

# 3. TUTORIUM

DATENKOMMUNIKATION UND SICHERHEIT

TUTORIUMSGRUPPE 18

MATTHIS FRANZGROTE

COMSYS

RWTH AACHEN

19.05.2021

- 1 Aufgabe 3.1: Abtasttheorem
- 2 Aufgabe 3.2: Code Division Multiplexing
- 3 Aufgabe 3.3: Bit-Stuffing
- 4 Aufgabe 3.4: CRC
- 5 Aufgabe 3.5: Fehlerkorrigierende Codes

## **AUFGABE 3.1: ABSTASTTHEOREM**

### Aufgabe

Mit der Digitalisierung des Telefonnetzes (durch ISDN) wurde festgelegt, dass zur digitalen Telefonie ein Kanal mit 64 kBit/s benötigt wird.  
Auf welchen Annahmen basiert die Bitrate von 64 kBit/s?

### Aufgabe

Mit der Digitalisierung des Telefonnetzes (durch ISDN) wurde festgelegt, dass zur digitalen Telefonie ein Kanal mit 64 kBit/s benötigt wird.  
Auf welchen Annahmen basiert die Bitrate von 64 kBit/s?

### Abtasttheorem

$f_A$ : Abtastfrequenz,  $f_{\text{Grenz}}$ : Grenzfrequenz

$$f_A \geq 2 \cdot f_{\text{Grenz}}$$

### Aufgabe

Mit der Digitalisierung des Telefonnetzes (durch ISDN) wurde festgelegt, dass zur digitalen Telefonie ein Kanal mit 64 kBit/s benötigt wird.  
Auf welchen Annahmen basiert die Bitrate von 64 kBit/s?

### Abtasttheorem

$f_A$ : Abtastfrequenz,  $f_{\text{Grenz}}$ : Grenzfrequenz

$$f_A \geq 2 \cdot f_{\text{Grenz}}$$

- Wichtigste Frequenzen der menschlichen Sprache als 300 Hz - 3,4 kHz definiert (Bandbreite 3,1 kHz)
- $\Rightarrow f_A \geq 6800 \text{ Hz}$

### Aufgabe

Mit der Digitalisierung des Telefonnetzes (durch ISDN) wurde festgelegt, dass zur digitalen Telefonie ein Kanal mit 64 kBit/s benötigt wird.  
Auf welchen Annahmen basiert die Bitrate von 64 kBit/s?

- $f_A \geq 6800 \text{ Hz}$
- Wegen Störfaktoren und Imperfektionen der Hardware auf  $f_A = 8000 \text{ Hz}$  geeinigt
- 256 Signalstufen (recht arbiträr gewählt), also 8 Bit pro Schritt

$\Rightarrow 8000 \text{ Hz} \cdot 8 \text{ Bit} = \underline{64 \text{ kBit/s}}$

↳ Nyquist



### Aufgabe

Mit der Digitalisierung des Telefonnetzes (durch ISDN) wurde festgelegt, dass zur digitalen Telefonie ein Kanal mit 64 kBit/s benötigt wird.

Hängt diese Bitrate auch mit dem verwendeten Medium (Twisted Pair) zusammen?



### Aufgabe

Mit der Digitalisierung des Telefonnetzes (durch ISDN) wurde festgelegt, dass zur digitalen Telefonie ein Kanal mit 64 kBit/s benötigt wird.

Hängt diese Bitrate auch mit dem verwendeten Medium (Twisted Pair) zusammen?

- In o.g. Berechnung nicht (nur beachtung des Frequenzbereichs der Stimme und nötigen Signalstufen)
- In der Realität natürlich schon: **Shannon-Theorem**
- Dämpfung und Störfaktoren auf der Leitung beschränken die erreichbare Datenrate

## **AUFGABE 3.2: CODE DIVISION MULTIPLEXING**

# MULTIPLEXING

Multiplexing: Aufteilung des Mediums auf mehrere Sender

- Frequenz
- Code
- Raum
- Zeit



Multiplexing: Aufteilung des Mediums auf mehrere Sender

- Zeitmultiplex

Multiplexing: Aufteilung des Mediums auf mehrere Sender

- Zeitmultiplex
- Frequenzmultiplex

Multiplexing: Aufteilung des Mediums auf mehrere Sender

- Zeitmultiplex
- Frequenzmultiplex
- **Codemultiplex**

Multiplexing: Aufteilung des Mediums auf mehrere Sender

- Zeitmultiplex
- Frequenzmultiplex
- **Codemultiplex**
- Raummultiplex

Multiplexing: Aufteilung des Mediums auf mehrere Sender

- Zeitmultiplex
- Frequenzmultiplex
- **Codemultiplex**
- Raummultiplex

Codemultiplex:

- Geteiltes Frequenzband
- Chipping-Sequenz (Spreizcode) zur eindeutigen Codierung
- Orthogonale Codes um andere Sender nicht zu stören



# CODEMULTIPLEX

Datenbit  $x \in \{0, 1\}$

Spreizcode  $a = a_0 a_1 \dots a_{m-1}, a_i \in \{-1, +1\}$

$x$	gesendete Bitfolge
0	$a = a_0 a_1 \dots a_{m-1}$
1	$a = -a_0 - a_1 \dots - a_{m-1}$

$a = -1, 1, -1, 1$   
↑

$1, -1, 1, -1$

Codes  $a$  und  $b$  sind **orthogonal** gdw.

$$\sum_{i=0}^{m-1} a_i \cdot b_i = 0$$

$a = -1, 1, -1$

$b = 1, 1, 1$

$$a \cdot b = -1 + 1 - 1 = -1 \neq 0$$

## AUFGABE 3.2 A)

### Aufgabe

Betrachten Sie in dieser Aufgabe Spreizcodes der Länge  $m = 4$ .  
Wie viele verschiedene Spreizcodes sind bei dieser Länge möglich?  
Sind alle Spreizcodes orthogonal zueinander?

$$q_0, q_1, q_2, q_3 \quad 2^4 = 16$$

### Aufgabe

Betrachten Sie in dieser Aufgabe Spreizcodes der Länge  $m = 4$ .  
Wie viele verschiedene Spreizcodes sind bei dieser Länge möglich?  
Sind alle Spreizcodes orthogonal zueinander?

$$2^m = 16 \text{ Codes}$$

### Aufgabe

Betrachten Sie in dieser Aufgabe Spreizcodes der Länge  $m = 4$ .  
Wie viele verschiedene Spreizcodes sind bei dieser Länge möglich?  
Sind alle Spreizcodes orthogonal zueinander?

