

Министерство науки и высшего образования Российской Федерации

Томский государственный университет систем
управления и радиоэлектроники

Кологривов В.А.

Родзик М.В.

Всяких К.А.

**МОДЕЛЬНОЕ ПРОЕКТИРОВАНИЕ ЭЛЕМЕНТОВ ЦИФРОВОГО КАНАЛА
ПЕРЕДАЧИ ДАННЫХ**

Методические указания по выполнению курсовой работы по дисциплине «Общая теория связи» в среде функционального моделирования Simulink системы MatLab для студентов радиотехнических специальностей

Томск 2024

УДК 621.391.6

ББК 32.811

К 61

Рецензент:

Мещеряков А.А., доцент кафедры радиотехнических систем ТУСУР, канд. техн. наук

Кологривов В.А., Родзик М.В., Всяких К.А.

К 61 Модельное проектирование элементов цифрового канала передачи данных: методические указания по выполнению курсовой работы для студентов радиотехнических специальностей / Кологривов В.А., Родзик М.В., Всяких К.А. – Томск: Томск. гос. ун-т систем упр. и радиоэлектроники, 2024 – 94 с.

Настоящие методические указания по курсовой работе составлены с учетом требований федерального государственного образовательного стандарта высшего образования (ФГОС ВО).

Курсовая работа «Модельное проектирование элементов цифрового канала передачи данных» с использованием пакета функционального моделирования **Simulink** системы для инженерных и научных расчетов **MatLab**.

В описании сформулирована цель курсового проекта, приведены краткие теоретические сведения о возможностях функциональной среды **Simulink** системы **MatLab**, дискретизации сигнала по Котельникову, аналого-цифровом и цифро-аналоговом преобразованиях сигнала, скремблировании, описание спектров, низкочастотных и полосовых фильтров, модели канала распространения радиосигнала, методике исследования помехоустойчивости модема, энергетической и спектральной эффективностях, цифровых модуляциях, методах приема (демодуляции), помехоустойчивом кодировании и способах декодирования, совместное использование помехоустойчивого кода в модеме с целью получения энергетического выигрыша от кодирования (**ЭВК**).

Одобрено на заседании каф. РТС протокол №7 от 17.04.2024 г.

УДК 621.391.6

ББК 32.811

© Кологривов В.А., Родзик М.В., Всяких К.А., 2024

© Томск. гос. ун-т систем упр. и радиоэлектроники, 2024

ОГЛАВЛЕНИЕ

Введение	5
1 ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ, ИЗЛОЖЕНИЮ И ОФОРМЛЕНИЮ КУРСОВОЙ РАБОТЫ	7
1.1 Общие требования.....	7
1.2 Структура пояснительной записки.....	8
2 ВОЗМОЖНОСТИ И ОСОБЕННОСТИ ФУНКЦИОНАЛЬНОЙ СРЕДЫ SIMULINK СИСТЕМЫ MATLAB	10
2.1 Запуск и работа с пакетом Simulink	10
3 ДИСКРЕТИЗАЦИЯ ПО КОТЕЛЬНИКОВУ, АНАЛОГО-ЦИФРОВОЕ И ЦИФРО-АНАЛОГОВОЕ ПРЕОБРАЗОВАНИЯ И СКРЕМБЛИРОВАНИЕ СИГНАЛА.....	12
3.1 Относительность моделирования по времени и частоте в Simulink	12
3.2 Дискретизация по теореме Котельникова. АЦП и ЦАП	12
3.3 Модель аддитивного скремблера.....	18
4 ОПИСАНИЕ ЦИФРОВЫХ СИГНАЛОВ, СПЕКТРОВ, НИЗКОЧАСТОТНЫХ И ПОЛОСОВЫХ ФИЛЬТРОВ.....	25
4.1 Описание модели источника цифровой информационной последовательности... ..	25
4.2 Описание спектров информационной последовательности и модулированного радиосигнала	29
4.3 Полосовые и низкочастотные фильтры	30
4.4 Измерение помехоустойчивости	32
4.5 Способы приема (демодуляции)	36
5 ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ ЦИФРОВЫХ МОДУЛЯЦИЙ	38
5.1 Описание моделей модемов на основе BPSK модуляции.....	38
5.2 Описание моделей модемов на основе BFSK модуляции.....	42
5.3 Описание моделей модемов на основе QPSK модуляции.....	47
6 ОПИСАНИЕ ТЕОРИИ И МОДЕЛЕЙ ПОМЕХОУСТОЙЧЕВЫХ КОДЕКОВ И БЛОЧНЫХ КОДОВ.....	58
6.1 Синдромное декодирование	58
6.2 Систематическое циклическое кодирование	61
6.3 Мажоритарное декодирование блочных кодов	67
7 ИССЛЕДОВАНИЕ КОДЕМОВ.....	70
7.1 QBPSK модем	70
7.2 Пример QBPSK кодема.....	72
7.3 BFSK модем	74
7.4 Пример BFSK кодема.....	76
7.5 QPSK модем.....	78
7.6 Пример QPSK кодема.....	80

Заключение	84
Список использованных источников.....	85
ПРИЛОЖЕНИЕ А (обязательное) Функциональная модель квадратурной реализации BPSK	86
ПРИЛОЖЕНИЕ Б (обязательное) Функциональная модель BFSK модема.....	88
ПРИЛОЖЕНИЕ В (обязательное) Функциональная схема исследования QPSK модема	90
ПРИЛОЖЕНИЕ Г (обязательное) Функциональная модель QBPSK модема с синдромным декодером.....	92
ПРИЛОЖЕНИЕ Д (обязательное) Функциональная модель BFSK модема с синдромным декодером.....	93
ПРИЛОЖЕНИЕ Е (обязательное) Функциональная модель схема QPSK модема с мажоритарным декодером.....	94

Введение

Общая теория связи (**ОТС**) – это теория о способах, законах и методах коммуницирования между абонентами, с помощью определенных устройств, и о способах взаимодействия двух и более устройств, а также о том, как обрабатываются передаваемые сигналы (цифровые или аналоговые), и как осуществляется связь.

Задачей дисциплины **ОТС** является, ознакомление с основными принципами и методами современной статистической теории обработки сигналов, с основными технологиями электрической связи, например, с технологиями и системами беспроводного доступа, принципами их функционирования и методами оценки пропускной способности; влиянием многолучёвости каналов распространения на пропускную способность беспроводных каналов; используемыми методами модуляции и помехоустойчивого кодирования; использованием пространственно-временных методов передачи; способами выравнивания характеристик канала; технологией модуляции на нескольких несущих; широкополосными системами передачи; технологиями мультиплексирования каналов; сотовой организацией сетей связи.

Цифровая связь — область техники, связанная с передачей цифровых данных на расстояние. В настоящее время цифровая связь повсеместно используется также и для передачи аналоговых непрерывных по уровню и времени, (например, речь, изображение) сигналов, которые для этой цели оцифровываются (дискретизируются). Такое преобразование всегда связано с потерями, т.е. аналоговый сигнал представляется в цифровом виде с некоторой неточностью.

Современные системы цифровой связи используют кабельные (в том числе волоконно-оптические), спутниковые, радиорелейные и другие линии и каналы связи, в том числе и аналоговые.

Цели и задачи курсовой работы по дисциплине Общая теория связи (ОТС) по теме: Модельное проектирование элементов цифрового канала передачи данных (в среде *Simulink* системы *MatLab*).

Целью курсового проектирования является, создание более систематизированного представления о работе модема системы связи на конкретных примерах, используя функциональную среду *Simulink* системы *MatLab* и в объеме, предоставленном индивидуальным техническим заданием (**ТЗ**). Ознакомиться с содержанием, составом и структурой связного модема, включая особенности модельной реализации. Составить структурную схему модема с использованием **ТЗ** (исходные данные которого, тип передаваемого сообщения, разрядность **АЦП** и **ЦАП**, тип скремблирования, тип модуляции сигнала, тип демодуляции, тип кодирования, тип декодирования). Получить опыт работы с учебной литературой, закрепить навыки оформления результатов курсовой работы в виде структурированной пояснительной записки, выполненной по стандарту ОС ТУСУРа.

Задачи курсовой работы:

1. Ознакомление с возможностями и особенностями функциональной среды *Simulink* системы *MatLab*.

При моделировании (построении функциональных моделей) и интерпретации результатов моделирования целесообразно использовать принцип относительности вычислений во временной и частотной областях.

2. Модельная иллюстрация дискретизации и восстановления аналогового сигнала по Котельникову, а также преобразований **АЦП** и **ЦАП**.

Обратить внимание на сравнение ширины спектров исходного аналогового сигнала и полученного цифрового сигнала.

3. Модельная реализация аддитивного скремблера на основе регистра сдвига с обратными связями по структуре примитивного полинома соответствующего порядка.

Обратить внимание на способ построения сквозного разностного уравнения регистра сдвига для простого рекуррентного вычисления, включая табличный, формируемой псевдослучайной последовательности.

5. Модельная реализация модуляторов разновидностей частотной и фазовой манипуляции.

Обратить внимание на использование для контроля измерительных блоков построения гистограммы плотности распределения, анализатора спектра для отображения спектральной плотности мощности и блока вычисления автокорреляционной функции потока данных.

6. Модельная реализация канала распространения радиосигнала, позволяющего, в частности, менять соотношение сигнал/шум при исследовании помехоустойчивости модемов.

Обратить внимание на методику модельного исследования помехоустойчивости, подсистем детектора ошибок и измерителя мощности сигнала и шумов.

7. Модельная реализация демодуляторов (приемников) на основе принципа синхронного детектирования (прямого преобразования спектра) либо корреляционного приема.

Обратить внимание на методику настройки параметров полосовых фильтров (**ПФ**) и **ФНЧ** и модельного вычисления функции корреляции, опорного и принятого сигналов.

8. Модельная реализация простых помехоустойчивых групповых и блочных кодов, включая циклические.

Обратить внимание на модели последующего декодирования синдромным, квазисиндромным на основе восстановления информационных битов по принятым избыточным, квазисиндромным на основе повторного кодирования, мажоритарным методами гарантирующих обнаружение и исправление однократных ошибок.

9. Модельная реализация включения помехоустойчивых кодеков в модем.

Обратить внимание на исследование зависимости числа возникающих в канале и остающихся после декодера ошибок от отношения сигнал/шум и энергетического выигрыша кодирования (**ЭВК**).

1 ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ, ИЗЛОЖЕНИЮ И ОФОРМЛЕНИЮ КУРСОВОЙ РАБОТЫ

Курсовая работа выполняется по индивидуальному техническому заданию (ТЗ) с использованием учебной литературы, методических пособий и консультаций преподавателя. Техническое задание содержит пункты как общего плана, так и частного, опирающиеся на конкретные исходные данные. В процессе выполнения работы необходимо правильно организовать регулярную работу над заданием, проявлять максимум инициативы и самостоятельности для решения поставленных задач.

Пояснительная записка должна быть оформлена в соответствии с образовательным стандартом вуза «ОС ТУСУР 01-2021» [1].

1.1 Общие требования

Работа должна быть выполнена печатным способом с использованием компьютера и принтера на одной стороне листа белой бумаги одного сорта формата А4 (210 × 297 мм).

При создании текстового файла работы устанавливаются следующие размеры полей: левое – 30 мм, правое – 15 мм, верхнее и нижнее – 20 мм. Абзацный отступ должен быть равен 1,25 см. Выравнивание текста производится по ширине страницы.

Тип шрифта для всего текста работы – Times New Roman черного цвета размером 14 пт. Междустрочный интервал – 1,5 строки.

Страницы работы следует нумеровать арабскими цифрами. Номер страницы проставляется в центре нижнего поля листа (страницы) без точки. Первым листом является титульный лист. На титульном листе номер не проставляется.

Выделение в тексте необходимых фрагментов производится с помощью шрифта иного начертания, чем шрифт основного текста, но того же кегля и гарнитуры (например, прямой полужирный шрифт или полужирный курсив). Выделение фрагментов текста путем подчеркивания не допускается.

Заголовки должны быть выделены полужирным шрифтом. Заголовки разных уровней должны отличаться по оформлению (например, заголовки первого уровня (наименования разделов/глав – ПРОПИСНЫЕ БУКВЫ, набранные прямым полужирным шрифтом; второго уровня (наименование подразделов/параграфов) – строчные буквы, набранные прямым полужирным шрифтом. Заголовки следует размещать посередине страницы с прописной буквы без точки в конце, не подчеркивая. В нумеруемых разделах перед заголовком помещают номер соответствующего раздела или подраздела. Переносы слов в заголовках не допускаются. Если заголовок состоит из двух предложений, их разделяют точкой.

Рисунок должен иметь тематическое название, а также может иметь пояснительные данные (подрисуночный текст). Слово «рисунок», его номер и тематическое наименование помещают ниже изображения и пояснительных данных симметрично иллюстрации. Сам рисунок (иллюстрация) располагается по центру страницы без абзацного отступа. Если наименование рисунка состоит из нескольких строк, то его следует записывать через одинарный междустрочный интервал. На все рисунки должны быть ссылки.

Формулы следует выделять из текста в отдельную строку и оформлять в редакторе формул. Значения символов и числовых коэффициентов, входящих в формулу, должны быть приведены непосредственно под формулой. Значение каждого символа дают с новой строки в той последовательности, в какой они приведены в формуле. Первая строка расшифровки должна начинаться со слова «где» без двоеточия после него. Формулы, следующие одна за другой и не разделенные текстом, отделяют запятой. Формулы, на которые имеются ссылки в тексте работы, должны быть пронумерованы в пределах раздела (приложения) арабскими цифрами. Номер формулы должен состоять из номера раздела и порядкового номера

формулы, разделенных точкой, например, «(1.4)». Номер указывают с правой стороны листа на уровне формулы в круглых скобках.

1.2 Структура пояснительной записки

Объем пояснительной записки подготавливаемого студентом в процессе выполнения курсового проектирования составляет приблизительно 20-25 страниц машинописного текста формата А4.

В текстовый документ последовательно включаются следующие части:

- титульный лист,
- реферат,
- задание,
- список условных сокращений и обозначений (при необходимости),
- содержание,
- введение,
- основная часть,
- заключение,
- список использованных источников,
- приложения.

Реферат.

Реферат размещается на отдельной странице. Реферат должен содержать:

- сведения о количестве страниц, иллюстраций, таблиц, использованных источников, приложений, листов графического материала;

- ключевые слова,
- текст реферата.

Текст реферата должен отражать:

- объект разработки или исследования;
- цель работы; - назначение работы и область применения;
- использованное программное обеспечение для разработки;
- полученные результаты и их новизну;
- дополнительные сведения.

Оглавление.

Оглавление содержит рубрикацию и наименование разделов отчета и должно отражать все материалы, представленной к защите работы.

Введение.

В разделе «Введение» указываются цели и задачи работы, ее назначение и область применения. Указывается значение работы для науки (техники) и народного хозяйства.

Основная часть.

В основной части отражается работа студента по выполнению технического задания.

Основная часть, как правило, содержит следующие разделы:

- относительность моделирования по времени и частоте в функциональной среде Simulink системы MatLab. Элементы модельных исследований;
- дискретизация и аналого-цифровое преобразование аналогового сигнала;
- моделирование источника информационного потока данных;
- модель аддитивного скремблера на основе регистра сдвига с обратными связями;
- установка параметров полосовой и низкочастотной фильтрации;
- заданный вид частотной либо фазовой модуляции;
- демодуляция путем прямого преобразования спектра радиосигнала или на основе корреляционного приема;
- заданный вид помехоустойчивого блочного кодера и декодера;

- включение помехоустойчивого кодека в модем и модельное исследование энергетического выигрыша кодирования.

Заключение.

Заключение должно содержать краткие выводы по наиболее важным результатам выполненной работы.

Список использованных источников.

В разделе «Список использованных источников» включаются все источники, использованные студентом в процессе выполнения работы. В тексте обязательны ссылки на все использованные источники. Список источников располагается в порядке ссылок.

Приложения.

В приложения рекомендуется выносить материалы иллюстративного и вспомогательного характера, например, для модели в тексте изложения использовать мелкий масштаб, а в приложение (по ссылке) выносить модель в более крупном масштабе на отдельный лист. На все приложения в тексте работы должны быть даны ссылки. Приложения располагают в порядке их упоминания в тексте работы.

Каждое приложение следует размещать с нового листа (страницы) с указанием наверху посередине страницы слова «Приложение» и его обозначение, а под ним заголовок, который записывают с прописной буквы отдельной строкой и без точки в конце. Рекомендуется в скобках над заголовком приложения указывать – «обязательное» (если его выполнение предусмотрено заданием, *ТЗ*) или «справочное».


Приложения должны иметь общую с остальной частью работы сквозную нумерацию страниц. Все приложения должны быть перечислены в оглавлении работы с указанием их обозначений, статуса и наименования.

2 ВОЗМОЖНОСТИ И ОСОБЕННОСТИ ФУНКЦИОНАЛЬНОЙ СРЕДЫ SIMULINK СИСТЕМЫ MATLAB


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

2.1 Запуск и работа с пакетом Simulink

Для запуска пакета *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.

Для создания модели в новом файле нужно открыть окно обозревателя библиотеки блоков (*Simulink Library Browser* ) (рисунок 2.1).

На рисунке 2.1 выведена библиотека системы *Simulink* (в левой части окна) и показаны ее разделы (в правой части окна), так же для нахождения определенного блока можно воспользоваться поисковой строкой *Enter search term* (в левой верхней 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 маршрутизации сигналов;
- *Signal Routing* – блоки маршрутизации сигналов;
- *Sinks* – блоки приема и отображения сигналов;
- *Sources* – блоки источников сигнала;
- *User-Defined Function* – функции, определяемые пользователем.

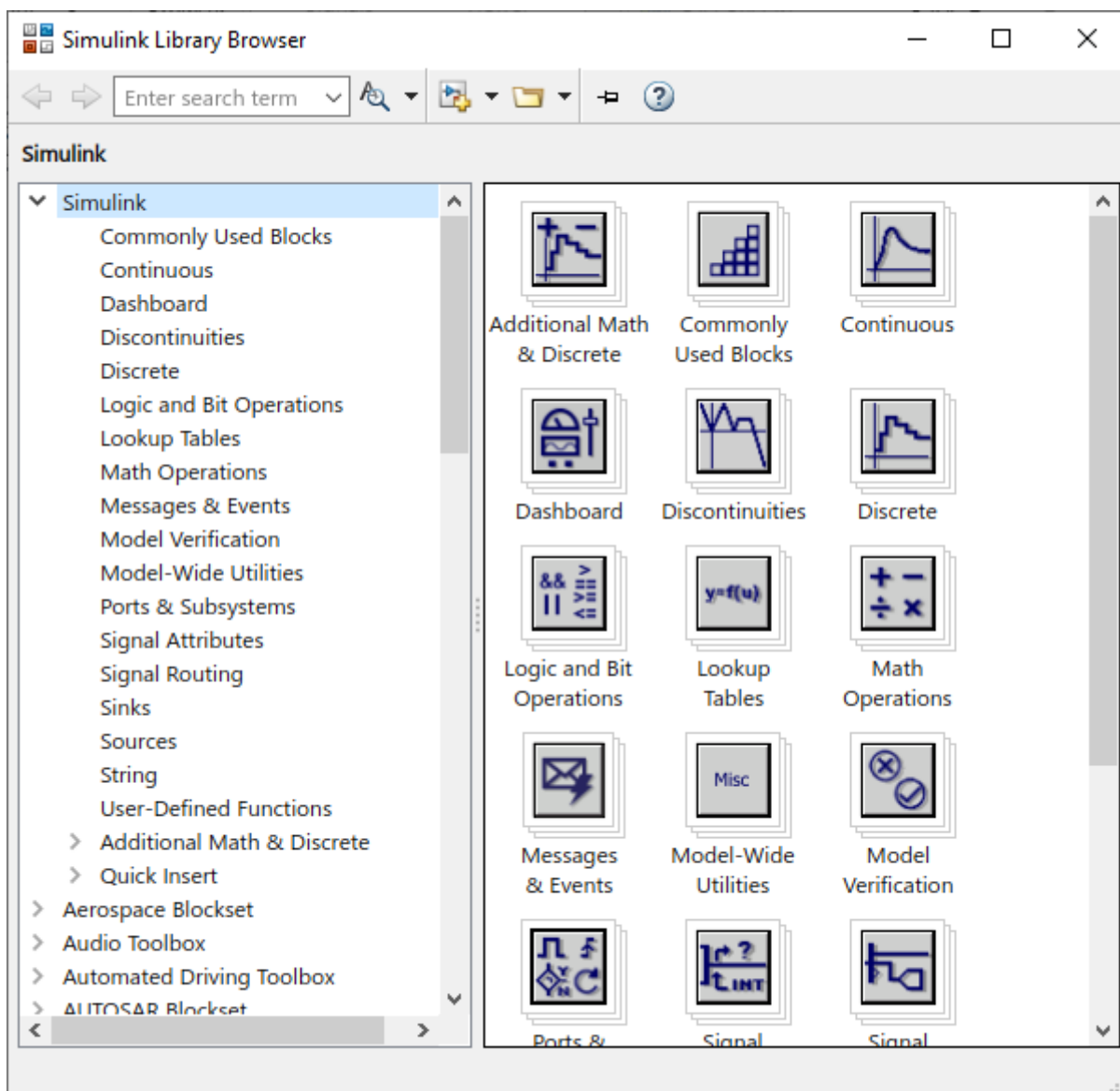


Рисунок 2.1 – Библиотека блоков **Simulink Library Browser**

3 ДИСКРЕТИЗАЦИЯ ПО КОТЕЛЬНИКОВУ, АНАЛОГО-ЦИФРОВОЕ И ЦИФРО-АНАЛОГОВОЕ ПРЕОБРАЗОВАНИЯ И СКРЕМБЛИРОВАНИЕ СИГНАЛА

3.1 Относительность моделирования по времени и частоте в *Simulink*

При моделировании (построении функциональных моделей) и интерпретации результатов моделирования, целесообразно использовать принцип относительности вычислений во временной и частотной областях.

Так если длительность бита последовательности составляет одну компьютерную секунду $\tau_b = 1$, то ширина основного лепестка спектра составляет порядка $\Delta\Omega = 2\pi/\tau_b$ радиан в секунду или $\Delta F = 1$ Гц, если длительность бита принять за одну наносекунду, то ширина основного лепестка спектра соответствует одному ГигаГерцу и т.д.

Процесс расчета при моделировании выполняется во временной области. Самый медленный процесс при моделировании — это информационные последовательности. Длительности битов берут в районе единицы модельного времени, например, $1/2, 1, 2$. При модуляции информационные последовательности являются модулирующими, т.е. во время длительности бита должно уложиться значимое число периодов несущего колебания ω_0 . На практике это число периодов составляет тысячи и миллионы. При моделировании в **Simulink** такое не допустимо, поскольку для качественного отображения несущего гармонического колебания необходимо **50-100** точек, всего колебаний за бит пусть будет **1000**, а для исследования помехоустойчивости приходится работать с последовательностями длиной **1000-10000** битов. Итого получаем колоссальное количество требуемых вычислений, с которыми не справится ни один современный компьютер. В связи с этим используется минимально возможное число периодов несущего колебания за время бита, при котором ещё не теряется смысл модуляции, скажем **8-10** периодов несущей ω_0 . При этом время счета информационных последовательностей длиной **10000** битов выливается в десятки минут компьютерного времени.

Для примера приведем расчет параметров несущего колебания. Пусть длительность бита составляет $\tau_b = 1$, а несущее колебание укладывается в эту длительность **8** раз. Тогда $T_0 = \tau_b/8 = 1/8$, т.е. $f_0 = 1/T_0 = 8$. Тогда круговая частота несущего колебания составляет $\omega_0 = 2\pi f_0 = 16\pi$ радиан. Частота кратная π берется исключительно для удобства. В нашем примере несущее колебание укладывается за время длительности бита равно **8** раз.

3.2 Дискретизация по теореме Котельникова. АЦП и ЦАП

В **ОТС** рассматриваются случаи передачи полезного сигнала по цифровому каналу, но в большинстве устройств на их вход подаётся аналоговый сигнал [2].

Для передачи аналогового сигнала по цифровому каналу необходимо осуществить преобразование в три этапа, для преобразования аналогового сигнала в цифровой:

- 1) дискретизация по времени;
- 2) квантование по уровню;
- 3) цифровое представление уровней.

Рисунок 3.1 иллюстрирует преобразование аналогового сигнала в цифровой.

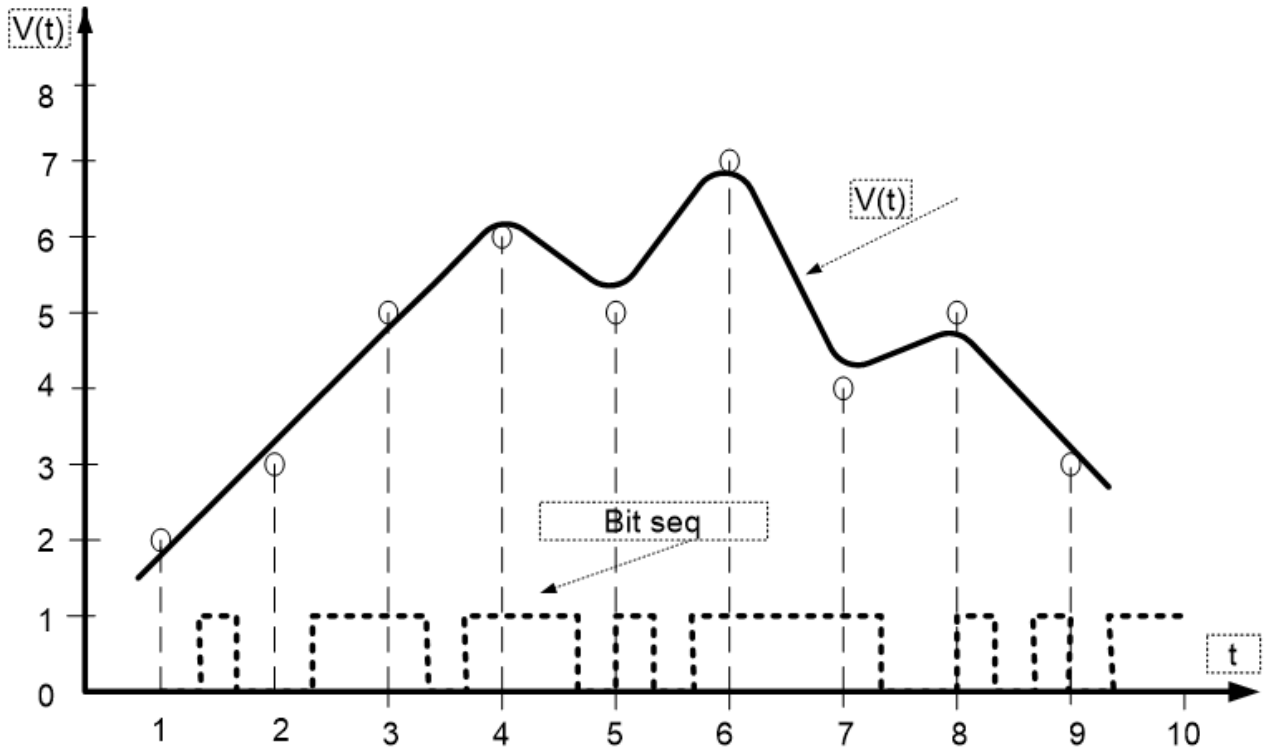


Рисунок 3.1 – Иллюстрация преобразования аналогового сигнала в цифровой

На рисунке 3.1 изображен фрагмент аналогового сигнала $v(t)$, к которому применили дискретизацию по времени, квантование по уровню, а также представление квантовых уровней трех битовым цифровым кодом.

По оси времени t отложены дискретные моменты времени $k = 1, 2, 3, \dots$, удовлетворяющие условию Котельникова $T_d = \Delta t \leq 1/2f_v = \pi/\omega_v$, где $T_d = \Delta t$ - период дискретизации во времени, f_v - линейная верхняя частота спектра аналогового сигнала, ω_v - круговая верхняя частота спектра аналогового сигнала. Если использовать понятие частоты дискретизации, то условие запишется $F_d = 1/T_d \geq 2f_v = \omega_v/\pi$. При этом полагается, что непрерывное время дискретизируется как $t = kT_d$. На практике верхняя частота спектра соответствует частоте, при которой энергия сигнала спадает, например, до уровня $E \leq 1\%$.

Математически дискретизация описывается умножением непрерывного сигнала $v(t)$ на периодическую последовательность дискретизирующих импульсов $\delta_t = \sum_k \delta(t - kT_d)$. Отклик дискретизатора $x_k = x(t) \sum_k \delta(t - kT_d)$ соответствует последовательности квантованных по уровню узких импульсов τ_0 с длительностью много меньше периода дискретизации, например, $\tau_0 = (0.05 \div 0.1)T_d$. Если считать, что ширина основного лепестка спектра $\Delta\omega = 2\pi/\tau_0$ примерно равна требуемой полосе пропускания дискретной системы, то при $T_d = \pi/\omega_v$, получаем $\Delta\omega = (40 \div 20)\omega_v$ неприемлемо большую величину.

По оси абсцисс $v(t)$ отложены уровни квантования $n = 0, \dots, L - 1$ с шагом квантования Δq и порогом квантования $h^{(n)}$, где L - число уровней квантования. Реальные дискретные отсчеты сигнала округляются до ближайшего уровня.

Если учесть, что с вероятностью 0.997 гауссовский случайный процесс находится в диапазоне $6\sigma_v = 6\sqrt{P_v}$, σ_v - дисперсия, и в этом диапазоне разместить $(L - 2)$ уровней, а два уровня отвести на области вне диапазона, т.е. $v < v_{min}$ и $v > v_{max}$, то шаг квантования можно определить из выражения $\Delta q = 6\sqrt{P_v}/(L - 2)$, а порог квантования –

$$h^{(n)} = 3\sqrt{P_v} \left(\frac{n-1}{0.5L-1} - 1 \right), \quad n = 1, \dots, L-1,$$

где $h^{(0)} = -\infty$, $h^{(L)} = +\infty$ - крайние пороги квантования равны, P_v - мощность (дисперсия) сигнала.

В простейшем случае уровни квантования определяются соотношениями:

$$v^{(n)} = \frac{h^{(n+1)} + h^{(n)}}{2} = v^{(0)} + n\Delta q,$$

$$v^{(0)} = -\frac{\Delta q}{2}(L-1), \quad n = 0, \dots, L-1.$$

Функциональная модель аналого-цифрового (АЦП) и цифро-аналоговое (ЦАП) преобразования сигнала, представлена на рисунке 3.2.

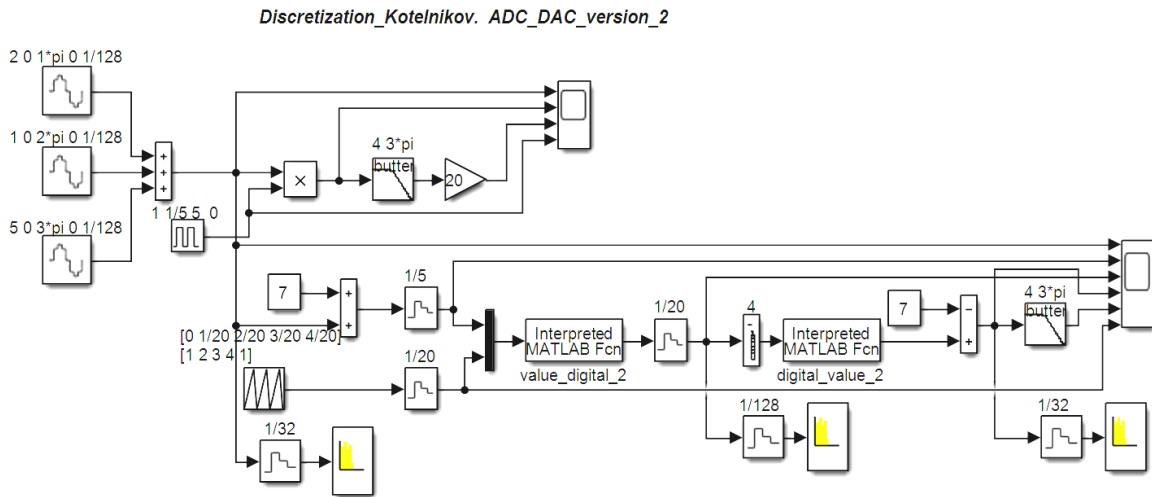


Рисунок 3.2 – Функциональная модель иллюстрации теоремы Котельникова и преобразований АЦП и ЦАП

Дискретизация аналогового сигнала по Котельникову

Так как источник аналогового сигнала периодический и сформирован суммой трех гармоник ($\omega_1 = \pi$, $\omega_2 = 2\pi$, $\omega_3 = 3\pi$) радиан с разными амплитудами. Такое формирование позволяет без измерения спектра определить $\omega_v = 3\pi$ радиан в секунду. Амплитуды гармоник подобраны таким образом, чтобы сигнал изменялся от -7 В до $+7 \text{ В}$. Шаг квантования для удобства выбран равным 1 В , итого имеем $L = 15$ уровней квантования. Для представления такого числа уровней квантования требуется 4-е двоичных разряда. Граничное значение периода дискретизации $T_d = \pi/\omega_v = \pi/3\pi = 1/3$. Нами для модельного эксперимента с запасом взят период дискретизации $T_d = 1/5$.

На рисунке 3.2 в рабочей среде *MatLab* с пакетом *Simulink*, выполнен дискретизатор на основе умножителя (блок *Product*). Аналоговый сигнал приходит на первый вход умножителя, а на второй вход поступают дискретизирующие импульсы (блок *Pulse Generator* с параметрами: *Amplitude=1*, *Period=1/5*, *Pulse Width (% of period)=5*, *Phase delay=0*). Таким образом, дискретизирующие импульсы по длительности равны $\tau_0 = 0.05 \cdot T_d = 0.05 \cdot 0.2 = 0.001$.

Процесс, описанный выше, это дискретизация сигнала, превращение аналогового сигнала в n количество отсчетов (уровней). Обратный процесс дискретизации - это восстановление аналогового сигнала по дискретным отсчетам. Такой процесс осуществляется с помощью фильтра нижних частот (**ФНЧ**), (блок *Analog Filter Design* с параметрами: *Design method=Butterworth*, *Filter type=Lowpass*, *Filter order=4*, *Passband edge frequency=3*pi*.)

Для нормирования амплитуды восстановленного сигнала использован блок **Gain=20**. На рисунке 3.3 приведены фрагменты осциллограмм дискретизируемого сигнала и его восстановления.

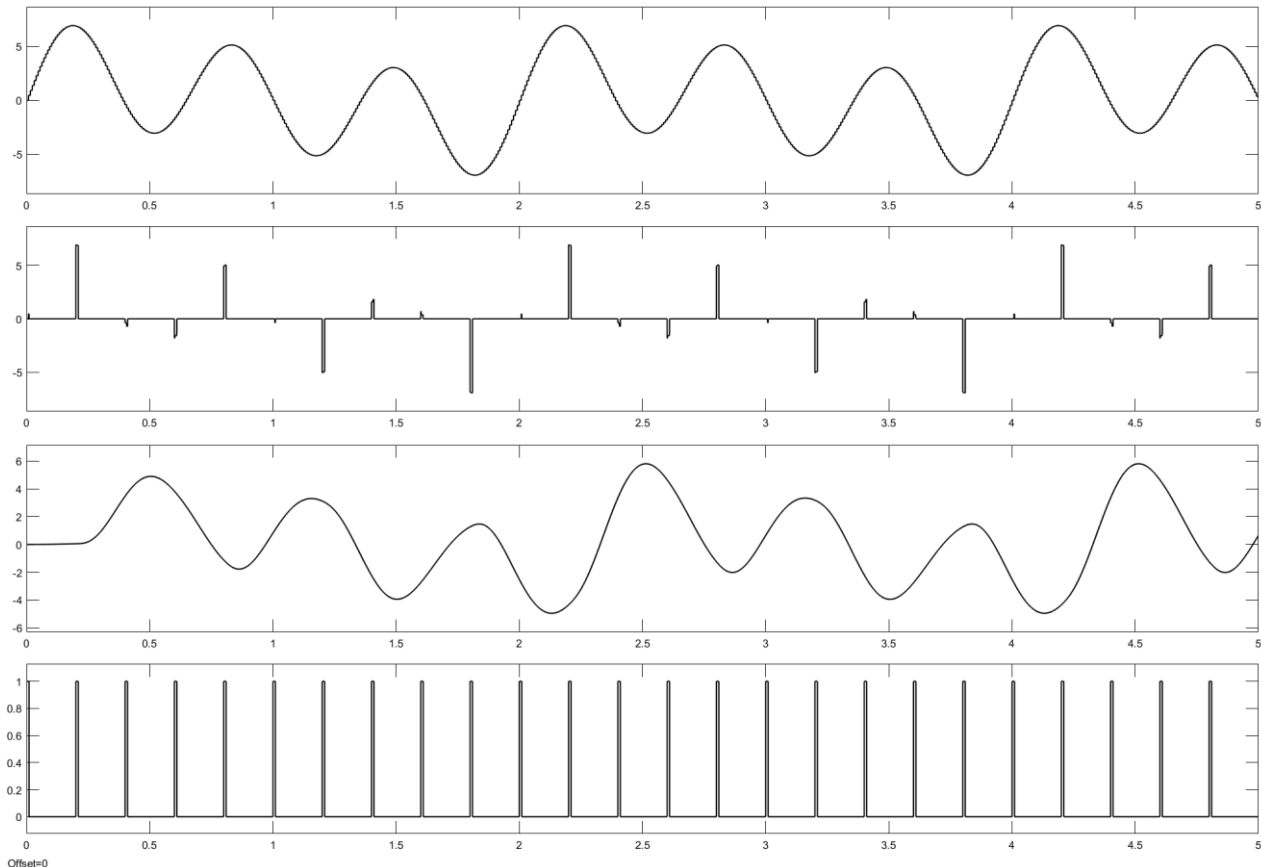


Рисунок 3.3 – Осциллограммы дискретизации аналогового сигнала по Котельникову

Из рисунка 3.3, видно, что качество восстановления дискретизированного сигнала хорошее, но имеется сдвиг из-за фазовой задержки **ФНЧ**.

