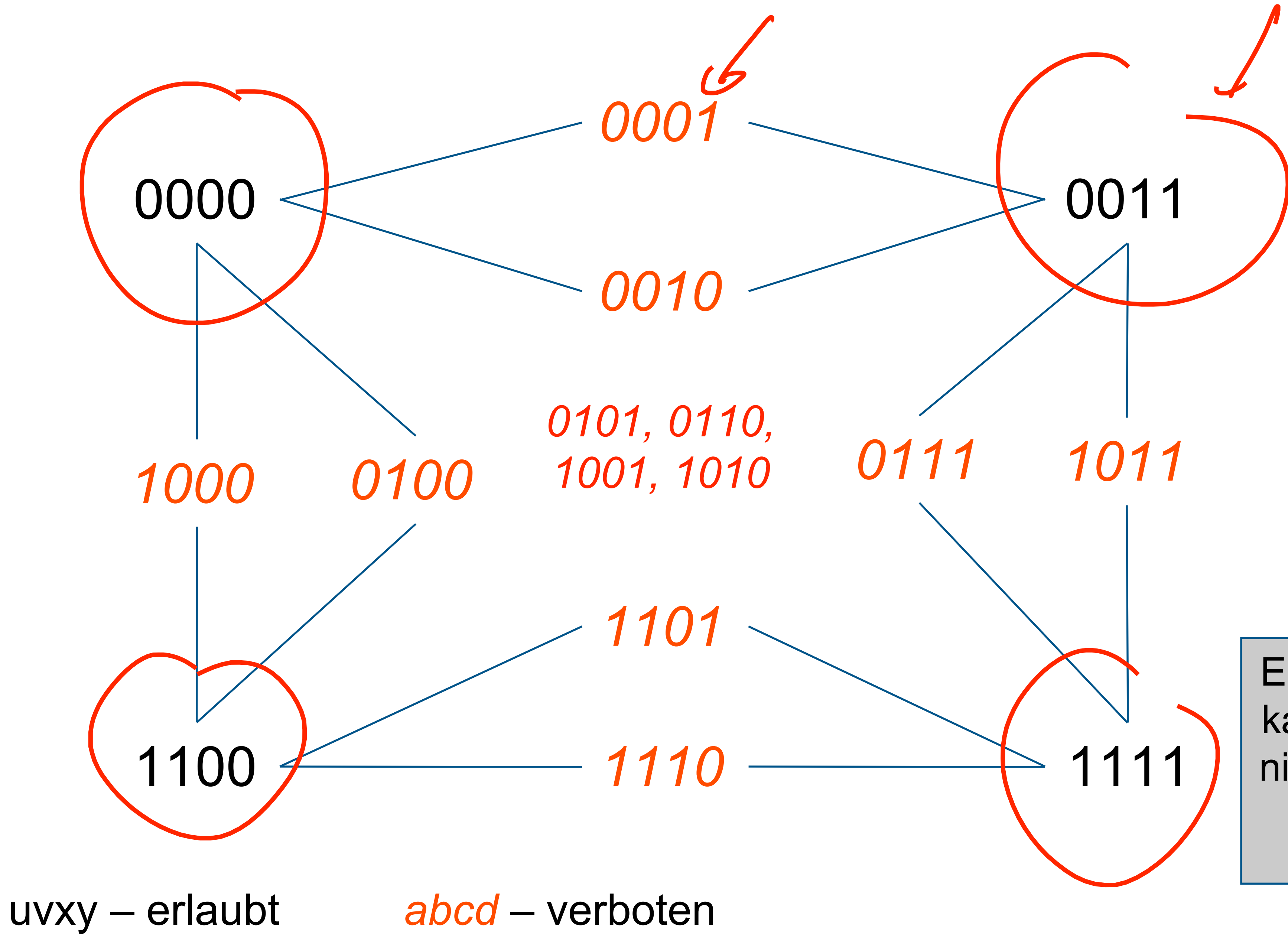
Systeme II
3. Die Datensicherungsschicht

Christian Schindelhauer
Technische Fakultät
Rechnernetze und Telematik
Albert-Ludwigs-Universität Freiburg
Version 14.05.2013

Veränderung der Frames durch Bitfehler

- Angenommen die folgenden Frames sind erlaubt:
  0000, 0011, 1100, 1111



Kanten verbinden Frames, die sich nur in einem Bit unterscheiden

Ein einfacher Bitfehler kann erlaubte Frames nicht in einen anderen erlaubten Frame umformen!

Hamming-Distanz

0000 0110
1010 0000
1 1 11 4

- Der “Abstand” der erlaubten Nachrichten zueinander war immer zwei Bits
- Definition: Hamming-Distanz
  - Seien $x = x_1, \ldots, x_n$ und $y = y_1, \ldots, y_n$ Nachrichten
  - Dann sei d(x,y) = die Anzahl der 1er Bits in x XOR y
- Intuitiver: die Anzahl der Positionen, in denen sich x und y unterscheiden

Hamming-Distanz

- Die Hamming-Distanz ist eine Metrik
  - Symmetrie
    - $d(x,y) = d(y,x)$
  - Dreiecksungleichung:
    - $d(x,y) \leq d(x,z) + d(z,y)$
  - Identität
    - $d(x,x) = 0$ und
      $d(x,y) = 0$ gdw. $x = y$
- Beispiel:
  - x= 0011010111
  - y= 0110100101
  - x XOR y= 0101110010
  - $d(x,y) = 5$

Hamming-Distanz von Nachrichtenmengen

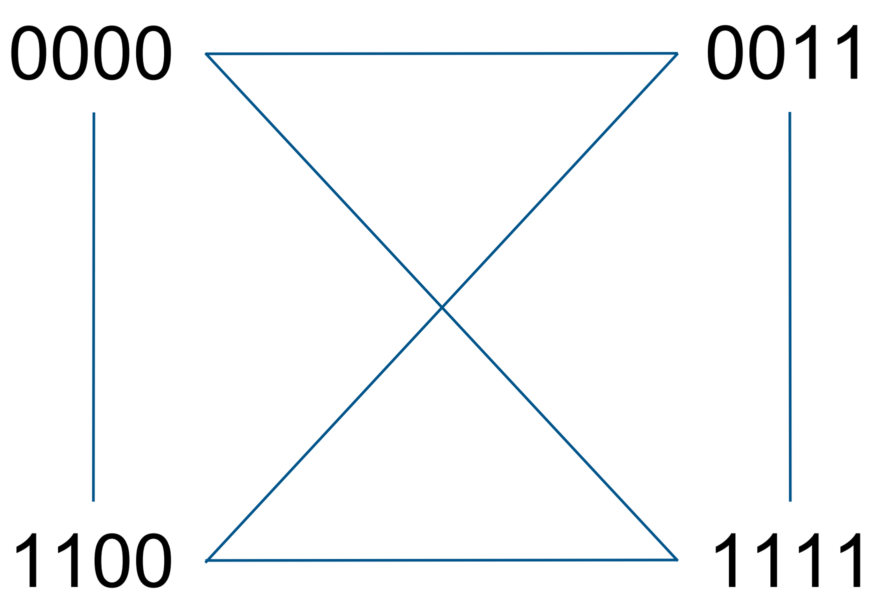
000001
000000

- Die Hamming-Distanz einer Menge von (gleich langen) Bit-Strings S ist:

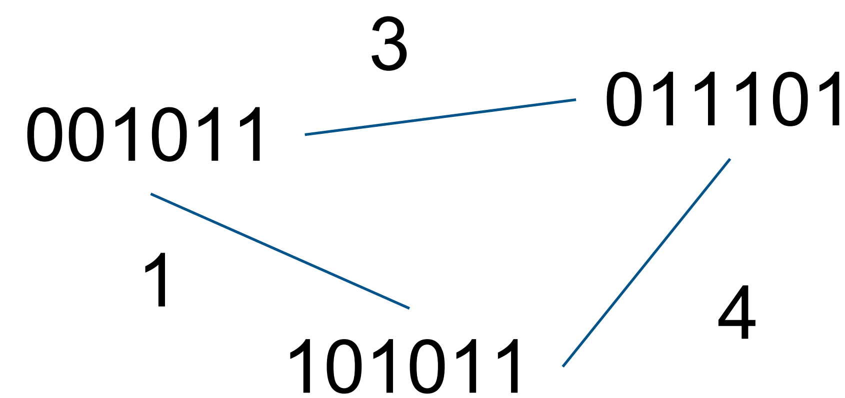
$$d(S) = \min_{x,y \in S, x \neq y} d(x,y)$$

  - d.h. der kleinste Abstand zweier verschiedener Wörter in S

Beispiel:



Alle Abstände sind 2



Ein Abstand ist 1!

