Федеральное государственное автономное образовательное учреждение высшего образования “Национальный исследовательский университет ИТМО”

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по дисциплине “Информатика”

Синтез помехоустойчивого кода

Вариант 50

Выполнил:

Шулай Роман Юрьевич P3115

Проверил:

Миняев Илья Андреевич

Санкт-Петербург 2025

Оглавление

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 отчете в виде изображения.
3. Показать, исходя из выбранных вариантов сообщений (по 4 у каждого – часть №1 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 15-символьного кода.
5. Построить схему декодирования классического кода Хэмминга (15;11), которую представить в отчете 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.

Основные этапы вычисления

1*

Для варианта 50 набор из 4 полученных сообщений следующий: 35, 67, 99, 19

2*

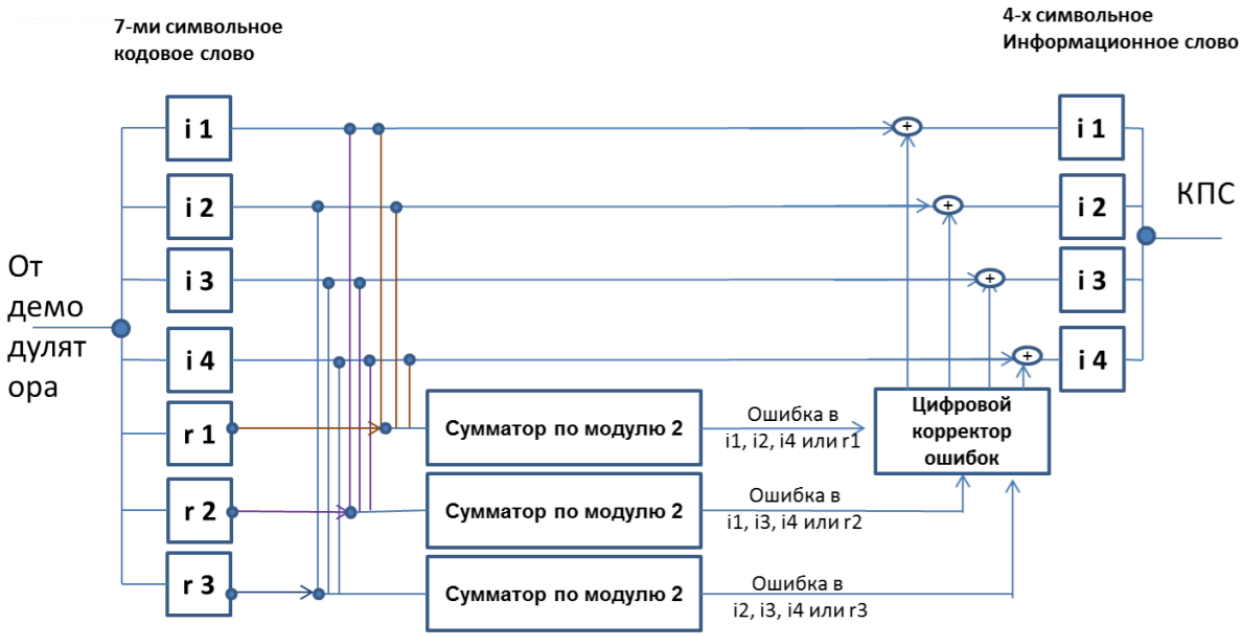

Рисунок 1 Схема декодирования классического кода Хэмминга (7,4)

3*

а) Построим таблицу для сообщения 35:

N	1	2	3	4	5	6	7
Сообщение	0	1	1	1	0	1	0
2^x	r1	r2	i1	r3	i2	i3	i4
1	x		x		x		x
2		x	x			x	x
4				x	x	x	x

Проверочный бит r1 должен быть равен i1⊕i2⊕i4=1, но равен 0

Проверочный бит r2 должен быть равен i1⊕i3⊕i4=0, но равен 1

Проверочный бит r3 должен быть равен i2⊕i3⊕i4=1 и равен 1

Номера битов r1, r2 - 1 и 2 соответственно, значит ошибка в 1+2=3 бите по счету, а это i1

Верное сообщение: 0101010

б) Построим таблицу для сообщения 67:

N	1	2	3	4	5	6	7
Сообщение	1	1	0	0	1	0	0
2^x	r1	r2	i1	r3	i2	i3	i4
1	x		x		x		x
2		x	x			x	x
4				x	x	x	x

Проверочный бит r1 должен быть равен i1⊕i2⊕i4=1 и равен 1

Проверочный бит r2 должен быть равен i1⊕i3⊕i4=0, но равен 1

Проверочный бит r3 должен быть равен i2⊕i3⊕i4=1, но равен 0

Номер битов r2, r3 - 2 и 4 соответственно, значит ошибка в 2+4=6 бите по счету, а это i3

Верное сообщение: 1100110

в) Построим таблицу для сообщения 99:

N	1	2	3	4	5	6	7
Сообщение	0	0	0	0	1	1	1
2^x	r1	r2	i1	r3	i2	i3	i4
1	x		x		x		x
2		x	x			x	x
4				x	x	x	x

Проверочный бит r1 должен быть равен i1⊕i2⊕i4=0 и равен 0

Проверочный бит r2 должен быть равен i1⊕i3⊕i4=0 и равен 0

Проверочный бит r3 должен быть равен i2⊕i3⊕i4=1, но равен 0

Номер бита r3 - 4, значит ошибка в 4 бите - бите r3

Верное сообщение: 0001111

г) Построим таблицу для сообщения 19:

N	1	2	3	4	5	6	7
Сообщение	0	1	0	1	0	0	1
2^x	r1	r2	i1	r3	i2	i3	i4
1	x		x		x		x
2		x	x			x	x
4				x	x	x	x

Проверочный бит r1 должен быть равен i1⊕i2⊕i4=1, но равен 0

Проверочный бит r2 должен быть равен i1⊕i3⊕i4=1 и равен 1

Проверочный бит r3 должен быть равен i2⊕i3⊕i4=1 и равен 1

Номер бита r1 - 1, значит ошибка в 1 бите - бите r1

Верное сообщение: 1101001

4*

Для варианта 50 получено сообщение 50

5*

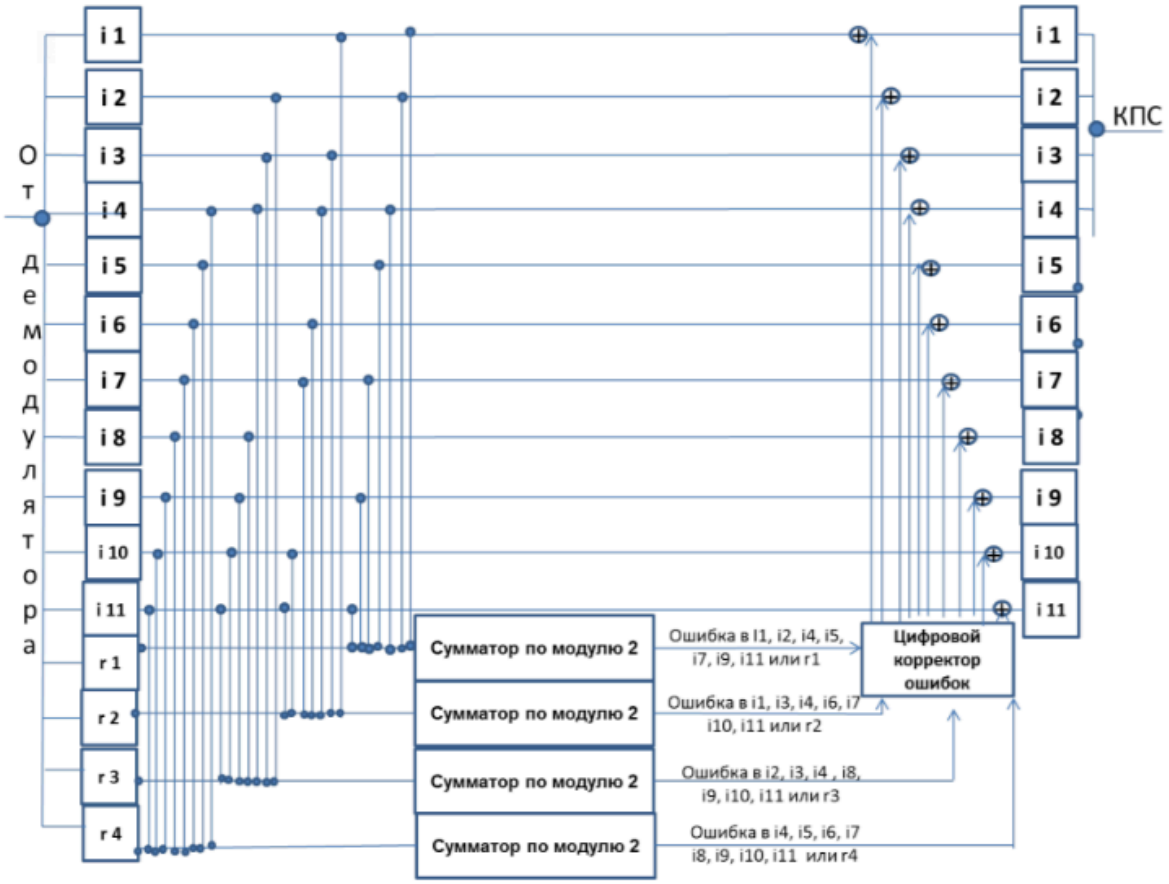

Рисунок 2 Схема декодирования классического кода Хэмминга (15,11)

6*

Построим таблицу для сообщения 50:

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

Проверочный бит r1 должен быть равен i1⊕i2⊕i4⊕i5⊕i7⊕i9⊕i11=0 и равен 0

Проверочный бит r2 должен быть равен i1⊕i3⊕i4⊕i6⊕i7⊕i10⊕i11=1 и равен 1

Проверочный бит r3 должен быть равен i2⊕i3⊕i4⊕i8⊕i9⊕i10⊕i11=0 и равен 0

Проверочный бит r4 должен быть равен i5⊕i6⊕i7⊕i8⊕i9⊕i10⊕i11=1, но равен 0

Номер бита r4 - 8, значит ошибка в 8 бите - бите r4

Верное сообщение: 01000110100011

7*

i = (35+67+99+19+50)*4 = 1080 - информационных битов в передаваемом сообщении.

