Московский авиационный институт  
(государственный технический университет)

Курсовая работа по дисциплине

«Цифровые системы передачи информации»

Выполнил  
студент группы 04-501  
Маляренко Ю.В.

Проверил  
Рощин А.Е.

Москва, 2008

## Аналого-цифровое преобразование сигналов.

Реализация исходного аналогового сигнала во временной области:

0

1

2

3

4

5

6

7

8

9

10

**t**

**U**

динамический диапазон U =0÷10 В.

Общий вид спектра такого сигнала:

fmax f

σ(f)

Примем fmax равной 24,4 кГц, определим период и частоту дискретизации:

*,*

*-3 c,*

*,*

Изображение дискретного сигнала:

4

5,5

6

6,5

7,2

8

8,7

9,1

9,2

8,7

8,5

8,1

8

7,5

7,2

7,1

6,5

6,3

7

8,5

9

7,5

0

1

2

3

4

5

6

7

8

9

10

0

t

Выберем число уровней квантования L равным 11. Определим шаг квантования Δ:

Выполним квантование полученных дискретных выборок по амплитуде. Для этого значение дискретной выборки будем присваивать одному из уровней квантования, при этом, если значение выборки меньше чем Δ/2, то выборку присваиваем меньшему уровню, если значение выборки больше или равно чем Δ/2, то выборку присваиваем большему уровню.

0

1

2

3

4

5

6

7

8

9

10

округл.

0

+0,5

0

+0,5

-0,2

0

+0,3

-0,1

-0,2

+0,3

+0,5

-0,1

0

+0,5

-0,2

-0,1

+0,5

-0,3

0

+0,5

0

+0,5

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

Найдем разрядность двоичного кода, требуемую для представления полученного многоуровневого квантованного сигнала в двоичном виде:

Таблица соответствия номера уровня двоичной комбинации:

|  |  |
| --- | --- |
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |
| 10 | 1010 |

Преобразуем ранее полученные квантованные выборки в двоичные комбинации, в соответствии с таблицей:

0100 0110 0110 0111 0111 1000 1001 1001 1001 1001 1001 1000 1000 1000 0111 0111 0111 0110 0111 1001 1001 1000.

Определение битовой скорости двоичного потока и объем информации, соответствующий выбранному количеству дискретных значений:

**Рассмотрим дифференциальную импульсно-кодовую модуляцию сигнала (ДИКМ).**

При данной модуляции передаются не текущие значения отсчетов, а разница между соседними отсчетами.

Разностный сигнал будет иметь следующий вид:



Если исходный сигнал имел динамический диапазон U 0÷10В, то сигнал, прошедший через ДИКМ, как видно, имеет динамический диапазон U -2÷4В. Уменьшение динамического диапазона позволяет уменьшить число уровней квантования до 6 при том же шаге квантования 1В.

Квантованный сигнал имеет вид:

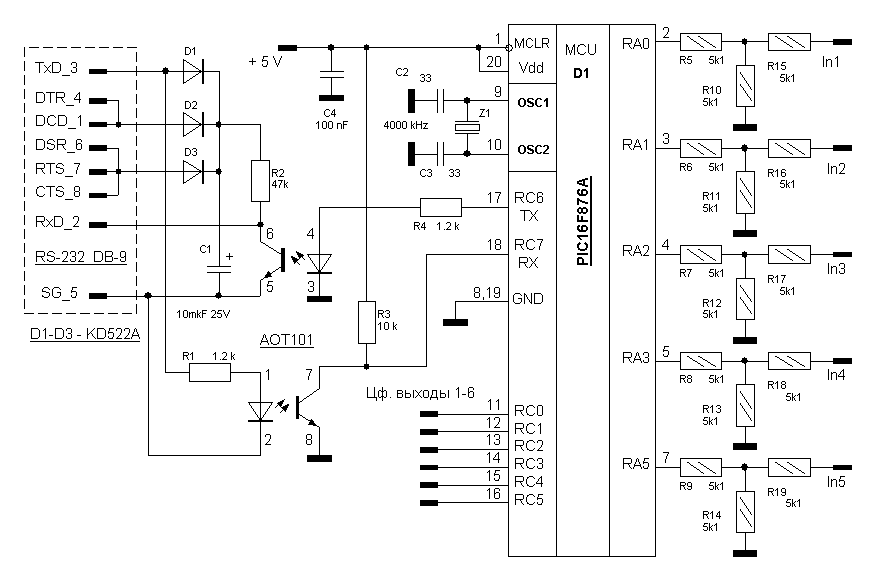


Рассчитаем битовую скорость кодированного методом ДИКМ сигнала:

Сравним ее со скоростью, полученной при простой ИКМ:

За счет уменьшения уровней квантования, мы получили меньший объем передаваемой информации.

## Вариант реализации АЦП.



Устройство собрано на PIC16F876A. Делители (R10-R19) определяют ширину диапазона. Подбор делителей на входе устройства позволяет измерять сигналы в широком диапазоне. Программно можно корректировать смещение сигнала +/- в случае погрешности номиналов сопротивлений делителя.

Точность измерения определяется по формуле:

Например, если делителями задан диапазон 10 В, то точность составляет

**10 / 1023 = 0,0097 В**

## Кодирование источника, сжатие двоичной информации.

**Рассмотрим алгоритм Шенона-Фано.**

Для имеющейся последовательности

0100 0110 0110 0111 0111 1000 1001 1001 1001 1001 1001 1000 1000 1000 0111 0111 0111 0110 0111 1001 1001 1000

применим статистический алгоритм сжатия Шенона-Фано.

