Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Лабораторная работа № 2

“Синтез помехоустойчивого кода”

Вариант № 72

Выполнил:

Сандов Кирилл Алексеевич

Группа:

P3113

Проверил:

к.т.н преподаватель Белозубов Александр Владимирович

г. Санкт-Петербург

2022

Оглавление

Задание

1. Построить схему декодирования классического кода Хэмминга (7;4) и предоставить её изображение.

2. Показать для каждого из приведённых в таблице 1 сообщений, имеются ли в принятом сообщении ошибки, и если имеются, то какие. Подробно прокомментировать и записать правильное сообщение.

№	Сообщение						
	r1	r2	i1	r3	i2	i3	i4
1	1	1	0	1	0	1	1
2	0	1	1	1	1	1	0
3	0	0	0	1	0	0	1
4	1	0	1	0	0	1	1

Таблица 1 - «Сообщения для задания 1»

3. Построить схему декодирования классического кода Хэмминга (15;11) и предоставить её изображение.

4. Показать для сообщения, приведённого в таблице 2, имеются ли в принятом сообщении ошибки, и если имеются, то какие. Подробно прокомментировать и записать правильное сообщение.

№	Сообщение														
	r1	r2	i1	r3	i2	i3	i4	r4	i5	i6	i7	i8	i9	i10	i11
1	0	0	1	1	1	0	0	0	1	1	1	0	1	0	0

Таблица 2 – «Сообщение для задания 2»

5. Сложить номера всех 5 вариантов заданий (54, 91, 16, 51, 71). 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.

6*. Необязательное задания для получения оценки «5». Написать программу на любом языке программирования, которая на вход из командной строки 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.

Задание 1

Схема декодирования классического кода Хэмминга (7;4) представлена на рисунке 1.

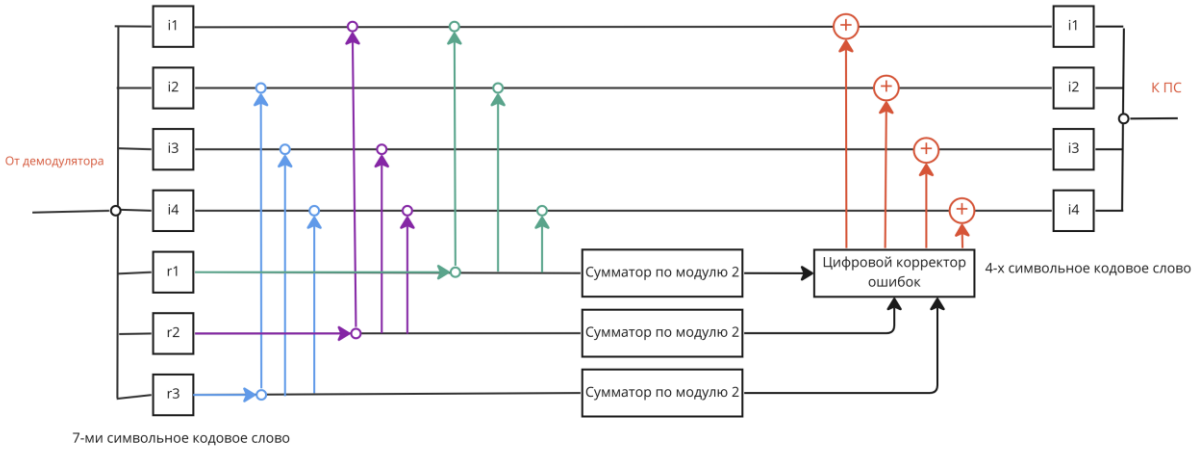

Рисунок 1 – «Схема декодирования кода Хэмминга (7;4)»

Задание 2

Для каждого сообщения будем строить таблицу кодов Хэмминга. Затем вычислим синдром S из s_1, s_2, s_3, сложив отмеченные напротив биты в таблице по модулю 2. Если S равен 0, то ошибки нет, иначе найдём бит с ошибкой, сопоставив двоичной число, состоящее из синдромов, с отметками в таблице.

Сообщение 1

Таблица кодов Хэмминга (7;4) с рассматриваемым сообщением представлена в виде таблицы 3.

