Kanalkodierung

Leistungsnachweis

Rico Pöhland (3299447) Mandy Korzetz (3338750)

Wintersemester 2009/10

1 Faltungskodes - Aufgabe (E)

Gegeben sei folgende Empfangsfolge:

$$
\begin{aligned}
b = (&000111011010011110010111000111011010100001001010111000\\
&000011111001001111001011111101111001111010101100111000\\
&101000001110111111101101000001110100001101001111001000\\
&010001010010000001010001001100100101011111110000100101\\
&0111)
\end{aligned}
$$

Sowie ein Faltungskodierer (siehe Abb.1), der durch folgende Koeffizientenmatrix beschrieben ist:

$$G = (13_8, 15_8, 17_8)^T = \begin{pmatrix} 1011 \\ 1101 \\ 1111 \end{pmatrix}$$

Mittels programmtechnischer Realisierung des VITERBI-Algorithmus soll die Empfangsfolge b dekodiert und anschließend der Quelltext aus der Informationsfolge b^* ermittelt werden. Das Ergebnis soll mit der freien Distanz d_f des Faltungskodes verglichen werden (siehe Vorlesung S.93, Tabelle OFD-Faltungskodes) und Schlussfolgerungen gezogen werden. Weiterhin besteht die Fragestellung ob der Einsatz der Terminierung die Anzahl rekonstruierbarerer Fehler beeinflusst und ob ein Zusammenhang mit Bezug auf die freie Distanz d_f ableitbar ist.

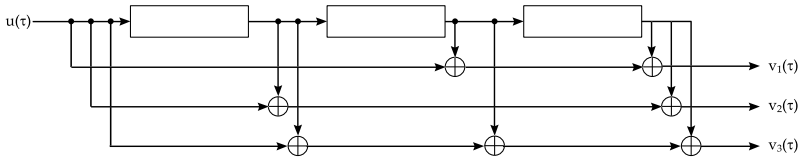


Abbildung 1: Faltungskodierer G

2 Benutzung des Programms

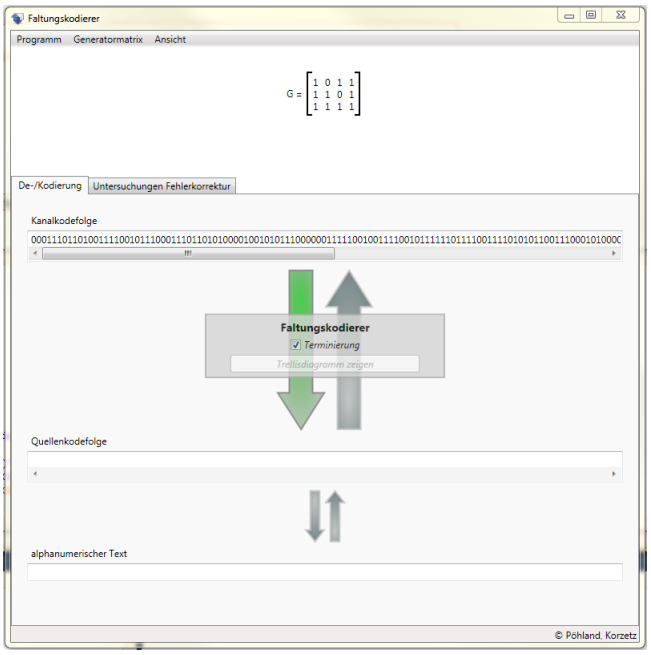


Abbildung 2: Ansicht nach Programmstart

Nach erfolgreichem Start des Programms[1] öffnet sich das Hauptfenster der Anwendung (siehe Abb. 2). Dieses besteht aus einer Menüleiste, einem Anzeigebereich für die momentan gesetzte Generatormatrix des Faltungskodierers, einem Registerkarten-Bereich mit den Hauptfunktionen der Anwendung, sowie einer Statusleiste.

Die Menüleiste besitzt zum einen die typische Funktion zum Beenden einer Anwendung. Desweiterenkann man zwischen den zwei Registerkarten *De-/Kodierung* und *Untersuchungen Fehlerkorrektur* im Hauptbereich des Fenster wechseln. Zu finden ist diese Option unter *Ansicht*. Unter dem Menüpunkt *Generatormatrix* der Menüleiste besteht die Möglichkeit eine beliebige Generatormatrix selbst einzugeben (*Generatormatrix* > *Bearbeiten*). Es öffnet sich ein neues Fenster (Abb. 3), in dem man zum einem die Größe der Generatormatrix verändern kann (maximale Zeilen- und Spaltenanzahl liegt bei 15) und zum anderen natürlich auch die einzelnen Werte $\in \{0, 1\}$ der Matrix setzen kann. Zum endgültigen Setzen der Generatormatrix muss noch der Button *Übernehmen* gedrückt werden.

Fehler, die bei der Benutzung auftreten, werden in der Statusleiste ausgegeben.

1 WPF-Anwendung entstanden unter .NET 3.5 SP1

Abbildung 3: beliebige Generatormatrix setzen

2.1 De-/Kodierung

Über die Registerkarte *De-/Kodierung* erreichen Sie die Kernfunktionalität der Anwendung. Diese Registerkarte wird beim Start geöffnet (siehe Abb. 2). Das Dekodieren einer Kanalkodefolge (standardmäßig ist hier die gegebene Kanalkodefolge b eingetragen) erfolgt über den Pfeilbutton in Richtung Quellenkodefolge, umgekehrt erfolgt das Kodieren einer Quellenkodefolge über den Pfeilbutton in Richtung der Kanalkodefolge. Möchte man Kodieren oder Dekodieren ohne Terminierung, so entfernt man das Häkchen bei dieser Option. Betätigt man den Button *Trellis anzeigen* öffnet sich ein Fenster (Abb. 4) mit dem zu Grunde liegenden Trellisdiagramm des zuvor durchgeführten Dekodiervorgangs. Wie aus der Vorlesung gewohnt, bedeutet eine gestrichelte Linie eine *0* und eine durchgezogene Linie eine *1*. Rotgefärbte Linien stehen für die dekodierte Folge mittels des VITERBI-Algorithmus[1].

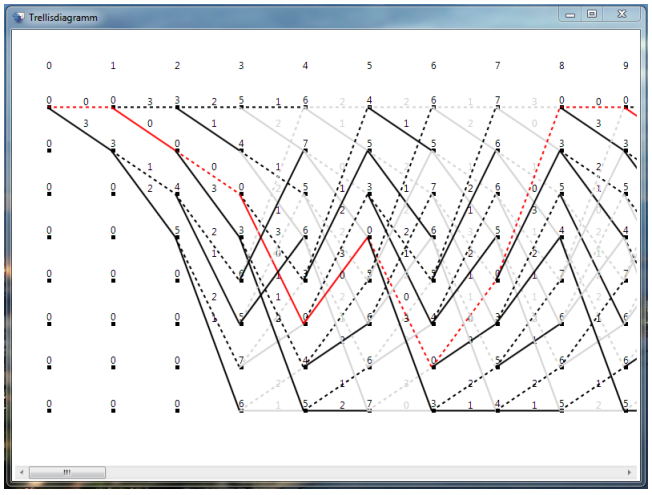


Abbildung 4: Fenster mit dem Trellisdiagramm

Um die Quellenkodefolge für den Benutzer lesbar zu machen kann man die kleineren Pfeilbutton zwischen der Quellenkodefolge und dem alphanumerischem Text

1 Hierbei ist zu erwähnen, dass MD-Dekodierung angewandt wurde und bei gleichen Metrikwerten der jeweils von "unten kommende" Pfad erhalten bleibt.

benutzen. Hierbei wird die Binärzeichenfolge des Quellenkodewortes mittels einfacher 5-Bit-Kodierung in eine Buchstabenfolge (A,...,Z,␣) überführt bzw. umgekehrt.

2.2 Untersuchungen zur Fehlerkorrektur

Die Ansicht der Registerkarte *Untersuchung Fehlerkorrektur* sehen Sie in Abbildung 5.

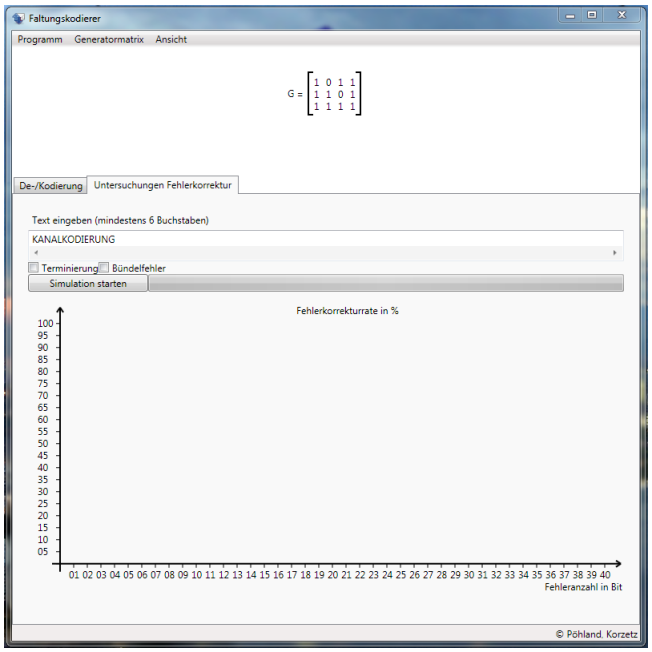


Abbildung 5: Ansicht der Registerkarte *Untersuchung Fehlerkorrektur*

Mit dem Button *Simulation starten* starten Sie 100 Dekodiervorgänge mit der in der Anwendung gesetzten Generatormatrix. Grundlage ist der eingegebene Text, der standardmäßig *KANALKODIERUNG* lautet. Am Anfang eines jeden Dekodiervorganges steht intern das Kodieren der Texteingabe. In diese kodierte Folge werden nun nacheinander zufällig 1 Bit-, 2 Bit-,... und 40 Bit-Fehler eingebaut. Bei ausgewählter *Bündelfehler*-Option handelt es sich dabei um zusammenhängende Bitfehler. Ist diese Option deaktiviert, werden die Bitfehler zufällig[1] bestimmt. Diese kodierte und jeweils fehlerhafte Bitfolge wird nun dekodiert und anschließend verglichen, ob der Faltungskodierer in der Lage war, die Bitfolge vollständig zu rekonstruieren. Diese Ergebnisse werden anschließend grafisch in einem Diagramm dargestellt und könnten beispielsweise wie in Abbildung 6 aussehen. An der Abszisse werden die Anzahl der Bitfehler abgetragen. Der zugehörige Ordinatenwert enthält die erfolgreichen Korrekturen in Prozent. Auch in dieser Registerkarte hat man die Möglichkeit den Faltungskodierer mit und ohne Terminierung zu benutzen.

1 Der Zufall kann hierbei auch vereinzelt Bündelfehler entstehen lassen.

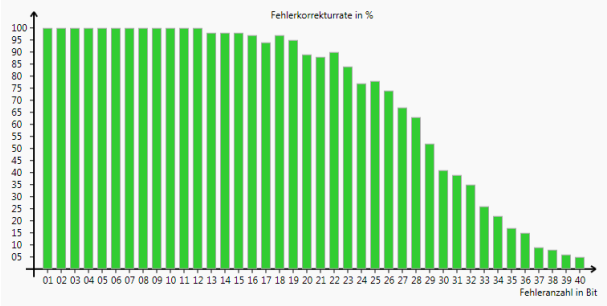


Abbildung 6: Beispiel eines Diagramms zur Fehlerkorrekturuntersuchung

3 Ergebnisse

Gegeben war der Faltungskodierer

$$G = (13_8, 15_8, 17_8)^T = \begin{pmatrix} 1011 \\ 1101 \\ 1111 \end{pmatrix}$$

sowie die Kanalkodefolge b (siehe Seite 1).

Der Faltungskodierer G lässt sich auch als einen determinierten Automaten (dA) mit $k = 3$ (sog. *Gedächtnis*) Zuständen und $m = 3$ Ausgängen beschreiben.

