Systeme II

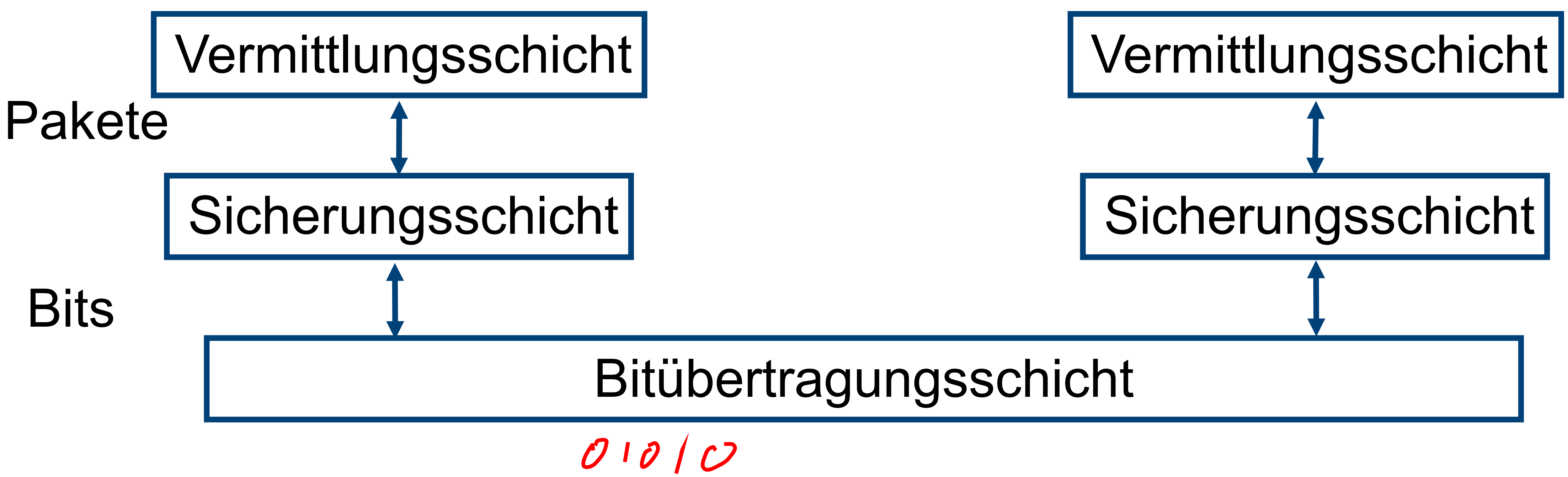
3. Die Datensicherungsschicht

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

Version 15.05.2017

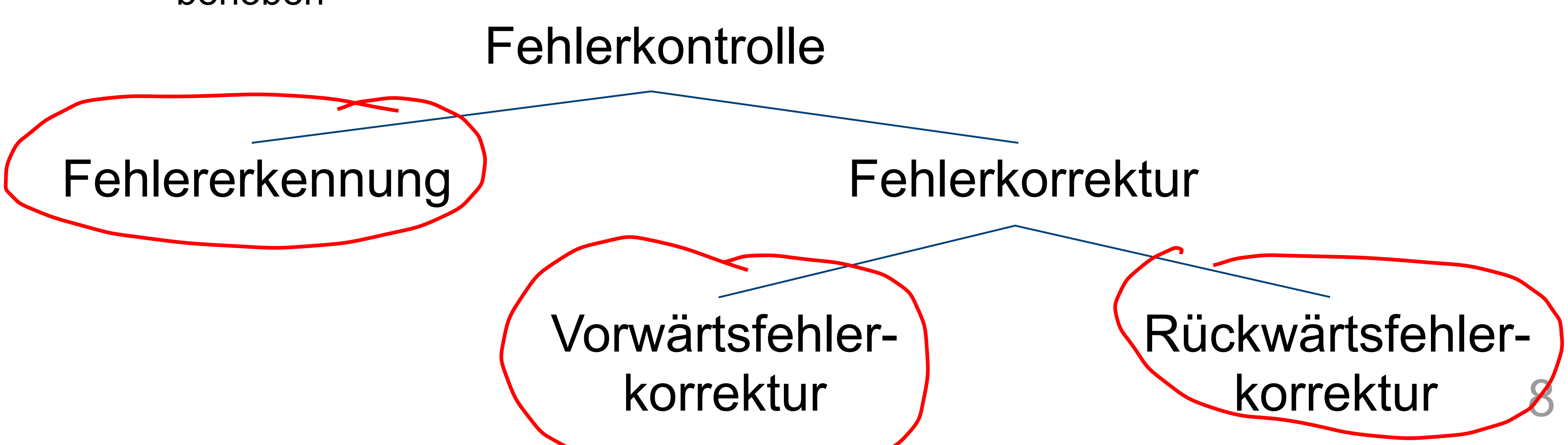
Dienste der Sicherungsschicht

- Situation der Sicherungsschicht
  - Die Bitübertragungsschicht überträgt Bits
  - Aber unstrukturiert und möglicherweise fehlerbehaftet
- Die Vermittlungsschicht erwartet von der Sicherungsschicht
  - Fehlerfreie Übermittlung
  - Übermittlung von strukturierten Daten
    - Datenpakete oder Datenströme
  - Störungslosen Datenfluss



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



Verbindungsaufbau

- Nutzen von Verbindungen
  - Kontrolle des Verbindungsstatus
    - Korrektheit des Protokolls
  - Fehlerkontrolle
    - Verschiedene Fehlerkontrollverfahren vertrauen auf gemeinsamen Kontext von Sender und Empfänger
- Aufbau und Terminierung von Verbindungen
  - “Virtuelle Verbindungen”
    - Es werden keine Schalter umgelegt
    - Interpretation des Bitstroms
  - Kontrollinformationen in Frames
  - Besonders wichtig bei drahtlosen Medien
- Das Problem wird im Rahmen der Transportschicht ausführlich diskutiert
  - Vgl. Sitzungsschicht vom OSI-Modell

Flusskontrolle

- Problem: Schneller Sender und langsamer Empfänger
  - Der Sender lässt den Empfangspuffer des Empfängers überlaufen
  - Übertragungsbandweite wird durch sinnlosen Mehrfachversand (nach Fehlerkontrolle) verschwendet
- Anpassung der Frame-Sende-Rate an dem Empfänger notwendig

Langsamer Empfänger

Schneller Sender

Frames

- Wo fängt der Frame an und wo hört er auf?
- Achtung:
  - Die Bitübertragungsschicht kann auch Bits liefern, wenn der Sender tatsächlich nichts sendet
  - Der Empfänger
    - könnte das Rauschen auf dem Medium interpretieren
    - könnte die Folge 00000000.... liefern
  - Daten oder Kontrollinformation?

Zeit

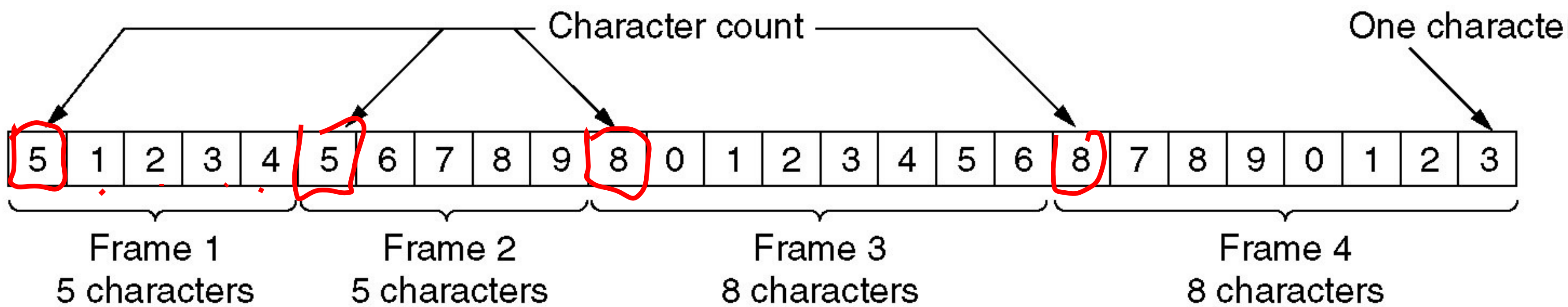
Übertragener Bitstrom 0110010101110101110010100010101010101010101100010

Frame-Anfang?

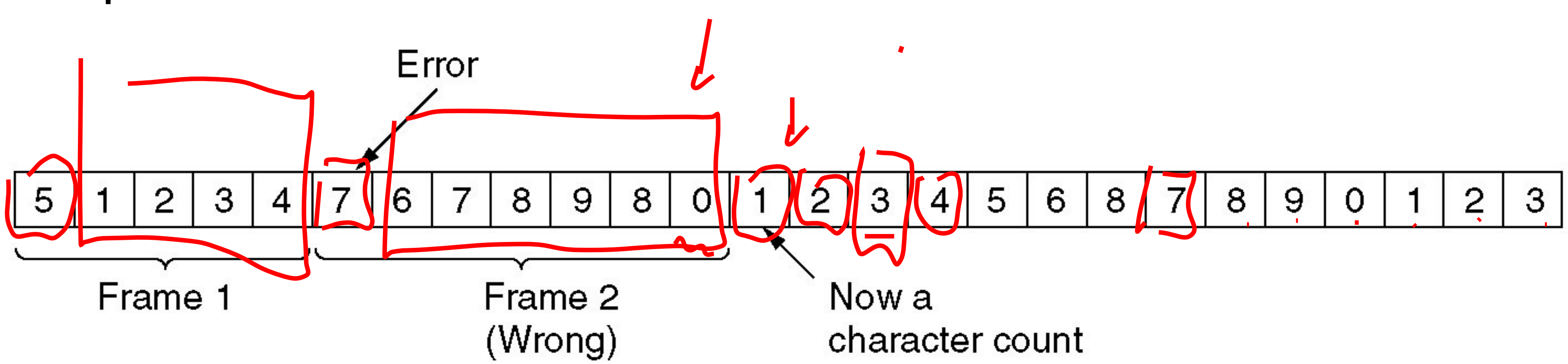
Frame-Ende?

Frame-Grenzen durch Paketlängen?

