Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

Радиотехнический факультет (РТФ)

Кафедра радиотехнических систем (РТС)

СБОРНИК ЭЛЕКТРОННЫХ ЛАБОРАТОРНЫХ РАБОТ ПО СИСТЕМАМ СВЯЗИ

Учебно-методическое пособие для выполнения лабораторных работ
и самостоятельной работы студентов по направлениям
11.03.01, 11.03.02, 11.04.01 и **11.04.02**
и следующим дисциплинам:
Общая теория радиосвязи / Цифровая связь /
Теория и техника передачи информации / Теория радиосвязи

РАЗРАБОТЧИК:
доц. каф. РТС, к.т.н.,
_____ А.В. Новиков

2017

Новиков А.В.
Сборник электронных работ по системам связи: учебно-методическое пособие для выполнения лабораторных работ и самостоятельной работы студентов по направлениям **11.03.01, 11.03.02, 11.04.01** и **11.04.02** и следующим дисциплинам: Общая теория радиосвязи / Цифровая связь / Теория и техника передачи информации / Теория радиосвязи. — Томск: Радиотехнический факультет, ТУСУР, 2017. — 158 с.

Предлагаемый сборник содержит семь разноплановых лабораторных работ по системам связи.

В первых двух работах изучаются циклические двоичные коды Хемминга. Рассмотрено кодирование и декодирование алгебраически, а также с помощью цифровых фильтров; показана взаимосвязь обоих способов. Подробно исследуются пространственные свойства кода (7, 4).

В третьей работе изучаются сверточные коды. Подробно рассматривается декодирование алгоритмом Витерби по решетке, а также пороговое декодирование на примере простейшего систематического сверточного кода со скоростью кодирования 1/2. Показан пример вывода формулы для вероятности ошибки после декодирования для порогового метода.

Четвертая работа посвящена изучению двоичной частотной манипуляции. Разъяснен способ формирования частотно модулированного сигнала с непрерывной фазой. Иллюстрируется ручная установка фазы тактовых импульсов в цепи тактовой синхронизации при некогерентной демодуляции.

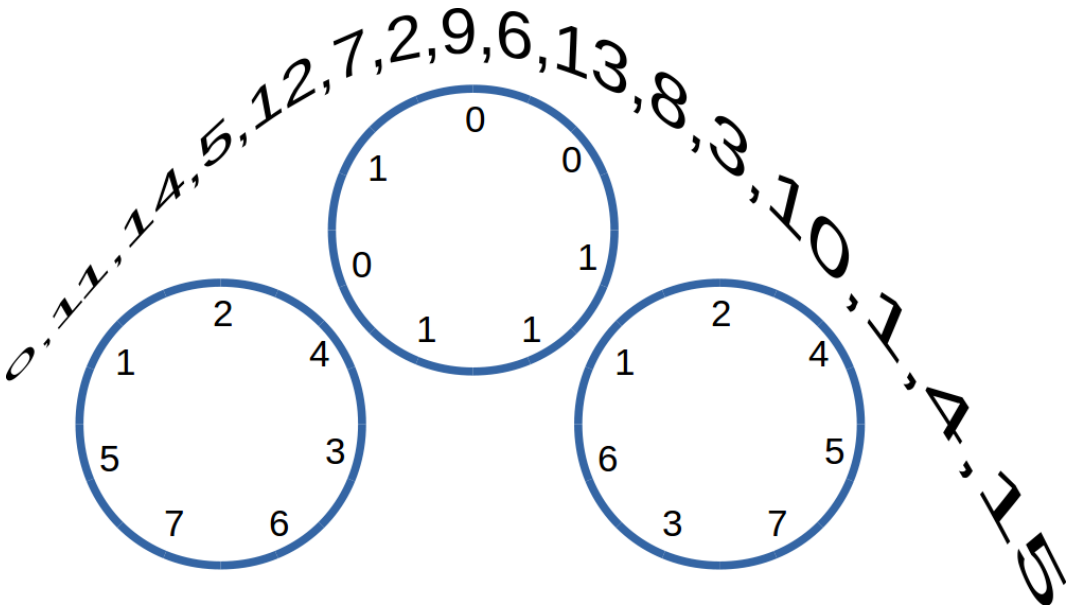
В пятой работе рассматриваются спектры сигналов с линейной модуляцией. Поставлен акцент на влияние корреляционных свойств цифровой последовательности на спектр модулированного сигнала.

В шестой работе изучается дельта-модуляция. Даны принципы формирования однобитового цифрового сигнала, а также способ его демодуляции. Исследуются такие погрешности дельта-модуляции как перегрузка по скорости, ошибка по уровню и неидеальность интегратора. Показывается применение теории z-преобразования к расчету напряжения на выходе цифрового фильтра нижних частот в режиме молчания.

Седьмая работа посвящена методике кодирования отсчетов дискретного сигнала. Иллюстрируется метод последовательного счета и поразрядного приближения. В лабораторном макете реализовано кодирование поразрядным приближением весами из чисел Фибоначчи; иллюстрируется возможность обнаружения сбоев при кодировании за счет избыточности кода Фибоначчи.

ОГЛАВЛЕНИЕ

ЦИКЛИЧЕСКИЕ КОДЫ-1



1. Введение

Тема работы относится к разделу кодирование канала [1].

В современных цифровых системах передачи информации кодирование канала как правило используется совместно с кодированием источника [1], несмотря на то, что они как «лебедь, рак и щука» тянут в разные стороны. Канальное кодирование направлено на получение выигрыша в помехоустойчивости, что достигается искусственной избыточностью, которая снижает скорость передачи информации; кодирование источника направлено на уменьшение естественной избыточности, что дает плюс к скорости передачи информации и минус к помехоустойчивости. Примерами кодирования источника являются, например, универсальные архиваторы файлов **zip** или **rar**, звуковые кодеки типа **mp3** или **flac,** видеокодеки типа **mp4** или **HuffYUV** и тому подобное.

Естественная избыточность, содержащаяся практически в любой информации, в общем случае не может быть использована для защиты от помех, так как для технического устройства она носит случайный характер. Например, сумму в 12000 рублей можно записать **многими, заранее не предсказуемыми** способами: «12000 руб.», «Двенадцать тысяч рублей 00 копеек», «12 тыс. руб. 00 к.» и так далее и тому подобное. Поэтому кодированием источника эту избыточность уменьшают, после чего добавляют искусственную избыточность для получения требуемой помехоустойчивости. Естественной избыточностью может пользоваться только обученный языку человек.

Яркий пример искусственной избыточности — контрольная сумма некоторого файла, вычисляемая по известному алгоритму. Это, например, циклические суммы **CRC16, CRC32,** хэш-суммы **MD5, SHA-1, SHA-2, SHA-3.** Контрольная сумма — это проверочные символы, по которым можно оценить целостность записанного, например, на оптический диск файла.

Кодер источника — это «черный ящик» для экономной упаковки (сжатия) информации. Здесь уместно провести аналогию с конструктором **LEGO,**

когда кодер разбирает массивную конструкцию и компактно упаковывает мелкие детали в коробку, прилагая рядом схему сборки. Декодер, соответственно, служит для распаковки архива и сборки информации (рис. 1).

Кодер канала — «черный ящик» для упаковки информации в «оболочку», которая предназначена для защиты от помех, неизбежно присутствующих в любых каналах связи, а в особенности — в радиоканалах. Декодер канала распаковывает эту «оболочку», по возможности делает «косметический ремонт» принятой информации, и затем передает ее следующему устройству в цепочке преобразования информации, а «оболочка» за ненадобностью отбрасывается.

Кодер и декодер принято называть одним словом — *кодек.*

Предполагается, что студент знает основную суть процессов модуляции-демодуляции цифровых сигналов (символы или цифры не могут передаваться по каналу связи без соответствующих сигналов), и понимает, что для упрощения эти блоки на рис. 1 не показаны.

В данной лабораторной работе изучаются возможности циклического кода Хемминга и алгоритмы кодирования и декодирования. Логическим продолжением является работа **Циклические коды-2,** которая предназначена для отработки схемных реализаций кодера и декодера (регистров сдвига с линейной обратной связью).

Методические требования к выполнению работы:

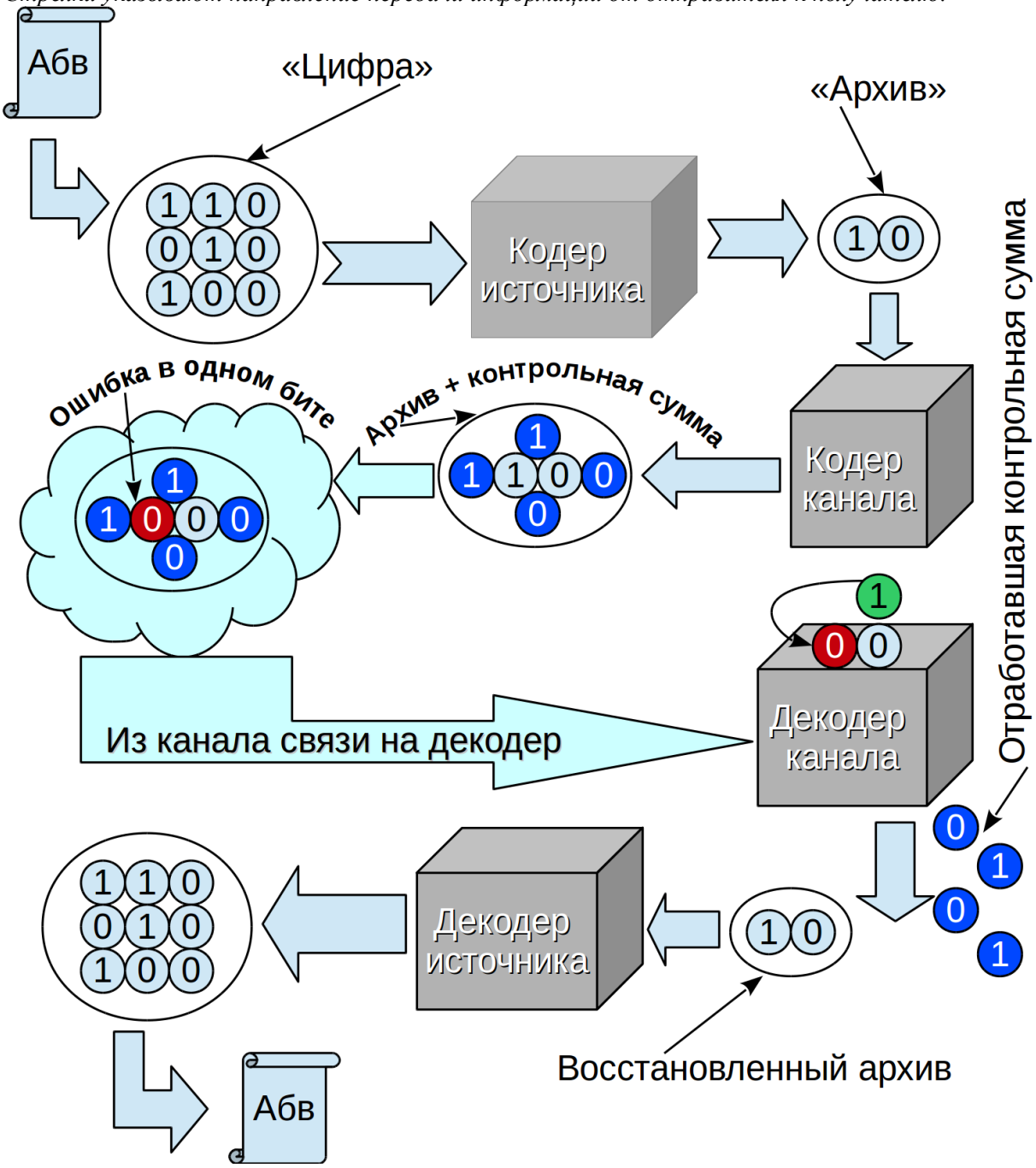
- Если в тексте встречается выделенное жирным шрифтом сказуемое **(показать, заполнить...),** то процесс выполнения соответствующего задания должен быть отражен в отчете по ходу работы;
- Работу по возможности выполнять последовательно, так как задания связаны между собой. Например, ответ на текущий вопрос предполагает знание ответа на некоторые предыдущие. Теорию рекомендуется читать полностью, проверяя на «бумажке с карандашом» выводы формул.

Структура отчета должна соответствовать следующему:

- ✔ Титульный лист;
- ✔ Ход работы;
- ✔ Ответы на вопросы;
- ✔ Выводы.

Принимаются форматы файлов **doc, docx, odt** и **pdf.**

Свиток «Абв» — аналоговая информация (текст, речь, видео, изображение…).
Овалы с нулями и единицами — пакеты с цифровой информацией.
Стрелки указывают направление передачи информации от отправителя к получателю.



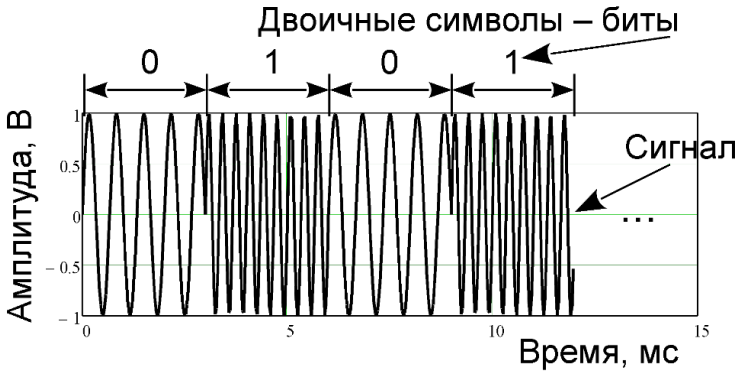
Информация с помощью аналого-цифрового преобразователя переходит в цифровую форму, сжимается кодером источника, а затем канальным кодером вносится искусственная избыточность, после чего упакованная информация передается по каналу связи с помехами.

Канальный декодер анализирует принятую информацию на предмет наличия ошибок, по возможности исправляет их и передает информацию декодеру источника, который ее распаковывает. И, наконец, исходный цифровой поток поступает на цифро-аналоговый преобразователь для отображения аналоговой информации, годной для получателя.

Рисунок 1 Роль двух видов кодирования и их место в цифровых системах передачи информации

2. Сведения из теории

Циклические коды, относящиеся к классу *помехоустойчивых кодов,* предназначены для борьбы с помехами в канале связи, которые могут приводить к ошибкам при демодуляции принимаемого сигнала (при угадывании символа по форме принятого сигнала).



Факт использования помехоустойчивых кодов принято называть *кодированием канала.* Основное свойство данных кодов — наличие избыточной информации, введенной в канал связи по заранее известным правилам, проверка которых на приемной стороне позволяет обнаружить и даже исправить некоторые ошибки.

Самый понятный способ кодирования-декодирования — табличный, когда составляется таблица, состоящая из двух колонок, в одной из которых перебираются всевозможные информационные слова, а в другой записываются соответствующие разрешенные (табл. 1).

Табл. 1. Пример табличного помехоустойчивого кодирования

Информационное слово	**Разрешенное кодовое слово**
0 0	0 1 0 1 0 1
0 1	1 1 0 0 1 1
1 0	1 1 1 0 0 0
1 1	1 1 1 1 1 1

При этом нет никаких ограничений на правило кодирования, кроме требования **однозначности декодирования** и наличия **избыточности:** можно кодировать «на глаз», можно использовать формулы и т. д. В качестве примера табличного кодирования приведем код 4B/5B, используемый на физическом уровне сетевой технологии Ethernet 100Мбит/с (100Base-TX) для само-

синхронизации. Данный код гарантирует не более трех нулевых битов подряд в выходном потоке. Его кодовая таблица содержит всего 16 строк, что не накладывает серьезных ограничений на использование памяти кодера/декодера. Избыточность кода 4B/5B проявляется в том, что на входе четыре бита, а на выходе — пять, то есть $2^4=16$ слов кодируются пятью битами.

Если оценить количество требуемых строк таблицы, например, для кода с информационными словами из ста **двоичных** символов, то получается «**два** в сотой» или около десяти в тридцатой степени! Очевидно, что никакое электронное устройство не обеспечит такой объем памяти с приемлемым быстродействием поиска кодовых слов в таблице. Однако коды таких размеров и даже бóльших (~40000 [2]) успешно используются в реальных системах связи. Яркий пример тому — цифровое телевидение **DVB-T2** [2], а причиной успеха является выделение класса *линейных кодов.*

Основная особенность линейных кодов — линейность операций кодирования (вычисления проверочных символов) и декодирования (анализа принятых символов). Вместо кодовой таблицы используется матрица из k строк, которая позволяет сгенерировать 2^k **двоичных** кодовых слов. Разница в объемах — налицо, и она тем больше, чем больше длина кода. Ограничение на правило кодирования у линейных кодов — требование линейной независимости строк порождающей матрицы, так как разрешенные кодовые слова получаются с помощью всевозможных сумм строк этой матрицы. Однозначность декодирования никто не отменял, и эта однозначность табличных кодов вылилась в линейную независимость строк порождающей матрицы.

Стремясь к компактности кодеков, на этом не останавливаются, и среди линейных кодов выделяют класс циклических, основная особенность которых состоит в том, что *циклические перестановки* разрешенных кодовых слов дают разрешенные кодовые слова. При этом используется не матричный, а полиномиальный способ кодирования, который требует задания всего одного вектора, состоящего из коэффициентов некоторого полинома. Цифровые

устройства, реализующие данный способ, — обычные цифровые фильтры, состоящие из регистра сдвига, набора сумматоров и умножителей. Для двоичных кодов умножители заменяются ключами с состояниями «включено/выключено».

По сути, для циклических кодов достаточно знать лишь одну строку матрицы, с помощью которой осуществляется кодирование, а оставшиеся строки выражаются через циклические перестановки исходной.

Естественно, что у циклических кодов есть как матричный способ записи процесса кодирования, так и табличный, то есть данные коды — это ограниченный по количеству (не всякий линейный код будет циклическим), но при этом весьма интересный и не совсем простой для изучения класс помехоустойчивых кодов.

В данной работе изучаются **два** двоичных циклических кода Хемминга (7, 4) с генераторными полиномами [1]

$$\begin{aligned} g_1(x) &= x^3 + x + 1, \\ g_2(x) &= x^3 + x^2 + 1. \end{aligned} \qquad (1)$$

Параметр (7, 4) означает то, что на вход кодирующего устройства подаются четыре двоичных символа (бита), а с выхода считываются семь символов, в которых содержатся четыре входных и три проверочных символа (трехбитовая контрольная сумма). Избыточность данного кода равна $3/7$.

Пусть соответствие кодового слова коэффициентам некоторого полинома строится по правилу «младший символ слева»

$$(s_0 s_1 s_2 s_3 s_3 s_5 s_6) \rightarrow s(x) = s_0 + s_1 x + s_2 x^2 + s_3 x^3 + s_4 x^4 + s_5 x^5 + s_6 x^6 \quad . \qquad (2)$$

Тогда, например, двум генераторным полиномам (1) будут соответствовать два кодовых слова

$$\begin{aligned} g_1 &= (1101000), \\ g_2 &= (1011000). \end{aligned} \qquad (3)$$

Циклические коды являются *блочными,* то есть все кодовые слова имеют постоянную длину, а так как мы рассматриваем пример кода (7, 4), то длина входных слов равна четырем, выходных — семи.

Для двоичных кодов применяется правило сложения элементов кодовых слов (и, естественно, коэффициентов полиномов) «по модулю два»

$$\begin{aligned} 0+0=0, 1+0=1, \\ 0+1=1, 1+1=0, \end{aligned} \qquad (4)$$

которое сформировано на основе обычного сложения, но так, чтобы результат никогда не выходил за пределы 0 и 1 (четные числа равносильны нулю, нечетные — единице).

Циклические коды являются блочными (*n, k*)-кодами [1], поэтому степени полиномов должны быть меньше *n*

$$x^n = x^0 = 1 \quad .$$

Рассмотрим, например, полином для $n = 7$

$$s(x) = x^4 + x + 1 \quad , \qquad (5)$$

которому по правилу (2) соответствует кодовое слово

$$(1100100) \quad .$$

Умножению полинома на *x* соответствует сдвиг кодового слова на один символ вправо. Математически, равенство $x^n = 1$ соответствует переносу бита из последней ячейки регистра памяти в первую (рис. 2). Длину выходного блока для циклических кодов можно назвать **длиной цикла** (периодом).

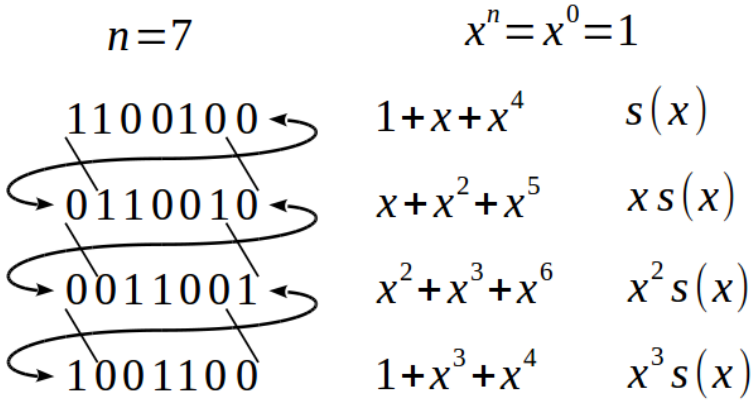


Рисунок 2 Последовательные циклические сдвиги кодового слова на один, два и три бита вправо

Правило кодирования для циклических кодов простое: входное кодовое слово преобразуется в выходное с помощью умножения входного полинома $a(x)$ на генераторный

$$s(x) = a(x)g(x) \quad . \qquad (6)$$

При этом степень генераторного полинома нет смысла брать больше, чем $r = n - k$, так как степень входного полинома по определению не боль-

ше $k-1$, что после перемножения даст степень полинома (6), меньшую n. Коэффициенты при крайних степенях генераторного (порождающего) полинома не должны быть равны нулю, иначе код снизит свою помехоустойчивость[1]. Для двоичных кодов единственным ненулевым элементом является единица, поэтому

$$g_0=g_r=1.$$

При способе кодирования (6) нельзя дать гарантию того, что в блоке на выходе кодера можно будет выделить два последовательных блока: блок информационных символов плюс блок проверочных символов. Помимо перестановок информационных символов, могут наблюдаться их функциональные преобразования (сложения по модулю два)*.

---

*Например, выходное кодовое слово кода (7, 4) с $g(x)=x^3+x+1$ для любого входного слова $(a_0a_1a_2a_3)$ равно $(a_0, a_0+a_1, a_1+a_2, a_0+a_2+a_3, a_1+a_3, a_2, a_3)$, откуда следует, что на последний двух позициях стоят два информационных символа, а на первой позиции — один информационный. Информационный символ a_1 скрыт среди второго, третьего или пятого символа выходного слова.

Зачастую бывает удобно кодировать и декодировать с разделением на два последовательных блока: блок информационных плюс блок проверочных символов. Для этого информационные символы сдвигают на r позиций вправо. Этому, как мы уже знаем, соответствует умножение $a(x)x^r$. Тем самым резервируется r нулевых позиций под проверочные символы, которые еще предстоит вычислить. Добавление вычисленных символов дает гарантию того, что итоговое кодовое слово будет разрешенным.

Чтобы некоторое кодовое слово было разрешенным, из правила кодирования (6) следует, что ему соответствующий полином должен делиться без

---

1 Желающие могут снизить степень, составить код и убедиться в этом, логически сравнивая «сниженный» и исходный коды.

остатка на генераторный (функция нахождения остатка обозначается как mod[2], полином остаток — как *res*(*x*))

$$res(x) = s(x) \mod g(x) \equiv 0. \quad (7)$$

Поэтому дальше находят остаток от деления «сдвинутого» полинома

$$[a(x) \cdot x^r] \mod g(x). \quad (8)$$

По определению степень полинома остатка всегда **меньше** степени делителя. Чтобы полином $a(x) \cdot x^r$ делился без остатка, необходимо вычесть из него найденный остаток (8), но для двоичных кодов вычитание эквивалентно сложению по модулю два

$$1 - 1 = 1 + 1 = 0.$$

В итоге выходной полином будет иметь вид

$$s(x) = a(x) \cdot x^r + [a(x) \cdot x^r] \mod g(x). \quad (9)$$

Правило (9) называют *кодированием в систематической форме* [1]. В практической части данной работы исследуется именно оно. Правило (6) — *кодирование в несистематической форме* [1].

Процесс декодирования — это:

А) **вычисление остатка** от деления полинома $v(x)$, соответствующего принятому кодовому слову, на генераторный полином;

В) **определение номеров ошибочных символов,** которые в случае двоичных кодов исправляются на обратные.

Пункт **В** является необязательным, так как декодер может работать лишь в режиме обнаружения ошибок.

Процесс декодирования можно выразить формулами

$$v(x) = s(x) + e(x), \quad res(x) = v(x) \mod g(x) = e(x) \mod g(x), \quad (10)$$

где $e(x)$ — полином, соответствующий *вектору ошибок.*

Для анализа типов ошибок вводят такое понятие как *кратность ошибки q,* которая численно равна количеству ненулевых элементов в векторе ошибок *e,* или, что одно и то же, его весу $q = w(e)$.

---

2 Например, часовая стрелка использует арифметику по модулю 12, минутная — по модулю 60.

Из (10) следует, что остаток от деления определяется лишь вектором ошибки, и не зависит от того, какое кодовое слово передавалось.

Если остаток от деления равен нулю, то ошибки либо нет, либо она такой *кратности,* что не обнаруживается декодером. Если остаток не равен нулю, то ошибка есть, и в некоторых случаях может быть исправлена.

Подытоживая предыдущий абзац, можно сказать, что декодирование — это анализ принятых символов и вынесение определенного решения из заранее составленного набора решений.

