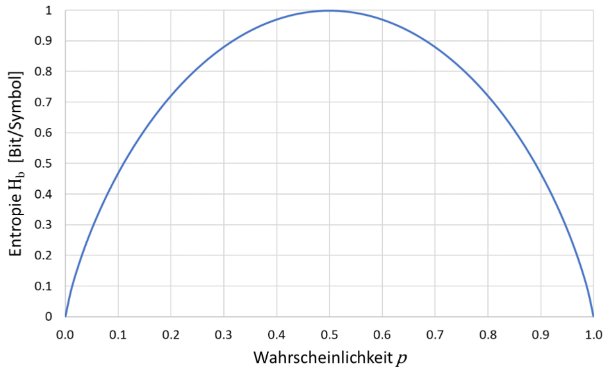
$$H(X) = \sum_{n=0}^{N-1} P(x_n) \cdot log_2(\frac{1}{P(x_n)}) = \sum_{n=0}^{N-1} P(x_n) \cdot I(x_n)$$

Die Masseinheit der Entropie ist $Bit/Symbol$.

3.5.1 Entropie Binary Memoryless Source

Eine BMS kennt nur zwei Symbole. Ist p die Auftretungswahrscheinlichkeit des eines Symbols, folgt dass $(1-p)$ jene des anderen Symbols ist.

$$H_b = p \cdot log_2(\frac{1}{p}) + (1-p) \cdot log_2(\frac{1}{1-p})$$



4 Quellencodierung

4.1 Redundanz

4.1.1 Codewortlänge ℓ_n

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

4.1.2 Mittlere Länge der Codierung L

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

4.1.3 Redundanz R

In $Bit/Symbol$:

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

4.1.4 Theorem zu Quellencodierung

- Falls $R > 0$, dann kann verlustfrei komprimiert werden.
- Falls $R \leq 0$, dann kann nur verlustbehaftet komprimiert werden.

4.2 Kompressionsrate

$$R = \frac{\text{Grösse komprimierte Daten}}{\text{Grösse Originaldaten}}$$

$$\text{Kompressionsrate } R \neq \text{Redundanz } R$$

4.3 Lauflängencodierung

Lauflängencodierung oder Run-Length Encoding (RLE) ist eine einfache Methode zur verlustfreien Datenkompression.

- Marker bestimmen, z.B. selten genutzes Zeichen.
- Marker und Anzahl der Wiederholungen speichern.

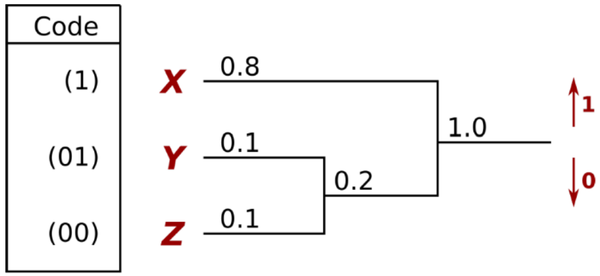
```
Hier verwenden wir als Marker z.B. Z:
Orginal: ASKEEEEEEEEEEEFEIIIIIPPPP ...
Codiert: ASKZ10EZ05IZ04P ...
```

4.4 Huffman-Codierung

Um Huffman-Codierung anzuwenden, muss die Wahrscheinlichkeit $P(x_n)$ der Symbole bekannt sein.

- Ordne alle Symbole nach aufsteigenden Auftretenswahrscheinlichkeiten auf einer Zeile. Dies sind die Blätter des Huffman-Baums.
- Notiere unter jedes Blatt seine Wahrscheinlichkeit.
- Schliesse die beiden Blätter mit der kleinsten Wahrscheinlichkeit an einer gemeinsamen Astgabel an und ordne dem Ast die Summe der Wahrscheinlichkeiten der beiden Blätter zu.
- Wiederhole den vorherigen Schritt mit Blättern und Ästen so lange, bis nur noch der Stamm des Baums übrig bleibt.
- Nun wird bei jeder Astgabel dem einen Zweig eine 0 und dem anderen eine 1 zugeordnet. (Die Zuordnung ist frei wählbar, muss aber über den ganzen Baum einheitlich sein).
- Nun werden auf dem Pfad vom Stamm zu jedem Blatt die Nullen und Einsen ausgelesen und von links nach rechts nebeneinander geschrieben. Dies sind die Huffman-Codeworte.