- Idee: Ankündigung der Bitanzahl im Frame-Header

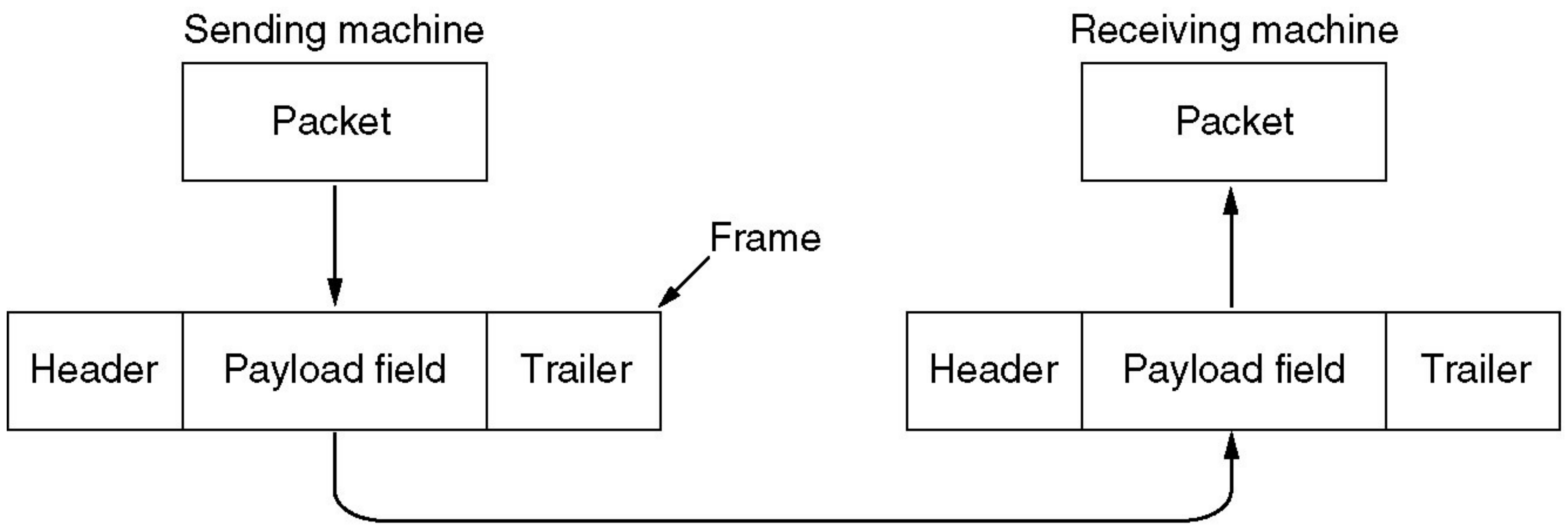


- Problem: Was, wenn die Frame-Länge fehlerhaft übertragen wird?
  - Der Empfänger kommt aus dem Takt und interpretiert neue, sinnlose Frames
  - Variable Frame-Größen mit Längeninformation sind daher kein gutes Konzept



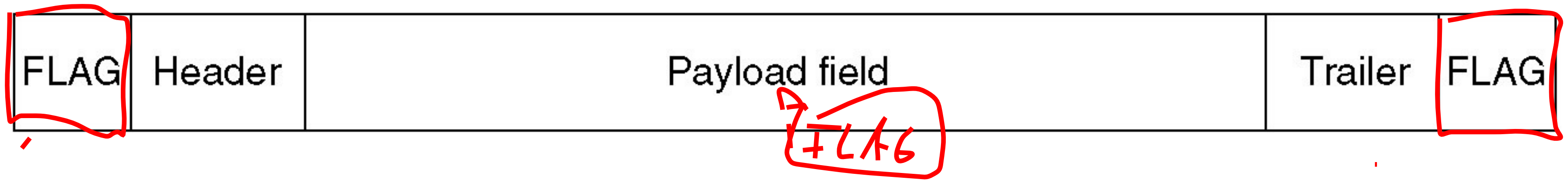
Header und Trailer

- Header und Trailer
  - Zumeist verwendet man Header am Anfang des Frames, mitunter auch Trailer am Ende des Frames
  - signalisieren den Frame-Beginn und das Frame-Ende
  - tragen Kontrollinformationen
    - z.B. Sender, Empfänger, Frametypen, Fehlerkontrollinformation



Flag Bytes und Bytestopfen

- Besondere “Flag Bytes” markieren Anfang und Ende eines Frames

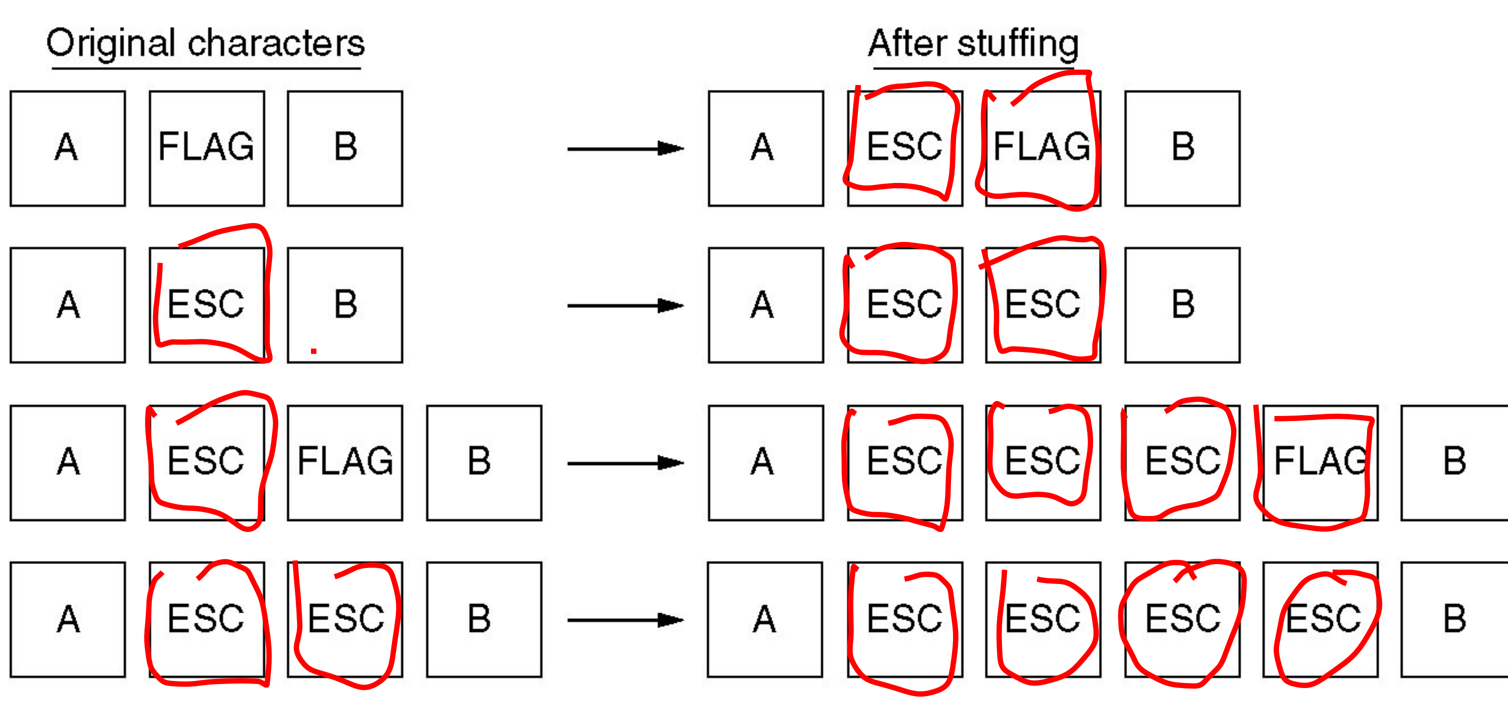


- Falls diese Marker in den Nutzdaten vorkommen
  - Als Nutzdatenbyte mit Sonderzeichen (Escape) markieren
    - Bytestopfen (byte stuffing)
  - Falls Sonderzeichen und “Flag-Byte” erscheinen, dito,
    - etc., etc.

"FLAG"

$\frac{2}{256}$

1



UNI FREIBURG

Frames durch Bit-Sequenzen/Bitstopfen

CoNe Freiburg

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

- Bytestopfen verwendet das Byte als elementare Einheit
  - Das Verfahren funktioniert aber auch auf Bitebene
- Flag Bits und Bitstopfen (bit stuffing)
  - Statt flag byte wird eine Bit-Folge verwendet
    - z.B.: 01111110
  - Bitstopfen
    - Wenn der Sender eine Folge von fünf 1er senden möchte, wird automatisch eine 0 in den Bitstrom eingefügt
      - Außer bei den Flag Bits
- Der Empfänger entfernt eine 0 nach fünf 1ern

0 1 1 1 1 1 1 1 0 5 0

Originale Nutzdate (a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Nach dem Bitstopfen (b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0

Stuffed bits

Nach der "Entstopfung" (c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

UNI FREIBURG

Frames durch Code-Verletzung

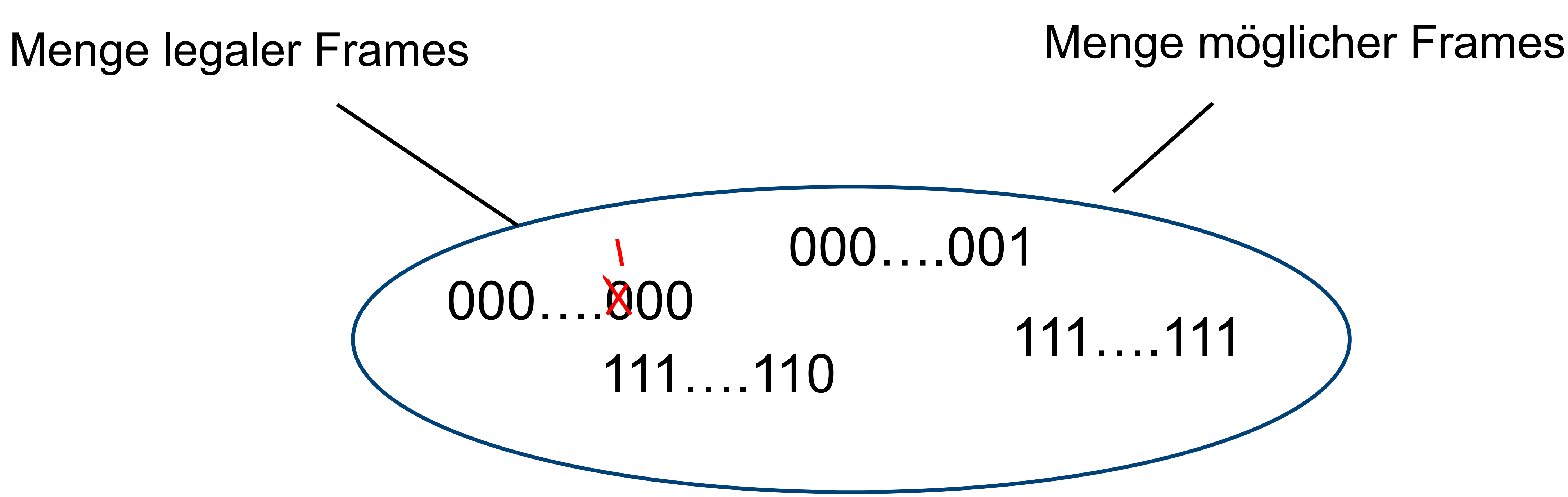
- Möglicher Spielraum bei Bitübertragungsschicht bei der Kodierung von Bits auf Signale
  - Nicht alle möglichen Kombination werden zur Kodierung verwendet
  - Zum Beispiel: Manchester-Kodierung hat nur tief/hoch und hoch/tief–Übergang
- Durch “Verletzung” der Kodierungsregeln kann man Start und Ende des Rahmens signalisieren
  - Beispiel: Manchester – Hinzunahme von hoch/hoch oder tief/tief
    - Selbsttaktung von Manchester gefährdet?
- Einfache und robuste Methode
  - z.B. verwendet in Ethernet
  - Kosten? Effiziente Verwendung der Bandbreite?

