МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт кибернетики
Базовая кафедра №252 – информационной безопасности

КУРСОВАЯ РАБОТА

По дисциплине
«Сети и системы передачи информации»

Тема курсовой работы
«Код Миллера»

Студент группы ККСО-03-19	Николенко В.О.	_____ *(подпись)*
Руководитель курсовой работы	Чернышев Н.Н.	_____ *(подпись)*

Работа представлена к защите «__» _____ 2022 г.

Допущен к защите «__» _____ 2022 г.

Москва – 2022

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

Различные методы кодирования широко используются в практической деятельности человека с незапамятных времён. Например, десятичная позиционная система счисления - это способ кодирования натуральных чисел. Другой способ кодирования натуральных чисел - римские цифры, причем этот метод более наглядный и естественный, действительно, палец - I, пятерня - V, две пятерни - X. Однако при этом способе кодирования труднее выполнять арифметические операции над большими числами, поэтому он был вытеснен способом кодирования, основанном на позиционных системах счисления, в частности, на десятичной системе счисления.

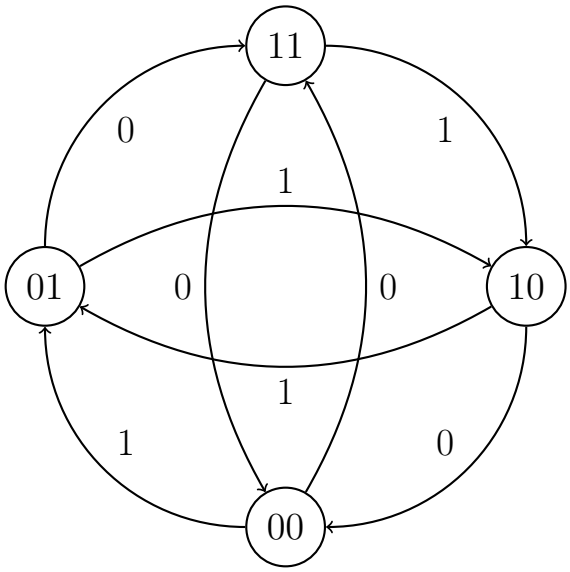
Широко известны способы числового кодирования геометрических объектов и их положения в пространстве: декартовы координаты и полярные координаты, каждый из которых имеет свои особенности.

Из этих примеров можно заключить, что различные способы кодирования обладают присущими только им специфическими особенностями, которые в зависимости от целей кодирования могут быть как достоинством конкретного способа кодирования, так и его недостатком.

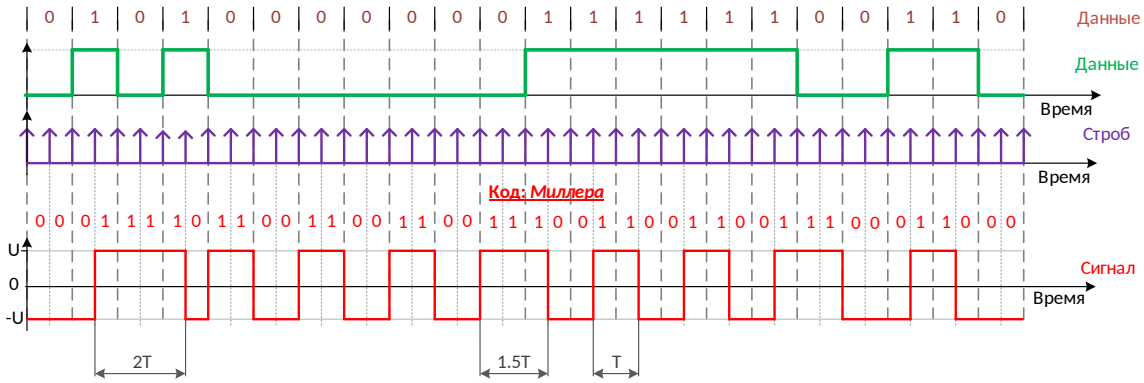
В данной курсовой работе я бы хотел рассмотреть так называемый Код Миллера, который был разработан для налаживания коммуникации между цифровой аппаратурой и физическими каналами связи.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Код Миллера (иногда называют трёхчастотным) — один из способов линейного кодирования(физического кодирования, канального кодирования, импульсно-кодовая модуляция, манипуляция сигнала). Применяется для передачи информации, представленной в цифровом виде от передатчика к приёмнику, например, по оптоволокну. Код, формируемый согласно правилу кода Миллера, является двухуровневым (сигнал может принимать два потенциальных значения, например: высокий и низкий уровень напряжения) кодом, в котором каждый информационный бит кодируется комбинацией из двух значений потенциала, всего таких комбинаций 4: $\{00, 01, 10, 11\}$, а переходы из одного состояния в другое описываются графом:



При непрерывном поступлении логических «нулей» или «единиц» на кодирующее устройство переключение полярности происходит с интервалом T, а переход от передачи «единиц» к передаче «нулей» с интервалом 1,5T. При поступлении на кодирующее устройство последовательности 101 возникает интервал 2T, по этой причине данный метод кодирования называют трёхчастотным. Переход с одного уровня на другой обеспечивает процесс синхронизации передатчика с приёмником, в данном способе передачи осуществляется переключение с одного уровня на другой с минимальной частотой 2T, что обеспечивает синхронизацию передатчика с приёмником. Принцип формирования кода Миллера показан на картинке ниже:



Главная особенность кода Миллера - использование импульсов одной и той же длительности для передачи различных символов. Импульсы тактовой длительности T передают и 0, и 1, а импульс 1,5T - символы 01 и 10. По этой причине могут возникать сбои в канале синхронизации, что ведет к появлению ошибок. Защита от таких сбоев основана на привязке синхроимпульсов к комбинации символов 101 - единственной, для которой выделен импульс 2T.

Давайте рассморим, какие же у такого метода кодирования существуют плюсы:

- Высокая эффективность
- Способность к самосинхронизации[1]
- Полоса пропускания кода Миллера вдвое меньше полосы пропускания в сравнении с манчестерским кодированием[2]. Тем не менее, за время, когда передается один символ исходной последовательности, надо принять решение - 0 это или 1, что приводит к снижению помехозащищенности, а следовательно для избежания этого надо расширять, как минимум вдвое, полосу частот.

Однако, есть и недостатки, например, в данном спсособе кодирования всё же присутствует постоянная составляющая, при этом достаточно велик и низкочастотный компонент, что преодолено в модифицированном коде Миллера в квадрате[3].

---

[1]Способность формировать сигнал (кодировать, манипуляция) в котором происходят детерминированные изменения по которым можно синхронизировать такты формирующего сигнала в передатчике с тактами принимающего сигнала в приёмнике. Такие кодовые последовательности получили название "самосинхронизирующиеся коды".

[2]Один из способов кодирования двоичным цифровым сигналом исходных двоичных данных для передачи по одному двухуровневому каналу связи или записи на носитель информации.

[3]В коде Миллера постоянная составляющая появляется при передаче четного числа единиц между двумя нулями. Модификация данного кода исправляет данный недостаток. В подобных комбинациях попросту вычеркивается последний перепад полярностей. **MFM-кодирование** является двуполярным двухуровневым (сигнал может принимать два значения, соответствующие низкому уровню и высокому уровню) кодом, в котором каждый информационный бит кодируется комбинацией из двух битов.

3. ПРИМЕР ВЫЧИСЛЕНИЙ

Дано:

$\exists$ двоичная последовательность: 11100011011. Тогда закодируем её.

Решение:

Тактовый сигнал должен быть выше частоты поступающей последовательностей в два раза, поскольку каждый бит поступающей последовательности кодируется двумя битами. Далее рассмотрим, с какой вершины нам начинать кодировать. Т.к. первый бит информации у нас 1, то он кодируется комбинацией 01 (если бы был 0, кодировали бы двумя нулями).

Следующая комбинация должна формироваться исходя из следующего входящего символа, он равен 1, следовательно по графу, попадаем в комбинацию 10. Далее мы просто следуем по графу в зависимости от того, какой последующий символ поступает на обработку...

Итоговая закодированная последовательность будет выглядеть: 01 10 01 11 00 11 10 01 11 10 01.

Далее декодируем полученное сообщение:

Т.к. каждый бит был закодирован двумя, то будем всякий раз делать шаг дины 2, а не 1, как это было при кодировании. В самом начале не может быть кодов $\{10, 11\}$. Таким образом, мы начинаем с вершины $\{01\}$ и сразу пишем в новый массив 1(если бы первой парой были $\{00\}$, то в массив был бы помещён 0), далее ориентируемся по графу и замечаем, что следующая пара это $\{10\} \Rightarrow$ в массив пишем вес ребра, которое приведёт нас в эту вершину - 1, на данном этапе наш массив будет выглядеть следующим образом: $\{1, 1\}$. Продолжим наше путешествие по графу, видим следующую пару $\{01\} \Rightarrow$ дополняем массив весом ребра, которое ведёт туда из нашей предыдущей вершины - 1. Таким образом мы гуляем по графу пока наши пары битов не закончатся, и в конце мы получим ровно ту же последовательность, что мы и кодировали ранее.

4. ПРАКТИЧЕСКАЯ ЧАСТЬ

Программа была написана на языке $C++$. И с полной её версией с целью протестировать весь функционал можно ознакомиться на моём $GitHub$. Ниже представлены листинги частей прогаммы: кодирования и декодирования.