Beschreibung in dA-Notation:

$$\begin{aligned} U &= \{0,1\} \\ V &= \{0,1\}^3 \\ Z &= \{0,1\}^3 \\ z(0) &= (z_1(0), z_2(0), z_3(0)) \\ &= (0,0,0) \end{aligned}$$

$$\begin{aligned} R: v_1(\tau) &= u(\tau) \oplus z_2(\tau) \oplus z_3(\tau) \\ v_2(\tau) &= u(\tau) \oplus z_1(\tau) \oplus z_3(\tau) \\ v_3(\tau) &= u(\tau) \oplus z_1(\tau) \oplus z_2(\tau) \oplus z_3(\tau) \\ S: z_1(\tau+1) &= u(\tau) \\ z_2(\tau+1) &= z_1(\tau) \\ z_3(\tau+1) &= z_2(\tau) \end{aligned}$$

Wobei $u \in U$ Eingangsvariable $\in$ Eingabealphabet, $v \in V$ Ausgangsvariable $\in$ Ausgabealphabet, $z \in Z$ Zustandsvariable $\in$ Zustandsmenge, $R : U \times Z \to V$ Ergebnisabbildung, $S : U \times Z \to Z$ Folgezustandsabbildung und $z(0) \in Z$ Anfangszustand.

Die Beschreibung der Verhaltensweise des Automaten erfolgt durch einen Zustandsgraphen (siehe Abb.7) oder einer Zustandsübergangstabelle (siehe Tab.1).

Für weitere Dekodierungsbetrachtungen ist dies jedoch nicht ausreichend. Es wird eine Darstellungsart benötigt, die die Zusammenhänge in zeitlicher Entwicklung darstellt. Hierfür eignen sich die aus der Vorlesung *Kanalkodierung* bekannten Trellisdiagramme. Diese ermöglichen eine taktweise Betrachtung. Das verkürzte Trellis ist in Abbildung 8 zu sehen, es wurde aus dem Zustandsgraphen (Abb. 7) abgeleitet.

$u(\tau)$	$z(\tau)$ $= (z_1(\tau), z_2(\tau), z_3(\tau))$	$z(\tau+1)$ $= (z_1(\tau+1), z_2(\tau+1), z_3(\tau+1))$	$v(\tau)$ $= (v_1(\tau), v_2(\tau), v_3(\tau))$
0	0,0,0	0,0,0	0,0,0
1	0,0,0	1,0,0	1,1,1
0	1,0,0	0,1,0	0,1,1
1	1,0,0	1,1,0	1,0,0
0	0,1,0	0,0,1	1,0,1
1	0,1,0	1,0,1	0,1,0
0	1,1,0	0,1,1	1,1,0
1	1,1,0	1,1,1	0,0,1
0	0,0,1	0,0,0	1,1,1
1	0,0,1	1,0,0	0,0,0
0	1,0,1	0,1,0	1,0,0
1	1,0,1	1,1,0	0,1,1
0	0,1,1	0,0,1	0,1,0
1	0,1,1	1,0,1	1,0,1
0	1,1,1	0,1,1	0,0,1
1	1,1,1	1,1,1	1,1,0

Tabelle 1: Zustandsübergangstabelle

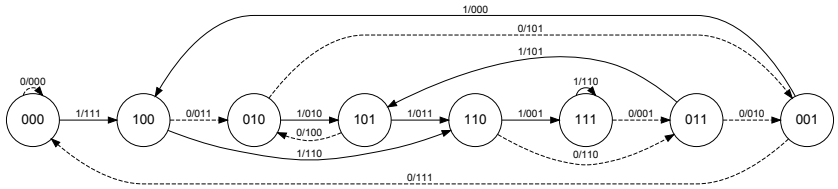


Abbildung 7: Zustandsgraph

Nach VITERBI wird zu jedem Zeitpunkt τ die Empfangssequenz mit jeder Kodesequenz des verkürzten Trellisdiagramms verglichen. Das bedeutet, es wird eine Teilmetrik berechnet, die sich in unserem Fall - da wir uns für MD-Dekodierung entschieden haben - aus der minimalen HAMMING-Distanz der beiden Sequenzen ergibt. Diese Teilmetrik wird zum Metrikwert des Zustandes im Takt τ addiert und ergibt damit den Metrikwert für den Zustand im Takt $\tau + 1$. Nach $\tau > k$ Takten laufen in jedem Zustand jeweils zwei Übergänge zusammen. Im Normalfall wird die kleinste Metrik weitergeführt und die zugehörige Kante der größeren Metrik kann gestrichen werden. Der Pfad im gesamten Trellis mit der gerinsten Metrik zum Endtaktzeitpunkt ist die gesuchte ggf. rekonstruierte Quellenkodefolge. Jedoch kann es auch vorkommen, dass die Metriken den gleichen Wert besitzen. In unserer Anwendung führen wir jeweils

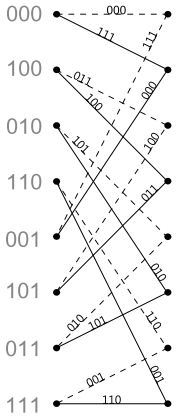


Abbildung 8: verkürztes Trellisdiagramm

den unteren Pfad fort und kommen zu einer rekonstruierten Empfangsfolge

$$\begin{aligned} b =&(000111011010011110010111000111011010\mathbf{011}00100101011100 \\ &0000111\mathbf{0}1\mathbf{1}0100111100101111111011\mathbf{0}10011110101011001110000 \\ &1010000011101111111011\mathbf{101}000011101000011010011110010000 \\ &01000\mathbf{010}0\mathbf{1}10\mathbf{11}000101000100110010010101111111000010010 1 \\ &0111) \end{aligned}$$

und damit zu einer geschätzten Quellenkodefolge

$$\begin{aligned} b^* =&(010110000101110000010110001010110111100100 \\ &0100100101100111010101110000111000) \\ =&KANALKODIESUNG \end{aligned}$$

Zum Vergleich kommt man auf ein anderes Ergebnis, wenn man den jeweils oberen Pfad bei gleichen Metriken beibehält. Durch händisches Dekodieren[1] ergibt sich in dem Fall eine rekonstruierte Empfangsfolge

$$\begin{aligned} b =&(000111011010011110010111000111011010\mathbf{011}00100101011100 \\ &0000111\mathbf{0}1\mathbf{1}0100111100101111111011\mathbf{0}10011110101011001110000 \\ &1010000011101111111011\mathbf{101}000011101000011010011110010000 \\ &0100\mathbf{1}10\mathbf{01}0\mathbf{0}000\mathbf{11}0101000100110010010101111111000010010 1 \\ &0111) \end{aligned}$$

1 Das zugehörige Trellis mit gekennzeichnetem VITERBI ist dem Anhang beigefügt.

und daher eine geschätzte Quellenkodefolge

$$\begin{aligned} b^* =&(0101100001011100000101100010110111100100 \\ &01001001011001100101011000111000) \\ =&KANALKODIESENG \end{aligned}$$

Erwartet wurde vermutlich die Quellenkodefolge $KANALKODIERUNG$, d.h. 1 Bit konnte vom Faltungskodierer nicht richtig korrigiert werden.

Aus der OFD-Tabelle der Vorlesungsunterlagen ist ableitbar, dass vorliegender Faltungskodierer eine freie Distanz d_f von 10 aufweist. Auch für Faltungskodes gilt der bekannte Ansatz für den Fehlerkorrekturgrad f_k:

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

d.h. für unseren vorliegenden Faltungskodierer ergibt sich ein f_k von 4. Die Leistungsfähigkeit hinsichtlich korrigierbarer Fehler kann aber auch deutlich über f_k liegen - je nach Fehlermusterstruktur. Bei "guter" Fehlermusterstruktur kann man davon ausgehen, dass die Anzahl korrigierbarer Fehler bei ungefähr d_f liegt. "Gute" Fehlermusterstruktur bedeutet, dass möglichst keine Bündelfehler vorkommen - im besten Fall also ausschließlich Einzelbitfehler. Anhand des Trellisdiagramms (siehe Anwendung) ist erkennbar, dass unsere gegebene Empfangsfolge b 15 Bitfehler aufweist. Es ergibt sich eine Fehlermuster

$$\begin{aligned} e =&(0000000000000000000000000000000000011100000000000000 \\ &0000000101000000000000000000000010000000000000000000000 \\ &0000000000000000000000111000000000000000000000000000000 \\ &0111000000110100000000000000000000000000000000000000000 \\ &0000) \end{aligned}$$

Damit liegt die Fehleranzahl $w(e) = 15$ deutlich über der Fehlerkorrekturrate von $f_k = 4$ und auch über der freien Distanz von $d_f = 10$. Demnach ist die Wahrscheinlichkeit, dass der Faltungskodierer die Empfangsfolge vollständig korrekt rekonstruieren kann, gering. Und wie wir nach der Rekonstruktion sehen konnten, erfolgte tatsächlich keine Korrektur auf die Quellenkodefolge $KANALKODIERUNG$.

Verzichtet man in der vorliegenden Konstellation auf Terminierung ergibt sich das gleiche Rekonstruktionsergebnis. In Bezug auf den VITERBI-Algorithmus bestimmt Terminierung eindeutig welcher Pfad im Trellisdiagramm der möglichen Kanalkodefolge am ähnlichsten ist, da exakt ein Pfad übrig bleibt. Wendet man keine Terminierung an, bleiben mehrere - genauer gesagt 2^k - nicht verworfene Pfade übrig. Selbstverständlich sollte dieser gewählt werden, welcher den kleinsten Metrikwert[1] enthält. Enthalten mehrere Pfade die kleinste Metrik, müsste zufällig entscheiden werden, welcher Pfad als Rekonstruktionsergebnis gewählt wird. Das stellt natürlich eine zusätzliche

1 bei MD-Dekodierung

Fehlerquelle dar, weswegen Terminierung hinsichtlich der besseren Rekonstruktionsergebnisse im Allgemeinen vorzuziehen ist. Mit den vorliegenden Größen ergibt sich im Takt 70, dem letzten Takt vor der Terminierungssequenz, des VITERBI-Algorithmus, genau ein solcher Pfad mit einer minimalen Metrik von 15. Da desweiteren keine Fehler in den Terminierungsnullen aufgetreten sind, entspricht dies folglich dem gleichen Rekonstruktionsergebnis wie dem mit dem Einsatz von Terminierung.

Bleibt noch die Frage offen, inwiefern der Einsatz von Terminierung die Anzahl rekonstruierbarer Fehler beeinflusst. Die Abbildungen 9-12 sollen darüber Aufschluss geben. Abbildung 9 zeigt die Verteilung der Fehlerkorrekturraten im Bereich von 1-40

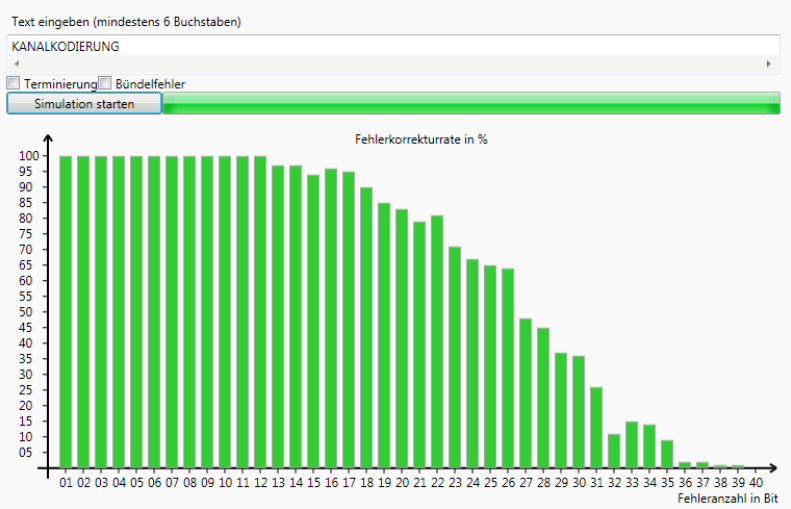


Abbildung 9: Fehlerkorrekturrate mit zufallsverteilten Einzelbitfehlern ohne Einsatz von Terminierung

Einzelbitfehlern des vorliegenden Faltungskodierers ohne den Einsatz von Terminierung. Es wird deutlich, dass der Faltungskodierer (mindestens) bis $d_f = 10$ Einzelbitfehler vollständig rekonstruieren kann. Nimmt man Terminierung hinzu (Abb. 10), kann er sogar weit über die Anzahl d_f hinaus Einzelbitfehler rekonstruieren.

Bei Bündelfehlern verhält sich das anders. Wie man der Abbildung 11 entnehmen kann, ist der Faltungskodierer zumindest in der Lage $f_k = 4$ Bit-Bündelfehler vollständig zu rekonstruieren. Führt man die Analyse ohne Terminierung aus (Abb. 12), erkennt man ein starkes Leistungsdefizit. Die Länge der rekonstruierbaren Bündelfehler bewegt sich im Bereich von $\left\lfloor \frac{f_k}{2} \right\rfloor$.