Fehlerkontrolle

- Aufgaben
  - Erkennung von Fehlern (fehlerhafte Bits) in einem Frame
  - Korrektur von Fehlern in einem Frame
- Jede Kombination dieser Aufgaben kommt vor
  - Erkennung ohne Korrektur
    - Löschen eines Frames ohne weiter Benachrichtigung (drop a frame)
    - Höhere Schichten müssen sich um das Problem kümmern
  - Korrektur ohne Erkennung
    - Es werden bestmöglich Bitfehler beseitigt, möglicherweise sind aber noch Fehler vorhanden
    - Sinnvoll, falls Anwendung Fehler tolerieren kann
      - Beispiel: Tonübertragung
    - Prinzipiell gerechtfertigt, weil immer eine positive Restfehlerwahrscheinlichkeit bleibt

Redundanz

- Redundanz ist eine Voraussetzung für Fehlerkontrolle
- Ohne Redundanz
  - Ein Frame der Länge m kann ~~2m~~ 2^m mögliche Daten repräsentieren
  - Jede davon ist erlaubt
- Ein fehlerhaftes Bit ergibt einen neuen Dateninhalt

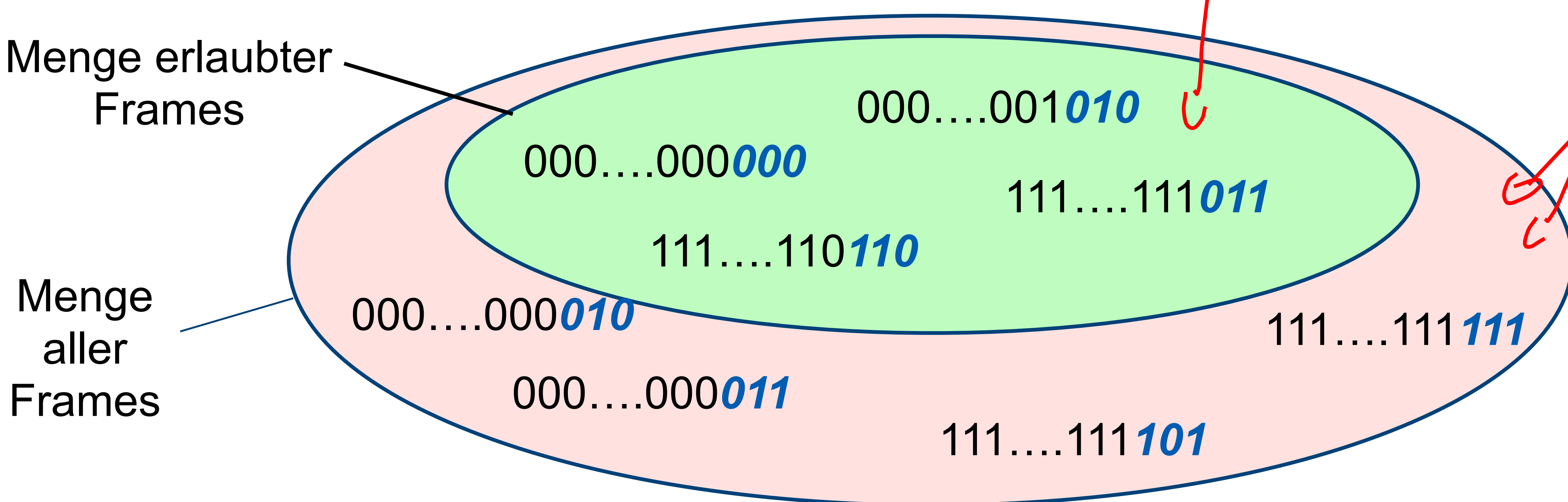


Redundanz

~~Myade~~

- Kernidee:
  - Einige der möglichen Nachrichten sind verboten
  - Um dann 2^m legale Frames darzustellen
    - werden mehr als 2^m mögliche Frames benötigt
    - Also werden mehr als m Bits in einem Frame benötigt
  - Der Frame hat also Länge $n > m$
  - $r = m - n$ sind die redundanten Bits
    - z.B. Im Header oder Trailer
- Nur die Einschränkung auf erlaubte und verbotene (legal/illegal) Frames ermöglicht die Fehlerkontrolle

Myriade



Einfachste Redundanz: Das Paritätsbit

$$x_1 \oplus x_2 \oplus x_3 \oplus c = \begin{cases} 1 & \text{odd} \\ 0 & \text{even} \end{cases}$$

- Eine einfache Regel um ein redundantes Bit zu erzeugen
  (d.h. n=m+1)

$$a \oplus b = c$$

$$a \oplus c = b$$

- Parität
  - Odd parity
    - Eine Eins wird hinzugefügt, so dass die Anzahl der 1er in der Nachricht ungerade wird (ansonsten eine Null)
  - Even parity
    - Eine Eins wird hinzugefügt, so dass die Anzahl der 1er in der Nachricht gerade wird (ansonsten wird eine Null hinzugefügt)

$$\frac{2^n}{2^m} = \frac{2^{m+1}}{2^m} = 2$$

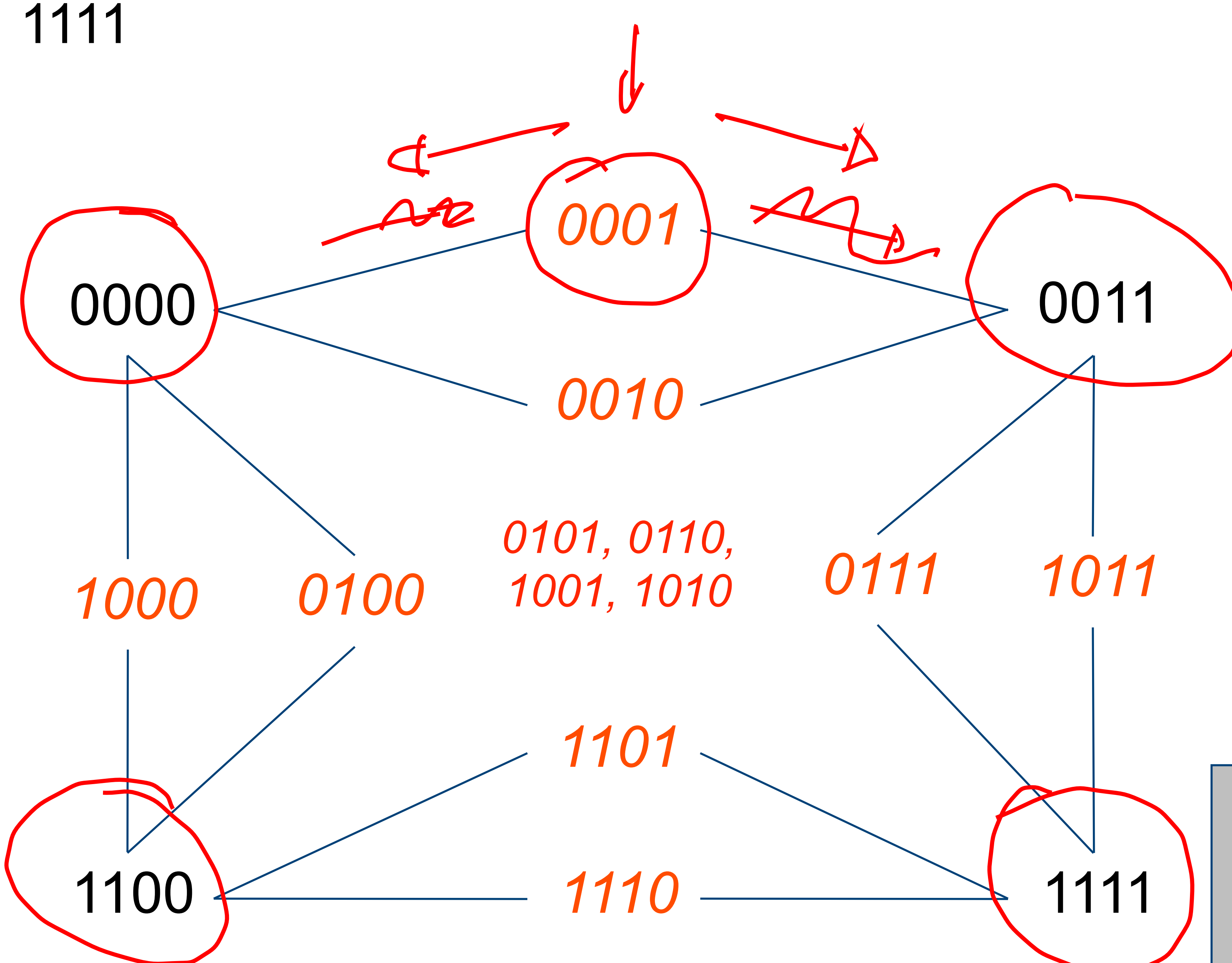
- Beispiel:
  - Originalnachricht ohne Redundanz: 01101011001
  - Odd parity: 011010110011
  - Even parity: 011010110010

Der Nutzen illegaler Frames

- Der Sender sendet nur erlaubte Frames
- In der Bitübertragungsschicht könnten Bits verfälscht werden
- Hoffnung:
  - Legale Frames werden nur in illegale Nachrichten verfälscht
  - Und niemals ein legaler Frame in einen anderen Legalen
- Notwendige Annahme
  - In der Bitübetragungsschicht werden nur eine bestimmte Anzahl von Bits verändert
    - z.B. k Bits pro Frame
  - Die legalen Nachrichten sind verschieden genug, um diese Frame-Fehlerrate zu erkennen

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!

uvxy – erlaubt *abcd* – verboten

Hamming-Distanz

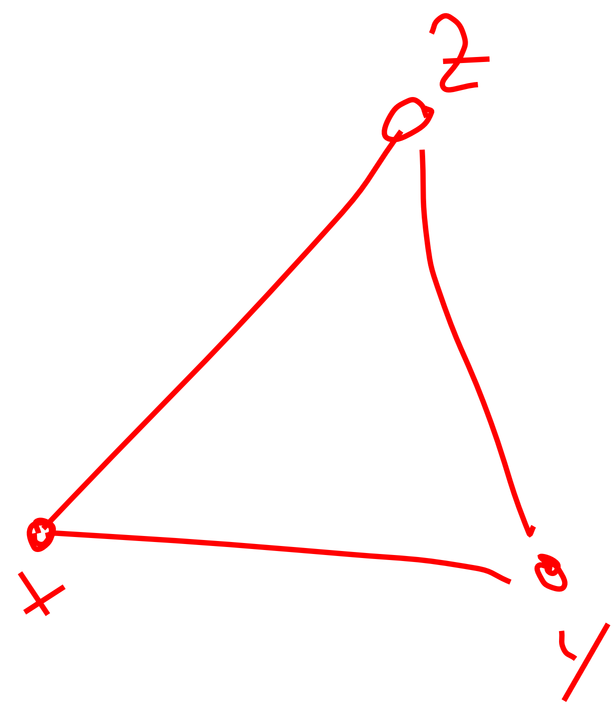
- Der "Abstand" der erlaubten Nachrichten zueinander war immer zwei Bits
- Definition: Hamming-Distanz
  - Seien $x = x_1, \ldots, x_n$ und $y = y_1, \ldots, y_n$ Nachrichten
  - Dann sei $d(x,y)$ = die Anzahl der 1er Bits in x XOR y
- Intuitiver: die Anzahl der Positionen, in denen sich x und y unterscheiden

