[**Введение** 2](#_Toc197347269)

[**Вступление** 3](#_Toc197347270)

[**Постановка задачи** 6](#_Toc197347271)

[**Анализ основных блоков алгоритма** 7](#_Toc197347272)

[**S2P** 7](#_Toc197347273)

[**Mapper** 8](#_Toc197347274)

[**IFFT** 12](#_Toc197347275)

[**P2S** 19](#_Toc197347276)

[**Реализация алгоритма** 20](#_Toc197347277)

[**Заключение** 28](#_Toc197347278)

[**Приложение 1** 29](#_Toc197347279)

[**Принцип работы приемопередатчика на основе OFDM** 29](#_Toc197347280)

[**Существующие реализации** 33](#_Toc197347281)

[**Приложение 2.** 49](#_Toc197347282)

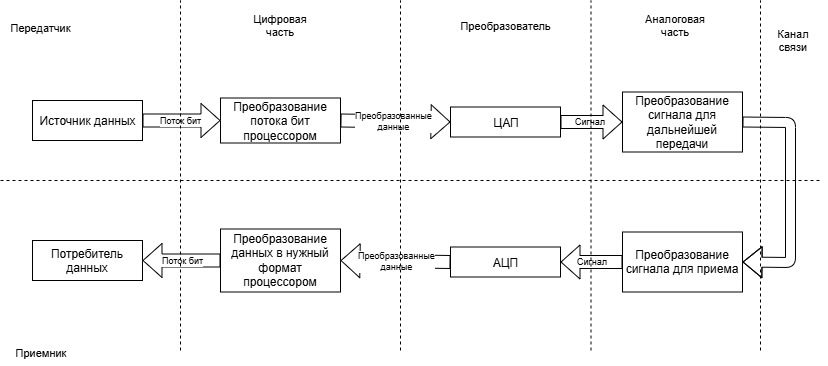
[**Модуляция** 49](#_Toc197347283)

[**Приложение 3.** 54](#_Toc197347284)

[**Алгоритмы БПФ.** 54](#_Toc197347285)

# **Введение**

Телефон стал неотъемлемой частью жизни, и сложно представить себе современного человека, обходящегося без этого устройства. Одной из важнейших функций телефона является передача информации. Системы приема и передачи информации состоят условно из аналоговой и цифровой части.

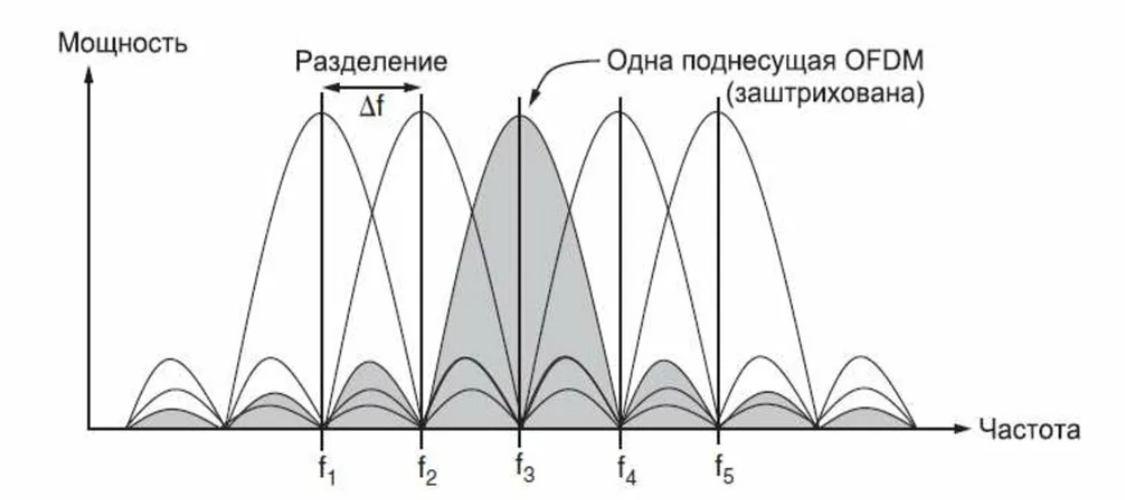


В аналоговой части осуществляется преобразование частоты, усиление и фильтрация сигнала. В цифровой части передатчика входящий в него поток бит преобразуется в многочастотный сигнал с нужным спектром, переносящий ускоренный в тысячи раз поток бит. В приемнике, с учетом многолучевого распространения сигнала и эффекта Доплера, осуществляются обратные действия. В данной работе рассматривается цифровая часть блока приема и передачи информации.

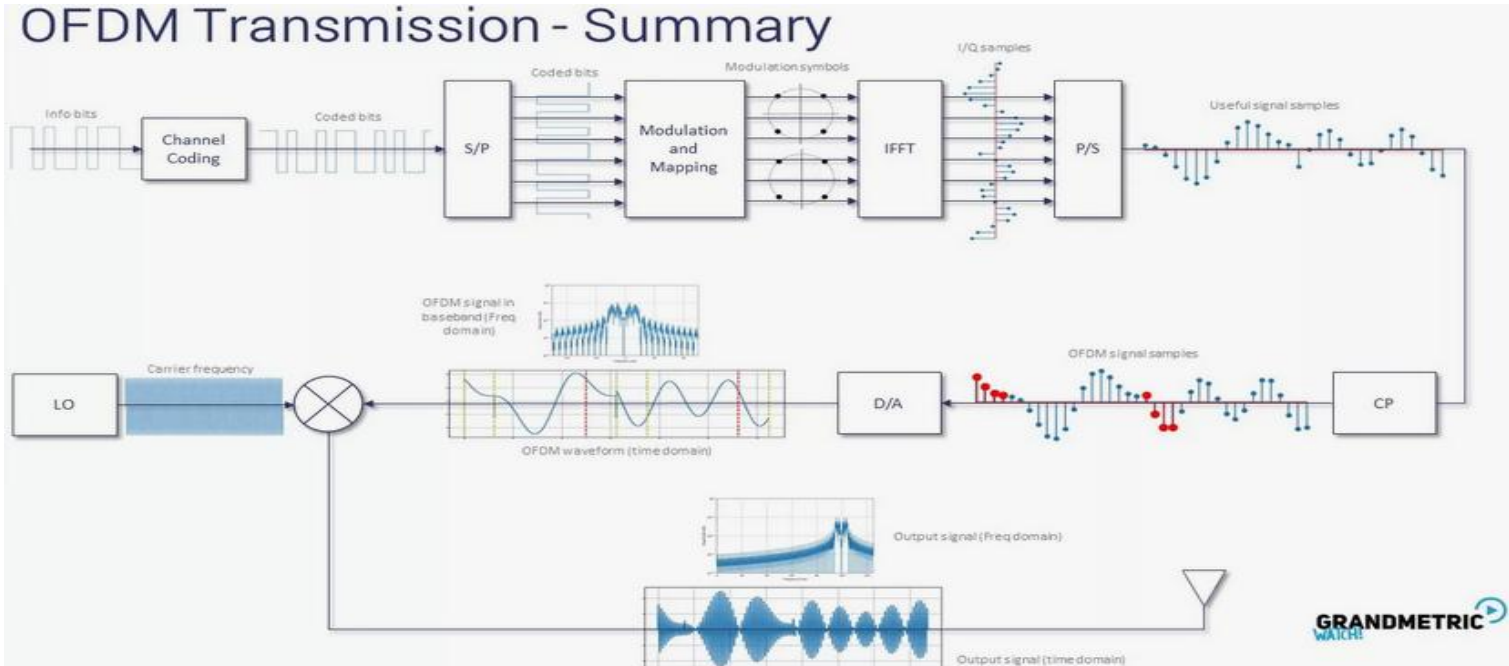
# **Вступление**

Наиболее распространенным способом преобразования потока бит в сигнал является алгоритм с использованием OFDM.

OFDM (мультиплексирование с ортогональным частотным разделением каналов) – технология цифровой модуляции с использованием большого количества близко расположенных ортогональных поднесущих. То есть данные будут разделяться на части, преобразовываться в сигнал и объединяться так, что спектры разных поднесущих перекрываются, но не мешают друг другу. Итоговый сигнал представлен на рисунке **\*\*\*N\*\*\*:**

****

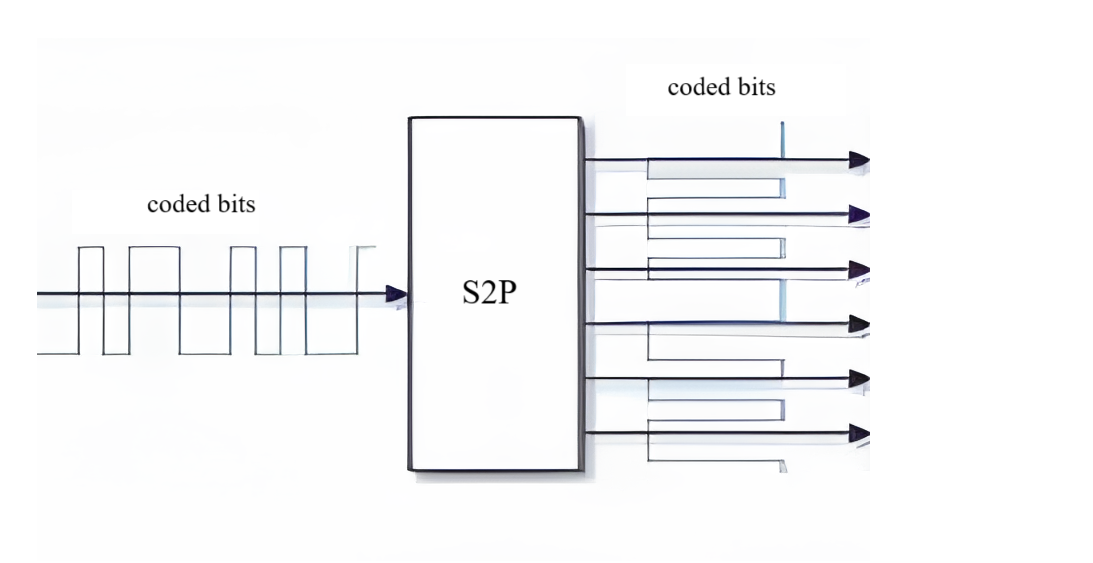
Данный алгоритм имеет следующую структуру:

****

**\*\*\*Нарисовать или найти более качественную картинку\*\*\***

Разберем подробнее алгоритм.

Входящий поток бит разделяется на битовые слова, которые имеют фиксированную длину (блок S2P). Стоит отметить, что из-за разделения единого потока бит на битовые слова, последующая обработка битового слова должна производиться со скоростью, равной , где – Скорость обработки входящего битового потока, – количество битовых слов, которые блок выдает за один цикл работы (количество выходных битовых слов). Также возможно организовать параллельную работу последующих блоков, как на рисунке **\*\*\*N\*\*\***. При организации параллельной обработки важно правильно учитывать и рассчитывать скорости работы блоков.

****Далее слова преобразуются в соответствующее комплексное число (блок mapper). Сопоставление битового слова и комплексного числа зависит от алгоритма в mapper. Также от этого алгоритма зависит и длина битового слова. Более подробная информация описана в главе Mapper.

После этого комплексные числа с помощью обратного преобразования Фурье преобразуются в сигнал (блок IFFT). **\*\*\* не уверен насчет формулировки, но пока не могу нормально сформулировать\*\*\***

Последним этапом является соединение рассчитанных данных в единый сигнал, который далее передается в определенном частотном диапазоне (блок P2S).

