**“Университет ИТМО”**

Факультет программной инженерии

И компьютерных технологий.

Лабораторная работа №2

Синтез помехоустойчивого кода

Вариант № 97

Выполнил

Студент группы Р31112

Шаматульский Роман Константинович

Преподаватель

Рыбаков Степан Дмитриевич

Санкт-Петербург, 2023

Оглавление

[Задание 3](#_Toc147788228)

[Основные этапы вычисления 4](#_Toc147788229)

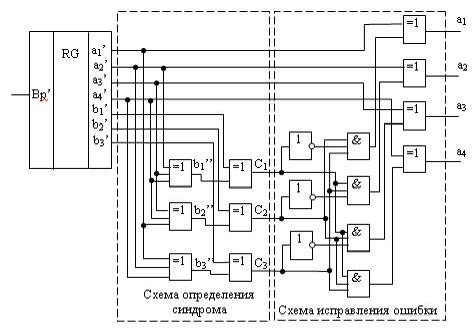
[Заключение 8](#_Toc147788230)

[Источники 9](#_Toc147788231)

# Задание

1. На основании номера варианта задания выбрать набор из 4 полученных сообщений в виде последовательности 7-символьного кода.
2. Построить схему декодирования классического кода Хэмминга (7;4), которую представить в отчёте в виде изображения.
3. Показать, исходя из выбранных вариантов сообщений (по 4 у каждого – часть No1 в варианте), имеются ли в принятом сообщении ошибки, и если имеются, то какие. **Подробно прокомментировать** и записать правильное сообщение.
4. На основании номера варианта задания выбрать 1 полученное сообщение в виде последовательности 11-символьного кода.
5. Построить схему декодирования классического кода Хэмминга (15;11), которую представить в отчёте в виде изображения.
6. Показать, исходя из выбранного варианта сообщений (по 1 у каждого – часть №2 в варианте), имеются ли в принятом сообщении ошибки, и если имеются, то какие. **Подробно прокомментировать** и записать правильное сообщение.
7. Сложить номера всех 5 вариантов заданий. **Умножить полученное число на 4**. Принять данное число как число информационных разрядов в передаваемом сообщении. Вычислить для данного числа минимальное число проверочных разрядов и коэффициент избыточности.
8. Написать программу на любом языке программирования, которая на вход получает набор из 7 цифр «0» и «1», записанных подряд, анализирует это сообщение на основе классического кода Хэмминга (7,4), а затем выдает правильное сообщение (только информационные биты) и указывает бит с ошибкой при его наличии.

# Основные этапы вычисления



*Схема декодирования классического кода Хэмминга (7;4)*

(+) – Сумматор по модулю 2.

№1 – Полученное число **1100101**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2^x | R1 | R2 | I1 | R3 | I2 | I3 | I4 | S |
| 1 | **1** | **1** | **0** | **0** | **1** | **0** | **1** | *1* |
| 2 | **1** | **1** | **0** | **0** | **1** | **0** | **1** | *0* |
| 4 | **1** | **1** | **0** | **0** | **1** | **0** | **1** | *0* |

S1 = 1 (+) 1 (+) 1 (+) 0 = 1

S2 = 1 (+) 1 (+) 1 (+) 0 = 0

S4 = 0 (+) 1 (+) 1 (+) 0 = 0

1. Переводим **001** в десятичную СС => 1 – номер ошибочного бита
2. Меняем первый бит на противоположный => **100101**

№2 – Полученное число **1011000**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2^x | R1 | R2 | I1 | R3 | I2 | I3 | I4 | S |
| 1 | **1** | **0** | **1** | **1** | **0** | **0** | **0** | *0* |
| 2 | **1** | **0** | **1** | **1** | **0** | **0** | **0** | *1* |
| 4 | **1** | **0** | **1** | **1** | **0** | **0** | **0** | *1* |

S1 = 1 (+) 1 (+) 0 (+) 0 = 0

S2 = 0 (+) 1 (+) 0 (+) 0 = 1

S4 = 1 (+) 0 (+) 0 (+) 0 = 1

1. **110** – синдром,по таблице: 6 – номер ошибочного бита
2. Меняем шестой бит на противоположный => **1011010**

