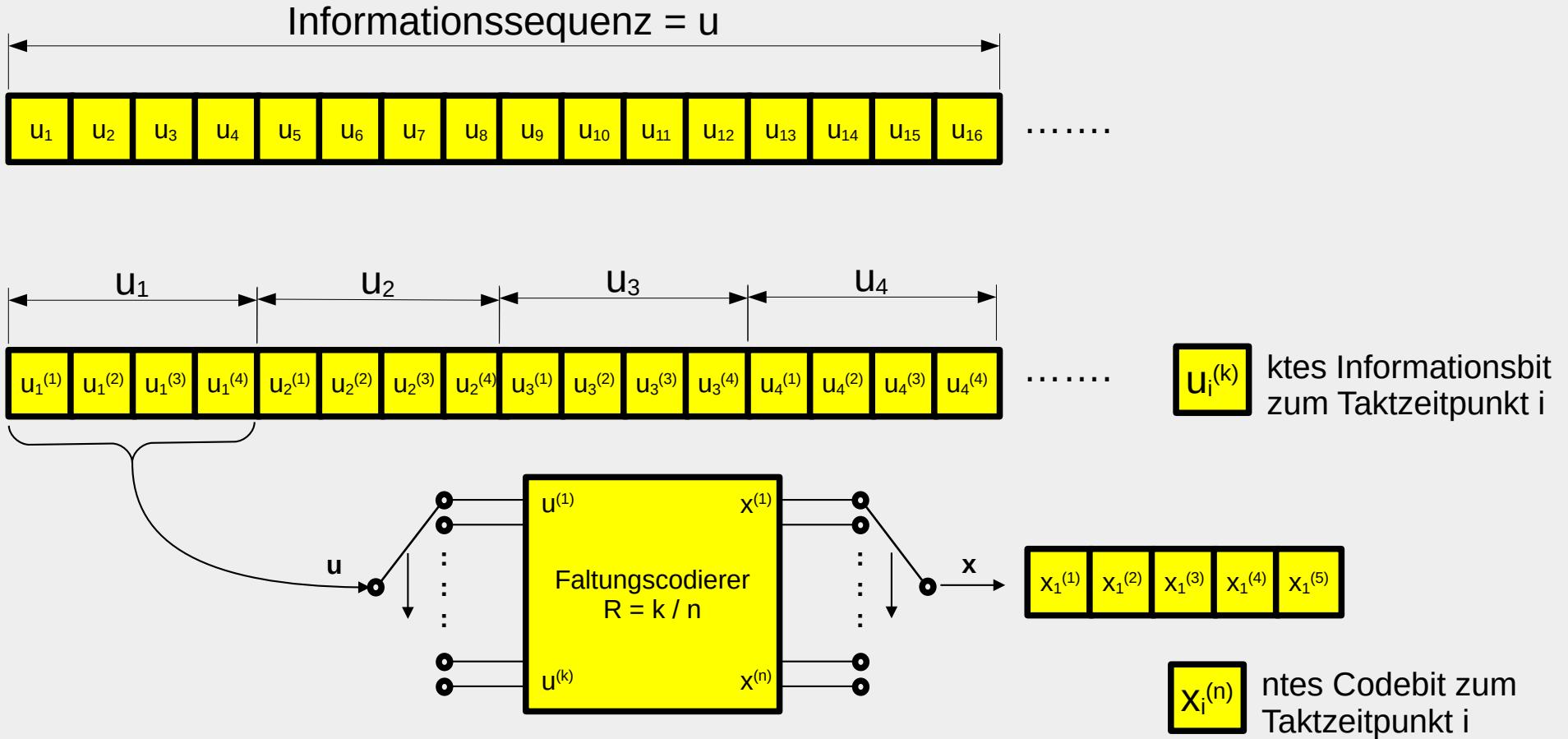
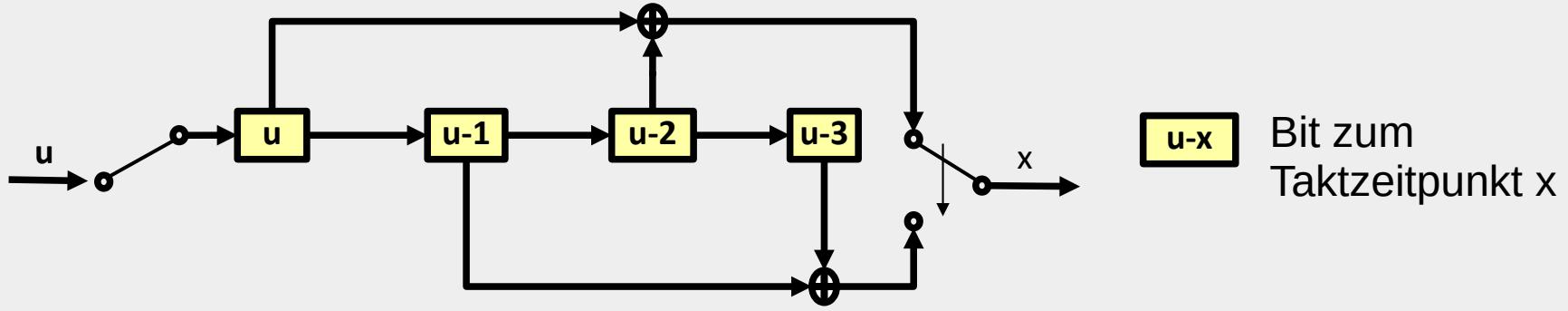
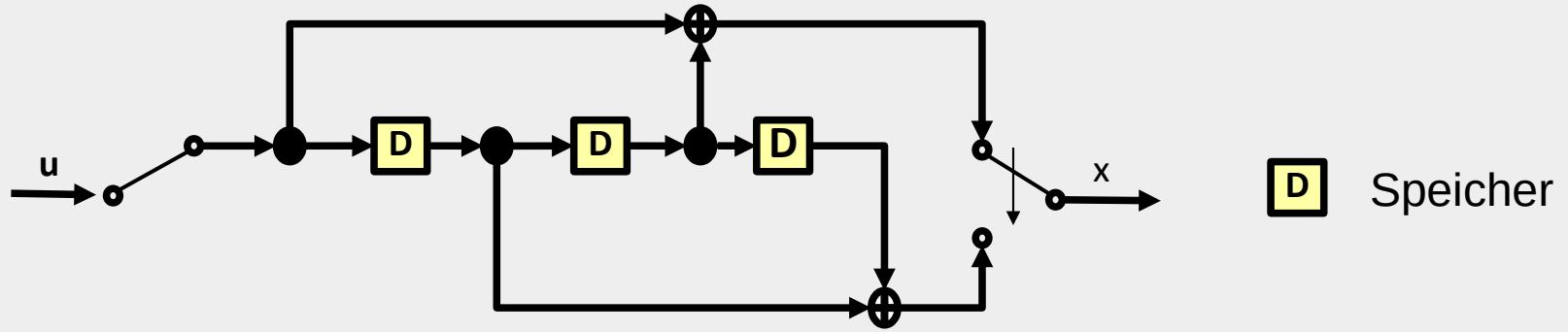


Faltungscodierer

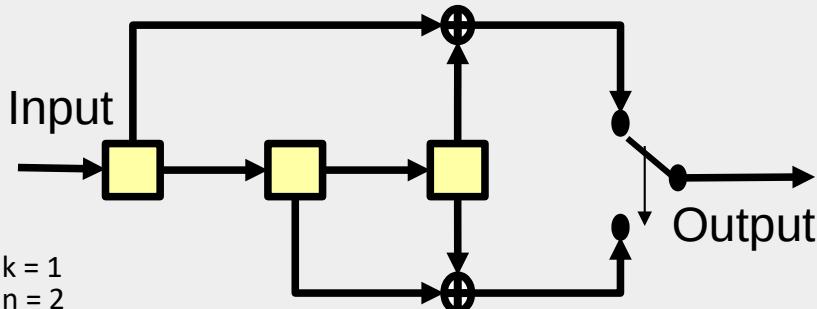
Faltungscodierer (allgemein) Teil-1



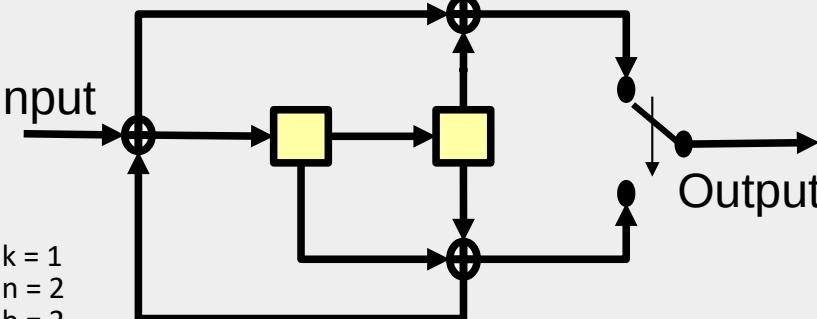
Faltungscodierer (allgemein) Teil-2



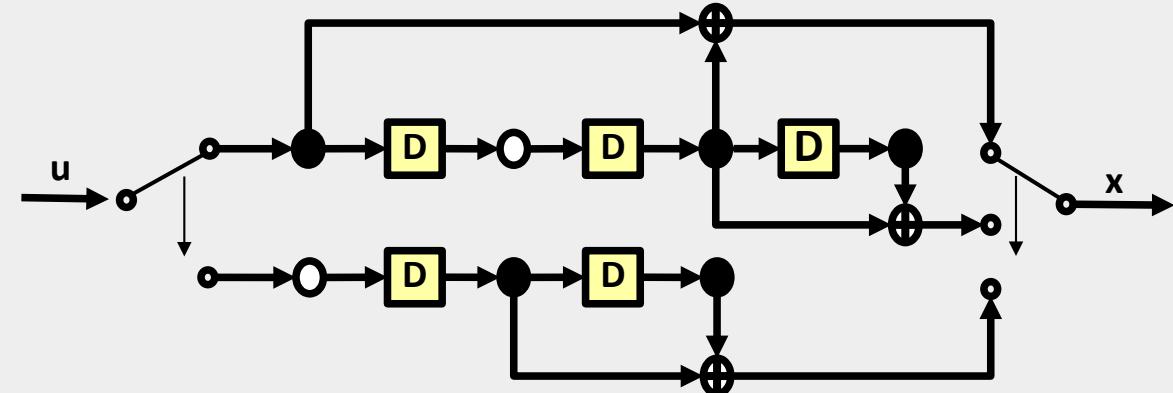
Faltungscodierer (allgemein) Teil-3



Nicht rekursiver
Faltungscodierer
mit einer Speicherkette



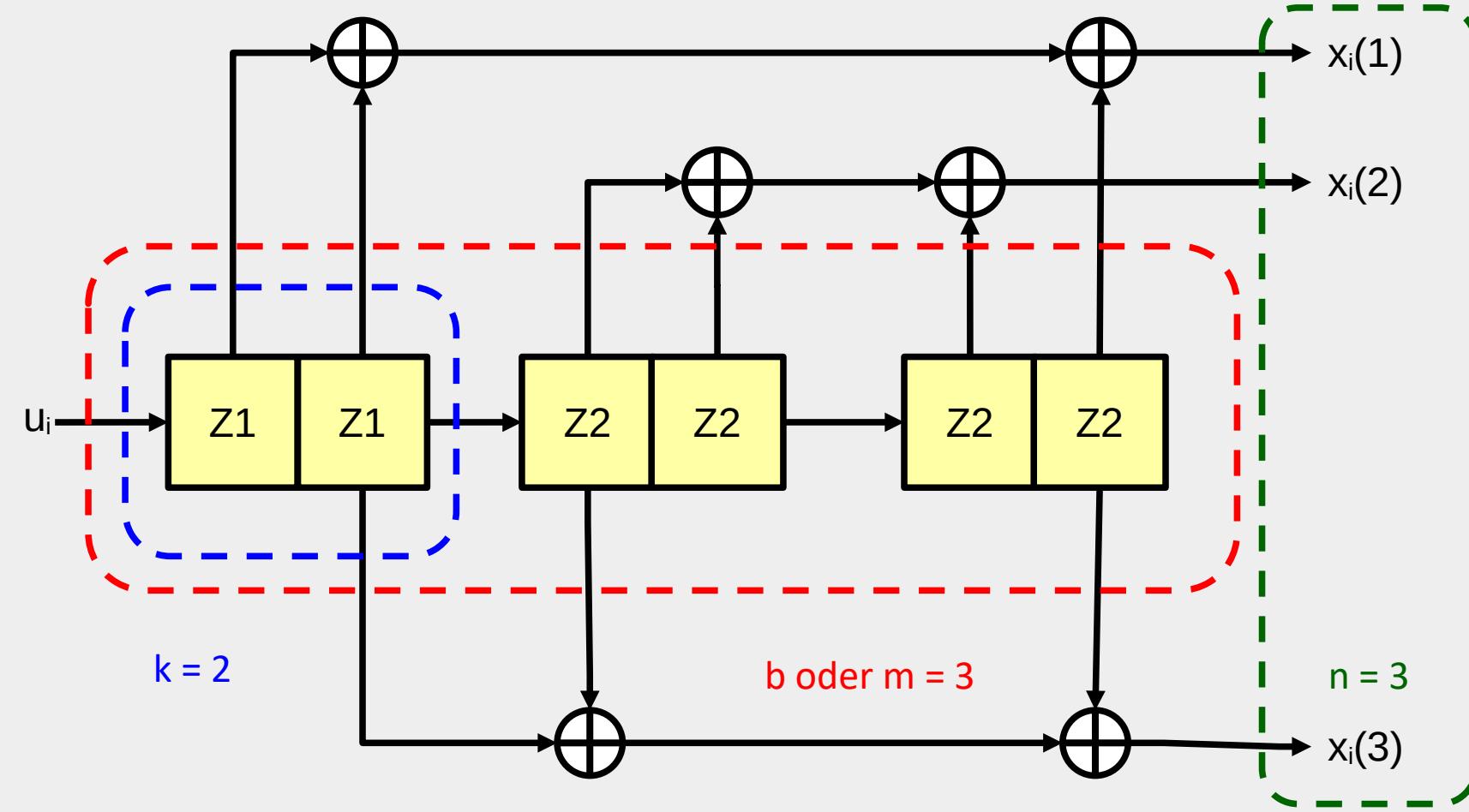
Rekursiver
Faltungscodierer
mit einer Speicherkette



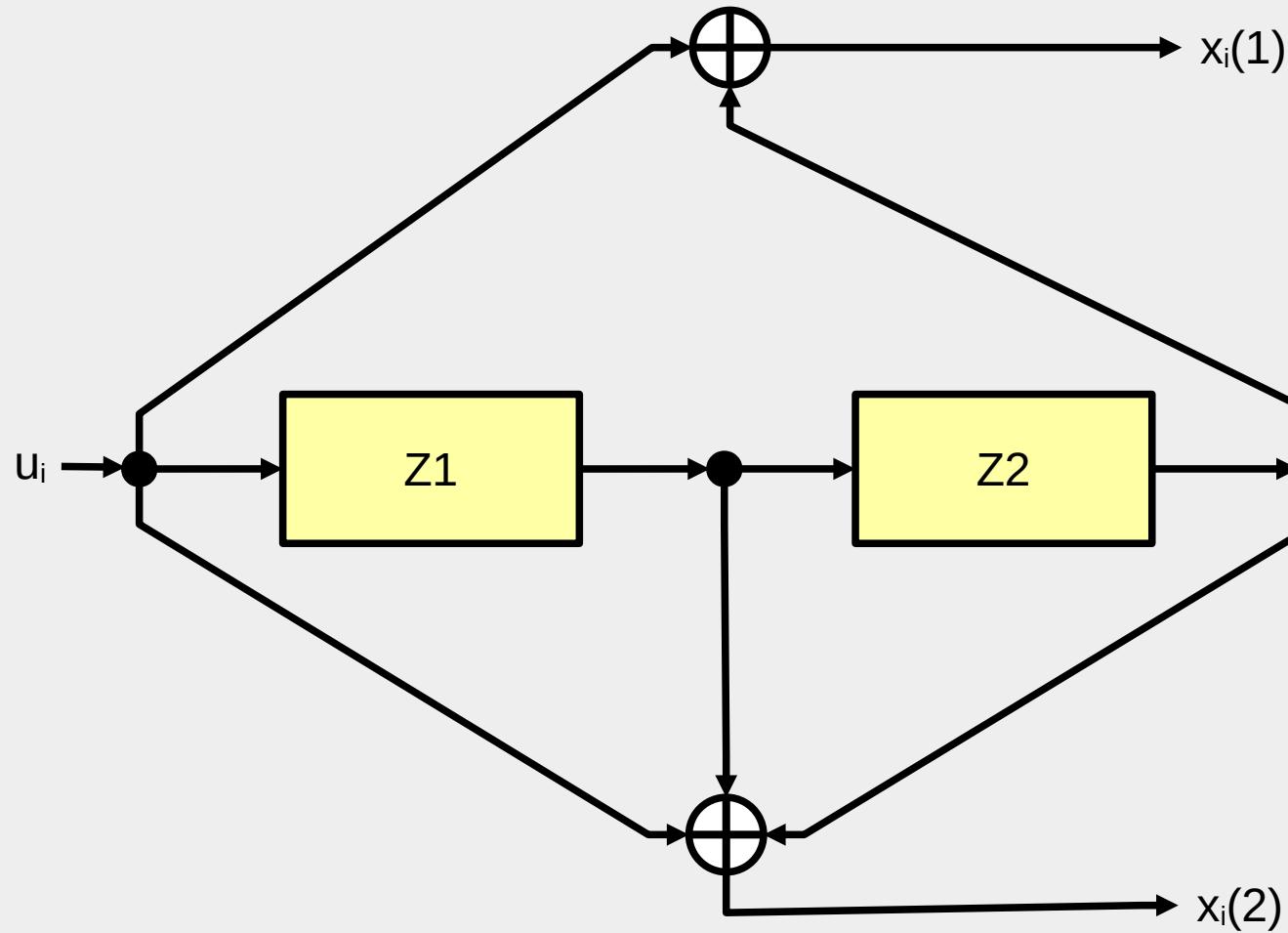
Nicht rekursiver
Faltungscodierer
mit 2 Speicherketten

$k = 2$
 $n = 3$
 $b = 3$ (längste Speicherkette)
 $L_c = 8$ (nicht verwendete Punkte werden mitgezählt)

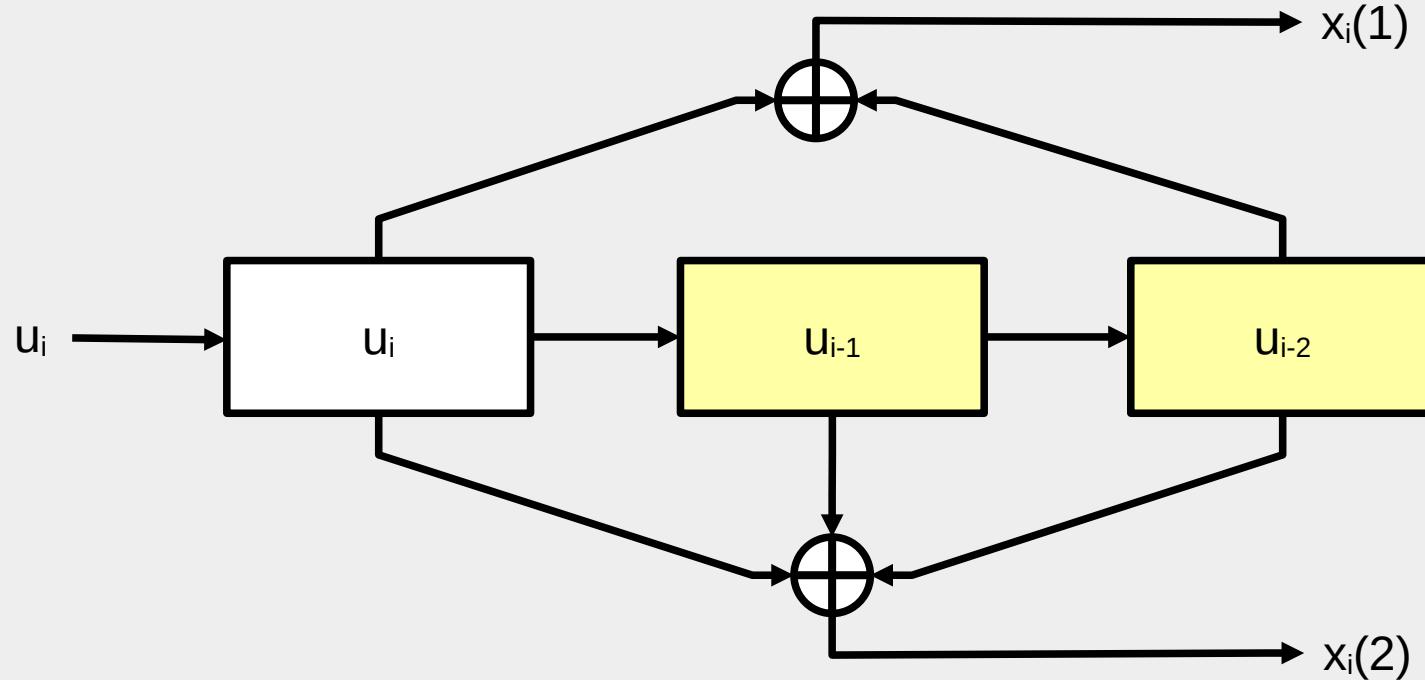
Faltungscodierer (allgemein) Teil-4



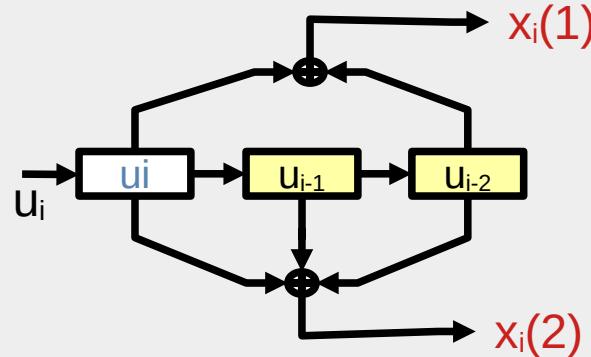
Faltungscodierer-Beispiel



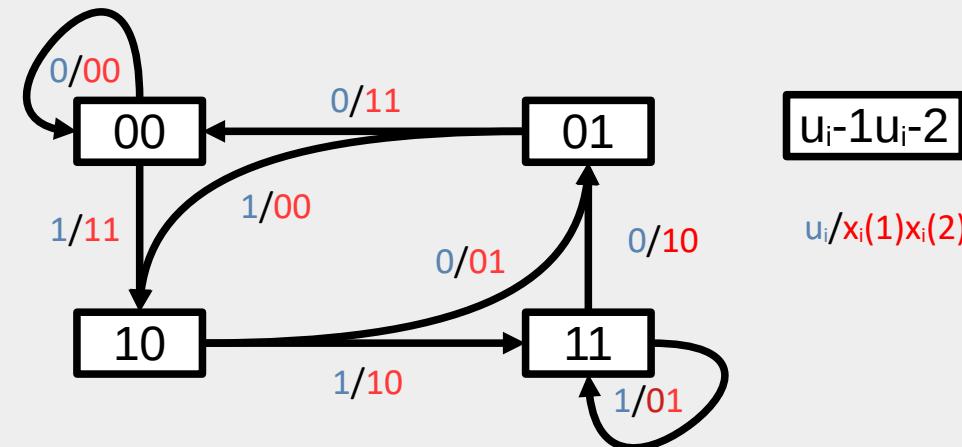
Faltungscodierer-Beispiel



Vom Faltungscodierer zum Zustandsdiagramm

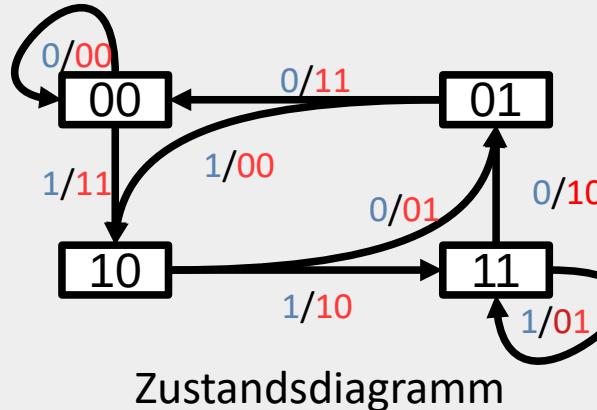


u_i		0	1	0	1	0	1	0	1
u_{i-1}	0	0	0	1	1	0	0	1	1
u_{i-2}	0	0	0	0	0	1	1	1	1
$x_i(1)$	0	0	1	0	1	1	0	1	0
$x_i(2)$	0	0	1	1	0	1	0	0	1