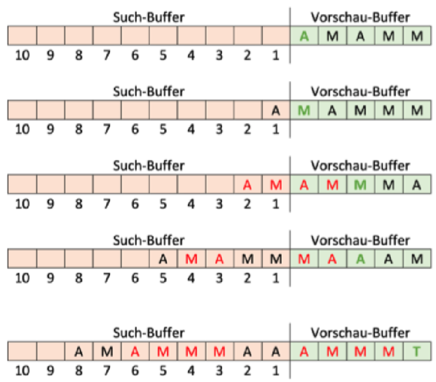
$P(X) = 0.80 \qquad P(Y) = 0.10 \qquad P(Z) = 0.10$



4.5 LZ77

Für LZ77 ein Suchbuffer n_s und Vorschaubuffer n_v definiert. Der Token hat das Format (Offset, Länge, Zeichen).

- Alle Zeichen werden durch Tokens fixer Länge ersetzt.
- Zur Bildung der Tokens werden die Daten durch ein Sliding Window, bestehend aus Such- und Vorschaubuffer, geleitet.
- Im Such-Buffer wird die längste Übereinstimmung mit dem Vorschau-Buffer als Token ausgegeben.
- Keine Übereinstimmung: Token $(0, 0, \text{Zeichen})$ wird verwendet.



Offset	Länge	Zeichen
0	0	A
0	0	M
2	2	M
4	2	A
6	4	T

4.5.1 Einzelne Tokengrösse

Grösse Such-Buffer + Grösse Vorschau-Buffer + Zusätzliches Zeichen

4.5.2 Gesamt

$$\text{Anzahl Tokens} \times \text{Tokengrösse}$$

4.5.3 Kompressionsrate

$$R = \frac{\text{LZ77 Codierung in Bits}}{\text{String ohne Codierung in Bits}}$$

Beim Token: Die Buffergrösse ist jeweils roundUp($\log_2(N)$), wobei N die Zahl ist, bis zu der wir maximal zählen können.
Beispiel Such-Buffer (1 bis 10): roundUp($\log_2(10)$) = 3.321... = 4 Bit

4.5.4 Decoding

- Start mit leerem Such-Buffer
- Bei Tokens ohne Referenz in den Such-Buffer einfach Zeichen ausgeben und in Such-Buffer schieben
- Bei Tokens mit Referenz in den Such-Buffer wird zunächst der referenzierte String ausgegeben und das Symbol im Token

4.6 LZW

4.6.1 Verfahren

- Basic vorinitialisiertes Wörterbuch (Bsp: Einzelsymbole von ASCII), Einzelne Zeichen sind immer im Wörterbuch vorhanden!
- String von links nach rechts lesen, bis der gelesene Teilstring nicht mehr im Wörterbuch vorkommt
- Token enthält nur den Index des schon bestehenden Eintrags im Wörterbuch, nicht aber das neue Zeichen

4.6.2 Decoding

Index	Eintrag	Output Token
0	(0)	
...	...	
65	A	
...	...	
77	M	
...	...	
84	T	
...	...	
255	(255)	
256	AM	65
257	MA	77

Index	Eintrag	Output Token
258	AMM	256
259	MM	77
260	MAA	257
261	AA	65
262	AMM_	258
263	_	
264		
265		
266		
267		
268		

Das Zeichen ganz rechts vom Eintrag wird erst bei dem nächsten Token übertragen (Überlappung)

5 Kanalcodierung

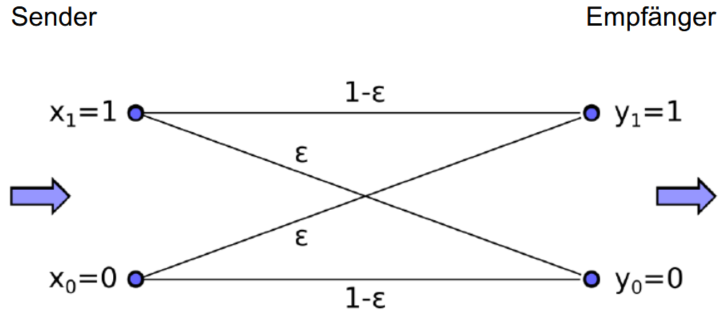
5.1 Bitfehlerwahrscheinlichkeit ε

ε ist die Wahrscheinlichkeit, dass ein Bitfehler auftritt (BER - Bit Error Rate).

