

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:

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

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, Asymetrie 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:

1001

+0001

-----

1010

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

0010

1010

-----

























1100

= -4

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

2 Digitaltechnik

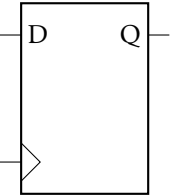
2.1 Operatoren

Function	Boolean Algebra <sup>(1)</sup>	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)			

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:

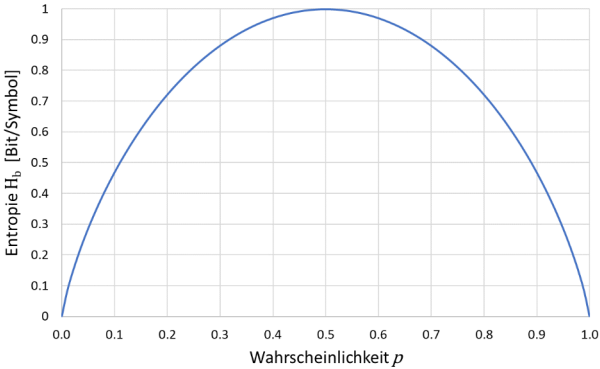
$$H(X) = \sum_{n=0}^{N-1} P(x_n) \cdot \log_2(\frac{1}{P(x_n)}) = \sum_{n=0}^{N-1} P(x_n) \cdot I(x_n)$$

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  $\ell_n$

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

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

Kompressionsrate  $R \neq$  Redundanz  $R$

4.3 Lauflängencodierung

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

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

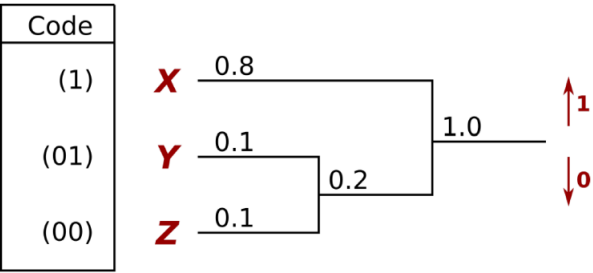
Hier verwenden wir als Marker z.B. Z:  
Original: ASKEEEEEEEFEIIIIIPPPP ...  
Codiert: ASKZ10EZ05IZ04P ...

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.

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



4.5 LZ77

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

Suchbuffer ( $n_s$ )	Vorschaubuffer ( $n_v$ )	Restdaten	Token
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

4.6 LZW

5 Kanalcodierung

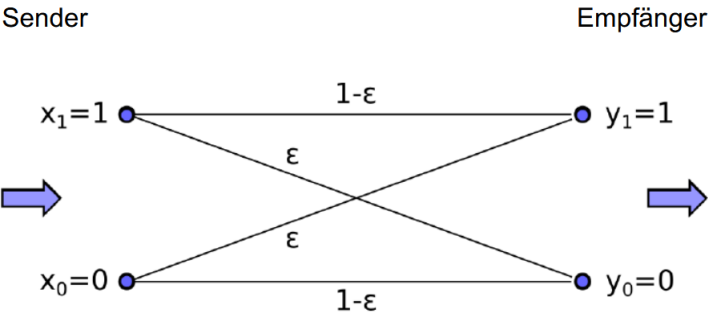
5.1 Bitfehlerwahrscheinlichkeit  $\varepsilon$

$\varepsilon$  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  $\varepsilon$ . 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.
- $\epsilon^F$ : Wahrscheinlichkeit, dass  $F$  Fehler auftreten.
- $(1 - \epsilon)^{N-F}$ : Wahrscheinlichkeit, dass Alle restlichen Bits  $N - F$  keinen Fehler haben.

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

$$\binom{6}{2} = \frac{6!}{2! \cdot (6-2)!} \text{ bzw. } \binom{6}{2} \text{ rechnet man wie folgt:}$$
$$\binom{6}{2} = \frac{6!}{2! \cdot (6-2)!} = \frac{6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}{2 \cdot 1 \cdot (4 \cdot 3 \cdot 2 \cdot 1)} = \frac{6 \cdot 5 \cdot \cancel{4} \cdot \cancel{3} \cdot \cancel{2} \cdot \cancel{1}}{2 \cdot 1 \cdot (\cancel{4} \cdot \cancel{3} \cdot \cancel{2} \cdot \cancel{1})} = \frac{6 \cdot 5}{2 \cdot 1} = 15$$

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.  $Paritätsbits + Informationsbits = N$  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.

