Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра радиотехнических систем (РТС)

Кологривов В. А., Куулар Ч. М.

КВАЗИСИНДРОМНОЕ ДЕКОДИРОВАНИЕ II ТИПА АЛГЕБРАИЧЕСКИХ БЛОКОВЫХ КОДОВ

Учебно-методическое пособие по лабораторной и самостоятельной работе и практическим занятиям

2019

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра радиотехнических систем (РТС)

Утверждаю:

Зав. кафедрой РТС, проф., д.т.н.

_____ С.В. Мелихов

«_____»_____2019 г.

КВАЗИСИНДРОМНОЕ ДЕКОДИРОВАНИЕ II ТИПА АЛГЕБРАИЧЕСКИХ БЛОКОВЫХ КОДОВ

Учебно-методическое пособие по лабораторной и самостоятельной работе и практическим занятиям для студентов направления

«Инфокоммуникационные технологии и системы связи»

Разработчики:

Доц. каф. РТС Кологривов В. А. _____

Студентка гр. 1В5 Куулар Ч. М. _____

2019

Кологривов В. А., Куулар Ч. М.

«Квазисиндромное декодирование II типа алгебраических блоковых кодов»: учебно-методическое пособие по лабораторной и самостоятельной работе и практическим занятиям для студентов направления «Инфокоммуникационные технологии и системы связи» - Томск: ТУСУР. Образовательный портал, 2019.– 25 с.

Учебно-методическое пособие содержит описание процесса квазисиндромного декодирования схемы, разработанной в среде Simulink пакета прикладных программ MatLab.

В данном пособии приведены краткие теоретические сведения по кодированию и квазисиндромному декодированию алгебраических блоковых кодов, краткая характеристика функциональных блоков библиотеки среды Simulink пакета прикладных MatLab, описание функциональной схемы, а также требования к экспериментальному исследованию и контрольные вопросы для допуска к выполнению лабораторной работы.

Аннотация

Лабораторная работа **«Квазисиндромное декодирование II типа алгебраических блоковых кодов»** разработана для экспериментального исследования модели кодека, исправляющего одиночные ошибки, в среде Simulink пакета прикладных программ MatLab.

Работа «Квазисиндромное декодирование II типа алгебраических блоковых кодов» относится к циклу лабораторных работ по разделу «помехоустойчивое кодирование», входящему в дисциплины по направлению «Инфокоммуникационные технологии и системы связи».

В данном пособии сформулирована цель лабораторной работы, приведены описание алгоритма, краткая характеристика среды Simulink пакета прикладных программ MatLab, описание виртуального лабораторного макета и используемых блоков библиотеки Simulink, а также требования к экспериментальному исследованию и контрольные вопросы для допуска к выполнению лабораторной работы.

Оглавление

1 Цель работы, краткие сведения

Цель работы: изучение принципов блокового кодирования и квазисиндромного декодирования II типа алгебраических блоковых кодов на основе модели кодека, разработанной в среде Simulink пакета прикладных программ MatLab.

Теоретические сведения

Идея помехоустойчивого кодирования состоит в таком добавлении к информационным битам избыточных битов, чтобы в приемнике ошибки могли быть найдены и исправлены. В данной работе рассмотрены алгебраические блоковые коды. Различают два вида кодов: блоковые и непрерывные. Блоковые коды в отличие от непрерывных кодов делят информацию на фрагменты постоянной длины, и каждый из них обрабатывается в отдельности. Код, в котором информационные символы заданы в явном виде, называется систематическим.

Помехоустойчивый код задает k информационных битов n битовыми кодами, называемыми кодовыми символами, за счет добавления $r = n - k$ битов четности к информационным битам [1].

Под символом будем понимать последовательность битов. Различают два вида символов: информационные и контрольные. Кодовый символ будет состоять из информационных и контрольных символов.