	1	2	3	4	5	6	7	
Сообщение	1	1	0	1	0	1	1	
2^x	r1	r2	i1	r3	i2	i3	i4	s
1	+		+		+		+	s_1
2		+	+			+	+	s_2
4				+	+	+	+	s_3

Таблица 3 – «Таблица кодов Хэмминга (7;4)»

Вычислим синдром S:

$$s_1 = r_1 \oplus i_1 \oplus i_2 \oplus i_4 = 1 \oplus 0 \oplus 0 \oplus 1 = 0$$

$$s_2 = r_2 \oplus i_1 \oplus i_3 \oplus i_4 = 1 \oplus 0 \oplus 1 \oplus 1 = 1$$

$$s_3 = r_3 \oplus i_2 \oplus i_3 \oplus i_4 = 1 \oplus 0 \oplus 1 \oplus 1 = 1$$

$$S = \overline{s_1 s_2 s_3}_{(2)} = 011_{(2)}$$

Синдрому S соответствует столбец 6, так как отметки стоят только у s_1 и s_2. Значит, ошибка в символе i_3. Изменим его значение с 1 на 0, чтобы исправить ошибку. Получим исправленное сообщение: 1101001.

Ответ: ошибка в символе i_3, исправленное сообщение: 1101001.

Сообщение 2

Таблица кодов Хэмминга (7;4) с рассматриваемым сообщением представлена в виде таблицы 4.

	1	2	3	4	5	6	7	
Сообщение	0	1	1	1	1	1	0	
2^x	r1	r2	i1	r3	i2	i3	i4	s
1	+		+		+		+	s_1
2		+	+			+	+	s_2
4				+	+	+	+	s_3

Таблица 4 - «Таблица кодов Хэмминга (7;4)»

Вычислим синдром S:

$$s_1 = r_1 \oplus i_1 \oplus i_2 \oplus i_4 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$s_2 = r_2 \oplus i_1 \oplus i_3 \oplus i_4 = 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$s_3 = r_3 \oplus i_2 \oplus i_3 \oplus i_4 = 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$S = \overline{s_1 s_2 s_3}_{(2)} = 011_{(2)}$$

Синдрому S соответствует столбец 6, так как отметки стоят только у s_2 и s_3. Значит, ошибка в символе i_3. Изменим его значение с 1 на 0, чтобы исправить ошибку. Получим исправленное сообщение: 0111100.

Ответ: ошибка в символе i_3, исправленное сообщение: 0111100.

Сообщение 3

Таблица кодов Хэмминга (7;4) с рассматриваемым сообщением представлена в виде таблицы 5.

	1	2	3	4	5	6	7	
Сообщение	1	0	1	0	0	1	1	
2^x	r1	r2	i1	r3	i2	i3	i4	s
1	+		+		+		+	s_1
2		+	+			+	+	s_2
4				+	+	+	+	s_3

Таблица 5 - «Таблица кодов Хэмминга (7;4)»

Вычислим синдром S:

$$s_1 = r_1 \oplus i_1 \oplus i_2 \oplus i_4 = 1 \oplus 1 \oplus 0 \oplus 1 = 1$$

$$s_2 = r_2 \oplus i_1 \oplus i_3 \oplus i_4 = 0 \oplus 1 \oplus 1 \oplus 1 = 1$$

$$s_3 = r_3 \oplus i_2 \oplus i_3 \oplus i_4 = 0 \oplus 0 \oplus 1 \oplus 1 = 0$$

$$S = \overline{s_1 s_2 s_3}_{(2)} = 110_{(2)}$$

Синдрому S соответствует столбец 3, так как отметки стоят только у s_1 и s_2. Значит, ошибка в символе i_1. Изменим его значение с 1 на 0, чтобы исправить ошибку. Получим исправленное сообщение: 1000011.

Ответ: ошибка в символе i_1, исправленное сообщение: 1000011.

Задание 3

Схема декодирования классического кода Хэмминга (15;11) представлена на рисунке 2.