Erkennung und Korrektur mit Hamming-Distanzen

- 1. Fall $d(S) = 1$
  - Keine Fehlerkorrektur
  - Legale Frames unterscheiden sich in nur einem Bit
- 2. Fall $d(S) = 2$
  - Dann gibt es nur $x, y \in S$ mit $d(x,y) = 2$
  - Somit ist jedes u mit $d(x,u) = 1$ illegal,
    - wie auch jedes u mit $d(y,u) = 1$

$$x \xrightarrow{\text{1 Bit Unterschied}} u \xrightarrow{\text{1 Bit Unterschied}} y$$

  - 1-Bit-Fehler
    - können immer erkannt werden
    - aber nicht korrigiert werden

Erkennung und Korrektur mit Hamming-Distanzen

0000
2 1111 2
0101

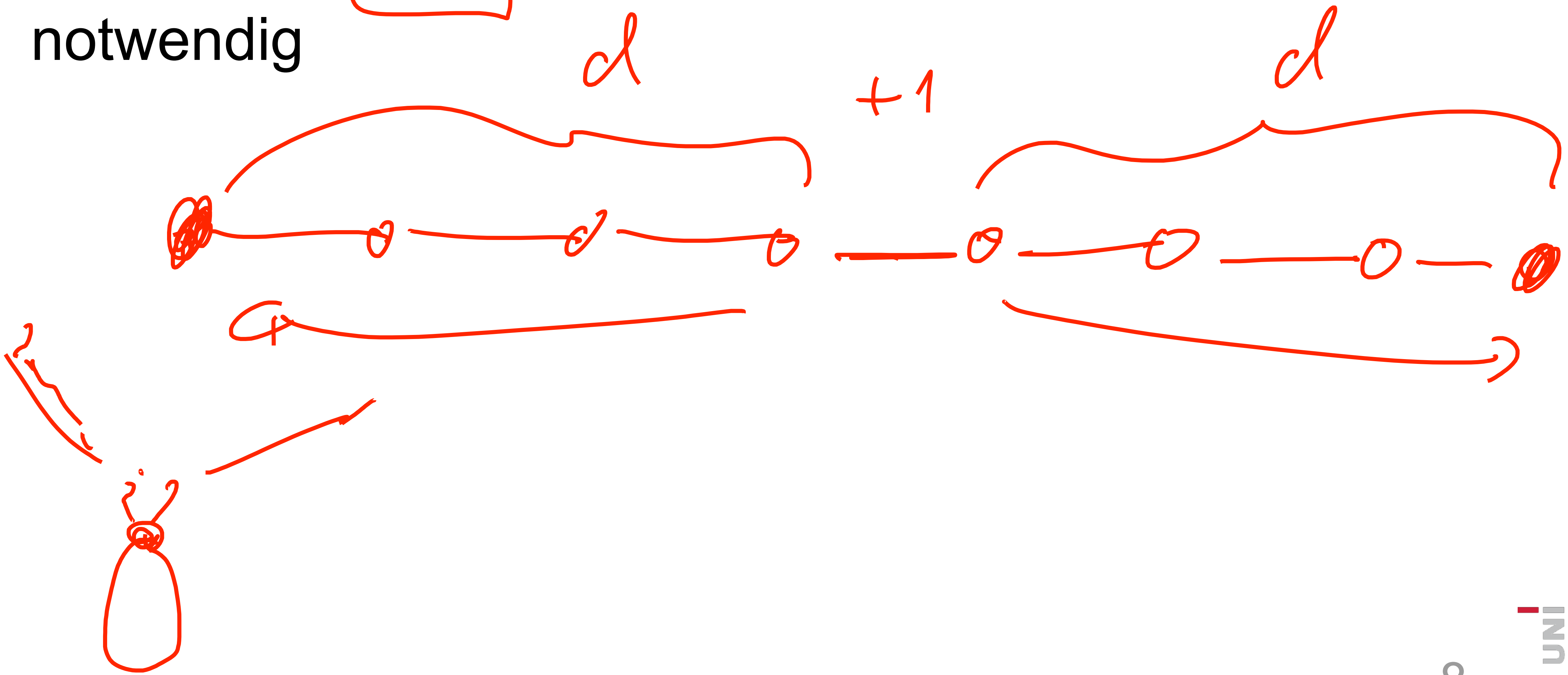
- 3. Fall d(S) = 3
  - Dann gibt es nur x, y ∈ S mit d(x,y) = 3
  - Jedes u mit d(x,u) = 1 illegal und d(y,u) > 1

  - Falls u empfangen wird, sind folgende Fälle denkbar:
    - x wurde gesendet und mit 1 Bit-Fehler empfangen
    - y wurde gesendet und mit 2 Bit-Fehlern empfangen
    - Etwas anderes wurde gesendet und mit mindestens 2 Bit-Fehlern empfangen
  - Es ist also wahrscheinlicher, dass x gesendet wurde, statt y

Erkennung und Korrektur mit Hamming-Distanzen

- Um d Bit-Fehler zu erkennen ist eine Hamming-Distanz von d+1 in der Menge der legalen Frames notwendig

- Um d Bit-Fehler zu korrigieren, ist eine Hamming-Distanz von $\boxed{2d+1}$ in der Menge der legalen Frames notwendig



Codebücher und Kodierungen

- Die Menge der legalen Frames S $\in \{0,1\}^n$ wird **das Code-Buch** oder einfach Kodierung genannt.
  - Die Rate R eines Codes S ist definiert als
    - Die Rate charakterisiert die Effizienz des Codes

$$R_S = \frac{\log |S|}{n} \in [0,1)$$

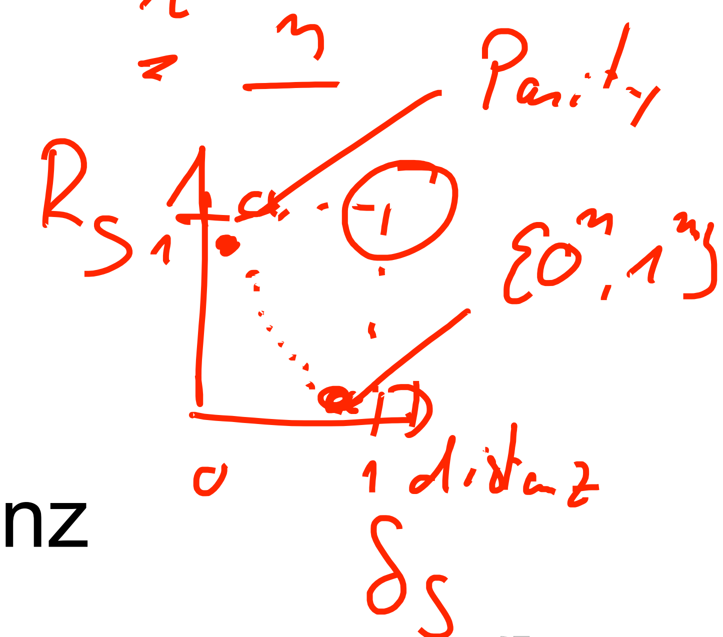
$$\frac{k}{n} = \frac{n-1}{n} = 1 - \frac{1}{n}$$

$$\overbrace{\underbrace{10110}_{k}\ 1}^{n}$$

  - Die Distanz δ des Codes S ist definiert als
    - charakterisiert die Fehlerkorrektur oder Fehlererkennungsmöglichkeiten

$$\delta_S = \frac{d(S)}{n} \in [0,1]$$

$$\frac{1}{n}$$



- Gute Codes haben hohe Raten und hohe Distanz
  - Beides lässt sich nicht zugleich optimieren

CoNe Freiburg

Block-Codes

- Block-Codes kodieren k Bits Originaldaten in n kodierte Bits
  - Zusätzlich werden n-k Symbole hinzugefügt
  - Binäre Block-Codes können höchstens bis zu t Fehler in einem Code-Wort der Länge n mit k Originalbits erkennen, wobei (Gilbert-Varshamov-Schranke):

$$\boxed{2^{n-k} \geq \sum_{i=0}^{t} \binom{n}{i}}$$

q

q

$$\overbrace{0\,00\,1\,1\,1\,0\,1\,1\,0\,1\,1\,1\,0\,0}^{n}$$

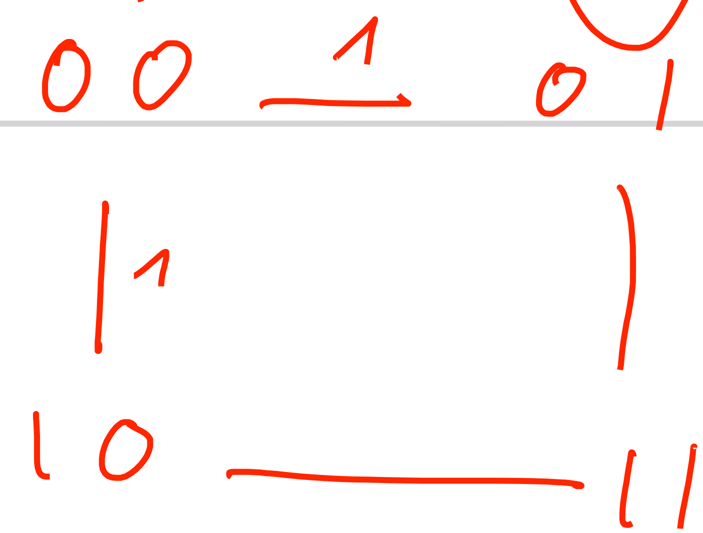
$$\underbrace{0\,1\,0\,1\,1\,0}_{4}$$

  - Das ist eine theoretische obere Schranke
- Beispiele
  - Bose Chaudhuri Hocquenghem (BCH) Codes
    - basierend auf Polynomen über endlichen Körpern (Galois-Körpern)
  - Reed Solomon Codes
    - Spezialfall nichtbinärer BCH-Codes