- Alle Bits falsch: $BER = 1$
- Kein Bit falsch: $BER = 0$
- 1 von 2 Bits falsch: $BER = 0.5$
- 1 von 1000 Bits falsch: $BER = 0.001$

5.2 Binary Symmetric Channel (BSC)

Bei x_1 und x_0 kommen jeweils 0 oder 1 hinen. Die Wahrscheinlichkeit, dass ein Bitfehler auftritt, ist ε. Die Wahrscheinlichkeit dass kein Bitfehler auftritt, ist $1-\varepsilon$.



5.2.1 Erfolgswahrscheinlichkeit

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

5.2.2 Fehlerwahrscheinlichkeit

Auf N Datenbits:

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

Wobei für $N \cdot \varepsilon \ll 1$ folgende Näherung gilt: $1-(1-\varepsilon)^N \approx (1 - N \cdot \varepsilon)$

5.2.3 Mehr-Bit-Fehlerwahrscheinlichkeit

- $\binom{N}{F}$: Anzahl der Möglichkeiten, F Fehler in N Bits zu platzieren.
- ε^F: Wahrscheinlichkeit, dass F Fehler auftreten.
- $(1-\varepsilon)^{N-F}$: Wahrscheinlichkeit, dass Alle restlichen Bits $N-F$keinen Fehler haben.

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

$\binom{N}{F} = \frac{N!}{F! \cdot (N-K)!}$ bzw. $\binom{6}{2}$ rechnet man wie folgt:

$$\binom{6}{2} = \frac{6!}{2! \cdot (6-2)!} = \frac{6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}{2 \cdot 1 \cdot (4 \cdot 3 \cdot 2 \cdot 1)} = \frac{6 \cdot 5 \cdot \cancel{4 \cdot 3 \cdot 2 \cdot 1}}{2 \cdot 1 \cdot \cancel{(4 \cdot 3 \cdot 2 \cdot 1)}} = \frac{6 \cdot 5}{2 \cdot 1} = 15$$

5.2.4 Coderate R

- Die Coderate R ist das Verhältnis von Nutzdatenbits zu gesendeten Bits.
- K ist die Anzahl der Nutzdatenbits und N die Anzahl der gesendeten Bits.
- Z.B. $Paritaetsbits + Informationsbits = N$ und $K = Informationsbits$.

$$R = \frac{K}{N}$$

5.2.5 Kanalkapazität C

Die Kanalkapazität C in bit/bit ist die maximale Datenrate, die über einen Kanal übertragen werden kann.

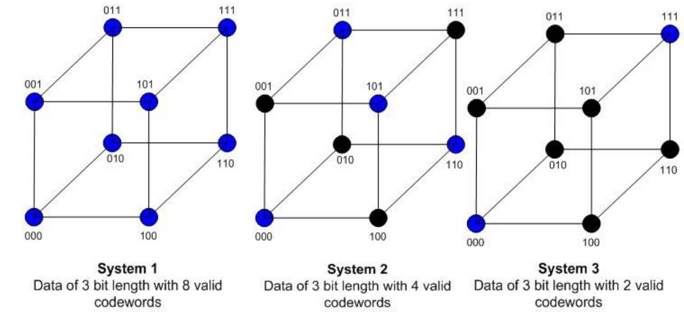
$$C_{BSC}(\varepsilon) = 1 - H_b(\varepsilon)$$

5.2.6 Kanalkodierungstheorem

Möchte man die Restfehlerwahrscheinlichkeit eines Fehlerschutzcodes beliebig klein machen, so muss $R < C$ sein.

5.3 Hamming-Distanz

Die Hamming-Distanz d_H ist die Anzahl der unterschiedlichen Bits zwischen zwei Codewörtern.



