Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Телекоммуникационные системы

Отчет по лабораторной работе №8
Модель телекоммуникационного канала

Работу выполнил:
Еременко Д.Ю.
Группа: 33531/2
Преподаватель:
Богач Н.В.

Санкт-Петербург
2019

Содержание

1. Цель работы

По имеющейся записи сигнала из эфира и коду модели передатчика создать модель приемника, в которой найти позицию начала пакета и, выполнив операции демодуляции, деперемежения и декодирования, получить передаваемые параметры

2. Программа работы

Пакетный сигнал длительностью 200 мкс состоит из 64 бит полезной информации и 8 нулевых tail-бит. В нулевом 16-битном слове пакета передается ID, в первом - период излучения в мс, во втором – сквозной номер пакета, в третьем - контрольная сумма (CRC-16). На передающей стороне пакет сформированнй таким образом проходит следующие этапы обработки:

1. Помехоустойчивое кодирование.
2. Перемежение бит.
3. Модуляция символов.
4. Прямое расширение спектра.

3. Ход выполнения работы

3.1. Листинг

```
clear all;
close all;

data = int16([0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 ...
0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 1 ...
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]);

msg = transmit(data);

fid=fopen('test.sig', 'w','l');
fwrite(fid, msg, 'int16');
fclose(fid);


function message = transmit(data)

%% Parameters

PRS = int16([1;  1;  1;  1;  1;  1; -1; -1; -1; -1;  1;  1;  1;  1; -1;  1;
1;  1; -1; -1; -1; -1;  1; -1;  1;  1; -1; -1;  1;  1; -1;  1;
1; -1;  1;  1;  1;  1; -1;  1; -1; -1; -1; -1;  1;  1;  1; -1;
-1;  1;  1; -1; -1; -1; -1;  1; -1; -1;  1; -1; -1; -1;  1; -1;
1; -1;  1;  1;  1; -1;  1; -1;  1;  1;  1;  1; -1; -1;  1; -1;
-1;  1; -1;  1;  1;  1; -1; -1;  1;  1;  1; -1; -1; -1; -1; -1;
-1;  1;  1;  1; -1;  1;  1;  1; -1;  1; -1; -1;  1;  1;  1;  1;
-1;  1; -1;  1; -1; -1;  1; -1;  1; -1; -1; -1; -1; -1; -1;  1;
-1;  1; -1;  1; -1;  1; -1;  1;  1;  1;  1;  1; -1;  1; -1;  1;
1; -1;  1; -1; -1; -1; -1; -1;  1;  1; -1;  1;  1;  1; -1;  1;
1; -1;  1;  1; -1;  1; -1;  1;  1; -1; -1; -1; -1; -1;  1; -1;
1;  1;  1; -1;  1;  1;  1;  1;  1; -1; -1; -1;  1;  1;  1;  1;
-1; -1;  1;  1; -1;  1; -1; -1;  1;  1; -1;  1; -1;  1;  1;  1;
-1; -1; -1;  1;  1; -1;  1; -1; -1; -1;  1; -1;  1;  1;  1;  1;
```

```
1;  1;  1; -1;  1; -1; -1;  1; -1;  1;  1; -1; -1; -1;  1; -1;
1; -1; -1;  1;  1; -1; -1; -1;  1;  1; -1; -1; -1; -1; -1; -1;
-1;  1;  1; -1; -1;  1;  1; -1; -1;  1; -1;  1; -1;  1;  1; -1;
-1;  1; -1; -1;  1;  1;  1;  1;  1;  1; -1;  1;  1; -1;  1; -1;
-1;  1; -1; -1;  1; -1; -1;  1;  1; -1;  1;  1;  1;  1;  1;  1;
-1; -1;  1; -1;  1;  1; -1;  1; -1;  1; -1; -1; -1; -1;  1; -1;
1; -1; -1; -1;  1; -1; -1;  1;  1;  1; -1;  1;  1; -1; -1;  1;
-1;  1;  1;  1;  1; -1;  1;  1; -1; -1; -1; -1;  1;  1; -1;  1;
-1;  1; -1;  1; -1; -1;  1;  1;  1; -1; -1;  1; -1; -1; -1; -1;