Zustandsdiagramm

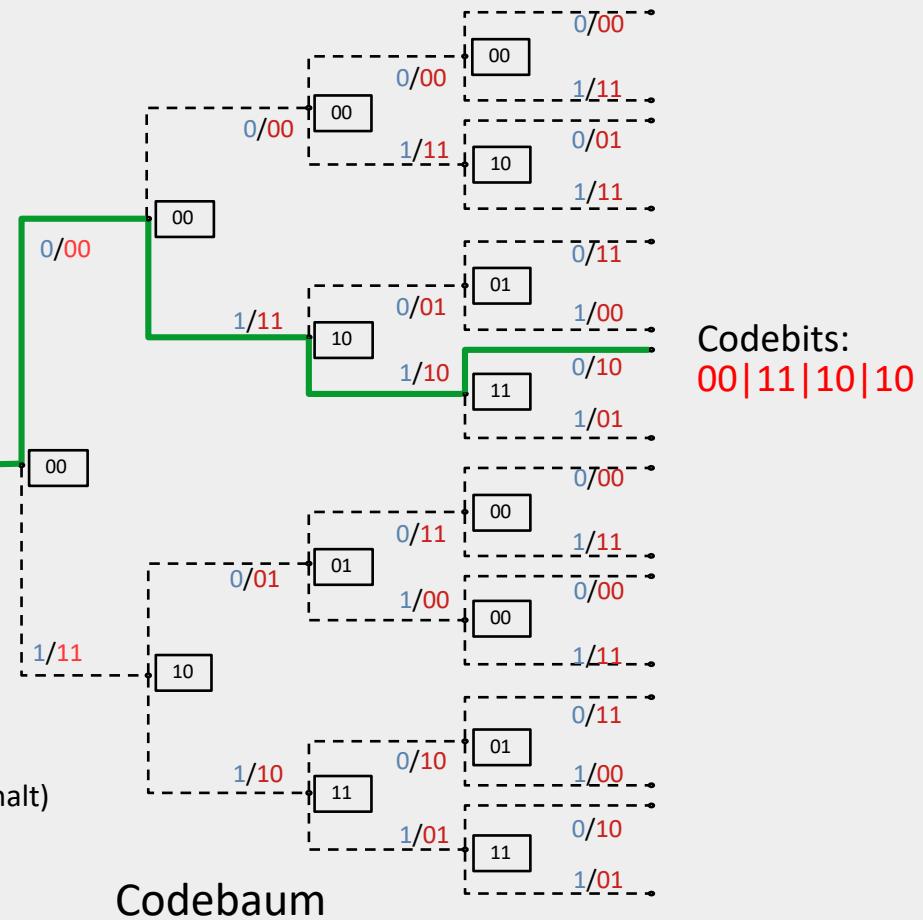
Vom Zustandsdiagramm zum Codebaum



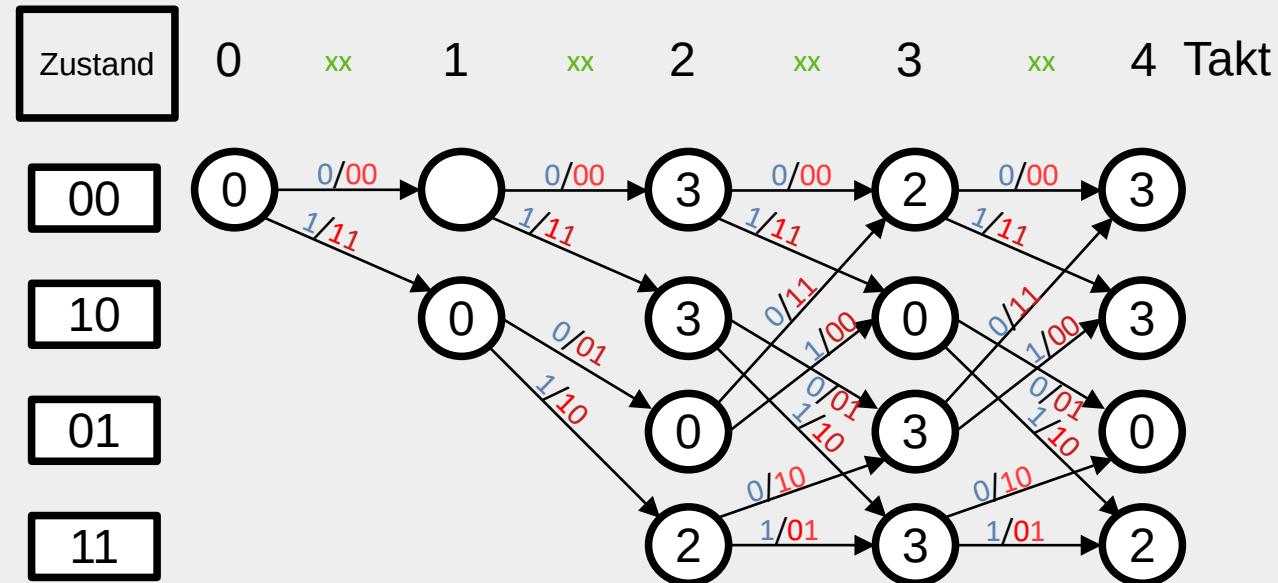
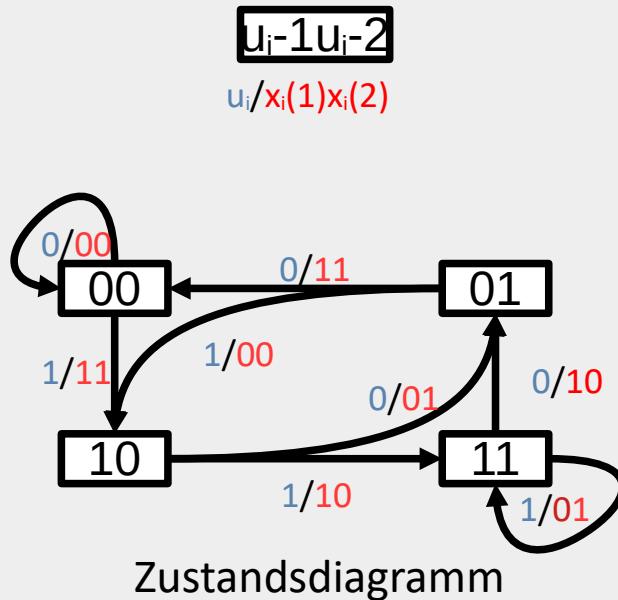
$$u_{i-1} u_i - 2 \\ u_i / x_i(1) x_i(2)$$

Informationsbits:
0|1|1|0

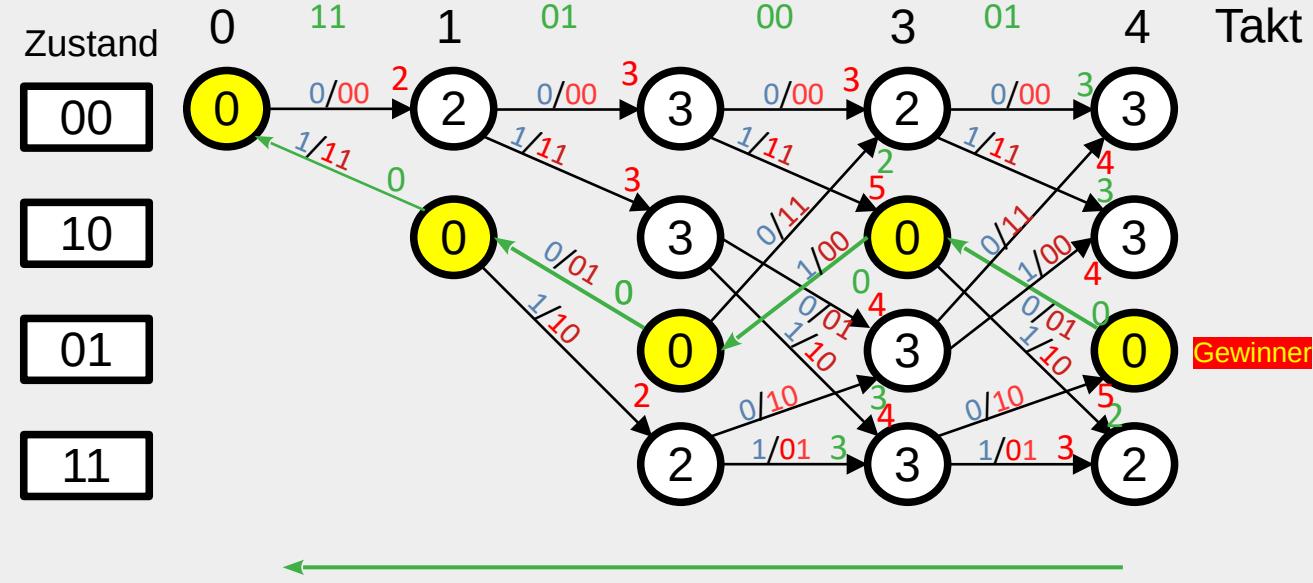
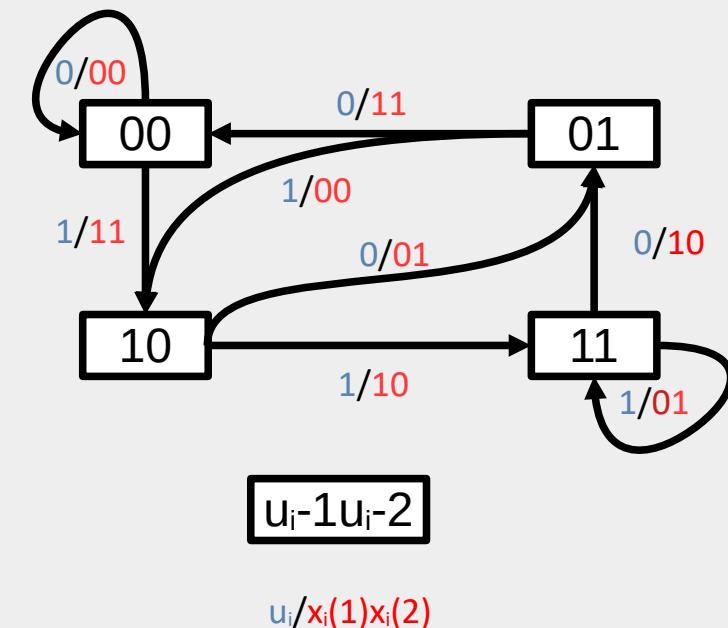
00 Zustand (Speicherinhalt)
↓ Informationsbit am Eingang
0/00 ↑ Code am Ausgang



Vom Zustandsdiagramm zum Trellis



Trellisdiagramm - Bearbeitung

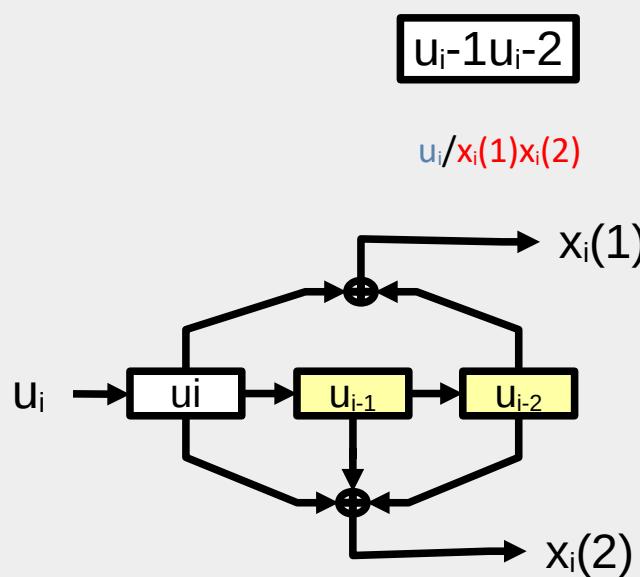


Vom Anfang an ausgehend werden die Eingangswerte übernommen

Ergebnis-Bitfolge: 1|0|1|0

Beispiel-Bitfolge: 1|0|1|0

Beispiel: Viterbi-Algorithmus Schritt - 1



Beispiel-Bitfolge: 1|0|1|0

In Codierer

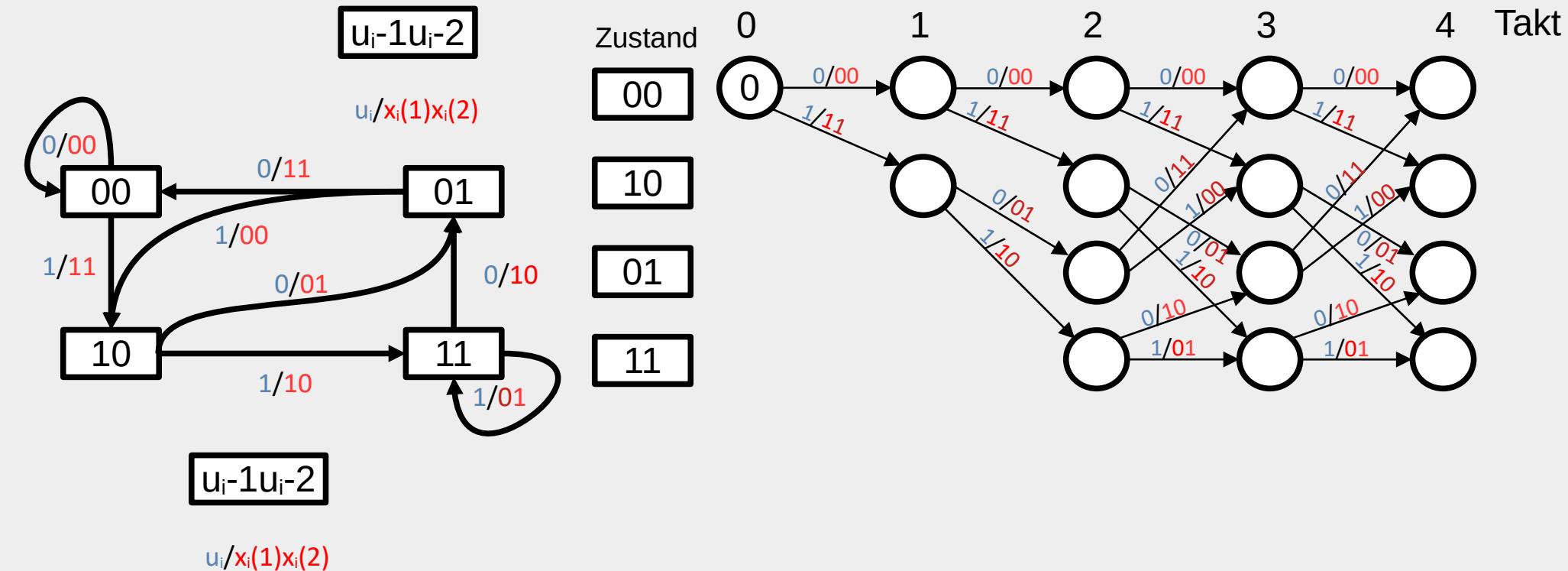
i	init	1	2	3	4
u_i	0	1	0	1	0
u_{i-1}	0	0	1	0	1
u_{i-2}	0	0	0	1	0
$x_i(1)$	0	1	0	0	0
$x_i(2)$	0	1	1	0	1

Aufgereiht

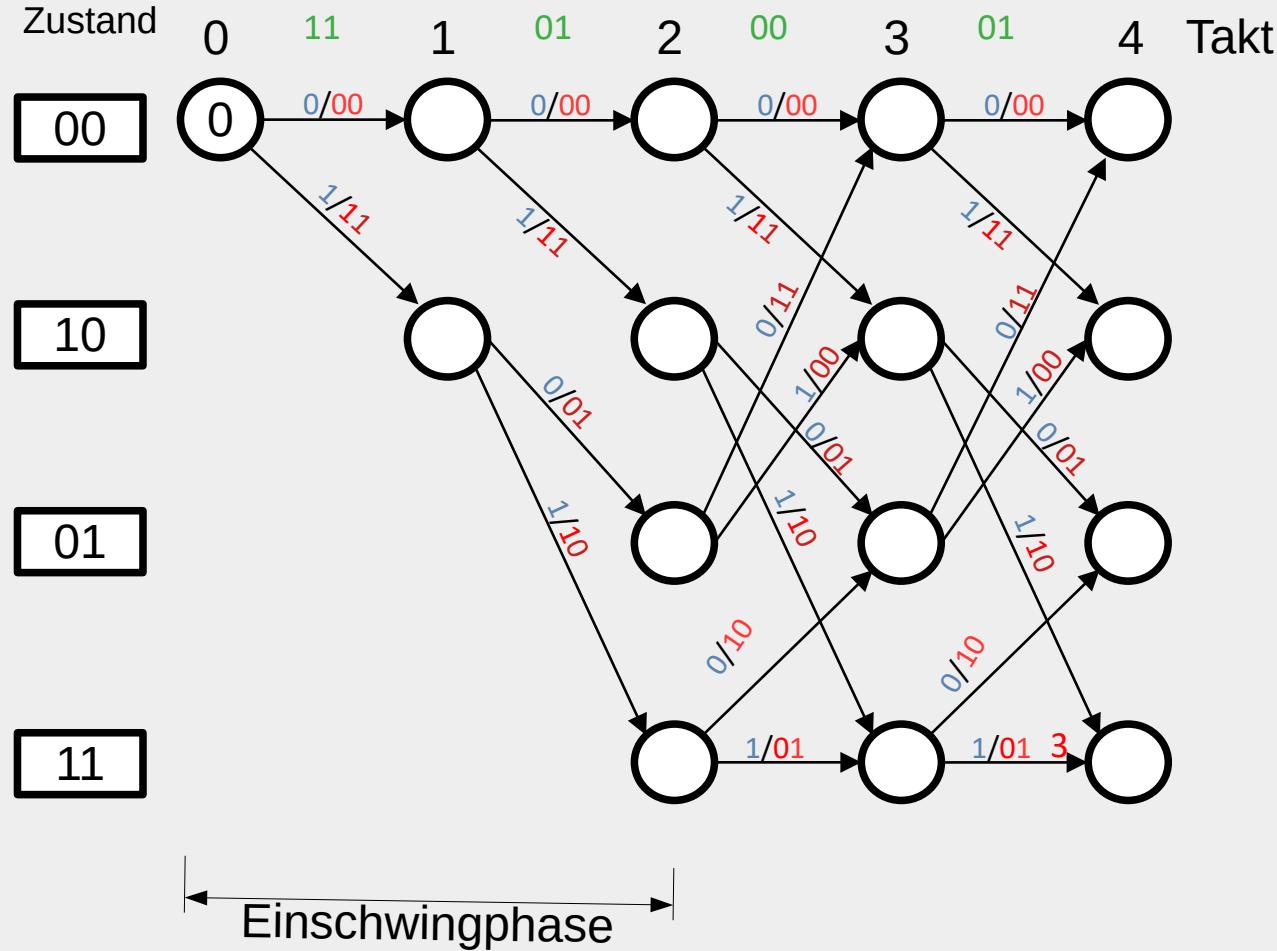
11|01|00|01

Senden

Beispiel: Viterbi-Algorithmus Schritt - 2



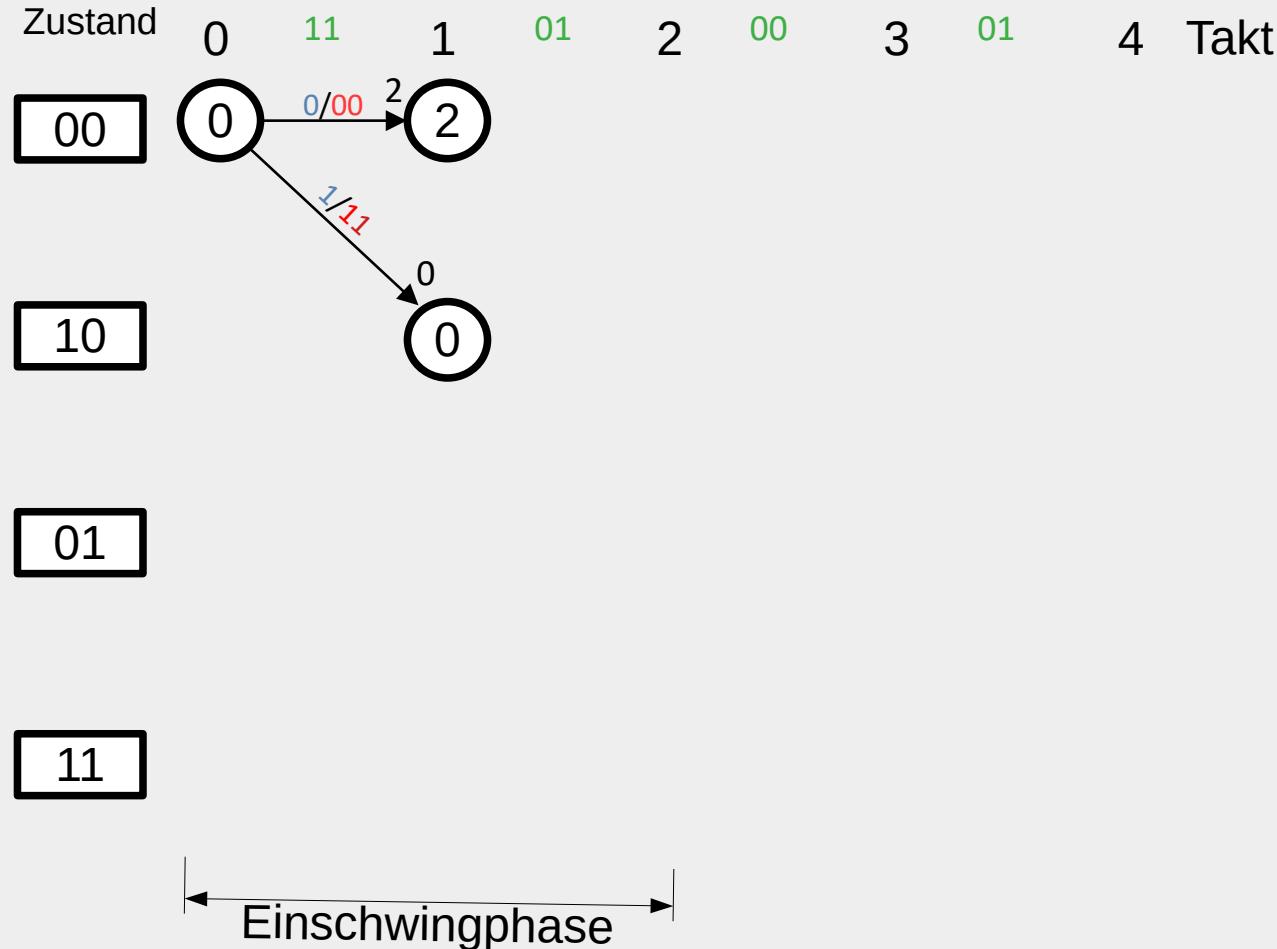
Beispiel: Viterbi-Algorithmus Schritt - 3



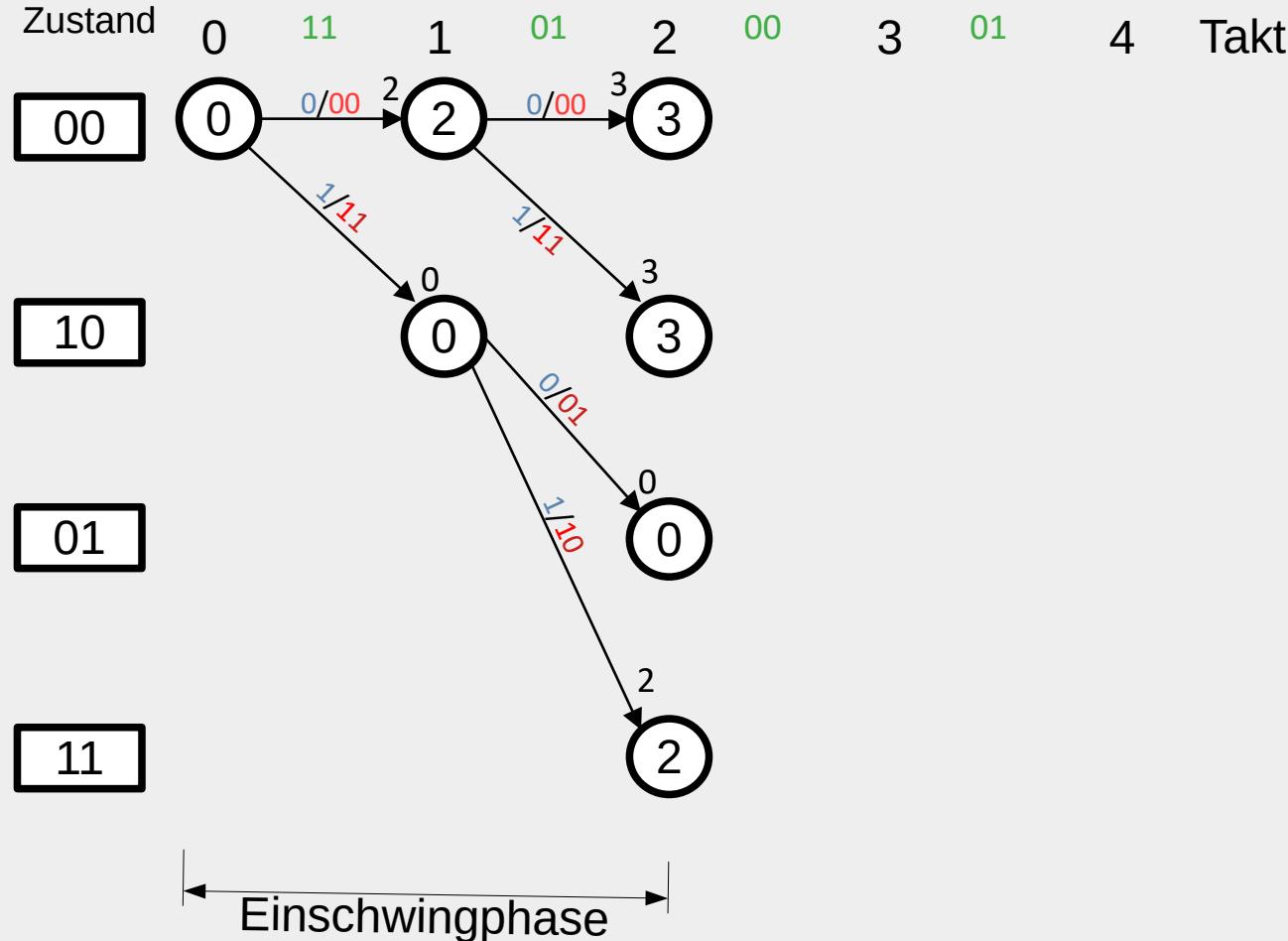
Beispiel: Viterbi-Algorithmus Schritt - 4