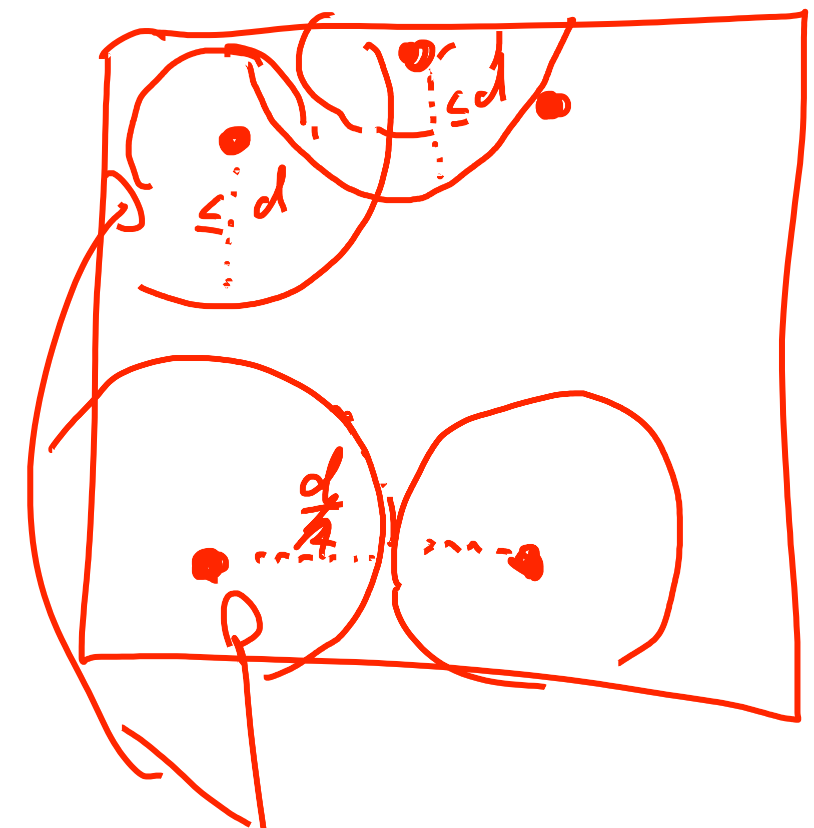
UNI FREIBURG

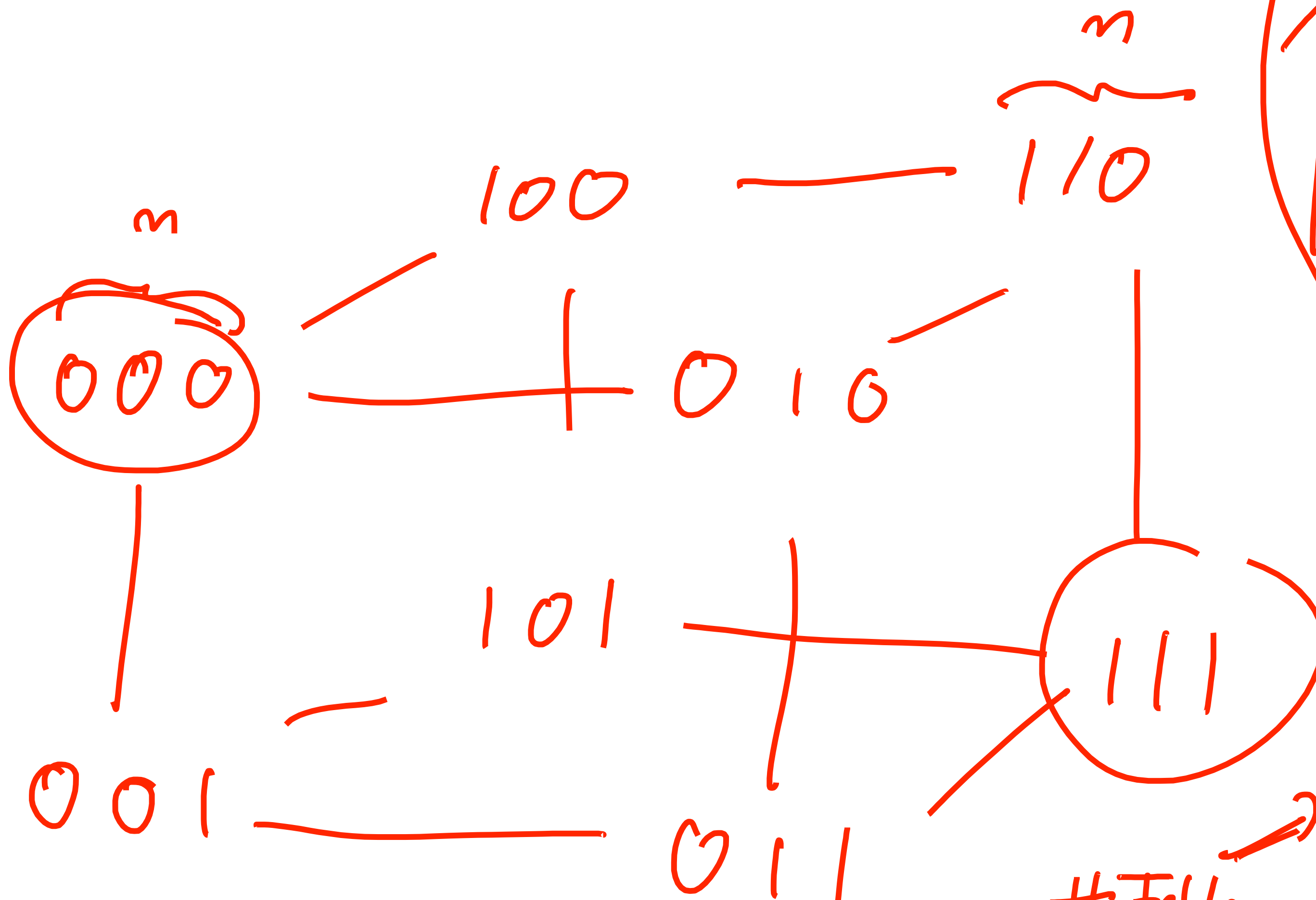
$\underbrace{000}_{4}$

$\frac{d}{2}$ Fehler



$\{0,1\}^n$





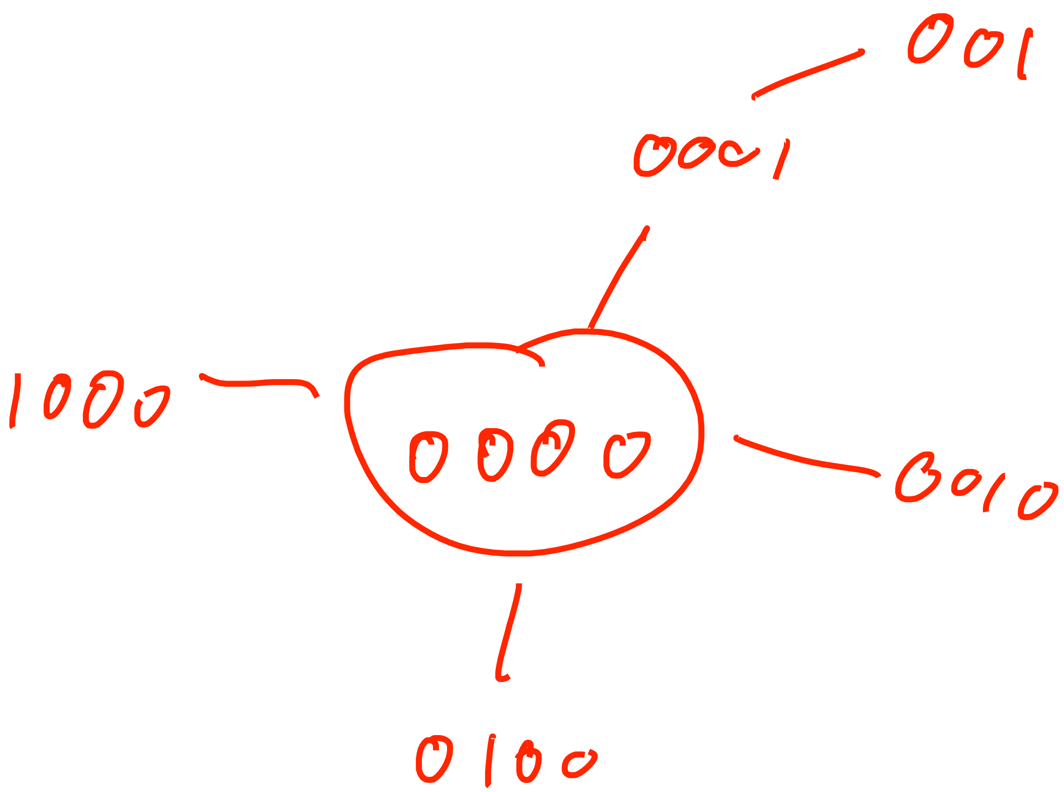
$$\left(\sum_{t=0}^{d/2}\binom{n}{t}\right)\cdot 2^4 \le 2^n$$

\# Fehler

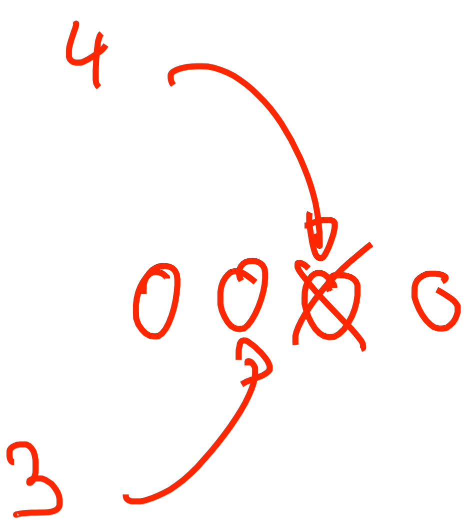
$$\sum_{t=0}^{d/2}\binom{n}{t} \le 2^{n-4}$$