Далее для примера будет рассматриваться систематический код Хэмминга (7,4). Составим порождающую систему уравнений систематического блокового кода (7,4).

$$1 \cdot x_k \oplus 0 \cdot x_{k+1} \oplus 0 \cdot x_{k+2} \oplus 0 \cdot x_{k+3} = y_k,$$

$$0 \cdot x_k \oplus 1 \cdot x_{k+1} \oplus 0 \cdot x_{k+2} \oplus 0 \cdot x_{k+3} = y_{k+1},$$

$$0 \cdot x_k \oplus 0 \cdot x_{k+1} \oplus 1 \cdot x_{k+2} \oplus 0 \cdot x_{k+3} = y_{k+2},$$

$$0 \cdot x_k \oplus 0 \cdot x_{k+1} \oplus 0 \cdot x_{k+2} \oplus 1 \cdot x_{k+3} = y_{k+3},$$

$$0 \cdot x_k \oplus 1 \cdot x_{k+1} \oplus 1 \cdot x_{k+2} \oplus 1 \cdot x_{k+3} = y_{k+4},$$

$$1 \cdot x_k \oplus 1 \cdot x_{k+1} \oplus 0 \cdot x_{k+2} \oplus 1 \cdot x_{k+3} = y_{k+5},$$

$$1 \cdot x_k \oplus 1 \cdot x_{k+1} \oplus 1 \cdot x_{k+2} \oplus 0 \cdot x_{k+3} = y_{k+6}.$$

В матричном виде коэффициенты данной системы уравнений представляют собой транспонированную порождающую матрицу:

$$G^t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}.$$

Структура порождающей матрицы систематического кода имеет вид [1]:

$$G = \left[I_{(k,k)} P_{(k,r)} \right],$$

где $I_{(k,k)}$ - единичная подматрица размерностью (k, k);

$P_{(k,r)}$– подматрица, состоящая из контрольных символов, размерностью (k, r).

Тогда порождающая матрица выглядит так:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

Так как главной идеей помехоустойчивого кодирования является введение избыточных символов, кодовый символ формируется следующим образом [2]:

$$Y = X \cdot G,$$

где G – порождающая матрица;

X – входная информационная последовательность битов.

Формирование кодового символа на выходе кодера в матричном виде:

$$Y = [x_k, x_{k+1}, x_{k+2}, x_{k+3}] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} =$$

$$[y_k, y_{k+1}, y_{k+2}, y_{k+3}, y_{k+4}, y_{k+5}, y_{k+6}].$$

Для примера закодируем некоторую часть входного потока с помощью нашей порождающей матрицы:

$$Y = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Таким образом, в результате кодирования из четырех битовых информационных векторов получается семибитовые кодовые векторы, что показывает избыточность кода.

При распространении сигнала в канале связи кодовый символ искажается. Рассмотрим процесс декодирования искаженного сигнала.

Наиболее распространенным методом декодирования является метод синдромного декодирования. Идея такого метода состоит в том, что формируется такая матрица, называемая проверочной *H*, с помощью которой находится синдром, указывающий на место возникновения ошибки. Такой подход позволяет обнаруживать и исправлять в данном случае одиночные ошибки [1].

На основе идеи синдромного декодирования был предложен метод квазисиндромного декодирования, суть которого состоит в том, что находятся вектора, обладающие свойствами синдромов, то есть зависящие только от места возникновения ошибок. Такие вектора определяются с помощью восстанавливающей матрицы, которая, в свою очередь, формируется путем составления уравнений восстановления

информационных битов из кодовых при условии отсутствия ошибок. Полученные таким образом вектора называются квазисиндромами (назовем их условно квазисиндромами I типа) [2].

В данной работе рассмотрен квазисиндромный метод (квазисиндромы II типа), основанный на составлении уравнений восстановления *контрольных* битов из кодовых при условии отсутствия ошибок. В дальнейшем такой метод будем называть методом квазисиндромного декодирования II типа.

Возникшие в процессе передачи одиночные ошибки в кодовом символе можно исправить, используя восстанавливающую матрицу. Уравнения восстанавливающей системы составляются путем суммирования по модулю 2 битов кодового символа. Количество таких уравнений соответствует числу позиций контрольных битов *(n-k)* в кодовом символе. У кода Хэмминга (7,4) контрольные биты занимают 3 позиций.