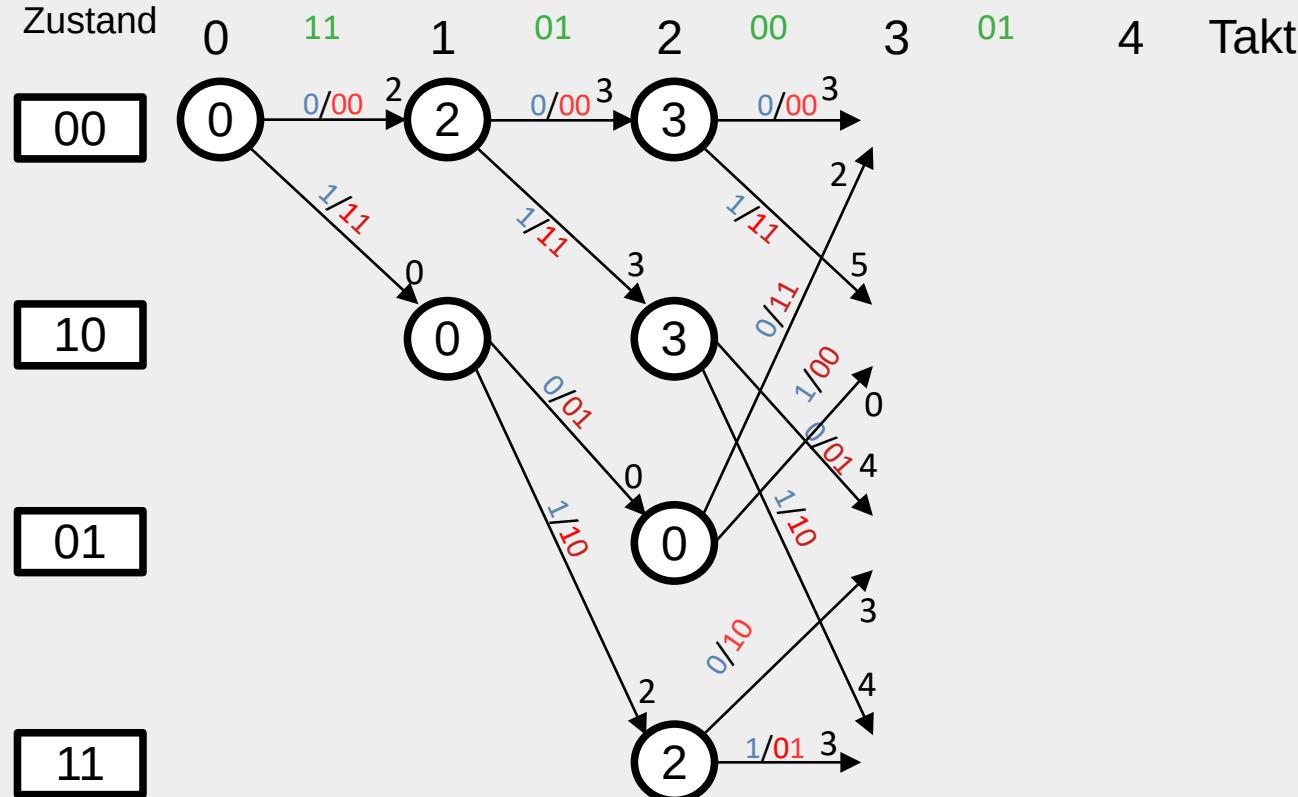
Beispiel: Viterbi-Algorithmus Schritt - 5



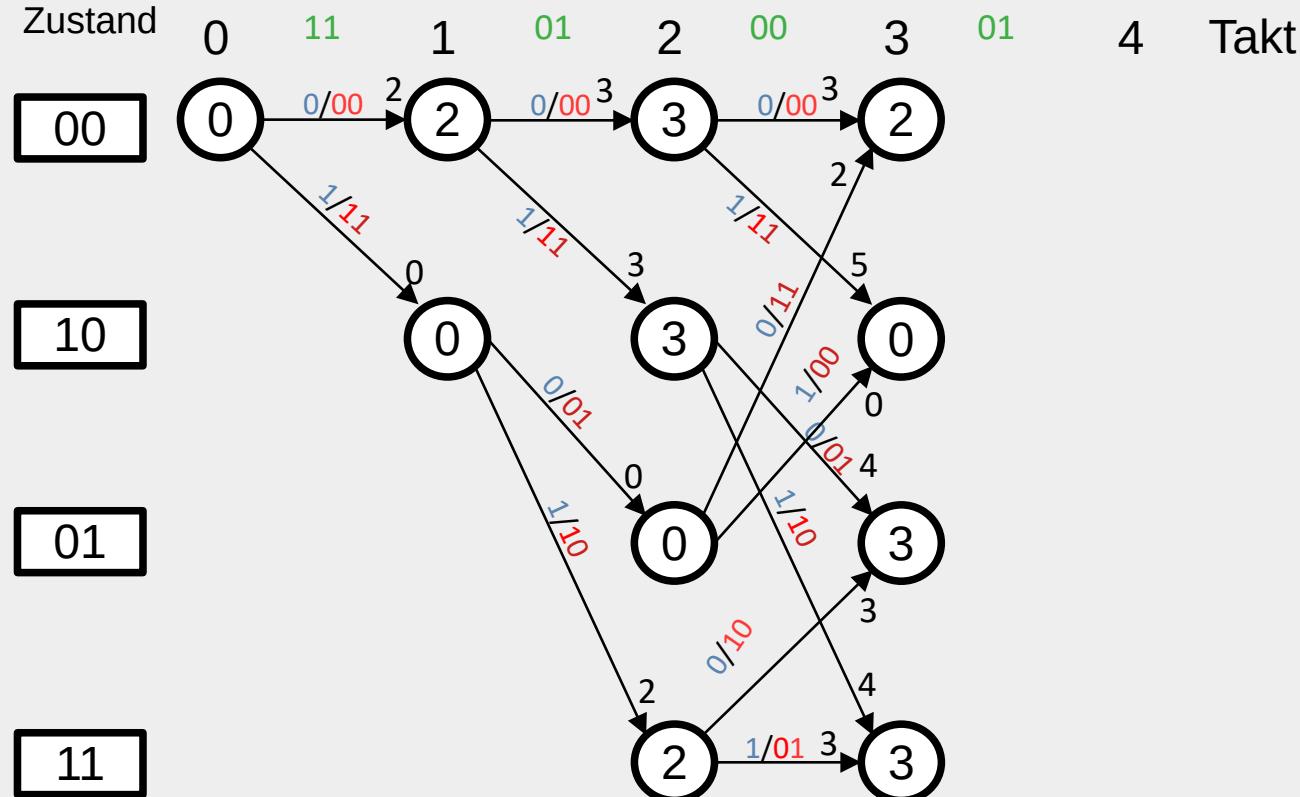
Beispiel: Viterbi-Algorithmus Schritt - 6



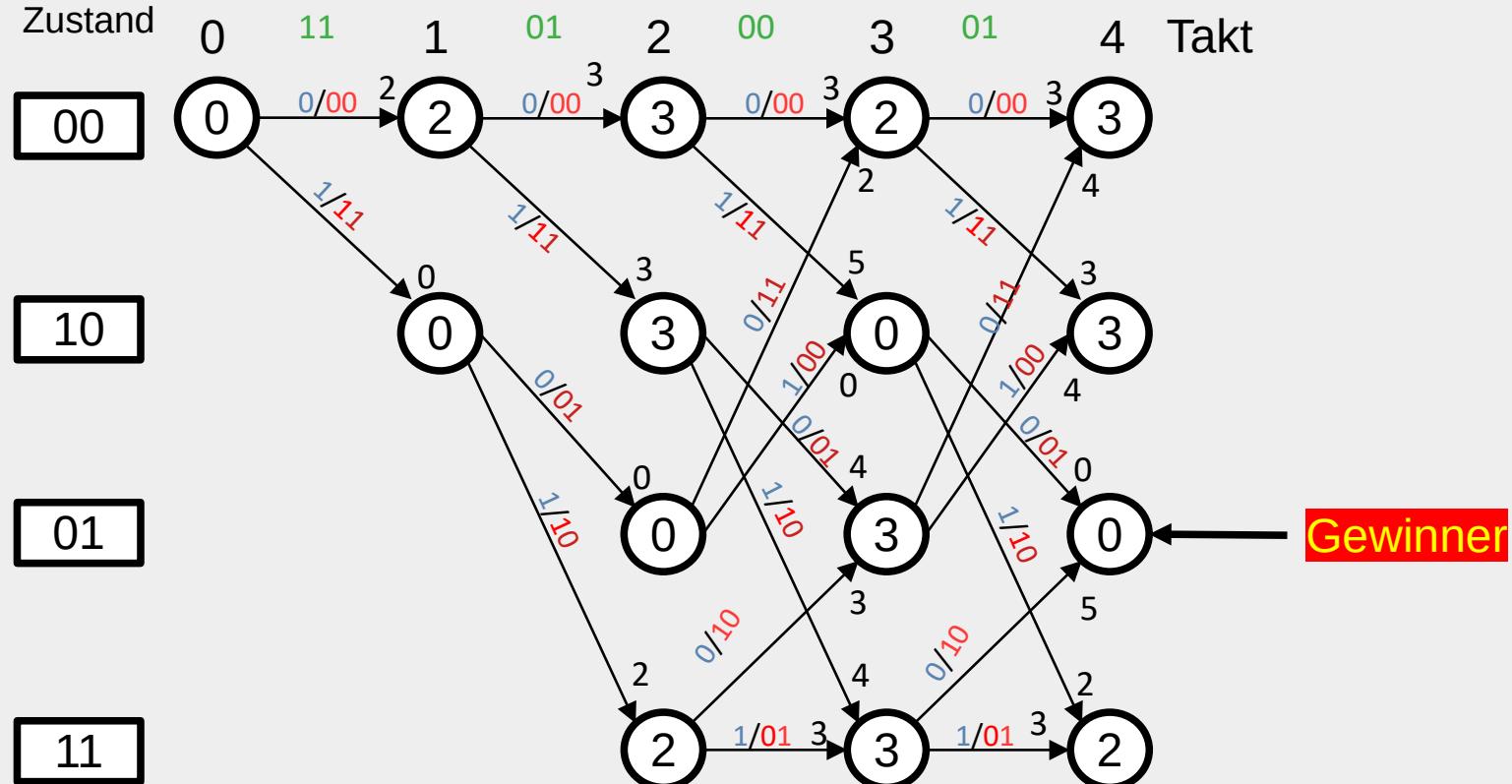
Beispiel: Viterbi-Algorithmus Schritt - 7



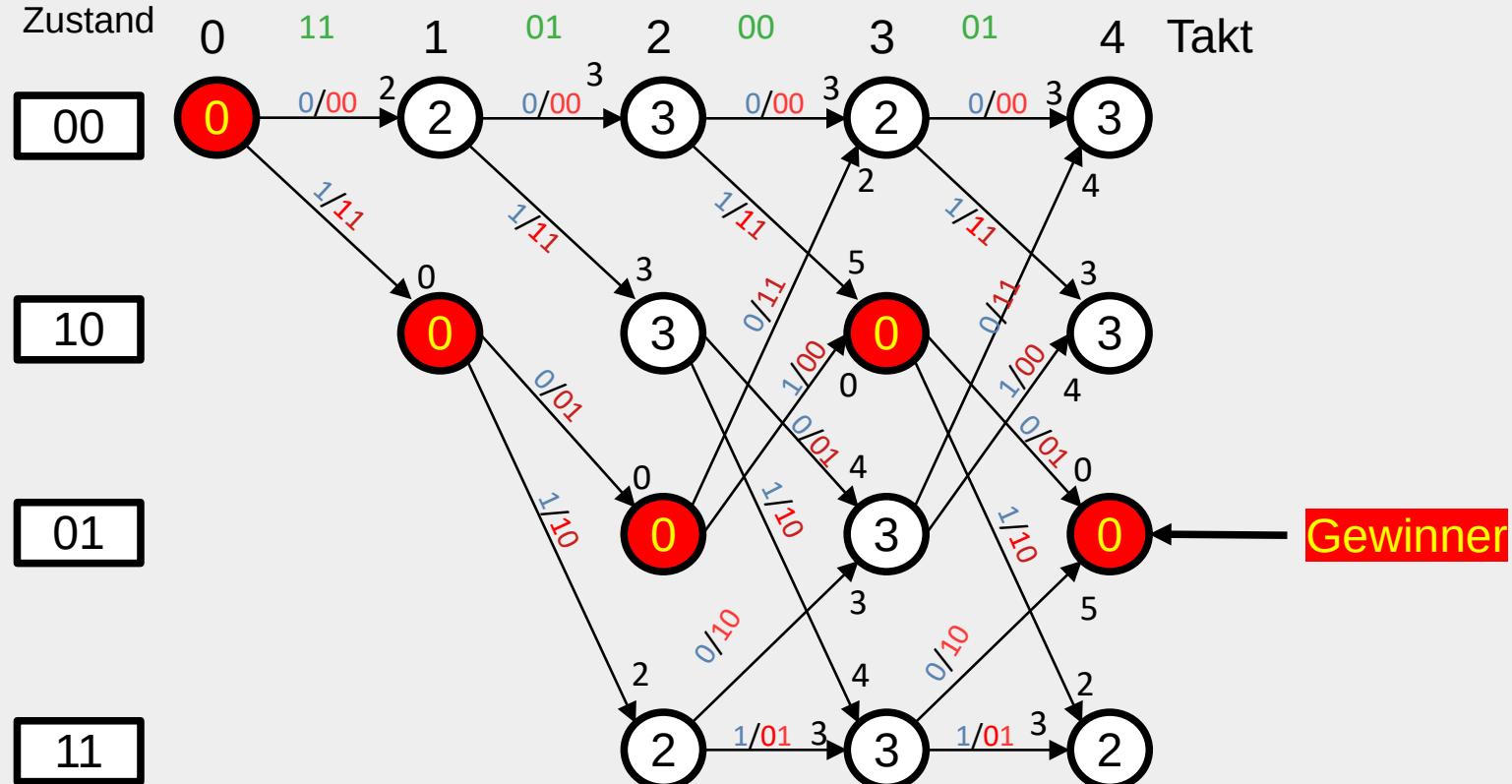
Beispiel: Viterbi-Algorithmus Schritt - 8



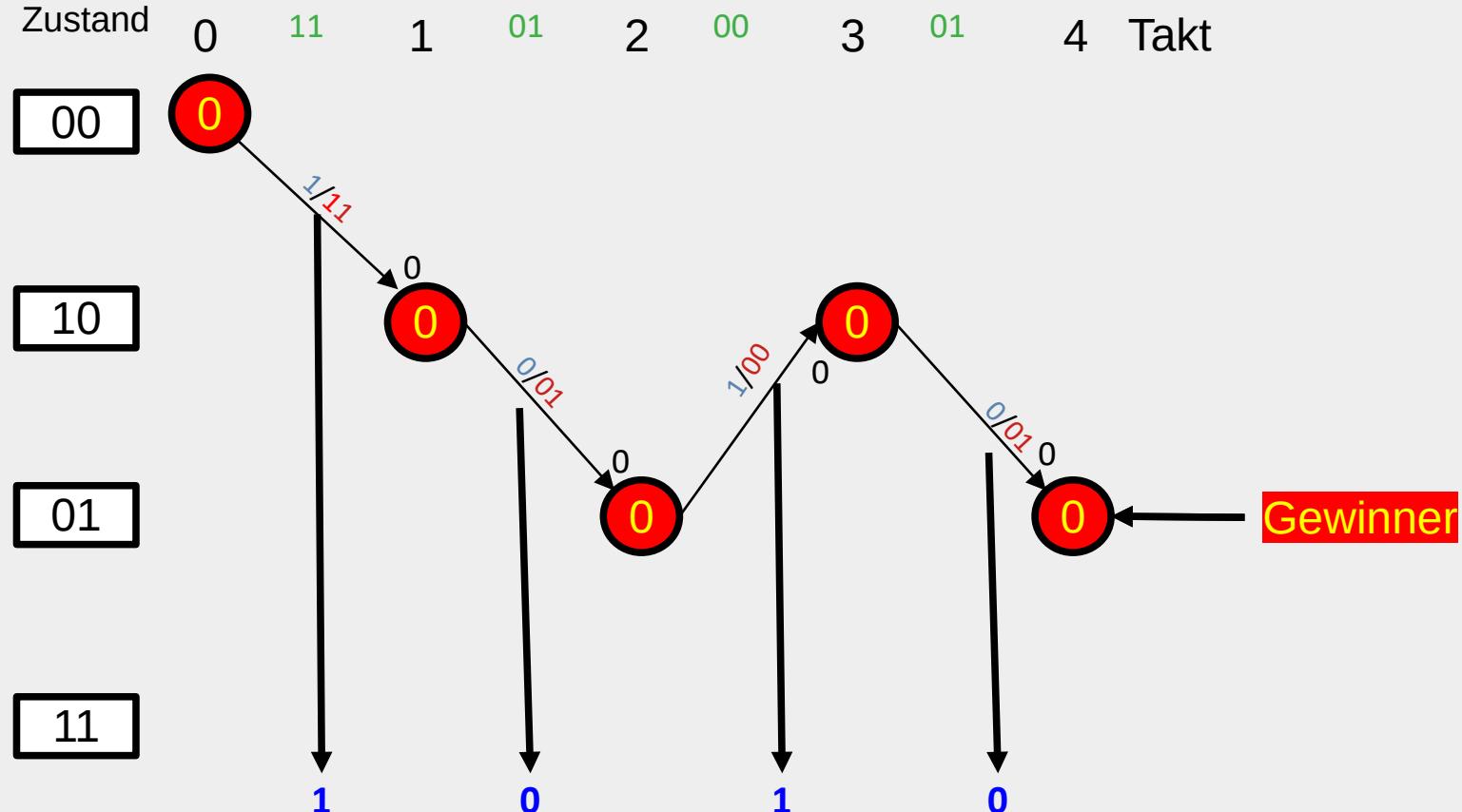
Beispiel: Viterbi-Algorithmus Schritt - 9



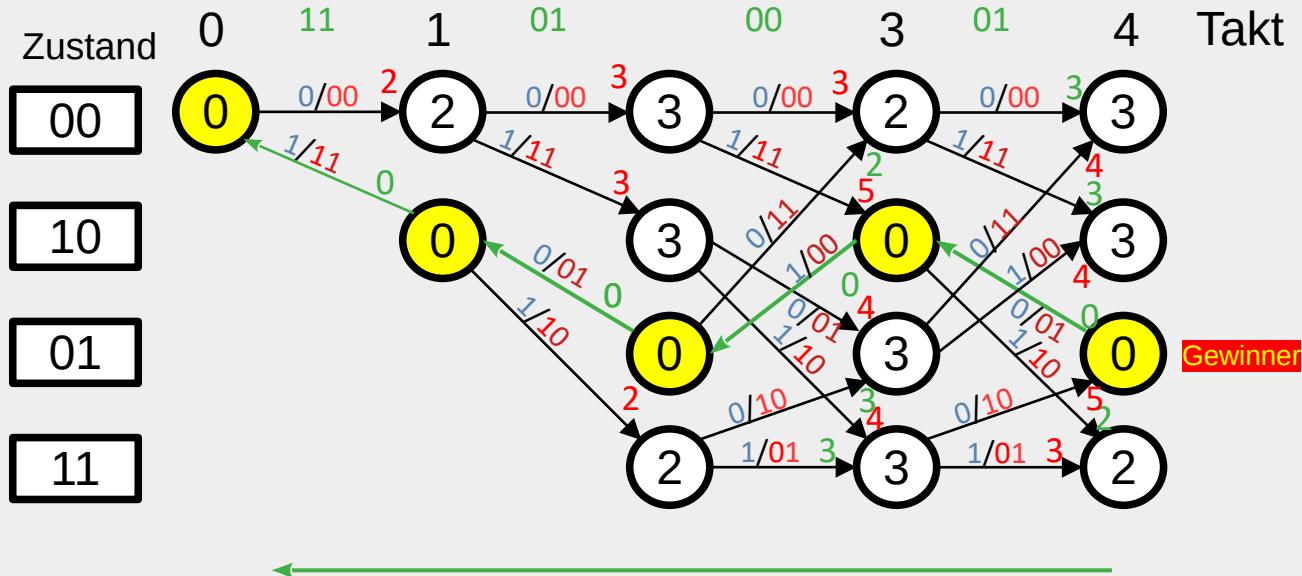
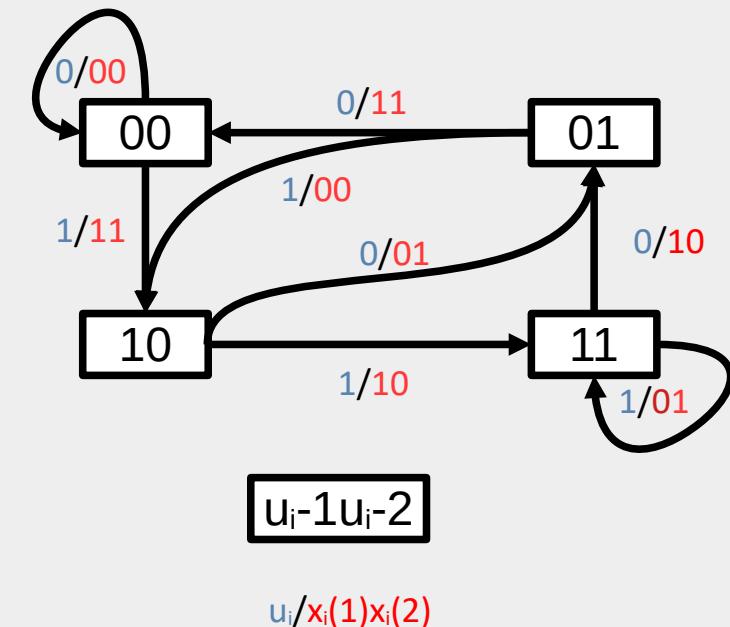
Beispiel: Viterbi-Algorithmus Schritt - 10



Beispiel: Viterbi-Algorithmus Schritt - 11



Beispiel: Viterbi-Algorithmus Zusammenfassung



Backtracking vom Gewinner zum Anfang

Vom Anfang an ausgehend werden die Eingangswerte übernommen

Ergebnis-Bitfolge: 1|0|1|0

Beispiel-Bitfolge: 1|0|1|0

Beispiel: Faltungscodierer mit Punktierung Schritt - 1

Beispiel-Bitfolge: 1|0|1|1|0|1



i	1	2	3	4	5	6
u_i	1	0	1	1	0	1
u_{i-1}	0	1	0	1	1	0
u_{i-2}	0	0	1	0	1	1
$x_i(1)$	1	0	0	1	1	0
$x_i(2)$	1	1	0	0	0	0

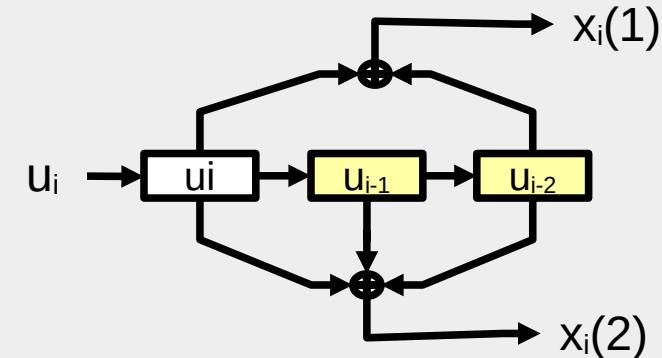


1	0	0	1	1	0
1	1	0	0	0	0

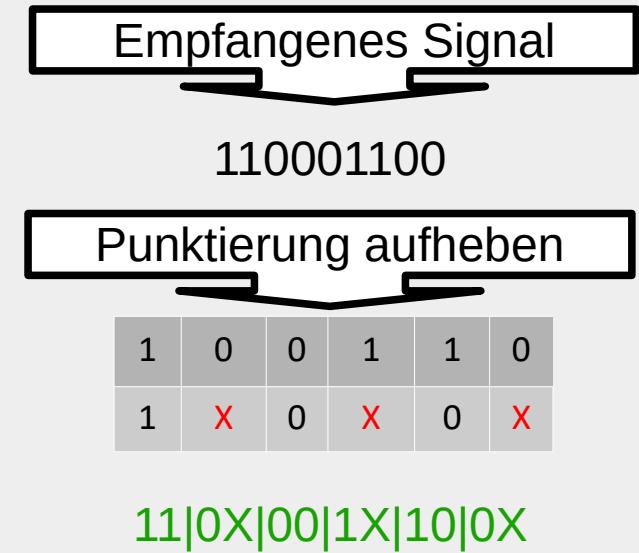
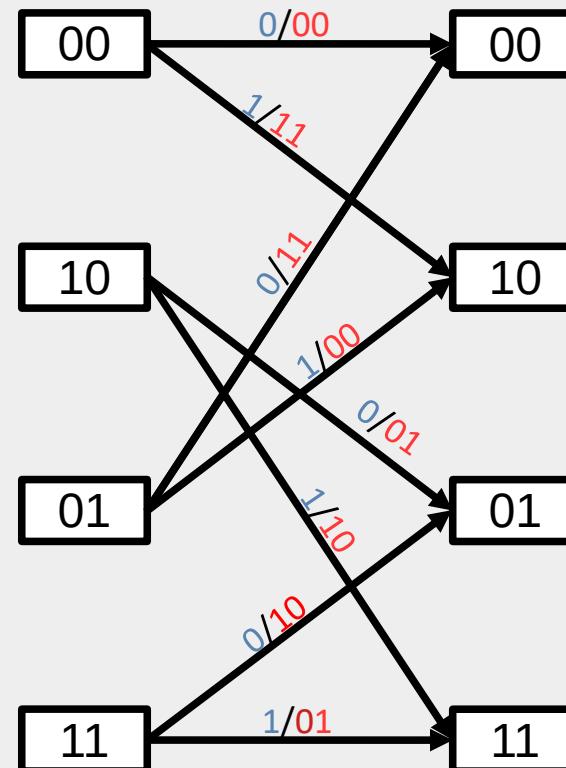
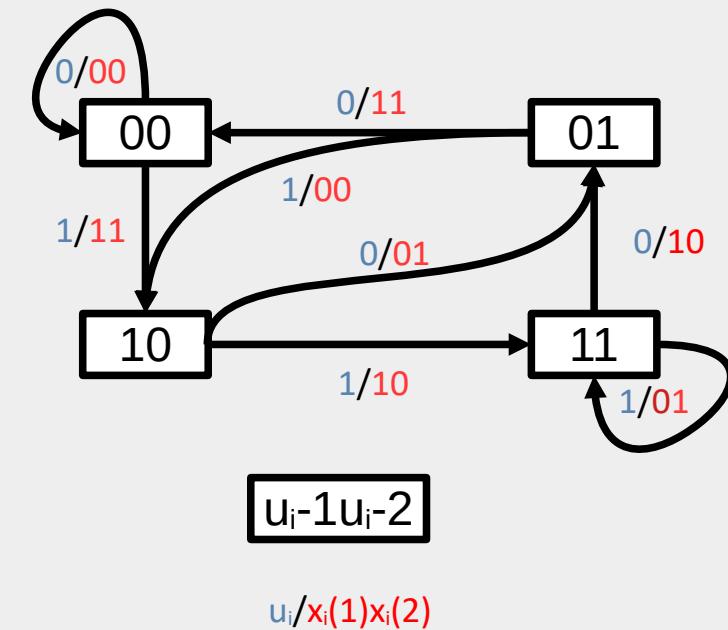


110001100

Senden

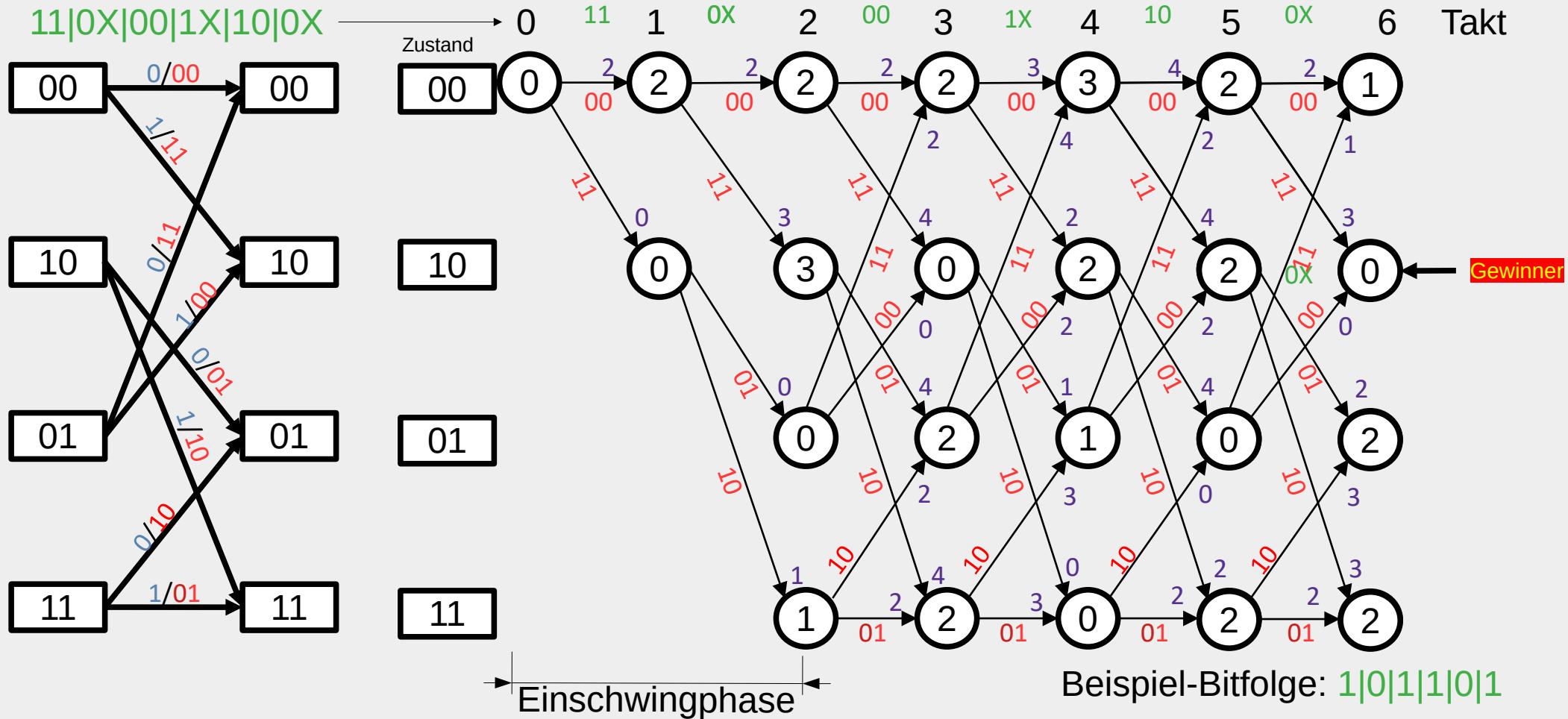


Beispiel: Faltungscodierer mit Punktierung Schritt - 2

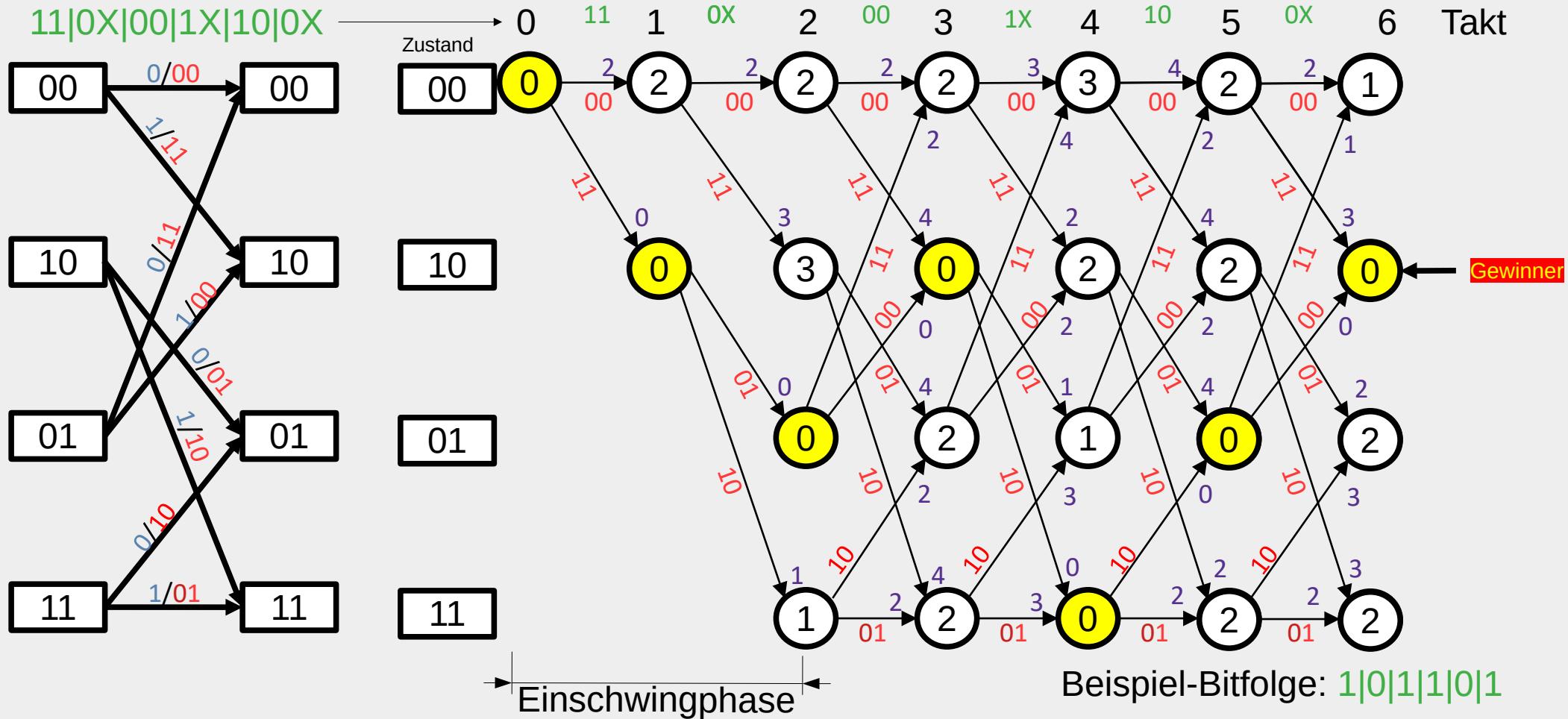


Beispiel-Bitfolge: 1|0|1|1|0|1

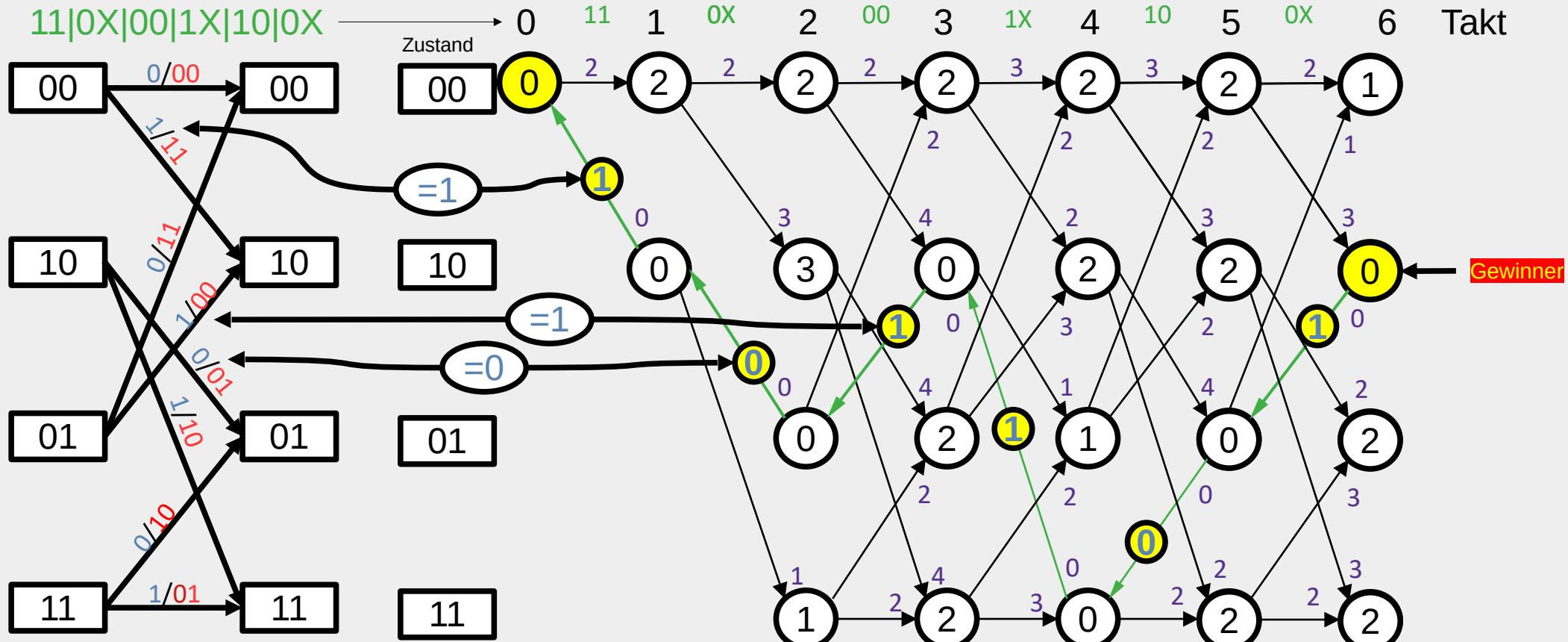
Beispiel: Faltungscodierer mit Punktierung Schritt - 3



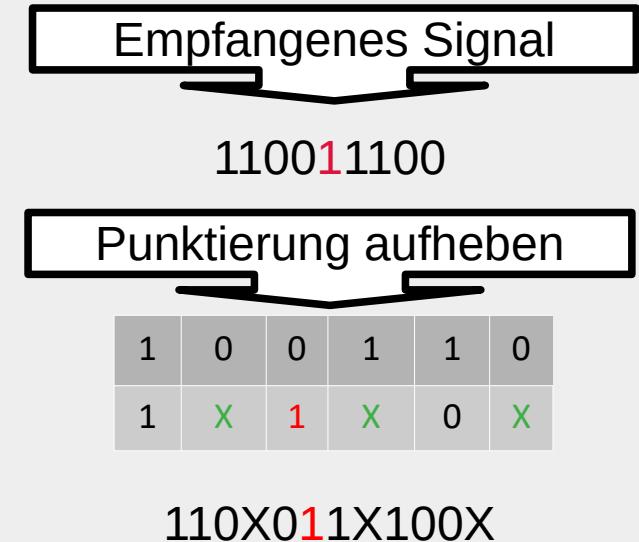
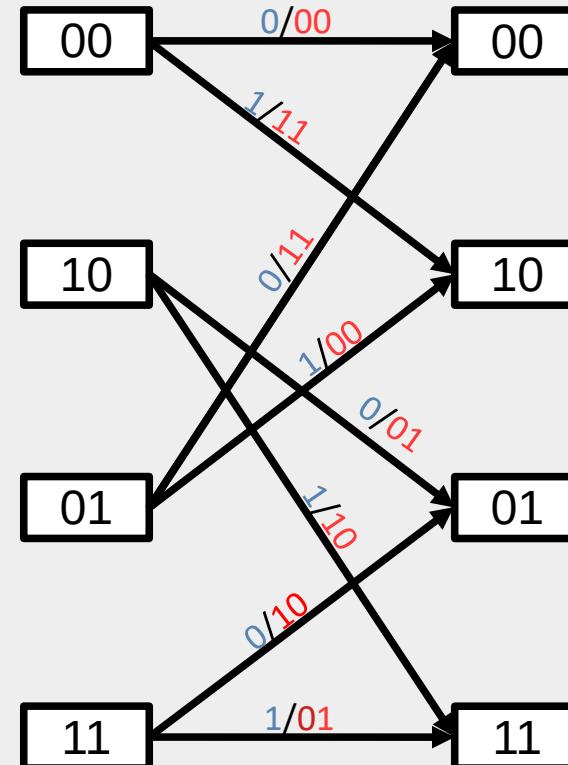
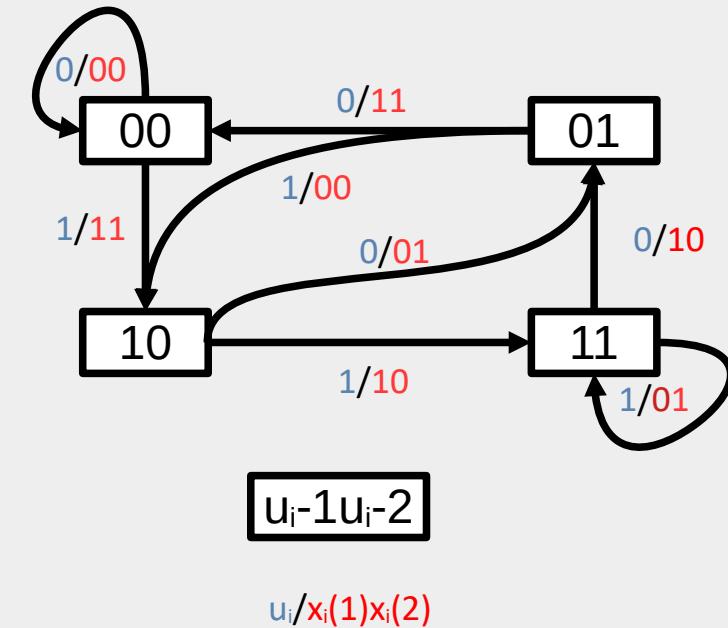
Beispiel: Faltungscodierer mit Punktierung Schritt - 4



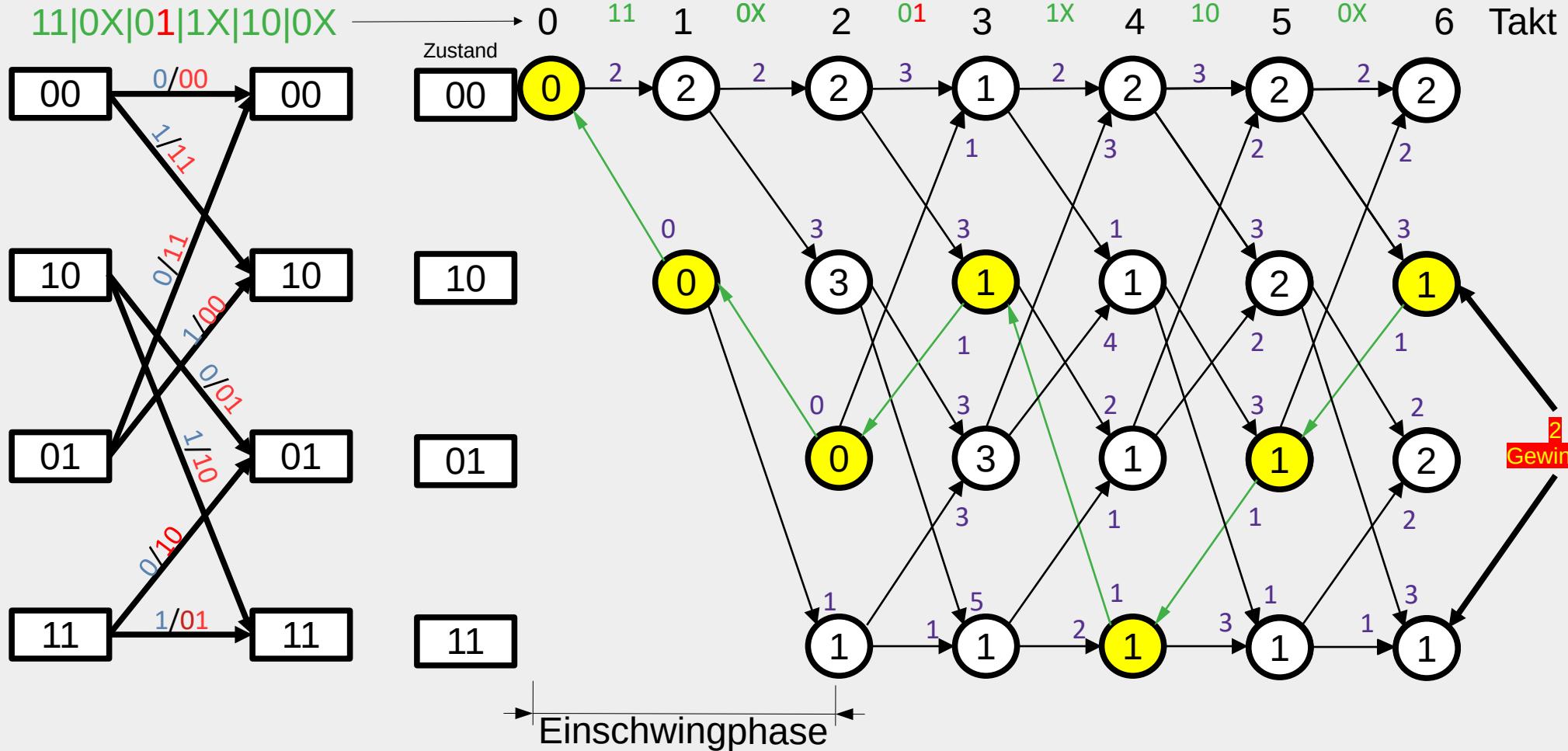
Beispiel: Faltungscodierer mit Punktierung Schritt - 5