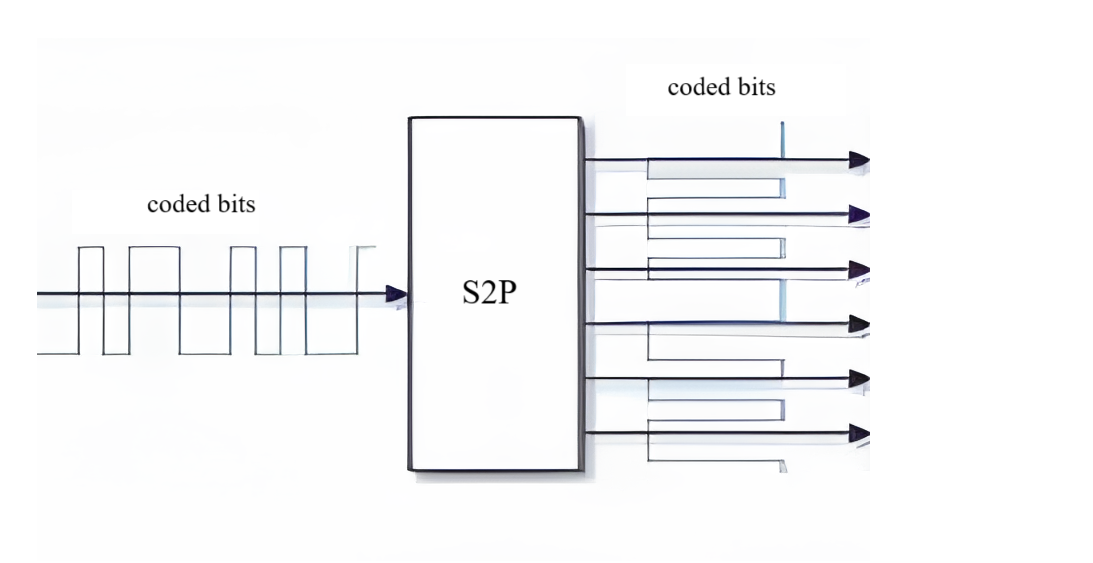
**Задача** заключается в: проведении анализа существующих реализаций алгоритмов OFDM; оптимизации и реализации алгоритма под конкретные условия на основе анализа.

# **Постановка задачи**

Проанализировать существующие реализации алгоритма OFDM, как в аппаратной, так и в программной формах; конкретизировать функциональные блоки OFDM-приемопередатчика, определяя параметры, зависящие от конкретных условий работы (частотный диапазон, скорость передачи данных, тип модуляции, характеристики канала связи). На основе этого анализа, выделить наиболее нагруженные места в каждом из блоков, выявить узкие места и провести оптимизацию этих блоков под конкретную задачу, учитывая требования к производительности, энергопотреблению и занимаемой памяти. Реализовать алгоритм в имитаторе и на микроконтроллере.

# **Анализ основных блоков алгоритма**

## **S2P**

****

Данный блок разделяет входящий поток бит на группы бит фиксированной длины. Из описания полного алгоритма OFDM требуется отметить, что длина выходного из блока битового слова зависит от выбранного в блоке Mapper алгоритма. Также, как было ранее описано, важно помнить, что последующая скорость обработки битовых слов будет ниже, чем обработка битового потока, так как иначе блок S2P не будет успевать обрабатывать слова, а последующие блоки будут простаивать, ожидая окончания выполнения работы блока S2P. Это уточнение будет важно при реализации данного алгоритма на микроконтроллере либо при расчете частоты работы функций в имитаторе. Кодовая реализация данного блока имеет следующий вид:

Листинг

// Преобразование символов в кадры по 4 бита. (Проверено)

inline void Symbol\_produce(unsigned char\* cadrs, char\* buff)

{

    for (i = 0; i < COUNT\_CADR; ++i, ++iGl)

    {

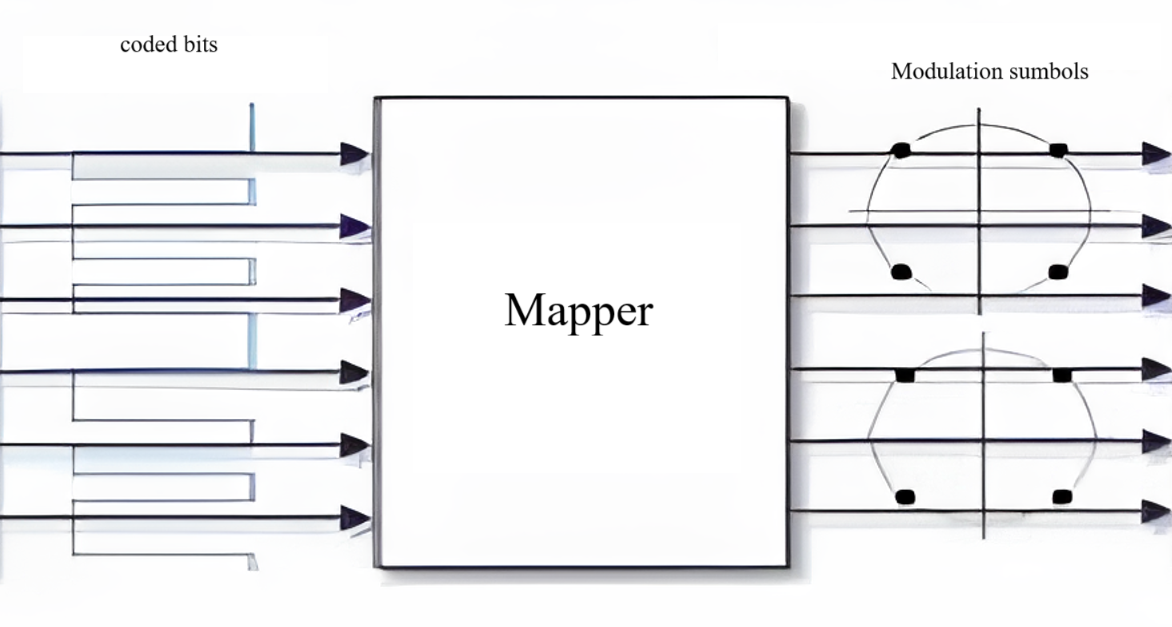
        cadrs[i] = buff[iGl] & 0b1111;

        cadrs[++i] = buff[iGl] >> 4 & 0b1111;

    }

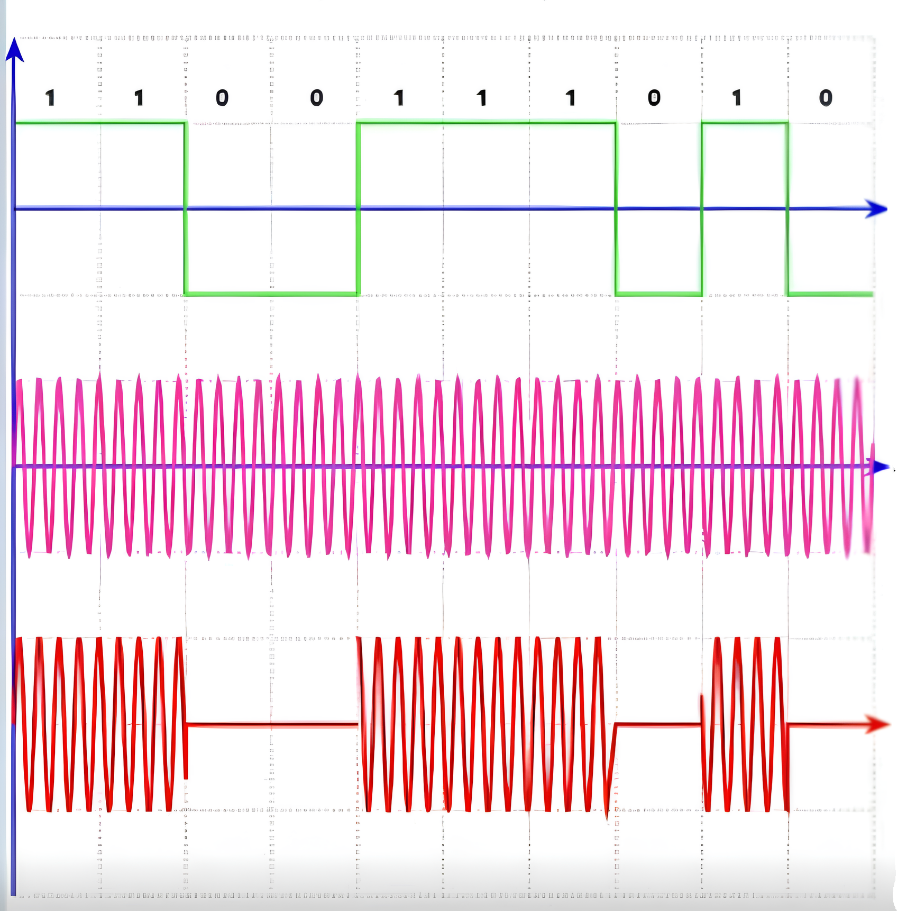
}

## **Mapper**

****

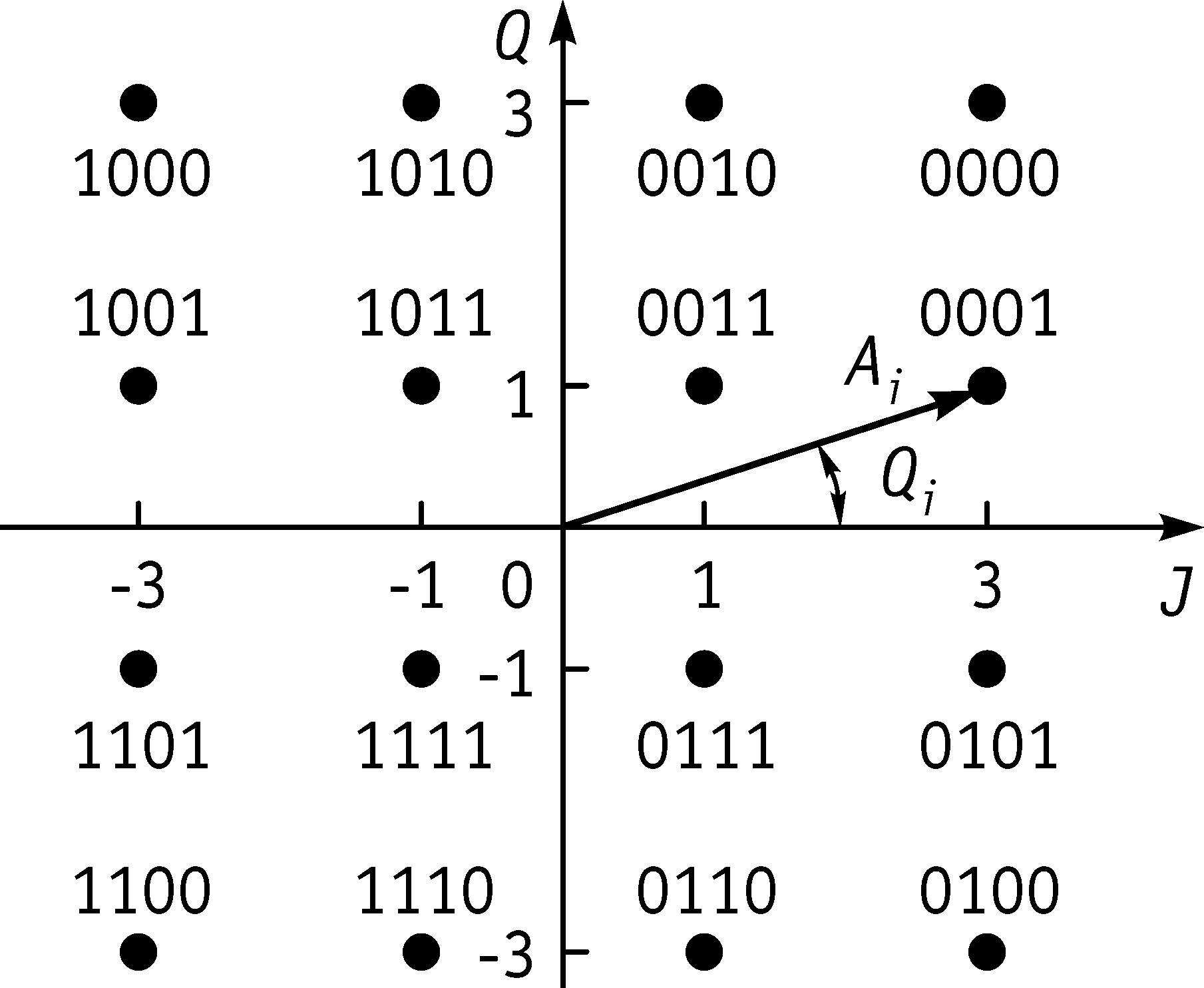
Блок Mapper принимает на вход битовые слова и по определенному алгоритму преобразует их в соответствующее комплексное число. Данная операция нужна для реализации передачи слов в виде сигнала.

