МИНСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра радиотехнических систем (РТС)

Утверждаю:

Зав. каф. РТС, проф., д.т.н.

_____Мелихов С.В.

_____2018 г.

Кологривов В.А., Чыпсынак Ч.А.

ИССЛЕДОВАНИЕ КОДЕКА С ИСПОЛЬЗОВАНИЕМ МАЖОРИТАРНОГО ДЕКОДЕРА

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.

_____Чыпсынак Ч.А.

2018 г.

Кологривов В. А., Чыпсынак Ч. А.

«Исследование кодека с использованием мажоритарного декодера»: Учебно-методическое пособие по лабораторной работе и самостоятельной практическим занятиям для студентов направления «Инфокоммуникационные технологии и системы связи» - Томск: ТУСУР. Научно-образовательный портал, 2018. – 26 с.

Учебно-методическое пособие содержит описание функциональной модели кодека для исследования мажоритарного декодирования блочных кодов, выполненной в среде функционального моделирования ***Simulink***, системы для инженерных и научных расчетов ***MatLab***.

В пособии приведены краткие теоретические сведения о мажоритарном декодировании блочных кодов, краткая характеристика пакета ***Simulink*** системы ***MatLab***, описание виртуальных лабораторных макетов и используемых блоков библиотеки ***Simulink***.

Аннотация

Лабораторная работа **«Исследование кодека с использованием мажоритарного декодера»** посвящена экспериментальному исследованию кодека с использованием мажоритарного декодера, пакета функционального моделирования Simulink и системы для инженерных и научных расчетов MatLab.

Работа **«Исследование кодеков с использованием мажоритарного декодера»** относится к циклу лабораторных работ «Помехоустойчивое кодирование», входящему в дисциплины по направлению «Инфокоммуникационные технологии и системы связи».

В описании сформулирована цель лабораторной работы, приведены краткие теоретические сведения о мажоритарном декодировании, краткая характеристика пакета Simulink системы MatLab, описание виртуальных лабораторных макетов и используемых блоков библиотеки Simulink, а также требования к экспериментальному исследованию и контрольные вопросы, ответы на которые необходимы для успешной защиты лабораторной работы.

Лабораторная работа рассчитана на одно лабораторное занятие на 4 часа.

Содержание

1 Цель работы. Краткие сведения из теории

Цель работы: разработка моделей каналов передачи, блочных кодеров и мажоритарных декодеров. Разработку ввести в среде функционального моделирования **Simulink** системы **Matlab.**

1.1 Теоретическая часть

Ранее средства кодирования играли вспомогательную роль и не рассматривались как отдельный предмет математического изучения, но с появлением компьютеров ситуация радикально изменилась. Кодирование буквально пронизывает информационные технологии и является центральным вопросом при решении самых разных (практически всех) задач программирования:

- представление данных произвольной природы (например, чисел, текста, графики) в памяти компьютера;
- защита информации от несанкционированного доступа;
- обеспечение помехоустойчивости при передаче данных по каналам связи;
- сжатие информации в базах данных.

Кодирование – это преобразования информации в форму удобную для передачи по определенному каналу связи.

Декодирование – восстановление принятого сообщения из закодированного вида в вид доступный для потребителя [1].

Задача декодирования состоит в получении k - элементной комбинации из принятого n - разрядного кодового слова при одновременном обнаружении или исправлении ошибок.

Энергетический выигрыш кодирования. Положительным эффектом *помехоустойчивого кодирования* является либо снижение вероятности ошибки,

либо снижение энергетики передачи при той же вероятности ошибки, либо и то, и другое одновременно. Таким образом, кодирование расширяет возможности компромисса между полосой и энергетикой канала, присущего любой системе связи.

Линейные блочные коды позволяют представить информационные и кодовые слова в виде двоичных векторов, что позволяет описать процессы кодирования и декодирования с помощью аппарата линейной алгебры, с учетом того, что компонентами вводимых векторов и матриц являются символы «0» и «1». Операции над двоичными компонентами производятся при этом по правилам арифметики по модулю 2.

Поскольку между информационными и кодовыми словами существует взаимно однозначное соответствие, процесс кодирования может быть осуществлен с использованием таблицы соответствий, хранящейся в памяти кодера. Однако, для длинных кодов такой метод неприемлем, так как требует большой объем памяти для хранения таблицы.

