Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра радиотехнических систем (РТС)

Кологривов В.А., Алишери А.А.

ПОМЕХОУСТОЙЧИВОЕ КОДИРОВАНИЕ И КВАЗИСИНДРОМНОЕ ДЕКОДИРОВАНИЕ БЛОКОВЫХ КОДОВ

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

2018

Министерство образования и науки 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.

_____ Мелихов С.В.

«___» _____ 2018 г.

ПОМЕХОУСТОЙЧИВОЕ КОДИРОВАНИЕ И КВАЗИСИНДРОМНОЕ ДЕКОДИРОВАНИЕ 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. 1В4 Алишери А.А. _____

2018

Кологривов В.А., Алишери А.А.

«Помехоустойчивое кодирование и квазисиндромное декодирование 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, 2013.- 22 с.

Учебно-методическое пособие содержит описание систематического кода (7, 3) с квазисиндромным декодированием, выполненной в среде функционального моделирования ***Simulink*** системы для инженерных и научных расчетов ***Matlab.***

В пособии приведены краткие теоретические сведения по кодированию и квазисиндромному декодированию блоковых кодов, краткая характеристика пакета ***Simulink*** системы ***Matlab,*** описание виртуального лабораторного макета и используемых блоков библиотеки ***Simulink,*** а также требования к экспериментальному исследованию и контрольные вопросы, ответы на которые необходимы для успешной защиты лабораторной работы.

АННОТАЦИЯ

Лабораторная работа **«Помехоустойчивое кодирование и квазисиндромное декодирование блоковых кодов»** посвящена экспериментальному исследованию модели кодека с использованием пакета функционального моделирования ***Simulink*** системы для инженерных и научных расчетов ***Matlab.***

Работа «Помехоустойчивое кодирование и квазисиндромное декодирование блоковых кодов» относится к циклу лабораторных работ по разделу «Помехоустойчивое кодирование» входящему в дисциплины по направлению «Инфокоммуникационные технологии и системы связи».

В описании сформулирована цель лабораторной работы, приведены краткие теоретические сведения по блоковым кодам, краткая характеристика пакета ***Simulink*** системы ***Matlab***, описание виртуального лабораторного макета и используемых блоках библиотеки ***Simulink,*** а также требования к экспериментальному исследованию и контрольные вопросы, ответы на которые необходимы для успешной защиты лабораторной работы.

СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ. КРАТКИЕ СВЕДЕНИЯ

Цель работы: изучение структуры и принципа блочного кодирования и квазисиндромного декодирования алгебраических кодов с использование пакета функционального моделирования Simulink.

Теоретические сведения

Идея помехоустойчивого кодирования заключается во введении в информационный поток дозированной избыточности. Это позволяет обнаруживать и частично исправлять ошибки передачи данных. Коды делятся на блочные и непрерывные.

Информационный поток разбивается на блоки длиной $\boldsymbol{k}$ битов, представляющие информационные символы. При кодировании эти блоки наращиваются избыточными битами до длины $\boldsymbol{n}$ и называются канальными символами. Избыточные биты $\boldsymbol{n-k}$ называются битами четности. Коды, содержащие информационные биты в чистом виде, называются систематическими [1].

Операции кодирования и декодирования удобно представить в векторно-матричной форме. Кодовый вектор-символ $\boldsymbol{y}$ формируется из информационного вектора $\boldsymbol{x}$ путем введения избыточных битов. На примере кодека кода (7, 3) формирование кодового символа можно представить в виде:

$$\boldsymbol{x \cdot G = y} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix} \cdot \begin{vmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} \end{vmatrix} = \begin{vmatrix} \boldsymbol{y_k} \\ \boldsymbol{y_{k+1}} \\ \boldsymbol{y_{k+2}} \\ \boldsymbol{y_{k+3}} \\ \boldsymbol{y_{k+4}} \\ \boldsymbol{y_{k+5}} \\ \boldsymbol{y_{k+6}} \end{vmatrix},$$

где $\boldsymbol{G}$ - это матрица формирующей системы. Кодирование удобно представлять в виде вектор-строк:

$$\boldsymbol{x \cdot G = y},$$

где $\boldsymbol{G}$ - это порождающая матрица. Для систематического кода структура порождающей матрицы имеет вид:

$$\boldsymbol{G} = \left[\boldsymbol{I}_{\boldsymbol{k}\cdot\boldsymbol{k}}\ \boldsymbol{P}_{\boldsymbol{k}\cdot(\boldsymbol{n}-\boldsymbol{k})}\right],$$

где $\boldsymbol{I}_{\boldsymbol{k}\cdot\boldsymbol{k}}$- единичный блок; $\boldsymbol{P}_{\boldsymbol{k}\cdot(\boldsymbol{n}-\boldsymbol{k})}$- проверочный блок.