Может возникнуть вопрос, почему бы не передавать сигнал без преобразования сразу в битовом виде, как, например, на рисунке **\*\*\*N\*\*\*.**

****

Этот способ передачи данных неэффективен, так как если передавать в виде «1 – есть сигнал, 0 – нет сигнала», то сигнал будет сильно подвержен влиянию шума **\*\*\*Возможно стоит немного рассказать, что такое шум\*\*\***, а при передаче бита на более широкой частоте такой способ передачи будет в разы проигрывать в скорости передачи информации методам кодирования (модуляции), которые реализуются в данном блоке.

Для рассмотрения способов модуляции, рассмотрим комплексную плоскость. На ней имеется ось реальных значений и ось мнимых значений.



Ось реальных значений определяет амплитуду волны. Ось мнимых значений - фазу волны. На основе этого имеется 3 вида алгоритмов, в основе которых: изменения фазы при фиксированной амплитуде; изменение амплитуды при фиксированной фазе; изменение фазы и амплитуды. **\*\*\* можно добавить просто картинки с тем, как они выглядят не вдаваясь в подробности, но стоит ли\*\*\***

Заметим, что самое эффективное кодирование является кодирование изменением и амплитуды, и фазы. **\*\*\*Можно описать, почему, но надо ли\*\*\***. В данных алгоритмах битовые слова сопоставляются определенным точкам на комплексной плоскости. Изображение сопоставления битовых слов определенным точкам на комплексной плоскости называют картой. Пример карты для алгоритма 16-QAM, который и будет реализован, представлен на рисунке **\*\*\*N, который выше\*\*\***

Стоит отметить, что данная карта составлена так, чтобы соседние точки в ней отличались лишь на 1 бит. Данный способ кодирования позволяет уменьшить вероятность ошибки при передаче и облегчает восстановление данных при возможных ошибках при приеме и дешифровки сигнала.

Программная реализация данного блока:

inline void Modulation(unsigned char \* cadrs, fftw\_complex \*symbolCoding)

{

    // Вариант 1: swithc-case.

    for (i = 0; i < COUNT\_CADR; ++i)

    {

        switch (cadrs[i]) {

            case 0b0000: symbolCoding[i][0] = -3.0; symbolCoding[i][1] =  3.0; break; // 0

            case 0b0001: symbolCoding[i][0] = -1.0; symbolCoding[i][1] =  3.0; break; // 1

            case 0b0010: symbolCoding[i][0] = -3.0; symbolCoding[i][1] =  1.0; break; // 2

            case 0b0011: symbolCoding[i][0] = -1.0; symbolCoding[i][1] =  1.0; break; // 3

            case 0b0100: symbolCoding[i][0] = -3.0; symbolCoding[i][1] = -3.0; break; // 4

            case 0b0101: symbolCoding[i][0] = -1.0; symbolCoding[i][1] = -3.0; break; // 5

            case 0b0110: symbolCoding[i][0] = -3.0; symbolCoding[i][1] = -1.0; break; // 6

            case 0b0111: symbolCoding[i][0] = -1.0; symbolCoding[i][1] = -1.0; break; // 7

            case 0b1000: symbolCoding[i][0] =  3.0; symbolCoding[i][1] =  3.0; break; // 8

            case 0b1001: symbolCoding[i][0] =  1.0; symbolCoding[i][1] =  3.0; break; // 9

            case 0b1010: symbolCoding[i][0] =  3.0; symbolCoding[i][1] =  1.0; break; // 10

            case 0b1011: symbolCoding[i][0] =  1.0; symbolCoding[i][1] =  1.0; break; // 11

            case 0b1100: symbolCoding[i][0] =  3.0; symbolCoding[i][1] = -3.0; break; // 12

            case 0b1101: symbolCoding[i][0] =  1.0; symbolCoding[i][1] = -3.0; break; // 13

            case 0b1110: symbolCoding[i][0] =  3.0; symbolCoding[i][1] = -1.0; break; // 14

            case 0b1111: symbolCoding[i][0] =  1.0; symbolCoding[i][1] = -1.0; break; // 15

            default:   symbolCoding[i][0] = 0.0; symbolCoding[i][1] = 0.0; // Обработка неожиданной ситуации (опционально)

        }

    }

    /\*Вариант 2: Использование выражения\*/

    /\*

    for (i = 0; i < COUNT\_CADR; i++)

    {

        symbolCoding[0][i] = (cadrs[i] >> 3) \* 2 - 1;

        symbolCoding[1][i] = ((cadrs[i] & 0b0010) >> 1) \* 2 - 1;

        if ((cadrs[i] & 0b0100)) symbolCoding[0][i] = symbolCoding[0][i] \* 3;

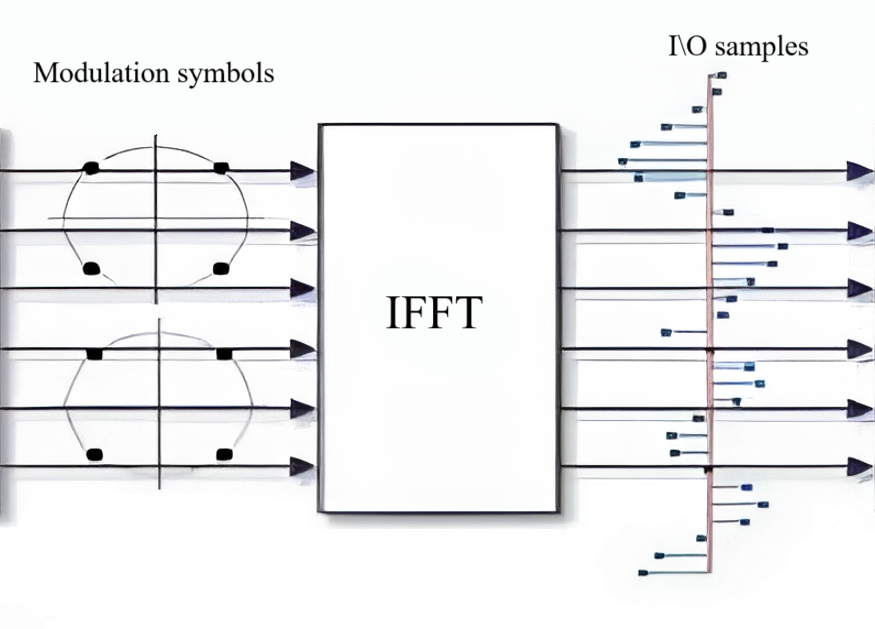
        if ((cadrs[i] & 0b0001)) symbolCoding[1][i] = symbolCoding[1][i] \* 3;

    }

    \*/

}

## **IFFT**

****

Данный блок преобразовывает комплексное число или комплексные числа, полученные на прошлом этапе, в сигнал. Это позволяет сделать обратное преобразование Фурье. Рассмотрим преобразование Фурье:

Для дискретного сигнала x[n] из N отсчетов (где n = 0, 1, ..., N-1), ДПФ, обозначаемое X[k], определяется как:

для k = 0, 1, ..., N-1, где n = 0 до N-1,

- n-ый отсчет входной последовательности (сигнала во временной области);

- k-ый отсчет выходной последовательности (спектр сигнала в частотной области);

- длина последовательности;

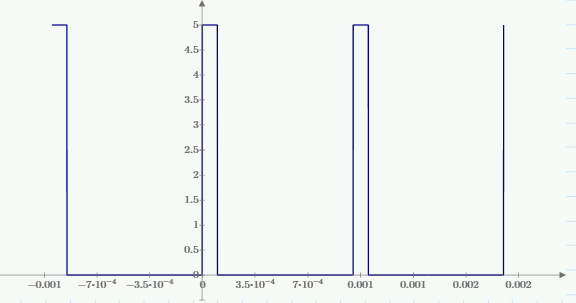
Обратное дискретное преобразование Фурье (ОДПФ) определяется как:

для n = 0, 1, ..., N-1, где k = 0 до N-1.

Разлагая комплексную экспоненту по формуле Эйлера (e^(j \* θ)) = cos(θ) + j \* sin(θ)) получим:

Данную формулу можно разложить на реальную и мнимую составляющие. Они будут равняться соответственно:

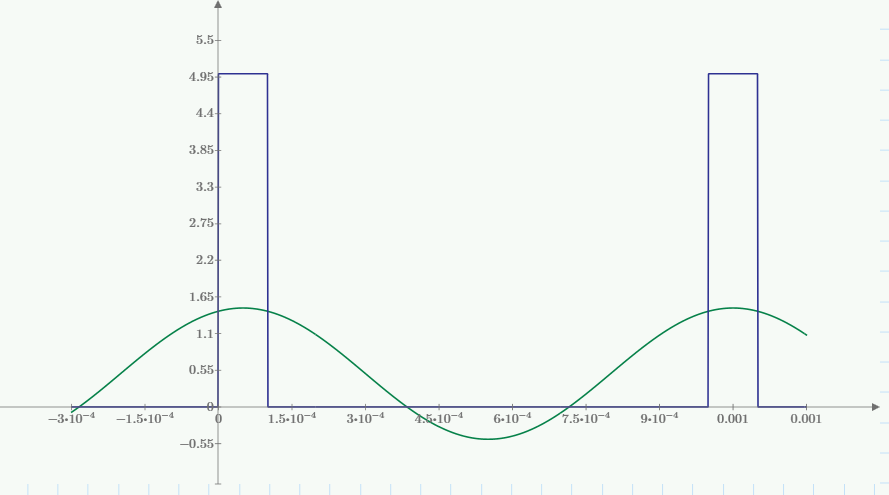
Рассмотрим на практическом примере. Имеем следующий сигнал, представленный на рисунке **\*\*\*N\*\*\***, который необходимо передать.



При преобразовании данного сигнала с помощью ОПФ мы получим сигнал, представленный на рисунке **\*\*\*N\*\*\***, по форме напоминающий исходный, но который возможно передать в цифро-аналоговый преобразователь для последующей передачи. **\*\*\*Возможно по формулировке непонятно, нужно уточнить\*\*\***

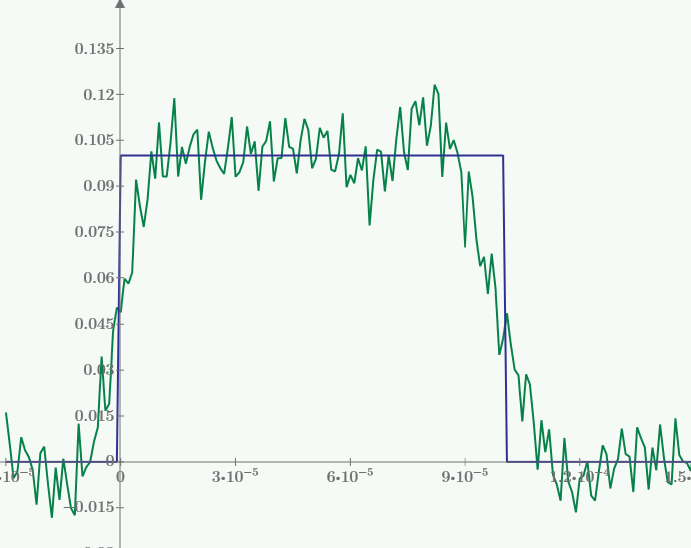


Важным параметром в вычислениях ОПФ является число гармоник, на которые будет разложен входной сигнал. Как можно увидеть по рисунку **\*\*\*N\*\*\***(на рисунок выше, где первый сигнал раскладывается), изначально прямоугольный сигнал разложился на сигнал, являющийся суммой синусоид и косинусоид. Эти синусоиды и косинусоиды и есть гармоники сигнала. От их количества зависит то, насколько точно выходной сигнал повторит входной. Преобразование на рисунке **\*\*\*N\*\*\***(на рисунок выше, где первый сигнал раскладывается) использует 40 гармоник. В качестве примера, рассмотрим преобразование того же входного сигнала, но используя одну гармонику. Получим следующий результат:

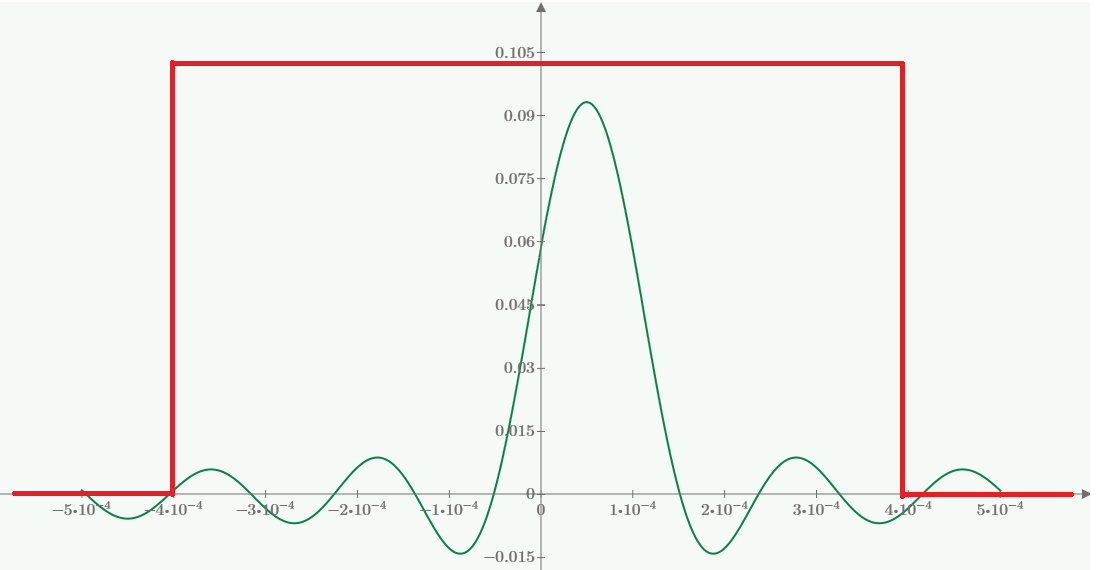


По данному выходному сигналу очень сложно определить изначальный вид. Используя 4 гармоники, получится следующий результат: 

Данный сигнал лучше похож на первоначальный, но все еще не так точно повторяет сигнал. Первоначальное разложение на 40 гармоник идеально раскладывает входной сигнал, однако, при увеличении количества точек вплоть до бесконечности, на которые разбивается сигнал, не удастся идеально восстановить исходных сигнал. График исходного и восстановленного сигнала при разложении на 4000 точек представлен на рисунке **\*\*\*N\*\*\***.



При дальнейшем увеличении точек будет увеличиваться количество пульсаций, а их амплитуда останется той же. Это связанно с тем, что то, что получается после преобразования, является сверткой частотной характеристики и прямоугольного окна. То есть, когда ограничивается число гармоник, это эквивалентно умножению бесконечной частотной характеристики на прямоугольную функцию, которая равна 1 в пределах выбранного диапазона частот и 0 за его пределами. Это вырезание части спектра и есть применение прямоугольного окна. Наглядный пример на рисунке **\*\*\*N\*\*\***. Также можно использовать другие окна (Ханна, Хэмминга), это поможет побороть пульсации.



Свертка – комбинирование двух сигналов.

Учитывая ограниченность времени вычисления, квадратичную (в случае ОДПФ) или логарифмическую (в случае ОБПФ) сложность вычисления и то, что от числа гармоник зависит число вычислений, требуется использовать ограниченное число гармоник, но такое, чтобы сигнал однозначно определялся.

Из-за конкретизированных преобразований в блоке Mapper на вход данного блока могут поступать 16 конкретных комплексных чисел. Имея фиксированные данные, можно заранее рассчитать каждый коэффициент в вышеописанных формулах, что в разы увеличит скорость вычислений, хоть и потребует хранить большой объем данных.

Заранее определив число гармоник, можно рассчитать, сколько времени уйдет на проход полного цикла работы.

Если использовать стандартные алгоритмы ДПФ, сложность все равно будет квадратичной, но данные вычисления позволяют сократить число операций на каждом пункте вычислений до одного сложения.

Тесты показали, что данный алгоритм быстрее стандартных вычислений в 5 раз.

Итог выполнения оптимизированного ОДПФ 10000 раз (Для усреднения данные вычисления проходили 1000 раз):



Итог выполнения стандартного ОДПФ с такими же параметрами:



Для последующей оптимизации требуется использования алгоритмов БПФ, которые являются более сложными в реализации. Алгоритмы БПФ имеют логарифмическую сложность, в то время как стандартные ДПФ, даже оптимизированные, имеют квадратичную.

**\*\*\*Попробовать найти реализации БПФ и дописать дальше описание алгоритма\*\*\***

## **P2S**

****

Данный блок получает на вход сигналы, вычисленные в предыдущем блоке, и объединяет их для последующей передачи. По основному свойству OFDM, сигналы располагаются так, чтобы основные части сигнала не перекрывали друг друга, но при этом довольно плотно. Ширина полезной части сигнала заранее известна, поэтому заранее можно рассчитать величину разницы между соседними сигналами.

**\*\*\*Код блока P2S\*\*\***

# **Реализация алгоритма**

**\*\*\* Пока что здесь описан только старый код для тестов производительности разных версий ОДПФ. Далее тут будет описание работы самого эмулятора, который будет использовать вышеописанные функции и контролировать скорости выполнения функций(блоков программ) с возможностью параллельной работы некоторых модулей. Это нужно чтобы рассчитать необходимую производительность процессоров в зависимости от организации выполнения кода. Этот эмулятор уже написан и протестирован.\*\*\***

Программа эмулирует процесс обработки сигнала, включающий модуляцию и дискретное преобразование Фурье, с целью оценки временных характеристик алгоритма.

Программа содержит функцию Symbol\_produce, осуществляющую разложение символов из входного буфера на последовательность “кадров”, представленных беззнаковыми символами. Функция short\_modulation (код которой не представлен) предположительно выполняет модуляцию этих символов.

В основной функции main инициализируется трехмерный вектор offtDataIn, который предназначен для хранения предопределенных значений, используемых в последующих вычислениях ОДПФ, и вектор offtDataOut для хранения результатов обработки. Затем происходит заполнение offtDataIn значениями согласно формулам выше, полученными путем применения функции short\_modulation, которая возвращает комплексное число по входному значению в соответствии 16QAM, и умножения на комплексную экспоненту, соответствующую дискретному преобразованию Фурье. Таким образом формируется базис для последующего анализа сигнала.

Далее программа запрашивает ввод строки символов, определяет ее длину и приступает к серии тестовых прогонов, предназначенных для оценки времени выполнения ключевых участков кода. В каждом прогоне измеряется время, затраченное на многократное выполнение следующей последовательности операций:

1. Разложение символов из входного буфера с помощью функции Symbol\_produce, формирующей “кадры” данных.
2. Итеративное вычисление значений в векторе offtDataOut на основе данных, хранящихся в offtDataIn и определяемых значениями cadrs. Происходит суммирование комплексных чисел.
3. Применение нормировочного коэффициента к полученным значениям.

После завершения всех тестовых прогонов программа выводит в консоль время, затраченное на каждый прогон, а также среднее время выполнения. Эти данные позволяют оценить производительность и стабильность алгоритма обработки сигнала.

Листинг 1. Код программы.

*/\**

Обязательные флаги при компиляции: -lfftw3 -lm

\*/

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<cstring>

#include<iostream>

#include<math.h>

#include <bitset>

#include <vector>

#include <complex>

#include <chrono>

constexpr int N = 21; *// Число комплексных чисел, которые одновременно обрабатывает одна операция ОБПФ*

*// В будущем сюда добавятся комплексные числа для поддержания точности.*

constexpr int DATA\_LEN = 14191;

constexpr int COUNT\_CADR = 2;  *// Количество кадров*

constexpr int COUNT\_TESTS = 100;

constexpr double normalizeCoef = 1/sqrt(N); *// Нормализующий коэффициент0*

constexpr int COUNT\_SR\_TIME = 10000;

*// #define RECOWER // Раскомментировать если нужно провести обратный процесс*

unsigned long timeStart;

int i;

int iGl = 0;

void Data\_produce(char\*, int\*);

void Symbol\_produce(unsigned char\*, char\*);

std::complex<double> short\_modulation(unsigned char);

int main()

{

*// запишем комплексные числа как двумерный массив float.*

 std::vector<std::vector<std::vector<std::complex<double>>>> offtDataIn(17, std::vector<std::vector<std::complex<double>>>(N, std::vector<std::complex<double>>(N)));

 std::vector<std::complex<double>> offtDataOut(N);

*// Инициализируем начальный массив данных:*

 for (i = 0; i < 17; ++i)

 {

  for (unsigned int n = 0; n < N; ++n)

  {

   for (unsigned int k = 0; k < N; ++k)

   {

    offtDataIn[i][n][k] = short\_modulation(i) \* std::exp(std::complex<double>(0, 2 \* M\_PI \* k \* n / static\_cast<double>(N)));

   }

  }

 }

  unsigned char cadrs[COUNT\_CADR];

  char buff[DATA\_LEN] = "asdefrga";

  int maxLen = strlen(buff);

*// memset(buff, 0, sizeof(char)\*DATA\_LEN);*

  printf("Введите строку\n");

*// Data\_produce(buff, &maxLen);*

  printf("Введенная строка: %s\nЧисло символов в строке: %d\n", buff, maxLen);

*// timeStart = time(NULL);*

  long l = 0;

  long long glob = 0;

  int lss = 1;

  std::chrono::system\_clock::time\_point startTime;

  std::chrono::duration<double> srTime[COUNT\_TESTS];

  std::chrono::duration<double> nowSrTime;

  for (lss = 0; lss < COUNT\_TESTS; ++lss)

  {

   printf("Тест %d\n", lss);

   l = 0;

   startTime = std::chrono::system\_clock::now();

   for(unsigned long long kk = 0; kk < COUNT\_SR\_TIME; ++kk, iGl = 0)

   while (iGl < maxLen)

   {

    if (iGl >= maxLen)

     iGl = 0;

*// printf("\nКадр: %ld\n", iGl/6+1);*

*// printf("Строка обработки:\n");*

*// for (i = (iGl/6) \* 6; i < (iGl/6 + 1) \* 6; i++)*

*//  printf("%c", buff[i]);*

*// printf("\n");*

*// printf("Данные, обработанные на этом этапе:\n");*

*// for (i = (iGl/6) \* 6; i < (iGl/6 + 1) \* 6; i++)*

*//  printf("%8b\n", buff[i]);*

    Symbol\_produce(cadrs, buff);

*// for (i = 0; i < COUNT\_CADR; i++)*

*//  std::cout << i << " "<< std::bitset<4>(cadrs[i]) << std::endl;*

    for (i = 0; i < N; ++i)

     offtDataOut[i] = offtDataIn[cadrs[0]][i][0];

    for (unsigned int l = 0; l < N; ++l)

    {

     for (i = 1; i < COUNT\_CADR; ++i)

     {

      offtDataOut[l] += offtDataIn[cadrs[i]][l][i];

     }

     for (; i < N; ++i)

     {

      offtDataOut[l] += offtDataIn[16][l][i];

     }

     offtDataOut[l] \*= normalizeCoef;

    }

*// std::cout << "Rezult OFFT: \n ";*

*// for (i = 0; i < N; ++i)*

*//  std::cout << "Real = " << offtDataOut[i].real() << " Mnim = " << offtDataOut[i].imag() << std::endl;*

    ++l;

   }

   printf("Число пройденных циклов: %ld\n", l);

   glob += l;

   srTime[lss] = std::chrono::duration<double>(std::chrono::system\_clock::now() - startTime);

   nowSrTime += srTime[lss];

  }

  printf("Итоговое среднее число пройденных циклов: %lld\n", glob/COUNT\_TESTS);

  std::cout << "Время прохождения:" << std::endl;

  for (i = 0; i < COUNT\_TESTS; ++i)

   std::cout << srTime[i].count() << std::endl;

  std::cout << "Среднее время прохождения = " << nowSrTime.count() / COUNT\_TESTS << std::endl;

*// Здесь делаем ОБПФ для каждого кадра по отдельности*

 return 0;

}

*// Тут просто будет ввод текста и ожидание подтверждения отправки.*

void Data\_produce(char\* buff, int\* maxLen)

{

 char c;

 int i = -1;

 while((buff[++i] = getchar()) !='\n');

 buff[i] = 0;

 \*maxLen = i;

}

*// Преобразование символов в кадры по 4 бита. (Проверено)*

inline void Symbol\_produce(unsigned char\* cadrs, char\* buff)

{

    for (i = 0; i < COUNT\_CADR; ++i, ++iGl)

