I International Scientific Practical Conference of graduate and postgraduate students,
lecturers «APPLIED ISSUES OF EXACT SCIENCES»
20-21 October 2017, Armavir

ФИЗИЧЕСКОЕ И ЛОГИЧЕСКОЕ КОДИРОВАНИЕ ИНФОРМАЦИИ

Л.А. Манукян[1)], Д.А. Трухан[2)]

1) студент Армавирского механико-технологического института (филиала) ФГБОУ ВО «Кубанский государственный технологический университет», г.Армавир, Россия, leo.manukian@yandex.ru.ru
2) к.т.н., зав. кафедрой ВЭА Армавирского механико-технологического института (филиала) ФГБОУ ВО «Кубанский государственный технологический университет», г. Армавир, Россия, neozavkaf@gmail.com

Аннотация: в данной статье рассматривалась разработка визуального интерактивного приложения, которая дает возможность осуществить анализ физического и логического кодирования информации в цифровых компьютерных сетях, а также визуально проследить формирование кода для разных систем кодирования.

Ключевые слова: физическое кодирование, логическое кодирование, алгоритм, анализа кодировки, C#.

PHYSICAL AND LOGIC CODING OF INFORMATION

Leo A. Manukian[1)], Dmitriy A. Truhan[2)]

1) the student Armavir Institute of Mechanics and Technology (branch) of Federal State Budgetary Institution of Higher Education “Kuban State Technological University”, city of Armavir, Russia, leo.manukian@yandex.ru.ru
2) Ph. D., associate Professor, Armavir Institute of Mechanics and Technology (branch) of Federal State Budgetary Institution of Higher Education “Kuban State Technological University”, city of Armavir, Russia, neozavkaf@gmail.com

Abstract: this article examined the development of visual interactive app which enables to carry out analysis of physical and logic coding of information in digital computer networks, as well as to visually trace the code generation for different systems codarovaniya.

Key words: physical encoding, logical encoding, algorithm, analysis, coding, C#.

Обучение основам построения сигналов на физическом уровне, а также формирования наглядного представления о физическом и логиче-

ском кодировании информации – первично при изучении дисциплин связанных с передачей информации по компьютерным сетям.

Разрабатываемый программный продукт позволит пользователю осуществлять физическое и логическое кодирование сигнала и анализировать его по модели наглядной визуализации, предлагается построение алгоритма кодирования и применения языка программирования C# для создания программного продукта.

Приложение, кодирующее вводимую с клавиатуры строку, которая представлена в виде последовательности двоичных Unicode-символов методами физического и логического кодирования реализует следующие виды физического кодирования: NRZ, NRZI, MLT-3, AMI, RZ, Манчестерский код, 2B1Q, а также предусматривает задание таких видов логического кодирования как: 4B/5B, Скремблирование, B8ZS, HDB3.

Разработка алгоритма программы

Язык программирования C#, наиболее удобный для реализации поставленной задачи. В программе будут использованы следующие классы данного языка: RadioButton, Timer, PictureBox, RichTextBox, VScrollBar, PrintDocument, Label, GroupBox, MenuStrip, StatusStrip, Button.

Программа написана в виде визуального приложения, состоящего из четырех форм:

Главная форма (Main.cs) – появляться перед пользователем при запуске программы. Содержит две канвы, в одной из которых отображается представление символов строки в виде Unicode-символов, а в другой – наглядное графическое изображение кодирования строки выбранными пользователем методами физического и логического кодирования.

Новый проект (New_project.cs) – вводит строку, подлежащую кодировке, а также выбрать вид физического и логического кодирования строки. Осуществляет передачу данных в главную форму.

Анализ кодировки (Analize.cs) – позволяет пользователю проанализировать эффективность выбранного им кода. Отображает основные параметры кодировки.

Выбор кодировки (Choise.cs) – определяет наиболее эффективную кодировку, соответствующую выбранным пользователем требованиям.