Составим уравнения восстановления контрольных битов на основе рассмотренного примера.

$$x_{k+4} = 0 \cdot y_k \oplus 1 \cdot y_{k+1} \oplus 0 \cdot y_{k+2} \oplus 0 \cdot y_{k+3} \oplus + 0 \cdot y_{k+4} \oplus 1 \cdot y_{k+5} \oplus 1 \cdot y_{k+6},$$

$$x_{k+5} = 0 \cdot y_k \oplus 0 \cdot y_{k+1} \oplus 1 \cdot y_{k+2} \oplus 1 \cdot y_{k+3} \oplus + 0 \cdot y_{k+4} \oplus 0 \cdot y_{k+5} \oplus 1 \cdot y_{k+6},$$

$$x_{k+6} = 1 \cdot y_k \oplus 0 \cdot y_{k+1} \oplus 0 \cdot y_{k+2} \oplus 1 \cdot y_{k+3} \oplus + 1 \cdot y_{k+4} \oplus 0 \cdot y_{k+5} \oplus 0 \cdot y_{k+6}.$$

Записывая коэффициенты этих уравнений по столбцам, получим восстанавливающую матрицу D.

$$D = \begin{vmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{vmatrix}.$$

Влияние помех на передаваемое сообщение в канале связи представляются в виде аддитивных векторов ошибок. Вектора ошибок представляют собой совокупность битов (нулей и единиц), причем единица

указывает на позицию некорректно принятого бита кодового символа. Вектора всевозможных одиночных ошибок формируют единичную матрицу одиночных ошибок размерностью (n, n).

Перемножим матрицу всевозможных одиночных ошибок E с матрицей восстановления D:

$$E \cdot D = \begin{bmatrix} 1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & 1 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & 0 & 1 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & 0 & 0 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \cdot \begin{vmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{vmatrix}.$$

Далее добавив матрицу W, содержащую несистематическую (проверочную) часть принятых кодовых символов, к полученной матрице, получим матрицу, строки которой представляют собой квазисиндромы II типа.

$$S = \begin{vmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{vmatrix} \oplus \begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{vmatrix}.$$

Таким образом, процесс нахождения квазисиндрома II типа можно представить следующим образом:

$$E \cdot D = D \oplus W = S.$$

Из двух последних выражений видно соответствие синдромов и векторов ошибок.

Искаженные биты можно исправить с помощью векторов ошибок найденных из таблицы соответствия:

$$Y = Z \oplus e_i.$$

Проверим правильность полученных квазисиндромов на примере, когда в принятых кодовых символах Z в последнем бите появилась одиночная ошибка.

$$S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & 1 & 1 & 1 & \mathbf{1} \\ 0 & 0 & 1 & 0 & 1 & 0 & \mathbf{0} \\ 0 & 0 & 1 & 1 & 0 & 1 & \mathbf{0} \\ 0 & 1 & 0 & 0 & 1 & 1 & \mathbf{0} \\ 0 & 1 & 0 & 1 & 0 & 0 & \mathbf{0} \\ 0 & 1 & 1 & 0 & 0 & 1 & \mathbf{1} \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \oplus \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Видно, что полученный квазисиндром совпадает с квазисиндромом вектора ошибки в последнем бите. Данный пример показывает правильность определения квазисиндромов. В таблице 1.1 представлены соответствия между векторами одиночных ошибок и квазисиндромами на основе данного примера.

Таблица 1.1 – Соответствие между векторами одиночных ошибок и квазисиндромами II типа

Квазисиндром	001	100	010	011	101	110	111
Вектор ошибки	e_1 (1000000)	e_2 (0100000)	e_3 (0010000)	e_4 (0001000)	e_5 (0000100)	e_6 (0000010)	e_7 (0000001)

Из данной таблицы видно, что различным векторам одиночных ошибок соответствуют различные квазисиндромы, это означает, что все одиночные ошибки могут быть обнаружены и исправлены.

Восстановим искаженные символы с ошибкой в последнем бите:

$$Y = Z \oplus e_7,$$

$$Y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & 1 & 1 & 1 & \mathbf{1} \\ 0 & 0 & 1 & 0 & 1 & 0 & \mathbf{0} \\ 0 & 0 & 1 & 1 & 0 & 1 & \mathbf{0} \\ 0 & 1 & 0 & 0 & 1 & 1 & \mathbf{0} \\ 0 & 1 & 0 & 1 & 0 & 0 & \mathbf{0} \\ 0 & 1 & 1 & 0 & 0 & 1 & \mathbf{1} \end{bmatrix} \oplus \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} \\ 0 & 0 & 0 & 1 & 1 & 1 & \mathbf{0} \\ 0 & 0 & 1 & 0 & 1 & 0 & \mathbf{1} \\ 0 & 0 & 1 & 1 & 0 & 1 & \mathbf{1} \\ 0 & 1 & 0 & 0 & 1 & 1 & \mathbf{1} \\ 0 & 1 & 0 & 1 & 0 & 0 & \mathbf{1} \\ 0 & 1 & 1 & 0 & 0 & 1 & \mathbf{0} \end{bmatrix}.$$

Таким образом, было показано, что квазисиндром обладает теми же свойствами, что и синдром, то есть зависит только от местоположения ошибки и различным синдромам соответствуют различные вектора одиночных ошибок.