    {

        cadrs[i] = buff[iGl] & 0b1111;

        cadrs[++i] = buff[iGl] >> 4 & 0b1111;

    }

}

std::complex<double> short\_modulation(unsigned char cadrs)

{

 std::complex<double> symbolCoding;

 switch (cadrs) {

  case 0b0000: symbolCoding.real(-3.0); symbolCoding.imag(3.0); break; *// 0*

  case 0b0001: symbolCoding.real(-1.0); symbolCoding.imag(3.0); break; *// 1*

  case 0b0010: symbolCoding.real(-3.0); symbolCoding.imag(1.0); break; *// 2*

  case 0b0011: symbolCoding.real(-1.0); symbolCoding.imag(1.0); break; *// 3*

  case 0b0100: symbolCoding.real(-3.0); symbolCoding.imag(3.0); break; *// 4*

  case 0b0101: symbolCoding.real(-1.0); symbolCoding.imag(3.0); break; *// 5*

  case 0b0110: symbolCoding.real(-3.0); symbolCoding.imag(1.0); break; *// 6*

  case 0b0111: symbolCoding.real(-1.0); symbolCoding.imag(1.0); break; *// 7*

  case 0b1000: symbolCoding.real( 3.0); symbolCoding.imag(3.0); break; *// 8*

  case 0b1001: symbolCoding.real( 1.0); symbolCoding.imag(3.0); break; *// 9*

  case 0b1010: symbolCoding.real( 3.0); symbolCoding.imag(1.0); break; *// 10*

  case 0b1011: symbolCoding.real( 1.0); symbolCoding.imag(1.0); break; *// 11*

  case 0b1100: symbolCoding.real( 3.0); symbolCoding.imag(3.0); break; *// 12*

  case 0b1101: symbolCoding.real( 1.0); symbolCoding.imag(3.0); break; *// 13*

  case 0b1110: symbolCoding.real( 3.0); symbolCoding.imag(1.0); break; *// 14*

  case 0b1111: symbolCoding.real( 1.0); symbolCoding.imag(1.0); break; *// 15*

  default: symbolCoding.real(0.0); symbolCoding.imag(0.0);

 }

 return symbolCoding;

}

# **Заключение**

В рамках данной работы был проведен анализ и реализована оптимизация алгоритма OFDM. Также был разработан эмулятор для анализа скорости работы блоков, возможности их объединения и требуемой скорости вычислений выполняющего устройства. Далее на основе этих данных выбраны микроконтроллеры определенной скорости и реализован вышеописанный алгоритм. Следующим шагом является использование полученных данных для оптимального подбора компонентов с целью реализации более быстрого и эффективного устройства приема и передачи данных с использованием OFDM.

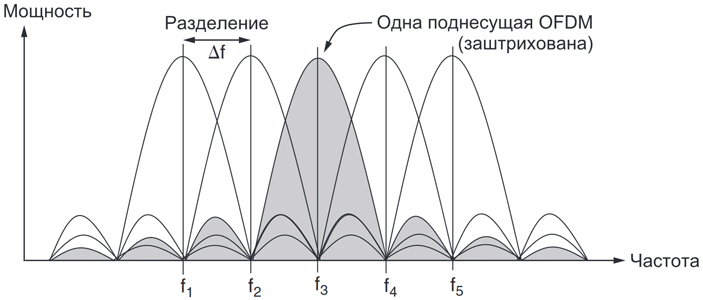
# **Приложение 1**

## **Принцип работы приемопередатчика на основе OFDM**

Прежде чем перейти к детальному рассмотрению алгоритмов формирования и обработки сигналов OFDM на микропроцессорной платформе, необходимо разобрать теоретические основы данной технологии модуляции.

Ортогональное частотное мультиплексирование (OFDM) - это метод цифровой модуляции, который делит широкополосный канал связи на множество узкополосных, ортогональных поднесущих. Вместо передачи одного высокоскоростного потока данных по всему каналу, OFDM передает несколько низкоскоростных потоков данных параллельно по этим поднесущим.

В основе OFDM лежит принцип разделения полосы пропускания на множество узкополосных поднесущих, каждая из которых модулируется независимо. Эти поднесущие выбираются таким образом, чтобы они были ортогональны друг другу. Ортогональность означает, что в точке максимальной амплитуды одной поднесущей амплитуда всех остальных поднесущих равна нулю. Это позволяет поднесущим перекрываться по частоте, не создавая взаимных помех.



Основной принцип работы OFDM заключается в следующем:

1. Высокоскоростной поток данных делится на несколько параллельных низкоскоростных потоков;
2. Каждый низкоскоростной поток данных модулирует свою поднесущую. Поднесущие выбираются таким образом, чтобы они были ортогональными друг другу;
3. Все модулированные поднесущие суммируются вместе с использованием обратного быстрого преобразования Фурье (IFFT). Это создает OFDM-сигнал во временной области;
4. К каждому OFDM-символу добавляется циклический префикс, который представляет собой копию части OFDM-символа, добавленную в начало. CP помогает бороться с межсимвольной интерференцией и межподнесущей интерференцией, вызванными многолучевым распространением.

На приемной стороне выполняются обратные операции: удаление CP, быстрое преобразование Фурье (FFT), демодуляция поднесущих и объединение низкоскоростных потоков данных в высокоскоростной поток.

Преимущества и недостатки OFDM:

Преимущества:

* OFDM эффективно борется с многолучевым распространением, так как узкополосные поднесущие менее подвержены частотно-избирательному замиранию (явление, при котором разные частотные компоненты сигнала затухают по-разному из-за многолучевого распространения);
* Так как поднесущие ортогональны друг другу, данные плотно упаковываются в частотный диапазон;
* Простота реализации с использованием БПФ/ОБПФ;
* OFDM позволяет адаптировать параметры передачи (модуляцию, кодирование) к условиям канала для каждой поднесущей.

Недостатки:

* Небольшие отклонения частоты могут привести к потере ортогональности между поднесущими и ухудшению производительности системы.
* OFDM-сигнал может иметь высокий пик-фактор, что требует использования линейных усилителей мощности.
* Фазовый шум может привести к межподнесущей интерференции (ICI).

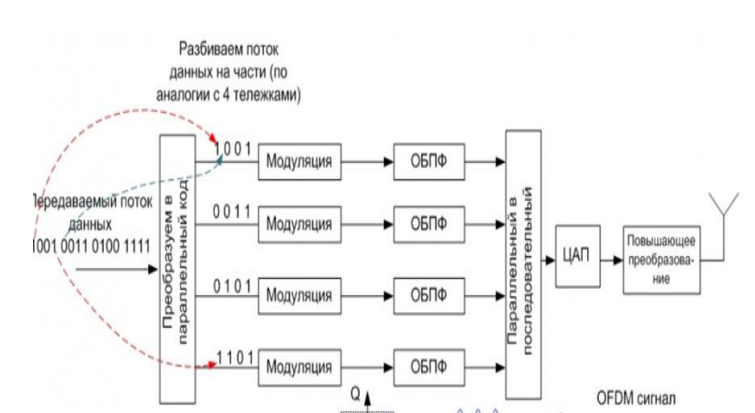
Пример:

Представим, что существует полоса частот шириной 1 МГц, которую необходимо использовать для передачи данных. Вместо того чтобы использовать один широкий канал, эта полоса разделяется, допустим, на 128 узких каналов (поднесущих) шириной примерно 7.8 кГц каждый. Каждая из этих поднесущих модулируется независимо с использованием определенного алгоритма модуляции.

Таким образом, вместо передачи одного символа с высокой скоростью передачи данных, передается 128 символов с более низкой скоростью передачи данных на каждой поднесущей. Это снижает влияние многолучевого распространения, так как узкополосные поднесущие менее подвержены частотно-избирательному замиранию (явление, при котором разные частотные компоненты сигнала затухают по-разному из-за многолучевого распространения).

В итоге, суммарная скорость передачи данных остается прежней, но система становится более устойчивой к помехам и многолучевому распространению.

Пример строения блоков:



## **Существующие реализации**

Пример 1

“Hybrid Analog and Digital Beamforming for mmWave OFDM Large-Scale Antenna Arrays” (“Гибридное аналого-цифровое формирование луча для OFDM-систем миллиметрового диапазона с крупномасштабными антенными решетками”):

Рассмотрим крупномасштабную MIMO-систему на основе OFDM, в которой передатчик, оснащенный Nt антеннами, обслуживает приемник, оснащенный Nr антеннами, отправляя Ns символов данных на каждый частотный тон. В общем случае, количество символов данных может быть различным для разных частотных тонов, однако для простоты, в этой статье внимание сосредоточено на случае с равным количеством потоков данных для всех поднесущих, поскольку для систем миллиметрового диапазона с сильно коррелированными каналами все подканалы обычно имеют низкий ранг. Более того, в практических крупномасштабных MIMO-системах количество доступных РЧ-цепей1, NRF, обычно намного меньше, чем количество антенн приемопередатчика, т.е. NRF min(Nt; Nr). Это исключает возможность реализации традиционных полностью цифровых методов формирования луча, которые требуют одной РЧ-цепи на каждый антенный элемент. В этой статье используется архитектура гибридного аналого-цифрового формирования луча, показанную на рисунке 1, для решения проблемы ограничения аппаратных ресурсов. В архитектуре гибридного формирования луча общий формирователь луча состоит из низкоразмерного цифрового (в полосе основной частоты) формирователя луча и высокоразмерного аналогового (радиочастотного) формирователя луча, реализованного с использованием простых аналоговых компонентов.

**A. Модель сигнала в гибридном формировании луча**

В архитектуре гибридного формирования луча на основе OFDM, показанной на рисунке 1, передатчик сначала предварительно кодирует Ns символов данных s[k] на каждой поднесущей k = 1, … , K, используя низкоразмерный цифровой предварительный кодировщик, VD[k] ∈ CNRF×Ns, затем преобразует сигналы во временную область, используя NRF K-точечных обратных быстрых преобразований Фурье (IFFT). После добавления циклических префиксов, передатчик использует аналоговую матрицу предварительного кодирования VRF ∈ CNt×NRF, чтобы сгенерировать финальный передаваемый сигнал. Поскольку аналоговый предварительный кодировщик является модулем, следующим за IFFT, аналоговый предварительный кодировщик идентичен для всех поднесущих. Это является ключевой проблемой при проектировании гибридных формирователей луча в OFDM системах по сравнению с однонесущими системами. Учитывая это, финальный передаваемый сигнал на поднесущей k равен:

x[k] = VRF VD[k] s[k], (1)\*

где s[k] ∈ CNs×1 — вектор передаваемых символов данных на поднесущей k с E{s[k]s[k]H} = INs.

Предполагая модель блочно-замирающего канала, полученный сигнал на поднесущей k равен:

y[k] = H[k] VRF VD[k] s[k] + z[k], (2)\*

где H[k] ∈ CNr×Nt и z[k] ∼ CN (0, σ2INr) являются, соответственно, матрицей канала и аддитивным белым гауссовским шумом для поднесущей k.

На приемнике, полученные сигналы всех поднесущих изначально обрабатываются с использованием аналогового объединителя, WRF ∈ CNr×NRF. Затем циклический префикс удаляется, и NRF K-точечные быстрые преобразования Фурье (FFT) применяются для восстановления сигналов в частотной области. Наконец, используя низкоразмерный цифровой объединитель для каждой поднесущей, WD[k] ∈ CNRF×Ns, приемник получает финальный обработанный сигнал в виде:

y˜[k] = WtH[k] H[k] Vt[k] s[k] + WtH[k] z[k], (3)\*

в котором Vt[k] = VRF VD[k] и Wt[k] = WRF WD[k] являются общим гибридным предварительным кодировщиком и объединителем для k-ой поднесущей, соответственно.

• s[k]: Вектор символов данных, которые нужно передать на k-ой поднесущей. E{s[k]s[k]H} = INs означает, что символы некоррелированы и имеют единичную мощность. INs - единичная матрица размера Ns x Ns.

• VD[k]: Матрица цифрового предварительного кодирования (digital precoder) для k-ой поднесущей. Эта матрица применяется в цифровой области после модуляции OFDM. Размерность NRF x Ns означает, что она преобразует Ns входных потоков в NRF потоков для отправки через РЧ-цепи.