Отличие квазисиндромного от синдромного декодирования, заключается в том, что здесь находятся уравнения, восстанавливающее в отсутствия ошибок исходные биты. Матрицу коэффициентов этой системы уравнении $\boldsymbol{D}$ назовем восстанавливающей.

В матрице формирующей системы ($\boldsymbol{G}^{\boldsymbol{t}}$) можем увидеть, что восстановить информационные биты по кодовому символу можно в виде трех уравнении:

$$\begin{gathered} x_k = y_{k+4} \oplus y_{k+5} \oplus y_{k+6} \\ x_{k+1} = y_{k+2} \oplus y_{k+4} \oplus y_{k+5}. \\ x_{k+2} = y_k \oplus y_{k+3} \oplus y_{k+4} \end{gathered}$$

Восстанавливающая матрица этой системы имеет вид:

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

Идею образования квазисиндрома можно проиллюстрировать на примера кода (7, 3). Поскольку квазисиндром, как и синдром зависит от места локализации ошибок, то вместо кодовых символов можно взять все возможные вектора одиночных ошибок. Тогда, умножая вектора ошибок на восстанавливающую матрицу, и, добавляя информационную часть символов, получаем квазисиндром:

$$E \cdot D = D \oplus X = S;$$

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

где E – матрица векторов одиночных ошибок,

D – восстанавливающая матрица,

X - систематическая часть символов ошибок,

S – матрица квазисиндромов.

Из приведенного примера можно сразу увидеть соответствия между квазисиндромами и векторами ошибок. А так же, что различные вектора ошибок имеют разные квазисиндромы. Этот факт говорит о том, что данный квазисиндромный декодер способен исправить все одиночные ошибки.

Ниже приведена таблица 1.1 соответствия квазисиндромов и векторов ошибок для кода (7, 3), составленная по данным приведенного примера. В таблице каждому синдрому соответствует свой вектор ошибки.

Таблица 1.1 – Таблица соответствия синдромов и векторов ошибок для исправления кода (7, 3)

Синдром	101	010	011	001	111	110	100
Вектор ошибки	e_1	e_2	e_3	e_4	e_5	e_6	e_7

Таким образом, квазисиндром связан с местоположением ошибочного бита в принятой кодовой комбинации на входе декодера.

Можно сформулировать алгоритм квазисиндромного декодирования. ***Алгоритм квазисиндромного декодирования блоковых кодов*** состоит в следующем:

1. Формирование восстанавливающей матрицы кода $\boldsymbol{D}$.

2. Для каждого принятого символа находим предполагаемый информационный символ.

3. Суммируя, по модулю два предполагаемый информационный символ с систематической частью принятого символа, получаем квазисиндром.

4. Из таблицы соответствия находим вектор ошибки.

5. Исправление ошибки в кодовой комбинации выполняется по правилу

$$\boldsymbol{y_i = z_i \oplus e_i.}$$

2. ЗАПУСК И РАБОТА С ПАКЕТОМ 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:

-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**) [2].

На рисунке 2.1 выведена библиотека системы **Simulink** (в левой части окна) и показаны ее разделы (в правой части окна). 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.