$\binom{4}{2}$

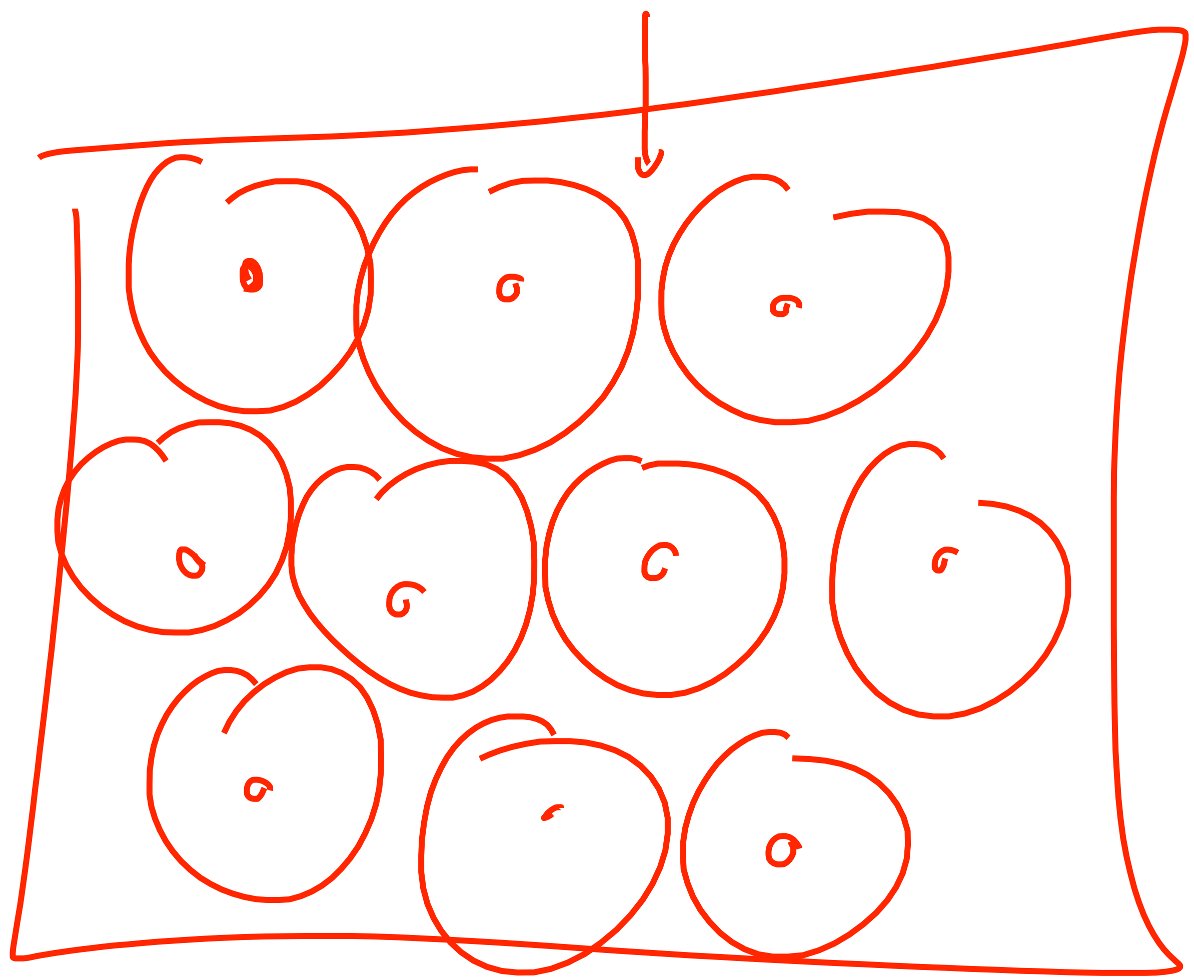
$\sum_{t=0}^{d} \binom{n}{t}$



dist	#	
0	1	$1 \binom{n}{0}$
1	4	$\binom{n}{1}$
2	$\binom{4}{2}$	$\binom{n}{2}$
3	$\binom{4}{3}$	$\binom{n}{3}$
4	$1 = \binom{4}{4}$	$\binom{n}{4}$



$$\frac{4\cdot 3}{2\cdot 1} = \frac{4\cdot 3\cdot 2\cdot 1}{(2\cdot 1)(2\cdot 1)} = \frac{4!}{2!\cdot 2!} = \binom{4}{2}$$



Fehlerkontrolle

- Zumeist gefordert von der Vermittlungsschicht
  - Mit Hilfe der Frames
- Fehlererkennung
  - Gibt es fehlerhaft übertragene Bits?
- Fehlerkorrektur
  - Behebung von Bitfehlern
  - Vorwärtsfehlerkorrektur (Forward Error Correction)
    - Verwendung von redundanter Kodierung, die es ermöglicht Fehler ohne zusätzliche Übertragungen zu beheben
  - Rückwärtsfehlerkorretur (Backward Error Correction)
    - Nach Erkennen eines Fehlers, wird durch weitere Kommunikation der Fehler behoben