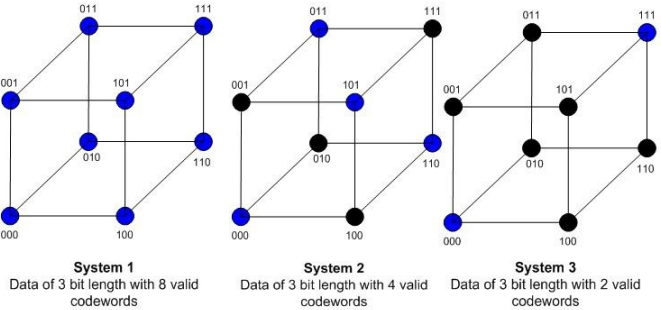
$$C_{BSC}(\epsilon) = 1 - H_b(\epsilon)$$

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 *n*perfekter Codez, 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

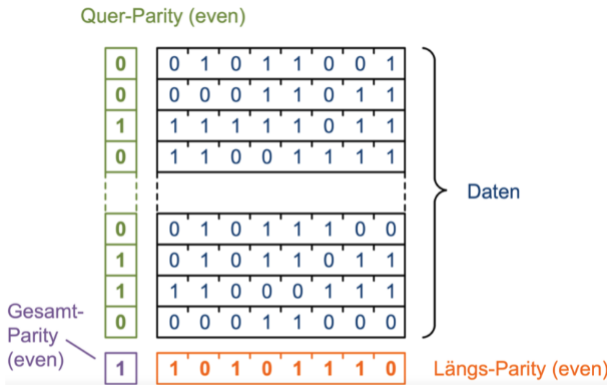
5.4.1 1D Paritätscheck

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.