Функция NRZ

Основное предназначение главной формы, код которой хранится в файле Main.cs – реализация функций, осуществляющих логическое и физическое кодирование данных.

В качестве параметра все функции, реализующие логическое и физическое кодирование данных, принимают строковую переменную strbin. В основе построения кода лежит определенное поведение графика функции каждого вида кода при подаче 0 и 1. Используются такие переменные как: l – счетчик цикла, принимающий значения от 0 до номера последнего символа строки; m - величина размера стороны клетки масштабной сетки; i – текущая координата импульса по оси OX; m – текущая координата импульса по оси OY; dy – смещение оси построения импульса относительно

начала координат.

При значении очередного символа строки strbin, равном 1, с помощью функции DrawLine() прорисовывается сначала левая сторона, затем верх и после этого правая сторона импульса. При этом проверяется следующее и предыдущее значение строки strbin, чтобы не построить лишние линии. Перед тем, как прорисовывать верх импульса, необходимо проверить, не будет ли достигнут край канвы. Если такая ситуация возникнет, то построение перейдет на следующую строку.

```
if (strbin[l] == '1')
{
if (l > 0) //если символ не первый
//если предыдущий символ равен 0, рисуем левую сторону импульса
{
      if (strbin[l - 1] == '0') gr.DrawLine(penG, i, m + dy, i, m * 2 + dy);
}
else gr.DrawLine(penG, i, m+dy, i, m * 2+dy);
//проверка не будет ли достигнут край канвы на следующем шаге
if (i+m > pictureBox1.Width) { dy = dy + 3 * m; i = 0; }
gr.DrawString(Convert.ToString(strbin[l]), Num_2, Brushes.Black, i + (int)m / 3, (int)m / 4 + dy);
gr.DrawLine(penG, i, m * 2 + dy, i += m, m * 2 + dy);
//если последний символ, то рисуем правую сторону импульса
if (l == strbin.Length - 1) gr.DrawLine(penG, i, m * 2 + dy, i, m + dy);
else if (strbin[l + 1] == '0') gr.DrawLine(penG, i, m * 2 + dy, i, m + dy);
};
```

Аналогичным образом совершается построение импульса при очередном нулевом значении строки strbin, только алгоритм упрощается за счет того, что не надо проверять следующее и предыдущее значение строки strbin:

```
if (strbin[l] == '0')
{
//проверка не будет ли достигнут край канвы на следующем шаге
if (i+m > pictureBox1.Width) { dy = dy + 3 * m; i = 0; }
gr.DrawString(Convert.ToString(strbin[l]), Num_2, Brushes.Black, i + (int)m / 3, (int)m / 4 + dy);
gr.DrawLine(penG, i, m + dy, i += m, m + dy);
}
```

Проведение анализа кодировки, выбранной пользователем

Основное предназначение формы «Анализ кодировки», код которой хранится в файле Analize.cs – определение эффективности того иного кода, путем проведения анализа выбранной кодировки.

Анализ кодировки, выбранной пользователем, производится по трем критериям:

1. сложность оборудования, реализующего выбранную кодировку

сигнала. Может быть либо простым, либо сложным;

2. уровень самосинхронизации сигнала. Три варианта уровня: низкий, нормальный, высокий;

3. ширина полосы спектра сигнала, от которой зависит пропускная способность линии: чем шире полоса спектра сигнала, тем большая требуется пропускная способность канала связи. Может принимать три значения: узкая (0,25 МГц), средняя (0,5 МГц), широкая (1 МГц).

Считается, что скорость передачи данных по каналу связи равна 100 Мбит/с.

Для обеспечения сравнения каждого сигнала в программе реализована функция blok_analiza(string name), в которой в качестве параметра принимается значение кодировки, используемой пользователем приложения. Наименования кодировок хранятся в строковом массиве name_kod.

Для анализа кодировок в программе реализованы три матрицы, каждая из которых описывает параметры каждой кодировки. Характеристика параметра кодировки обозначается значением 1 соответствующего элемента матрицы.