Beispiel: Faltungscodierer mit verfälschtem Bit Schritt - 1

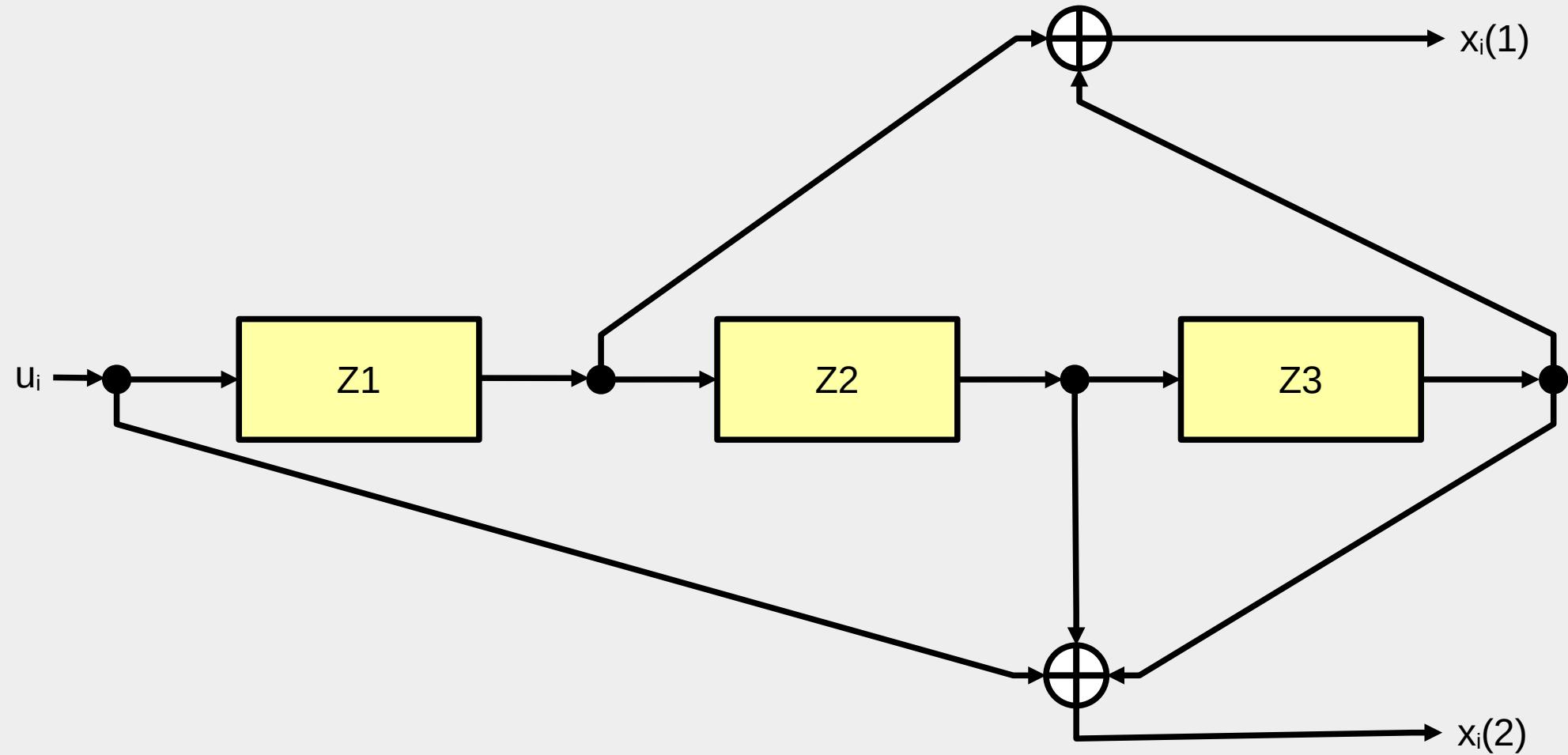


Beispiel: Faltungscodierer mit verfälschtem Bit Schritt - 2

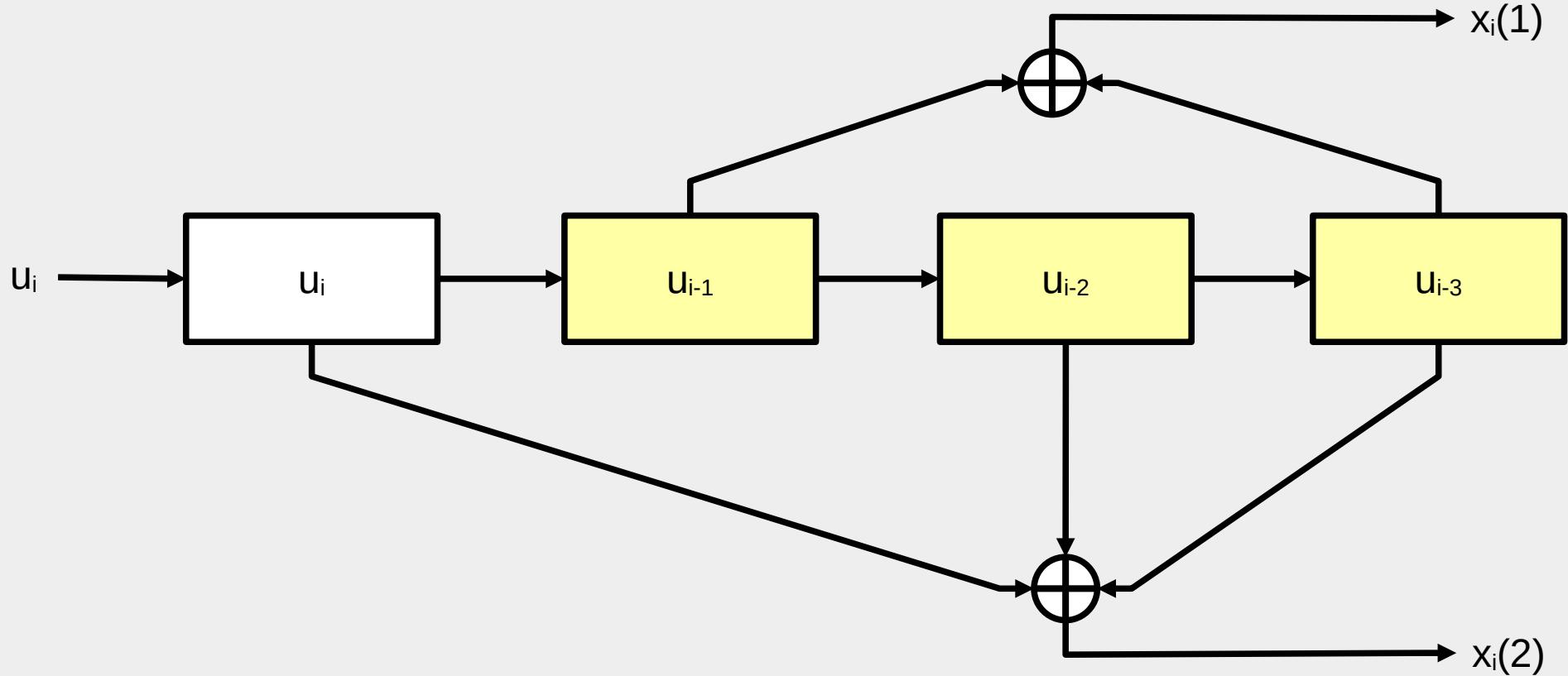


Beispiel: Faltungscodierer mit 3 Speichern

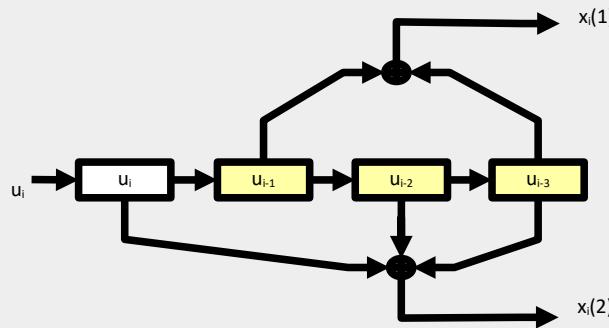
Schritt - 1



Beispiel: Faltungscodierer mit 3 Speichern Schritt - 2



Beispiel: Faltungscodierer mit 3 Sperichern Schritt - 3



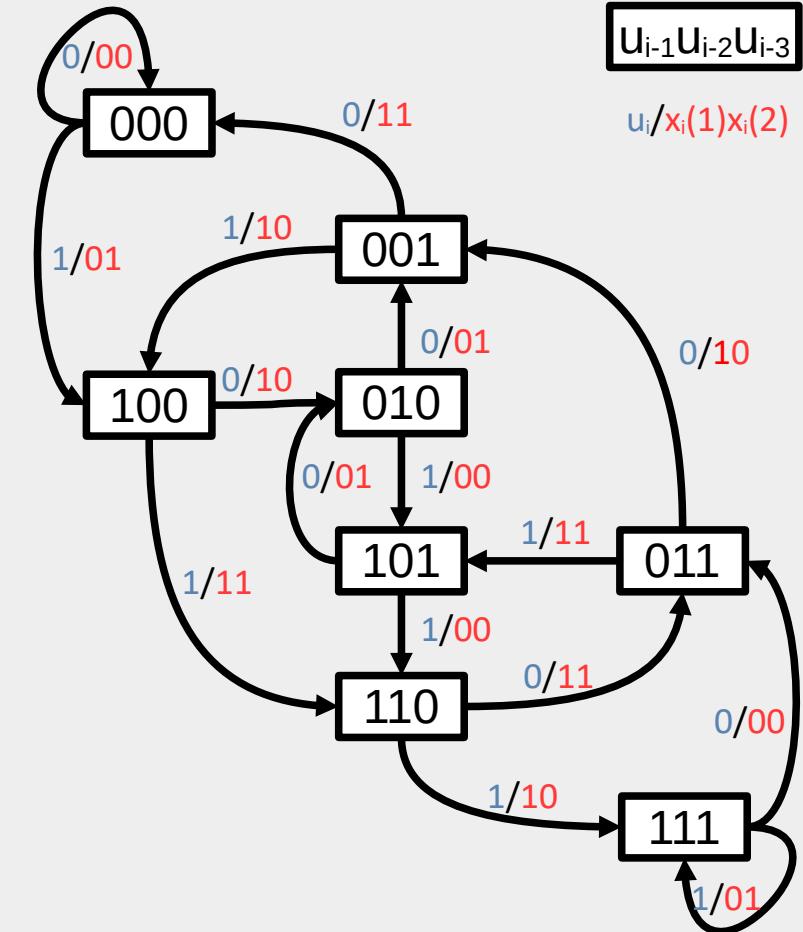
u_i		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
u_{i-1}	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
u_{i-2}	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
u_{i-3}	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$x_i(1)$	0	0	0	1	1	0	0	1	1	1	0	0	1	1	0	0	0
$x_i(2)$	0	0	1	0	1	1	0	1	0	1	0	1	0	0	1	0	1

Beispiel: Faltungscodierer mit 3 Speichern Schritt - 4

u_i		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
u_{i-1}	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
u_{i-2}	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
u_{i-3}	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$x_i(1)$	0	0	0	1	1	0	0	1	1	1	0	0	1	1	0	0	0
$x_i(2)$	0	0	1	0	1	1	0	1	0	1	0	0	1	0	1	0	1

Beispiel-Bitfolge: 0-0-1-1-0-1-0-1-1

i	1	2	3	4	5	6	7	8	9
u_i	0	0	1	1	0	1	0	1	1
u_{i-1}	0	0	0	1	1	0	1	0	1
u_{i-2}	0	0	0	0	1	1	0	1	0
u_{i-3}	0	0	0	0	0	1	1	0	1
$x_i(1)$	0	0	0	1	1	1	0	0	0
$x_i(2)$	0	0	1	1	1	1	1	0	0



Beispiel: Faltungscodierer mit 3 Sperichern Schritt - 5

i	1	2	3	4	5	6	7	8	9
u_i	0	0	1	1	0	1	0	1	1
$x_i(1)$	0	0	0	1	1	1	0	0	0
$x_i(2)$	0	0	1	1	1	1	1	0	0

Aufgereiht

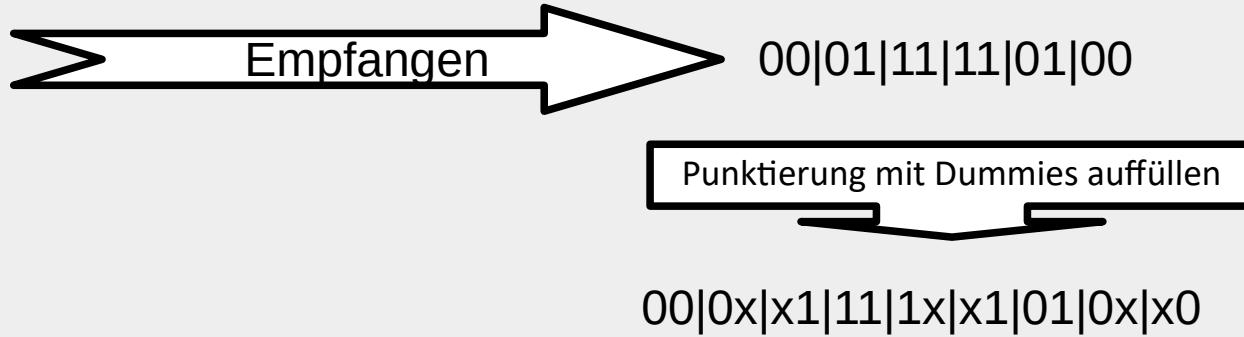
00|00|01|11|11|11|01|00|00

Punktierung

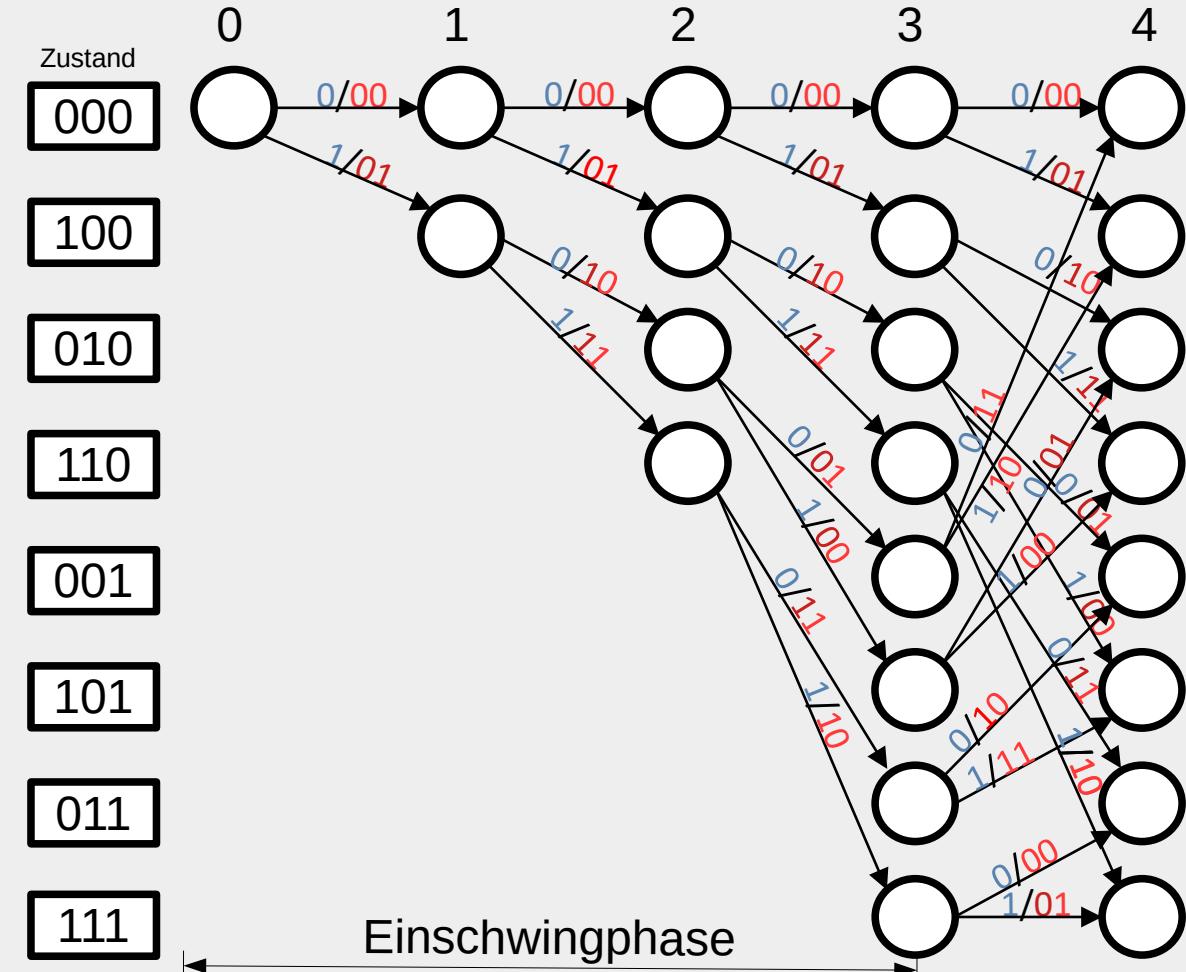
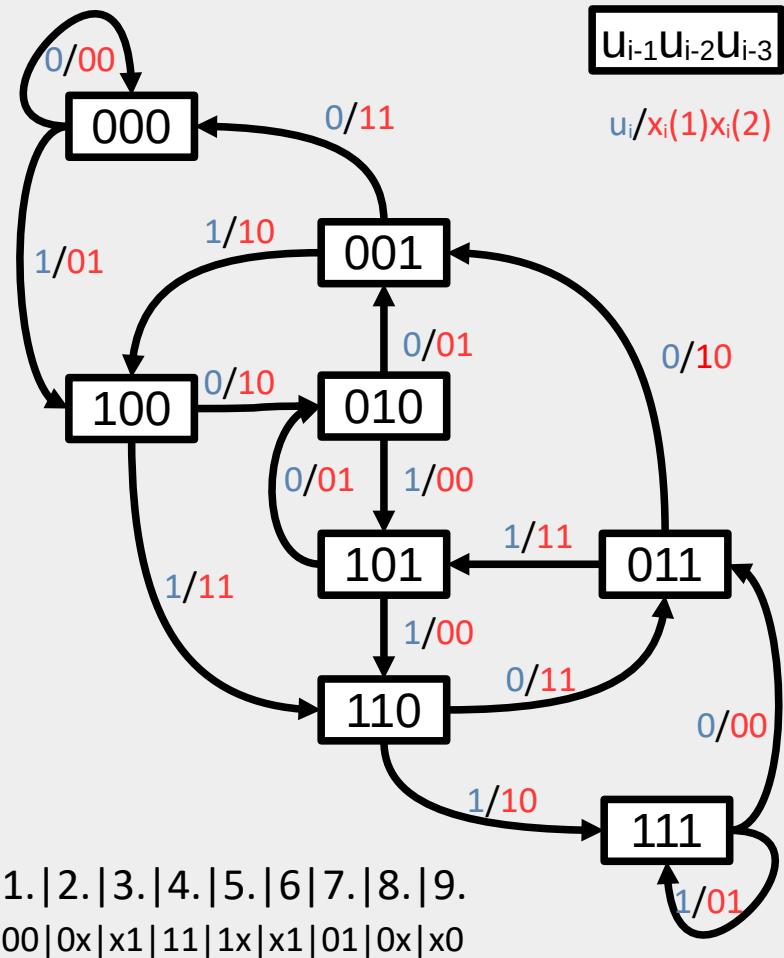
00|01|11|11|01|00

Senden

Beispiel: Faltungscodierer mit 3 Sperichern Schritt - 6

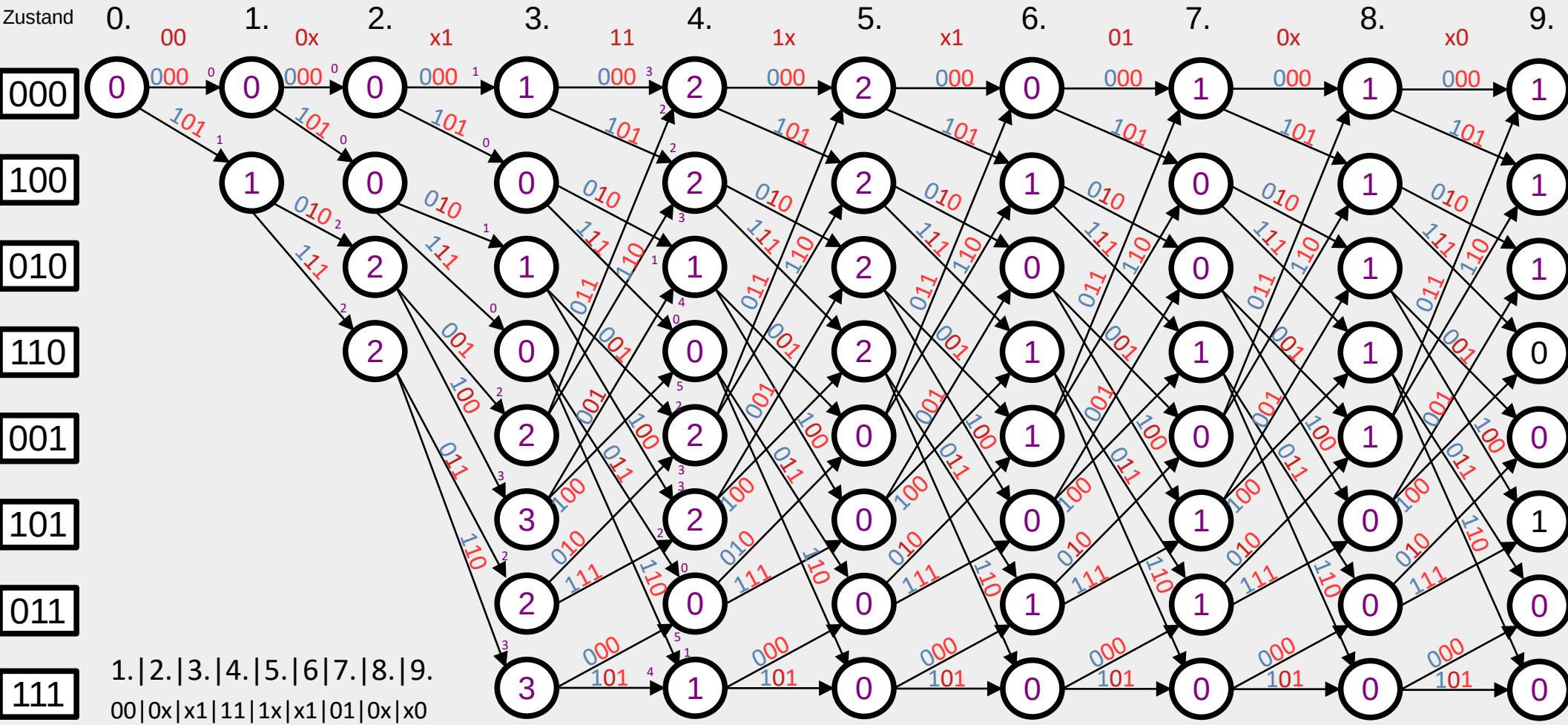


Beispiel: Faltungscodierer mit 3 Sperichern Schritt - 7

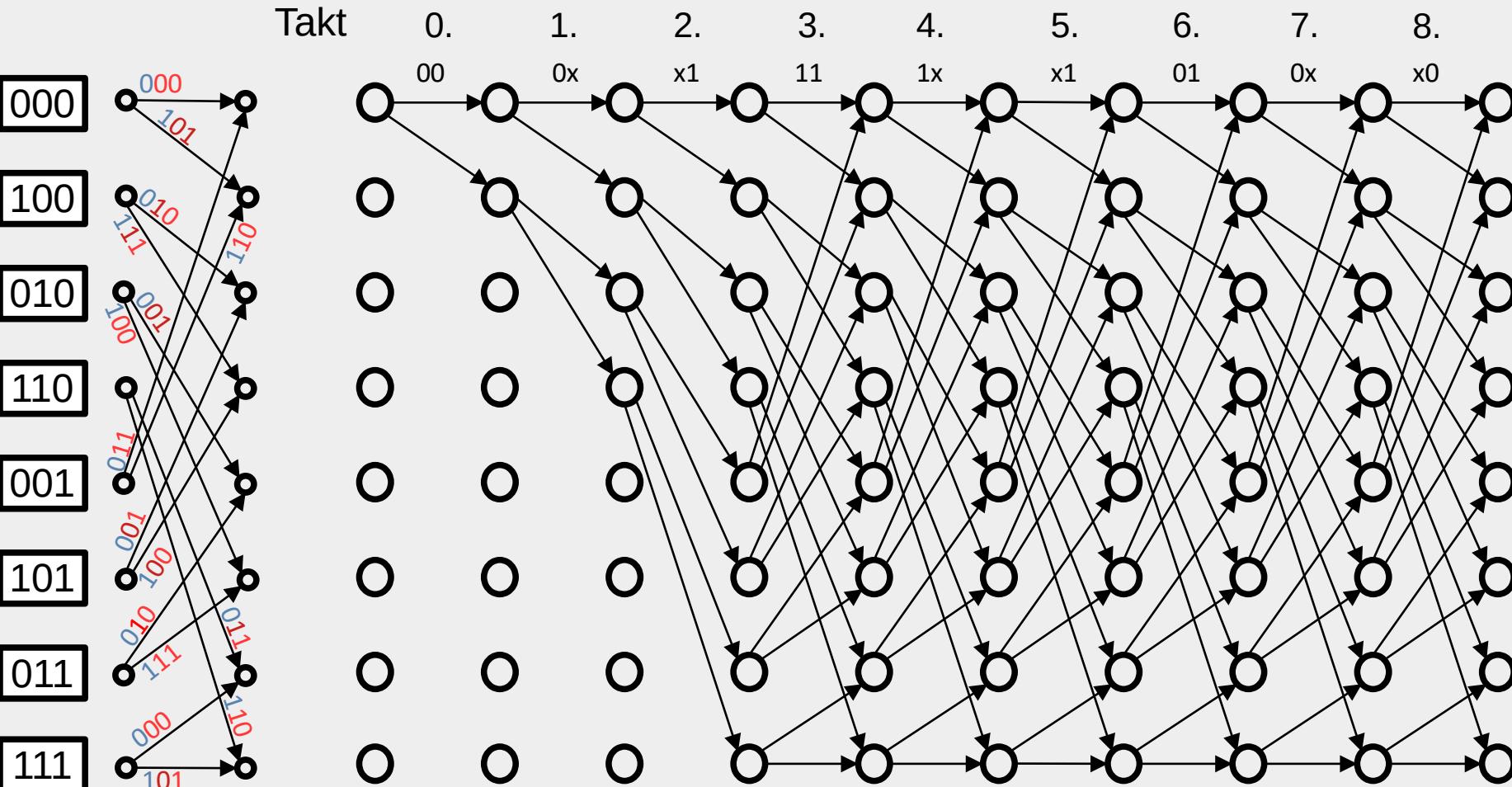


Beispiel: Faltungscodierer mit 3 Sperichern

Schritt - 8

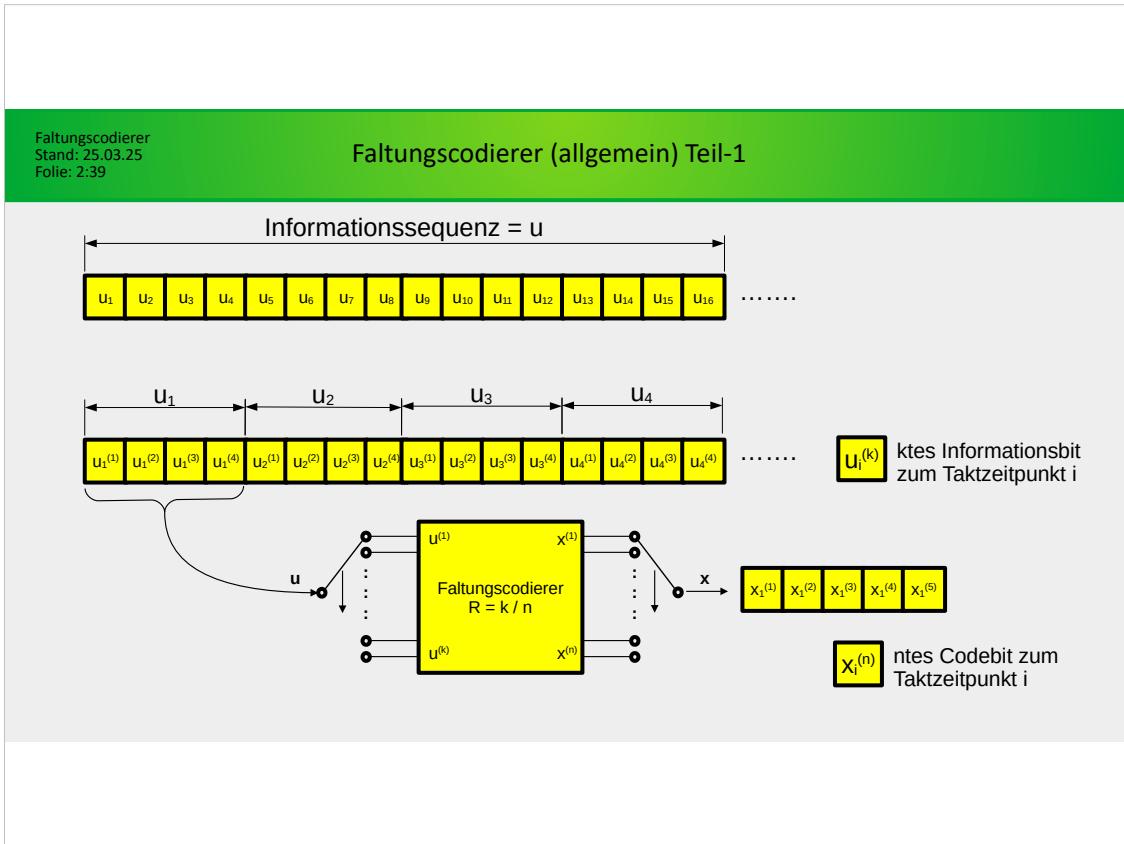


Beispiel: Faltungscodierer mit 3 Sperichern Schritt - 9



Faltungscodierer

Stand: 25.03.25



Allgemeine Betrachtung:

Ein Faltungscodierer erzeugt einen Faltungscode (engl. convolutional code) indem er Informationsbits in einem Speicher (Schieberegister) zwischenspeichert und mit einer Verknüpfungslogik zu einem Codewort zusammenstellt.

Zu Beginn wird eine unendlich lange binäre Informationssequenz (u) angenommen.

Die Informationssequenz wird in Informationsblöcke der Größe $k = 4$ aufgeteilt.

Am Anfang (beim Takt $i = 0$) werden die Speicherzellen des Faltungscodierers (normalerweise mit Nullen) initialisiert.

Pro Takt wird dem Faltungscodierer ein Informationsblock mit $k = 4$ **Informationsbits** zugeführt.

Pro Takt wird am Ausgang ein Codeblock mit $n = 5$ **Codebits** ausgegeben.

Im obigen Beispiel ist der Takt $i = 1$ angenommen.

Damit lässt sich die **Coderate** mit $r = k / n = 4 / 5$ bestimmen.

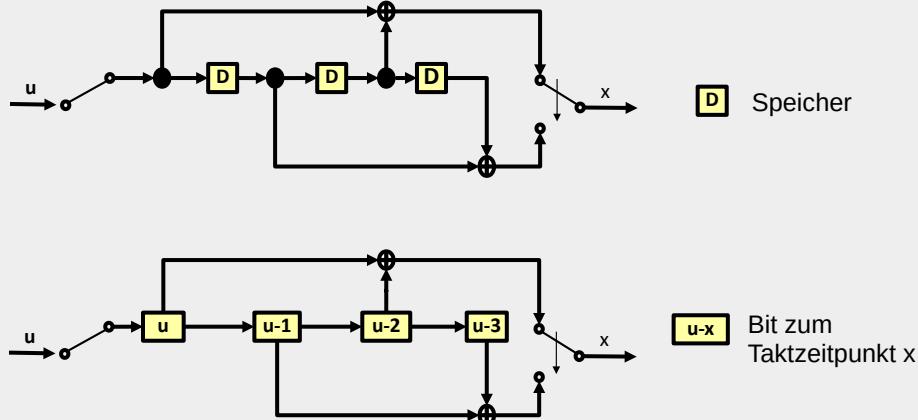
Der Speicher kann aus einem oder mehreren Flipflops aufgebaut sein. Er kann auch aus mehreren parallelen Ketten bestehen. Die Verknüpfungslogik entnimmt an diversen Punkten (Eingang oder Speicherausgängen) die durchgeschobenen Bits und verknüpft sie mit einer Modulo-2 Operation.