5.3.1 Eigenschaften

- **Systematisch**: Ein Code ist Systematisch wenn die Nutzdatenbits unverändert im Codewort übernommen werden. Hierfür müssen legendlich z.B. die Paritätsbits entfernt werden.
- **Linear**: Ein Code ist linear wenn jede Kombination ($EXOR$) von Codewörtern wieder ein Codewort ist.
- **Zyklisch**: Ein zyklischer Code ist eine spezielle Art eines linearen Codes, bei dem jede zyklische Verschiebung eines Codeworts ebenfalls ein gültiges Codewort ist.
- **Perfekt**: Ein Code heisst ein ńperfekter Codeż, wenn jedes empfangene Wort w genau ein Codewort c hat, zu dem es eine geringste Hamming-Distanz hat und zu dem es eindeutig zugeordnet werden kann.

5.4 Paritätscheck

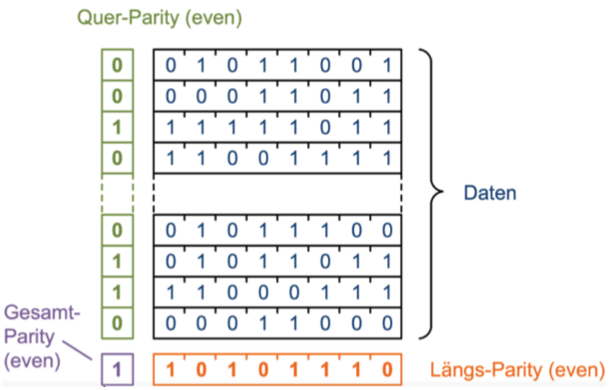
5.4.1 1D Paritätscheck

Ein Parity-Bit bestimmt, ob die Anzahl Einer in einem Codewort gerade oder ungerade ist. Even und Odd Parity sind gleichwertig, wobei nur even parity lineare Codes ermöglicht.

5.4.2 2D Paritätscheck

Ein 2D Paritätscheck...

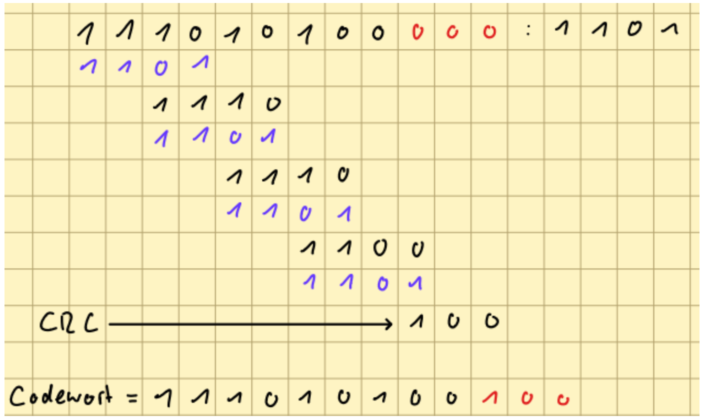
- ...ist ein Paritätscheck, der auf zwei Dimensionen angewendet wird.
- ...kann einen Fehler in einer Zeile oder Spalte erkennen und korrigieren.
- ...kann zwei Fehler erkennen, aber nicht korrigieren.
- ...kann drei Fehler nicht erkennen.



5.5 CRC

5.5.1 CRC-Polynomdivision

- Nutzdatenwort: 111010100
- Generatorpolynom: $x^3 + x^2 + 1 \Rightarrow 1101$
- Nutzdatenwort mit Nullen (Grad des Generatorpolynoms [3]) erweitern: 111010100000
- Rest bestimmen durch Division: 100
- Rest an Nutzdatenwort anhängen: 111010100100