3. Примеры кодирования и декодирования

Выберем генераторный полином кода Хемминга (7, 4)

$$g_2(x)=x^3+x^2+1.$$

Зададим входное (информационное) слово (0111), которому соответствует полином

$$a(x)=x^3+x^2+x.$$

Рассмотрим оба способа кодирования (6) и (9), а также пример декодирования по правилу (10).

Кодирование в систематической форме

$$\begin{array}{l|l} x^6+x^5+x^4 & x^3+x^2+1 \\ \oplus\ x^6+x^5+x^3 & \hline x^3+x \\ \hline x^4+x^3 & \\ \oplus\ x^4+x^3+x & \\ \hline x & \end{array}$$

Число проверочных символов $r=n-k=7-4=3$. Умножим полином $a(x)$ на x^3 и найдем остаток от деления методом деления «в столбик». Получим остаток $res(x)=x$ и выходной полином $s(x)=x^6+x^5+x^4+x$, что соответствует кодовому слову $s=(\mathbf{010}0111)$, где проверочные символы выделены жирным шрифтом.

При делении в столбик следует учесть, что

$$x^m+x^m=x^m(1+1)=x^m\cdot 0=0.$$

Кодирование в несистематической форме

$$s(x)=a(x)g(x)=(x^3+x^2+x)(x^3+x^2+1)=$$
$$=x^6+x^5+x^3+x^5+x^4+x^2+x^4+x^3+x=x^6+x^2+x,$$

или $s=(\mathbf{01}10001)$. В данном случае нельзя напрямую (выделением блоков) отделить информационные и проверочные символы. **Показать,** что первые два и последний символ в выходном слове (выделены жирным шрифтом) — это информационные символы a_0 , a_1 и a_3 соответственно.

Декодирование

Разделим «в столбик» полином $s(x)=x^6+x^5+x^4+x$, соответствующий найденному разрешенному кодовому слову **систематического** кода, на выбранный генераторный полином.

$$\begin{array}{l|l} x^6+x^5+x^4+x & x^3+x^2+1 \\ \oplus\ x^6+x^5+x^3 & \hline x^3+x \\ \hline \quad x^4+x^3+x & \\ \quad \oplus\ x^4+x^3+x & \\ \hline \qquad 0 & \end{array}$$

Получили нулевой остаток, что говорит о том, что принятое кодовое слово является разрешенным, то есть оно содержится в таблице кодирования. Таблица из-за больших размеров в явном виде не составляется. Кодирующее устройство состоит всего из r (в данном случае из трех) ячеек памяти, соединенных между собой в цепочку (смотри лабораторную работу **Циклические коды-2**).

Вводя однократную ошибку, например, в предпоследний символ, то есть задавая вектор и полином ошибок в виде

$$\mathbf{e}=(0000010)\rightarrow e(x)=x^5 \ ,$$

получим полином

$$v(x)=s(x)+e(x)=(x^6+x^5+x^4+x)+x^5=x^6+x^4+x \ ,$$

деление которого даст ненулевой остаток, который можно вычислить, разделив $e(x)$ на генераторный полином.

Остаток от деления равен $x+1$ или (110) в двоичном представлении (младший бит слева).

$$
\begin{array}{l|l}
\oplus\ x^5 & x^3+x^2+1 \\
\quad x^5+x^4+x^2 & \hline
\quad\ \ x^4+x^2 & x^2+x+1 \\
\quad\ \oplus\ x^4+x^3+x & \hline
\quad\ \ \ x^3+x^2+x & \\
\quad\ \ \oplus\ x^3+x^2+1 & \hline
\quad\ \ \ \ x+1 &
\end{array}
$$

Вводя разные **однократные** ошибки (только в первый, только во второй, … , только в седьмой символы) можно убедиться, что **все остатки получаются разными.** Этот факт доказывает стопроцентное исправление рассматриваемым кодом всех однократных ошибок.

4. Описание лабораторного макета

Лабораторный макет — исполняемый файл **cyclic**, лежащий в папке **./windows** и предназначенный для запуска в операционных системах **Windows**. Программа написана на языке C++ с использованием открытых библиотек **Qt** [3] (рис. 3).

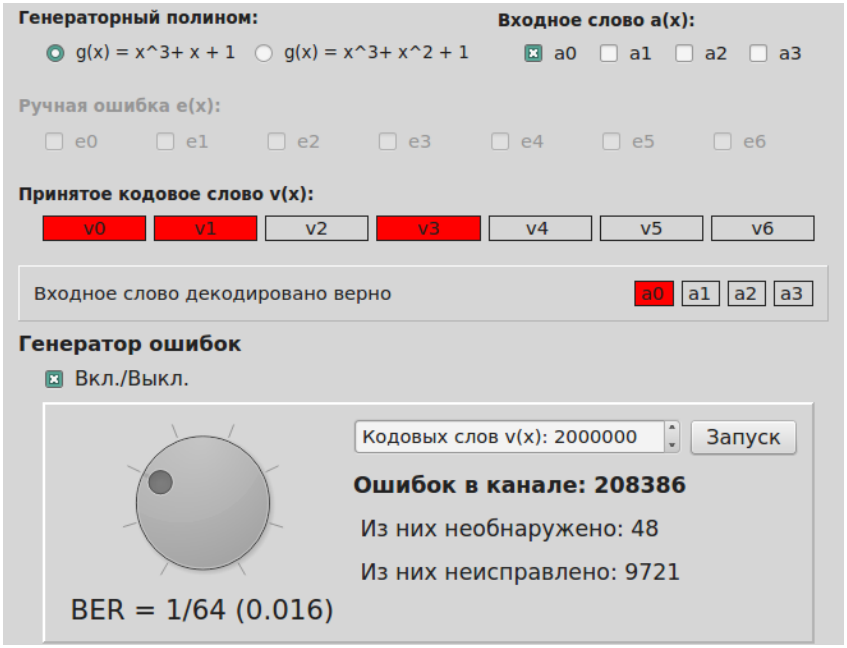


Рисунок 3 Программный макет для изучения циклического кода Хемминга (7, 4)

В макете реализовано кодирование и декодирование в систематической форме.

Генераторный полином $g(x)$ выбирается радиокнопками; входное слово $a(x)$ и ручная ошибка $e(x)$ задаются флажками (бит **1** — флажок отмечен, бит **0** — снят). Автоматически вычисляется принятое кодовое слово (красный цвет — **1**, цвет фона окна — **0**) с учетом ручной ошибки, которое затем декодируется, а результат выводится на панель, расположенную ниже панели Принятое кодовое слово v(x).

На рис. 3 результат декодирования дан в виде текста Входное слово декодировано верно и четырех информационных символов $(a_0 a_1 a_2 a_3)$, совпадающих с символами входного слова.

Панель Генератор ошибок предназначена для генерации случайных ошибок в автоматическом режиме и их последующем подсчете. Используется модель *двоичного симметричного канала связи с независимыми ошибками* (§5.3). *Битовая вероятность ошибки* в канале выставляется ручкой BER. Количество генерируемых кодовых слов задается в поле ввода Кодовых слов v(x).

Запуск статистических испытаний осуществляется соответствующей кнопкой, после чего в текстовые поля выводятся показания трех счетчиков. При включении генератора ошибок, ручная ошибка автоматически выключается.

Декодер может работать в двух режимах (рис. 4):

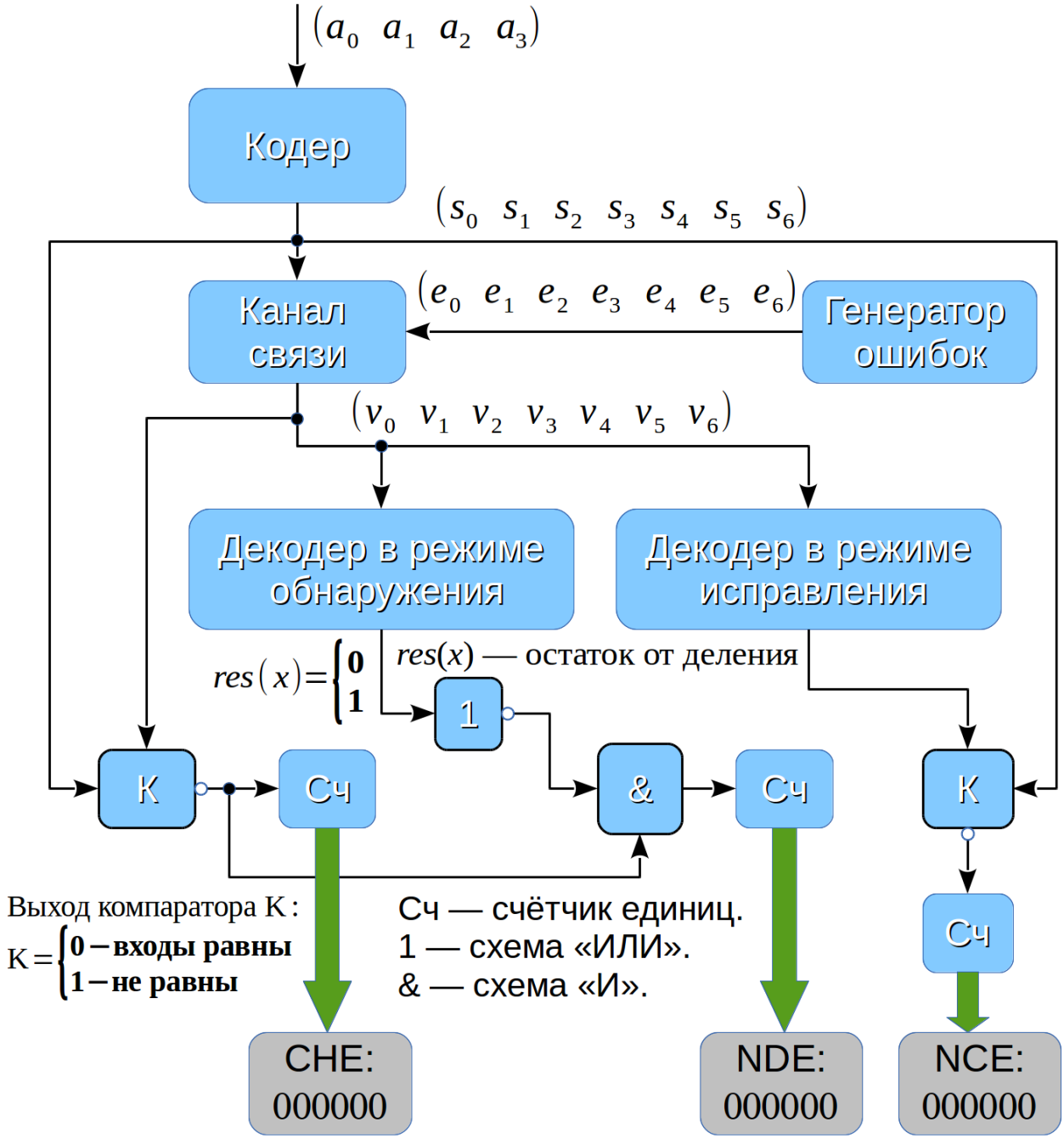
- Обнаружение ошибок;
- Исправление ошибок.

Рассматриваемый код Хемминга (7, 4) способен гарантированно обнаружить все ошибки вплоть до кратности два включительно, а гарантированно исправить — лишь однократные (детали этого выяснятся после выполнения §§5.1 и 5.2).

Если декодер работает в режиме обнаружения, то он, после обработки принятого слова, выносит одно из двух возможных решений: либо ошибка обнаружена, либо — нет. Это можно сделать благодаря сравнению остатка от деления с нулем: он либо отличен от нуля, либо — равен. Программа подсчитывает (счетчик **NDE, No Detected Error,** рис. 4) количество кодовых слов с имеющимися, но необнаруженными ошибками. Не будут обнаружены такие ошибки, которым соответствуют полиномы $e(x)$, **делящиеся без остатка** на генераторный.

Если же декодер работает в режиме исправления, то он вычисляет остаток от деления и по номеру остатка корректирует один бит из семи принятых. Если кратность ошибки больше единицы, то коррекция только навредит, зато все однократные ошибки будут исправлены. Вводя в программе ошибку вручную, в этом нужно убедиться. Программа подсчитывает (счетчик **NCE, No Corrected Error,** рис. 4) количество неисправленных кодовых слов, то есть все варианты с кратностями ошибки выше единицы.

Счетчик **CHE** (**Channel Error,** рис. 4) выдает количество кодовых слов, пришедших на вход декодера с ошибкой (хотя бы в одном бите из семи).



CHE — число ошибочных кодовых слов на выходе канала связи (на входе декодера).
NDE — число необнаруженных ошибок на выходе декодера.
NCE — число неисправленных ошибок на выходе декодера.

Рисунок 4 Блок-схема работы лабораторного макета «Циклический код (7, 4)»

5. Порядок выполнения работы

Для экономии времени желательно проделать три пункта по порядку:

1. Сделать расчетное задание;

2. Ответить на вопросы, по возможности соблюдая порядок 1, 2, 3,...;

3. Выполнить экспериментальную часть работы.

5.1 Расчетное задание

Найти путем деления в столбик **все** разрешенные кодовые слова для полиномов $g_1(x)$ и $g_2(x)$ **систематического** кода. **Свести** результаты в две таблицы А и В. Вычислить вес *w* (количество единиц) всех кодовых слов.

Таблица А. Разрешенные кодовые слова систематического циклического кода (7, 4)
$g_1(x)=x^3+x+1$

№	r_0	r_1	r_2	a_0	a_1	a_2	a_3	w
				0	0	0	0	
1				1	0	0	0	
2				0	1	0	0	
3				1	1	0	0	
4				0	0	1	0	
5				1	0	1	0	
6				0	1	1	0	
7				1	1	1	0	
8				0	0	0	1	
9				1	0	0	1	
10				0	1	0	1	
11				1	1	0	1	
12				0	0	1	1	
13				1	0	1	1	
14				0	1	1	1	
15				1	1	1	1	

Таблицу В сделать аналогично таблице А, но для зеркального полинома

$$g_2(x)=x^3+x^2+1 \quad .$$

Проверить найденные кодовые слова с помощью программы **cyclic**. Взять наугад из таблиц А и В по одному полиному и разделить его на соответствующий генераторный полином. Показать, что остатки от деления равны нулю, тем самым вручную подтвердив правильность кодирования.

Выписать в таблицу С полиномы остатки *res*(*x*) от деления целых степеней x^i на полиномы $g_1(x)$ и $g_2(x)$.

Таблица С. Остатки от деления целых степеней на генераторные полиномы

Генераторный полином $g_1(x)$		Генераторный полином $g_2(x)$	
Степень x	Остаток	Степень x	Остаток
x^0		x^0	
x^1		x^1	
x^2		x^2	
x^3		x^3	
x^4		x^4	
x^5		x^5	
x^6		x^6	

После **правильного** выполнения расчетного задания ответить на вопросы, подготовив себя к практической части лабораторной работы. Ответы **привести** в отчете по ходу работы.

5.2 Анализ результатов выполнения расчетного задания

На каждый вопрос дать два ответа: для первого кода и второго

1. Есть ли среди разрешенных кодовые слова с весом пять (то есть состоящие из пяти единиц)? Шесть? Семь?

2. Определить веса, по которым сразу определяются запрещенные кодовые слова (например, **один,** так как если в слове только одна единица, то оно однозначно запрещенное. Проверить **два, три, ... ,семь**).

3. Какие веса может иметь разрешенное кодовое слово?

4. Обязано ли произвольное слово из семи бит с весом четыре быть разрешенным? С весом ноль (то есть все семь битов — нули)?

5. Сколько в принципе существует разных двоичных слов из семи битов с весом четыре? Сколько подобных слов среди разрешенных?

6. Если к любому разрешенному кодовому слову прибавить любое, имеющее вес два (то есть ввести двукратную ошибку),то обнаружится ли эта ошибка? Всегда? Анализ сделать на основании остатков от деления.

7. Если ввести трехкратную ошибку, обнаружится ли она? Всегда? Если не всегда, то каков процент того, что ошибка не будет обнаружена?

8. Справится ли декодер с двукратной ошибкой, если он работает в режиме исправления? С трех-, четырех-, пяти-, шести-, семикратной? Для анализа использовать таблицу С.

5.3 Экспериментальная часть

В программном лабораторном макете предусмотрены два режима ввода ошибки в кодовые слова кода Хемминга (7, 4):

1. Режим ручной ошибки, когда оператор отмечает/снимает отметки на панели **Ручная ошибка *e*(*x*)** (рис. 2). Если флажок отмечен, то соответствующий бит в принятом кодовом слове инвертируется (**0** на **1**, **1** на **0**).
2. Режим генерации ошибок, когда оператор задает количество испытаний и после нажатия кнопки **Запуск** программа автоматически случайным образом будет вводить ошибки в неизменное разрешенное кодовое слово.

Режим ручной ошибки

Запустив программу и выбрав наугад полином и входное кодовое слово, вручную ввести следующие ошибки:

- Всевозможные однократные (их семь);
- Две трехкратные, но так, что первую обнаруживает, а вторую — нет;

- Две четырехкратные, одну из которых обнаруживает, другую — нет;
- Все шестикратные (их семь);
- Семикратную (она одна).

Результаты декодирования **привести** в отчете (копируя результаты из программы, но желательно не скриншотами).

Почему одну выбранную Вами трехкратную ошибку обнаруживает, другую — нет? Четырехкратную? **Ответить** на два данных вопроса.

Режим генерации ошибок

В данной работе рассматривается модель двоичного симметричного канала связи с независимыми ошибками (рис. 5), возникающими в каждом двоичном символе (бите) с вероятностью p (вероятность битовой ошибки, BER, *Bit Error Rate*).

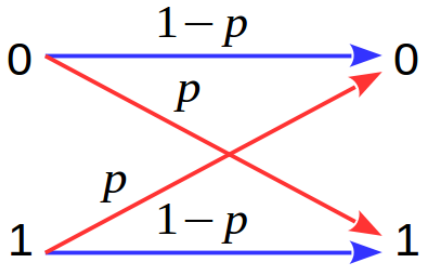


Рисунок 5 Модель двоичного симметричного канала связи с независимыми ошибками

Предварительно сделав расчетное задание (§5.1) и ответив на вопросы (§5.2), **для двух** рассматриваемых кодов $(7, 4)$ **определить** в виде формул три вероятности:

- Вероятность P_1 того, что кодовое слово длиной семь бит, поступающее на вход декодера из канала связи, **будет содержать ошибку;**
- Вероятность P_2 того, что содержащаяся в принятом кодовом слове ошибка **не будет обнаружена;**
- Вероятность P_3 того, что имеющаяся ошибка **не будет исправлена.**

При выводе трех формул использовать формулу Бернулли для вероятности $P(q)$ количества успехов q при проведении n независимых статистических испытаний

$$P(q)=C_n^q p^q (1-p)^{n-q},$$

где p — вероятность успеха (для нас «успех» — это битовая ошибка);

n — длина кодового слова $(n=7)$;

$C_n^q = \frac{n!}{q!(n-q)!}$ — число сочетаний из n по q .

Например, если $q=2$, то это означает, что в кодовом слове из семи бит произошла ошибка в двух любых битах. Число разных двукратных ошибок в нашем случае составляет $C_7^2 = 21$. Помимо факториальной формулы, числа сочетаний легко могут быть вычислены по треугольнику Паскаля.

Также при выводе формул **учесть** результаты ответов на вопросы из §5.2.

Вычислить по найденным формулам три вероятности $\{P_1, P_2, P_3\}$ для набора BER

$$p = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{128}, \frac{1}{256} \right).$$

Занести результаты в таблицу.

С помощью программы **cyclic,** включив генератор ошибок, провести статистический эксперимент по обнаружению и исправлению ошибок **для обоих** рассматриваемых кодов. Результаты измерений **занести** в таблицу. Входное слово можно задавать любым (кстати, почему?). Число испытаний выбрать из диапазона 2’000’000–5’000’000 и **не изменять** на всем протяжении измерений.

Оценить по показаниям трех счетчиков три вероятности $\{\hat{P}_1, \hat{P}_2, \hat{P}_3\}$ и **сравнить** их с вычисленными теоретическими значениями $\{P_1, P_2, P_3\}$.

Перенести файлы **decoder_error_distrib_inv_p_X,** созданные программой при проведении статистических испытаний, в таблицу, где параметр **X** означает обратную вероятность битовой ошибки в канале p . В данных файлах записано экспериментальное распределение кратностей m ошибок **на выходе декодера.** Оценить по этим данным битовую вероятность ошибки на выходе декодера $\hat{p}_{\text{out}}$ в режиме исправления и построить график зависимости $\hat{p}_{\text{out}}(p)$ в логарифмическом масштабе по обеим осям.

Дозаполнить с помощью программы **cyclic** (в режиме ручной ошибки) таблицу D.

Таблица D. К определению частоты битовых ошибок на выходе декодера

Кратность ошибки q	Число разных ошибок кратности q, искажающих m битов на выходе декодера			
	$m=1$	$m=2$	$m=3$	$m=4$
2	… из 21	… из 21	… из 21	… из 21
3	7 из 35	15 из 35	13 из 35	0 из 35
4	13 из 35	15 из 35	7 из 35	0 из 35
5	3 из 21	9 из 21	9 из 21	0 из 21
6	… из 7	… из 7	… из 7	… из 7
7	0 из 1	0 из 1	0 из 1	1 из 1

На основании таблицы D **вывести** формулу для вычисления вероятности битовой ошибки на выходе декодера (7, 4) в режиме исправления p_{out} в зависимости от вероятности битовой ошибки на его входе p. Отобразить найденную зависимость $p_{\text{out}}(p)$ вместе с экспериментальным графиком $\hat{p}_{\text{out}}(p)$.

При какой BER эффективность использования кода равна нулю (когда битовые вероятности на входе и выходе декодера становятся равными)?

В отчете все сделанные выводы собрать воедино. **Ответить на вопросы:**

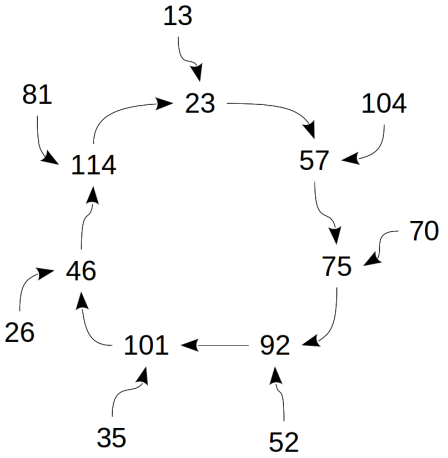
1. Чем отличаются два рассмотренных кода?
2. Какой из них более помехоустойчивый и, если это так, то почему?

6. Литература

1. Теория и техника передачи информации: Учебное пособие / Ю.П. Акулиничев, А.С. Бернгардт. — 2012, 210 с.
2. И. Шахнович. DVB-T2 – новый стандарт цифрового телевизионного вещания // Электроника: наука, технология, бизнес, 6/2009, http://www.electronics.ru/files/article_pdf/0/article_258_849.pdf.
3. Qt | Cross-platform application & UI development framework, http://www.qt.io/.

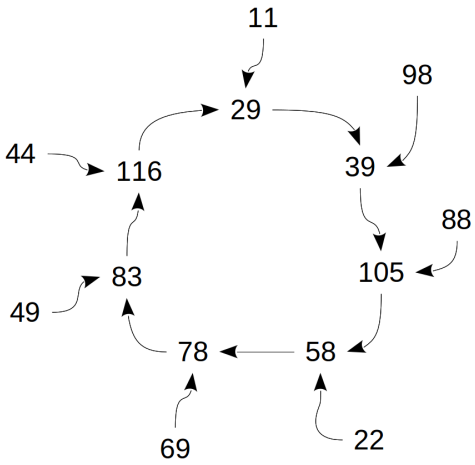
Для заметок

ЦИКЛИЧЕСКИЕ КОДЫ-2



23=0010111

$$\frac{1}{x^3+x+1}=1+x+x^2+x^4+\ldots$$



29=0011101

$$\frac{1}{x^3+x^2+1}=1+x^2+x^3+x^4+\ldots$$

1. Введение

Для более подробного введения в теорию помехоустойчивых кодов обратитесь к работе **Циклические коды-1,** которую рекомендуется изучить и сделать в первую очередь.

В данной лабораторной работе изучаются регистры сдвига с линейной обратной связью, являющиеся базовыми устройствами при схемной реализации кодеров и декодеров циклических кодов. Выбор пал на достаточно короткие коды Хемминга (7, 4) и (15, 11).

Методические требования к выполнению работы:

- Работу по возможности выполнять последовательно, так как задания связаны между собой;
- Теорию рекомендуется читать полностью, проверяя на «бумажке с карандашом» выводы формул.

Структура отчета должна соответствовать следующему:

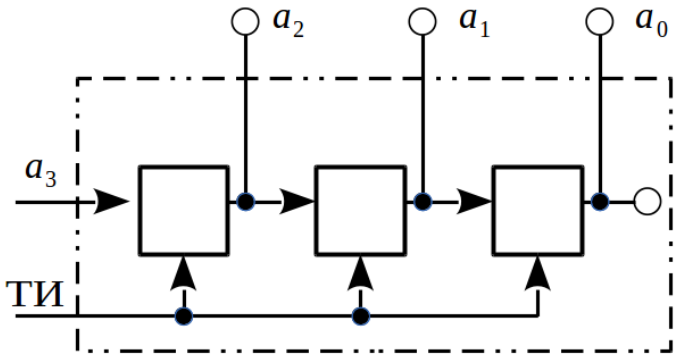
✔ Титульный лист;
✔ Ход работы;
✔ Ответы на вопросы;
✔ Выводы.