Auf der Ausgangsseite können ein oder mehrere codierte Bits pro Takt entnommen werden.

Eigenschaften:

- Informationsbits sind explizit im Codewort enthalten.
- Folgezustände hängen vom aktuellen Zustand, dem Eingangswert und der Rückkopplungsstruktur ab.
- Um definierte Zustände zu erreichen werden Faltungscodierer mit Nullen initialisiert und den Informationsbits werden **Tailbits** (Nullen) angehängt.
- Hauptmerkmal von Faltungscodes: Im Unterschied zu Blockcodes beruht die Decodierung nicht auf einer Prüfgleichung sondern auf der optimalen Schätzung der Empfangsfolge.

Faltungscodierer (allgemein) Teil-2



Die Innereien eines Faltungscodierers können unterschiedlich dargestellt werden.

In beiden Beispielen geht es um den selben Faltungscodierer. Den Faltungscodierern wird jeweils am Eingang pro Takt jeweils $k = 1$ Bit zugeführt.
Am Ausgang werden pro Takt $n = 2$ Bits erzeugt.
Damit ist die Coderate $r = k/n = 1/2$

In der obigen Abbildung ist ein Faltungscodierer mit 3 Speichern dargestellt. Damit hat das **Gedächtnis** den Wert $m = 3$ (memory)
In der Literatur wird das auch als **Eindringtiefe** (b) bezeichnet, denn ein zugeführtes Bit dringt in 3 Speicher vor.
Hat ein Faltungscodierer mehrere Speicherketten mit unterschiedlicher Länge, so gilt für die Ermittlung der Eindringtiefe die längste Kette.

Im unteren Faltungscodierer sind nicht mehr die Speicher explizit dargestellt, sondern nur noch die Werte zu einem Taktzeitpunkt welche verknüpft werden.
Dabei sind die Positionen eines Bits zu einem entsprechenden Taktzeitpunkt dargestellt.
Hier ist zu erkennen, dass ein Bit an 4 Stellen den Ausgang beeinflusst. (Direkt am Eingang und jeweils nach einem Speicher).

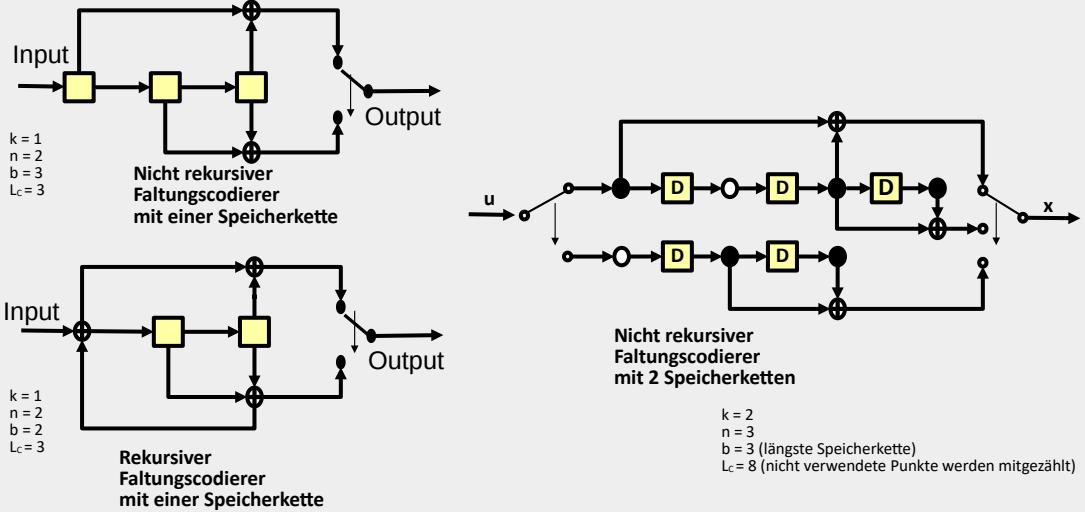
Je nach Darstellung (oben oder unten) finden sich in der Literatur unterschiedliche Festlegungen für die **Einflusslänge**. Damit wird beschrieben, wie oft ein Eingangsbit über die Takte hinweg das Ergebnis beeinflussen kann..

Oben: Zählung der Speicherelemente
Die **Einflusslänge** (L_c) ergibt sich aus der Summe aller Speicherelemente multipliziert mit der Anzahl der Informationsbits pro Takt. $L_c = b * k$. Damit ist die Einflusslänge $L_c = 3 * 1 = 3$

Unten: Summe der Takte, die für das Erzeugen des Codewortes relevant sind (b), multipliziert mit Der Anzahl der Informationsbits pro Takt (k). Hierbei sind 4 Stellen für die Bildung des Codewortes relevant. Damit errechnet sich die **Einflusslänge** $L_c = (b + 1) * k = 4 * 1 = 4$

Da die untere Interpretation der Einflusslänge (L_c) für die weitere Bearbeitung besser ist, soll sie ab hier verwendet werden.

Faltungscodierer (allgemein) Teil-3



Bauformen:

Es gibt:

- Nicht rekursive Faltungscodierer. Hier werden die Informationsbits nur durchgeschoben.
- Rekursive Faltungscodierer.
Hierbei wird ein Teil der Information wieder auf den Eingang zu rückgeführt und mitverwendet.

Faltungscodierer können auch aus mehreren Speicherketten bestehen, deren Länge unterschiedlich sein kann. Bei Unterschieden in der Kettenlänge wird zur Berechnung der Parameter die längste Kette verwendet.

Im rechten Beispiel gelten deshalb die folgenden Parameter

m oder $b = 3$ (längste Kette)

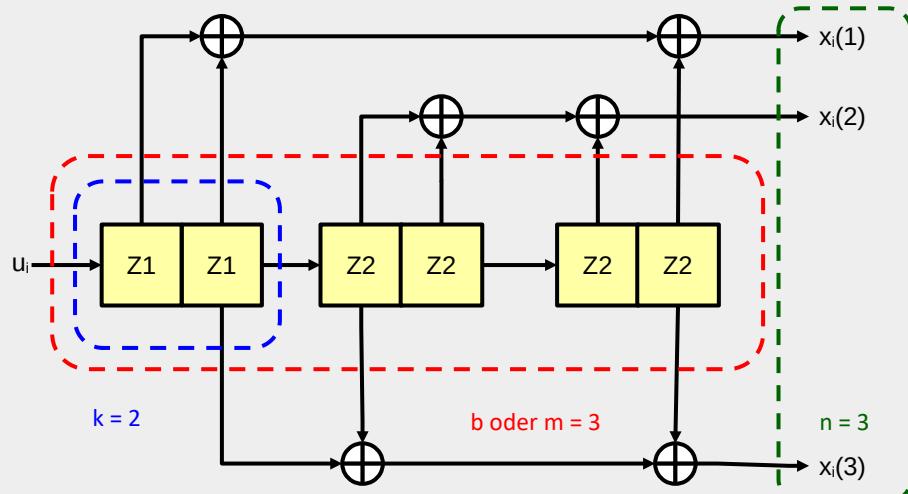
$k = 2$

$R = k/n = 2/3$

$L_c = (m + 1) * k = (3 + 1) * 2 = 8$

Bei der Ermittlung der Einflusslänge L_c werden alle (möglichen) Punkte ungeachtet deren Verwendung gezählt.

Faltungscodierer (allgemein) Teil-4



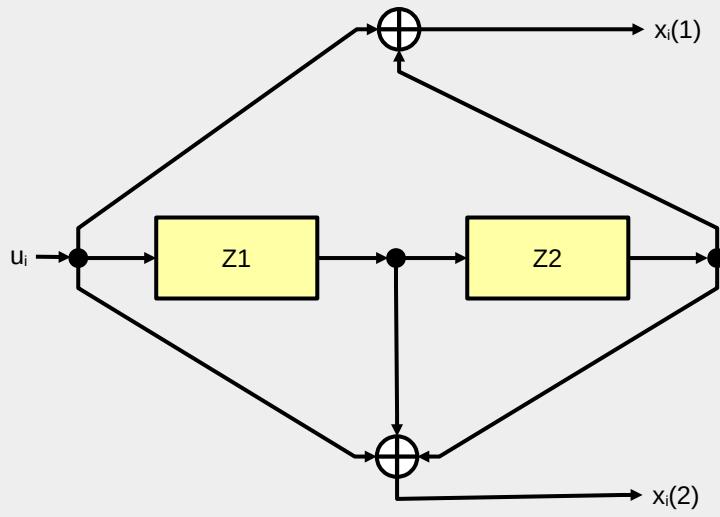
Kenngrößen von Faltungscodes:

(n, k, K) code

Wobei folgendes gilt:

- n** = Anzahl der Faltungscodes-Bits pro Takt
- k** = Anzahl der Informationsbits pro Takt
- K** = Begrenzungsfaktor / Eindringtiefe
(Länge des Schieberegisters / Anzahl der Speicherstellen (m oder b))

Faltungscodierer-Beispiel



Im Beispiel ist ein Faltungscodierer mit 2 Speicherzellen dargestellt.

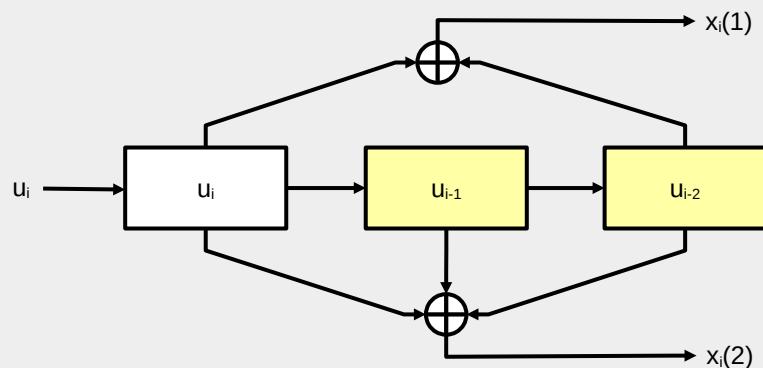
Es wird jeweils $k = 1$ Datenbit in den Codierer eingespeist.

Es werden jeweils $n = 2$ Codebits vom Codierer geliefert.

Damit ist die Coderate $r = k / n = 1 / 2$.

Durch die 2 Speicherzellen ergibt sich eine Eindringtiefe von $b = 2$.

Durch die 3 verwendeten Takte berechnet sich Einflusslänge $L_c = (b + 1) * k = (2 + 1) * 1 = 3$.



Da für die XOR-Operation an 3 Stellen die Bits abgegriffen werden, kann man den Codierer mit 3 Blöcken wie in der Folie darstellen.

i gibt den Takt an.

u_i stellt das aktuell zugeführte Datenbit dar.

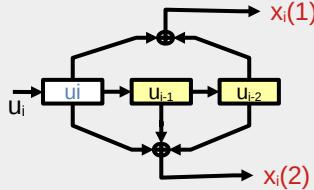
u_{i-1} gibt das vorhergehende Datenbit an. Es wird vom ersten Schieberegister gehalten.

u_{i-2} gibt das Datenbit an welches vorher von u_{i-1} gehalten wurde.
Es wird vom zweiten Schieberegister gehalten.

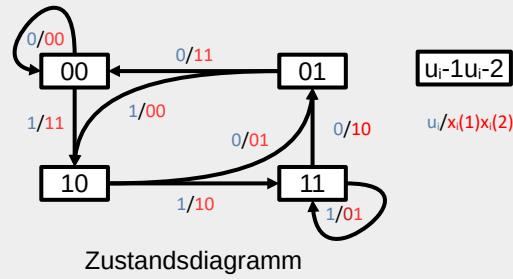
$X_i(1)$ ergibt sich als XOR-Verknüpfung von $u_i \oplus u_{i-2}$.

$X_i(2)$ ergibt sich als XOR-Verknüpfung von $u_i \oplus u_{i-1} \oplus u_{i-2}$.

Vom Faltungscodierer zum Zustandsdiagramm



u_i	0	0	1	0	1	0	1	0	1
u_{i-1}	0	0	0	1	1	0	0	1	1
u_{i-2}	0	0	0	0	0	1	1	1	1
$x_i(1)$	0	0	1	0	1	1	0	1	0
$x_i(2)$	0	0	1	1	0	1	0	0	1



In der Tabelle ist für jeden Speicherzustand und jedem möglichen zugeführtem **Informationsbit** (u_i) der Ausgang mit den **Codebits** ($x_i(1), x_i(2)$) dargestellt.

Zu Beginn werden die Speicher mit Nullen initialisiert. Dabei sind die Codebits am Ausgang Null.

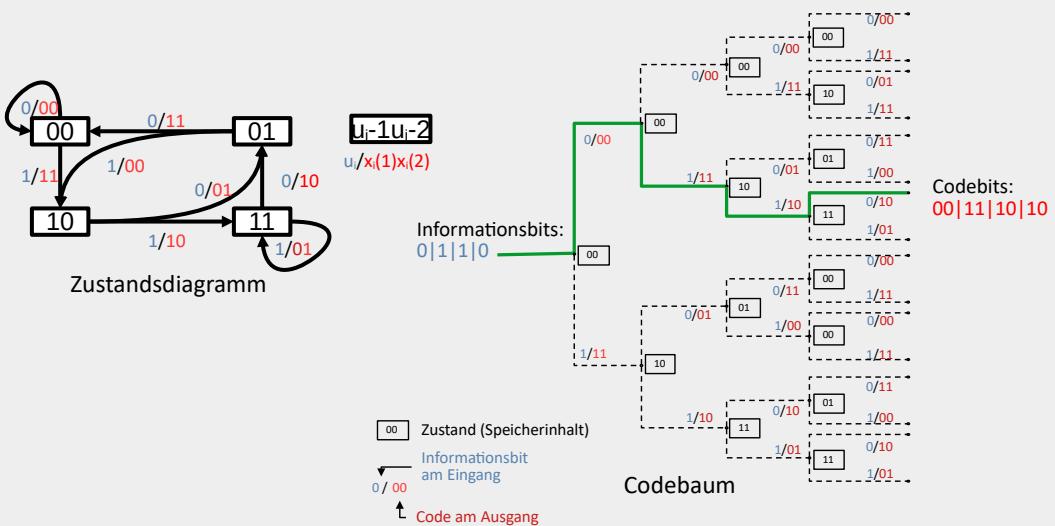
Damit lässt sich das Zustandsdiagramm für alle möglichen Werte der Speicher aufbauen.

Die Anzahl der Speicherzellen (b) bestimmt die Anzahl der Zustände. Es gibt also 2^b Zustände. Der Inhalt der Speicherzellen ist in den Zuständen abgebildet. Der Inhalt ist von rechts nach links zu lesen. Das bedeutet, dass der älteste Speicherinhalt ganz rechts steht, der jüngste Speicherinhalt ganz links.

In **Blau** sind die zugeführten **Informationsbits** (u_i) dargestellt, die zu dem Zustand geführt haben.

In **Rot** sind die beiden **Codebits** ($x_i(1), x_i(2)$) dafür dargestellt.

Vom Zustandsdiagramm zum Codebaum



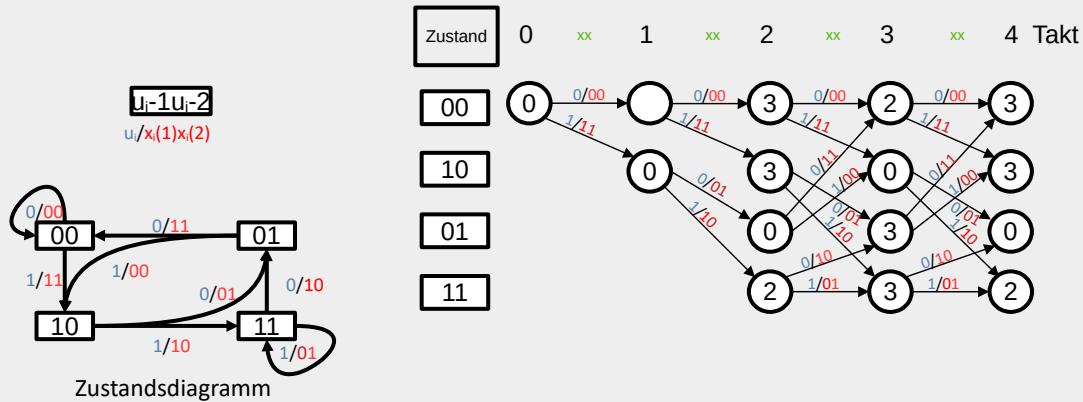
Eine weitere Möglichkeit die Funktionalität eines Faltungscodierers darzustellen ist ein Codebaum.

Beim Codebaum führt eine Null zu einer Verzweigung nach oben.
Eine 1 führt zu einer Verzweigung nach unten.
Dadurch wächst der Baum nach rechts exponentiell mit dem Takt!

Die Informationsbitfolge 0110 (in blau) am Eingang führt zur Codebitfolge 00111010 (in rot) am Ausgang. Der jeweilige Zustand ist mit einem Rahmen gekennzeichnet.

Wie in der Folie zu sehen ist, ergibt eine Informationsbitfolge einen Pfad der zu den Codebits führt.

Vom Zustandsdiagramm zum Trellis



Auf der Empfängerseite gibt es keine Schaltung, welche den empfangenen Datenstrom in die ursprüngliche Bitfolge zurückführen kann.

Stattdessen wird mit einem Viterbi-Algorithmus das wahrscheinlichste Ergebnis geschätzt. Dabei handelt es sich um das **Maximum-Likelihood-Verfahren**. Deshalb heißt der Decoder auch **ML-Decoder**.

Eine Kombination aus Codebaum und Zustandsdiagramm ist das Netzdiagramm (engl. Trellis).

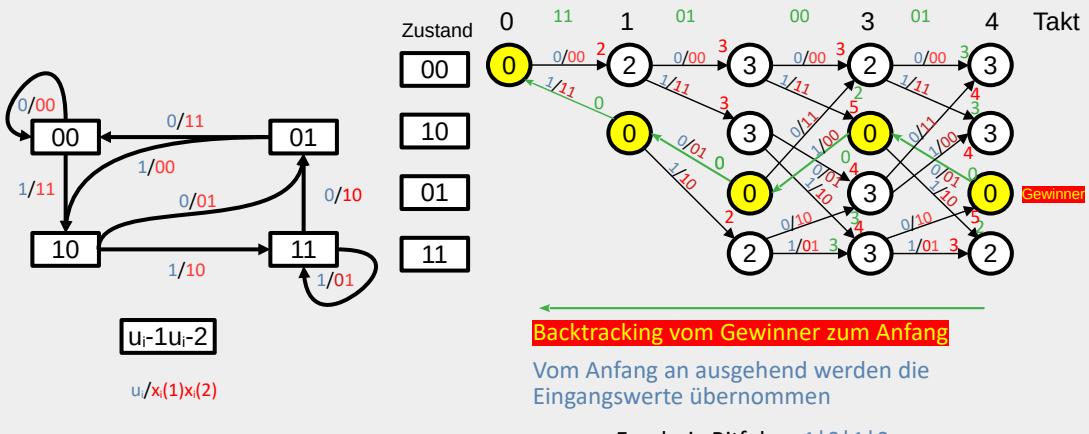
Darin werden für jeden Takt die Zustände, die Informationsbits, die Codebits und die Summe der Hammingabstände des jeweiligen Pfades dargestellt. Mit den Pfeilen können die Informationsbits (in blau) und die Codebits (in rot) dargestellt werden.

Zu Beginn wird mit Nullen initialisiert. Das bedeutet im Zustand 00 wird gestartet.

Während der ersten beiden Takte wächst das Trellisdiagramm noch. Im eingeschwungenen Zustand bleibt es bei den 4 Zuständen, da es nur zwei Speicherelemente gibt. Jeder Zustand hat zwei Eingänge und zwei Ausgänge

Bei der Bearbeitung werden für die empfangenen Bits der Hammingabstand zu den möglichen Codebits ermittelt und in den Kreisen für jeden Takt aufsummiert.

Trellisdiagramm - Bearbeitung

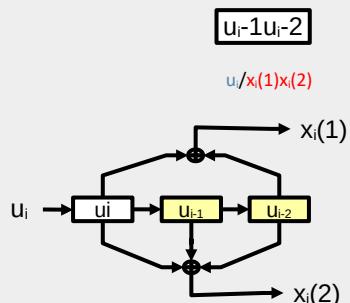


Am Ende wird der Pfad ausgewählt, der die kleinste Summe der Hamming-Abstände hat.

Ist der Weg vom Gewinner-Knoten zum Anfang ermittelt, werden die Eingangszustände (blau) für jeden Weg zwischen den Knoten, beginnend mit dem Anfang, hintereinander aufgeschrieben.

Die aufgeschriebenen Werte ergeben die empfangene Bitfolge.

Beispiel: Viterbi-Algorithmus
Schritt - 1



Beispiel-Bitfolge: 1|0|1|0

In Codierer

i	init	1	2	3	4
u_i	0	1	0	1	0
u_{i-1}	0	0	1	0	1
u_{i-2}	0	0	0	1	0
$x_i(1)$	0	1	0	0	0
$x_i(2)$	0	1	1	0	1

Aufgereiht

11|01|00|01

Senden

Beispiel:

Der Faltungscodierer wird mit 0,0,0 initiiert.

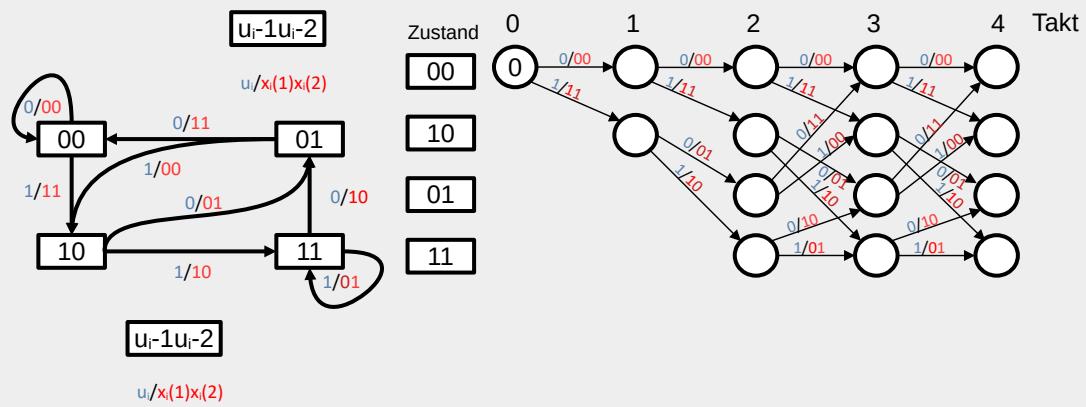
Danach wird die Bitfolge 1010 in den Faltungscodierer eingespeist und durchgeschoben.

Am Ausgang können die Signale $x_i(1)$ und $x_i(2)$ abwechselnd abgegriffen werden.

Um das Beispiel einfach zu halten wurde auf eine Punktierung verzichtet.

Das Ergebnis, die Bitfolge 11-01-00-01, wird danach gesendet.

Beispiel: Viterbi-Algorithmus Schritt - 2

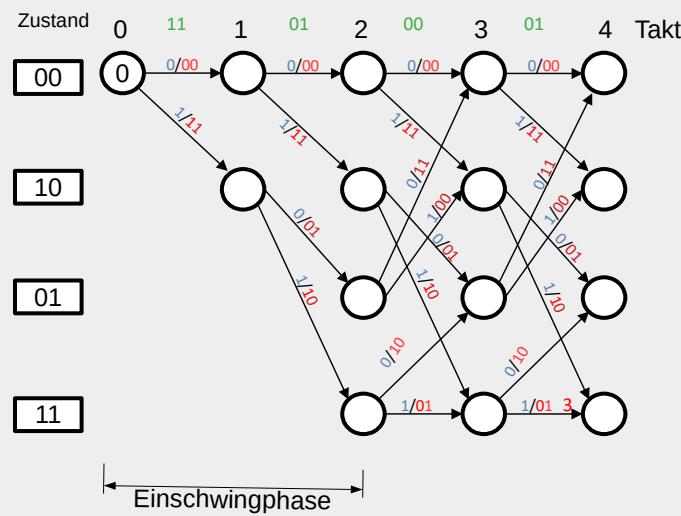


Vorbereitung:

Auf der Empfängerseite ist ein Trellis aus dem Zustandsdiagramm zu erstellen.

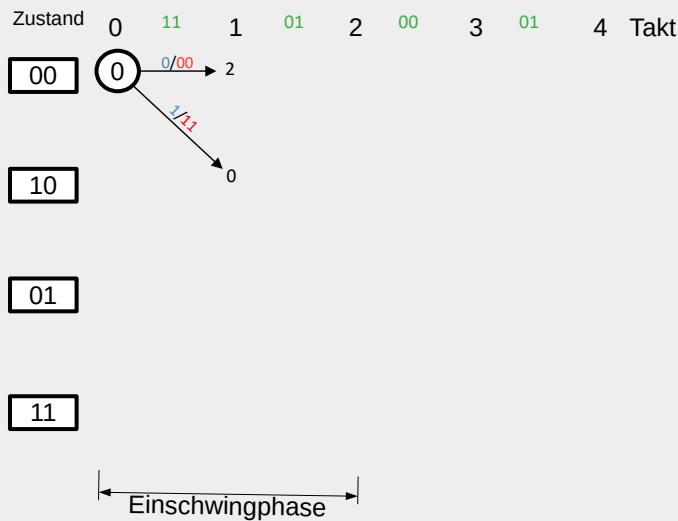
Zu Beginn wird mit Nullen initialisiert. Das bedeutet im Zustand 00 wird gestartet.

