МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования
«Крымский федеральный университет имени В.И. Вернадского»

Физико-технический институт

Кафедра компьютерной инженерии и моделирования

Лабораторная работа № 6
«Циклический код»
по дисциплине
«Теория информации и кодирование»

Выполнил:
студент 3 курса
группа ИВТ-222
Гоголев В. Г

Проверил:
Филиппов Д.М.
«____» _____20___г.
Подпись: _____

Симферополь, 2024

Цель работы: построить помехоустойчивый циклический код, позволяющий обнаруживать и исправлять все однократные ошибки.

Техническое задание: источник информации вырабатывает сообщения, содержащие k информационных разрядов. Значения разрядов генерируются в двоичной системе счисления счетчиком случайных чисел. Необходимо: 1. разработать программное обеспечение для передатчика, которое будет строить циклический код, позволяющий обнаруживать и исправлять все однократные ошибки; 2. разработать программное обеспечение на приемной стороне, позволяющее обрабатывать принятый циклический код и определять позицию ошибки; 3. провести комплекс численных экспериментов, в ходе которых продемонстрировать работу системы «передатчик-приемник» с использованием циклического кода

Ход работы:

Вариант № 4

Задание I.

С использованием разработанного программного обеспечения для передатчика необходимо по количеству информационных разрядов (k) определить значность кода (n), рассчитать количество проверочных разрядов (p), выбрать образующий полином P(x).

Задание II.

Провести цикл комплексных экспериментов (не менее 6), в ходе которого необходимо: а) сгенерировать случайным образом информационную кодовую комбинацию, состоящую из k разрядов, на передающей стороне; б) построить для информационной кодовой комбинации на передающей стороне циклический код, позволяющий обнаруживать и исправлять все однократные ошибки; в) передать образующий полином с выходы программного обеспечения на передающей стороне на вход программного обеспечения приемной стороны; г) передать циклический код от передатчика к приемнику, сгенерировав случайным образом однократную ошибку в любом разряде циклического кода; д) при помощи программного обеспечения на приемной стороне построить таблицу соответствия позиции ошибки и вида остатка; е) по принятому циклическому коду на приемной стороне определить полином, рассчитать остаток от деления полинома на образующий полином, по таблице соответствия определить позицию ошибки и откорректировать циклический код.

```
Образующий полином: [1 1 0 1]

Эксперимент 1
Исходное сообщение: [1 1 0 1 0 1 1 1 1 1 1 1]
Закодированное сообщение: [1 1 0 1 0 1 1 1 1 1 1 1 0 0 0]
Сообщение с ошибкой: [1 1 0 1 0 1 1 1 0 1 1 1 0 0 0]
Ошибка введена в позиции 8
Обнаруженная позиция ошибки: 1
Синдром: [1 1 0]
Исправленное сообщение: [1 0 0 1 0 1 1 1 0 1 1 1 0 0 0]
Сообщение без проверочных разрядов: [1 0 0 1 0 1 1 1 0 1 1 1]
Исходное сообщение совпадает с исправленным: False

Эксперимент 2
Исходное сообщение: [1 1 1 1 1 1 0 0 1 1 1 1]
Закодированное сообщение: [1 1 1 1 1 1 0 0 1 1 1 1 1 0 1]
Сообщение с ошибкой: [1 1 1 1 0 1 0 0 1 1 1 1 1 0 1]
Ошибка введена в позиции 4
Обнаруженная позиция ошибки: 4
Синдром: [1 0 1]
Исправленное сообщение: [1 1 1 1 1 1 0 0 1 1 1 1 1 0 1]
Сообщение без проверочных разрядов: [1 1 1 1 1 1 0 0 1 1 1 1]
Исходное сообщение совпадает с исправленным: True
```