Вводится понятие так называемой порождающей матрицы G. Оно основано на том, что подпространство всех кодовых слов линейного блочного *(n, k)* – кода имеет некоторый базис $(v_0, v_1, \dots, v_{k-1})$, через который может быть выражено любое кодовое слово этого кода.

$$v = u_0 \cdot v_0 + u_1 \cdot v_1 + \cdots + u_{k-1} \cdot v_{k-1}, \qquad (1.1)$$

где $u_0 \in \{0,1\}, 0 < i < k$.

Векторы базиса образуют порождающую матрицу G размера $k \times n$.

$$G = \begin{bmatrix} v_0 \\ v_1 \\ \cdots \\ v_{k-1} \end{bmatrix} = \begin{bmatrix} v_{0,1} & v_{0,1} & \cdots & v_{0,n-1} \\ v_{1,0} & v_{1,1} & \cdots & v_{1,n-1} \\ \cdots & \cdots & \cdots & \cdots \\ v_{k-1,0} & v_{k-1,1} & \cdots & v_{k-1,n-1} \end{bmatrix}$$

Тогда уравнение (1.1) принимает вид

$$v = u \cdot G,$$

где $u = (u_0, u_1, \dots, u_{k-1})$ – информационное слово;

$v = (v_0, v_1, \dots, v_{k-1})$ - кодовое слово.

Мажоритарное декодирование (*majority* –большинство). Для линейных кодов, рассчитанных на исправление многократных ошибок, часто более простыми оказываются декодирующие устройства, построенные по мажоритарному принципу. Этот метод декодирования называют также принципом голосования или способом декодирования по большинству проверок.

Некоторые блоковые коды допускают реализацию простого алгоритма декодирования – алгоритма **мажоритарного декодирования**, который основан на возможности выразить каждый информационный символ кодовой комбинации несколькими способами через другие принятые символы. Для иллюстрации этого алгоритма рассмотрим систематический код (7, 3) с порождающей матрицей [2]:

$$G = \begin{vmatrix} 1\,0\,0\,1\,1\,1\,0 \\ 0\,1\,0\,0\,1\,1\,1 \\ 0\,0\,1\,1\,1\,0\,1 \end{vmatrix}.$$

Данной матрице соответствуют проверочная матрица:

$$H = \begin{vmatrix} 1\,0\,1\,1\;\;0\,0\;\;0 \\ 1\,1\,1\,0\;\;1\,0\;\;0 \\ 1\,1\,0\,0\;\;0\,1\;\;0 \\ 0\,1\,1\,0\;\;0\,0\;\;1 \end{vmatrix}, \qquad (1.2)$$

а также транспонированная проверочная матрица:

$$H^t = \begin{vmatrix} 1\,1\,10 \\ 0\,1\,11 \\ 1\,1\,01 \\ 1\,0\,00 \\ 0\,1\,00 \\ 0\,0\,10 \\ 0\,0\,01 \end{vmatrix}. \qquad (1.3)$$

Обозначим принятую из канала кодовую комбинацию как $b = (b_1, b_2, b_3, b_4, b_5, b_6, b_7)$.

Поскольку рассматриваемый код – систематический, первые три символа (b_1, b_2, b_3) являются *информационными.* Используя структурные свойства этого кода, можно при декодировании сформировать как ***тривиальные,*** так и ***составные оценки*** *информационных символов*, которые представлены в таблице 1.1. На основе столбцов проверочной матрицы (1.3) запишем *проверочные соотношения*

$$b_1 \oplus b_3 \oplus b_4 = 0,\ b_1 \oplus b_2 \oplus b_3 \oplus b_5 = 0,\ b_1 \oplus b_2 \oplus b_6 = 0,\ b_2 \oplus b_3 \oplus b_7 = 0 \quad (1.4)$$

которые позволяют сформировать ***составные оценки.***

Например, на основе первого равенства из (1.4) следует *составная оценка* первого информационного символа $b_1 = b_3 \oplus b_4$. *Тривиальная оценка* этого символа и есть, собственно, этот символ $b_1 = b_1$, поскольку код систематический.

Выражения для остальных информационных символов составлены аналогично. Они представлены в этой же таблице 1.1.

После формирования оценок они подаются на *мажоритарный элемент*, в котором решение о каждом информационном символе выносится «*по большинству голосов*». К примеру, если оценки информационного символа b_1 имеют вид:

$$b_1 = b_1 = 1,\ b_1 = b_3 \oplus b_4 = 1,\ b_1 = b_5 \oplus b_7 = 1,\ b_1 = b_2 \oplus b_6 = 0,$$

среди которых количество оценок «1» превышает количество оценок «0», то мажоритарный элемент выносит *решение «**по большинству**»:* $b_1 = 1$.