Fehlerkontrolle

Fehlererkennung

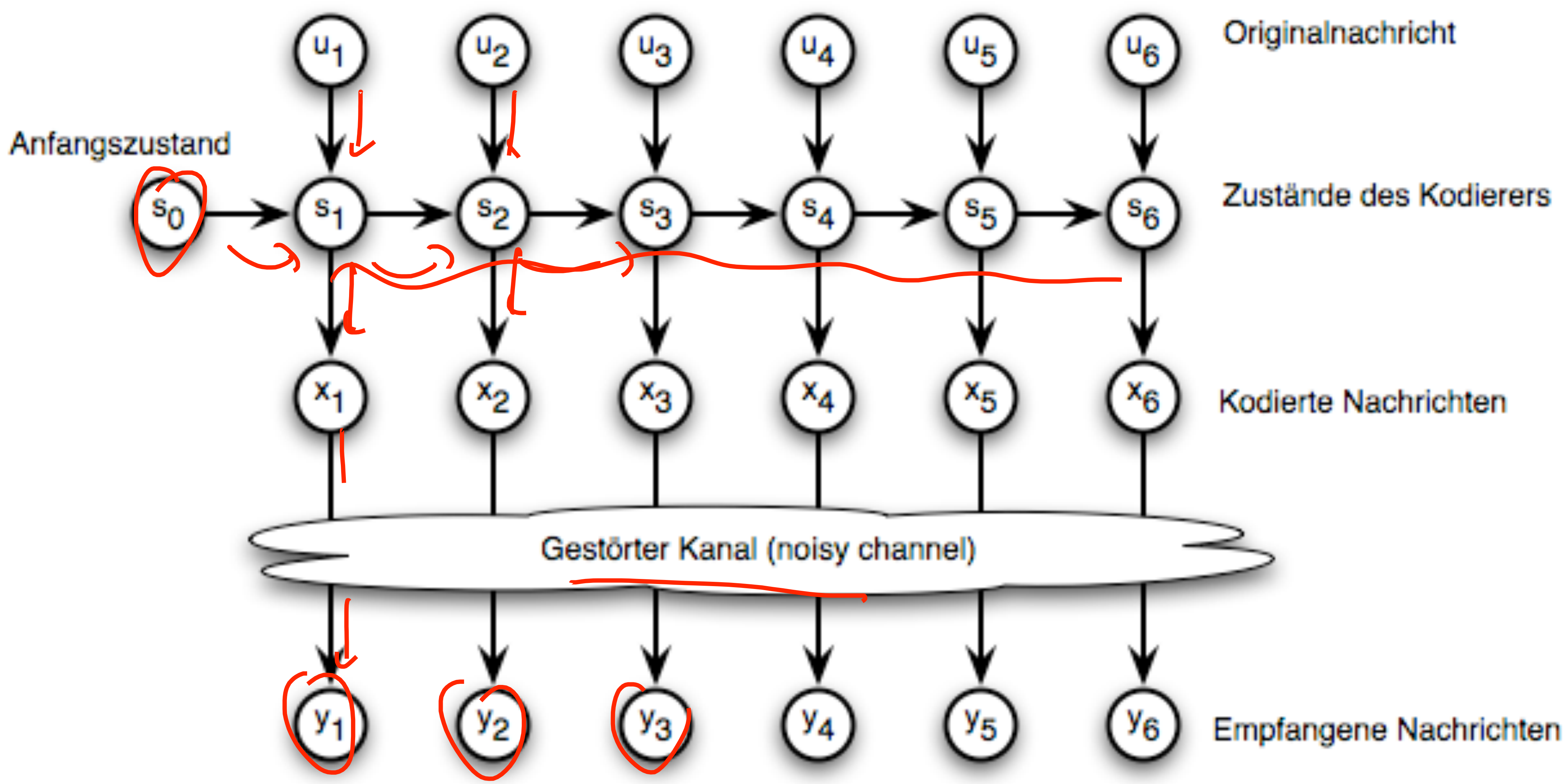
Fehlerkorrektur

Vorwärtsfehler-
korrektur

Rückwärtsfehler-
korrektur

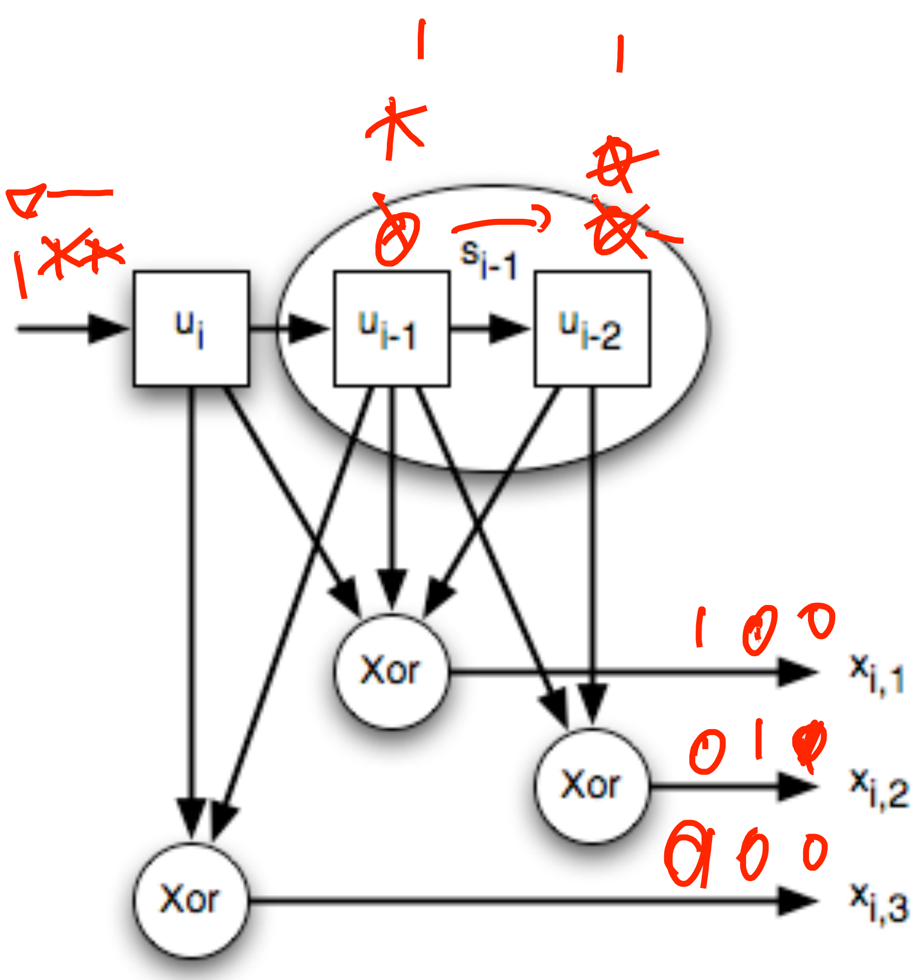
Faltungs-Codes

- Faltungs-Codes (Convolutional Codes)
  - Daten und Fehlerredundanz werden vermischt.
  - k Bits werden auf n Bits abgebildet
  - Die Ausgabe hängt von den k letzten Bits und dem internen Zustand ab.