4 Fazit

Bündelfehler stellen für Faltungskodierer ein großes Problem bei der Dekodierung einer Empfangsfolge dar. Da sich diese nur in den seltensten Fällen vermeiden lassen, sollte nicht auf Terminierung verzichtet werden, denn nur mit Terminierung kann eine

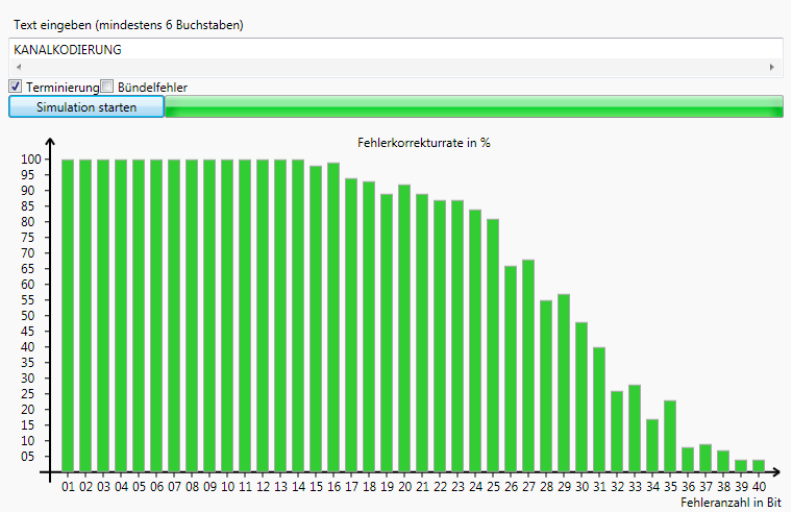


Abbildung 10: Fehlerkorrekturrate mit zufallsverteilten Einzelbitfehlern unter Einsatz von Terminierung

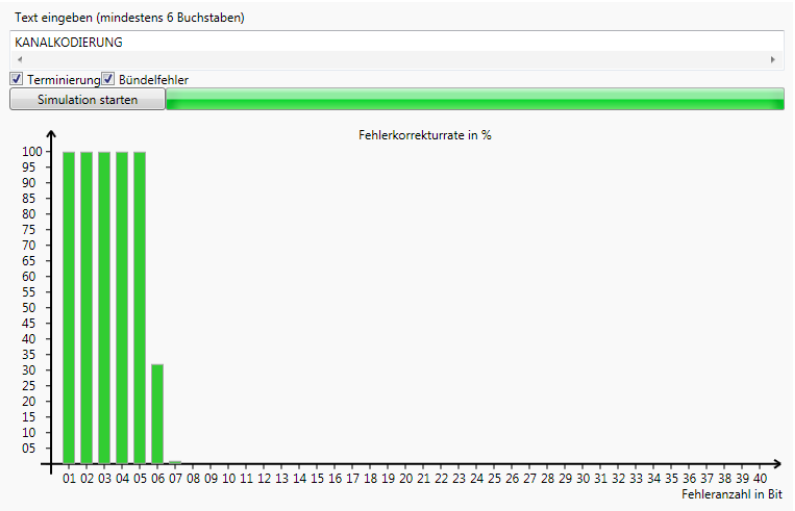


Abbildung 11: Fehlerkorrekturrate mit Bündelfehlern unter Einsatz von Terminierung

Leistungsfähigkeit bzgl. der Bitfehleranzahl von mindestens f_k eines Faltungskodierers gesichert werden. Natürlich muss man bei Terminierung mit einem Koderatenverlust rechnen, da zur Quellkodefolge jeweils k Bit Nullen angehängt werden müssen. Weitere Techniken zur Verbesserung der Leistungsfähigkeit des Faltungskodierers in Bezug auf Bündelfehler wären Interleaving und/oder Kodeverkettung, sollen aber an dieser Stelle nicht weiter ausgeführt werden.

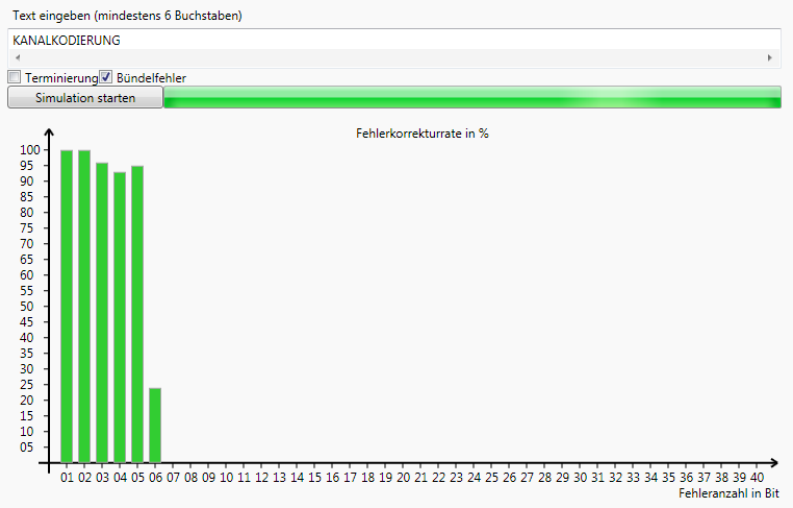


Abbildung 12: Fehlerkorrekturrate mit Bündelfehlern ohne Einsatz von Terminierung

Literatur

[KPS06] KLIMANT, Herbert ; PIOTRASCHKE, Rudi ; SCHÖNFELD, Dagmar: *Informations- und Kodierungstheorie*. Teubner Verlag, 2006. – ISBN 3–8351–0042–4

[Sch10] SCHÖNFELD, Dagmar: *Lehrmaterial Kanalkodierung*, 2009/2010

Anhang

Quellcode

Listing 1: App.xaml

```
ï»¿<Application x:Class="Faltungskodierer.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    StartupUri="Window1.xaml">

    <Application.Resources>
         <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary
                  Source="ResDictionary.xaml">
                </ResourceDictionary>
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Application.Resources>

</Application>
```

Listing 2: App.xaml.cs

```
ï»¿using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Windows;

namespace Faltungskodierer
{
    /// <summary>
    /// Interaktionslogik fÃ¼r "App.xaml"
    /// </summary>


```

```
15      public partial class App : Application
16      {
17          public ConvEncoder encoder = new ConvEncoder();
18          public Trellis trellis;
19          public Viterbi viterbi;
20      }
21  }
```

Listing 3: Window1.xaml

```
1   <Window x:Class="Faltungskodierer.Window1"
2       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4       xmlns:local="clr-namespace:Faltungskodierer"
5       Title="Faltungskodierer" Height="800" Width="800">
6
7       <DockPanel>
8           <Menu DockPanel.Dock="Top">
9               <MenuItem Header="Programm">
10                  <MenuItem Name="mClose" Header="Schließen" Click="mClose_Click" />
11              </MenuItem>
12              <MenuItem Header="Generatormatrix">
13                  <MenuItem Name="mEdit" Header="Bearbeiten" Click="btnChangeMatrix_Click"/>
14                  <MenuItem Header="Standardmatrix wählen">
15                      <MenuItem Name="mG1" Header="(13&#x2088;, 15&#x2088;, 17&#x2088;)" Click="mG1_Click" />
16                      <MenuItem Name="mG2" Header="(5&#x2088;, 7&#x2088;)" Click="mG2_Click" />
17                      <MenuItem Name="mG3" Header="(7&#x2088;, 4&#x2088;, 5&#x2088;)" Click="mG3_Click" />
18                  </MenuItem>
19              </MenuItem>
20              <MenuItem Header="Ansicht">
21                  <MenuItem  Name="mTab1" Header="De-/Kodierung" Click="mTab1_Click" />
22                  <MenuItem Name="mTab2" Header="Untersuchungen Fehlerkorrektur" Click="mTab2_Click" />
23              </MenuItem>
24          </Menu>
25
26          <StatusBar DockPanel.Dock="Bottom" Height="20">
27              <StatusBarItem Name="sbText"></StatusBarItem>
28              <StatusBarItem HorizontalContentAlignment="Right" Content="&#x00A9; Pöhland, Korzetz"/>
29          </StatusBar>
30
31          <ScrollViewer Height="160" DockPanel.Dock="Top" VerticalScrollBarVisibility="Auto">
32              <StackPanel>
33                  <Canvas Name="caMatrix" Margin="15"/>
34              </StackPanel>
35          </ScrollViewer>
36
37          <TabControl>
38              <TabItem Name="tab1" Header="De-/Kodierung">
39                  <Frame Name="frMainTab" NavigationUIVisibility="Hidden" Source="MainTab.xaml"></Frame>
40              </TabItem>
41              <TabItem Name="tab2" Header="Untersuchungen Fehlerkorrektur">
42                  <Frame NavigationUIVisibility="Hidden" Source="ConvStudy.xaml"></Frame>
43              </TabItem>
44          </TabControl>
45      </DockPanel>
46  </Window>
```

Listing 4: Window1.xaml.cs

```
1   ï»¿using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Windows;
6   using System.Windows.Controls;
7   using System.Windows.Data;
8   using System.Windows.Documents;
9   using System.Windows.Input;
10  using System.Windows.Media;
11  using System.Windows.Media.Imaging;
12  using System.Windows.Navigation;
13  using System.Windows.Shapes;
14  using System.Collections;
15
16  namespace Faltungskodierer
17  {
18      /// <summary>
19      /// Interaktionslogik fÃ¼r Window1.xaml
20      /// </summary>
21      public partial class Window1 : Window
```

```
22      {
23          int[,] generatorMatrix;
24          App a;
25
26          public Window1()
27          {
28              InitializeComponent();
29
30              //generatormatrix anlegen
31              generatorMatrix = new int[3, 4];
32              generatorMatrix[0, 0] = 1;
33              generatorMatrix[0, 1] = 0;
34              generatorMatrix[0, 2] = 1;
35              generatorMatrix[0, 3] = 1;
36
37              generatorMatrix[1, 0] = 1;
38              generatorMatrix[1, 1] = 1;
39              generatorMatrix[1, 2] = 0;
40              generatorMatrix[1, 3] = 1;
41
42              generatorMatrix[2, 0] = 1;
43              generatorMatrix[2, 1] = 1;
44              generatorMatrix[2, 2] = 1;
45              generatorMatrix[2, 3] = 1;
46
47
48              a = (App)App.Current;
49              a.encoder.generatorMatrix = generatorMatrix;
50              drawMatrix();
51
52          }
53
54          private void btnChangeMatrix_Click(object sender, RoutedEventArgs e)
55          {
56              MatrixWindow mWindow = new MatrixWindow();
57              mWindow.ShowDialog();
58              drawMatrix();
59          }
60
61          public void drawMatrix()
62          {
63              caMatrix.Children.Clear();
64              int height = a.encoder.generatorMatrix.GetLength(0) * 15 + 30;
65              int width = a.encoder.generatorMatrix.GetLength(1) * 15 + 50;
66              caMatrix.Height = height;
67              caMatrix.Width = width;
68
69              Label g = new Label();
70              g.Content = "G = ";
71              caMatrix.Children.Add(g);
72              Canvas.SetLeft(g, 2);
73              Canvas.SetTop(g, height/2 -8);
74
75              Line left = new Line();
76              left.X1 = 33;
77              left.Y1 = 15;
78              left.X2 = 33;
79              left.Y2 = height - 5;
80              left.StrokeThickness = 2;
81              left.Stroke = Brushes.Black;
82              caMatrix.Children.Add(left);
83
84              left = new Line();
85              left.X1 = 33;
86              left.Y1 = 16;
87              left.X2 = 39;
88              left.Y2 = 16;
89              left.StrokeThickness = 2;
90              left.Stroke = Brushes.Black;
91              caMatrix.Children.Add(left);
92
93              left = new Line();
94              left.X1 = 33;
95              left.Y1 = height - 6;
96              left.X2 = 39;
97              left.Y2 = height - 6;
98              left.StrokeThickness = 2;
99              left.Stroke = Brushes.Black;
100             caMatrix.Children.Add(left);
101
102             Line right = new Line();
103             right.X1 = width - 12;
104             right.Y1 = 15;
105             right.X2 = width - 12;
```

```
106             right.Y2 = height - 5;
107             right.StrokeThickness = 2;
108             right.Stroke = Brushes.Black;
109             caMatrix.Children.Add(right);
110 
111             right = new Line();
112             right.X1 = width - 12;
113             right.Y1 = 16;
114             right.X2 = width - 18;
115             right.Y2 = 16;
116             right.StrokeThickness = 2;
117             right.Stroke = Brushes.Black;
118             caMatrix.Children.Add(right);
119 
120             right = new Line();
121             right.X1 = width - 12;
122             right.Y1 = height - 6;
123             right.X2 = width - 18;
124             right.Y2 = height - 6;
125             right.StrokeThickness = 2;
126             right.Stroke = Brushes.Black;
127             caMatrix.Children.Add(right);
128 
129             for (int i = 0; i < a.encoder.generatorMatrix.GetLength(0); i++)
130             {
131                 for (int j = 0; j < a.encoder.generatorMatrix.GetLength(1); j++)
132                 {
133                     Label t = new Label();
134                     t.Name = "m" + i.ToString() + j.ToString();
135                     t.Content = a.encoder.generatorMatrix[i, j].ToString();
136                     caMatrix.Children.Add(t);
137                     Canvas.SetLeft(t, j * 15 + 35);
138                     Canvas.SetTop(t, i * 15 + 15);
139                 }
140             }
141 
142         }
143 
144         private void mClose_Click(object sender, RoutedEventArgs e)
145         {
146             this.Close();
147         }
148 
149         private void mTab1_Click(object sender, RoutedEventArgs e)
150         {
151             tab1.IsSelected = true;
152             tab2.IsSelected = false;
153         }
154 
155         private void mTab2_Click(object sender, RoutedEventArgs e)
156         {
157             tab2.IsSelected = true;
158             tab1.IsSelected = false;
159         }
160 
161         private void mG1_Click(object sender, RoutedEventArgs e)
162         {
163             generatorMatrix = new int[3, 4];
164             generatorMatrix[0, 0] = 1;
165             generatorMatrix[0, 1] = 0;
166             generatorMatrix[0, 2] = 1;
167             generatorMatrix[0, 3] = 1;
168 
169             generatorMatrix[1, 0] = 1;
170             generatorMatrix[1, 1] = 1;
171             generatorMatrix[1, 2] = 0;
172             generatorMatrix[1, 3] = 1;
173 
174             generatorMatrix[2, 0] = 1;
175             generatorMatrix[2, 1] = 1;
176             generatorMatrix[2, 2] = 1;
177             generatorMatrix[2, 3] = 1;
178 
179 
180             a = (App)App.Current;
181             a.encoder.generatorMatrix = generatorMatrix;
182             drawMatrix();
183         }
184 
185         private void mG2_Click(object sender, RoutedEventArgs e)
186         {
187             generatorMatrix = new int[2, 3];
188             generatorMatrix[0, 0] = 1;
189             generatorMatrix[0, 1] = 0;
```

```
190             generatorMatrix[0, 2] = 1;
191
192             generatorMatrix[1, 0] = 1;
193             generatorMatrix[1, 1] = 1;
194             generatorMatrix[1, 2] = 1;
195
196             a = (App)App.Current;
197             a.encoder.generatorMatrix = generatorMatrix;
198             drawMatrix();
199         }
200
201         private void mG3_Click(object sender, RoutedEventArgs e)
202         {
203             generatorMatrix = new int[3, 3];
204             generatorMatrix[0, 0] = 1;
205             generatorMatrix[0, 1] = 1;
206             generatorMatrix[0, 2] = 1;
207
208             generatorMatrix[1, 0] = 1;
209             generatorMatrix[1, 1] = 0;
210             generatorMatrix[1, 2] = 0;
211
212             generatorMatrix[2, 0] = 1;
213             generatorMatrix[2, 1] = 0;
214             generatorMatrix[2, 2] = 1;
215
216             a = (App)App.Current;
217             a.encoder.generatorMatrix = generatorMatrix;
218             drawMatrix();
219         }
220
221
222     }
223 }
```