Принимаются форматы файлов **doc, docx, odt** и **pdf.**

2. Сведения из теории

Так как циклические коды полностью задаются полиномом (генераторным или проверочным), то схемная реализация кодера-декодера является **самой экономной** среди линейных блочных кодов и представляет собой **один** (меньше не бывает) *регистр сдвига.*

Регистр сдвига — это цифровое устройство, состоящее из последовательно соединенных ячеек памяти (рис. 6). Слово «цифровое» в данном контексте означает то, что по физическому состоянию ячейки регистра можно однозначно определить какой символ из алфавита A хранится в ней в данный момент.



ТИ — тактовые импульсы

Рисунок 6 Пример функциональной схемы трехразрядного регистра сдвига

Исторически сложилось так, что для представления информации зачастую используется алфавит всего лишь из двух символов — битов $A=\{0,1\}$. Примером тому являются все широко распространенные модули оперативной памяти компьютеров, реализованные в транзисторной логике. Слово «транзистор» здесь является ключевым, так как транзистор имеет два устойчивых режима работы: режим «открыт» (максимальное потребление тока) и режим «закрыт» (минимальное потребление тока). Поэтому с их помощью выгодно хранить именно биты, для чего потребуется как минимум два транзистора — *триггер* [1].

Если мы попытаемся с помощью того же количества транзисторов хранить трехзначные символы (триты), то потребуется вводить транзистор в «средний» (полуоткрытый, линейный) режим работы. Это приведет к усложнению схемотехники и, кроме того, повысится вероятность ошибки при чтении-записи данных, так как необходимо различать три интервала уровней напряжения (тока, заряда), а не два.

Однако, в настоящее время имеются карты памяти и твердотельные диски с тремя [2] и даже четырьмя [3] битами на ячейку памяти! Это связано с желанием повысить плотность записи в накопителях (в конечном итоге — их объем в гигабайтах), а с возникающими ошибками помогают бороться корректирующие коды (в частности, циклические).

Слово «сдвиг» в словосочетании «регистр сдвига» означает то, что все символы, хранящиеся в текущих ячейках, после прихода *тактового импульса* практически одновременно переходят в следующие ячейки в соответствии со стрелками.

Индексы у символов a_i на рис. 6 означают *дискретные моменты времени.* Во время прихода тактового импульса состояние текущей ячейки передается следующей, и после завершения всех передач появляется интервал времени, когда с регистра можно считывать данные. Этот интервал времени нумеруется индексом.

Например, если состояние $a_2 a_1 a_0$ регистра на рис. 6 соответствует моменту времени с индексом 0, то в следующий момент времени (индекс 1) регистр перейдет в состояние $a_3 a_2 a_1$: цепочка $a_2 a_1 a_0$ сдвинется вправо, добавится **новый** символ a_3 , а символ a_0 в зависимости от назначения регистра потеряется или пойдет на вход следующего устройства.

В принципе, можно себе представить, что **новые** символы хранятся в неограниченно длинном регистре, расположенном со стороны входа рассматриваемого регистра, и такт за тактом поступают на его вход. Аналогичная ситуация и с потерявшимися символами: можно представить, что они уходят в неограниченно длинный регистр, находящийся со стороны выхода рассматриваемого.

Зачастую используются двоичные регистры, двоичные коды (хотя в настоящее время это под вопросом), поэтому символы a_i могут принимать всего лишь два значения, которые условно обозначаются как **0** и **1.** Однако,

никто не мешает группировать биты, например, в байты и анализировать регистры, **за один такт** передвигающие информацию **на один байт.**

По сути, регистр сдвига — это буфер, предназначенный для хранения и преобразования цифровой информации, последовательно поступающей на его вход. На базе таких регистров строятся цифровые фильтры, в частности, кодеры и декодеры циклических кодов.

Любую точку регистра сдвига математически можно представить в виде полинома, что активно используется при изучении циклических кодов.

Любая ячейка памяти имеет вход и выход. Направление передвижения символов указывается стрелкой. В любой момент времени на входе имеется один символ, на выходе — другой, так как выход текущей ячейки соединяется со входом следующей.

В зависимости от индексации символов, логически различают два типа ячеек памяти: **повышающая и понижающая,** хотя физически они ничем не отличаются (рис. 7).

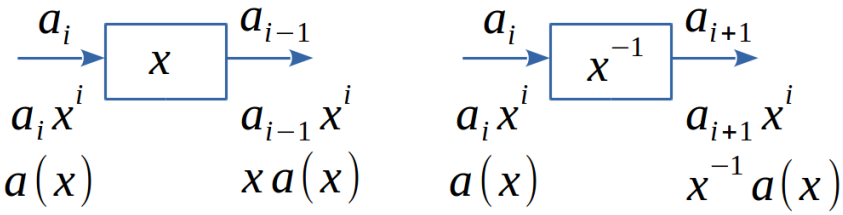


Рисунок 7 Ячейки памяти двух логических типов: повышающая и понижающая

Состояние ячеек на рис. 7 «сфотографировано» в момент времени с индексом *i,* чему по определению соответствует степень x^i . Слагаемое $a_i x^i$ содержится в полиноме $a(x)$, а слагаемое $a_{i-1} x^i$ — в полиноме $x\,a(x)$ и так далее.

Математически выгодно количество слагаемых в полиноме ничем не ограничивать, то есть полагать, что

$$a(x)=\sum_{i\in\mathbb{Z}} a_i x^i \quad , \quad \mathbb{Z} \quad \text{— множество целых чисел.}$$

Это дает право не задумываться про индексы при разного рода преобразованиях: умножении двух полиномов, сложении, делении.

Рассмотрим для примера регистр сдвига, изображенный на рис. 8. Он меняет свое состояние с каждым приходом **тактового импульса** (ТИ) clock(t), реагируя на перепад напряжения снизу-вверх (на передний фронт). После смены состояния в течение времени с индексом i можно считывать данные.

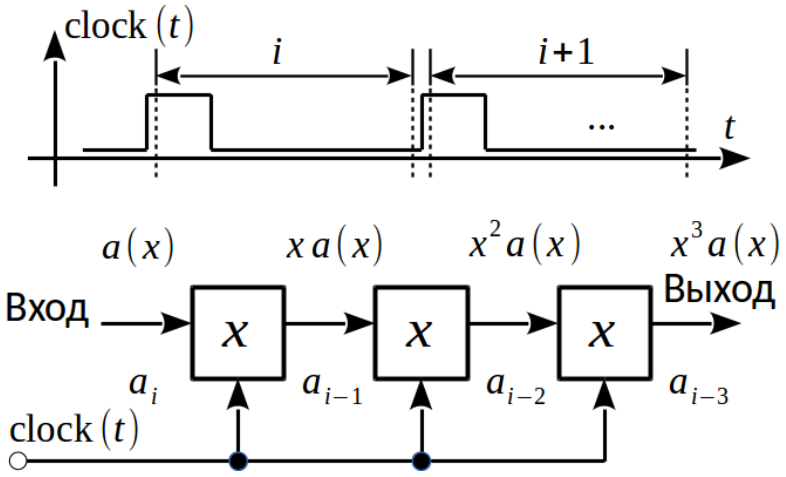


Рисунок 8 Трехразрядный регистр на повышающих ячейках

Глядя на рис. 8, можно вычислить коэффициент передачи данного регистра как отношение выходного полинома ко входному

$$K(x)=\frac{x^3 a(x)}{a(x)}=x^3 \quad .$$

Последнее выражение говорит о том, что рассматриваемый фильтр выполняет задержку входных символов (цифр) на три такта.

Простейший способ кодирования с помощью циклического кода — умножение информационного полинома $a(x)$ на генераторный $g(x)$. Из только что сделанных выводов мы можем заключить, что для такого кодирования необходим фильтр с коэффициентом передачи, совпадающим с генераторным полиномом.

Рассмотрим, например, полином циклического кода Хемминга (7, 4)

$$g(x)=x^3+x+1 \quad ,$$

в котором присутствуют три слагаемых, откуда следует, что на рис. 8 не хватает *сумматоров*.

Сумматоры складывают цифры из ячеек регистра в некоторый момент времени. Считается, что **они** являются безынерционными элементами, то есть мгновенно складывающими, но на самом деле (схемотехнически) их инерционность много меньше длительности одного такта и ею можно пренебречь. Сумматор имеет, как минимум, два входа и один выход.

Математические операции с символами-цифрами не должны приводить к переполнениям (количество цифр ограничено), поэтому сумматоры выполняют операцию **суммирования по модулю** некоторого числа, являющегося основанием кода. Для двоичных кодов используются **сумматоры по модулю два ⊕,** принцип сложения которых следующий.

Складываем две единицы, получаем двойку, которой нет в алфавите, поэтому берем двойку по модулю два, что дает ноль. Модуль — это число, которое следует вычитать из некоторого результата суммирования (или прибавлять к нему) до тех пор, пока сумма не совпадет с одним из чисел-символов используемого алфавита. Приведение по модулю однозначно. Числовая ось из целых чисел отобразится по модулю два в числовую ось из чередующихся нулей и единиц ...0,1,0,1,0,1,0,1..., что соответствует простому признаку четное нечетное.

Например, рассмотрим алфавит из трех символов $\{-1,0,1\}$. Модуль равен трем, так как в алфавите три символа. Вычислим произвольную сумму

$$-1+1+1-1-1+1+0+1+1=2\equiv(2-3)\bmod 3=-1 \quad .$$

Знак $\equiv$ означает то, что число 2 **сравнимо** с числом –1 по модулю 3. Коротко данный факт описывается так: $2\equiv -1 \bmod 3$.

Исходя из только что озвученных правил, составим схему кодера двоичного циклического кода Хемминга (7, 4), записанного в **несистематической** форме (рис. 9).

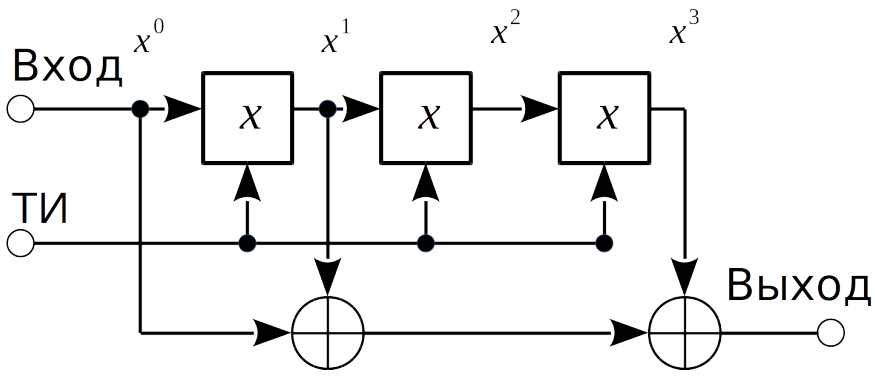


Рисунок 9 Схема несистематического кодера двоичного циклического кода Хемминга (7, 4) *с генераторным полиномом* x^3+x+1

Как видно из рис. 9, тактовые импульсы на сумматоры не подаются, так как они (теоретически) не нуждаются в тактировании.

Проверим правильность работы схемы. Зададим входное кодовое слово $a=(0011)$, которому соответствует полином

$$a(x)=0x^0+0x^1+1x^2+1x^3=x^3+x^2 .$$

Проведем кодирование алгебраически

$$s(x)=a(x)g(x)=(x^3+x^2)(x^3+x+1)=x^6+x^4+x^3+x^5+x^3+x^2=x^6+x^5+x^4+x^2 .$$

Выходное кодовое слово будет равно $s=(0010111)$.

Закодируем то же самое кодовое слово с помощью рассматриваемой схемы. Для этого подадим входное кодовое слово младшим битом вперед и составим соответствующую таблицу (табл. 1), каждая строка которой соответствует некоторому дискретному моменту времени. В начальный момент времени регистр обнулен.

Табл. 1. Потактовая работа кодера (рис. 9)

x^0	x^1	x^2	x^3	s_i
0	0	0	0	0
0	0	0	0	0
1	0	0	0	1
1	1	0	0	0
0	1	1	0	1
0	0	1	1	1
0	0	0	1	1

Столбец x^0 и вход схемы совпадают. Столбец s_i вычисляется как сумма по модулю два столбцов x^0, x^1 и x^3, то есть в точном соответствии с генераторным полиномом. Старшая степень выходного полинома определилась на седьмом такте.

На данном этапе мы детально разобрали **принцип умножения двух полиномов** с помощью регистра сдвига и набора сумматоров. Для завершения теоретического обзора необходимо разобрать **принцип деления двух полиномов,** что заметно сложнее, так как появляется остаток от деления.

Для возможности деления в схему регистра сдвига с помощью сумматоров необходимо ввести обратную связь. Именно с помощью сумматоров, так как в поле двоичных чисел нет другой аддитивной операции, кроме как суммы по модулю два.

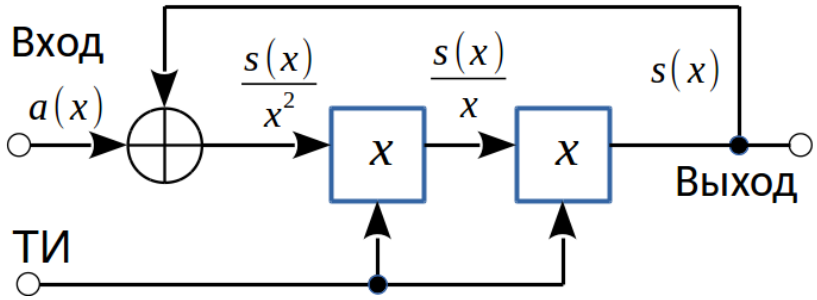


Рисунок 10 Двухразрядный регистр сдвига с обратными связями — пример схемы деления двух полиномов

Рассмотрим регистр из двух ячеек памяти с обратной связью (рис. 10).

Так как схема содержит один сумматор, то для определения коэффициента передачи потребуется одно уравнение, связывающее два полинома

$$a(x)+s(x)=\frac{s(x)}{x^2} \quad \Rightarrow \quad K(x)=\frac{s(x)}{a(x)}=\frac{x^2}{x^2+1} \quad .$$

Сумматор имеет два входа: на первый поступает входной полином $a(x)$, на второй — полином $s(x)$ с выхода схемы по цепи обратной связи.

На выходе сумматора — опереженный на два такта выходной полином $s(x)$, что полностью согласуется с правилом умножения или деления полинома на степени переменной x (рис. 7). Также следует учесть, что знак ⊕ в рассматриваемом примере является суммой по модулю два, поэтому слагаемые переносятся за знак равенства **без изменения знака:** вычитание эквивалентно сложению по модулю два, так как $1-1=1+1=0$.

Множитель x^2 в числителе коэффициента передачи — результат того, что выход помещен в конец схемы деления. Знаменатель коэффициента передачи соответствует структуре схемы деления:

- ✔ Две ячейки, значит степень полинома-знаменателя равна двум;
- ✔ Сумматора между двумя ячейками нет, значит слагаемое x в знаменателе отсутствует.

Для более детального усвоения принципа деления двух полиномов рассмотрим трехразрядный регистр сдвига с обратными связями (рис. 11).

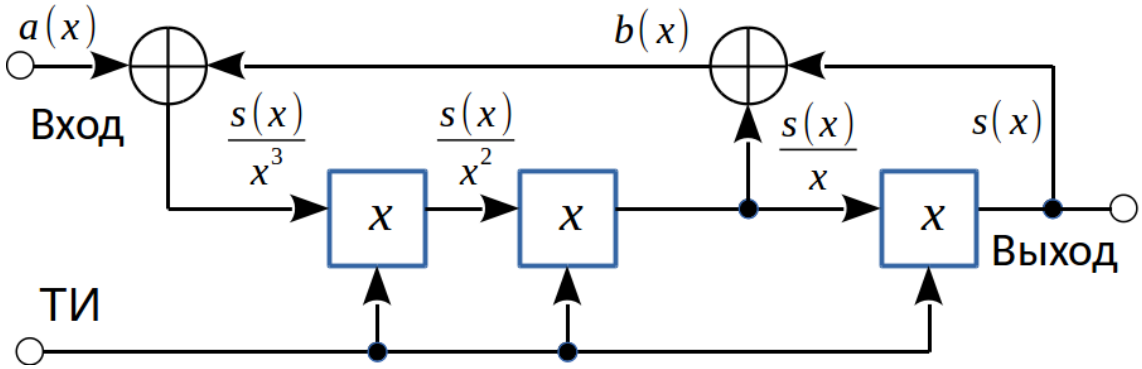


Рисунок 11 Трехразрядный регистр сдвига с обратными связями

В трехразрядном регистре появляется возможность несимметричного расположения сумматоров, что позволяет определить направление возрастания степени знаменателя коэффициента передачи.

Вывод выражения для коэффициента передачи достаточно прост, стоит лишь составить два уравнения (количество уравнений совпадает с количеством сумматоров)

$$s(x)+\frac{s(x)}{x}=b(x) \quad , \quad b(x)+a(x)=\frac{s(x)}{x^3} \quad .$$

Исключая полином $b(x)$, получим результат

$$K(x)=\frac{s(x)}{a(x)}=\frac{x^3}{x^3+x^2+1} \quad .$$

Отсюда следует, что степень полинома делителя возрастает слева-направо, то есть от входа к выходу схемы. Запоминать это не желательно, так как дальше будут модификации схем, которые изменят этот порядок, хотя составить на листочке шаблон (их будет четыре) весьма полезно.

Для того, чтобы проверить правильность найденного выражения для коэффициента передачи, разложим его в ряд Тейлора по степеням переменной x . Для этого очень удобно (экономит время) использовать программы компьютерной алгебры, например, *Maxima, Mathcad, SymPy (Python)* и другие подобные

$$\frac{x^3}{x^3+x^2+1}=x^3-x^5-x^6+x^7+2x^8-3x^{10}-2x^{11}+3x^{12}+5x^{13}-x^{14}+\ldots\equiv$$
$$\equiv x^3+x^5+x^6+x^7+x^{10}+x^{12}+x^{13}+x^{14}+\ldots.$$

При выводе последнего выражения (сравнение $\equiv$) учитывалось, что коэффициенты полиномов — биты, то есть числа **0** и **1,** поэтому коэффициенты обычного разложения в ряд брались по модулю два.

Коэффициент передачи любой схемы, состоящей из регистра сдвига и набора сумматоров, позволяет напрямую записать последовательность на выходе схемы, если на вход была подана последовательность, соответствующая единичному полиному

$$a(x)=\sum_{i\geq 0} a_i x^i=1x^0+0x^1+0x^2+0+\ldots=1 \quad .$$

Такую последовательность называют *дельта-последовательностью.*

Исходя из выше найденного разложения для коэффициента передачи, сразу определяем последовательность на выходе, считывая биты-коэффициенты начиная с младшей степени *x*

$$s(x)=K(x)a(x)=K(x)\rightarrow s=(000\mathbf{1011100}10111\ldots) \quad .$$

Первые три нуля — результат действия числителя x^3, вносящего лишь задержку на три такта и ни на что более не влияющего. После трех нулей идет некоторая последовательность битов, которая неизбежно должна повториться (период выделен **жирным** шрифтом), так как состояние регистра (рис. 11) полностью определяется состоянием трех ячеек (их входов или выходов — как удобнее).

В общем, если полностью убрать вход и задать начальную единицу в любой точке регистра (или несколько единиц), то он начнет генерировать периодическую последовательность символов.

Максимально возможный период определяется количеством различных ненулевых кодовых слов с разрядностью, равной количеству ячеек памяти регистра. В данном случае у нас три ячейки, поэтому максимально возможный период равен семи. Если регистр обнулить и подавать на него тактовые

импульсы, то он не изменит своего состояния, поэтому нулевую комбинацию исключают.

Анализируя последовательность на выходе рассматриваемого регистра, можно убедиться, что ее период равен семи, то есть максимальному периоду при заданной длине регистра. Подтвердим это непосредственным расчетом по схеме, составив таблицу (табл. 2). В начальный момент времени регистр как всегда обнулен.

Табл. 2. Потактовая работа регистра на рис. 11 при подаче на его вход дельта-последовательности

$a(x)$	$b(x)$	$\frac{s(x)}{x^3}$	$\frac{s(x)}{x^2}$	$\frac{s(x)}{x}$	$s(x)$
1	0	1	0	0	0
0	0	0	1	0	0
0	1	1	0	1	0
0	1	1	1	0	1
0	1	1	1	1	0
0	0	0	1	1	1
0	0	0	0	1	1
0	1	1	0	0	1
0	0	0	1	0	0
0	1	1	0	1	0
0	1	1	1	0	1

В качестве контрольных точек, определяющих состояние регистра, мы взяли три входа ячеек памяти, за которые отвечают третий, четвертый и пятый столбцы таблицы (выделены серым). Остальные столбцы являются зависимыми (вспомните два уравнения при выводе коэффициента передачи), и приведены для удобства расчетов и наглядности.

Анализируя табл. 2, отмечаем полное совпадение последнего столбца с результатами разложения коэффициента передачи в ряд Тейлора. Можно убедиться, что точка схемы $s(x)/x^3$ соответствует коэффициенту передачи с единичным числителем и, по сути, является опереженной (сдвинутой на три строки вверх) версией точки $s(x)$.

К сведению: если полином *r*-разрядного регистра сдвига с обратными связями выбран так, что регистр при подаче на его вход дельта-последовательности выдает последовательность с максимальным периодом, то такое устройство становится генератором *псевдослучайной последовательности (М-последовательности)* с периодом $M=2^r-1$.

Во всех рассмотренных случаях деления полиномов мы подавали последовательности **младшим** коэффициентом вперед, поэтому на выходе получали последовательность, соответствующую разложению коэффициента передачи в ряд Тейлора по степеням *x*. Однако для кодирования **систематическим** циклическим кодом необходим **остаток от деления** полинома $x^r a(x)$ на генераторный полином *g*(*x*)

$$s(x)=x^r a(x)+[x^r a(x)] \bmod g(x) \quad . \tag{11}$$

Остаток от деления легко находится путем деления в столбик, процедура которого проводится начиная со **старшей** степени делимого, так что делимое шаг за шагом уменьшает свою степень до тех пор, пока не станет строго меньше степени делителя.

Рассмотрим остатки $res_i(x)$ от деления x^i , $i\geq 0$, на $g(x)$:

$g(x)=x^3+x^2+1$,

$res_0(x)=x^0 \bmod g(x)=x^0$, $res_1(x)=x^1$, $res_2(x)=x^2$,

$res_3(x)=x^3 \bmod g(x)=x^2+1$ так как $x^3=1\cdot(x^3+x^2+1)+x^2+1$,

$res_4(x)=x^4 \bmod g(x)=(x\cdot x^3) \bmod g(x)=x(x^2+1)=$
$=x^3+x=res_3(x)+res_1(x)=x^2+x+1$,

$res_5(x)=(x\cdot x^4) \bmod g(x)=x(x^3+x)=x^4+x^2=res_4(x)+res_2(x)$,

и так далее, что в итоге дает **рекуррентную** формулу для последовательного вычисления всех базовых остатков

$$res_n(x)=res_{n-1}(x)+res_{n-3}(x) \quad , \tag{12}$$

которые начнут повторяться с некоторого *n*, то есть $res_n(x)$ — это периодическая функция дискретного аргумента *n*.

Если мы захотим разделить сумму двух слагаемых, например, x^3+x^5, и найти остаток, то он будет равен сумме остатков от деления x^3 и x^5 по отдельности (это доказывается легко, если учесть, что $(a+b)/c=a/c+b/c$). Поэтому, в итоге, любые остатки выражаются через сумму базовых. Мало того, сами базовые остатки выражаются через сумму действительно базовых

$$res_0(x)\ ,\quad res_1(x)\ ,\ldots,\quad res_{r-1}(x)\ .$$

Значит, чтобы с помощью схемы получить остаток от деления, необходимо входную последовательность подавать **старшим коэффициентом вперед.**

Делаем вывод: порядок следования входных и выходных коэффициентов задается исходя из назначения схемы, и именно этот порядок определяет правило деления и умножения полиномов на степени *x*.

Учитывая выше сказанное, преобразуем схему на рис. 11, чтобы знаменатель у коэффициента передачи не изменился (рис. 12), при этом не следует забывать, что последовательности на входе и выходе будут идти старшим коэффициентом вперед.

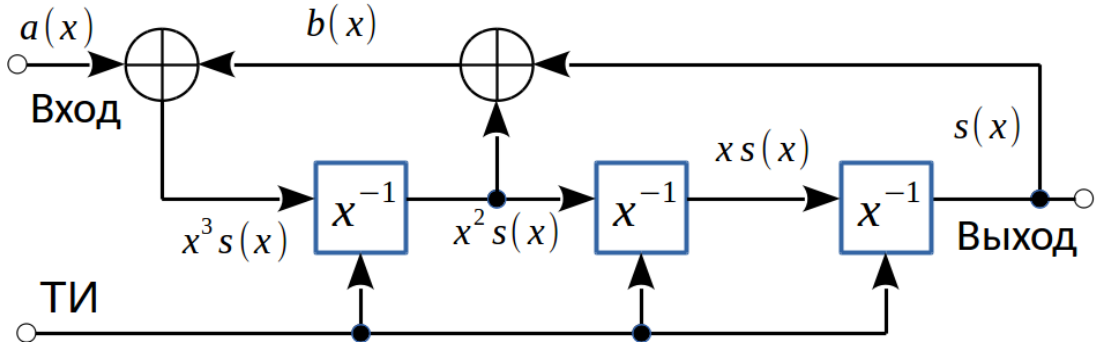


Рисунок 12 Трехразрядный регистр сдвига с обратными связями

Коэффициент передачи рассматриваемой схемы равен

$$K(x)=\frac{1}{x^3+x^2+1}\ .$$

Теперь задержка выхода на три такта (три первых гарантированных нуля на выходе) будет означать то, что частные от деления $1, x$ и x^2 на полином-знаменатель будут равны нулю.

Чтобы коэффициент передачи (знаменатель) не изменился, сумматоры мы расставили в том же порядке, но справа-налево, а не слева-направо, по сути, сохранив структуру полинома, если следовать порядку возрастания или убывания степеней *x*.