Матрица Slog_oborud описывает сложность оборудования, реализующего каждую кодировку. Номер столбца соответствует номеру кодировки, номер строки – степени сложности оборудования

```
byte[,] Slog_oborud = {{1,1,1,0,0,1,0}, //простое оборудование
                       {0,0,0,1,1,0,1}}; //сложное оборудование
```

Матрица Urov_sinhr описывает уровень самосинхронизации сигнала. Номер столбца соответствует номеру кодировки, номер строки – уровню самосинхронизации каждого сигнала.

```
byte[,] Urov_sinhr = {{1,0,0,0,0,0,0},//низкий уровень
                      {0,1,1,1,0,0,1},//нормальный уровень
                      {0,0,0,0,1,1,0}};//высокий уровень
```

Матрица Shir_sp описывает ширину спектра сигнала. Номер столбца соответствует номеру кодировки, номер строки – ширина полосы спектра.

```
byte[,] Shir_sp  = {{0,0,1,0,0,0,1},//узкая пососа спектра
                    {1,1,0,1,0,0,0},//средняя полоса спектра
                    {0,0,0,0,1,1,0}};//широкая полоса спектра
```

Приоритетным параметром для анализа является ширина полосы спектра сигнала, так как это качественная характеристика кодировки. Следующим по значимости приоритетом обладает сложность оборудования, а наименьшим – уровень синхронизации. Исходя из этого, были определены коэффициенты для каждого параметра для анализа сигнала. Переменная pokazatel представляет собой сумму коэффициентов каждого параметра кодировки. Её значение сравнивается с пороговым значением. Если это пороговое значение превышено, то сигнал считается эффективным, в противном случае – неэффективным. В последнем случае пользователю предлагается перейти в окно выбора более эффективной кодировки.

```
public void blok_analiza(string name)
{
```

```
    int num_kod = Array.IndexOf(name_kod, name);
    float pokazatel = 0;
    if (Slog_oborud[0, num_kod] == 1) {pokazatel += 0.35F; }
    else if (Slog_oborud[1, num_kod] == 1) {pokazatel += 0.25F; }
```

Выбор эффективной кодировки

Основное предназначение формы «Выбор кодировки», код которой хранится в файле Choice.cs – определение оптимальной кодировки, соответствующей введенным пользователем параметрам.

Выбор эффективной кодировки основан на анализе выбранных пользователем параметров кодировки, которые являются для него наиболее оптимальными.

Для анализа кодировок используются матрицы Slog_oborud, Urov_sinhr, Shir_sp. Выбранные пользователем параметры соответствуют номерам строк в каждой из них.

Для выбора оптимальной кодировки в программе реализована функция choice_kod(), в которой в каждой матрице для каждой кодировки просматривается строка, соответствующая критерию, выбранному пользователем. При обнаружении значения 1 для какой-либо кодировки, в массиве tmp_sch инкрементируется элемент, соответствующий этой кодировке. После прохода по трем матрицам между собой сравниваются элементы массива tmp_sch. Те элементы, которые подходят по максимальному числу критериев, считаются наиболее эффективными.

Результаты тестирования

При запуске программы появляется окно, после ввода всех данных в окне «Новый проект», необходимо нажать ОК, после чего в главной форме появится изображение кодированной строки, представленное на рисунке 2, на котором нанесены соответствующие обозначения. Для анализа используемой кодировки необходимо во вкладке меню «Проект» выбрать пункт «Анализ проекта». В появившемся окне, отображаются параметры кодировки и сведения об ее эффективности.