$2^m = 16$  Codes

Nicht alle orthogonal, z.B.

$$a = (1, 1, 1, 1), b = (-1, -1, -1, -1)$$

$$a \cdot b = -4 \neq 0$$

## AUFGABE 3.2 B)

### Aufgabe

Betrachten Sie in dieser Aufgabe Spreizcodes der Länge  $m = 4$ .  
Geben Sie alle zu  $a = (1, 1, 1, 1)$  orthogonalen Spreizcodes an.

Orthogonale Codes sind genau in der Hälfte der Stellen gleich

$$\bullet (1, 1, -1, -1) \quad 1 \cdot 1 + 1 \cdot 1 - 1 \cdot 1 - 1 \cdot 1 = 0$$

$$\bullet (-1, -1, 1, 1)$$

$$\bullet (-1, 1, -1, 1)$$

$$\bullet (1, -1, 1, -1)$$

## AUFGABE 3.2 B)

### Aufgabe

Betrachten Sie in dieser Aufgabe Spreizcodes der Länge  $m = 4$ .  
Geben Sie alle zu  $a = (1, 1, 1, 1)$  orthogonalen Spreizcodes an.

Orthogonale Codes sind genau in der Hälfte der Stellen gleich

■  $(1, 1, -1, -1)$

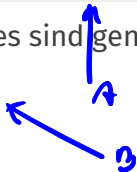
■  $(1, -1, 1, -1)$

■  $(1, -1, -1, 1)$

■  $(-1, 1, 1, -1)$

■  $(-1, 1, -1, 1)$

■  $(-1, -1, 1, 1)$



## AUFGABE 3.3: BIT-STUFFING

- Benutzung eines Flags zur markierung von Blockanfang und -ende
  - Auftreten des Flag im Nutzdatenteil problematisch
- ⇒ Verändere Nutzdaten an diesen Stellen nach Schema

