Martin Nocker

universität innsbruck

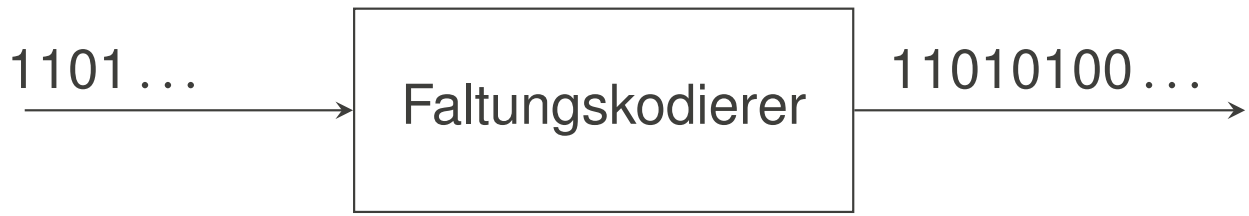
R-Paket für Kanalkodierung mit Faltungskodes

Gliederung

1. Blockkodes
2. R
3. **Faltungskodes**
4. **C++, Rcpp**
5. Turbo-Kodes
6. RMarkdown

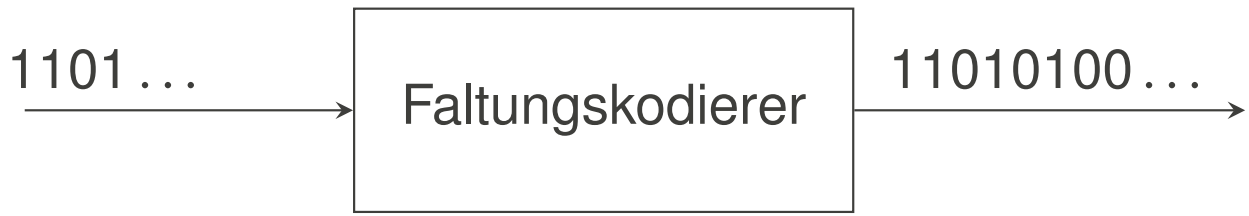
Faltungskodes

- Informationsstrom - keine Blöcke fester Länge
- ein einziges Kodewort als Resultat
- Redundanz wird kontinuierlich eingefügt



Faltungskodes

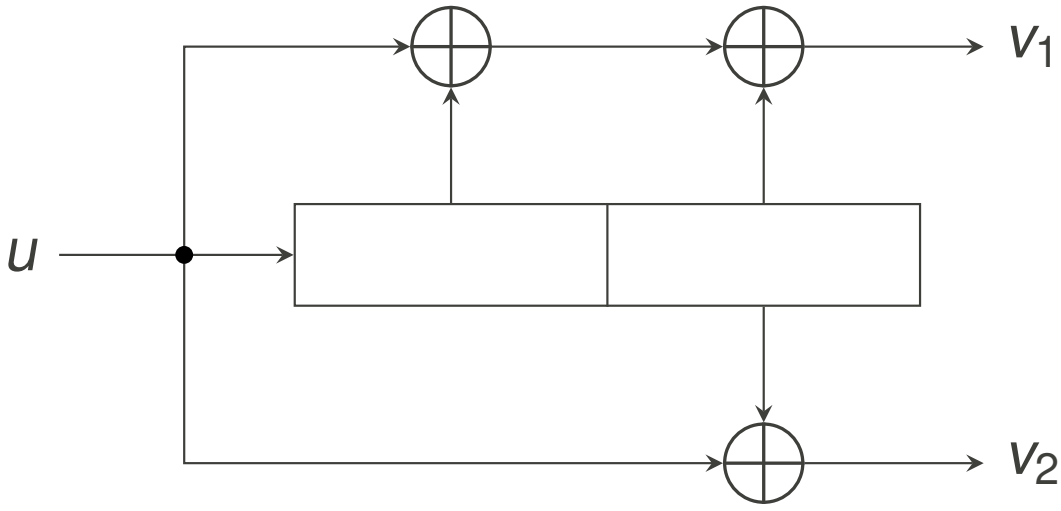
- Informationsstrom - keine Blöcke fester Länge
- ein einziges Kodewort als Resultat
- Redundanz wird kontinuierlich eingefügt



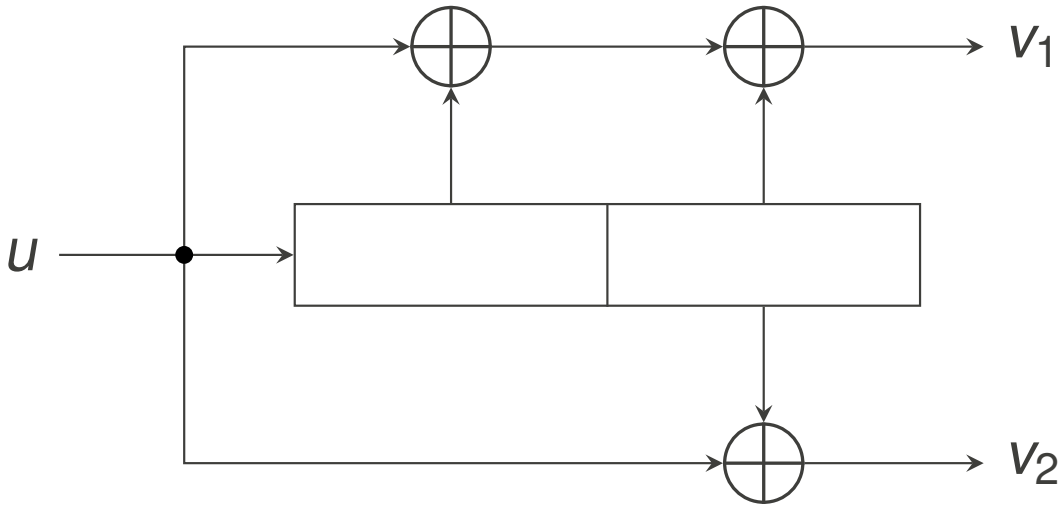
Verwendung

- Mobil- und Satellitenkommunikation
- Turbokodes

Faltungskodierer

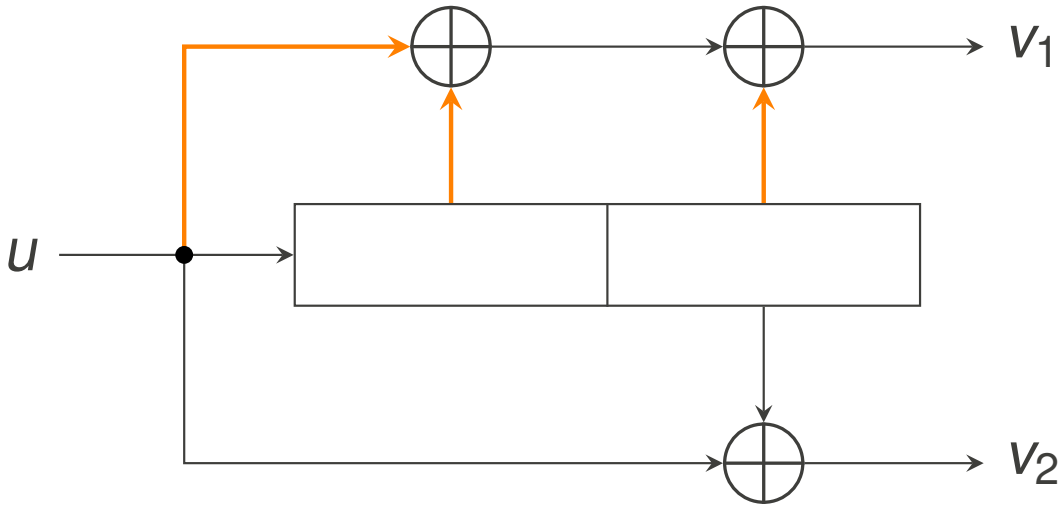


Faltungskodierer



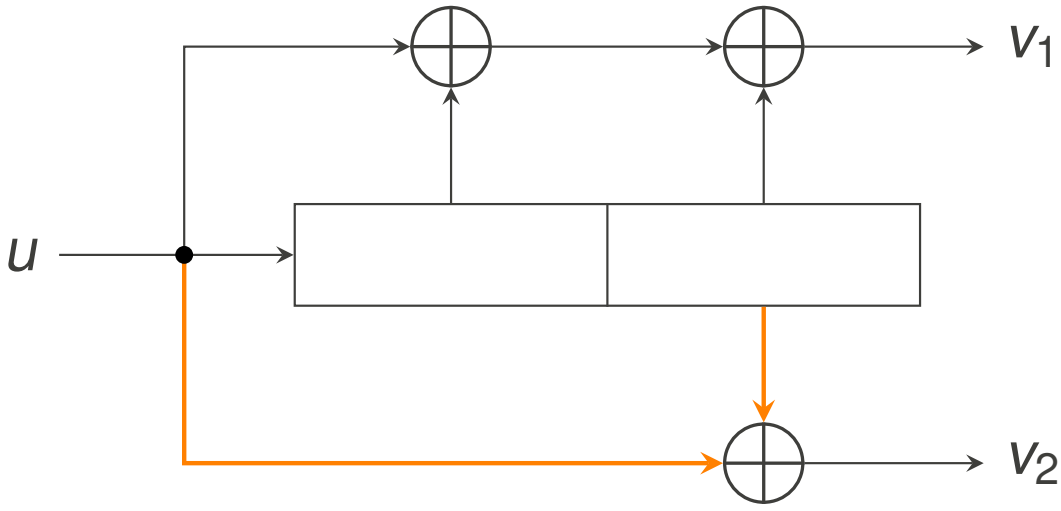
$$G = \begin{pmatrix} & \end{pmatrix}$$

Faltungskodierer



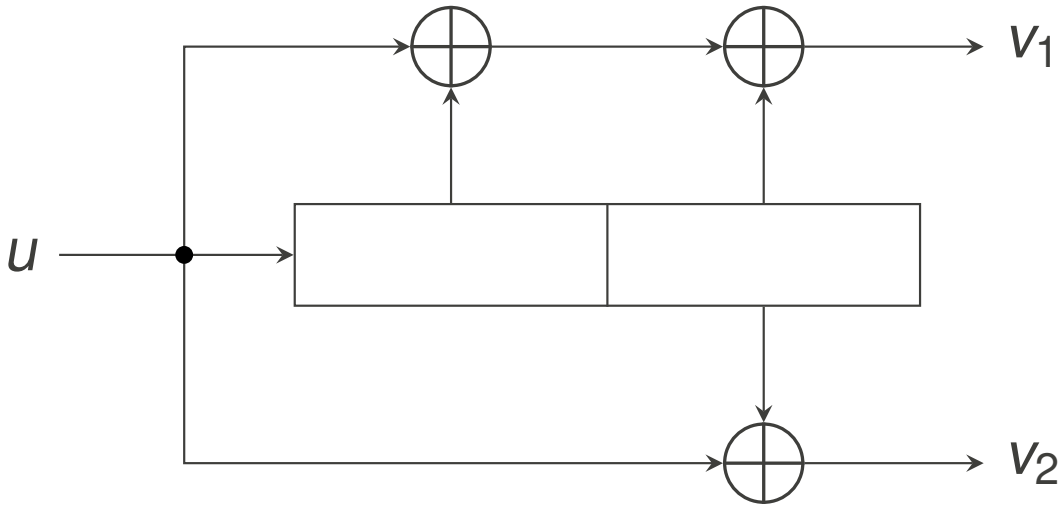
$$G = \begin{pmatrix} 111 \end{pmatrix}$$

Faltungskodierer



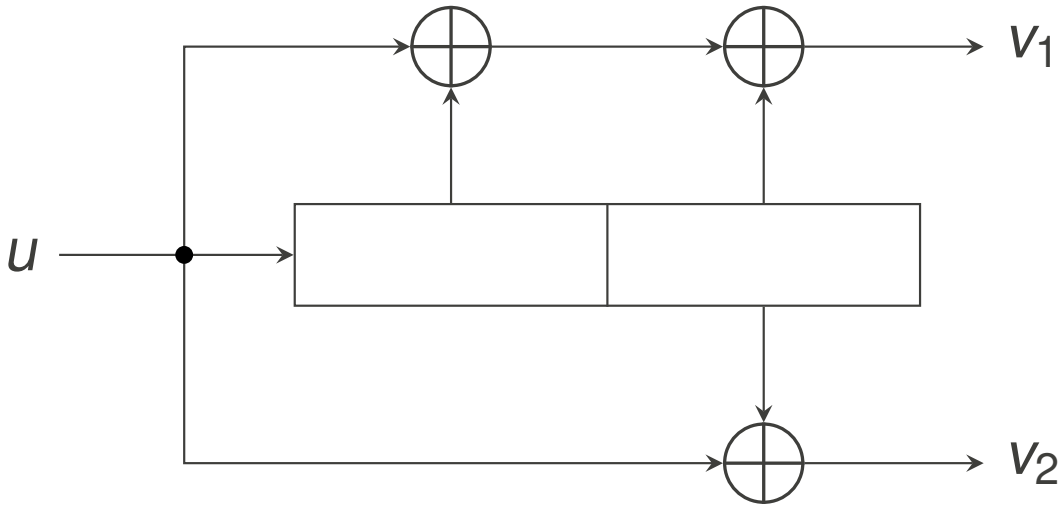
$$G = \begin{pmatrix} 111 \\ 101 \end{pmatrix}$$

Faltungskodierer



$$G = \begin{pmatrix} 111 \\ 101 \end{pmatrix} = (7_8, 5_8)$$

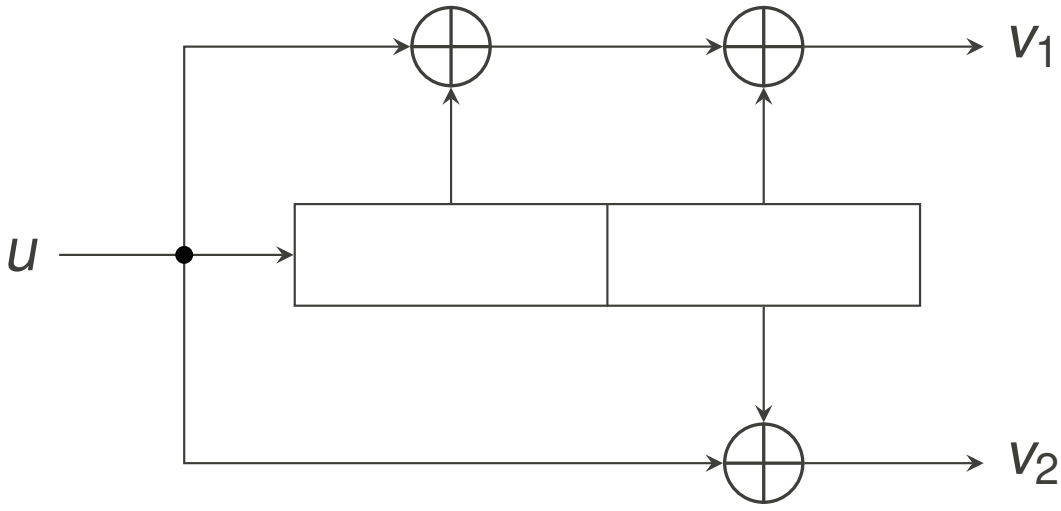
Faltungskodierer



$$G = \begin{pmatrix} 111 \\ 101 \end{pmatrix} = (7_8, 5_8)$$

$$f_k = \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor$$

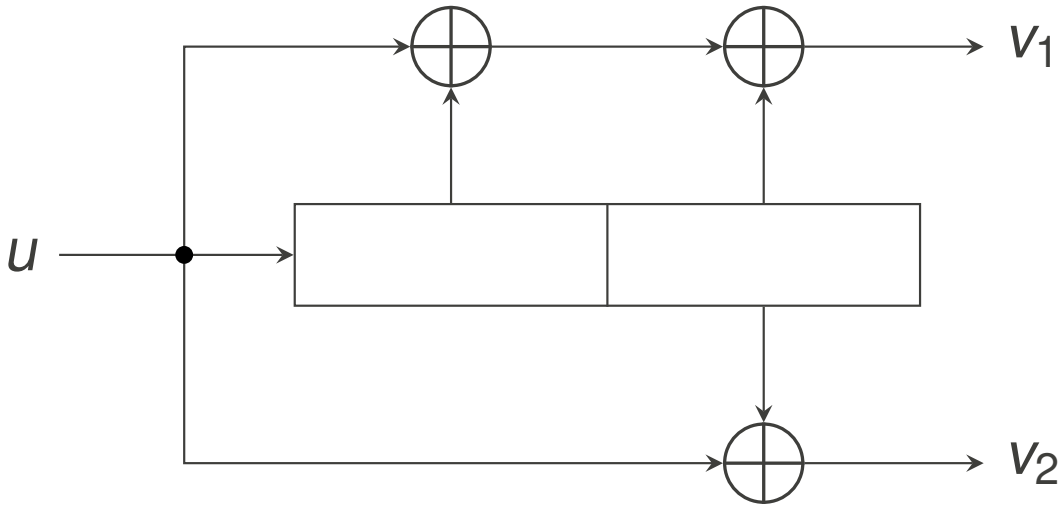
Faltungskodierer



$$G = \begin{pmatrix} 111 \\ 101 \end{pmatrix} = (7_8, 5_8)$$

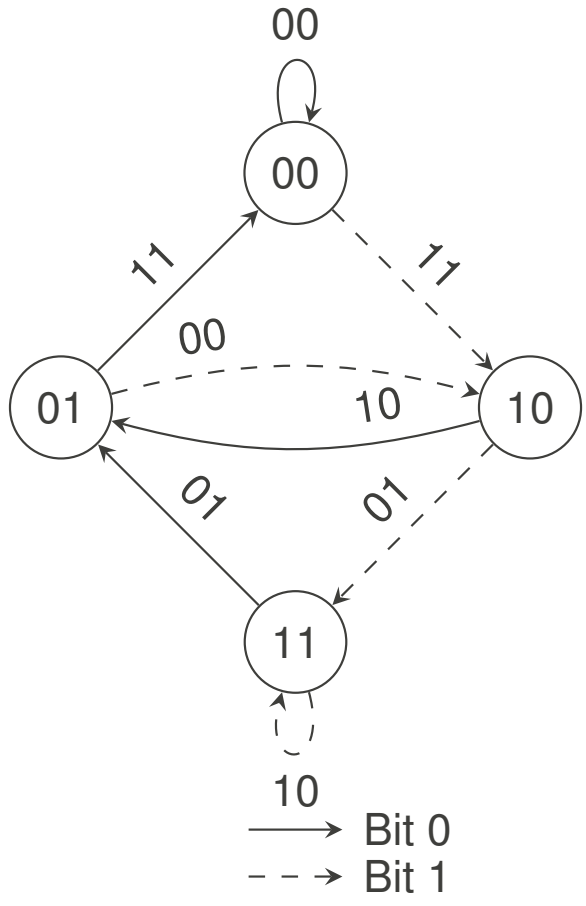
$$f_k = \left\lfloor \frac{w(G) - 1}{2} \right\rfloor$$