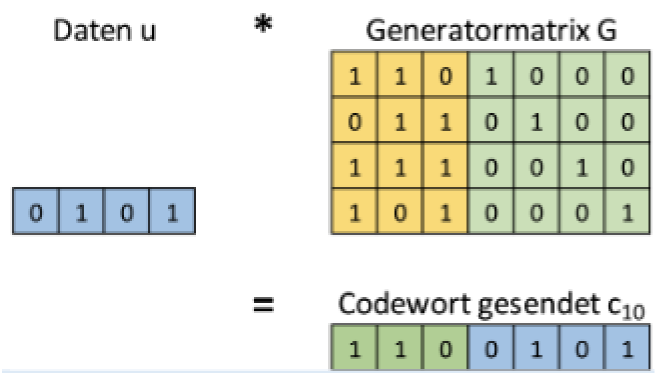


5.6 Blockcodes

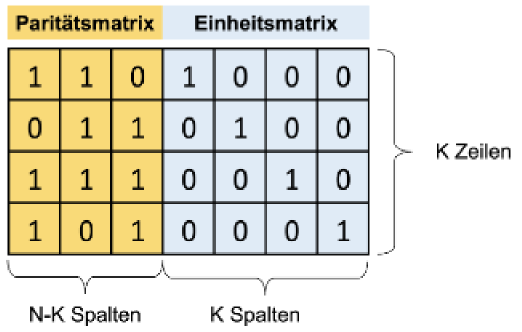
5.6.1 Eigenschaften

- Lineaer → Jede Kombination von Codewörtern ist wieder ein Codewort.
- Systematisch → Nutzdatenbits sind unverändert im Codewort enthalten.

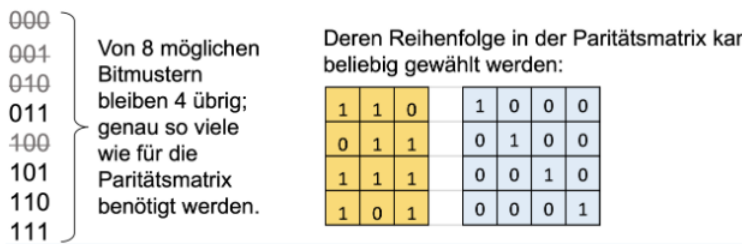
Lineare Blockcodes werden über eine Generatormatrix definiert. Das Codewort entsteht indem die Daten mit der Generatormatrix multipliziert werden:



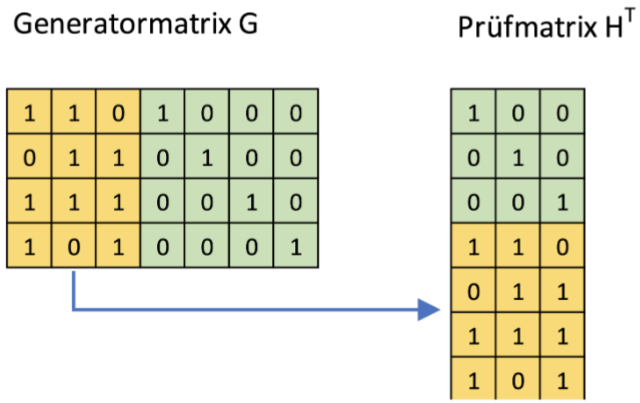
5.6.2 Bildung Generatormatrix



Die Paritaetsbits (Zeilen) müssen voneinander linear unabhängig sein. Für Hamming-Codes gilt: Jede Zeile hat gleich viele Einsen wie d_{min}.



5.6.3 Bildung Prüfmatrix



Generatormatrix G

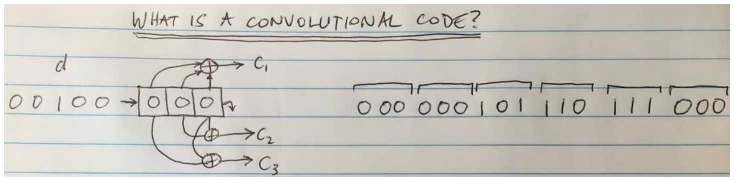
$$
G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}
\rightarrow
H^T = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}
$$

Prüfmatrix H^T

5.6.4 Fehlererkennung

Distance d	Visualization	# errors detected	# errors corrected
1	c_1—c_2	0	0
2	c_1—○—c_2	1	0
3	c_1—○—○—c_2	2	1
4	c_1—○—○—○—c_2	3	1
5	c_1—○—○—○—○—c_2	4	2
6	c_1—○—○—○—○—○—c_2	5	2
d		$d-1$	$\left\lfloor \frac{d-1}{2} \right\rfloor$

5.7 Faltungscodes



5.7.1 Trellis Diagramm

6 JPEG

1. Y Cr Cb Conversion
2. 8x8 Pixel Blocks
3. Discrete Cosine Transformation (DCT)
4. Quantization
5. Zig-Zag Scanning
6. DC and AC Seperation
7. Run-Length Encoding
8. Huffman Encoding
9. JFIF File Creation