Beispiel: Viterbi-Algorithmus Schritt - 3



Einfügen der empfangenen Bits pro Takt

Beispiel: Viterbi-Algorithmus Schritt - 4



Ablauf:

Für den ersten Takt wird die Hamming-Distanz für die möglichen **Eingaben (0 oder 1)** gebildet.

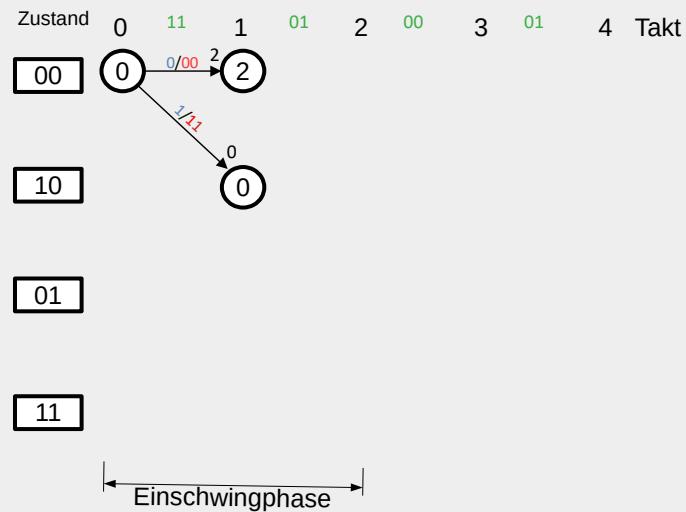
Dabei wird für den ersten Takt der Hamming-Abstand (HA) zwischen den empfangenen Bits (**11**) mit den beiden Pfaden (**00** und **11**) des ersten Zustands ermittelt.

Status 00: HA (**11** **00**) = 2
 HA (**11** **11**) = 0

Der obere Pfad hat einen Hammingabstand von 2 (**11** / **00**).
Der untere Pfad hat einen Hammingabstand von 0 (**11** / **11**).

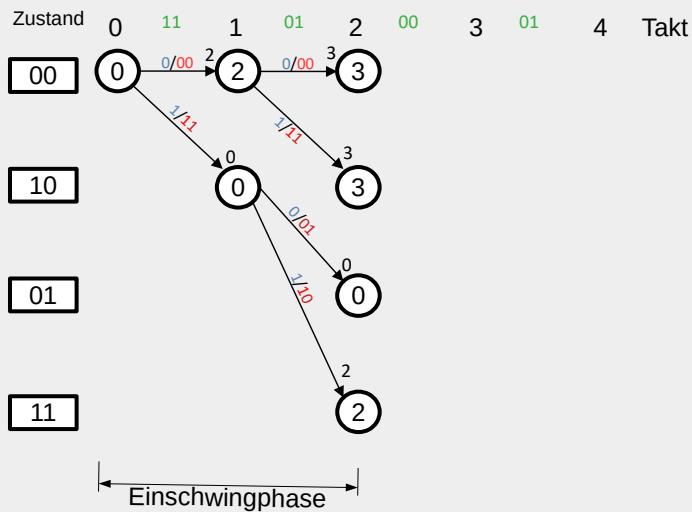
Der ermittelte Hamming-Abstand wird an das Pfad- / Pfeilende geschrieben.

Beispiel: Viterbi-Algorithmus Schritt - 5



Der ermittelte Hammingabstand wird in den Kreis als Ergebnis des ersten Taktes eingetragen.

Beispiel: Viterbi-Algorithmus Schritt - 6



Ausgehend von den beiden Stati nach dem ersten Takt werden die Stati für den zweiten Takt ermittelt.

Für die empfangene Bitfolge **01** wird mit den Ausgangswerten aller möglicher Pfade der Hamming-Abstand (HA) gebildet und auf den bisher aufgelaufenen Hamming-Abstand des Pfades addiert.

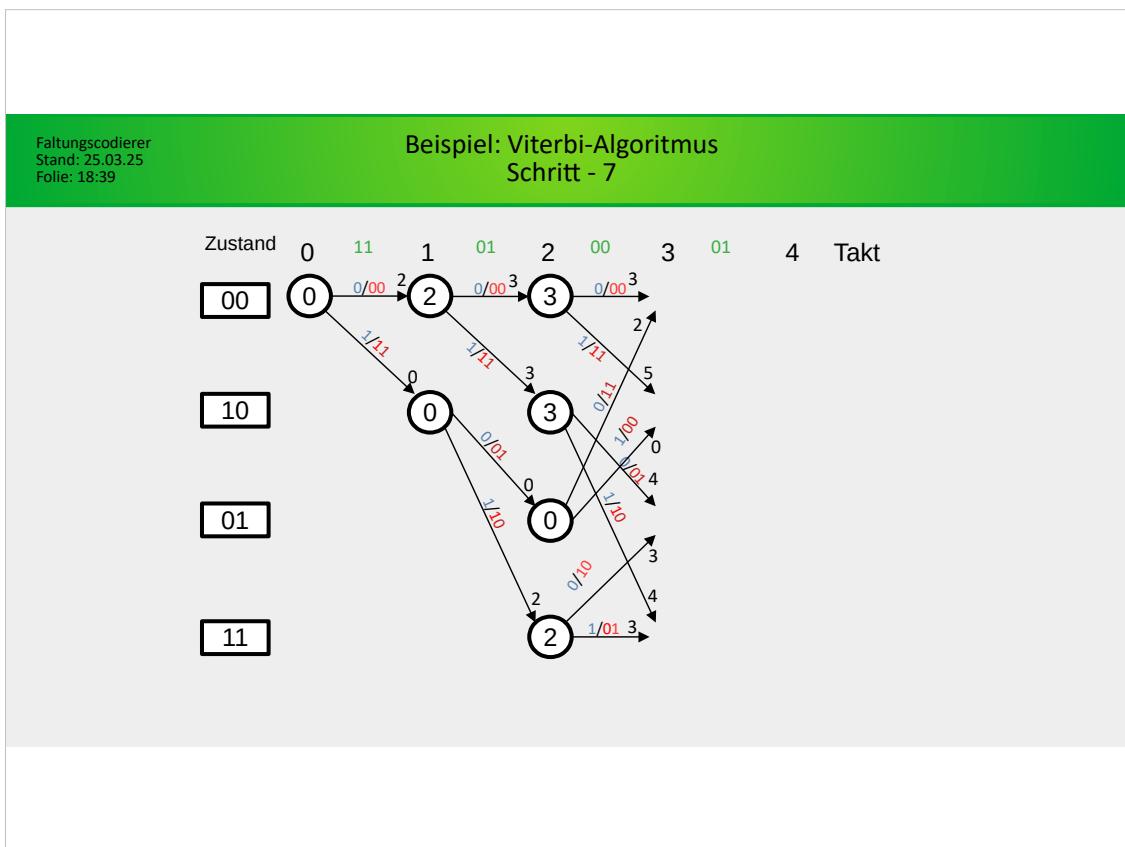
Das Ergebnis wird an das Pfeil- / Pfadende geschrieben

Status 00: HA (**01 00**) = 1 → Pfadsumme = 2 + 1 = 3
 HA (**01 11**) = 1 → Pfadsumme = 2 + 1 = 3

Status 10: HA (**01 01**) = 0 → Pfadsumme = 0 + 0 = 0
 HA (**01 10**) = 2 → Pfadsumme = 0 + 2 = 2

Die ermittelten Pfadsummen werden in das Ergebnis für den zweiten Takt in den Kreis geschrieben.

Da jetzt alle Stati erreichbar sind, ist die Einschwingphase abgeschlossen.



Ausgehend von den Stati nach dem zweiten Takt werden die Stati für den dritten Takt ermittelt.

Für die empfangene Bitfolge **00** wird mit den Ausgangswerten aller möglicher Pfade der Hamming-Abstand (HA) gebildet und auf den bisher aufgelaufenen Hamming-Abstand des Pfades addiert.

Das Ergebnis wird an das Pfeil- / Pfadende geschrieben

Status 00: HA (00 00) = 0 → Pfadsumme = 3 + 0 = 3
HA (00 11) = 2 → Pfadsumme = 3 + 2 = 5

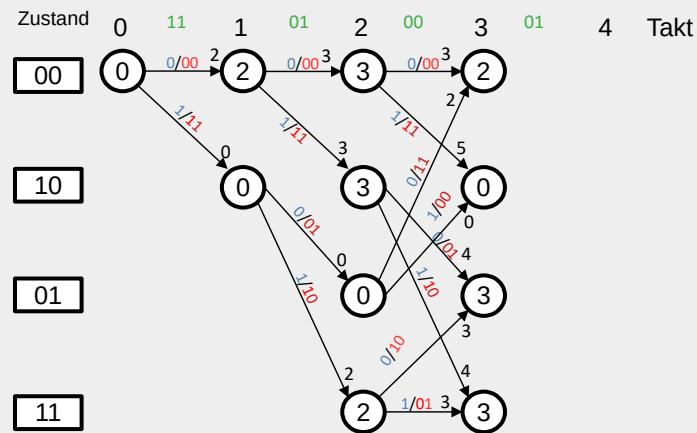
Status 10: $\text{HA (00 01)} = 1 \rightarrow \text{Pfadsumme} = 3 + 1 = 4$
 $\text{HA (00 10)} = 1 \rightarrow \text{Pfadsumme} = 3 + 1 = 4$

Status 01: HA (00 00) = 0 → Pfadsumme = 3 + 0 = 3
HA (00 11) = 2 → Pfadsumme = 3 + 2 = 5

Status 11: HA (00 01) = 1 → Pfadsumme = 3 + 1 = 4
HA (00 10) = 1 → Pfadsumme = 3 + 1 = 4

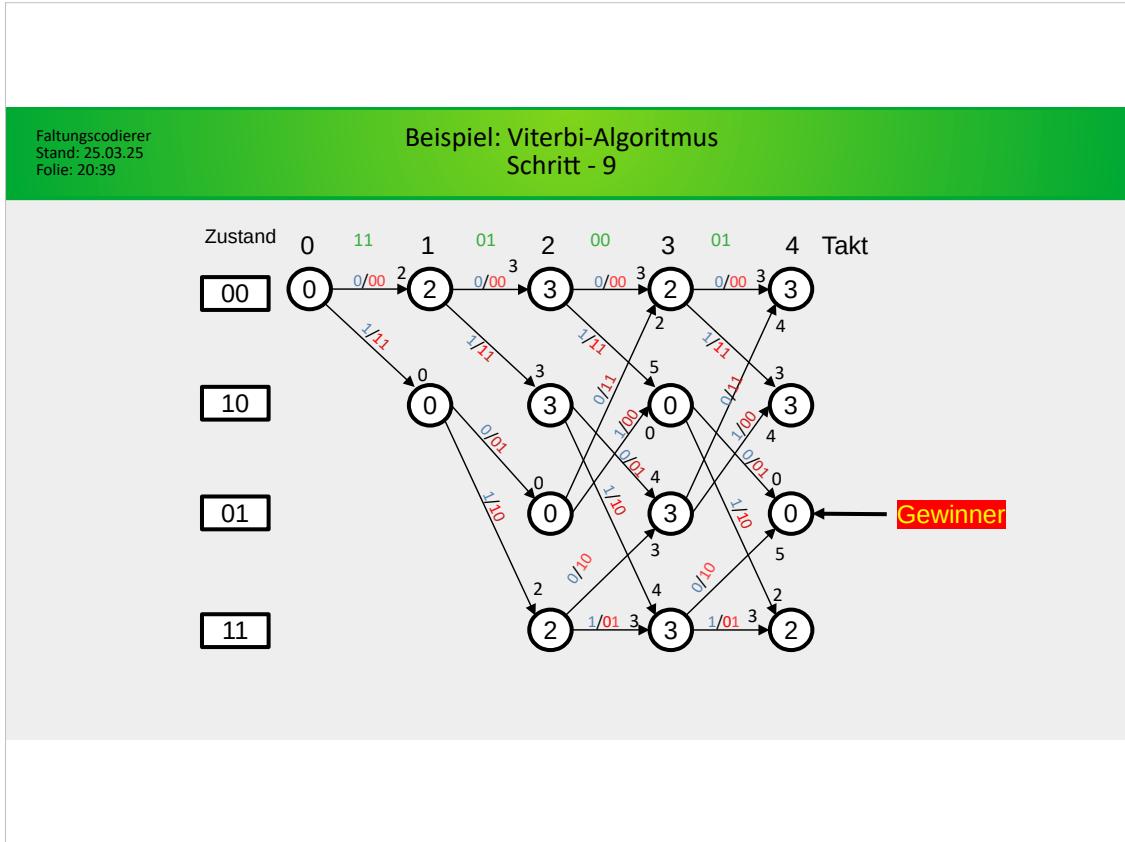
Die Pfadsummen werden in das Ergebnis für den zweiten Takt in den Kreis geschrieben.

Beispiel: Viterbi-Algorithmus Schritt - 8



Nun gibt es für jedes Taktergebnis zwei mögliche Werte, die übernommen werden können.

Sind die Werte unterschiedlich, wird immer der kleinste von beiden Werten übernommen und in den Kreis geschrieben.

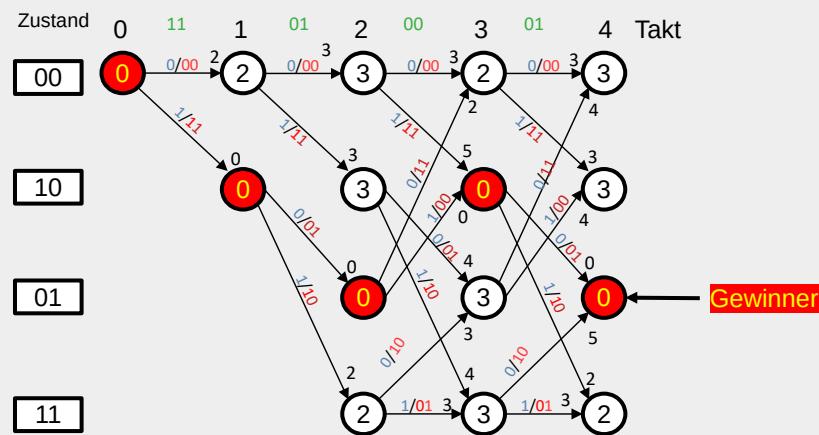


Die Vorgehensweise wird für alle weiteren Takte fortgesetzt, bis man am Ende der empfangenen Bitfolge ankommt.

Am Ende angekommen, stellt der Pfad mit der kleinsten Summe aller Hamming-Abstände die wahrscheinlichste gesendete Bitfolge dar.

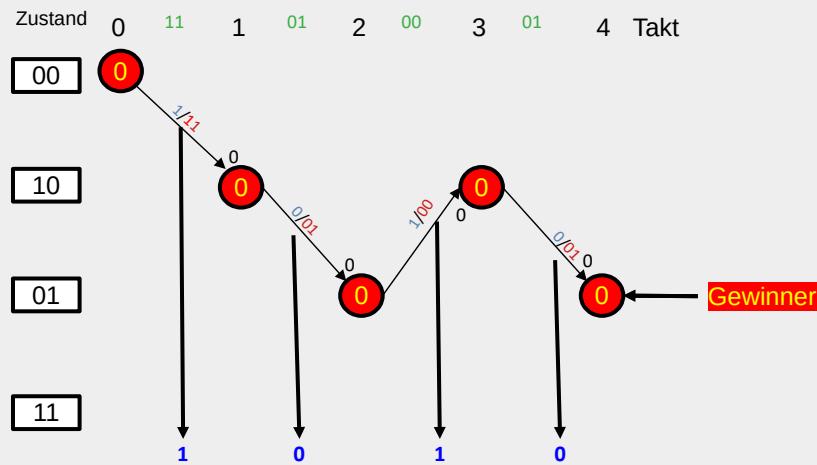
Damit ergibt der Wert Null beim Status 01 den Gewinner der Bearbeitung.
(Ende des wahrscheinlichsten Pfades)

Beispiel: Viterbi-Algorithmus Schritt - 10



Ausgehend vom Gewinner wird der Pfad mit den kleinsten Werten zurück verfolgt bis zum Takt 0.

Beispiel: Viterbi-Algorithmus Schritt - 11

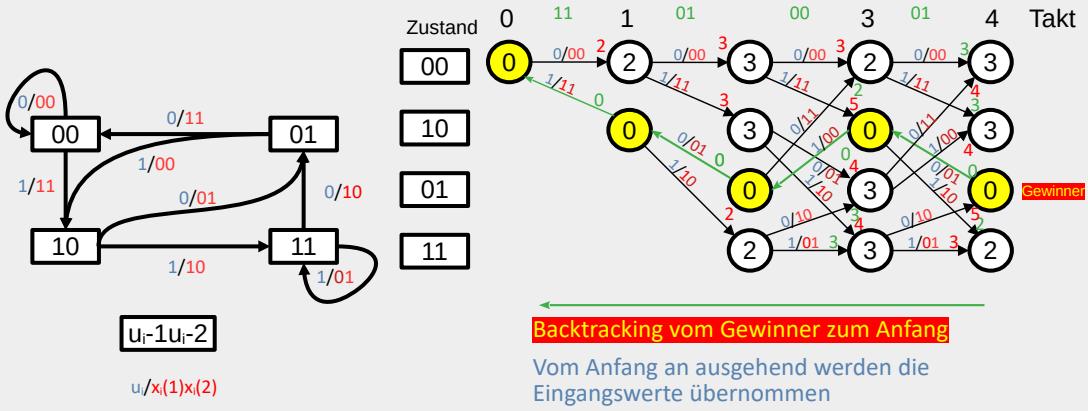


Die restlichen Pfade wurden für eine bessere Übersicht entfernt.

Nun können die **blauen Eingangswerte**, die für die Berechnung der Stati gedient haben und bisher nur mitgeschleppt wurden, als die ursprünglich zu übertragende Bitfolge festgelegt werden.

1 0 1 0

Beispiel: Viterbi-Algorithmus Zusammenfassung



Ist der Weg vom Gewinner-Knoten zum Anfang ermittelt, werden die Eingangszustände (blau) für jeden Weg zwischen den Knoten, beginnend mit dem Anfang, hintereinander aufgeschrieben.

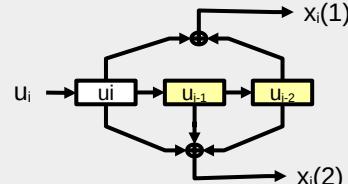
Die aufgeschriebenen Werte ergeben die empfangene Bitfolge.

Beispiel: Faltungscodierer mit Punktierung Schritt - 1

Beispiel-Bitfolge: 1|0|1|1|0|1

In Codierer

i	1	2	3	4	5	6
u_i	1	0	1	1	0	1
u_{i-1}	0	1	0	1	1	0
u_{i-2}	0	0	1	0	1	1
$x_i(1)$	1	0	0	1	1	0
$x_i(2)$	1	1	0	0	0	0



Punktierung

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

Aufgereiht

110001100 Senden

Das folgende Beispiel beinhaltet eine Punktierung als Verschärfung:

Es soll die Bitfolge 1-0-1-1-0-1 übertragen werden. Sie wird in den folgenden Folien zur Überprüfung immer unten rechts eingebettet.

Als Faltungscodierer kommt die bekannte Schaltung zum Zug.

Das Ergebnis des Faltungscodierers

$$x_i(1)=1-0-0-1-1-0$$

$$X_i(2)=1-1-0-0-0-0$$

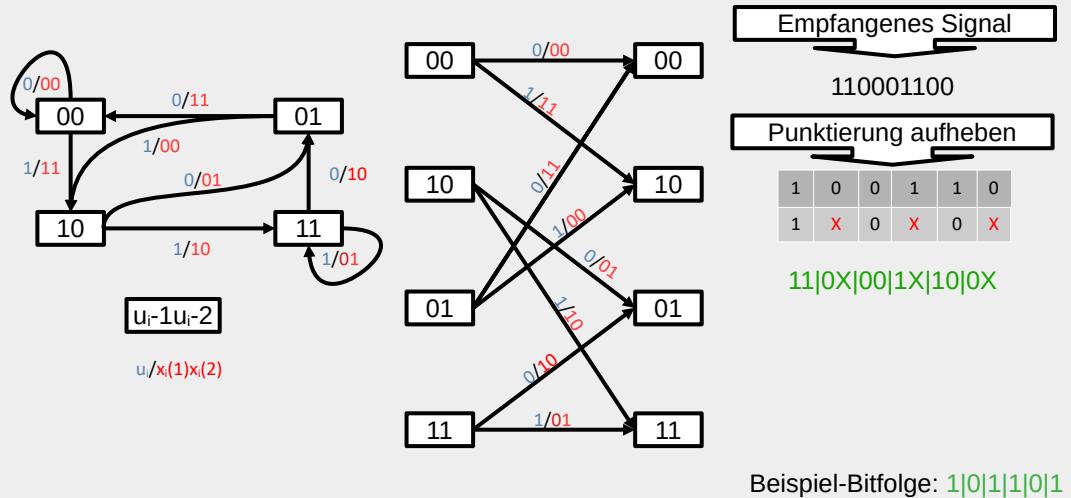
wird einer 2 / 3 - Punktierung zugeführt.

Das Ergebnis der Punktierung lautet:

$$1-1-0-0-0-1-1-0-0$$

Diese Bitfolge wird gesendet.

Beispiel: Faltungscodierer mit Punktierung Schritt - 2



Um die sendeseitige Punktierung rückgängig zu machen ist auf der Empfängerseite zuerst die Bitfolge mit Dummy-Werten (**X**) aufzufüllen.

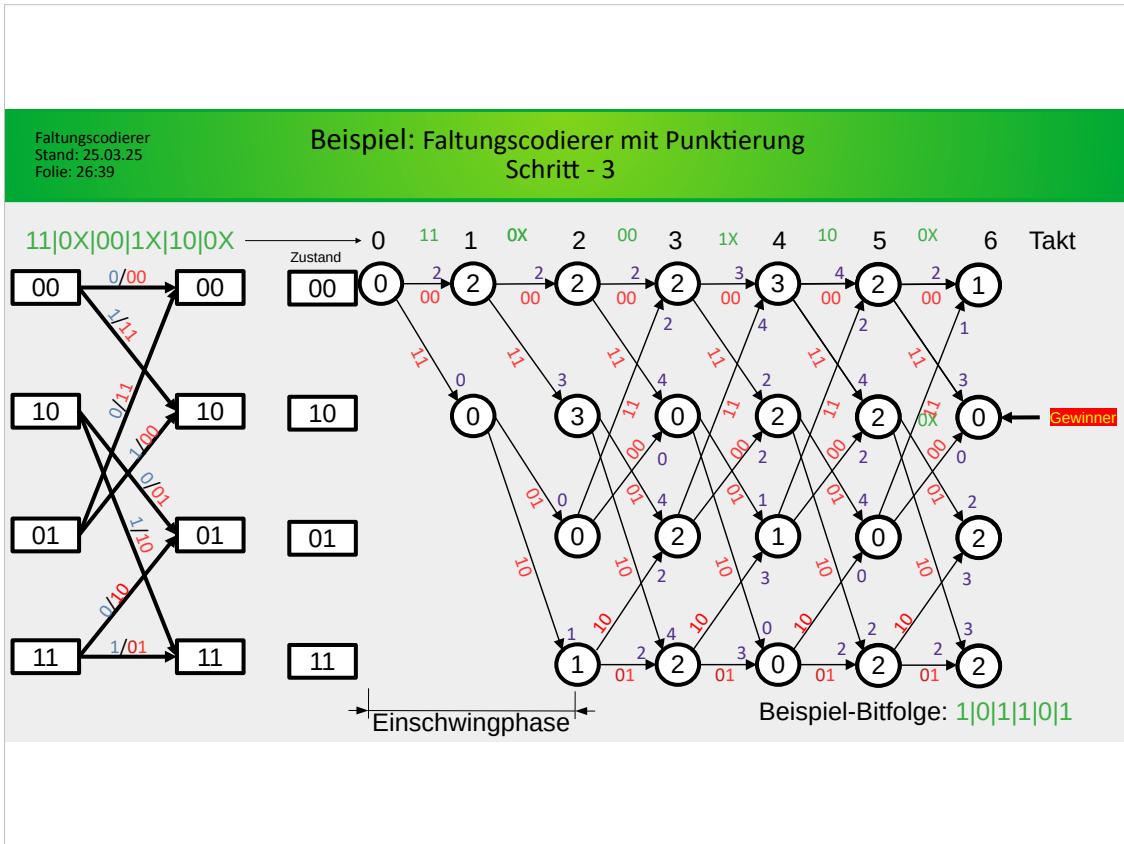
Dadurch entsteht die Bitfolge:
11|0X|00|1X|10|0X

Da die Vorgehensweise auf Folie 7 etwas unübersichtlich war, wird hier eine Alternative eingeführt.

Dazu wird das Statusdiagramm mit seinen Übergängen wie in der Folienmitte auseinandergezogen.

Jeder Zustand ist auf der linken und auf der Rechten Seite einmal aufgeführt. Von jedem Status gehen zwei Verbindungen ab und es werden auch zwei Verbindungen jedem Zustand zugeführt.

Die neue Darstellung des Zustandsdiagramms wird in der Folgefolie als Grundlage auf der linken Seite platziert.