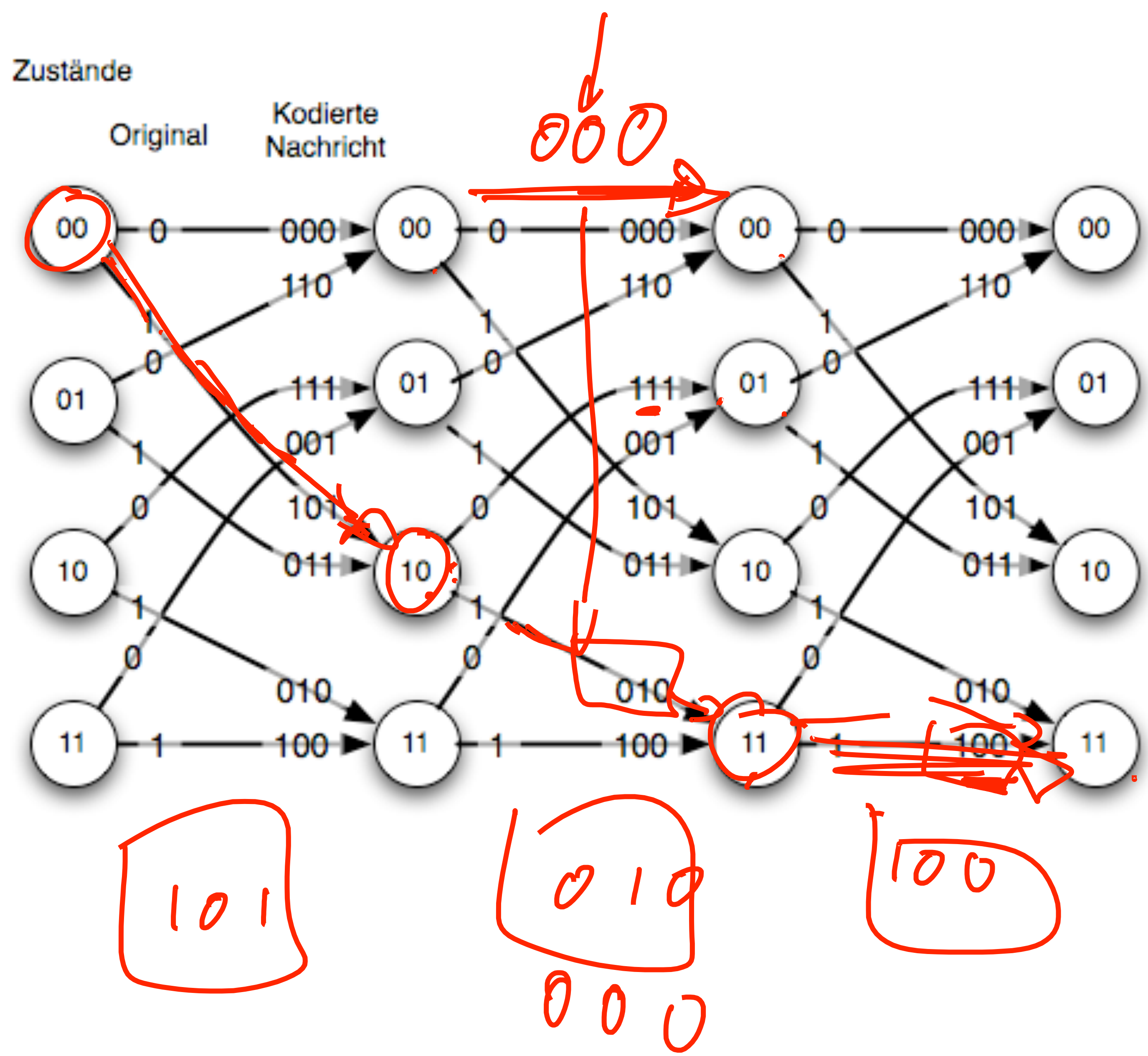


Beispiel

Faltungs-Kodierer



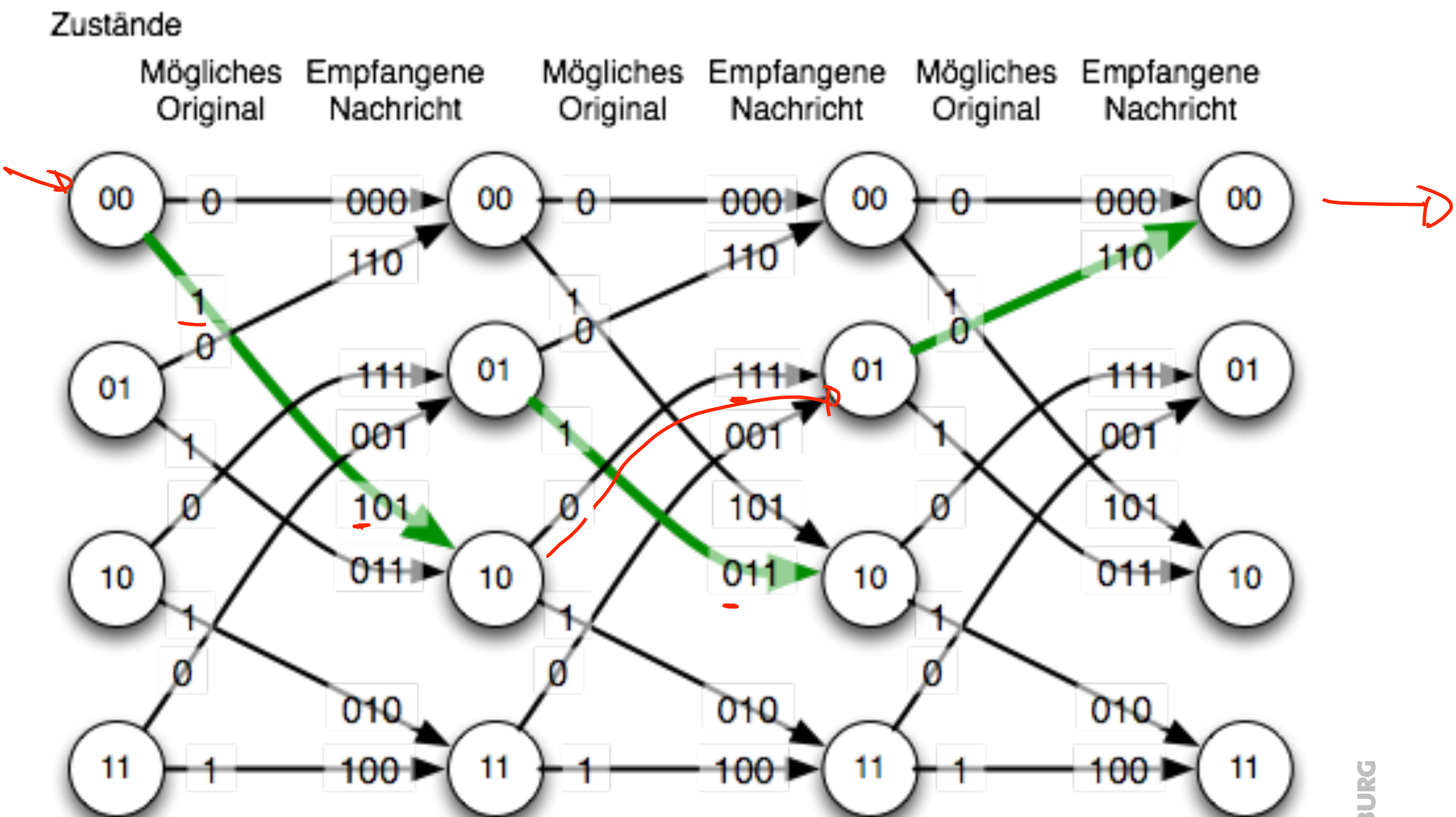
Trellis-Diagramm



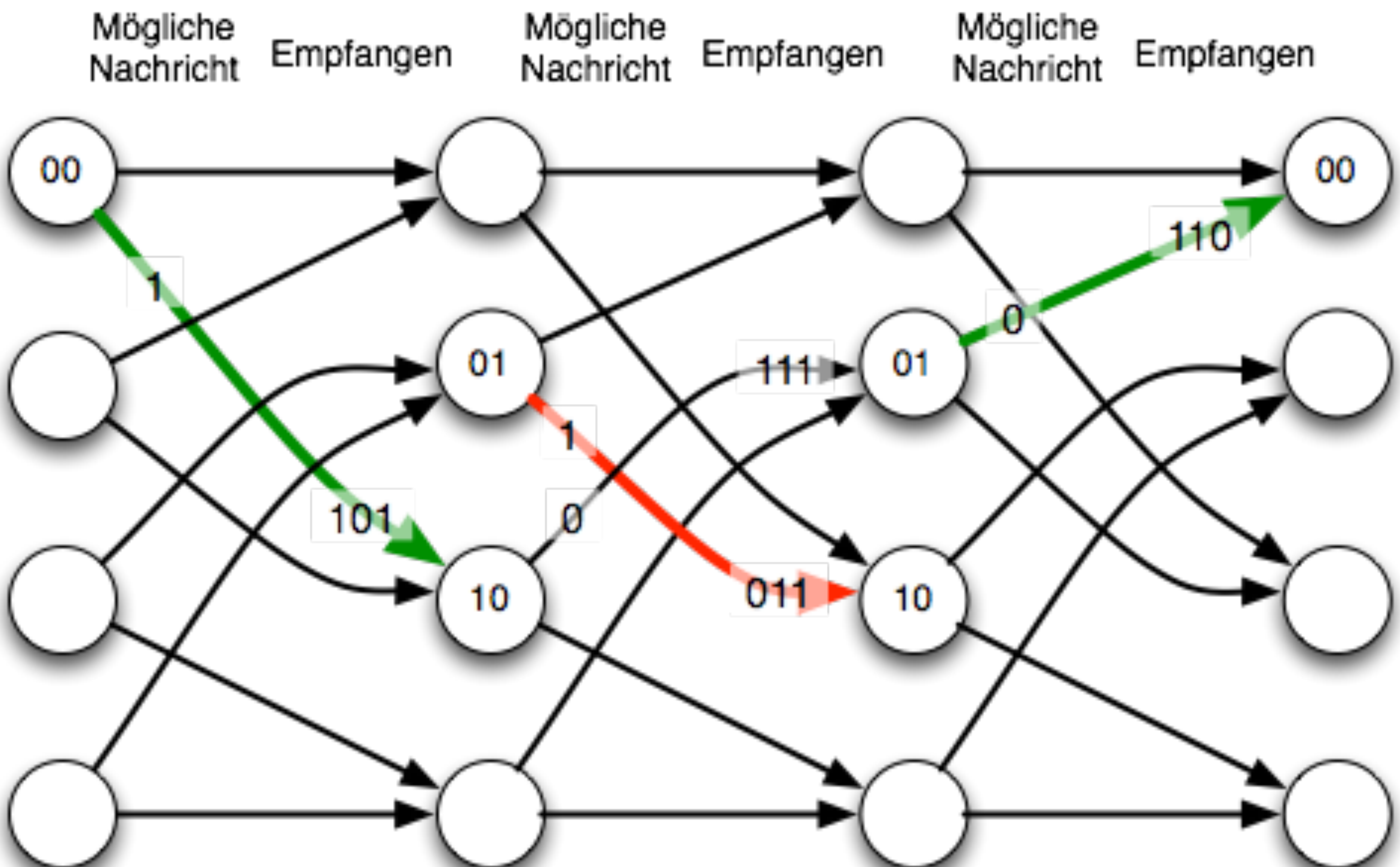
Dekodierung der Faltungs-Codes: Algorithmus von Viterbi

- Dynamische Programmierung
- Zwei notwendige Voraussetzungen für Dekodierung
  - (für den Empfänger) unbekannte Folge von Zuständen
  - beobachtete Folge von empfangenen Bits (möglicherweise mit Fehler)
- Der Algorithmus von Viterbi bestimmt die warscheinlichste Folge von Zuständen, welches die empfangenen Bits erklärt
  - Hardware-Implementation möglich

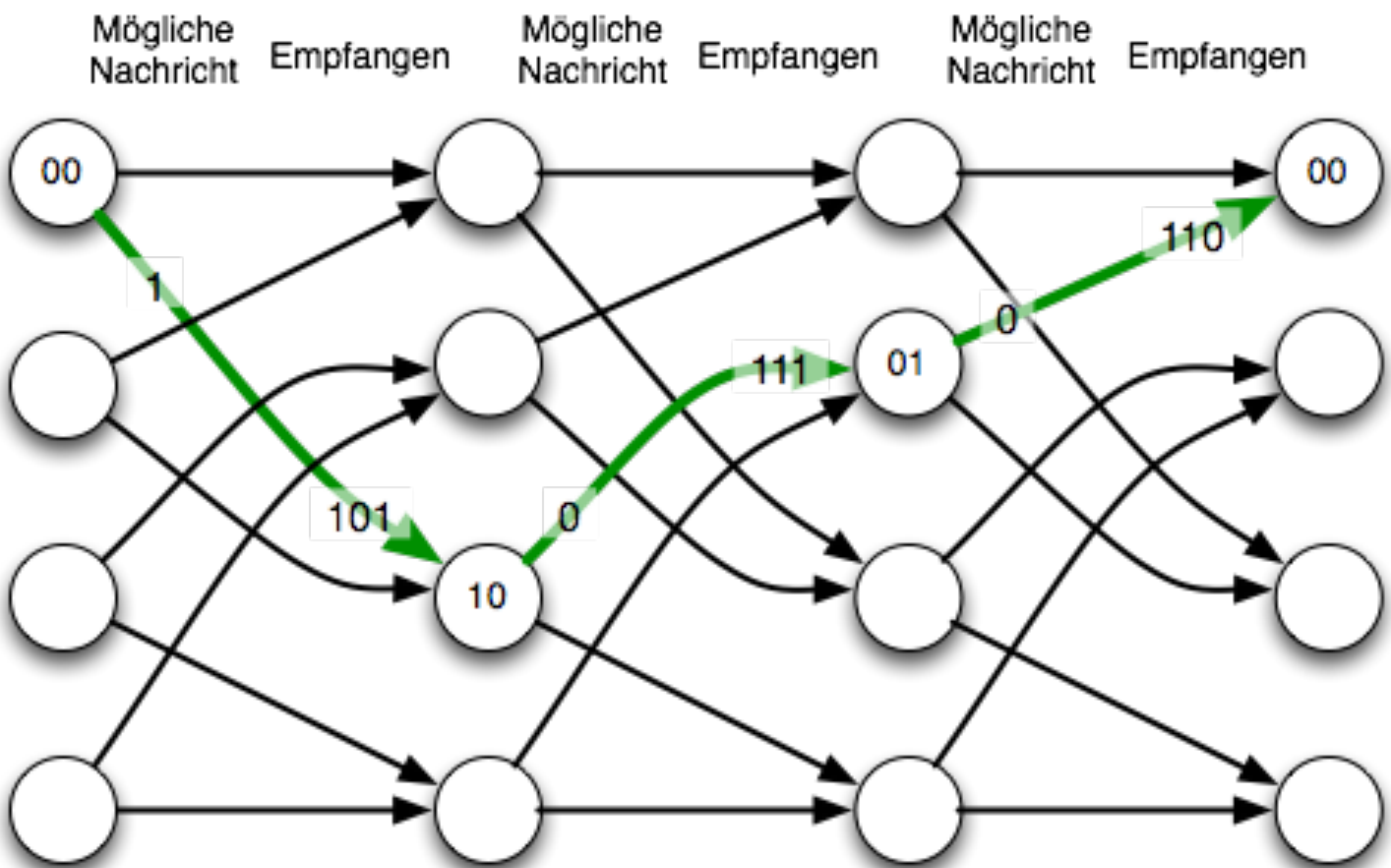
Dekodierung (I)



