

Modulacja i Kodowanie

Labolatorium

Kodowanie kanałowe – kody konwolucyjne

Kody splotowe (konwolucyjne)

Główną różnicą pomiędzy kodami blokowymi a konwolucyjnymi (splotowymi) polega na konstrukcji ciągu kodowego. W przypadku kodów blokowych mamy do czynienia z blokiem kodu o pewnej, ustalonej, i najczęściej, niezmiennej długości. Część bloku stanowią informacyjne, a resztę tzw. bity kontrolne. Np. w kodzie Hamminga (7,4) blok kody zawiera 7 bitów, z czego 4 to bity informacyjne, a 3 to bity kontrolne. Bity kontrolne są rozmieszczone nie przypadkowo, lecz na ściśle określonych pozycjach – w przypadku kodu Hamminga znajdują się na pozycji pierwszej, a następnie na pozycjach będących potęgami liczby 2.

Natomiast kody splotowe nie wymagają dzielenia strumienia informacji na bloke. Kodowany ciąg jest niejako w “locie”, a sam process kodowania można przyrównać do klasycznego splotu jednowymiarowego (konwolucji) sygnału z maską filtru. Bity kontrolne są rozmieszczane wzdłuż strumienia. Kody splotowe zwykle są określane przez trzy parametry: (n, k, m) . Ideą kodowania splotowego jest przekształcenie wejściowego k -bitowego ciągu informacyjnego na n -bitowy ciąg wyjściowy. Sprawność kodu splotowego wynosi k/n ($n \geq k$). Dodatkowym parametrem jest m , który oznacza liczbę przerzutników „D” w rejestrze układu kodującego albo ilość boksów, z których ten rejestr się składa. Kody konwolucyjne generacją pary (lub większe) liczby bitów w odpowiedzi na bity wejściowe.

Kody konwolucyjne charakteryzują się trema, podstawowymi wartościami:

$$(n, k, m)$$

gdzie:

n – liczba bitów wejściowych,

k – liczba bitów wyjściowych,

m - długość rejestru pamięci.

sprawność kodu (*code rate*) jest dana :

$$code\ rate = k / n$$

Długość rejestru pamięci (*Constraint length*) L jest dana wzorem:

$$L = k (m - 1)$$

L oznacza liczbę bitów w pamięci enkodera, która wpływa na generowanie n bitów wyjściowych.

- Koder splotowy

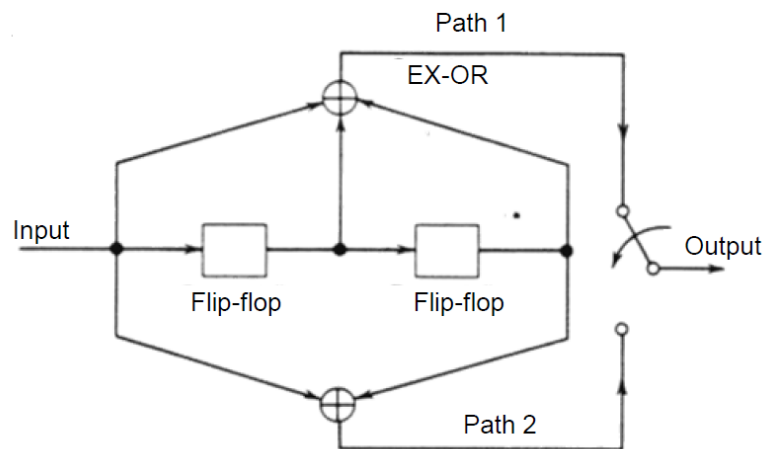


Figure 1: Przykład kodera splotowego

Implementację kodera splotowego pokazano na rysunku 1. Kod generowany przez ten kod jest niesystematycznym kodem. W przeciwieństwie do kodów lokowych, zwykle stosuje się niesystematyczne kody splotowe.