0 1 1 0 1
0 0 1 1 1

0 1 0 1 0 ⇒ 2

Hamming-Distanz

- Die Hamming-Distanz ist eine Metrik
  - Symmetrie
    - d(x,y) = d(y,x)
  - Dreiecksungleichung:
    - d(x,y) ≤ 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

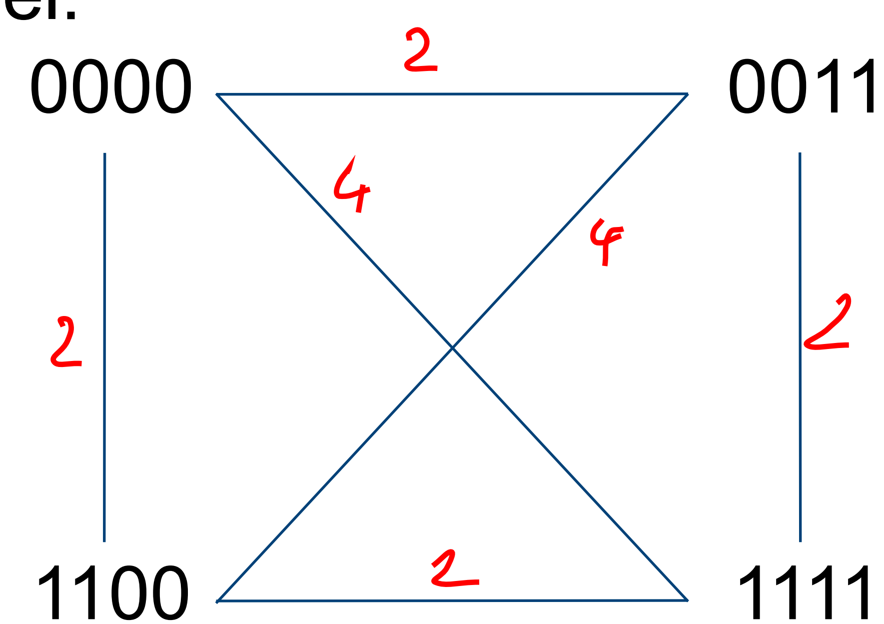
Hamming-Distanz von Nachrichtenmengen

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

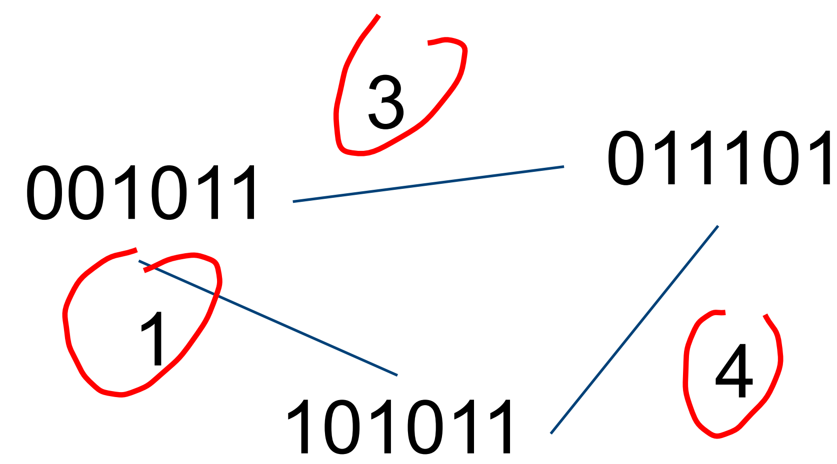
$$d(S) = \min_{x,y \in S, x \neq y} d(x,y)$$

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

Beispiel:



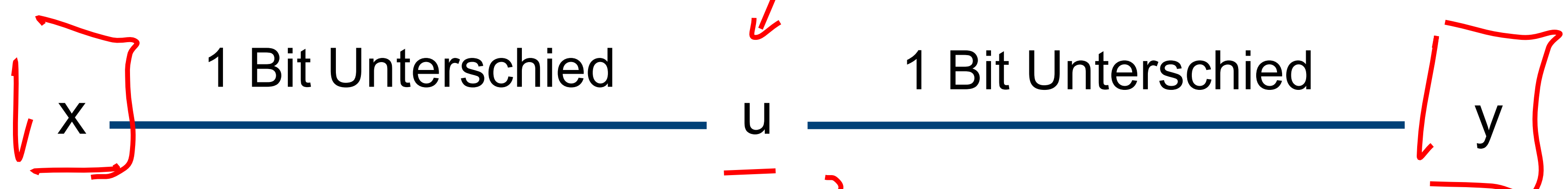
Alle Abstände sind 2



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



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

Erkennung und Korrektur mit Hamming-Distanzen

- 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

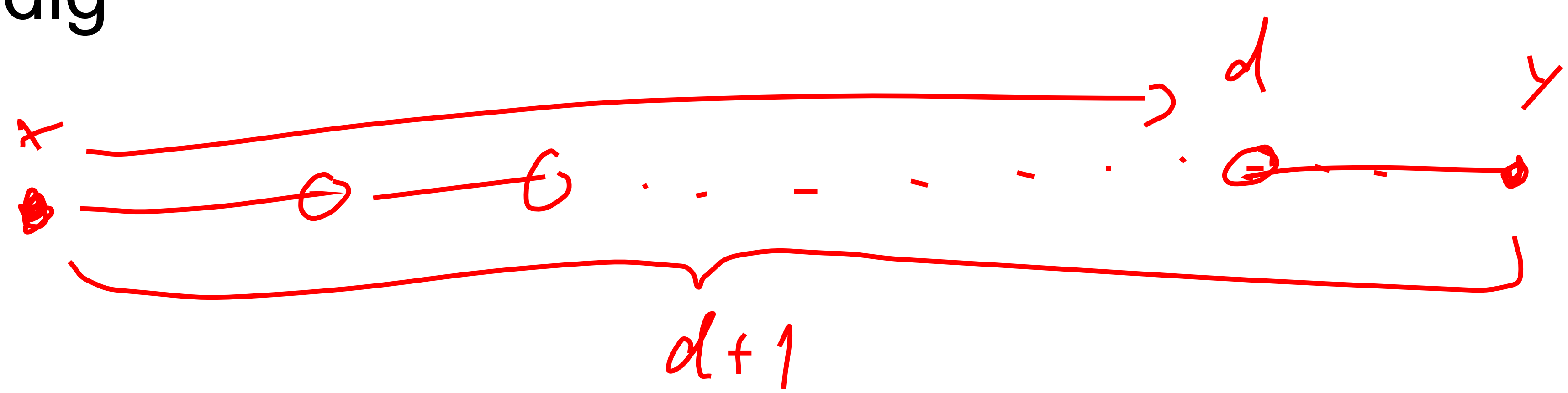
x — 1 Bit Unterschied — u — 1 Bit Unterschied — s — 1 Bit Unterschied — y

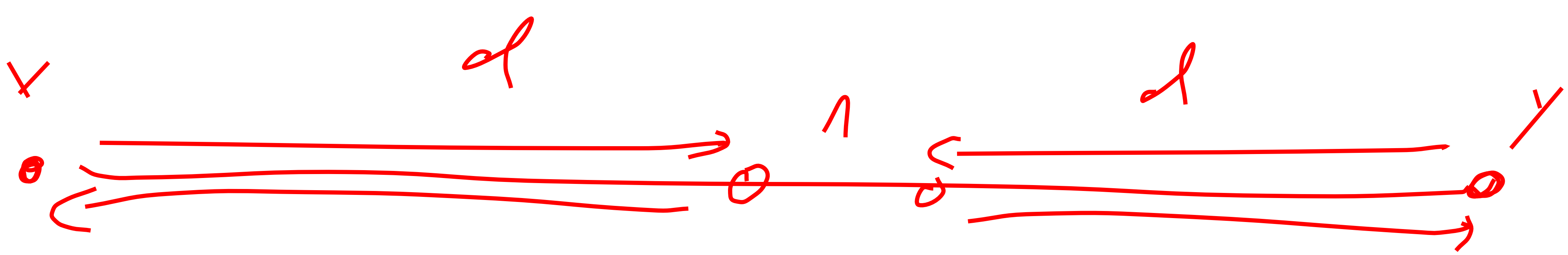
  - Falls u empfangen wird, sind folgende Fälle denkbar:
    - x wurde gesendet und mit 1 Bit-Fehler empfangen
    - y wurde gesendet und mit 2 Bit-Fehlern empfangen
    - Etwas anderes wurde gesendet und mit mindestens 2 Bit-Fehlern empfangen
  - Es ist also wahrscheinlicher, dass x gesendet wurde, statt 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 2d+1 in der Menge der legalen Frames notwendig





Codebücher und Kodierungen

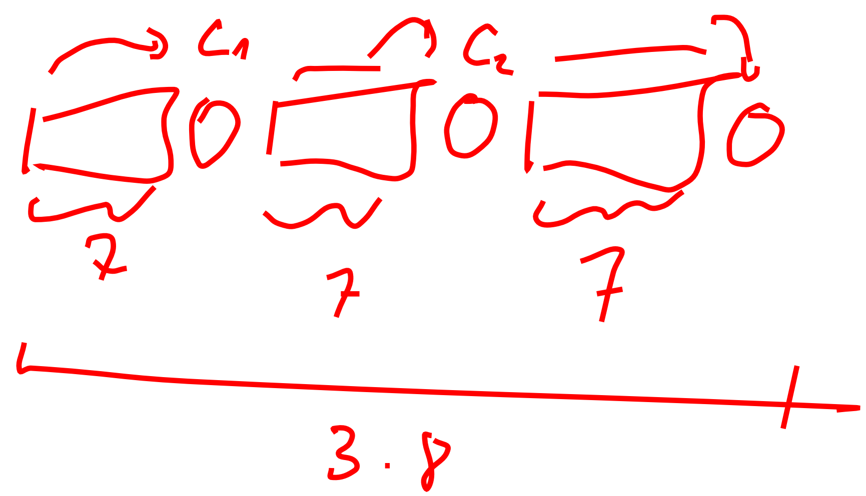
- Die Menge der legalen Frames $S \in \{0,1\}^n$ wird **das Code-Buch** oder einfach Kodierung genannt.
  - Die Rate R eines Codes S ist definiert als
    - Die Rate charakterisiert die Effizienz des Codes

$$R_S = \frac{\log |S|}{n}$$

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

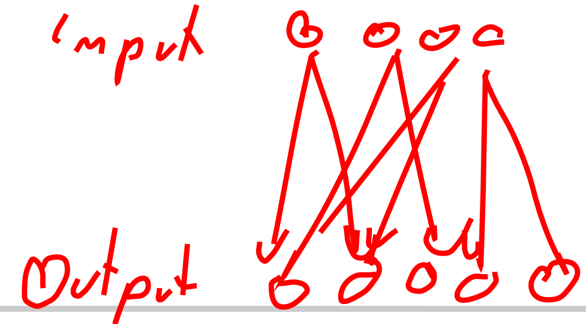
  - Die Distanz δ des Codes S ist definiert als
    - charakterisiert die Fehlerkorrektur oder Fehlererkennungsmöglichkeiten