Цифро-аналоговое и аналого-цифровое преобразования

Преобразования сигналов из аналоговой формы в цифровую и из цифровой в аналоговую осуществляют аналого-цифровые и цифро-аналоговые преобразователи (**АЦП** и **ЦАП**) [3].

Перед **АЦП** преобразованием биполярный сигнал преобразуется в однополярный путем добавления $7 В$ с периодом дискретизации $T_d = 1/5$ (блок **Zero Order Hold**) и подается на первый вход мультиплексора **Mux**. На второй вход мультиплексора подается четырехступенчатый импульс (блок *Repeating Sequence* с параметрами: *Time values=[0 1/20 2/20 3/20 4/20]*, *Output values=[1 2 3 4 1]*) длительностью равной периоду дискретизации $1/5$

предназначенный для считывания четырехразрядного двоичного кода. С выхода мультиплексора объединенный сигнал поступает в функцию *MatLab* (блок *Interpreted MatLab Fcn*). Функция *value_digital_2* программно реализует АЦП преобразования, что представлено на рисунке 3.4.

```
function d = value_digital_2(y)
% Value ----> Digital

x=y(1);
if (x<0.5) z=[0 0 0 0]; end;
if (x>=0.5) && (x<1.5) z=[0 0 0 1]; end;
if (x>=1.5) && (x<2.5) z=[0 0 1 0]; end;
if (x>=2.5) && (x<3.5) z=[0 0 1 1]; end;
if (x>=3.5) && (x<4.5) z=[0 1 0 0]; end;
if (x>=4.5) && (x<5.5) z=[0 1 0 1]; end;
if (x>=5.5) && (x<6.5) z=[0 1 1 0]; end;
if (x>=6.5) && (x<7.5) z=[0 1 1 1]; end;
if (x>=7.5) && (x<8.5) z=[1 0 0 0]; end;
if (x>=8.5) && (x<9.5) z=[1 0 0 1]; end;
if (x>=9.5) && (x<10.5) z=[1 0 1 0]; end;
if (x>=10.5) && (x<11.5) z=[1 0 1 1]; end;
if (x>=11.5) && (x<12.5) z=[1 1 0 0]; end;
if (x>=12.5) && (x<13.5) z=[1 1 0 1]; end;
if (x>=13.5) z=[1 1 1 0]; end;

k=y(2);
k=round(k);
d=z(k);

end
```

Рисунок 3.4 – Программная реализация АЦП

Из рисунка 3.4 видно, что каждому уровню квантования назначается определенный двоичный четырехразрядный код и поразрядно передается на выход функции.

Программное ЦАП преобразование с помощью функции *digital_value_2* представлена на рисунке 3.5.

На выходе функции появляются импульсы амплитудой пропорциональной передаваемому уровню аналогового сигнала.


```

function v = digital_value_2(b)
% Digital ---> Value

v=0;
if all(b==[0; 0; 0; 0]) v=0; end;
if all(b==[0; 0; 0; 1]) v=1; end;
if all(b==[0; 0; 1; 0]) v=2; end;
if all(b==[0; 0; 1; 1]) v=3; end;
if all(b==[0; 1; 0; 0]) v=4; end;
if all(b==[0; 1; 0; 1]) v=5; end;
if all(b==[0; 1; 1; 0]) v=6; end;
if all(b==[0; 1; 1; 1]) v=7; end;
if all(b==[1; 0; 0; 0]) v=8; end;
if all(b==[1; 0; 0; 1]) v=9; end;
if all(b==[1; 0; 1; 0]) v=10; end;
if all(b==[1; 0; 1; 1]) v=11; end;
if all(b==[1; 1; 0; 0]) v=12; end;
if all(b==[1; 1; 0; 1]) v=13; end;
if all(b==[1; 1; 1; 0]) v=14; end;

end

```

Рисунок 3.5 – Программная реализация ЦАП

В ЦАП восстановление биполярного сигнала реализуется вычитанием $7B$. Восстановление аналоговой формы сигнала по дискретным отсчетам осуществляется с помощью ФНЧ блок *Analog Filter Design* с параметрами: *Design method=Butterworth, Filter type=Lowpass, Filter order=4, Passband edge frequency=3*pi*. На рис. 3.6 приведены фрагменты осциллограмм АЦП и ЦАП преобразований сигнала.

Как и следовало ожидать, качество восстановленного сигнала после АЦП и ЦАП преобразований хорошее.

Изменение спектра при преобразовании аналогового сигнала в цифровой

Разберем, как меняется спектр при преобразовании аналогового сигнала в цифровой. Пусть цифровая система предназначена для передачи голосового сообщения. Как известно спектр голосового аналогового сообщения после частичного удаления избыточности занимает примерно диапазон порядка $f_v = 3.1 \text{ кГц}$. Частота дискретизации по Котельникову (Найквисту) выбирается из условия $f_d \geq 2f_v$. Для гарантированного восстановления на практике выбирают $f_d = 8 \text{ кГц}$.

Для оцифровки голоса обычно используют микросхему АЦП с разрядностью 8. В результате получаем скорость цифрового потока $R = 64 \text{ кб/сек}$. В цифровых системах реального времени часто используется блочный код со скоростью помехоустойчивого кодирования равная $k/n = 1/2$, где k - число информационных битов, n - число кодовых битов. В результате скорость цифрового потока возрастает до $R = 128 \text{ кб/сек}$. Для обеспечения синхронизации и передачи служебной информации требуемую скорость передачи необходимо увеличить, например, еще на 5.5%. Итого скорость цифрового потока цифровой системы передачи может составить $R \approx 135 \text{ кб/сек}$. Соответственно требуемая полоса пропускания для импульсов цифрового потока составляет $\Delta F \approx 135 \text{ кГц}$. При передаче на радиочастоте требуемая полоса пропускания (например, при использовании BPSK модуляции ($\tau_{sym} = \tau_b$)) удваивается и будет составлять порядка $\Delta f_R \approx 270 \text{ кГц}$.

Таким образом, цифровое представление аналогового сигнала существенно повышает требование к полосе пропускания, предоставляя возможности цифровой обработки и помехоустойчивого кодирования.

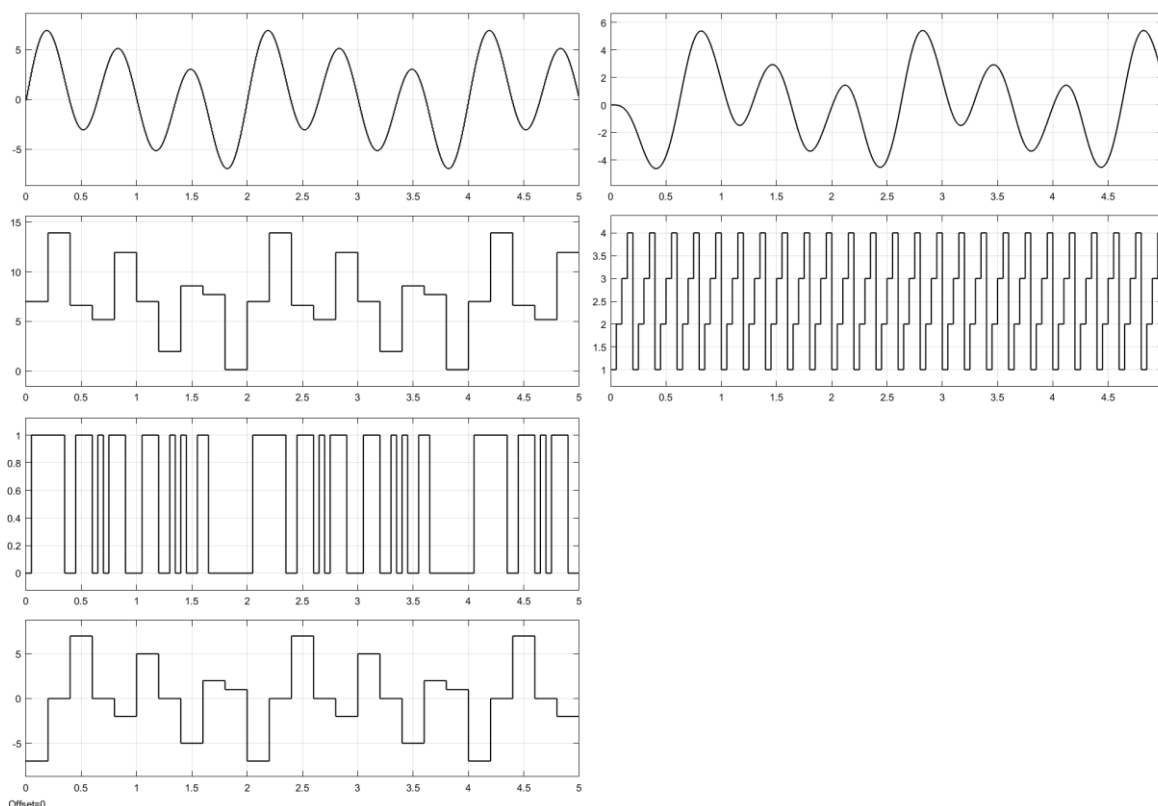


Рисунок 3.6 – АЦП и ЦАП преобразования сигнала

3.3 Модель аддитивного скремблера

Скремблирование передаваемого потока данных – это придание потоку псевдослучайного характера, т.е. исключение протяженных последовательностей нулей или единиц, которые неблагоприятно сказываются на устойчивости систем синхронизации. В системах передачи различают аддитивные и мультипликативные скремблеры, которые по-разному влияют на помехоустойчивость. Суть скремблирования состоит в обратимом изменении бит передаваемого потока [4].

При мультипликативном скремблировании передаваемая последовательность делится на скремблирующий полином, при восстановлении передаваемого потока на приемной стороне умножается на скремблирующий полином, что соответствует использованию трансверсального фильтра и не требует синхронизации. Такой подход, однако приводит к размножению ошибок, например, одиночная ошибка на входе приводит к появлению нескольких ошибок на выходе (их число соответствует числу ненулевых коэффициентов скремблирующего полинома). Таким образом мультипликативный скремблер повышает вероятность битовой ошибки системы передачи.

Структура аддитивного скремблера и дескремблера приведена на рисунке 3.7, где передаваемый поток на входе суммируется (по модулю два) с псевдослучайной последовательностью (**ПСП**) генерируемой на основе регистра сдвига с обратными связями, а на выходе (в приемнике) повторно суммируется с синхронизируемым аналогичным псевдослучайным потоком, что эквивалентно его вычитанию. Максимальная длина псевдослучайной последовательности $N = 2^n - 1$ реализуется, если структура регистра

сдвига соответствует так называемому примитивному полиному, где n – число элементов задержек регистра и, соответственно, порядок полинома. При нормальной синхронизации аддитивный скремблер не ухудшает помехоустойчивость системы передачи.

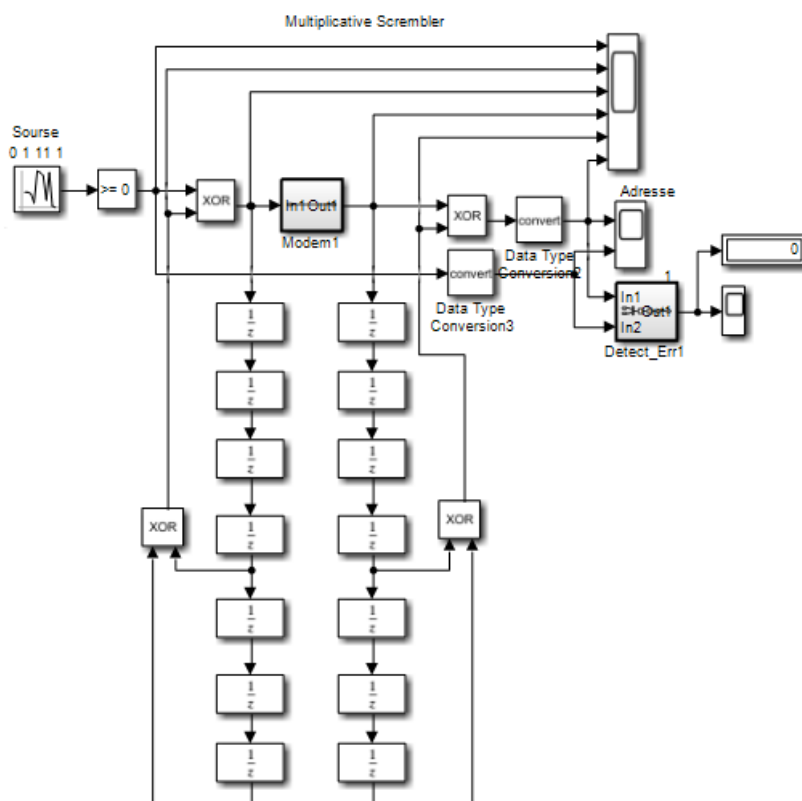


Рисунок 3.7 - Модель аддитивного скремблера на основе примитивного полинома $P(x) = 1 + x^4 + x^7$

На рисунке 3.8 приведены фрагменты осциллограмм аддитивного скремблера.

Время работы функциональной модели скремблера установлено равным **512** при $\tau_b = 1$. В качестве *Modema* использован стандартный пустой блок подсистемы *SubSystem*. Из рисунка 3.8 следует, что функциональная модель аддитивного скремблера функционирует верно – исходная скремблированная и восстановленная дескремблированная последовательности совпадают, о чем свидетельствует нулевое значение детектора ошибок *Detect_Err*. Кроме того, вторая и пятая осциллограммы показывают, что период повторения скремблирующей последовательности составляет **127**, что соответствует порядку примитивного полинома равному **7**.

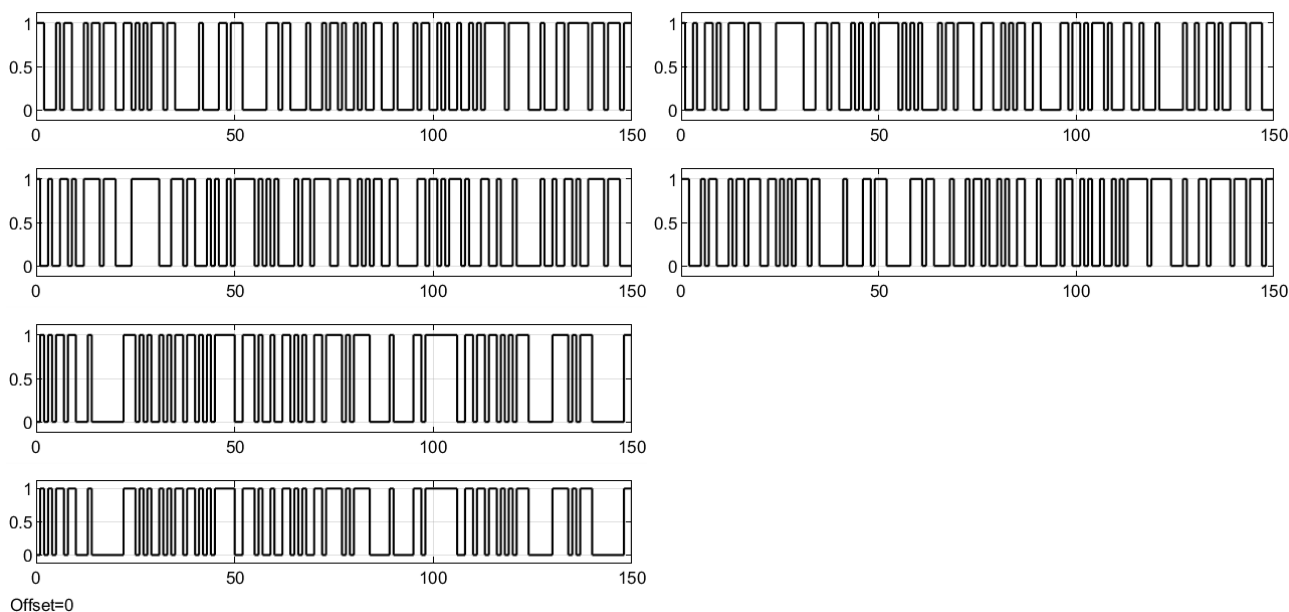


Рисунок 3.8 – Фрагменты осциллограмм аддитивного скремблера

Математическое описание регистров сдвига с обратными связями

Регистры представляют собой цепочки триггеров и предназначены для записи, хранения, сдвига и считывания из них двоичной информации (полубайта, байта и т. д.).

Регистры сдвига с обратными связями находят в цифровой связи самое широкое применение не только в качестве генераторов псевдослучайных последовательностей (*ПСП*), но и, например, при построении помехоустойчивых циклических кодов, позволяющих обнаруживать и исправлять ошибки передачи битового потока. При кодировании блок информационных битов наращивается нулями и делится на полином циклического кодера, представленного регистром сдвига с обратными связями. Остаток от деления выталкивается из кодера и представляет собой избыточные биты или биты четности. Совокупность информационных битов и битов четности представляет собой кодовый символ. Кодовый символ используется при модуляции и в форме радиосигнала поступает в канал распространения, где подвергается воздействию помех. При приеме радиосигнал демодулируется и извлекаются передаваемые биты кодового символа, возможно пораженные ошибками. Для обнаружения и исправления ошибок передачи принятые кодовые символы поступают на циклический декодер, выполненный также в виде регистра сдвига с обратными связями. После декодирования (деления на порождающий полином) в регистрах образуется остаток, называемый синдром, который выталкивается на выход следующим символом. Синдром определяет вектор ошибки, который, суммируясь с информационной частью принятого символа, исправляет ее. В итоге на выходе получаем исходные информационные биты [5].

Рассмотрим две функциональные схемы генераторов *ПСП* (рисунок 3.9) на основе регистров сдвига с обратными связями *LFSR*, отвечающими одному полиному $P(x) = 1 + x^2 + x^5$ и отличающиеся только местом включения запускающего генератора на входе либо выходе регистра. Длина неповторяющейся псевдослучайной последовательности определяется как $M = 2^n - 1$, где n - порядок примитивного полинома.

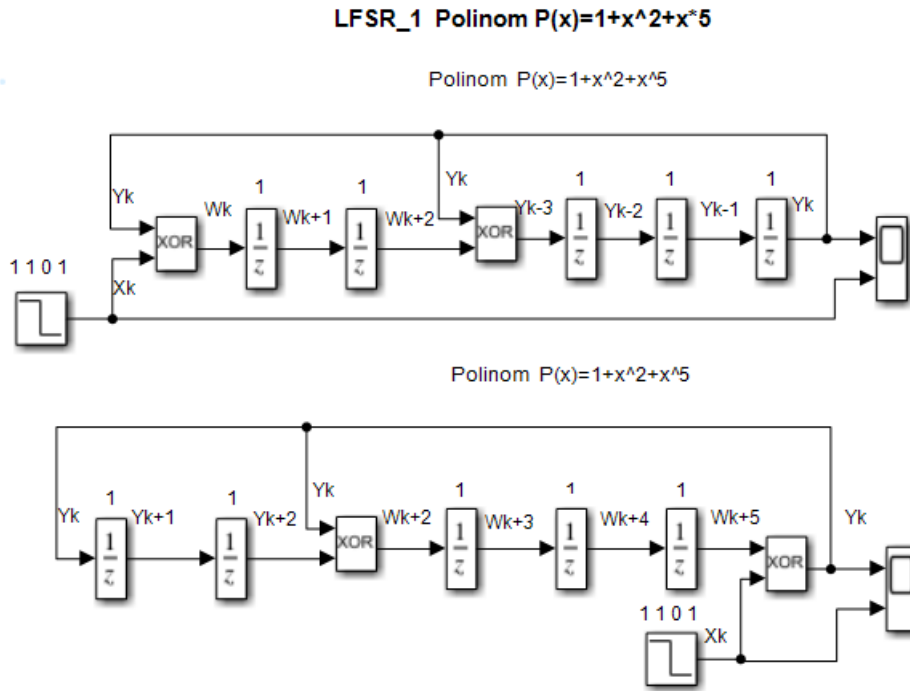


Рисунок 3.9 – Регистры сдвига с источником запуска на входе и выходе

Выходные осциллограммы регистра сдвига с генератором запуска на входе и на выходе приведены на рисунке 3.10. Генератор запускается единичным импульсом.

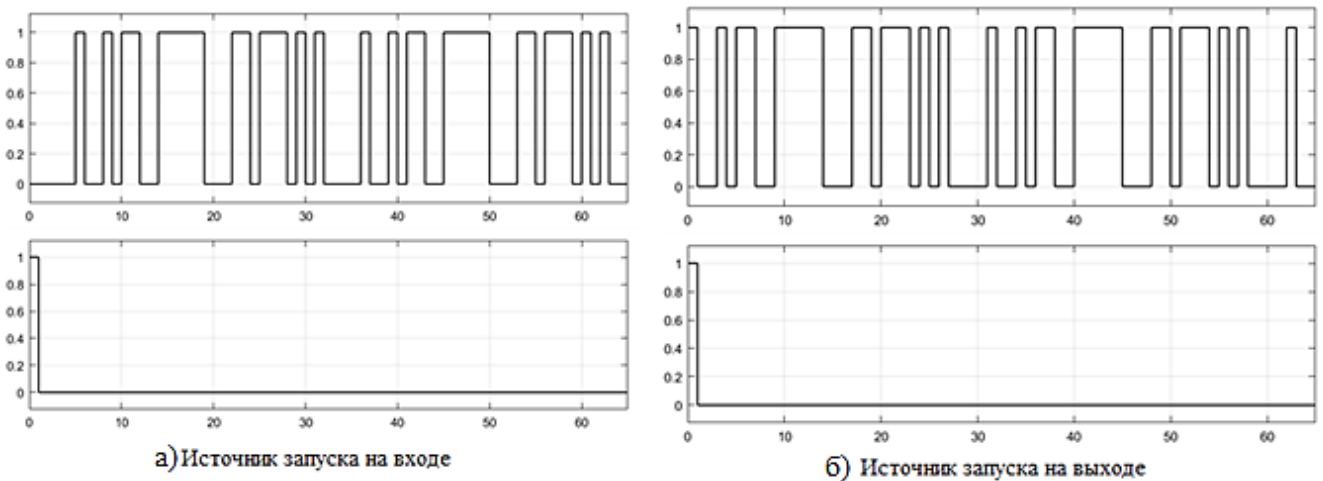


Рисунок 3.10 – Осциллограммы регистра сдвига с источником запуска на входе (а) и на выходе (б)

Для проверки правильности работы функциональных моделей необходимо получить итоговое рекуррентное уравнение регистра сдвига.

Отличие первой и второй функциональной модели состоит в том, что при подаче запускающего импульса на вход регистра сдвига происходит задержка всей последовательности на число регистров сдвига, определяющих порядок полинома.

Рассмотрим один из возможных способов вывода итогового уравнения. Обозначаем дискретные последовательности на входах и выходах сумматоров по модулю 2 (*XOR*) и элементов задержек на такт (*Delay*). Для определенности, импульс генератора запуска

обозначаем x_k , выходную последовательность регистра обозначаем через y_k , последовательности в остальных точках регистра обозначаем сдвигом индекса прежней переменной либо новой переменной после операции **XOR**. Дискретный индекс переменной отражает задержку на элементах **Delay**.

Для каждого элемента **XOR** регистра в терминах назначенных переменных записываем свое уравнение. Так для первой схемы рисунка 3.9 имеем: $w_k = y_k + x_k$; $y_{k-3} = y_k + w_{k+2}$. Здесь знак $+$ означает суммирование по модулю 2. Перепишем второе уравнение со сдвигом на 3 такта $y_k = y_{k+3} + w_{k+5}$. Далее, подставляя первое уравнение с учетом сдвига во второе уравнение, получим

$$y_k = y_{k+3} + y_{k+5} + x_{k+5}.$$

Запишем полное разностное уравнение регистра сдвига и совместим его с таблицей, в которой проследим несколько тактов работы, сдвигая данные таблицы вправо и вниз. В таблицу записываются все компоненты для отслеживания их изменения во времени, начиная с $k = 1$. Компоненты, отсутствующие в уравнении, запишем с весовым коэффициентом 0. Компоненты, присутствующие в уравнении для удобства выделим синим цветом.

Таблица 3.1 – Изменения компонент разностного уравнения **LFSR** во времени

$y_k =$	$0y_{k+1} +$	$0y_{k+2}$	$y_{k+3} +$	$0y_{k+4}$	$y_{k+5} +$	$0x_k +$	$0x_{k+1}$	$0x_{k+2}$	$0x_{k+3}$	$0x_{k+4}$	$x_{k+5} +$
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0	0	0	0	0
0	0	1	1	0	1	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	0
1	1	0	0	1	1	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0	0	0
0	0	1	1	1	1	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0
1	0	0	0	1	1	0	0	0	0	0	0

Сопоставляя данные столбца y_k , видим, что их значения совпадают с осциллограммой рисунок 3.10 (а). Таким образом, убеждаемся в правильности работы функциональной модели.

Для второй схемы имеем следующие уравнения сумматоров **XOR**: $w_{k+2} = y_k + y_{k+2}$; $y_k = w_{k+5} + x_k$. Подставляя второе уравнение со сдвигом в первое уравнение, получаем

$$y_k = y_{k+3} + y_{k+5} + x_k.$$

Запишем полное разностное уравнение регистра сдвига и совместим его с таблицей, в которой проследим несколько тактов работы, сдвигая данные таблицы вправо и вниз.

Сопоставляя данные столбца y_k , видим, что их значения совпадают с осциллограммой рисунок 3.10 (б). Таким образом, убеждаемся в правильности работы функциональной модели.

Таблица 3.2 – Изменения компонент разностного уравнения *LFSR* во времени

$y_k =$	$0y_{k+1} +$	$0y_{k+2} +$	$y_{k+3} +$	$0y_{k+4} +$	$y_{k+5} +$	x_k
1	0	0	0	0	0	1
0	1	0	0	0	0	0
0	0	1	0	0	0	0
1	0	0	1	0	0	0
0	1	0	0	1	0	0
1	0	1	0	0	1	0
1	1	0	1	0	0	0
0	1	1	0	1	0	0
0	0	1	1	0	1	0
1	0	0	1	1	0	0
1	1	0	0	1	1	0
1	1	1	0	0	1	0
1	1	1	1	0	0	0
1	1	1	1	1	0	0
0	1	1	1	1	1	0
0	0	1	1	1	1	0
0	0	0	1	1	1	0
1	0	0	0	1	1	0

Приведем небольшую таблицу 3.3 неприводимых полиномов.

Таблица 3.3 – Неприводимые полиномы

№	Порядок	Аналитическое представление
1	1	$x + 1$
2	2	$x^2 + x + 1$
3	3	$x^3 + x + 1$
4		$x^3 + x^2 + 1$
5	4	$x^4 + x + 1$
6		$x^4 + x^2 + 1$
7	5	$x^5 + x^2 + 1$
8		$x^5 + x^3 + 1$
9		$x^5 + x^3 + x^2 + x + 1$
10		$x^5 + x^4 + x^2 + x + 1$
11		$x^5 + x^4 + x^3 + x + 1$
12	$x^5 + x^4 + x^3 + x^2 + 1$	
13	6	$x^6 + x + 1$
14		$x^6 + x^3 + 1$
15		$x^6 + x^5 + 1$
16		$x^6 + x^4 + x^2 + x + 1$
17		$x^6 + x^4 + x^3 + x + 1$
18		$x^6 + x^5 + x^2 + x + 1$
19		$x^6 + x^5 + x^3 + x + 1$
20		$x^6 + x^5 + x^4 + x + 1$
21		$x^6 + x^5 + x^4 + x^2 + 1$
22	7	$x^7 + x + 1$
23		$x^7 + x^3 + 1$

24		$x^7 + x^3 + x^2 + x + 1$
25		$x^7 + x^4 + 1$
26		$x^7 + x^4 + x^3 + x^2 + 1$
27		$x^7 + x^5 + x^2 + x + 1$
28		$x^7 + x^5 + x^3 + x + 1$
29		$x^7 + x^5 + x^4 + x^3 + 1$
30		$x^7 + x^5 + x^4 + x^3 + x^2 + x + 1$
31		$x^7 + x^6 + 1$
32		$x^7 + x^6 + x^3 + x + 1$
33		$x^7 + x^6 + x^4 + x + 1$
34		$x^7 + x^6 + x^4 + x^2 + 1$
35		$x^7 + x^6 + x^5 + x^2 + 1$
36		$x^7 + x^6 + x^5 + x^3 + x^2 + 1$
37		$x^7 + x^6 + x^5 + x^4 + 1$
38		$x^7 + x^6 + x^5 + x^4 + x^2 + 1$
39		$x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + 1$

4 ОПИСАНИЕ ЦИФРОВЫХ СИГНАЛОВ, СПЕКТРОВ, НИЗКОЧАСТОТНЫХ И ПОЛОСОВЫХ ФИЛЬТРОВ

4.1 Описание модели источника цифровой информационной последовательности

При функциональном моделировании рано или поздно встает вопрос, откуда взять информационную последовательность? В нашем случае применен самый простой подход. В его основе используется идея, что информационный процесс содержит элемент неопределенности, иными словами можно использовать псевдослучайные последовательности. Самый простой способ получения биполярной псевдослучайной последовательности заключается в использовании, какого либо генератора псевдослучайного процесса, например *Random Number* с шагом τ , на выходе которого стоит идеальный ограничитель $(-1, 1)$ (для этой цели подходит функция *Sign*).

На рисунке 4.1 приведена модель создания двоичного биполярного источника цифровых данных с гауссовым законом распределения.

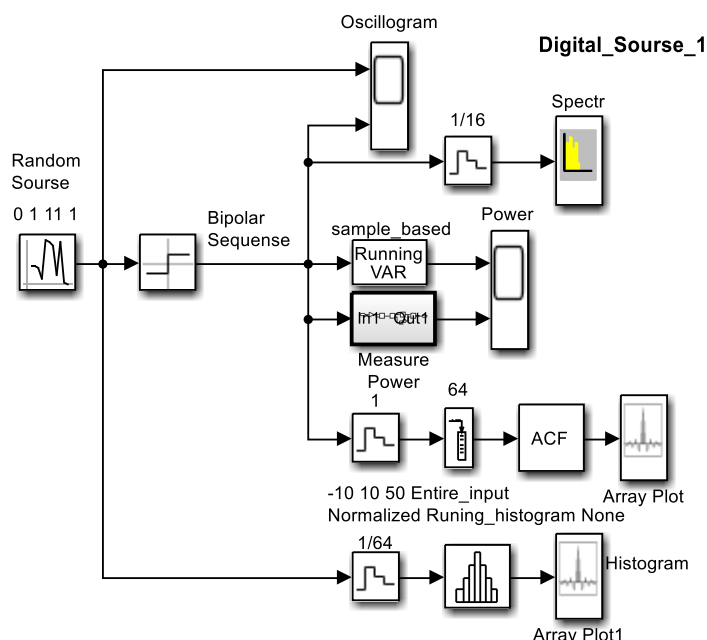


Рисунок 4.1 – Модель двоичного источника данных

Над каждым блоком функциональной модели в помощь студентам через пробелы записаны установленные параметры в порядке их перечисления во вкладке “Параметры блока”.

При моделировании систем передачи информации, в качестве источника информационной последовательности удобно взять генератор псевдослучайной последовательности с гауссовым распределением, как источник, обладающий необходимой степенью неопределенности. Меняя шаг по времени можем изменять длительность бита информационной последовательности или скорость передачи. Для преобразования в биполярную последовательность с амплитудой ± 1 достаточно поставить блок *Sign*. Учитывая относительный характер моделирования в масштабах времени и частоты, удобно выбирать длительность бита в районе $\tau_b = 1$, тогда скажем период несущего колебания

можно выбрать на порядок меньше $T_0 = (0.05 \div 0.1) \cdot \tau_b$. В результате получим круговую частоту несущего колебания порядка $\omega_0 = 2\pi/T_0 = 2\pi/((0.05 \div 0.1) \cdot \tau_b)$.

При моделировании шумов канала распространения также удобно использовать генератор псевдослучайной последовательности с гауссовым распределением и шагом, например, $\Delta t = 0.0125$, что позволяет получить ширину основного лепестка спектра шума порядка $\omega_{noise} = 2\pi/\Delta t = 80\pi$, то есть получить практически равномерную плотность мощности шумов в рабочем диапазоне частот (районе несущей). Меняя мощность генератора можем изменять отношение сигнал/шум SNR и исследовать скажем помехоустойчивость модема – зависимость вероятности битовой ошибки P_b от SNR .

Справка: *Random Number* – генератор псевдослучайной последовательности с Гауссовым распределением. Параметры: *Meam* – среднее, *Variance* – дисперсия (мощность), *Initial seed* – число, определяющее новую уникальную последовательность (рекомендую использовать простые числа), *Sample time* – шаг по времени.

Итак, основу модели составляет источник псевдослучайной последовательности с гауссовым распределением (блок *Random Number*). Эта последовательность проходит через идеальный двухсторонний ограничитель (блок *Sign*) и преобразуется в биполярную псевдослучайную последовательность данных с гауссовым распределением.

Параметры блока: Дисперсия (параметр *Variance*), т.е. (мощность) источника псевдослучайной последовательности с нормальным распределением установлена равной единице $\sigma = 1$, шаг (параметр *Sample time*) псевдослучайной последовательности установлен равным 1 , т.е. $\tau_b = 1$ секунда, амплитуда биполярной битовой последовательности после блока *Sign* соответствует 1 (условно, например, милливольту).

Для наблюдения и измерения в модели представлены блоки осциллографа (*Scope*), спектроанализатора (*Spectrum Analyzer*), блоки *Array Plot*, для отображения гистограммы (плотности распределения) и автокорреляционной функции (*ACF*). Автокорреляционная функция вычисляется блоком *Autocorrelation*, а плотность распределения – блоком *Histogram*. Мощность псевдослучайной битовой последовательности вычисляется блоком *Running VAR* и разработанной подсистемой *Measure Power*.

На рисунке 4.2 приведен фрагмент осциллограмм исходной псевдослучайной последовательности и сформированной из нее блоком *Sign* биполярной последовательности.

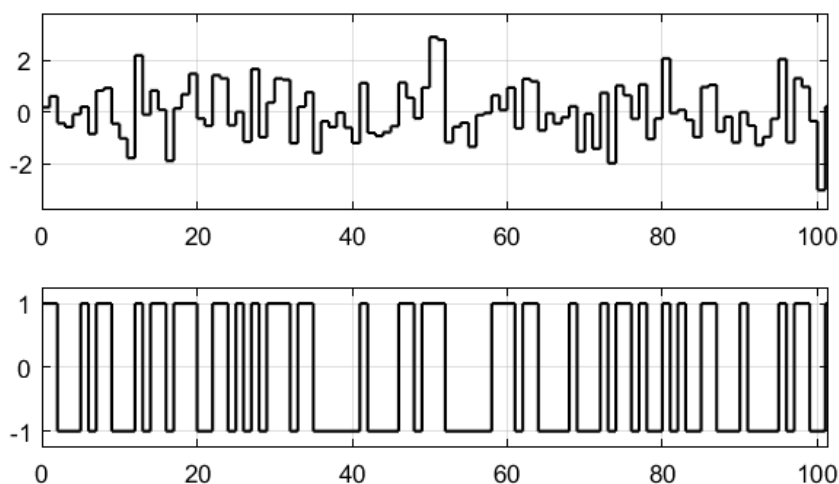


Рисунок 4.2 - Исходная псевдослучайная последовательность и сформированная биполярная последовательность

На рисунке 4.3 приведен график спектральной плотности мощности, сформированной биполярной псевдослучайной последовательности. Из рисунка 4.3 видно, что на самом деле ширина основного лепестка спектральной плотности составляет $\Delta F = 1$ Гц.