Если на рассматриваемую схему подать дельта-последовательность $a(x)=1$, то на выходе будет последовательность, отвечающая за частные от деления целых степеней *x* на полином-знаменатель: первая строка отвечает за деление x^0, вторая строка — за деление *x,* третья строка — за деление x^2 и т. д. Это можно объяснить тем, что левее входа можно представить неограниченно длинный регистр, ячейкам которого соответствуют полиномы $xa(x)$, $x^2a(x)$ и т. д. при $a(x)=1$.

Подтвердим сказанное таблицей (табл. 3). Результаты ручного деления в столбик целых степеней *x* на полином x^3+x^2+1 даны в табл. 4.

Выбирая для примера седьмую строку табл. 4 (выделена серым), получаем частное от деления в виде полинома x^3+x^2+x.

В табл. 3 из столбца $s(x)$ **выделим** первые семь элементов. Мы помним, что старшая степень идет первой по времени, то есть она отображена в первой строке. Три первых элемента — нули, поэтому шестой, пятой и четвертой степеней в частном не будет. Далее идут три единицы, значит будут степени 3, 2 и 1. И, наконец, последний нуль говорит о том, что нулевой степени не будет.

Табл. 3. Потактовая работа регистра (рис. 12)

$a(x)$	$b(x)$	$x^3s(x)$	$x^2s(x)$	$xs(x)$	$s(x)$
1	0	1	0	0	**0**
0	1	1	1	0	**0**
0	1	1	1	1	**0**
0	0	0	1	1	**1**
0	1	1	0	1	**1**
0	0	0	1	0	**1**
0	0	0	0	1	**0**
0	1	1	0	0	1

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

Табл. 4. Деление степеней x на полином x^3+x^2+1

Делимое	Частное	Остаток
x^0	0	1
x^1	0	x
x^2	0	x^2
x^3	1	x^2+1
x^4	$x+1$	x^2+x+1
x^5	x^2+x+1	$x+1$
x^6	x^3+x^2+x	x^2+x
x^7	$x^4+x^3+x^2+1$	1

В итоге мы показали, что оба частных — по результатам деления в столбик и с помощью схемы регистра — совпадают, чего не скажешь про остатки от деления, которые требуются при кодировании и декодировании: сравните третий, четвертый и пятый столбцы табл. 3 с остатками табл. 4.

Остатки от деления должны храниться в ячейках памяти регистра. Схема на рис. 12 не дает нам остатки в чистом виде, так как обратная связь организована не только с выхода, но и с промежуточных ячеек (в данном случае есть отвод между первой и второй). Пошаговая процедура нахождения остатков (деление в столбик) (12) требует подачи обратной связи **только с выхода.**

Формула $res_n=res_{n-1}+res_{n-3}$ для регистра сдвига с обратными связями означает то, что последний бит (крайний правый) переходит (суммируется) в предпоследний и расположенный двумя битами левее предпоследнего.

В нашем варианте степень полинома x^3+x^2+1 равна трем, поэтому регистр содержит три ячейки, на входах которых могут формироваться трехбитовые остатки. Четвертый бит (на выходе последней ячейки) переходит в третий и первый, то есть, например, состояние регистра $001(0)$ — это соответствует делению x^2 на $g(x)$ — на следующем такте должно перейти в $101(1)$ — вот он остаток от деления x^3 на $g(x)$. Остаток от деления

x^4 будет равен сдвигу комбинации $101(1)$ на один бит вправо $010(1)$ с добавлением комбинации 101, которая получилась из-за наличия четвертой единицы $000(1) \rightarrow 101(1)$. В итоге получим результат $010+101=111$.

Преобразуем схему на рис. 12 так, чтобы сумматоры располагались непосредственно между ячейками памяти (рис. 13).

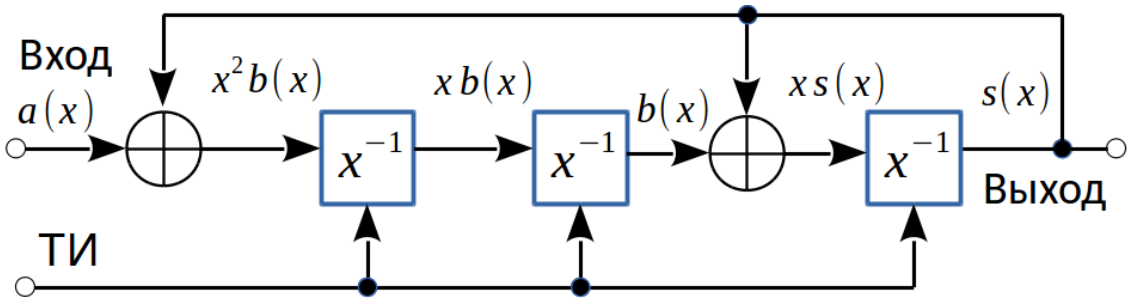


Рисунок 13 Трехразрядный регистр сдвига с обратными связями. Удобен для вычисления остатков от деления на полином.

Для сохранения коэффициента передачи порядок следования сумматоров следует снова зеркально отобразить **(проверьте это!).**

Составим таблицу работы рассматриваемой схемы (табл. 5). В начальный момент времени регистр, как всегда, обнулен.

Сравнивая табл. 5 и табл. 3, видим, что частные от деления совпадают, значит коэффициент передачи не изменился. Однако, анализируя остатки в табл. 4 и три столбца табл. 5, которые выделены серым и соответствуют входам ячеек памяти, замечаем, что они также совпадают. Старшие коэффициенты остатков в табл. 5 находятся справа, так как мы извлекаем из схемы правило деления в столбик и первыми подаем старшие коэффициенты.

Табл. 5. Потактовая работа регистра на рис. 13 при подаче на его вход дельта-последовательности

$a(x)$	$x^2 b(x)$	$x b(x)$	$x s(x)$	$b(x)$	$s(x)$
1	**1**	**0**	**0**	0	0
0	0	1	0	0	0
0	0	0	1	1	0
0	1	0	1	0	1
0	1	1	1	0	1
0	1	1	0	1	1
0	0	1	1	1	0

0	**1**	**0**	**0**	1	1
0	0	1	0	0	0
0	0	0	1	1	0
0	1	0	1	0	1

Таким образом, **делаем вывод: схема, пригодная для явного вычисления остатка от деления, это схема, содержащая сумматоры непосредственно между ячейками памяти, то есть когда сумматоры включаются в разрыв соединительной линии. Старшие коэффициенты входных и выходных полиномов располагаются правее (поступают на вход и выход первыми), младшие — левее (входят и выходят последними). Тот же самый порядок сохраняется и для порядка включения сумматоров: сумматор, ближайший к выходу, соответствует самой старшей степени генераторного полинома циклического кода.**

Подводя итог, составим схему (рис. 14) систематического кодера, основываясь на схеме рис. 13 и правиле кодирования (11), которое для выбранного кода (7, 4) будет записано в виде

$$s(x)=x^3\cdot a(x)+\left[x^3\cdot a(x)\right] \mod \left[x^3+x^2+1\right] . \qquad (13)$$

Степени слагаемых в полиноме $a(x)$ могут быть 0, 1, 2 и 3, так как число входных символов равно четырем. Тогда степени полинома $x^3a(x)$ будут 3, 4, 5 и 6, поэтому для кодирования необходимо знать остатки от деления степеней x^3, x^4, x^5 и x^6. Чтобы сэкономить три такта, входные биты подаются с конца схемы. Подача с конца, например, a_3, равносильна его подаче на первую ячейку и трем сдвигам вправо.

После четырех тактов, отведенных для ввода четырех информационных символов, в ячейках регистра будет храниться требуемый остаток от деления. Процессу его нахождения соответствует положение 1 переключателя.

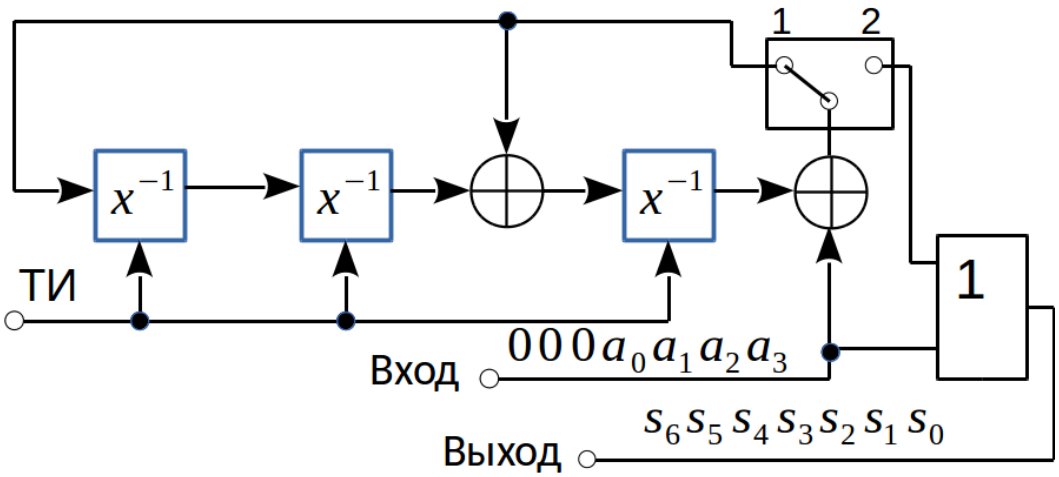


Рисунок 14 Схема систематического кодера для полинома $g(x)=x^3+x^2+1$

В процессе нахождения остатка информационные биты через схему **ИЛИ** (обозначена символом **1**) поступают на выход.

Для вывода найденного остатка переключатель переводится во второе положение. Вывод происходит за три такта, поэтому на весь процесс кодирования требуется семь тактов. Три нуля, поступающие после входных символов, предназначены для вставки проверочных символов (коэффициентов полинома-остатка).

3. Порядок выполнения работы

1. Определить с помощью какого из генераторных полиномов двоичного кода можно построить генератор последовательности максимальной длины (*M*-последовательности, псевдослучайной последовательности)

$$g_1(x)=x^4+x^2+1,\ \ g_2(x)=x^4+x+1,\ \ g_3(x)=x^4+x^2+x+1.$$

Привести в отчете доказательство, подобно рис. 11 и табл. 2. Выделить период.

2. Составить схему **несистематического** кодера двоичного циклического кода для найденного в п.1 полинома. Привести таблицу и рисунок, подобно рис. 9 и табл. 1, взяв $n=2^r-1$. Входное слово a взять случайным ненулевым.

3. Найти **все** остатки от деления целых степеней x^i, $i=0\ldots14$, на найденный в п. 1 генераторный полином. Задание сделать методом деления в столбик, приведя таблицу, подобную табл. 4, и непосредственно по схеме деления (подобно рис. 13 и табл. 5).

4. **Сравнить** степени

$$x^4, x^5, x^6, x^7, x^8, x^9, x^{10}, x^{11}, x^{12}, x^{13}, x^{14}$$

по модулю найденного полинома, то есть выразить каждую через базисные степени x^3, x^2, x^1 и x^0 . Например, для полинома x^4+x^2+x+1 справедливо сравнение $x^4 \equiv (x^2+x+1) \mod (x^4+x^2+x+1)$.

5. Используя схему из рис. 14, привести схему систематического кодера для найденного полинома. По тактам (в таблице) объяснить ход кодирования для выбранного входного слова, содержащего не менее двух единиц. Подтвердить правильность кодирования непосредственным расчетом по формуле (11).

6. Вычислить по схеме и методом деления в столбик остаток для найденной в п. 5 разрешенной кодовой комбинации систематического кода. Должен ли он быть равен нулю? Если да, то что это означает?

7. Ответить на следующие далее вопросы письменно.

4. Вопросы

1. Чему равен период последовательности, генерируемой троичным** трехразрядным регистром сдвига с обратными связями и коэффициентом передачи

$$K(x) = \frac{1}{x^3 - x + 1} \quad .$$

Равен ли он максимальному периоду? Для вычисления разложения в ряд Тейлора желательно использовать программу компьютерной алгебры (символьные вычисления). Привести отрезок (один период) найденной периодической последовательности.

**Используются ячейки памяти с тремя состояниями $(-1,1,+1)$ и сумматоры по модулю три.

2. Составьте таблицу сложения троичных символов $(-1,1,+1)$.

3. Приведите свой пример генераторного полинома третьей степени для троичного циклического кода, дающего максимальный период.

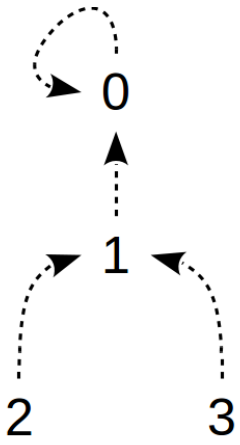
---

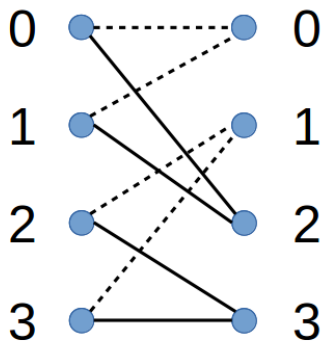
5. Литература

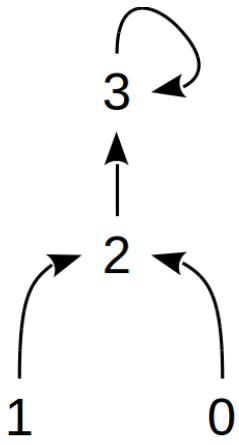
4. Триггер, http://ru.wikipedia.org/wiki/%D2%F0%E8%E3%E3%E5%F0.
5. 3BIT/CELL MLC NAND FLASH, http://www.samsung.com/global/business/semiconductor/minisite/SSD/uk/html/about/MlcNandFlash.html.
6. SANDISK SHIPS WORLD'S FIRST FLASH MEMORY CARDS WITH 64 Gigabit X4 (4-BITS-PER-CELL) NAND FLASH TECHNOLOGY, http://www.sandisk.com/about-sandisk/press-room/press-releases/2009/2009-10-13-sandisk-ships-world's-first-flash-memory-cards-with-64-gigabit-x4-(4-bits-per-cell)-nand-flash-technology/.

Для заметок

СВЁРТОЧНЫЕ КОДЫ







1. Введение

Свёрточные коды являются помехоустойчивыми, и успешно применяются для кодирования информации, передаваемой по каналам связи с помехами [1].

Любое помехоустойчивое кодирование *(канальное кодирование)* устроено так, что оно вносит *избыточность* в передаваемое сообщение. Простым примером её добавления является запись некоторой суммы денег прописью. В этом случае вероятность сделать ошибку при считывании прописи меньше, чем при считывании цифр, так как буквы в прописи связаны между собой смысловым контекстом, а цифры суммы денег — нет. Естественно, что качество записи всех анализируемых символов (цифр, букв) при этом должно быть одинаковым, иначе некорректно сравнивать эти способы по вероятности ошибки.

Избыточность, внесенная по известным правилам, и позволяет обнаруживать и исправлять некоторые ошибки. Это основная идея помехоустойчивого кодирования, без усвоения которой нет смысла изучать конкретные коды.

В данной работе ошибка вносится генератором случайных чисел с вероятностью, называемой *вероятностью битовой ошибки*.

Кодирование предназначено для снижения вероятности ошибки **после декодирования.** Декодирование должно давать выигрыш, который, как правило, тем больше, чем меньше вероятность ошибки на входе декодера. При достаточно большой вероятности ошибки (0,1–0,5) может сложиться ситуация, когда вероятность ошибки на входе и выходе декодера будет практически одинаковой. В этом случае эффективность кода будет близка к нулю (нет выигрыша), и лучше вовсе от него отказаться, так как за возможность кодирования мы платим или полосой частот, или скоростью передачи данных, или мощностью передатчика, или, наконец, дополнительной задержкой в приемнике (также возможны сочетания перечисленных ресурсов).

Процесс кодирования свёрточными кодами очень простой, чего не скажешь про процесс декодирования. В этом смысле, например, циклические коды являются симметричными.

Кодирование осуществляется с помощью *регистра сдвига* (последовательно соединенных ячеек памяти) и набора *сумматоров* (логических схем). Для упрощения материала дальше рассматриваются двоичные коды, в которых суммой является *сумма по модулю два*, а логической схемой, осуществляющей это суммирование, — схема исключающего **ИЛИ.**

Особенностью свёрточных кодов является **память,** которая приводит к тому, что в процессе декодирования текущего бита необходимо дождаться несколько последующих битов. В этом смысле декодер должен заглядывать в будущее до тех пор, пока не станет однозначным значение декодируемого бита. Такой интересный способ декодирования является следствием того, что свёрточные коды можно полностью описать диаграммой состояний, в которой показаны разрешенные переходы. Наличие запрещенных переходов позволяет бороться с некоторыми ошибками.

Абзацы, набранные в стиле

- Вычислить, сравнить, ...

являются упражнениями и предназначены для их выполнения и записи в отчёт по проделанной работе.

Структура отчета должна соответствовать следующему:

- ✔ Титульный лист;
- ✔ Цель работы;
- ✔ Ход работы
- ✔ Выводы;
- ✔ Список литературы;
- ✔ Приложения.

2. Сведения из теории

2.1 Кодирование

Так как свёрточный код является избыточным, то на один входной бит a приходится несколько (как минимум два) выходных битов, то есть математически кодер выполняет отображение

$$a \to b^{(1)} b^{(2)} b^{(3)} \ldots b^{(n)} \quad , \quad n>1 \quad .$$

Для краткости записи выходное слово $b^{(1)} b^{(2)} b^{(3)} \ldots b^{(n)}$ можно рассматривать как число, принимающее значение от 0 до $2^n - 1$.

Для формирования выходного слова выбирают **линейные** операции по отношению к входному биту. Так, например, можно просто скопировать входной бит на все позиции выходного слова. Однако, если учесть заданное количество предыдущих входных битов, то вариантов становится больше: на разные позиции можно ставить *значения* разных *линейных комбинаций* битов, хранящихся в регистре сдвига.

Произвольное кодовое слово удобно записывать в виде полинома, коэффициенты которого совпадают с элементами этого слова, а возрастающим степеням x^i соответствует увеличение задержки по времени[3]. Правило кодирования задается *генераторными (порождающими) полиномами,* каждый из которых отвечает за конкретный бит выходного слова.

Рассмотрим простой свёрточный код с отображением $a \to b^{(1)} b^{(2)}$, где одному входному биту соответствует два выходных; в этом случае *скорость кодирования* равна ½. Для реализации такого кода необходимо задать два полинома. Возьмем, например, полиномы степени не больше первой

$$G_1(x)=1 \quad , \quad G_2(x)=1+x \quad . \qquad (14)$$

Непрекращающуюся последовательность входных битов a_i запишем в виде полинома[4]

$$A(x)=\sum_{i \in \mathbb{Z}} a_i x^i \quad ,$$

3 Это следует из свойства z-преобразования, если x заменить на z^{-1}.

4 Математически выгодно индекс суммирования ничем не ограничивать

где символом $\mathbb{Z}$ обозначено множество целых чисел. Две последовательности на выходе кодера определим через произведение двух полиномов

$$B^{(1)}(x)=A(x)G_1(x)=A(x)\ ,$$
$$B^{(2)}(x)=A(x)G_2(x)=A(x)+x\,A(x)=\sum_{i\in\mathbb{Z}}(a_i+a_{i-1})x^i\ . \qquad (15)$$

На основании (15) определяем правило работы кодера

$$b_i^{(1)}=a_i\ ,\quad b_i^{(2)}=a_i+a_{i-1}\ .$$

Произведение двух полиномов — это полином, коэффициенты которого вычисляются через *свёртку* коэффициентов перемножаемых полиномов[5]. Для доказательства данного факта не требуется ничего, кроме знания правил перемножения двух полиномов и приведения подобных слагаемых. Также данные факты известны тем, кто знает элементы теории z-преобразования.

Если для рассматриваемого свёрточного кода попытаться построить *генераторную матрицу,* которая есть у любого *линейного блочного кода,* то окажется, что она будет неограниченной по размерам

$$\left(\ldots b_0^{(1)}b_0^{(2)}b_1^{(1)}b_1^{(2)}\ldots\right)=\left(\ldots a_0 a_1\ldots\right)\begin{pmatrix} & & & & \ldots & & & & \\ & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & \\ & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & \\ \ldots & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \ldots \\ & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & \\ & & & & & \ldots & & & & \end{pmatrix},$$

поэтому свёрточные коды не являются блочными. Правда, их можно назвать блочными по отношению к генераторным полиномам, так как полиномы имеют конечную степень (количество ячеек памяти ограничено, обратных связей нет).

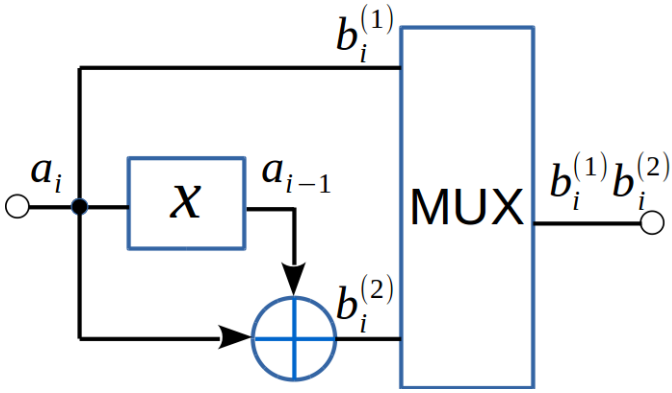


Рисунок 15: Свёрточный кодер ½

Составим функциональную схему кодера (рис. 15). Блок **MUX** — это мультиплексор, последо-

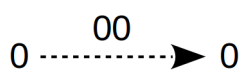
5 Отсюда и название данных кодов — свёрточные

вательно выдающий все входные биты; в данном случае выдающий попарно, начиная сверху вниз. Блок x — это триггер, хранящий предыдущий входной бит. Сумматор по модулю два обозначен окружностью со знаком +. Он имеет два входа и один выход. Таблица сложения простая

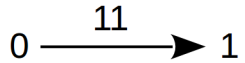
$$0+0=0,\quad 0+1=1,\quad 1+0=1,\quad 1+1=0\ .$$

Выходные биты кодера однозначно определяются входным битом и набором предыдущих входных битов, запомненных в регистре (в последовательно соединенных триггерных ячейках). Эти предыдущие биты называются *состоянием кодера.* Состояние иногда удобно записывать в виде десятичного числа, соответствующего двоичному коду. В рассматриваемом примере состояние определяется лишь одним битом a_{i-1} .

Считается, что в начальный момент кодер имеет нулевое состояние. Если на вход поступил бит **0,** то на выходе будет пара **00,** а в момент прихода следующего бита кодер перейдет в состояние **0,** так как до этого на входе был



0. Данную ситуацию можно отобразить веткой, в которой штриховая линия обозначает приход нулевого бита, а слово слева — состояние кодера в момент кодирования, то есть в момент выдачи пары битов **00** (эти биты обозначены над линией). Слово на конце стрелки — состояние кодера после прихода следующего кодируемого бита, то есть это будущее состояние.



Определим ветку, если пришел бит **1,** а кодер находится в состоянии **0.** Очевидно, что выходная пара — **11,** а будущее состояние — **1.** В данном случае будущее состояние определяется только текущим входным битом, так как ячейка памяти (триггер) одна; в общем случае будущее зависит и от некоторых предыдущих битов. Здесь линия стрелки сплошная, так как пришел бит **1.**

Осталось определить две ветки для единичного начального состояния:

а) пришел **0,** на выходе **01,** будущее состояние — **0;**

б) пришла **1,** на выходе **10,** будущее состояние — **1.**

Выписывая всевозможные состояния кодера в виде прямоугольных блоков с надписями внутри и соединяя их линиями со стрелками, можно получить *диаграмму состояний* (рис. 16).

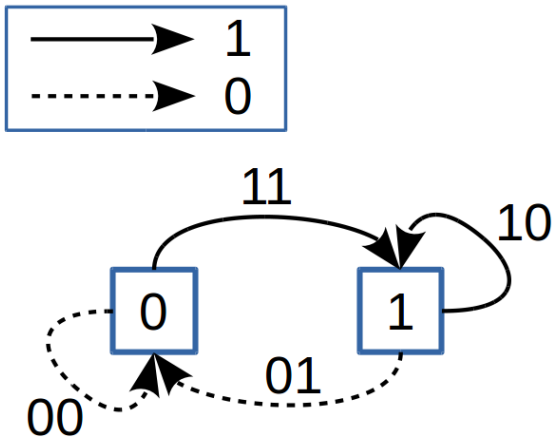


Рисунок 16: Диаграмма состояний свёрточного кодера ½

Если кодер переходит, например, из единицы в единицу, то получается *петля* или *цикл.* Пример цикла — подача на кодер одних единиц, что в установившемся режиме даст периодическую выходную последовательность ...101010....

С помощью диаграммы состояний очень просто кодировать входную битовую последовательность. Рассмотрим, например, дельта-последовательность *(цифровую дельта-функцию)*

$$\delta_n = 100\ldots0\ldots \ ,$$

отклик на которую есть *импульсная характеристика*

$$h_n = 11\,01\,00\ldots00\ldots \ ,$$

два ненулевых слова (**11** и **01**) которой говорят о том, что кодер обладает памятью. Индекс *n* у импульсной характеристики отвечает **за пару битов,** то есть за **слово.** Избыточность рассматриваемого кода равна 50%, так как на один входной бит кодер выдает два выходных. Кстати, можно отметить, что строки генераторной матрицы состоят из сдвинутых импульсных характеристик.