$$\delta_S = \frac{d(S)}{n}$$

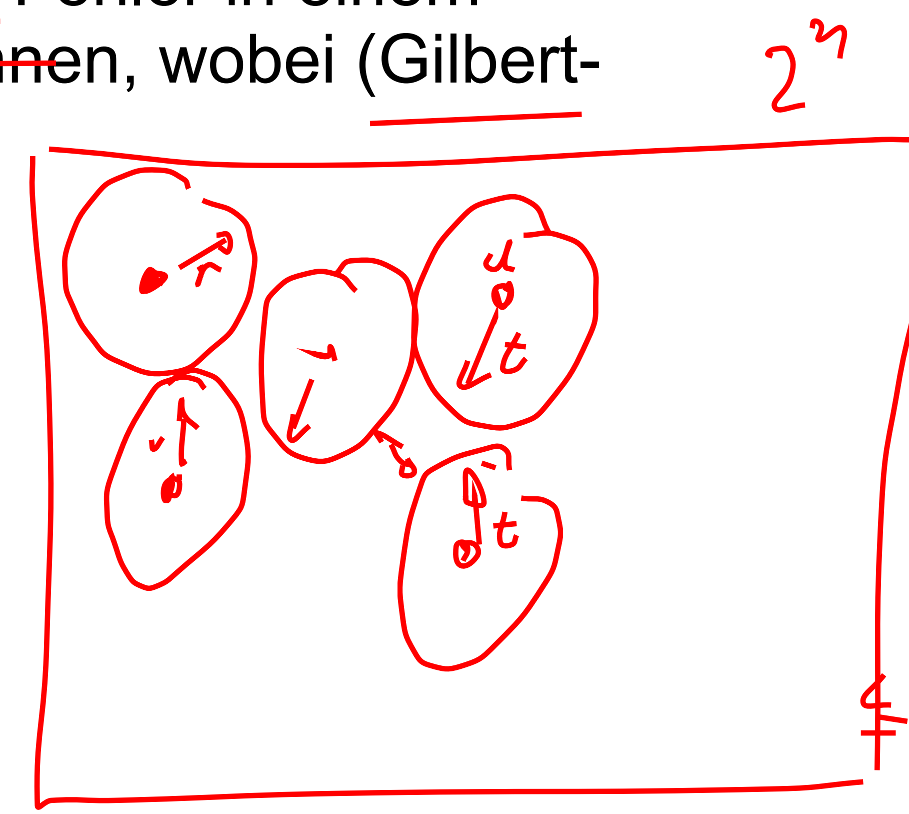


- Gute Codes haben hohe Raten und hohe Distanz
  - Beides lässt sich nicht zugleich optimieren

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~~ korrigieren, wobei (Gilbert-Varshamov-Schranke):



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

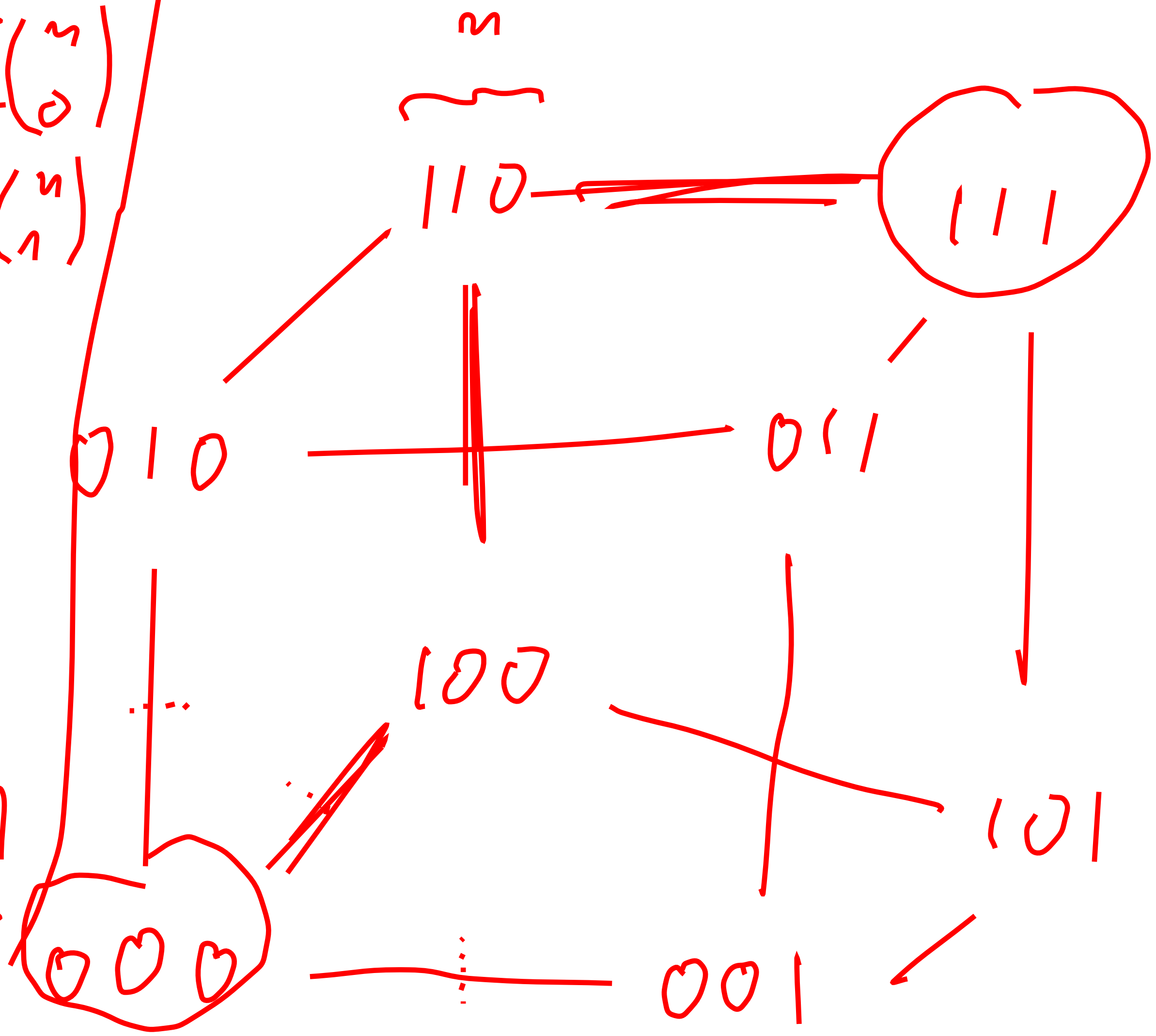
  - Das ist eine theoretische obere Schranke
- Beispiele
  - Bose Chaudhuri Hocquenghem (BCH) Codes
    - basierend auf Polynomen über endlichen Körpern (Galois-Körpern)
  - Reed Solomon Codes
    - Spezialfall nichtbinärer BCH-Codes

$$\frac{2^n}{\sum_{i=0}^{t}\binom{n}{i} = \mathrm{Vol}(\text{Kugel})} \geq 2^k = \#\text{Symbol}$$

n-dimensionale Hyperwürfel

00000

d	#
0	$1 = \binom{n}{0}$
1	$n = \binom{n}{1}$
2	$\binom{n}{2}$
3	$\binom{n}{3}$
t	$\binom{n}{t}$
n	$\binom{n}{n} = 1$



2^n Ecken

Grad n : (Nachbarn)