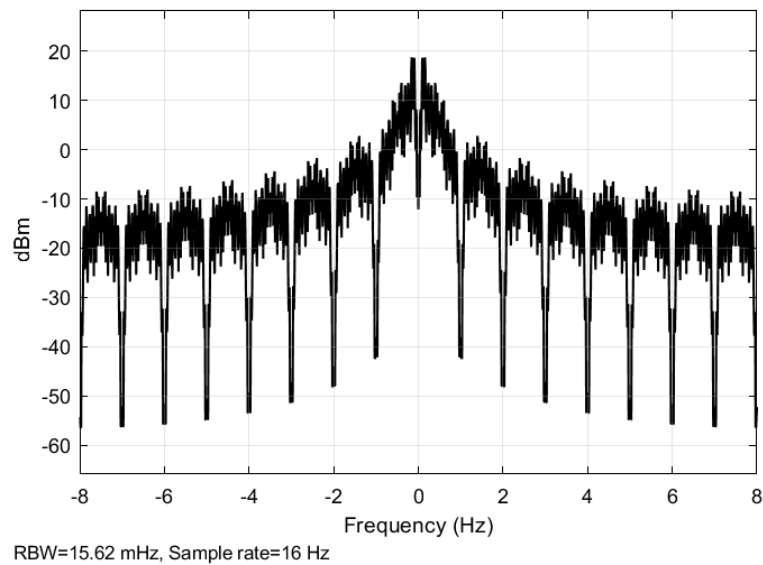


Рисунок 4.3 - График спектральной плотности мощности сформированной информационной биполярной псевдослучайной последовательности

На рисунке 4.4 приведены осциллограммы измеренных мощностей блоком *Running VAR* и разработанной подсистемой *Measure Power*.

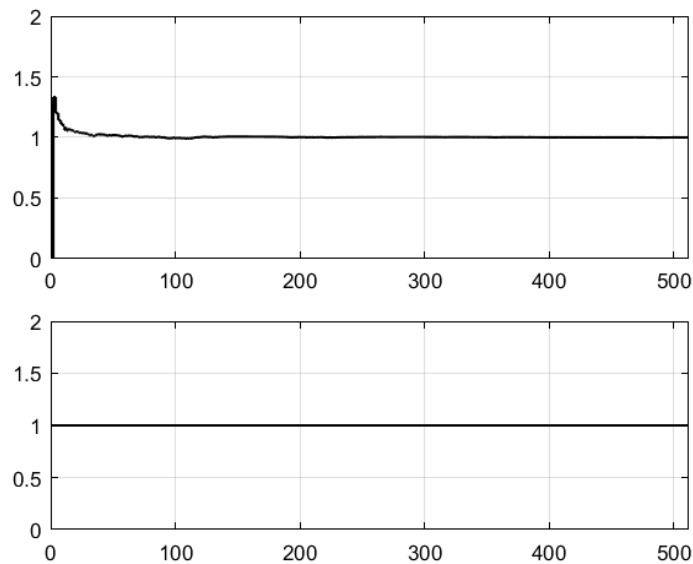


Рисунок 4.4 - Осциллограммы измеренных мощностей блоком **Running VAR** и разработанной подсистемой **Measure Power**

Из рисунка 4.4 видно, что измеренная мощность на самом деле составляет $P = \sigma = 1$ мВт.

На рисунке 4.5 приведена гистограмма исходной псевдослучайной последовательности с нормальным распределением. Из рисунка 4.5 видно, что огибающая спектральной плотности соответствует нормальному закону распределения.

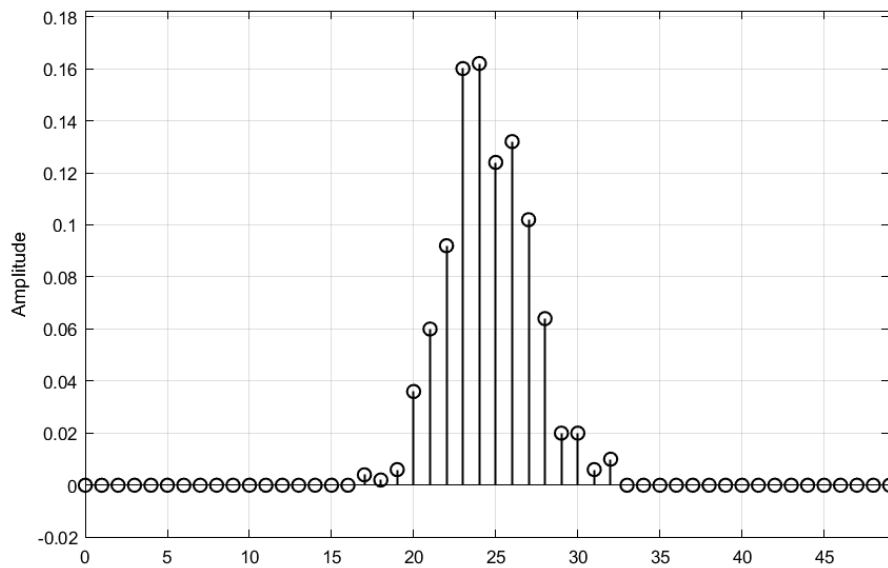


Рисунок 4.5 – Гистограмма исходной псевдослучайной последовательности с нормальным распределением

На рисунке 4.6 приведена автокорреляционная функция псевдослучайной биполярной последовательности с нормальным распределением.

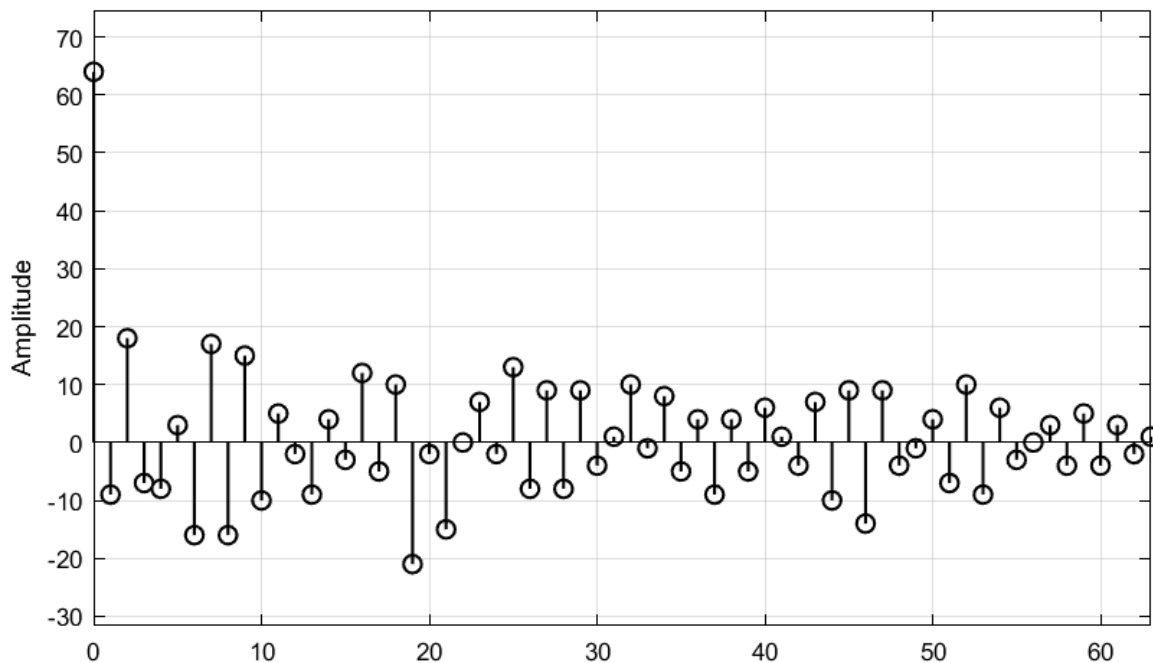


Рисунок 4.6 - Автокорреляционная функция псевдослучайной биполярной последовательности с нормальным распределением

4.2 Описание спектров информационной последовательности и модулированного радиосигнала

Цифровой сигнал представляет собой обычно псевдослучайный поток однополярных или биполярных битов. Пример цифрового сигнала представлен на рисунке 4.7.

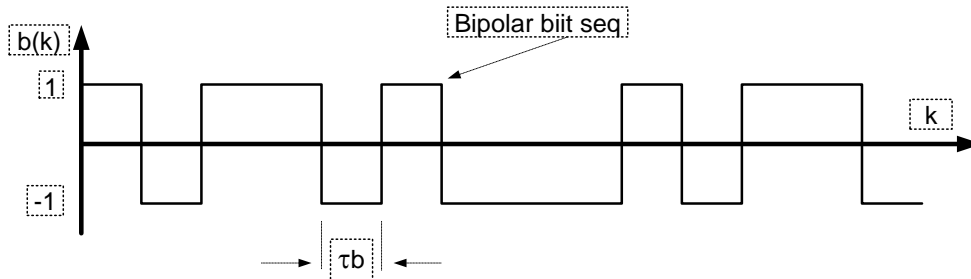


Рисунок 4.7 – Псевдослучайный поток биполярных битов длительности τ_b

Длительность, периодичность и форма сигнала во времени $s(t)$ определяют его спектр. Спектр $S(\omega)$ — это частотное представление сигнала исходя из преобразования Фурье. Согласно идее Фурье преобразования, всякий повторяющийся во времени процесс с периодом T может быть сколь угодно точно представлен набором амплитуд и фаз кратных периоду повторения гармоник, суммируя которые получим во времени тот же сигнал. Набор амплитуд кратных гармоник называется амплитудный спектр, а набор фаз — фазовый спектр.

Цифровой биполярный сигнал, в общем случае носит псевдослучайный характер и ему соответствует понятие текущего мгновенного спектра, поэтому условно его спектр отождествляют со спектром меандровой биполярной последовательности со скважностью 2 и длительностью импульсов τ_b , огибающая амплитудного спектра $|S(\omega)|$ описывается функцией $|\sin(x)/x|$ (рисунок 4.8).

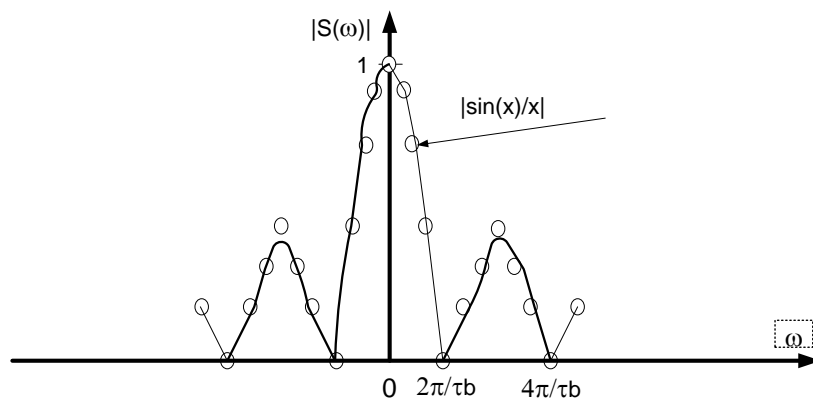


Рисунок 4.8 – Амплитудный спектр однополярной цифровой последовательности

Спектр повторяющегося сигнала линейчатый и линии спектра отстоят на $1/T$. Для биполярного меандра и биполярной псевдослучайной цифровой последовательности постоянная составляющая спектра отсутствует $S(0) = 0$. Модуль огибающей спектра периодически обращается в 0 с шагом $2\pi/\tau_b$ по оси ω , как в отрицательную, так и в положительную стороны.

Любой сигнал, в том числе и цифровой предполагает его передачу по тракту с целью его передачи либо приема и обработки. В связи с этим возникает вопрос о требуемой полосе пропускания тракта. За требуемую полосу пропускания удобно взять ширину основного лепестка, составляющую $2\pi/\tau_b$ и содержащую основную, более 90% энергии импульса. Берется часть лепестка с положительными частотами. Если импульс скругляется, то доля энергии основного лепестка увеличивается, а боковые лепестки спадают с частотой быстрее.

Перенос спектра при модуляции несущей

Особое место среди цифровых систем занимают модемы передачи информации. Модем расшифровывается как модулятор-демодулятор или передатчик-приемник.

Рисунок 4.9, иллюстрирует спектральную картину модема на основе *PSK* модуляции, например, *BPSK*.

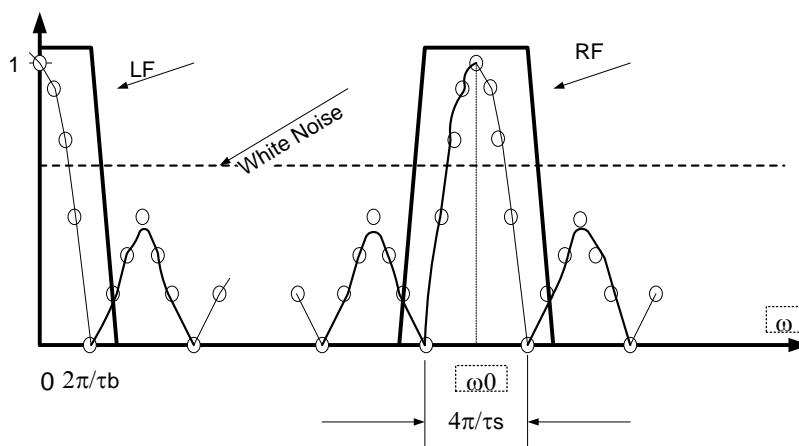


Рисунок 4.9 – Спектральная картина модема на основе *PSK*

Следует обратить внимание на тот факт, что ширина основного лепестка информационного битового процесса $2\pi/\tau_b$ связана с длительностью бита τ_b , а ширина основного лепестка спектра модулированного радиосигнала $4\pi/\tau_s$ связана с длительностью модулирующего символа τ_s . Так при *BPSK* модуляции $\tau_s = \tau_b$, при *QPSK* - $\tau_s = 2\tau_b$, при *8PSK* - $\tau_s = 3\tau_b$, при *16PSK* - $\tau_s = 4\tau_b$ и так далее.

Обратный перенос спектра сигнала в приемной части (демодуляция) осуществляется методом прямого преобразования или синхронного детектирования. Прямое преобразование — это разновидность супергетеродинного приема, когда частота гетеродина приемника совпадает с несущей частотой ω_0 , то есть промежуточная частота равна нулю, что позволяет обойтись без детектора.

4.3 Полосовые и низкочастотные фильтры

На выходе передатчика с целью ограничения внеполосных излучений и на входе приемника с целью ослабления влияния помех за пределами требуемой полосы частот используются полосовые фильтры (*ПФ (RF)*).

В приемной части для фильтрации высокочастотных продуктов демодуляции (преобразования) используются (*ФНЧ (LF)*).

В аппаратных частях передатчика и приемника и в канале распространения радиосигнал подвергается воздействию шумов и помех, что в приемной части

характеризуется отношением сигнал/шум (SNR). Параметр SNR , как отношение мощности сигнала к мощности шумов в точке принятия решений, определяет важнейший параметр модема его помехоустойчивость. Помехоустойчивость характеризуется зависимостью вероятности битовой ошибки (P_b) от SNR и имеет вид водопадopodobной кривой.

Условные изображения нормированных амплитудно-частотных характеристик ($AЧХ$) $ПФ$ и $ФНЧ$ фильтров, представленный на рисунках 4.10, 4.11.

При построении модемов блоки $ФНЧ$ фильтров берут обычно 2-го порядка с верхней граничной частотой $\omega_{gr} = 2\pi/\tau_b$ рад./сек.. На рисунке 4.10 приведена ($AЧХ(FRF)$) LF фильтра.

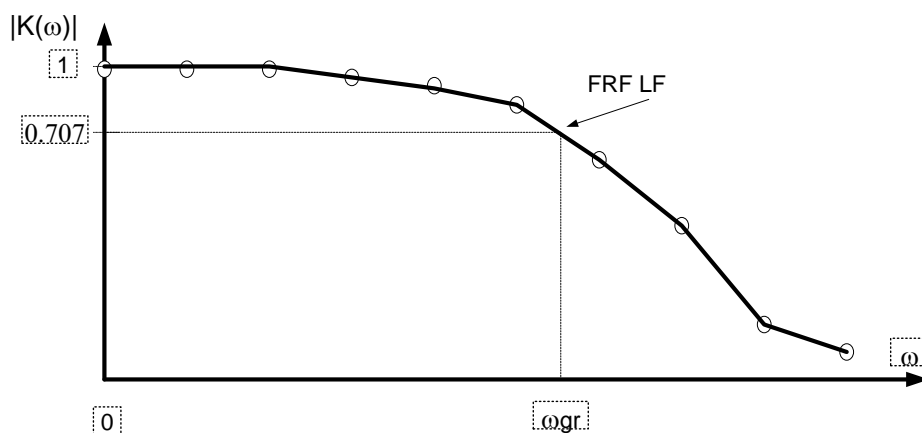


Рисунок 4.10 – $AЧХ(FRF)$ LF фильтра

На рисунке 4.11 приведена ($AЧХ(FRF)$) полосового фильтра $ПФ$. Блоки $ПФ$ берут обычно 2-го порядка для уменьшения выбросов переходных процессов и снижения помехоустойчивости. В настройке граничных частот (полосы пропускания) фильтров есть особенности. Дело в том, что кривая избирательности полосового фильтра на самом деле не симметрична в абсолютных значениях относительно несущей частоты. Симметричность любого полосового фильтра соблюдается в относительном масштабе, т.е. $\omega_0/\omega_L = \omega_U/\omega_0$, где ω_L, ω_U - нижняя и верхняя граничные частоты. Отсюда следует методика выбора граничных частот $ПФ$: например, берем нижнюю граничную частоту равной $\omega_L = \omega_0 - 2\pi/\tau_s$, тогда из симметрии относительного масштаба следует $\omega_U = \omega_0^2/\omega_L$. Можно взять верхнюю граничную частоту равной $\omega_U = \omega_0 + 2\pi/\tau_s$, тогда из симметрии относительного масштаба следует $\omega_L = \omega_0^2/\omega_U$. Несоблюдение этой методики приведет к искажению фазо-частотной характеристики $ПФ$, повороту диаграммы фазовых состояний и значительному снижению помехоустойчивости.

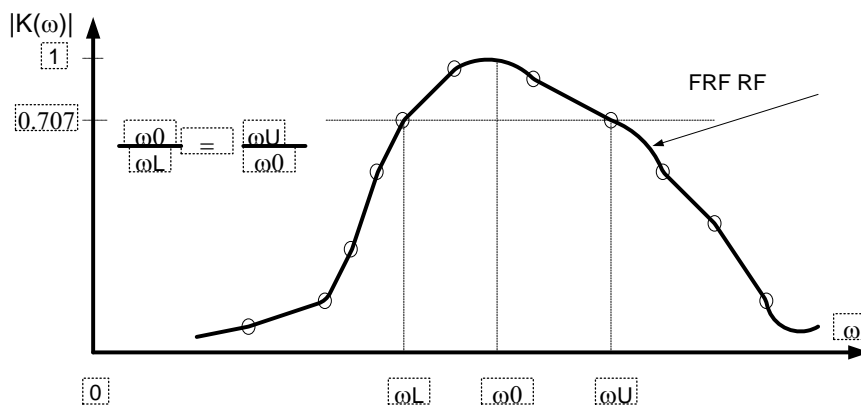


Рисунок 4.11 - АЧХ (FRF) RF фильтра

4.4 Измерение помехоустойчивости

При исследовании помехоустойчивости необходим детектор ошибок [6].

Детектор ошибок. На рисунке 4.12 приведена функциональная модель детектора ошибок передачи. На входе детектора вычитаются принятая модемом последовательность и, задержанная на необходимое число битов передаваемая последовательность. Далее для исключения влияния знака ошибки выполняется операция вычисления модуля блок *Abs*. Для накопления ошибок в виде разности площадей под отображаемыми последовательностями используется блок *Integrator*. На выходе детектора стоит нормирующий множитель блок *Gain* соответствующий площади биполярного либо однополярного бита, который переводит площадь разности в число ошибочных битов.

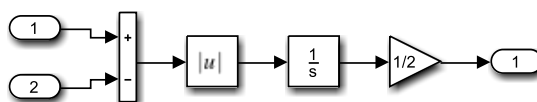


Рисунок 4.12 – Модель подсистемы детектора ошибок *Detect Err*

В результате на выходе детектора ошибок получаем число ошибок передачи модема. Для отображения числа ошибок используют блок *Display*, а для обеспечения возможности фиксации времени возникновения ошибки используют блок *Scope*.

Пусть имеем однополярные последовательности амплитудой равной 1 и длительностью бита равной 1 . Тогда площадь бита равна 1 и нормирующий параметр $Gain=1$. При увеличении амплитуды в 2 раза параметр $Gain=1/2$, соответственно, при уменьшении амплитуды до $1/2$ параметр $Gain=2$.

При тех же параметрах амплитуды и длительности битов биполярной последовательности площадь бита становится вдвое больше, поэтому параметр $Gain=1/2$.

Измеритель мощности сигнала и/или шума. На рисунке 4.13 приведена подсистема измерения мощности вещественных и комплексных, регулярных и случайных процессов.

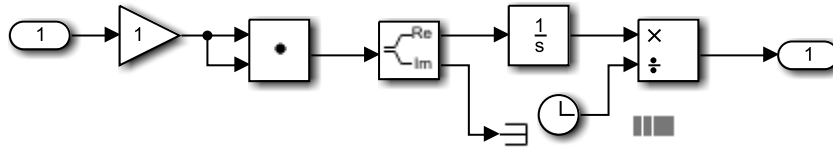


Рисунок 4.13 - Подсистема измерения мощности вещественных и комплексных, регулярных и случайных процессов

Описание модели. На входе подсистемы использован необязательный блок усиления *Gain* с параметром **1**. Далее использован блок *Dot Product*, который перемножает поступившие на входы процессы, предварительно выполняя операцию комплексного сопряжения с процессом второго входа. Благодаря этому измеритель работает не только с вещественными случайными и/или детерминированными процессами, но и комплексными. Далее, поскольку обычно имеем дело с действительной мощностью, используем блок *Complex to Real-Imag*, который отделяет действительную часть, а мнимую часть отправляет на заглушку *Terminator*. Действительная часть произведения подаётся на блок *Integrator*, в итоге получаем энергию. Используя блок *Divide* для деления на текущее время (блок *Clock*), получаем скорость поступления энергии, то есть мощность.

С математической точки зрения, энергия процесса $s(t)$ записывается выражением

$$E(t) = \int_{-\infty}^{\infty} s(t) \cdot s(t)^* dt,$$

где * - знак комплексного сопряжения. При делении энергии на время, получаем выражение мощности

$$P(t) = \frac{1}{T} \cdot \int_{-\infty}^{\infty} s(t) \cdot s(t)^* dt.$$

Замечание. Для формирования однополярного псевдослучайного источника битовой последовательности достаточно после блока *Sign* использовать блок сравнения с нулем *Compare To Zero* с параметром ≥ 0 .

Методика измерения помехоустойчивости. Помехоустойчивость оценивается по кривой зависимости вероятности битовой ошибки от соотношения сигнал/шум (см. рис. 4.14).

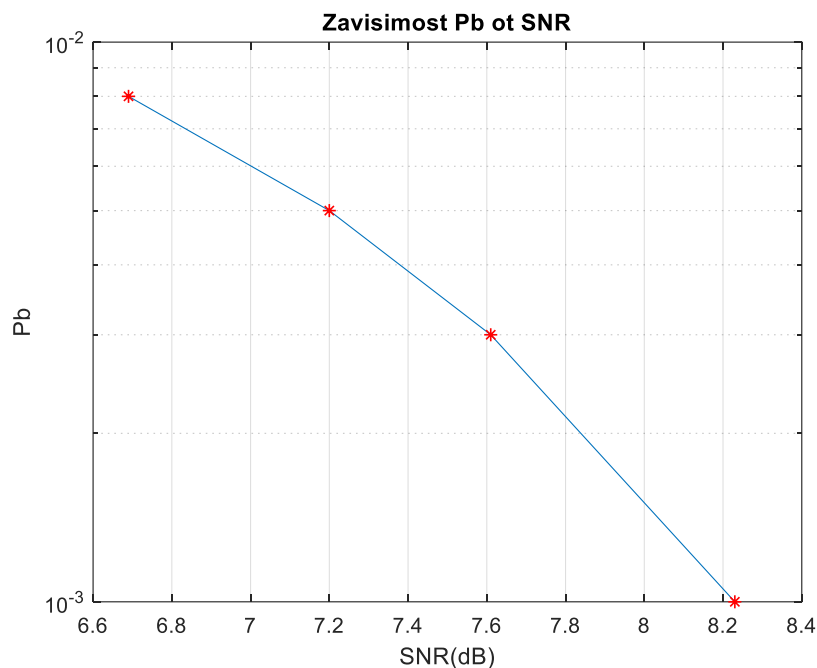


Рисунок 4.14 – Зависимость вероятности битовой ошибки от соотношения сигнал/шум

Помехоустойчивость зависит от соотношения сигнал/шум **SNR** или определяется величиной отношения сигнал/шум необходимого для достижения конкретной вероятности битовой ошибки **P_b**. За вероятность принимается частота появления ошибки при большом числе испытаний.

При моделировании помехоустойчивости между передающей и приемной частями модема ставится простейшая модель канала распространения в виде сумматора **Sum** на один вход которого подается передаваемый сигнал, а на другой вход подключается широкополосный шум в виде генератора псевдослучайной последовательности, например, с нормальным распределением. Изменяя интенсивность шумов, меняем соотношение сигнал/шум **SNR** и замеряем вероятность появления ошибок **P_b**. Длина информационной последовательности должна быть достаточно большой, скажем не менее **1000** битов. Соотношение сигнал/шум **SNR** вычисляем в единицах **дБ**. Измерения сигнала или смеси сигнала с шумом производим на выходе **ФНЧ** демодулятора, т.е. на входе схемы принятия решения блока **Zero Order Hold**.

Методика измерения. Методика измерения выглядит примерно следующим образом. Изменяя уровень шумов канала распространения добиваемся, например **1** ошибки на **1000** битов **P_b = 0.001** и измеряем мощность смеси сигнала с шумом **SN** на выходе демодулятора. Затем, отключаем источник шумов канала распространения просчитываем и измеряем уровень мощности сигнала **S**. Отношение сигнал/шум в **дБ** вычисляем из выражения

$$SNR = S/N = S/(SN - S) \text{ [dB]}.$$

Таким образом, имеем точку водопадоподобной кривой. Далее плавно увеличивая интенсивность шумов канала добиваемся, скажем, **3, 5, 8** ошибок и измеряя мощности смесей сигнала с шумом, вычисляем соответствующие **SNR**. Мощность сигнала измеряется один раз. Теперь имея **4-е** измеренные точки набираем массивы измеренных **SNR** и вероятностей **P_b** с помощью короткого программного кода с использованием функции

semilogy отображаем в полулогарифмическом масштабе водопадоподобную кривую по аналогии с приведенным ниже фрагментом.

Фрагмент *MatLab* программы отображения графика водопадоподобной кривой в полулогарифмическом масштабе

```
clc; clear all; format compact; %clf;
SNR_dB=[8.23 7.61 7.20 6.69];
Pb=[1e-3 3e-3 5e-3 8e-3];
figure(1);
semilogy(SNR_dB, Pb, SNR_dB, Pb, 'r*');
title('Zavisimost Pb ot SNR');
xlabel('SNR (dB) ');
ylabel('Pb');
grid;
```

Рисунок 4.15 - *Script*-файл программы отображения графика водопадоподобной кривой

Взаимосвязь отношения сигнал/шум с нормированным отношением сигнал/шум.

Такой показатель как помехоустойчивость системы передачи зависит от соотношения сигнал/шум *SNR*, определяемый как

$$SNR = S/N,$$

где *S*- мощность сигнала; *N*- мощность шума. *SNR*- измеряется в приемной части модема на выходе демодулятора, после блока *ФНЧ*, т.е. на входе блока принятия решения, обычно это блок *Zero Order Hold*.

Для цифровых систем передачи более распространено нормированное отношение сигнал/шум или энергетическая эффективность E_b/N_0 , где E_b - энергия бита; N_0 - односторонняя спектральная плотность шума приходящаяся на *1 Гц* используемой полосы частот.

Энергетическая эффективность определяет удельный расход энергии на передачу бита данных с заданной вероятностью ошибок.

Для установления связи *SNR* и энергетической эффективности распишем составляющие

$$SNR = S/N = \frac{E_b/T_b}{(N_0 \cdot W)} = \frac{(E_b \cdot R_b)}{(N_0 \cdot W)} = E_b/N_0 \cdot R_b/W,$$

где T_b - длительность бита; $R_b = 1/T_b$ - скорость передачи бита; *W*- используемая полоса частот. R_b/W - спектральная эффективность т.е. скорость передачи данных приходящаяся на *1 Гц* используемой полосы частот (бит/сек/Гц).

Спектральная эффективность характеризует скорость передачи данных в заданной полосе частот. Обратная величина спектральной эффективности $W/R_b = W \cdot T_b$ называется *базой сигнала* и представляет собой удельный расход полосы частот на передачу одного бита. Из развернутого выражения *SNR* следует

$$E_b/N_0 = S/N \cdot W/R_b = SNR \cdot W/R_b.$$

Помехоустойчивость цифровых систем принято характеризовать зависимостью вероятности битовой ошибки P_b от удельного расхода энергии на передачу одного бита данных E_b/N_0 .

4.5 Способы приема (демодуляции)

Корреляционный приемник – приемная часть модема может быть реализована по принципу корреляционного приема. В каждом канале несущей стоит по коррелятору. Коррелятор (рисунок 4.16) имеет два плеча накопления дискретизированных блоками **Zero Order Hold** отсчетов принятого в шумах радиосигнала и отсчетов гармонического колебания синхронизированного с передающей частью опорного генератора. Накопление отсчетов производится в буферах блок **Buffer**. Поскольку предполагается побитовый прием, то, например, шаг дискретизации берется равным $\Delta t = 1/64$, а размер буфера равен $N = 64$, тогда за время действия бита накапливается вектор размерностью **64**.

Далее блоками **Product** и **Sum** выполняется скалярное произведение векторов отсчетов. Даже при наличии помех при приеме бита равного 1 на выходе сумматора будет накапливаться нарастающий положительный всплеск, а при приеме бита равного -1 на выходе сумматора будет накапливаться нарастающий отрицательный всплеск. Использование блока **Sign** нормирует принимаемые значения всплесков до ± 1 . Корреляционный прием сигнала возможен при отношении сигнал/шум меньше единицы.

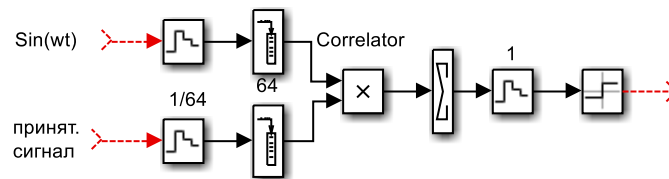


Рисунок 4.16 – Корреляторы каналов приема

Каждый коррелятор в данном случае выдает отсчет корреляционной функции $R(\tau)$ текущего бита при смещении $\tau = 0$. На выходе каналов обработки используется комбинация блоков **Zero Order Hold** и **Sign** для регенерации формы принимаемых битов. Далее принятый сигнал поступает для визуализации на блок **Scope** и для контроля ошибок передачи на подсистему детектора ошибок **Detect Err**.

Приемник на основе прямого преобразования или синхронного детектирования – на входе приемника прямого преобразования (рис. 4.17), для снижения влияния шумов канала распространения, используют полосовой фильтр блок **Analog Filter Design** с полосой пропускания, например, порядка $\Delta\omega = 4\pi$. На входе канала приема и обработки стоит преобразователь блок **Product**. На один из входов преобразователя поступает принятый сигнал, а на второй вход, синхронизированное с передающей частью, опорное колебание несущей частоты.

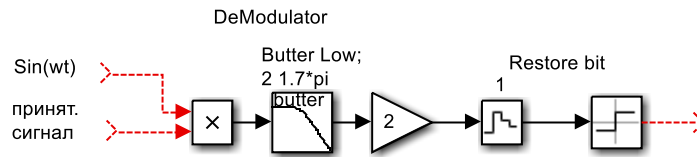


Рисунок 4.17 - Модель канала приемника прямого преобразования принятого сигнала

На выходе преобразователя используется **ФНЧ (LF)**, например, с верхней граничной частотой порядка $\omega_{gr} = 2\pi$, выделяющий спектр принимаемой информационной последовательности и отфильтровывающий высокочастотные продукты преобразования блок **Analog Filter Design** и блок **Gain=2** для компенсации коэффициента тригонометрического преобразования равного $1/2$. На выходе каналов обработки используется комбинация блоков **Zero Order Hold** и **Sign** для регенерации формы принимаемых битов. Далее принятый сигнал поступает для визуализации на блок **Scope** и для контроля ошибок передачи на подсистему детектора ошибок **Detect Err**.

5 ОПИСАНИЕ ИСПОЛЪЗУЕМЫХ ЦИФРОВЫХ МОДУЛЯЦИЙ

5.1 Описание моделей модемов на основе BPSK модуляции

Цифровые данные можно представлять с помощью изменения состояния фазы; этот метод называется фазовой манипуляцией (*PSK, phase shift keying*). Наиболее простой тип *PSK* называется двоичной фазовой манипуляцией (*BPSK, binary phase shift keying*), где «двоичный» относится к использованию двух фазовых состояний (одно для логической единицы и одно для логического нуля) [7].

Описание модели. На рисунке 5.1 и в приложении А (рисунок А.1), приведена функциональная модель квадратурной реализации *BPSK* модема *Quadrature BPSK Modem_1 QBPSK* с приемником прямого преобразования. Квадратурный вариант *BPSK* модуляции позволит в перспективе по аналогии *QPSK* модуляцией реализовать различные подвиды *BPSK* модуляции, обладающие новыми свойствами.

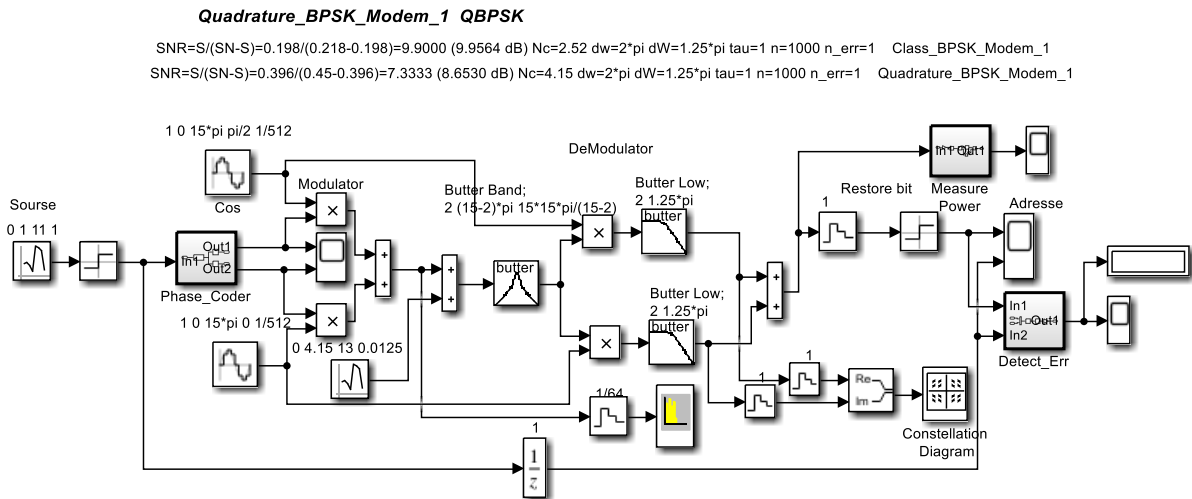


Рисунок 5.1 - Функциональная модель квадратурной реализации *BPSK* модема с приемником прямого преобразования

Входная биполярная информационная последовательность с $\tau_b = 1$ сформирована блоками *Random Number* и *Sign*. Далее информационная последовательность подается на вход подсистемы фазового кодера *Phase_Coder* (см. рис. 5.2).

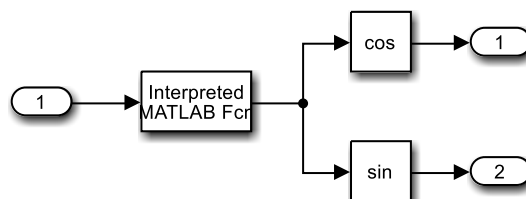


Рисунок 5.2 – Функциональная модель подсистемы фазового кодера *Phase_Coder* квадратурной реализации *BPSK* модема

На входе подсистемы стоит блок *Interpreted MatLab Fcn*, содержащий обращение к *MatLab*-функции *bit_phase_1*. *Script*-файл *MatLab*-функции приведен на рис. 5.3 *MatLab*-

функция ставит в соответствие поступающему биту фазовое состояние φ_k , так значению бита 1 соответствует фазовое состояние $\varphi_k = \pi/4$, а значению -1 , фазовое состояние $\varphi_k = -3\pi/4$. Далее вычисляются значения $\cos(\varphi_k)$ и $\sin(\varphi_k)$ и передаются на входы квадратурного модулятора *BPSK*. Возможно и другое назначение фазовых состояний. Так на рис. 5.4 приведен *Script*-файл *MatLab*-функции *bit_phase_2*. В этом случае значению бита 1 соответствует фазовое состояние $\varphi_k = -\pi/4$, а значению -1 , фазовое состояние $\varphi_k = 3\pi/4$. Основное различие в использовании этих функций состоит в том, что при использовании функции *bit_phase_1* в качестве фазового декодера в приемнике выступает сумматор, а при использовании функции *bit_phase_2* в качестве фазового декодера в приемнике выступает вычитатель.

```
function fi = bit_phase_1(x)
% transformation bit--->phase
% x- bit
% fi- phase

if x==1; fi=pi/4; end
if x==-1; fi=-3*pi/4; end
end
```

Рисунок 5.3 – *Script*-файл *MatLab*-функции *bit_phase_1*

```
function fi = bit_phase_2(x)
% transformation bit--->phase
% x- bit
% fi- phase

if x==1; fi=-pi/4; end
if x==-1; fi=3*pi/4; end
end
```

Рисунок 5.4 - *Script*-файл *MatLab*-функции *bit_phase_2*

Квадратурный модулятор *BPSK* реализован на двух умножителях *Product* и сумматора *Sum* на выходе. Как уже отмечалось на первые входы умножителей поступают управляющие импульсы пропорциональные значениям $\cos(\varphi_k)$ и $\sin(\varphi_k)$, а на вторые входы модулятора подаются квадратурные колебания несущих $\cos(\omega_0 t)$ и $\sin(\omega_0 t)$ с частотой $\omega_0 = 15\pi$.