Faltungskodierer



$$G = \begin{pmatrix} 111 \\ 101 \end{pmatrix} = (7_8, 5_8)$$

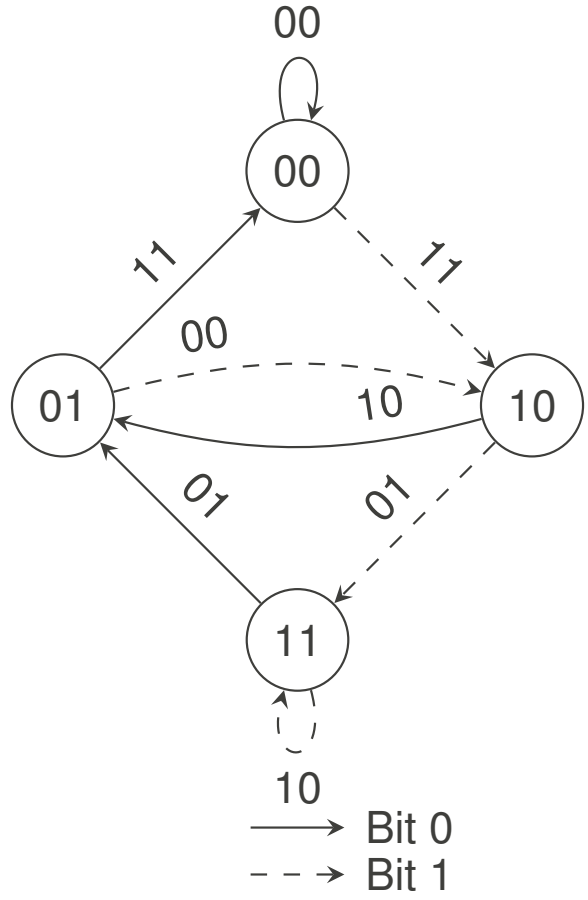
$$f_k = \left\lfloor \frac{w(G) - 1}{2} \right\rfloor$$



Faltungskodierung

Nachricht: (1, 1, 0, 1, 0, 0)

Input	Zustand	Folgezustand	Output

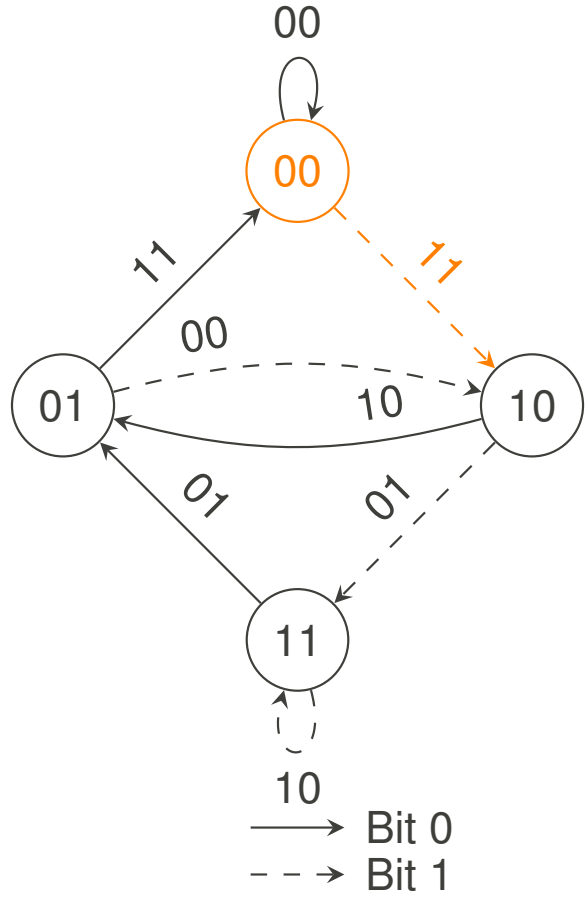


Kode :

Faltungskodierung

Nachricht: (1, 1, 0, 1, 0, 0)

Input	Zustand	Folgezustand	Output
1	00	10	11

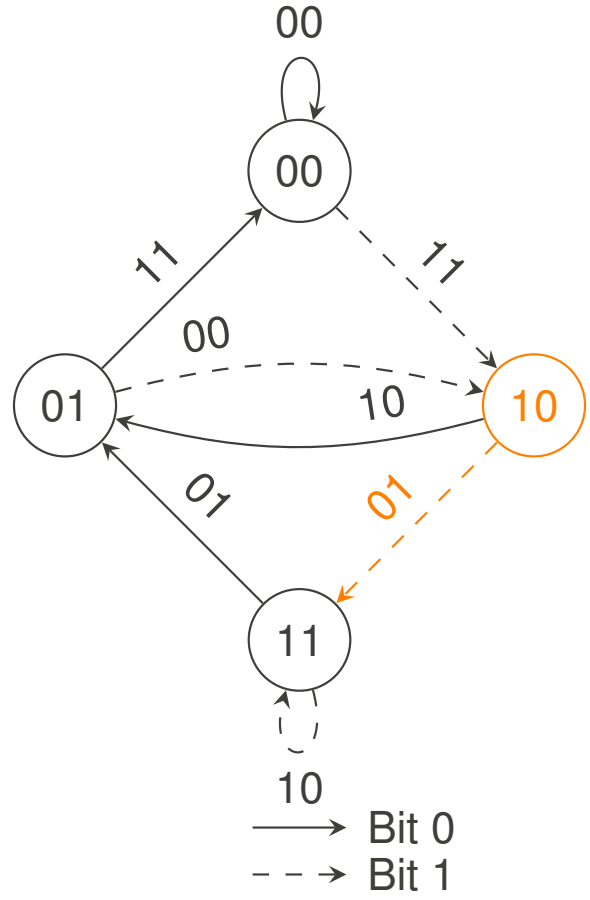


Kode : (1, 1

Faltungskodierung

Nachricht: (1, 1, 0, 1, 0, 0)

Input	Zustand	Folgezustand	Output
1	00	10	11
1	10	11	01

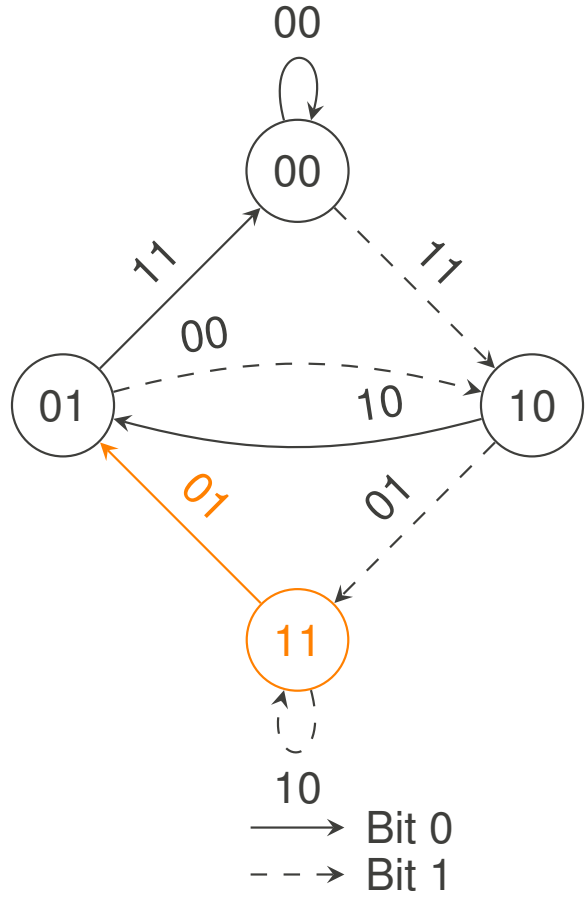


Kode : (1, 1, 0, 1

Faltungskodierung

Nachricht: (1, 1, 0, 1, 0, 0)

Input	Zustand	Folgezustand	Output
1	00	10	11
1	10	11	01
0	11	01	01

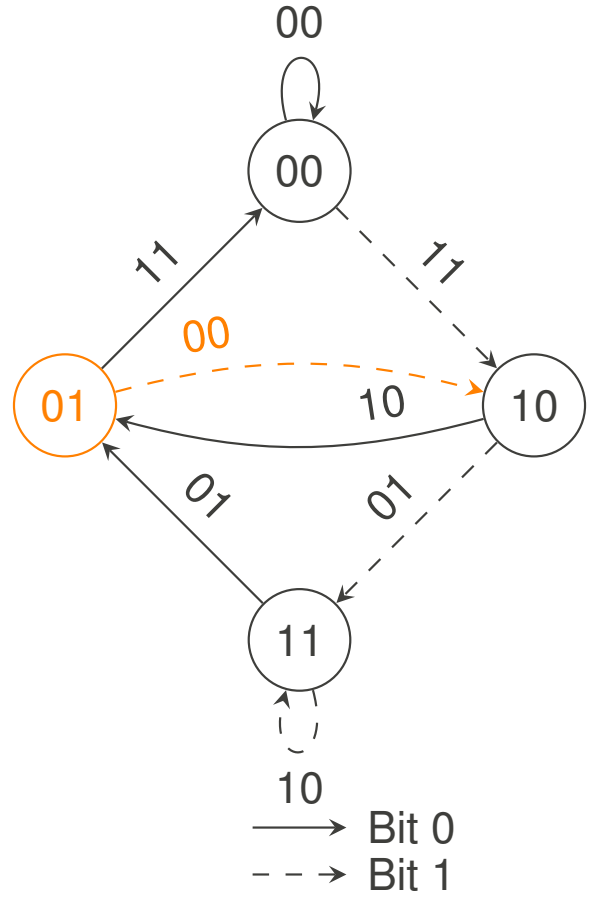


Kode : (1, 1, 0, 1, 0, 1

Faltungskodierung

Nachricht: (1, 1, 0, 1, 0, 0)

Input	Zustand	Folgezustand	Output
1	00	10	11
1	10	11	01
0	11	01	01
1	01	10	00

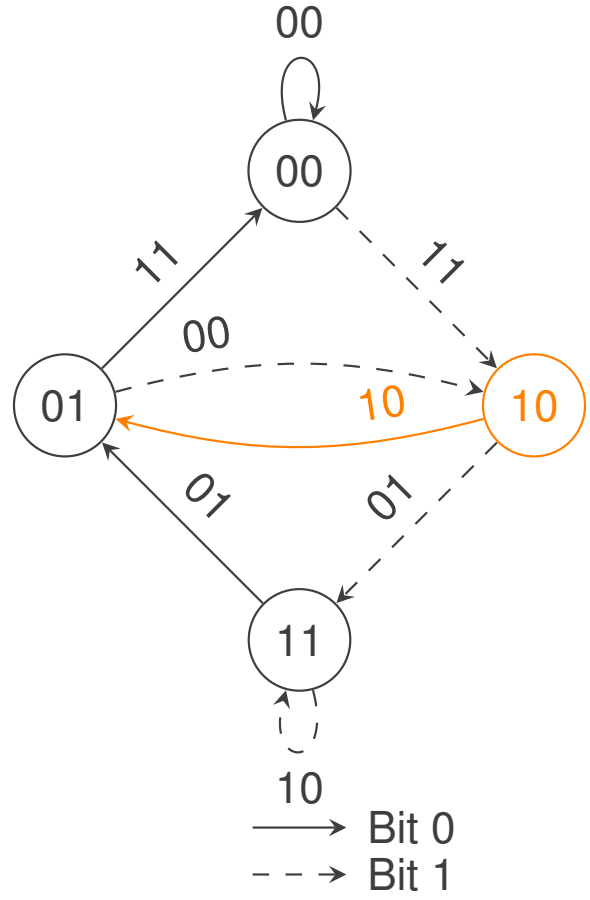


Kode : (1, 1, 0, 1, 0, 1, 0, 0

Faltungskodierung

Nachricht: (1, 1, 0, 1, 0, 0)

Input	Zustand	Folgezustand	Output
1	00	10	11
1	10	11	01
0	11	01	01
1	01	10	00
0	10	01	10

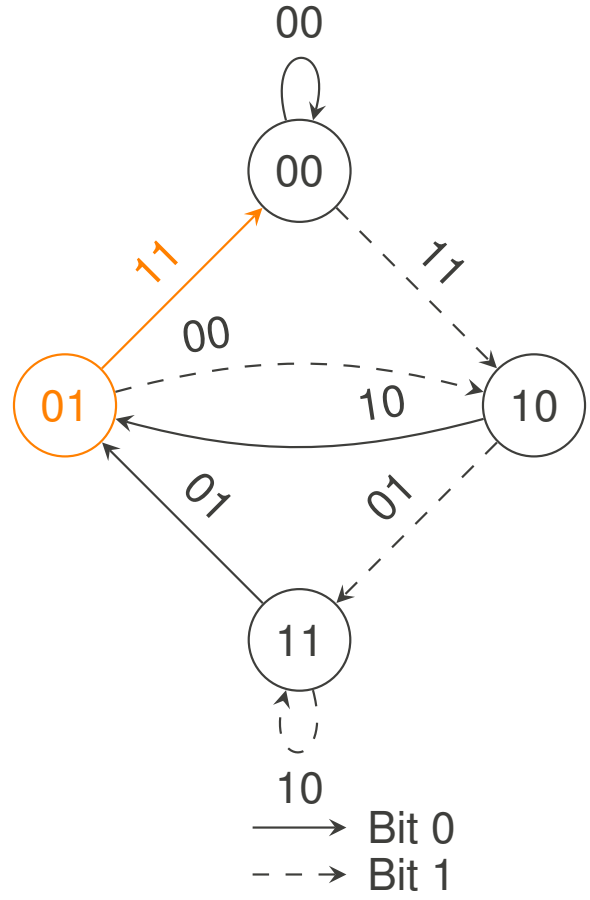


Kode : (1, 1, 0, 1, 0, 1, 0, 0, 1, 0

Faltungskodierung

Nachricht: (1, 1, 0, 1, 0, 0)

Input	Zustand	Folgezustand	Output
1	00	10	11
1	10	11	01
0	11	01	01
1	01	10	00
0	10	01	10
0	01	00	11

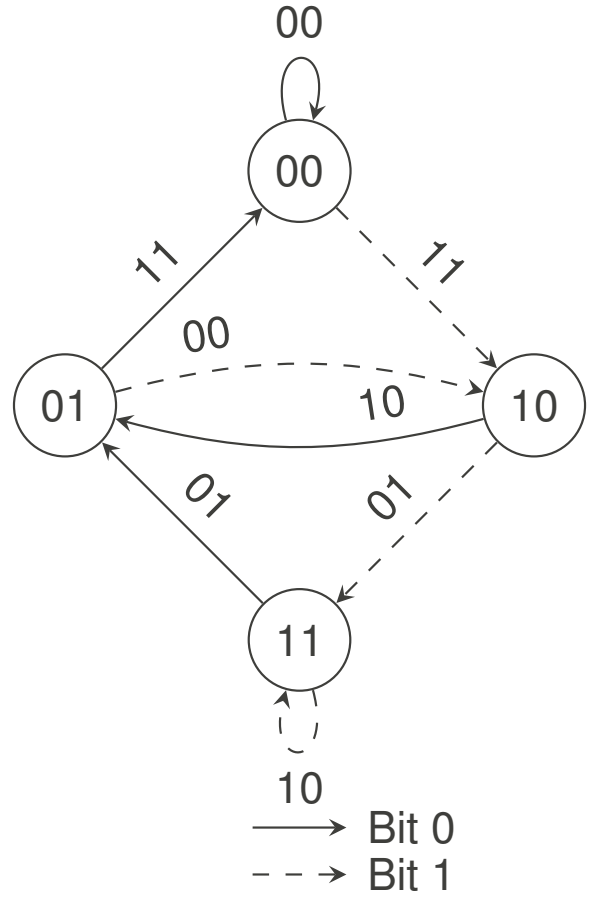


Kode : (1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1)