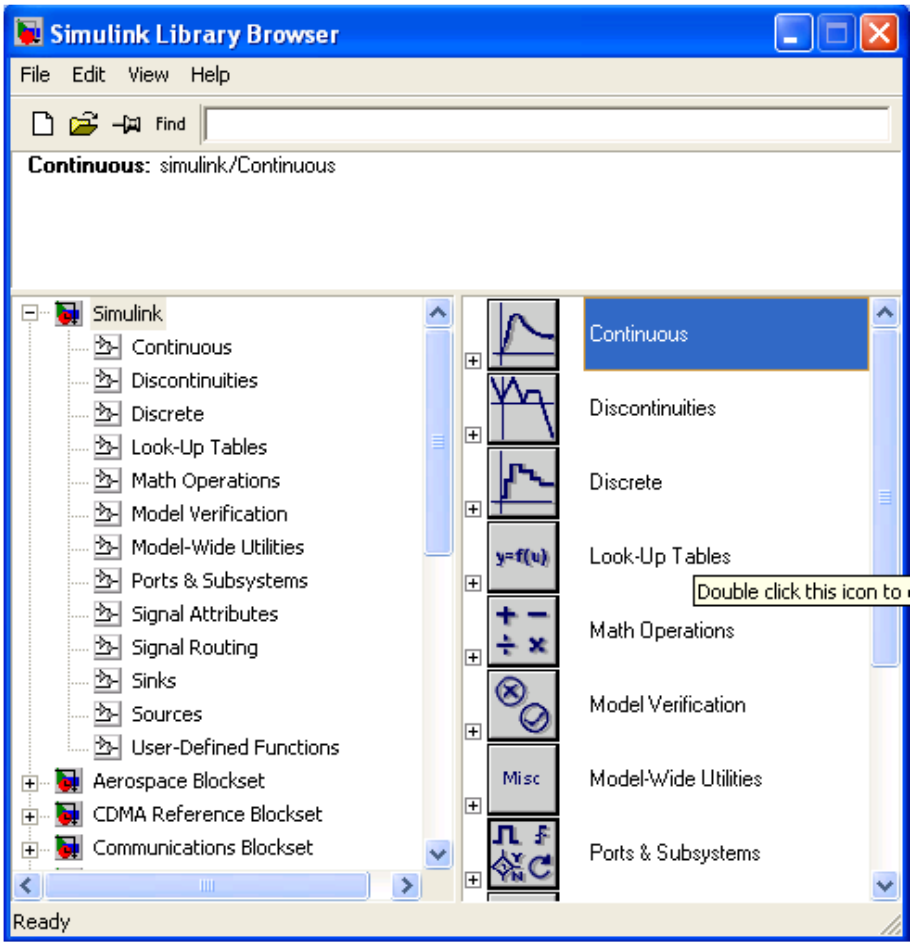


Рисунок 2.1. – Библиотека блоков **Simulink Library Browser**

Список разделов библиотеки представлен в виде дерева, и правила работы с ним являются общими для списков такого вида: пиктограмма свернутого узла дерева содержит символ «+», а пиктограмма развернутого – символ «-».

Для того чтобы развернуть или свернуть узел дерева, достаточно щелкнуть на его пиктограмме левой клавишей мыши (***ЛКМ***). При выборе соответствующего раздела библиотеки его содержимое отображается в правой части окна.

При работе элементы разделов библиотек **"перетаскивают"** в рабочую область удержанием ***ЛКМ*** 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 ***ЛКМ*** 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. ОПИСАНИЕ ЛАБОРАТОРНОГО МАКЕТА

Приведем краткое описание работы блочного кода с квазисиндромным декодированием кодов, на основе **Sim**-модели представленной на рисунке 4.1.

На вход кодера приходит поток битов, которые накапливаются в блоке **Buffer,** затем демультиплексируются блоком **DEMUX** и с помощью блоков **XOR,** суммируя сочетания различных информационных битов, получаем избыточные биты.

Далее с помощью блока **Multiport Switch** увеличивается скорость передачи битов кодового символа, так как были добавлены биты четности.

Затем кодовые символы проходят через имитатор ошибок, состоящие из блоков **Pulse generator** и **XOR.** Изменяя период и скважность **Pulse generator** можно имитировать, например, одинаковые одиночные ошибки в каждом передаваемом символе.

После имитатора ошибок блоком **Buffer** формируются принятый, возможно с ошибками, кодовый символ. Далее блоком **DEMUX** он разделяется на биты и с помощью блоков **XOR** реализуются уравнения восстанавливающие информационные биты. Затем еще раз, суммируя по модулю два восстановленные и принятые информационные биты, получаем квазисиндром.

Каждому квазисиндрому с помощью блока **Combinatorial Logic** ставится в соответствие вектор ошибок и он, суммируясь по модулю два (блок **XOR)** с систематической частью принятого символа, исправляет ошибку.