В качестве модели канала распространения использован блок *Sum*, на второй вход которого подается псевдослучайная гауссова последовательность, сформированная блоком *Random Number*. Интенсивность или мощность шумов канала распространения задается параметром *Variance*. Параметр *Sample time* определяет необходимую широкополосность шумов канала распространения в районе несущего колебания $\sin(\omega_0 t)$. Изменение параметра *Variance* позволяет менять отношение сигнал/шум (*SNR*) в процессе измерения зависимости вероятности битовой ошибки P_b от *SNR*, то есть помехоустойчивости.

На входе приемника прямого преобразования, для снижения влияния шумов канала распространения, используется полосовой фильтр блок *Analog Filter Design (LF)* с полосой пропускания порядка $\Delta\omega = 4\pi$. После полосового фильтра *RF* принятый сигнал разветвляется на входы квадратурного демодулятора.

Квадратурный демодулятор выполнен на двух умножителях блоки *Product*, на первые входы которых поступает принятый и отфильтрованный от лишних помех сигнал. На вторые входы умножителей подаются синхронизированные с передающей стороной колебания опорных генераторов несущей $\cos(\omega_0 t)$ и $\sin(\omega_0 t)$. На выходе умножителей используются

ФНЧ (LF) с верхней граничной частотой, в данном случае, порядка $\omega_{gr} = 1.25\pi$, выделяющий спектр принимаемой информационной последовательности и отфильтровывающие высокочастотные продукты преобразования блоки *Analog Filter Design* и блоки *Gain=2* для компенсации коэффициента тригонометрического преобразования равного $1/2$. Далее в качестве фазового декодера используется сумматор блок *Sum*, если в фазовом кодере значениям битов ± 1 устанавливались фазовые состояния $(\pi/4, -3\pi/4)$. В качестве фазового декодера используется вычитатель блок *Sum*, если в фазовом кодере значениям битов ± 1 устанавливались фазовые состояния $(-\pi/4, 3\pi/4)$.

На выходе фазового декодера используется комбинация блоков *Zero Order Hold* и *Sign* для регенерации формы принимаемых битов. Далее принятый сигнал поступает для визуализации на блок *Scope* и для контроля ошибок передачи на подсистему детектора ошибок *Detect Err*.

Описание модели. На рисунке 5.5 и в приложении А (рисунок А.2), приведена функциональная модель квадратурной реализации **BPSK** модема *Quadrature_BPSK_Modem_2 QBPSK_Corr* с корреляционным приемником. Квадратурный вариант **BPSK** модуляции позволит в перспективе по аналогии **QPSK** модуляцией реализовать различные подвиды **BPSK** модуляции, обладающие новыми свойствами.

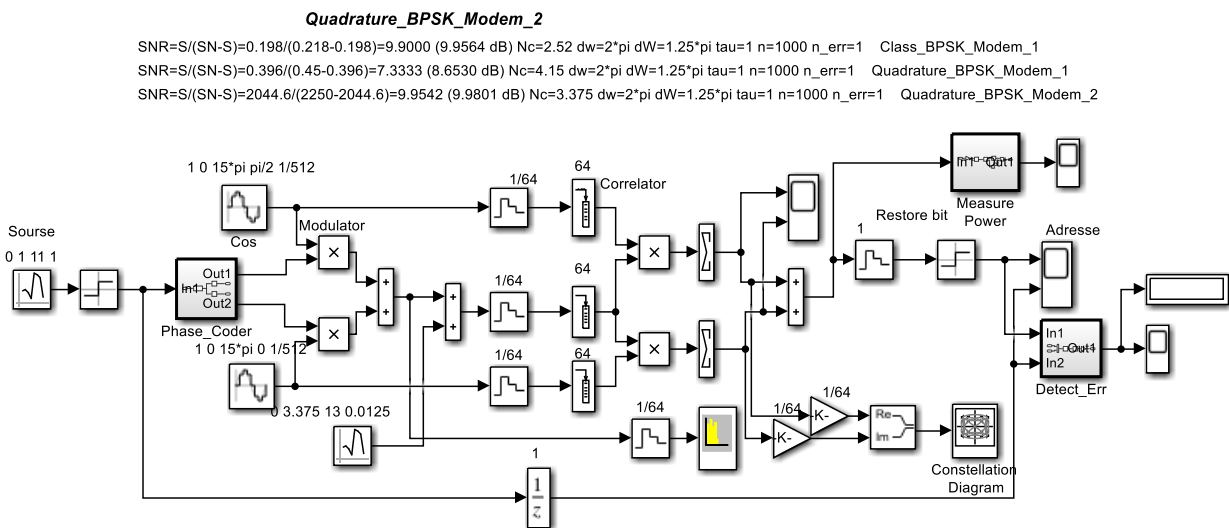


Рисунок 5.5 - Функциональная модель квадратурной реализации **BPSK** модема с корреляционным приемником

Входная биполярная информационная последовательность с $\tau_b = 1$ сформирована блоками *Random Number* и *Sign*. Далее информационная последовательность подается на вход подсистемы фазового кодера *Phase_Coder* (см. рис. 5.6).

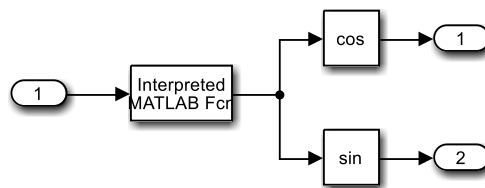


Рисунок 5.6 – Функциональная модель подсистемы фазового кодера *Phase_Coder* квадратурной реализации **BPSK** модема

На входе подсистемы стоит блок *Interpreted MatLab Fcn*, содержащий обращение к *MatLab*-функции *bit_phase_1*. *Script*-файл *MatLab*-функции приведен на рис. 5.7 *MatLab*-функция ставит в соответствие поступающему биту фазовое состояние φ_k , так значению бита *1* соответствует фазовое состояние $\varphi_k = \pi/4$, а значению *-1*, фазовое состояние $\varphi_k = -3\pi/4$. Далее вычисляются значения $\cos(\varphi_k)$ и $\sin(\varphi_k)$ и передаются на входы квадратурного модулятора *BPSK*. Возможно и другое назначение фазовых состояний. Так на рис. 5.8 приведен *Script*-файл *MatLab*-функции *bit_phase_2*. В этом случае значению бита *1* соответствует фазовое состояние $\varphi_k = -\pi/4$, а значению *-1*, фазовое состояние $\varphi_k = 3\pi/4$. Основное различие в использовании этих функций состоит в том, что при использовании функции *bit_phase_1* в качестве фазового декодера в приемнике выступает сумматор, а при использовании функции *bit_phase_2* в качестве фазового декодера в приемнике выступает вычитатель.

```
function fi = bit_phase_1(x)
% transformation bit--->phase
% x- bit
% fi- phase

if x==1; fi=pi/4; end
if x==-1; fi=-3*pi/4; end
end
```

Рисунок 5.7 – *Script*-файл *MatLab*-функции *bit_phase_1*

```
function fi = bit_phase_2(x)
% transformation bit--->phase
% x- bit
% fi- phase

if x==1; fi=-pi/4; end
if x==-1; fi=3*pi/4; end
end
```

Рисунок 5.8 - *Script*-файл *MatLab*-функции *bit_phase_2*

Квадратурный модулятор *BPSK* реализован на двух умножителях *Product* и сумматора *Sum* на выходе. Как уже отмечалось на первые входы умножителей поступают управляющие импульсы пропорциональные значениям $\cos(\varphi_k)$ и $\sin(\varphi_k)$, а на вторые входы модулятора подаются квадратурные колебания несущих $\cos(\omega_0 t)$ и $\sin(\omega_0 t)$ с частотой $\omega_0 = 15\pi$.

В качестве модели канала распространения использован блок *Sum*, на второй вход которого подается псевдослучайная гауссова последовательность, сформированная блоком *Random Number*. Интенсивность или мощность шумов канала распространения задается параметром *Variance*. Параметр *Sample time* определяет необходимую широкополосность шумов канала распространения в районе несущего колебания $\sin(\omega_0 t)$. Изменение параметра *Variance* позволяет менять отношение сигнал/шум (*SNR*) в процессе измерения зависимости вероятности битовой ошибки P_b от *SNR*, то есть помехоустойчивости.

Корреляционный приемник – приемная часть модема реализована по принципу корреляционного приема. В каждом канале несущей стоит по коррелятору. Коррелятор (см. рис. 5.9) имеет два плеча накопления дискретизированных блоками *Zero Order Hold* отсчетов принятого в шумах радиосигнала и отсчетов гармонических колебаний синхронизированного с передающей частью опорного генератора. Накопление отсчетов производится в буферах блок *Buffer*. Поскольку предполагается побитовый прием, то,

например, шаг дискретизации берется равным $\Delta t = 1/64$, а размер буфера равен $N = 64$, тогда за время действия бита накапливается вектор размерностью **64**. Далее блоками *Product* и *Sum* выполняется скалярное произведение векторов отсчетов. Даже при наличии помех при приеме бита равного I на выходе сумматора будет накапливаться нарастающий положительный всплеск, а при приеме бита равного $-I$ на выходе сумматора будет накапливаться нарастающий отрицательный всплеск. Использование блока *Sign* нормирует принимаемые значения всплесков до ± 1 . Корреляционный прием сигнала возможен при отношении сигнал/шум меньше единицы.

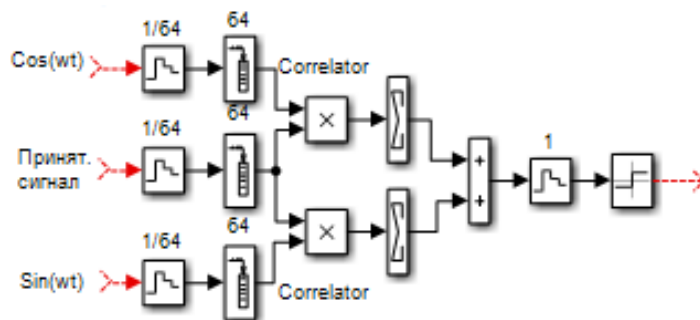


Рисунок 5.9 – Корреляторы каналов приема

Каждый коррелятор в данном случае выдает отсчет корреляционной функции $R(\tau)$ текущего бита при смещении $\tau = 0$. На выходе каналов обработки используется комбинация блоков *Zero Order Hold* и *Sign* для регенерации формы принимаемых битов. Выходные сигналы каналов обработки суммируются блок *Sum*, так как для передачи I и $-I$ использовались фазовые состояния $(\pi/4, -3\pi/4)$. Далее принятый сигнал поступает для визуализации на блок *Scope* и для контроля ошибок передачи на подсистему детектора ошибок *Detect Err*.

В качестве фазового декодера используется сумматор блок *Sum*, если в фазовом кодере значениям битов ± 1 устанавливались фазовые состояния $(\pi/4, -3\pi/4)$. В качестве фазового декодера используется вычитатель блок *Sum*, если в фазовом кодере значениям битов ± 1 устанавливались фазовые состояния $(-\pi/4, 3\pi/4)$.

На выходе фазового декодера используется комбинация блоков *Zero Order Hold* и *Sign* для регенерации формы принимаемых битов. Далее принятый сигнал поступает для визуализации на блок *Scope* и для контроля ошибок передачи на подсистему детектора ошибок *Detect Err*.

5.2 Описание моделей модемов на основе BFSK модуляции

Модуляция **BFSK** позволяет реализовать различные подвиды **BFSK** модуляции, включая варианты со сложением (вычитанием) каналов обработки и варианты **MSK** модуляции с минимальным сдвигом не имеющие скачков фаз, что позволяет снизить требования к линейности усилителей мощности передатчика. Основным недостатком **FSK** модуляции, является наличие так называемых разностных продуктов, которые не подавляются должным образом **ФНЧ** демодулятора и существенно снижают помехоустойчивость модемов на их основе.

Описание модели. На рисунке 5.10 и в приложении Б (рисунок Б.1), приведена функциональная модель классической реализации **BFSK** модема *Class_BFSK_Modem_1_Sum_New Sin(15pi*t)_ Sin(16pi*t)* с приемником прямого преобразования. Модуляция **BFSK** позволяет реализовать различные подвиды **BFSK**

модуляции, включая варианты со сложением (вычитанием) каналов обработки и варианты **MSK** модуляции с минимальным сдвигом не имеющие скачков фаз, что позволяет снизить требования к линейности усилителей мощности передатчика. Основным недостатком **FSK** модуляции, является наличие так называемых разностных продуктов, которые не подавляются должным образом **ФНЧ** демодулятора и существенно снижают помехоустойчивость модемов на их основе.

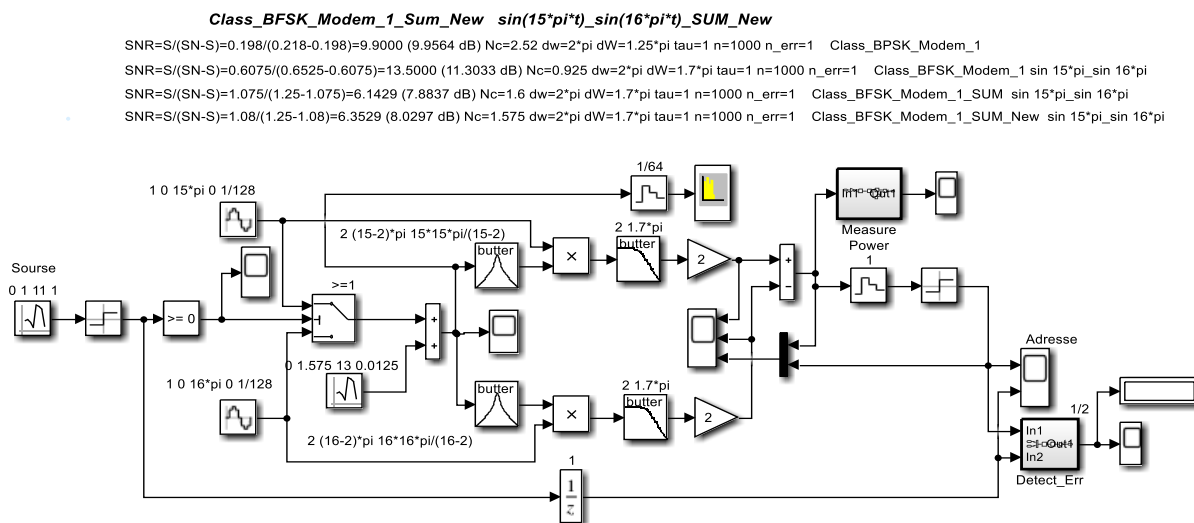


Рисунок 5.10 - Функциональная модель **BFSK** модема с суммированием канала обработки и прямым преобразованием спектра

Структура и принцип работы **BFSK** модема с приемником прямого преобразования и сложением (вычитанием) каналов обработки.

Источник информационной последовательности выполнен на основе генератора дискретного псевдослучайного процесса **Random Number** с шагом равным длине будущего бита τ_b . Псевдослучайный процесс после двухстороннего ограничителя на основе функции **sign** превращается в биполярную псевдослучайную информационную последовательность. С помощью блока **Relational Operator** биполярная последовательность превращается в однополярную.

BFSK модулятор выполнен на основе двухпозиционного переключателя **Switch**, управляемого информационной последовательностью и переключающего гармонические генераторы **Sine Wave** и **Sine Wave1**. Обычно разнос несущих частот генераторов выбирается из условия ортогональности $\Delta\omega = |\omega_2 - \omega_1| = 2\pi k / (2\tau)$, при когерентном приеме и $\Delta\omega = |\omega_2 - \omega_1| = 2\pi k / \tau$ при некогерентном приеме, где $k = 1, 2, \dots$. В нашем случае при $\tau_b = 1$ использованы несущие колебания с частотами $\omega_1 = 15\pi$ и $\omega_2 = 16\pi$.

При моделировании в **Simulink** генераторы передатчика и приемники строго синхронизированы по фазе, что соответствует когерентному приему. Модулированный **BFSK** сигнал представляет собой послышки несущих колебаний, чередование частот которых, определяется информационной последовательностью.

Модель канала распространения выполнена на основе сумматора **Sum**, на первый вход которого и поступает **BFSK** модулированный сигнал. На второй вход поступают шумы с генератора псевдослучайной последовательности **Random Number**. Изменяя параметр **Sample Time**, управляем широкополосностью шума, а параметр **Variance** (дисперсия) соответствует мощности, что позволяет изменять соотношение сигнал/шум при исследовании помехоустойчивости.

Приемник прямого преобразования. Приемная часть представляет собой два канала обработки на каждую несущую частоту. На входе канала стоит полосовой фильтр (**RF**) *Analog Filter Design* для фильтрации соответствующей несущей частоты. Демодулятор выполнен на умножителе **Product** (смесителе), на один вход которого подается принятый фильтрованный частотно-манипулированный сигнал. На второй вход умножителя поступает колебание соответствующей частоты с опорного генератора. На выходе умножителя стоит **ФНЧ (LF)**, фильтрующий высокочастотные составляющие демодулятора. Блок **Gain** с параметром **Gain=2** поставлен для компенсации коэффициента тригонометрических преобразований демодулятора, равного **1/2**.

Вариант схемы со сложением (вычитанием). Сигналы с каналов обработки после **ФНЧ (LF)** и усилителей **Gain** подаются на вычитатель (сумматор) **Sum**. С выхода сумматора объединенный демодулированный сигнал подается на блок **Zero Order Hold**, нормирующий блок **sign** и на осциллограф **Scope** и подсистему детектора ошибок **Detect_Err**. На выходе детектора ошибок включен **display** и осциллограф для визуализации момента возникновения ошибок.

Для исследования помехоустойчивости модема подсистема измерения мощности сигнала и шумов **Measure Power** подключается после вычитателя (блок **Sum**), на входе блока сравнения **Relational Operator**.

Для обеспечения визуального сравнения и возможности вычисления ошибок на вторые входы осциллографа и детектора ошибок подается задержанная исходная информационная последовательность.

Вычитание каналов обработки в **BFSK** модемах существенно улучшает его помехоустойчивость. Это связано с тем, что, вычитая каналы, мы не изменяем мощность сигнала, так как сигналы в каналах существуют попеременно. Шумы в каналах имеют общее происхождение - это канал распространения и в каналах обработки они сильно коррелированы, но при вычитании оказываются сильно антикоррелированы и при вычитании мощность шумов даже уменьшается за счет взаимокорреляционной составляющей и соотношение сигнал/шум увеличивается.

Еще раз отметим, что случае **BFSK** модуляции появляются так называемые разностные продукты, которые практически без ослабления проходят через **ФНЧ** и существенно ухудшают помехоустойчивость.

Описание модели. На рисунке 5.11 и в приложении Б (рисунок Б.2), приведена функциональная модель классической реализации **BFSK** модема **Class_BFSK_Modem_2_Sum_Corr_New Sin(15pi*t)_ Sin(16pi*t)** с корреляционным приемником. Модуляция **BFSK** позволяет реализовать различные подвиды **BFSK** модуляции, включая варианты со сложением (вычитанием) каналов обработки и варианты **MSK** модуляции с минимальным сдвигом не имеющие скачков фаз, что позволяет снизить требования к линейности усилителей мощности передатчика. Основным недостатком **FSK** модуляции, является наличие так называемых разностных продуктов, которые не подавляются должным образом **ФНЧ** демодулятора и существенно снижают помехоустойчивость модемов на их основе.

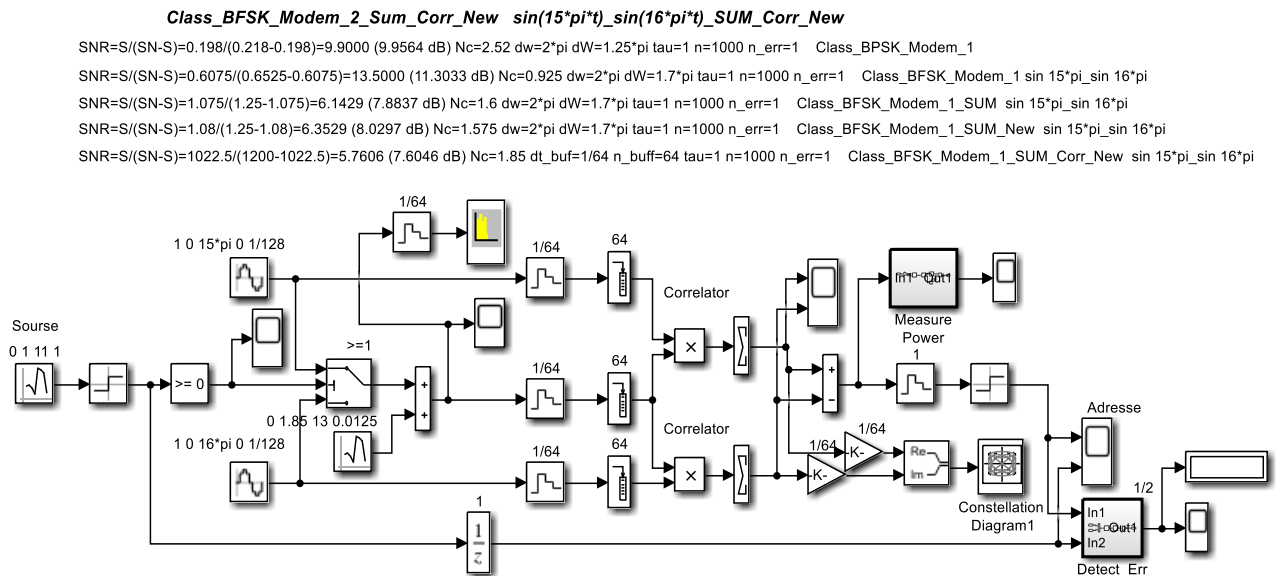


Рисунок 5.11 - Функциональная модель **BFSK** модема с суммированием канала обработки и корреляционным приемником

Структура и принцип работы *BFSK* модема с приемником прямого преобразования и сложением (вычитанием) каналов обработки.

Источник информационной последовательности выполнен на основе генератора дискретного псевдослучайного процесса *Random Number* с шагом равным длине будущего бита τ_b . Псевдослучайный процесс после двухстороннего ограничителя на основе функции *sign* превращается в биполярную псевдослучайную информационную последовательность. С помощью блока *Relational Operator* биполярная последовательность превращается в однополярную.

BFSK модулятор выполнен на основе двухпозиционного переключателя *Switch*, управляемого информационной последовательностью и переключающего гармонические генераторы *Sine Wave* и *Sine Wave1*. Обычно разнос несущих частот генераторов выбирается из условия ортогональности $\Delta\omega = |\omega_2 - \omega_1| = 2\pi k / (2\tau)$, при когерентном приеме и $\Delta\omega = |\omega_2 - \omega_1| = 2\pi k / \tau$ при некогерентном приеме, где $k = 1, 2, \dots$. В нашем случае при $\tau_b = 1$ использованы несущие колебания с частотами $\omega_1 = 15\pi$ и $\omega_2 = 16\pi$.

При моделировании в *Simulink* генераторы передатчика и приемники строго синхронизированы по фазе, что соответствует когерентному приему. Модулированный *BFSK* сигнал представляет собой послышки несущих колебаний, чередование частот которых, определяется информационной последовательностью.

Модель канала распространения выполнена на основе сумматора *Sum*, на первый вход которого и поступает *BFSK* модулированный сигнал. На второй вход поступают шумы с генератора псевдослучайной последовательности *Random Number*. Изменяя параметр *Sample Time*, управляем широкополосностью шума, а параметр *Variance* (дисперсия) соответствует мощности, что позволяет изменять соотношение сигнал/шум при исследовании помехоустойчивости.

Корреляционный приемник – приемная часть модема реализована по принципу корреляционного приема. В каждом канале несущей стоит по коррелятору. Коррелятор (см. рис. 5.12) имеет два плеча накопления дискретизированных отсчетов блоками *Zero Order Hold* отсчетов принятого в шумах радиосигнала и отсчетов гармонических колебаний синхронизированного с передающей частью опорного генератора. Накопление отсчетов

производится в буферах блок **Buffer**. Поскольку предполагается побитовый прием, то, например, шаг дискретизации берется равным $\Delta t = 1/64$, а размер буфера равен $N = 64$, тогда за время действия бита накапливается вектор размерностью **64**. Далее блоками **Product** и **Sum** выполняется скалярное произведение векторов отсчетов. Даже при наличии помех при приеме бита равного I на выходе сумматора будет накапливаться нарастающий положительный всплеск, а при приеме бита равного $-I$ на выходе сумматора будет накапливаться нарастающий отрицательный всплеск. Корреляционный прием сигнала возможен при отношении сигнал/шум меньше единицы.

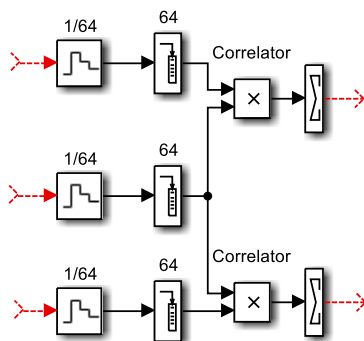


Рисунок 5.12 – Корреляторы каналов приема

Каждый коррелятор в данном случае выдает отсчет корреляционной функции $R(\tau)$ текущего бита при смещении $\tau = 0$. Сигналы с корреляторов поступают на вычитатель блок **Sum**, а далее на схему масштабирования – блоки **Zero Order Hold** и **sign**.

Далее включены осциллограф **Scope** и подсистема детектора ошибок **Detect_Err**. На выходе детектора ошибок включен **display** и осциллограф для визуализации момента возникновения ошибок.

Для обеспечения визуального сравнения и возможности вычисления ошибок на вторые входы осциллографа и детектора ошибок подается задержанная исходная информационная последовательность.

Для обеспечения исследования помехоустойчивости модема на выходе вычитателя, т.е. на входе блока **Zero Order Hold** включена подсистема измерения мощности сигнала и шумов **Measure_Power**.

Вариант схемы со сложением (вычитанием).

Сигналы с корреляторов каналов обработки на вычитатель (сумматор) **Sum**. С выхода сумматора объединенный демодулированный сигнал подается на блок **Zero Order Hold**, нормирующий блок **sign** и на осциллограф **Scope** и подсистему детектора ошибок **Detect_Err**. На выходе детектора ошибок включен **display** и осциллограф **Scope** для визуализации момента возникновения ошибок.

Для исследовании помехоустойчивости модема подсистема измерения мощности сигнала и шумов **Measure_Power** подключается после вычитателя (блок **Sum**), на входе блока сравнения **Relational Operator**.

Для обеспечения визуального сравнения и возможности вычисления ошибок на вторые входы осциллографа и детектора ошибок подается задержанная исходная информационная последовательность.

Вычитание каналов обработки в **BFSK** модемах существенно улучшает его помехоустойчивость. Это связано с тем, что, вычитая каналы, мы не изменяем мощность

сигнала, так как сигналы в каналах существуют попеременно. Шумы в каналах имеют общее происхождение - это канал распространения и в каналах обработки они сильно коррелированы, но при вычитании оказываются сильно антикоррелированы и при вычитании мощность шумов даже уменьшается за счет взаимокорреляционной составляющей и соотношение сигнал/шум увеличивается.

В случае **BFSK** модуляции появляются так называемые разностные продукты, которые практически без ослабления проходят через **ФНЧ** и существенно ухудшают помехоустойчивость.

5.3 Описание моделей модемов на основе QPSK модуляции

Следующей версией **PSK** модуляции является квадратурная фазовая манипуляция **QPSK** модуляция. Скорость передачи в два раза выше, чем у **BPSK**, передача одного символа эквивалентна передаче двух битов [8].

При квадратурной модуляции **QPSK** из текущих информационных битов формируются дибиты, каждому дибиту программно ставится в соответствие фазовое состояние φ_k и далее формируются модулирующие символы квадратурного модулятора $\cos(\varphi_k)$ и $\sin(\varphi_k)$.

Так как модулирующие символы имеют длительность 2τ , то по сравнению **BPSK** модуляцией ширина спектра оказывается в два раза уже, а энергия управляющего символа в два раза больше. Межсигнальное расстояние вместо $2E$ для **BPSK** становится $\sqrt{2}E$ для **QPSK**.

Описание модели. На рисунке 5.13 и в приложении В (рисунок В.1), приведена **Sim**-модель **Class_QPSK_Modem_1** классического **QPSK** модема с приемником демодулятором на основе прямого преобразования (синхронного детектирования).

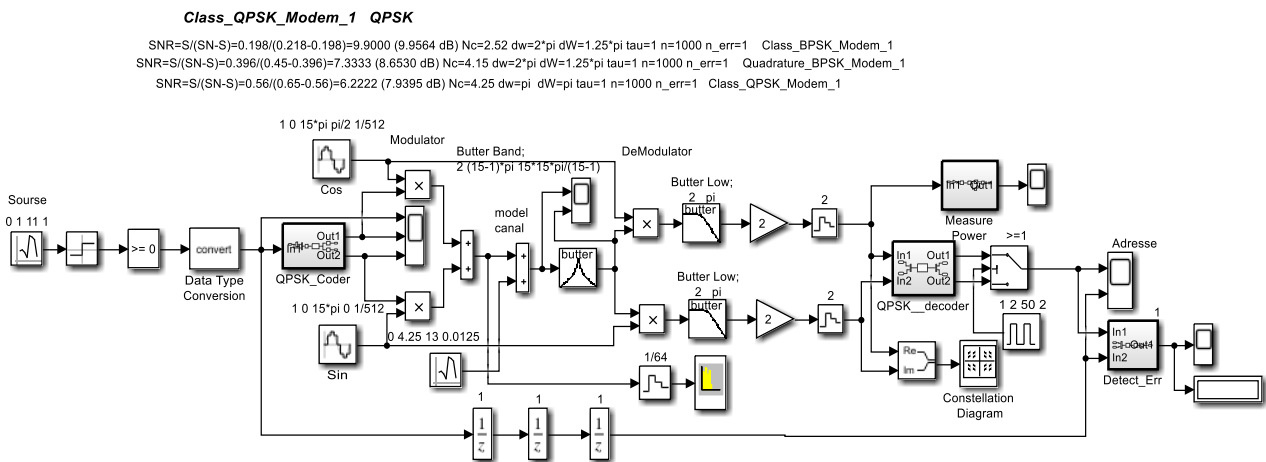


Рисунок 5.13 - **Sim**-модель классического **QPSK** модема с приемником демодулятором на основе прямого преобразования

Структура и принцип работы **QPSK** модема.

Источник информационной последовательности выполнен на основе генератора дискретного псевдослучайного процесса **Random Number** с шагом равным длине будущего бита τ . Псевдослучайный процесс после двухстороннего ограничителя на основе функции **sign** превращается в биполярную псевдослучайную информационную последовательность. С помощью блока **Compare To Zero** биполярная последовательность превращается в

однополярную. Блок **Data Type Conversion** переводит логический тип **boolean** после операции сравнения в тип **double**.

Далее однополярный поток поступает на подсистему фазового кодера **QPSK_Coder** рис. 5.14.

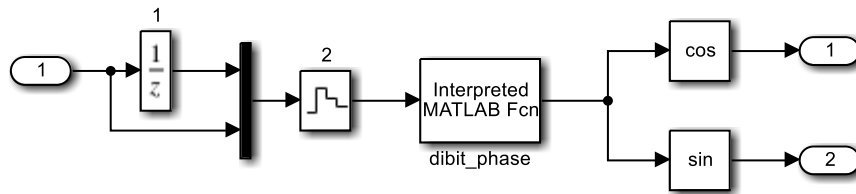


Рисунок 5.14 – Функциональная модель фазового кодера **QPSK_Coder**

С помощью блоков задержки **Unit Delay** и мультиплексора **Mux** входная битовая последовательность превращается в последовательность дибитов (вектор из двух битов, текущий и последующий). Блок **Zero Order Hold** с параметром шага равного 2τ запрещает повторное использование бита в другом дибите (остаются дибиты **1-2, 3-4, 5-6** и т.д.). Далее дибит подается на программный блок **MatLab Function**, содержащий функцию **dibit_phase**, скрипт файл которой приведен ниже (см. рис. 5.15).

```
function fi=dibit_phase(x);
% Преобразование дибита в фазу
% x- вектор (дибит)
% fi- фаза

if x==[0; 0]; fi=pi/4; end;
if x==[0; 1]; fi=3*pi/4; end;
if x==[1; 1]; fi=-3*pi/4; end;
if x==[1; 0]; fi=-pi/4; end;
```

Рисунок 5.15 – Функция преобразования дибита в фазу **dibit_phase**

Данная функция каждому значению дибита ставит в соответствие значение фазы φ_k .

Далее значения $\cos(\varphi_k)$ и $\sin(\varphi_k)$ поступают на выход подсистемы фазового кодера и на входы умножителей **product** квадратурного модулятора. На вторые входы умножителей подаются квадратурные колебания несущей частоты $\cos(\omega t)$ и $\sin(\omega t)$ (на практике используют один генератор несущей и фазовращатель на $\pi/2$ радиан, получая соответственно, $\cos(\omega t)$ и $-\sin(\omega t)$). Квадратурный модулятор завершается суммированием квадратурных составляющих умножителей.

Далее следует простейшая модель канала распространения, состоящая из сумматора **Sum** и генератора шумов канала **Random Number**. Изменяя параметр **Sample Time**, управляем широкополосностью шума, а параметр **Variance** (дисперсии) соответствует мощности, что позволяет изменять соотношение сигнал/шум при исследовании помехоустойчивости.

После модели канала распространения включен полосовой фильтр **ПФ (RF)** призванный отобразить фильтрацию внеполосных излучений на выходе передатчика и входе приемника.

Прием основан на принципе прямого преобразования или синхронного детектирования, когда в приемнике частота (гетеродина) опорного генератора совпадает с частотой несущей сигнала. Приемник состоит из двух квадратурных каналов обработки. На входе каждого канала обработки стоит демодулятор на основе умножителя. На один вход умножителя подается фильтрованный входной сигнал, а на другой вход синхронизированный опорный генератор квадратуры. На выходе умножителя используется **LF** фильтр для фильтрации высокочастотных составляющих и усилитель **Gain = 2**, для компенсации коэффициента тригонометрических преобразований. Далее использовано устройство взятия отсчета в фиксированные моменты времени на основе блока **Zero Order Hold**.

Сигналы с квадратурных каналов обработки поступают на вход подсистемы фазового декодера **QPSK_DeCoder**. Функциональная модель подсистемы фазового декодера приведена на рис. 5.16.

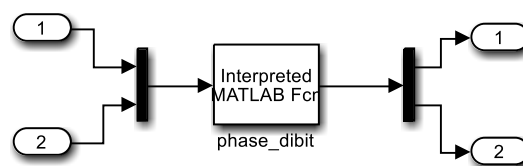


Рисунок 5.16 – Функциональная модель подсистемы фазового декодера **QPSK_DeCoder**

На вход подсистемы поступают квадратурные проекции принятых фазовых кодовых символов $\cos(\varphi_k)$ и $\sin(\varphi_k)$. Блок мультиплексора **Mux** создает из проекций векторное представление и подает его на вход блока **MatLab Function**, содержащего функцию **phaze_dibit**, скрипт файл которой приведен ниже(см. рис. 5.17) .

```
function y=phaze_dibit(x);
% Преобразование фазы в виде проекций в дибиты
% x- вектор (проекции cos и sin)

if ((x(1)==0) & (x(2)==0)); y=[0 0]; end;
if ((x(1)>=0) & (x(2)>=0)); y=[0 0]; end;
if ((x(1)<0) & (x(2)>=0)); y=[0 1]; end;
if ((x(1)<0) & (x(2)<0)); y=[1 1]; end;
if ((x(1)>=0) & (x(2)<0)); y=[1 0]; end;
```

Рисунок 5.17 – Функция преобразования фазы в дибиты **phaze_dibit**

Данная функция по значениям квадратурных составляющих восстанавливает дибит, и через демультимплексор **DeMux** разделяет его на составляющие биты длиной в дибит.

С выхода подсистемы биты поступают на двухпортовый переключатель **Switch**, управляемый импульсным генератором **Pulse Generator**. В первую половину дибита снимается первый бит, во вторую половину дибита - второй бит, таким образом блок **Switch** является преобразователем параллельного (векторного) представления в последовательное (скалярное).