Faltungskodierung

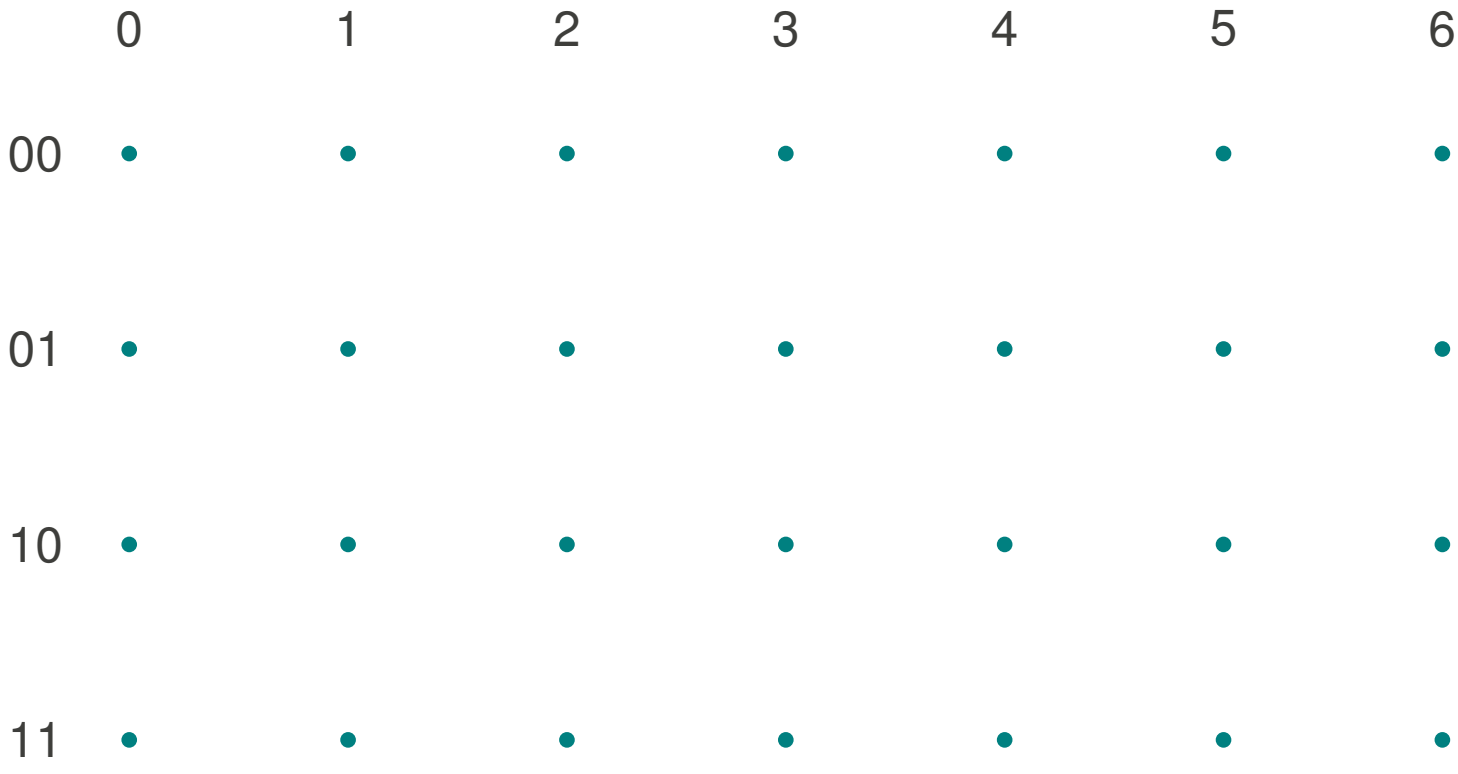
Nachricht: (1, 1, 0, 1, 0, 0)

Input	Zustand	Folgezustand	Output
1	00	10	11
1	10	11	01
0	11	01	01
1	01	10	00
0	10	01	10
0	01	00	11

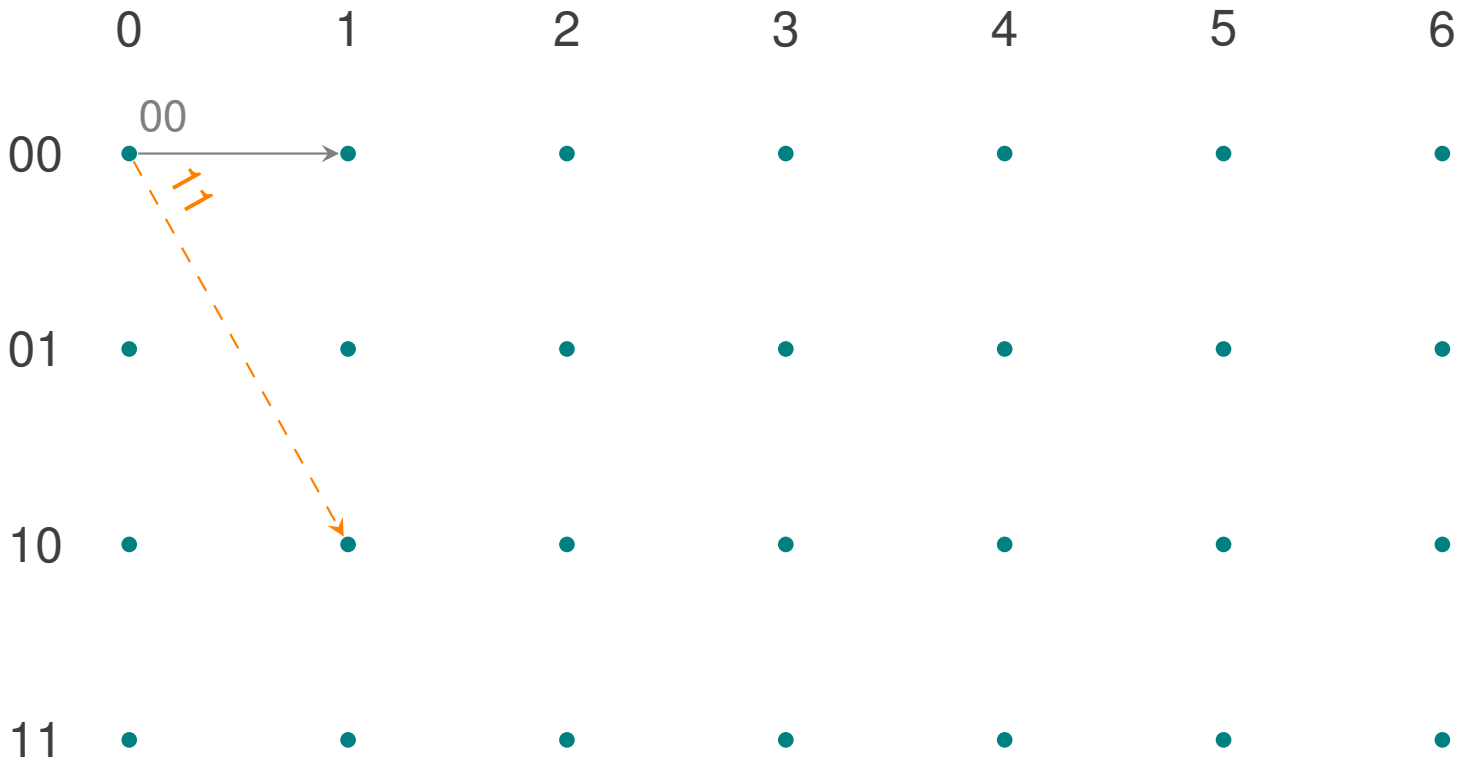


Kode : (1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1)