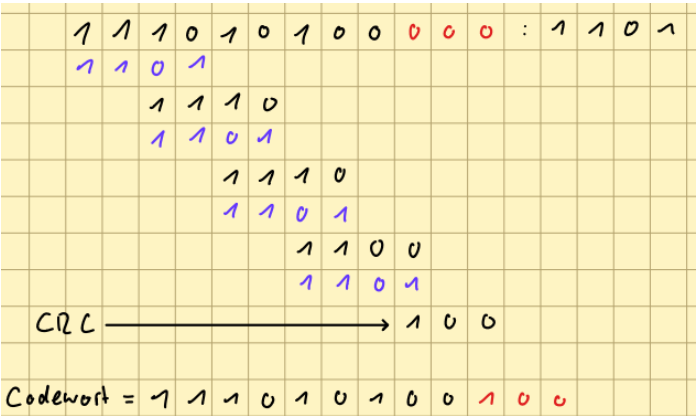


5.5 CRC

5.5.1 CRC-Polynomdivision

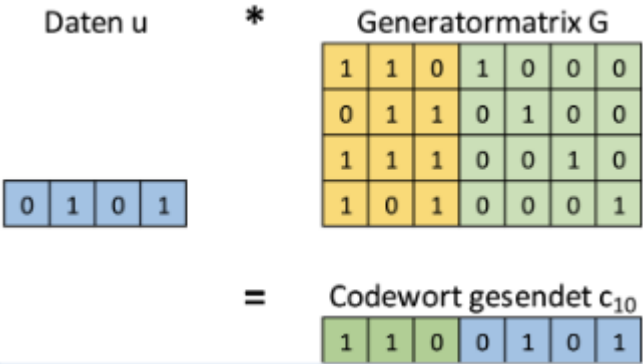
- Nutzdatenwort: 111010100

- 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

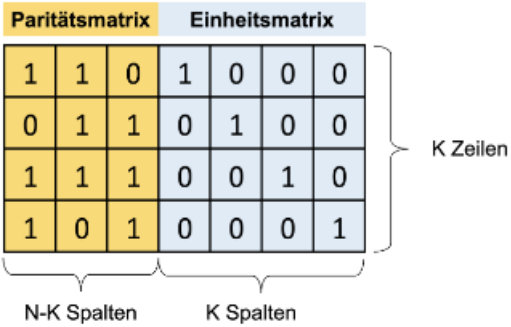


5.6 Blockcodes

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



5.6.1 Bildung Generatormatrix



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

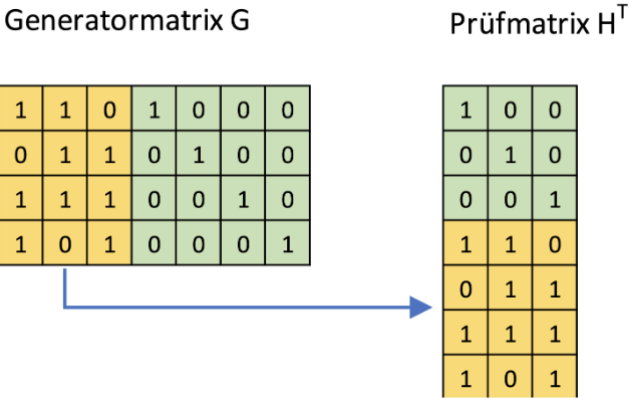
000  
001  
010  
011  
100  
101  
110  
111

Von 8 möglichen Bitmustern bleiben 4 übrig; genau so viele wie für die Paritätsmatrix benötigt werden.

Deren Reihenfolge in der Paritätsmatrix kann beliebig gewählt werden:

1	1	0
0	1	1
1	1	1
1	0	1

5.6.2 Bildung Prüfmatrix



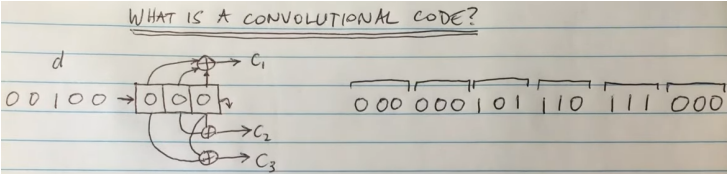
Generatormatrix G

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

Prüfmatrix H<sup>T</sup>

1	1	0
0	1	1
1	1	1
1	0	1
1	0	0
0	1	0
0	0	1

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 Separation
- 7. Run-Length Encoding
- 8. Huffman Encoding
- 9. JFIF File Creation

6.1 y Cr Cb Conversion

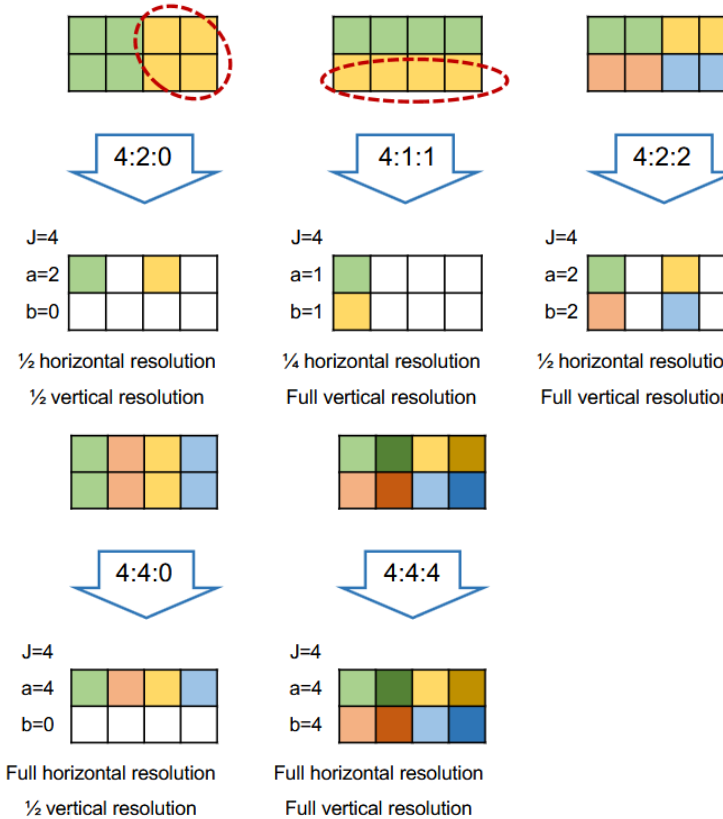
Das Bild wird in Lumineszenz (Y) und Chrominanz (C<sub>r</sub>, C<sub>b</sub>) aufgeteilt. C<sub>b</sub> ist der Blauanteil und C<sub>r</sub> der Rotanteil.

6.1.1 Subsampling

R = Resultierende Pixel / 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) = 1/4 \* C(u) \* C(v) \* sum\_{x=0}^7 sum\_{y=0}^7 f(x,y) \* cos((2x+1)u\*pi/16) \* cos((2y+1)v\*pi/16)

Vor DCT:

