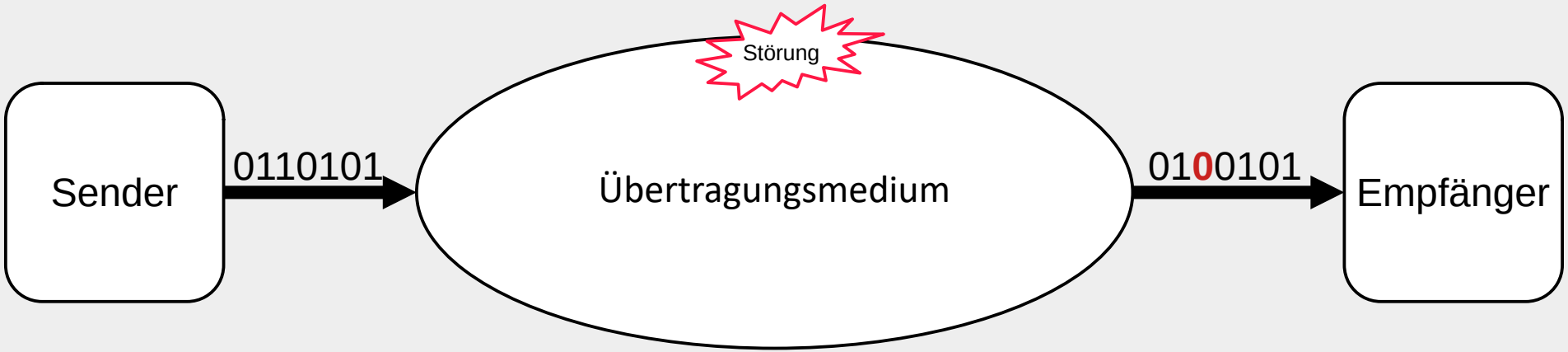
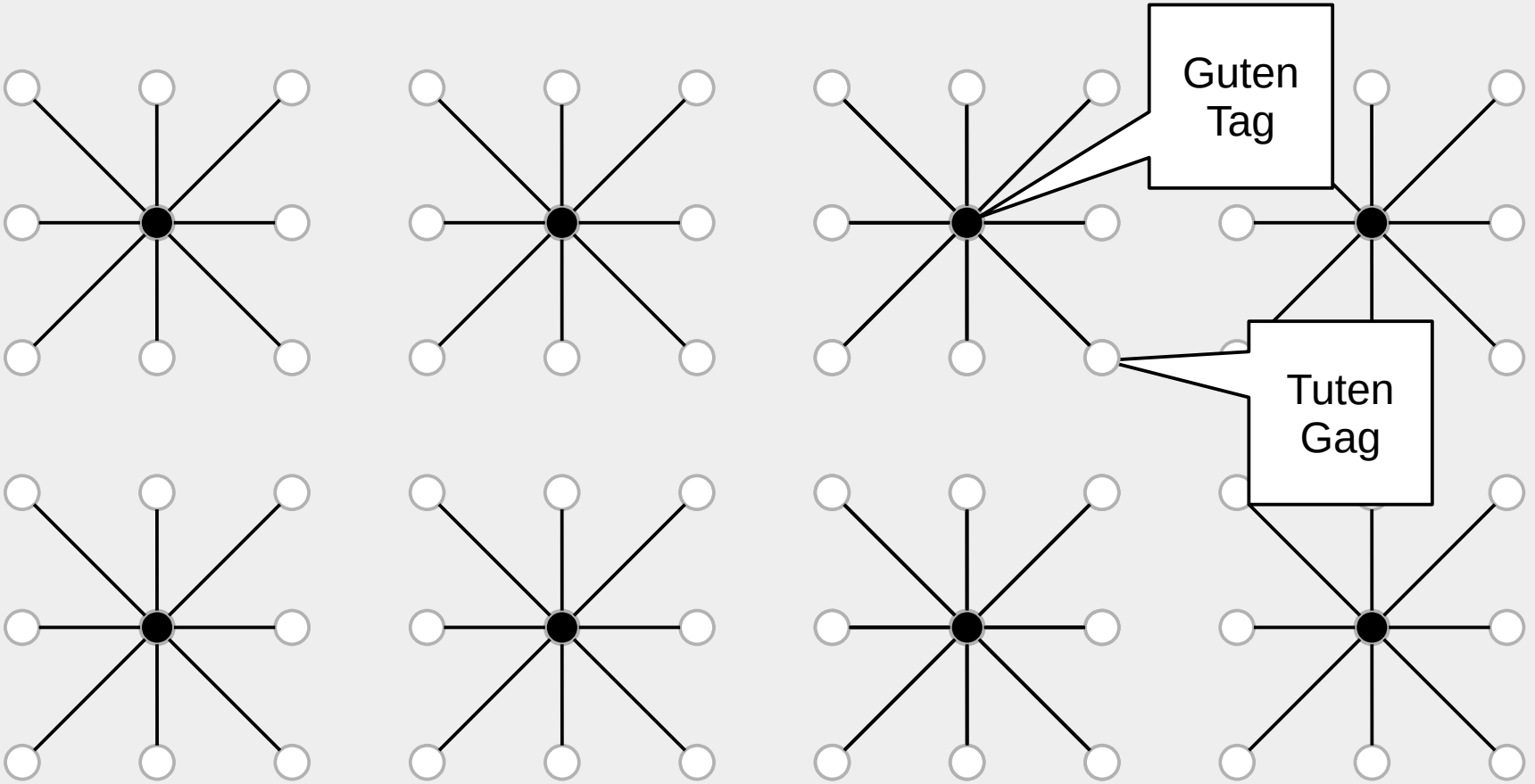
Hamming-Code

Problem: Fehler bei der Datenübertragung

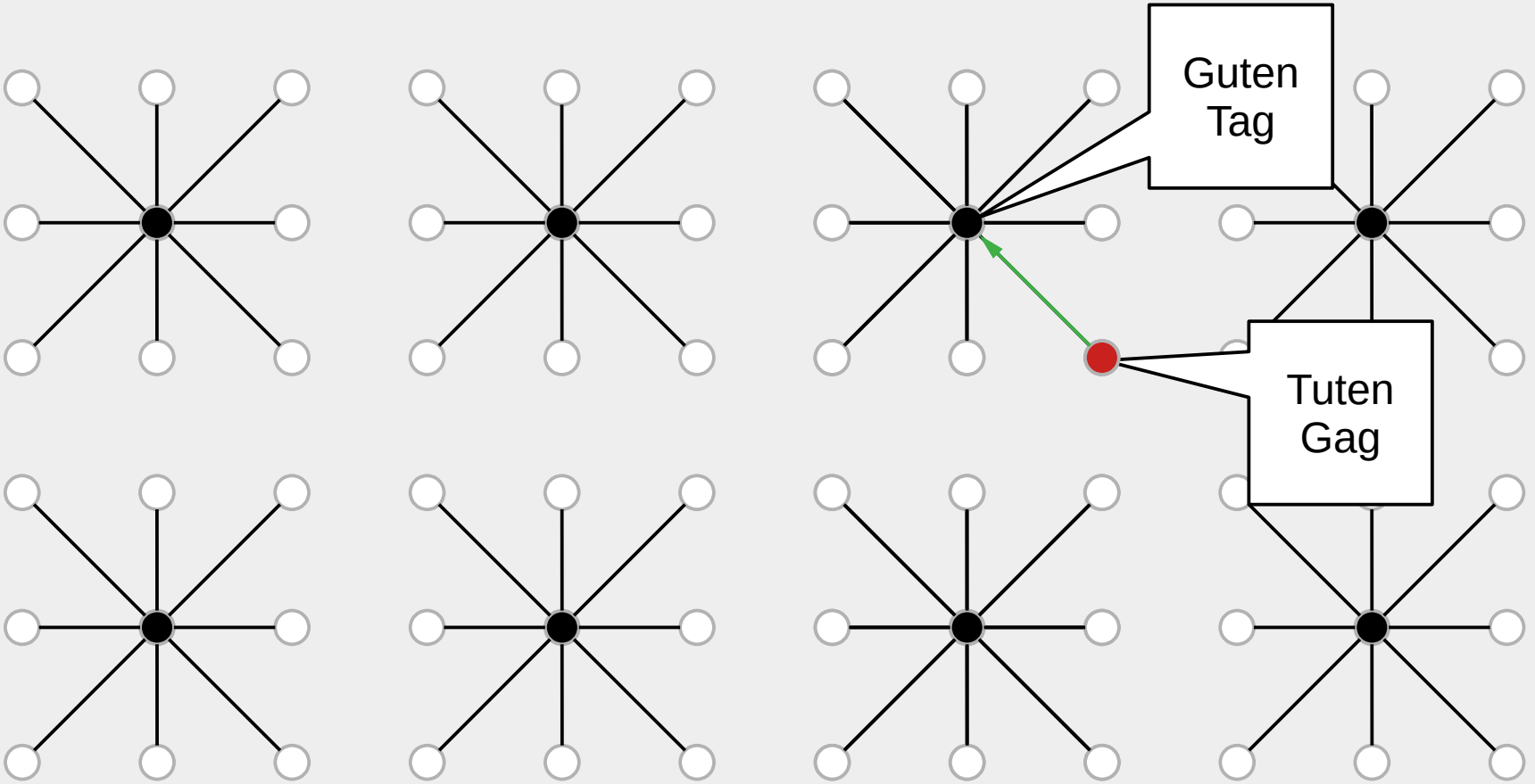
Probleme beim Übertragen von Daten durch Störungen



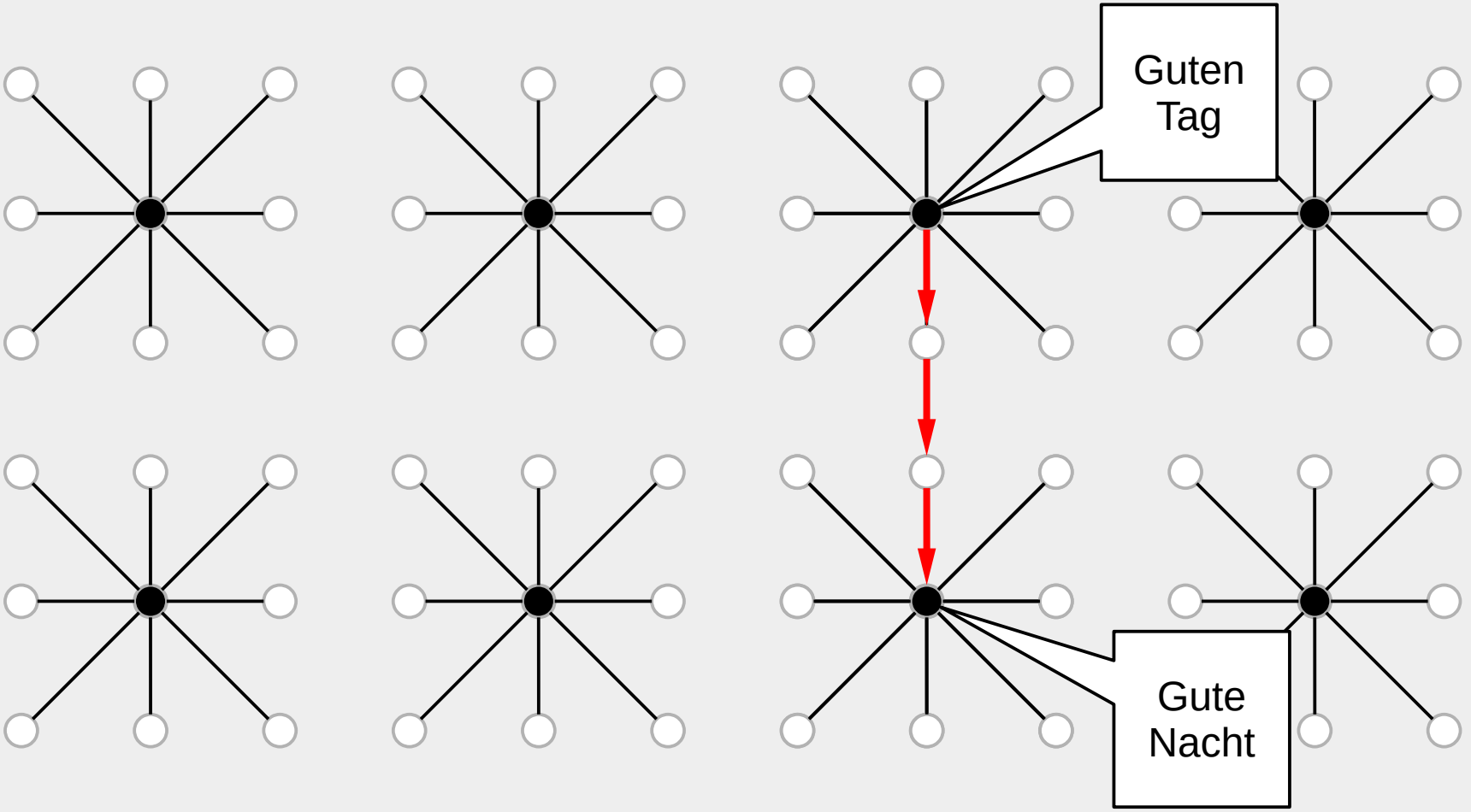
Richtige Nachricht ⊂ Alle möglichen Nachrichten
Einzelne Bitfehler



Korrigieren einzelner Bitfehler



Mehrfache Bitfehler können zu anderen richtigen Nachrichten führen



1. Möglichkeit der Fehlererkennung und Behebung

Zweimaliges Kopieren der Originaldaten

```
1000111000101011 Originaldaten
1000101000101011 1. Kopie
1000111000101011 2. Kopie
-----------------------------
1000111000101011 Ergebnis
```

Auf der Empfängerseite werden die drei Blöcke miteinander verglichen. Einzelne Bitfehler können durch bitweisen Vergleich korrigiert werden.

Nachteil: 66% Redundanz bei der Datenübertragung
Gibt es dafür eine bessere Lösung?

2. Möglichkeit der Fehlererkennung mit Hamming-Code

Richard Hamming war in den 1940er Jahren des letzten Jahrhunderts mit Lesefehlern bei Lochkarten beschäftigt.

Mit dem Hamming-Code kann die Redundanz bei z. B. Blöcken von 256 Bit auf 3,515% reduziert werden. (Bei einem Block von 256 Bits sind 9 Bits redundant)

Grundlage seiner Überlegungen ist die Einführung eines Parity-Bits.

$0 \oplus 0 = 0$
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$
$1 \oplus 1 = 0$

$\oplus$ = XOR

Parity-Bit

P	1	0	1
1	0	0	0
0	1	0	0
1	0	1	1

Parity-Bit
Daten-Bit

Ermittlung des Parity-Bits

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

Summe aller Einsen der Datenbits ist **7**
Sie ist also ungerade.

Will man eine **gerade Parität** (Summe aller Bits ist gerade) erzeugen, muss noch **ein Bit** hinzugefügt werden.

Will man eine **ungerade Parität** (Summe aller Bits ist ungerade) erzeugen, muss **kein Bit** hinzugefügt werden.

Im Folgenden soll immer von einer geraden Parität ausgegangen werden.

Damit ist **P = 1** einzusetzen, um eine gerade Parität zu erzeugen.

Parity-Bit

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

Wird ein Bit verfälscht ergibt die Summe der Datenbits Eine gerade Zahl (8)

Damit ist die Summe aller Bits (inclusive des Parity-Bits) ungerade.

Bei einer geraden Parität weist das auf einen Bitfehler hin.

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

Werden 2 Bits verfälscht ergibt die Summe der Datenbits Eine ungerade Zahl (7)

Damit ist die Summe aller Bits (inclusive des Parity-Bits) gerade.

Bei einer geraden Parität weist das auf keinen Bitfehler hin.

Position der Parity-Bits

Zu übertragende Daten werden in 11 Bit-Blöcke zerlegt

```
01110010101011110010000110110101 1
01110010101 - 01111001000 - 01101101011
```

Die Felder werden zeilenweise mit ihrem Binärwert gekennzeichnet.

Damit haben sie auch eine Adresse

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

Die grün schraffierten Felder können nicht für die Datenübertragung genutzt werden

Sie werden als Paritätsbits verwendet

Damit können 11 Bits als Datenbits verwendet werden

Paritätsbits sind Grundlage für den Hamming-Code
Auswahl der Position für die Parity-Bits

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

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

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

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

Die hellblauen Felder sind die Parity-Bits der blauen Bereiche
Das Grüne Feld bleibt vorerst unbeachtet

Ermittlung von Doppelfehlern

Ist kein Fehler vorhanden sind alle Parity-Bedingungen erfüllt.
Wird nun mit dem grünen Feld eine Parity-Berechnung über alle anderen Felder (auch die Parity-Bits) gemacht ist das grüne Parity-Bit = 0

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

Kein Bitfehler
→ Resultierendes grünes Parity-Bit = 0

0->1	0->1	1	0
0->1	0	1	1
1	1	0	1
0	1	1	1

Ein Bitfehler
→ Mit Parity-Bits kann die **Position** ermittelt werden
→ Resultierendes grünes Parity-Bit = 1

0->0	0->1	1	0
0->1	1	1	1
1->0	0	0	1
0	1	1	1

Zwei Bitfehler
→ Mit Parity-Bits kann nur ein Fehler ermittelt werden
→ Resultierendes grünes Parity-Bit = 0 → Hinweis auf 2 Bit-Fehler

Hamming-Code
skaliert bei großen Blöcken hervorragend

2

0	0

4

0	0
0	1

8

0	0	1	0
0	1	1	1

16

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

32

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

64

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

Block-Größe	Anzahl Paritätsbits	ProzentualerAnteil
1	1	100,000000%
2	2	100,000000%
4	3	75,000000%
8	4	50,000000%
16	5	31,250000%
32	6	18,750000%
64	7	10,937500%
128	8	6,250000%
256	9	3,515625%
512	10	1,953125%
1024	11	0,976563%
2048	12	0,537109%
4096	13	0,292969%
8192	14	0,158691%
16384	15	0,085449%
32768	16	0,045776%
65536	17	0,024414%
131072	18	0,012970%
262144	19	0,006866%
524288	20	0,003624%
1048576	21	0,001907%

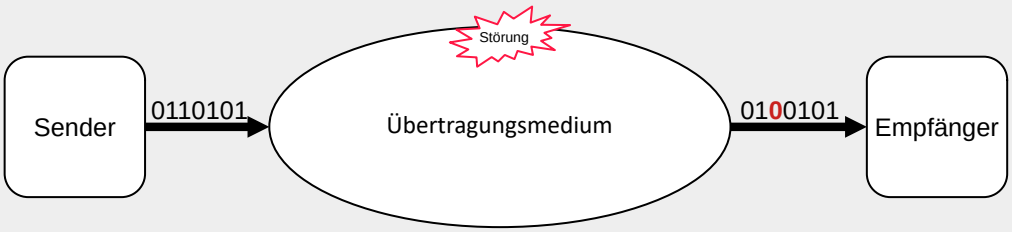
Hamming-Code

Lets go EXCEL

Hamming-Code

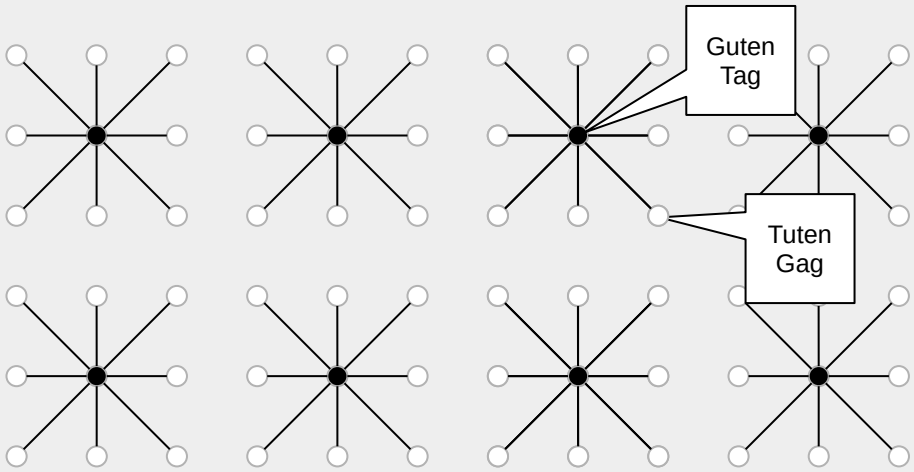
Problem: Fehler bei der Datenübertragung

Probleme beim Übertragen von Daten durch Störungen



Störungen können bei der Übertragung von Daten immer auftreten.

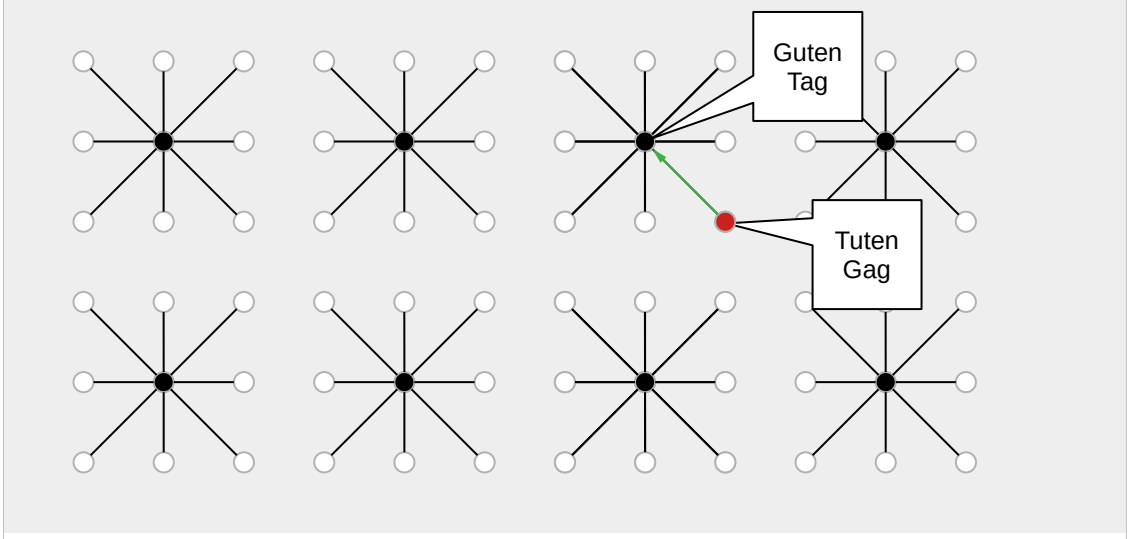
Richtige Nachricht ⊂ Alle möglichen Nachrichten
Einzelne Bitfehler



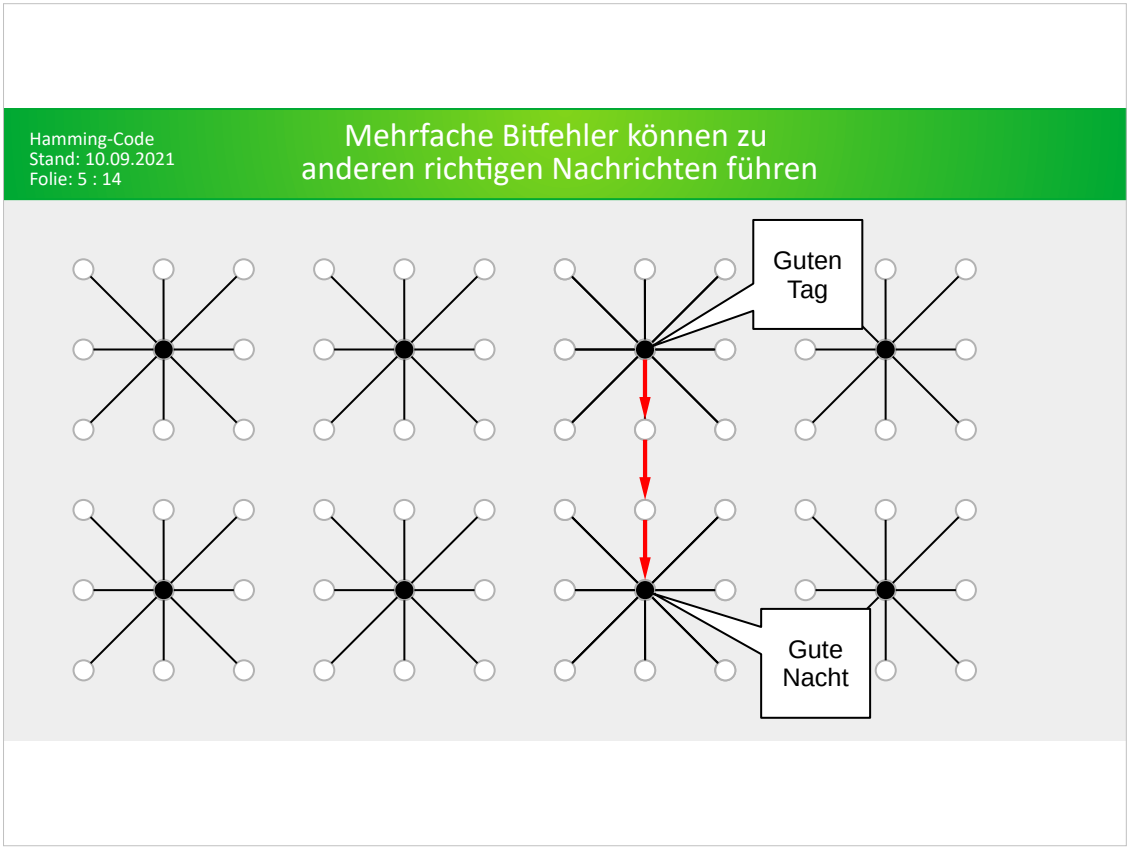
Die Menge der richtigen Nachrichten ist eine Untermenge aller Nachrichten.

Einzelfehler können zur Verfälschung der Nachricht führen.

Korrigieren einzelner Bitfehler



Einzelfehler können durch Verwendung eines Hamming-Codes korrigiert werden.



Mehrere Fehler hintereinander werden im Englischen Burstfehler genannt.

Das kann zu anderen Nachrichten führen, die zwar zulässig und trotzdem falsch sind.

Deshalb ist es wichtig Burstfehler zu eliminieren.

Das kann dadurch erreicht werden indem Burstfehler zu Einzelfehlern gemacht werden. Eine mögliche Vorgehensweise zur Vermeidung von Burst-Fehlern ist die Verwendung eines Interleavers. (siehe WLAN-Vorlesung)

Einzelfehler können wiederum durch Verwendung eines Hamming-Codes eliminiert werden.

1. Möglichkeit der Fehlererkennung und Behebung

Zweimaliges Kopieren der Originaldaten

```
1000111000101011 Originaldaten
10001O1000101011 1. Kopie
1000111000101011 2. Kopie
-----------------
1000111000101011 Ergebnis
```

Auf der Empfängerseite werden die drei Blöcke miteinander verglichen. Einzelne Bitfehler können durch bitweisen Vergleich korrigiert werden.

Nachteil: 66% Redundanz bei der Datenübertragung
Gibt es dafür eine bessere Lösung?

2. Möglichkeit der Fehlererkennung mit Hamming-Code

Richard Hamming war in den 1940er Jahren des letzten Jahrhunderts mit Lesefehlern bei Lochkarten beschäftigt.

Mit dem Hamming-Code kann die Redundanz bei z. B. Blöcken von 256 Bit auf 3,515% reduziert werden. (Bei einem Block von 256 Bits sind 9 Bits redundant)

Grundlage seiner Überlegungen ist die Einführung eines Parity-Bits.

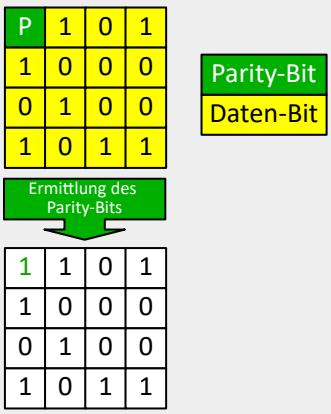
$0 \oplus 0 = 0$
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$
$1 \oplus 1 = 0$

$\oplus$ = XOR

Mit dem nach dem Erfinder benannten Hamming-Code ist es möglich, Einzelbitfehler zu erkennen und zu korrigieren.

Die mathematische Grundlage für den Parity-Code ist die XOR-Funktion.

Parity-Bit



Summe aller Einsen der Datenbits ist **7**
Sie ist also ungerade.

Will man eine **gerade Parität** (Summe aller Bits ist gerade) erzeugen, muss noch **ein Bit** hinzugefügt werden.

Will man eine **ungerade Parität** (Summe aller Bits ist ungerade) erzeugen, muss **kein Bit** hinzugefügt werden.

Im Folgenden soll immer von einer geraden Parität ausgegangen werden.

Damit ist **P = 1** einzusetzen, um eine gerade Parität zu erzeugen.

Bei einem 16 Bit großen Feld wird ein Bit zum Parity-Bit gemacht.

Das Ergebnis einer XOR-Funktion entspricht dem Wert des Parity-Bits bei gerader Parität.

Damit können nicht mehr 16 sondern nur noch 15 Daten-Bits verwendet werden!

Parity-Bit

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

Wird ein Bit verfälscht ergibt die Summe der Datenbits Eine gerade Zahl (8)

Damit ist die Summe aller Bits (inclusive des Parity-Bits) ungerade.

Bei einer geraden Parität weist das auf einen Bitfehler hin.

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

Werden 2 Bits verfälscht ergibt die Summe der Datenbits Eine ungerade Zahl (7)

Damit ist die Summe aller Bits (inclusive des Parity-Bits) gerade.

Bei einer geraden Parität weist das auf keinen Bitfehler hin.