Обобщив действия при декодировании, можно составить следующий алгоритм квазисиндромного декодирования II типа:

1. Составление уравнений для матрицы восстановления при условии отсутствия ошибок.

2. Формирование восстанавливающей матрицы D кода по коэффициентам составленных уравнений.

3. Нахождение для каждого принятого кодового символа составляющей квазисиндрома.

4. Определение квазисиндрома путем суммирования, по модулю два составляющей квазисиндрома с несистематической частью принятого кодового символа.

5. Определение вектора одиночной ошибки из таблицы соответствия.

6. Исправление ошибки в кодовом символе с помощью вектора ошибки, определяемого квазисиндромом из таблицы соответствия.

Структурная схема кодека, исправляющего одиночные ошибки методом квазисиндромного метода декодирования II типа, изображена на рисунке 1.1.

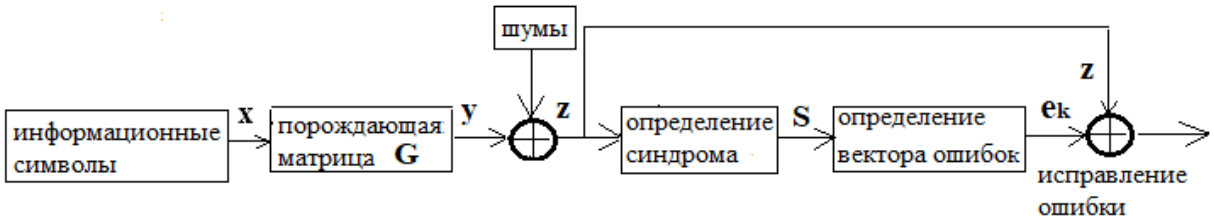


Рисунок 1.1 – Структурная схема кодека

На основе данной схемы составим модель кодека в среде Simulink.

2 Запуск и работа в среде Simulink

Для запуска системы Simulink необходимо предварительно выполнить запуск системы **MatLab**. После открытия командного окна системы **MatLab** нужно запустить систему **Simulink**. Это можно сделать одним из трех способов [3]:

- нажать кнопку (**Simulink**) на панели инструментов системы **MatLab**;
- в строке командного окна **MatLab** напечатать **Simulink** и нажать клавишу **Enter;**
- выполнить опцию **Open** в меню **File** и открыть файл модели (**mdl**-файл).

Последний способ предпочтителен при запуске уже готовой и отлаженной модели, когда требуется лишь провести моделирование и не нужно добавлять новые блоки в модель. При применении двух первых способов открывается окно обозревателя библиотеки блоков (**Simulink Library Browser**) [3].

На рисунке 2.1 выведена библиотека системы **Simulink** и показаны ее разделы. Основная библиотека системы содержит следующие разделы:

- **Continuous** – блоки аналоговых элементов;
- **Discontinuous** – блоки нелинейных элементов;
- **Discrete** – блоки дискретных элементов;
- **Look-Up Tables** – блоки таблиц;
- **Math Operations** – блоки элементов, определяющие математические операции;
- **Model Verification** – блоки проверки свойств сигнала;
- **Model-Wide Utilities** – раздел дополнительных утилит;
- **Port&Subsystems** – порты и подсистемы;
- **Signal Attributes** – блоки задания свойств сигналов;
- **Signal Routing** – блоки маршрутизации сигналов;

- **Sinks** – блоки приема и отображения сигналов;
- **Sources** – блоки источников сигнала;
- **User-Defined Function** – функции, определяемые пользователем.