Listing 5: MainTab.xaml

```
1   ï»¿<Page x:Class="Faltungskodierer.MainTab"
2         xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3         xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
5         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
6         mc:Ignorable="d"
7         d:DesignHeight="776" d:DesignWidth="800"
8           Title="MainTab">
9
10      <StackPanel Margin="15">
11          <Label>Kanalkodefolge</Label>
12          <TextBox ScrollViewer.HorizontalScrollBarVisibility="Visible" Name="tbB" TextChanged="tbB_TextChanged"
                PreviewTextInput="tbB_PreviewTextInput">000111011010011110010111000111011010100001001010111000000111110010011110010111111101111000
13          <Grid>
14              <StackPanel Margin="0,10,0,0" HorizontalAlignment="Center" Orientation="Horizontal">
15                  <Button Name="btnDecode" Template="{StaticResource ArrowDown}" HorizontalAlignment="Center"
                        ToolTip="Dekodieren" Click="btnDecode_Click" />
16                  <Button Name="btnEncode" Template="{StaticResource ArrowUp}" HorizontalAlignment="Center"
                        IsEnabled="False" ToolTip="Kodieren" Click="btnEncode_Click" />
17              </StackPanel>
18
19              <StackPanel VerticalAlignment="Center" Width="300">
20                  <Border Name="spBorder" BorderThickness="2" BorderBrush="DarkGray" Background="LightGray"
                        Opacity="0.8">
21                      <StackPanel>
22                          <Label HorizontalAlignment="Center" FontWeight="Bold" FontSize="14">Faltungskodierer</Label>
23                          <CheckBox IsChecked="True" Content="Terminierung" Height="16" Name="cbTerm"
                                HorizontalAlignment="Center" FontStyle="Italic"/>
24                          <Button Margin="30,5" Content="Trellisdiagramm zeigen" Name="btnShowTrellis"
                                Click="btnShowTrellis_Click" FontStyle="Italic" IsEnabled="False" />
25                      </StackPanel>
26                  </Border>
27              </StackPanel>
28
29          </Grid>
30
31          <Label>Quellenkodefolge</Label>
32          <TextBox ScrollViewer.HorizontalScrollBarVisibility="Visible" Name="tbBStar"
                TextChanged="tbBStar_TextChanged" PreviewTextInput="tbBStar_PreviewTextInput"></TextBox>
33          <StackPanel Margin="0,10,0,0" HorizontalAlignment="Center" Orientation="Horizontal">
34              <Button Name="btnToText" Template="{StaticResource ArrowDownS}" HorizontalAlignment="Center"
                    Click="btnToText_Click" IsEnabled="False" />
35              <Button Name="btnToBin" Template="{StaticResource ArrowUpS}" HorizontalAlignment="Center"
                    IsEnabled="False" Click="btnToBin_Click" />
36          </StackPanel>
```

```
37
38          <Label>alphanumerischer Text</Label>
39          <TextBox ScrollViewer.HorizontalScrollBarVisibility="Auto" Name="tbText" TextChanged="tbText_TextChanged"
                CharacterCasing="Upper" PreviewTextInput="tbText_PreviewTextInput"></TextBox>
40      </StackPanel>
41  </Page>
```

Listing 6: MainTab.xaml

```
1   <Page x:Class="Faltungskodierer.MainTab"
2         xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3         xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
5         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
6         mc:Ignorable="d"
7         d:DesignHeight="776" d:DesignWidth="800"
8           Title="MainTab">
9
10      <StackPanel Margin="15">
11          <Label>Kanalkodefolge</Label>
12          <TextBox ScrollViewer.HorizontalScrollBarVisibility="Visible" Name="tbB" TextChanged="tbB_TextChanged"
                PreviewTextInput="tbB_PreviewTextInput">0001110110100111100101110001110110101000010010101110000001111100100111100101111110111100
13          <Grid>
14              <StackPanel Margin="0,10,0,0" HorizontalAlignment="Center" Orientation="Horizontal">
15                  <Button Name="btnDecode" Template="{StaticResource ArrowDown}" HorizontalAlignment="Center"
                        ToolTip="Dekodieren" Click="btnDecode_Click" />
16                  <Button Name="btnEncode" Template="{StaticResource ArrowUp}" HorizontalAlignment="Center"
                        IsEnabled="False" ToolTip="Kodieren" Click="btnEncode_Click" />
17              </StackPanel>
18
19              <StackPanel VerticalAlignment="Center" Width="300">
20                  <Border Name="spBorder" BorderThickness="2" BorderBrush="DarkGray" Background="LightGray"
                        Opacity="0.8">
21                      <StackPanel>
22                          <Label HorizontalAlignment="Center" FontWeight="Bold" FontSize="14">Faltungskodierer</Label>
23                          <CheckBox IsChecked="True" Content="Terminierung" Height="16" Name="cbTerm"
                                HorizontalAlignment="Center" FontStyle="Italic"/>
24                          <Button Margin="30,5" Content="Trellisdiagramm zeigen" Name="btnShowTrellis"
                                Click="btnShowTrellis_Click" FontStyle="Italic" IsEnabled="False" />
25                      </StackPanel>
26                  </Border>
27              </StackPanel>
28
29          </Grid>
30
31          <Label>Quellenkodefolge</Label>
32          <TextBox ScrollViewer.HorizontalScrollBarVisibility="Visible" Name="tbBStar"
                TextChanged="tbBStar_TextChanged" PreviewTextInput="tbBStar_PreviewTextInput"></TextBox>
33          <StackPanel Margin="0,10,0,0" HorizontalAlignment="Center" Orientation="Horizontal">
34              <Button Name="btnToText" Template="{StaticResource ArrowDownS}" HorizontalAlignment="Center"
                    Click="btnToText_Click" IsEnabled="False" />
35              <Button Name="btnToBin" Template="{StaticResource ArrowUpS}" HorizontalAlignment="Center"
                    IsEnabled="False" Click="btnToBin_Click" />
36          </StackPanel>
37
38          <Label>alphanumerischer Text</Label>
39          <TextBox ScrollViewer.HorizontalScrollBarVisibility="Auto" Name="tbText" TextChanged="tbText_TextChanged"
                CharacterCasing="Upper" PreviewTextInput="tbText_PreviewTextInput"></TextBox>
40      </StackPanel>
41  </Page>
```

Listing 7: TrellisWindow.xaml

```
1   ï»¿<Window x:Class="Faltungskodierer.TrellisWindow"
2           xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4           Title="Trellisdiagramm" Width="800" Height="600" ResizeMode="NoResize">
5
6       <ScrollViewer Height="558" Width="776" VerticalAlignment="Center" Name="scrViewer"
            ScrollViewer.VerticalScrollBarVisibility="Auto" ScrollViewer.HorizontalScrollBarVisibility="Auto">
7           <Canvas Name="caTrellis" />
8       </ScrollViewer>
9
10  </Window>
```