• VRF: Матрица аналогового предварительного кодирования (analog precoder). Важный момент: она одинакова для всех поднесущих. Это ограничение связано с тем, что аналоговая часть формирователя луча работает после IFFT, то есть во временной области, где информация о поднесущих уже смешана. Размерность Nt x NRF означает, что она преобразует NRF потоков, выходящих из РЧ-цепей, в Nt сигналов для отправки через антенны.

• x[k]: Передаваемый сигнал на k-ой поднесущей.

• H[k]: Матрица канала для k-ой поднесущей. Описывает, как сигнал распространяется от передатчика к приемнику на этой частоте. Размерность Nr x Nt означает, что она отображает Nt передаваемых сигналов на Nr принимаемых сигналов.

• z[k]: Аддитивный белый гауссовский шум (AWGN) на k-ой поднесущей. CN (0, σ2INr) означает, что шум имеет комплексное нормальное распределение со средним значением 0 и дисперсией σ2 на каждой принимающей антенне.

• y[k]: Принимаемый сигнал на k-ой поднесущей.

• WRF: Матрица аналогового объединения (analog combiner) на приемнике. Аналогично VRF, она одинакова для всех поднесущих. Размерность Nr x NRF означает, что она преобразует Nr принимаемых сигналов в NRF потоков для обработки в РЧ-цепях.

• WD[k]: Матрица цифрового объединения (digital combiner) для k-ой поднесущей. Размерность NRF x Ns означает, что она преобразует NRF потоков, выходящих из РЧ-цепей, в Ns выходных потоков данных.

• y˜[k]: Финальный обработанный сигнал на k-ой поднесущей.

• Vt[k]: Общий гибридный предварительный кодировщик.

• Wt[k]: Общий гибридный объединитель.

• Блочно-замирающий канал: Модель канала, в которой канал остается постоянным в течение некоторого времени (блока), а затем изменяется случайным образом.



**B. Структура аналогового формирователя луча**

Аналоговая часть гибридного формирователя луча обычно реализуется с использованием простых аналоговых компонентов, таких как аналоговые сумматоры и аналоговые фазовращатели, которые могут изменять только фазу сигналов. Это приводит к некоторым ограничениям на матрицу аналогового формирования луча в зависимости от структуры аналогового формирователя луча. В этой статье мы сосредоточимся на двух широко используемых структурах аналогового формирования луча: полностью-связанной и частично-связанной структурах.

• Полностью-связанная архитектура: В этой структуре каждая РЧ-цепь соединена со всеми антенными элементами через сеть фазовращателей, как показано на рисунке 2(a). Это приводит к ограничению постоянного модуля нормы на все элементы матриц аналогового формирования луча, то есть |VRF(i, j)| = |WRF(i, j)| = 1, ∀i, j. Кроме того, как видно на рисунке 2(a), общее количество фазовращателей в этой архитектуре составляет NtNRF.

• Частично-связанная архитектура: В отличие от полностью-связанной структуры, в частично-связанной структуре каждая РЧ-цепь соединена только с одной подрешеткой с Nt/ NRF антеннами, как показано на рисунке 2(b). Следовательно, матрица аналогового формирования луча в частично-связанной архитектуре имеет блочно-диагональный формат, то есть аналоговый прекодер имеет вид:

VRF =

[ v1 0 ... 0 ]

[ 0 v2 0 ]

[ 0 0 ... ]

[ . . 0 ]

[ 0 0 ... vNRF ]

(4)

где каждый элемент вектора vi удовлетворяет ограничению постоянного модуля. Общее количество фазовращателей в этой структуре составляет Nt, что означает, что сложность аппаратной части РЧ-формирователя луча снижается в NRF раз.

• Analog Adders (Аналоговые сумматоры): Компоненты, которые складывают аналоговые сигналы.

• Analog Phase Shifters (Аналоговые фазовращатели): Компоненты, которые изменяют фазу аналогового сигнала. Они являются ключевыми компонентами для формирования луча, так как позволяют управлять направлением излучения/приема антенной решетки.

• Fully-connected Architecture (Полностью-связанная архитектура): Каждая РЧ-цепь соединена со всеми антеннами. Это обеспечивает большую гибкость в формировании луча, но требует большего количества фазовращателей.

• Partially-connected Architecture (Частично-связанная архитектура): Каждая РЧ-цепь соединена только с подмножеством антенн (подрешеткой). Это упрощает аппаратную реализацию (меньше фазовращателей), но может снижать гибкость формирования луча.

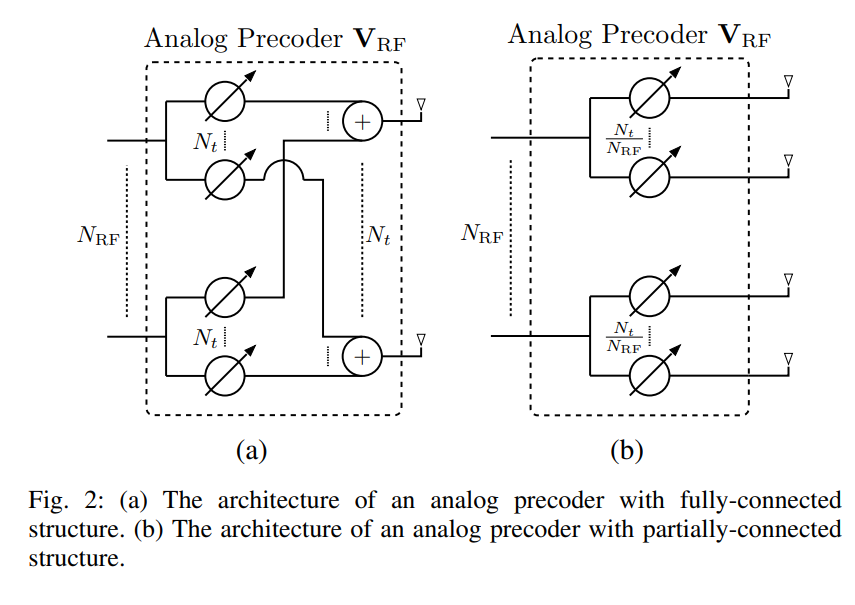
• Constant Modulus Constraint (Ограничение постоянного модуля): |VRF(i, j)| = |WRF(i, j)| = 1. Это означает, что амплитуда сигнала на выходе фазовращателя остается неизменной (равной 1), изменяется только его фаза. Это упрощает проектирование фазовращателей.

• NtNRF: Общее количество фазовращателей в полностью-связанной архитектуре.

• Block Diagonal Format (Блочно-диагональный формат): Матрица, в которой элементы за пределами диагональных блоков равны нулю.

• Nt/NRF: Количество антенн в каждой подрешетке в частично-связанной архитектуре.

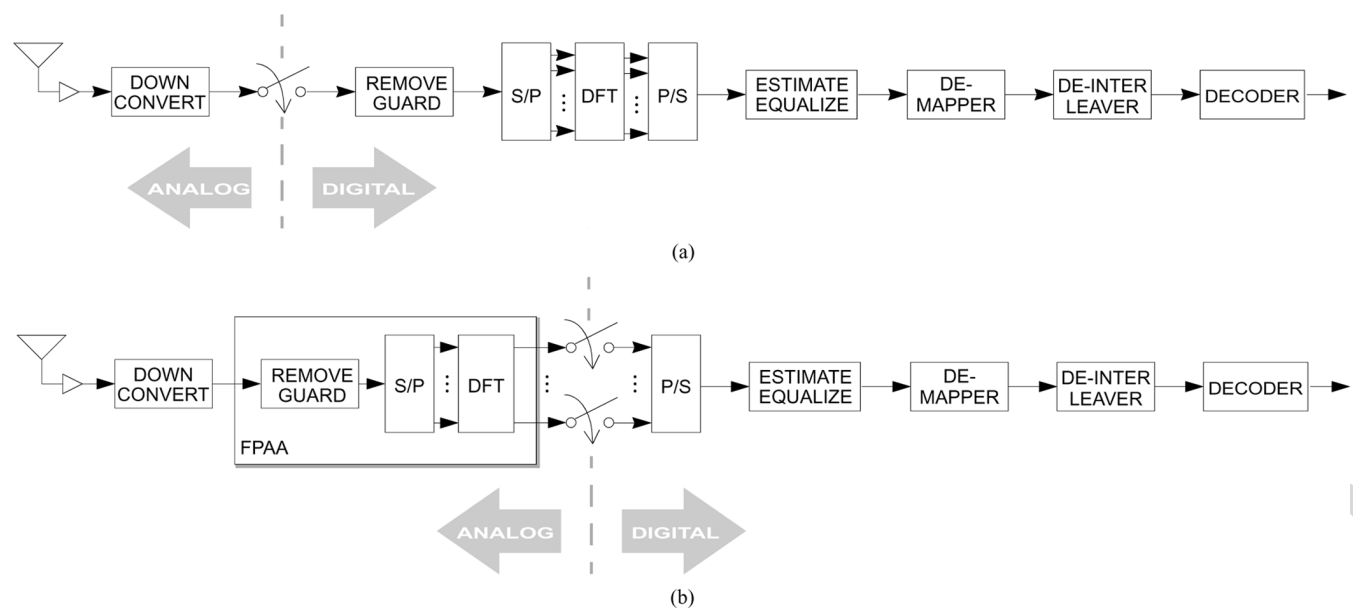
• Nt: Общее количество фазовращателей в частично-связанной архитектуре.



В 19 показано, что существует компромисс между производительностью и сложностью при выборе вышеупомянутых структур. Полностью-связанная структура может достичь полного усиления формирования луча с полным контролем фазы, в то время как частично-связанная структура имеет ограниченный контроль фазы, и, следовательно, не может достичь полного усиления формирования луча во всех случаях. С другой стороны, сложность аппаратной реализации и энергопотребление частично-связанной архитектуры значительно ниже по сравнению с таковыми у полностью-связанной структуры.

**Пример 2**

Low-Power Discrete Fourier Transform for OFDM: A Programmable Analog Approach /Малопотребляющее дискретное преобразование Фурье для OFDM: Программируемый аналоговый подход



*Рисунок 1. OFDM приемник. (a) Традиционная реализация, в которой дискретизация происходит перед цифровым ДПФ. (b) Предлагаемая реализация, в которой дискретизация происходит после аналогового ДПФ.*

На рисунке 1(a) мы показываем упрощенную блок-схему обычного OFDM-приемника, например, для системы 802.11 a/g, где принятый сигнал дискретизируется сразу после понижения частоты. Демодуляция OFDM выполняется цифровым способом с использованием ДПФ (DFT).

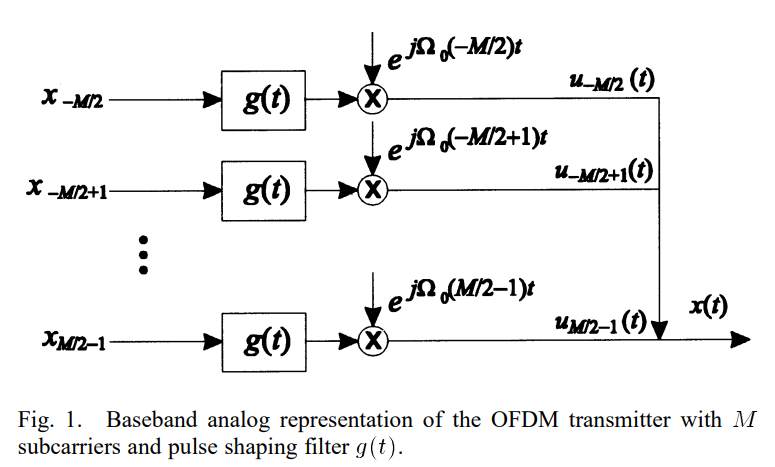
В качестве альтернативы предлагается аналоговая реализация, как показано на рисунке 1(b), где выход понижающего преобразователя подается непосредственно на FPAA (программируемую аналоговую матрицу), без дискретизации. Функциональность ДПФ реализуется аналоговым способом с использованием FPAA. Выходы FPAA — по одному на каждую поднесущую — дискретизируются отдельно.