Рассмотрим более сложную входную последовательность

$$a = 110100\ldots0\ldots \ ,$$

отклик на которую по диаграмме состояний будет следующим

$$b = 11\,10\,01\,11\,01\,00\,00\ldots00\ldots \ .$$

Этот же результат можно получить несколько иным способом, если учесть, что отклик на сумму входных последовательностей равен сумме

откликов[6], и что любую входную последовательность можно выразить через сумму дельта последовательностей, отличающихся лишь задержкой

$$a=1101=1000+0100+0001=\delta_n+\delta_{n-1}+\delta_{n-3}\ ,$$

$$b=h_n+h_{n-1}+h_{n-3}=(11\,01\,00\,00\,00\ldots)+(00\,11\,01\,00\,00\ldots)+(00\,00\,00\,11\,01\ldots)=$$
$$=(11\,10\,01\,11\,01\ldots)\ .$$

Слагаемые в скобках складываются по модулю два.

- Составить схему свёрточного кодера с генераторными полиномами $G_1(x)=1$, $G_2(x)=1+x+x^2$, и диаграмму состояний.
- Изучив §2.2 Декодирование Витерби, на основании диаграммы состояний построить решетку кода до момента её установления.

6 Это следует из свойства умножения полинома на сумму полиномов:
$A(x)[B(x)+C(x)]=A(x)B(x)+B(x)C(x)$

2.2 Декодирование Витерби

Декодирование гораздо интереснее кодирования.

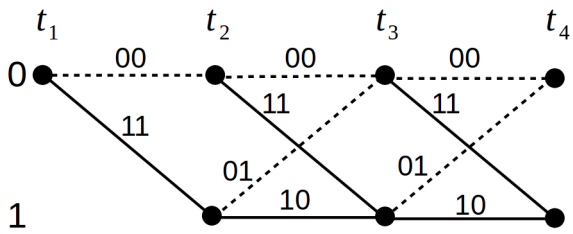


Рисунок 17: Решётка свёрточного кода ½

При декодировании удобно использовать решётку (рис. 17), которая получается после развертки во времени рассмотренной диаграммы состояний. Процесс построения решётки начинается с записи в столбик всех возможных состояний кодера. В нашем случае состояний два — это **0** и **1.** Кодер (и декодер) стартует с нулевого состояния, поэтому только от него отводятся линии так, как на это указывает диаграмма состояний. Получается второй столбик (момент времени t_2). Дальше отводятся четыре линии, так как кодер может перейти как из нуля в ноль или в единицу, так и из единицы в единицу или в ноль.

Естественно, что процесс углубления рано или поздно установится, так как число состояний конечно. Время установления зависит от количества ячеек памяти K , а количество разных состояний равно 2^{K-1} .

Решётка отображает эволюцию состояний свёрточного кодера, показывая все возможные переходы из одного состояния в другое. Моменты времени t_i — это тактовые моменты (система дискретного времени).

Декодируем с помощью решётки безошибочную последовательность

$$b = 11\,10\,01\,11\,01\,00\,00 \quad .$$

Так как ошибок заведомо нет, то декодирование осуществляется очень просто. Начинаем с нулевого состояния (момент времени t_1). На вход декодера пришла пара битов **11.** По какому пути пошел кодер, когда выдал эту пару? Естественно, по пути из **0** в **1,** выдавая **11.** Линия сплошная, значит декодируем **1.** Дальше пришла пара **10,** что соответствует переходу из **1** в **1** (моменты t_2 и t_3). Декодируем **1.** И так далее, что даст 110100.

Если нет уверенности в том, что входная последовательность безошибочная (а помехоустойчивые коды нет смысла использовать в каналах без

ошибок), то необходимо использовать полный метод декодирования, который заметно сложнее и называется *декодированием Витерби* [2].

Декодируем последовательность с ошибкой в первом бите третьего слова

$$b = 11\,10\,\mathbf{11}\,11\,01\,00\,00 \quad .$$

Начинаем, как всегда, с нулевого состояния. Строим решётку до тех пор, пока не будет по два пути, **входящих** в каждое состояние в текущем конце решетки (рис. 18). Так как в начальный момент решётка не установилась, то сначала потребуется некоторое время для того, чтобы определить эти пары входящих путей. Потом, когда будет решаться судьба второго бита, процесс углубления будет идти через один шаг, то есть появится бегущее окно фиксированного размера, называемое *окном декодирования*.

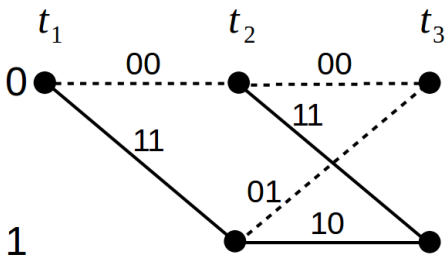


Рисунок 18: Первый шаг декодирования — нашли две пары путей

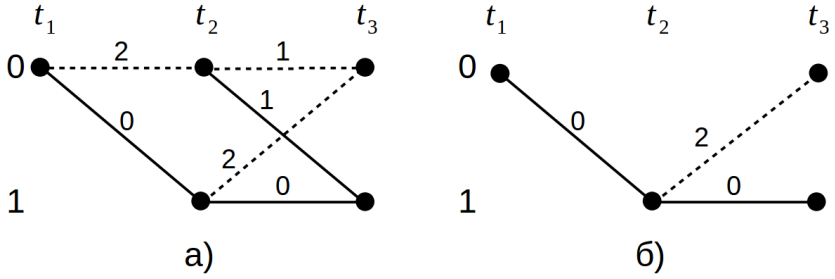


Рисунок 19: Первый шаг декодирования — исключили из каждой пары пути с наибольшей метрикой

Теперь, когда имеются по два пути, входящих в **0** и в **1,** необходимо оставить по одному, предварительно пометив линии этих путей числами, указывающими на расстояние между принятым словом и тем, что дает линия выбранного пути (рис. 19а). Верхний путь (из **0** в **0** и опять в **0**) дает выход кодера $00\,00$, который отличается от принятой пары слов $11\,10$. Эти два отличия дадут два расстояния Хемминга: 2 и 1, так как 00 от 11 отличается двумя битами, а 00 от 10 — одним. Сумма расстояний для верхнего пути оказалась равна трем. Проделываем аналогичные вычисления для всех оставшихся путей. Другой переход из **0** в **0** — это переход вниз-вверх по решётке (то есть из **0** в **1,** из **1** в **0**). Метрика этого пути равна двум, что мень-

ше метрики верхнего пути. Значит верхний путь вычеркиваем, V-образный — оставляем. Для путей, входящих в **1** метрики путей равны нулю и трём (рис. 19а). Оставляем L-образный путь, **объединяя** его с V-образным (рис. 19б).

В результате, для определения «судьбы» первого декодируемого бита остается решётка, изображенная на рис. 19б, в которой только одна линия между моментами времени t_1 и t_2, что дает однозначное декодирование — **1.**

Второй шаг декодирования, призванный определить «судьбу» второго бита, начинается с эволюции усеченной решётки так, чтобы была возможность исключить по одному пути из каждой пары **входящих** путей и в конце-концов оставить одну линию между моментами t_2 и t_3. Основная цель, по сути, в этом-то и состоит: оставить одну **наиболее вероятную** линию, по которой декодирование происходит автоматически. Момент времени t_1 можно выбросить, так как он никак не повлияет на дальнейшие результаты. Это говорит о марковости свёрточных кодов; этим экономится память декодера.

Казалось бы, а почему сразу не декодировать второй бит, выбрав нижнюю ветку на рис. 19б, ведь у неё метрика нулевая, в отличии от диагональной с метрикой два? Если мы так сделаем, то не будет учтена память кодера, то есть его возможности не будут до конца использованы. За счет памяти следующее (третье) слово несет информацию о втором бите, и это надо использовать, иначе зачем тогда городить ячейки памяти? В итоге, решётка строится вглубь до тех пор, пока не будет по два пути в каждое состояние (рис. 20а). Далее мы исключаем из каждой пары путь с

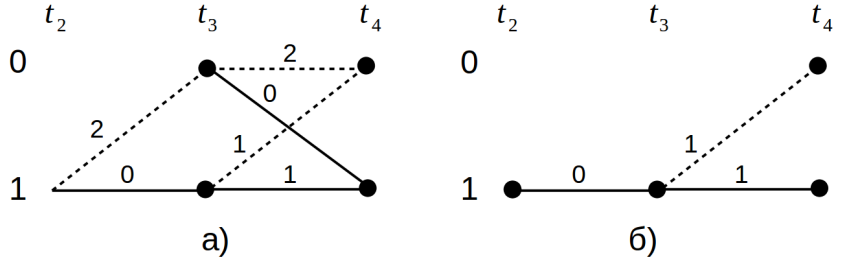


Рисунок 20: Второй шаг декодирования — определили второй декодируемый бит

наибольшей метрикой, то есть маловероятный путь (рис. 20б), и однозначно декодируем второй бит как **1**.

Между моментами t_3 и t_4 нет пути с нулевой метрикой (рис. 20б). Это напрямую связано с ошибкой в третьем принятом слове (слово **01** заменено на **11**). Разрешить возникшую неопределенность **может** следующее принятое слово, которое, к счастью, без ошибки, но декодер этого не знает. Здесь выделено слово **может,** так как за один такт неопределенность может и не разрешиться, и тогда, убрав из каждой пары путей пути с наибольшей метрикой, придется сделать ещё один такт, вычислив метрики и удалив лишние пути.

Всё это продолжается до тех пор, пока неопределенность не исчезнет, или пока не кончатся принимаемые слова. В последнем варианте придется вычислить метрики всех путей и выбрать путь с минимальной, разрешив в конечном итоге ситуацию буриданова осла. Если окажется, что метрики выбираемых путей равны между собой, то выбирается любой *(рандомный путь)*.

На третьем шаге декодирования принимаем **четвертое** слово **11.** За счет памяти кодера оно несет информацию о третьем бите. Рассмотрим путь из **1** в **1** (рис. 21а).

Зигзагообразный путь дает на выходе 0111, что приводит к суммарной метрике $1+0=1$, а горизонтальный путь — выход 1010 или метрику $1+1=2$. Видно, что первый путь предпочтительнее. Путь из **1** в **0** предпочтительнее пройти как $1\to1\to0$, так как метрика будет равна двум, что меньше метрики 3, даваемой переходом $1\to0\to0$. Таким образом, неопределенность относительно третьего бита не разрешилась. Шагаем дальше, достраивая решетку в момент времени t_6 (рис. 21б), что требует **пятого** принятого слова **01.**

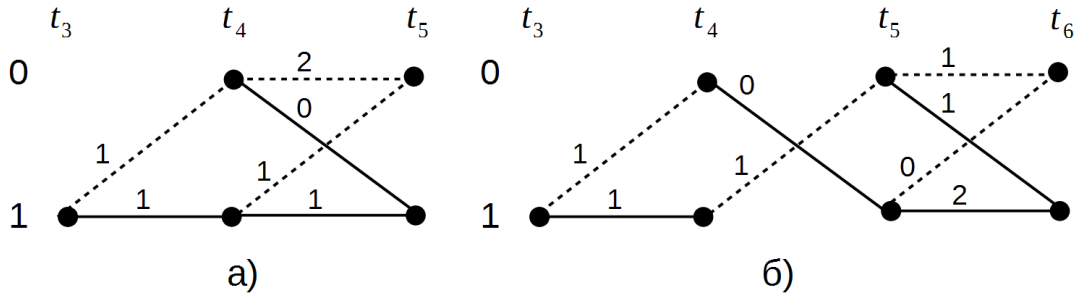


Рисунок 21: Третий шаг декодирования — третий бит не определился

Анализ рис. 21б дает интересный результат: путь в **0** предпочтительнее выбрать *N*-образным (метрика $1+0+0=1$), поэтому между моментами t_3 и t_4 пока остается штриховая линия, которая соответствует декодируемому биту **0**, но ещё есть путь в **1** — он также даст свой вес, который пропускать нельзя. Метрики двух путей в **1** оказались одинаковыми, так как две суммы равны между собой

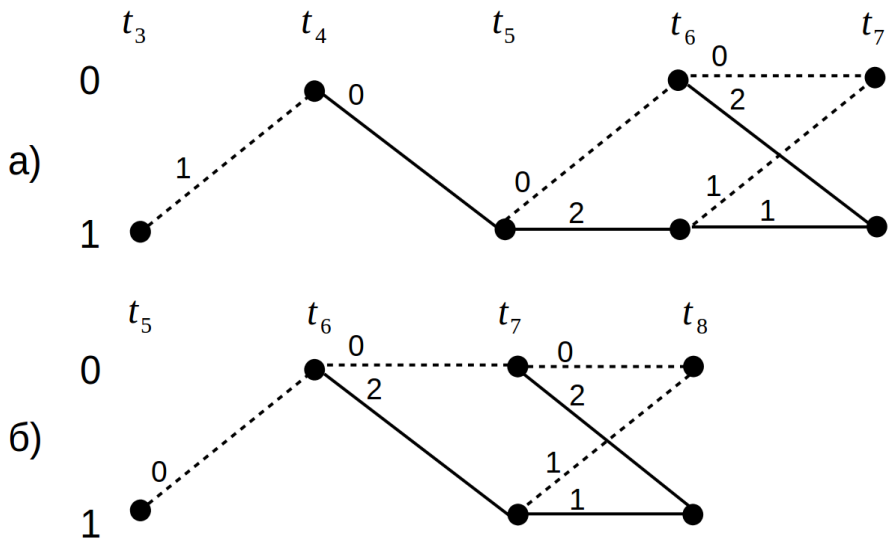


Рисунок 22: Третий шаг декодирования — определились третий и четвертый биты

$$1+0+2= \\ =1+1+1=3\ ,$$

поэтому выбираем любой путь.

Выгоднее выбрать тот, который исключает все ещё имеющуюся неопределенность относительно третьего бита — это путь $1\to0\to1\to1$ с метриками $1+0+2$. В результате разрешилась неопределенность сразу относительно двух битов: третьего **0** и четвертого **1** (рис. 22а). Решетку от t_6 до t_7 достраиваем по предпоследнему принятому слову **00.** Исключаем лишние пути и достраиваем решётку по последнему слову **00** (рис. 22б). Пятый

бит — это **0.** После дальнейшего исключения лишних путей декодируется шестой бит — опять **0.**

Таким образом, декодированная последовательность совпала с исходной, несмотря на обнаруживаемую ошибку в одном принятом слове: она была исправлена.

В рассмотренном кодере память равна одному биту, поэтому при кодировании, например, шести битов, на выходе декодера необходимо сохранить не шесть слов, а семь, так как седьмое слово потребуется при декодировании шестого бита.

2.3 Пороговое декодирование

Пороговое декодирование гораздо проще декодирования Витерби. Чем же придется заплатить за эту простоту?.. Вероятностью ошибки? Если да, то всегда ли?..

Интересно было бы узнать в каких случаях пороговый декодер даст вероятность ошибки не хуже декодера Витерби. Если такие случаи найдутся, то это будет говорить о том, что не всегда имеет смысл использовать сравнительно медленный декодер Витерби: всё должно быть промоделировано и обосновано.

Рассмотрим пороговое декодирование на примере изученного свёрточного кодера (рис. 15).

На выходе кодера имеется попарное чередование битов. Первый бит совпадает с *информационным битом*, второй *(проверочный бит)* равен сумме информационных: текущего и предыдущего. Раз кодер умеет вычислять проверочные биты, то и декодер может это сделать, беря в качестве входного бита принятый информационный.

Обозначим информационный бит как a_i , а соответствующий ему проверочный как b_i . Поместим эти биты в таблицу (табл. 1), показывающую зависимости между битами (то есть какой бит от какого зависит). Символом $b_i^{(d)}$ обозначим проверочный бит, **вычисляемый декодером.**

a	a_1	a_2	$\boldsymbol{a_3}$	a_4	a_5	Кодер
b	b_1	b_2	$\boldsymbol{b_3}$	$\boldsymbol{b_4}$	b_5	
$b^{(d)}$	$b_1^{(d)}$	$b_2^{(d)}$	$\boldsymbol{b_3^{(d)}}$	$\boldsymbol{b_4^{(d)}}$	$b_5^{(d)}$	Декодер

Таблица 1: К пороговому декодированию свёрточного кода

Жирным выделен **один** информационный бит и зависящие от него проверочные биты. Например, кодер вычислял $b_4 = a_4 + a_3$, $b_3 = a_3 + a_2$. То же самое делает и декодер, чтобы в конце-концов решить судьбу рассматриваемого информационного бита a_3 . Если ошибок нет, то биты b_3 и $b_3^{(d)}$, b_4 и $b_4^{(d)}$ должны попарно совпадать. Сравнение делает декодер, и в зависимости от количества несовпадений либо исправляет информационный бит, либо — нет. Если декодер исправил **бит,** то проверочные символы, на которые **он** влияет, пересчитываются заново.

Как определить порог по количеству несовпадений? Порог определяется по минимуму вероятности ошибки на выходе декодера для малой вероятности ошибки в канальных битах; это не так сложно сделать экспериментально. В рассматриваемом случае количество несовпадений может быть равно 0, 1 и 2, так как декодером сравниваются две пары битов.

Для анализа порогового декодирования в табл. 1 был специально выбран третий бит, чтобы оставить место слева. Так как кодер имеет память на один бит, то неверно декодированный бит a_2 повлияет на процесс декодирования бита a_3 . Этот факт является очень важным, так как вскрывает неблочную структуру свёрточных кодов. В блочных кодах ошибка декодирования кодового слова не влияет на ошибку декодирования следующего слова.

Определим логику порогового декодирования для двух вариантов:

A) Предыдущий бит декодирован верно (простой вариант);

B) Предыдущий бит декодирован ошибочно (вариант посложнее).

Такое разделение позволит упростить вывод формулы для вероятности ошибки на выходе декодера.

Результаты декодирования предпредыдущих битов никак не влияют на декодирование текущего бита, поэтому выведенная далее формула для вероятности ошибки на выходе декодера будет точной для рассматриваемого кода. Считается также, что каждый бит (что **0**, что **1**) может быть искажён с постоянной вероятностью, не зависящей от результатов определения предыдущих битов *(двоичный симметричный канал с независимыми ошибками)*.

В процессе декодирования бита a_3 участвуют биты a_3, a_4, b_3 и b_4, поступающие на вход декодера, и биты $b_3^{(d)}$ и $b_4^{(d)}$, вычисляемые декодером по принятым битам a_3 и a_4. Таким образом, в принимаемой четверке битов может быть искажен один бит, два, три, четыре или ни одного. Если искажен один бит, то следует рассмотреть два случая: 1) это информационный и 2) это проверочный. Если два бита, то три случая: 1) информационный плюс проверочный, 2) два проверочных, 3) два информационных. Если три бита, то два случая. При четырех искаженных битах и при отсутствии ошибок (ноль искаженных битов) — всего по одному случаю. Подробности идут ниже.

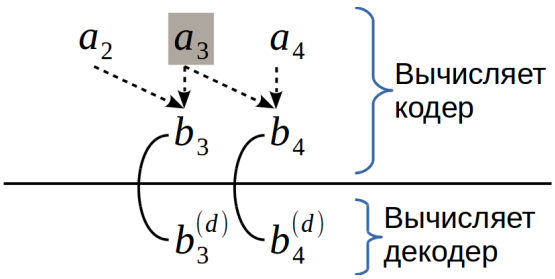


Рисунок 23: Пороговое декодирование — предыдущий бит декодирован верно

Рассматриваем вариант А), показанный на рис. 23. Декодируется выделенный серым фоном бит.

Если в принятой четверке ошибок нет, то декодер вычислит верные проверочные биты, так как предыдущий бит был декодирован верно. Количество несовпадений равно нулю, поэтому декодер не будет исправлять бит a_3, что даст верное декодирование. Вероятность такого события — это вероятность ноль-кратной ошибки, вычисляемая по биномиальной формуле

$$P(q)=\binom{n}{q}p^q(1-p)^{n-q}\ , \qquad (16)$$

где $\binom{n}{q}=\frac{n!}{q!(n-q)!}$ — число сочетаний из n по q ;

p — битовая вероятность ошибки на входе декодера (определяется качеством линии связи, видом модуляции).

Параметр q в (16) как раз эту кратность ошибки и определяет. Так как рассматривается четверка бит, то $n=4$.

Если в принятой четверке одна ошибка $(q=1)$, то следует рассмотреть два случая:

- Ошибка в каком-либо информационном бите;
- Ошибка в каком-либо проверочном.

Если ошибка в бите a_3 , то декодер неверно вычислит оба проверочных бита $b_3^{(d)}$ и $b_4^{(d)}$, что даст наибольшее количество несовпадений. Декодер исправит бит a_3 , и это будет верным декодированием. Если ошибка в бите a_4 , то декодер неверно вычислит только бит $b_4^{(d)}$, что даст одно несовпадение. Порог несовпадений выбирается так, чтобы минимизировать вероятность ошибки. Мы, естественно, хотим, чтобы с однократной ошибкой код справлялся, тогда мы просто обязаны при одном несовпадении оставлять анализируемый бит в исходном состоянии. Тогда бит a_3 не изменится, что даст верное декодирование, так как мы задали ошибку в бите a_4 (но до него очередь декодирования ещё не дошла).

Если ошибка в любом проверочном бите, то будет одно несовпадение, и, согласно принятому порогу, декодер ничего исправлять не будет, что опять даст верное декодирование.

Таким образом, видно, что если предыдущее декодирование было сделано верно, то с текущей однократной ошибкой декодер всегда справится. Это даёт право добавить к вероятности правильного декодирования слагаемое $P(1)$, взятое из (16).

Если в принятой четверке битов две ошибки $(q=2)$, то необходимо рассмотреть три случая. Их удобно условно обозначить как **ИИ** (ошибка в

двух информационных), **ПП** (в двух проверочных), **ИП** (в информационном и проверочном). Детализация всех возможных сочетаний **И** и **П** дана ниже.

- **ИИ.** Один вариант:
  - Ошибка в a_3 и a_4. Бит $b_3^{(d)}$ не совпадет с b_3, а бит $b_4^{(d)}$ совпадет с b_4, так как $b_4^{(d)}=a_4+a_3=a_4+1+a_3+1$. Одно несовпадение, декодер не тронет бит a_3 и ошибется, так как этот бит надо бы исправить.
- **ПП.** Один вариант:
  - Ошибка в b_3 и b_4. Два несовпадения — декодер исправит бит a_3 и опять ошибется.
- **ИП.** Четыре варианта:
  - Ошибка в a_3 и b_3. Одно несовпадение. Декодер ошибся;
  - Ошибка в a_3 и b_4. Одно несовпадение. Ошибка декодера;
  - Ошибка в a_4 и b_3. Два несовпадения. Ошибка декодера;
  - Ошибка в a_4 и b_4. Ноль несовпадений. Декодер справился.

Таким образом, видно, что если предыдущее декодирование было сделано верно, то с текущей двукратной ошибкой декодер почти всегда не справится. Это дает право добавить к вероятности правильного декодирования лишь небольшой вклад в виде слагаемого

$$\frac{1}{6}P(2) \quad .$$

Если в принятой четверке битов три ошибки $(q=3)$, то необходимо рассмотреть два случая типа **ИИП** и **ИПП.**

- **ИИП.**
  - $a_3 a_4 b_3$. Ноль несовпадений. Ошибка будет;
  - $a_3 a_4 b_4$. Два несовпадения. Верное декодирование.
- **ИПП.**
  - $a_3 b_3 b_4$. Ноль несовпадений. Ошибка;

- $a_4 b_3 b_4$. Одно несовпадение. Верное декодирование.

Добавляем к вероятности правильного декодирования слагаемое

$\frac{1}{2}P(3)$ (половина трехкратных ошибок исправится).

И, наконец, четырехкратная ошибка. Вроде как декодер вообще не должен с ней справиться. Она всего одна: **ИИПП,** то есть $a_3 a_4 b_3 b_4$. Одно несовпадение, поэтому декодер, не затронув бит a_3 , допустит ошибку. Предположение о крахе декодера при четырехкратной ошибке подтвердилось.

Рассмотрев пункт А), можно утверждать, что вероятность правильного декодирования равна

$$P_{\text{п/п}} = P(0) + P(1) + \frac{1}{6}P(2) + \frac{1}{2}P(3) \quad , \qquad (17)$$

где индекс п/п означает правильное декодирование/правильное декодирование, причем буква справа от знака / определяет условие декодирования предыдущего бита.

Рассмотрим, наконец, пункт В). Можно использовать рис. 23, считая, что в бите a_2 всегда сделана ошибка. Все сочетания опишем кратко.

- $q=0$
  - «_».
    - Одно несовпадение. Верное декодирование.
- $q=1$
  - **И**
    - a_3 . Одно несовпадение. Ошибка;
    - a_4 . Два несовпадения. Ошибка.
  - **П**
    - b_3 . Ноль несовпадений. Верное декодирование;
    - b_4 . Два несовпадения. Ошибка.
- $q=2$
  - **ИИ**

- $a_3 a_4$. Ноль несовпадений. Ошибка.

- **ПП**
  - $b_3 b_4$. Одно несовпадение. Верное декодирование.
- **ИП**
  - $a_3 b_3$. Два несовпадения. Верное декодирование;
  - $a_3 b_4$. Ноль несовпадений. Ошибка;
  - $a_4 b_3$. Одно несовпадение. Верное декодирование;
  - $a_4 b_4$. Одно несовпадение. Верное декодирование.

- $q=3$
  - **ИИП**
    - $a_3 a_4 b_3$. Одно несовпадение. Ошибка;
    - $a_3 a_4 b_4$. Одно несовпадение. Ошибка.
  - **ИПП**
    - $a_3 b_3 b_4$. Одно несовпадение. Ошибка;
    - $a_4 b_3 b_4$. Ноль несовпадений. Верное декодирование.

- $q=4$
  - **ИИПП**
    - $a_3 a_4 b_3 b_4$. Два несовпадения. Как ни странно, но верное декодирование (неверное декодирование предыдущего бита a_2 при четырехкратной ошибке помогло).