Определим статистику повторения комбинаций в последовательности:

|  |  |
| --- | --- |
| 0100 | 1 |
| 0110 | 3 |
| 0111 | 6 |
| 1000 | 5 |
| 1001 | 7 |

Построим дерево алгоритма



Полученная кодировка:

|  |  |
| --- | --- |
| 0100 | 011 |
| 0110 | 010 |
| 0111 | 11 |
| 1000 | 10 |
| 1001 | 00 |

Запишем исходную последовательность в новой кодировке:

011 010 010 11 11 10 00 00 00 00 00 10 10 10 11 11 11 010 11 00 00 10

Определим коэффициент сжатия:

**Применим статистический алгоритм сжатия Хаффмана.**

Определим статистику повторения комбинаций в последовательности:

|  |  |
| --- | --- |
| 0100 | 1 |
| 0110 | 3 |
| 0111 | 6 |
| 1000 | 5 |
| 1001 | 7 |

Построим дерево алгоритма:



Полученная кодировка:

|  |  |
| --- | --- |
| 0100 | 111 |
| 0110 | 110 |
| 0111 | 01 |
| 1000 | 00 |
| 1001 | 10 |

Запишем исходную последовательность в новой кодировке:

111 110 110 01 01 00 10 10 10 10 10 00 00 00 01 01 01 110 01 10 10 00

Определим коэффициент сжатия:

**Применим арифметическое кодирование (реализация IBM).**

Кодирование происходит по следующему алгоритму: каждому символу сопоставляется его вес: вначале он для всех равен 1. Все символы располагаются в естественном порядке, например, по возрастанию. Вероятность каждого символа устанавливается равной его весу, деленному на суммарный вес всех символов. После получения очередного символа и постройки интервала для него, вес этого символа увеличивается на 1.

Рассчитаем вероятности получения следующего символа, с учетом пришедших ранее символов и границы отрезка.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Номер итерации | Веса символов | | | | | Вероятность символа | Граница интервала |
| 0100 | 0110 | 0111 | 1000 | 1001 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1/5 | (0;1/5) |
| 2 | 2 | 1 | 1 | 1 | 1 | 1/6 | (2/30;3/30) |
| 3 | 2 | 2 | 1 | 1 | 1 | 2/7 | (16/210;18/210) |
| 4 | 2 | 3 | 1 | 1 | 1 | 1/8 | (69/840;70/840) |
| 5 | 2 | 3 | 2 | 1 | 1 | 2/9 | (626/7560;628/7560) |
| 6 | 2 | 3 | 3 | 1 | 1 | 1/10 | (3238/38800;3239/38800) |
| 7 | 2 | 3 | 3 | 2 | 1 | 1/11 | (35628/415800;35629/415800) |
| 8 | 2 | 3 | 3 | 2 | 2 | 2/12 | (427546/4989800;427548/4989800) |
| 9 | 2 | 3 | 3 | 2 | 3 | 3/13 | (2779059/16216200;2779062/16216200) |
| 10 | 2 | 3 | 3 | 2 | 4 | 4/14 | (12968952/75675600;12968956/75675600) |
| 11 | 2 | 3 | 3 | 2 | 5 | 5/15 | (48633580/283783500;  48633585/283783500) |
| 12 | 2 | 3 | 3 | 2 | 6 | 2/16 | (155627464/454053600;  155627466/454053600) |
| 13 | 2 | 3 | 3 | 3 | 6 | 3/17 | (1322833452/3859455600;  1322833455/3859455600) |
| 14 | 2 | 3 | 3 | 4 | 6 | 4/18 | (7937000720/23156733600;  7937000724/23156733600) |
| 15 | 2 | 3 | 3 | 5 | 6 | 3/19 | (37700753425/109994484600;  37700753428/109994484600) |
| 16 | 2 | 3 | 4 | 5 | 6 | 4/20 | (251338356171/733296564000;  251338356175/733296564000) |
| 17 | 2 | 3 | 5 | 5 | 6 | 5/21 | (1319526369902/3849806961000;  1319526369907/3849806961000) |
| 18 | 2 | 3 | 6 | 5 | 6 | 3/22 | (5805916027571/16939150628400;  5805916027574/16939150628400) |
| 19 | 2 | 4 | 6 | 5 | 6 | 6/23 | (4451202287810/129866821484400;  4451202287816/129866821484400) |
| 20 | 2 | 4 | 7 | 5 | 6 | 6/24 | (17804809151258/519467285937600;  17804809151264/519467285937600) |
| 21 | 2 | 4 | 7 | 5 | 7 | 7/25 | (74186704796326/2164447024740000;  74186704796333/2164447024740000) |
| 22 | 2 | 4 | 7 | 5 | 8 | 2/26 | (27550617814927/80393746633200000;  27550617814927/80393746633200000) |
| 23 | 2 | 4 | 7 | 6 | 8 | 6/27 | (371933340501547/24644094144309000000;  371933340501553/24644094144309000000  ) |

Возьмем число 371933340501550/24644094144309000000, принадлежащее последнему интервалу и переведем его в двоичный вид, в результате чего получим число

0.00000000000000000000000000000000000000000000001011010000100100011010000111111111100100111111101111

**Рассмотрим адаптивный алгоритм Хаффмана.**

Адаптивная модель позволяет не передавать модель сообщения вместе с ним самим и ограничиться всего одним проходом по сообщению как при кодировании, так и при декодировании.  
Для реализации адаптивного алгоритма Хаффмана необходимо работать с упорядоченным деревом кодирования. (Упорядоченным называется дерево, все узлы которого могут быть перечислены в порядке возрастания веса и в этом перечислении каждый узел находится рядом со своим братом.) Обновление дерева при считывании нового символа состоит из двух операций: увеличиваем вес узла, увеличивая при этом вес его родителя, доходя до корня дерева, и перестановка узлов, когда это необходимо для сохранения свойств упорядоченности.

Моделирование будем начинать с пустого дерева, добавляя в него новые символы по мере их появления в сообщении, символ, появившейся впервые, будем передавать не кодируя, добавляя специальный ЕSC-код, показывающий декодеру, что символ передан без кодирования. Также имеется EOF-код, показывающий декодеру, что достигнут конец сообщения.