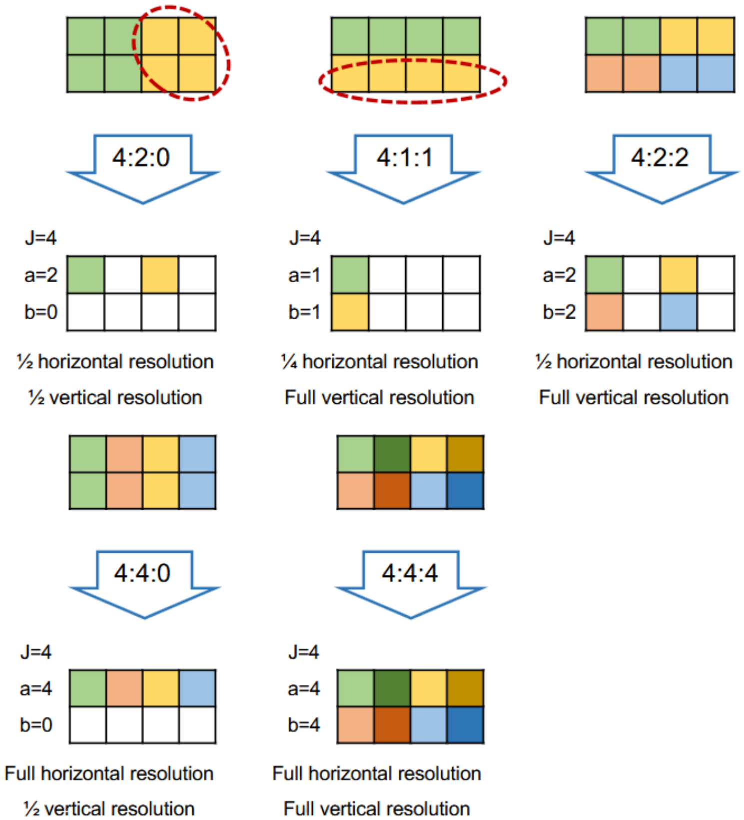
6.1 y C_r C_b Conversion

Das Bild wird in Lumineszenz (Y) und Chrominanz (C_r, C_b) aufgeteilt. C_b ist der Blauanteil und C_r der Rotanteil.

6.1.1 Subsampling

$$R = \frac{\text{Resultierende Pixel}}{\text{Ursprüngliche Pixel}}$$

- Subsampling meint, dass in beiden Chrominanz-Ebenen in der Horizontalen oder Vertikalen mehrere Pixel zusammengefasst werden.
- Der Schema-Indikator gibt die Art des Subsamplings an und hat die Form J:a:b (z.B. 4:2:0)
- Diese Notation basiert auf einem Referenzbildblock, der J Pixel breit und 2 Pixel hoch ist. Üblich ist J = 4.



6.2 8x8 Pixel Blocks

Das Bild wird in 8x8 Pixel Blöcke aufgeteilt. Diese Blöcke werden dann einzeln bearbeitet.

6.3 Discrete Cosine Transformation (DCT)

Jeder wert der 8x8 Matrix wird durch die DCT in einen Frequenzraum transformiert.

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

Vor DCT:

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

Nach DCT (Q_{vu}):

- **grosser DC-Wert (Mass für Mittelwert)** (Hoch = Weiss, Tief = Schwarz)
- **wenig tieffrequente grössere AC-Werte**
- **viele kleine AC-Werte (vernachlässigbar!)**

1260	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	-0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

6.4 Quantization

Die Frequenzanteile von der DCT werden nun durch eine Quantizationstabelle (Q_{vu}) geteilt. Die Quantizationstabelle wird je nach JPEG Verfahren anders gewählt. Quantizationstabelle für Luminanz:

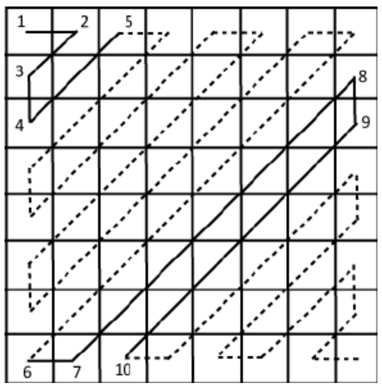
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98				