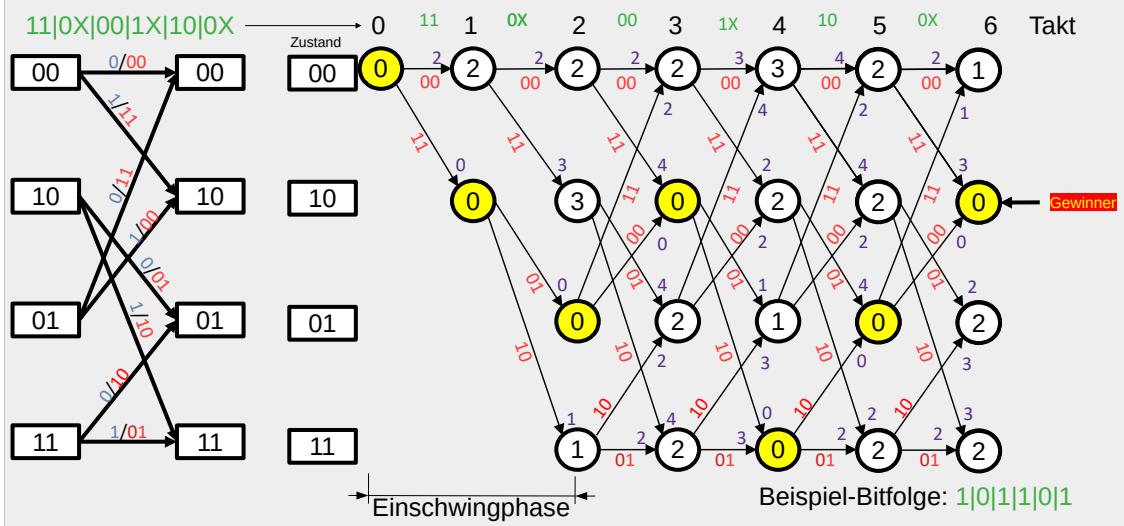
Das Trellis ist nun etwas übersichtlicher. Allerdings muss beim Aufbau der Pfade immer die Übergänge von der linke Seite mit übernommen werden.

Die Bitfolge 11|0X|00|1X|10|0X wird mit dem Trellis bearbeitet.

Dabei werden die Stellen mit dem Dummy-X nicht mitgerechnet!

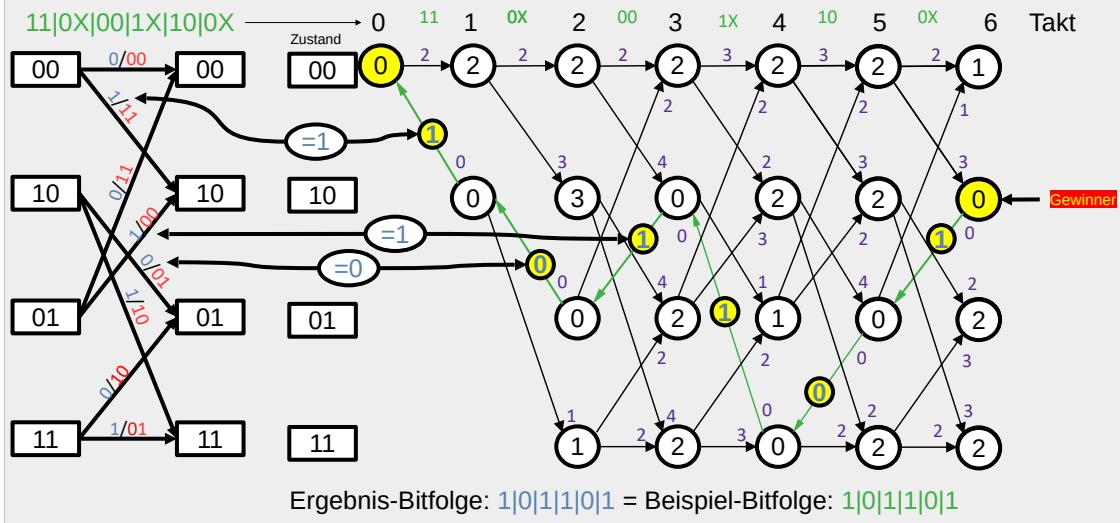
Es gibt einen Gewinner!

Beispiel: Faltungscodierer mit Punktierung Schritt - 4



Ausgehend vom Gewinner wird der Pfad rückwärts aufgebaut . Dabei wird immer der Pfeil mit dem kleinsten Wert zurückverfolgt.

Beispiel: Faltungscodierer mit Punktierung Schritt - 5

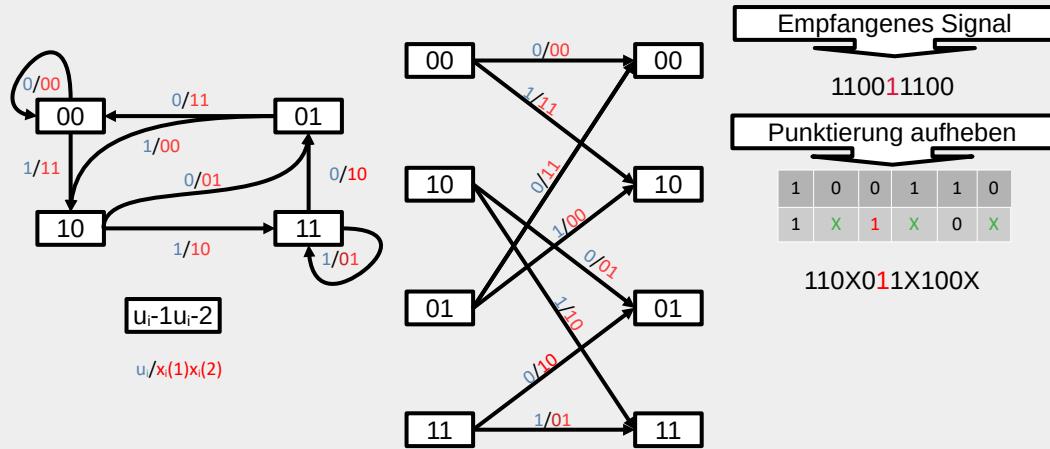


Für jeden Teil-Pfad wird jetzt der zugehörige Eingangswert (blauer Wert aus dem Zustandsdiagramm) ermittelt und vom Gewinner-Knoten aus rückwärts aufgereiht.

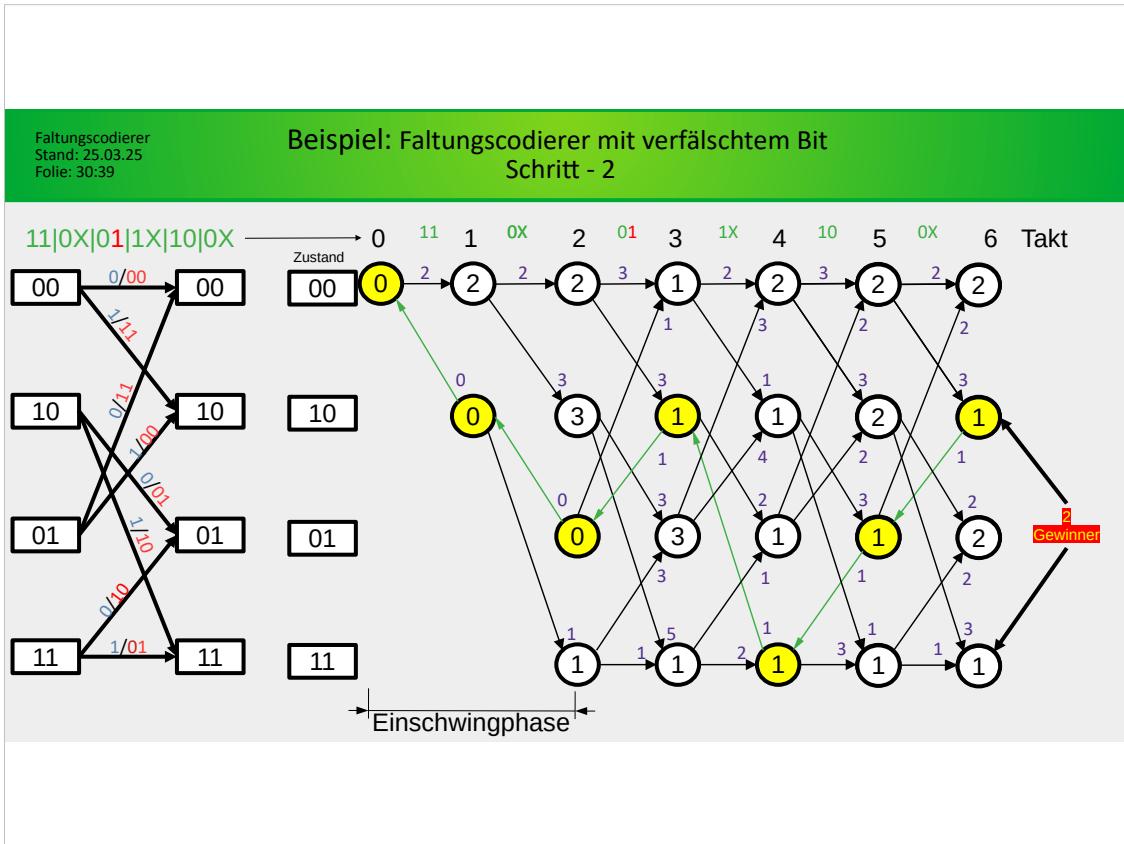
Die so entstehende Bitfolge lautet: 1|0|1|1|0|1

Das ist genau die gesendete Bitfolge (siehe unten)

Beispiel: Faltungscodierer mit verfälschtem Bit Schritt - 1



Als nächste Verschärfung wird bei der Übertragung ein Bitfehler eingebaut.



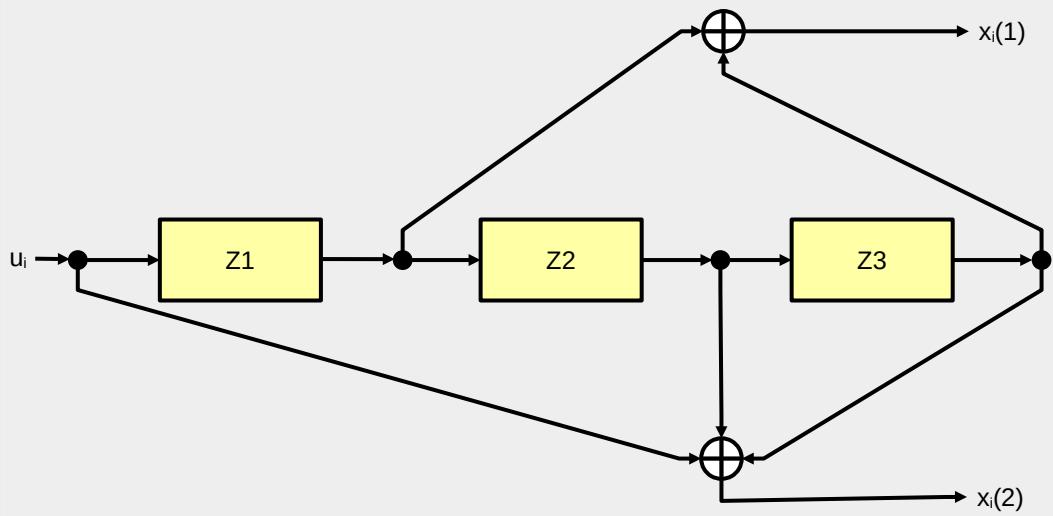
Die Bearbeitung im Trellis führt zu zwei Ergebnissen mit dem Wert = 1.

Jetzt erhebt sich die Frage, welcher Pfad der richtige ist.

Um die Frage beantworten zu können, müssten die zu dekodierenden Daten länger sein.

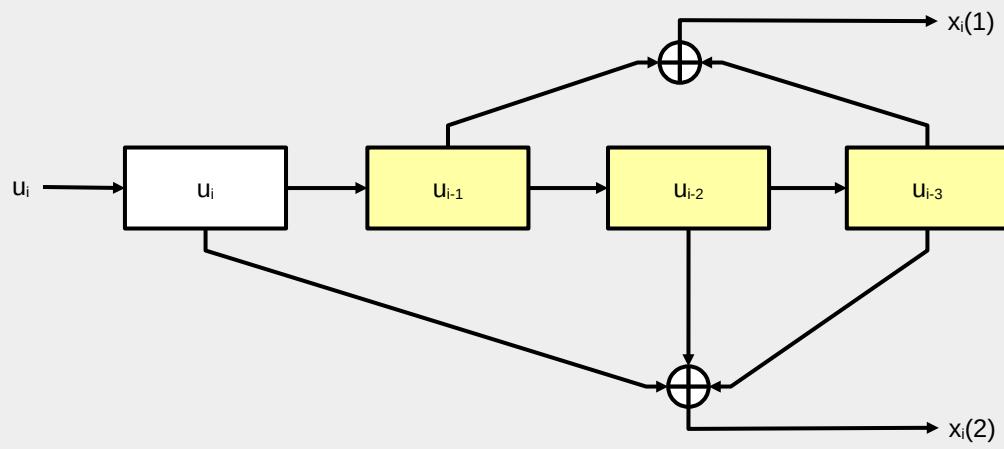
Deshalb werden an die Datenbits immer noch definierte Tail-Bits (Nullen) angehängt, die dazu führen, dass das Netzdigramm (trellis) ein eindeutiges Ergebnis liefert.

Beispiel: Faltungscodierer mit 3 Speichern
Schritt - 1



Zum Abschluss folgt noch ein Beispiel für einen Faltungscodierer mit 3 Speicherzellen und ein Viterbi-Algorithmus.

Beispiel: Faltungscodierer mit 3 Speichern
Schritt - 2



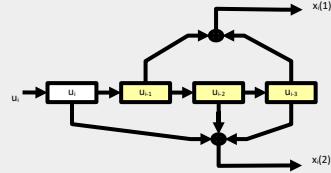
Das Blockdiagramm hat das obige Aussehen.

Die Generatorpolynom lauten:

$$g_0 = x + x^3$$

$$g_1 = 1 + x^2 + x^3$$

Beispiel: Faltungscodierer mit 3 Sperichern Schritt - 3



u_i		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
u_{i-1}	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
u_{i-2}	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1
u_{i-3}	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
$x_i(1)$	0	0	0	1	1	0	0	1	1	1	1	0	0	1	1	0	0	0	0
$x_i(2)$	0	0	1	0	1	1	0	1	0	1	0	1	0	0	1	0	1	0	1

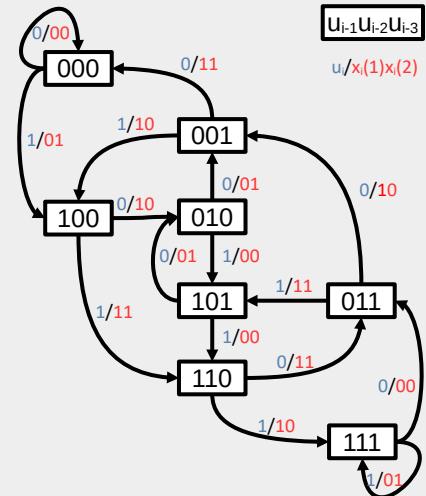
Aus dem Blockschaltbild kann die Tabelle mit den Statuszuständen, den Eingangswerten und den Ausgangswerten aufgebaut werden.

Beispiel: Faltungscodierer mit 3 Speichern Schritt - 4

u_i	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
u_{i-1}	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
u_{i-2}	0	0	0	0	0	1	1	1	0	0	0	0	1	1	1	1
u_{i-3}	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$x_i(1)$	0	0	0	1	1	0	0	1	1	1	0	0	1	1	0	0
$x_i(2)$	0	0	1	0	1	1	0	1	0	1	0	0	1	0	1	0

Beispiel-Bitfolge: 0-0-1-1-0-1-0-1-1

i	1	2	3	4	5	6	7	8	9
u_i	0	0	1	1	0	1	0	1	1
u_{i-1}	0	0	0	1	1	0	1	0	1
u_{i-2}	0	0	0	0	1	1	0	1	0
u_{i-3}	0	0	0	0	0	1	1	0	1
$x_i(1)$	0	0	0	1	1	1	0	0	0
$x_i(2)$	0	0	1	1	1	1	0	0	0



Aus der Tabelle kann das aufwändigere Statusdiagramm auf der rechten Seite erstellt werden.

Jeder Status wird mit den 3 Bits der Speicher beschrieben.
Es gibt nach wie vor am Eingang ein Informationsbit und am Ausgang zwei Codebits.

Es soll die folgende Bitfolge übertragen werden: 0-0-1-1-0-1-0-1-1

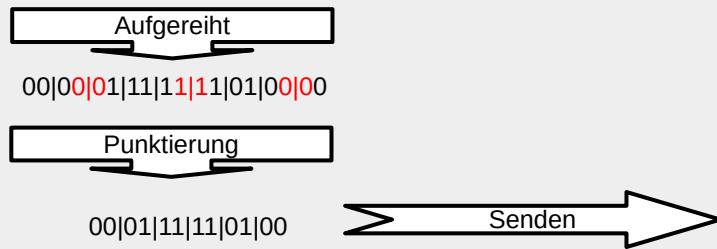
Das Ergebnis nachdem Faltungscodierer lautet:

$$x_i(1)=0-0-0-1-1-0-0-0$$

$$x_i(2)=0-0-1-1-1-1-0-0$$

Beispiel: Faltungscodierer mit 3 Sperichern Schritt - 5

i	1	2	3	4	5	6	7	8	9
u_i	0	0	1	1	0	1	0	1	1
$x(1)$	0	0	0	1	1	1	0	0	0
$x(2)$	0	0	1	1	1	1	1	0	0



Das Ergebnis des Faltungscodierers

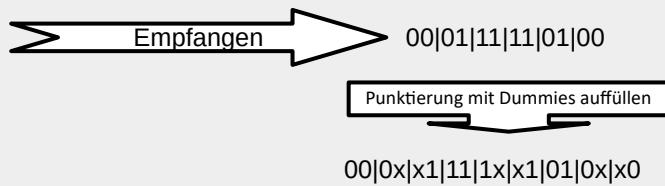
$$x_i(1)=0-0-0-1-1-1-0-0-0$$

$$x_i(2)=0-0-1-1-1-1-1-0-0$$

wird mit einer Coderate von 3 / 4 punktiert.

Das Ergebnis der Punktierung lautet 00-01-11-11-01-00 und wird gesendet

Beispiel: Faltungscodierer mit 3 Sperichern Schritt - 6

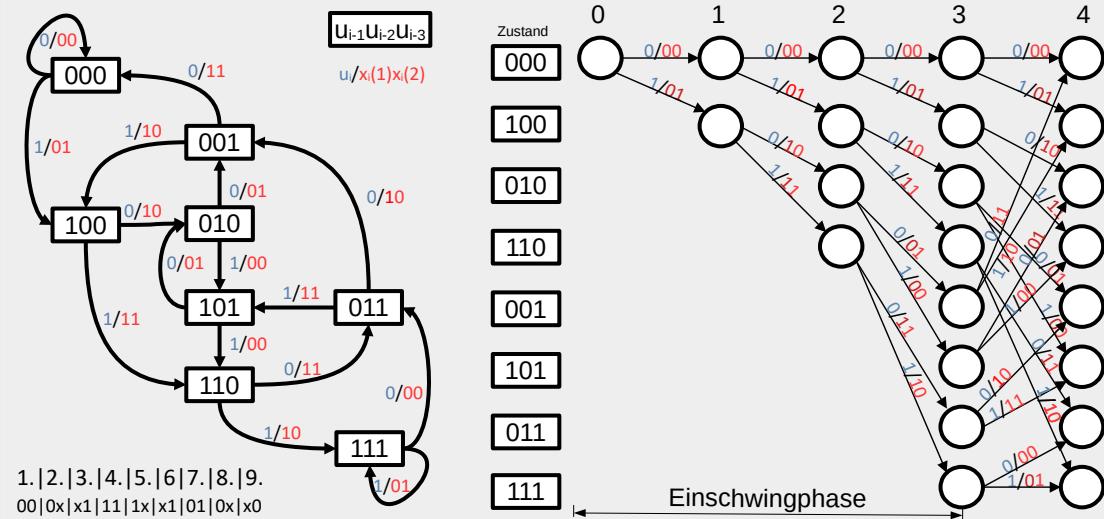


Die Bitfolge 00-01-11-11-01-00 wird empfangen.
Die durch die Punktierung verlorenen Bits müssen zuerst ergänzt werden.

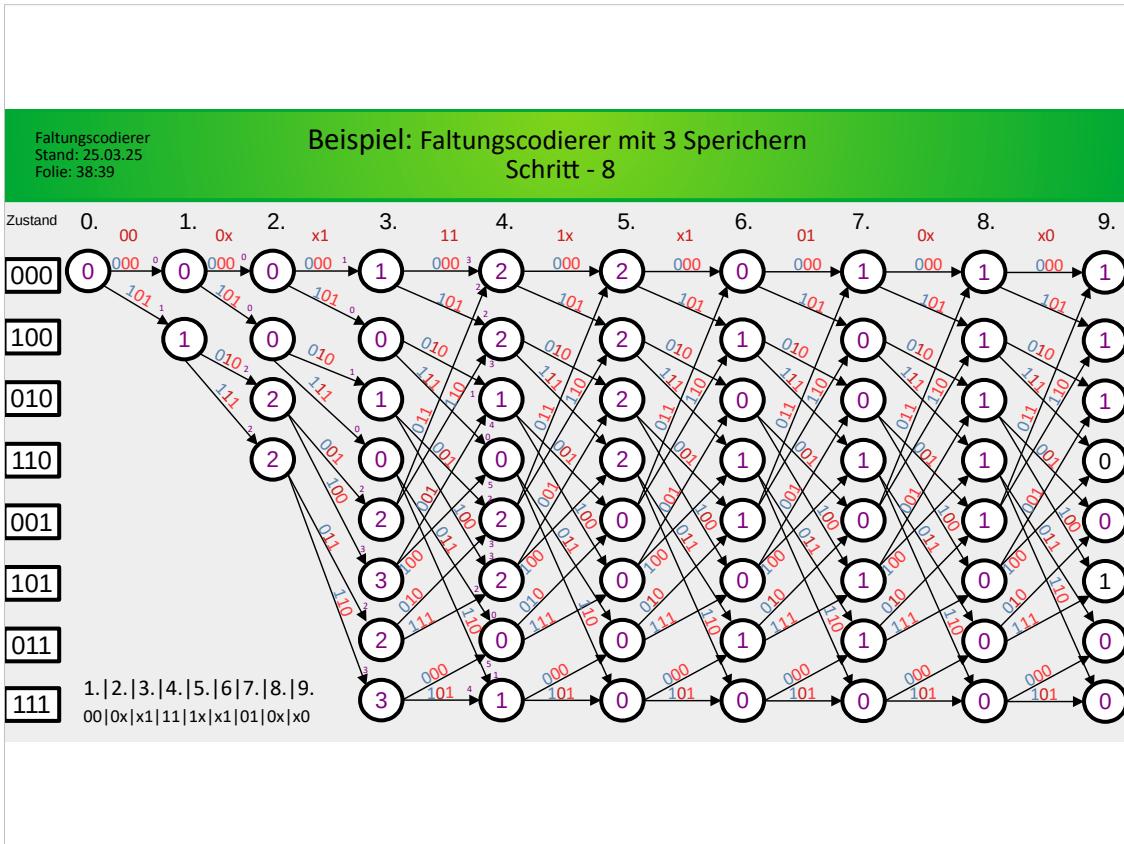
Nach dem Auffüllen mit Dummy-Bits lautet die Bitfolge:
00-0x-x1-11-1x-x1-01-0x-x0

Diese Bitfolge wird dem Viterbi-Algorithmus zugeführt.

Beispiel: Faltungscodierer mit 3 Speichern Schritt - 7



Der im Vorfeld zu erstellende Trellis ist etwas umfänglicher da 8 Zustände und die zugehörigen Übergänge abzubilden sind.



Als Ergebnis gibt es mehrere Pfade die einen Gesamt-Hammingabstand von 0 aufweisen.

Beide Ergebnisse sind nicht eindeutig.

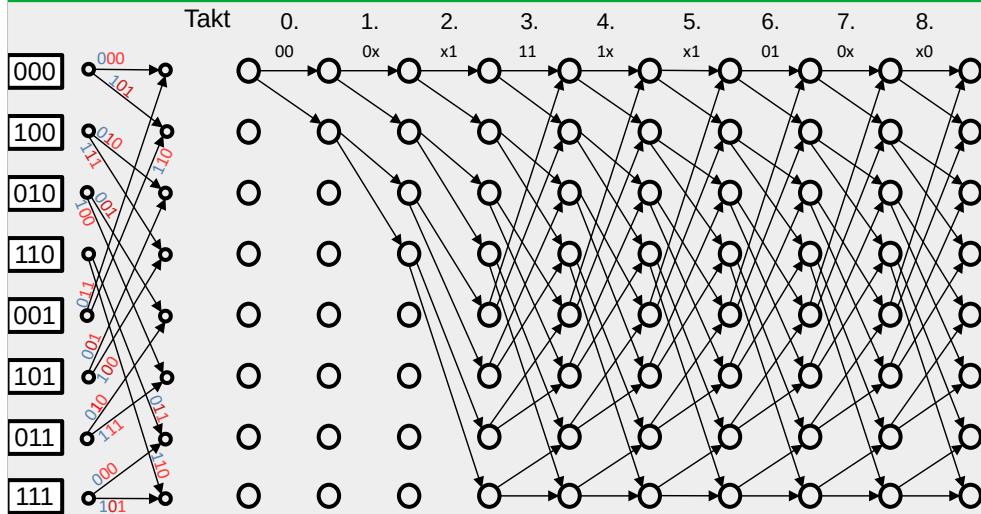
Das deutet darauf hin, dass die Bitfolge länger sein müsste um ein vernünftiges Ergebnis zu bekommen.

Deshalb werden immer Tailbits (bei WLANS 6 Bits) an die Informationsbits angehängt um ein eindeutiges Ergebnis zu erhalten.

Außerdem waren die Beispiele alles andere als übersichtlich.

Die folgende und letzte Folie zeigt ein etwas aufgeräumteres Aussehen, jedoch ist das Ergebnis das gleiche.

Beispiel: Faltungscodierer mit 3 Sperichern
Schritt - 9



Hier sind die Übergänge und der Eingang auf der linken Seite übersichtlich dargestellt.

Das Ergebnis ist allerdings genauso unbefriedigend, da sich die Bitfolge nicht geändert hat und immer noch genauso kurz ist.

Viel Spaß beim Ausprobieren!