Trellis - Kodierung



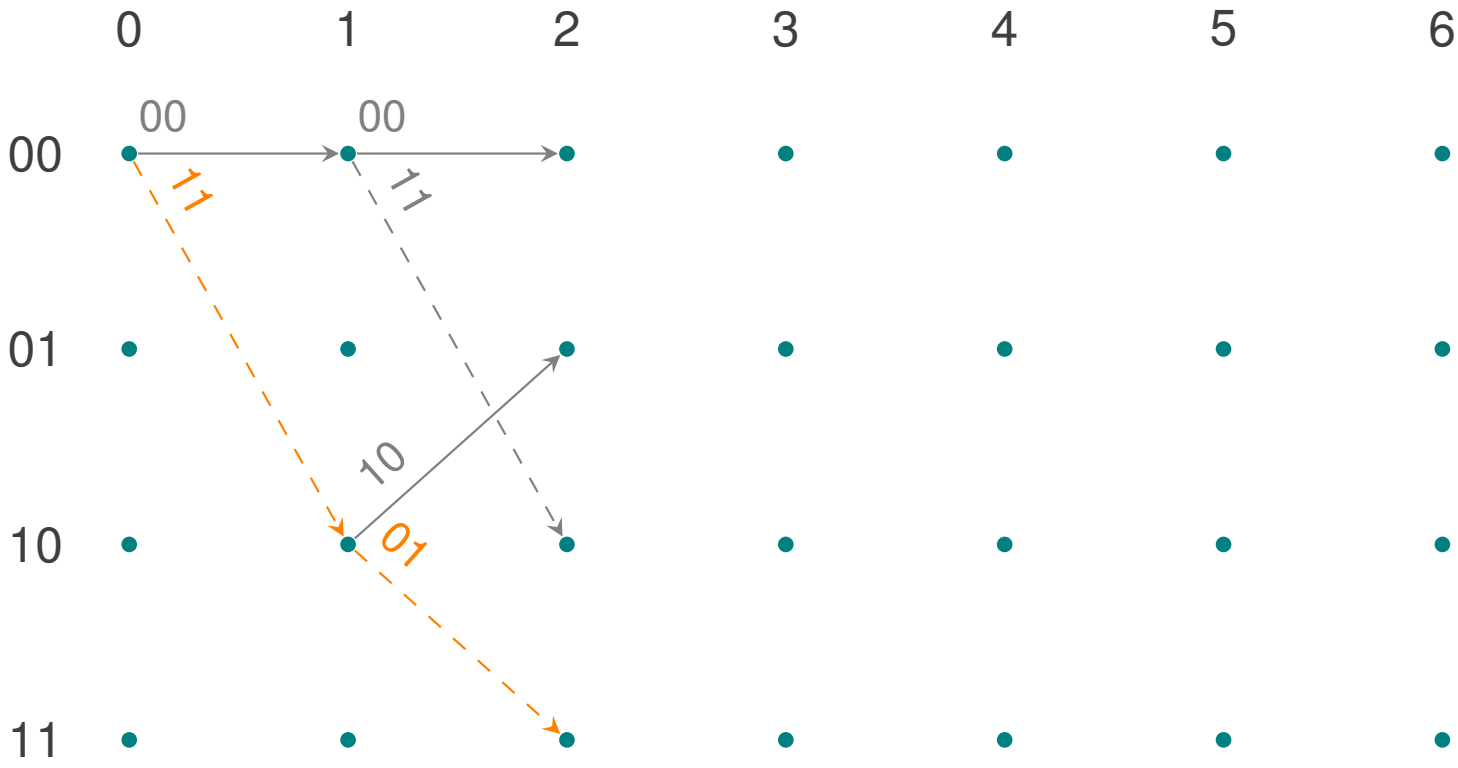
Trellis - Kodierung



Nachricht: 1 1 0 1 0 0

Kode: 11

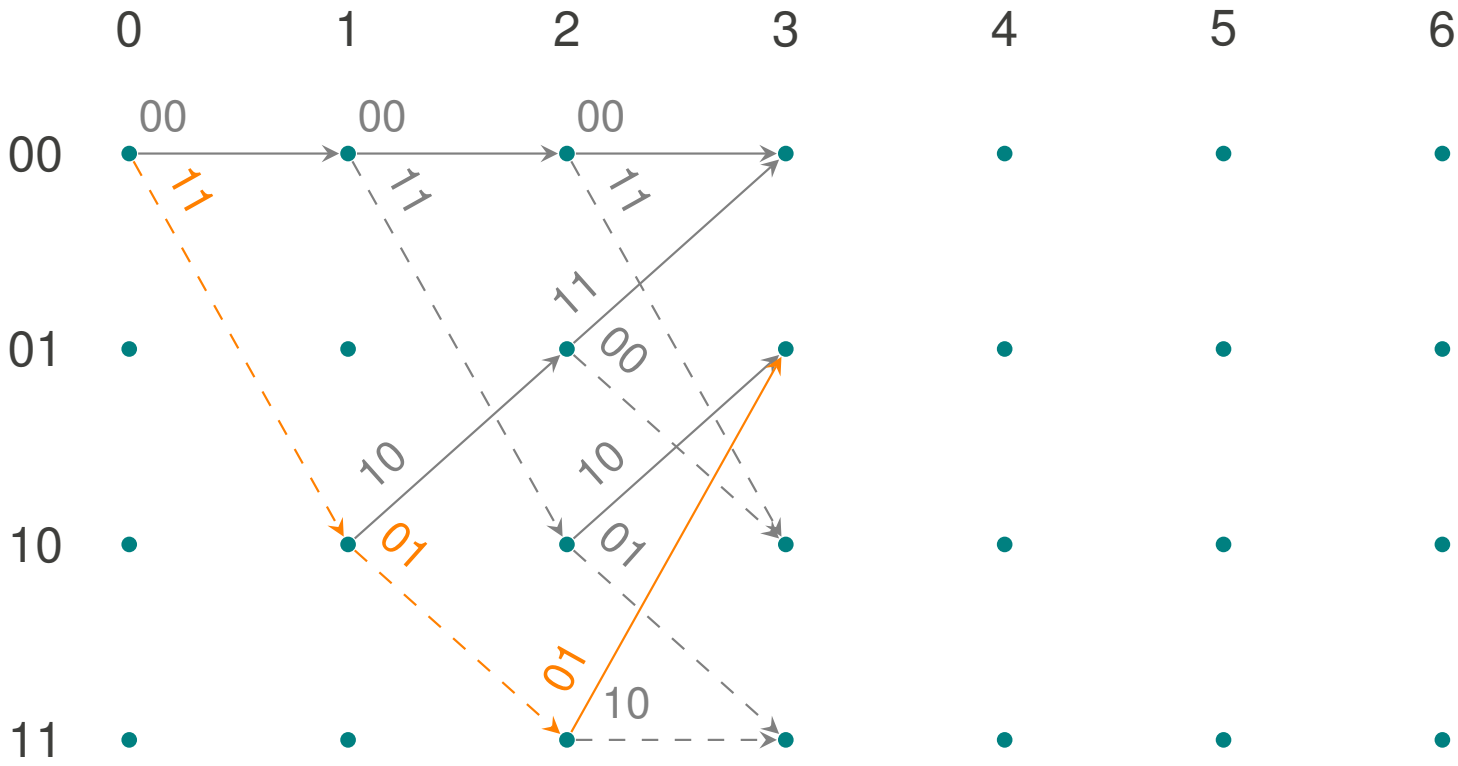
Trellis - Kodierung



Nachricht: 1 1 0 1 0 0

Kode: 11 01

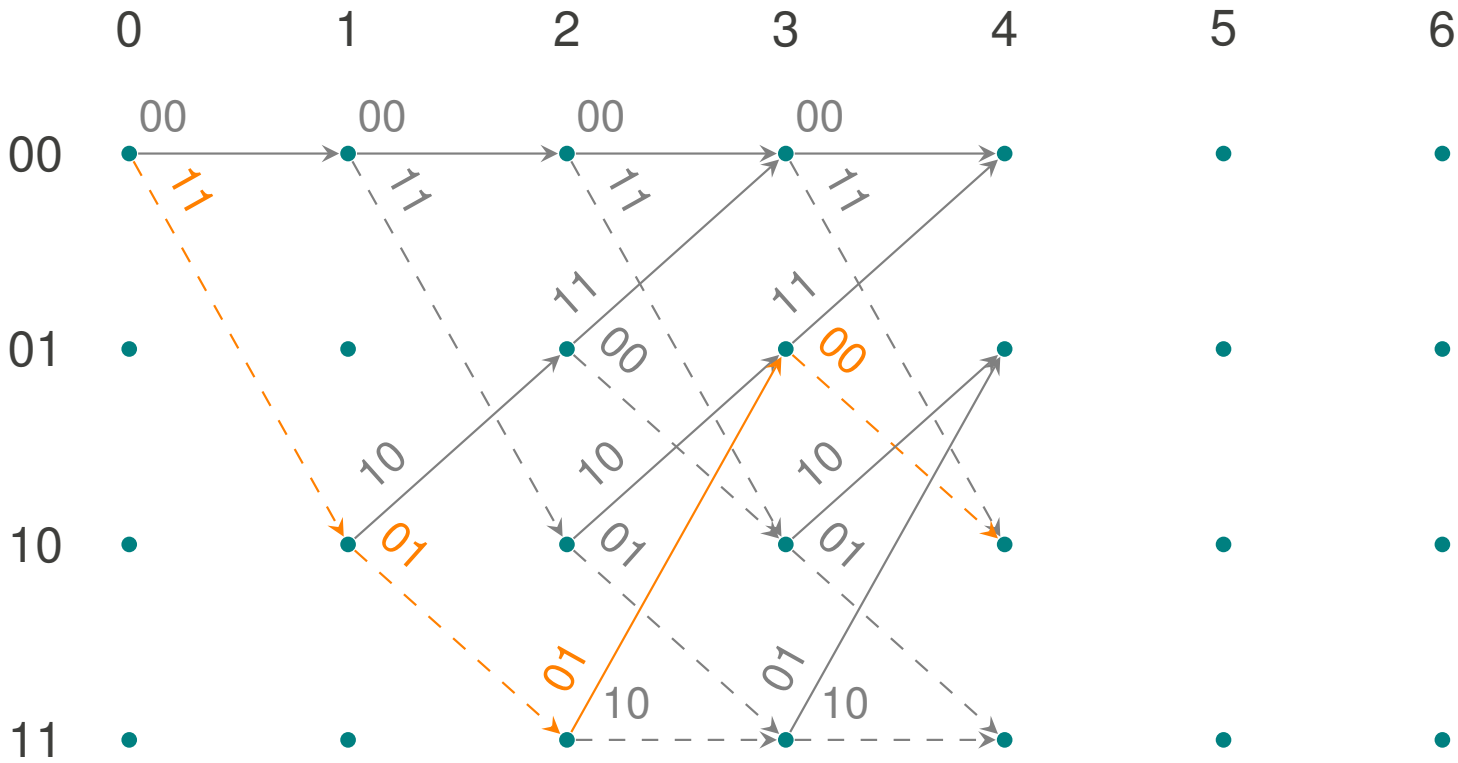
Trellis - Kodierung



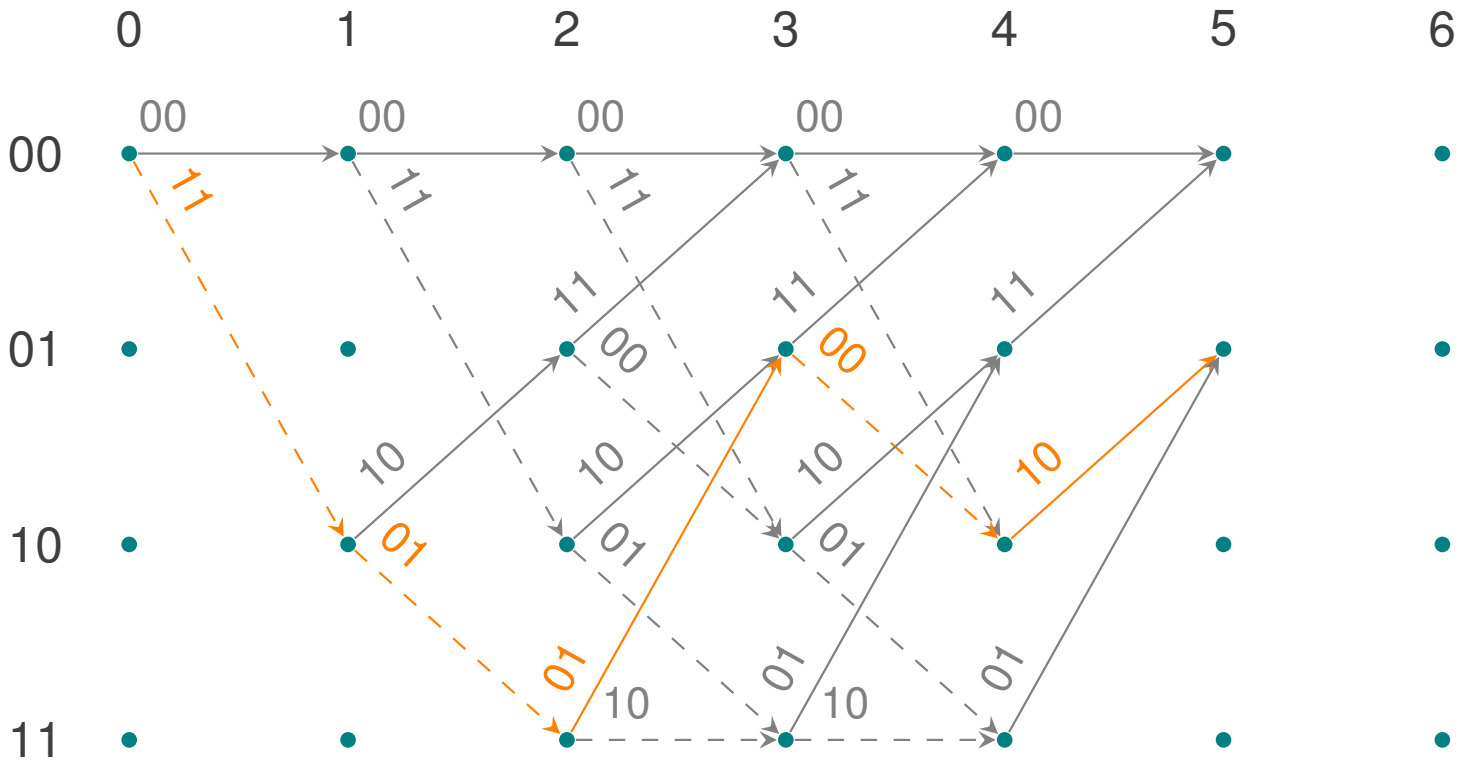
Nachricht: 1 1 0 1 0 0

Kode: 11 01 01

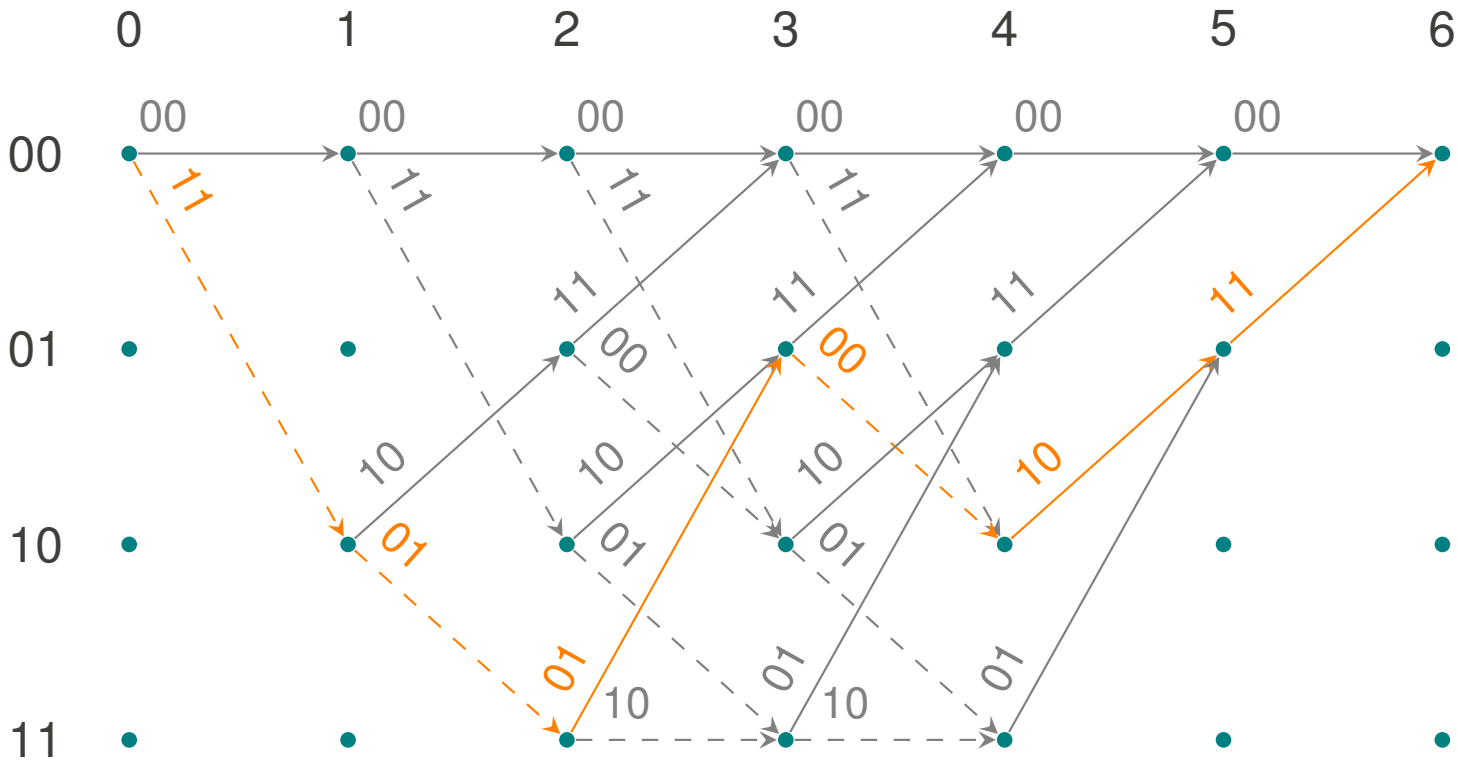
Trellis - Kodierung



Trellis - Kodierung



Trellis - Kodierung

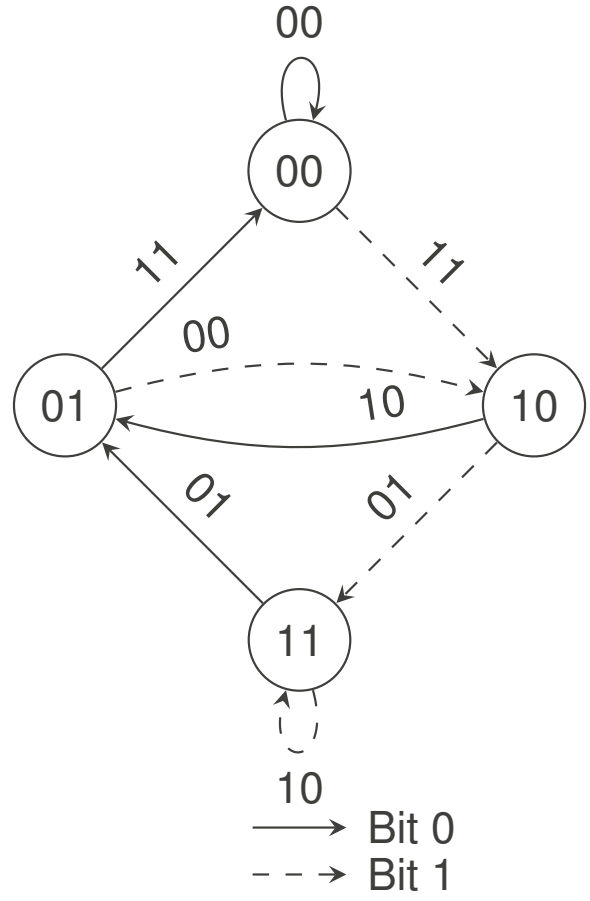


Nachricht:	1	1	0	1	0	0
Kode:	11	01	01	00	10	11

Faltungskodierung

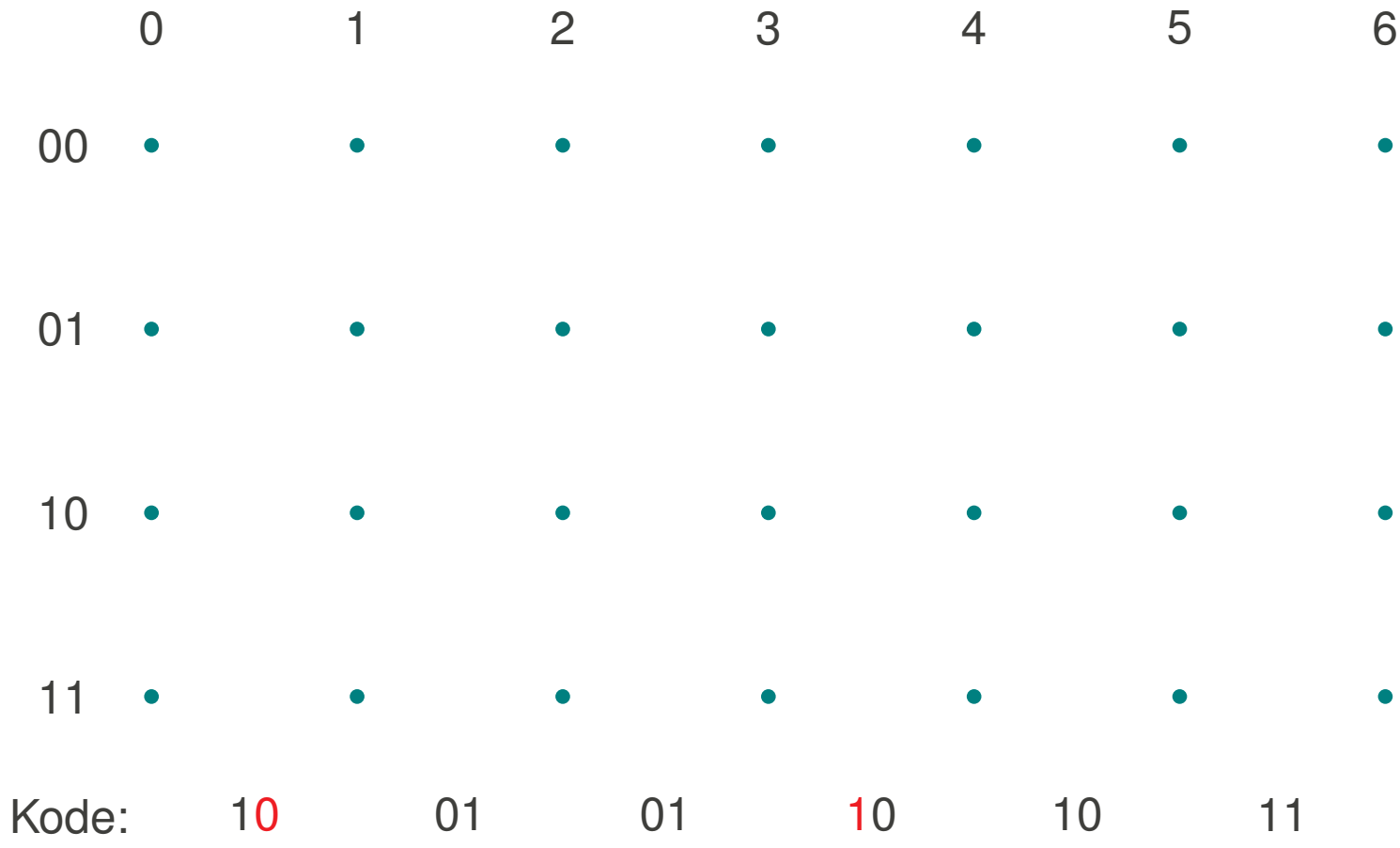
Nachricht: (1, 1, 0, 1, 0, 0)

Input	Zustand	Folgezustand	Output
1	00	10	11
1	10	11	01
0	11	01	01
1	01	10	00
0	10	01	10
0	01	00	11

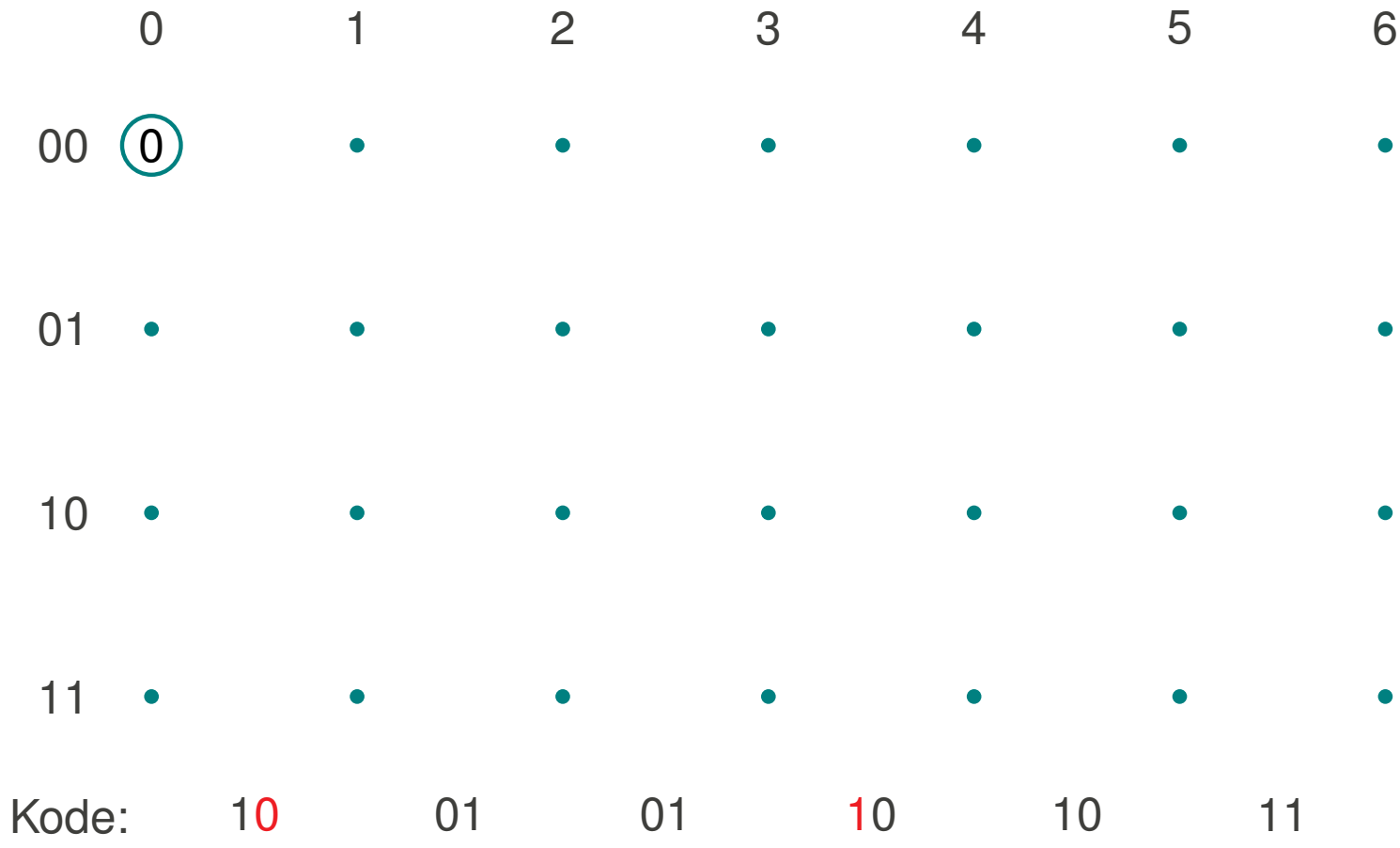


Kode : (1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1)