№3 – Полученное число **1100011**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2^x | R1 | R2 | I1 | R3 | I2 | I3 | I4 | S |
| 1 | **1** | **1** | **0** | **0** | **0** | **1** | **1** | *0* |
| 2 | **1** | **1** | **0** | **0** | **0** | **1** | **1** | *1* |
| 4 | **1** | **1** | **0** | **0** | **0** | **1** | **1** | *0* |

S1 = 1 (+) 0 (+) 0 (+) 1 = 0

S2 = 1 (+) 0 (+) 1 (+) 1 = 1

S4 = 0 (+) 0 (+) 1 (+) 1 = 0

1. **010** – синдром, по таблице:ошибочного бита нет.

№4 – Полученное число **1001110**

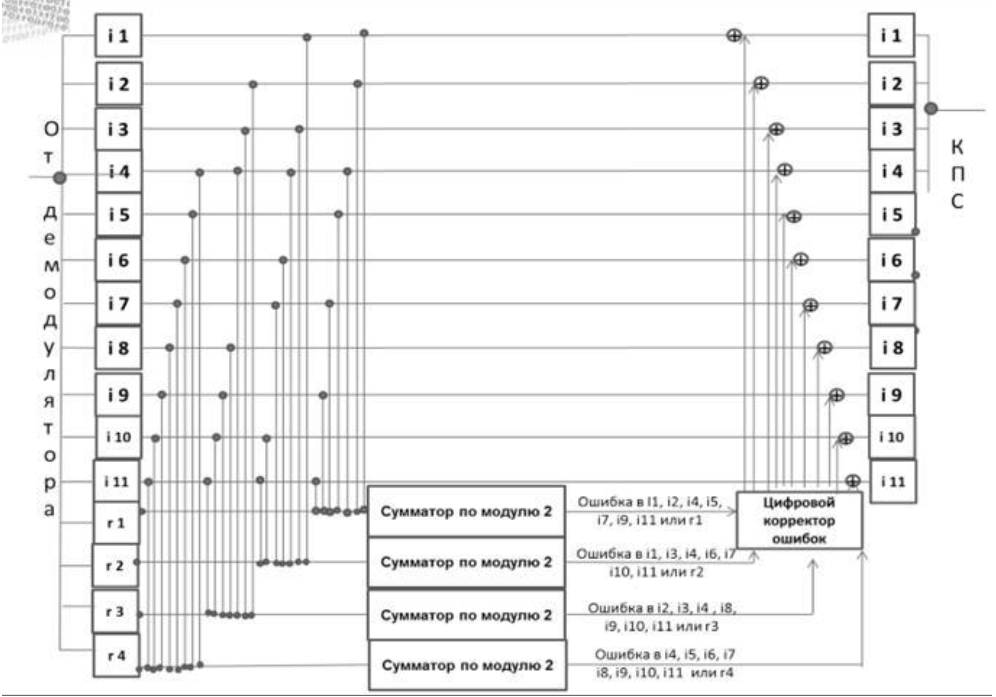
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2^x | R1 | R2 | I1 | R3 | I2 | I3 | I4 | S |
| 1 | **1** | **0** | **0** | **1** | **1** | **1** | **0** | *0* |
| 2 | **1** | **0** | **0** | **1** | **1** | **1** | **0** | *1* |
| 4 | **1** | **0** | **0** | **1** | **1** | **1** | **0** | *1* |

S1 = 1 (+) 0 (+) 1 (+) 0 = 0

S2 = 0 (+) 0 (+) 1 (+) 0 = 1

S4 = 1 (+) 1 (+) 1 (+) 0 = 1

1. **110** – синдром, по таблице: 6 – номер ошибочного бита
2. Меняем шестой бит на противоположный => **1001100**



*Схема декодирования классического кода Хэмминга (15;11)*

№5 – Полученное число **001010111110101**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2^x | R1 | R2 | I1 | R3 | I2 | I3 | I4 | R4 | I5 | I6 | I7 | I8 | I9 | I10 | I11 | S |
| 1 | **0** | **0** | **1** | **0** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **0** | **1** | *1* |
| 2 | **0** | **0** | **1** | **0** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **0** | **1** | *1* |
| 4 | **0** | **0** | **1** | **0** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **0** | **1** | *0* |
| 8 | **0** | **0** | **1** | **0** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **0** | **1** | *0* |