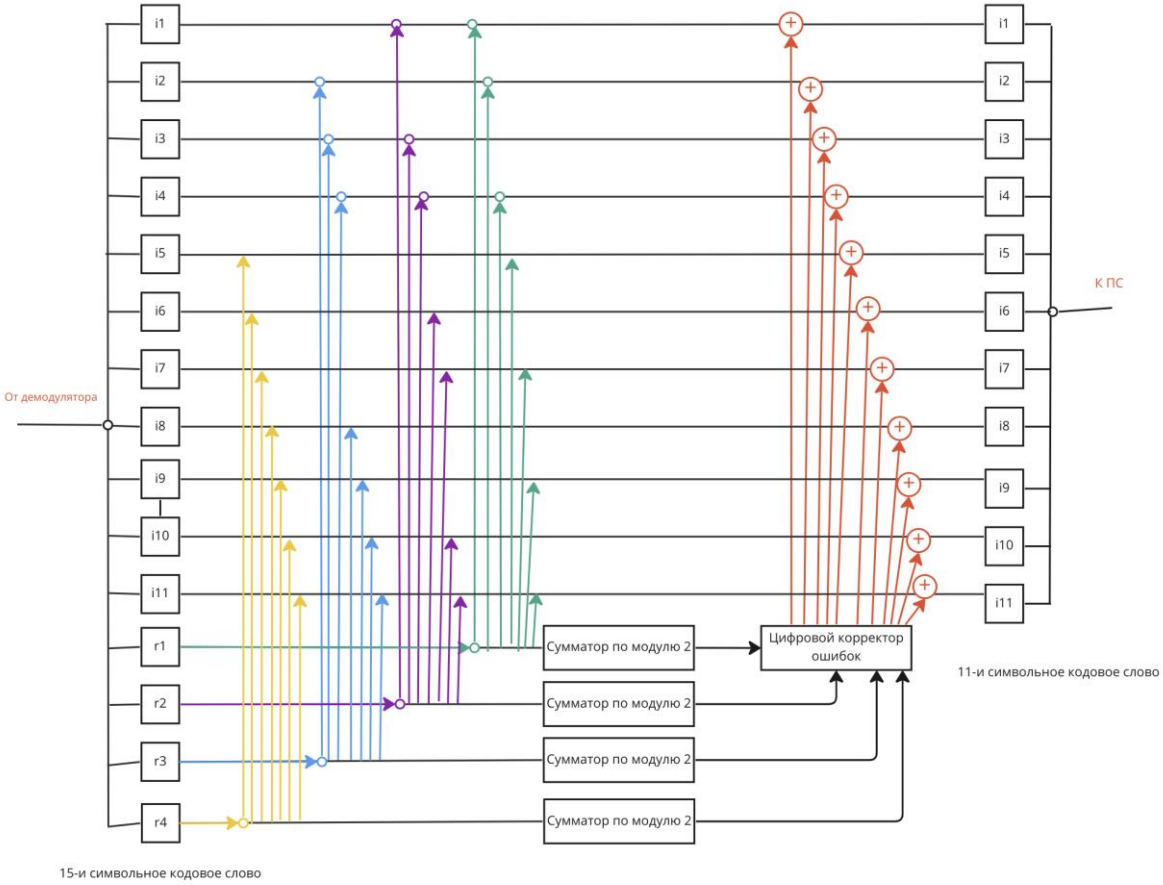

Рисунок 2 – «Схема декодирования кода Хэмминга (15;11)»

Задание 4

Построим таблицу кодов Хэмминга (15;11) . Затем вычислим синдром S из s_1, s_2, s_3, s_4, сложив отмеченные напротив биты в таблице по модулю 2. Если S равен 0, то ошибки нет, иначе найдём бит с ошибкой, сопоставив двоичной число, состоящее из синдромов, с отметками в таблице.

Таблица кодов Хэмминга (15;11) с рассматриваемым сообщением представлена в виде таблицы 6.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Сообщение	0	0	1	1	1	0	0	0	1	1	1	0	1	0	0	
2^x	r1	r2	i1	r3	i2	i3	i4	r4	i5	i6	i7	i8	i9	i10	i11	s
1	+		+		+		+		+		+		+		+	s_1
2		+	+			+	+			+	+			+	+	s_2
4				+	+	+	+					+	+	+	+	s_3
8								+	+	+	+	+	+	+	+	s_4

Таблица 6 - «Таблица кодов Хэмминга (15;11)»