Listing 8: TrellisWindow.xaml.cs

```
ï»¿using System;
using System.Collections;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace Faltungskodierer
{
    /// <summary>
    /// Interaction logic for TrellisWindow.xaml
    /// </summary>
    public partial class TrellisWindow : Window
    {
        Trellis trellis;

        public TrellisWindow()
        {
            InitializeComponent();

            App a = (App)App.Current;
            this.trellis = a.trellis;

            caTrellis.Width = trellis.Count * 80 + 40;
            Hashtable t = (Hashtable)trellis[0];

            caTrellis.Height = a.encoder.stateTable.GetLength(0) * 27 + 40;


            //scrViewer.Width = trellis.Count * 80 + 40;
            //scrViewer.Height = t.Count * 27 + 40;

            caTrellis.Children.Clear();
            drawTrellis();
        }

        public void drawTrellis()
        {

            int steps = trellis.Count;
            for (int i = 0; i < steps; i++)
            {
                Label l = new Label();
                l.Content = i.ToString();
                caTrellis.Children.Add(l);
                Canvas.SetLeft(l, i * 80 + 40 - 6);
                Canvas.SetTop(l, -5);
                Hashtable t = (Hashtable)trellis[i];
                foreach (object o in t.Keys)
                {
                    Node n = (Node)t[o];

                    Line p = new Line();
                    p.X1 = n.getX();
                    p.Y1 = n.getY();
                    p.X2 = n.getX() + 5;
                    p.Y2 = n.getY();
                    p.Stroke = Brushes.Black;
                    p.StrokeThickness = 5;
                    caTrellis.Children.Add(p);

                    Label aktV = new Label();
                    aktV.Content = n.getMinValue();
                    caTrellis.Children.Add(aktV);
                    Canvas.SetLeft(aktV, n.getX() - 6);
                    Canvas.SetTop(aktV, n.getY() - 22);

                    if (n.getNextEdge_0() != null)
                    {
                        Line q = new Line();
                        q.X1 = n.getNextEdge_0().getX1();
                        q.Y1 = n.getNextEdge_0().getY1();
                        q.X2 = n.getNextEdge_0().getX2();
                        q.Y2 = n.getNextEdge_0().getY2();
                        Label lv = new Label();
                        if (n.getNextEdge_0().getEnable())
                        {
                            if (n.getNextEdge_0().getPath())
```

```
                                q.Stroke = Brushes.Red;
                            else
                                q.Stroke = Brushes.Black;

                            lv.Foreground = Brushes.Black;
                        }
                        else
                        {
                            q.Stroke = Brushes.LightGray;
                            lv.Foreground = Brushes.LightGray;
                        }

                        q.StrokeThickness = 2;
                        q.StrokeDashArray = new DoubleCollection() { 2.0, 2.0 };
                        caTrellis.Children.Add(q);
                        lv.Content = n.getNextEdge_0().getMatch();
                        caTrellis.Children.Add(lv);
                        int x = (n.getNextEdge_0().getX1() + n.getNextEdge_0().getX2()) / 2;
                        int y = (n.getNextEdge_0().getY1() + n.getNextEdge_0().getY2()) / 2;
                        Canvas.SetLeft(lv, x);
                        Canvas.SetTop(lv, y - 20);

                    }

                    if (n.getNextEdge_1() != null)
                    {
                        Line r = new Line();

                        r.X1 = n.getNextEdge_1().getX1();
                        r.Y1 = n.getNextEdge_1().getY1();
                        r.X2 = n.getNextEdge_1().getX2();
                        r.Y2 = n.getNextEdge_1().getY2();
                        Label lv1 = new Label();
                        if (n.getNextEdge_1().getEnable())
                        {
                            if (n.getNextEdge_1().getPath())
                                r.Stroke = Brushes.Red;
                            else
                                r.Stroke = Brushes.Black;

                            lv1.Foreground = Brushes.Black;
                        }
                        else
                        {
                            r.Stroke = Brushes.LightGray;
                            lv1.Foreground = Brushes.LightGray;
                        }

                        r.StrokeThickness = 2;
                        caTrellis.Children.Add(r);
                        lv1.Content = n.getNextEdge_1().getMatch();
                        caTrellis.Children.Add(lv1);
                        int x = (n.getNextEdge_1().getX1() + n.getNextEdge_1().getX2()) / 2;
                        int y = (n.getNextEdge_1().getY1() + n.getNextEdge_1().getY2()) / 2;
                        Canvas.SetLeft(lv1, x);
                        Canvas.SetTop(lv1, y - 20);
                    }

                }


            }
        }
    }
}
```

Listing 9: MatrixWindow.xaml

```
<Window x:Class="Faltungskodierer.MatrixWindow"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        Title="Matrix" mc:Ignorable="d" d:DesignHeight="198" SizeToContent="WidthAndHeight" d:DesignWidth="451">

    <StackPanel Margin="15">
        <StackPanel Orientation="Horizontal" >
        <Canvas Name="caMatrix" Margin="10"></Canvas>
        <StackPanel>
            <Expander Header="Matrixgröße" Width="100">
                <Expander.BorderBrush>Gray</Expander.BorderBrush>
                <Grid HorizontalAlignment="Center" Margin="10">
```

```
15                      <Grid.RowDefinitions>
16                          <RowDefinition/>
17                          <RowDefinition/>
18                      </Grid.RowDefinitions>
19                      <Grid.ColumnDefinitions>
20                          <ColumnDefinition/>
21                          <ColumnDefinition Width="30"/>
22                      </Grid.ColumnDefinitions>
23
24                      <Label Grid.Column="0" Grid.Row="0" HorizontalAlignment="Right">Zeilen:</Label>
25                      <Label Grid.Column="0" Grid.Row="1" HorizontalAlignment="Right">Spalten:</Label>
26                      <TextBox Grid.Column="1" Name="tbRow" PreviewTextInput="tbRow_PreviewTextInput"
                              TextChanged="tbRow_TextChanged"></TextBox>
27                      <TextBox Grid.Column="1" Grid.Row="1" Name="tbColumn"
                              PreviewTextInput="tbColumn_PreviewTextInput" TextChanged="tbColumn_TextChanged"></TextBox>
28                  </Grid>
29              </Expander>
30          </StackPanel>
31      </StackPanel>
32          <Button Margin="0,10" Click="Button_Click">Übernehmen</Button>
33      </StackPanel>
34  </Window>
```

Listing 10: MatrixWindow.xaml.cs

```
1   ï»¿using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Windows;
6   using System.Windows.Controls;
7   using System.Windows.Data;
8   using System.Windows.Documents;
9   using System.Windows.Input;
10  using System.Windows.Media;
11  using System.Windows.Media.Imaging;
12  using System.Windows.Navigation;
13  using System.Windows.Shapes;
14
15  namespace Faltungskodierer
16  {
17      /// <summary>
18      /// Interaction logic for Matrix.xaml
19      /// </summary>
20      public partial class MatrixWindow : Window
21      {
22          int[,] generatorMatrix;
23          public MatrixWindow()
24          {
25              InitializeComponent();
26              App a = (App)App.Current;
27              tbRow.Text = a.encoder.generatorMatrix.GetLength(0).ToString();
28              tbColumn.Text = a.encoder.generatorMatrix.GetLength(1).ToString();
29              this.generatorMatrix = (int[,])a.encoder.generatorMatrix.Clone();
30              drawMatrix();
31          }
32
33          public void drawMatrix()
34          {
35              caMatrix.Children.Clear();
36              int height = generatorMatrix.GetLength(0) * 25 - 5;
37              int width = generatorMatrix.GetLength(1) * 25 - 5;
38              caMatrix.Height = height;
39              caMatrix.Width = width;
40
41              for (int i = 0; i < generatorMatrix.GetLength(0); i++)
42              {
43                  for (int j = 0; j < generatorMatrix.GetLength(1); j++)
44                  {
45                      TextBox t = new TextBox();
46                      t.Name = "m"+i.ToString()+j.ToString();
47                      t.Text = generatorMatrix[i, j].ToString();
48                      t.Height = 20;
49                      t.Width = 20;
50                      t.HorizontalContentAlignment = System.Windows.HorizontalAlignment.Center;
51                      t.MaxLength = 1;
52                      t.PreviewTextInput += tbMatrix_PreviewTextInput;
53                      caMatrix.Children.Add(t);
54                      Canvas.SetLeft(t, j * 25);
55                      Canvas.SetTop(t, i * 25);
56                  }
57              }
```

```
58
59          }
60
61          private void tbRow_PreviewTextInput(object sender, TextCompositionEventArgs e)
62          {
63              e.Handled = !ValidNumeric(e.Text);
64              base.OnPreviewTextInput(e);
65          }
66
67          private void tbColumn_PreviewTextInput(object sender, TextCompositionEventArgs e)
68          {
69              e.Handled = !ValidNumeric(e.Text);
70              base.OnPreviewTextInput(e);
71          }
72
73          private void tbMatrix_PreviewTextInput(object sender, TextCompositionEventArgs e)
74          {
75              e.Handled = !BinaryValue(e.Text);
76              base.OnPreviewTextInput(e);
77          }
78
79          private bool BinaryValue(string str)
80          {
81              bool ret = true;
82
83              int l = str.Length;
84              for (int i = 0; i < l; i++)
85              {
86                  char ch = str[i];
87                  // nur 0 und 1 zugelassen
88                  ret &= (ch.ToString() == "0" || ch.ToString() == "1");
89              }
90
91              return ret;
92          }
93
94          private bool ValidNumeric(string str)
95          {
96              bool ret = true;
97
98              int l = str.Length;
99              for (int i = 0; i < l; i++)
100             {
101                 char ch = str[i];
102
103
104                 ret &= (Char.IsDigit(ch));
105             }
106
107             return ret;
108         }
109
110         private void createGMatrix(int row, int column)
111         {
112             int[,] newGMatrix= new int[row,column];
113             newGMatrix.Initialize();
114             this.generatorMatrix = newGMatrix;
115             drawMatrix();
116         }
117
118
119         private void tbColumn_TextChanged(object sender, TextChangedEventArgs e)
120         {
121             if (!tbRow.Text.Equals("") && !tbColumn.Text.Equals(""))
122             {
123                 if(Convert.ToInt16(tbColumn.Text) < 2)
124                     tbColumn.Text = "2";
125                 if (Convert.ToInt16(tbColumn.Text) > 15)
126                     tbColumn.Text = "15";
127
128                 createGMatrix(Convert.ToInt16(tbRow.Text), Convert.ToInt16(tbColumn.Text));
129             }
130
131         }
132
133         private void tbRow_TextChanged(object sender, TextChangedEventArgs e)
134         {
135             if (!tbRow.Text.Equals("") && !tbColumn.Text.Equals(""))
136             {
137                 if (Convert.ToInt16(tbRow.Text) < 2)
138                     tbRow.Text = "2";
139                 if (Convert.ToInt16(tbRow.Text) > 15)
140                     tbRow.Text = "15";
141
```

```
                createGMatrix(Convert.ToInt16(tbRow.Text), Convert.ToInt16(tbColumn.Text));
            }

        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            saveNewGMatrix();
        }

        private void saveNewGMatrix()
        {
            int i = 0;
            int j = 0;
            foreach (Object o in caMatrix.Children)
            {
                TextBox t = (TextBox)o;
                int column = Convert.ToInt16(tbColumn.Text);
                int row = Convert.ToInt16(tbRow.Text);

                this.generatorMatrix[j%row,i%column] = Convert.ToInt16(t.Text);
                if (i % column == column - 1)
                    j++;
                i++;
            }
            App a = (App)App.Current;
            a.encoder.generatorMatrix = this.generatorMatrix;
            this.Close();

        }


    }
}
```

Listing 11: ConvStudy.xaml

```
ï»¿<Page x:Class="Faltungskodierer.ConvStudy"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      mc:Ignorable="d"
        Title="ConvStudy">


    <StackPanel Margin="15">
        <Label>Text eingeben (mindestens 6 Buchstaben)</Label>
        <TextBox ScrollViewer.HorizontalScrollBarVisibility="Visible" Name="tbLetter" CharacterCasing="Upper"
             TextChanged="tbLetter_TextChanged"
             PreviewTextInput="tbLetter_PreviewTextInput">KANALKODIERUNG</TextBox>
        <DockPanel>
            <CheckBox Name="cbTerm">Terminierung</CheckBox>
            <CheckBox Margin="10,0" Name="cbBundel">BÃ¼ndelfehler</CheckBox>
        </DockPanel>
        <DockPanel>
            <Button IsEnabled="true" Name="btRun" Click="btRun_Click" Width="150">Simulation starten</Button>
            <ProgressBar Name="pbProgress" Maximum="4000"></ProgressBar>
        </DockPanel>
        <Canvas Name="caChart" Height="500" Width="800"></Canvas>
    </StackPanel>
</Page>
```

Listing 12: ConvStudy.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Collections;
using System.Threading;
```

```
16
17  namespace Faltungskodierer
18  {
19      /// <summary>
20      /// Interaction logic for ConvStudy.xaml
21      /// </summary>
22      public partial class ConvStudy : Page
23      {
24          string inputLetters = "";
25          Alphabet alphabet;
26          Encoding encode;
27          ArrayList inputBits;
28          Trellis trellis;
29          Viterbi viterbi;
30          ArrayList words;
31          Random rand;
32          int[] errorList;
33          string binary;
34          bool term, bundel;
35          App a;
36
37          public ConvStudy()
38          {
39              InitializeComponent();
40              a = (App)App.Current;
41              rand = new Random();
42              encode = new Encoding();
43              alphabet = new Alphabet();
44              inputBits = new ArrayList();
45              errorList = new int[40];
46              errorList.Initialize();
47              drawAxis();
48
49          }
50
51          public void convertInputbitToArray(string input)
52          {
53              int lenght = input.Length;
54              char[] inputChars = input.ToCharArray();
55              for (int i = 0; i < lenght; i++)
56              {
57                  inputBits.Add(Int32.Parse(inputChars[i].ToString()));
58              }
59          }
60
61          public string transformToLetters(ArrayList words)
62          {
63              string text = "";
64              for (int i = words.Count - 1; i >= 0; i--)
65                  text += alphabet.getChar((string)words[i]);
66              return text;
67          }
68
69          private void btRun_Click(object sender, RoutedEventArgs e)
70          {
71              if (trellis != null)
72              {
73                  trellis.Clear();
74                  viterbi = null;
75                  inputBits.Clear();
76                  for (int w = 0; w < 40; w++)
77                      errorList[w] = 0;
78
79              }
80
81              pbProgress.Value = 0;
82
83              binary = alphabet.charToBinary(tbLetter.Text);
84
85              string term_0 = "";
86              for (int l = 0; l < a.encoder.generatorMatrix.GetLength(1) - 1; l++)
87                  term_0 += "0";
88
89
90              if ((bool)cbTerm.IsChecked)
91                  binary += term_0;
92
93              encode.convertStateTableToHashTable();
94              string a_Star = encode.encode(binary);
95
96
97              convertInputbitToArray(a_Star);
98
99              term = (bool) cbTerm.IsChecked;
```