Faltungsdekodierung (Viterbi-Algorithmus)

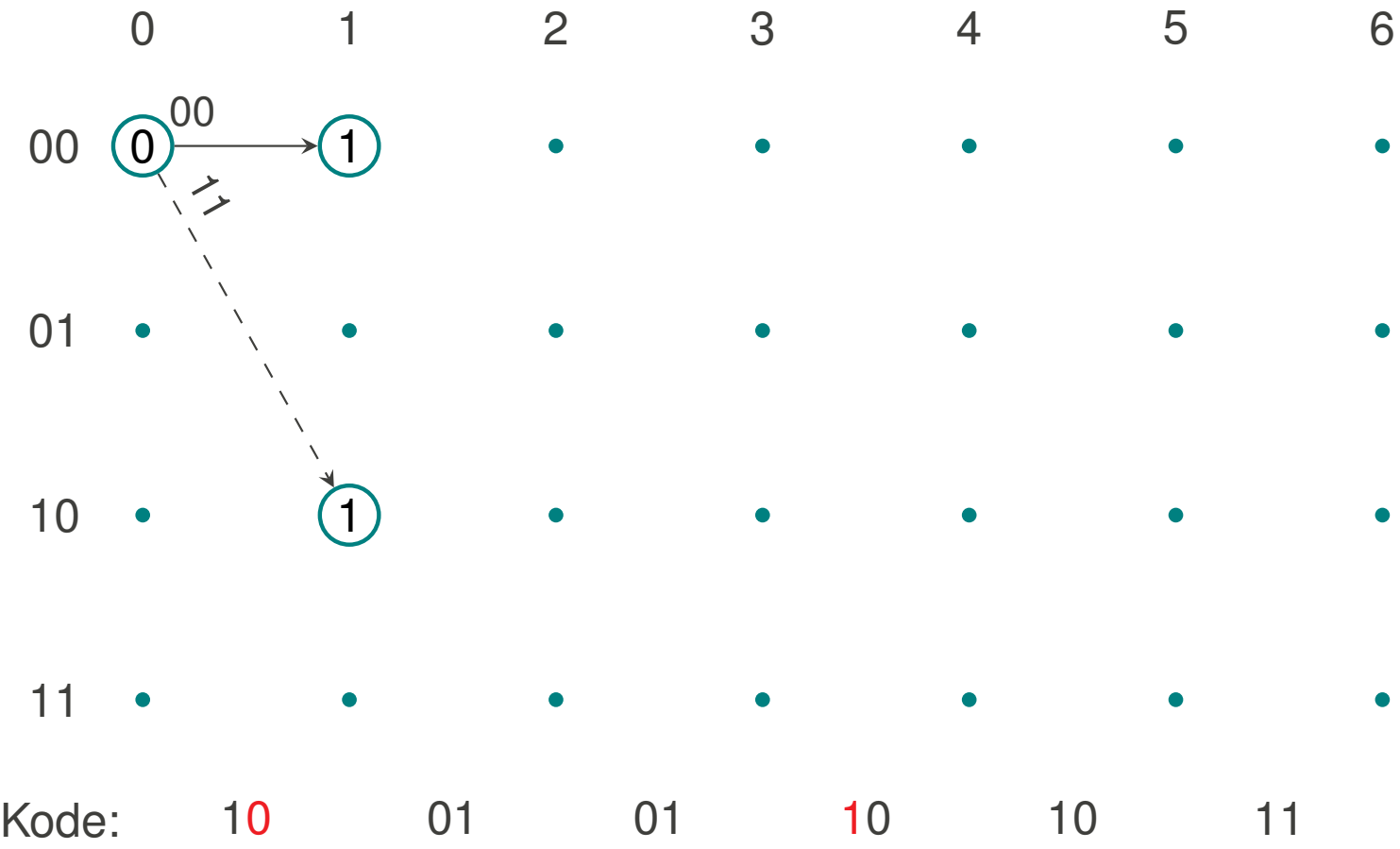


Faltungsdekodierung (Viterbi-Algorithmus)



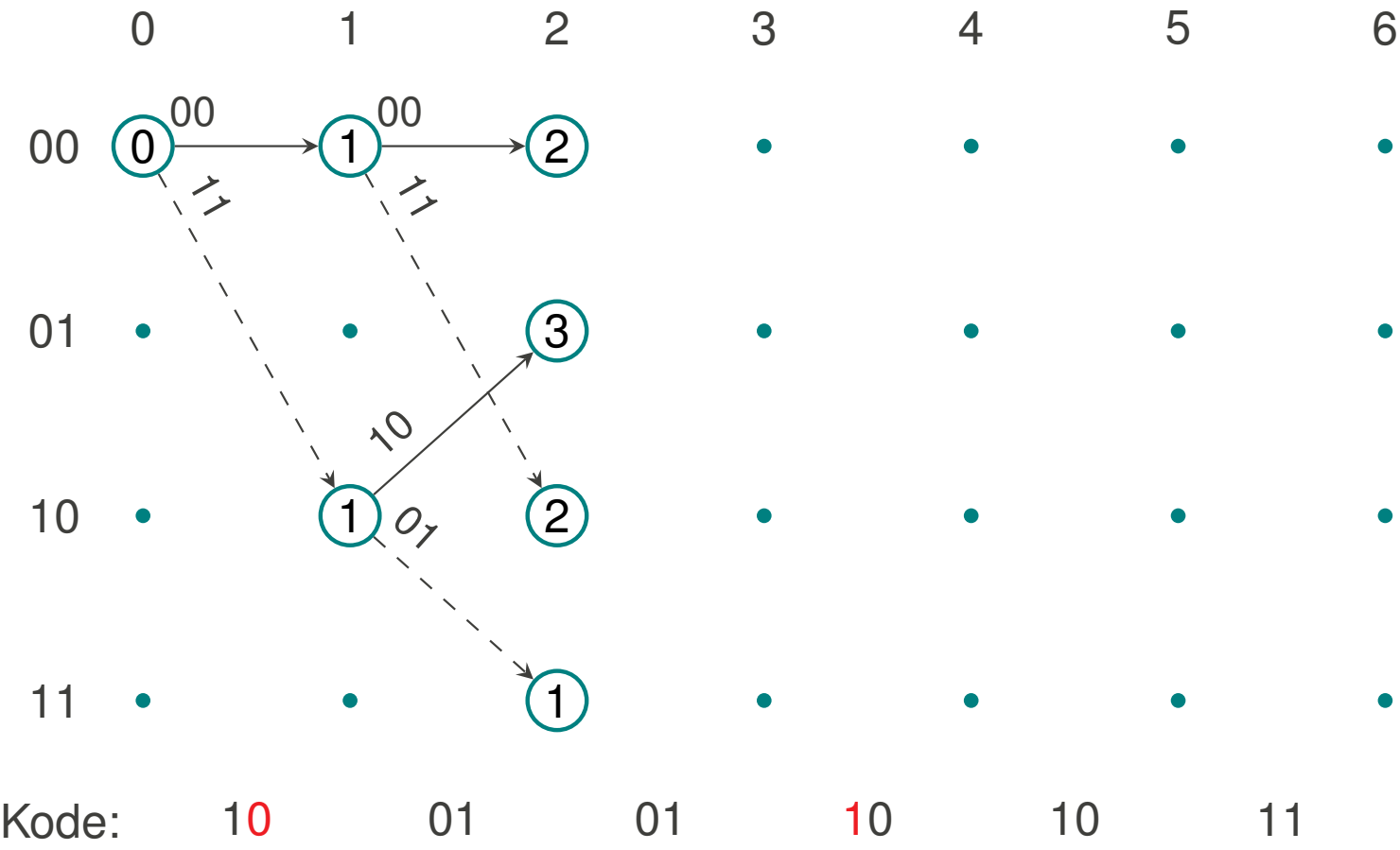
Nachricht:

Faltungsdekodierung (Viterbi-Algorithmus)



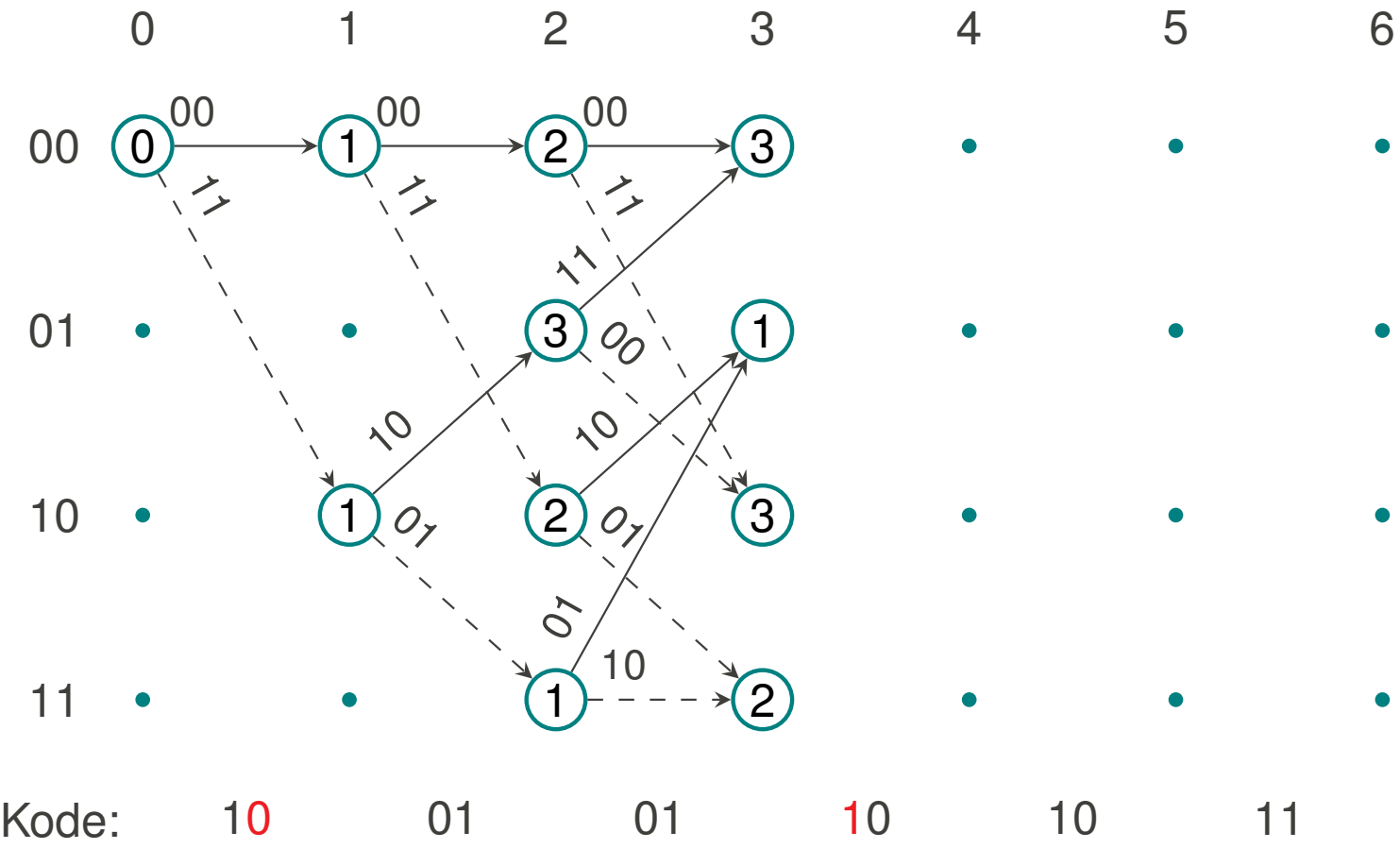
Nachricht:

Faltungsdekodierung (Viterbi-Algorithmus)



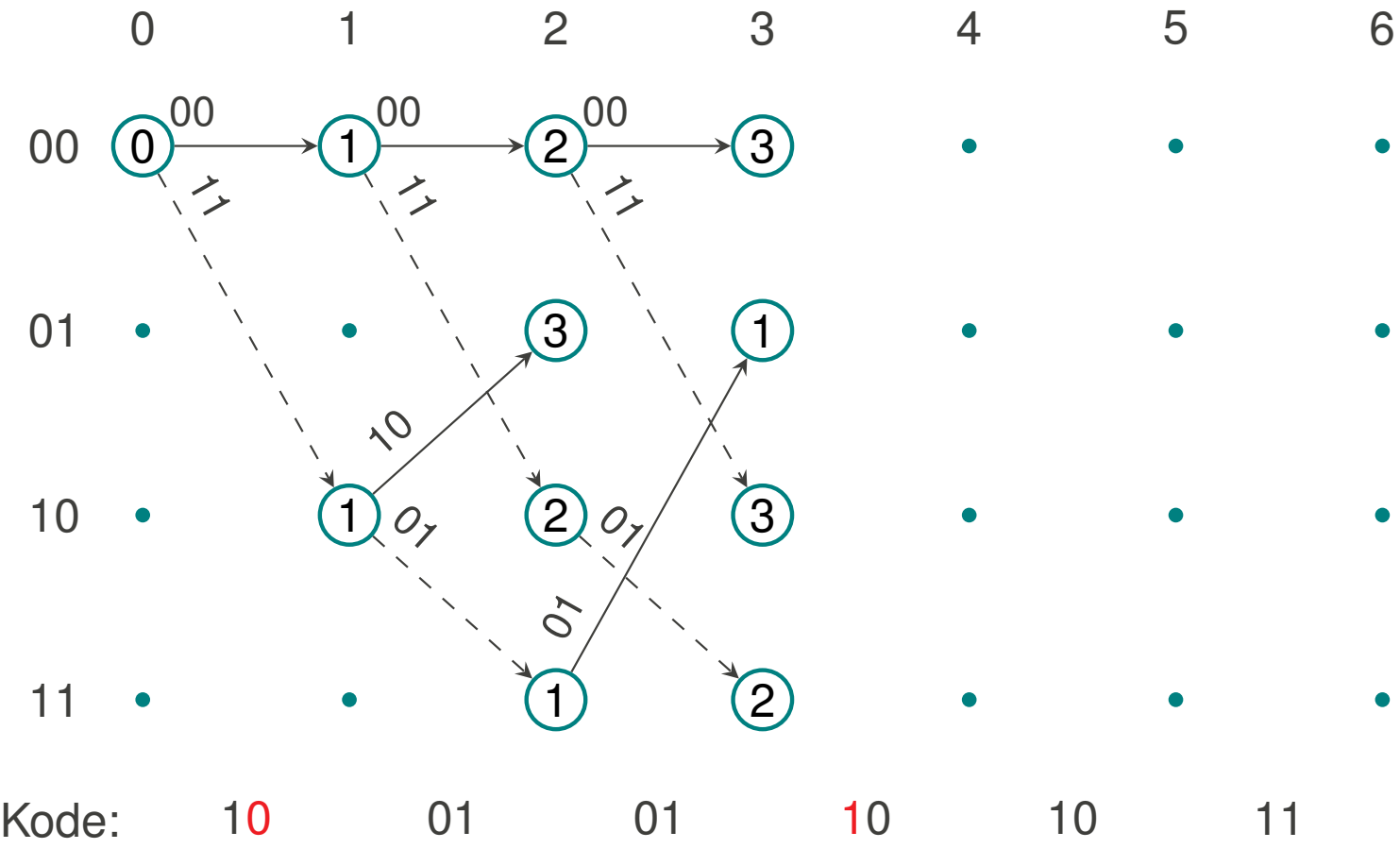
Nachricht:

Faltungsdekodierung (Viterbi-Algorithmus)