Рассмотрев пункт В), можно утверждать, что вероятность правильного декодирования равна

$$P_{\text{п/о}} = P(0) + \frac{1}{4}P(1) + \frac{4}{6}P(2) + \frac{1}{4}P(3) + P(4) \ , \qquad (18)$$

где буква о в индексе обозначает ошибку декодирования.

Осталось собрать формулы (17) и (18) в одну.

Процесс декодирования можно представить себе так, как это изображено на рис. 24, где буквой **П** обозначено событие Правильное декодирование, а буквой **О** — Ошибочное декодирование. Этот процесс — это постоянные переходы из **О** в **П** или **О,** или из **П** в **П** или **О,** то есть, например, после правильного декодирования **П** мы можем либо ошибиться **О,** либо опять правильно декодировать **П,** и так далее, и тому подобное.

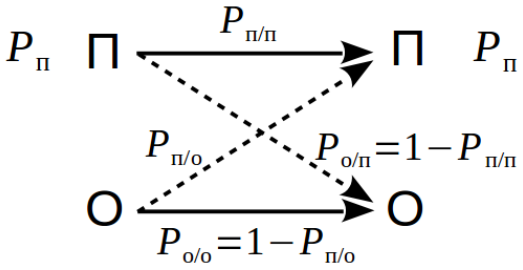


Рисунок 24: К определению вероятности правильного декодирования

Так как вероятность определяется по бесконечному количеству шагов декодирования, то слева и справа на рис. 24 стоит одинаковая величина — искомая вероятность правильного декодирования $P_{п}$, поэтому получим

$$P_{п}=\frac{P_{п/о}}{P_{п/о}-P_{п/п}+1} \quad . \tag{19}$$

Итак, подставляем (17) и (18) в (19)

$$P_{п}(p)=\frac{4p^4-8p^3+7p^2-3p+1}{8p^4-16p^3+12p^2-3p+1} \quad . \tag{20}$$

Графически удобно анализировать вероятность ошибки

$$P_{о}(p)=1-P_{п}(p)=\frac{4p^4-8p^3+5p^2}{8p^4-16p^3+12p^2-3p+1} \quad ,$$

построив её зависимость в логарифмическом масштабе (рис. 25).

Если вероятность битовой ошибки до декодирования (BER) была 0,001, то после декодирования она падает в 200 раз, что говорит об определённой эффективности применения свёрточного кодирования. Коэффициент уменьшения вероятности ошибки (выигрыш декодера), к сожалению, не остаётся постоянным: например, для BER = 0,01 он будет равен около 20, что в десять раз меньше. При BER > 0,2 выигрыш практически отсутствует.

На рис. 25 можно отметить особенные точки $(0{,}5;0{,}5)$ и $(1{,}0;0{,}5)$, причем последней точке соответствует ситуация, когда канал постоянно вносит ошибку, и, если обратиться к рис. 24, то результат декодирования можно

выразить змейкой вида **ОПОПОП...,** так как $P_{п/п}=0$ и $P_{п/о}=1$. Это и означает, что битовая вероятность ошибки после декодирования в данном случае равна ровно 0,5.

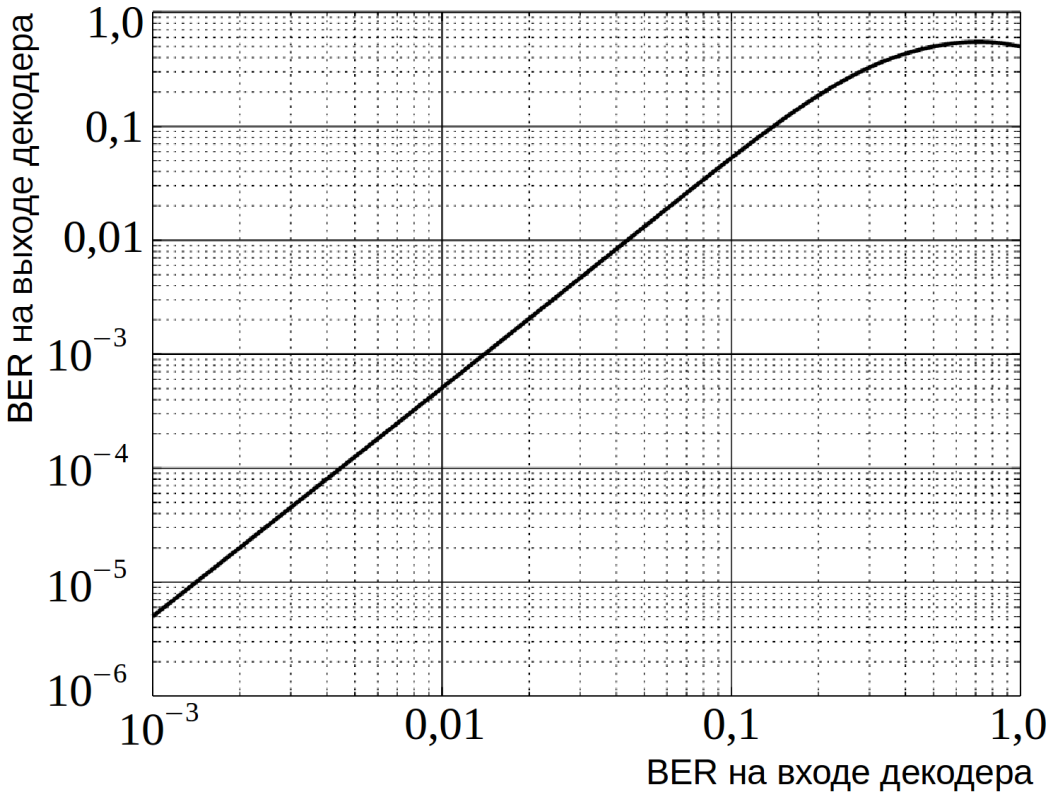


Рисунок 25: Вероятность битовой ошибки на выходе свёрточного декодера в зависимости от вероятности битовой ошибки на его входе

Даже такой простой свёрточный код, как рассмотренный, может сделать из одной ошибки на тысячу принятых бит одну ошибку на двести тысяч принятых. За такую эффективность мы платим избыточностью в 50%, и чтобы сохранить битовую скорость передачи информации, следует в два раза увеличить полосу пропускания канала связи, или пожертвовать четырехкратным увеличением мощности, если полосу нет возможности расширить[7].

Остался вопрос: как регулировать избыточность в сторону менее 50%? Для этого применяется *перфорирование* (**puncturing,** удаление некоторых проверочных битов после нескольких тактов кодирования), что дает возможность на основании кода ½ построить код со скоростью кодирования, например, 2/3, у которого избыточность равна 1/3. Рассмотрение таких кодов вы-

7 Про решётчато-кодированную модуляцию (trellis coded modulation) здесь речь не идет.

ходит за рамки данной работы. Вообще, надо сказать, что коды со скоростью кодирования ½ являются базовыми и в итоге, после вставки перфорированных битов, декодируются именно они. Это делается из-за желания унифицировать декодеры.

3. Описание лабораторного макета

Программный макет — консольная программа (рис. 26), предназначенная для анализа свёрточного кодирования и рассмотренных выше методов декодирования. Написана на языке C++ с использованием открытых библиотек **Qt** [3].

После запуска программы **thr_svert** в консоль выводится заголовок и перечисляются коэффициенты полиномов свёрточного кода, которые по умолчанию совпадают с рассматриваемым кодом (14). Если требуется другой набор полиномов, то это можно сделать, изменив файл **polynoms.txt** по инструкции, в нем содержащейся.

Далее предлагается последовательно ввести количество бит на входе кодера и вероятность битовой ошибки (на входе декодера). Если выбран режим **Выводить битовые последовательности на экран,** то в консоль выведутся все символы-биты во всех контрольных точках: вход и выход кодера, вход и выход декодера. Данный режим имеет смысл включать при достаточно коротких последовательностях. Наконец, предлагается ввести порог для порогового декодера, который приближенно можно вычислить как половину от числа сравниваемых пар.

```
Свёрточный код (систематический)
Пороговое декодирование и декодер Витерби
_____________________________________
Коэффициенты полиномов G(x):
G1 = (10)
G2 = (11)
Коэффициенты полиномов установлены из файла:
   polynoms.txt
Введите количество бит на входе кодера:
10
Введите битовую вероятность ошибки (BER):
0.1
Выводить битовые последовательности на экран? (да, нет)
у
Введите порог для порогового декодера:
1
 Кодирование...

---ПОРОГОВЫЙ ДЕКОДЕР---
 Декодирование...
Битовые последовательности:
 на входе кодера:    0011000010
 на выходе кодера:   00001110010000000110100
 на входе декодера:  11010110010000000010100
 на выходе декодера: 1011000010
Количество искажённых в канале бит: 5 из 22
Количество неисправленных декодером бит: 1 из 10
BER на входе декодера: 0.227273
BER на выходе декодера: 0.1

---ДЕКОДЕР ВИТЕРБИ---
 Декодирование...
Битовые последовательности:
 на выходе декодера: 1011000010
Количество неисправленных декодером бит: 1 из 10
BER на выходе декодера: 0.1
```

Рисунок 26: Консольная программа свёрточного кодирования и декодирования систематического кода

Входные биты задаются датчиком случайных чисел (равновероятно для **0** и **1**). Ошибки в выходную последовательность кодера вносятся датчиком случайных чисел с заданной пользователем вероятностью. При малом количестве выходных битов (менее 20–50) требования заданной вероятности выполняются грубо, поэтому в любом случае в программе выводятся текущие оценки этих вероятностей (до декодера и после).

Пороговое декодирование выполняется в полном соответствии с рассмотренным на рис. 23 алгоритмом. Если порог превышен, то анализируемый бит инвертируется.

В самом конце вывода программы помещается результат декодирования по Витерби. Следует учесть, что от раза к разу результаты декодирования по Витерби могут отличаться друг от друга. Это связано с тем, что процедура выбора наиболее короткого пути может быть неоднозначна. Для устранения неоднозначности применяется датчик равновероятных случайных чисел.

4. Порядок выполнения работы

1. Закодировать свёрточным кодом (рис. 15) последовательность битов для своего варианта из приложения Б.
2. Запустить программу **thr_svert** и выбрать следующие параметры

- Количество бит на входе кодера — 10;
- Вероятность битовой ошибки (BER) — 0,1;
- Выводить битовые последовательности на экран — **да, yes.**

Добиться случая двух-трех канальных ошибок, введенных так, что **пороговый** декодер их полностью исправляет. Выписать все битовые последовательности в тетрадь (отчет), **в том числе и для декодера Витерби** (как справится этот декодер — без разницы). Привести процедуру кодирования (методом импульсных характеристик или по диаграмме состояний). Расписать по шагам процедуру (порогового) декодирования по примеру Приложения А.

3. Выполнить предыдущий пункт, выбрав ошибки так, что **пороговый** декодер не справится (останется **хотя бы один** ошибочный бит).
4. Декодировать по алгоритму Витерби последовательности из пунктов 2 и 3. Привести необходимые решётки декодирования. Сравнить ручные результаты с программными.
5. Сравнить битовые вероятности ошибки для обоих декодеров, выбрав проверочные полиномы вида

- $G_2(x)=1+x$ (порог* — 1),
- $G_2(x)=1+x+x^2+x^3$ (порог* — 3),
- $G_2(x)=1+x+x^3$ (порог* — 2).

*Порог выбран по минимуму вероятности ошибки после декодирования для малых BER на входе декодера.

Вероятности битовых ошибок на входе задавать из ряда 0,005; 0,01; 0,02; 0,05; 0,1; 0,3; 0,5. Количество бит на входе — миллион. Параллельно фиксировать время кодирования и декодирования, а также среднюю длину решётки (например, на рис. 21б длина решётки равна трем). На графиках привести теоретический график вероятности битовой ошибки на выходе декодера кода Хемминга (7, 4)[8]

$$p_{\text{Нb}}=9p^2-26p^3+30p^4-12p^5 \ ,$$

у которого избыточность 3/7, что немного меньше 50%.

6. Рассмотреть избыточное кодирование простым копированием бит $a_1a_1a_2a_2a_3a_3a_4a_4\ldots$, где избыточность также равна 50%. Привести полиномы данного кода. Оценить исправляющую способность данного кода по битовой вероятности ошибки на выходе с помощью программы **thr_svert,** откорректировав файл с коэффициентами полиномов. Вероятности BER на входе декодера задавать также, как и раньше.

5. Контрольные вопросы

1. Является ли свёрточный код блочным? Почему?
2. Что из себя представляет генераторная матрица свёрточного кода?
3. Выгодно ли[9] по вероятности ошибки использовать простое копирование входного бита на две позиции вместо линейных комбинаций с задержанным, как это делается в рассмотренном свёрточном коде?
4. Какой декодер лучше по вероятности ошибки: пороговый или Витерби? В каких случаях? Какой декодер быстрее по скорости декодирования?

8 Это помехоустойчивый код без памяти

9 В двоичном симметричном канале с независимыми ошибками

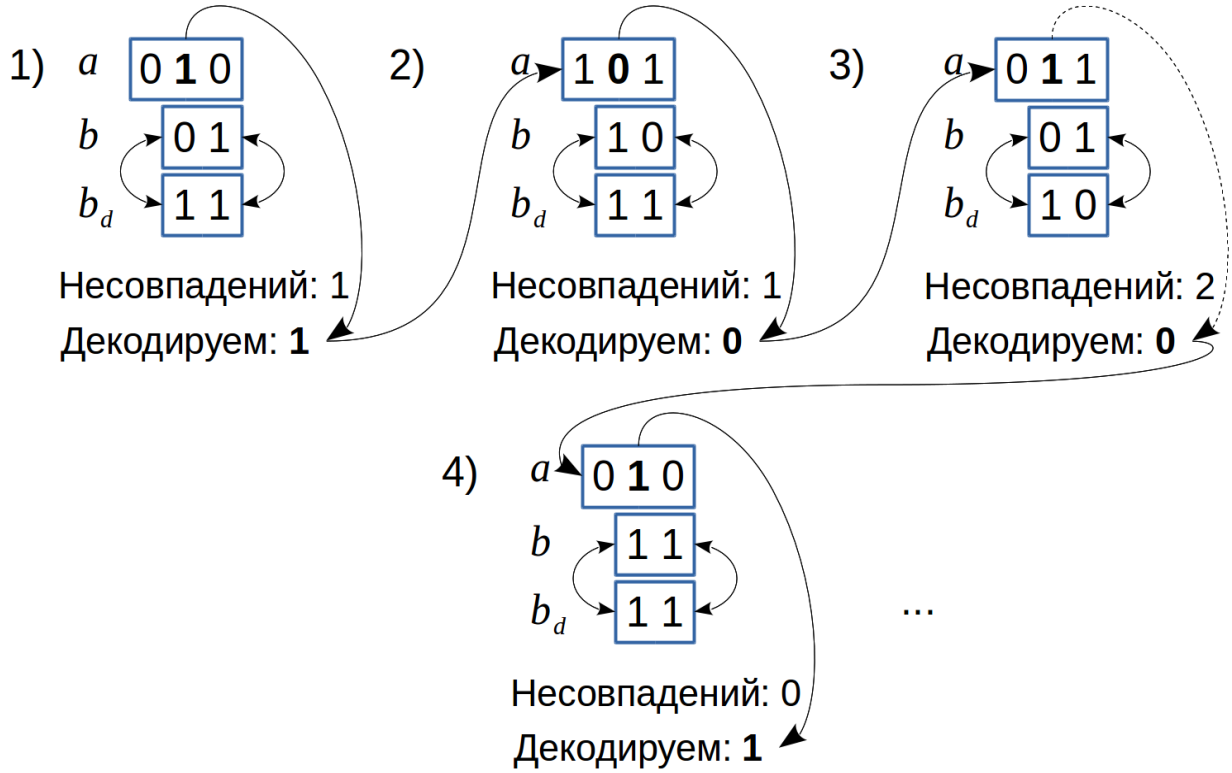
5. Какие коды лучше по вероятности ошибки: Хемминга или свёрточные? В каких случаях?

6. Литература

1. И. Шахнович. DVB-T2 — новый стандарт цифрового телевизионного вещания// Электроника: наука, технология, бизнес, 6/2009, http://www.electronics.ru/files/article_pdf/0/article_258_849.pdf.
2. Скляр Бернард. Цифровая связь. Теоретические основы и практическое применение. — 2003.
3. Qt | Cross-platform application & UI development framework, https://www.qt.io/.
4. State and Trellis, Error Correcting Codes by Dr. P. Vijay Kumar, Department of Electrical Communication Engineering, https://www.youtube.com/watch?v=6j9dcKhsYYU.
5. The Viterbi Decoder, Error Correcting Codes by Dr. P. Vijay Kumar, Department of Electrical Communication Engineering, https://www.youtube.com/watch?v=JwWBfhfHq_M.

Приложение А Пример нескольких шагов порогового декодирования для свёрточного кода ½

Вход кодера: 1001001001
Выход кодера: 11 01 00 11 01 00 11 01 00 11 01
Вход декодера: 1**0** 01 **1**0 11 01 00 11 **1**1 00 11 01



Канальная ошибка произошла в первом, третьем и восьмом слове, причем в первом слове искажен проверочный бит, а в третьем и восьмом — информационные биты. Так как однобитные ошибки повредили слова, отстоящие друг от друга как минимум на одно слово, то все они будут исправлены (память кодера равна единице).

Приложение Б Последовательности для свёрточного кодирования

Номер варианта	Последовательность битов									
1	1	0	0	1	0	1	0	1	1	1
2	1	1	1	0	0	0	0	0	1	0
3	1	0	1	1	1	0	1	1	0	0
4	1	1	1	0	1	1	0	0	1	0
5	0	1	0	1	1	1	1	1	0	1
6	0	1	1	0	0	0	0	0	0	1
7	1	0	0	1	0	1	1	0	0	0
8	0	0	0	1	1	1	1	1	1	1
9	0	1	1	1	0	1	1	1	1	1
10	1	1	1	1	0	0	0	0	0	1
11	1	0	0	1	1	0	1	0	0	1
12	0	1	1	0	1	1	0	1	1	1
13	1	1	0	1	1	0	0	1	1	1
14	0	1	0	1	1	1	1	0	1	0
15	0	1	0	1	0	0	1	0	0	1
16	1	0	1	0	0	1	0	1	1	1
17	0	0	0	0	0	1	1	1	0	0
18	0	0	1	0	0	0	1	0	0	1
19	1	1	0	0	0	0	1	0	1	0
20	1	1	1	1	0	0	1	1	0	1

НЕКОГЕРЕНТНАЯ ДЕМОДУЛЯЦИЯ БИНАРНОГО ЧМ СИГНАЛА

(ЧМ-сигнал — сигнал с частотной манипуляцией[10])

1. Введение

Частотная манипуляция является нелинейным методом модуляции и содержит в себе много интересных моментов, особенно при демодуляции сигнала. В частности, таким моментом является возможность формирования ЧМ-сигнала с непрерывной фазой. Это ведет к наличию памяти и возможности когерентной демодуляции с использованием решётки фаз, аналогично декодированию свёрточных кодов по алгоритму Витерби. В добавок ЧМ-сигнал с непрерывной фазой по умолчанию (без дополнительного фильтра) имеет более компактный спектр[11] по сравнению, например, с сигналом фазовой манипуляции. Это связано с отсутствием скачков уровня в квадратурах ЧМ-сигнала.

В данной работе изучается некогерентная демодуляция как наиболее простая. Для выполнения работы требуется программа **Octave** с установленным пакетом **signal.** В этом пакете содержатся функции по обработке сигналов. Версия **Octave** должна быть не ниже **3.8.0**. Рекомендуется сайт http://www.tatsuromatsuoka.com/octave/Eng/Win/.

2. Основные сведения из теории

Частотной манипуляцией называют кодирование потока битов в виде гармонического сигнала с переключением частоты (ЧМ-сигнал), при этом амплитуда сигнала не меняется, а начальная фаза не определена (может быть любой). Слово «кодирование» понимается буквально, так как сейчас при формировании сигналов широко используется техника прямого цифрового синтеза **(DDS, Direct Digital Synthesis),** то есть ЧМ-сигнал сначала формиру-

---

10 Манипуляцией называют модуляцию для **цифровых** систем связи

11 Под компактностью подразумевается не столько ширина спектра, сколько мера сосредоточенности мощности сигнала в заданном диапазоне частот. Это определяется скоростью убывания мощности с ростом отклонения частоты от центральной.

ется в коде, а затем идет процедура цифро-аналогового преобразования (ЦАП) и полосовая фильтрация.

Фаза ЧМ-сигнала формируется из принципа непрерывности, так как это ослабляет требования к выходному усилителю мощности. Скачки фазы при смене частоты приводят к скачк**а**м в ЧМ-сигнале, которые имеют широкий спектр. Соблюдение принципа непрерывности фазы ЧМ-сигнала позволяет сохранить гладкость сигнала при скачкообразном изменении частоты (рис. 1).

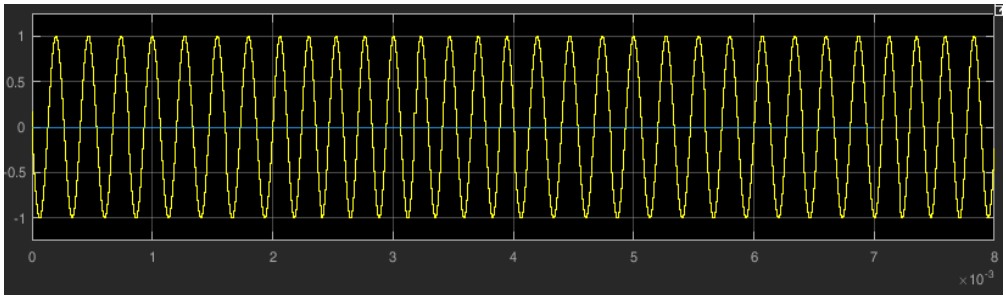


Рис. 1 ЧМ-сигнал с непрерывной фазой и скачкообразным переключением частот: 3,75 кГц и 4,25 кГц.

Длительность символа (бита) на рис. 1 равна 1 мс. Первый импульс имеет 3,75 периодов на символ, то есть частоту 3,75 кГц (допустим, это бит **0**). Второй — аналогично, но начальная фаза у него другая[12], а третий — 4,25 периодов, что дает частоту 4,25 кГц (это бит **1**). ЧМ-сигнал при этом сохраняет непрерывность и декодируется как **00110001.**

Начальная фаза импульса зависит от текущей частоты и от предыдущей фазы, что говорит о наличии памяти в ЧМ-сигнале без разрыва фазы. Кратность ¼ (4,25=4,00+0,25) дает четыре начальных фазы, например, 0, 90, 180 и 270 градусов. Число ¼ означает четверть окружности, поэтому и шаг кратен 90 градусам; или по-другому: девиация частоты 250 Гц составляет четвертую часть от частоты следования символов 1 кГц.

Демодуляция с учетом памяти возможна лишь при когерентной реализации (то есть при наличии петли ФАПЧ[13]). Некогерентный демодулятор память не учитывает, так как для этого нужна информация о фазах.

12 Проследите это!
13 Фазовая автоподстройка частоты

ЧМ-сигнал с центральной несущей частотой f_0 записывается в виде

$$s_{FM}(t)=A\cos\left[2\pi\left(f_0 t+f_d\int_{-\infty}^{t}s_m(t)\,dt\right)+\varphi\right]\ , \tag{21}$$

где f_d — частота девиации (на рис. 1 она равна 250 Гц),

$$s_m(t)=\sum_{n\in\mathbb{Z}}a_n g(t-nT)\ ,\quad |s_m(t)|\le 1\ ,$$ —

модулирующий сигнал[14], a_n — передаваемые биты в биполярном коде ±1[15], $g(t)\ge 0$ — формирующий импульс, отличный от нуля на интервале $0\le t<T$. Этот импульс, после интегрирования, отвечает за форму спектра ЧМ-сигнала.

При скачкообразном изменении частоты формирующий импульс является прямоугольным. Именно этот вариант изучается в данной работе. В системе сотовой связи GSM формирующий импульс сглаживается гауссовским фильтром, поэтому там частота изменяется не скачкообразно, но достаточно быстро, что позволяет выдержать частотную маску, диктуемую стандартами радиосвязи.

ЧМ-сигнал, генерируемый по (21), является полосовым **(band-pass).** Практически же генерируются лишь квадратуры, то есть эквивалентный низкочастотный сигнал **(baseband).** Выведем практическую формулу для формирования квадратур без разрыва фазы.

Так как формируются лишь квадратуры, то приравняем центральную частоту к нулю $f_0=0$ и запишем полную фазу

$$\Theta(t)=2\pi f_d\int_{-\infty}^{t}\sum_{n\in\mathbb{Z}}a_n g(t-nT)\,dt+\varphi\ , \tag{22}$$

и определим правило формирования непрерывной фазы на символьном интервале k

$$\Theta_k(kT+t)=2\pi f_d\int_{0}^{t}a_k g(t)\,dt+\varphi_{k-1}\ ,\quad 0\le t<T\ , \tag{23}$$

---

14 Сигнал с линейной модуляцией. ЧМ-сигнал имеет **не**линейную модуляцию.

15 Может рассматриваться не только поток битов ±1, но и поток цифр ±1, ±3, … .

где $\varphi_{k-1}=\Theta_{k-1}(kT)$ — конечная фаза на предыдущем символьном интервале.

Функциональная схема формирования ЧМ-сигнала без разрыва фазы приведена на рис. 2.