```
100             bundel = (bool) cbBundel.IsChecked;
101             Thread run = new Thread(calculate);
102             run.Start();
103
104
105         }
106
107         public void calculate()
108         {
109             for (int i = 1; i <= 40; i++)
110             {
111                 for (int j = 0; j < 100; j++)
112                 {
113                     if (words != null)
114                     {
115                         words.Clear();
116                         viterbi = null;
117                         trellis = null;
118                     }
119                     trellis = new Trellis((inputBits.Count / a.encoder.generatorMatrix.GetLength(0)),
                            a.encoder.stateTable, a.encoder.generatorMatrix.GetLength(0),
                            a.encoder.generatorMatrix.GetLength(1), term);
120                     viterbi = new Viterbi(trellis, makeErrors(inputBits, i, bundel),
                            a.encoder.generatorMatrix.GetLength(1) - 1, term);
121                     viterbi.setConv(false);
122                     viterbi.execute();
123                     viterbi.getCodeWord();
124                     string output = viterbi.showCodeWord().Substring(0, binary.Length);
125                     if (binary.Equals(output))
126                     {
127                         errorList[i - 1] += 1;
128                     }
129
130                     this.Dispatcher.BeginInvoke(new Action(() => this.pbProgress.Value = this.pbProgress.Value +
                            1), null);
131                 }
132
133             }
134             this.Dispatcher.BeginInvoke(new Action(() => drawChart()), null);
135         }
136
137         public ArrayList makeErrors(ArrayList input, int numberOfErrors, bool bundle)
138         {
139             ArrayList cloneInput = (ArrayList) input.Clone();
140             int count = cloneInput.Count;
141             Hashtable positions = new Hashtable();
142             int randomPos = rand.Next(count);
143             positions.Add(randomPos, null);
144             if (!bundle)
145                 for (int i = 0; i < numberOfErrors; i++)
146                 {
147                     cloneInput[randomPos] = ((int)cloneInput[randomPos] + 1) % 2;
148                     randomPos = rand.Next(count);
149                     if (!positions.ContainsKey(randomPos))
150                     {
151                         positions.Add(randomPos, null);
152                     }
153                     else
154                     {
155                         while (positions.ContainsKey(randomPos))
156                         {
157                             randomPos = rand.Next(count);
158                         }
159                         positions.Add(randomPos, null);
160                     }
161                 }
162             else
163             {
164                 randomPos = rand.Next(count-numberOfErrors);
165                 for (int j = 0 + randomPos; j < numberOfErrors + randomPos; j++)
166                 {
167                     cloneInput[j] = ((int)cloneInput[j] + 1) % 2;
168                 }
169             }
170             return cloneInput;
171         }
172
173         public string inputtoString(ArrayList cloneInput)
174         {
175             string codestring = "";
176             for (int i = 0; i < cloneInput.Count; i++)
177             {
178                 codestring += cloneInput[i].ToString();
179             }
```

```
180             return codestring;
181
182         }
183
184         public void drawAxis()
185         {
186
187             Line a1 = new Line();
188             a1.X1 = 40;
189             a1.Y1 = 20;
190             a1.X2 = 44;
191             a1.Y2 = 26;
192             a1.Stroke = Brushes.Black;
193             a1.StrokeThickness = 2;
194             caChart.Children.Add(a1);
195
196             Line a2 = new Line();
197             a2.X1 = 40;
198             a2.Y1 = 20;
199             a2.X2 = 36;
200             a2.Y2 = 26;
201             a2.Stroke = Brushes.Black;
202             a2.StrokeThickness = 2;
203             caChart.Children.Add(a2);
204
205
206             Line y = new Line();
207             y.X1 = 40;
208             y.Y1 = 20;
209             y.X2 = 40;
210             y.Y2 = 350;
211             y.Stroke = Brushes.Black;
212             y.StrokeThickness = 2;
213             caChart.Children.Add(y);
214
215             for (int i = 100 ; i >= 0; i -= 5)
216             {
217                 Line p = new Line();
218                 p.X1 = 35;
219                 p.Y1 = i * 3 + 40;
220                 p.X2 = 40;
221                 p.Y2 = i * 3 + 40;
222                 p.Stroke = Brushes.Black;
223                 p.StrokeThickness = 1;
224                 caChart.Children.Add(p);
225
226                 Label pl = new Label();
227                 int q = 100 - i;
228                 string number = "0" + (q).ToString();
229                 if (100 - i >= 10 )
230                     number = number.Substring(1, 2);
231                 if (q == 100)
232                     number += "0";
233                 if (q == 0)
234                     number = "";
235                 pl.Content = number;
236                 pl.Foreground = Brushes.Black;
237                 caChart.Children.Add(pl);
238                 Canvas.SetLeft(pl, 7);
239                 Canvas.SetTop(pl, i * 3 + 26);
240             }
241
242             Line x = new Line();
243             x.X1 = 30;
244             x.Y1 = 340;
245             x.X2 = 740;
246             x.Y2 = 340;
247             x.Stroke = Brushes.Black;
248             x.StrokeThickness = 2;
249             caChart.Children.Add(x);
250
251             Line a3 = new Line();
252             a3.X1 = 740;
253             a3.Y1 = 340;
254             a3.X2 = 734;
255             a3.Y2 = 344;
256             a3.Stroke = Brushes.Black;
257             a3.StrokeThickness = 2;
258             caChart.Children.Add(a3);
259
260             Line a4 = new Line();
261             a4.X1 = 740;
262             a4.Y1 = 340;
263             a4.X2 = 734;
```

```
264             a4.Y2 = 336;
265             a4.Stroke = Brushes.Black;
266             a4.StrokeThickness = 2;
267             caChart.Children.Add(a4);
268
269             for (int i = 1; i <= 40; i++)
270             {
271                 Line p = new Line();
272                 p.X1 = i * 17 + 40;
273                 p.Y1 = 340;
274                 p.X2 = i * 17 + 40;
275                 p.Y2 = 345;
276                 p.Stroke = Brushes.Black;
277                 p.StrokeThickness = 1;
278                 caChart.Children.Add(p);
279
280                 Label pl = new Label();
281                 string number = "0" + i.ToString();
282                 if (i >= 10)
283                     number = number.Substring(1, 2);
284                 pl.Content = number;
285                 pl.Foreground = Brushes.Black;
286                 caChart.Children.Add(pl);
287                 Canvas.SetLeft(pl, i * 17 + 29);
288                 Canvas.SetTop(pl, 340);
289             }
290
291             //Achsenbezeichener
292             Label l = new Label();
293             l.Content = "Fehleranzahl in Bit";
294             l.Foreground = Brushes.Black;
295             caChart.Children.Add(l);
296             Canvas.SetLeft(l, 640);
297             Canvas.SetTop(l, 355);
298
299             Label lpro = new Label();
300             lpro.Content = "Fehlerkorrekturrate in %";
301             lpro.Foreground = Brushes.Black;
302             caChart.Children.Add(lpro);
303             Canvas.SetLeft(lpro, 330);
304             Canvas.SetTop(lpro, 10);
305         }
306
307         public void drawChart()
308         {
309             caChart.Children.Clear();
310             drawAxis();
311             for (int i = 0; i < 40; i++)
312             {
313                 Rectangle rect = new Rectangle();
314                 rect.Height = errorList[i] * 3;
315                 rect.Width = 12;
316                 rect.Fill = Brushes.LimeGreen;
317                 rect.Stroke = Brushes.DarkGray;
318                 rect.StrokeThickness = 1;
319                 rect.ToolTip = errorList[i].ToString();
320
321                 caChart.Children.Add(rect);
322                 Canvas.SetLeft(rect, (i * 17 + 40 + 17) - 6);
323                 Canvas.SetTop(rect, 340 - errorList[i] * 3);
324             }
325         }
326
327         private void tbLetter_TextChanged(object sender, TextChangedEventArgs e)
328         {
329             if(btRun != null)
330             if (tbLetter.Text.Length >= 10)
331             {
332                 btRun.IsEnabled = true;
333             }
334             else
335             {
336                 btRun.IsEnabled = false;
337             }
338         }
339
340         private void tbLetter_PreviewTextInput(object sender, TextCompositionEventArgs e)
341         {
342             e.Handled = !ValidText(e.Text);
343             base.OnPreviewTextInput(e);
344         }
345
346         bool ValidText(string str)
347         {
```

```
348             bool ret = true;
349 
350             int l = str.Length;
351             for (int i = 0; i < l; i++)
352             {
353 
354                 char ch = str[i];
355                 ret &= (Char.IsLetter(ch) && !ch.ToString().Equals("ß") && !ch.ToString().Equals("ö") &&
                        !ch.ToString().Equals("ä") && !ch.ToString().Equals("ü") && !ch.ToString().Equals("Ö") &&
                        !ch.ToString().Equals("Ä") && !ch.ToString().Equals("Ü"));
356             }
357 
358             return ret;
359         }
360     }
361 }
```

Listing 13: Alphabet.cs

```
1   ï»¿using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Collections;
6   
7   namespace Faltungskodierer
8   {
9       public class Alphabet:Hashtable
10      {
11          public Alphabet()
12          {
13              this.Add("00001", "A");
14              this.Add("00010", "B");
15              this.Add("00011", "C");
16              this.Add("00100", "D");
17              this.Add("00101", "E");
18              this.Add("00110", "F");
19              this.Add("00111", "G");
20              this.Add("01000", "H");
21              this.Add("01001", "I");
22              this.Add("01010", "J");
23              this.Add("01011", "K");
24              this.Add("01100", "L");
25              this.Add("01101", "M");
26              this.Add("01110", "N");
27              this.Add("01111", "O");
28              this.Add("10000", "P");
29              this.Add("10001", "Q");
30              this.Add("10010", "R");
31              this.Add("10011", "S");
32              this.Add("10100", "T");
33              this.Add("10101", "U");
34              this.Add("10110", "V");
35              this.Add("10111", "W");
36              this.Add("11000", "X");
37              this.Add("11001", "Y");
38              this.Add("11010", "Z");
39              this.Add("11011", " ");
40  
41              this.Add("A", "00001");
42              this.Add("B", "00010");
43              this.Add("C", "00011");
44              this.Add("D", "00100");
45              this.Add("E", "00101");
46              this.Add("F", "00110");
47              this.Add("G", "00111");
48              this.Add("H", "01000");
49              this.Add("I", "01001");
50              this.Add("J", "01010");
51              this.Add("K", "01011");
52              this.Add("L", "01100");
53              this.Add("M", "01101");
54              this.Add("N", "01110");
55              this.Add("O", "01111");
56              this.Add("P", "10000");
57              this.Add("Q", "10001");
58              this.Add("R", "10010");
59              this.Add("S", "10011");
60              this.Add("T", "10100");
61              this.Add("U", "10101");
62              this.Add("V", "10110");
63              this.Add("W", "10111");
```

```
64              this.Add("X", "11000");
65              this.Add("Y", "11001");
66              this.Add("Z", "11010");
67              this.Add(" ", "11011");
68          }
69
70          public string getChar(string pattern)
71          {
72              App a = (App)App.Current;
73              Window1 win1 = (Window1)a.Windows[0];
74
75              try
76              {
77                  return this[pattern].ToString();
78              }
79              catch(Exception e)
80              {
81                  win1.sbText.Content = "Eingabefolge enthielt ungÃ¼ltige/nicht im Alphabet vereinbarte Zeichen";
82                  return "";
83              }
84
85          }
86
87          public string charToBinary(string chars)
88          {
89
90              chars = chars.ToUpper();
91
92              string binary = "";
93              for (int i = 0; i < chars.Length; i++)
94              {
95                  string letter = chars[i].ToString();
96                  binary += this[letter].ToString();
97              }
98
99              return binary;
100         }
101     }
102 }
```

Listing 14: Clk.cs

```
1   ï»¿using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Collections;
6
7   namespace Faltungskodierer
8   {
9       public class Clk:Hashtable
10      {
11          int step;
12          int[,]states;
13          int row;
14          int column;
15          int wDist;
16          int hDist;
17
18          public Clk(int step,int[,] stateTable, int row, int column)
19          {
20              this.step = step;
21              this.states = stateTable;
22              this.row = row;
23              this.column = column;
24              this.wDist = 80;
25              this.hDist = 27;
26              createNodes();
27          }
28
29          public void createNodes()
30          {
31              for (int i = 0; i < states.GetLength(0); i += 2)
32              {
33                  string state = "";
34                  for (int j = 1; j < column; j++)
35                  {
36                      state += states[i, j].ToString();
37                  }
38                  int x = step * wDist + 40;
39                  int y = i * hDist + 60;
40                  this.Add(state, new Node(step, state, x, y));
```