000 | 10 1 0 0 1 1 | 000  
~~~~~



# BITSTUFFING

- Benutzung eines **Flags** zur markierung von Blockanfang und -ende
- Auftreten des Flag im Nutzdatenteil problematisch

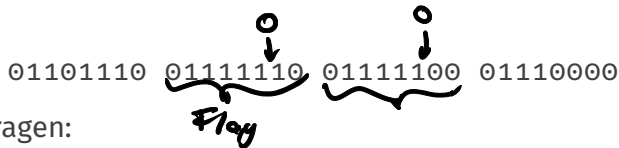
⇒ Verändere Nutzdaten an diesen Stellen nach Schema

Bsp. aus der Vorlesung:

- Flag: 01111110
- Stuffing: Nach 5 1en eine 0 einfügen (und entfernen!)

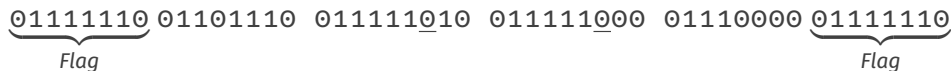
Nutzdaten:

01101110 01111110 01111110 01110000



Tatsächlich übertragen:

01111110 01101110 01111110 01111110 01110000 01111110



## AUFGABE 3.3 A)

### Aufgabe

Führen Sie anhand des folgenden Beispiels Bit-Stuffing durch:

Bitfolge: 1011 0001 0011 0110 0110 0101 1101 01

Flag: 1001101

Stuffing durch Einfügen einer 0

**Aber wo?**

## AUFGABE 3.3 A)

### Aufgabe

Führen Sie anhand des folgenden Beispiels Bit-Stuffing durch:

Bitfolge: 1011 0001 0011 0110 0110 0101 1101 01

Flag: 1001101

Stuffing durch Einfügen einer 0

### Aber wo?

Am besten so weit hinten wie möglich für geringeren Overhead

10011001

## AUFGABE 3.3 A)

### Aufgabe

Führen Sie anhand des folgenden Beispiels Bit-Stuffing durch:

Bitfolge: 1011 0001 0011 0110 0110 0101 1101 01

Flag: 1001101

Stuffing durch Einfügen einer 0

### Aber wo?

Am besten so weit hinten wie möglich für geringeren Overhead

10011001

⇒ 1011 0001 0011 00110 01100 0101 1101 01

## AUFGABE 3.4: CRC

## AUFGABE 3.4 A)

### Aufgabe

Gegeben ist die Bitsequenz 1011 1001, die mittels CRC gesichert übertragen werden soll. Berechnen Sie die Prüfsumme unter Verwendung des Generatorpolynoms  $G(x) = x^4 + x + 1$ .

Generatorpolynom (Divisor):  $G(x) = x^4 + x + 1 \rightarrow 10011$   
 $x^4 + x^3 + x^2 + x^1 + 1$

## AUFGABE 3.4 A)

Generatorpolynom (Divisor):  $G(x) = x^4 + x + 1 \Rightarrow 10011$

Dividend:  $1011 \ 1001 \ 0000 : 10011$

$$\begin{array}{r} \underline{10011} \\ 0010000 \\ \underline{10011} \\ 00011100 \\ \underline{10011} \\ 011110 \\ \underline{10011} \\ 011010 \\ \underline{10011} \\ 01001 \\ \hline \text{FCS} \end{array}$$

## AUFGABE 3.4 A)

Generatorpolynom (Divisor):  $G(x) = x^4 + x + 1 \Rightarrow 10011$

Dividend: 1011 1001 0000

$$\begin{array}{r} 101110010001 : 10011 \\ \underline{10011} \phantom{000000000000} \\ 100000 \phantom{000000000000} \\ \underline{10011} \phantom{000000000000} \\ 11100 \phantom{000000000000} \\ \underline{10011} \phantom{000000000000} \\ 11110 \phantom{000000000000} \\ \underline{10011} \phantom{000000000000} \\ 11010 \phantom{000000000000} \\ \underline{10011} \phantom{000000000000} \\ \underline{1001} \phantom{000000000000} \end{array}$$

7001

Übertragung OK  
0 → Fehlerhaft

FCS: 1001  $\Rightarrow$  zu übertragene Bitsequenz: 1011 1001 1001