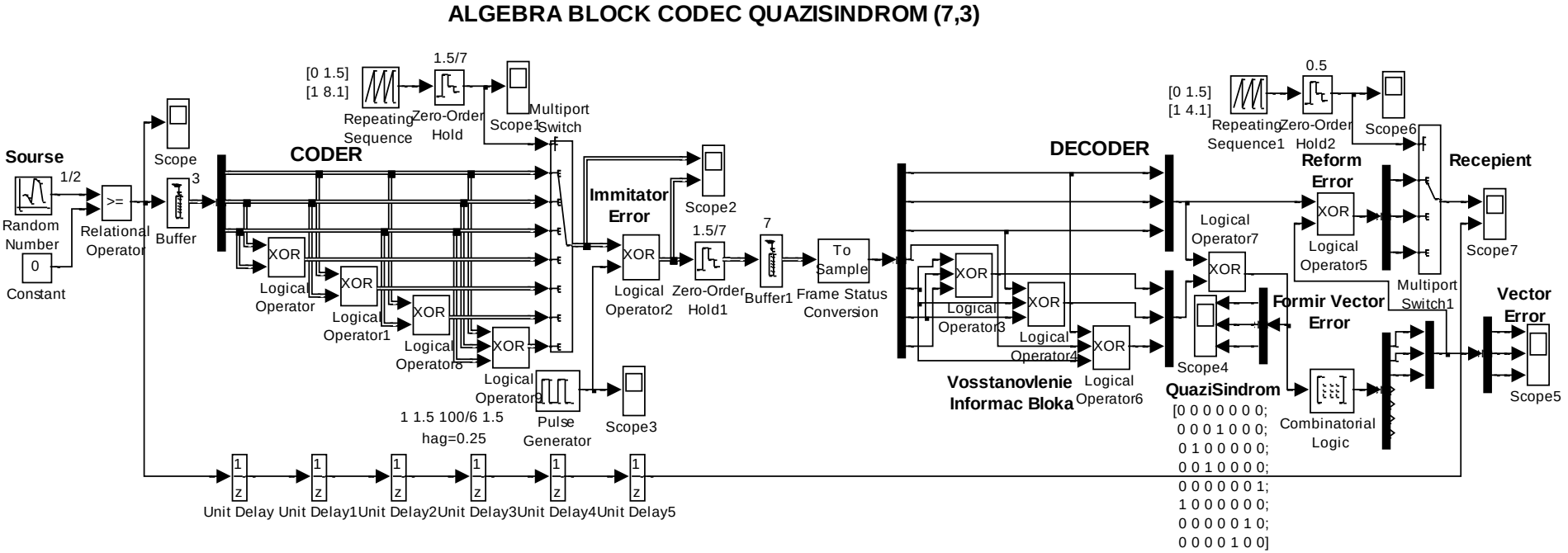


Рисунок 4.1 – Функциональная схема кодека систематического кода (7, 3) с квазисиндромным декодером

После исправления ошибок с помощью блока **Multiport Switch** возвращаемся к прежней скорости передачи битов.

Вектора ошибок смотрим в Scope 5 и так как исправляем только информационную часть принятого символа, то с блока **Combinatorial Logic** берем только часть вектора ошибоки. В связи с этим, начиная, с ошибки в четвертом бите на Scope 5 наблюдается, нулевые вектора ошибок.

4. ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ БЛОКОВ БИБЛИОТЕКИ SIMULINK

Ниже описаны основные блоки базовых разделов библиотеки Simulink [3, 4, 5], используемые в функциональной схеме.

Random number – источник случайного сигнала с нормальным распределением. *Назначение:* Формирование случайного сигнала с равномерным распределением уровня сигнала. *Параметры блока:* **Minimum –** минимальный уровень сигнала; **Maximum –** максимальный уровень сигнала; **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.

На рисунке 2.1 выведена библиотека системы **Simulink** (в левой части окна) и показаны ее разделы (в правой части окна). 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;

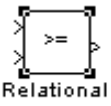
-**Signal Attributes** – блоки задания свойств сигналов;

-**Signal Routing** – блоки маршрутизации сигналов;

-**Sinks** – блоки приема и отображения сигналов;

-**Sources** – блоки источников сигнала;

-**User-Defined Function** – функции, определяемые пользователем.



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;

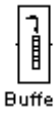
< = - меньше или равно;

> = - больше или равно;

> - больше.

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

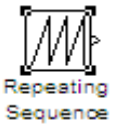
Scope – блок осциллографа. *Назначение:* построение графиков исследуемых сигналов как функций времени. Открытие окна осциллографа производится двойным щелчком **ЛКМ** на пиктограмме блока. В случае векторного сигнала каждая компонента вектора отображается отдельным цветом. Настройка окна осциллографа 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.



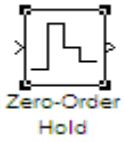
Buffer – блок буферизации. *Назначение:* служит для буферизации сигналов. Его работу можно уподобить получению воды ищ единственного крана с помощью ведер – заполняется одно ведро, затем другое и т.д. Таким оброзом, поток данных сигнала дробится на части (фреймы) заданного размера. Размер буфера, выделяемого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

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.



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.**



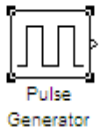
Zero-Older Hold – экстраполятор нулевого порядка. *Назначение:* экстраполяция входного сигнала на интервале дискретизации. Блок фиксирует значение входного сигнала в начале интервала дискретизации и поддерживает на выходе это значение до окончания интервала дискретизации. Затем выходной сигнал изменяется 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.

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**.



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.

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].**

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.

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.**

5. ЭКСПЕРИМЕНТАЛЬНОЕ ЗАДАНИЕ

1. Собрать **Sim**-модель для исследования квазисиндромного декодера кода (7, 3) в соответствии с рисунком 4.1.

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 = 0.5.

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 квазисиндром?
2. Что такое вектор ошибки?
3. Что такое кодовый символ?
4. Что такое информационный сигнал?
5. Что такое порождающая матрица?
6. Что такое восстанавливающая матрица?
7. Что такое избыточные биты?

8. Что такое канальные символы?
9. Что такое матрица формирующей системы?
10. Как образуется квазисиндром?

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Банкет В.Л., Иващенко П.В., Ищенко Н.А. Помехоустойчивое кодирование в телекоммуникационных системах. Учеб. пособие. – Одесса: ОНАС, 2011.– 104 с.

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