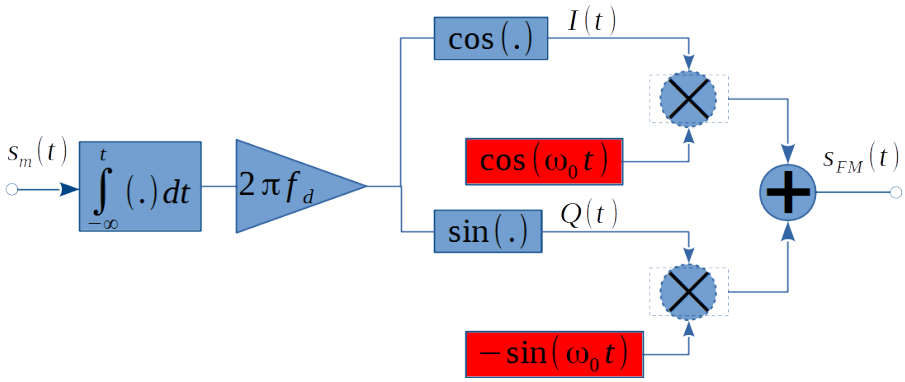


Рис. 2 Функциональная схема формирования ЧМ-сигнала без разрыва фазы.

Кодирование битов **00110001** в квадратурах показано на рис. 3 и рис. 4.

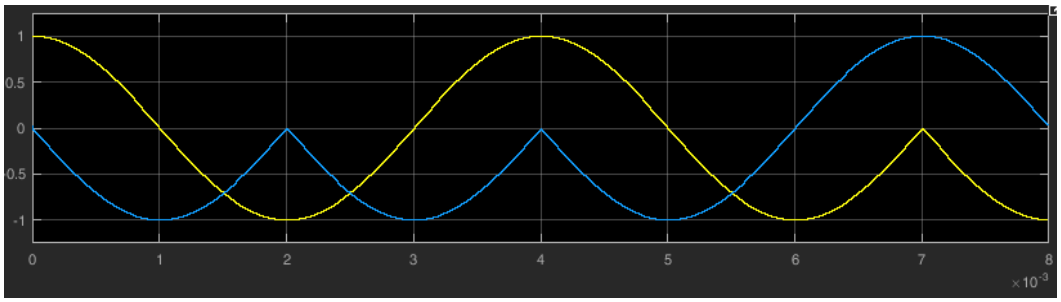


Рис. 3 Квадратуры ЧМ-сигнала, соответствующие рис. 1.

На рис. 4 показан относительный угловой ход вектора в процессе кодирования рассматриваемой битовой последовательности. Начальное состояние вектора $(I,Q)=(1,0)$. Бит **0** соответствует частоте 3,75 кГц, что ниже центральной частоты 4 кГц, поэтому вращение вектора при бите **0** должно быть приторможено. Вектор с частотой 4 кГц вращается против часовой стрелки (так принято в радиотехнике). На рис. 4 показаны квадратуры, то есть вращения относительно вектора 4 кГц. Торможению соответствует вращение по часовой стрелке, ускорению (бит **1**) — против часовой стрелки.

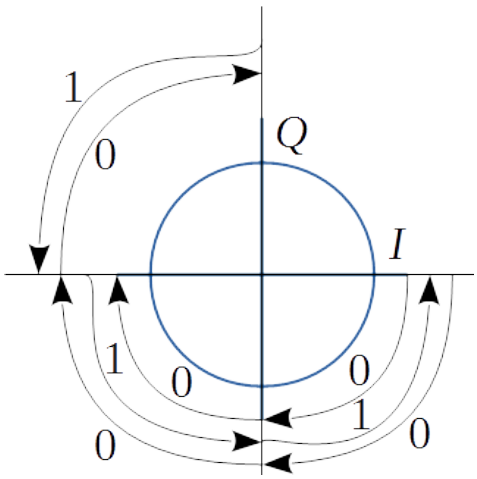


Рис. 4 Квадратуры ЧМ-сигнала, соответствующие рис. 1.

На рис. 3 квадратура I показана желтым цветом, Q — синим. После первого бита квадратуры переходят из состояния $(I,Q)=(1,0)$ в состояние $(I,Q)=(0,-1)$. Отследите этот момент и дальнейшие на обоих рисунках.

Наконец, определим индекс модуляции m — это отношение разноса частот к символьной скорости передачи данных. В рассматриваемом варианте индекс равен

$$m=\frac{(4{,}25-3{,}75)\ \text{кГц}}{1\ \text{кГц}}=0{,}5\ .$$

Оценка среднего спектра мощности ЧМ-сигнала с индексом модуляции ½ дана на рис. 5.

Видно, что уровень первого бокового лепестка составляет −23 дБ. Этот выигрыш достигается за счёт непрерывности фазы. Для сигнала с разрывом фазы уровень первого бокового равен −13 дБ.

Измеренная скорость затухания боковых лепестков составляет 24 дБ/октаву[16]. Такая скорость соответствует асимптотике спектра мощности $1/f^8$! Это проверяется расчетом

$$10\lg 2^8\approx 80\cdot 0{,}3=24\ \text{дБ}.$$

Спектр мощности ЧМ-сигнала с разрывом фазы соответствует асимптотике $\frac{1}{f^2}$.

16 Октава — отношение граничных частот диапазона как один к двум. Например, диапазон (6…12) кГц.

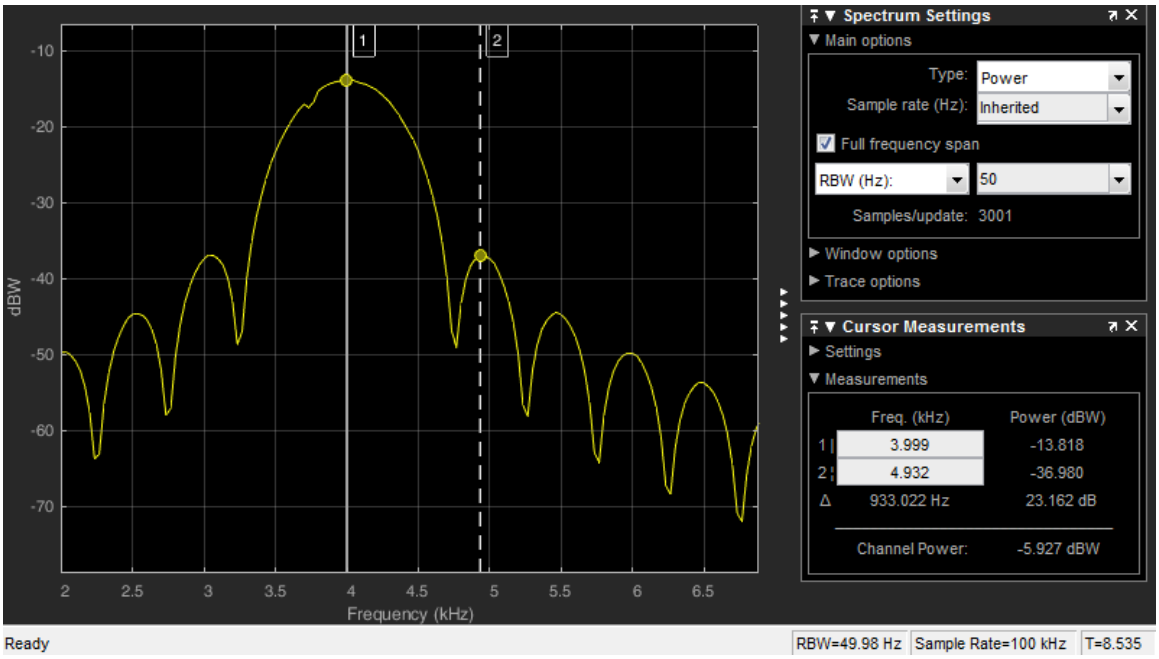


Рис. 5 Оценка среднего спектра мощности ЧМ-сигнала с непрерывной фазой и индексом модуляции ½.

Полоса частот ЧМ-сигнала, определенная по первым нулям, вычисляется по формуле

$$B_0 = 3m/T \quad . \tag{24}$$

Индекс модуляции ½ — это минимальный вариант, обеспечивающий ортогональность сигналов, соответствующих биту **0** и **1.** Ортогональность (равенство корреляционного интеграла нулю) важна при когерентной демодуляции, когда есть контроль над начальной фазой генератора опорного колебания в приемнике. При некогерентном приеме ортогональность не важна — там важен индекс модуляции: чем он больше, тем меньше коэффициент корреляции между сигналами **0** и **1.** Какая при этом у них разность фаз φ — не важно. Для иллюстрации сказанного рассмотрим корреляционный интеграл (нормированный)

$$r_I = \frac{2}{T}\int_0^T \cos\left(2\pi(f_0 - f_d)t\right)\cos\left(2\pi(f_0 + f_d)t + \varphi\right)dt = \\ = \frac{\sin\left(4\pi f_0 T + \varphi\right)}{4\pi f_0 T} + \frac{\sin\left(4\pi f_d T + \varphi\right)}{4\pi f_d T} = \frac{\sin\left(2\pi k + \varphi\right)}{2\pi k} + \frac{\sin\left(2\pi m + \varphi\right)}{2\pi m} \quad , \tag{25}$$

где $k = 2f_0 T$, в рассматриваемом варианте $k = 8$. Интеграл для квадратуры Q получается дифференцированием (25) по фазе φ

$$r_Q = \frac{d r_I}{d \varphi} = \frac{\cos(2\pi k + \varphi)}{2\pi k} + \frac{\cos(2\pi m + \varphi)}{2\pi m} \quad . \tag{26}$$

При некогерентной демодуляции важна общая степень неортогональности, так как такой приемник содержит устройство вычисления модуля квадратур

$$r = \sqrt{r_I^{\,2} + r_Q^{\,2}} \quad . \tag{27}$$

Мера неортогональности r в зависимости от индекса модуляции отражена на рис. 6. Параметру $k = 100$ соответствует 50 периодов центральной несущей частоты на символьном интервале.

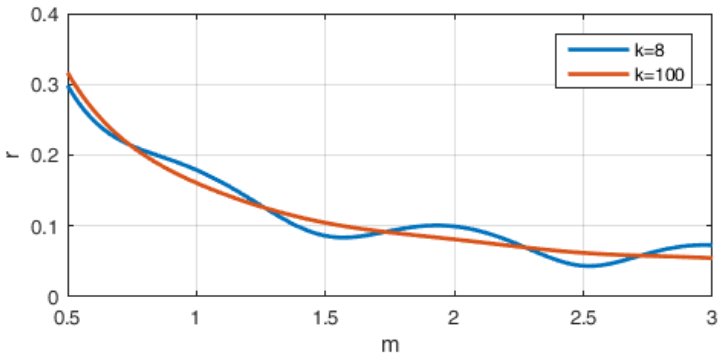


Рис. 6 Мера неортогональности двух ЧМ-сигналов в зависимости от индекса модуляции.

При индексе модуляции $m = 1$ средний спектр мощности ЧМ-сигнала содержит особенности в виде игл на частотах модуляции (4500 и 3500 Гц, рис. 7). Это связано с формированием квадратур.

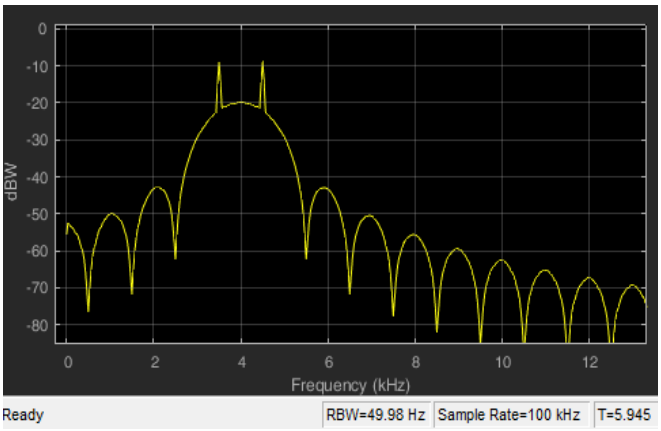


Рис. 7 Средний спектр мощности ЧМ-сигнала при индексе модуляции 1.

Набег фазы за длительность символа для $m = 1$ равен 180 градусам. Косинус-квадратура I при этом нечувствительна к знаку набега фазы, так как косинус — четная функция, поэтому как бы биты ни сочетались, косинус-квадратура никогда не будет иметь изломов — всегда останется гладкой. Синус-квадратура при смене бита будет иметь излом. Чистота косинус-квадратуры и дает иглы (практическое наблюдение дельта-функции) в спектре на частотах 4500 и 3500 Гц.

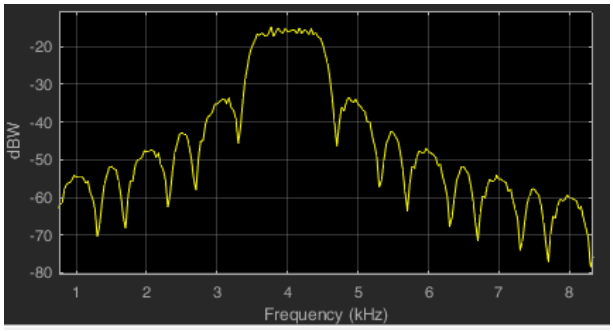


Рис. 8 Средний спектр ЧМ-сигнала при индексе модуляции 0,625.

Чтобы исключить такое доминирование частот, индекс модуляции в данной работе рекомендуется брать 0,625. В этом случае набег фазы за длительность символа равен 5/16 от длины окружности (от 360 градусов), и даже при неизменных битах фаза последовательно примет все 16 значений (5 и 16 — взаимно простые числа) по закону

$$(5n) \bmod 16 = 0,5,10,15,4,9,14,3,8,13,2,7,12,1,6,11,\ldots \ ,$$

что способствует прямоугольности среднего спектра (рис. 8). При этом уровень первого бокового лепестка 18 дБ, а скорость убывания — около 20 дБ/октаву. Пример квадратур показан на рис. 9.

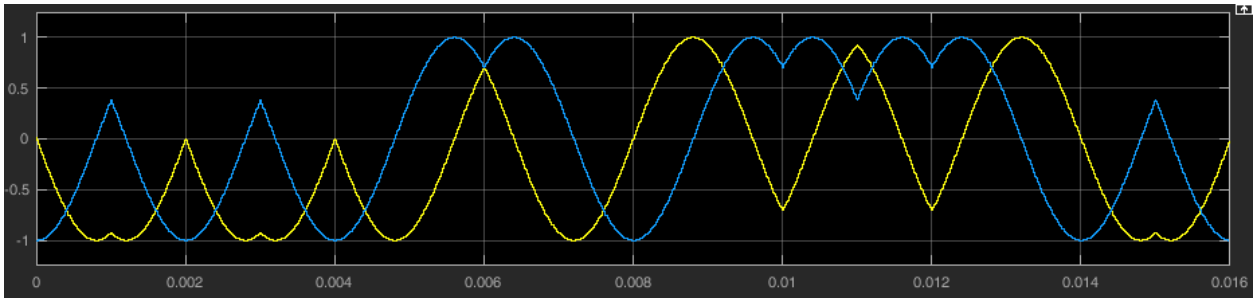


Рис. 9 Квадратуры ЧМ-сигнала с индексом модуляции 0,625. Для 16 символов.

Функциональная схема некогерентного ЧМ-демодулятора показана на рис. 10.

В программно реализованном макете нет устройства восстановления фазы тактовых импульсов, так как для упрощения моделируется канал с заранее известной задержкой, определяемой постоянной задержкой в фильтрах (в частности, в согласованном фильтре). Фаза выставляется заранее в виде вычисленной константы. Также в макете после фильтра нижних частот (ФНЧ) идет блок прореживания отсчетов **(downsampling)** для увеличения быстродействия. Отсчетное устройство в макете упрощено до блока прореживания отсчетов с шагом N и заданным смещением **(offset).** Параметр **offset** и есть фаза тактовых импульсов.

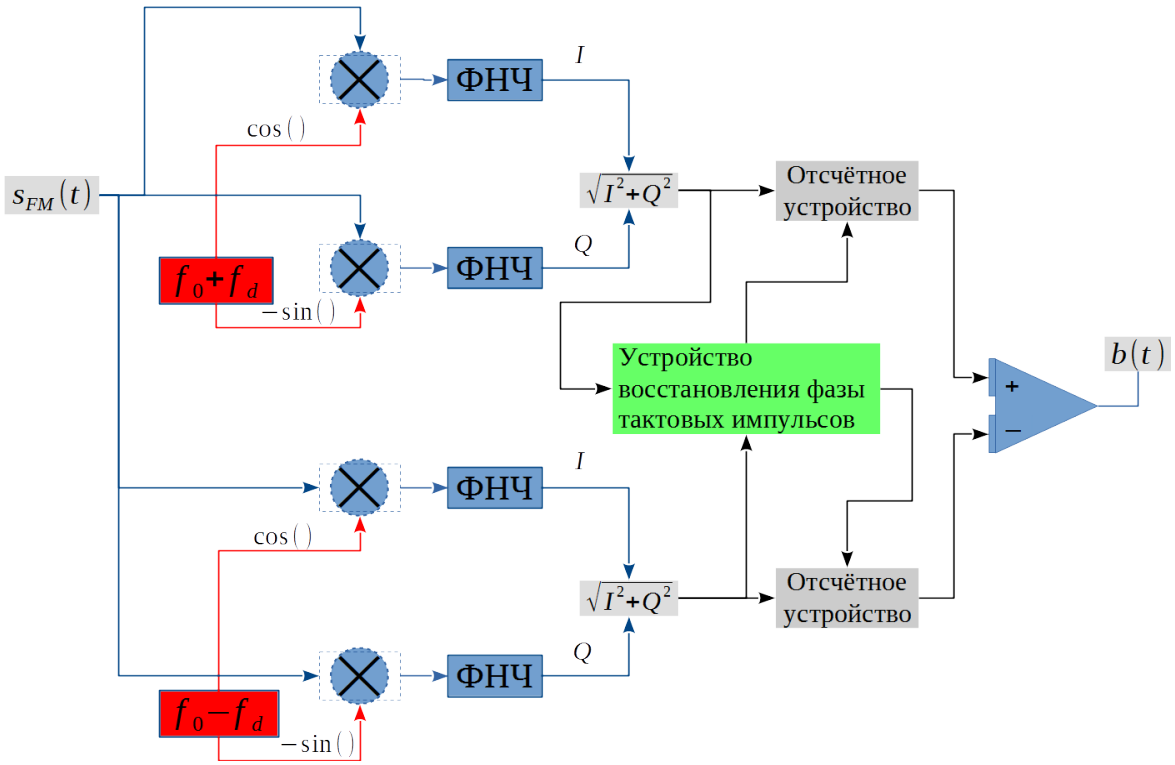


Рис. 10 Функциональная схема некогерентного демодулятора ЧМ-сигнала

ЧМ-сигнал $s_{FM}(t)$ поступает на четыре смесителя (умножителя), на вторые входы которых поступают гармонические опорные сигналы. Верхний канал, имеющий два смесителя, настроен на верхнюю частоту f_0+f_d, нижний — на нижнюю f_0-f_d. Начальные фазы опорных генераторов произвольные, но одинаковые. Сигналы с выхода смесителей поступают на ФНЧ, задача которого — подавить сигнал в окрестности суммарной частоты. После ФНЧ, по-хорошему, должен идти согласованный фильтр; в рассматриваемом варианте фильтр согласован с прямоугольным импульсом. Его задача — максимально ослабить действие теплового шума и сохранить при этом энергию полезного сигнала (максимизировать отношение сигнал-шум на выходе). Дальше по квадратурам вычисляется модуль, который поступает на отсчетное устройство, выбирающее отсчеты в моменты времени, задаваемые тактовым генератором, закольцованным петлей ФАПЧ (устройство восстановления фазы тактовых импульсов). Отсчетное устройство является интерполятором, на один вход которого подается отсчитываемый сигнал, а на другой — сигнал, определяющий момент взятия отсчета относительно длительности символа

(это и есть фаза тактовых импульсов). Далее выходы двух отсчетных устройств сравниваются компаратором и выставляется соответствующий бит (**0** или **1**).

ФНЧ в макете спроектирован в виде цифрового КИХ-фильтра с переходной полосой (2500...5000) Гц с подавлением 60 дБ и неравномерностью в полосе пропускания 1 дБ. Согласованный фильтр, идущий после ФНЧ, является КИХ-фильтром с коэффициентами-константами (согласование с прямоугольным импульсом).

Частота дискретизации для ЧМ-сигнала в макете выбрана равной 100 кГц. После ФНЧ перед согласованным фильтром стоит блок **downsample** с коэффициентом 4, поэтому согласованный фильтр имеет порядок (длину импульсной характеристики КИХ-фильтра)

$$\frac{100\,\text{кГц}}{4\cdot 1\,\text{кГц}}=25 \quad .$$

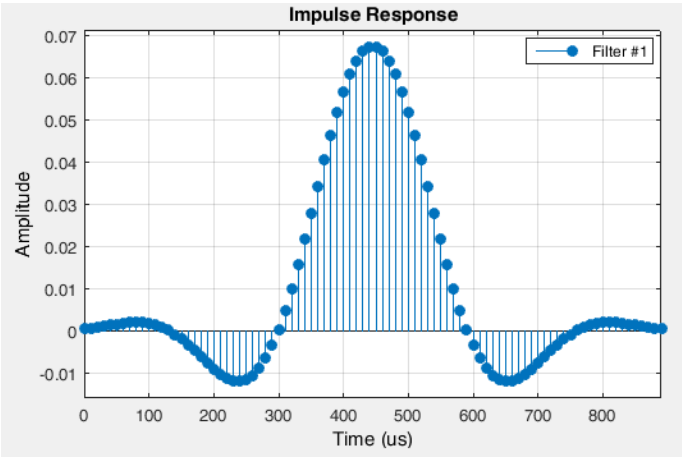


Рис. 11 Импульсная характеристика ФНЧ в некогерентном ЧМ-демодуляторе

Отсчетное устройство делает выборки через каждые 25 отсчетов, но начинает выборку с задаваемого момента времени (согласно фазе тактовых импульсов). Этот момент времени в макете зависит от задержки в ФНЧ, которая равна 44,5 отсчета или 0,445 мс (рис. 11).

В согласованном фильтре сигнал задерживается ровно на длительность символа, поэтому на фазу тактовых импульсов этот фильтр не повлияет. Единственно, это повлияет на процедуру сравнения битов, так как принятые биты будут запаздывать на один такт. Задержка в ФНЧ не кратна длительности символа, поэтому на фазу она повлияет. Так как после ФНЧ стоит блок **downsample,** то длительность символа равна 25 отсчетам, значит фаза должна быть равна $25\cdot 0{,}445\approx 11$. Фаза округлена до целого числа, так как в ма-

кете отсчетное устройство не является интерполятором, определяющим отсчет, задаваемый **дробным** тактом, по известным соседним отсчетам, находящимся в обычных целых тактах времени. В данном случае такой точности временной синхронизации хватает.

3. Ход работы

Запустить программу **Octave.** В командной строке выполнить

```
pkg load signal
```

Открыть с помощью редактора **Octave** файл-скрипт **fsk.m.**

Выключить шум с помощью установки отношения сигнал-шум `EbNo` на 120 дБ. Индекс модуляции выставить на `m=0,625.` Выключить режим статистических испытаний, выставив `Nstat=1`. Набрать в переменную `inputBits` для своего варианта битовую последовательность (Приложение А).

1. Привести в отчёте осциллограммы во всех контрольных точках схемы формирования ЧМ-сигнала:

- Битовую последовательность в биполярном коде: переменная `input;`
- Квадратуры *I* и *Q:* переменные `cpfskRe` и `cpfskIm;`
- Сформированный ЧМ-сигнал: переменная `s_FM;`
- Сигналы с выходов смесителей: переменные `mix0Re, mix0Im, mix1Re` и `mix1Im;`
- Сигналы с выходов ФНЧ: переменные `Lpf0Re, Lpf0Im, Lpf1Re` и `Lpf1Im;`
- Сигналы после **downsample:** переменные `Lpf0ReDown, Lpf0ImDown, Lpf1ReDown` и `Lpf1ImDown;`
- Сигналы с выходов согласованного фильтра: переменные `I0, Q0, I1` и `Q1;`
- Сигналы с выходов вычислителя модуля: переменные `out0` и `out1;`

- Сигналы после **offset-downsample:** переменные `Detector0` и `Detector1;`
- Демодулированные биты до удаления задержки `bitsDelay` и после: переменная `demodulatedBits.`

Единое для всех графиков время вывести в единицах символьного интервала `Tsym.`

В качестве примера вывода графиков использовать приведённый в скрипте код, выводящий всего три графика. Убедиться, что биты демодулированы верно. Убедиться, что ЧМ-сигнал сформирован без разрыва фазы несущей частоты. Привести частотные характеристики ФНЧ, воспользовавшись функцией `freqz(h).`

Повторить этот пункт для индекса модуляции `m=1.`

Индекс модуляции выставить на `m=0,625.` Включить режим статистических испытаний, количество испытаний выставив на **10000.**

2. Для ряда значений сигнал-шум `EbNo`

0, 3, 6, 9, 12 и 15 дБ,

записать в таблицу выданные программой вероятности ошибки демодуляции **(Symbol Error Rate).** Здесь же привести вероятности, рассчитанные по формуле для некогерентной ЧМ-демодуляции [3]

$$P_s = \frac{1}{2}\exp\left(-\frac{1}{2}\frac{E_b}{N_0}\right) \quad \text{, (здесь сигнал-шум } \textbf{в разах}\text{).} \qquad (28)$$

Построить соответствующие графики.

3. Выставить сигнал-шум на 120 дБ. Вывести график спектра мощности ЧМ-сигнала[17]: переменная `psdFM.` Измерить полосу сигнала по уровню −3 дБ. Измерить уровень первого бокового лепестка, а также скорость спада мощности в зависимости от частоты (измеряется в дБ/октаву).

4. Сделать §§2–3 для индекса модуляции `m=1.`

17 В макете спектр мощности оценивается с использованием окна Хеннинга, расширяющего в 1,6 раз полосу по уровню −3 дБ.