### Aufgabe

Warum hat sich CRC zur Fehlerprüfung in der Datenkommunikation durchgesetzt?  
Welche Vorteile bietet sie verglichen mit z.B. der Verwendung eines Paritätsbit.

### Aufgabe

Warum hat sich CRC zur Fehlerprüfung in der Datenkommunikation durchgesetzt?  
Welche Vorteile bietet sie verglichen mit z.B. der Verwendung eines Paritätsbit.

- CRC ist sehr leistungsfähig: Fehlerbursts der Länge  $n$  zuverlässig erkannt ( $n$  = Grad des Generatorpolynoms)

### Aufgabe

Warum hat sich CRC zur Fehlerprüfung in der Datenkommunikation durchgesetzt?  
Welche Vorteile bietet sie verglichen mit z.B. der Verwendung eines Paritätsbit.

- CRC ist sehr leistungsfähig: Fehlerbursts der Länge  $n$  zuverlässig erkannt ( $n$  = Grad des Generatorpolynoms)
- (Längere Fehlerbursts zu großem Prozentsatz auch erkannt)

### Aufgabe

Warum hat sich CRC zur Fehlerprüfung in der Datenkommunikation durchgesetzt?  
Welche Vorteile bietet sie verglichen mit z.B. der Verwendung eines Paritätsbit.

- CRC ist sehr leistungsfähig: Fehlerbursts der Länge  $n$  zuverlässig erkannt ( $n$  = Grad des Generatorpolynoms)
- (Längere Fehlerbursts zu großem Prozentsatz auch erkannt)
- **Welche Fehlermuster können nicht erkannt werden?**

### Aufgabe

Warum hat sich CRC zur Fehlerprüfung in der Datenkommunikation durchgesetzt?  
Welche Vorteile bietet sie verglichen mit z.B. der Verwendung eines Paritätsbit.

- CRC ist sehr leistungsfähig: Fehlerbursts der Länge  $n$  zuverlässig erkannt ( $n$  = Grad des Generatorpolynoms)
  - (Längere Fehlerbursts zu großem Prozentsatz auch erkannt)
  - **Welche Fehlermuster können nicht erkannt werden?**
- ⇒ Vielfache des Generatorpolynoms, dort ist der Rest auch 0

### Aufgabe

Warum hat sich CRC zur Fehlerprüfung in der Datenkommunikation durchgesetzt?  
Welche Vorteile bietet sie verglichen mit z.B. der Verwendung eines Paritätsbit.

- CRC ist sehr leistungsfähig: Fehlerbursts der Länge  $n$  zuverlässig erkannt ( $n$  = Grad des Generatorpolynoms)
  - (Längere Fehlerbursts zu großem Prozentsatz auch erkannt)
  - **Welche Fehlermuster können nicht erkannt werden?**
- ⇒ Vielfache des Generatorpolynoms, dort ist der Rest auch 0
- Paritätsbits erkennen nur ungerade Anzahlen an falschen Bits

### Aufgabe

Warum hat sich CRC zur Fehlerprüfung in der Datenkommunikation durchgesetzt?  
Welche Vorteile bietet sie verglichen mit z.B. der Verwendung eines Paritätsbit.

- CRC ist sehr leistungsfähig: Fehlerbursts der Länge  $n$  zuverlässig erkannt ( $n$  = Grad des Generatorpolynoms)
  - (Längere Fehlerbursts zu großem Prozentsatz auch erkannt)
  - **Welche Fehlermuster können nicht erkannt werden?**