Дерево алгоритма:



Таблица кодировки:

|  |  |  |
| --- | --- | --- |
| Символ | Число повторов | Код |
| 0100 | 1 | 0100 |
| 0110 | 3 | 0001 |
| 0111 | 6 | 001 |
| 1000 | 5 | 01 |
| 1001 | 7 | 1 |
| ESC | 5 | 000001 |
| EOF | 1 | 000000 |

Кодированная последовательность:

000001 0100 000001 0110 0001 000001 0111 001 000001 1000 000001 1001 1 1 1 1 01 01 01 001 001 001 0001 001 1 1 01 000000

Лучшими являются и , равные 1,83. Для дальнейших преобразований выберем последовательность, полученную в результате работы метода Хаффмана.

## Защита информации, шифрование двоичной последовательности.

Поточное шифрование на основе регистра сдвига с обратными связями. В качестве ключа выбрано начальное заполнение 1000. Вектор связи регистра 0011.



Построим таблицу ключевой последовательности:

|  |  |  |
| --- | --- | --- |
| N | Состояние РС | Выход |
| 0 | 1000 |  |
| 1 | 0100 | 0 |
| 2 | 0010 | 0 |
| 3 | 1001 | 0 |
| 4 | 1100 | 1 |
| 5 | 0110 | 0 |
| 6 | 1011 | 0 |
| 7 | 0101 | 1 |
| 8 | 1010 | 1 |
| 9 | 1101 | 0 |
| 10 | 1110 | 1 |
| 11 | 1111 | 0 |
| 12 | 0111 | 1 |
| 13 | 0011 | 1 |
| 14 | 0001 | 1 |
| 15 | 1000 | 1 |

γ=000100110101111, период N=15.

Шифрование открытого текста посредством полученной последовательности.

111110110010100101010101000000001010111001101000  
+  
000100110101111000100110101111000100110101111000  
=  
111010000111011101110011101111001110001100010000

Данная схема является очень уязвимой по отношению к атакам. Чтобы определить вектор связи регистра, начальное состояние и всю последовательность кода, криптоаналитику требуется 2n бит открытого текста и соответствующий шифртекст.  
 Пусть криптоаналитику это удалось:

Открытый текст: 11011111

Шифрованный текст: 00010111

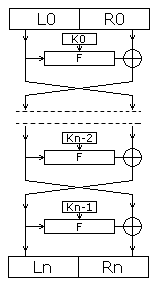
Крайний правый бит получен первым. Чтобы получить фрагмент ключевого потока 11001000, криптоаналитик складывает обе последовательности по модулю 2. Ключевой поток показывает содержание РС в различные моменты времени, причем крайние правые 4 бита — начальное состояние РС. Если их последовательно сдвигать влево, то получим содержимое регистра в моменты *t2, t3, t4.* Используя линейную структуру РС можно записать следующее:

*,*

где — цифра, поданная через контур обратной связи обратно на вход, а (=1 или 0) определяет *i*-ое состояние обратной связи. Составляем 4 уравнения с четырьмя неизвестными:

Решением этой системы является Таким образом, зная начальное состояние РС и его внутренние связи, криптоаналитик может узнать последовательность в любой момент времени.

Лучшую устойчивость будет иметь шифрование через **сеть Фейстеля**.  
Она реализуется по следующим принципам:

* Вся информация разбивается на [блоки фиксированной длины](http://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80). В случае, если длина входного блока меньше, чем размер, который шифруется заданным [алгоритмом](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC), то блок удлиняется каким-либо способом. Как правило длина блока является [степенью](http://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B7%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2_%D1%81%D1%82%D0%B5%D0%BF%D0%B5%D0%BD%D1%8C) двойки, например : 64 [бита](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82), 128 [бит](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82). Далее будем рассматривать операции происходящие только с одним блоком, так как с другими в процессе шифрования выполняются те же самые операции.
* Выбранный блок делится на два равных подблока — «левый» (*L*0) и «правый» (*R*0).
* «Левый блок» *L*0 видоизменяется функцией *f*(*L*0,*K*1) в зависимости от ключа *K*1, после чего он [складывается по модулю 2](http://ru.wikipedia.org/wiki/Xor) с «правым блоком» *R*0.
* Результат сложения присваивается новому левому подблоку *L*1, который будет половиной входных данных для следующего раунда, а «левый блок» *L*0 присваивается без изменений новому правому подблоку*R*1(см. схему), который будет другой половиной.
* После чего операция повторяется N-1 раз, при этом при переходе от i-го к i+1-му этапу могут меняться ключи(*Ki* на *Ki* + 1) по какому-либо правилу, где N — количество раундов в заданном алгоритме.

Примем, что будут шифроваться блоки длинной 64 бит, исходный текст будет расширяться до данного значения путем добавления нужного количества нулей.  
Первоначальный ключ выбирается случайным образом из некоего подмножества ключей К, далее он меняется циклическим сдвигом влево. Функция изменения ключом – сложение по модулю 2.

На первой итерации будем иметь:

Начальный текст: 111110110010100101010101000000001010111001101000 (48 бит).

Расширенный текст: 1111101100101001010101010000000010101110011010000000000000000000  
(64 бит).

Первоначальный ключ:  
00110110110111010110001000000111  
(32 бит).

Изменение ключом левого блока:

11111011001010010101010100000000  
+  
00110110110111010110001000000111  
=  
11001101111101000011011100000111.

Сложение измененного левого и правого блоков:

11001101111101000011011100000111  
+  
10101110011010000000000000000000  
=  
01100011100111000011011100000111.

Текст для второй итерации:  
0110001110011100001101110000011111111011001010010101010100000000.