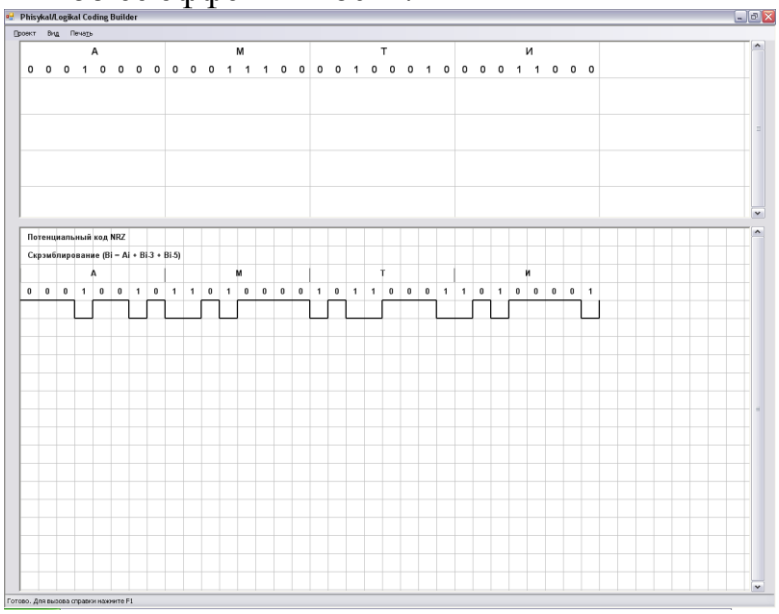

Рисунок 1 – Окно главной формы после ввода параметров

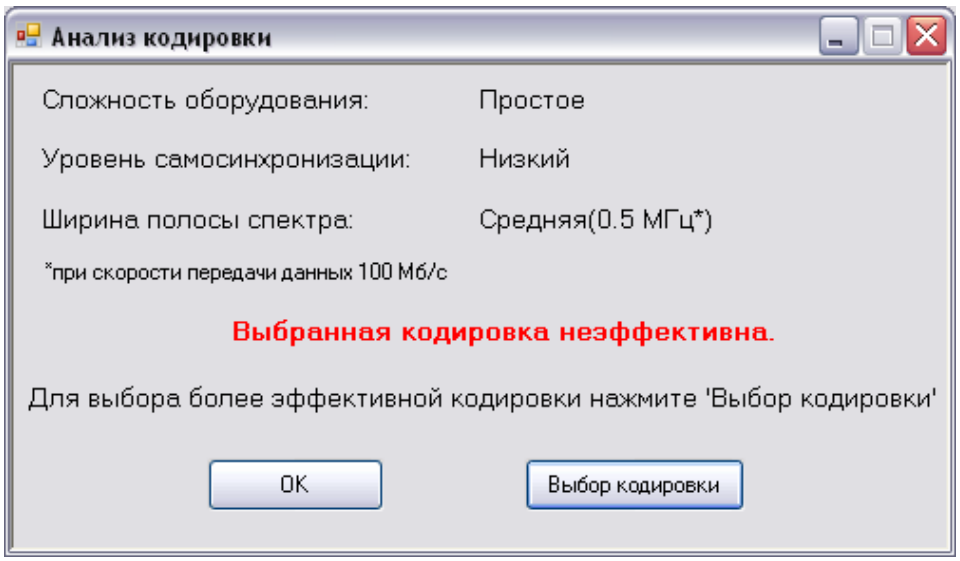

Рисунок 2 – Окно анализа кодировки

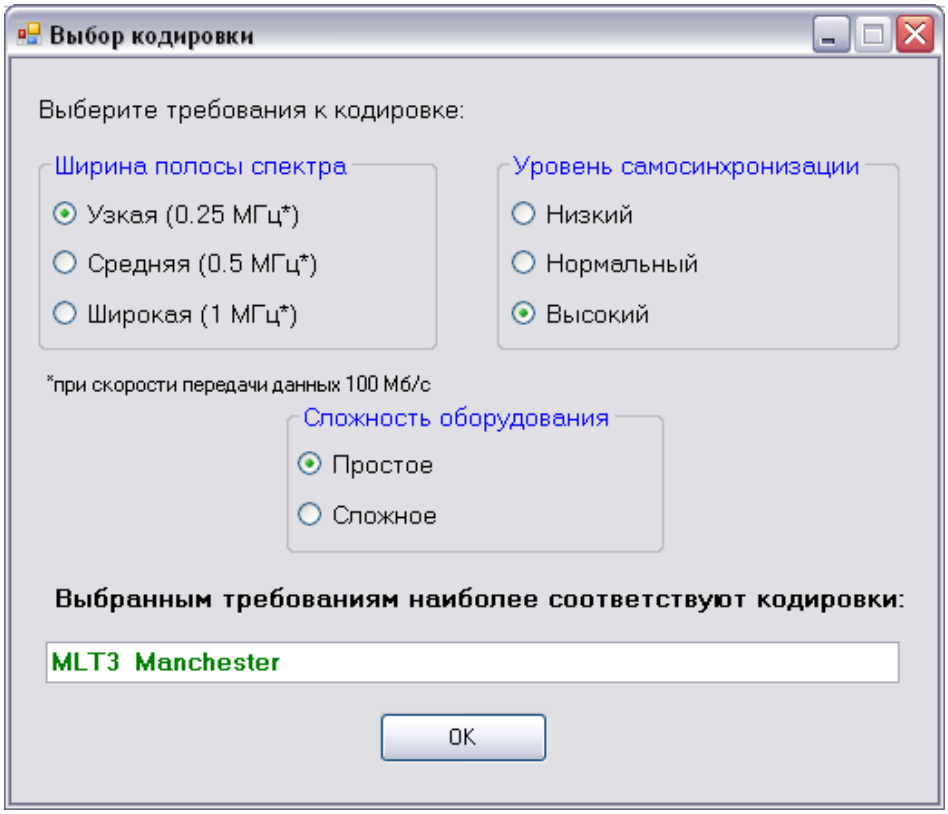

Рисунок 3 – Окно анализа кодировки

Разработаное приложение, осуществляющее визуализацию кодирования символьной строки, может быть использовано при разработке сетевых аппаратных средств для контроля правильности построения цифрового сигнала.

Использование предлагаемой разработки позволит решить задачи изучения студентами методов передачи данных на физическом уровне, а также обеспечения программными продуктами, предназначенных для создания сетевых технологий.

Список использованных источников:

1. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы. – СПб.: Питер,2016. – 672 с.: ил. ISBN 5-8046-0133-4.
2. Таненбаум Э. Компьютерные сети. 4-е изд. – СПб.: Питер, 2003. – 992 с.: ил. ISBN 5-318-00492-X.
3. Трухан Д.А. Визуализация учебной информации в обучении математике, ее значение и роль / Трухан Д.А., Трухан И.А. // Успехи современного естествознания. Изд.: Издательский Дом "Академия Естествознания" 2013. №10. Режим доступа: http://elibrary.ru/item.asp?id=20161681, 0,18 п.л.
4. Трухан Д.А. Высшее профессиональное образование: интеграция общеобразовательной и профессиональной подготовки / Трухан Д.А., Тряпицин Ю.Д., Часов К.В., Коврига Е.В. // Международный журнал экспериментального образования. 2016. №6-1. Режим доступа: http://elibrary.ru/item.asp?id=26008873, 0,12 п.л.
5. Трухан Д.А. Высшее профессиональное образование: интеграция общеобразовательной и профессиональной подготовки. Монография / Трухан Д.А., Тряпицин Ю.Д., Часов К.В., Коврига Е.В. // Кубанский государственный технологический университет. – Краснодар: Изд. ФГБОУ ВПО «КубГТУ», 2015. Режим доступа: http://elibrary.ru/item.asp?id=23778556, 8,25 п.л.
6. Трухан Д.А. Вычислительные системы, сети и телекоммуникации. Методическое пособие по самостоятельной работе студентов специальности 080801 - Прикладная информатика в экономике / Трухан Д.А. // Кубан. гос. технол. ун-т. Армавирский механико-технологический институт. – Армавир: Изд. АМТИ, 2009. Режим доступа: http://elibrary.ru/item.asp?id=28413202, 3,75 п.л.