Вычислим синдром S:

$$s_1 = r_1 \oplus i_1 \oplus i_2 \oplus i_4 \oplus i_5 \oplus i_7 \oplus i_9 \oplus i_{11} = 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$s_2 = r_2 \oplus i_1 \oplus i_3 \oplus i_4 \oplus i_6 \oplus i_7 \oplus i_{10} \oplus i_{11} = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 1$$

$$s_3 = r_3 \oplus i_2 \oplus i_3 \oplus i_4 \oplus i_8 \oplus i_9 \oplus i_{10} \oplus i_{11} = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 = 1$$

$$s_4 = r_4 \oplus i_5 \oplus i_6 \oplus i_7 \oplus i_8 \oplus i_9 \oplus i_{10} \oplus i_{11} = 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 = 0$$

$$S = \overline{s_1 s_2 s_3 s_4}_{(2)} = 1110_{(2)}$$

Синдрому S соответствует столбец 7, так как отметки стоят только у s_1, s_2 и s_3. Значит, ошибка в символе i_4. Изменим его значение с 0 на 1, чтобы исправить ошибку. Получим исправленное сообщение: 001110101110100.

Ответ: ошибка в символе i_4, исправленное сообщение: 001110101110100.

Задание 5

Вычислим число, необходимое для выполнения задания.

$4(54 + 91 + 16 + 51 + 71) = 4 \times 283 = 1132$

Получаем, что передаваемое сообщение состояло из 1132 информационных разрядов.

Определим минимальное количество контрольных разрядов для такого сообщения по формуле:

$2^r \geq r + i + 1$, где r – количество контрольных разрядов, $r \in \mathbb{N}$;

i – количество информационных разрядов, $i \in \mathbb{N}$.

$2^r \geq r + 1132 + 1$

$2^r \geq r + 1133$

При r = 10: $1024 \geq 1143$ – неверно

При r = 11: $2048 \geq 1144$ – верно

Следовательно, $r \geq 11$, то есть для сообщения, состоящего из 1132 информационных символов нужно как минимум 11 проверочных разрядов.

Теперь вычислим коэффициент избыточности k как отношение числа проверочных разрядов r (при r = 11) к общему числу разрядов, равному r + i.

$$k = \frac{r}{r+i} = \frac{11}{11+1132} = \frac{11}{1143} = 0{,}00962379 \ldots \approx 0{,}01$$

Ответ: $r_{min} = 11,\ k = 0{,}01$

Задание 6*

Для выполнения этого задания была написана программа на языке Java. Далее представлен её код.