**Достоинства:**

* Простота аппаратной реализации на современной электронной базе
* Простота программной реализации в силу того, что все значительная часть функций поддерживаются на аппаратном уровне в современных комьютерах
* Хорошая изученность алгоритмов на основе сетей Фейстеля[[5]](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D1%8C_%D0%A4%D0%B5%D0%B9%D1%81%D1%82%D0%B5%D0%BB%D1%8F#cite_note-4)

**Недостатки:**

* За один раунд шифруется только половина входного блока

## Шифрование по алгоритму DES.

**DES** (*Data Encruption Standart*) — Симметричный алгоритм шифрования, в котором один ключ используется как и для шифрования так и для расшифрования данных. DES разрабртан фирмой [IBM](http://ru.wikipedia.org/wiki/IBM) и утвержен правительством [США](http://ru.wikipedia.org/wiki/%D0%A1%D0%A8%D0%90) в 1977 году как официальный стандарт. DES имеет блоки по 64 бит и 16 цикловую структуру [сети Фейстеля](http://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B8_%D0%A4%D0%B5%D0%B9%D1%81%D1%82%D0%B5%D0%BB%D1%8F&action=edit&redlink=1), для шифрования использует ключ с длиной 56 бит.

Исходный текст расширяеи до 64 бит добавлением 0.

Начальный ключ: 11101100111000100110000011011100010101001000001111011101011110

Обозначения:

CD – блоки ключа до перестановки  
KS – ключ для конкретной итерации  
Е – блок текста после функции расширения

Input bits: 11111011 00101001 01010101 00000000 10101110 01101000 00000000 00000000  
Key bits: 00111011 00111000 10011000 00110111 00010101 00100000 11110111 01011110  
CD[0]: 0100010 0110000 0001101 0111101 1100100 1110110 0010000 1111111  
CD[1]: 1000100 1100000 0011010 1111010 1001001 1101100 0100001 1111111  
KS[1]: 010111 000000 100001 001100 010101 011000 111101 001111  
CD[2]: 0001001 1000000 0110101 1110101 0010011 1011000 1000011 1111111  
KS[2]: 010100 010010 110111 110000 011001 001001 011111 001100  
CD[3]: 0100110 0000001 1010111 1010100 1001110 1100010 0001111 1111100  
KS[3]: 110101 001110 010010 000101 110110 001011 010011 101111  
CD[4]: 0011000 0000110 1011110 1010001 0111011 0001000 0111111 1110010  
KS[4]: 010100 111000 011100 000110 011011 101101 111010 101001  
CD[5]: 1100000 0011010 1111010 1000100 1101100 0100001 1111111 1001001  
KS[5]: 011010 001001 000010 100111 000110 100111 110101 111011  
CD[6]: 0000000 1101011 1101010 0010011 0110001 0000111 1111110 0100111  
KS[6]: 101100 011000 000001 101110 101011 111101 100100 110000  
CD[7]: 0000011 0101111 0101000 1001100 1000100 0011111 1111001 0011101  
KS[7]: 101000 000100 001010 110010 110000 010110 111101 110010  
CD[8]: 0001101 0111101 0100010 0110000 0010000 1111111 1100100 1110110  
KS[8]: 101101 000001 101100 110100 111111 011000 101000 011100  
CD[9]: 0011010 1111010 1000100 1100000 0100001 1111111 1001001 1101100  
KS[9]: 001000 101101 110101 000010 100100 111000 011001 111100  
CD[10]: 1101011 1101010 0010011 0000000 0000111 1111110 0100111 0110001  
KS[10]: 011010 000110 000101 010111 110110 011011 111110 000100  
CD[11]: 0101111 0101000 1001100 0000011 0011111 1111001 0011101 1000100  
KS[11]: 001001 011100 010100 011001 001110 000110 011010 111101  
CD[12]: 0111101 0100010 0110000 0001101 1111111 1100100 1110110 0010000  
KS[12]: 010001 110000 000110 110011 011110 110111 100010 000111  
CD[13]: 1110101 0001001 1000000 0110101 1111111 0010011 1011000 1000011  
KS[13]: 101111 111000 100010 010001 101001 100110 000110 111011  
CD[14]: 1010100 0100110 0000001 1010111 1111100 1001110 1100010 0001111  
KS[14]: 000111 110010 001010 001010 101001 110011 101101 000111  
CD[15]: 1010001 0011000 0000110 1011110 1110010 0111011 0001000 0111111  
KS[15]: 001110 100001 010010 011100 111101 101000 001111 110010  
CD[16]: 0100010 0110000 0001101 0111101 1100100 1110110 0010000 1111111  
KS[16]: 000100 010111 110010 000001 110101 111110 000101 001110  
L[0]: 00100101 00000101 00010100 00000111  
R[0]: 00010001 00110011 00110011 00010001  
Round 1  
E : 100010 100010 100110 100110 100110 100110 100010 100010  
KS : 010111 000000 100001 001100 010101 011000 111101 001111  
E xor KS: 110101 100010 000111 101010 110011 111110 011111 101101  
Sbox: 0011 1110 1001 1011 1111 0110 0110 1000  
P : 11101101 01111100 00000111 11011010  
L[i]: 00010001 00110011 00110011 00010001  
R[i]: 11001000 01111001 00010011 11011101  
Round 2  
E : 111001 010000 001111 110010 100010 100111 111011 111011  
KS : 010100 010010 110111 110000 011001 001001 011111 001100  
E xor KS: 101101 000010 111000 000010 111011 101110 100100 110111  
Sbox: 0001 0001 0101 1101 0100 0011 1011 0000  
P : 10000110 00100101 01110100 01000011  
L[i]: 11001000 01111001 00010011 11011101  
R[i]: 10010111 00010110 01000111 01010010  
Round 3  
E : 010010 101110 100010 101100 001000 001110 101010 100101  
KS : 110101 001110 010010 000101 110110 001011 010011 101111  
E xor KS: 100111 100000 110000 101001 111110 000101 111001 001010  
Sbox: 0010 0000 1011 1010 1110 0100 1110 1111  
P : 00001101 01010110 00001111 11101101  
L[i]: 10010111 00010110 01000111 01010010  
R[i]: 11000101 00101111 00011100 00110000  
Round 4  
E : 011000 001010 100101 011110 100011 111000 000110 100001  
KS : 010100 111000 011100 000110 011011 101101 111010 101001  
E xor KS: 001100 110010 111001 011000 111000 010101 111100 001000  
Sbox: 1011 1000 1011 1011 0110 1101 1001 0110  
P : 10010110 11001110 00100011 11101111  
L[i]: 11000101 00101111 00011100 00110000  
R[i]: 00000001 11011000 01100100 10111101