На выходе преобразователя из параллельного представления в последовательное включены осциллограф **Scope** и подсистема детектора ошибок **Detect_Err**. На выходе

детектора ошибок включен *display* и осциллограф *Scope* для визуализации момента возникновения ошибок.

Для обеспечения визуального сравнения и возможности вычисления ошибок на вторые входы осциллографа и детектора ошибок подается задержанная исходная информационная последовательность.

Для обеспечения исследования помехоустойчивости модема на выходах *LF* каналов обработки демодуляторов, т.е. на входах блоков взятия отсчетов *Zero Order Hold Hold* могут быть включены подсистемы измерения мощности *Measure_Power* сигнала и шумов.

Описание модели. В рисунке 5.18 и в приложении В (рисунок В.2), приведена *Sim*-модель *Class_QPSK_Modem_2_Corr* классического *QPSK* модема с корреляционным приемником демодулятором.

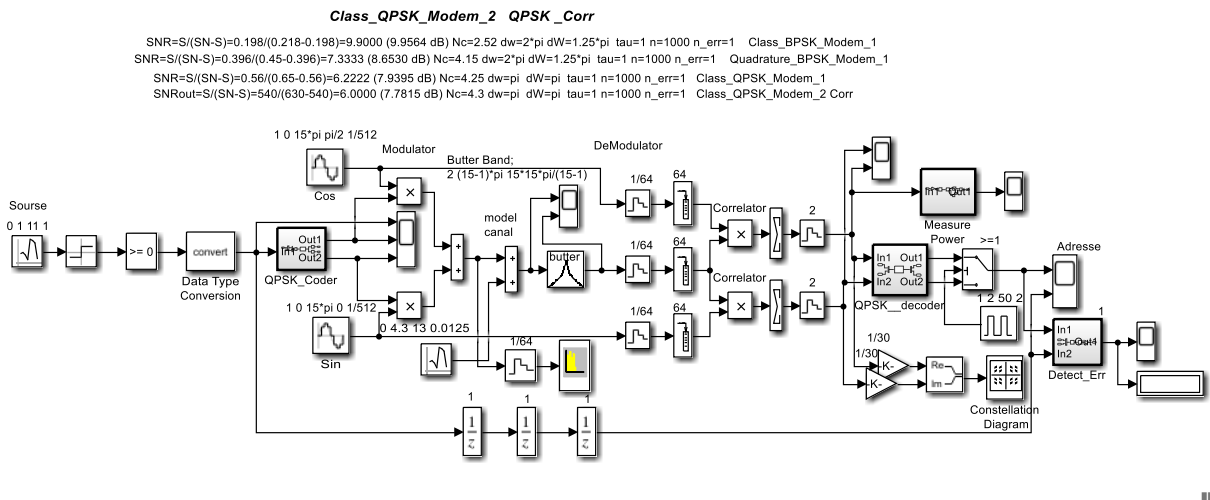


Рисунок 5.18 - *Sim*-модель классического *QPSK* модема с корреляционным приемником демодулятором

Структура и принцип работы *QPSK* модема с коррелятором.

Источник информационной последовательности выполнен на основе генератора дискретного псевдослучайного процесса *Random Number* с шагом равным длине будущего бита τ . Псевдослучайный процесс после двухстороннего ограничителя на основе функции *sign* превращается в биполярную псевдослучайную информационную последовательность. С помощью блока *Compare To Zero* биполярная последовательность превращается в однополярную. Блок *Data Type Conversion* переводит логический тип *boolean* после операции сравнения в тип *double*.

Далее однополярный поток поступает на подсистему фазового кодера *QPSK_Coder* рис. 5.19.

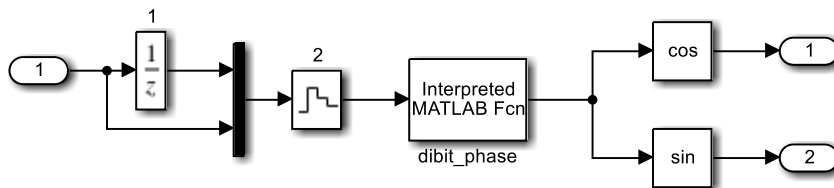


Рисунок 5.19 – Функциональная модель фазового кодера *QPSK_Coder*

С помощью блоков задержки **Unit Delay** и мультиплексора **Mux** входная битовая последовательность превращается в последовательность дибитов (вектор из двух битов, текущий и последующий). Блок **Zero Order Hold** с параметром шага равного 2τ запрещает повторное использование бита в другом дибите (остаются дибиты **1-2, 3-4, 5-6** и т.д.). Далее дибит подается на программный блок **MatLab Function**, содержащий функцию **dibit_phase**, скрипт файл которой приведен ниже (см. рис. 5.20).

```
function fi=dibit_phase(x);
% Преобразование дибита в фазу
% x- вектор (дибит)
% fi- фаза

if x==[0; 0]; fi=pi/4; end;
if x==[0; 1]; fi=3*pi/4; end;
if x==[1; 1]; fi=-3*pi/4; end;
if x==[1; 0]; fi=-pi/4; end;
```

Рисунок 5.20 – Функция преобразования дибита в фазу **dibit_phase**

Данная функция каждому значению дибита ставит в соответствие значение фазы φ_k .

Далее значения $\cos(\varphi_k)$ и $\sin(\varphi_k)$ поступают на выход подсистемы фазового кодера и на входы умножителей **product** квадратурного модулятора. На вторые входы умножителей подаются квадратурные колебания несущей частоты $\cos(\omega t)$ и $\sin(\omega t)$ (на практике используют один генератор несущей и фазовращатель на $\pi/2$ радиан, получая соответственно, $\cos(\omega t)$ и $-\sin(\omega t)$). Квадратурный модулятор завершается суммированием квадратурных составляющих умножителей.

Далее следует простейшая модель канала распространения, состоящая из сумматора **sum** и генератора шумов канала **Random Number**. Изменяя параметр **Sample Time**, управляем широкополосностью шума, а параметр **Variance** (дисперсии) соответствует мощности, что позволяет изменять соотношение сигнал/шум при исследовании помехоустойчивости.

После модели канала распространения включен полосовой фильтр **ПФ (RF)** призванный отобразить фильтрацию внеполосных излучений на выходе передатчика и входе приемника.

Корреляционный прием квадратурных составляющих осуществляется корреляторами (см рис. 5.21).

Каждый коррелятор имеет два плеча накопления дискретизированных отсчетов блоками **Zero Order Hold** отсчетов принятого в шумах радиосигнала и отсчетов гармонических колебаний синхронизированного с передающей частью опорного генератора. Накопление отсчетов производится в буферах блок **Buffer**. Поскольку предполагается побитовый прием, то, например, шаг дискретизации берется равным $\Delta t = 1/64$, а размер буфера равен $N = 64$, тогда за время действия бита накапливается вектор размерностью **64**. Далее блоками **Product** и **Sum** выполняется скалярное произведение векторов отсчетов. Даже при наличии помех при приеме бита равного **1** на выходе сумматора будет накапливаться нарастающий положительный всплеск, а при приеме бита равного **-1** на выходе сумматора будет накапливаться нарастающий отрицательный всплеск. Использование блока **Sign** нормирует принимаемые значения всплесков до ± 1 . Корреляционный прием сигнала возможен при отношении сигнал/шум меньше единицы.

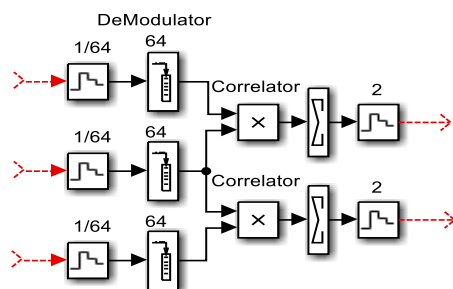


Рисунок 5.21 – Подсистема корреляционной обработки квадратурных составляющих *QPSK* модема

Далее сигналы с квадратурных каналов обработки поступают на вход подсистемы фазового декодера *QPSK_DeCoder*. Функциональная модель подсистемы фазового декодера приведена на рис. 5.22.

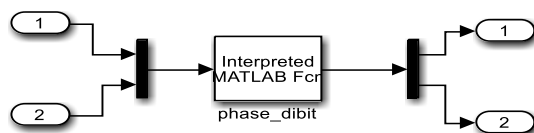


Рисунок 5.22 – Функциональная модель подсистемы фазового декодера *QPSK_DeCoder*

На вход подсистемы поступают квадратурные проекции принятых фазовых кодовых символов $\cos(\varphi_k)$ и $\sin(\varphi_k)$. Блок мультиплексора *Mux* создает из проекций векторное представление и подает его на вход блока *MatLab Function*, содержащего функцию *phaze_dibit*, скрипт файл которой приведен ниже (см. рис. 5.23).

Данная функция по значениям квадратурных составляющих восстанавливает дибит, и через демультимплексор *DeMux* разделяет его на составляющие биты длиной в дибит.

С выхода подсистемы биты поступают на двухпортовый переключатель *Switch*, управляемый импульсным генератором *Pulse Generator*. В первую половину дибита снимается первый бит, во вторую половину дибита второй бит, таким образом блок *Switch* является преобразователем параллельного (векторного) представления в последовательное (скалярное).

На выходе преобразователя из параллельного представления в последовательное включены осциллограф *Scope* и подсистема детектора ошибок *Detect_Err*. На выходе детектора ошибок включен *display* и осциллограф *Scope* для визуализации момента возникновения ошибок.

```
function y=phaze_dibit(x);
% Преобразование фазы в виде проекций в дибиты
% x- вектор (проекции cos и sin)

if ((x(1)==0) & (x(2)==0)); y=[0 0]; end;
if ((x(1)>=0) & (x(2)>=0)); y=[0 0]; end;
if ((x(1)<0) & (x(2)>=0)); y=[0 1]; end;
if ((x(1)<0) & (x(2)<0)); y=[1 1]; end;
if ((x(1)>=0) & (x(2)<0)); y=[1 0]; end;
```

Рисунок 5.23 – Функция преобразования фазы в дибиты *phaze_dibit*

Для обеспечения визуального сравнения и возможности вычисления ошибок на вторые входы осциллографа и детектора ошибок подается задержанная исходная информационная последовательность.

Для обеспечения исследования помехоустойчивости модема на выходах **LF** каналов обработки демодуляторов, т.е. на входах блоков взятия отсчетов **Zero Order Hold** могут быть включены подсистемы измерения мощности **Measure_Power** сигнала и шумов.

Математическое описание QPSK модема с корреляционным приемом

Дадим простое математическое описание **QPSK** модуляции, используя структуру модема. Информационный поток битов поступает на фазовый кодер, который формирует из них дибиты, ставит им в соответствие фазовое состояние φ_k и, используя функции **cos** и **sin**, преобразует их в управляющие символы модулятора. Управляющими символами, поступающими с фазового кодера на квадратурный модулятор являются $d_I = \cos(\varphi_k)$ и $d_Q = \sin(\varphi_k)$. Рассмотрим квадратурный модулятор.

Синфазный (**Inphase**) канал модулятора

$$I_k = \cos(\varphi_k) \cdot \cos(\omega t) = \frac{1}{2} \cdot \langle \cos(\varphi_k - \omega t) + \cos(\omega t + \varphi_k) \rangle.$$

Квадратурный (**Quadrature**) канал модулятора

$$Q_k = \sin(\varphi_k) \cdot \sin(\omega t) = \frac{1}{2} \cdot \langle \cos(\varphi_k - \omega t) - \cos(\omega t + \varphi_k) \rangle.$$

После суммирования квадратур на выходе модулятора имеем

$$\begin{aligned} U_k &= d_I \cdot \cos(\omega t) + d_Q \cdot \sin(\omega t) = \\ &= \cos(\varphi_k) \cdot \cos(\omega t) + \sin(\varphi_k) \cdot \sin(\omega t) = \cos(\varphi_k - \omega t). \end{aligned}$$

На выходе модели канала распространения имеем

$$S_k = \cos(\varphi_k - \omega t) + n_k.$$

Рассмотрим квадратурный демодулятор.

Синфазный (**Inphase**) канал демодулятора

$$\langle \cos(\varphi_k - \omega t) + n_k \rangle \cdot \cos(\omega t) = \frac{1}{2} \cdot \langle \cos(\varphi_k) + \cos(2\omega t - \varphi_k) + n_k \cdot \cos(\omega t) \rangle.$$

Квадратурный (**Quadrature**) канал демодулятора

$$\langle \cos(\varphi_k - \omega t) + n_k \rangle \cdot \sin(\omega t) = \frac{1}{2} \cdot \langle \sin(\varphi_k) + \sin(2\omega t - \varphi_k) + n_k \cdot \sin(\omega t) \rangle.$$

После прохождения через **ФНЧ** отфильтровывающего высокочастотные составляющие на выходах квадратурных каналов обработки оказываются модулирующие символы $d_I = \cos(\varphi_k)$ и $d_Q = \sin(\varphi_k)$, которые фазовый декодер преобразует в дибиты, а преобразователь параллельного представления в последовательное выдает поток принятых информационных битов.

Пример представления результатов модельного исследования QPSK модема

Для исследования модема, рассмотрим функциональную схему на рисунке 5.24, ниже. Блоки отмеченные красным, это блоки *Scope*, с которых в дальнейшем будут сниматься осциллограммы. Так же на схеме имеется метка для второго осциллографа слева, что для правильной осциллограммы, нужно отключить генератор внешних шумов. В свою очередь генератор внешних шумов помечен фиолетовым цветом.

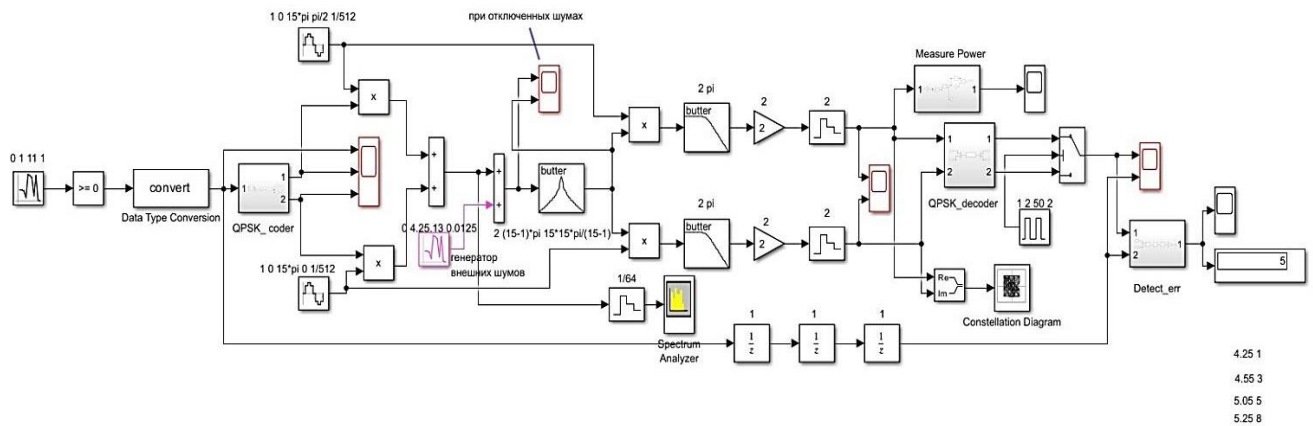


Рисунок 5.24 – Функциональная схема исследования *QPSK* модема с приемником прямого преобразования

Для иллюстрации работы *QPSK* модема приведены фрагменты осциллограмм передаваемого информационного потока и сформированных модулирующих символов.

Осциллограмма на рисунке 5.25 снята с первого блока *Scope*, и содержит три графика. На самом верхнем графике изображен сигнал, приходящий на вход кодера "*QPSK_coder*", на среднем и нижнем графике представлены косинусоидальная и синусоидальная части сигнала.

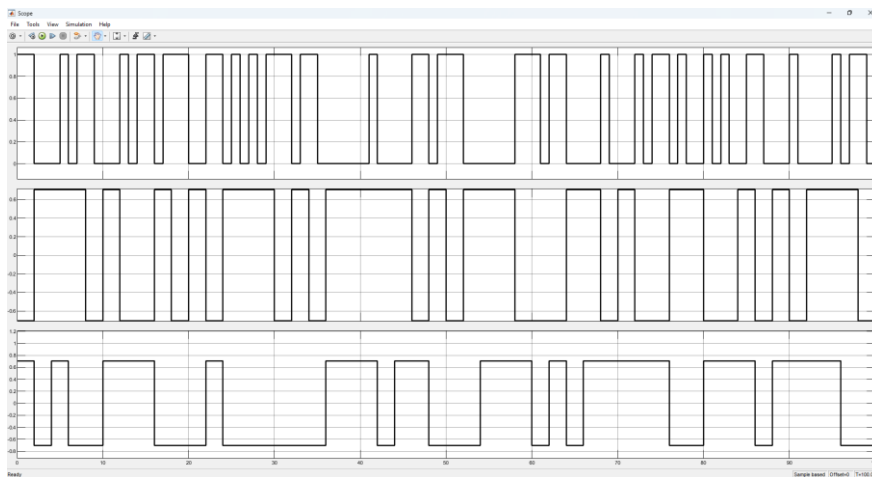


Рисунок 5.25 - Фрагменты осциллограмм передаваемого информационного потока и сформированных модулирующих символов

На рисунке 5.26 представлены графики радиосигнала с отключёнными шумами до и после полосового фильтра.

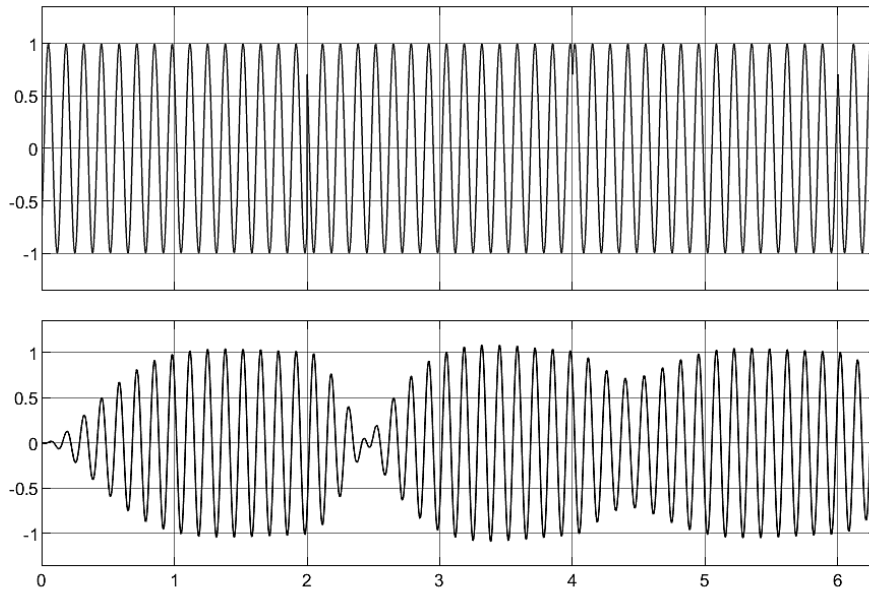


Рисунок 5.26 – Фрагменты осциллограмм квадратурно-модулированного сигнала *QPSK* до и после полосового фильтра в отсутствии шумов канала распространения

На рисунке 5.27 приведен фрагмент осциллограмм квадратурных потоков на выходе квадратурного демодулятора в шумах.

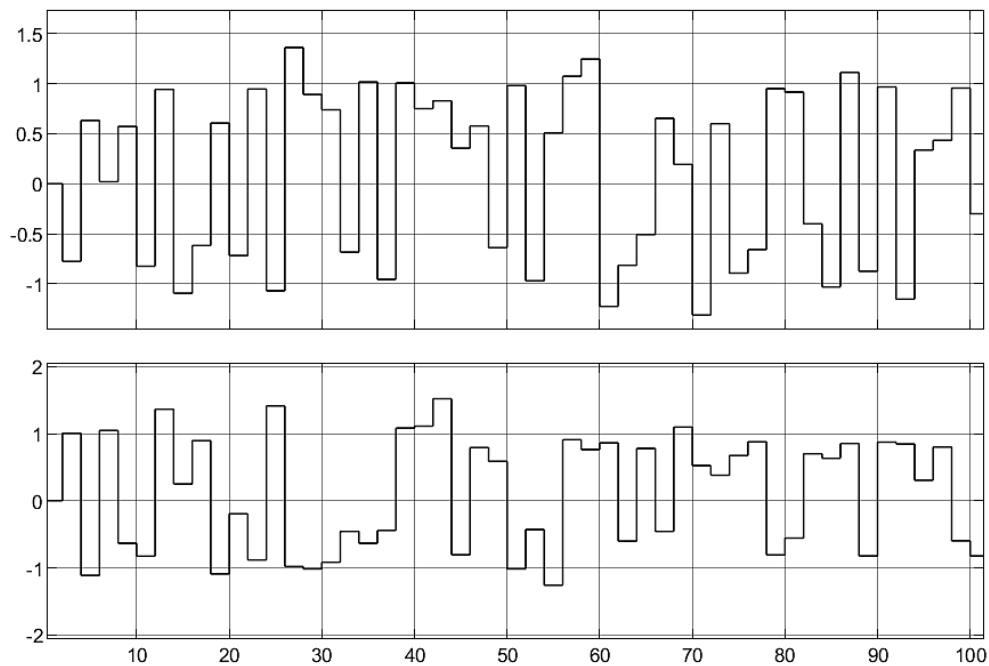


Рисунок 5.27 – Фрагменты принятых осциллограмм квадратурных потоков на выходе квадратурного демодулятора в шумах канала распространения

На рисунке 5.28 приведены фрагменты осциллограмм принятого и передаваемого информационных потоков.

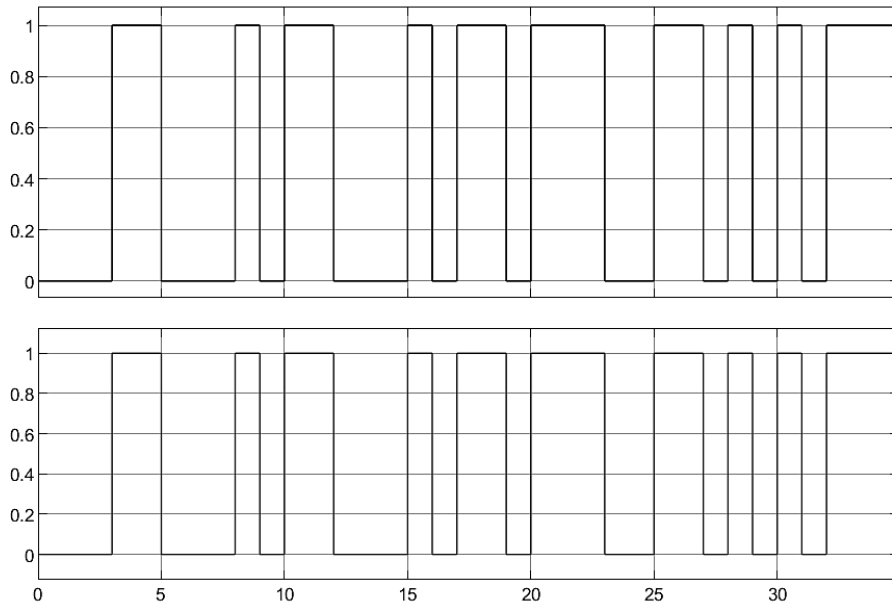


Рисунок 5.28 – Фрагменты осциллограмм принятого и передаваемого информационных потоков в отсутствии шумов канала распространения

Приведём диаграмму фазовых переходов в приёмнике *QPSK* модема в отсутствии шумов канала распространения (см. рис. 5.29).

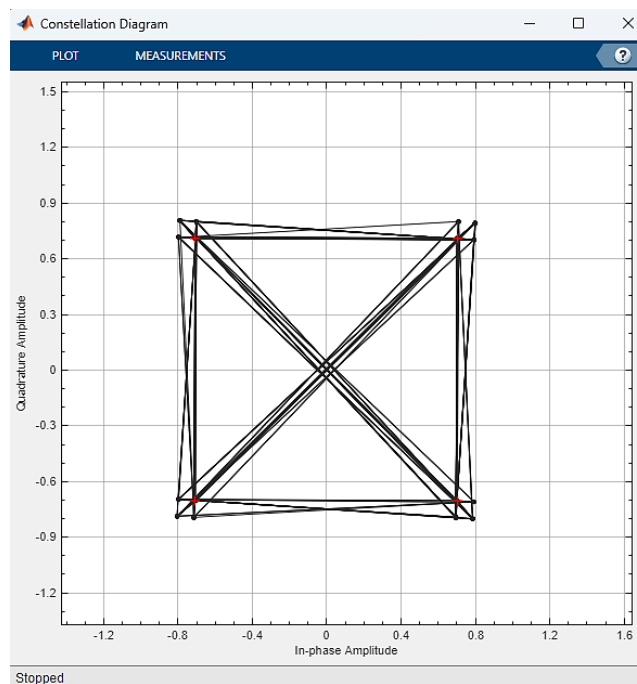


Рисунок 5.29 – Диаграмма фазовых переходов в приёмнике *QPSK* модема в отсутствии шумов канала распространения

После представления сигналов в узловых точках модема, приступим к практической части. Так в блоке *Display*, отображается сколько ошибок возникает в исследуемом модеме и при какой дисперсии "*Variance*", генератора шумов канала распространения возникает 1, 3, 5, 8 ошибок при передаче скажем 1000 битов. В данном *QPSK_modem*, с представленными

параметрами дисперсии шумов генератора канала распространения составляют **4.25, 4.55, 5.05, 5.25** соответственно. Значения **SNR** на выходе скажем квадратурного канала демодулятора составляют **7, 6.59, 5.6, 5.09**, как рассчитывать **SNR** показано в теории выше. По значениям **SNR** и частоте появления ошибок (вероятности появления **P_b**) строим в полулогарифмическом масштабе водопадоподобную кривую, как зависимость вероятности битовой ошибки от **SNR**.

Для построения графика воспользуемся кодом ниже.

```
clc; clear all; format compact; %clf;
SNR_dB=[7, 6.59, 5.6, 5.09];
Pb=[1e-3 3e-3 5e-3 8e-3];
figure(1);
semilogy(SNR_dB, Pb, SNR_dB, Pb, 'r*');
title('Zavisimost Pb ot SNR');
xlabel('SNR(dB)');
ylabel('Pb');
grid;
```

Рисунок 5.30 – Код для построения водопадоподобной кривой

График зависимости вероятности появления битовой ошибки **P_b** от **SNR** представлен на рисунке 5.31.

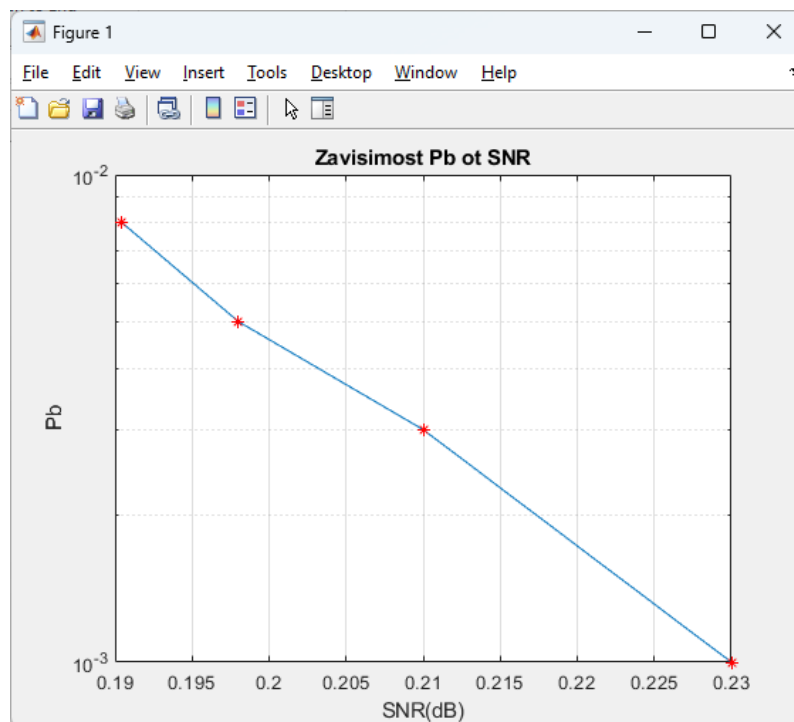


Рисунок 5.31 – График водопадоподобной кривой помехоустойчивости

6 ОПИСАНИЕ ТЕОРИИ И МОДЕЛЕЙ ПОМЕХОУСТОЙЧЕВЫХ КОДЕКОВ И БЛОЧНЫХ КОДОВ

6.1 Синдромное декодирование

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 [9].

Структура и принцип работы двоичного блочного кодера (6, 3) (см. рис.6.1). Кодер это объединение кодера и декодера. Модель кодера для наглядности выполнена на основе битовой последовательности без использования радиоканала, т.е. модуляции/демодуляции несущей и воздействия шумов канала распространения. В связи с этим в модели между кодером и декодером использован имитатор одиночных ошибок, позволяющий вносить ошибку в любой бит каждого кодового символа. Согласно обозначения, код (6, 3) к каждому 3-м информационным битам добавляет 3-и избыточных бита.

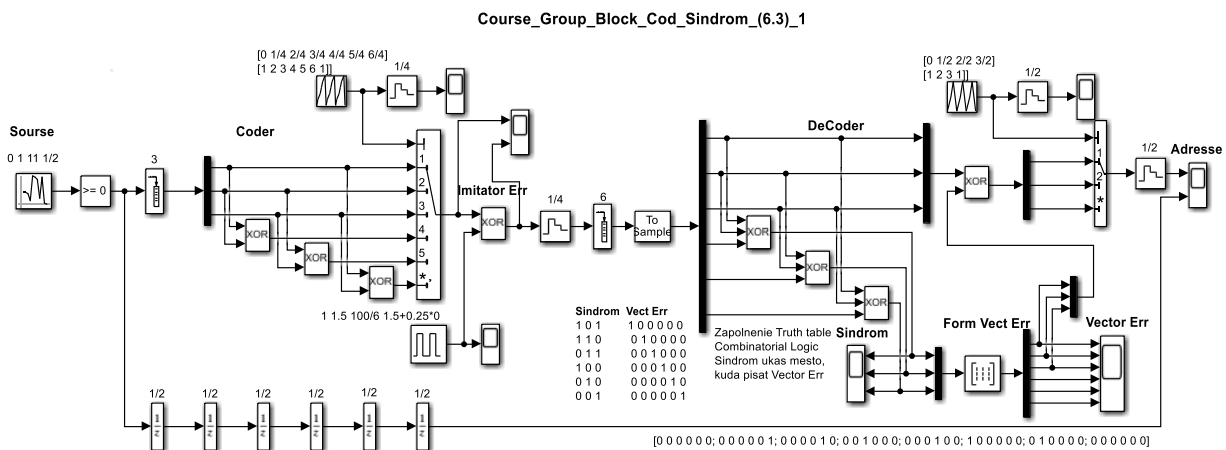


Рисунок 6.1 – Функциональная модель двоичного блочного кодера (6, 3)

Источник информационной битовой последовательности выполнен на основе генератора псевдослучайной последовательности (**ПСП**) **Random Number** с заданным шагом **Sample Time=1/2**. Однополярная битовая последовательность реализуется блоком сравнения **Compare To Zero**. Формирование блока информационных символов реализуется использованием блока **Buffer** с параметром **Output buffer size=3**. Блочный кодер реализуется демультиплексированием информационных битов блоком **DeMux**, а образование избыточных битов реализуется блоками суммирования по модулю 2 **XOR**. Изменение скорости передачи битов кодового символа реализуется блоком **Multiport Switch**, который управляется генератором пилообразного напряжения **Repeating Sequence** с параметрами **Time values: [0 1/4 2/4 3/4 4/4 5/4 6/4]** и **Output values: [1 2 3 4 5 6 1]**, с заданной длиной и крутизной импульсов. Таким образом, блок **Multiport Switch** преобразует параллельное представление кодового символа в последовательный поток выхода кодера.

Имитатором однократных ошибок является блок суммирования по модулю 2 **XOR**, на второй вход которого подаются импульсы необходимой длины, скважности и задержки, которые, суммируясь по модулю 2 с определенным битом каждого символа, вносит ошибку. Так, записывая амплитуду, период, обратное значение скважности, и начальной задержки в виде **1, 1.5, 100/6, 1.5+0.25*0**, получаем ошибки в каждом первом бите кодовых символов.

При смене 0 на 1 в параметре начальной задержки импульсов генератора имитирует ошибки во вторых битах символов, и так далее до 5 .

Декодер начинается с блоков фиксации длительности битов **Zero Order Hold** и **Buffer** - формирователя кодового символа с параметром *Output buffer size=6*. Далее стоит преобразователь из фреймового в векторный тип **Frame Status Conversion** и демультиплексор **DeMux**, битов кодового символа. С помощью блоков **XOR** формируется синдром принятого символа и через мультиплексор **Mux** подается на блок **Combinatorial Logic** с параметром *Truth table: [0 0 0 0 0 0; 0 0 0 0 0 1; 0 0 0 0 1 0; 0 0 1 0 0 0; 0 0 0 1 0 0; 1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 0 0 0 0]*, формирующего по синдрому вектор ошибки, который через мультиплексор подается на блок исправления ошибки информационных битов на основе **XOR**. Строки комбинаторной матрицы нумеруются от нуля до $2^m - 1$, где m - число избыточных битов. Значение синдрома показывает номер строки комбинаторной матрицы, в которую записывается вектор ошибки, не упомянутые в синдромах строки матрицы остаются нулевыми. Вектор синдрома и соответствующий ему фрагмент вектора ошибки информационных битов для визуального контроля выведены на осциллографы. Далее исправленный информационный символ через демультиплексор **DeMux** подается на мультипортовый ключ **Multiport Switch**, управляемый генератором пилообразного напряжения **Repeating Sequence** с параметрами *Time values: [0 1/2 2/2 3/2]* и *Output values: [1 2 3 1]*, для возвращения к первоначальной скорости передачи. С выхода декодера принятый поток битов направляется для наблюдения на двухлучевой осциллограф **Scope**, на второй вход которого с необходимой задержкой подается исходный информационный поток.

Блочное алгебраическое кодирование и синдромное декодирование на примере кода (6, 3). Блочные алгебраические коды являются наиболее красивым разделом дисциплины Теоретические основы систем мобильной связи.

Идея. Входной информационный поток разбивается на блоки (символы или вектора) x длиной k битов. В кодере к каждому информационному блоку добавляются m избыточных (проверочных) битов предназначенных для использования в приемнике с целью обнаружения и исправления ошибок передачи. В результате с выхода кодера в канал подаются уже кодовые символы y длиной $n = k + m$ битов. Блочные коды представляются символически в виде (n, k) .

Для понимания природы и процесса появления избыточных битов рассмотрим формирующую систему уравнений блочного кода (6, 3)

$$G^t \cdot x = y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_k \\ x_{k+1} \\ x_{k+2} \end{bmatrix} = \begin{bmatrix} y_k \\ y_{k+1} \\ y_{k+2} \\ y_{k+3} \\ y_{k+4} \\ y_{k+5} \end{bmatrix},$$

где G^t - транспонированная образующая (формирующая) матрица

Из примера видно, что первые кодовые биты совпадают с информационными, а избыточные биты представляют собой суперпозиции информационных битов. Коды, содержащие информационные биты в кодовом символе, называются систематическими.

Зачастую информационные и кодовые символы представляются вектор - строками. Так кодирование всевозможных информационных символов из 3-х битов можно записать в виде

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

Образующая матрица систематического кода имеет структуру

$$G = \begin{matrix} k & m \\ k & [I \ P] \end{matrix},$$

где I - единичная матрица; P - проверочная матрица.

На приемной стороне принятый кодовый вектор z трансформируется так называемой проверочной матрицей H в вектор s , называемый синдромом

$$y \rightarrow z \cdot H^t = s.$$

Формально проверочная матрица H может быть получена из условия ортогональности

$$G \cdot H^t = \begin{matrix} k & m \\ k & [I \ P] \end{matrix} \cdot \begin{matrix} m \\ k \\ m \\ [P^t \\ I] \end{matrix} = 0.$$

Таким образом, структура и содержание проверочной матрицы для нашего примера имеет вид

$$H = \begin{matrix} k & m \\ m & [P \ I] \end{matrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Прагматический подход к интерпретации проверочной матрицы состоит в том, что строка проверочной матрицы показывает какие строки транспонированной формирующей матрицы можно просуммировать по модулю 2 и получить 0 при отсутствии ошибок передачи. Всего возможно не более m таких независимых наборов.

Безошибочно принятые кодовые вектора-символы трансформируются матрицей H^t в нулевые векторы - синдромы

$$y \cdot H^t = s = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Наличие ошибки, например, в первом бите кодовых символов дает одинаковые синдромы равные

$$z \cdot H^t = s = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}.$$

При ошибках во вторых битах кодовых векторов дает одинаковые, но другие синдромы

$$z \cdot H^t = s = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}.$$

Таким образом, синдром зависит от места возникновения ошибки в символе и не зависит от символа.