Таблица 1.1 – Мажоритарное декодирование блокового кода

Оценки информационных символов		
Оценки символа b_1	Оценки символа b_2	Оценки символа b_3
Тривиальные		
$b_1 = b_1$	$b_2 = b_2$	$b_3 = b_3$
Составные		
$b_1 = b_3 \oplus b_4$ $b_1 = b_5 \oplus b_7$ $b_1 = b_2 \oplus b_6$	$b_2 = b_4 \oplus b_5$ $b_2 = b_6 \oplus b_1$ $b_2 = b_3 \oplus b_7$	$b_3 = b_5 \oplus b_6$ $b_3 = b_7 \oplus b_2$ $b_3 = b_4 \oplus b_1$

Перечисленные в таблице 1.1 составные оценки называются *ортогональными* проверками, поскольку в них входят *несовпадающие символы.* Число ортогональных проверок N и кратность ошибок $q_{\text{исп}}$, исправляемых при мажоритарном декодировании, находятся в соотношении:

$$q_{\text{исп}} \le (N - 1)/2.$$

Код с порождающей матрицей (1.3) позволяет сформировать $N = 3$ ортогональных проверки и, соответственно, *исправлять однократные ошибки* в информационных символах *при значительном упрощении алгоритма декодирования.* Необходимо отметить, что правила формирования проверок могут иметь *циклические свойства*, что упрощает процедуру декодирования.

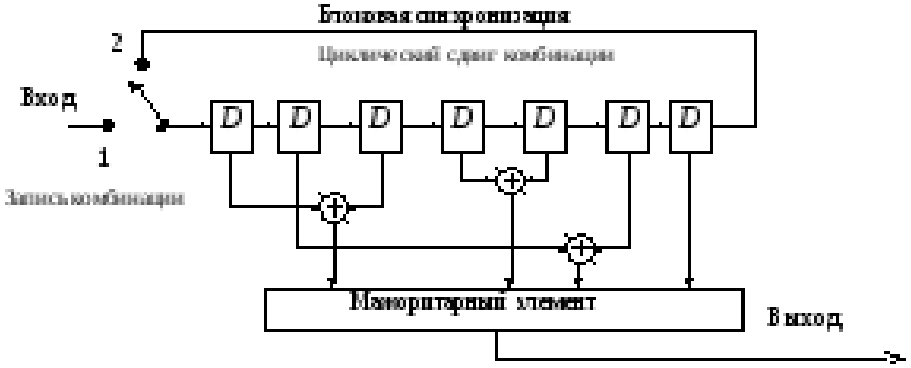


Рисунок 1.1 – Структура мажоритарного декодера (7,3)

Декодер состоит из регистра сдвигов, коммутатора на входе, управляемого системой блоковой синхронизации, схем формирования проверок и мажоритарного элемента.

Декодер работает следующим образом. Вначале коммутатор на входе устанавливается в положение «1» и декодируемая кодовая комбинация $\boldsymbol{b} = (b_1, b_2, b_3, b_4, b_5, b_6, b_7)$ вписывается в ячейки регистра сдвигов. При этом на входах мажоритарного элемента действуют как тривиальные, так и составные проверки, определяемые в таблице 1.1. Решение о передаваемом информационном символе b_1 считывается с выхода мажоритарного элемента. Затем коммутатор устанавливается в положение «2» и происходит сдвиг комбинации на один символ. На этом такте, в силу циклических свойств проверок, формируются проверки относительно второго информационного символа и решение об информационном символе b_2 считывается с выхода мажоритарного элемента. Далее процесс повторяется вплоть до получения на выходе символа b_3 [2].

2 Краткое описание пакета Simulink

2.1 Общая характеристика пакеты Simulink

Пакет **Simulink** распространяется в составе математического пакета **MatLab**. Пакет основан на графическом интерфейсе и является типичным средством визуально-ориентированного программирования. Пакет **Simulink** обладает обширной библиотекой готовых блоков с модифицируемыми параметрами для построения моделей рассматриваемых систем и наглядными средствами визуализации результатов моделирования.

2.2 Запуск и работа с пакетом Simulink

Для запуска системы **Simulink** необходимо выполнить запуск системы **MatLab**. После открытия командного окна системы **MatLab** нужно запустить систему **Simulink**. Существует три способа запуска системы **Simulink**:

- 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).

При применении двух первых способов открывается окно обозревателя библиотеки блоков (**Simulink Library Browser**). Если нам не требуется добавление новых блоков, а нужно лишь открыть уже готовую модель и провести моделирование, то следует воспользоваться третьим способом.

