Transmisja Danych – Lab 07

Krystian Bartosik 213A, nr 44266

Kod źródłowy:

```
// Krystian Bartosik
// bk44266@zut.edu.pl
//   FEDCBA
#define _USE_MATH_DEFINES
#include <iostream>
#include <string>
#include <fstream>
#include "math.h"
#include <complex>
#include <cstddef>
#include <bitset>

using namespace std;

double ManchesterSignal[1601];
double NRZISignal[1601];
double BAMISignal[1601];
int M_iterator = 0;
int N_iterator = 0;
int B_iterator = 0;



string S2BS(const char* s, string Endian)
{
    string result = "";

    if (Endian == "BigEndian")
    {
        for (int j = 0; j < strlen(s); j++)
        {
            result = result + bitset<8>(s[j]).to_string();
        }
    }
    // "test" = 01110100011001010111001101110100

    if (Endian == "LittleEndian")
    {
        for (int j = 0; j < strlen(s); j++)
        {
            result = bitset<8>(s[j]).to_string() + result;
        }
    }

    cout << result << endl;
    return result;
}

void Zegar(double Czestotliwosc, double Probkowanie)
{
    fstream File1;
    File1.open("C:/Users/Qrystian/Desktop/results1.txt", ios::out);

    int Check = 0;
    int Bit = 1;
    for (double j = 0; j < Czestotliwosc; j = j + Probkowanie)
```

```
    {
        File1 << j << " " << Bit << endl;

        if (Check == 50)
        {
            Check = 0;
            if (Bit == 0)
                Bit = 1;
            else
                Bit = 0;
        }

        Check++;
    }

    File1.close();
}

double TTL(char t)
{
    double tt = (double)t - '0'; // Konwersja na liczbę

    if (tt == 0)
    {
        return 0.0;
    }

    if (tt == 1)
    {
        return 1.0;
    }
}

double TTLdec(double t)
{
    if (t > 0)
    {
        return 1.0;
    }
    else
    {
        return 0.0;
    }
}

double Manchester(string S, double Probkowanie)
{
    int Previous = 0;
    double tSaved = 0.0;
    fstream File3;
    File3.open("C:/Users/Qrystian/Desktop/results3.txt", ios::out);

    for (int j = 0; j < S.length(); j = j + 1)
    {
        if (((double)S[j] - '0') == 0)
        {
            for (double k = 0; k < 50; k = k + 1)
            {
                File3 << tSaved << " " << 5 << endl;
                ManchesterSignal[M_iterator] = 5;
                M_iterator++;
                tSaved = tSaved + Probkowanie;
            }
```

```
            for (double k = 0; k < 50; k = k + 1)
            {
                File3 << tSaved << " " << -5 << endl;
                ManchesterSignal[M_iterator] = -5;
                M_iterator++;
                tSaved = tSaved + Probkowanie;
            }
        }

        if (((double)S[j] - '0') == 1)
        {
            for (double k = 0; k < 50; k = k + 1)
            {
                File3 << tSaved << " " << -5 << endl;
                ManchesterSignal[M_iterator] = -5;
                M_iterator++;
                tSaved = tSaved + Probkowanie;
            }

            for (double k = 0; k < 50; k = k + 1)
            {
                File3 << tSaved << " " << 5 << endl;
                ManchesterSignal[M_iterator] = 5;
                M_iterator++;
                tSaved = tSaved + Probkowanie;
            }
        }
    }

    File3.close();
    return 0;
}

double Manchesterdec(double * Signal, double Probkowanie)
{
    int Info = 0;
    double tSaved = 0.0;
    bool Free = true;
    fstream File4;
    File4.open("C:/Users/Qrystian/Desktop/results4.txt", ios::out);

    for (int j = 0; j <= M_iterator; j++)
    {
        if ((Signal[j + 1] < 0) && (Signal[j] > 0) && Free)
        {
            Info = 0;
            Free = false;
        }

        if ((Signal[j + 1] > 0) && (Signal[j] < 0) && Free)
        {
            Info = 1;
            Free = false;
        }

        if ((j % 100 == 0) && (j != 0))
        {
            for (int k = 0; k < 100; k++)
            {
                File4 << tSaved << " " << Info << endl;
                tSaved = tSaved + Probkowanie;
            }

            Free = true;
```

```
        }
    }

    File4.close();
    return 0;
}

double NRZI(string S, double Probkowanie)
{
    double tSaved = 0.0;
    double NRZI_Bit = -5;
    fstream File3;
    File3.open("C:/Users/Qrystian/Desktop/results3.txt", ios::out);

    for (int j = 0; j < S.length(); j = j + 1)
    {
        double tt = (double)S[j] - '0'; // Konwersja na liczbę
        if (tt == 0)
        {
            for (double k = 0; k < 100; k = k + 1)
            {
                File3 << tSaved << " " << NRZI_Bit << endl;
                NRZISignal[N_iterator] = NRZI_Bit;
                N_iterator++;
                tSaved = tSaved + Probkowanie;
            }
        }

        if (tt == 1)
        {
            NRZI_Bit = NRZI_Bit * -1;

            for (double k = 0; k < 100; k = k + 1)
            {
                File3 << tSaved << " " << NRZI_Bit << endl;
                NRZISignal[N_iterator] = NRZI_Bit;
                N_iterator++;
                tSaved = tSaved + Probkowanie;
            }
        }

    }

    File3.close();
    return 0;
}

double NRZIdec(double* Signal, double Probkowanie)
{

    double tSaved = 0.0;
    int Info = 0;
    bool Free = true;
    fstream File4;
    File4.open("C:/Users/Qrystian/Desktop/results4.txt", ios::out);

    for (int j = 0; j <= N_iterator; j++)
    {
        if ((Signal[j] != Signal[j + 1]) && (Free))
        {
            Free = false;
            Info = 1;
        }
```