Для нахождения синдромов всех одиночных ошибок удобно воспользоваться теоремой, что синдром ошибки в символе совпадает с синдромом соответствующего вектора ошибки

$$z \cdot H^t = s = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

В результате сразу получили синдромы всех одиночных ошибок. Как видим, для ошибок в первом и втором битах синдромы соответствующих векторов ошибок совпали. Из последнего соотношения следует, что строки матрицы H^t соответствуют синдромам одиночных ошибок.

Вектор ошибки - это вектор, у которого стоит 1 на месте ошибочного бита, а остальные нули. У двоичных кодов для исправления одиночной ошибки достаточно сложить по модулю 2 принятый вектор с соответствующим вектором ошибки.

Поскольку синдромы одиночных ошибок различны, это означает, что данный код способен обнаружить и исправить все одиночные ошибки.

6.2 Систематическое циклическое кодирование

Рассмотрим функциональное представление систематических циклических кодов на примере кода **(6, 3)** на основе примитивного полинома $P(x) = 1 + x + x^3$ (см. рис. (6.2)).

Кодек - это объединение кодера и декодера. Модель кодера для наглядности выполнена на основе битовой последовательности без использования радиоканала, т.е. модуляции/демодуляции несущей и воздействия шумов канала распространения. В связи с

этим в модели между кодером и декодером использован имитатор одиночных ошибок, позволяющий вносить ошибку в любой бит каждого кодового символа.

В циклических кодах разрешенные символы реализуются с помощью операции циклического сдвига (каждый такт сдвига эквивалентен переносу бита с последней позиции на первую и сдвигу всех остальных битов).

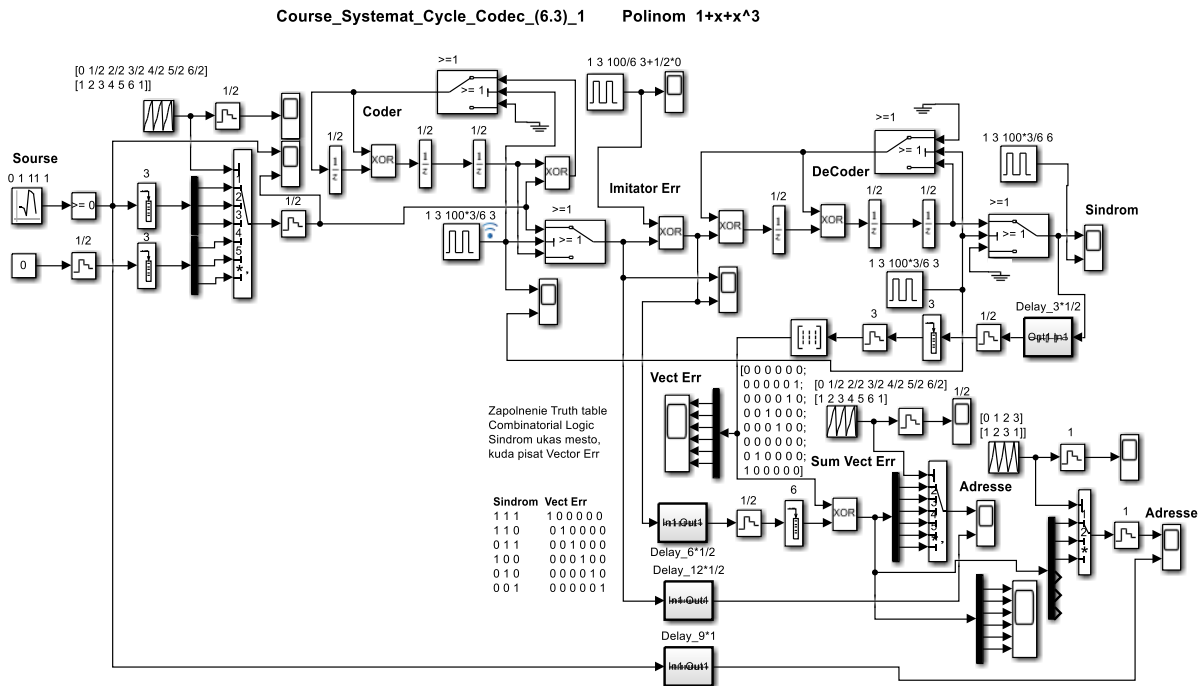


Рисунок 6.2 – Систематический циклический кодек (6, 3) на основе полинома $P(x) = 1 + x + x^3$

Источник информационной битовой последовательности выполнен на основе генератора псевдослучайной последовательности (ПСП) **Random Number** с заданным шагом **Sample Time**. Однополярная битовая последовательность реализуется блоком сравнения **Compare To Zero**. Формирование информационных символов реализуется использованием блока **Buffer** с параметром **Output buffer size=3**. Для кода (6,3) 3-и информационных бита наращиваются 3-мя избыточными битами в кодере, с помощью блоков **Constant Zero Order Hold** и **Buffer** с параметром **Output buffer size=3**. Поскольку размер блока увеличивается, то с помощью блока **Multiport Switch** управляемого блоком генератора пилообразного напряжения **Repeating Sequence** с параметрами **Time values: [0 1/2 2/2 3/2 4/2 5/2 6/2]** и **Output values: [1 2 3 4 5 6 1]** вдвое увеличивается скорость передачи. Блок **Frame Status Conversion** преобразует тип данных из фреймового после блоков **Buffer** в скалярный. Далее бинарный поток данных направляется на систематический циклический кодер, с ключами **Switch** управляемыми блоком **Pulse Generator** с параметрами - амплитудой, периодом, обратным значением скважности, и начальной задержкой в виде: **1 3 100*3/6 3**.

Кодер. На рис. 6.3 приведена Sim-модель систематического циклического кодера (6,3). Структура кодера на основе регистров сдвига с обратной связью соответствует примитивному полиному $P(x) = 1 + x + x^3$.

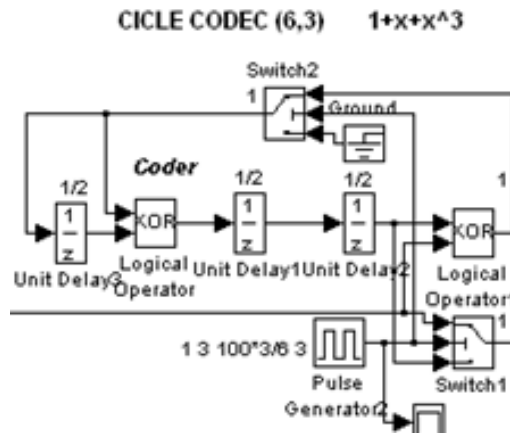


Рисунок 6.3 – *Sim*-модель систематического циклического кодера (6, 3) на основе полинома $P(x) = 1 + x + x^3$

Роль регистров сдвига выполняют блоки задержки на такт *Unit Delay*. Поскольку полином 3-го порядка, регистр содержит три элемента задержки. На входе регистра степень 0, на выходе степень 3. С выхода на вход всегда есть обратная связь (*OC*), в которую в случае систематического кода включен ключ *Switch2*. Кроме нулевой и третьей степени полином содержит компонент степени 1, поэтому после первой задержки ставится сумматор по модулю 2 *XOR*, на один из входов которого берется сигнал из *OC*.

Предварительно информационные биты наращиваются дополнительными нулевыми битами по числу битов четности. Входной блок информационной последовательности поступает на *Switch1* и одновременно через сумматор *XOR* на выход регистра в цепь *OC* и на выход кодера (выход блока *Switch1*). В начальный момент *Switch1* и *Switch2* в верхнем положении, т.е. информационный блок поступает на выход кодера и по цепи *OC* в регистры сдвига. Биты информационного блока продвигаются дискретно тактами. По окончании информационного блока *Pulse Generator* переключает ключи в нижнее положение, т.е. *OC* разрывается и содержимое регистров через *Switch1* выталкивается на выход кодера в качестве дополнительных битов (битов четности). Затем процесс повторяется для следующего блока.

Рассмотренный процесс соответствует делению предварительно сдвинутого информационного полинома на полином кодера. Остаток от деления этих полиномов то есть синдром соответствует полиному представляющего биты четности.

Имитатором однократных ошибок является блок суммирования по модулю 2 *XOR*, на второй вход которого подаются импульсы необходимой длины, скважности и задержки, которые, суммируясь по модулю 2 с определенным битом каждого символа, вносит ошибку. Так, записывая амплитуду, период, обратное значение скважности, и начальной задержки в виде 1, 3, 100/6, 3+0.5*0, получаем ошибки в каждом первом бите кодовых символов. При смене 0 на 1 в параметре начальной задержки импульсов генератора имитирует ошибки во вторых битах символов, и так далее до 5. Далее бинарный поток данных направляется на систематический циклический декодер, с ключами *Switch* управляемыми блоком *Pulse Generator*.

Декодер. На рис. 6.4 приведена *Sim*-модель систематического циклического декодера (6,3). Структура декодера на основе регистров сдвига с обратной связью также соответствует примитивному полиному $P(x) = 1 + x + x^3$. Только здесь сумматор по модулю 2 *XOR* ставится на входе. Процесс декодирования соответствует делению полинома кодового символа на полином кодера. Остаток от деления этих полиномов соответствует полиному представляющего синдром. Если в процессе передачи битов кодового символа ошибок не было, то в остатке получаем нулевой синдром, в противном случае синдром отличен от нулевого. Если мощности кода достаточно, то каждой ошибке соответствует

уникальный синдром. В процессе проектирования кода строится таблица соответствия ошибок и синдромов, что позволяет корректировать (исправлять) ошибки. В случае двоичных кодов для исправления ошибки к принятому символу (вектору) по модулю 2 добавляется вектор ошибки, содержащий на ошибочных позициях 1 и нули 0 в остальных позициях.

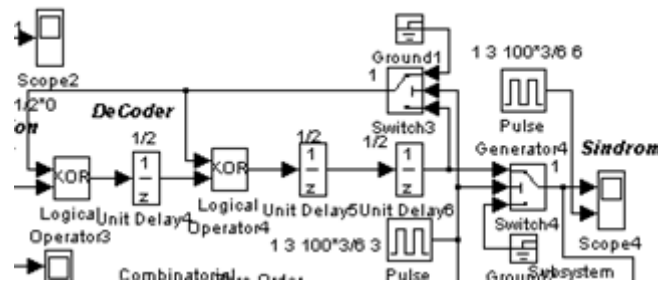


Рисунок 6.4 – *Sim*-модель систематического циклического декодера (6, 3) на основе полинома $P(x) = 1 + x + x^3$

Вначале в регистр поступает систематическая информационная часть кодового символа, выталкивая на выход декодера через **Switch4**, находящемся в верхнем положении, синдром предыдущего кодового символа. **Switch3** при этом тоже находится в верхнем положении, т.е. **ОС** разомкнута. По окончании систематической информационной части ключи **Switch3** и **Switch4** генератором **Pulse Generator** переводятся в нижнее положение. **ОС** при этом замыкается и в регистрах сдвига формируется синдром, который будет вытолкнут систематической частью следующего кодового символа. Синдром как вектор отображается на осциллографе, на котором для облегчения считывания снизу с помощью **Pulse Generator** создается подсветка в виде широкого импульса.

Далее синдром с помощью блока **Combinatorial Logic** с параметром **Truth table: [0 0 0 0 0; 0 0 0 0 0 1; 0 0 0 0 1 0; 0 0 1 0 0 0; 0 0 0 1 0 0; 0 0 0 0 0 0; 0 1 0 0 0 0; 1 0 0 0 0 0]** преобразуется в вектор ошибки. Для синхронизации вектора ошибки синдром вначале задерживается подсистемой из 3-х элементов задержки **Unit Delay** с параметром 0.5, преобразуется в векторную форму буфером **Buffer** и через преобразователь типа подается на **Combinatorial Logic**. Параметры блока **Combinatorial Logic** отображены на скриншоте в виде матрицы.

Строки комбинаторной матрицы нумеруются от нуля до $2^m - 1$, где m - число избыточных битов. Значение синдрома показывает номер строки комбинаторной матрицы, а местоположение синдрома в таблице соответствия показывает номер столбца, на пересечении которых записывается 1, а остальные значения нулевые.

Сформированный вектор ошибки для исправления обнаруженной ошибки принятого символа подается на вход сумматора по модулю 2 **XOR**. На второй вход этого блока подается сформированный **Buffer** принятый символ и задержанный подсистемой из 6-ти элементов задержки **Unit Delay** с параметром 0.5.

Далее принятые и исправленные кодовые символы-векторы преобразуются в последовательный поток блоком **Multiport Switch** и визуализируется блоком **Scope**, на второй вход которого подается задержанный, подсистемой из 12-ти элементов задержки с параметром 0.5, поток, взятый до имитатора ошибок.

Исправленный кодовый символ-вектор демультиплексируется блоком **DeMux**, выделяется систематическая часть и с помощью блока **Multiport Switch** восстанавливается исходная скорость информационного потока. Визуализация результатов декодирования и исправления ошибок информационного потока реализуется блоком **Scope**, на второй вход

которого подается исходный информационный поток, предварительно задержанный подсистемой из 9-ти элементов задержки *Unit Delay* с параметром 1.0.

На основании достаточно подробного описания можно строить функциональные модели и других не очень сложных циклических кодов.

Рассмотрим математическое представление систематических циклических кодов на примере кода (6,3). Математическое представление соответствует операции деления полиномов информационных или кодовых символов на полиномы соответственно кодера и декодера. При описании операций деления старшие степени полиномов записываются впереди, например полином кодера $x^3 + x + 1$. Для удобства полином удобно представить вектором, заменяя присутствующие степени 1, а отсутствующие степени 0, т.е. вектор кодера **1 0 1 1**. Информационным блокам **1 1 1**, **1 1 0**, **1 0 1**, соответствуют полиномы $x^3 + x^2 + x$, $x^3 + x^2 + 0 \cdot x$, $x^3 + 0 \cdot x^2 + x$. Сдвиг на три порядка по числу бит четности приводит к полиномам и векторам $x^6 + x^5 + x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x \rightarrow$ **1 1 1 0 0 0**, $x^6 + x^5 + 0 \cdot x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x \rightarrow$ **1 1 0 0 0 0**, $x^6 + 0 \cdot x^5 + x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x \rightarrow$ **1 0 1 0 0 0**.

Найдем кодовые символы вектора путем деления сдвинутых информационных векторов на вектор кодера **1 0 1 1**.

Возьмем вектор **1 1 1 0 0 0**.

$$\begin{array}{r} \oplus \frac{111000}{1011} \left| \begin{array}{l} 1011 \\ 11 \end{array} \right. \\ \hline 1010 \\ \oplus \frac{1010}{1011} \\ \hline 010 \\ \rightarrow \end{array}$$

В итоге биты четности **0 1 0**, а кодовый символ **1 1 1 0 1 0**.

Найдем синдромы неискаженных кодовых символов-векторов, поделив их на вектор декодера 1 0 1 1.

Возьмем вектор **1 1 1 0 1 0**.

$$\begin{array}{r} \oplus \frac{111010}{1011} \left| \begin{array}{l} 1011 \\ 11 \end{array} \right. \\ \hline 1011 \\ \oplus \frac{1011}{1011} \\ \hline 000 \\ \rightarrow \end{array}$$

В итоге синдром неискаженного символа нулевой **0 0 0**.

Аналогично можно искать другие возможные кодовые символы.

Приведем таблицу всех кодовых символов кода (6.3) на основе полинома $P(x) = 1 + x + x^3$

0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	1	0	1
1	0	0	1	1	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	0	1	0

и убедимся, что все они различны. Можно также и убедиться, что для неискаженных кодовых символов синдромы нулевые.

Для поиска синдромов одиночных ошибок кодовых символов уместно вспомнить теорему о том, что синдром ошибки символа совпадает с синдромом вектора ошибки. Вектор ошибки - это вектор, у которого на месте ошибочных битов стоят 1. Для нахождения синдромов вектора ошибок поделим на вектор декодера **1 0 1 1**.

Возьмем вектор ошибки в первом бите **1 0 0 0 0 0**

$$\begin{array}{r} \oplus \quad \frac{100000}{1011} \quad \left| \begin{array}{l} 1011 \\ 11 \end{array} \right. \\ \hline \oplus \quad \frac{1100}{1011} \\ \hline \underline{111} \\ \rightarrow \end{array}$$

Как видим, синдром вектора ошибки совпал с синдромами ошибок в первом бите. Возьмем вектор ошибки во втором бите **0 1 0 0 0 0**

$$\begin{array}{r} \oplus \quad \frac{010000}{1011} \quad \left| \begin{array}{l} 1011 \\ 1 \end{array} \right. \\ \hline \underline{110} \\ \rightarrow \end{array}$$

Как видим, синдром вектора ошибки совпал с синдромами ошибок во втором бите. Возьмем вектор ошибки в третьем бите **0 0 1 0 0 0**

$$\begin{array}{r} \oplus \quad \frac{001000}{1011} \quad \left| \begin{array}{l} 1011 \\ 1 \end{array} \right. \\ \hline \underline{011} \\ \rightarrow \end{array}$$

Как видим, синдром вектора ошибки совпал с синдромами ошибок в третьем бите. Возьмем вектор ошибки в четвертом бите **0 0 0 1 0 0**

$$\begin{array}{r} \frac{000100}{100} \quad \left| \begin{array}{l} 1011 \\ 0 \end{array} \right. \\ \hline \underline{100} \\ \rightarrow \end{array}$$

Как видим, синдром вектора ошибки в четвертом бите равен **1 0 0**. Возьмем вектор ошибки в пятом бите **0 0 0 0 1 0**

$$\begin{array}{r} \frac{000010}{010} \quad \left| \begin{array}{l} 1011 \\ 0 \end{array} \right. \\ \hline \underline{010} \\ \rightarrow \end{array}$$

Как видим, синдром вектора ошибки в пятом бите равен **0 1 0**. Возьмем вектор ошибки в шестом бите **0 0 0 0 0 1**

$$\begin{array}{r} \frac{000001}{001} \quad \left| \begin{array}{l} 1011 \\ 0 \end{array} \right. \\ \hline \underline{001} \\ \rightarrow \end{array}$$

Как видим, синдром вектора ошибки в шестом бите равен **0 0 1**.
 В итоге имеем следующую таблицу соответствия векторов ошибок и синдромов.

Таблица соответствия векторов ошибок и синдромов

1 0 0 0 0 0	→	1 1 1
0 1 0 0 0 0	→	1 1 0
0 0 1 0 0 0	→	0 1 1
0 0 0 1 0 0	→	1 0 0
0 0 0 0 1 0	→	0 1 0
0 0 0 0 0 1	→	0 0 1

Используя таблицу соответствия легко организовать исправление одиночных ошибок двоичных циклических кодов, сложив по модулю 2 принятый символ с соответствующим вектором ошибки.

Поскольку синдромы одиночных ошибок различны, это означает, что данный код способен обнаружить и исправить все одиночные ошибки.

6.3 Мажоритарное декодирование блочных кодов

Структура и принцип работы мажоритарного декодирования блочного кода (6, 3), кодек, в данном случае это объединение блочного кодера и мажоритарного декодера. Модель кодека для наглядности выполнена на основе битовой последовательности без использования радиоканала, т.е. модуляции/демодуляции несущей и воздействия шумов канала распространения. В связи с этим в модели между кодером и декодером использован имитатор одиночных ошибок, позволяющий вносить ошибку в любой бит каждого кодового символа. Согласно обозначения, код (6, 3) к каждому 3-м информационным битам добавляет 3-и избыточных бита. Функциональная модель двоичного блочного кодера (6, 3) с мажоритарным декодером представлена на рисунке 6.5.

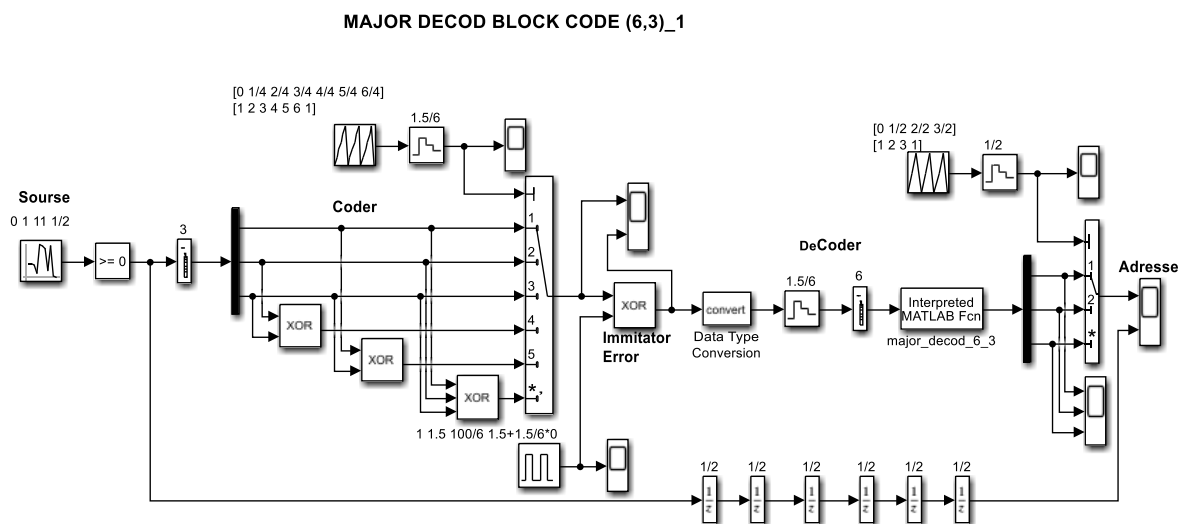


Рисунок 6.5 – Функциональная модель двоичного блочного кодера (6, 3) с мажоритарным декодером

Источник информационной битовой последовательности выполнен на основе генератора псевдослучайной последовательности (ПСП) **Random Number**. Однополярная битовая последовательность реализуется блоком сравнения **Compare To Zero**.

Формирование блока информационных символов реализуется использованием блока **Buffer**. Блочный кодер реализуется демультиплексированием информационных битов блоком **DeMux**, а образование избыточных битов реализуется блоками суммирования по модулю 2 **XOR**. Изменение скорости передачи битов кодового символа реализуется блоком **Multiport Switch**, который управляется генератором пилообразного напряжения **Repeating**, с заданной длиной и крутизной импульсов. Таким образом, блок **Multiport Switch** преобразует параллельное представление кодового символа в последовательный поток выхода кодера.

Имитатором однократных ошибок является блок суммирования по модулю 2 **XOR**, на второй вход которого подаются импульсы необходимой длины, скважности и задержки, которые, суммируясь по модулю 2 с определенным битом каждого символа, вносит ошибку. Так, записывая амплитуду, период, обратное значение скважности, и начальной задержки в виде $1, 1.5, 100/6, 1.5+0.25*0$, получаем ошибки в каждом первом бите кодовых символов. При смене 0 на 1 в параметре начальной задержки импульсов генератора имитирует ошибки во вторых битах символов, и так далее до 5.

Декодер начинается с преобразователя из фреймового в векторный тип данных **Frame Status Conversion**, далее стоят блоки фиксации длительности битов **Zero Order Hold** и **Buffer** - формировавателя кодового символа. Мажоритарное декодирование осуществляется блоком *MatLab Fcn* с использованием *MatLab*-функции, которая представлена на рисунке 6.6.