Nun nimmt man die 8x8 Tabelle von nach der DCT und teilt sie durch die Quantizationstabelle bzw. Folgende Funktion:

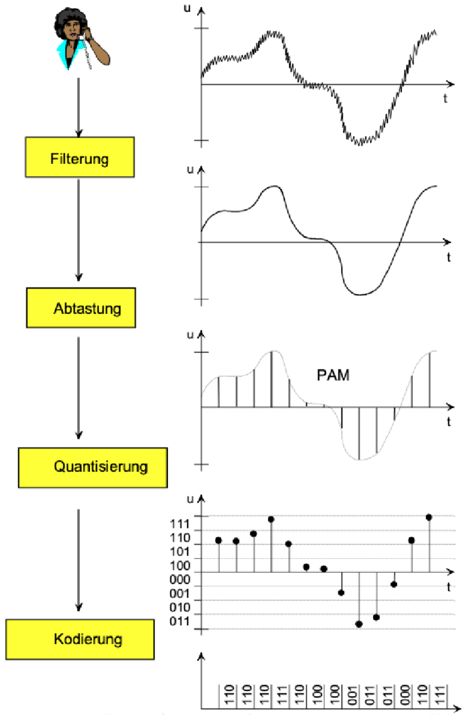
$$F_{vu} = round(F_{vu}/Q_{vu})$$

79	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

6.5 Zig-Zag Scanning



7 Audiocodierung

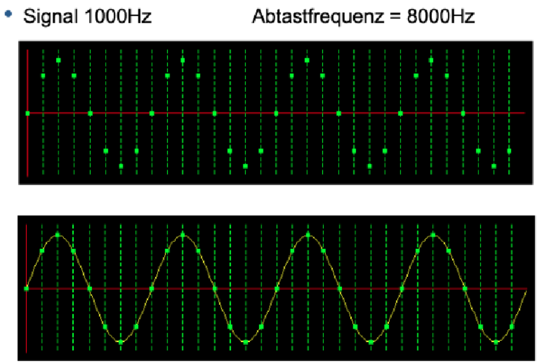


7.1 Filterung

Mit Hilfe eines Filters werden die zu hohen und die zu tiefen Frequenzen entfernt. Das Signal wird also auf einen hörbaren Frequenzbereich (typischerweise 20-20kHz) begrenzt.

7.2 Abtastung

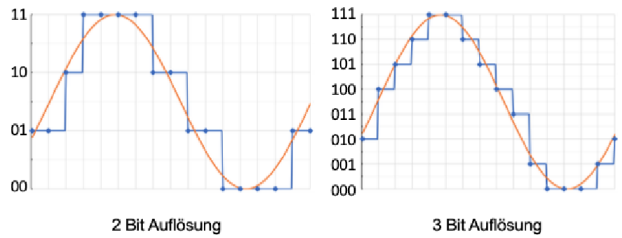
Abtastfrequenz muss mindestens doppelt so gross sein, wie die höchste im analogen Signal vorkommende Frequenz (siehe Abtasttheorem)



7.3 Quantisierung

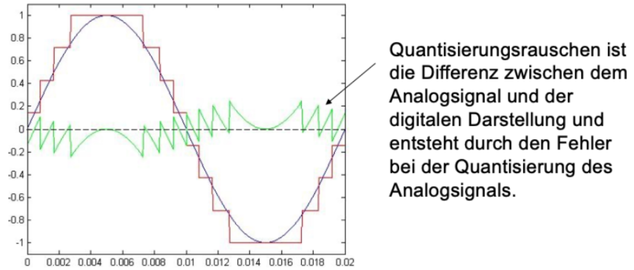
Anzahl Bit, die für die Quantisierung verwendet werden, bestimmt die Anzahl Stufen, welche für die Messung der Amplitude zur Verfügung stehen.

Die gemessenen Werte werden der nächsten Stufe zugeordnet (gerundet). Dabei entsteht ein sogenannter **Quantisierungsfehler**.



Quantisierungsrauschen: Differenz Quantisierung und dem Signal

Das **Quantisierungsrauschen** wird kleiner bei grösserer Anzahl verwendeter Bits.



- **Mit jedem Bitanstieg** nimmt das Quantisierungsrauschen um **6dB** ab.
- Jede Bit-Zunahme bedeutet eine **Verdoppelung** der Anzahl Stufen und damit eine Halbierung des Quantisierungsrauschens.