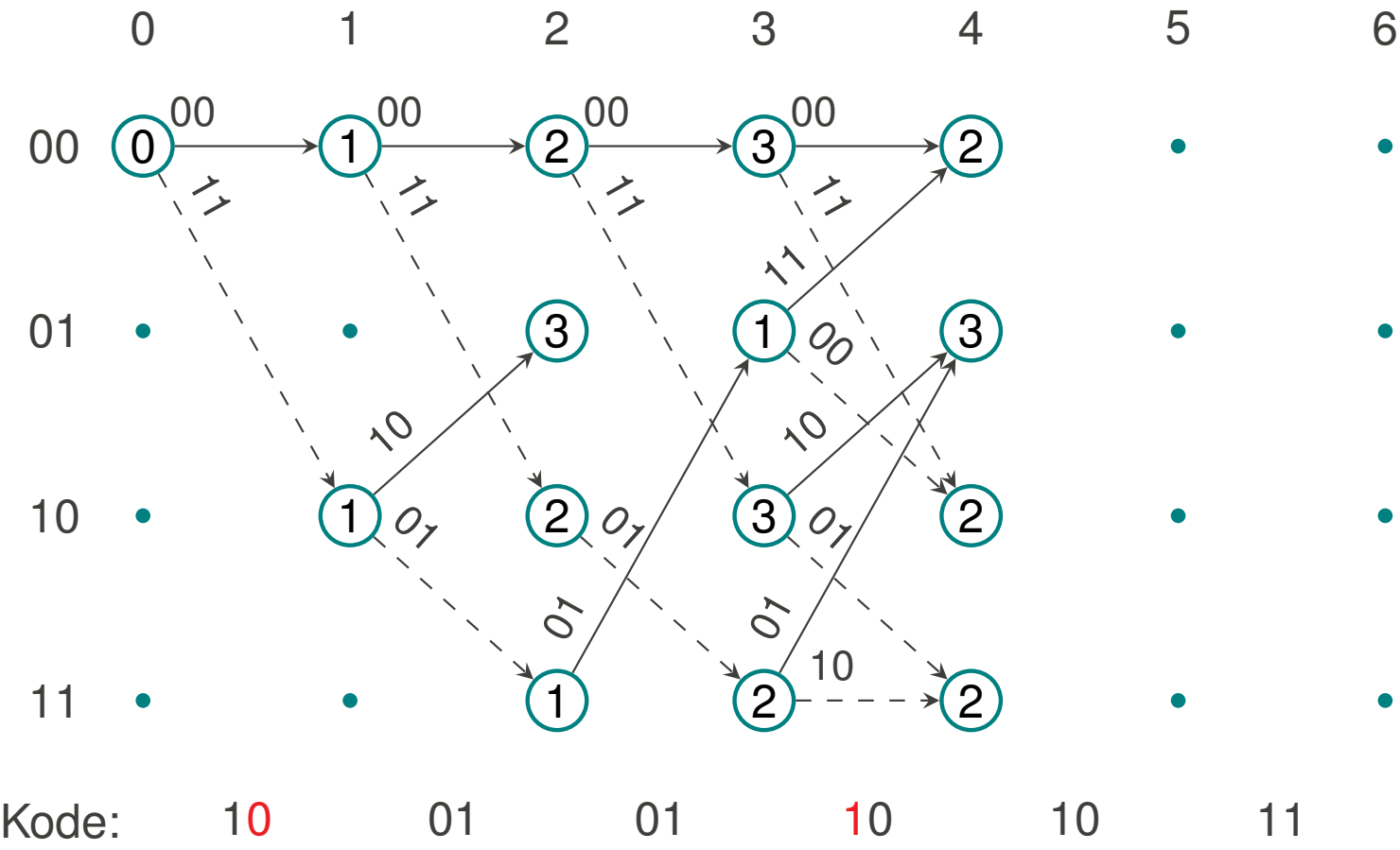
Nachricht:

Faltungsdekodierung (Viterbi-Algorithmus)

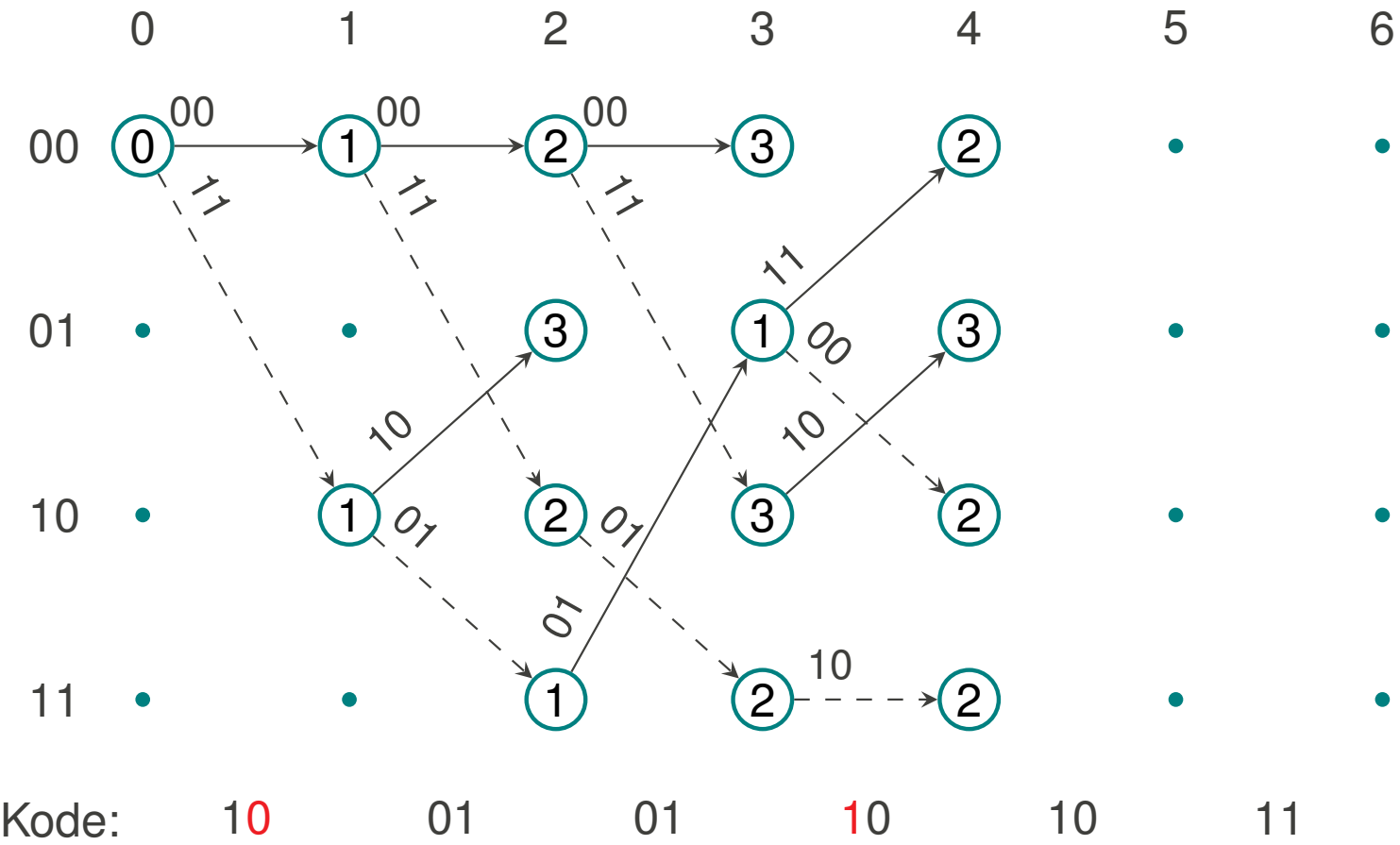


Nachricht:

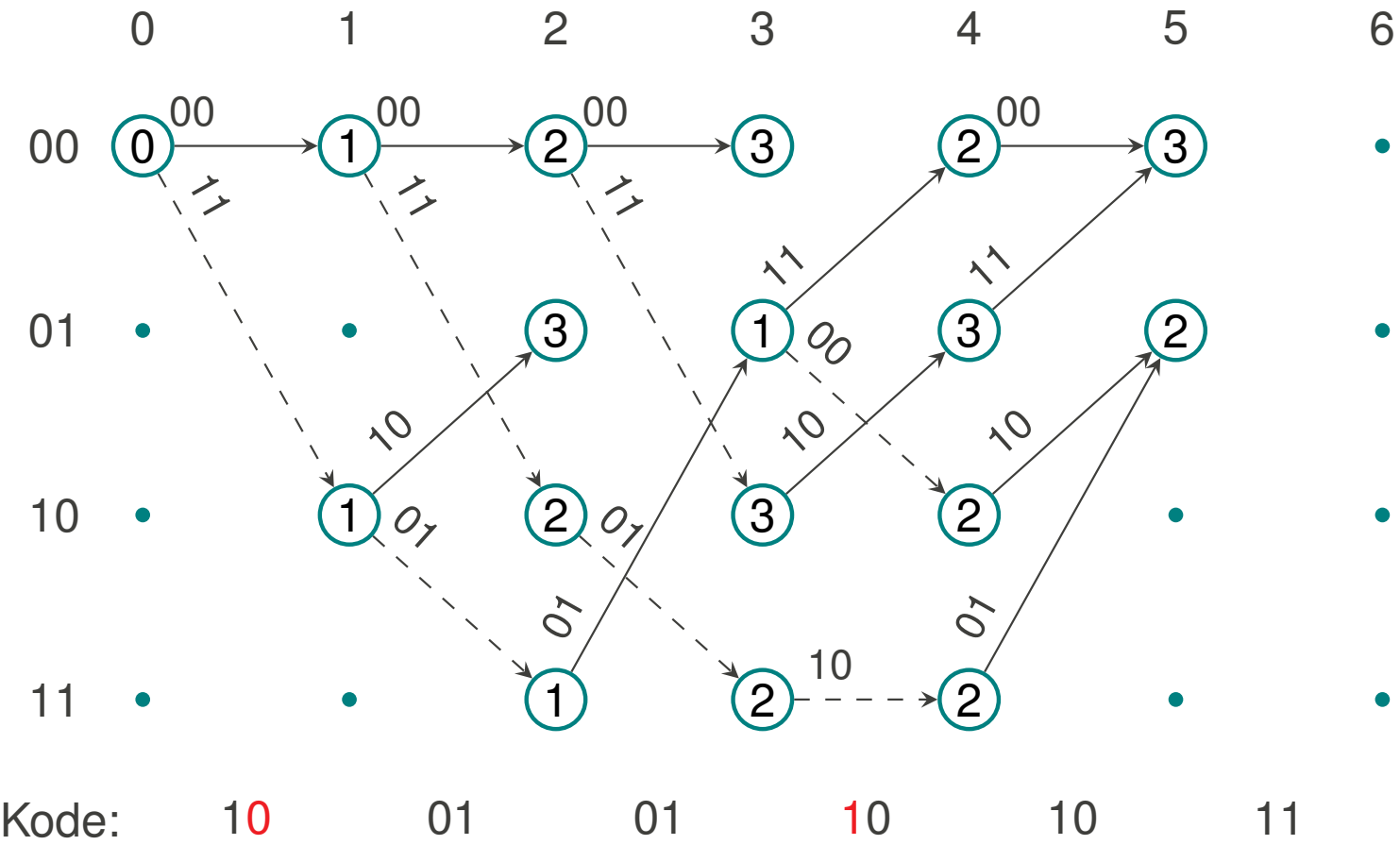
Faltungsdekodierung (Viterbi-Algorithmus)



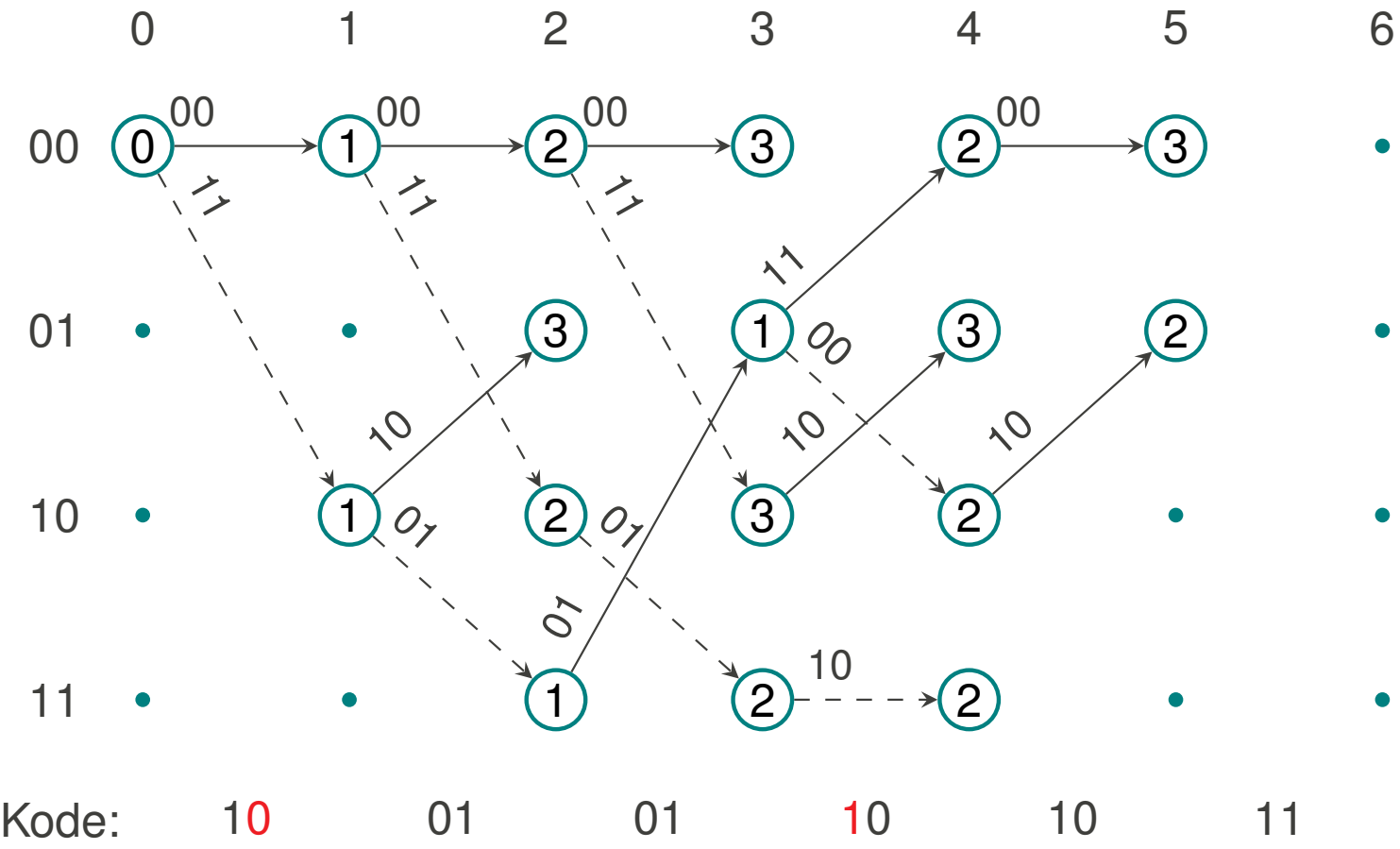
Faltungsdekodierung (Viterbi-Algorithmus)



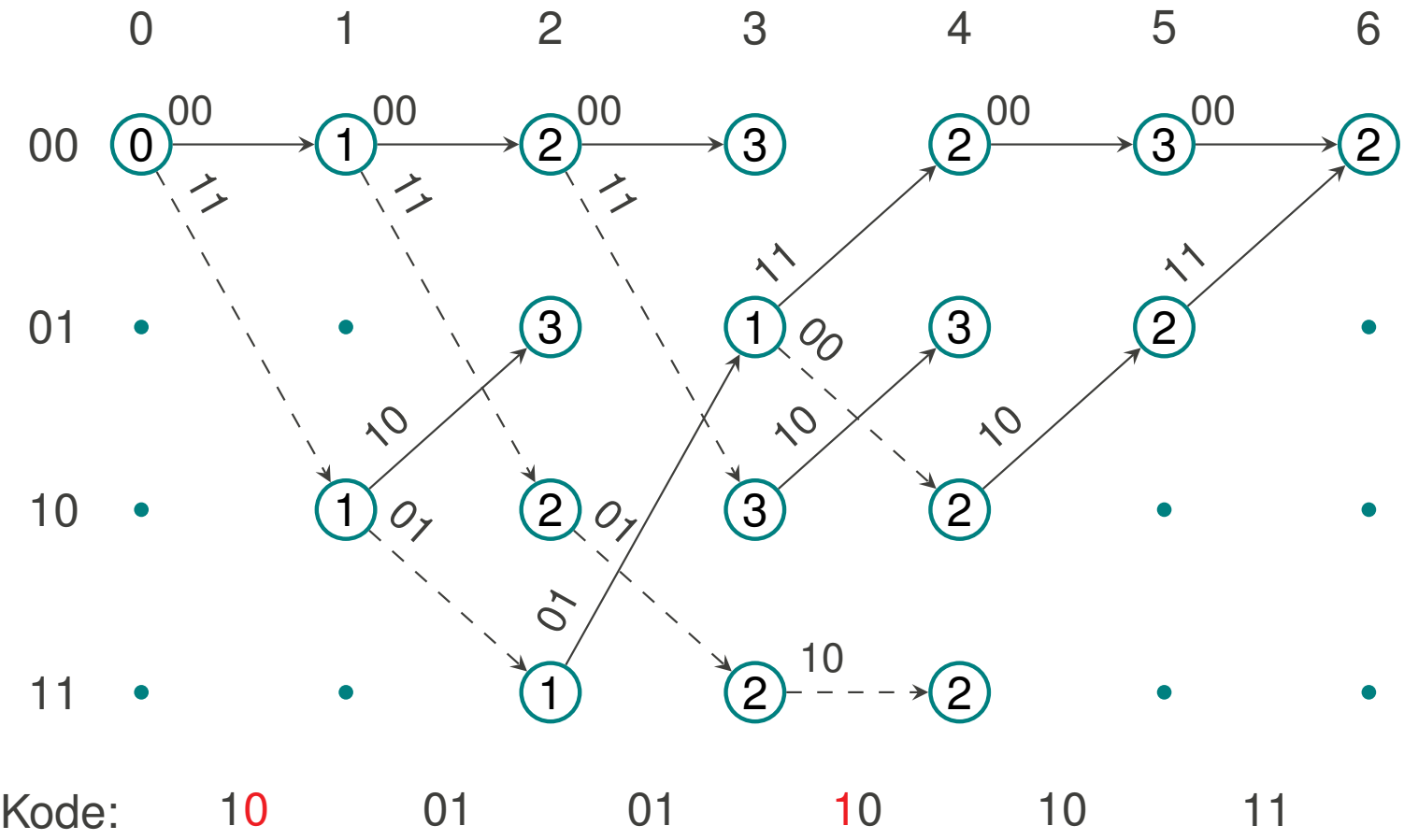
Faltungsdekodierung (Viterbi-Algorithmus)