- Koder splotowy jako maszyna stanowa (skończony automat deterministyczny)

Enkoder konwolucyjny może być również reprezentowany jako maszyna skończona. Zauważ, że (w binarnym kanale) możemy zdefiniować tylko cztery stany możliwości: 00, 01, 10, 11 (są to wszystkie, możliwe kombinacje pary bitów 0 i 1), patrz rys. 2.

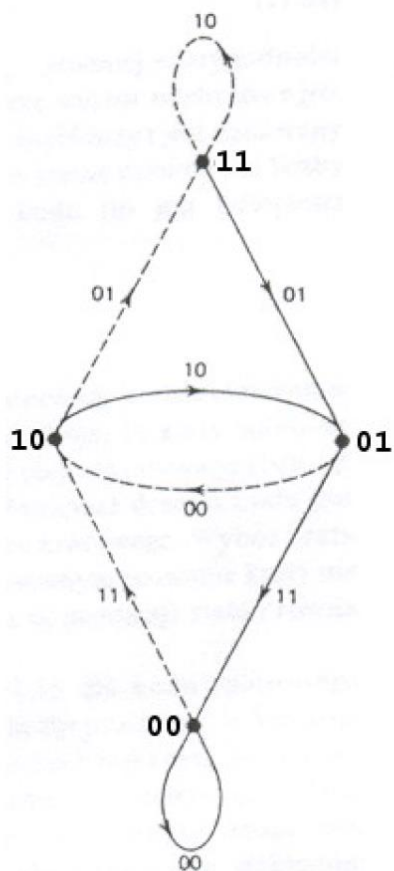


Figure 2 Koder splotowy jako deterministyczny automat skończony.

Zgodnie z rys. 2: linia ciągła jest używana, gdy bity wejściowe mają wartość 0, a linia przerywana jest używana, gdy bit wejściowy wynosi 1. Sekwencja bitów, umieszczona obok linii, jest wyjściową parą bitów (wyjściem kodera). Gdy skończą się bity informacyjne, koder wraca najkrótszą ścieżką do stanu S_{00}

Schemat diagramu kratowego to reprezentacja zakodowanej wiadomości w dziedzinie czasu:

- Każde możliwe słowo kodowe jest reprezentowane przez pojedynczą, unikalną ścieżkę przez kratę
- Każde słowo kodu zaczyna się i zatrzymuje w S_{00} (stan zerowy)

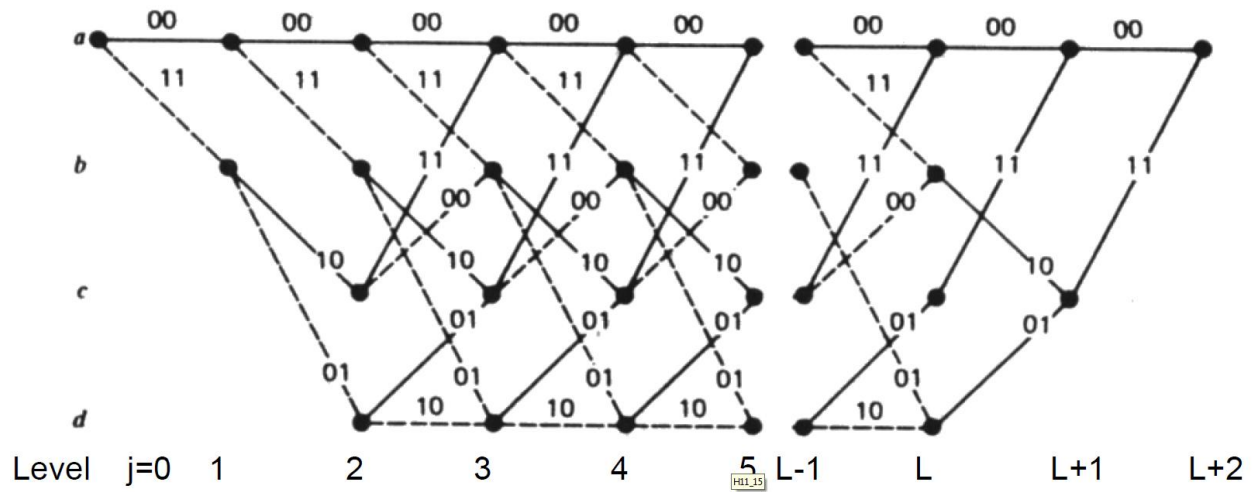


Figure 3: Example of trellis diagram

Symulacja kanału komunikacyjnego z kodowaniem kanałowym

%parametry kanału i modulacji:

```
nbits = 100000;           % liczba bitów do przesłania
M = 16;                   % liczba stanów dla modulacji QAM
k = log2(M);
EbNo = 11;                % stosunek energii n bit do mocy szumu
```