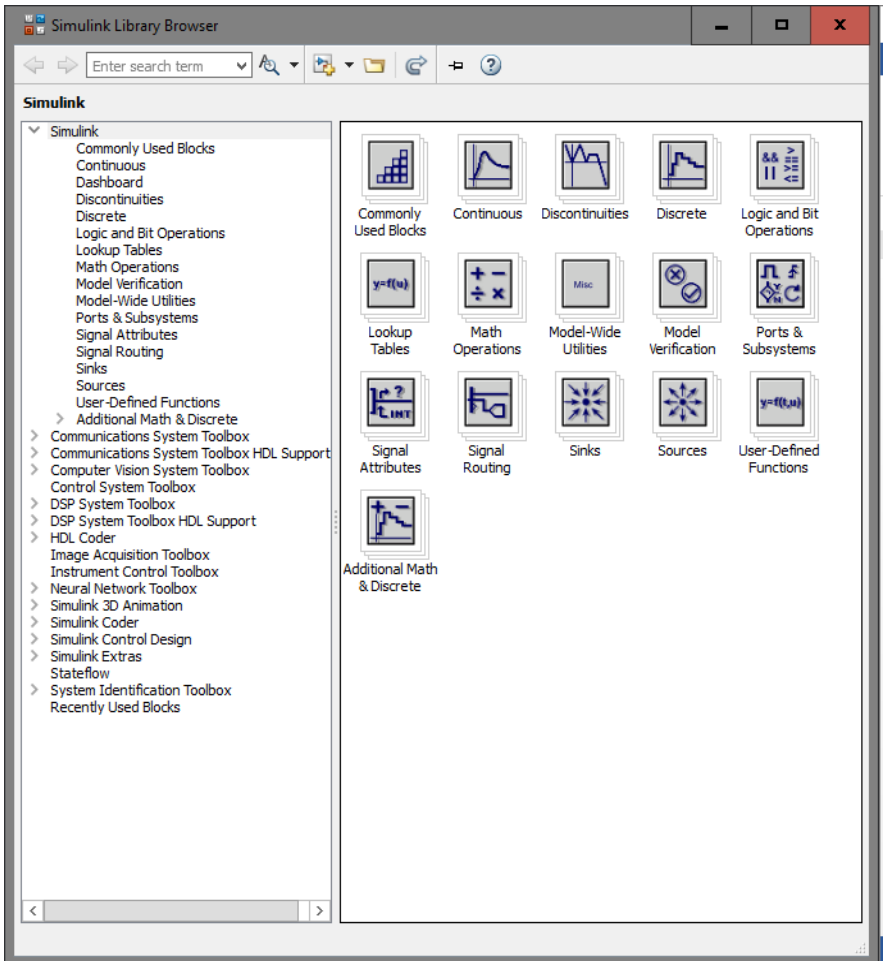


Рисунок 2.1 – Библиотека блоков **Simulink Library Browser**

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

- **Continuous**– блоки аналоговых элементов;
- **Dashboard** – панель приборов
- **Discontinuous** – блоки нелинейных элементов;
- **Discrete** – блоки дискретных элементов;
- **Logic and Bit Operations** – логические и битовые операции;
- **Lookup Tables** – блокитаблиц;
- **Math Operations** – блоки элементов, определяющие математические операции;

- **Model-Wide Utilities** – раздел дополнительных утилит;
- **Model Verification** – блоки проверки свойств сигнала;
- **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-DefinedFunction** – функции, определяемые пользователем;
- **Additional Math & Discrete** – дополнительная и дискретная математика.

Правила работы со списком разделов библиотеки: пиктограмма свернутого узла содержит символ «+»,а пиктограмма развернутого – символ «-».

Для того чтобы развернуть или свернуть узел, достаточно щелкнуть на его пиктограмме левой клавишей мыши.

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 приведёт к выходу из **Simulink.**

3 Описание лабораторных макетов

3.1 Кодек с использованием блочного кодера и мажоритарного декодера

Реализация функциональной модели кодека с использованием блочного кодера и мажоритарного декодера на примере кода Хэмминга в среде **Matlab Simulink** (7,4) представлена на рисунке 3.1:

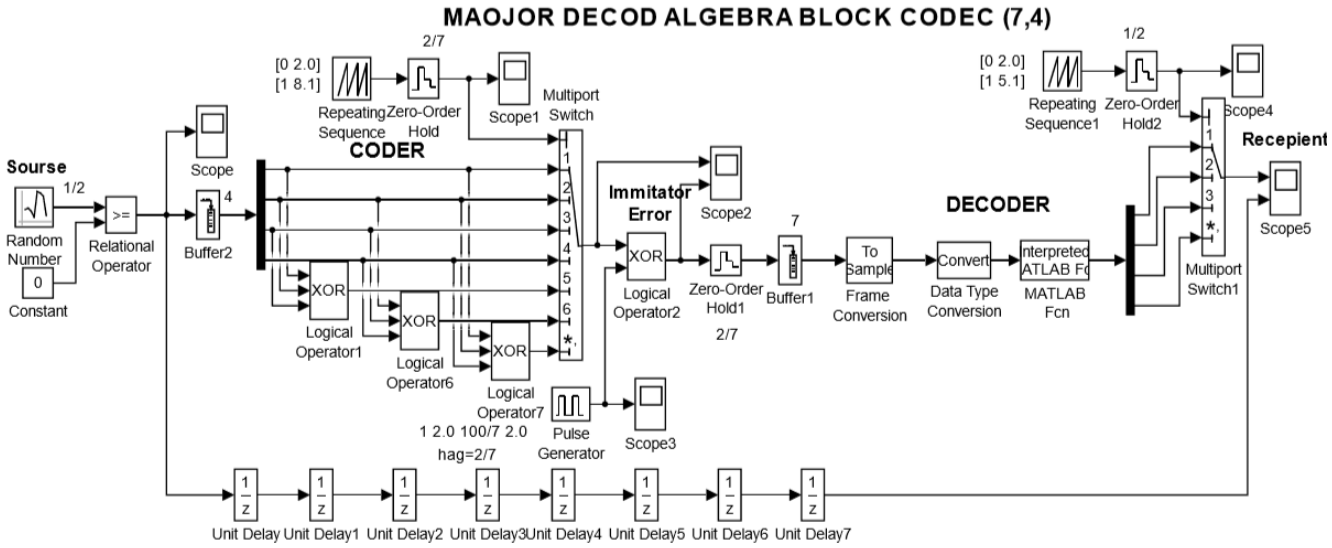


Рисунок 3.1 – Функциональная схема с использованием блочного кодера и мажоритарного декодера на примере кода (7,4)

Функциональная модель состоит из кодера (блочный кодер), имитатора ошибок и декодера (мажоритарный декодер).

При помощи блоков **Random Number** и **Relational Operator** формируем случайный поток с нормальным распределением нулей и единиц.

Подается четыре бита (информационный символ), созданный при помощи блока **Buffer**, получая на выходе **Multiport Switch** биты четности с увеличенной скоростью. Биты четности формируются при помощи блоков логического

оператора **XOR**. **Multiport Switch** увеличивает скорость передачи информационного потока.

Блоки **Repeating Sequence** и **Zero-Order Hold** управляют блоком **Multiport Switch**.

При помощи блока **Pulse Generator** и **XOR** создается имитатор одиночных ошибок.

Декодирование производится блоком **Matlab Fnc**, в котором содержится **Matlab**-функция мажоритарного декодирования кода Хэмминга (7,4) (рисунок 3.2), на выходе которого после блока **Multiport Switch** получаем изначальный информационный поток с уменьшенной скоростью.

Блоки **Unit Delay** создают искусственную задержку для сравнения результатов в блоке **Scope6**, т.е. задержки учитывают время необходимое для кодирования и декодирования потока данных.

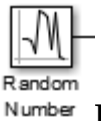
major_decod_7_4(1).m

```
function y=major_decod_7_4_1(x);
% function y=major_decod_7_4(x);
% функциЯ мажоритарного декодирования блочного кода 7.4
% x- входной массив - принятый кодовый вектор
% y- выходной массив - мажоритарно декодированный вектор данных

y=zeros(4,1);
z1=x(1);
z2=xor(x(2),xor(x(3),x(5)));
z3=xor(x(2),xor(x(4),x(7)));
z4=xor(x(4),xor(x(5),x(6)));
z5=xor(x(3),xor(x(6),x(7)));
y(1)=(z1+z2+z3+z4+z5); if y(1)>=2.5; y(1)=1; else y(1)=0; end;

z1=x(2);
z2=xor(x(1),xor(x(3),x(5)));
z3=xor(x(1),xor(x(4),x(7)));
z4=xor(x(3),xor(x(4),x(6)));
z5=xor(x(5),xor(x(6),x(7)));
y(2)=(z1+z2+z3+z4+z5); if y(2)>=2.5; y(2)=1; else y(2)=0; end;

z1=x(3);
z2=xor(x(1),xor(x(2),x(5)));
z3=xor(x(2),xor(x(4),x(6)));
z4=xor(x(4),xor(x(5),x(7)));
z5=xor(x(1),xor(x(6),x(7)));
y(3)=(z1+z2+z3+z4+z5); if y(3)>=2.5; y(3)=1; else y(3)=0; end;

z1=x(4);
z2=xor(x(2),xor(x(3),x(6)));
z3=xor(x(1),xor(x(2),x(7)));
z4=xor(x(1),xor(x(5),x(6)));
z5=xor(x(3),xor(x(5),x(7)));
y(4)=(z1+z2+z3+z4+z5); if y(4)>=2.5; y(4)=1; else y(4)=0; end;
```