Round 5  
E : 100000 000011 111011 110000 001100 001001 010111 111010  
KS : 011010 001001 000010 100111 000110 100111 110101 111011  
E xor KS: 111010 001010 111001 010111 001010 101110 100010 000001  
Sbox: 1010 1011 1011 1100 1010 0011 0100 0001  
P : 01000101 10111000 01111011 11000100  
L[i]: 00000001 11011000 01100100 10111101  
R[i]: 10000000 10010111 01100111 11110100  
Round 6  
E : 010000 000001 010010 101110 101100 001111 111110 101001  
KS : 101100 011000 000001 101110 101011 111101 100100 110000  
E xor KS: 111100 011001 010011 000000 000111 110010 011010 011001  
Sbox: 0101 0110 1000 0111 1100 0000 1010 0000  
P : 11000001 01000100 10010101 00010011  
L[i]: 10000000 10010111 01100111 11110100  
R[i]: 11000000 10011100 11110001 10101110  
Round 7  
E : 011000 000001 010011 111001 011110 100011 110101 011101  
KS : 101000 000100 001010 110010 110000 010110 111101 110010  
E xor KS: 110000 000101 011001 001011 101110 110101 001000 101111  
Sbox: 1111 0100 1100 1111 1000 0001 1111 1101  
P : 10001011 11010001 10111111 01110011  
L[i]: 11000000 10011100 11110001 10101110  
R[i]: 00001011 01000110 11011000 10000111  
Round 8  
E : 100001 010110 101000 001101 011011 110001 010000 001110  
KS : 101101 000001 101100 110100 111111 011000 101000 011100  
E xor KS: 001100 010111 000100 111001 100100 101001 111000 010010  
Sbox: 1011 1010 1001 1100 0001 1001 0000 1001  
P : 01111100 10001000 00111011 01000010  
L[i]: 00001011 01000110 11011000 10000111  
R[i]: 10111100 00010100 11001010 11101100  
Round 9  
E : 010111 111000 000010 101001 011001 010101 011101 011001  
KS : 001000 101101 110101 000010 100100 111000 011001 111100  
E xor KS: 011111 010101 110111 101011 111101 101101 000100 100101  
Sbox: 1000 0001 0011 0001 0101 1111 0010 1110  
P : 10111100 10100110 01100100 00101100  
L[i]: 10111100 00010100 11001010 11101100  
R[i]: 10110111 11100000 10111100 10101011  
Round 10  
E : 110110 101111 111100 000001 010111 111001 010101 010111  
KS : 011010 000110 000101 010111 110110 011011 111110 000100  
E xor KS: 101100 101001 111001 010110 100001 100010 101011 010011  
Sbox: 0010 0011 1011 0101 1011 1110 0100 0101  
P : 11110101 00110000 01011011 10101100  
L[i]: 10110111 11100000 10111100 10101011  
R[i]: 01001001 00100100 10010001 01000000  
Round 11  
E : 001001 010010 100100 001001 010010 100010 101000 000000  
KS : 001001 011100 010100 011001 001110 000110 011010 111101  
E xor KS: 000000 001110 110000 010000 011100 100100 110010 111101  
Sbox: 1110 0100 1011 0001 1110 1111 1111 0110  
P : 10010111 10110110 10100111 10111101  
L[i]: 01001001 00100100 10010001 01000000  
R[i]: 00100000 01010110 00011011 00010110  
Round 12  
E : 000100 000000 001010 101100 000011 110110 100010 101100  
KS : 010001 110000 000110 110011 011110 110111 100010 000111  
E xor KS: 010101 110000 001100 011111 011101 000001 000000 101011  
Sbox: 1100 0101 1111 1001 1000 1010 0100 1010  
P : 10011101 10110011 11000001 01010100  
L[i]: 00100000 01010110 00011011 00010110  
R[i]: 11010100 10010111 01010000 00010100  
Round 13  
E : 011010 101001 010010 101110 101010 100000 000010 101001  
KS : 101111 111000 100010 010001 101001 100110 000110 111011  
E xor KS: 110101 010001 110000 111111 000011 000110 000100 010010  
Sbox: 0011 1100 1011 1110 1011 1111 0010 1001  
P : 00111101 01101000 00111111 11011110  
L[i]: 11010100 10010111 01010000 00010100  
R[i]: 00011101 00111110 00100100 11001000  
Round 14  
E : 000011 111010 100111 111100 000100 001001 011001 010000  
KS : 000111 110010 001010 001010 101001 110011 101101 000111  
E xor KS: 000100 001000 101101 110110 101101 111010 110100 010111  
Sbox: 1101 0110 1000 1110 0010 1101 0110 1011  
P : 01011000 11010010 10111101 11011010  
L[i]: 00011101 00111110 00100100 11001000  
R[i]: 10001100 01000101 11101101 11001110  
Round 15  
E : 010001 011000 001000 001011 111101 011011 111001 011101  
KS : 001110 100001 010010 011100 111101 101000 001111 110010  
E xor KS: 011111 111001 011010 010111 000000 110011 110110 101111  
Sbox: 1000 0000 0100 1100 0010 1110 1000 1101  
P : 00011000 10100001 00011000 11101001  
L[i]: 10001100 01000101 11101101 11001110  
R[i]: 00000101 10011111 00111100 00100001  
Round 16  
E : 100000 001011 110011 111110 100111 111000 000100 000010  
KS : 000100 010111 110010 000001 110101 111110 000101 001110  
E xor KS: 100100 011100 000001 111111 010010 000110 000001 001100  
Sbox: 1110 0101 1101 1110 0101 1111 1101 1011  
P : 00111110 11110111 11111011 01011001  
L[i]: 00000101 10011111 00111100 00100001  
R[i]: 10110010 10110010 00010110 10010111  
LR[16] 10110010 10110010 00010110 10010111 00000101 10011111 00111100 00100001  
Output 10100011 01110101 10101101 00101000 01111101 01011010 00000000 01110001