Faltungsdekodierung (Viterbi-Algorithmus)

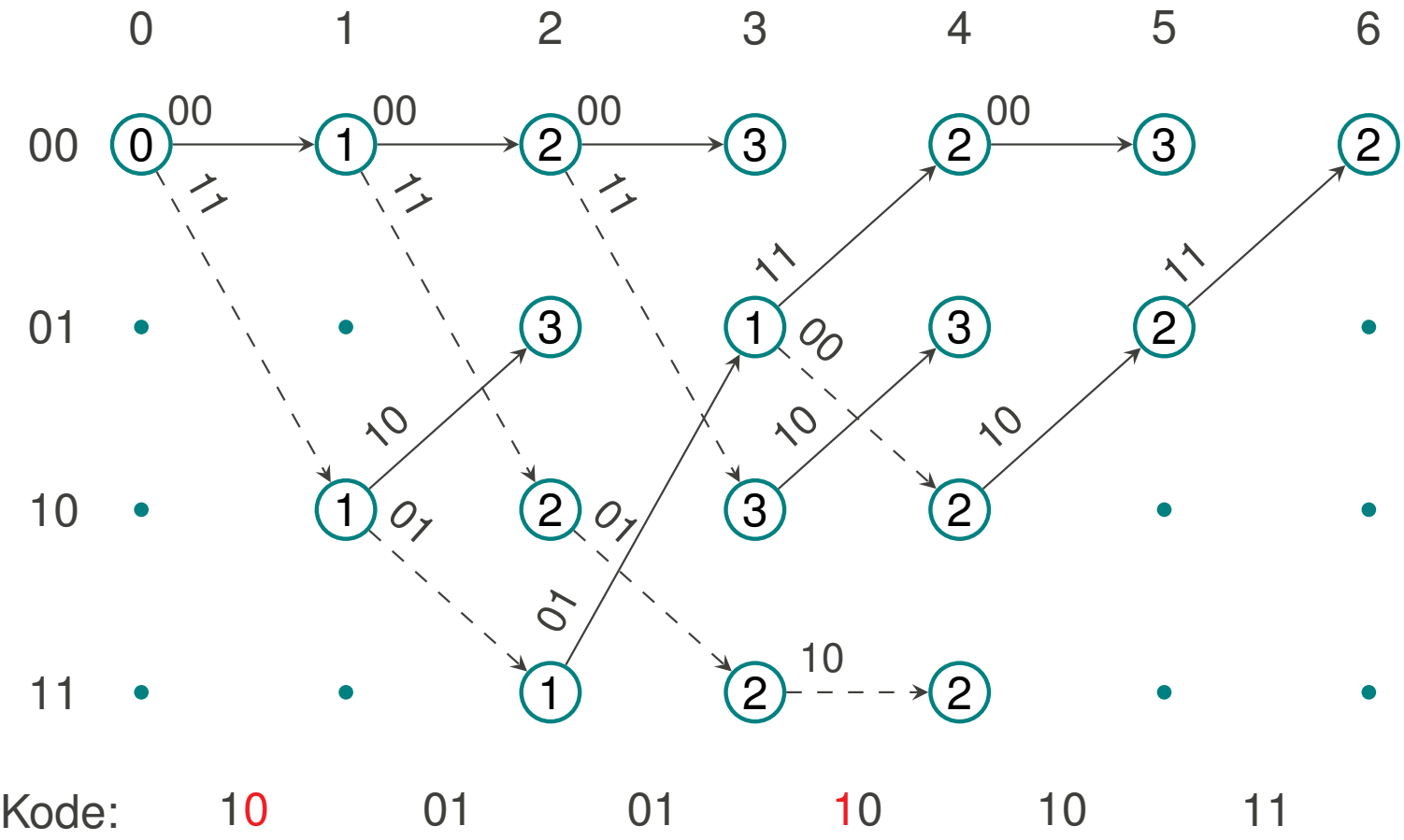


Faltungsdekodierung (Viterbi-Algorithmus)

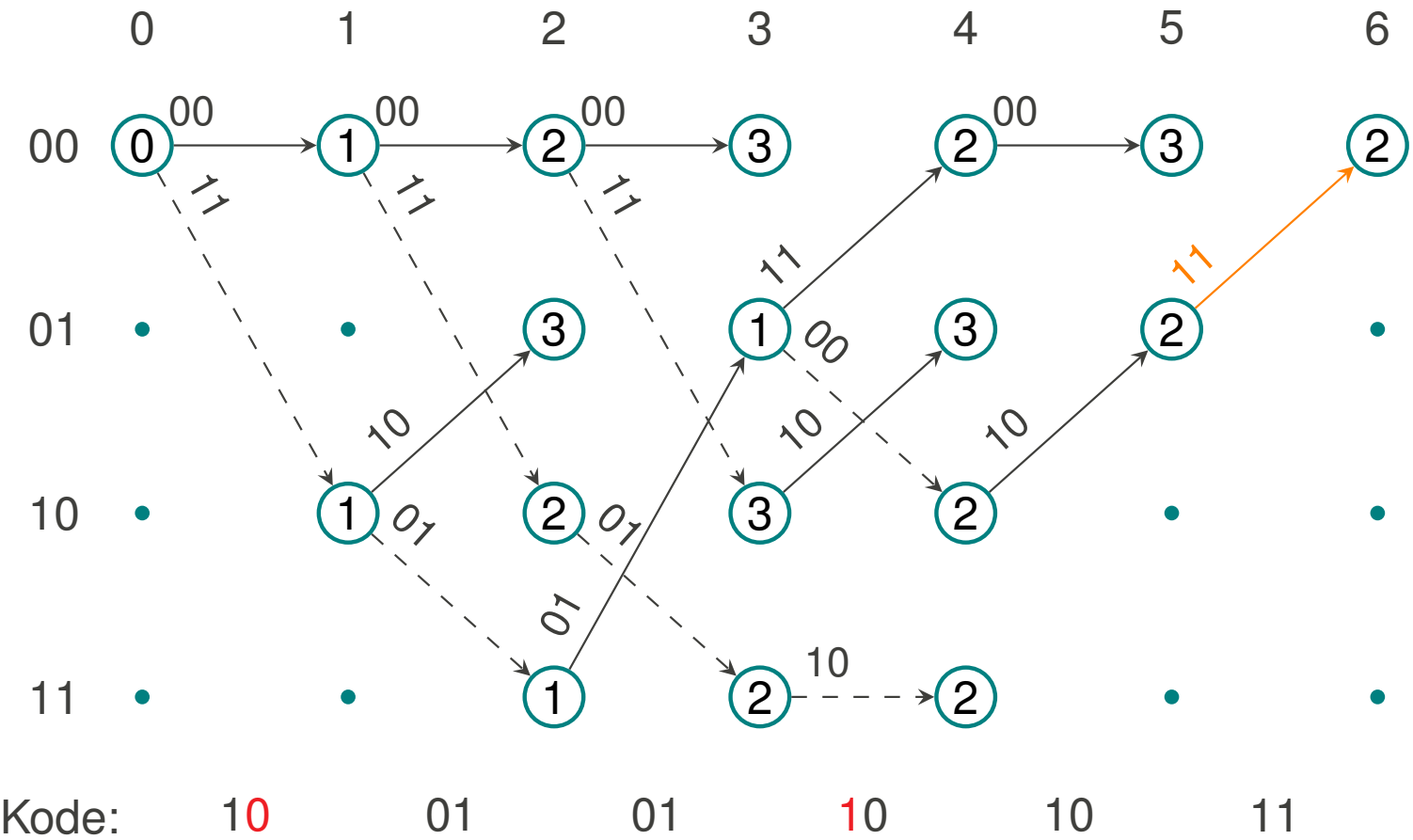


Nachricht:

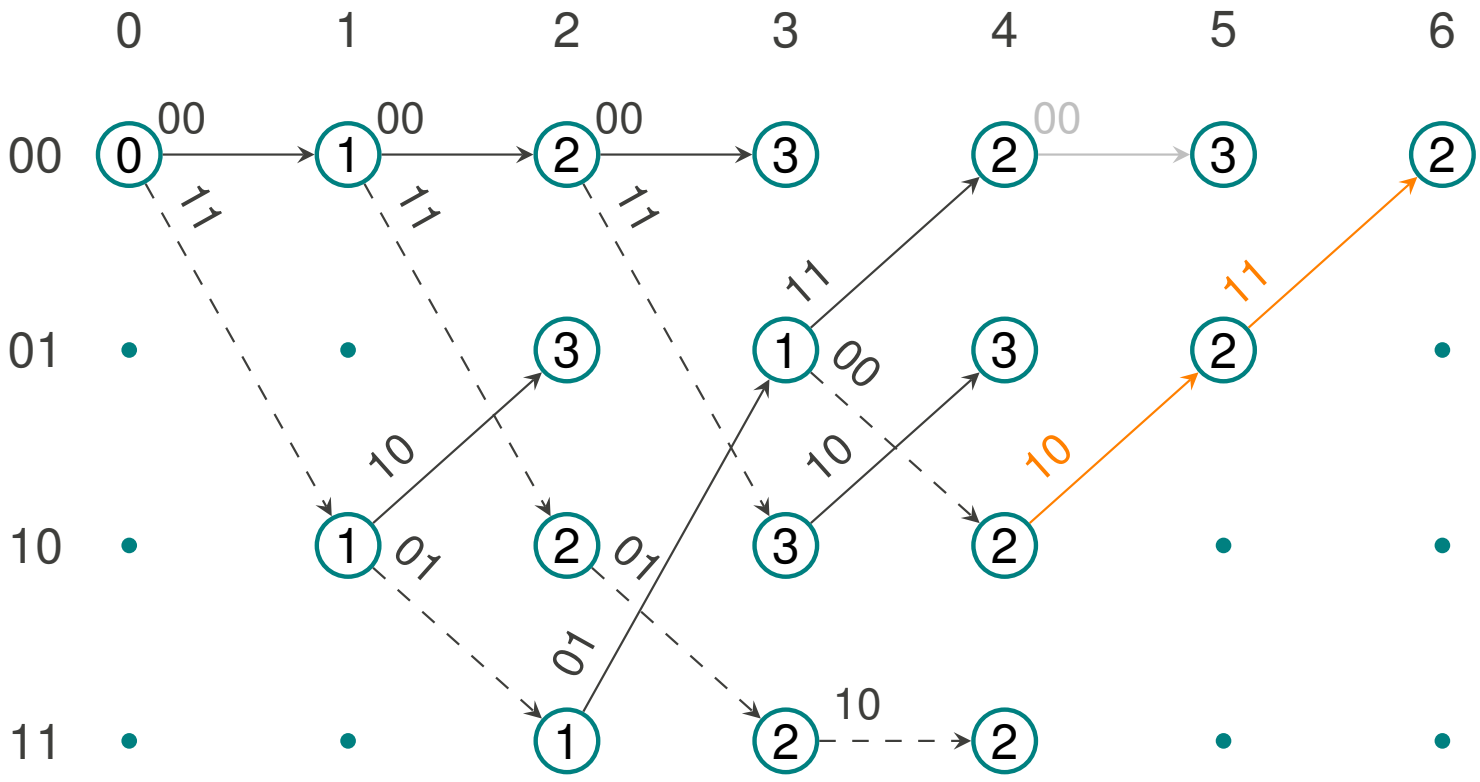
Faltungsdekodierung (Viterbi-Algorithmus)



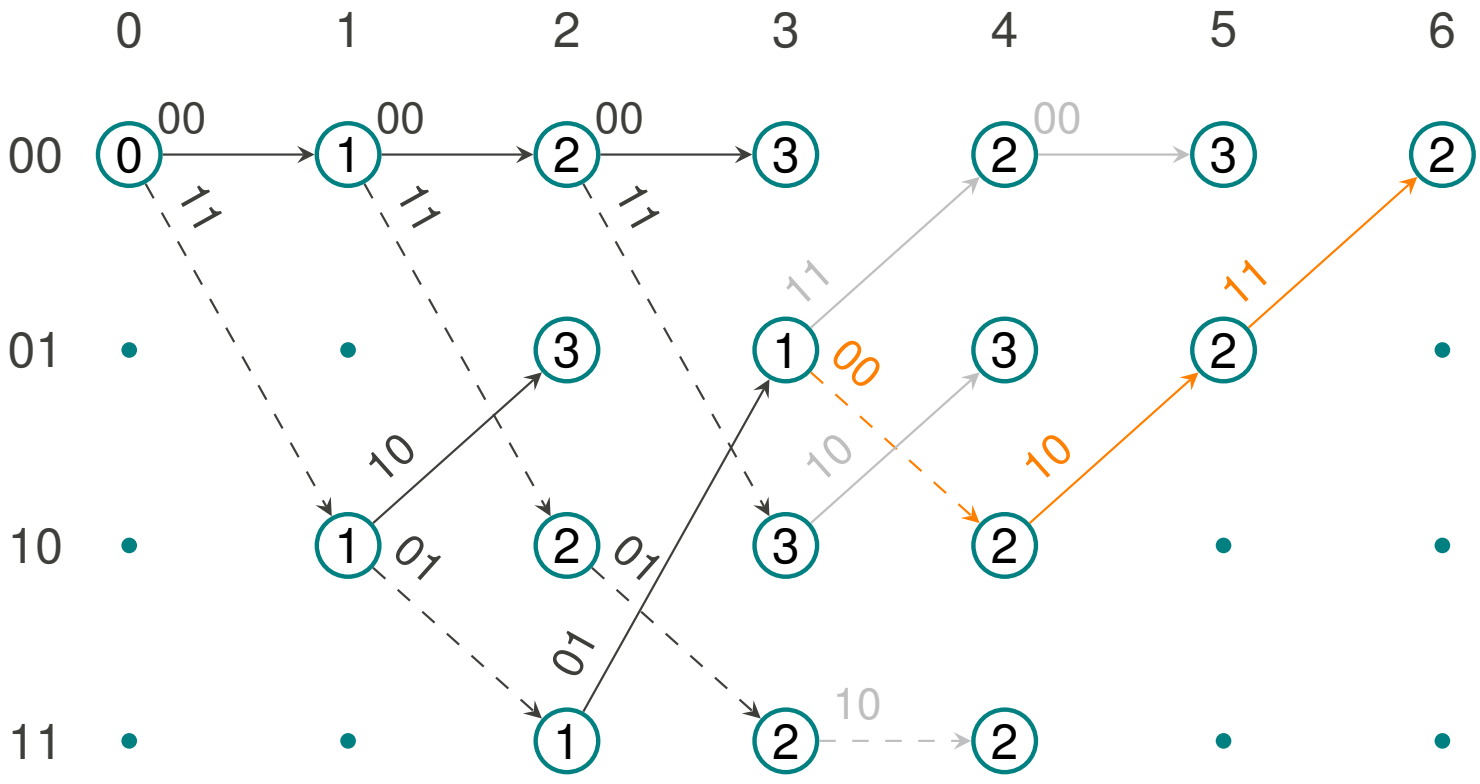
Faltungsdekodierung (Viterbi-Algorithmus)