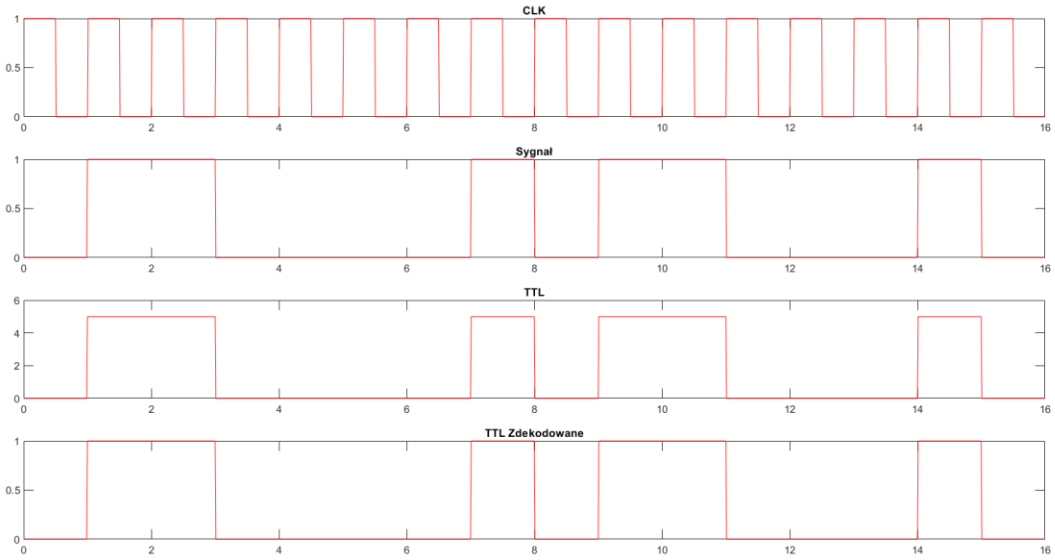
```
        if ((j % 100 == 1) && (j != 0))
        {
            for (int k = 0; k < 100; k++)
            {
                File4 << tSaved << " " << Info << endl;
                tSaved = tSaved + Probkowanie;
            }

            Info = 0;
            Free = true;
        }

    }

    File4.close();
    return 0;
}

double BAMI(string S, double Probkowanie)
{
    double tSaved = 0.0;
    double BAMI_Bit = 5;
    fstream File3;
    File3.open("C:/Users/Qrystian/Desktop/results3.txt", ios::out);

    for (int j = 0; j < S.length(); j = j + 1)
    {
        double tt = (double)S[j] - '0'; // Konwersja na liczbę

        if (tt == 0)
        {
            for (double k = 0; k < 100; k = k + 1)
            {
                File3 << tSaved << " " << 0 << endl;
                BAMISignal[B_iterator] = 0;
                B_iterator++;
                tSaved = tSaved + Probkowanie;
            }
        }

        if (tt == 1)
        {
            if (BAMI_Bit == 5)
            {
                BAMI_Bit = -5;
            }
            else
            {
                BAMI_Bit = 5;
            }

            for (double k = 0; k < 100; k = k + 1)
            {
                File3 << tSaved << " " << BAMI_Bit << endl;
                BAMISignal[B_iterator] = BAMI_Bit;
                B_iterator++;
                tSaved = tSaved + Probkowanie;
            }
        }
    }

    File3.close();
    return 0;
}
```

```
double BAMIdec(double * Signal, double Probkowanie)
{
    double tSaved = 0.0;
    fstream File4;
    File4.open("C:/Users/Qrystian/Desktop/results4.txt", ios::out);

    for (int j = 0; j <= B_iterator; j++)
    {
        if (Signal[j] == 0)
        {
            File4 << tSaved << " " << 0 << endl;
            tSaved = tSaved + Probkowanie;
        }
        else
        {
            File4 << tSaved << " " << 1 << endl;
            tSaved = tSaved + Probkowanie;
        }
    }

    File4.close();
    return 0;
}

int main()
{
    fstream File2;
    fstream File3;
    fstream File4;
    File2.open("C:/Users/Qrystian/Desktop/results2.txt", ios::out);
    File3.open("C:/Users/Qrystian/Desktop/results3.txt", ios::out);
    File4.open("C:/Users/Qrystian/Desktop/results4.txt", ios::out);
    string S = S2BS("ab", "BigEndian");
    double Probkowanie = 0.01;
    Zegar(S.length(), Probkowanie);

    for (double t = 0; t < S.length(); t = t + Probkowanie)
    {
        // TTL - Return to Zero/*
        File2 << t << " " << TTL(S[floor(t)]) << endl;
        File3 << t << " " << TTL(S[floor(t)]) << endl;
        File4 << t << " " << TTLdec( TTL(S[floor(t)]) ) << endl;*/
        // Manchester
        /*
        File2 << t << " " << TTL(S[floor(t)]) << endl;
        if (t == 0) Manchester(S, Probkowanie);
        if (t == 0) Manchesterdec(ManchesterSignal, Probkowanie);
        */
        // NRZI - Non Return to Zero Inverted
        /*
        File2 << t << " " << TTL(S[floor(t)]) << endl;
        if (t == 0) NRZI(S, Probkowanie);
        if (t == 0) NRZIdec(NRZISignal, Probkowanie);
        */
        // BAMI - Bipolar Alternate Mark Inversion
        File2 << t << " " << TTL(S[floor(t)]) << endl;
        if (t == 0) BAMI(S, Probkowanie);
        if (t == 0) BAMIdec(BAMISignal, Probkowanie);
    }

    File2.close();
    File3.close();
    File4.close();
}
```