7.4 Codierung

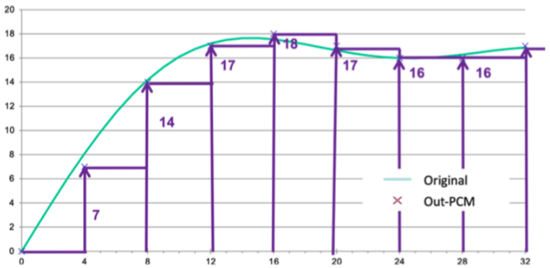
Jedem quantisierten Messwert wird ein Wert von einer bestimmten Bitlänge zugeordnet. Typischerweise werden 16Bit zur Aufzeichnung von Audio in hoher Qualität verwendet. (Professionell: 24Bit)

Aufgrund der Abtastrate entsteht somit ein Bitstrom, der berechnet werden kann:

$$\text{Anzahl Bit pro Messwert (Sample)} \times \text{Abtastfrequenz}$$

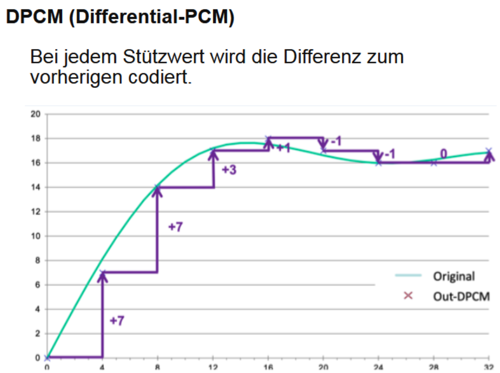
7.4.1 PCM (linear Quantisiert)

Bei jedem Stützwert wird ein absoluter Wert codiert und bei jedem Wert, wird die volle Anzahl Bits gebraucht.



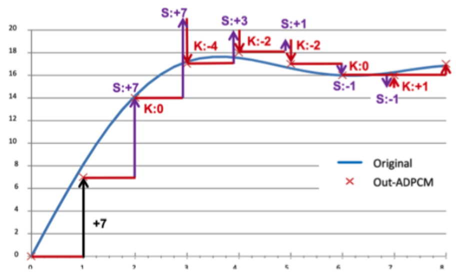
7.4.2 DPCM (Differential-PCM)

Bei jedem Stützwert wird die Differenz zum vorherigen codiert.



7.4.3 ADPCM (Adaptive-Differential-PCM)

Nur Korrekturwert K wird codiert (Korrektur des vorherigen Korrekturwertes).

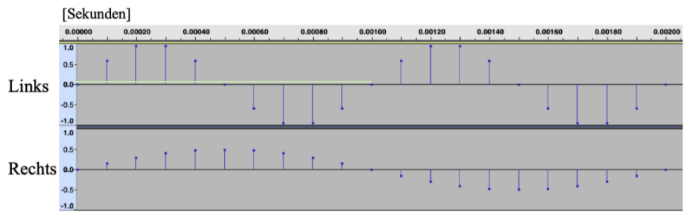


7.5 Abtastheorem

Die Abtastfrequenz muss mindestens doppelt so gross sein, wie die höchste im analogen Signal vorkommenden Frequenz.

$$f_{\text{abtast}} > 2 \times f_{\text{max}}$$

7.6 Beispiel



7.6.1 Abtastfrequenz finden

Samples (blaue Punkte) für eine Periode T zählen (hier 10).

$$f = 1/T = 1/0.01s = 100Hz$$

$$f_{\text{abtast}} = (1/T) \times \text{n-samples} = 1000Hz \times 10 = 10000Hz$$

7.6.2 dB Unterschied

Halbierung der Lautstärke/Amplitude entspricht einer Reduktion von 6dB.

7.6.3 Filesize

16-Bit-Tiefe = 2 Byte/Sample

Filesize = (Abtastfrequenz × Länge + 1) × Byte/Sample × Anzahl Kanäle + Header

Sollte ca. das gleiche sein wie:

$$\text{Filesize} = \frac{\text{Abtastfrequenz} \times \text{Länge} \times \text{Auflösung} \times \text{Anzahl Kanäle} + \text{Header}}{8}$$