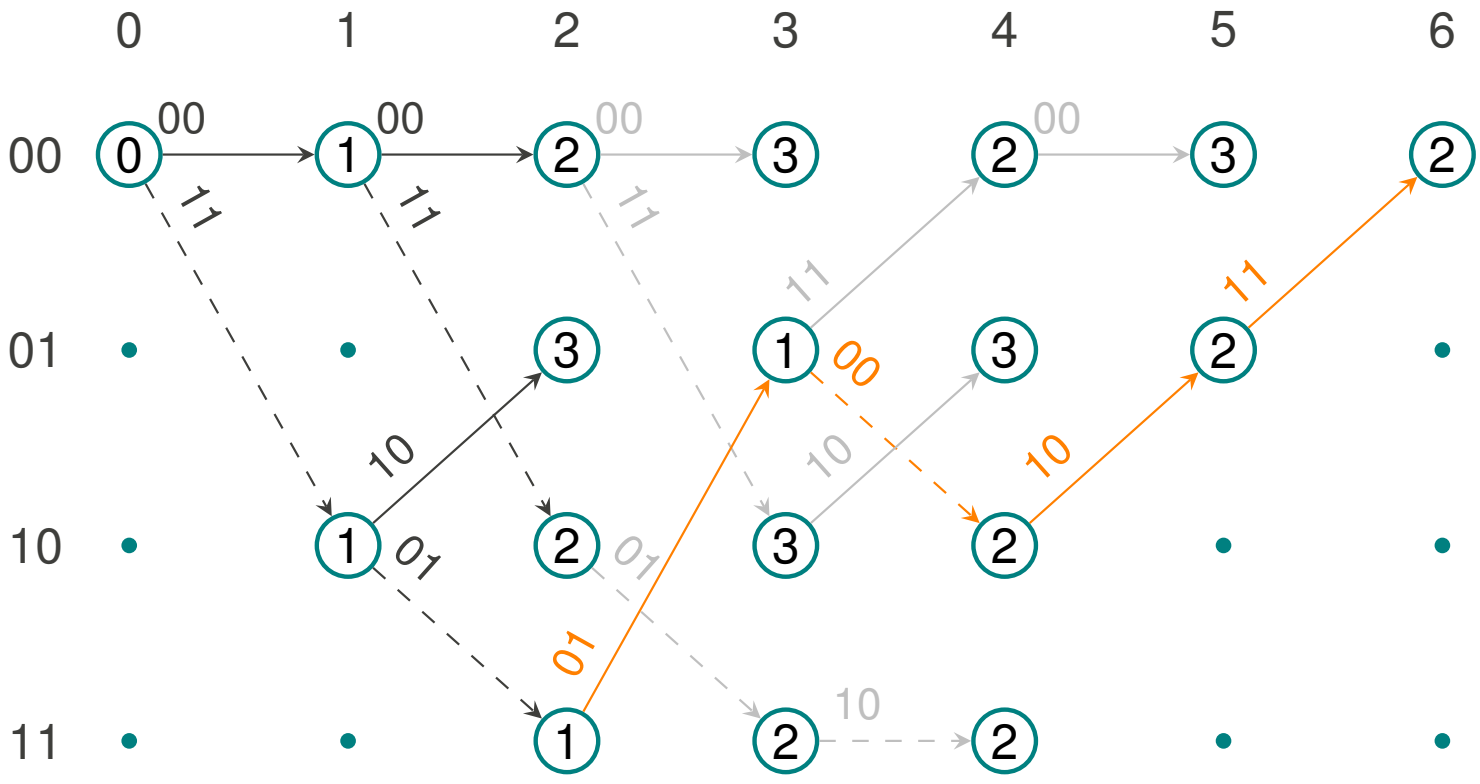
Faltungsdekodierung (Viterbi-Algorithmus)



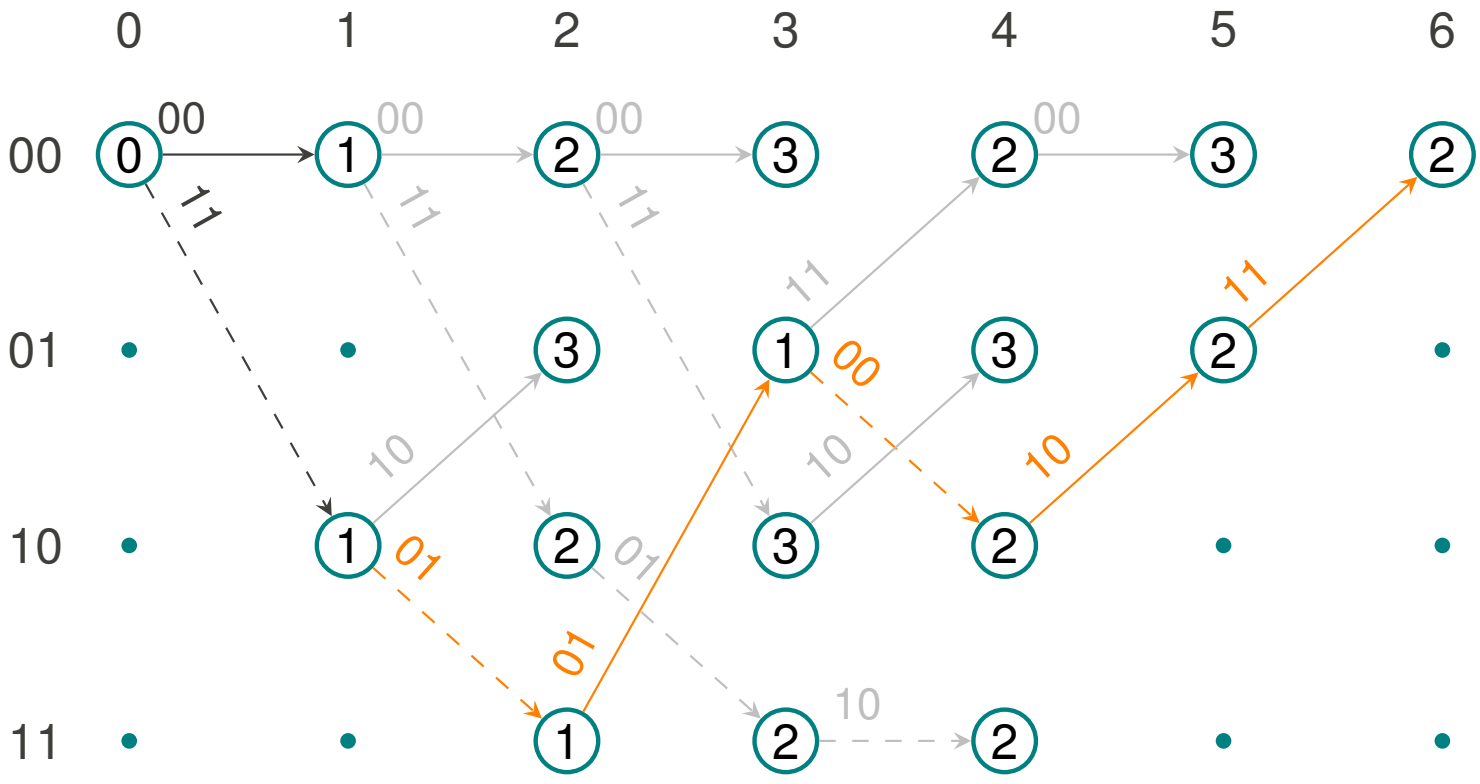
Faltungsdekodierung (Viterbi-Algorithmus)



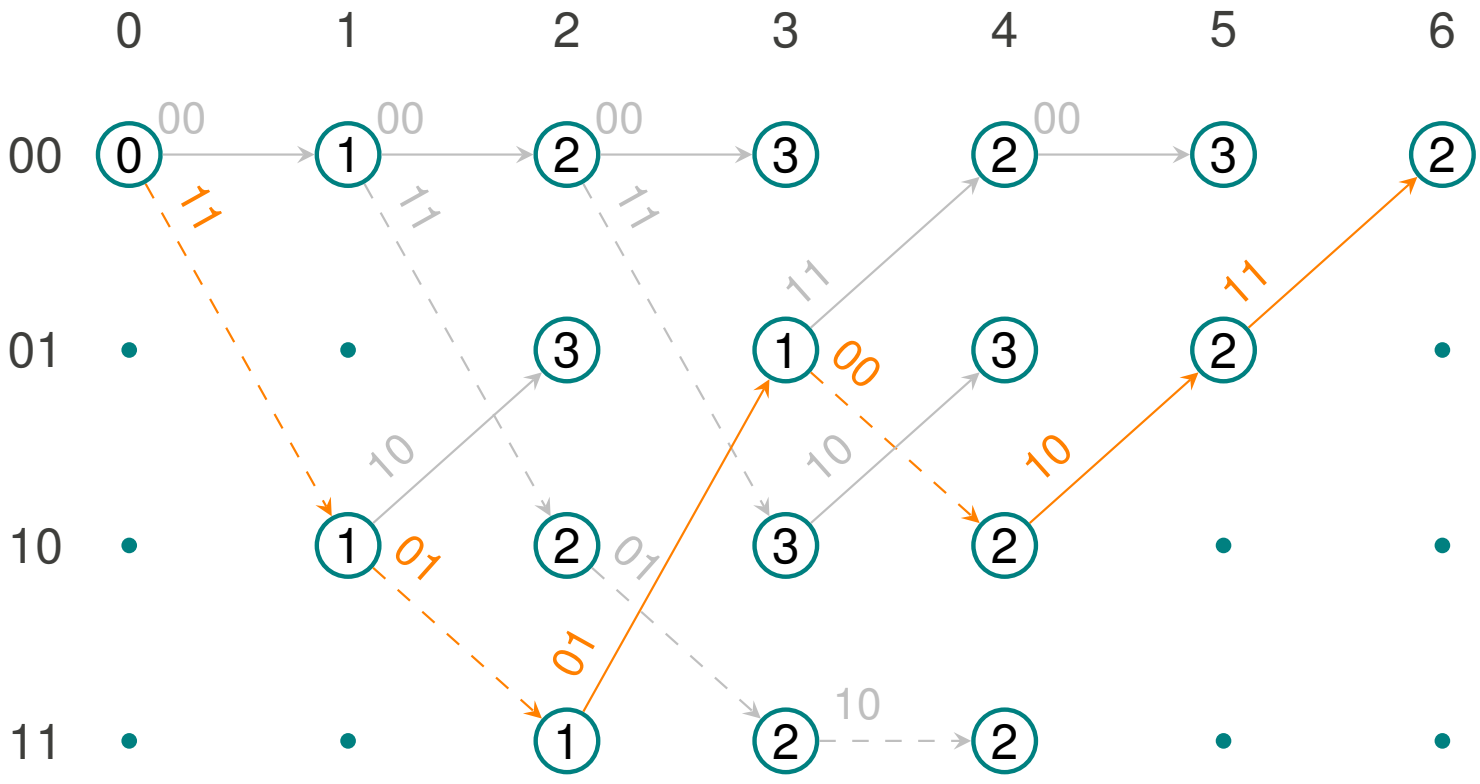
Faltungsdekodierung (Viterbi-Algorithmus)



Faltungsdekodierung (Viterbi-Algorithmus)



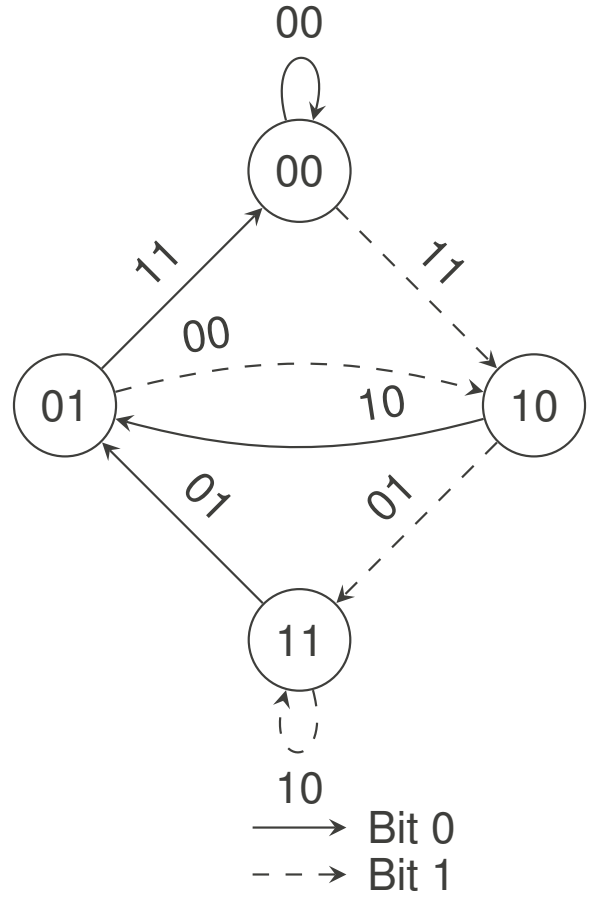
Faltungsdekodierung (Viterbi-Algorithmus)



Faltungskodierung

Nachricht: (1, 1, 0, 1, 0, 0)

Input	Zustand	Folgezustand	Output
1	00	10	11
1	10	11	01
0	11	01	01
1	01	10	00
0	10	01	10
0	01	00	11



Kode : (1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1)

Viterbi-Algorithmus

- Varianten
  - hard decision Dekodierung
  - soft decision Dekodierung
- Dekodierung aufwändig → Flaschenhals

Viterbi-Algorithmus

- Varianten
  - hard decision Dekodierung
  - soft decision Dekodierung
- Dekodierung aufwändig → Flaschenhals

R vs. C/C++

- R-Code wird interpretiert → langsam
- Flaschenhälse: Schleifen, rekursive Funktionen
- Performance mittels C/C++-Code verbessern (kompiliert)
- Rcpp-Paket

Rcpp

R-Datentypen

- `NumericVector`
- `IntegerVector`
- `CharacterVector`
- `LogicalVector`
- `NumericMatrix`
- `IntegerMatrix`
- `CharacterMatrix`
- `LogicalMatrix`

R-Datentypen

Beispiel

```
1  IntegerVector replicate(int x, int times)
2  {
3      IntegerVector v(times);
4
5      for (int i = 0; i < times; i++)
6      {
7          v[i] = x;
8      }
9
10     return v;
11 }
```

Rcpp

einfacher Funktionsaufruf

C++-Code: *vectors.cpp*

```
1  #include <Rcpp.h>
2  using namespace Rcpp;
3
4  // [[Rcpp::export]]
5  double vector_mean(NumericVector x) {
6
7     int n = x.size();
8     double total = 0;
9
10    for (int i = 0; i < n; i++) {
11       total += x[i];
12    }
13    return total / n;
14 }
```

Rcpp

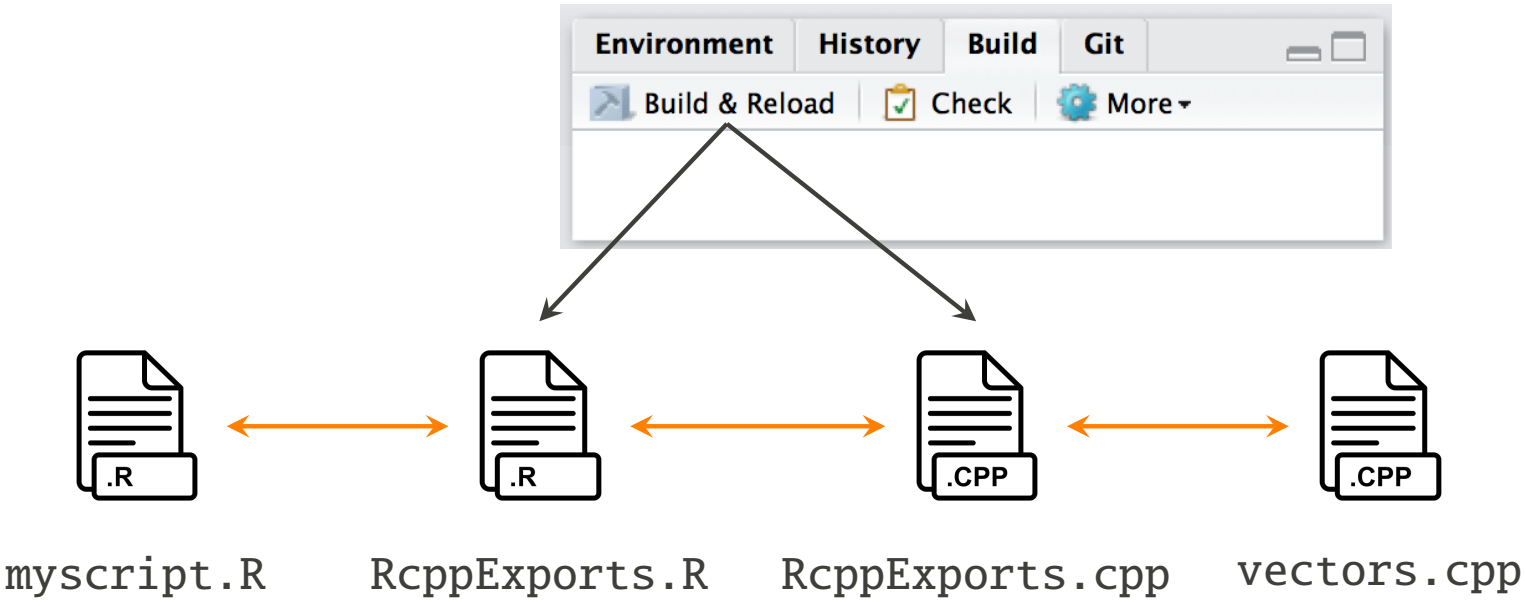
einfacher Funktionsaufruf

R-Code: *myscript.R*

```
1 mean <- vector_mean(c(5,12,9,1,3))
2 mean
3 # 6
```

Rcpp

einfacher Funktionsaufruf



Literatur

- W. C. Huffman und V. Pless. *Fundamentals of Error-Correcting Codes*. 2010
- R. H. Morelos-Zaragoza. *The Art of Error Correcting Coding*. 2006
- D. Schönfeld, H. Klimant und R. Piotraschke. *Informations- und Kodierungstheorie*. 2012
- H. Wickham. *Advanced R*. 2015
- H. Wickham. *R Packages*. 2015