1;  1; -1; -1; -1;  1; -1; -1; -1; -1;  1; -1; -1; -1; -1; -1;
-1; -1; -1;  1; -1; -1; -1;  1; -1; -1; -1;  1;  1; -1; -1;  1;
-1; -1; -1;  1;  1;  1; -1;  1; -1;  1; -1;  1;  1; -1;  1;  1;
-1; -1; -1;  1;  1;  1; -1; -1; -1;  1; -1; -1;  1; -1;  1; -1;
1; -1; -1; -1;  1;  1; -1;  1;  1; -1; -1;  1;  1;  1;  1;  1;
-1; -1;  1;  1;  1;  1; -1; -1; -1;  1; -1;  1;  1; -1;  1;  1;
1; -1; -1;  1; -1;  1; -1; -1;  1; -1; -1; -1; -1; -1;  1; -1;
-1;  1;  1; -1; -1;  1;  1;  1; -1;  1; -1; -1; -1;  1;  1;  1;
1;  1; -1;  1;  1;  1;  1; -1; -1; -1; -1; -1;  1;  1;  1  ]);


interleaver = int16([  0; 133; 122; 111; 100;  89;  78;  67;  56;  45;  34;  23;
12;   1; 134; 123; 112; 101;  90;  79;  68;  57;  46;  35;
24;  13;   2; 135; 124; 113; 102;  91;  80;  69;  58;  47;
36;  25;  14;   3; 136; 125; 114; 103;  92;  81;  70;  59;
48;  37;  26;  15;   4; 137; 126; 115; 104;  93;  82;  71;
60;  49;  38;  27;  16;   5; 138; 127; 116; 105;  94;  83;
72;  61;  50;  39;  28;  17;   6; 139; 128; 117; 106;  95;
84;  73;  62;  51;  40;  29;  18;   7; 140; 129; 118; 107;
96;  85;  74;  63;  52;  41;  30;  19;   8; 141; 130; 119;
108;  97;  86;  75;  64;  53;  42;  31;  20;   9; 142; 131;
120; 109;  98;  87;  76;  65;  54;  43;  32;  21;  10; 143;
132; 121; 110;  99;  88;  77;  66;  55;  44;  33;  22;  11]);


N = length(data);

%% Convolution Encoder

trellis = poly2trellis(9, [753 561]);
convolved_signal = convenc(data, trellis);

%% Interleaver

interleaved_signal = convolved_signal(interleaver+1);
signal_matrix = reshape(interleaved_signal(1:N*2), [N*2/6, 6]);

%% Spread spectrum

y = int16(hadamard(64));
row_number = int16(zeros(1, N*2/6));
for k=1:N*2/6
line = signal_matrix(k,:);
row_number(k) = bi2de(line) + 1;
end

signal = y(row_number, :);
signal_to_modulate = reshape(signal', 1, []);

n_repeat = int16(length(signal_to_modulate) / length(PRS'));
n_compl  = length(signal_to_modulate) - length(PRS') * n_repeat;
```

```
93
94 signal_to_modulate = signal_to_modulate .* [repmat(PRS', 1, n_repeat), PRS(1:
       ↪ n_compl)'];
95 signal_to_modulate = [PRS', signal_to_modulate];
96
97 %% BPSK modulation
98
99 mask = (signal_to_modulate == -1);
100 signal_to_modulate(mask) = 0;
101
102 IQ = pskmod(double(signal_to_modulate), 2);
103 %M = comm.BPSKModulator;
104 %IQ = M(signal_to_modulate')';
105 %add white gaussian noise
106 IQ = awgn(IQ, inf);
107 scatterplot(IQ);
108
109 % model oversampling by 2
110 IQ_oversampled = [IQ, IQ];
111 IQ_im_part=imag(IQ_oversampled);
112 IQ_re_part=real(IQ_oversampled);
113 IQ_record = [IQ_re_part, IQ_im_part];
114
115 %IQ_record= [IQ_record(101:180) IQ_record];
116
117 %% Calculate CRC-code
118
119 crc_gen = crc.generator('Polynomial', '0x1021');
120 crc = generate(crc_gen, data');
121 crc_send = crc(73:88);
122
123 %% Form message
124 message = int16(IQ_record);
125
126 end
```