$\frac{1}{2} n \cdot 2^n$: # Kanten

Durchmesser ist n

$\binom{n}{2} = \frac{n(n-1)}{2}$

Wie viele Knoten sind im Abstand d

01101101

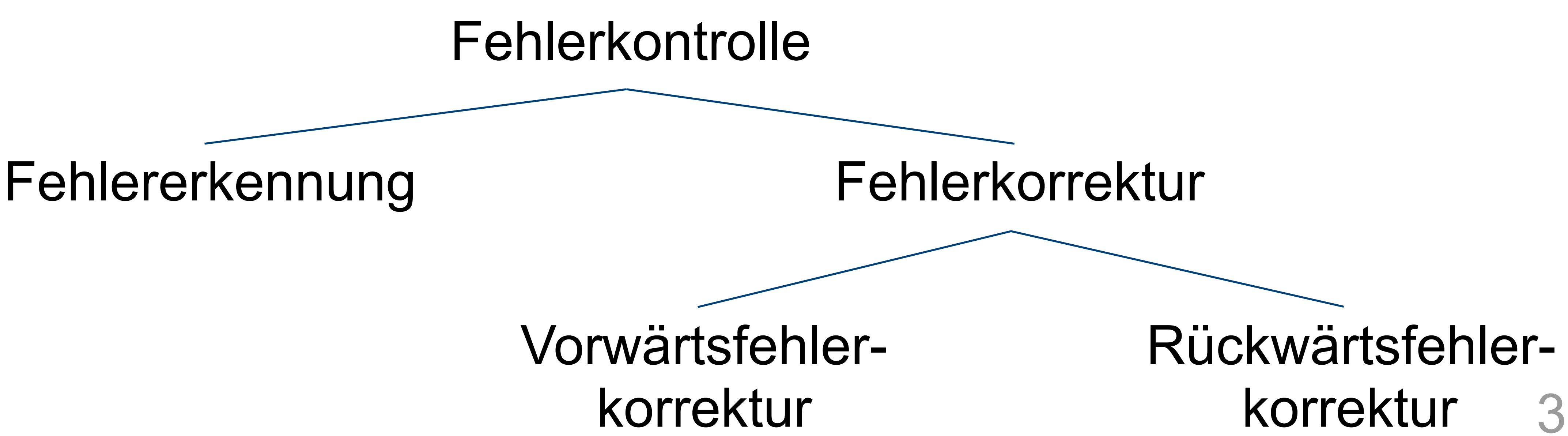
CoNe Freiburg

UNI FREIBURG

Fehlerkontrolle

CoNe Freiburg

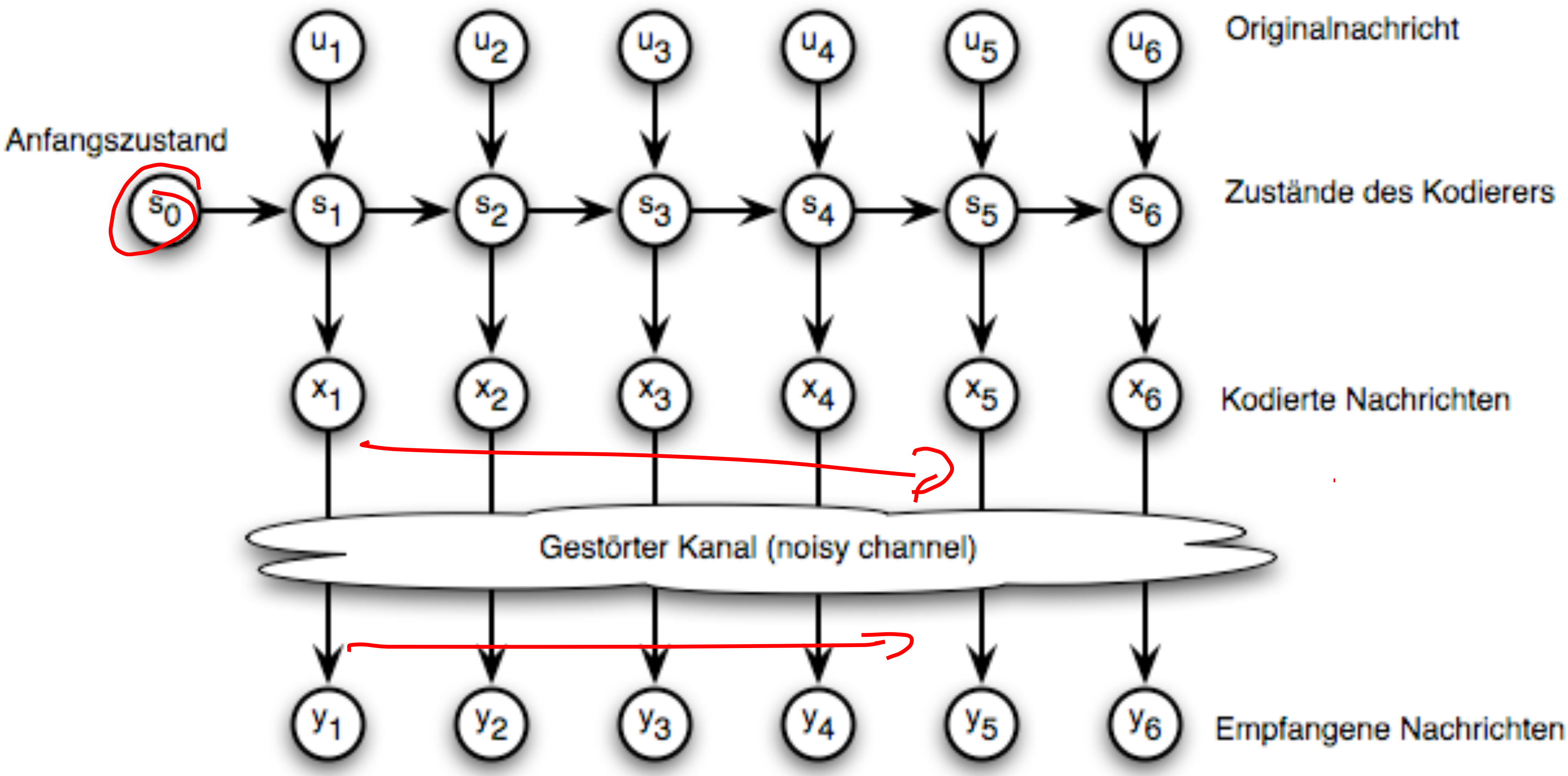
- Zumeist gefordert von der Vermittlungsschicht
  - Mit Hilfe der Frames
- Fehlererkennung
  - Gibt es fehlerhaft übertragene Bits?
- Fehlerkorrektur
  - Behebung von Bitfehlern
  - Vorwärtsfehlerkorrektur (Forward Error Correction)
    - Verwendung von redundanter Kodierung, die es ermöglicht Fehler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



UNI FREIBURG

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.