```
41              }
42
43          }
44
45          public string toString()
46          {
47              string values = "";
48              foreach(object o in this.Keys)
49              {
50                  Node n = (Node) this[o];
51                  values += n.getState() + "\n";
52              }
53              return values;
54          }
55
56          public int getStep()
57          {
58              return step;
59          }
60
61      }
62  }
```

Listing 15: ConvEncoder.cs

```
1   ï»¿using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5
6   namespace Faltungskodierer
7   {
8       public class ConvEncoder
9       {
10          private int[,] _generatorMatrix;
11          private int[,] _stateTable;
12
13          public int[,] generatorMatrix
14          {
15              get
16              {
17                  return _generatorMatrix;
18              }
19              set
20              {
21                  this.generateStateTable(value);
22                  _generatorMatrix = value;
23              }
24          }
25
26          public int[,] stateTable
27          {
28              get
29              {
30                  return _stateTable;
31              }
32
33          }
34
35          public void generateStateTable(int[,] generatorMatrix)
36          {
37              int row = generatorMatrix.GetLength(0);
38              int column = generatorMatrix.GetLength(1);
39
40              int r = (int)Math.Pow(2, column);
41              int c = (1 + 2 * (column - 1) + row);
42              //zeile X spalte
43              _stateTable = new int[r, c];
44              int k = 1;
45              int countz1 = 0;
46              int countz2 = 0;
47
48              for (int j = 0; j < column; j++)
49              {
50                  for (int i = 0; i < r; i++)
51                  {
52                      if (countz1 < k)
53                      {
54                          _stateTable[i, j] = 0;
55                          if (countz2 < column - 1)
56                              _stateTable[i, j + column] = 0;
57                      }
```

```
                    else
                    {
                        _stateTable[i, j] = 1;
                        if (countz2 < column - 1)
                            _stateTable[i, j + column] = 1;
                    }
                    countz1++;
                    countz1 %= k * 2;
                }
                countz2++;
                k *= 2;
            }

            int v = column + column - 1;
            for (int l = 0; l < row; l++)
            {
                for (int u = 0; u < r; u++)
                {
                    int sum = 0;
                    for (int m = 0; m < column; m++)
                    {
                        if (generatorMatrix[l, m] == 1)
                        {
                            sum += _stateTable[u, m];
                        }
                    }
                    sum %= 2;
                    _stateTable[u, v + l] = sum;
                }
            }
        }
    }
}
```

Listing 16: Encoding.cs

```
ï»¿using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;

namespace Faltungskodierer
{
    class Encoding
    {

        Hashtable stateHashTable;
        App a;

        public Encoding()
        {
            a = (App)App.Current;

            stateHashTable = new Hashtable();
            convertStateTableToHashTable();
        }

        public void convertStateTableToHashTable()
        {
            stateHashTable.Clear();
            for (int i = 0; i < a.encoder.stateTable.GetLength(0); i++)
            {
                string key = "";
                for (int j = 0; j < a.encoder.generatorMatrix.GetLength(1); j++)
                    key += a.encoder.stateTable[i, j];

                string nextnode = "";
                for (int l = a.encoder.generatorMatrix.GetLength(1); l < a.encoder.generatorMatrix.GetLength(1) +
                      (a.encoder.generatorMatrix.GetLength(1) - 1); l++)
                    nextnode += a.encoder.stateTable[i, l];

                string output = "";
                for (int k = a.encoder.stateTable.GetLength(1) - a.encoder.generatorMatrix.GetLength(0); k <
                     a.encoder.stateTable.GetLength(1); k++)
                    output += a.encoder.stateTable[i, k];

                string[] n = { nextnode, output };

                stateHashTable.Add(key, n);
            }
        }
```

```
45
46          public string encode(string b)
47          {
48              string output = "";
49              int lenght = b.Length;
50              string startpoint = "";
51              for (int l = 0; l < a.encoder.generatorMatrix.GetLength(1) - 1; l++)
52                  startpoint += "0";
53
54              string nextPoint = startpoint;
55              for (int i = 0; i < lenght; i++)
56              {
57                  string[] n = (string[]) stateHashTable[b[i] + nextPoint];
58                  nextPoint = n[0];
59                  output += n[1];
60              }
61
62              return output;
63          }
64      }
65  }
```

Listing 17: Viterbi.cs

```
1   ï»¿using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Collections;
6   using System.Windows;
7   using System.Windows.Controls;
8   using System.Windows.Data;
9   using System.Windows.Documents;
10  using System.Windows.Input;
11  using System.Windows.Media;
12  using System.Windows.Media.Imaging;
13  using System.Windows.Navigation;
14  using System.Windows.Shapes;
15  using System.Threading;
16
17
18  namespace Faltungskodierer
19  {
20      public class Viterbi
21      {
22          Trellis trellis;
23          ArrayList inputBits;
24          int oscillationTime;
25          bool termination;
26          ArrayList codeWord;
27          App a;
28          bool conv = true;
29
30
31
32          public Viterbi(Trellis trellis, ArrayList inputBits, int oscillationTime, bool termination)
33          {
34              this.trellis = trellis;
35              this.inputBits = inputBits;
36              this.oscillationTime = oscillationTime;
37              this.termination = termination;
38              codeWord = new ArrayList();
39              a = (App)App.Current;
40
41          }
42
43          public void setConv(bool conv){
44              this.conv = conv;
45          }
46
47          public void execute()
48          {
49              string startPoint = "";
50
51              for (int i = 0; i < a.encoder.generatorMatrix.GetLength(0); i++)
52                  startPoint += "0";
53
54              for (int i = 0; i <= trellis.getStepsCount(); i++)
55              {
56                  Hashtable step = (Hashtable)trellis[i];
57                  foreach (Object o in step.Keys)
58                  {
```

```
59                  Node n = (Node) step[o];
60                  Edge e_a = n.getPrevEdge_A();
61                  Edge e_b = n.getPrevEdge_B();
62
63                  if (e_a != null && e_b != null)
64                  {
65                      int aktValue_a = e_a.getAktValue();
66                      int aktValue_b = e_b.getAktValue();
67
68                      if (aktValue_a < aktValue_b)
69                      {
70                          n.setMinValue(aktValue_a);
71                          e_b.SetEnable(false);
72                      }
73                      else
74                      {
75                          n.setMinValue(aktValue_b);
76                          e_a.SetEnable(false);
77                      }
78                  }
79                  else
80                      if (e_a != null && e_b == null)
81                      {
82                          int aktValue_a = e_a.getAktValue();
83                          n.setMinValue(aktValue_a);
84                      }
85                      else
86                          if (e_b != null && e_a == null)
87                          {
88                              int aktValue_b = e_b.getAktValue();
89                              n.setMinValue(aktValue_b);
90                          }
91                          else
92                              n.setMinValue(0);
93
94                  if (i < trellis.getStepsCount())
95                  {
96                      ArrayList inputPattern = new ArrayList();
97                      for (int j = i * a.encoder.generatorMatrix.GetLength(0); j < i *
                             a.encoder.generatorMatrix.GetLength(0) + a.encoder.generatorMatrix.GetLength(0); j++)
98                          inputPattern.Add(inputBits[j]);
99
100
101                     int minvalue = n.getMinValue();
102                     Edge e_0 = n.getNextEdge_0();
103                     Edge e_1 = n.getNextEdge_1();
104
105                     if (e_0 != null)
106                         e_0.matchPattern(inputPattern, minvalue);
107                     if (e_1 != null)
108                         e_1.matchPattern(inputPattern, minvalue);
109                 }
110
111             }
112         }
113     }
114
115     public ArrayList getAlphabetCodes()
116     {
117         int start = 0;
118         if (termination)
119             start = oscillationTime;
120
121         ArrayList words = new ArrayList();
122
123         for (int i = start; i < codeWord.Count; i += 5)
124         {
125             try
126             {
127                 words.Add(codeWord[i + 4].ToString() +
128                       codeWord[i + 3].ToString() +
129                       codeWord[i + 2].ToString() +
130                       codeWord[i + 1].ToString() +
131                       codeWord[i].ToString());
132
133                 if(conv)
134                 try
135                 {
136                     Window1 win1 = (Window1)a.Windows[0];
137                     win1.sbText.Content = "Eingabe Korrekt";
138                     conv = true;
139                 }
140                 catch (Exception exeption)
141                 {
```

```
142                         conv = false;
143                     }
144
145
146                 }
147                 catch(Exception exept)
148                 {
149
150                 }
151
152             }
153
154             return words;
155         }
156
157         public ArrayList getCodeWord()
158         {
159             getprevStep(getBestNode());
160             return getAlphabetCodes();
161         }
162
163         public ArrayList getCodeWordArray()
164         {
165             return this.codeWord;
166         }
167
168         public Node getBestNode()
169         {
170
171             Hashtable lastNodes = (Hashtable) trellis[trellis.Count-1];
172             Node bestNode = null;
173             foreach (Object o in lastNodes.Keys)
174             {
175                 if (termination)
176                 {
177                     string startpoint = "";
178                     for (int l = 0; l < a.encoder.generatorMatrix.GetLength(1) - 1; l++)
179                         startpoint += "0";
180                     return (Node)lastNodes[startpoint];
181                 }
182
183
184                 if (bestNode == null)
185                     bestNode = (Node) lastNodes[o];
186                 else
187                 {
188                     Node aktNode = (Node)lastNodes[o];
189                     if (bestNode.getMinValue() > aktNode.getMinValue() && (aktNode.getPrevEdge_A() != null ||
                          aktNode.getPrevEdge_B() != null))
190                         bestNode = aktNode;
191
192                 }
193             }
194
195             return bestNode;
196         }
197
198         public void getprevStep(Node aktNode)
199         {
200             if (aktNode.getPrevEdge_A() == null && aktNode.getPrevEdge_B() == null)
201                 return;
202
203             if (aktNode.getPrevEdge_A().getEnable())
204             {
205                 aktNode.getPrevEdge_A().setPath(true);
206                 codeWord.Add(aktNode.getPrevEdge_A().getTyp());
207                 getprevStep(aktNode.getPrevEdge_A().getPrevNode());
208             }
209             else
210             {
211                 aktNode.getPrevEdge_B().setPath(true);
212                 codeWord.Add(aktNode.getPrevEdge_B().getTyp());
213                 getprevStep(aktNode.getPrevEdge_B().getPrevNode());
214             }
215         }
216
217
218
219         public string showCodeWord()
220         {
221             string codeWordString =  "";
222             for(int i = 0; i < codeWord.Count; i++){
223                 codeWordString = codeWord[i] + codeWordString;
224             }
```