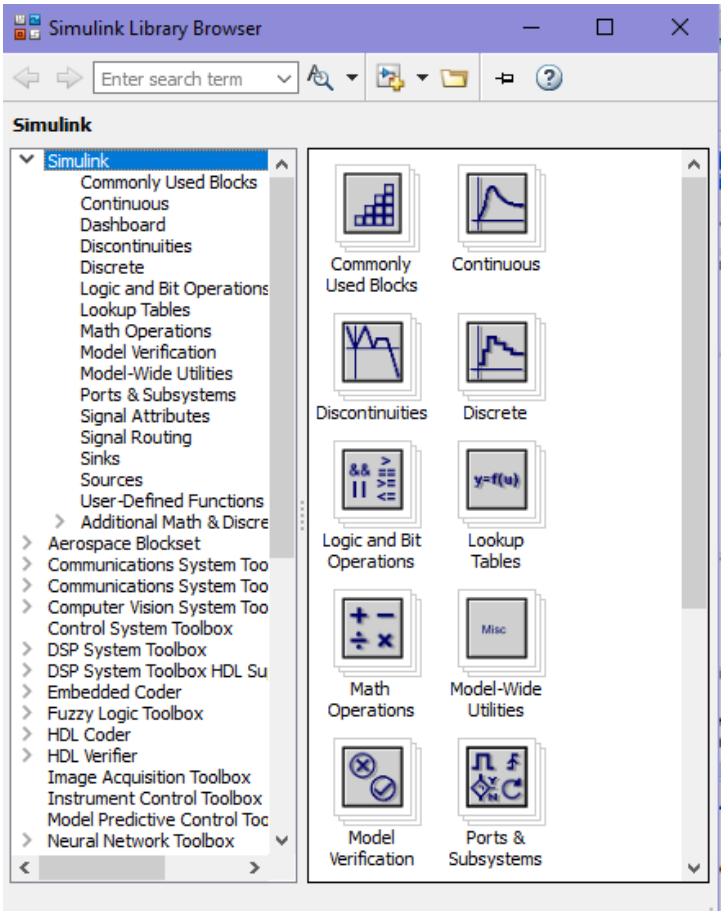


Рисунок 2.1. – Библиотека блоков **Simulink Library Browser**

При работе элементы разделов библиотек "перетаскивают" в рабочую область удержанием левой кнопки мыши на соответствующих изображениях. Для соединения элементов достаточно указать курсором мыши на начало соединения и затем при нажатии левой кнопки мыши протянуть соединение в его конец.

При двойном щелчке левой кнопки мыши на выделенном блоке всплывает меню, в котором задаются параметры блоков.

Работа **Simulink** происходит на фоне открытого окна системы MatLab, закрытие которого приведѐт к выходу из Simulink [3].

3 Описание лабораторного макета

Функциональная модель кодека, по которой будет проводиться исследование процессов кодирования и квазисиндромного декодирования на примере кода Хэмминга (7,4), представлена на рисунке 3.1.

В передатчике источник сигнала представлен с помощью блоков **Random Number**, **Constant** и **Relational Operator**, которые генерируют сигнал в виде однополярных псевдослучайных последовательностей. В буфере (блок **Buffer**) происходит накопление 4 бит, которые являются информационными. Затем получаем контрольные биты четности, демультиплексируя блоком **Demux** и суммируя по модулю 2 соответствующие информационные биты с помощью блоков **Xor**. Далее с помощью блока **Multiport Switch** увеличивается скорость передачи битов кодового символа, так как были добавлены биты четности.

Затем биты подаются на имитатор ошибок, состоящий из блоков **Pulse Generator** и **Xor 3**, где мы можем выбрать место одиночной ошибки, подбирая период и скважность и подавая с помощью задержки на нужном месте единицу. Далее стоит экстраполятор нулевого порядка, определяющий шаг дискретности перед буферизацией.

В приёмнике очередной буфер **Buffer 1** с накоплением принятых битов формирует блок из 7 битов. Принятые последовательные биты с помощью демультиплексора **Demux** преобразуются в параллельные биты. Суммируя по модулю 2 с помощью блоков **Xor** соответствующие биты, формируется восстанавливающая матрица. Коэффициенты полученной матрицы суммируются по модулю 2 с несистематической частью принятого кодового символа, тем самым, формируя квазисиндром. Затем с помощью блока комбинаторной логики **Combinatorial Logic** каждому квазисиндрому ставится в соответствие вектор ошибок и он, суммируясь по модулю два (блок **Xor 7**) с информационными битами принятого кодового символа, исправляет одиночные ошибки.

После исправления (коррекции) ошибок мультипортовый ключ **Multiport Switch** возвращает исходную скорость передачи битов.

В ходе исследования была составлена таблица соответствия квазисиндромов и векторов одиночных ошибок для исправления кода (7, 4).