Помимо снижения энергопотребления при аналоговой реализации ДПФ, важным преимуществом предлагаемой структуры приемника на рисунке 1(b) является значительное облегчение требований к скорости и точности аналого-цифрового преобразователя (АЦП). В частности, АЦП обычного приемника, показанного на рисунке 1(a), должен дискретизировать сигнал со скоростью, равной полной полосе пропускания сигнала, а его точность должна быть высокой (порядка 10 бит и более), чтобы соответствовать широкому динамическому диапазону и гауссовскому распределению OFDM-сигналов. В отличие от этого, предлагаемый приемник на рисунке 1(b) имеет не один АЦП, а N АЦП, по одному на каждую поднесущую, с частотой дискретизации, меньшей в N раз. Более того, каждый выход ДПФ представляет собой сигнал с конечным алфавитом, который может быть дискретизирован со значительно меньшей битовой точностью 14. По сравнению с АЦП на рисунке 1(a), каждый АЦП на рисунке 1(b) требует частоты дискретизации, которая ниже в N раз, и количества битов точности, которое меньше в три раза или больше, в зависимости от размера алфавита модуляции. Оба эффекта приводят к снижению энергопотребления АЦП, хотя точная величина зависит от типа структуры АЦП; энергопотребление АЦП находится между линейной и квадратичной функцией частоты дискретизации 19.

Таким образом, предлагаемый приемник на рисунке 1(b) выгоден не только из-за снижения энергопотребления при аналоговой реализации ДПФ, но и из-за дополнительной экономии энергии, возникающей из-за более низких требований к скорости и битовой точности последующего АЦП. Хотя здесь сосредоточены на стороне приемника, кратко отмечается, что те же преимущества справедливы и на стороне передатчика, где выходы от модулятора символов модулируются с помощью блока обратного ДПФ (IDFT), который имеет ту же структуру, что и блок ДПФ, но с другими коэффициентами. Помимо экономии энергии при аналоговой реализации IDFT, перенос блока IDFT после цифро-аналогового преобразователя (ЦАП) позволяет OFDM-передатчику заменить полноскоростной высокоточный ЦАП на N отдельных ЦАП, каждый из которых работает на частоте в 1/N раз ниже и с меньшей битовой точностью.

• FPAA (Field-Programmable Analog Array) / ПААМ (Программируемая аналоговая матрица): Интегральная схема, содержащая конфигурируемые аналоговые блоки, такие как операционные усилители, резисторы, конденсаторы и т. д., которые могут быть соединены для реализации различных аналоговых функций.

**Пример 3**

OFDM Transmitters: Analog Representation and DFT-Based Implementation” (“OFDM передатчики OFDM: Аналоговое представление и реализация на основе ДПФ”):

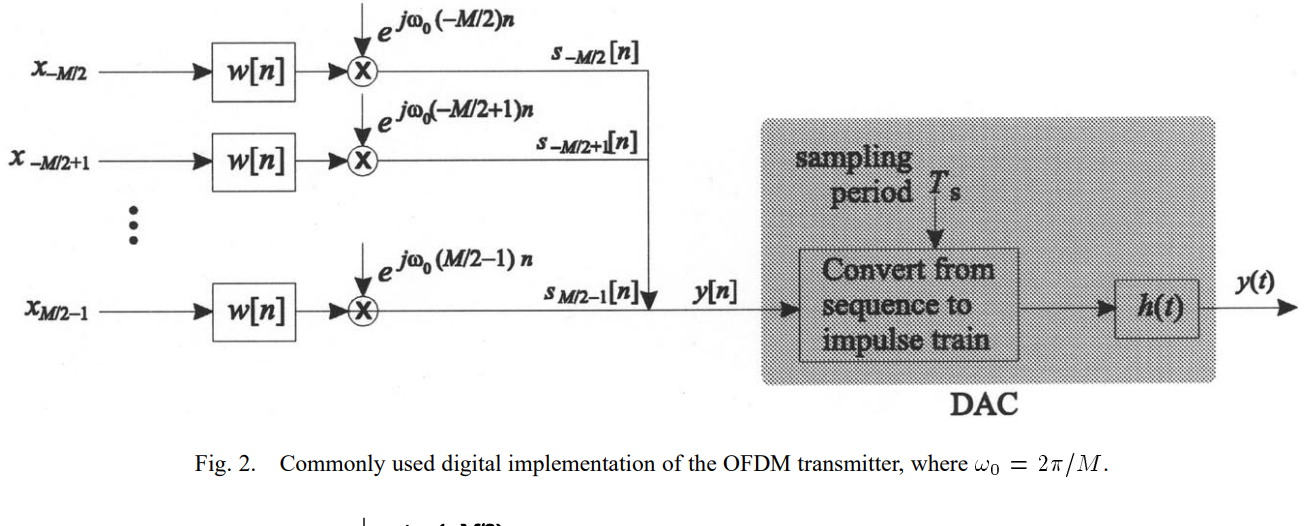
На рисунке 1 показана схема аналогового OFDM-передатчика с M поднесущими. Обозначив расстояние между поднесущими как Ω0, выход передатчика задается выражением:

x(t) = Σk=-M/2M/2-1 xk g(t) ejkΩ0t (1)

предполагая, что M четное. Фильтр формирования импульса g(t) обычно представляет собой прямоугольный импульс длительностью T0 = 2π/Ω0. Многие исследования OFDM-систем проводятся с использованием выражения (1), например, исследование спектрального спада выходных сигналов OFDM-передатчиков 3, 4, влияние смещения несущей частоты 5 и пик-факторов выходных сигналов передатчика 6. Был предложен ряд непрямоугольных форм импульсов g(t) для улучшения спектрального спада передаваемого сигнала x(t), например, в 4 и 6. Хотя аналоговое представление удобно для анализа, на практике модуляция поднесущих осуществляется в дискретном времени. Такой передатчик (см. рисунок 2) состоит из двух частей 3: цифро-аналогового преобразователя (ЦАП) и части, выполняющей цифровую модуляцию поднесущих, которая может быть эффективно реализована с использованием матрицы обратного дискретного преобразования Фурье (ОДПФ) размером M x M. Период дискретизации равен Ts = T0/ M, и дискретная последовательность w[n], показанная на рисунке 2, обычно представляет собой прямоугольное окно длины M.

Предположим, что фильтр восстановления ЦАП имеет импульсную характеристику h(t), как показано на рисунке 2. Выход ЦАП с периодом дискретизации Ts задается выражением:

y(t) = Σn=-∞∞ y[n] h(t - nTs)

где y[n] = w[n] Σk=-M/2M/2-1 xk ej(2π/M)kn: (2)

Как указано на рисунке 2, y[n] является входом ЦАП. Форма волны y(t) напоминает форму волны x(t) — выходного сигнала аналогового передатчика — особенно при больших значениях M, 1. В [3, гл. 5] упоминается, что при использовании цифровой реализации с идеальным низкочастотным фильтром восстановления h(t) в ЦАП, формирующий фильтр g(t) больше не является прямоугольным импульсом. В литературе ранее не было указано точной связи между передатчиком на основе ДПФ и аналоговым представлением.

• x(t): Аналоговое представление OFDM-сигнала во времени.

• M: Количество поднесущих.

• Ω0: Расстояние между поднесущими (в радианах в секунду).

• xk: Комплексный символ данных, модулирующий k-ую поднесущую.

• g(t): Импульсная характеристика формирующего фильтра. Определяет форму импульса каждой поднесущей во времени. Прямоугольный импульс является наиболее распространенным, но другие формы могут улучшить спектральные характеристики.

• ejkΩ0t: Комплексная экспонента, представляющая k-ую поднесущую.

• w[n]: Окно, применяемое к дискретным данным перед преобразованием в аналоговый сигнал.

• y[n]: Дискретные отсчеты, подаваемые на вход ЦАП.

• y(t): Аналоговый сигнал на выходе ЦАП.

• h(t): Импульсная характеристика фильтра восстановления ЦАП. Этот фильтр сглаживает ступенчатый выход ЦАП, чтобы приблизиться к исходному аналоговому сигналу.

• Ts: Период дискретизации.

В этой статье рассматриваются условия, при которых передатчик на основе ДПФ, показанный на рисунке 2, допускает аналоговое представление, показанное на рисунке 1. Для случая передатчика на основе ДПФ с прямоугольным окном w[n] и идеальным низкочастотным фильтром h(t) аналоговое представление не существует. Известно, что, если мы выберем прямоугольное окно g(t) в аналоговом представлении, выходной сигнал будет близок к выходному сигналу передатчика на основе ДПФ во временном окне, но выходные сигналы двух передатчиков могут иметь значительные различия в спектральном спаде. Покажем, что на самом деле, когда аналоговый передатчик имеет прямоугольный g(t), реализация на основе ДПФ не существует, независимо от выбора w[n] и h(t). Для произвольного импульса g(t) эквивалентная цифровая реализация, вообще говоря, не существует. Аналоговый и передатчик на основе ДПФ эквивалентны только в некоторых ограниченных случаях. Поэтому анализ OFDM-систем непосредственно с использованием схемы на основе ДПФ, показанной на рисунке 2, более полезен, чем использование аналоговой схемы, показанной на рисунке 1. Например, разработка w[n] и h(t) для улучшения спектрального спада более полезна, чем разработка g(t). Будет выведено необходимое и достаточное условие эквивалентности аналогового и передатчика на основе ДПФ. Будет приведен пример набора g(t), w[n] и h(t), удовлетворяющих этому условию.

Разбор и пояснения:

• Эквивалентность аналогового и цифрового представлений: Основная тема статьи - выяснить, когда аналоговый и цифровой (на основе ДПФ) OFDM-передатчики дают одинаковый результат.

• Прямоугольное окно w[n] и идеальный фильтр h(t): Это часто используемые упрощения в анализе, но в этой статье утверждается, что в этом случае не существует эквивалентного аналогового представления.

• Спектральный спад: Скорость уменьшения мощности сигнала по мере удаления от несущей частоты. Хороший спектральный спад важен для уменьшения помех соседним каналам.

• Разработка w[n] и h(t) вместо g(t): В статье утверждается, что для улучшения характеристик OFDM-передатчика лучше настраивать окно w[n] в цифровой части и фильтр восстановления h(t) в ЦАП, чем пытаться оптимизировать формирующий фильтр g(t) в аналоговом представлении.

• Необходимое и достаточное условие: Статья обещает вывести математическое условие, которое должно выполняться, чтобы аналоговый и цифровой передатчики были эквивалентны.

# **Приложение 2.**

### **Модуляция**

Модуляция поднесущих - это процесс изменения одного или нескольких параметров несущего сигнала (амплитуды, частоты, фазы) в соответствии с информацией, которую необходимо передать. В OFDM модуляция выполняется на каждой поднесущей независимо, что позволяет передавать несколько параллельных потоков данных. Выбор метода модуляции влияет на скорость передачи данных, устойчивость к шуму и сложность системы.

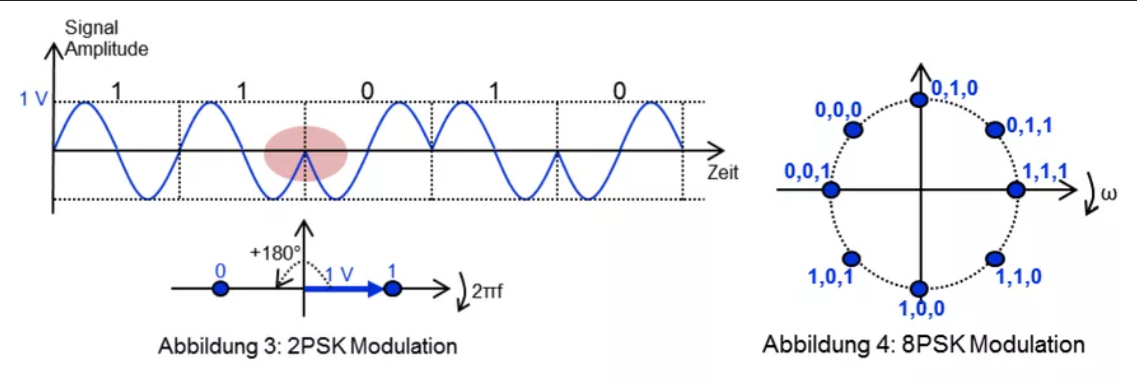
Рассмотрим наиболее распространенные методы модуляции, используемые в OFDM:

**Фазовая модуляция (Phase-Shift Keying - PSK)**

Данные кодируются путем изменения фазы несущей частоты. Амплитуда остается постоянной.