HammingAnalyser/Main.java:

```
package HammingAnalyser;

import java.util.Scanner;

final class Main {
    private final static Scanner inp = new Scanner(System.in);

    public static void main(String[] args) throws Exception {
        boolean messageCreated = false;
        Message msg = null;
        do {
            System.out.print("Введите сообщение: ");
            String msgStr = inp.next();
            try {
                msg = new Message(msgStr);
                messageCreated = true;
            } catch (Exception e) {
                System.out.println(e);
            }
        } while (!messageCreated);
        printErrorAndFix(msg);
    }

    private static void printErrorAndFix(Message m) {
        String errorBit = m.getErrorBit();
        if (errorBit != null) {
            System.out.printf("Ошибка в бите: %s\n", errorBit);
            Message fixed;
            try {
                fixed = m.getFixedMessage();
            } catch (Exception e) {
                System.out.println("Возникла проблема при исправлении ошибки");
                return;
            }
            System.out.printf("Исправленное                сообщение:                %s\n",
fixed.getCurrentMessageString());
            try {
                assert (fixed.getFixedMessage() == null);
            } catch (Exception e) {
                System.out.println(e);
                return;
            }
        }
        else
            System.out.printf("Ошибок нет");
    }
}

class Message {
    final static int MAX_CHECKING_BITS = 20;
    protected String msg;
    protected String S;

    public Message(String message) throws Exception {
        checkMessage(message);
        this.msg = message;
        calculateS();
    }
```

```
    public String getErrorBit() {
        int bitIndex = binToDec(reverse(this.S));
        int log2BitIndex = flooredLog2(bitIndex);
        if (bitIndex == 0)  // No error
            return null;
        else if ((1 << log2BitIndex) == bitIndex)  // Checking bits stand on the indexes those
are powers of 2
            return "r" + log2BitIndex;
        else  // Otherwise, it is a info bit
            return "i" + (bitIndex - log2BitIndex - 1);
    }

    public Message getFixedMessage() throws Exception {
        int bitIndex = binToDec(reverse(this.S)) - 1;
        String newMsg = this.msg.substring(0, bitIndex) + (this.msg.charAt(bitIndex) == '1' ?
'0' : '1') + this.msg.substring(bitIndex + 1);
        return new Message(newMsg);
    }

    public String getCurrentMessageString() {
        return this.msg;
    }

    private void checkMessage(String msg) throws Exception {
        if (countChars(msg, "1") + countChars(msg, "0") != msg.length())
            throw new Exception();
    }

    private void calculateS() {
        int n = msg.length();
        int r = getNumberOfCheckingBits(n);
        String S = "";
        for (int i = 1; i <= r; i++)
        S += calculateSyndrome(i);
        this.S = S;
    }

    private int calculateSyndrome(int idx) {
        int s = 0;
        for (int i = 1 << (idx - 1); i <= msg.length(); i += 2*idx) {
            for (int j = 0; j < 1 << (idx - 1); j++) {
                if (i + j - 1 >= msg.length())
                    break;
                s = (s + Character.getNumericValue(this.msg.charAt(i + j - 1))) % 2;
            }
        }
        return s;
    }

    private static int getNumberOfCheckingBits(int totalBits) {
        // 2^r >= r + i + 1, searching for the first r which fits this inequality
        int twoPowR = 1;
        for (int r = 1; r <= MAX_CHECKING_BITS; r++) {
            twoPowR *= 2;
            if (twoPowR >= totalBits + 1)
                return r;
        }
        throw new ArithmeticException();
    }

    private static int countChars(String s, String chars) {
        return s.length() - s.replace(chars, "").length();
    }

    private static int flooredLog2(int num) {
        int val = 1;
        for (int power = 0; val < Integer.MAX_VALUE; power++) {
            if (val == num || val * 2 > num)
                return power;
```

```
            val *= 2;
        }
        return -1;
    }

    private static int binToDec(String bin) {
        int dec = 0;
        for (int i = 0; i < bin.length(); i++)
            if (bin.charAt(i) == '1')
                dec += 1 << (bin.length() - 1 - i);
        return dec;
    }

    private static String reverse(String s) {
        String res = "";
        for (int i = s.length() - 1; i >= 0; i--)
            res += s.charAt(i);
        return res;
    }
}
```

Примеры вывода программы (в качестве переходных данных передадим сообщения из задания 2) показаны на рисунке 3.

```
[amphyx@bringer prog-task]$ java HammingAnalyser/Main.java
Введите сообщение: 1101011
Ошибка в бите: i3
Исправленное сообщение: 1101001
[amphyx@bringer prog-task]$ java HammingAnalyser/Main.java
Введите сообщение: 0111110
Ошибка в бите: i3
Исправленное сообщение: 0111100
[amphyx@bringer prog-task]$ java HammingAnalyser/Main.java
Введите сообщение: 1010011
Ошибка в бите: i1
Исправленное сообщение: 1000011
```

Рисунок 3 – «Вывод программы»

Заключение

В результате выполнения данной работы были получены знания о коде Хэмминга и его применении для проверки ошибок в сообщениях, возникших при передаче или хранении данных. Далее изучен алгоритм построения таблицы кода Хэмминга и метод вычисления синдрома последовательности. Затем рассмотрена схема декодирования кода Хэмминга (для случаев (7;4), (15;11)) и выполнены практические задания по поиску ошибки в некоторых сообщениях. Также были рассмотрены характеристики кода Хэмминга, такие как коэффициент избыточности, расстояние Хэмминга, кодовое расстояние, и произведено их вычисление для конкретного примера.

Список использованной литературы

Всё о Hi-Tech Коды Хэмминга [Электронный ресурс]. - Дата обращения: 14 Октябрь 2022 г. - URL: http://all-ht.ru/inf/systems/p_0_14.html.

Е.А. Балакшин П.В. Соснин В.В. Машина Информатика [Книга]. - Санкт-Петербург : Университет ИТМО, 2020.