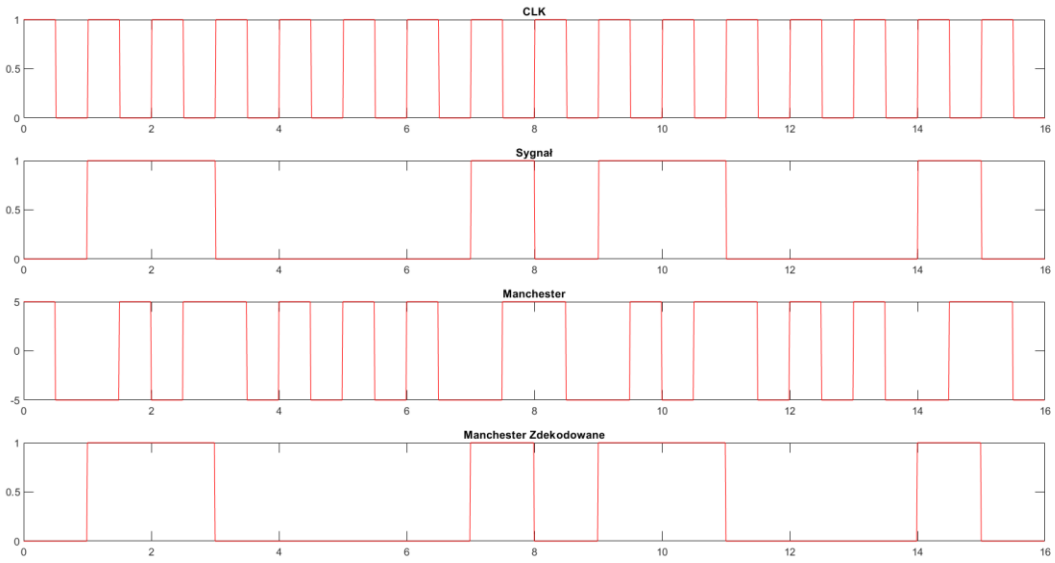
Opis kodu:

- Zegar
- Kod TTL + dekoder
- Kod Manchester + dekoder
- Kod NRZI + dekoder
- Kod BAMI + dekoder

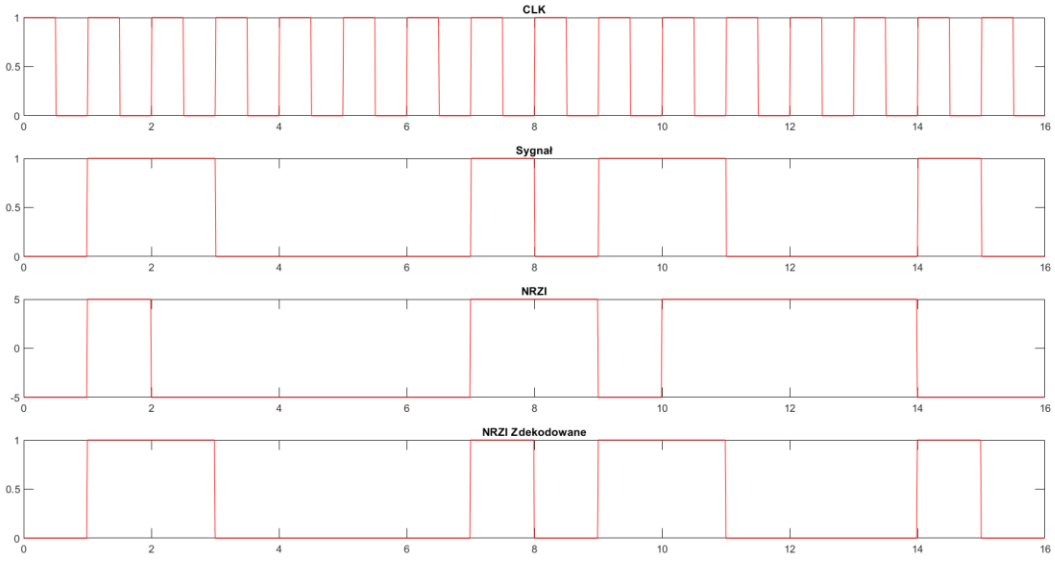
Wygenerowane wykresy (jest ich 4 ponieważ zegar umieściłem na każdym wykresie):



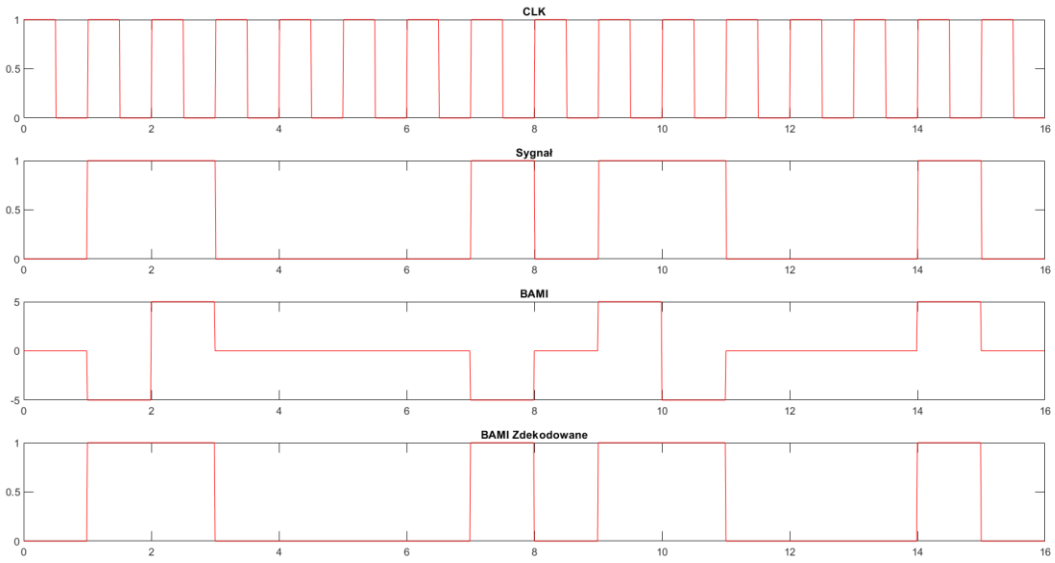
Wykres 1
Zegar + TTL



Wykres 2
Zegar + Manchester



Wykres 3
Zegar + NRZI



Wykres 4
Zegar + BAMI