Типы PSK:

* BPSK (Binary PSK): Использует две фазы (0 и 180 градусов) для представления битов 0 и 1.
* QPSK (Quadrature PSK): Использует четыре фазы (0, 90, 180 и 270 градусов) для представления двух битов информации на символ.
* 8-PSK: Использует восемь фаз для представления трех битов на символ.
* 16-PSK: Использует шестнадцать фаз для представления четырех битов на символ.



Пример в виде алгоритма QPSK:

* Входящий поток битов делится на пары.
* Каждой паре битов присваивается определенная фаза:

00 -> 45°

01 -> 135°

10 -> 225°

11 -> 315°

* Сигнал генерируется как: , где - амплитуда, - частота поднесущей, - фаза, соответствующая дибиту.

Преимущества: Простота реализации, постоянная огибающая (что упрощает работу усилителей мощности).

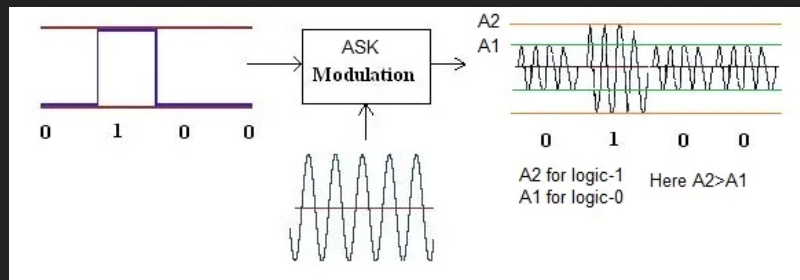
Недостатки: Ограниченная спектральная эффективность (малое количество битов на символ), чувствительность к фазовому шуму.

Вычислительные ресурсы: Низкие. Требуются простые операции для генерации синусоидальных сигналов с определенной фазой.

**Амплитудная модуляция (Amplitude-Shift Keying - ASK)**

Данные кодируются путем изменения амплитуды несущей частоты. Фаза остается постоянной.

* OOK (On-Off Keying): Простейший тип ASK, где наличие сигнала представляет бит 1, а отсутствие сигнала - бит 0.



Пример алгоритма:

Бит 1: (сигнал есть)

Бит 0: s(t) = 0 (сигнала нет)

Преимущества: Очень простая реализация.

Недостатки: Низкая спектральная эффективность, очень чувствительна к шуму и изменениям уровня сигнала. Практически не используется в OFDM.

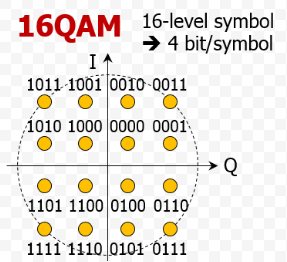
Вычислительные ресурсы: Очень низкие.

**Квадратурная амплитудная модуляция (Quadrature Amplitude Modulation - QAM)**

Данные кодируются путем изменения как амплитуды, так и фазы несущей частоты.

Типы QAM:

* 16-QAM: Использует 16 комбинаций амплитуды и фазы для представления четырех битов на символ.
* 64-QAM: Использует 64 комбинации амплитуды и фазы для представления шести битов на символ.
* 256-QAM: Использует 256 комбинаций амплитуды и фазы для представления восьми битов на символ.



Пример алгоритма 16-QAM:

1. Входящий поток битов делится на группы по 4 бита.
2. Каждой группе битов присваивается определенная амплитуда и фаза в соответствии с картой созвездия.
3. Сигнал генерируется как сумма двух квадратурных компонент: где *I* и - синфазная и квадратурная компоненты, определяемые амплитудой и фазой.

Преимущества: Высокая спектральная эффективность (большое количество битов на символ).

Недостатки: Более сложная реализация, чувствительность к шуму и искажениям, требует линейных усилителей мощности.

Вычислительные ресурсы: Средние. Требуются более сложные операции для генерации сигналов с переменной амплитудой и фазой.

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Биты | Устойчивость к шуму | Вычислительные ресурсы |
| BPSK | 1 | Высокая | Низкие |
| QPSK | 2 | Средняя | Низкие |
| 16-QAM | 4 | Средняя | Средние |
| 64-QAM | 6 | Низкая | Средние |
| 256-QAM | 8 | Очень низкая | Высокие |

Таблица 1. Сравнение методов модуляции

Выбор метода модуляции для каждой поднесущей в OFDM зависит от:

* Состояния канала: Если канал характеризуется высоким SNR, можно использовать более сложные схемы модуляции (например, 64-QAM) для увеличения скорости передачи данных. Если канал зашумлен, следует использовать более надежные схемы (например, QPSK или BPSK).
* Требований к скорости передачи данных: Если требуется высокая скорость передачи данных, используются схемы модуляции с большим количеством битов на символ.
* Требований к надежности: Если требуется высокая надежность передачи данных, используются схемы модуляции с меньшим количеством битов на символ и канальным кодированием.

# **Приложение 3.**

## **Алгоритмы БПФ.**

Преобразование Фурье является фундаментальным инструментом в цифровой обработке сигналов. Оно позволяет разложить сигнал во временной области на его составляющие частоты, предоставляя информацию о частотном спектре сигнала. Дискретное преобразование Фурье является дискретной версией преобразование Фурье и применяется к дискретным сигналам, полученным, например, в результате дискретизации аналогового сигнала.

Однако, непосредственное вычисление ДПФ требует большого количества вычислительных операций (порядка O(N2) для сигнала длиной N), что делает его неэффективным для обработки больших объемов данных. Быстрое преобразование Фурье - это класс алгоритмов, разработанных для эффективного вычисления ДПФ. БПФ значительно снижает вычислительную сложность до O(N\*log(N)), что делает его применимым для обработки сигналов в реальном времени и в различных приложениях.

Для дешифровки требуется дискретное преобразование Фурье последовательности x(n) длиной N определяется как:

для k = 0, 1, ..., N-1, где n = 0 до N-1,

- n-ый отсчет входной последовательности (сигнала во временной области);

- k-ый отсчет выходной последовательности (спектр сигнала в частотной области);

- длина последовательности;

Обратное дискретное преобразование Фурье (ОДПФ) определяется как:

для n = 0, 1, ..., N-1, где k = 0 до N-1.

Цель преобразования Фурье – это получить коэффициенты ряда Фурье. Общий вид коэффициентов ряда Фурье представлен ниже.

**Алгоритм Кули-Тьюки (Cooley-Tukey Algorithm**)

Является наиболее известным и часто используемым алгоритмом БПФ. Разделяет ДПФ на более мелкие ДПФ, которые затем рекурсивно вычисляются.

Наиболее эффективен, когда длина последовательности N является степенью двойки (radix-2 БПФ). Существуют также варианты с другими основаниями (radix-4, radix-8 и т.д.).

Имеет вычислительную сложность O(Nlog(N)).

**Алгоритм Radix-2 (основание 2)**

Алгоритм Radix-2 БПФ является частным случаем алгоритма Кули-Тьюки, который наиболее эффективно работает, когда длина входной последовательности N является степенью двойки (N = 2m, где m - целое число). Он основан на декомпозиции ДПФ на более мелкие ДПФ размером 2 (бабочки). Алгоритм рекурсивно разделяет ДПФ на две половины: четные и нечетные индексы, пока не достигнет ДПФ размером 2.

1. Переупорядочивание входной последовательности x[n] в соответствии с обращенным битовым представлением индексов.
2. Рекурсивное вычисление ДПФ размером 2 (бабочки) на каждой стадии. «Бабочка» - это базовая вычислительная единица, выполняющая следующие операции:

где: A и B - входы “бабочки” (результаты предыдущей стадии).

- вращающийся множитель.

N - размер ДПФ на текущей стадии.

1. Повторение шага 2 для каждой стадии рекурсии, пока не будут вычислены все значения ДПФ. Количество стадий равно log2(N).

Вычислительная сложность: O(N log2 N) комплексных умножений и O(N log2 N) комплексных сложений.

Отличия от Radix-4 и Radix-8:

Radix-4: Декомпозирует ДПФ на более мелкие ДПФ размером 4. Требует, чтобы длина последовательности была степенью 4 (N = 4m). Выполняет более сложные “бабочки”, но требует меньше стадий (log4 N).

Radix-8: Декомпозирует ДПФ на более мелкие ДПФ размером 8. Требует, чтобы длина последовательности была степенью 8 (N = 8m). Еще более сложные “бабочки” и еще меньше стадий (log8 N).

В общем случае, алгоритмы с большим основанием (Radix-4, Radix-8) могут быть более эффективными, чем Radix-2, за счет уменьшения количества стадий. Однако, они требуют более сложной реализации “бабочек” и более строгих ограничений на длину последовательности.

**Алгоритм БПФ Бэтчера (Bluestein’s FFT Algorithm)**

Алгоритм Бэтчера используется для вычисления ДПФ последовательностей, длина которых не является степенью двойки (или любого другого удобного основания для алгоритма Кули-Тьюки). Он основан на преобразовании ДПФ в операцию свертки, которую можно эффективно вычислить с использованием БПФ.

Алгоритм:

1. ДПФ выражается в виде свертки с использованием следующей замены:

Подставляя это в формулу ДПФ, получаем:

Это можно переписать как свертку:

где: n = 0 до N-1;

1. Вычисляются БПФ последовательностей и . Затем выполняется поэлементное умножение полученных спектров, и к результату применяется обратное БПФ.
2. Результат свертки умножается на для получения окончательного результата ДПФ.

Вычислительная сложность: O(N\*log N), так как основная часть вычислений приходится на БПФ и ОБПФ. Однако, константа перед N\*log N может быть выше, чем у алгоритма Кули-Тьюки для последовательностей, длина которых является степенью двойки.

Применяется, когда длина входной последовательности не является степенью двойки или когда требуется использовать БПФ фиксированного размера (например, из-за ограничений оборудования).

**Алгоритм Винограда(Радера) (Rader’s Algorithm)**

Алгоритм Радера используется для вычисления ДПФ последовательностей, длина которых является простым числом (N - простое число). Он также преобразует ДПФ в свертку, которую можно эффективно вычислить с использованием БПФ.

Алгоритм:

1. Поскольку N - простое число, можно найти образующий элемент *g* мультипликативной группы *ZN* \* (множество целых чисел от 1 до N-1, взаимно простых с N, с операцией умножения по модулю N). Это означает, что все числа от 1 до N-1 можно представить в виде степеней *g* по модулю N.

Выполняется замена индексов:

где n’, k’ = 1, 2, …, N-1.

1. После переиндексации формула ДПФ может быть преобразована в циклическую свертку:

Пусть , , тогда

где n' = 1 до N-1.

Последняя сумма является циклической сверткой y и h.

1. Вычисление свертки с использованием БПФ: Вычисляются БПФ последовательностей и . Затем выполняется поэлементное умножение полученных спектров, и к результату применяется обратное БПФ.
2. К результату свертки добавляется для получения окончательного результата ДПФ.

Вычислительная сложность: O(N\*log N) из-за использования БПФ для вычисления свертки. Константа перед N\*log N может быть выше, чем у алгоритма Кули-Тьюки.

Применение: Когда длина входной последовательности является простым числом.

**Применение БПФ для реализации OFDM**

БПФ играет ключевую роль в реализации OFDM. Он используется как для модуляции, так и для демодуляции сигнала.

Модуляция (Передатчик):

* Последовательность комплексных символов X[k], представляющих данные для каждой поднесущей, подается на вход обратного быстрого преобразования Фурье.
* ОБПФ преобразует частотное представление сигнала X[k] во временное представление x[n], создавая OFDM-символ.
* Выход ОБПФ представляет собой дискретный OFDM-сигнал во временной области.
* Добавление циклического префикса (CP) к OFDM-символу.

Демодуляция (Приемник):

* Принятый OFDM-сигнал (после удаления циклического префикса) подается на вход быстрого преобразования Фурье (БПФ или FFT).
* БПФ преобразует временное представление сигнала обратно в частотное представление, восстанавливая символы X[k] на каждой поднесущей.
* Символы демодулируются для получения исходных данных.

Преимущества использования БПФ в OFDM:

Эффективность: БПФ значительно снижает вычислительную сложность модуляции и демодуляции OFDM-сигналов, делая возможной реализацию систем OFDM с большим количеством поднесущих.

Простота реализации: БПФ является хорошо изученным и реализованным алгоритмом, для которого существуют оптимизированные библиотеки и аппаратные реализации.