- ⇒ Vielfache des Generatorpolynoms, dort ist der Rest auch 0
- Paritätsbits erkennen nur ungerade Anzahlen an falschen Bits
- ⇒ Viel mehr Fehlermuster unerkannt

## AUFGABE 3.4 C)

### Aufgabe

Kann man mittels einer CRC auch Fehler korrigieren?



### Aufgabe

Kann man mittels einer CRC auch Fehler korrigieren?

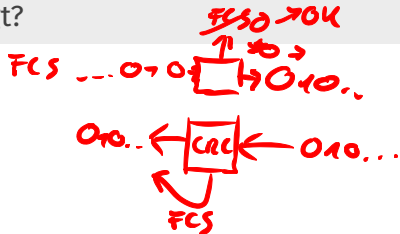
Nein? Doch!

- Bei geschickter Wahl von  $G$  und großem Overhead möglich
- ATM: ein Byte FCS zu vier Byte Header
- Jeder der 40 möglichen Einzelbitfehler erkennbar

## AUFGABE 3.4 D)

### Aufgabe

Warum wird die berechnete Prüfsumme nicht wie alle anderen Kontrollinformationen mit im Header übertragen, sondern an die Nutzdaten angehängt?



### Aufgabe

Warum wird die berechnete Prüfsumme nicht wie alle anderen Kontrollinformationen mit im Header übertragen, sondern an die Nutzdaten angehängt?

- Effizienz!
- Einfache Implementierung in Hardware (siehe Folien am Ende)
- FCS wird *on-the-fly* berechnet und dann angehängen
- Daten können währenddessen schon aufs Kabel gelegt werden
- Empfängerseite genauso

## **AUFGABE 3.5: FEHLERKORRIGIERENDE CODES**

## AUFGABE 3.5 A)

### Aufgabe

Sie wollen Nachrichten übertragen, welche nur aus den Zeichen "A" (000000), "B" (111000), "C" (000111) und "D" (111111) bestehen können.

Sie empfangen nun die im Folgenden dargestellten Bitfolgen. Welche Zeichen werden jeweils dekodiert? Lässt sich dies in jedem Fall eindeutig entscheiden?

## AUFGABE 3.5 A)

### Aufgabe

Sie wollen Nachrichten übertragen, welche nur aus den Zeichen "A" (000000), "B" (111000), "C" (000111) und "D" (111111) bestehen können.

Sie empfangen nun die im Folgenden dargestellten Bitfolgen. Welche Zeichen werden jeweils dekodiert? Lässt sich dies in jedem Fall eindeutig entscheiden?

■  $\overset{\downarrow}{1}00000 \rightarrow A$  *Wegung-Abstand: 1*

### Aufgabe

Sie wollen Nachrichten übertragen, welche nur aus den Zeichen "A" (000000), "B" (111000), "C" (000111) und "D" (111111) bestehen können.

Sie empfangen nun die im Folgenden dargestellten Bitfolgen. Welche Zeichen werden jeweils dekodiert? Lässt sich dies in jedem Fall eindeutig entscheiden?

■ 100000  $\Rightarrow$  A

### Aufgabe

Sie wollen Nachrichten übertragen, welche nur aus den Zeichen "A" (000000), "B" (111000), "C" (000111) und "D" (111111) bestehen können.

Sie empfangen nun die im Folgenden dargestellten Bitfolgen. Welche Zeichen werden jeweils dekodiert? Lässt sich dies in jedem Fall eindeutig entscheiden?

■ 100000  $\Rightarrow$  A

■ 001111



### Aufgabe

Sie wollen Nachrichten übertragen, welche nur aus den Zeichen "A" (000000), "B" (111000), "C" (000111) und "D" (111111) bestehen können.