Таблица 3.1 – Соответствия квазисиндромов II типа и векторов ошибок для исправления кода (7, 4)

Квазисиндром	001	100	010	011	101	110	111
Вектор ошибки	e_1 (1000000)	e_2 (0100000)	e_3 (0010000)	e_4 (0001000)	e_5 (0000100)	e_6 (0000010)	e_7 (0000001)

Из данной таблицы видно, что полученные экспериментально квазисиндромы и вектора одиночных ошибок совпадают с расчетными данными, что показывает правильность работы кодека.

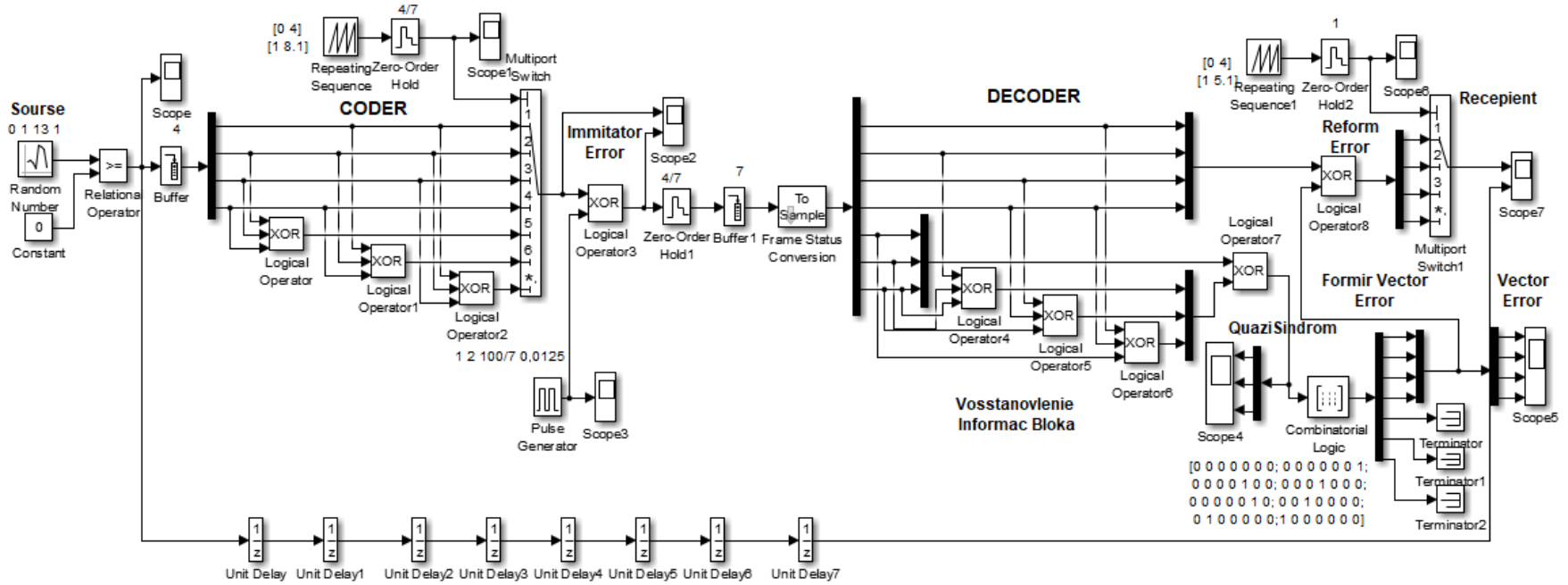
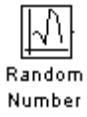


Рисунок 3.1 – Функциональная модель исследуемого кодека на основе кода Хемминга (7,4)

4 Описание функций используемых блоков библиотеки Simulink

Ниже описаны используемые в построении функциональной модели кодека блоки разделов библиотеки Simulink [3].



Random Number – источник случайного сигнала с нормальным распределением.

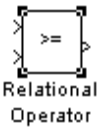
Назначение: формирование случайного сигнала с нормальным распределением уровня сигнала.

Параметры блока: Mean – среднее значение сигнала; Variance – дисперсия; Initial seed – начальное значение генератора случайного сигнала; Sample time – такт дискретности.

Constant – источник постоянного сигнала.

Назначение: задает сигнал постоянного уровня.

Параметры блока: Constant value – постоянная величина, значение которой может быть задано действительным или комплексным числом, вычисляемым выражением, вектором или массивом; флажок Interpret vector parameters as 1 - D – интерпретировать вектор как массив скаляров; флажок Show additional parameters – показать дополнительные параметры, в нашем случае не используется.