```
225             return codeWordString;
226         }
227 
228         public void setInputBits(ArrayList inputBits)
229         {
230             this.inputBits = inputBits;
231         }
232 
233         public string codeworttoString()
234         {
235             string codestring = "";
236             for (int i = 0; i < codeWord.Count; i++)
237             {
238                 codestring += codeWord[i].ToString();
239             }
240             return codestring;
241 
242         }
243     }
244 }
```

Listing 18: Trellis.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Collections;
6 
7   namespace Faltungskodierer
8   {
9       public class Trellis:Hashtable
10      {
11          int steps;
12          int row;
13          int column;
14          bool termination;
15          App a;
16 
17          public Trellis(int steps, int[,] stateTable, int row, int column, bool termination)
18          {
19              this.steps = steps;
20              this.row = row;
21              this.column = column;
22              this.termination = termination;
23              a = (App)App.Current;
24              initializeViterbiSteps();
25              initializeViterbiEdges();
26 
27          }
28 
29          public void initializeViterbiSteps()
30          {
31              for (int i = 0; i <= steps ; i++)
32              {
33                  this.Add(i, new Clk(i, a.encoder.stateTable, row, column));
34              }
35          }
36 
37          public void initializeViterbiEdges()
38          {
39              transientOscillation();
40              fullTrellisSteps();
41              if (termination)
42                  terminationOscillation();
43          }
44 
45          public void transientOscillation()
46          {
47              Hashtable allowedPoints = new Hashtable();
48 
49              string startpointGlobal = "";
50              for (int i = 0; i < column - 1; i++)
51              {
52                  startpointGlobal += "0";
53              }
54 
55              allowedPoints.Add(startpointGlobal, null);
56 
57              for (int s = 0; s < column - 1; s++)
58              {
59                  Hashtable current = (Hashtable)this[s];
```

```
60                  Hashtable next = (Hashtable)this[s + 1];
61                  Hashtable tempAllowedPoints = new Hashtable();
62
63                  for (int j = 0; j < a.encoder.stateTable.GetLength(0); j++)
64                  {
65                      //Vergleichspattern, Start- und Endpunkt der Kante Festlegen aus der Zustandstabelle
66                      ArrayList pattern = getPattern(j);
67                      string startpoint = getStartpoint(j);
68                      string endpoint = getEndpoint(j);
69
70                      //Kante Anlegen
71                      Edge e = new Edge(startpoint, endpoint, pattern, a.encoder.stateTable[j, 0]);
72
73                      //Test ob Startpunkt ein schon erreichter Punkt ist
74                      if (allowedPoints.ContainsKey(startpoint))
75                      {
76                          //eintragen des Endpunktes der Kante in die erreichte temporäre Kantenmenge
77                          if (!allowedPoints.ContainsKey(endpoint))
78                              tempAllowedPoints.Add(endpoint, null);
79
80                          //Punkte holen
81                          Node currentNode = (Node)current[startpoint];
82                          Node nextNode = (Node)next[endpoint];
83                          setPrevAndDrawPoints(e, currentNode, nextNode);
84
85                          //Kante an den Startpunkt binden
86                          if (a.encoder.stateTable[j, 0] == 0)
87                              currentNode.setNextEdge_0(e);
88                          else
89                              currentNode.setNextEdge_1(e);
90
91                          //Kante an den Endpunkt binden
92                          if (nextNode.getPrevEdge_A() == null)
93                              nextNode.setPrevEdge_A(e);
94                          else
95                              nextNode.setPrevEdge_B(e);
96                      }
97                  }
98                  //übertragen der temporären erreichbaren Punkte
99                  foreach (Object o in tempAllowedPoints.Keys)
100                     if (!allowedPoints.ContainsKey(o))
101                         allowedPoints.Add(o, null);
102
103
104             }
105         }
106
107         public void fullTrellisSteps()
108         {
109             //wenn Terminierung Hauptteil und Terminierung verkürzen
110             int term = 0;
111             if (termination)
112                 term = (column - 1);
113
114             for (int i = column - 1; i < steps - term; i++)
115             {
116                 Hashtable current = (Hashtable)this[i];
117                 Hashtable next = (Hashtable)this[i + 1];
118
119                 //über alle Kanten aus Zustandstabelle iterieren
120                 for (int j = 0; j < a.encoder.stateTable.GetLength(0); j++)
121                 {
122                     //Vergleichspattern, Start- und Endpunkt der Kante Festlegen aus der Zustandstabelle
123                     ArrayList pattern = getPattern(j);
124                     string startpoint = getStartpoint(j);
125                     string endpoint = getEndpoint(j);
126
127                     //Kante Anlegen
128                     Edge e = new Edge(startpoint, endpoint, pattern, a.encoder.stateTable[j, 0]);
129
130                     //Punkte holen und Koordinaten für Visualisierung in die Kante eintragen
131                     Node currentNode = (Node)current[startpoint];
132                     Node nextNode = (Node)next[endpoint];
133                     setPrevAndDrawPoints(e, currentNode, nextNode);
134
135                     //Kante an den Startpunkt binden
136                     if (a.encoder.stateTable[j, 0] == 0)
137                         currentNode.setNextEdge_0(e);
138                     else
139                         currentNode.setNextEdge_1(e);
140
141                     //Kante an den Endpunkt binden
142                     if (nextNode.getPrevEdge_A() == null)
143                         nextNode.setPrevEdge_A(e);
```

```
144                     else
145                         nextNode.setPrevEdge_B(e);
146                 }
147             }
148         }
149
150         public void terminationOscillation()
151         {
152             for (int i = steps - (column - 1); i < steps; i++)
153             {
154                 Hashtable current = (Hashtable)this[i];
155                 Hashtable next = (Hashtable)this[i + 1];
156
157                 for (int j = 0; j < a.encoder.stateTable.GetLength(0); j++)
158                 {
159                     //Vergleichspattern, Start- und Endpunkt der Kante Festlegen aus der Zustandstabelle
160                     ArrayList pattern = getPattern(j);
161                     string startpoint = getStartpoint(j);
162                     string endpoint = getEndpoint(j);
163
164                     Edge e = new Edge(startpoint, endpoint, pattern, a.encoder.stateTable[j, 0]);
165
166                     //Punkte holen und Koordinaten für Visualisierung in die Kante eintragen
167                     Node currentNode = (Node)current[startpoint];
168                     Node nextNode = (Node)next[endpoint];
169                     setPrevAndDrawPoints(e, currentNode, nextNode);
170
171                     if (a.encoder.stateTable[j, 0] == 0)
172                     {
173                         if (currentNode.getPrevEdge_A() != null || currentNode.getPrevEdge_A() != null)
174                         {
175                             currentNode.setNextEdge_0(e);
176
177
178                             if (nextNode.getPrevEdge_A() == null)
179                                 nextNode.setPrevEdge_A(e);
180                             else
181                                 nextNode.setPrevEdge_B(e);
182                         }
183                     }
184                 }
185             }
186         }
187
188         public int getStepsCount()
189         {
190             return steps;
191         }
192
193         public ArrayList getPattern(int j)
194         {
195             ArrayList pattern = new ArrayList();
196             for (int p = 2 * column - 1; p < 2 * column - 1 + row; p++)
197                 pattern.Add(a.encoder.stateTable[j, p]);
198             return pattern;
199         }
200
201         public string getStartpoint(int j)
202         {
203             string startpoint = "";
204             for (int l = 1; l < column; l++)
205                 startpoint += a.encoder.stateTable[j, l].ToString();
206             return startpoint;
207         }
208
209         public string getEndpoint(int j)
210         {
211             string endpoint = "";
212             for (int q = column; q < 2 * column - 1; q++)
213                 endpoint += a.encoder.stateTable[j, q].ToString();
214             return endpoint;
215         }
216
217         public void setPrevAndDrawPoints(Edge e, Node currentNode, Node nextNode )
218         {
219             //Koordinaten für Visualisierung in die Kante eintragen
220             e.setX1(currentNode.getX());
221             e.setY1(currentNode.getY());
222             e.setX2(nextNode.getX());
223             e.setY2(nextNode.getY());
224             //Vorgänger Knoten für rekursiven Durchlauf des Graphen eintragen
225             e.setPrevNode(currentNode);
226         }
227     }
```

```
228 }
```

Listing 19: Edge.cs

```
1   ï»¿using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Collections;
6
7   namespace Faltungskodierer
8   {
9       public class Edge
10      {
11          bool enable;
12          ArrayList pattern;
13          string startPoint;
14          string endPoint;
15          int match;
16          int typ;
17          int aktValue;
18          int x1, x2, y1, y2;
19          Node prevNode;
20          bool path;
21
22          public Edge(string startPoint, string endPoint, ArrayList pattern, int typ)
23          {
24              this.startPoint = startPoint;
25              this.endPoint = endPoint;
26              this.pattern = pattern;
27              this.typ = typ;
28              this.enable = true;
29              this.aktValue = 0;
30              this.prevNode = null;
31              this.path = false;
32          }
33
34          public void matchPattern(ArrayList iPattern, int minValue)
35          {
36              int diff = 0;
37              for (int i = 0; i < iPattern.Count; i++)
38              {
39                  if ((int)iPattern[i] != (int)pattern[i])
40                      diff++;
41              }
42              this.match = diff;
43              this.aktValue = minValue + diff;
44          }
45
46          public int getTyp()
47          {
48              return typ;
49          }
50
51          public void setPrevNode(Node n)
52          {
53              this.prevNode = n;
54          }
55
56          public Node getPrevNode()
57          {
58              return prevNode;
59          }
60
61          public void setX1(int x1)
62          {
63              this.x1 = x1;
64          }
65
66          public void setX2(int x2)
67          {
68              this.x2 = x2;
69          }
70
71          public void setY1(int y1)
72          {
73              this.y1 = y1;
74          }
75
76          public void setY2(int y2)
77          {
78              this.y2 = y2;
```

```
79          }
80
81          public int getX1()
82          {
83              return this.x1;
84          }
85
86          public int getX2()
87          {
88              return this.x2;
89          }
90
91          public int getY1()
92          {
93              return this.y1;
94          }
95
96          public int getY2()
97          {
98              return this.y2;
99          }
100
101         public void setMatch(int m)
102         {
103             this.match = m;
104         }
105
106         public void setAktValue(int a)
107         {
108             this.aktValue = a;
109         }
110
111         public void SetEnable(bool b)
112         {
113             this.enable = b;
114         }
115
116         public int getMatch()
117         {
118             return this.match;
119         }
120
121         public int getAktValue()
122         {
123             return this.aktValue;
124         }
125
126         public bool getEnable()
127         {
128             return this.enable;
129         }
130
131         public string getEndPoint()
132         {
133             return this.endPoint;
134         }
135
136         public void setPath(bool b)
137         {
138             this.path = b;
139         }
140
141         public bool getPath()
142         {
143             return this.path;
144         }
145
146     }
147 }
```

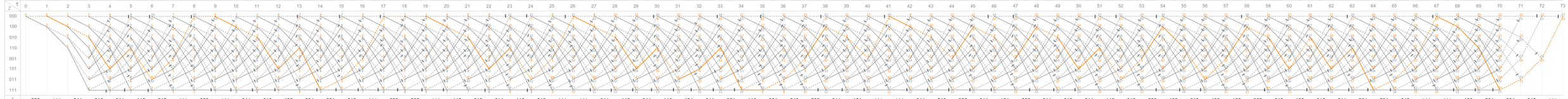
Listing 20: Node.cs

```
1  ï»¿using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5
6  namespace Faltungskodierer
7  {
8      public class Node
9      {
10         Edge prevEdge_A;
```

```
11          Edge prevEdge_B;
12          Edge nextEdge_0;
13          Edge nextEdge_1;
14          int step;
15          string state;
16          int minValue;
17          int x, y;
18
19          public Node(int step, string state,int x,int y)
20          {
21              this.prevEdge_A = null;
22              this.prevEdge_B = null;
23              this.nextEdge_0 = null;
24              this.nextEdge_1 = null;
25              this.step = step;
26              this.state = state;
27              minValue = 0;
28              this.x = x;
29              this.y = y;
30
31          }
32
33          public int getX()
34          {
35              return this.x;
36          }
37
38          public int getY()
39          {
40              return this.y;
41          }
42
43          public Edge getPrevEdge_A()
44          {
45              return this.prevEdge_A;
46          }
47
48          public Edge getPrevEdge_B()
49          {
50              return this.prevEdge_B;
51          }
52
53          public void setPrevEdge_A(Edge A)
54          {
55              this.prevEdge_A = A;
56          }
57
58          public void setPrevEdge_B(Edge B)
59          {
60              this.prevEdge_B = B;
61          }
62
63          public int getStep()
64          {
65              return this.step;
66          }
67
68          public void setStep(int step)
69          {
70              this.step = step;
71          }
72
73          public string getState()
74          {
75              return this.state;
76          }
77
78          public Edge getNextEdge_0()
79          {
80              return this.nextEdge_0;
81          }
82
83          public void setNextEdge_0(Edge N)
84          {
85              this.nextEdge_0 = N;
86          }
87
88          public Edge getNextEdge_1()
89          {
90              return this.nextEdge_1;
91          }
92
93          public void setNextEdge_1(Edge O)
94          {
```

```
95              this.nextEdge_1 = 0;
96          }
97
98          public void setMinValue(int v)
99          {
100             this.minValue = v;
101         }
102
103         public int getMinValue()
104         {
105             return this.minValue;
106         }
107
108
109     }
110 }
```

VITERBI für Empfangsfolge b mit Faltungskodierer G


y_t \ t
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
000
100
010
110
001
101
011
111
h 000 111 011 010 011 110 010 111 000 111 011 010 100 001 001 010 111 000 000 111 110 010 011 110 010 111 111 011 110 011 110 101 011 001 110 001 010 000 011 101 111 111 011 010 000 011 101 000 011 010 011 110 010 000 100 010 100 100 000 010 100 010 011 001 001 010 111 111 100 001 001 010 111
h^* 0 1 0 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1 0 1 1 0 0 0 1 0 1 1 1 1 0 0 1 0 0 0 1 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 1 1 0 0 0 1 1 1 0 0 0