Sie empfangen nun die im Folgenden dargestellten Bitfolgen. Welche Zeichen werden jeweils dekodiert? Lässt sich dies in jedem Fall eindeutig entscheiden?

■ 100000  $\Rightarrow$  A

■ 001111  $\Rightarrow$  C

## AUFGABE 3.5 A)

### Aufgabe

Sie wollen Nachrichten übertragen, welche nur aus den Zeichen "A" (000000), "B" (111000), "C" (000111) und "D" (111111) bestehen können.

Sie empfangen nun die im Folgenden dargestellten Bitfolgen. Welche Zeichen werden jeweils dekodiert? Lässt sich dies in jedem Fall eindeutig entscheiden?

■ 100000  $\Rightarrow$  A

■ 001111  $\Rightarrow$  C

■ 101111

### Aufgabe

Sie wollen Nachrichten übertragen, welche nur aus den Zeichen "A" (000000), "B" (111000), "C" (000111) und "D" (111111) bestehen können.

Sie empfangen nun die im Folgenden dargestellten Bitfolgen. Welche Zeichen werden jeweils dekodiert? Lässt sich dies in jedem Fall eindeutig entscheiden?

■ 100000  $\Rightarrow$  A

■ 001111  $\Rightarrow$  C

■ 101111  $\Rightarrow$  D

### Aufgabe

Sie wollen Nachrichten übertragen, welche nur aus den Zeichen "A" (000000), "B" (111000), "C" (000111) und "D" (111111) bestehen können.

Sie empfangen nun die im Folgenden dargestellten Bitfolgen. Welche Zeichen werden jeweils dekodiert? Lässt sich dies in jedem Fall eindeutig entscheiden?

- 100000  $\Rightarrow$  A
- 001111  $\Rightarrow$  C
- 101111  $\Rightarrow$  D
- 101010

## AUFGABE 3.5 A)

### Aufgabe

Sie wollen Nachrichten übertragen, welche nur aus den Zeichen "A" (000000), "B" (111000), "C" (000111) und "D" (111111) bestehen können.

Sie empfangen nun die im Folgenden dargestellten Bitfolgen. Welche Zeichen werden jeweils dekodiert? Lässt sich dies in jedem Fall eindeutig entscheiden?

■ 100000  $\Rightarrow$  A

■ 001111  $\Rightarrow$  C

■ 101111  $\Rightarrow$  D

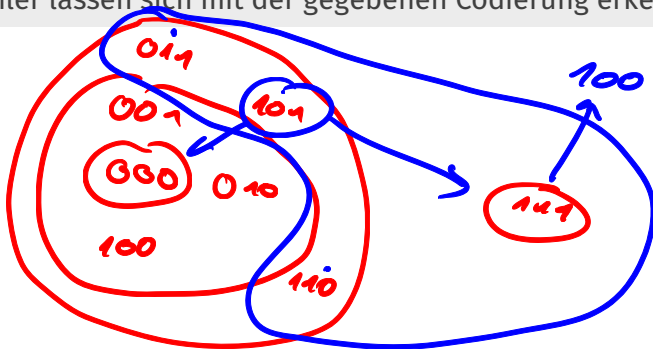
■ 101010  $\Rightarrow$  B

In allen Fällen nach geringstem Hamming-Abstand zuordenbar. (Annahme, dass Bitfehler selten sind und daher einfach zum nächsten dekodiert werden kann.)

## AUFGABE 3.5 B), c)

### Aufgabe

Sie wollen Nachrichten übertragen, welche nur aus den Zeichen "A" (000000), "B" (111000), "C" (000111) und "D" (111111) bestehen können. Wie viele Bitfehler lassen sich mit der gegebenen Codierung erkennen/korrigieren?



## AUFGABE 3.5 B), C)

### Aufgabe

Sie wollen Nachrichten übertragen, welche nur aus den Zeichen "A" (000000), "B" (111000), "C" (000111) und "D" (111111) bestehen können.  
Wie viele Bitfehler lassen sich mit der gegebenen Codierung erkennen/korrigieren?

### Hamming-Abstand (Folie III-24)

Hamming-Abstand  $D \leftarrow \text{minimale}$

$D \geq t + 1 \Rightarrow$  Code kann  $t$  Fehler erkennen