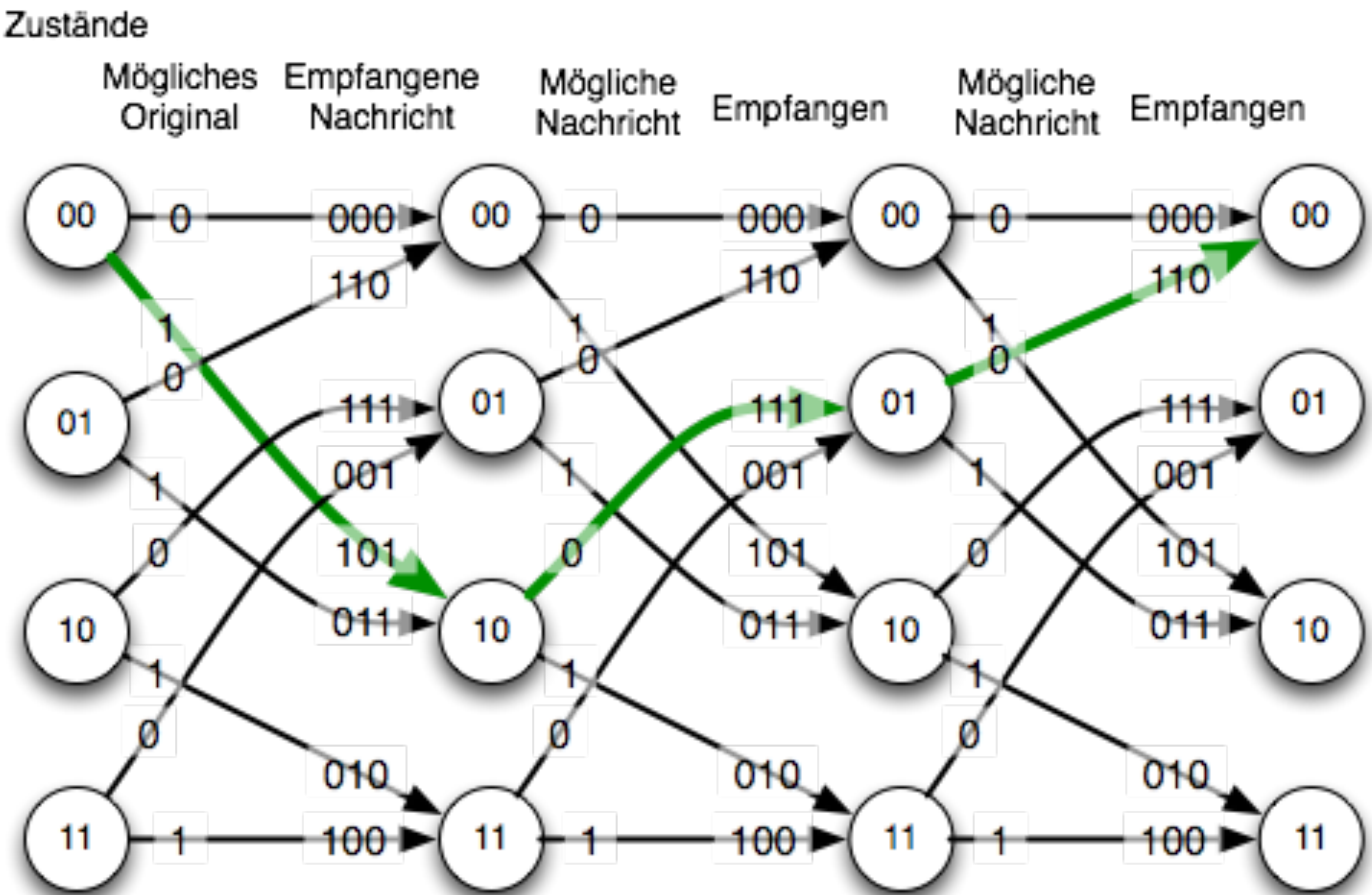
Dekodierung (II)



Dekodierung (III)

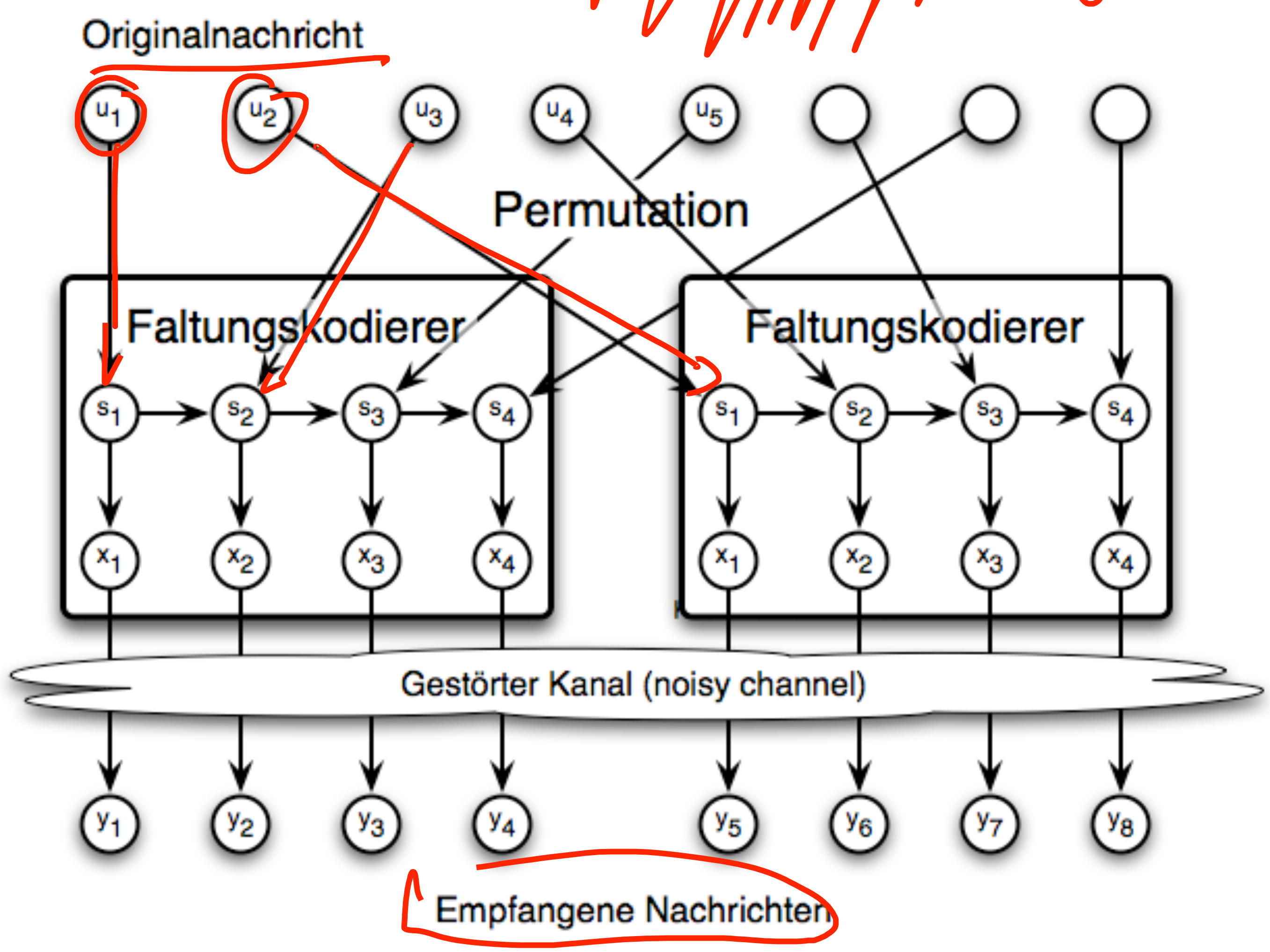


Dekodierung (IV)