## Канальное (помехоустойчивое) кодирование.

## Код Хэмминга.

Используем код Хэмминга (7,4), т.е. блок длинной 7 символов, из которых 4 разряда – информационные и 3 – контрольные. Для построения блоков необходимо получить производящую матрицу G = (), где - единичная матрица, а - матрица, столбцы которой задаются урвнениями:

G=

Перемножая построчно G с блоком информационной комбинации и складывая затем строки по модулю 2, получаем кодовую комбинацию:

ИК: 1110 1000 0111 0111 0111 0011 1011 1100 1110 0011 0001 0000  
КК: 1110100 1000101 0111010 0111010 0111010 0011101 1011000 1100010 1110100 0011101 0001011 0000000

Битовая скорость R равна:

,

где m – разрядность.

,

Кодовым расстоянием d для кода, содержащего m кодовых комбинаций, является минимальное расстояние между всеми парами кодовых комбинаций. Для кода Хэмминга d=3.  
Кратность обнаруживаемой ошибки:

Кратность исправляемой ошибки:

Декодирование и обнаружение/исправление ошибки.

Построим проверочную матрицу Хэмминга H=().

Введем ошибку во второй разряд комбинации 1110100 1010100.  
Перемножая столбец с разрядом комбинации, получим проверочную матрицу для комбинации:

Получим вектор S, путем сложения столбцов по модулю 2:

Ненулевой вектор показывает что есть ошибка. Находим такой столбец в проверочной матрице, его номер указывает на номер разряда в кодовой комбинации, в котором произошла ошибка, в данном случае – во втором. Изменяем значение второго бита и получаем 1110100, далее получаем проверочную матрицу для той комбинации:

Вектор S для нее:

,

что говорит о том, что ошибки нет и можно отбросить проверочные разряды.

## Код Хэмминга, заданный как циклический код.

Используем код Хэмминга, заданный как циклический код с образующим полиномом G(x)=x3+x2+1.  
Переведем исходные блоки в полиномиальный вид:

|  |  |
| --- | --- |
| Последовательность | Полином |
| 1110 | x3+x2+x |
| 1000 | x3 |
| 0111 | x2+x+1 |
| 0111 | x2+x+1 |
| 0111 | x2+x+1 |
| 0011 | x+1 |
| 1011 | x3+x+1 |
| 1100 | x3+x2 |
| 1110 | x3+x2+x |
| 0011 | x+1 |
| 0001 | 1 |
| 0000 | 0 |

Введем избыточность, умножая каждый полином на x3:

|  |  |
| --- | --- |
| Полином | Избыточный полином |
| x3+x2+x | x6+x5+x4 |
| x3 | x6 |
| x2+x+1 | x5+x4+x3 |
| x2+x+1 | x5+x4+x3 |
| x2+x+1 | x5+x4+x3 |
| x+1 | x4+x3 |
| x3+x+1 | x6+x4+x3 |
| x3+x2 | x6+x5 |
| x3+x2+x | x6+x5+x4 |
| x+1 | x4+x3 |
| 1 | x3 |
| 0 | 0 |

Деля избыточные полиномы на образующий полином, получаем кодовые полиномы:

|  |  |  |
| --- | --- | --- |
| Избыточный полином | Кодированный полином | Кодовая комбинация |
| x6+x5+x4 | x6+x5+x4+x | 1110010 |
| x6 | x6+x2+x | 1000110 |
| x5+x4+x3 | x5+x4+x3+1 | 0111001 |
| x5+x4+x3 | x5+x4+x3+1 | 0111001 |
| x5+x4+x3 | x5+x4+x3+1 | 0111001 |
| x4+x3 | x4+x3+x | 0111001 |
| x6+x4+x3 | x6+x4+x3+x2 | 0011010 |
| x6+x5 | x6+x5+x2+1 | 1100101 |
| x6+x5+x4 | x6+x5+x4+x | 1110010 |
| x4+x3 | x4+x3+x | 0011010 |
| x3 | x3+x2+1 | 0001101 |
| 0 | 0 | 0000000 |

Битовая скорость R равна:

,

Кратность обнаруживаемой ошибки:

Кратность исправляемой ошибки:

Декодирование и обнаружение/исправление ошибки.

Если при делении кодированного полинома на образующий полином, остаток от деления равен 0, то ошибок нет и служебную информацию можно отбросить.  
Внесем ошибку в четвертый разряд полинома x6+x5+x4+x, т.е. добавим член x3 (1111010) и разделим результат на образующий полином, сдвигая кодовую комбинацию влево на разряд, до тех пор, пока остаток от деления не будет равен 1.

|  |  |  |
| --- | --- | --- |
| Итерация | Делимое | Остаток |
| 1 | x6+x5+x4+x3 | x2+1 |
| 2 | x6+x5+x4+x2+1 | x2+x+1 |
| 3 | x6+x5+x3+x+1 | x+1 |
| 4 | x6+x4+x2+x+1 | x2+x |
| 5 | x5+x3+x2+x+1 | 1 |

Сдвигаем вектор 0000001 влево на 4 разряда, получаем 0001000, складываем этот вектор по модулю 2 с ошибочной кодовой комбинацией, получая в результате 1110010. Переводим комбинацию в полиномиальный вид, делим на образующий полином. Остаток от деления равен 0, значит проверочные разряды можно отбросить.