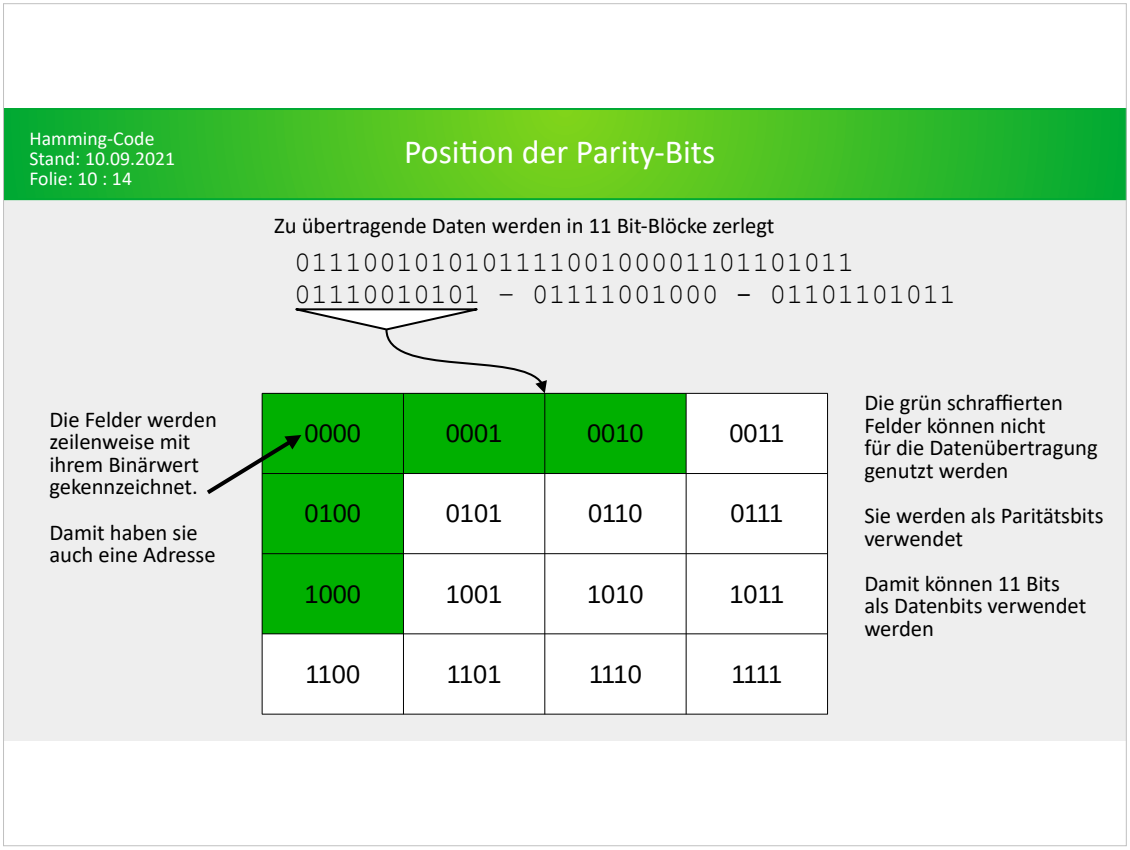
Wird nun bei der Datenübertragung ein Bit verfälscht, ist die Summe der Datenbits gerade.
Das auf 1 gesetzte Parity-Bit führt zu einer ungeraden Gesamtsumme.
Dies ist der Hinweis auf einen Bitfehler.

Ungeradzahlige (1, 3, 5, 7, ...) Bitfehler können erkannt werden.

Geradzahlige (2, 4, 6, 8, ...) Bitfehler können nicht erkannt werden.

Mit einem einzelnen Parity-Bit können also nur ungeradzahlige Fehler erkannt werden. Deshalb ist es wichtig Burst-Fehler zu vermeiden.

Mit nur einem Parity-Bit können Fehler erkannt werden, allerdings können sie nicht korrigiert werden, da ihre Position nicht bekannt ist.

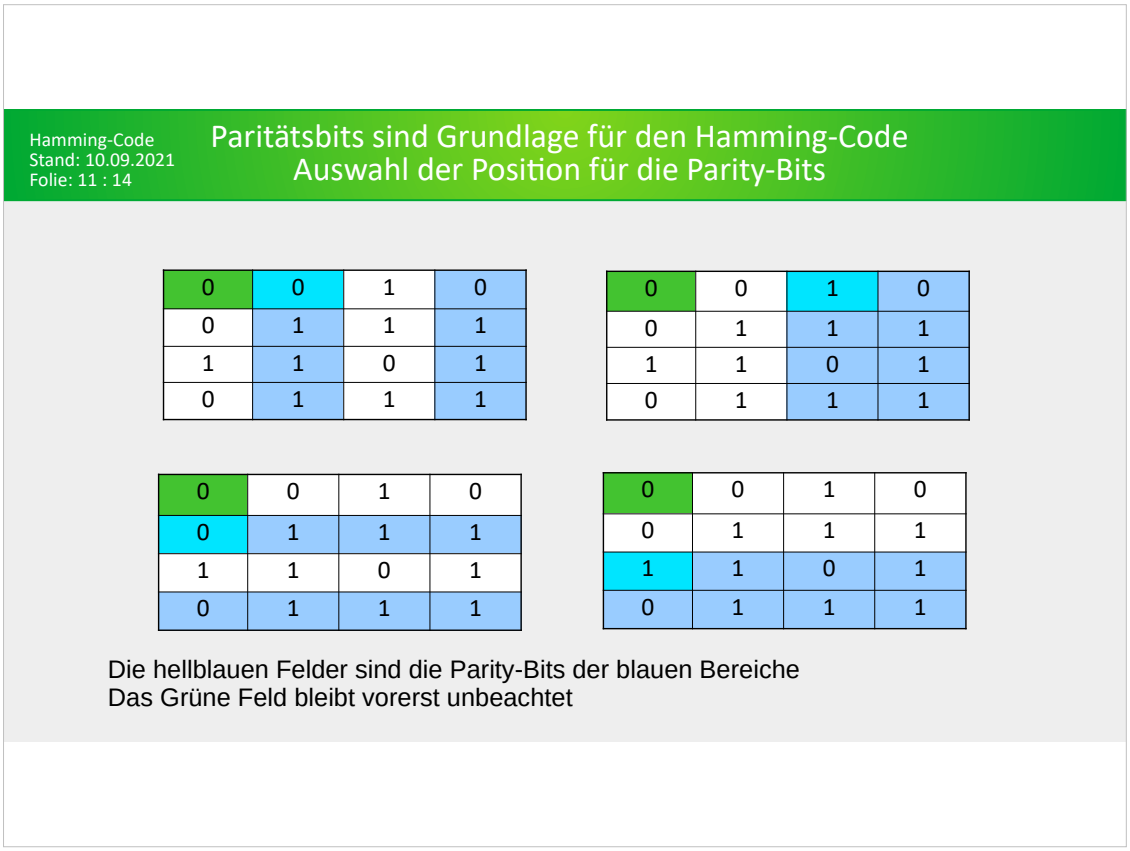


Um Bitfehler nicht nur zu erkennen, sondern auch zu korrigieren, müssen weitere Partiy-Bits eingerichtet werden, was die Anzahl der Daten-Bits weiter reduziert.