Закодируем пакетный сигнал функциями poly2trellis и convenc. 753, 561 - образующие полиномы, 9 - кодовое ограничение. На выходе кодера количество бит становится равным 144.

```
trellis = poly2trellis(9, [753 561]);
convolved_signal = convenc(data, trellis);
```

Перемежение бит происходит в соответствии с вектором interleaver, где значение элемента равно новой позиции бита по заданному адресу. Количество бит на этом этапе остается неизменным.

```
interleaver = int16([0;  133;  122;  111;  100;   89;   78;   67;   56;   45;   34;   23;
                     12;    1;  134;  123;  112;  101;   90;   79;   68;   57;   46;   35;
                     24;   13;    2;  135;  124;  113;  102;   91;   80;   69;   58;   47;
                     36;   25;   14;    3;  136;  125;  114;  103;   92;   81;   70;   59;
                     48;   37;   26;   15;    4;  137;  126;  115;  104;   93;   82;   71;
                     60;   49;   38;   27;   16;    5;  138;  127;  116;  105;   94;   83;
                     72;   61;   50;   39;   28;   17;    6;  139;  128;  117;  106;   95;
                     84;   73;   62;   51;   40;   29;   18;    7;  140;  129;  118;  107;
                     96;   85;   74;   63;   52;   41;   30;   19;    8;  141;  130;  119;
                    108;   97;   86;   75;   64;   53;   42;   31;   20;    9;  142;  131;
                    120;  109;   98;   87;   76;   65;   54;   43;   32;   21;   10;  143;
                    132;  121;  110;   99;   88;   77;   66;   55;   44;   33;   22;   11]);
```

На следующем этапе пакет из 144 полученных с выхода перемежителя бит разбивается на 24 символа из 6 бит. Генерируется таблица функций Уолша длиной 64 бита. Каждый 6- битный символ заменяется последовательностью Уолша, номер которой равен значению данных 6-ти бит. Т.о. на выходе модулятора получается 24 * 64 = 1536 знаковых символов.

```
signal_matrix = reshape(interleaved_signal(1:N*2), [N*2/6, 6]);
```

```
y = int16(hadamard(64));
```

```
row_number = int16(zeros(1, N*2/6));
for k=1:N*2/6
    line = signal_matrix(k,:);
    row_number(k) = bi2de(line) + 1;
end
```

Полученную последовательность из 1536 символов периодически умножим с учетом знака на псевдослучайную последовательность длиной 511 символов.

```
n_repeat = int16(length(signal_to_modulate) / length(PRS'));
n_compl = length(signal_to_modulate) - length(PRS') * n_repeat;

signal_to_modulate = signal_to_modulate .* [repmat(PRS', 1, n_repeat), PRS(1:
    ↪ n_compl)'];
```

Далее к началу сформированного символьного пакета прикрепим немодулированную ПСП. Т.о. символьная длина становится равной 2047.

```
signal_to_modulate = [PRS', signal_to_modulate];
```

Далее полученные символы промодулируем методом BPSK.

```
mask = (signal_to_modulate == -1);
signal_to_modulate(mask) = 0;

IQ = pskmod(double(signal_to_modulate), 2);
```

4. Выводы

В данной работе была разработана модель формирования пакета данных с использованием помехоустойчивого кодирования сверточным кодом, перемежением бит, модуляцией символов и расширением спектра.