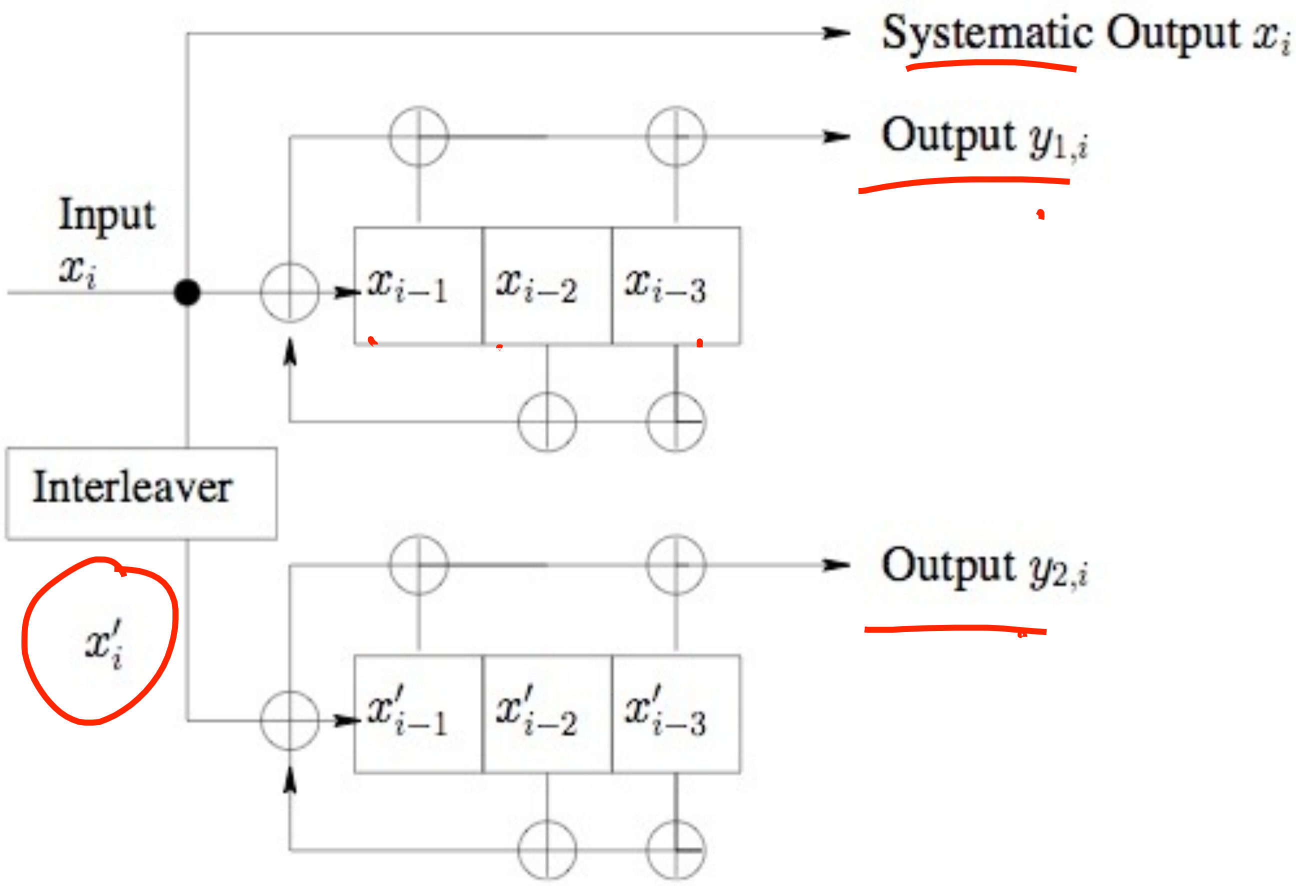
Turbo-Codes

- Turbo-Codes sind wesentlich effizienter als Faltungs-Codes
  - bestehen aus zwei Faltungs-Codes welche abwechselnd mit der Eingabe versorgt werden.
  - Die Eingabe wird durch eine Permutation (Interleaver) im zweiten Faltungs-Code umsortiert



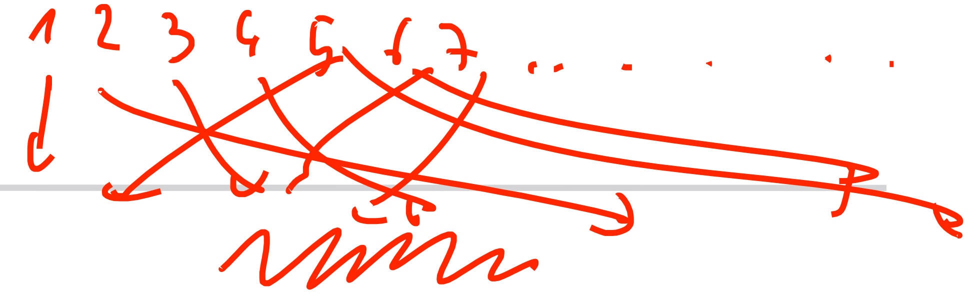
Turbo-Codes

- Beispiel:
  - UMTS Turbo-Kodierer
- Dekodierung von Turbo-Codes ist effizienter möglich als bei Faltungscodes
- Kompensation von Bursts

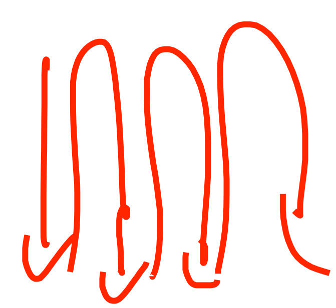


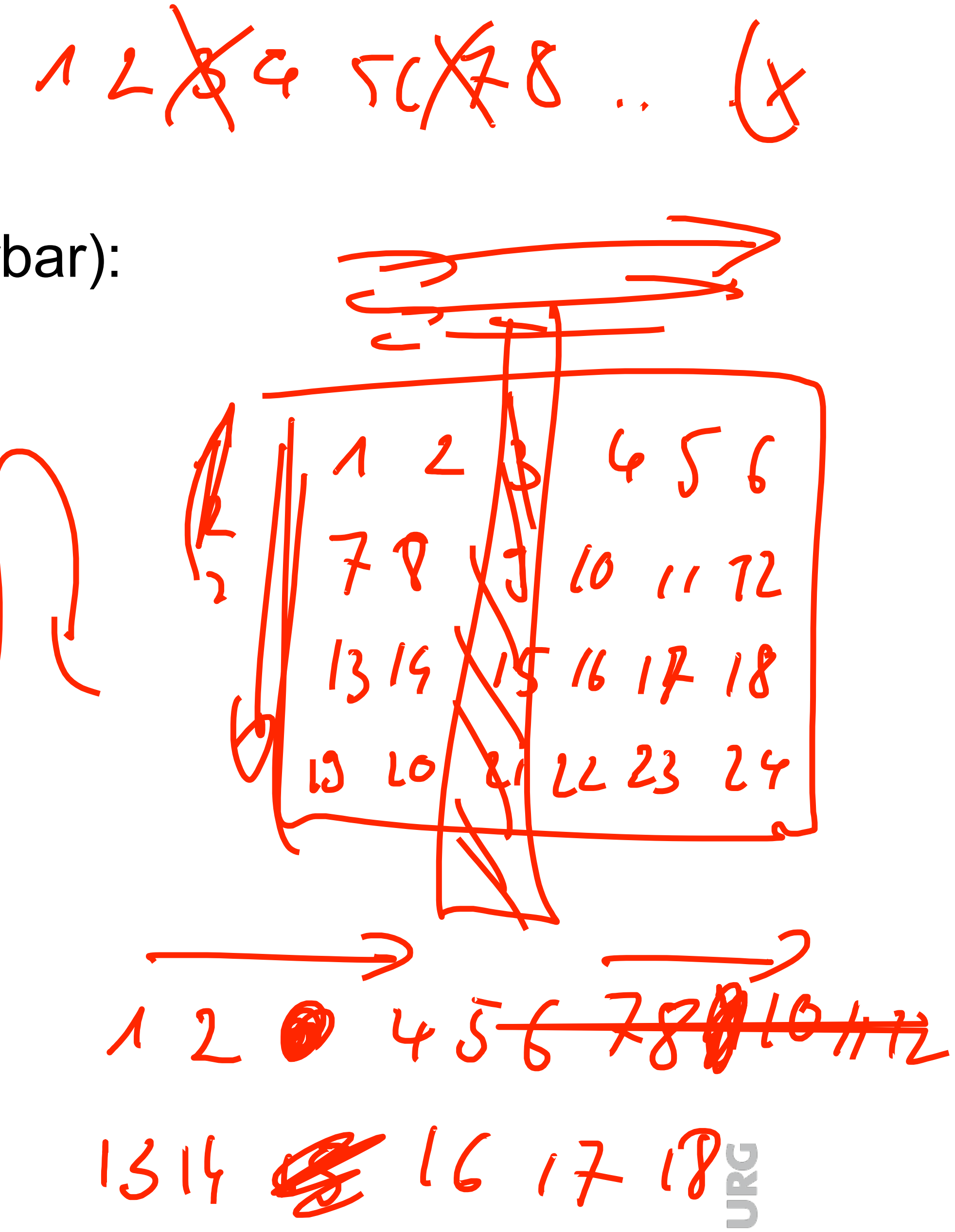
CoNe Freiburg

Interleavers



- Fehler treten oftmals gehäuft auf (Bursts)
  - z.B.: Daten: 0 1 2 3 4 5 6 7 8 9 A B C D E F
  - mit Fehler: 0 1 2 3 **? ? ? ? ?** 9 A B C D E F
- Dann scheitern klassische Kodierer ohne Interleavers
  - Nach Fehlerkorrektur (zwei Zeichen in Folge reparierbar):
    0 1 2 3 4 5 **?** 7 8 9 A B C D E F
- Interleaver:
  - Permutation der Eingabekodierung:
    0 1 2 3
    4 5 6 7
    8 9 A B
    C D E F





  - z.B. Row-column Interleaver:
    0 4 8 C 1 5 9 D 2 6 A E 3 7 B F
  - mit Fehler: 0 4 8 C **? ? ? ? ?** 6 A E 3 7 B F
  - Rückpermutiert: 0 **? ?** 3 4 **?** 6 7 8 **?** A B C D **?** F
  - nach FEC: 0 1 2 3 4 5 6 7 8 9 A B C D E F

Systeme II

3. Die Datensicherungsschicht

Christian Schindelhauer
Technische Fakultät
Rechnernetze und Telematik
Albert-Ludwigs-Universität Freiburg