Bei einem 16 Bit großen Feld müssen 5 Bits zu Partity-Bits deklariert werden und können nicht für Datenbits verwendet werden. Damit können nur noch 11 Datenbits überwacht werden.

Es fällt auf, dass bei den Parity-Bits-Adressen der immer nur ein Bit auf 1 gesetzt ist. Bei allen anderen Adressen sind alle Bits auf Null (links oben) oder mehr als ein Bit auf 1 gesetzt.

Das ist auf den ersten Blick eine massive Erhöhung des Overheads, der sich jedoch mit zunehmender Größe massiv relativiert.



Grundlagen zum Hamming-Codes:

- XOR-Funktion
- Parity-Bits (in der Form für gerade Parität)
- Parity-Bits stehen an der Stelle 2^n

Die Parity-Bits stehen immer an der ersten Stelle, des Bereichs, für den das Parity-Bit bestimmt wird.

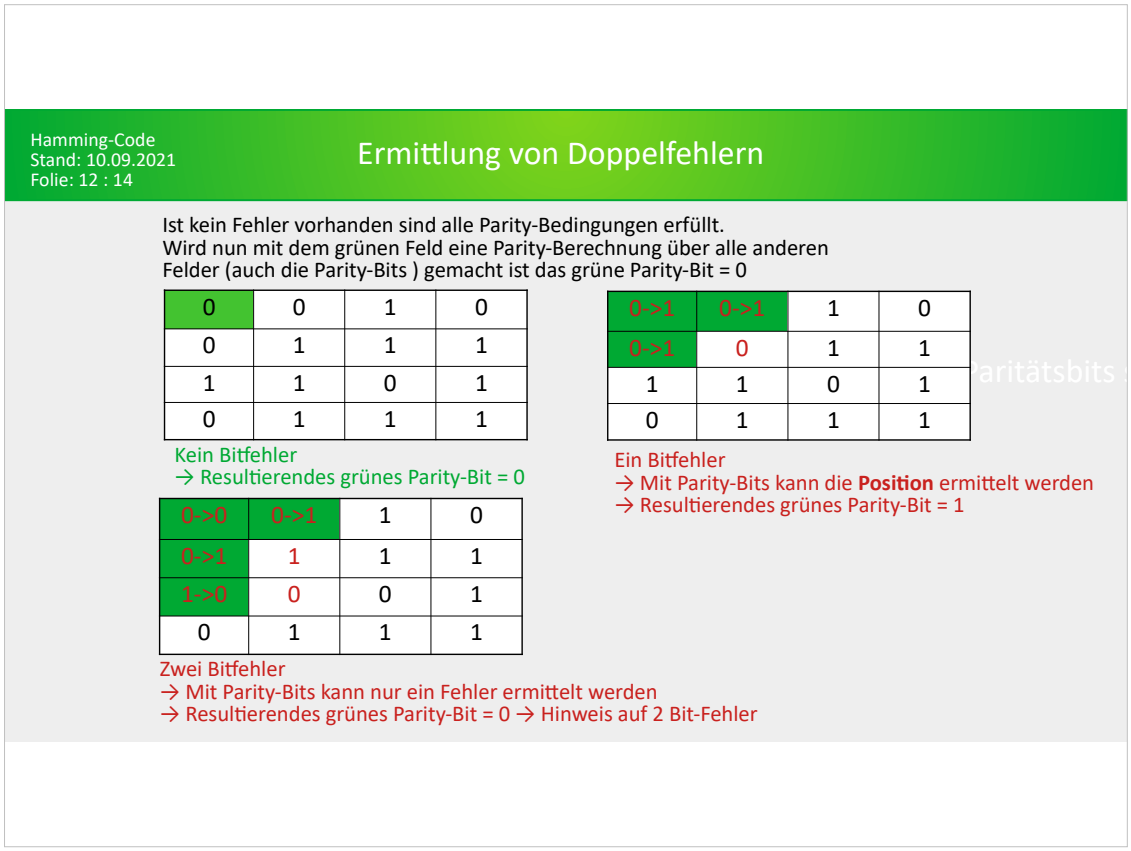
Wie zu sehen ist, können die beiden hellblauen Parity-Felder in der oberen Zeile zur Ermittlung der Spalte mit einem fehlerhaften Bit herangezogen werden.

Die beiden hellblauen Parity-Felder der linken Spalte können für die Ermittlung der Zeile mit dem fehlerhaften Bit herangezogen werden.

Ist in einem blauen Bereich ein Fehler, kann davon ausgegangen werden, dass man schon mal die Hälfte der Datenfelder identifiziert wurde, in der der Bitfehler liegt. Dann ist nur noch weiter zu unterteilen.