$D \geq 2t + 1 \Rightarrow$  Code kann  $t$  Fehler korrigieren

$D = 3$     2 Fehler erkennbar  
            1 Fehler korrigierbar

## AUFGABE 3.5 B), C)

### Aufgabe

Sie wollen Nachrichten übertragen, welche nur aus den Zeichen "A" (000000), "B" (111000), "C" (000111) und "D" (111111) bestehen können. Wie viele Bitfehler lassen sich mit der gegebenen Codierung erkennen/korrigieren?

### Hamming-Abstand (Folie III-24)

Hamming-Abstand  $D$

$D \geq t + 1 \Rightarrow$  Code kann  $t$  Fehler erkennen

$D \geq 2t + 1 \Rightarrow$  Code kann  $t$  Fehler korrigieren

Hamming-Abstand  $D = 3$

2 Fehler erkannt, 1 Fehler korrigiert



## AUFGABE 3.5 B), C)

### Aufgabe

Sie wollen Nachrichten übertragen, welche nur aus den Zeichen "A" (000000), "B" (111000), "C" (000111) und "D" (111111) bestehen können. Wie viele Bitfehler lassen sich mit der gegebenen Codierung erkennen/korrigieren?

Hamming-Abstand  $D = 3$   
2 Fehler erkannt, 1 Fehler korrigiert

Achtung! In a) konnte auch ein 2-Bit Fehler korrigiert werden, da er zufällig keinen 1-Bit-Fehler eines anderen Wortes darstellt (der Coderaum ist hier nicht vollständig ausgenutzt).

## AUFGABE 3.5 D)

### Aufgabe

Sie wollen nun fehlererkennende (CRC) und fehlerkorrigierende (FEC) Codes gemeinsam verwenden, um Fehler auf jeden Fall zu erkennen und soweit möglich auch korrigieren zu können. Dazu können Sie zwischen drei Szenarien wählen. Welches dieser Szenarien halten Sie für das Sinnvollste?



*Szenario 1* •



~~*Szenario 2*~~



*Szenario 3*

## AUFGABE 3.5 D)

- Szenario 2: CRC Falsch  $\Rightarrow$  FEC und Payload verworfen
- Alternative Implementierung: Erst Erkennung, dann Korrektur

## AUFGABE 3.5 D)

- Szenario 2: CRC Falsch  $\Rightarrow$  FEC und Payload verworfen
- Alternative Implementierung: Erst Erkennung, dann Korrektur
  - ▶ CRC OK  $\Rightarrow$  Trotzdem FEC, da CRC nicht alles erkennt

## AUFGABE 3.5 D)

- Szenario 2: CRC Falsch  $\Rightarrow$  FEC und Payload verworfen
- Alternative Implementierung: Erst Erkennung, dann Korrektur
  - ▶ CRC OK  $\Rightarrow$  Trotzdem FEC, da CRC nicht alles erkennt
  - ▶ CRC Falsch  $\Rightarrow$  Wird FEC funktionieren?  $\Rightarrow$  Einfach stur anwenden

## AUFGABE 3.5 D)

- Szenario 2: CRC Falsch  $\Rightarrow$  FEC und Payload verworfen
- Alternative Implementierung: Erst Erkennung, dann Korrektur
  - ▶ CRC OK  $\Rightarrow$  Trotzdem FEC, da CRC nicht alles erkennt
  - ▶ CRC Falsch  $\Rightarrow$  Wird FEC funktionieren?  $\Rightarrow$  Einfach stur anwenden $\Rightarrow$  CRC unnötig, da FEC so oder so angewandt wird

## AUFGABE 3.5 D)

- Szenario 2: CRC Falsch  $\Rightarrow$  FEC und Payload verworfen