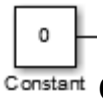
Рисунок 3.2 – Программный код мажоритарного декодирования для кода Хэмминга (7,4)

4 Описание используемых блоков библиотеки Simulink

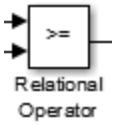
Ниже описаны основные блоки базовых разделов библиотеки Simulink, используемые в функциональной схеме блочного кодирования и мажоритарного декодирования кода (7,4).



Random Number – источник случайного сигнала с нормальным распределением. Назначение: формирование случайного сигнала с равномерным распределением уровня сигнала. Параметры блока: **Mean**– среднее значение сигнала; **Variance**– дисперсия; **Initia lseed** – начальное значение генератора случайного сигнала; **Sample time** – такт дискретности



Constant - источник постоянного сигнала задает постоянный по уровню сигнал. Параметры: Constant value - постоянная величина, Interpret vector parameters as 1-D – интерпретировать вектор параметров как одномерный (при установленном флажке). Значение константы может быть действительным или комплексным числом, вычисляемым выражением, вектором или матрицей.



Relational Operator - блок вычисления операции отношения. Блок сравнивает текущие значения входных сигналов. Параметры: тип операции отношения (выбирается из списка):

- = = - Тождественно равно.
- ~ = - Не равно.
- < - Меньше.
- < = - Меньше или равно.
- > = - Больше или равно.
- > - Больше.

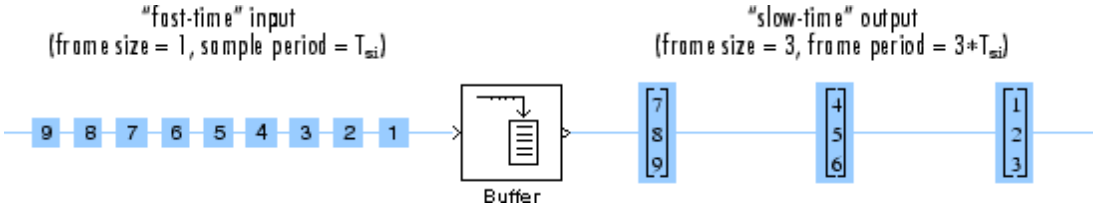
В операции отношения первым операндом является сигнал, подаваемый на первый (верхний) вход блока, а вторым операндом – сигнал, подаваемый на второй (нижний) вход. Выходным сигналом блока является **1**, если результат вычисления операции отношения есть **“ИСТИНА”** и **0,** если результат – **“ЛОЖЬ”**.

Входные сигналы блока могут быть скалярными, векторными или матричными. Если оба входных сигнала – векторы или матрицы, то блок выполняет поэлементную операцию сравнения, при этом размерность входных сигналов должна совпадать. Если один из входных сигналов – вектор или матрица, а другой входной сигнал – скаляр, то блок выполняет сравнение скалярного входного сигнала с каждым элементом массива. Размерность выходного сигнала, в этом случае, будет определяться размерностью векторного или матричного сигнала, подаваемого на один из входов.

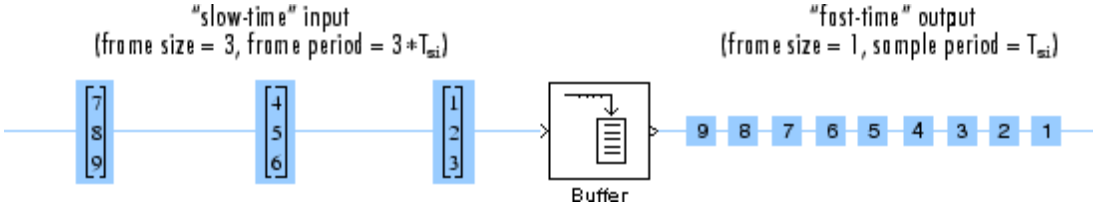
Для операций = = (тождественно равно) и ~ = (не равно) допускается использовать комплексные входные сигналы.

Входные сигналы также могут быть логического типа (**boolean**).