## Кодирование с исправлением двойной ошибки.

В качестве кода, исправляющего ошибку двойной кратности, выберем сверточный код. Применим 3-х разрядный сверточный кодер с векторами связи g1=111 и g2=101.Ниже представлен граф переходов этого кодера.



Переход по сплошной линии соответствует входному 0, по пунктирной – входной 1.Составим таблицу шифрования:

|  |  |  |  |
| --- | --- | --- | --- |
| Итерация | Входной символ | Состояние РС | Выходная посл. |
| 0 | - | 000 | - |
| 1 | 1 | 100 | 11 |
| 2 | 1 | 110 | 01 |
| 3 | 1 | 111 | 10 |
| 4 | 1 | 111 | 10 |
| 5 | 1 | 111 | 10 |
| 6 | 0 | 011 | 10 |
| 7 | 1 | 101 | 00 |
| 8 | 1 | 110 | 01 |
| 9 | 0 | 011 | 01 |
| 10 | 0 | 001 | 11 |
| 11 | 1 | 100 | 11 |
| 12 | 0 | 010 | 10 |
| 13 | 1 | 101 | 00 |
| 14 | 0 | 010 | 10 |
| 15 | 0 | 001 | 00 |
| 16 | 1 | 100 | 11 |
| 17 | 0 | 010 | 10 |
| 18 | 1 | 101 | 00 |
| 19 | 0 | 010 | 10 |
| 20 | 1 | 101 | 00 |
| 21 | 0 | 010 | 10 |
| 22 | 1 | 101 | 00 |
| 23 | 0 | 010 | 10 |
| 24 | 1 | 101 | 00 |
| 25 | 0 | 010 | 10 |
| 26 | 0 | 001 | 11 |
| 27 | 0 | 000 | 00 |
| 28 | 0 | 000 | 00 |
| 29 | 0 | 000 | 00 |
| 30 | 0 | 000 | 00 |
| 31 | 0 | 000 | 00 |
| 32 | 0 | 000 | 00 |
| 33 | 1 | 100 | 11 |
| 34 | 0 | 010 | 10 |
| 35 | 1 | 101 | 00 |
| 36 | 0 | 010 | 10 |
| 37 | 1 | 101 | 00 |
| 38 | 1 | 110 | 01 |
| 39 | 1 | 111 | 10 |
| 40 | 0 | 011 | 01 |
| 41 | 0 | 001 | 11 |
| 42 | 1 | 101 | 11 |
| 43 | 1 | 110 | 01 |
| 44 | 0 | 011 | 01 |
| 45 | 1 | 101 | 00 |
| 46 | 0 | 010 | 10 |
| 47 | 0 | 001 | 11 |
| 48 | 0 | 000 | 00 |

Построим по графу решетчатую диаграмму[[1]](#footnote-2):



Битовая скорость R равна:

,

Кратность обнаруживаемой ошибки:

Кратность исправляемой ошибки:

*,*

где d – минимальное свободное расстояние, определяемое как метрика Хэмминга между двумя минимальными путями, сходящимися в одной точке (d=5).

Декодирование и исправление двукратной ошибки.  
Для декодирования и исправления воспользуемся алгоритмом Витерби, согласно которому, необходимо посчитать метрики Хэмминга для всех путей в некоторый момент времени t и выбрать минимальную из них. Будем считать метрики Хэмминга для кодированной последовательности из 8 символов.

|  |  |  |
| --- | --- | --- |
| Путь | Метрика Хэмминга для верной последовательности  11 01 10 10 | Метрика Хэмминга для ошибочной последовательности 10 01 11 10 |
| 11 10 11 00 | 4 | 4 |
| 11 01 01 11 | 3 | 3 |
| 11 10 11 11 | 4 | 4 |
| 11 01 01 00 | 3 | 3 |
| 11 10 00 10 | 3 | 5 |
| 11 01 10 01 | 2 | 4 |
| 11 10 00 01 | 5 | 7 |
| **11 01 10 10** | **0** | **2** |

Выбираем последнюю ячейку, так как она содержит минимальные метрики Хэмминга для обоих случаев. Ошибочную последовательность интерпретируем как путь, с которым у нее минимальная метрика Хэмминга. Для декодирования сообщения необходимо пройти по решетчатой диаграмме, интерпретируя блок кода как значение, при котором совершается переход, т.е. последовательность 11 01 10 10 декодируем в 1 1 1 1.

## Импульсная модуляция

Представим сигнал кодировкой без возврата к нулю:



Достоинства метода :  
— Простота реализации.  
— Метод обладает хорошей распознаваемостью ошибок (благодаря наличию двух резко отличающихся потенциалов).  
— Основная гармоника f0 имеет достаточно низкую частоту (равную N/2 Гц), что приводит к узкому спектру.

Недостатки метода :  
— Метод не обладает свойством самосинхронизации. Даже при наличии высокоточного тактового генератора приёмник может ошибиться с выбором момента съёма данных, так как частоты двух генераторов никогда не бывают полностью идентичными. Поэтому при высоких скоростях обмена данными и длинных последовательностях единиц или нулей небольшое рассогласование тактовых частот может привести к ошибке в целый такт и, соответственно, считыванию некорректного значения бита.  
— Вторым серьёзным недостатком метода, является наличие низкочастотной составляющей, которая приближается к постоянному сигналу при передаче длинных последовательностей единиц и нулей. Из-за этого многие линии связи, не обеспечивающие прямого гальванического соединения между приёмником и источником, этот вид кодирования не поддерживают. Поэтому в сетях код NRZ в основном используется в виде различных его модификаций, в которых устранены как плохая самосинхронизация кода, так и проблемы постоянной составляющей.