S1 = 0 (+) 1 (+) 1 (+) 1 (+) 1 (+) 1 (+) 1 (+) 1 = 1

S2 = 0 (+) 1 (+) 0 (+) 1 (+) 1 (+) 1 (+) 0 (+) 1 = 1

S3 = 0 (+) 1 (+) 0 (+) 1 (+) 0 (+) 1 (+) 0 (+) 1 = 0

S4 = 1 (+) 1 (+) 1 (+) 1 (+) 0 (+) 1 (+) 0 (+) 1 = 0

1. **0011** – синдром, по таблице: 3 – номер ошибочного бита
2. Меняем третий бит на противоположный => **000010111110101**

№6 – (81 + 11 + 53 + 93 + 95) \* 4 = 1332

1332 – число информационных разрядов в передаваемом сообщении.

Вычислим для данного числа минимальное число проверочных разрядов по формуле:

*2^r >= r - i – 1*

Получим *r = 11*

Найдём коэффицент избыточности по формуле:

*n = r / r + i*

Получим *n ~= 0,008*

№9 (Дополнительное задание)

Приведу пример решения с помощью java:

import java.util.HashMap;

import java.util.Scanner;

public class hamming\_lab2 {

public static void main(String[] args) {

HashMap<String, Integer> table = new HashMap<>(); // таблица для превода синдрома

table.put("001", 3);

table.put("010", 1);

table.put("011", 5);

table.put("100", 0);

table.put("101", 4);

table.put("110", 2);

table.put("111", 6);

// массив с позициями пр. разрядов

int[] r\_places = {1 - 1, 2 - 1, 4 - 1}; // -1, тк индексы идут с 0

Scanner s = new Scanner(System.in);

String request = s.nextLine(); // request - введенное число

int sumer; // сюда поместим суммы проверочных разрядов для каждой степени двойки

String syndrome = ""; // в эту строку будем помещать единицы и нули - синдром посл-ти

for (int i: r\_places) {

sumer = 0;

for (int j = i; j < request.length(); j++) {

sumer = (((j + 1) / (i + 1)) % 2 == 1) ? sumer + Character.getNumericValue(request.charAt(j)) : sumer;

// ^ выбираем проверочные разряды

}

syndrome = syndrome.concat(Integer.toString(sumer % 2)); // складываем рез-ты для каждой ст. двойки

}

String reversedSyndrome = new StringBuilder(syndrome).reverse().toString(); // развернём синдром

if (!reversedSyndrome.equals("000")){

int result = table.get(reversedSyndrome); // получаем номер ошибочного бита

char[] pattern = request.toCharArray();

pattern[result] = (pattern[result] == '1') ? '0' : '1';

request = new String(pattern);

System.out.printf("Номер ошибочного бита: %s, ответ: %s\n", result + 1, request);

}

else{

System.out.println("Ошибочных битов нет");

}

}

}

Результат работы программы:

|  |  |  |  |
| --- | --- | --- | --- |
| **Ввод** | 0010000 | 1010101 | 1110011 |
| **Вывод** | 3 | 0 | 1 |

# Заключение

В ходе выполнения данной лабораторной работы я научился понимать принцип кодирования и декодирования кода Хэмминга, освоил принцип написания помехоустойчивого кода. Полученные мной в ходе выполнения данной работы знания лягут в основу моего дальнейшего обучения и освоения информатики.

# Источники

1. Код Хэмминга. Пример работы алгоритма / tltshnik – URL: <https://habr.com/ru/articles/140611/> (Дата обращения: 09.10.23)
2. Hamming code – URL: <https://en.wikipedia.org/wiki/Hamming_code> (Дата обращения: 09.10.23)
3. Помехоустойчивое кодирование. Часть 1: код Хэмминга / ladvip – URL: <https://habr.com/ru/articles/357666/> (Дата обращения: 09.10.23)