Buffer - буферный блок всегда выполняет обработку на основе кадра. Блок перераспределяет данные в каждом столбце ввода для получения результата с другим размером кадра. Буферизация сигнала на больший размер кадра дает выход с более низкой частотой кадров, чем вход. Например, рассмотрим следующую иллюстрацию для скалярного ввода.



Буферизация сигнала на меньший размер кадра дает выход с более высокой частотой кадров, чем вход. Например, рассмотрим следующую иллюстрацию скалярного вывода.



Блок координирует размер выходного кадра и частоту кадров неперекрывающихся буферов таким образом, чтобы период выборки сигнала был одинаковым как на входе, так и на выходе: Tso = Tsi.

Logical Operator - блок логических операций реализует одну из базовых логических операций. Параметры: вид реализуемой логической операции (выбирается из списка):

- AND – Логическое умножение (операция И).
- OR – Логическое сложение (операция ИЛИ).
- NAND – Операция И-НЕ.
- NOR – Операция ИЛИ-НЕ.
- XOR – Исключающее ИЛИ (операция сложения по модулю 2).
- NOT – Логическое отрицание (НЕ).
- Number of input ports – Количество входных портов.

Выходным сигналом блока является 1, если результат вычисления логической операции есть “ИСТИНА” и 0, если результат – “ЛОЖЬ”.

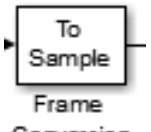
Входные сигналы блока могут быть скалярными, векторными или матричными. Если входные сигналы – векторы или матрицы, то блок выполняет поэлементную логическую операцию, при этом размерность входных сигналов должна совпадать. Если часть входных сигналов – векторы или матрицы, а

другая часть входных сигналов – скаляры, то блок выполняет логическую операцию для скалярных входных сигналов и каждого элемента векторных или матричных сигналов. Размерность выходного сигнала, в этом случае, будет определяться размерностью векторных или матричных входных сигналов.

При выполнении логической операции отрицания блок будет иметь лишь один входной порт.

Входные сигналы могут быть как действительного, так и логического типа (boolean).

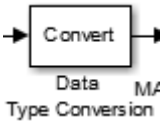
Zero-Order Hold - ‚лок экстраполятора нулевого порядка выполняет дискретизацию входного сигнала по времени. Параметры: Sample time – Величина шага дискретизации по времени. Блок фиксирует значение входного сигнала в начале интервала квантования и поддерживает на выходе это значение до окончания интервала квантования. Затем выходной сигнал изменяется скачком до величины входного сигнала на следующем шаге квантования.



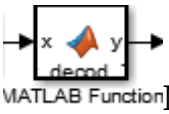
Frame Conversion - блок преобразования кадров передает входной сигнал на выход и устанавливает режим выходной выборки на значение параметра **Sampling mode of output signal**, который может быть основан как на **Frame-based** , так и на **Sample-based**. Режим выходной выборки также может быть унаследован от сигнала на входном порту **Ref** (reference), который вы делаете видимым, выбрав **Inherit output sampling <Ref> input port**.

Блок **Frame Conversion** не вносит никаких изменений во входной сигнал, отличный от режима выборки. В частности, блок не перестраивает или не

изменяет размер двухмерных (2-D) входов. Поскольку одномерные (1-D) векторы не могут быть основаны на кадрах, когда вход является 1-D вектором длины-M, а блок находится в режиме на основе кадра, выход представляет собой матрицу M-на-1 на основе кадра, то есть один канал. Параметры: **Inherit output sampling mode from <Ref> input port** - включается порт Ref, **Sampling mode of output signal** – выбирается режим выборки выходного сигнала (Frame-based or Sample-based).



Data Type Conversion - преобразователь типа сигнала. Назначение: преобразует тип входного сигнала. Параметры блока: **Data Type** - тип данных выходного сигнала. **Saturate On Integer Overflow** - подавлять переполнение целого.



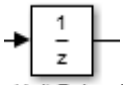
Matlab Function – блок задания M – функции. Назначение: задает выражение в стиле языка программирования MATLAB. Параметры блока: MATLAB Function – выражение на языке MATLAB. **Output Dimensions** – размерность выходного сигнала. **Output Signal Type** – тип выходного сигнала. Выбирается из списка: **real** - действительный сигнал; complex- комплексный сигнал; **auto**- автоматическое определение типа сигнала. **Collapse 2-Dresultsto 1-D**-преобразование двумерного выходного сигнала в одномерный.