Рисунок 1 – результат работы программы

```
Эксперимент 3
Исходное сообщение: [1 1 1 1 1 1 0 0 1 1 0 1]
Закодированное сообщение: [1 1 1 1 1 1 0 0 1 1 0 1 0 1 0]
Сообщение с ошибкой: [1 1 1 1 1 1 0 0 0 1 0 1 0 1 0]
Ошибка введена в позиции 8
Обнаруженная позиция ошибки: 1
Синдром: [1 1 0]
Исправленное сообщение: [1 0 1 1 1 1 0 0 0 1 0 1 0 1 0]
Сообщение без проверочных разрядов: [1 0 1 1 1 1 0 0 0 1 0 1]
Исходное сообщение совпадает с исправленным: False

Эксперимент 4
Исходное сообщение: [0 1 0 0 0 0 1 1 1 0 0 0]
Закодированное сообщение: [0 1 0 0 0 0 1 1 1 0 0 0 0 1 1]
Сообщение с ошибкой: [0 1 0 0 0 0 1 1 1 0 0 1 0 1 1]
Ошибка введена в позиции 11
Обнаруженная позиция ошибки: 4
Синдром: [1 0 1]
Исправленное сообщение: [0 1 0 0 1 0 1 1 1 0 0 1 0 1 1]
Сообщение без проверочных разрядов: [0 1 0 0 1 0 1 1 1 0 0 1]
Исходное сообщение совпадает с исправленным: False
```

Рисунок 2 – результат работы программы

ЗАКЛЮЧЕНИЕ

В результате выполнения работы были получены навыки по формированию Систематического кода, по созданию образующего полинома, в процессе передачи сообщения, генерации кодовой комбинации на передающей стороне, был построен систематический код на передающей стороне для кодовой комбинации, способный обрабатывать однократные ошибки, на принимающей стороне реализовано получение кодовой комбинации, полнима и преобразования для выявления позиции ошибки. Были проведены 6 экспериментов с разными значениями кодовой комбинации для проверки работы алгоритма.

ПРИЛОЖЕНИЕ

```
import numpy as np
import random

#Функция generate_random_message(k) генерирует случайное бинарное сообщение длиной
k.
def generate_random_message(k):
    return np.random.randint(0, 2, k)

#Функция polynomial_to_bits(p) преобразует полином в массив битов.
def polynomial_to_bits(p):
    return np.array([int(bit) for bit in bin(p)[2:]])

#Функция divide_polynomials(dividend, divisor) выполняет деление полиномов,
возвращая остаток.
def divide_polynomials(dividend, divisor):
    while len(dividend) >= len(divisor):
        if dividend[0] == 1:
            dividend[:len(divisor)] ^= divisor
        dividend = dividend[1:]
    return dividend

#Функция encode_message(message, generator_polynomial) кодирует сообщение,
добавляя проверочные биты.
def encode_message(message, generator_polynomial):
    padded_message = np.concatenate((message, np.zeros(len(generator_polynomial) -
1, dtype=int)))
    remainder = divide_polynomials(padded_message, generator_polynomial)
    return np.concatenate((message, remainder))

def introduce_error(encoded_message):
    error_position = random.randint(0, len(encoded_message) - 1)
    encoded_message[error_position] ^= 1
    return error_position, encoded_message

def detect_error(encoded_message, generator_polynomial):
    syndrome = divide_polynomials(encoded_message.copy(), generator_polynomial)
    if not any(syndrome):
        return -1, syndrome
    for i in range(len(encoded_message)):
        test_message = encoded_message.copy()
        test_message[i] ^= 1
        if not any(divide_polynomials(test_message.copy(), generator_polynomial)):
            return i, syndrome
    return -1, syndrome

def main():
    k = 12
    generator_polynomial_bits = polynomial_to_bits(0b1101)  # Пример полинома x^3
+ x^2 + 1
```

```
    print("Образующий полином:", generator_polynomial_bits)

    for experiment in range(6):
        print(f"\nЭксперимент {experiment + 1}")
        message = generate_random_message(k)
        print("Исходное сообщение:", message)

        encoded_message = encode_message(message, generator_polynomial_bits)
        print("Закодированное сообщение:", encoded_message)

        error_position, erroneous_message =
introduce_error(encoded_message.copy())
        print("Сообщение с ошибкой:", erroneous_message)
        print(f"Ошибка введена в позиции {error_position}")

        detected_error_position, syndrome = detect_error(erroneous_message.copy(),
generator_polynomial_bits)
        print("Обнаруженная позиция ошибки:", detected_error_position)
        print("Синдром:", syndrome)

        if detected_error_position != -1:
            erroneous_message[detected_error_position] ^= 1
        print("Исправленное сообщение:", erroneous_message)
        print("Сообщение без проверочных разрядов:", erroneous_message[:-
len(generator_polynomial_bits) + 1])
        print("Исходное сообщение совпадает с исправленным:",
              np.array_equal(message, erroneous_message[:-
len(generator_polynomial_bits) + 1]))

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

Приложение 1 – листинг программного кода