Далее исправленный информационный символ через демультиплексор **DeMux** подается на мультипортовый ключ **Multiport Switch**, управляемый генератором пилообразного напряжения **Repeating Sequence**, для возвращения к первоначальной скорости передачи. С выхода декодера принятый поток битов направляется для наблюдения на двухлучевой осциллограф **Scope**, на второй вход которого с необходимой задержкой подается исходный информационный поток.

```
function y=major_decod_6_3(x);
% функция y=major_decod_6_3(x);
% функция мажоритарного декодирования блочного кода 6.3
% x- входной массив - принятый кодовый вектор
% y- выходной массив - мажоритарно декодированный вектор данных

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

Рисунок 6.6 – Скрин файл *MatLab*-функции мажоритарного декодирования группового блочного кода (6, 3)

Блочное алгебраическое кодирование и мажоритарное декодирование, основной базис теории описан выше в подразделе 6.1.

Мажоритарное декодирование устроено по принципу голосования и является способом декодирования по большинству проверок.

Для пояснения перепишем формирующую систему уравнений блочного кода (6,3) для текущего блока

$$y = G^t \cdot x = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

Из этой системы выпишем очевидные соотношения

$$\begin{aligned} y_4 &= y_2 + y_3 \\ y_5 &= y_1 + y_3 \\ y_6 &= y_1 + y_2 + y_3 \\ y_1 + y_2 + y_3 + y_4 + y_5 &= 0 \\ y_2 + y_5 + y_6 &= 0 \\ y_1 + y_4 + y_6 &= 0 \end{aligned}$$

Эти соотношения являются разрешающей системой уравнений. Здесь знак + соответствует операции суммирования по модулю 2 - \oplus . Суммирование и вычитание по модулю 2 совпадают. Для независимости уравнений любое уравнение должно отличаться от других хотя бы одной компонентой.

Для декодирования из разрешающей системы уравнений выразим компоненты, соответствующие исходным информационным битам.

Для первого информационного бита имеем соотношения

$$\begin{aligned} y_1 &= y_1 \\ y_1 &= y_3 + y_5 \\ y_1 &= y_2 + y_3 + y_6 \\ y_1 &= y_4 + y_6 \\ y_1 &= y_2 + y_4 + y_5 \end{aligned}$$

Второй информационный бит определится соотношениями

$$\begin{aligned} y_2 &= y_2 \\ y_2 &= y_3 + y_4 \\ y_2 &= y_1 + y_3 + y_6 \\ y_2 &= y_5 + y_6 \\ y_2 &= y_1 + y_4 + y_5 \end{aligned}$$

И наконец, выразим третий информационный бит

$$\begin{aligned} y_3 &= y_3 \\ y_3 &= y_1 + y_5 \\ y_3 &= y_1 + y_2 + y_6 \\ y_3 &= y_2 + y_4 \\ y_3 &= y_4 + y_5 + y_6 \end{aligned}$$

Как видим, каждый информационный бит определяется 5-ю соотношениями, откуда резонно предположить порог «голосования» равным 2.5. Каждое уравнение в силу операции по модулю 2 дает либо истину 1, либо ложь 0. Суммируя ответы получаем, либо 1 при превышении порога, либо 0 в противном случае.

7 ИССЛЕДОВАНИЕ КОДЕМОВ

В данном пункте рассмотрены три вида модемов на основе **QPSK**, **BFSK**, **QPSK** модемов. Это сделано для того, чтобы на наглядном примере продемонстрировать эффективность помехоустойчивых кодеков при разных видах модуляции. Основным показателем эффективности использования помехоустойчивого кодека является энергетический выигрыш кодирования (**ЭВК**), показывающий на сколько **дБ** уменьшился **SNR** после включения кодека в модем. Дополнительно для иллюстрации эффективности кодирования в канал передачи введен детектор ошибок **Detect_Err**, который показывает сколько ошибок возникает в канале передачи на тот момент, когда на выходе кодема возникает, например, одна ошибка. В данном разделе используются простейшие помехоустойчивые коды обнаруживающие и исправляющие однократные ошибки.

Параллельно с помехоустойчивостью анализируется спектральная эффективность модемов и кодемов. В основе исследований лежит известный факт, что ширина основного лепестка низкочастотного спектра последовательности прямоугольных битов составляет $\Delta\Omega = 2\pi/\tau_b$ радиан или $\Delta F = 1/\tau_b$ Гц. Ширина основного лепестка радиочастотного спектра последовательности прямоугольных битов составляет $\Delta\omega = 4\pi/\tau_b$ радиан или $\Delta f = 2/\tau_b$ Гц.

7.1 QPSK модем

Функциональная схема квадратурного **BPSK** модема с приемником прямого преобразования представлена на рисунке 7.1. В классическом **BPSK** модеме фазовые состояния соответствующие значениям битов равных **1** и **-1** составляют, соответственно **0** и **π** радиан. В **QPSK** модеме диаграмма фазовых состояний повернута на **+45°** или **-45°** градусов - (**$\pi/4$** , **$-3\pi/4$**) или (**$-\pi/4$** , **$3\pi/4$**) и реализуется квадратурным модулятором и квадратурными несущими колебаниями **$\cos(\omega_0 t)$** и **$\sin(\omega_0 t)$** . Так как межсигнальное расстояние при **QPSK** аналогично **BPSK**, а длительности модулирующих символов равны длительности бита, их помехоустойчивости и энергетические и спектральные эффективности должны совпадать.

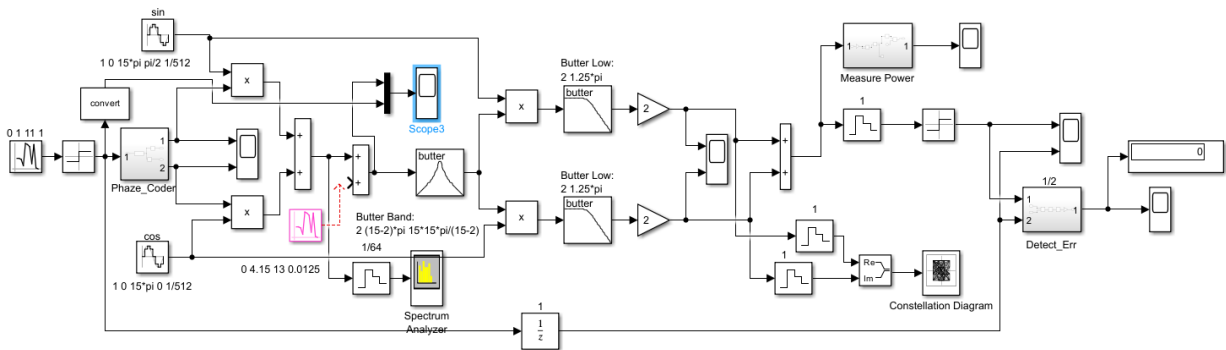


Рисунок 7.1 - Функциональная схема **QPSK** модема

Информационный битовый поток, сформированный блоком **Random Number** с параметрами **(0, 1, 11, 1)**, имеет длину бита **$\tau_b = 1$** (см. рис. 7.2).

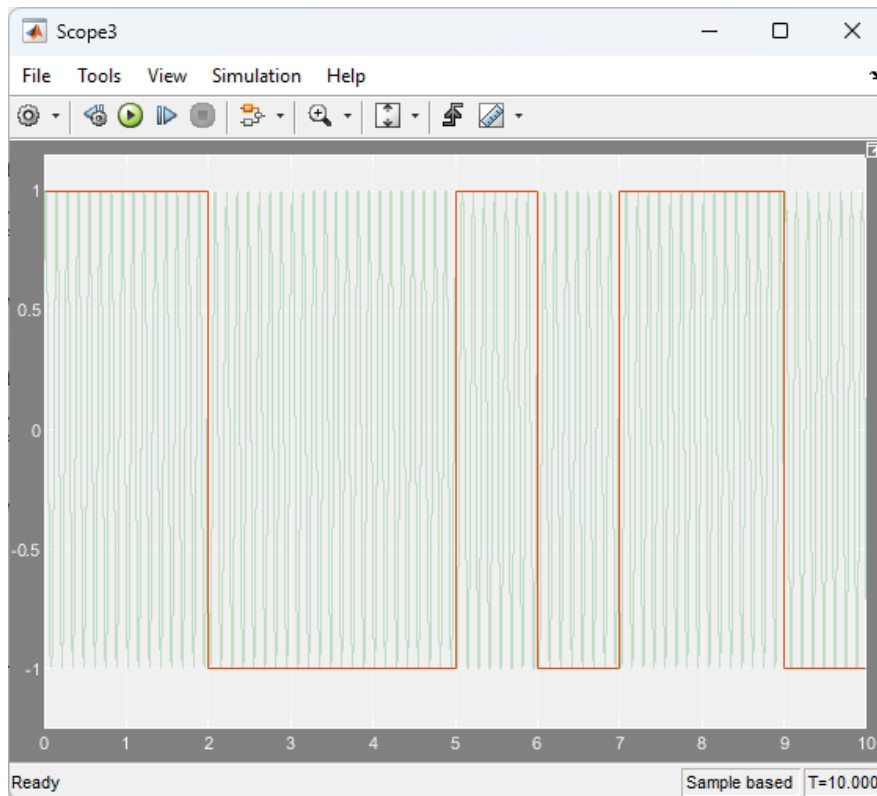


Рисунок 7.2 - Фрагмент *QPSK* радиосигнала модема с наложением битового потока

В нашем случае, при $\tau_b = 1$ использовано несущее колебание с частотой $\omega_0 = 15\pi$. Ширина основного лепестка спектра *QPSK* радиосигнала должна составлять $\Delta\omega = 4\pi i$ радиан или $\Delta f = 2$ Гц. В первом приближении требуемая полоса пропускания совпадает с шириной основного лепестка спектра и в данном случае составляет 4π радиан или 2 Гц.

Ширина спектра основного лепестка *QPSK* радиосигнала равна 2 Гц, что доказывает рисунок 7.3.

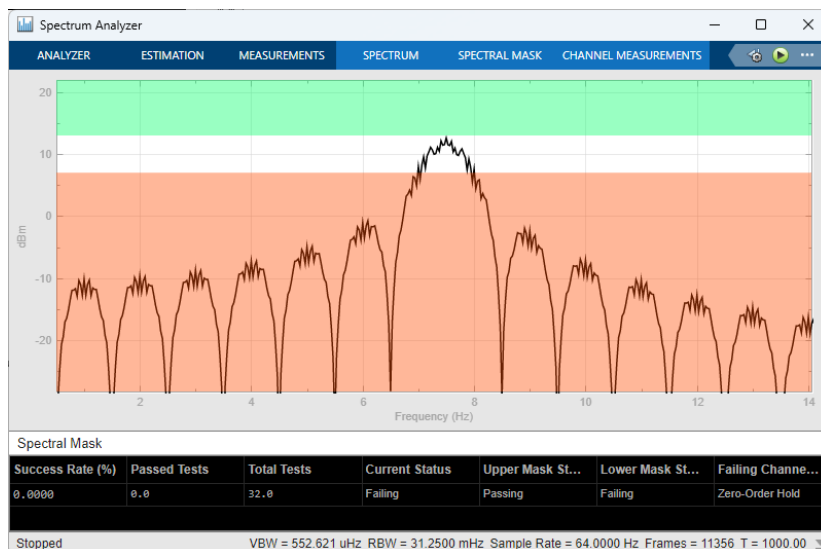


Рисунок 7.3 – Спектр *QPSK* радиосигнала

По результатам исследования помехоустойчивости **Q**PSK модема, получили значение **SNR** при вероятности битовой ошибки $P_b = 10^{-3}$ равное **6.4** раза или **8.06 дБ**.

7.2 Пример QPSK кода

Функциональная схема **Q**PSK кода с блочным кодом (6, 3) и синдромным декодером представлена на рисунке 7.4 и в приложении Г (рисунок Г.1).

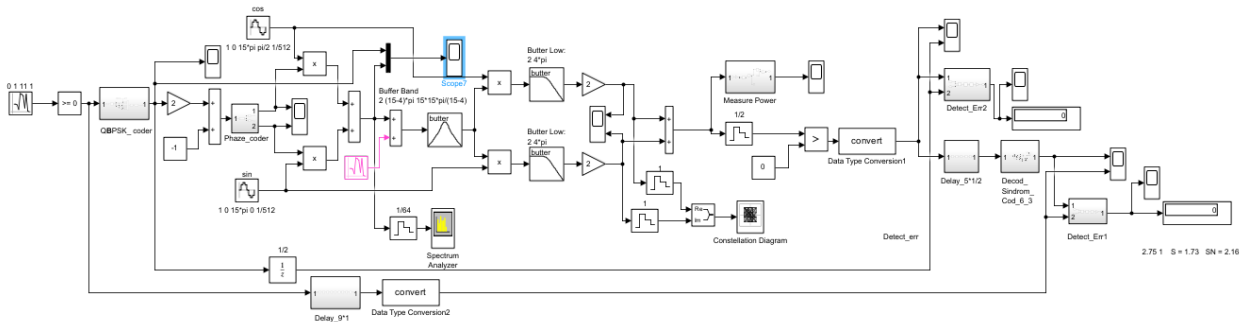


Рисунок 7.4 - Квадратурный **PSK** кодем блочным кодером (6, 3) и синдромным декодером

В канал передачи добавляется детектор для оценки ошибок блок **Detect_err**.

Для того, чтобы сделать из модема кодем был добавлены в передающую часть кодер (**QBPSK_coder**), а в приемную декодер (**Decod_Sindrom_Cod_6_3**). Подсистемы кодера и декодера представлены на рисунках 7.5 и 7.6, соответственно.

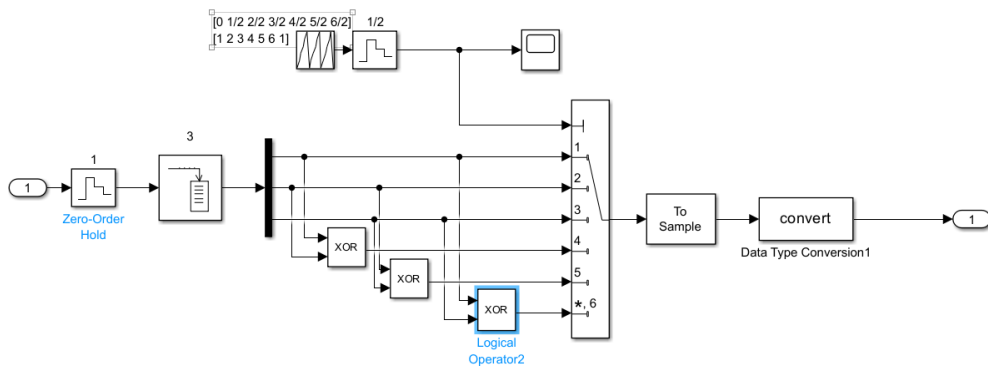


Рисунок 7.5 – Структурная схема подсистемы кодера с блочным кодом (6, 3)

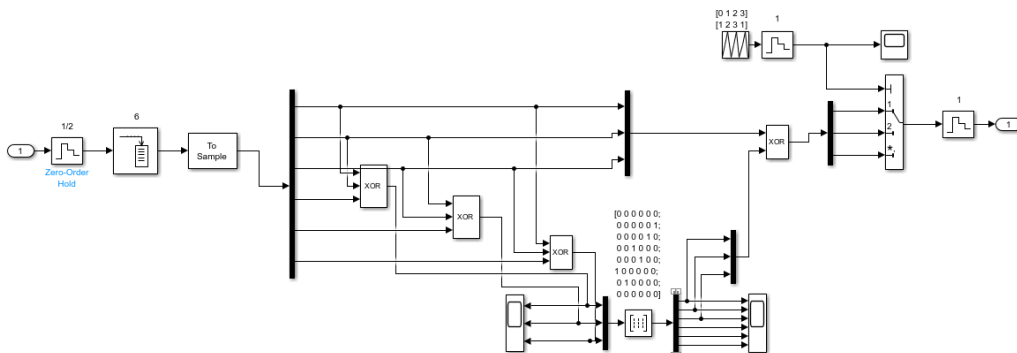


Рисунок 7.6 – Структурная схема подсистемы синдромным декодером с блочным кодом (6, 3)

Заметим, что перед декодером стоит блок задержек ($Delay_{5*1/2}$), это обусловлено фазовой задержкой по частоте в канале приема, и для того, чтобы совместить начало принятого символа с началом декодирования, необходимо добавить эту задержку.

На рисунке 7.7 приведен радиосигнал кодема с нанесенным на него битовым потоком кодового символа.

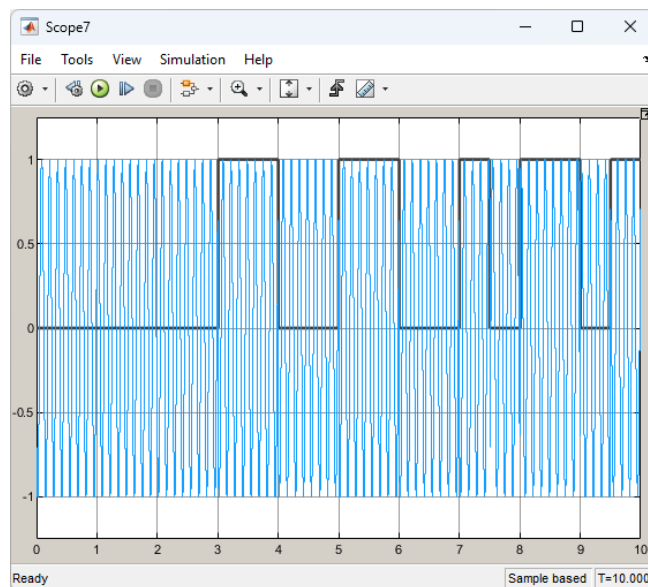


Рисунок 7.7 - Фрагмент *QPSK* радиосигнала модема с наложением битового потока после кодера

В нашем случае, при $\tau_b = 0.5$, использованы прежние несущие колебания с частотами $\omega_0 = 15\pi$. С учетом того, что при кодировании добавляются биты четности длительность битов сокращается вдвое и составляет $\tau_b = 1/2$, при этом ширина основного лепестка спектра радиосигнала и, соответственно требуемая полоса пропускания полосового фильтра кодема, составляет $\Delta\omega = 8\pi$ радиан или $\Delta f = 4$ Гц.

На рисунке 7.8 приведен основной лепесток спектра *QPSK* радиосигнала кодема с несущей частотой $\omega_0 = 15\pi$ и шириной основного лепестка спектра $\Delta f = 4$ Гц.

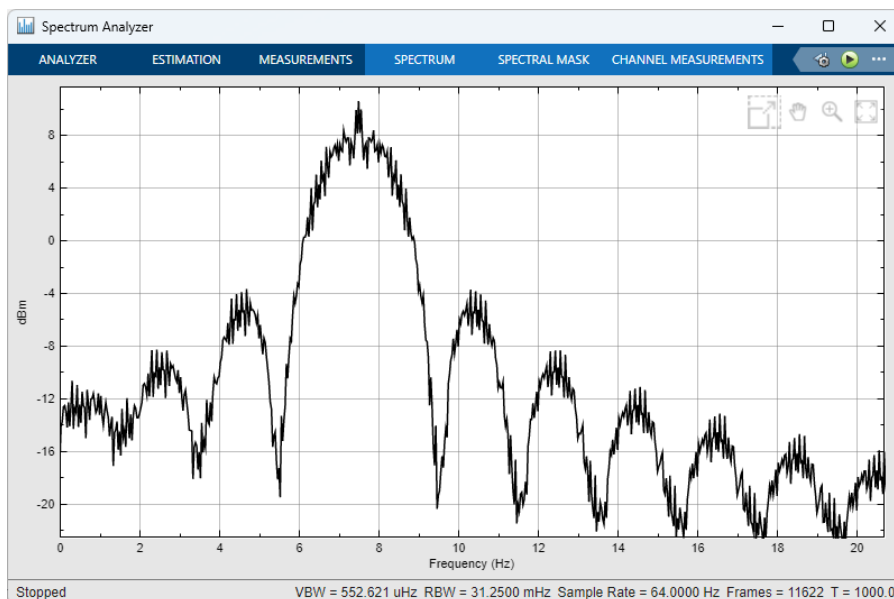


Рисунок 7.8 –Спектр *QPSK* радиосигнала кодема

Ширина основного лепестка спектра радиосигнала примерно совпадает с требуемой шириной полосы пропускания и составляет 4 Гц , что доказывает рисунок 7.8. У кодема по сравнению с модемом, ширина основного лепестка спектра удвоилась, при той же несущей.

По результатам исследования помехоустойчивости **QPSK** кодема, получено значение **SNR** при вероятности битовой ошибки $P_b = 10^{-3}$ равное **4.02** раз или **6.04 дБ**.

Так как значение помехоустойчивости **QPSK** модема составляет **8.06 дБ**, тогда **ЭВК** составляет **2.02 дБ**.

Детектор ошибок, установленный в канал передачи радиосигнала, показал, что при **34** ошибках канала возникает первая ошибка на выходе кодема. Это дополнительно характеризует эффективность использования помехоустойчивого кодема.

7.3 BFSK модем

Функциональная схема **BFSK** модема с приемником прямого преобразования и сложением (вычитанием) каналов обработки, представлена на рисунке 7.9. **BFSK** модуляция является частным случаем **FM (ЧМ)** при модулирующем сигнале в виде двоичной битовой последовательности. В **BFSK** информационный битовый поток модулирует высокочастотный сигнал путем переключения его между двумя несущими частотами.

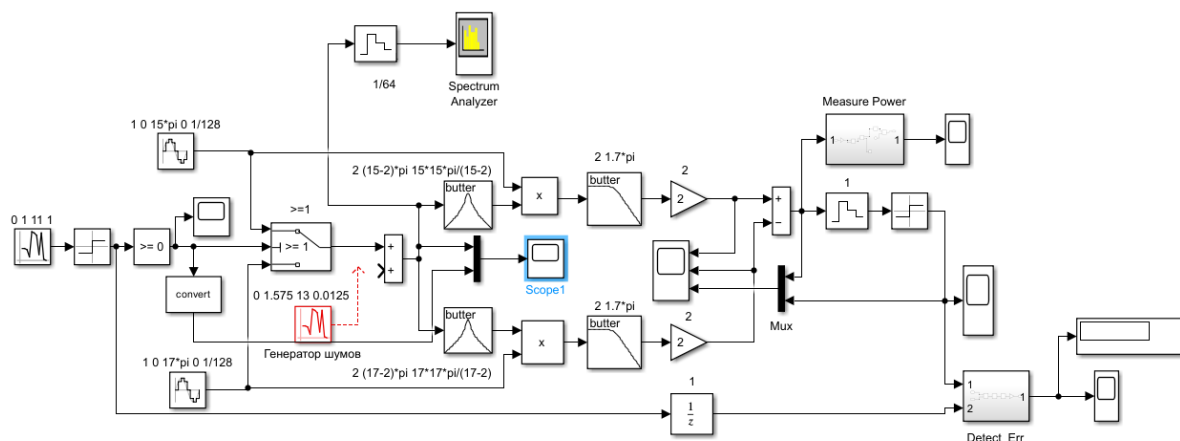


Рисунок 7.9 – Функциональная схема классического **BFSK** модема с суммированием каналов обработки

Изначальный битовый поток, сформированный блоком **Random Number** с параметрами $(0, 1, 11, 1)$, имеет длину бита $\tau_b = 1$ (см. рис. 7.10).

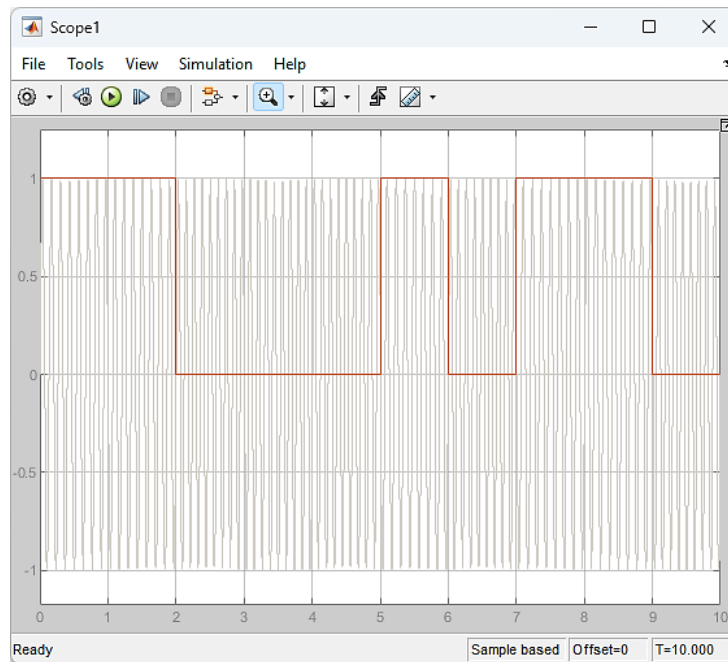


Рисунок 7.10 – Фрагмент *FSK* радиосигнала с наложением битового потока

В нашем случае, при $\tau_b = 1$ использованы ортогональные несущие колебания с частотами $\omega_1 = 15\pi$ и $\omega_2 = 17\pi$. Полоса пропускания на каждой несущей в данном случае, составляет 4π , а с учетом разнеса несущих 2π общая ширина основного лепестка спектра составляет 6π радиан или 3 Гц .

Если разнос несущих частот в радианах составляет 2π , то линейный разнос по частоте составляет 1 Гц , что доказывает рисунок 7.11. Ширина основного лепестка спектра радиосигнала имеет полосу пропускания 3 Гц .

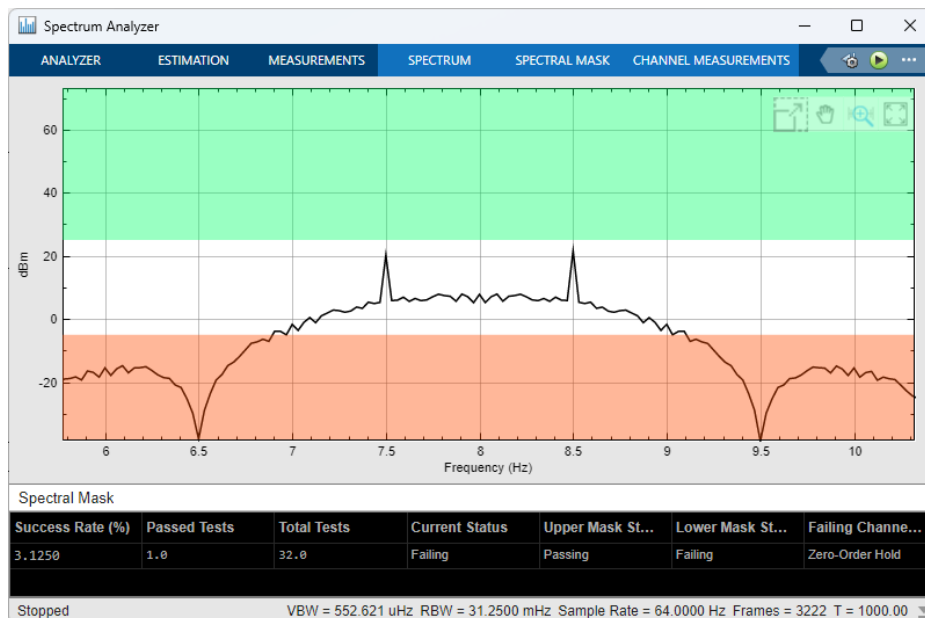


Рисунок 7.11 – Основной лепесток спектра *FSK* радиосигнала

По результатам исследования помехоустойчивости *BFSK* модема, получили значение *SNR* при вероятности битовой ошибки $P_b = 10^{-3}$ равное 9.5 раз или 9.78 дБ .

7.4 Пример BFSK кода

Функциональная схема классического **BFSK** кода с суммированием каналов и с циклическим кодером (6,3) на основе полинома $P(x) = 1 + x + x^3$ и синдромным декодером представлена на рисунке 7.12 и в приложении Д (рисунок Д.1).

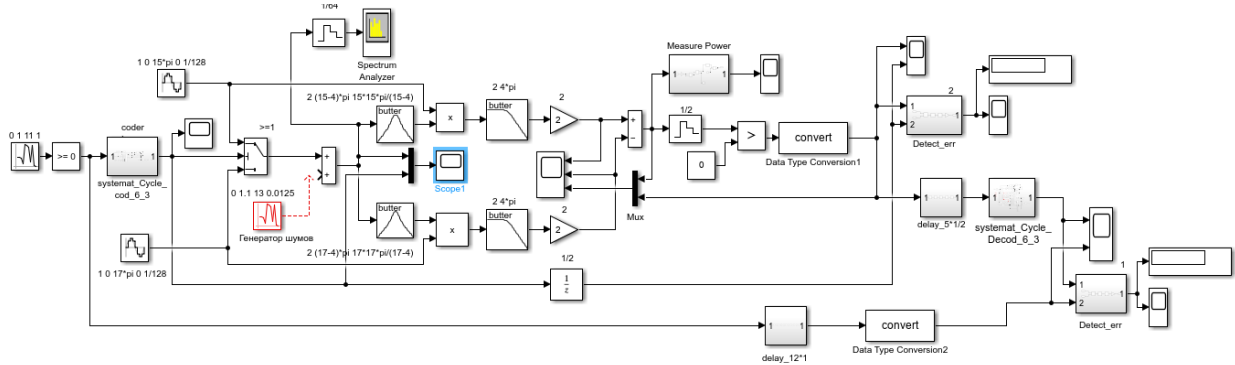


Рисунок 7.12 - Функциональная схема классического **BFSK** кода с суммированием каналов и систематическим циклическим кодером (6,3) на основе полинома $P(x) = 1 + x + x^3$ и синдромным декодером

В канал передачи добавляется детектор для оценки ошибок блок *Detect_err*.

Для того, чтобы сделать из модема кодем в передающую часть был добавлен кодер (*systemat_cycle_cod_6_3*), а в приемную часть - декодер (*systemat_cycle_decod_6_3*). Подсистемы кодера и декодера представлены на рисунках 7.13 и 7.14, соответственно.

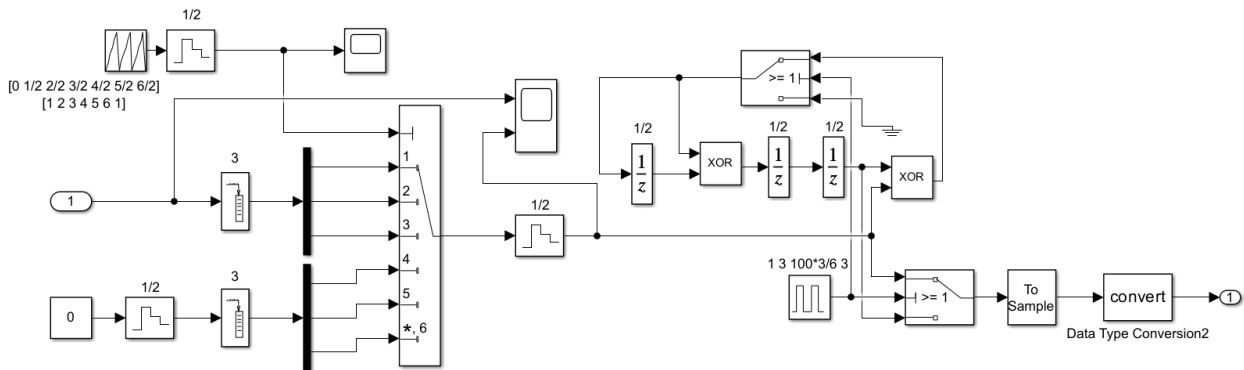


Рисунок 7.13 – Структурная схема подсистемы циклического кодера кода (6,3) на основе полинома $P(x) = 1 + x + x^3$

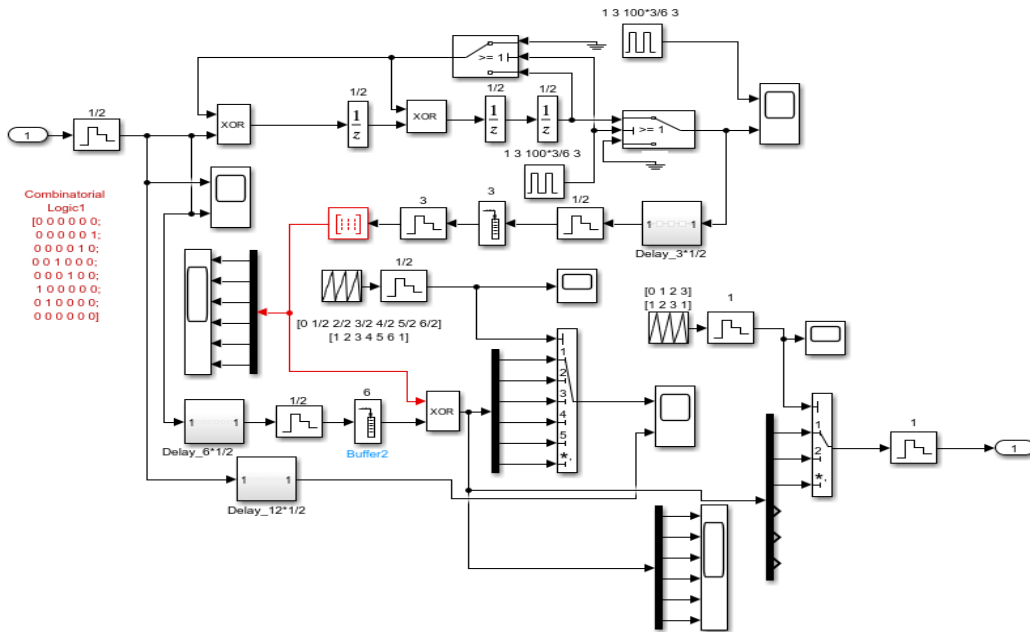


Рисунок 7.14 – Структурная схема подсистемы циклического декодера кода (6,3) на основе полинома $P(x) = 1 + x + x^3$

Заметим, что как и в предыдущем пункте перед декодером стоит блок задержек ($Delay_5*1/2$), обоснование этого смотрите в пункте 7.2.

На рисунке 7.15 приведен радиосигнал кода с нанесенным на него битовым потоком кодового символа. Обратим внимание биты кодового символа за счет добавления избыточных битов сановиться вдвое короче, а скорость вдвое больше.

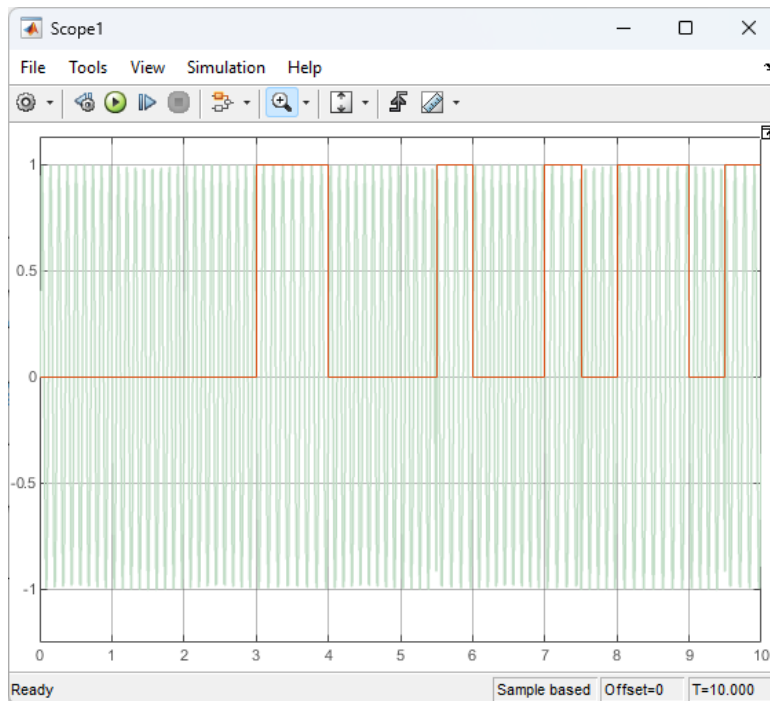


Рисунок 7.15 - Фрагмент *FSK* радиосигнала кода с наложением битового потока после кодера

Поскольку длительность бита изменилась и радиосигнал не является ортогональным, и наблюдаются скачки фазы.

В нашем случае, при $\tau_b = 0.5$ за счет добавления избыточных бит, использованы прежние несущие колебания с частотами $\omega_1 = 15\pi$ и $\omega_2 = 17\pi$. При этом ширина основного лепестка спектра на каждой несущей составляет 8π , а с учетом разноса несущих на 2π общая ширина основного лепестка спектра составляет 10π радиан или 5 Гц .

На рисунке 7.16 приведен основной лепесток спектра **BFSK** радиосигнала модема с несущими частотами $\omega_1 = 15\pi$ и $\omega_2 = 17\pi$ радиан или 7.5 и 8.5 Гц .

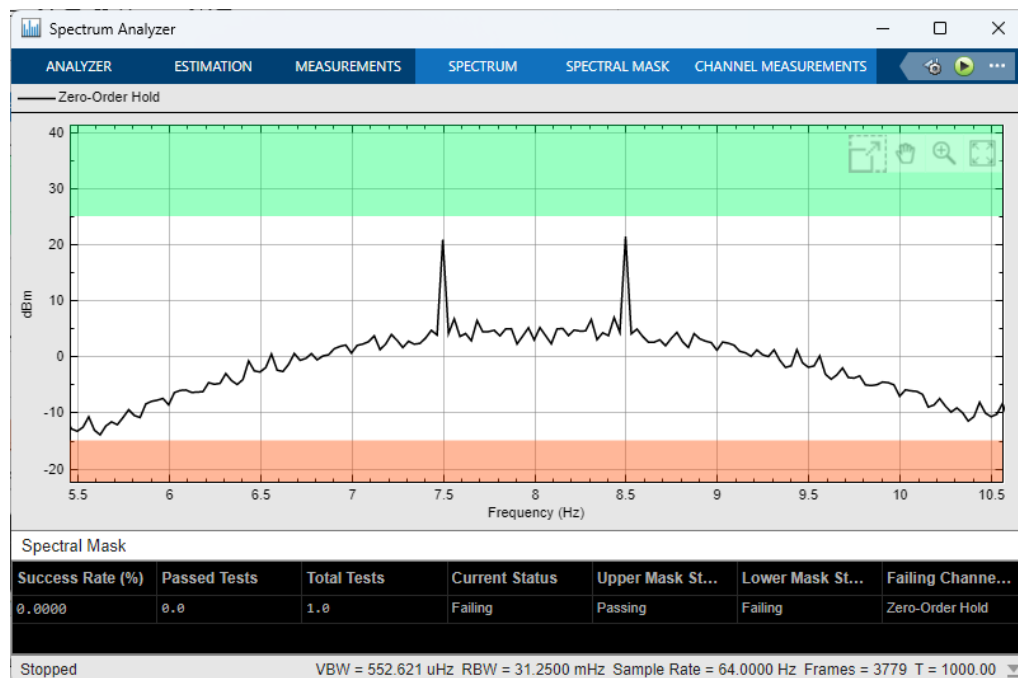


Рисунок 7.16 – Основной лепесток спектра **BFSK** радиосигнала модема

Если разнос несущих частот в радианах составляет 2π , то разнос по частоте составляет 1 Гц , что доказывает рисунок 7.16. Спектр основного лепестка радиосигнала имеет ширину полосы пропускания 5 Гц . У модема по сравнению с модемом, спектр основного лепестка стал шире на 2 Гц , при том же разноре между несущими.

По результатам исследования помехоустойчивости **BFSK** модема, получили значение SNR при вероятности битовой ошибки $P_b = 10^{-3}$ равное 4.76 раз или 6.78 дБ .

Зная, что значение помехоустойчивости **BFSK** модема составляет 9.78 дБ , тогда $ЭВК$ составляет 3 дБ .

Детектор ошибок, установленный в канал передачи радиосигнала, показал, что при 19 ошибках канала возникает первая ошибка на выходе модема. Это дополнительно характеризует выйгрыш в использовании модема.

7.5 QPSK модем

Функциональная схема **QPSK** модема с приемником прямого преобразования, представлена на рисунке 7.17. Модуляция **QPSK** предоставляет необходимый компромисс между скоростью передачи и помехоустойчивостью и применяется как самостоятельно, так и в комбинациях с другими методами.