Repeating Sequence – данный блок формирует периодический сигнал. Параметры: **Time values** - вектор значений модельного времени, **Output values** - вектор значений сигнала для моментов времени заданных вектором **Time values**. Блок выполняет линейную интерполяцию выходного сигнала для моментов времени не совпадающих со значениями заданными вектором **Time values.**

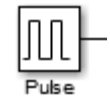
Multiport Switch – блок многоходового переключателя выполняет переключение входных сигналов по сигналу управления, задающему номер активного входного порта. Параметры: **Number of inputs** – количество входов. Блок многовходового переключателя **Multiport Switch**, пропускает на выход сигнал с того входного порта, номер которого равен текущему значению управляющего сигнала. Если управляющий сигнал не является сигналом целого типа, то блок **Multiport Switch** производит отбрасывание дробной части числа, при этом в командном окне **Matlab** появляется предупреждающее сообщение.

Scope – блок осциллографа. Назначение: построение графиков исследуемых сигналов как функций времени. Открытие окна осциллографа производится двойным щелчком **ЛКМ** на пиктограмме блока. 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 – демультиплексор. Назначение: разделяет входной векторный сигнал на отдельные составляющие. Параметры блока: **Number Of Outputs**-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. **Bus Selection Mode** - режим разделения векторных сигналов по шине.



Unit Delay - блок единичной дискретной задержки. Назначение: выполняет задержку входного сигнала на один шаг модельного времени. Параметры: Initial condition – начальное значение для выходного сигнала. Sample time – шаг модельного времени. Входной сигнал блока может быть как скалярным, так и векторным. При векторном входном сигнале задержка выполняется для каждого элемента вектора. Блок поддерживает работу с комплексными и действительными сигналами.



Pulse Generator - Источник импульсного сигнала **Pulse Generator** Назначение: формирование прямоугольных импульсов.

Параметры:

Pulse Type – Способ формирования сигнала.

Может принимать два значения: Time-based (по текущему времени), Sample-based (по величине модельного времени и количеству расчетных шагов).

Amplitude — Амплитуда.

Period — Период. Задается в секундах для Time-based Pulse Type или в шагах модельного времени для Sample-based Pulse Type.

Pulse width — Ширина импульсов. Задается в в % по отношению к периоду для Time-based Pulse Type или в шагах модельного времени для Sample-based Pulse Type.

Phase delay — Фазовая задержка. Задается в секундах для Time-based Pulse Type или в шагах модельного времени для Sample-based Pulse Type. Sample time — Шаг модельного времени. Задается для Sample-based Pulse Type.

5 Экспериментальная часть

Моделирование в среде Simulink ведется во временной области с использованием относительных масштабов по времени и частоте. Вариант простановки параметров блоков модели указан на функциональной модели кодека (рисунок 3.1).

Задание:

1. Для кода Хэмминга (7,4) провести оценку информационных символов по аналогии с приведенным примером п.1.1.

2. Написать программный код в Matlab, описывающий работу мажоритарного декодирования для кода Хэмминга (7,4) на основе полученных данных из задания 1.

3. Собрать функциональную модель кодека с мажоритарным декодированием в соответствии рисунком 3.1.

4. Проверить работоспособность модели: какие ошибки обнаруживает и исправляет декодер.

5. Написать отчет.

6 Контрольные вопросы

1. Какова основная идея помехоустойчивого кодирования?

2. В чем особенность линейного блочного кодирования?

3. Основная идея мажоритарного декодирования.

4. Опишите принцип работы мажоритарного декодирования.

5. Приведите упрощенную структурную схему работы кодека с мажоритарным декодированием.

6. Что такое составные и тривиальные оценки?

7. Что такое энергетический выигрыш кодирования?

8. Как понимать выражение «кодирование расширяет возможности компромисса между полосой и энергетикой канала».

9. Можно ли неограниченно наращивать избыточность?

10. Понятия порождающей и проверочной матриц.

Список использованных источников

1. Элементы теории кодирования информации [Электронный ресурс] – Режим доступа: http://www.urtt.ru/bib/dataindex/dm/glava_5~.htm (дата обращения 10.03.2018).

2. Мажоритарное декодирование блоковых кодов [Электронный ресурс] – Режим доступа: https://studfiles.net/preview/5157419/page:9/ (дата обращения 15.10.2017).

3. Черных И.В. "Simulink: Инструмент моделирования динамических систем [Электронный ресурс] – Режим доступа: http://matlab.exponenta.ru/simulink/book1/index.php (дата обращения 15.03.2018).

4. Справочник Матлаб [Электронный ресурс] – Режим доступа: https://www.mathworks.com/help/matlab/ (дата обращения 15.03.2018).