Пусть количество проверочных разрядов - r. Тогда верно неравенство:

$2^r \geq r + i + 1 \Leftrightarrow 2^r \geq r + 1081$, отсюда наименьшее целое r = 11.

Минимальное число проверочных разрядов: 11

Коэффициент избыточности: $\frac{r}{i+r} = \frac{11}{1091} \approx 0,0100825$

Дополнительное задание №1

Сделать себе учётную запись на https://gitlab.se.ifmo.ru/. Создал учетную запись https://gitlab.se.ifmo.ru/dxunvrs.

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.

Напишем программу на языке программирования Python

```python
import sys

def main():
    while True:
        try:
            message = input("Введите сообщение(двоичное число длины 7, для выхода quit): ")
            if message == "quit":
                sys.exit(0)
            if not isMessage(message):
                print("Ошибка ввода")
                continue
            print(findError(message))
        except KeyboardInterrupt:
            print("Ошибка ввода")
        except EOFError:
            print("Ошибка ввода")


def isMessage(message):
    symbols = set(message)
    if len(symbols) == 0 or len(symbols) > 2:
        return False
    if not(symbols == set("1") or symbols == set("0") or symbols == set("01")):
        return False
    if len(message) != 7:
        return False
    return True


def findError(message):
    infBits = []
    checkBits = []
    correctMessage = [int(x) for x in list(message)]
    errorBit = 0

    # находим проверочные биты и информационные
    for i in range(0,len(message)):
        if (i+1) in [1,2,4]:
            checkBits += [int(message[i])]
        else:
            infBits += [int(message[i])]

    # считаем сумму для проверочных битов
    r1 = (infBits[0] + infBits[1] + infBits[3]) % 2
    r2 = (infBits[0] + infBits[2] + infBits[3]) % 2
    r3 = (infBits[1] + infBits[2] + infBits[3]) % 2
    correctcheckBits = [r1, r2, r3]

    # проверяем все ли они совпадают и накапливаем номер в errorBit
    for i in range(0,len(checkBits)):
        if checkBits[i] != correctcheckBits[i]:
            errorBit += 2**i
```

Рисунок 3 Листинг программы на языке Python

```python
    # выводим все что нашли
    if errorBit == 0:
        return "Ошибок нет, введено верное сообщение: " + "".join([str(x) for x in infBits]) + f"(полное сообщение: {message})"

    correctMessage[errorBit-1] = (correctMessage[errorBit-1] + 1) % 2
    correctMessage = "".join([str(x) for x in correctMessage])

    if errorBit in [1,2,4]:
        return "Ошибка в проверочном бите, информация не пострадала: " + "".join([str(x) for x in infBits]) + f"(полное сообщение: {correctMessage})"
    else:
        infBitsMessage = f"{correctMessage[2]}{correctMessage[4]}{correctMessage[5]}{correctMessage[6]}"
        if errorBit == 3:
            return f"Ошибка в {errorBit-2} информационном бите, верное сообщение: {infBitsMessage}" + f"(полное сообщение: {correctMessage})"
        return f"Ошибка в {errorBit-3} информационном бите, верное сообщение: {infBitsMessage}" + f"(полное сообщение: {correctMessage})"


if __name__ == "__main__":
    main()
```

Рисунок 4 Листинг программы на языке Python

Функция isMessage проверяет корректность ввода, основная логика реализована в функции findError. Сначала программа разделяет сообщение на два массива с информационными битами и проверочными, затем считает суммы информационных битов и сравнивает с проверочными, если значения совпали, то в этих битах ошибок нет, если нет то программа запоминает номер бита с ошибкой. После цикла в переменной errorBit содержится номер ошибочного бита. После этого программа выводит место с ошибкой и верное сообщение.

Вывод

В ходе выполнения лабораторной работы я научился работать с кодами Хэмминга (7;4) и (15;11), а также определять биты информации, переданные ошибочно, и, таким образом, исправлять поврежденные в процессе передачи сообщения.

Список литературы

1. Основы цифровой радиосвязи. Помехоустойчивое кодирование: метод. указания / сост. Д. В. Пьянзин. – Саранск : Изд-во Мордов. ун-та, 2009. – 16 с.
2. Коды и устройства помехоустойчивого кодирования информации / сост. Королев А.И. – Мн.: , 2002. – с.286