Relational Operator – блок выполнения операций отношения.

Назначение: сравнение текущих значений входных сигналов поступающих на входы.

*Параметры блока**:*** Relational Operator – тип операции отношения выбираемый из списка:

= =- тождественно равно;

~ = - не равно;

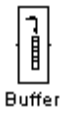
< - меньше;

< = - меньше или равно;

> = - больше или равно;

> - больше.

Флажок - Show additional parameters – показать дополнительные параметры – в нашем случае не используется.



Buffer – блок буферизации.

Назначение: служит для буферизации сигналов. Его работу можно уподобить получению воды ищ единственного крана с помощью ведер – заполняется одно ведро, затем другое и т.д. Таким образом, поток данных сигнала дробится на части (фреймы) заданного размера. Размер буфера, выделяемого под задержанный сигнал, в байтах (число, кратное 8, по умолчанию 1024 байта).

Scope – блок осциллографа.

Назначение: построение графиков исследуемых сигналов как функций времени. Открытие окна осциллографа производится двойным щелчком левой кнопки мыши на пиктограмме блока. В случае векторного сигнала каждая компонента вектора отображается отдельным цветом.

Настройка окна осциллографа выполняется с помощью панелей инструментов, позволяющих: осуществить печать содержимого окна осциллографа; установить *параметры,* в частности, Number of axes - число входов осциллографа, Time range – отображаемый временной интервал и

другие; изменить масштабы графиков; установить и сохранить настройки; перевести в плавающий режим и так далее.

 **Demux –** блок демультиплексора.

Назначение: разделение входного векторного сигнала на составляющие (последовательного представления в параллельное).

Параметры блока: Number of output – количество выходов; Display option – способ отображения выбирается из списка: bar – вертикальный узкий прямоугольник черного цвета; none – прямоугольник с белым фоном без отображения меток входных сигналов. Флажок Bus Selection Mode – режим разделения векторных сигналов в шине, используется для разделения сигналов, объединенных в шину.

Logical Operation - блок выполнения логических операций.

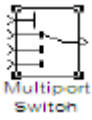
Назначение: реализует одну из базовых логических операций.

Параметры блока: Operator - вид реализуемой логической операции, выбирается из списка:

AND - логическое умножение (операция логическое И), OR - логическое сложение (операция логическое ИЛИ), NAND - операция И-НЕ, NOR - операция ИЛИ-НЕ, XOR - операция сложения по модулю 2 (операция ИСКЛЮЧАЮЩЕЕ ИЛИ), NOT - логическое отрицание (логическое НЕ); Number of input ports- количество входных портов; Флажок Show additional parameters – показать дополнительные параметры (в нашем случае не используется); Флажок Require all inputs to have same data type- установить одинаковый тип входных данных; Output data type mode - выбор типа выходных данных из списка: Boolean (двоичный), Logical (логический),

Specify via dialog (задаваемый дополнительным списком). В последнем случае появится окно списка Output data type - тип выходных данных.

Входные сигналы могут быть как действительного, так и логического типа (Boolean). Выходным сигналом блока является 1, если результат вычисления логической операции есть ИСТИНА, и 0, если результат – ЛОЖЬ.



Multiport Switch – блок многовходового переключателя.

Назначение: выполняет переключение входных сигналов на выход по сигналу управления, задающему номер активного входного порта.

Параметры блока: Number of inputs – количество входов; флажок Show additional parameters – показать дополнительные параметры, в нашем случае не используется. Блок Multiport Switch пропускает на выход сигнал с того входного порта, номер которого равен текущему значению управляющего сигнала. Если управляющий сигнал не является сигналом целого типа, то блок Multiport Switch производит округление значения в соответствии со способом, выбранным в графе дополнительного параметра Round integer calculations toward.

Repeating Sequence – источник периодического сигнала.

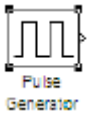
Назначение: формирование заданного пользователем периодического сигнала.

Параметры блока: Time values – вектор значений времени; Output values – вектор значений сигнала. Блок выполняет линейную интерполяцию выходного сигнала для моментов времени не совпадающих со значениями, заданными вектором Time values.

Zero-Older Hold – экстраполятор нулевого порядка.