%Definicja wielomianu określającego rodzaj kodu (rate 2/3 code).

```
trellis = poly2trellis([5 4],[23 35 0; 0 5 13]);
codeRate = 2/3;
```

%Generowanie danych wejściowych

```
dataIn = randi([0 1],nbits,1);
```

%Kodowanie splotowe danych wejściowych

```
codeword = convenc(dataIn,trellis);
```

```

%Reshape the encoded column vector into a matrix having k columns. Then,
convert the binary matrix into an integer column vector.

% przygotowanie danych binarnych (kodowanie ochronne)
codewordMat = reshape(codeword,length(codeword)/k,k);
dataSymbolsInCoding = bi2de(codewordMat);

% przygotowanie danych binarnych (bez kodowania ochronnego)

dataInMatrix = reshape(dataIn,length(dataIn)/k,k);    % Reshape data into
dataSymbolsIn = bi2de(dataInMatrix);

%modulacja M-QAM
'gray' - gray (differential) encoding (kodowanie roznicowe)

txSig = qammod(dataSymbolsIn,M,0,'gray');
txSigEC = qammod(dataSymbolsInCoding,M,0,'gray');

%model kanału z szumem (AWGN) dla danego Eb/N0
snr = EbNo + 10*log10(k*codeRate);
rxSig = awgn(txSig,snr,'measured');
rxSigEC = awgn(txSigEC,snr,'measured');

%Demoduacja sygnałów

demodSig = qamdemod(rxSig,M,0,'gray');
demodSigEC = qamdemod(rxSigEC,M,0,'gray');

demodSigMat = de2bi(demodSigEC,k);
demodSigBinaryEC = demodSigMat(:);

%%%% Sciezka dla sygnału z kodem ochronnym %%%%
%Przygotowanie dekodera Viterbiego.

traceBack = 16;

%Dekodowanie

dataOut = vitdec(demodSigBinaryEC,trellis,traceBack,'cont','hard');

%Obliczanie BER (z oknem synchronizacyjnym)

decDelay = 2*traceBack;
[numErrorsEC,berEC] = biterr(dataIn(1:end-decDelay),dataOut(decDelay+1:end));

```

```

%%%%%%%% Sciezka dla sygnalu bez kodu ochronnego %%%%

dataOutMatrix = de2bi(demodSig,k);
dataOut = dataOutMatrix(:); % Return data in column vector

%Obliczanie BER

[numErrors,ber] = biterr(dataIn,dataOut);

% wyswietlanie wynikow na ekranie

fprintf('\nThe binary coding bit error rate = %5.2e, based on %d errors\n',
...
    ber,numErrors)

fprintf('\n[Convolution error coding]The binary coding bit error rate = %5.2e,
based on %d errors\n', ...
    berEC,numErrorsEC)

```

2.1 Ćwiczenia:

- Zmierz współczynnik BER dla E_b/N_0 zmieniającego się w zakresie od 1 do 20 (z krokiem co 1).
- Porównaj uzyskane wyniki z teoretycznymi wartościami BER za pomocą Matlabowego narzędzia: *bertool*.
- Zmierz BER dla QAM4 i QAM64
- Zmierz BER dla innych typów kodera, np.: “7,[171 133]” (długość rejestru pamięci = 7, współczynnik kodowania = $\frac{1}{2}$)
- Porównaj uzyskane wyniki z innymi rodzajami kodów ochronnych (np Kod Hamminga – narzędzie *bertool*.)