Кодирование выглядит следующим образом:

```
vector<bitset<1>> encode(vector<bitset<1>> bits) {
  vector<bitset<1>> coded_bits;
  int position;

  //  -_-_-_setting the first bit-_-_-_
  coded_bits.push_back(0);    // cause we got only
                              // 2 var-s 00 or 01

  if(bits[0] == 0) {
    coded_bits.push_back(0);
    position = 0;
  } else {
    coded_bits.push_back(1);
    position = 1;
  }

  for(int i = 1; i < bits.size(); i++) {
    if(position == 0) {
      if(bits[i] == 0) {      // 00 --> 11
        coded_bits.push_back(1);
        coded_bits.push_back(1);
        position = 11;
      } else {                // 00 --> 01
        coded_bits.push_back(0);
        coded_bits.push_back(1);
        position = 1;
      }
    } else if(position == 1) {
      if(bits[i] == 0) {      // 01 --> 11
        coded_bits.push_back(1);
        coded_bits.push_back(1);
        position = 11;
```

```
33          } else {                    // 01 --> 10
34            coded_bits.push_back(1);
35            coded_bits.push_back(0);
36            position = 10;
37          }
38        } else if(position == 10) {
39          if(bits[i] == 0) {          // 10 --> 00
40            coded_bits.push_back(0);
41            coded_bits.push_back(0);
42            position = 0;
43          } else {                    // 10 --> 01
44            coded_bits.push_back(0);
45            coded_bits.push_back(1);
46            position = 1;
47          }
48        } else if(position == 11) {
49          if(bits[i] == 0) {          // 11 --> 00
50            coded_bits.push_back(0);
51            coded_bits.push_back(0);
52            position = 0;
53          } else {                    // 11 --> 10
54            coded_bits.push_back(1);
55            coded_bits.push_back(0);
56            position = 10;
57          }
58        }
59      }
60      return coded_bits;
61    }
62
```

Декодирование в свою очередь является полностью инвертированным:

```
vector<bitset<1>> decode(vector<bitset<1>> bits) {
  vector<bitset<1>> decoded_bits;
  int position;

  //  -_-_-_setting the first 2 bits-_-_-_
  if(bits[0] == 0) {
    if(bits[1] == 0) {
      decoded_bits.push_back(0);
      position = 0;
    } else {
      decoded_bits.push_back(1);
      position = 1;
    }
  }

  for(int i = 2; i < bits.size(); i += 2) {
    if(position == 0) {
      if(bits[i] == 0) {      // 00 --> 01
        decoded_bits.push_back(1);
        position = 1;
      } else {                // 00 --> 11
        decoded_bits.push_back(0);
        position = 11;
      }
    } else if(position == 1) {
      if(bits[i + 1] == 0) {  // 01 --> 10
        decoded_bits.push_back(1);
        position = 10;
      } else {                // 01 --> 11
        decoded_bits.push_back(0);
        position = 11;
      }
    } else if(position == 10) {
      if(bits[i + 1] == 0) {  // 10 --> 00
        decoded_bits.push_back(0);
        position = 0;
      } else {                // 10 --> 01
        decoded_bits.push_back(1);
```

```
39              position = 1;
40            }
41          } else if(position == 11) {
42            if(bits[i] == 0) {       // 11 --> 00
43              decoded_bits.push_back(0);
44              position = 0;
45            } else {                 // 11 --> 10
46              decoded_bits.push_back(1);
47              position = 10;
48            }
49          }
50        }
51
52        return decoded_bits;
53      }
54
```

Внутри программы использовались библиотеки *vector* и *bitset*, которые позволили структурировать данные и снизить количество потребляемой памяти, т.к. в коде миллера оперируются лишь бинарные данные.

5. ЗАКЛЮЧЕНИЕ

В заключение, хотелось бы отметить то, что Код Миллера хоть и довольно просто объясняется в Теории Графов, однако он необходим для преобразования сообщений в электрические сигналы с помощью этого физического (канального) кодера. При этом кодер устанавливает однозначное соответствие, называемое кодом, между элементами сообщения и элементами сигнала на его выходе (кодовыми символами). Таким образом, он является неотъемлемой частью взаимодействия между цифровыми носителями информации, представленной в цифровом виде, и физическими каналами связи и способствует её адаптации для передачи по оптоволокну, например.

6. СПИСОК ЛИТЕРАТУРЫ

1. Берлин А. Н. Коммутация в системах и сетях связи. — М.: Эко-трендз, 2006. — 344 с. — ISBN 5-88405-073-9.

2. Передача дискретных сообщений: Учебник для вузов/ В. П. Шувалов, Н. В. Захарченко, В. О. Шварцман и др. ; Под ред. В. П. Шувалова. — М.: Радио и связь, —1990—464 ISBN 5-256-00852-8

3. Campbell Parallel Programming with Microsoft Visual C++ / Campbell. - Москва: Гостехиздат, 2011. - 784 с.

4. Слепов Н. Н. Синхронные цифровые сети SDH. — М.: Эко-Трендз, -1998, 148с. ISBN - 5-88405-002-X

5. Гольдштейн Борис Соломонович. Протоколы сети доступа. — БХВ-Петербург. — 2005.