Представим сигнал кодировкой с возвратом к нулю:



Достоинства метода:  
— Высокая помехоустойчивость по сравнению с кодом без возврата к нулю.

Недостатки метода:  
— Более сложный процесс декодирования (на приемной стороне сигнал необходимо пропустить через дифференцирующую цепь для определения начала и конца импульса) .

Представим сигнал манчестерским кодом:

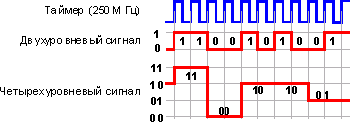


Достоинства метода:  
— Самосинхронизация.

Недостатки метода:  
— Требует вдвое большей пропускной способности чем прямое кодирование.

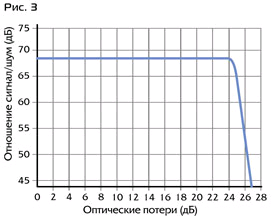
Реализация импульсной модуляции в витой паре.

*5-ти уровневая амплитудно-импульсная модуляция*  обеспечивает пропускную способность выше, чем бинарный сигнал, где каждый символ представлен одним битом (0 или 1). Так, каждый передаваемый символ представлен одним из 5ти разных уровней (-2, -1, 0, + 1, +2). Так как каждый символ может представляться 2-мя битами информации (4 уровня используется для представления 2х бит, плюс дополнительный пятый уровень используется для разработки метода поддержки коррекции ошибок (FEC)), скорость передачи данных вместе с пропускной способностью увеличивается. Распространенное четырехуровневое кодирование обрабатывает входящие биты парами. Всего существует 4 различных комбинации - 00, 01, 10, 11. Передатчик может каждой паре бит установить свой уровень напряжения передаваемого сигнал, что уменьшает в 2 раза частоту модуляции четырехуровневого сигнала, 125 МГц вместо 250 МГц, (рис.4), и следовательно частоту излучения. Пятый уровень добавлен для создания избыточности кода. В результате чего становится возможной коррекция ошибок на приеме. Это дает дополнительный резерв 6 дБ в соотношении сигнал/шум.



Реализация импульсной модуляции в коаксиальном кабеле.

Растущая потребность в повышении качества передачи изображения в системах охранного телевидения обусловила необходимость разработки полностью цифровых методов передачи видеосигнала. В этом случае для передачи двоичных символов в волоконнооптических системах связи обычно используется импульсно-кодовая модуляция; "0" соответствует отсутствию, "1" - наличию оптического излучения в волокне. Использование цифровых технологий обеспечило появление систем передачи одного или нескольких видеосигналов по одному оптическому кабелю с исключительно высоким качеством, не зависящим от длины линии связи .



При этом в 2-3 раза удалось снизить искажения типа "дифференциальная фаза" и "дифференциальное усиление" по сравнению с аналоговыми методами. Использование даже 8битного кодирования позволяет создать транспортную среду, удовлетворяющую требованиям ГОСТ Р 5072594 и RS250C MediumHaul Transmission.

Например, цифровые кодеки серий "CFO" и "OPX" финской фирмы "Teleste" c 8битным кодированием обеспечивают отношение "сигнал/шум" не хуже 60 дБ при искажениях типа "дифференциальное усиление" - 1%, "дифференциальная фаза" - 1°. Переход к 12битному кодированию в последних разработках "Teleste" позволил создать многомодовую аппаратуру, обеспечивающую студийное качество передаваемых изображений в соответствии с CCIR601 и RS250C ShortHaul Transmission. Так, передатчики серии "CFO 121" обеспечивают в линии связи отношение "сигнал/шум" не хуже 70 дБ при искажениях типа "дифференциальное усиление" -1%, "дифференциальная фаза" - 1°.

## Многоканальная передача.

Частотное уплотнение

Используются канальные сигналы, частотные спектры которых располагаются в неперекрывающихся частотных полосах. Формирование канальных сигналов осуществляется при помощи АМ, ЧМ или ФМ так, чтобы средние частоты спектров канальных сигналов соответствовали средним частотам отведенных полос каждого канала. В приемной части разделение каналов осуществляется набором частотных фильтров, каждый из которых пропускает спектр частот, принадлежащий только данному канальному сигналу. На рисунках показаны спектры сообщений, передаваемых по трем каналам и спектр сигнала, передаваемого по линии связи.





В многоканальных системах с временным разделением каналов (ВРК) канальные сигналы не перекрываются во времени, что обеспечивает их ортогональность.

Рассмотрим один из способов формирования канальных сигналов в системе с ВРК. Сообщения λk, поступающие от источников, подвергаются дискретизации по времени так, чтобы отсчеты одного сообщения не совпадали с отсчетами другого. В соответствии с моментами отсчетов вырабатываются импульсы, параметры которых меняются в зависимости от значений сообщений сообщения в каждом отсчете. Канальные сигналы, образованные из сообщения λ1, не совпадают по времени с канальными сигналами, образованными из сообщения λ2.





## Множественный доступ

Для обеспечения возможности одновременного использования канала передачи данных несколькими пользователями применяют системы множественного доступа:

- множественный доступ с частотным разделением — при этом каждому пользователью предоставляется отдельный диапазон частот.

- множественный доступ с временным разделением — каждому пользователю предоставляется определенный временной интервал Таймслот, в течение которого он производит передачу и прием данных.

- множественный доступ с кодовым разделением — при этом каждому пользователю выдается кодовая последовательность, полученная, например, с помощью фунуции Уолша. Данные пользователя накладываются на кодовую последовательность таким образом, что передаваемые сигналы различных пользователей не мешают друг другу, хотя и передаются на одних и тех же частотах.

1. Проведена селекция путей после такта t3 по алгоритму Витерби [↑](#footnote-ref-2)