- Alternative Implementierung: Erst Erkennung, dann Korrektur
  - ▶ CRC OK  $\Rightarrow$  Trotzdem FEC, da CRC nicht alles erkennt
  - ▶ CRC Falsch  $\Rightarrow$  Wird FEC funktionieren?  $\Rightarrow$  Einfach stur anwenden  
 $\Rightarrow$  CRC unnötig, da FEC so oder so angewandt wird
- Szenario 1: Fehler in CRC können korrigiert werden (vor fehlererkennung im Payload)

## AUFGABE 3.5 D)

- Szenario 2: CRC Falsch  $\Rightarrow$  FEC und Payload verworfen
- Alternative Implementierung: Erst Erkennung, dann Korrektur
  - ▶ CRC OK  $\Rightarrow$  Trotzdem FEC, da CRC nicht alles erkennt
  - ▶ CRC Falsch  $\Rightarrow$  Wird FEC funktionieren?  $\Rightarrow$  Einfach stur anwenden
  - $\Rightarrow$  CRC unnötig, da FEC so oder so angewandt wird
- Szenario 1: Fehler in CRC können korrigiert werden (vor fehlererkennung im Payload)
- Szenario 3: Fehler in CRC können korrigiert werden, dafür ist die CRC für weniger Bits zuständig, also mehr Fehler im Payload korrigierbar



## AUFGABE 3.5 D)

- Szenario 2: CRC Falsch  $\Rightarrow$  FEC und Payload verworfen
- Alternative Implementierung: Erst Erkennung, dann Korrektur
  - ▶ CRC OK  $\Rightarrow$  Trotzdem FEC, da CRC nicht alles erkennt
  - ▶ CRC Falsch  $\Rightarrow$  Wird FEC funktionieren?  $\Rightarrow$  Einfach stur anwenden
  - $\Rightarrow$  CRC unnötig, da FEC so oder so angewandt wird
- Szenario 1: Fehler in CRC können korrigiert werden (vor fehlererkennung im Payload)
- Szenario 3: Fehler in CRC können korrigiert werden, dafür ist die CRC für weniger Bits zuständig, also mehr Fehler im Payload korrigierbar

Praxis: Szenario 1 (WiFi: CRC auf Schicht 2, FEC zusammen mit Codierung auf Schicht 1)

ÜBUNGSBLATT 3 ABGABEFRIST:  
31.05.2021 18:00

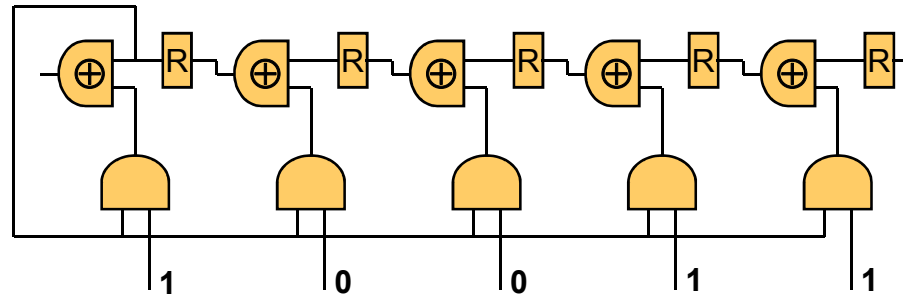
NÄCHSTES TUTORIUM:  
MITTWOCH 02.06.2021 12:30

# CRC-Berechnung: Schieberegister

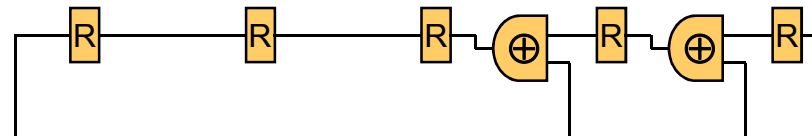
Implementation by Shift Registers:

- XOR for subtraction
- AND for applying subtraction:
  - first register = 0: no subtraction
  - first register = 1: subtraction

Generator polynomial  $x^4 + x + 1$ :



Simplified realization:

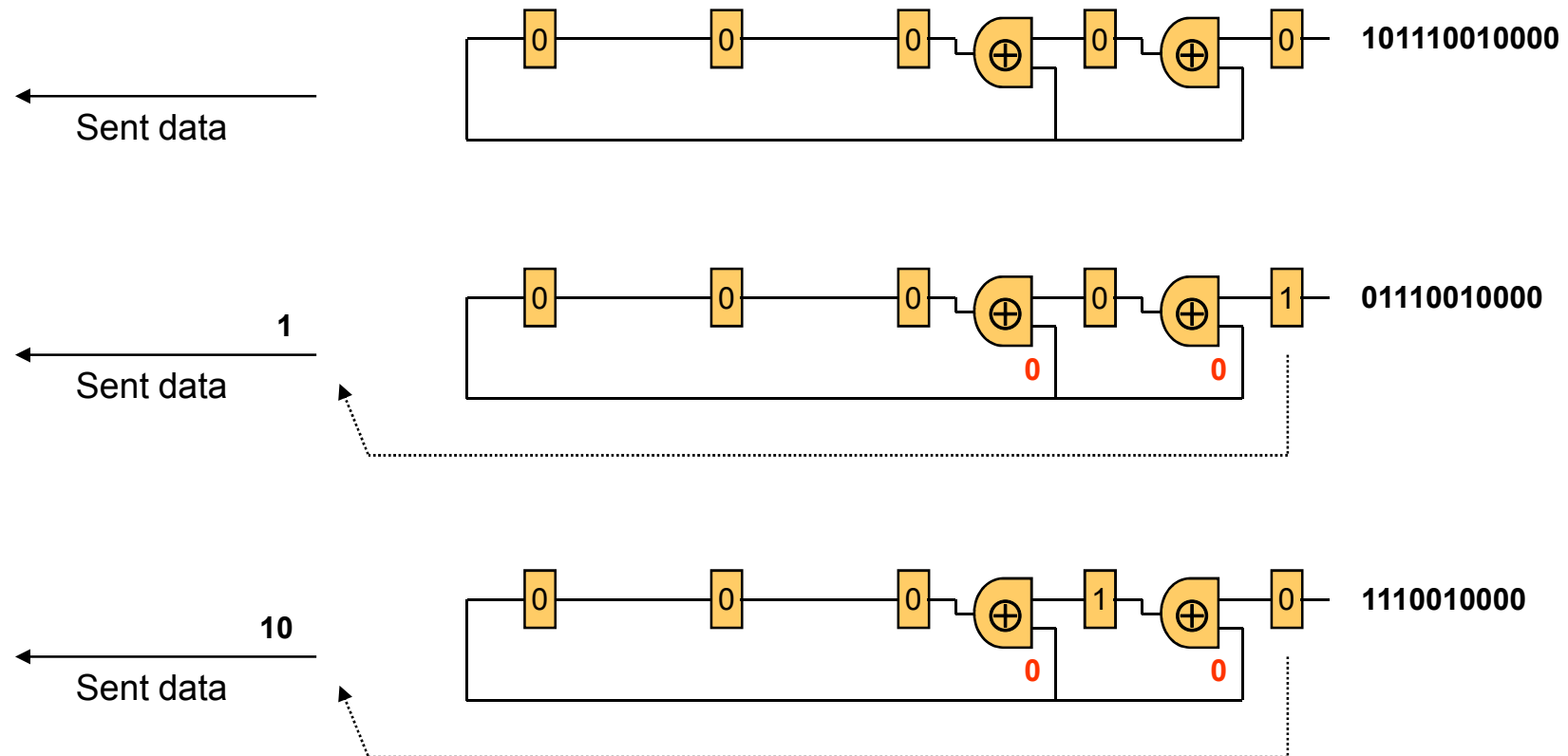


When no more input is given in the rightmost register, the other registers contain the CRC result.

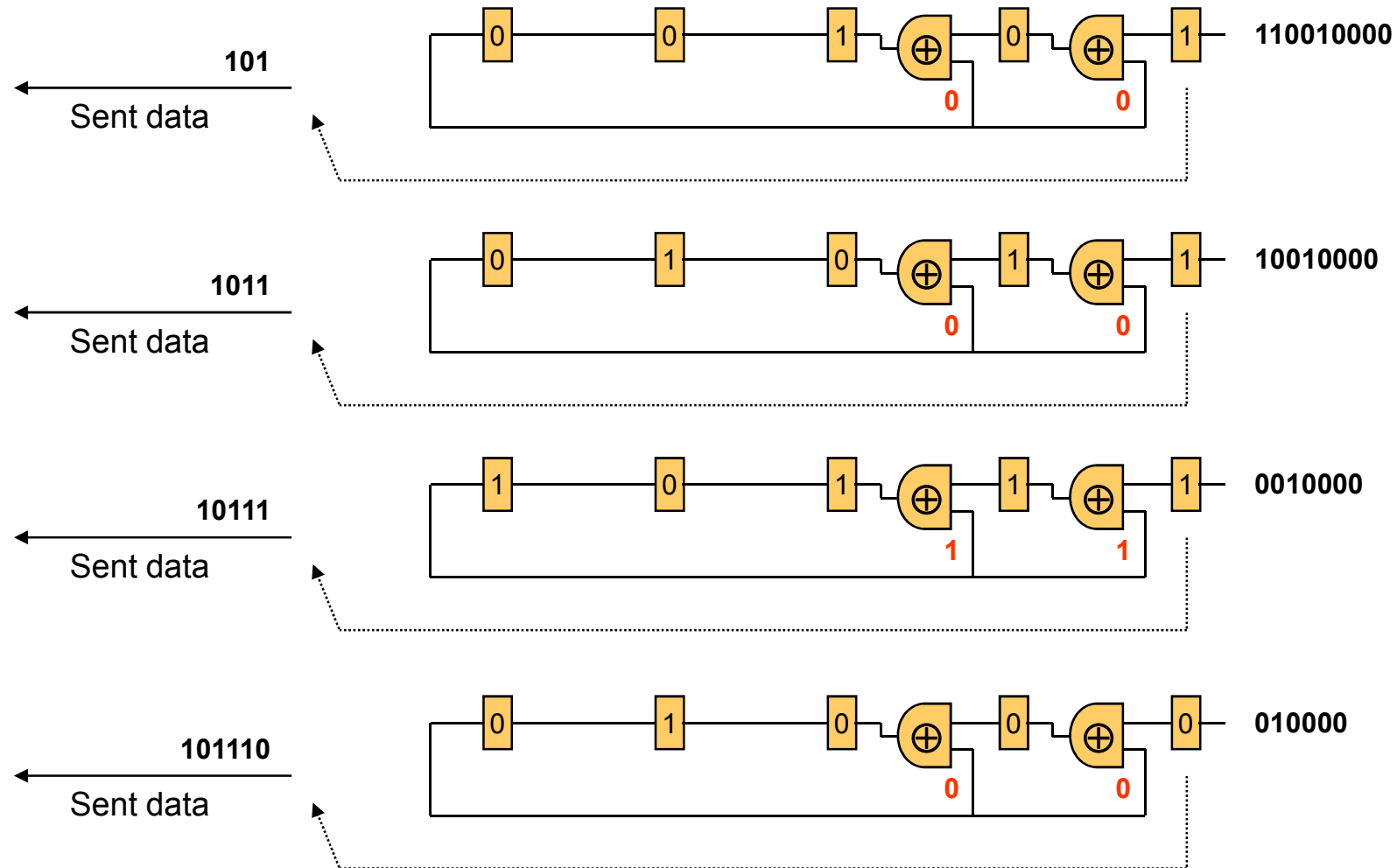
# CRC-Berechnung: Schieberegister

Data to be transmitted: 10111001

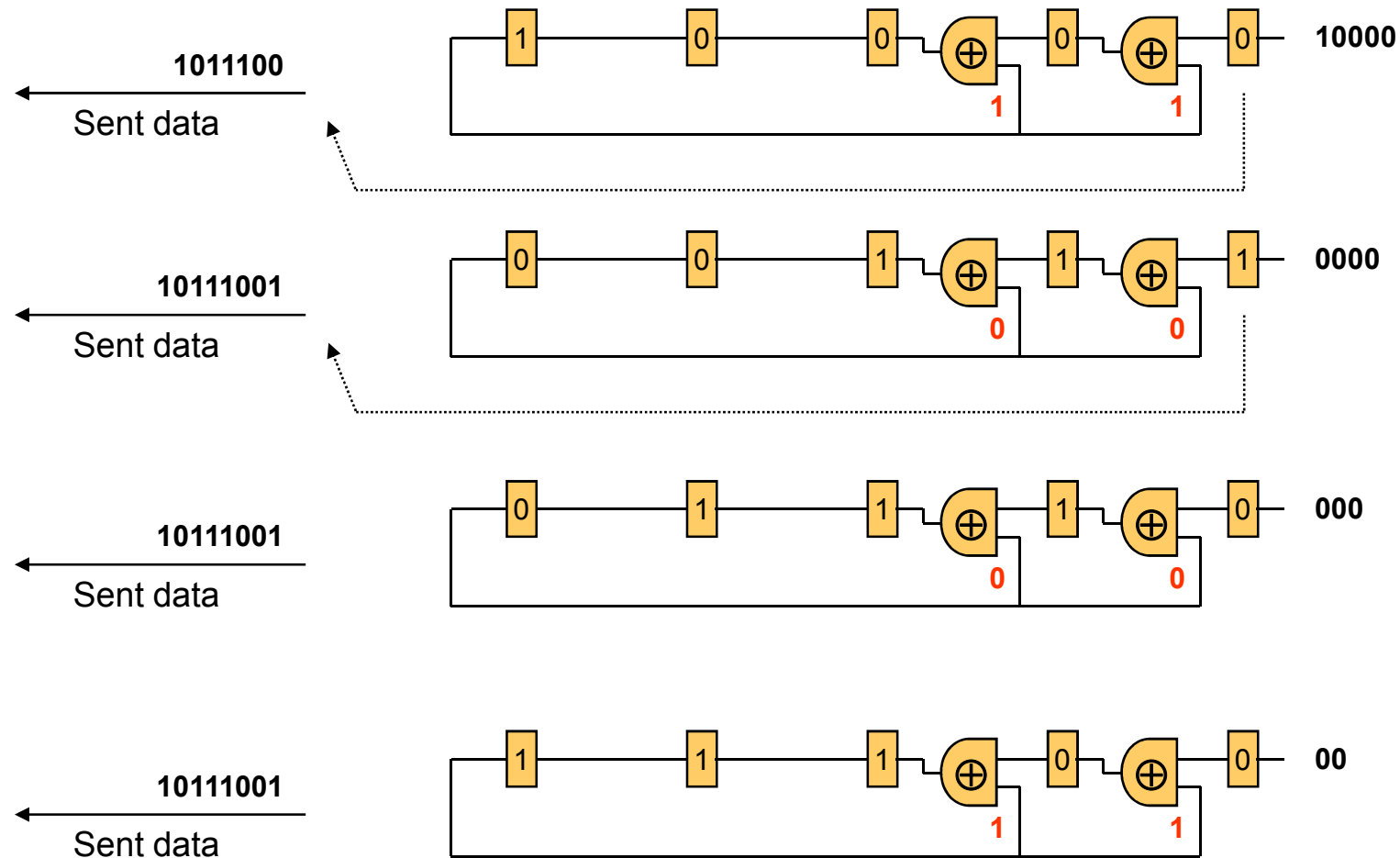
Generator polynomial:  $x^4 + x + 1$



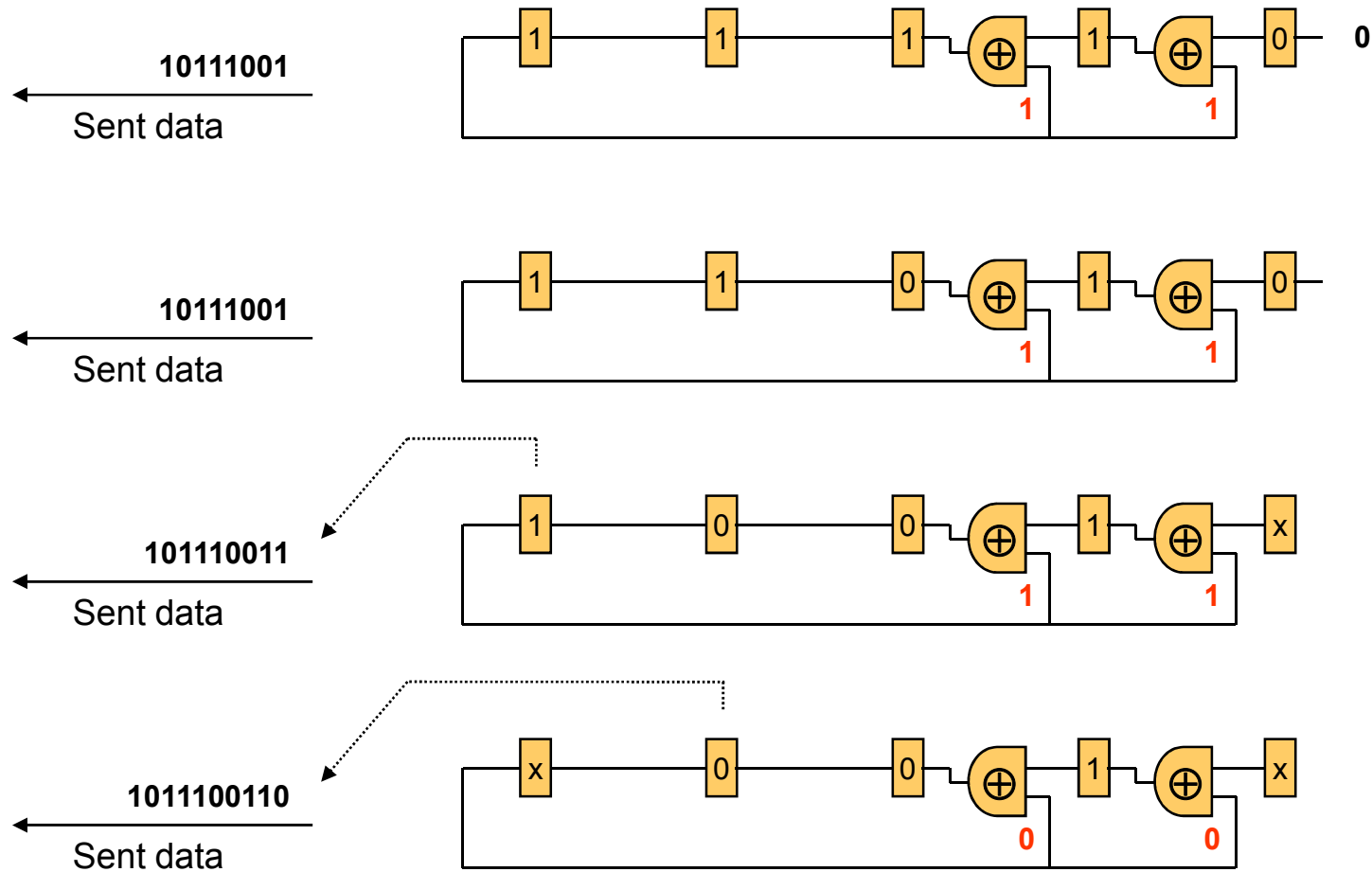
# CRC-Berechnung: Schieberegister



# CRC-Berechnung: Schieberegister



# CRC-Berechnung: Schieberegister



# CRC-Berechnung: Schieberegister