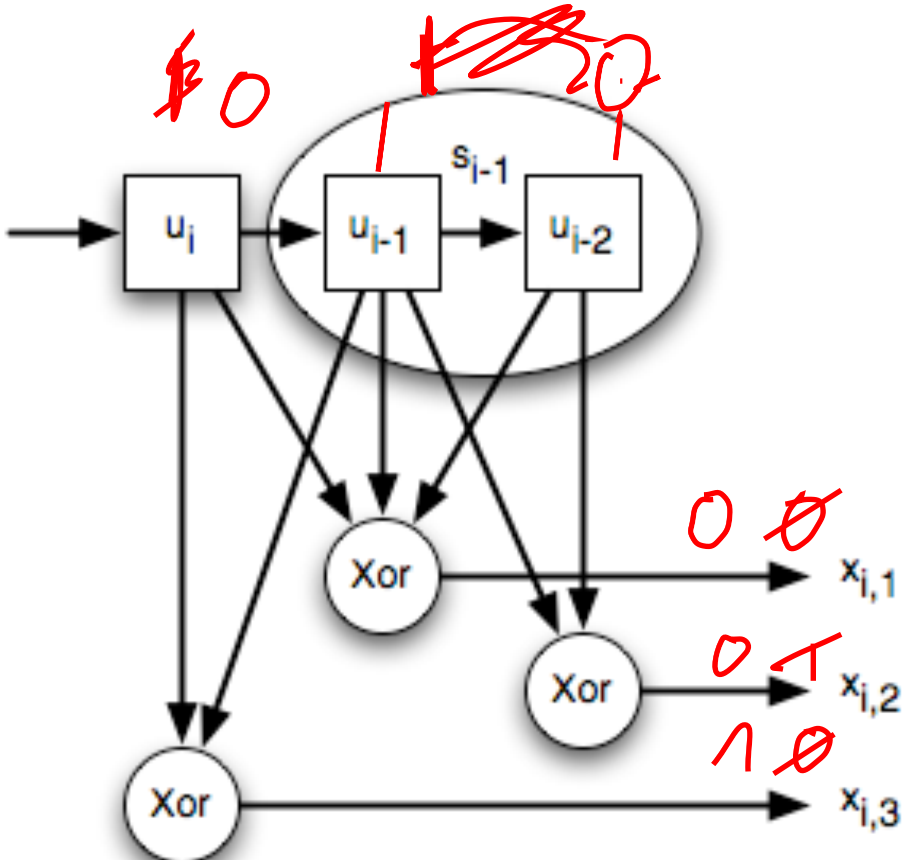
CoNe Freiburg

Beispiel

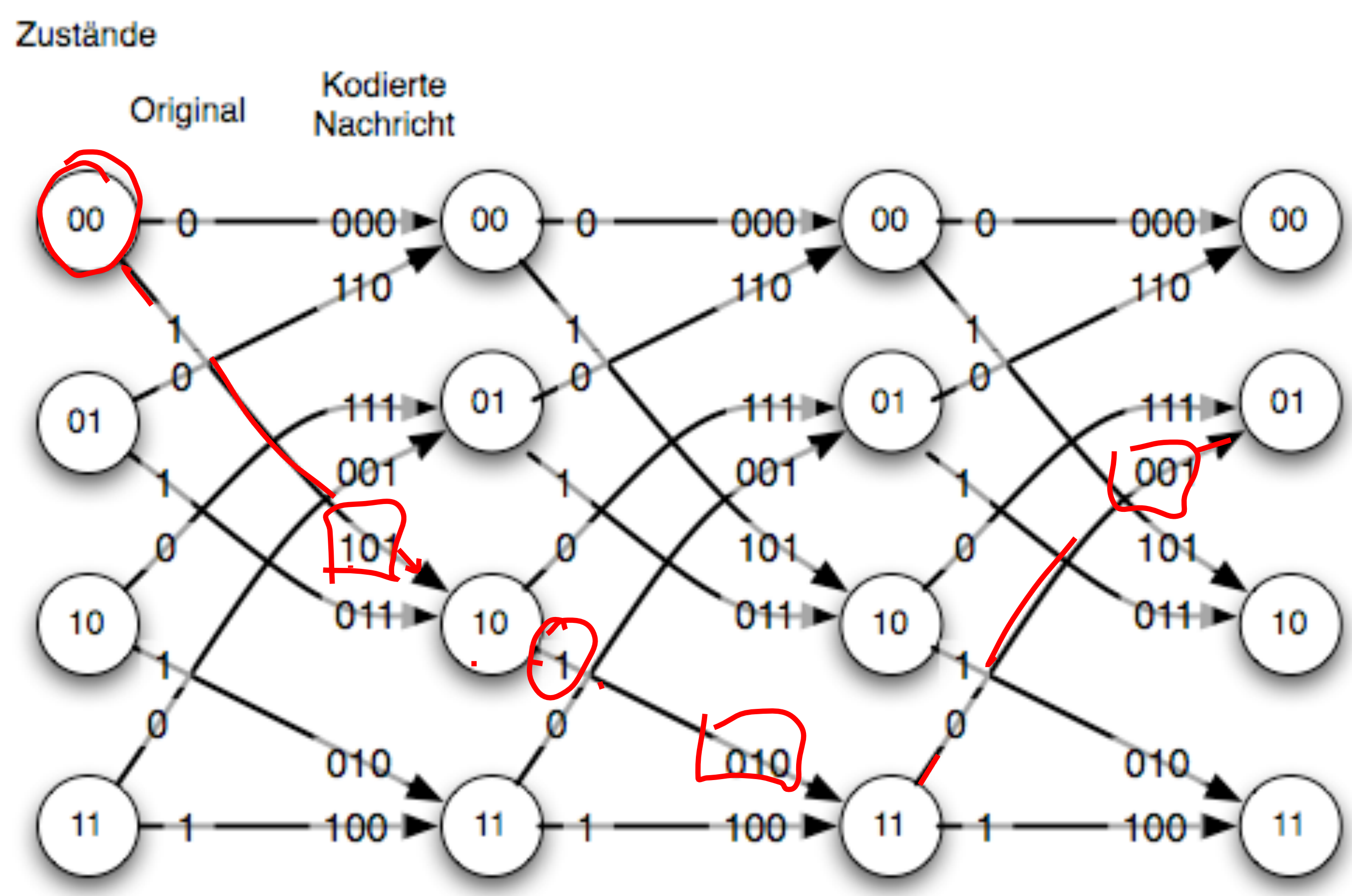
000

Trellis

Faltungs-Kodierer



Trellis-Diagramm



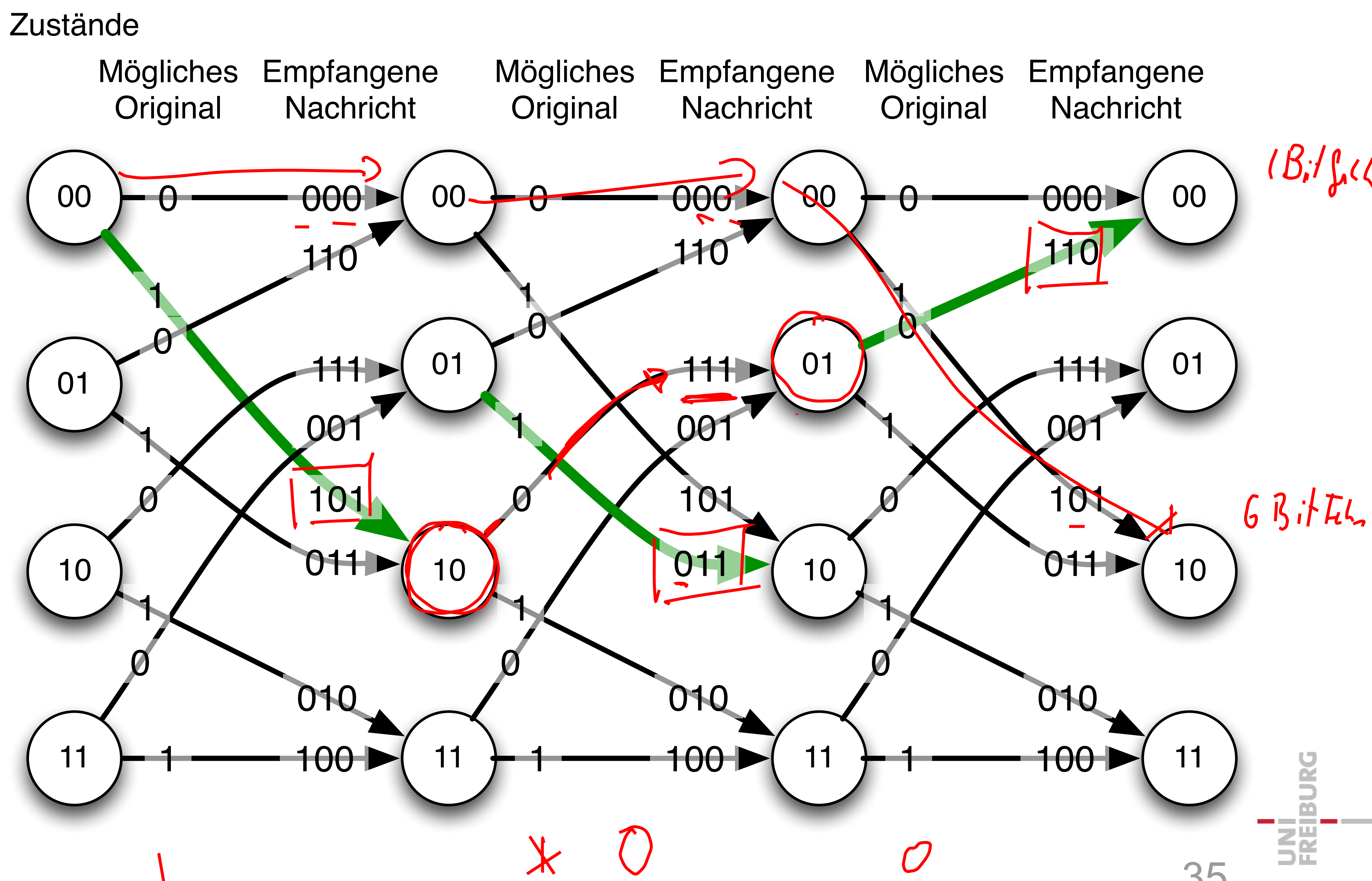
101 010

UNI FREIBURG

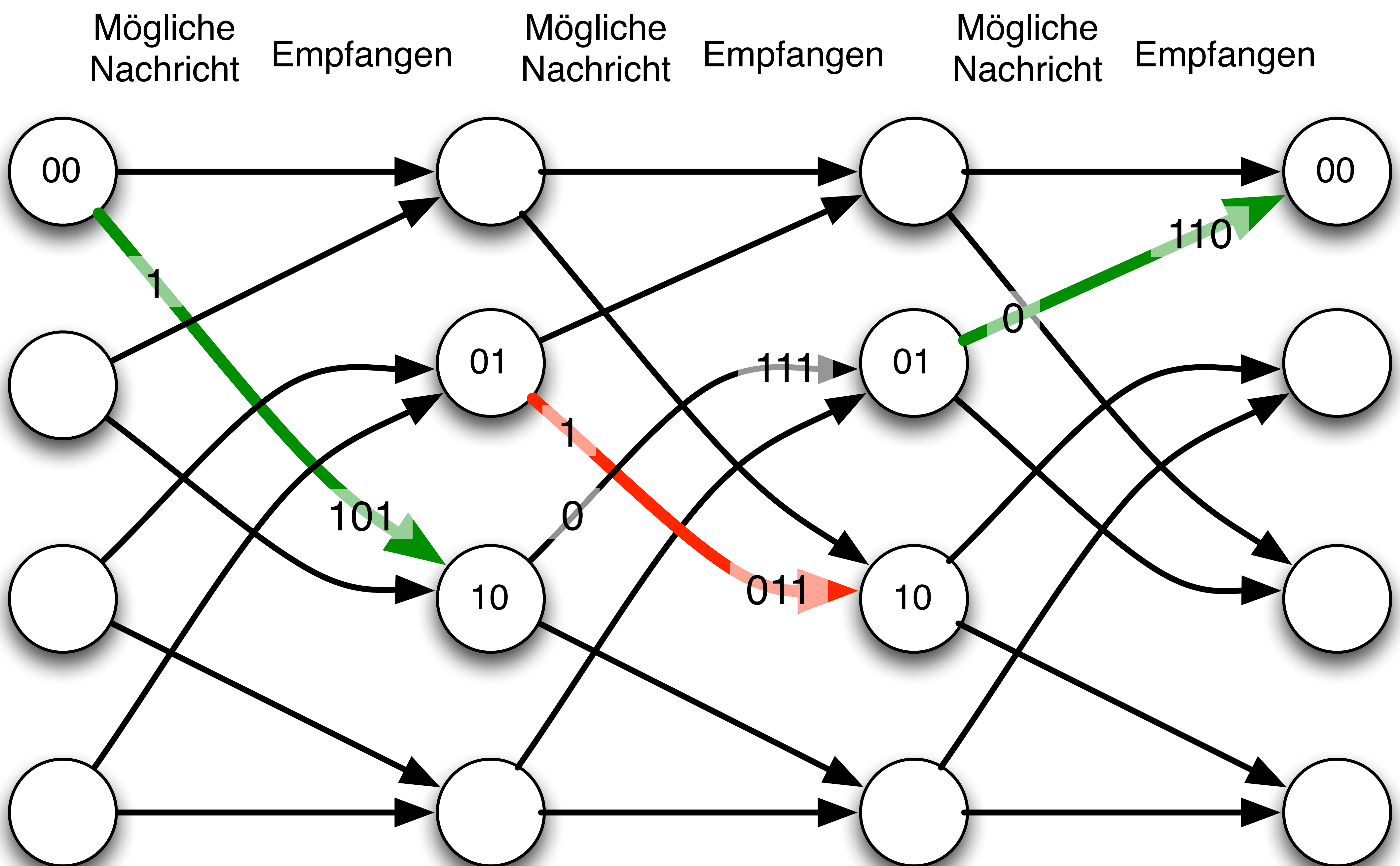
Dekodierung der Faltungs-Codes: Algorithmus von Viterbi

- 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

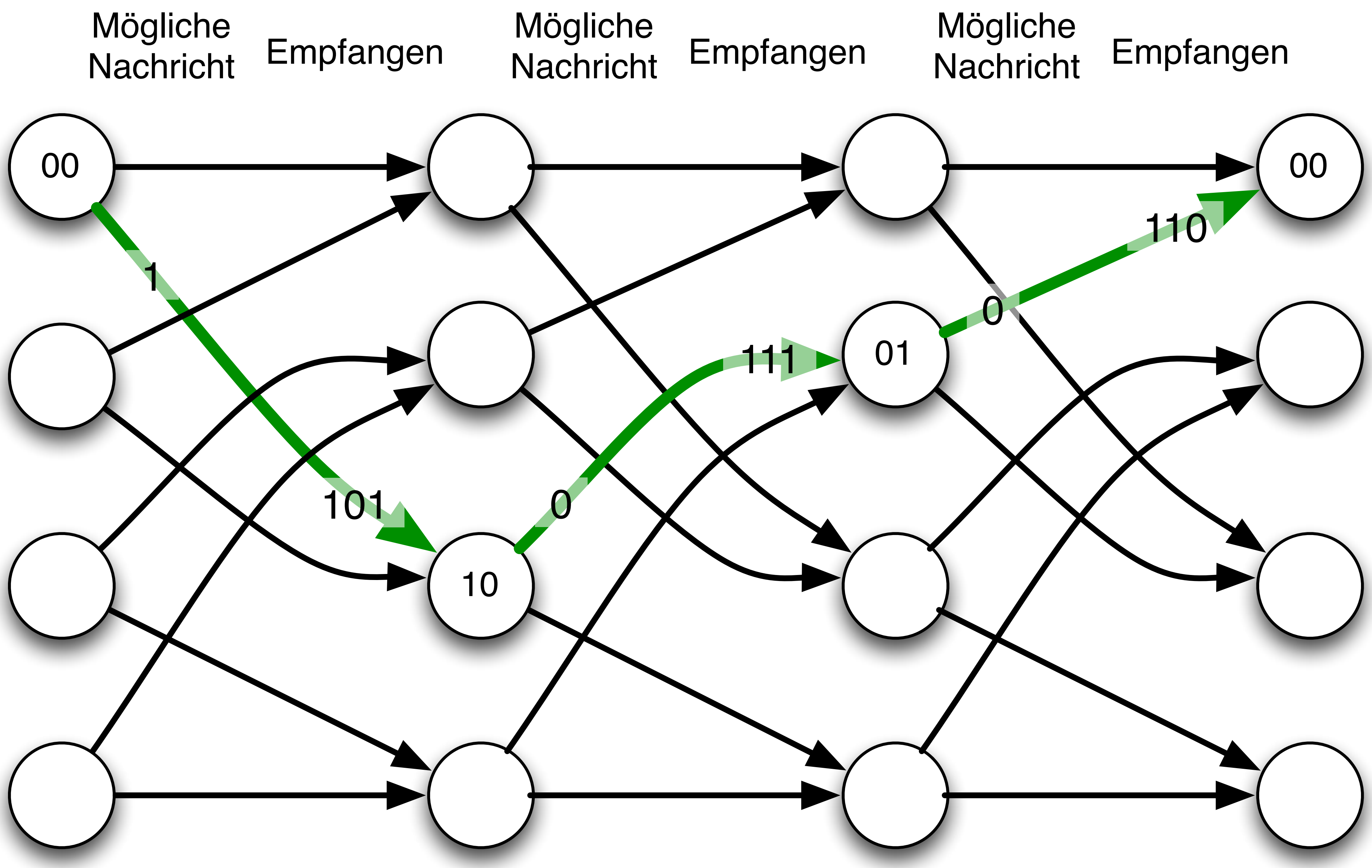
Dekodierung (I)