Назначение: экстраполяция входного сигнала на интервале дискретизации. Блок фиксирует значение входного сигнала в начале интервала дискретизации и поддерживает на выходе это значение до окончания интервала дискретизации. Затем выходной сигнал изменяется скачком до величины входного сигнала на следующем шаге дискретизации.

Параметры блока: Sample time – такт дискретности. Блок экстраполятора нулевого порядка может использоваться также для согласования работы дискретных блоков, имеющих разные такты дискретности.



Pulse generator – блок источника импульсного сигнала.

Назначение: формирование сигнала в форме прямоугольных импульсов.

Параметры блока: Pulse Type – способ формирования сигнала, может принимать два значения: Time-based – по текущему времени; Sample-based – по величине такта дискретности и количеству шагов моделирования. Вид окна параметров зависит от выбранного способа формирования сигнала. Amplitude – амплитуда; Period – период, задается в секундах при способе Time-based или количеством тактов при способе Sample-based; Pulse width – ширина импульса, задается в процентах от периода при способе Time-based или количеством тактов при способе Sample-based; Phase delay – фазовая задержка, задается в секундах при способе Time-based или количеством тактов при способе Sample-based; Sample time – такт дискретности; флажок Interpret vector parameters as 1 - D – интерпретировать вектор как массив скаляров.

 **Mux –** блок мультиплексора.

Назначение: объединяет входные сигналы в вектор.

Параметры блока: Number of Inputs – количество входов; Display option – способ отображения, выбирается из списка: bar – вертикальный узкий прямоугольник черного цвета; signals – прямоугольник с белым фоном и отображением меток входных сигналов; none – прямоугольник с белым фоном без отображения меток входных сигналов.

Combinatorial Logic – блок комбинаторной логики.

Назначение: преобразует входной векторный сигнал в соответствии с таблицей истинности. Таблица истинности представляет собой список возможных выходных значений блока. Каждому состоянию входного векторного сигнала соответствует определенное логическое состояние выходного сигнала.

Параметры блока: Truth table – таблица истинности. Так для возможных значений входного вектора, соответствующего дибитам, таблица истинности имеет вид [00; 01; 10; 11].

Terminator – концевой приемник.

Назначение: Блок применяется как заглушка для сигнала, поступающего с выхода другого блока. В том случае, когда выход блока оказывается не подключенным ко входу другого блока Simulink выдает предупреждение в командном окне системы MatLab. Для исключения таких ситуаций следует использовать блок Terminator.

Параметры блока: Нет.

5 Экспериментальное задание

В ходе выполнения данной лабораторной работы требуется выполнить следующие пункты:

1. Собрать модель для исследования квазисиндромного декодера кода (7, 4) в соответствии с рисунком 3.1.

2. Выставить параметры блоков Sim-модели, согласованные с исходными параметрами блока источника случайного сигнала с нормальным распределением (Random Number), например: Mean = 0; Variance = 1; Seed = 13; Sample time = 1.

3. Пронаблюдать и зафиксировать основные осциллограммы, иллюстрирующие работу квазисиндромного декодера (**Scope 2, Scope 4, Scope 5** и **Scope 7)**.

4. На блоке **Pulse generator** имитатора ошибок поочерёдно выставить задержки соответствующие всем одиночным ошибкам и с блоков **Scope 4** и **Scope 5** записать квазисиндромы и соответствующие им вектора ошибок (сформировать таблицу соответствия).

5. Написать отчет с кратким описанием принципа работы кодека.

6. Защитить отчет.

6 Контрольные вопросы

1. Что такое информационный сигнал?
2. Что такое избыточные биты?
3. Что такое кодовый символ?
4. Что такое порождающая матрица и как она формируется?
5. Что такое восстанавливающая матрица и как она формируется?
6. Что такое вектор ошибки?
7. Что такое квазисиндром, его свойства?
8. Как образуется квазисиндром?

Список использованных источников

1. Блейхут Р. Теория и практика кодов, контролирующих ошибки: Пер с англ. / Р. Блейхут – М.: Мир, 1986.– 576 с.

2. Помехоустойчивое кодирование и квазисиндромное декодирование блочных кодов: учебно-методическое пособие по лабораторной и самостоятельной работе и практическим занятиям / В.А. Кологривов, А.А. Алишери. – Томск. ТУСУР, 2018.– 22 с.

3. Черных И. В. Simulink: среда создания инженерных приложений / Под общ. ред. к. т. н. В. Г. Потемкина. – М.:ДИАЛОГ-МИФИ, 2003.– 496 с.