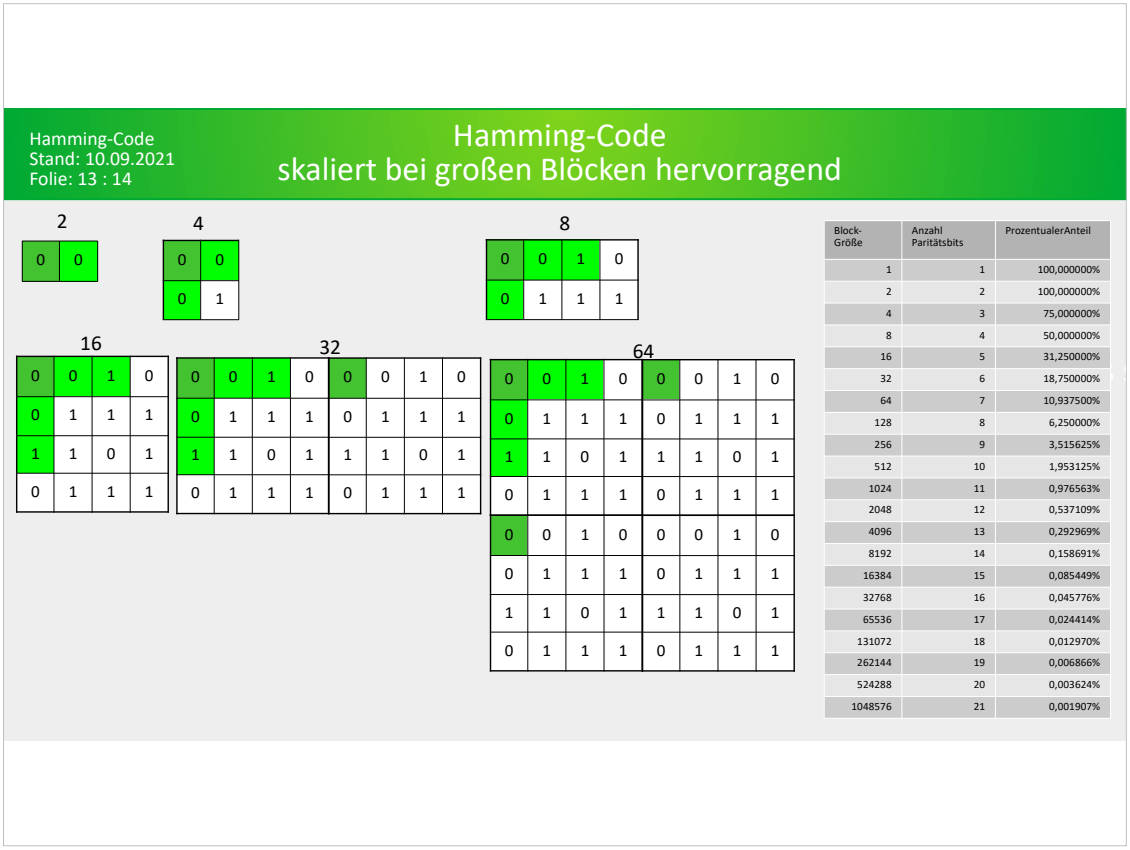
Ist in einem blauen Bereich kein Fehler, kann davon ausgegangen werden, dass es entweder keinen Fehler gibt, oder dass der Fehler im weißen Bereich liegt.

Damit wird auch klar, warum die Parity-Bits an der Stelle 2^n stehen müssen und warum die Regeln für die Ermittlung der Parity-Bits
Für n von 0 bis 3: n + 1 Bits nehmen, danach n + 1 Bits überspringen, usw.



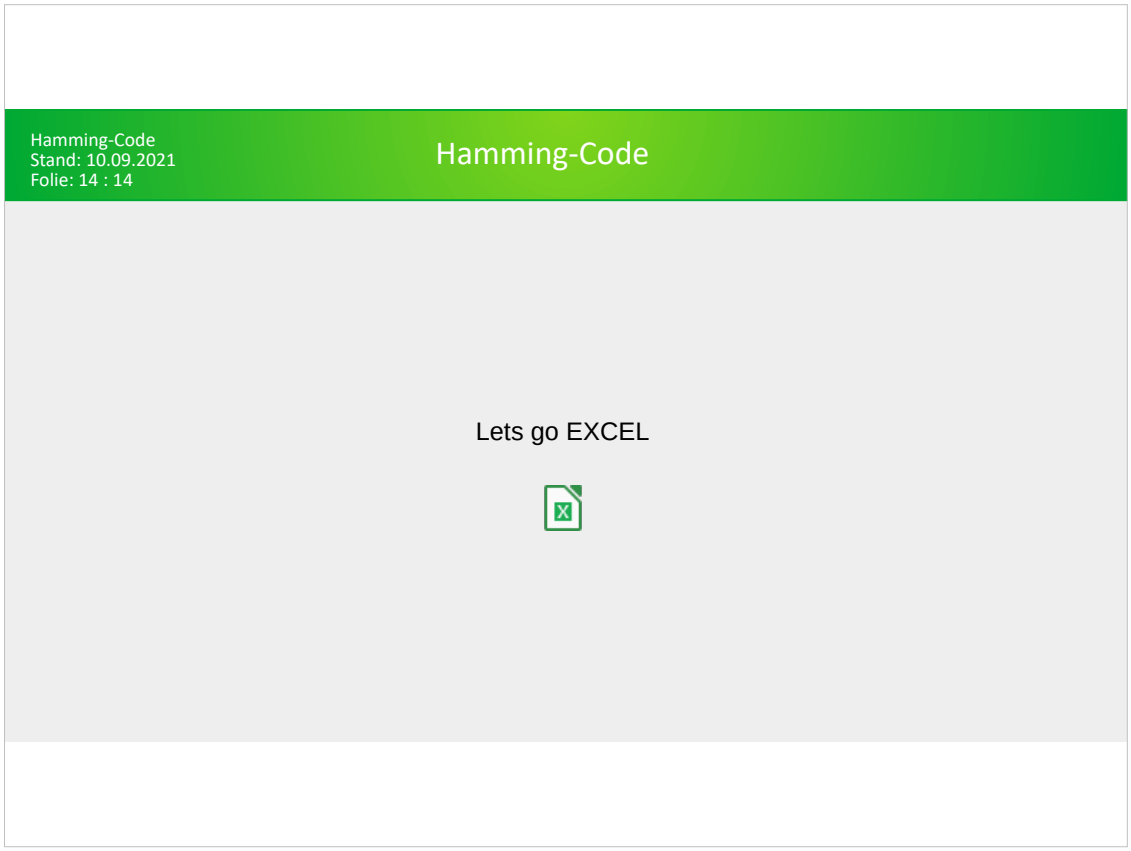
Mit dem grünen Parity-Bit kann ein Doppelfehler erkannt werden.

Eine Korrektur des Fehler ist jedoch nicht möglich.



Mit jedem zusätzlichen Parity-Bit kann der Bereich verdoppelt werden.

Das führt dazu, dass der Hamming-Code bei großen Feldern wunderbar skaliert.



In der zugehörigen EXCEL-Datei werden die nächsten Schritte visualisiert.