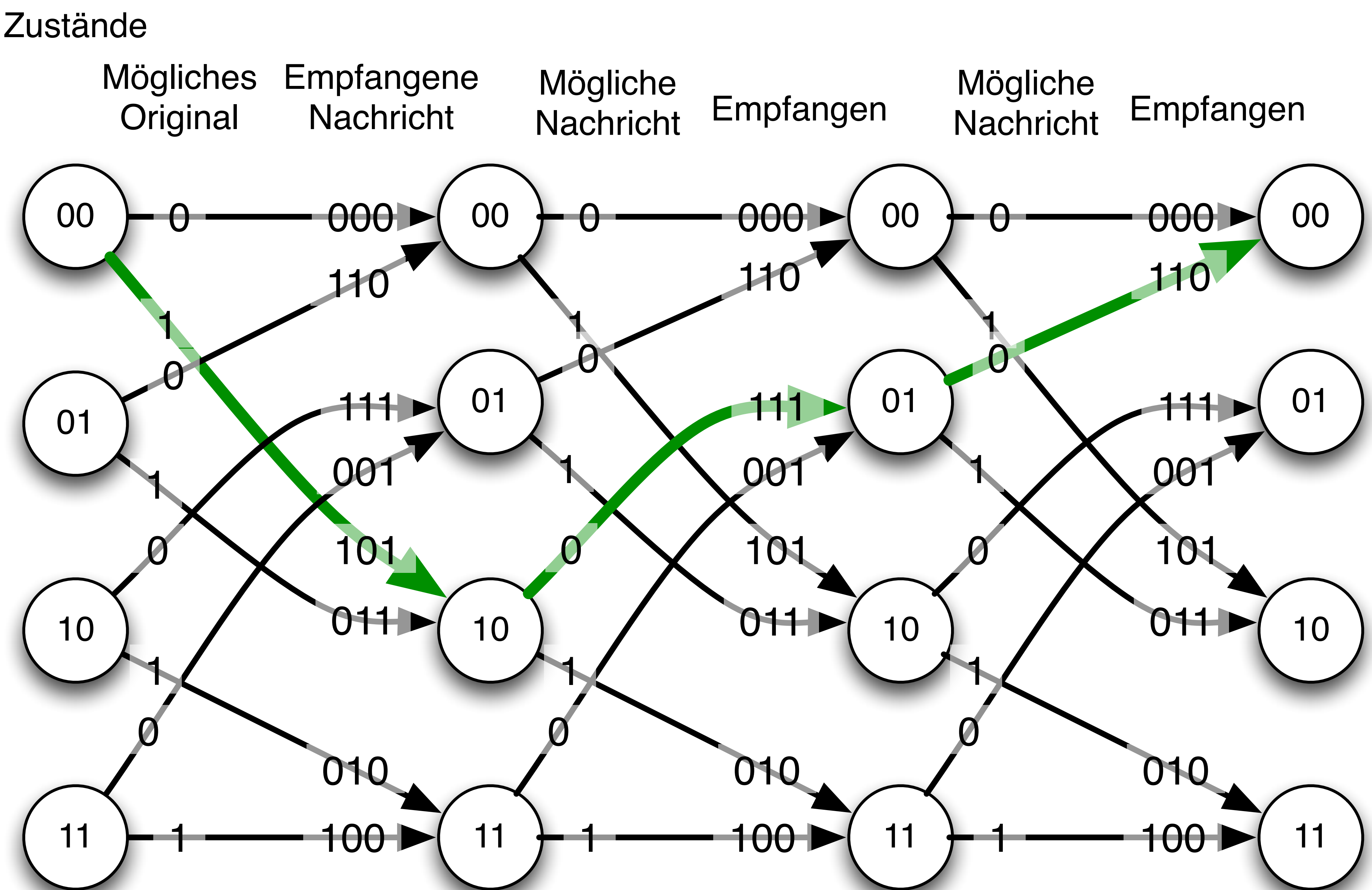
Dekodierung (II)



Dekodierung (III)

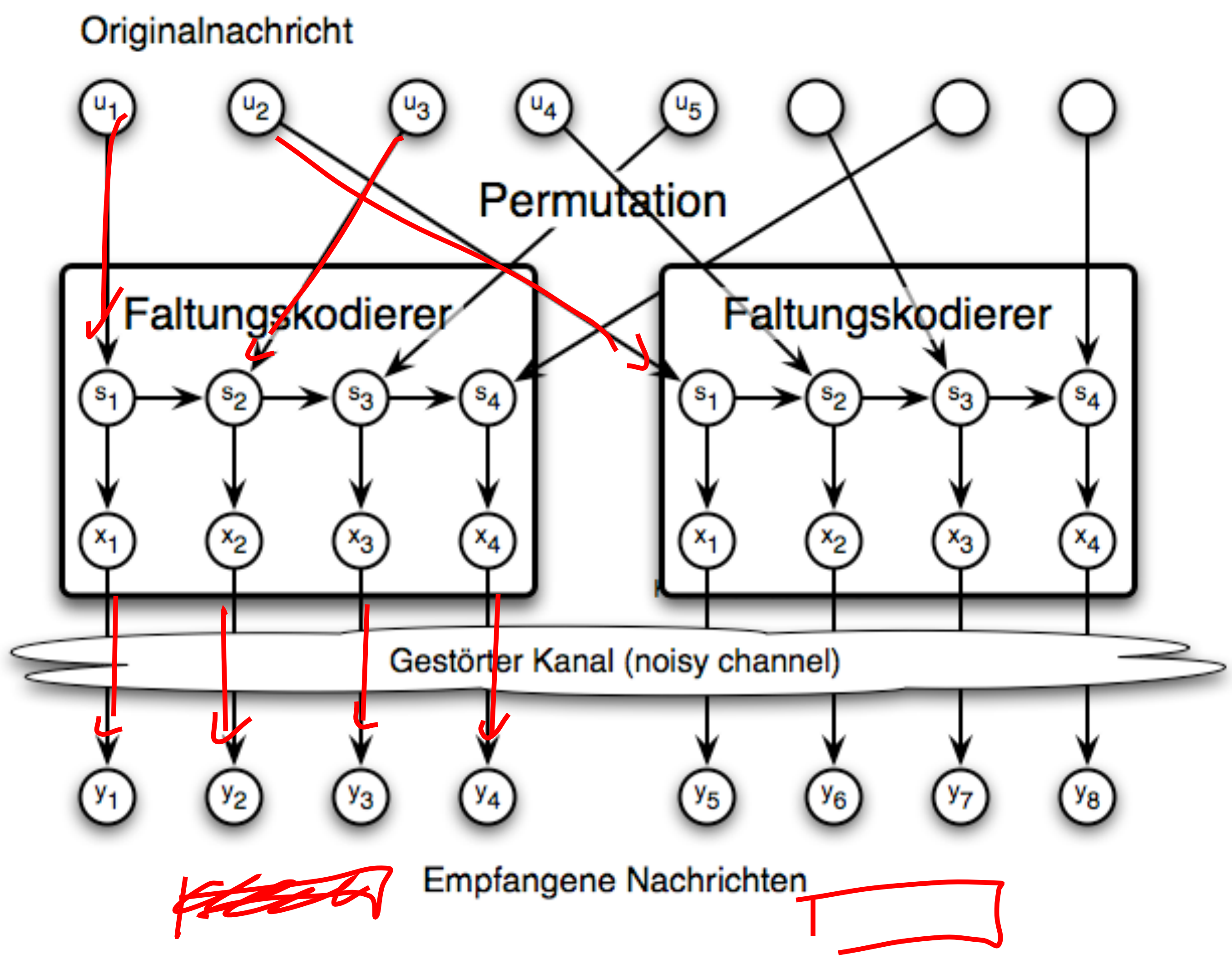


Dekodierung (IV)



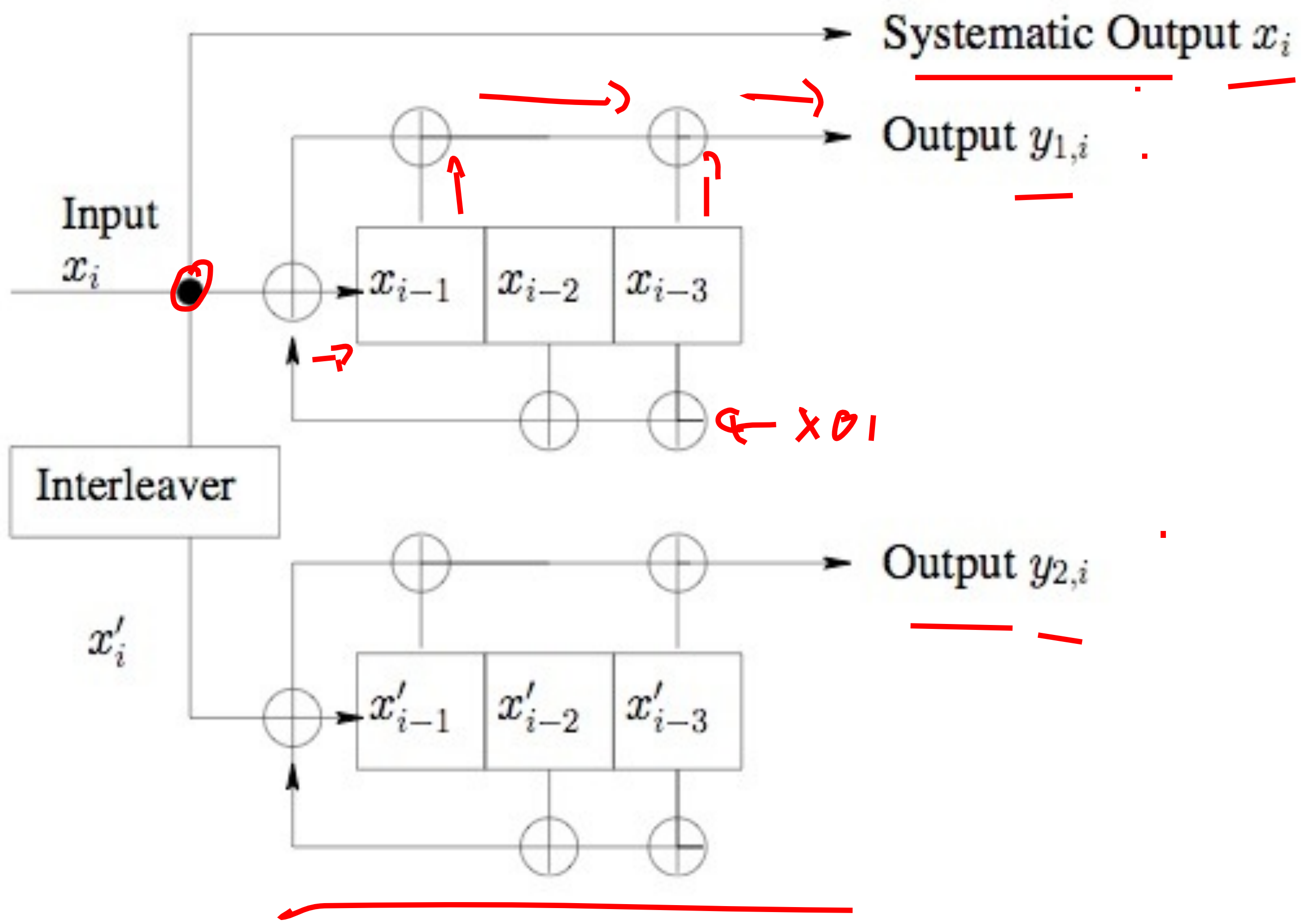
Turbo-Codes

- 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

12 8 4 11 7 3 10 6 2 9 5 1

- 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

UNI FREIBURG

Fehlererkennung: CRC

- Effiziente Fehlererkennung: Cyclic Redundancy Check (CRC)
- Praktisch häufig verwendeter Code
  - Hoher Fehlererkennungsrate
  - Effizient in Hardware umsetzbar
- Beruht auf Polynomarithmetik im Restklassenring Z_2
  - Zeichenketten sind Polynome
  - Bits sind Koeffizienten des Polynoms

Boole. $\{0,1\}$

XOR

$0+0=0$

$1+0=1$

$0+1=1$

$1+1=0$

And

$0\cdot 0=0$

$0\cdot 1=0$

$1\cdot 0=0$

$1\cdot 1=1$