В нашем случае, при $\tau_b = 1$ использованы квадратурные несущие колебания с частотами $\omega_0 = 15\pi$. Из битов формируются управляющие символы - дибиты. С учетом того, что управляющие символы имеют длину дибита, то есть $\tau_s = 2$ ширина основного лепестка спектра радиосигнала составляет 2π радиан или 1 Гц .

Основной лепесток спектра радиосигнала имеет ширину 1 Гц , что доказывает рисунок 7.20. Несущее колебание находится на частоте 7.5 Гц или 15π радиан.

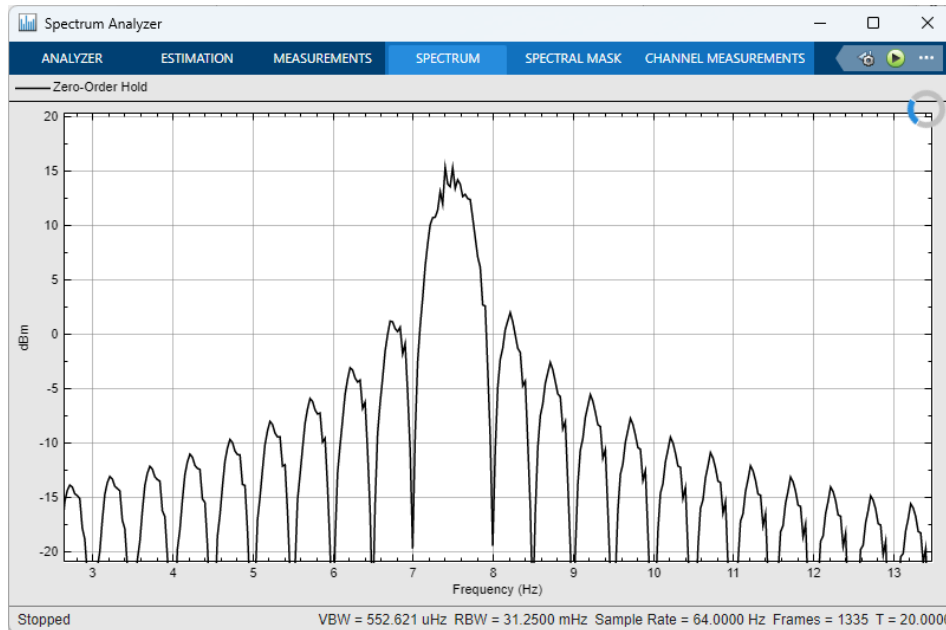


Рисунок 7.20 – Спектр *QPSK* радиосигнала

По результатам исследования помехоустойчивости *QPSK* модема, получили значение *SNR* при вероятности битовой ошибки $P_b = 10^{-3}$ равное 7 или 8.45 дБ.

7.6 Пример *QPSK* кода

Пример работы *QPSK* модема, представлен в разделе 5. Функциональная схема *QPSK* кода с блочным кодом (6,3) и мажоритарным декодером представлена на рисунке 7.21 и в приложении Е (рисунок Е.1).

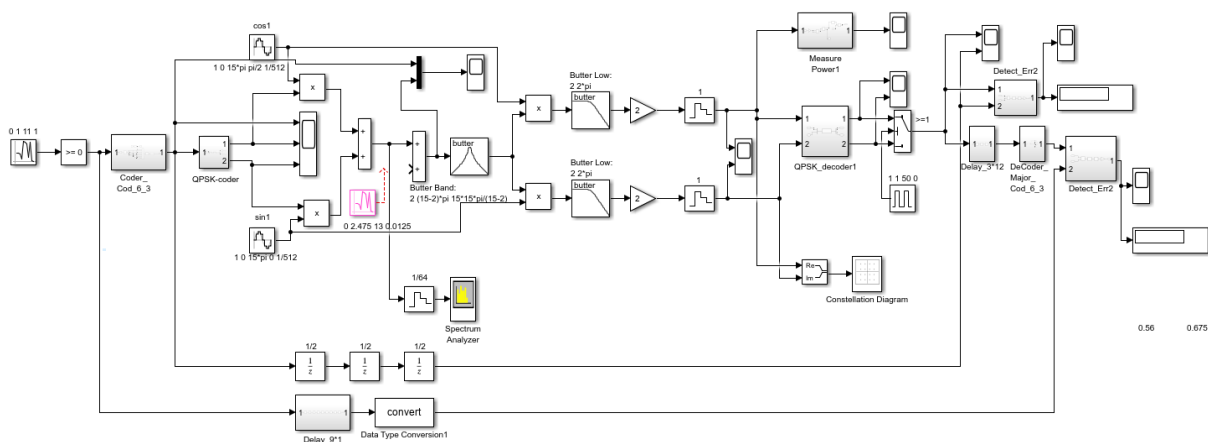


Рисунок 7.21 - Функциональная схема *QPSK* кода с блочным кодом (6,3) и мажоритарным декодером

В канал передачи добавляется детектор для оценки ошибок блок *Detect_err*.

Для того, чтобы сделать из модема кодем в передающую часть был добавлен кодер (*Coder_Cod_6_3*), а в приемную часть - декодер (*Decod_Major_Cod_6_3*). Подсистемы кодера и декодера представлены на рисунках 7.22 и 7.23, соответственно.

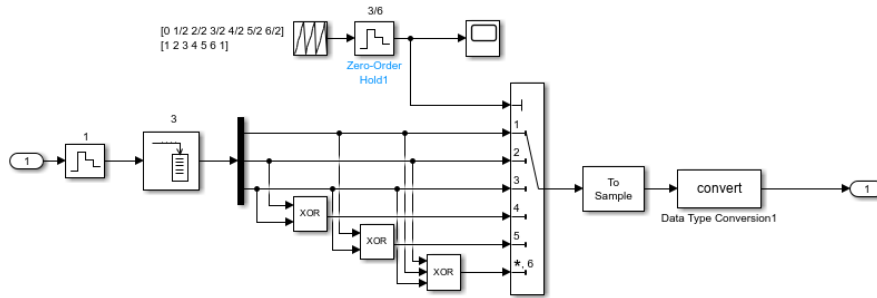


Рисунок 7.22 – Функциональная схема подсистемы кодера блочного кода (6,3)

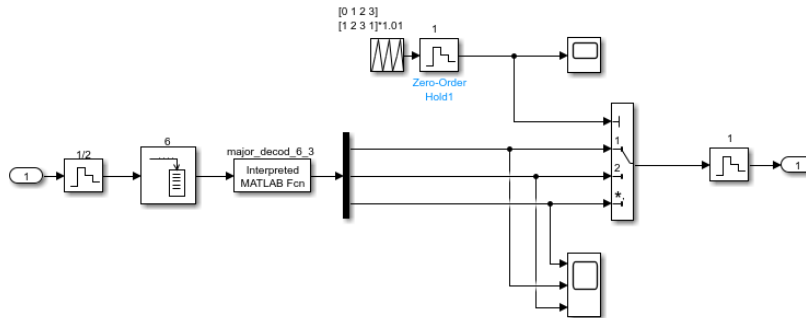


Рисунок 7.23 – Функциональная схема подсистемы мажоритарного декодера блочного кода (6,3)

Заметим, что как и в предыдущем пункте перед декодером стоит блок задержек (*Delay_3*1/2*), обоснование этого смотрите в подразделе 7.2.

На рисунке 7.24 приведен радиосигнал кодема с нанесенным на него битовым потоком кодового символа. Исходный битовый поток имеет $\tau_b = 1$. При коде (6,3) число битов удваивается и длительность бита становится равной $\tau_b = 1/2$. Из этих битов формируются управляющие символы - дибиты. С учетом того, что управляющие символы имеют длину дибита, то есть $\tau_s = 1$ ширина основного лепестка спектра радиосигнала составляет 4π радиан или 2 Гц .

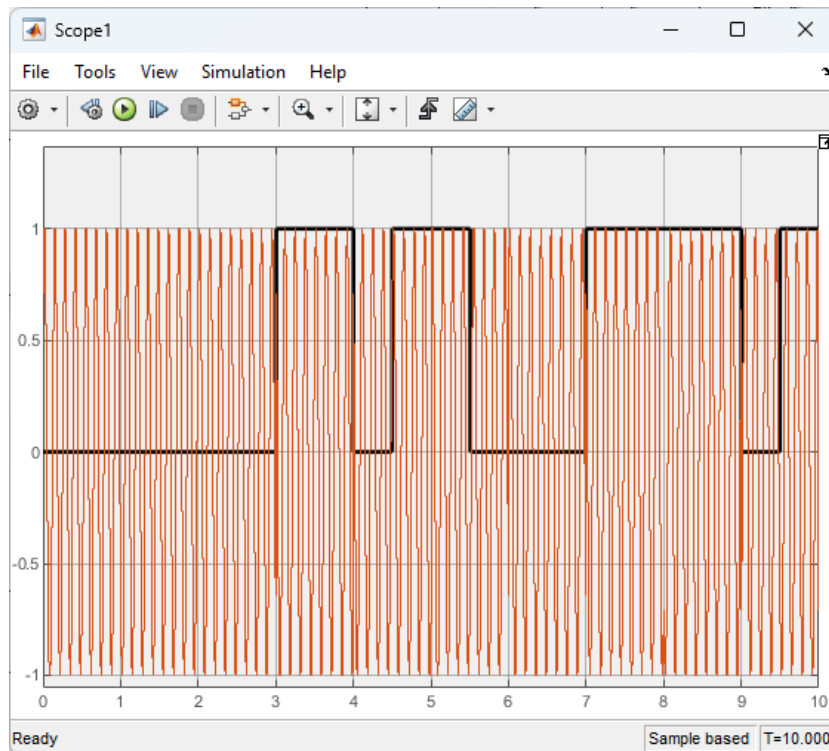


Рисунок 7.24 - Фрагмент *PSK* радиосигнала кодема с наложением битового потока после кодера

В нашем случае использованы квадратурные несущие колебания с частотой $\omega_0 = 15\pi$. Как было показано выше ширина основного лепестка спектра радиосигнала составляет 4π радиан или 2 Гц .

На рисунке 7.25 приведен спектр *QPSK* радиосигнала кодема с несущей частотой $\omega_0 = 15\pi$ и шириной основного лепестка спектра 2 Гц .

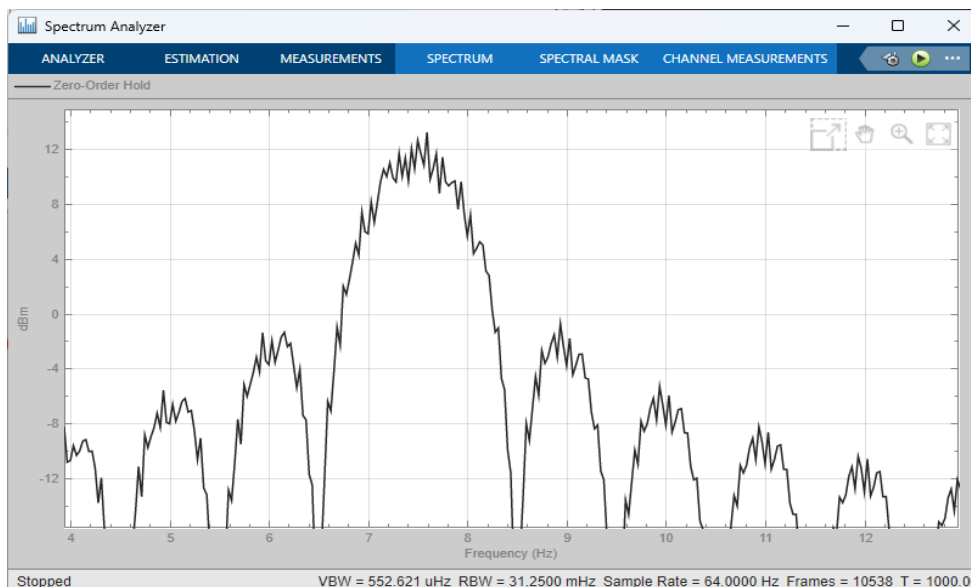


Рисунок 7.25 – Спектр *QPSK* радиосигнала кодема

Спектр основного лепестка радиосигнала имеет ширину полосы пропускания 2 Гц , что доказывает рисунок 7.25. У кодема по сравнению с модемом, ширина основного лепестка спектра удвоилась, при той же несущей частоте.

По результатам исследования помехоустойчивости **QBPSK** модема, получили значение **SNR** при вероятности битовой ошибки $P_b = 10^{-3}$ равное **4.86** раз или **6.88 дБ**.

Зная, что значение помехоустойчивости **QPSK** модема составляет **8.45 дБ**, тогда **ЭВК** составляет **1.57 дБ**.

Детектор ошибок, установленный в канал передачи радиосигнала, показал, что при **22** ошибках канала возникает первая ошибка на выходе модема. Это дополнительно характеризует выигрыш в использовании кодека.

В результате исследования, на наглядных примерах изучили энергетический выигрыш кодирования и эффективность помехоустойчивого кодирования. Все полученные данные для наглядности приведем в таблицу 7.1.

Таблица 7.1 – Полученные данные исследования

Тип модуляции	Помехоустойчивость при $P_b = 10^{-3}$, дБ		Энергетический выигрыш, дБ	Количество возникших ошибок	
	Модем	Кодек		Канал передачи	Кодек
QBPSK	8.06	6.04	2.02	34	1
BFSK	9.78	6.78	3	19	
QPSK	8.45	6.88	1.57	22	

Заключение

В ходе подготовки методических указаний по выполнению курсовой работы по дисциплине «Общая теория связи», были рассмотрены следующие задачи:

1. Ознакомление с возможностями и особенностями функциональной среды моделирования *Simulink* системы *MatLab*.

2. Модельная иллюстрация дискретизации и восстановления аналогового сигнала по Котельникову, а также преобразований *АЦП* и *ЦАП*.

3. Модельная реализация аддитивного скремблера на основе регистра сдвига с обратными связями по структуре примитивного полинома соответствующего порядка.

5. Модельная реализация модуляторов разновидностей частотной и фазовой манипуляции.

6. Модельная реализация канала распространения радиосигнала, позволяющего, в частности, менять соотношение сигнал/шум при исследовании помехоустойчивости модемов.

7. Модельная реализация демодуляторов (приемников) на основе принципа синхронного детектирования (прямого преобразования спектра) либо корреляционного приема.

8. Модельная реализация простых помехоустойчивых групповых и блочных кодов, включая циклические коды.

9. Модельная реализация включения помехоустойчивых кодеков в модем и исследование энергетического выигрыша кодирования (*ЭВК*).

Целью курсового проектирования является овладение методикой выполнения указанных выше задач, а так же расширение и закрепление теоретических знаний. На изучение были предложены и рассмотрены различные примеры модельного проектирования элементов цифрового канала передачи данных, которые помогут студенту сформировать представление о дисциплине *ОТС*.

Общая теория связи, как и теория электрических цепей, это основа многих дисциплин направления «Инфокоммуникационные технологии и системы связи». Она рассматривает многие значимые вопросы, такие как модуляция, помехоустойчивое кодирование, прием и цифровую обработку сигналов, которые в дальнейшем встретятся студентам при изучении других смежных дисциплин. Так после изучения курса *ОТС* студенты будут изучать такие дисциплины как: теоретические основы систем мобильной связи, сотовые и транкинговые системы связи, системы связи на основе шумоподобных сигналов, космические системы связи, системы радионавигации и так далее. Следовательно, дисциплина Общая теория связи является основой для изучения многих специальных дисциплин.

Список использованных источников

1. Образовательный стандарт ОС ТУСУР 01-2021 – Томск: Томск. гос. ун-т систем управления и радиоэлектроники, 2021. – 52 с. - [Электронный ресурс] Режим доступа: <https://regulations.tusur.ru/documents/70> (дата обращения 05.02.2024).
2. Демидов, А. Я. Системы и сети связи: Методическое пособие к лабораторным работам [Электронный ресурс] / А. Я. Демидов. — Томск: ТУСУР, 2012. — 24 с. — Режим доступа: <https://edu.tusur.ru/publications/1402> (дата обращения 09.02.2024).
3. Кулинич, А. П. Исследование аналого-цифрового и цифроаналогового преобразования сигналов: Руководство к лабораторной работе [Электронный ресурс] / А. П. Кулинич, В. Г. Козлов. — Томск: ТУСУР, 2012. — 21 с. — Режим доступа: <https://edu.tusur.ru/publications/1434> (дата обращения 12.02.2024).
4. Сергиенко А. Б. Цифровая связь: Учеб. Пособие. – СПб.: СПбГЭТУ «ЛЭТИ», 2012.- 164 с.
5. Кологривов, В. А. Систематическое циклическое кодирование: Учебно-методическое пособие по лабораторной и самостоятельной работе, и практическим занятиям [Электронный ресурс] / В. А. Кологривов, К. А. Гердт. — Томск: ТУСУР, 2018. — 20 с. — Режим доступа: <https://edu.tusur.ru/publications/7632> (дата обращения 14.02.2024).
6. Михайленко, С. А. Исследование помехоустойчивости FSK-модуляции от соотношения сигнал/шум: Учебно-методическое пособие по лабораторной работе для студентов направления «Инфокоммуникационные технологии и системы связи» по дисциплине «Сети и системы мобильной связи» [Электронный ресурс] / С. А. Михайленко, В. А. Кологривов. — Томск: ТУСУР, 2016. — 30 с. — Режим доступа: <https://edu.tusur.ru/publications/6139> (дата обращения 20.02.2024).
7. Кологривов, В. А. Исследование BPSK модема с синхронным детектированием и корреляционным приёмом: Методические указания по лабораторной работе в среде функционального моделирования Simulink системы MatLab для студентов радиотехнических специальностей [Электронный ресурс] / В. А. Кологривов, Е. В. Суздальцева. — Томск: ТУСУР, 2022. — 26 с. — Режим доступа: <https://edu.tusur.ru/publications/10152> (дата обращения 20.02.2024).
8. Кологривов, В. А. Исследование OQPSK модема: Методические указания по лабораторной работе в среде функционального моделирования Simulink системы MatLab для студентов радиотехнических специальностей [Электронный ресурс] / В. А. Кологривов, Е. С. Никифорова. — Томск: ТУСУР, 2022. — 25 с. — Режим доступа: <https://edu.tusur.ru/publications/10153> (дата обращения 24.02.2024).
9. Кологривов, В. А. Помехоустойчивое кодирование и синдромное декодирование блоковых кодов: Учебно-методическое пособие по лабораторной и самостоятельной работе и практическим занятиям [Электронный ресурс] / В. А. Кологривов, А. А. Алишери — Томск: ТУСУР, 2018. — 21 с. — Режим доступа: <https://edu.tusur.ru/publications/7541> (дата обращения 24.02.2024).

ПРИЛОЖЕНИЕ А
 (обязательное)
Функциональная модель квадратурной реализации BPSK модема

Quadrature_BPSK_Modem_1 QBPSK

SNR=S/(SN-S)=0.198/(0.218-0.198)=9.9000 (9.9564 dB) Nc=2.52 dw=2*pi dW=1.25*pi tau=1 n=1000 n_err=1 Class_BPSK_Modem_1
 SNR=S/(SN-S)=0.396/(0.45-0.396)=7.3333 (8.6530 dB) Nc=4.15 dw=2*pi dW=1.25*pi tau=1 n=1000 n_err=1 Quadrature_BPSK_Modem_1

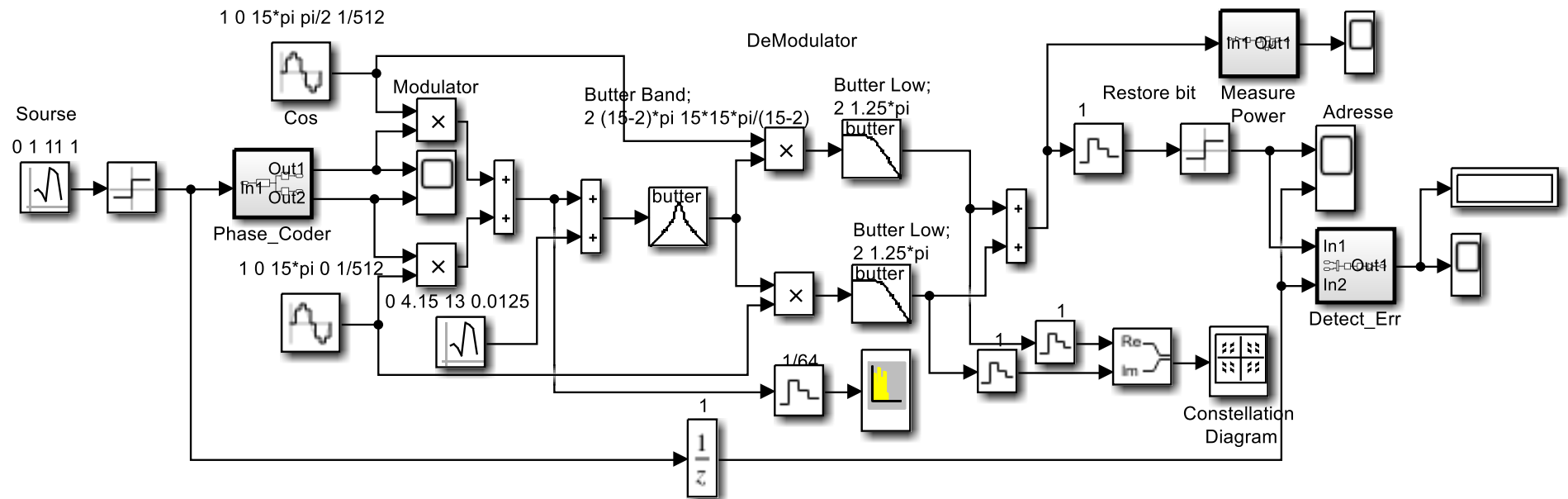


Рисунок А.1 - Функциональная модель квадратурной реализации BPSK модема с приемником прямого преобразования

Функциональная модель квадратурной реализации BPSK модема

Quadrature_BPSK_Modem_2

SNR=S/(SN-S)=0.198/(0.218-0.198)=9.9000 (9.9564 dB) Nc=2.52 dw=2*pi dW=1.25*pi tau=1 n=1000 n_err=1 Class_BPSK_Modem_1

SNR=S/(SN-S)=0.396/(0.45-0.396)=7.3333 (8.6530 dB) Nc=4.15 dw=2*pi dW=1.25*pi tau=1 n=1000 n_err=1 Quadrature_BPSK_Modem_1

SNR=S/(SN-S)=2044.6/(2250-2044.6)=9.9542 (9.9801 dB) Nc=3.375 dw=2*pi dW=1.25*pi tau=1 n=1000 n_err=1 Quadrature_BPSK_Modem_2

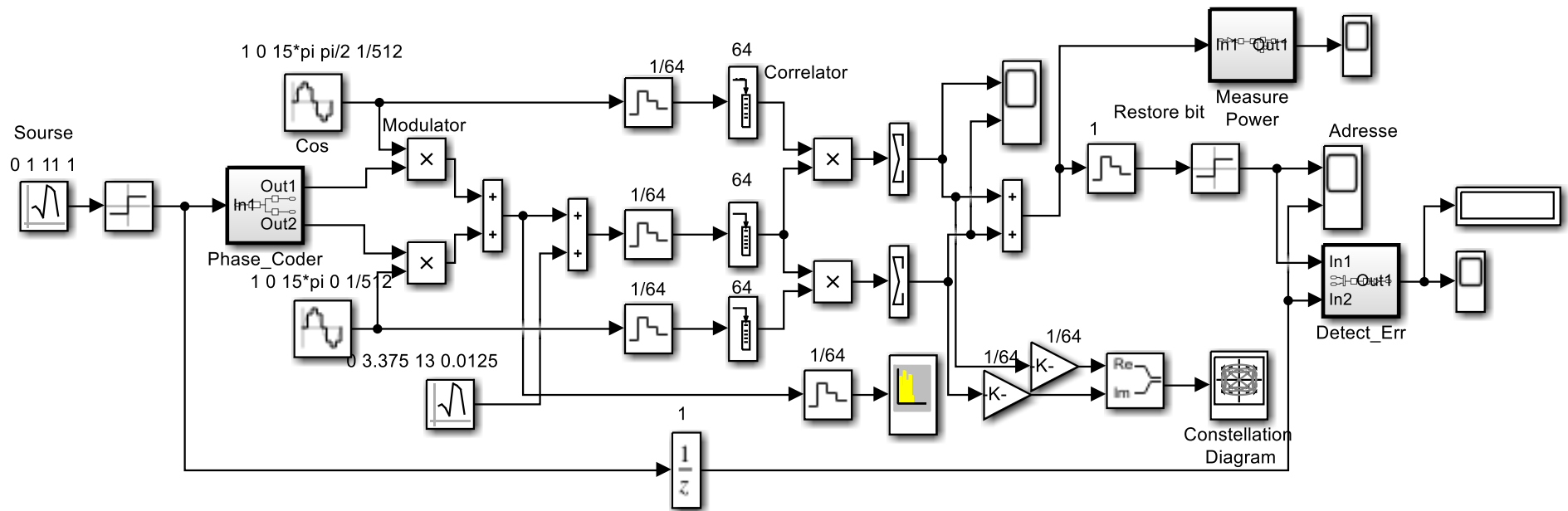


Рисунок А.2 - Функциональная модель квадратурной реализации BPSK модема с корреляционным приемником

ПРИЛОЖЕНИЕ Б
(обязательное)
Функциональная модель BFSK модема

Class_BFSK_Modem_1_Sum_New sin(15*pi*t)_sin(16*pi*t)_SUM_New

SNR=S/(SN-S)=0.198/(0.218-0.198)=9.9000 (9.9564 dB) Nc=2.52 dw=2*pi dW=1.25*pi tau=1 n=1000 n_err=1 Class_BPSK_Modem_1

SNR=S/(SN-S)=0.6075/(0.6525-0.6075)=13.5000 (11.3033 dB) Nc=0.925 dw=2*pi dW=1.7*pi tau=1 n=1000 n_err=1 Class_BFSK_Modem_1 sin 15*pi sin 16*pi

SNR=S/(SN-S)=1.075/(1.25-1.075)=6.1429 (7.8837 dB) Nc=1.6 dw=2*pi dW=1.7*pi tau=1 n=1000 n_err=1 Class_BFSK_Modem_1_SUM sin 15*pi sin 16*pi

SNR=S/(SN-S)=1.08/(1.25-1.08)=6.3529 (8.0297 dB) Nc=1.575 dw=2*pi dW=1.7*pi tau=1 n=1000 n_err=1 Class_BFSK_Modem_1_SUM_New sin 15*pi sin 16*pi

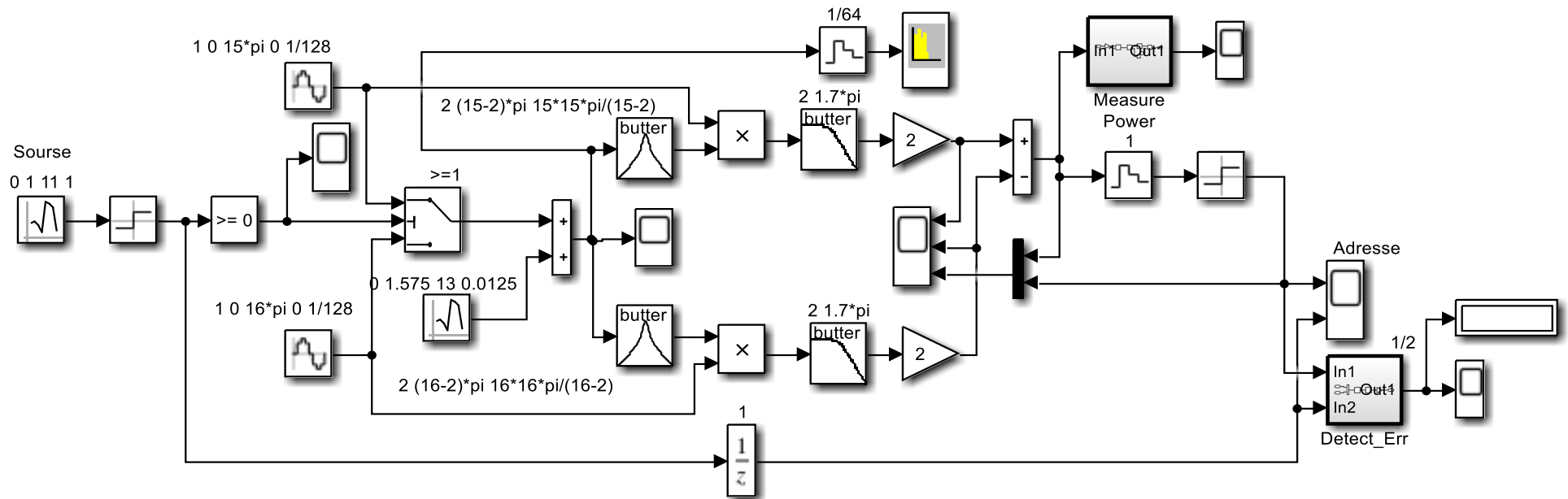


Рисунок Б.1 - Функциональная модель BFSK модема с суммированием канала обработки и прямым преобразованием спектра

Функциональная модель BFSK модема

Class_BFSK_Modem_2_Sum_Corr_New $\sin(15\pi t) \sin(16\pi t)$ _SUM_Corr_New

SNR=S/(SN-S)=0.198/(0.218-0.198)=9.9000 (9.9564 dB) Nc=2.52 dw=2*pi dW=1.25*pi tau=1 n=1000 n_err=1 Class_BPSK_Modem_1

SNR=S/(SN-S)=0.6075/(0.6525-0.6075)=13.5000 (11.3033 dB) Nc=0.925 dw=2*pi dW=1.7*pi tau=1 n=1000 n_err=1 Class_BFSK_Modem_1 $\sin 15\pi \sin 16\pi$

SNR=S/(SN-S)=1.075/(1.25-1.075)=6.1429 (7.8837 dB) Nc=1.6 dw=2*pi dW=1.7*pi tau=1 n=1000 n_err=1 Class_BFSK_Modem_1_SUM $\sin 15\pi \sin 16\pi$

SNR=S/(SN-S)=1.08/(1.25-1.08)=6.3529 (8.0297 dB) Nc=1.575 dw=2*pi dW=1.7*pi tau=1 n=1000 n_err=1 Class_BFSK_Modem_1_SUM_New $\sin 15\pi \sin 16\pi$

SNR=S/(SN-S)=1022.5/(1200-1022.5)=5.7606 (7.6046 dB) Nc=1.85 dt_buf=1/64 n_buf=64 tau=1 n=1000 n_err=1 Class_BFSK_Modem_1_SUM_Corr_New $\sin 15\pi \sin 16\pi$

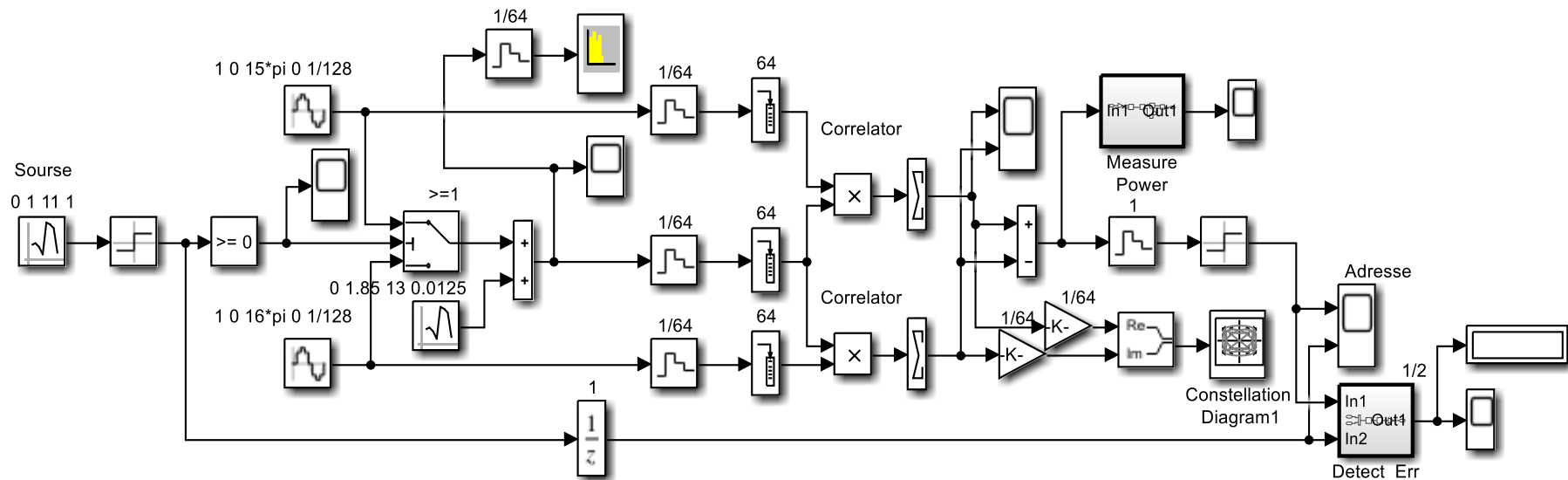


Рисунок Б.2 - Функциональная модель **BFSK** модема с суммированием канала обработки и корреляционным приемником

ПРИЛОЖЕНИЕ В
(обязательное)
Функциональная схема исследования QPSK модема

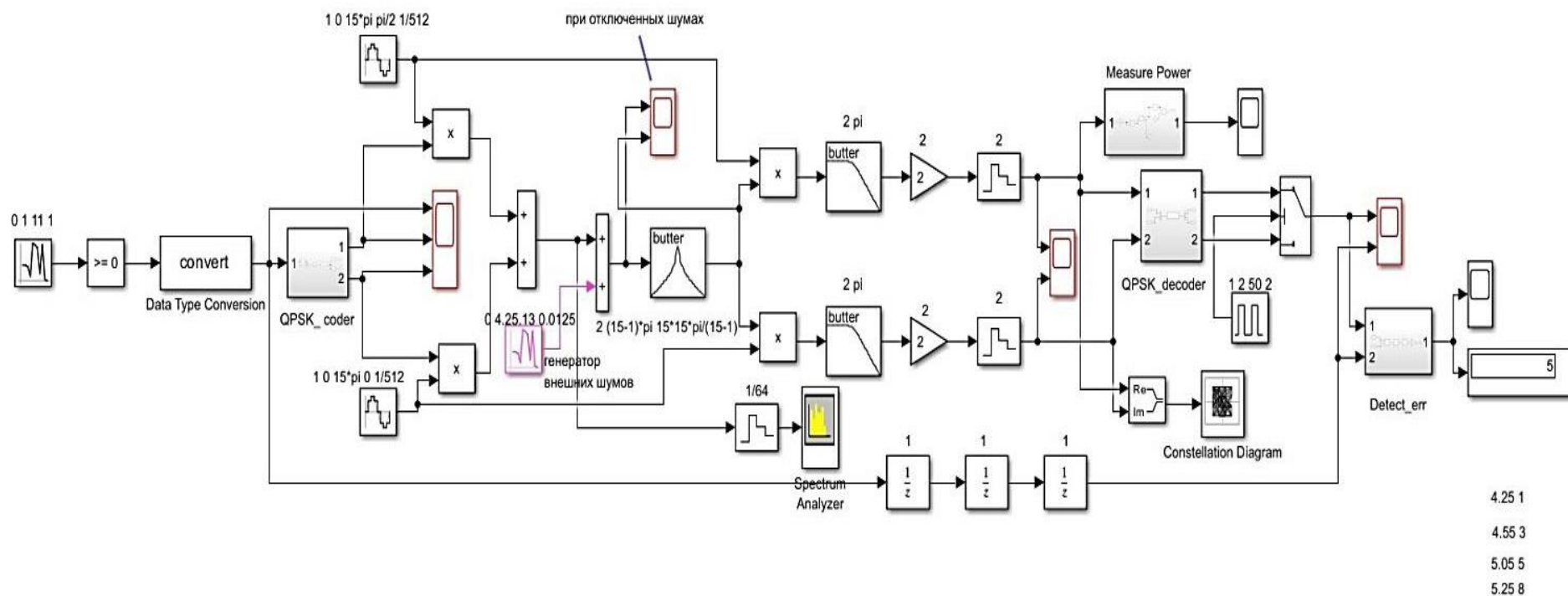


Рисунок В.1 – Функциональная схема исследования *QPSK* модема с приемником прямого преобразования

Функциональная схема исследования QPSK модема

Class_QPSK_Modem_2 QPSK_Corr

$SNR=S/(SN-S)=0.198/(0.218-0.198)=9.9000$ (9.9564 dB) $N_c=2.52$ $dw=2*\pi$ $dW=1.25*\pi$ $\tau=1$ $n=1000$ $n_err=1$ Class_BPSK_Modem_1
 $SNR=S/(SN-S)=0.396/(0.45-0.396)=7.3333$ (8.6530 dB) $N_c=4.15$ $dw=2*\pi$ $dW=1.25*\pi$ $\tau=1$ $n=1000$ $n_err=1$ Quadrature_BPSK_Modem_1
 $SNR=S/(SN-S)=0.56/(0.65-0.56)=6.2222$ (7.9395 dB) $N_c=4.25$ $dw=\pi$ $dW=\pi$ $\tau=1$ $n=1000$ $n_err=1$ Class_QPSK_Modem_1
 $SNR_{out}=S/(SN-S)=540/(630-540)=6.0000$ (7.7815 dB) $N_c=4.3$ $dw=\pi$ $dW=\pi$ $\tau=1$ $n=1000$ $n_err=1$ Class_QPSK_Modem_2 Corr

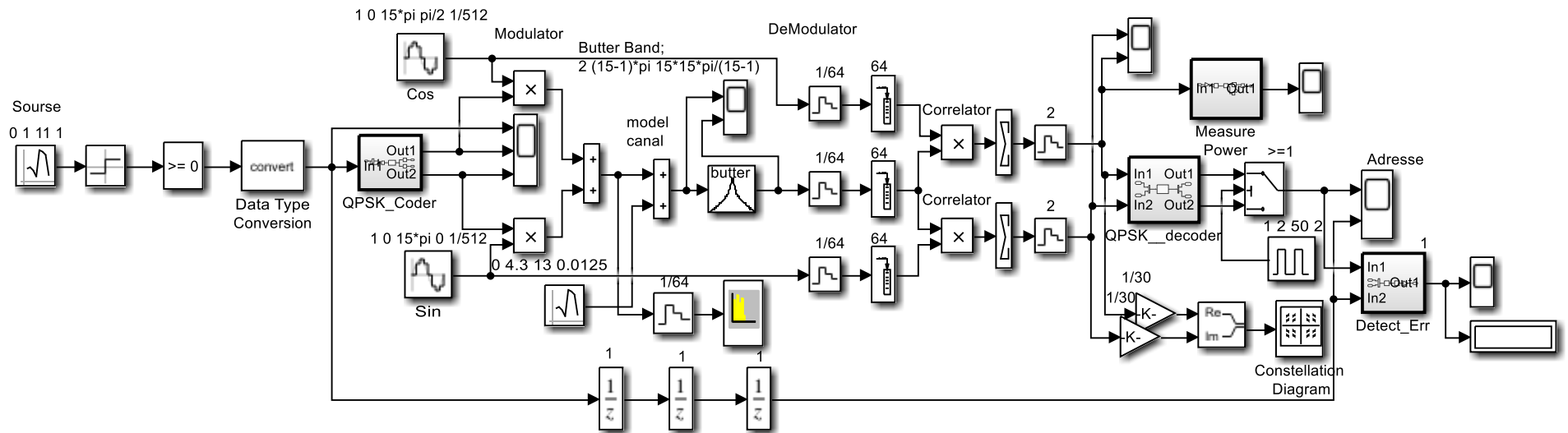


Рисунок В.2 - Sim-модель классического QPSK модема с корреляционным приемником демодулятором

ПРИЛОЖЕНИЕ Г
(обязательное)
Функциональная модель QPSK кода с синдромным декодером

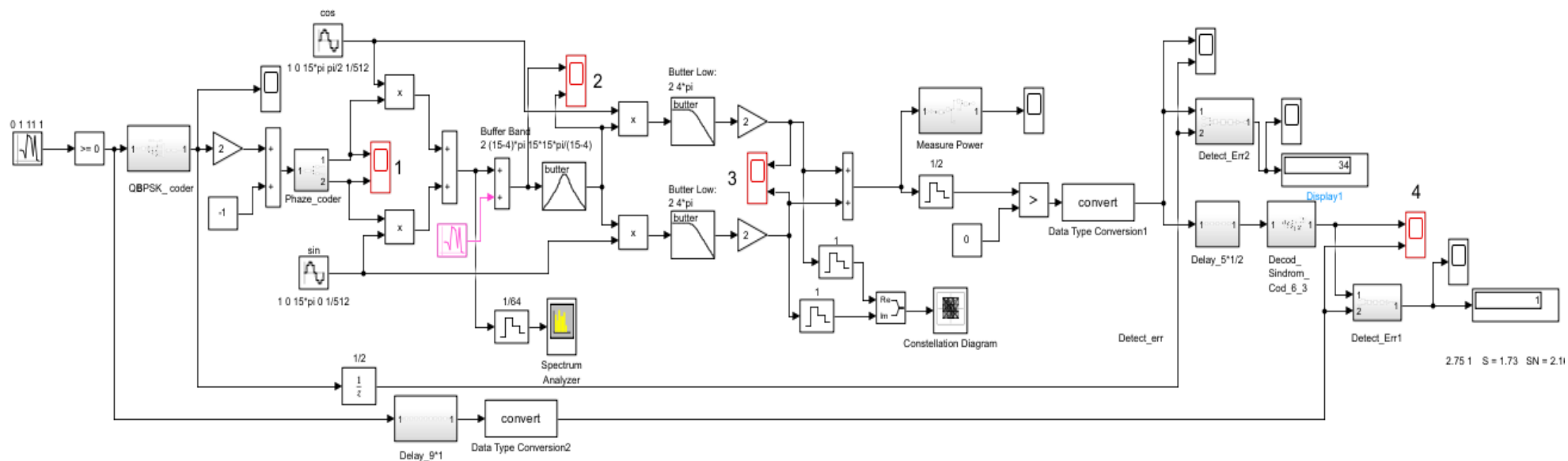


Рисунок Г.1 - Квадратурный *QPSK* кодем блочным кодером (6,3) и синдромным декодером

ПРИЛОЖЕНИЕ Д
(обязательное)
Функциональная модель BFSK кода с синдромным циклическим декодером

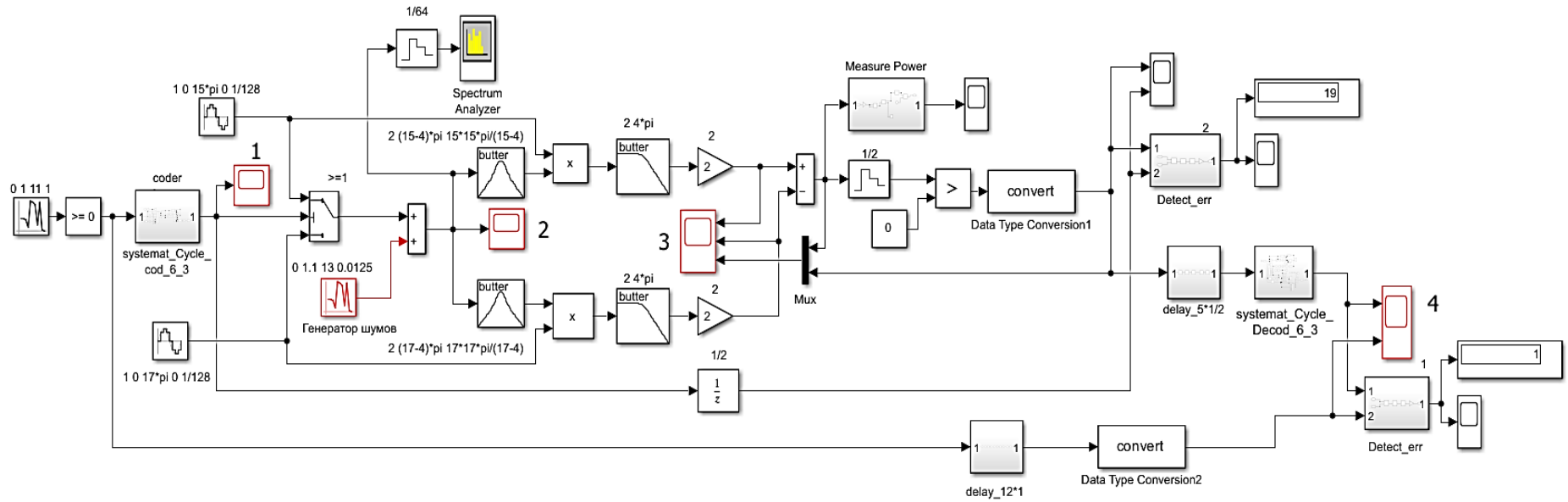


Рисунок Д.1 - Функциональная схема классического **BFSK** кода с суммированием каналов и с циклическим кодером (6, 3) на основе полинома $P(x) = 1 + x + x^3$ и синдромным декодером

ПРИЛОЖЕНИЕ Е
(обязательное)
Функциональная модель схема QPSK кодема с мажоритарным декодером

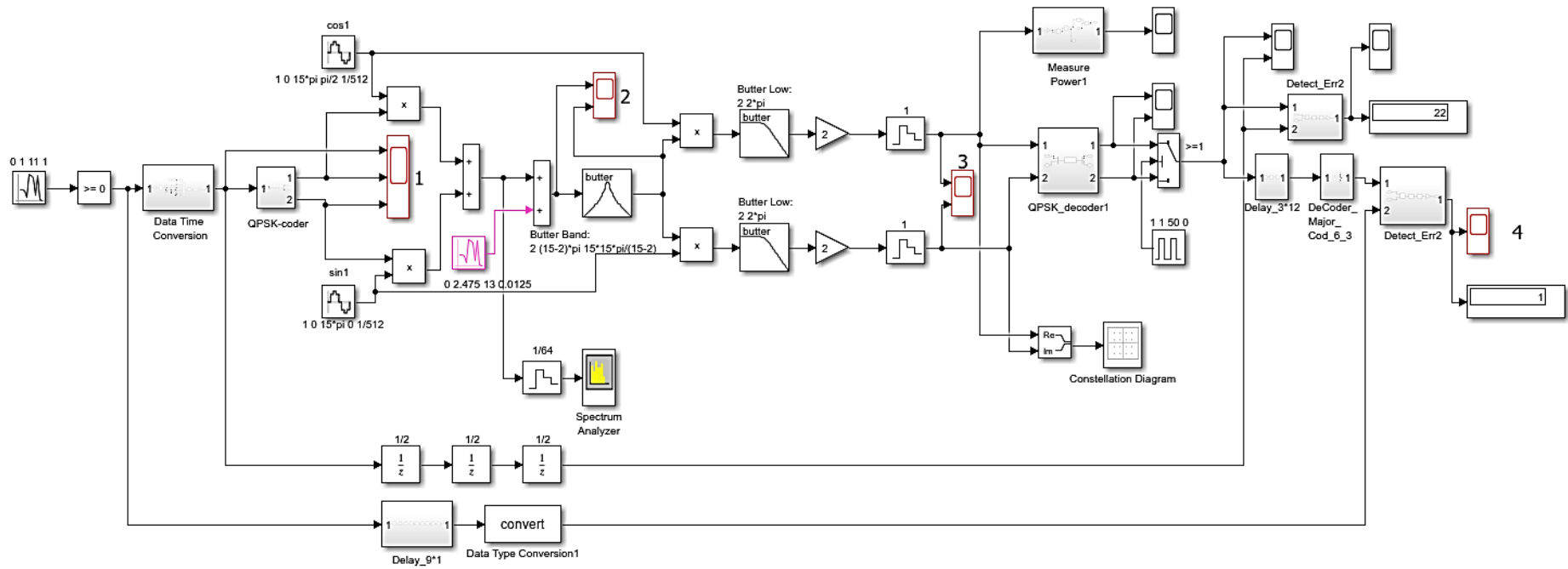


Рисунок Е.1 - Функциональная схема *QPSK* кодема с блочным кодером (6, 3) и мажоритарным декодером