5. Выставить `m=0,625, Nstat=1` и `EbNo=20 дБ`. Сравнить сигналы с выхода вычислителей модуля (`out0` и `out1`) при включенном и при выключенном согласованном фильтре (см. скрипт, строку «if matched filter is 'off', then use it»).

После каждого пункта сделать выводы. В конце работы сделать общие выводы и ответить на контрольные вопросы.

4. Контрольные вопросы

1. Чем хорош ЧМ-сигнал с непрерывной фазой?

2. Возможна ли некогерентная демодуляция ЧМ-сигнала по решётке фаз (например, по алгоритму Витерби)?

3. К какому виду модуляции относят частотную модуляцию: к нелинейной или линейной? Почему (в какой функции сидит нелинейность)?

4. Почему теоретический график вероятности ошибки заметно отличается от результатов статистических испытаний?

5. Список литературы

1. Бернард Скляр. Цифровая связь.

2. Джон Прокис. Цифровая связь.

3. Ю.П. Акулиничев. Теория и техника передачи информации.

Приложение А Последовательности для подачи на вход ЧМ-модулятора

Номер варианта	Последовательность битов									
1	1	0	0	1	0	1	0	1	1	1
2	1	1	1	0	0	0	0	0	1	0
3	1	0	1	1	1	0	1	1	0	0
4	1	1	1	0	1	1	0	0	1	0
5	0	1	0	1	1	1	1	1	0	1
6	0	1	1	0	0	0	0	0	0	1
7	1	0	0	1	0	1	1	0	0	0
8	0	0	0	1	1	1	1	1	1	1
9	0	1	1	1	0	1	1	1	1	1
10	1	1	1	1	0	0	0	0	0	1
11	1	0	0	1	1	0	1	0	0	1
12	0	1	1	0	1	1	0	1	1	1
13	1	1	0	1	1	0	0	1	1	1
14	0	1	0	1	1	1	1	0	1	0
15	0	1	0	1	0	0	1	0	0	1
16	1	0	1	0	0	1	0	1	1	1
17	0	0	0	0	0	1	1	1	0	0
18	0	0	1	0	0	0	1	0	0	1
19	1	1	0	0	0	0	1	0	1	0
20	1	1	1	1	0	0	1	1	0	1

СПЕКТРЫ СИГНАЛОВ С ЛИНЕЙНОЙ МОДУЛЯЦИЕЙ

Заведомо известный сигнал не несет информации

1. Введение

Данная работа адресована студентам, изучающим цифровую связь, где информация задается в виде последовательности символов, например, битов

...100101101010...

Символы, являющиеся абстракцией (например, числами), пересаживаются на *импульс-носитель* как на своеобразную лошадку, в результате чего получается цифровой сигнал, пригодный для передачи по линии связи. Процесс пересадки называется *модуляцией.*

Автор надеется, что студент усвоил что такое спектр сигнала и знаком с рядом и интегралом Фурье. Если нет, то здесь делать нечего.

Основная особенность данной работы заключается в случайности последовательности символов, что делает спектр Фурье соответствующего цифрового сигнала зависимым от конкретной *реализации* символов. Обратите внимание на слово **реализация**, что говорит о наличии *ансамбля* реализаций (целого *множества* реализаций).

Спектры, содержащие *случайную* составляющую, не всегда нужны, поэтому при необходимости ее ослабляют с помощью *усреднения,* которое выделяет *регулярную* (неслучайную) составляющую, обязательно присутствующую в спектре из-за неслучайной формы *импульса-носителя.*

Почему поток символов случаен?

Дело здесь в том, что передача информации происходит лишь тогда, когда приемник различает изменение сигнала и это изменение на сто процентов он предсказать не может — вот в этой частичной (или полной) непредсказуемости и кроется та случайность, которую можно назвать *информационной* и которая является столпом любой связи.

Поясним предыдущий абзац на житейском примере.

Допустим мы договорились с рыбаком, отплывающим на другой берег, что он в десять часов утра в случае удачного улова зажигает красный сигнальный огонь, а в случае неудачного — зеленый, но один из двух обяза-

тельно должен быть зажжен. Какой это будет цвет, нам (приемнику) заранее не известно, зато *алфавит* (все возможные цвета), разумеется, известен, иначе мы ничего не примем.

Информация от рыбака в час **X** будет получена лишь тогда, когда будет определен цвет (этому может помешать дым костра). В случае определения цвета происходит *разрешение* имевшейся до приема *неопределенности*. Если она снимается полностью, когда уверенность в цвете *на все сто*, то мы получаем всю информацию, а если частично, то — не всю (есть некоторая неуверенность в правильности распознавания цвета).

Сигнальный огонь определенного цвета — это оптический сигнал, в котором заложена информация о текущем символе. Сигнал генерируется передатчиком и распознается приемником. Случайность сигнала заключается в непредсказуемости выбора из заранее известного набора сигналов. Естественно, что у принимаемого сигнала есть и другая случайность: *мерцание* (случайное изменение яркости), которое на больших расстояниях может дать ошибку определения цвета, но такое явление в данной работе не рассматривается, потому что оно относится к процессу демодуляции (угадывания символа).

В системах связи идет последовательный поток сигналов-импульсов, где каждый импульс обозначает один из символов алфавита. Рассмотрим самый простой алфавит — *двоичный (битовый)*.

Предположим, что биту **1** соответствует прямоугольный импульс, а биту **0** — отсутствие импульса (пауза). Тогда можно заметить, что для битового потока из одних нулей **...000...** или одних единиц **...111...** будет генерироваться постоянное напряжение, что сводит на нет возможность передачи такого сигнала через стандартные линии связи. С другой стороны, если идет последовательность из чередующихся нулей и единиц **...010101...**, то спектр такого сигнала будет соответствовать спектру меандра: гармоника основного тона плюс треть от гармоники утроенного тона плюс и так далее по всем нечетным номерам гармоник.

Из этих двух примеров, где рассматривались неслучайные сигналы, видно, что на спектр влияет не только длительность импульса, но и структура последовательности символов — *шаблон,* который для неслучайных сигналов повторяется (копируется).

Чтобы избавиться от случайности в спектре, усредняют **квадрат модуля,** пропорциональный **мощности** текущей гармоники. Именно *средний квадрат модуля* в данной работе играет ключевую роль.

Структура отчета должна соответствовать следующему:

- ✔ Титульный лист;
- ✔ Ход работы;
- ✔ Ответы на вопросы;
- ✔ Выводы.

Принимаются форматы файлов **doc, docx, odt** и **pdf.**

2. Сведения из теории

2.1 Спектральная плотность энергии импульса-носителя

Электрические процессы, связанные с передачей и приемом информации и являющиеся физическим объектом, отображаются на математический объект — *сигнал* — функцию, зависящую от времени.

Например, сигналом является *прямоугольный импульс,* равный разности двух функций *Хевисайда* $h(t)$ (рис. 27, рис. 28).

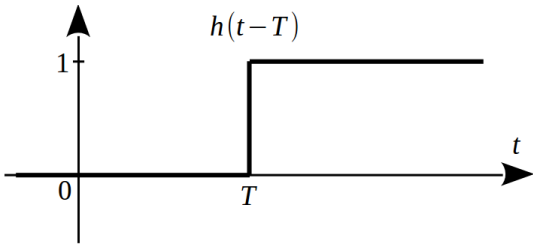


Рисунок 27: Функция Хевисайда — единичный скачок

Единичный скачок не ограничен во времени, а сигнал, переносящий один символ, обязан иметь конечную длительность. По этой причине при построении прямоугольного импульса амплитуды двух скачков выбираются равными

$$\mathrm{rect}(t)=A\cdot h(t)-A\cdot h(t-T)\ . \qquad (29)$$

Нулевой уровень не считается сигналом, поэтому прямоугольный импульс ограничен во времени. Параметр T в (29) называется *длительностью импульса,* а A — *амплитудой импульса.*

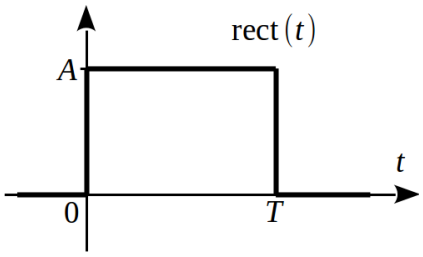


Рисунок 28: Прямоугольный импульс

Сигналы, существующие в физических электрических цепях, никогда не имеют строго прямоугольной формы, так как для скачкообразного изменения амплитуды с нуля до некоторой ограниченной величины требуется множество гармоник с неограниченно растущими частотами[18]. Плюс после такого скачка необходимо на некоторое время удерживать постоянный уровень, чего также не бывает, так как в цепях идут переходные процессы, связанные с неустранимыми емкостями и индуктивностями.

Вывод: прямоугольный импульс — теоретическая абстракция, к которой можно лишь приближаться, что и делается в разных генераторах по-разному.

Рассмотренный прямоугольный импульс можно считать одним из многих носителей, используемых для кодирования битов. Разные носители имеют разные энергетические спектры, отличающиеся, в частности, скоростью убывания энергии в зависимости от частоты. Скорость убывания играет важную роль, ведь чем более *компактным* будет спектр (компактным не означает узким), тем меньшим будет уровень внеполосного излучения (смотри понятие *спектральная маска сигнала*).

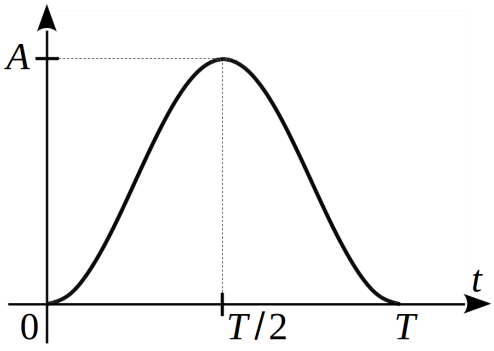


Рисунок 29: Импульс приподнятого косинуса

Примером более реального носителя является импульс *приподнятого косинуса* (рис. 29), задаваемый функцией

$$g_c(t)=\frac{A}{2}\left(1-\cos\frac{2\pi t}{T}\right) \quad . \qquad (30)$$

Но он также идеален, так как его спектр не ограничен по частоте, зато его энергия убывает с ростом частоты быстрее, чем энергия прямоугольного импульса.

18 Смотри ряд для меандра (38) и его первую производную в точках перепада уровня

Чтобы убедиться в различии спектров, вычислим их с помощью известного интеграла Фурье.

Для прямоугольного импульса

$$G_{rect}(f)=\int_0^T A\,e^{-2\pi jft}dt=\frac{A}{-2\pi jf}e^{-2\pi jft}\Bigg|_0^T=AT\,\frac{\sin(\pi fT)}{\pi fT}e^{-\pi jfT}\quad . \qquad (31)$$

Для расчета спектральной плотности (в данном случае — энергии) необходимо взять квадрат модуля от (31) (объяснение смотри в (45))

$$\Phi_{rect}(f)=|G_{rect}(f)|^2=(AT)^2\left(\frac{\sin(\pi fT)}{\pi fT}\right)^2\quad ,\quad B^2\cdot c/Гц\quad . \qquad (32)$$

Если A измеряется в вольтах, то (32) — в $вольт^2\cdot секунда/герц$. Для сопротивления 1 Ом это будет $джоуль/герц$, что и говорит о плотности энергии, то есть сколько джоулей приходится на полосу один герц.

Интеграл от (32) по всем частотам даст полную энергию сигнала, которую можно вычислить значительно проще через интеграл от квадрата сигнала

$$\int_{-\infty}^{\infty}(AT)^2\left(\frac{\sin(\pi fT)}{\pi fT}\right)^2df=A^2T\quad , \qquad (33)$$

что подтверждается известным в математике равенством $\int_{-\infty}^{\infty}\left(\frac{\sin x}{x}\right)^2dx=\pi$.

На нулевой частоте спектральная плотность энергии прямоугольного импульса принимает значение $(AT)^2$, так как здесь работает первый замечательный предел

$$\lim_{x\to 0}\frac{\sin x}{x}=1\quad .$$

Значение $(AT)^2$ говорит о том, что чем больше длительность импульса, тем больше плотность энергии постоянной составляющей, что логично, так как импульс при этом больше становится похожим на постоянное напряжение.

Исходя из равенства $\Phi_{rect}(1/T)=0$ делаем вывод, что гармоники частоты $1/T$ в данном импульсе нет. Этой гармоники нет и для случайной последовательности прямоугольных импульсов и пауз одинаковой длительности. Данный факт становится более понятным, если нарисовать *меандр* — неслучайное переключение напряжения с частотой $1/T$. Наглядно видно (рис. 30), что меандр хорошо приближается гармоникой с периодом $2T$ и частотой $f_1=1/2T$, причем эта гармоника приподнята на половину амплитуды *A*.

Частота f_1 определяет *первую гармонику* меандра. Амплитуда второй гармоники равна нулю, так как на ее период приходится постоянное напряжение меандра (*A* или 0), которое не содержит никаких гармоник.

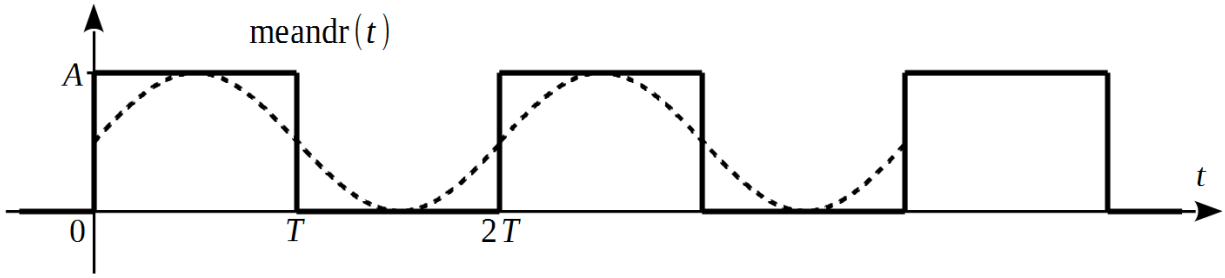
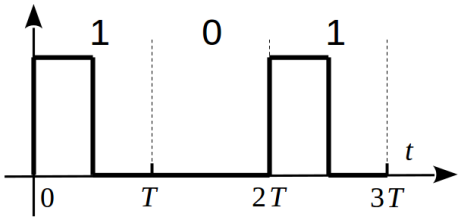


Рисунок 30: Меандр — закономерное переключение напряжения с частотой $1/T$

В цифровой связи ценится знание приемником моментов начала импульса и его окончания, так как в этом случае обеспечивается качественное распознавания импульса. Приемник должен уметь генерировать тактовые импульсы, *выровненные по фазе* с принимаемыми импульсами, и, значит, следующие с частотой $1/T$. Если система связи подразумевает специальный канал для передачи синхроимпульсов от передатчика к приемнику, то это хорошо, а если нет, то приемник должен их выделить из импульсов, содержащих только данные.

Мы только что выяснили, что энергетическая доля частоты $1/T$ равна нулю, так что никакие резонансные контуры не помогут выделить тактовые импульсы напрямую.



Один из выходов — сложить (по модулю два) исходный сигнал и сигнал, задержанный на половину такта. Это эквивалентно повышению частоты переключения в два раза, что при псевдослучайном исходном битовом потоке даст постоянный уровень средней мощности искомой тактовой частоты $1/T$. По такому принципу работает *схема восстановления тактовой частоты.*

Определение спектра меандра для «продвинутых»

==========

Меандр равен сумме большого количества прямоугольных импульсов, **задержанных** друг относительно друга на постоянную величину $2T$, поэтому его спектр можно вычислить, просуммировав спектры типа (31), отличающиеся только лишь **фазовым множителем,** отражающим задержку

$$G_{meandr}(f)=AT\frac{\sin\pi fT}{\pi fT}\left[\ldots+e^{7\pi jfT}+e^{3\pi jfT}+e^{-\pi jfT}+e^{-5\pi jfT}+e^{-9\pi jfT}+\ldots\right] . \quad (34)$$

Давайте упростим это выражение, вынося за скобки экспоненту $e^{-\pi jfT}$ и замечая, что сумма оставшихся экспонент дает сумму дельта-функций

$$\begin{aligned} G_{meandr}(f)&=G_{rect}(f)\left[\ldots+e^{2\pi j2f2T}+e^{2\pi j1f2T}+1+e^{-2\pi j1f2T}+\ldots\right]= \\ &=\frac{G_{rect}(f)}{2T}\left[\ldots+\delta\left(f+\frac{1}{2T}\right)+\delta(f)+\delta\left(f-\frac{1}{2T}\right)+\delta\left(f-\frac{2}{2T}\right)+\ldots\right] . \end{aligned} \quad (35)$$

Здесь использовался ряд Фурье для дельта-функции

$$\sum_{k\in\mathbb{Z}}\delta(x-2\pi k)=\frac{1}{2\pi}+\frac{1}{\pi}\sum_{k=1}^{\infty}\cos kx=\frac{1}{2\pi}\sum_{k\in\mathbb{Z}}e^{-jkx}$$

и ее свойство изменения масштаба

$$\delta(kx+b)=\frac{1}{k}\delta(x+b/k) .$$

Факт замены суммы экспонент на сумму масштабированных дельта-функций нетрудно понять, если построить график частичной суммы экспонент, взяв для начала 5 слагаемых, затем 21 и так далее (рис. 31).

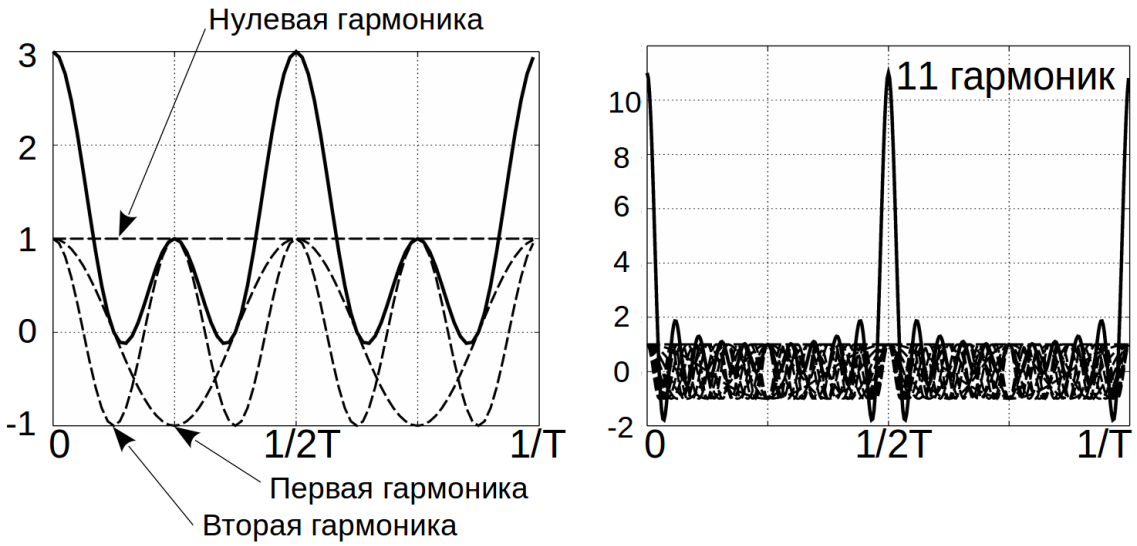


Рисунок 31: Частичные суммы косинусов кратных частот

Особенность этих экспонент в том, что каждая имеет комплексно-сопряженную пару, поэтому по сути суммируются только косинусы. Из рис. 31 видно, что сумма экспонент — периодическая функция, в основном колеблющаяся возле нуля, но в строго определенные моменты времени резко возрастающая. Возрастание происходит из-за пучностей, где все косинусы равны единице.

Спектр меандра (35) за счет дельта-функций является *линейчатым,* причем дельта-функции появляются только при неограниченном количестве слагаемых, то есть когда последовательность прямоугольных импульсов становится периодической. Когда количество импульсов велико, но ограничено, их спектр будет сплошным, но с ярко выраженной линейчатостью.

Наличие дельта-функций создает проблему неограниченности при численном расчете амплитуд спектра (35), так как $\delta(0)=\infty$. Чтобы ее избежать, спектр вычисляют как предел

$$G(f)=\lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2} g(t)e^{-j2\pi ft}dt \quad .$$

Сигналы, у которых энергия неограниченная, называют *мощностными.* Сигналы с ограниченной энергией относятся к *энергетическим.*

Мощностной сигнал не обязан быть периодическим, так как сигналы в связи часто отображают случайную последовательность битов, у которой нет периода.

Замечая из (31), что

$$G_{rect}\left(f=\frac{k}{2T}\right)=AT\,\frac{\sin\pi k/2}{\pi k/2}e^{-\pi jk/2}=\begin{cases}AT, & k=0\\ 0, & k-\text{чётное}\\ -j\dfrac{2AT}{\pi k}, & k-\text{нечётное}\end{cases},$$

и вычисляя **мощностной** спектр амплитуд меандра, получим

$$G_{meandr}(k)=\begin{cases}\dfrac{A}{2}, & k=0\\ 0, & k-\text{чётное}\\ -j\dfrac{A}{\pi}\dfrac{1}{k}, & k-\text{нечётное}\end{cases}. \qquad (36)$$

Дельта-функции при вычислении спектра для **мощностного** сигнала переходят в цифровые дельта-функции, принимающие значения **0** и **1.** В этом можно убедиться, если в (35) взять три слагаемых $N=3$ симметрично единице

$$G_{meandr}^{N=3}(f)=G_{rect}(f)\left[e^{2\pi j1f2T}+1+e^{-2\pi j1f2T}\right],$$

которые отвечают за три прямоугольных импульса, и разделить данный спектр на длительность **отрезка** меандра $5T$ (три импульса, две паузы), а затем вычислить предел при $N\to\infty$ на частоте дискретизации $k/2T$

$$\lim_{N\to\infty}\frac{G_{meandr}^{N}(f)}{T_{meandr}(N)}\bigg|_{f=k/2T}=G_{rect}(f=k/2T)\cdot\lim_{N\to\infty}\frac{N}{(2N-1)T}=\frac{1}{2T}G_{rect}(f=k/2T).$$

Из этого и следует искомый переход аналоговых дельта-функций в цифровые ...00**1**00....

Из выражения (36) видно, что в спектре меандра имеется постоянная составляющая и синусоидальные гармоники нечетных частот. Для уверенно-

сти в этом вычислим коэффициенты классического ряда Фурье с периодом первой гармоники $2T$

$$s_T(t)=\frac{a_0}{2}+\sum_{k=1}^{\infty} a_k\cos(\pi k t/T)+b_k\sin(\pi k t/T) \qquad (37)$$

по известным формулам

$$a_0/2=G_{meandr}(0)=A/2 \ , \ a_k=G_{meandr}(k)+G_{meandr}(-k)=0 \ ,$$

$$b_k=j\left[G_{meandr}(k)-G_{meandr}(-k)\right]=\frac{2A}{\pi k} \ , \ k-\text{нечётное} \ ,$$

и подставим их в (37)

$$s_{meandr}(t)=\frac{A}{2}+\frac{2A}{\pi}\sum_{k=1,3,5,\ldots}\frac{1}{k}\sin(\pi k t/T) \ , \text{(период равен } 2T\text{).} \qquad (38)$$

Выражение (38) полностью совпадает с рядом Фурье для меандра.

Рассмотрев вычисление мощностного спектра меандра, мы в какой-то степени забежали вперед, подготавливая почву для §2.3 Пороговое декодирование.

==========

Наконец, вычислим спектр Фурье приподнятого косинуса (30)

$$G_c(f)=\frac{1}{2}G_{rect}(f)-\frac{A}{2}\int_0^T\cos\left(\frac{2\pi t}{T}\right)e^{-2\pi jft}dt= \\ =\frac{AT}{2}\frac{1}{1-(fT)^2}\frac{\sin(\pi fT)}{\pi fT}e^{-j\pi fT} \ . \qquad (39)$$

Расчет спектральной плотности энергии дает

$$\Phi_c(f)=|G_c(f)|^2=\left(\frac{AT}{2}\right)^2\left[\frac{1}{1-(fT)^2}\right]^2\left[\frac{\sin(\pi fT)}{\pi fT}\right]^2 \ . \qquad (40)$$

Таким образом, энергетический спектр косинусного импульса убывает как $1/f^6$ с ростом частоты, то есть на четыре порядка быстрее спектра прямоугольного импульса. Но спектр косинусного шире, так как $\Phi_c(1/T)\neq 0$. Предлагается вычислить, чему равно это ненулевое значение,

а также построить сравнительные графики энергетических спектров обоих импульсов.

2.2 Спектр мощности случайной последовательности чисел

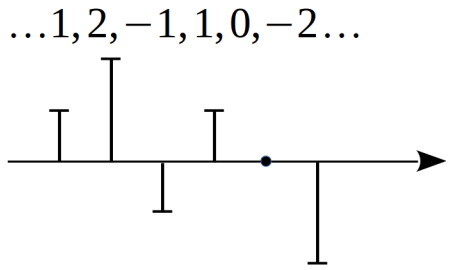


Рисунок 32: Последовательность чисел, представленная дельта-носителем

Считается, что числа, кодирующие символы, сыпятся из черного ящика, и нет им начала, нет им конца… По этой причине вычисляется спектр **мощности,** так как энергия будет неограниченно расти и никакой информации нам не даст.

Если есть слово *мощность*, то числам должен быть сопоставлен сигнал. Сигнал образуется носителем, два из которых мы рассмотрели в параграфе 2.1 Кодирование. Здесь предполагается дельта-носитель (рис. 32)

$$\delta_n = \begin{cases} 1, \text{ если } n=0; \\ 0, \text{ иначе.} \end{cases} , \tag{41}$$

то есть цифровая дельта-функция, имеющая константный спектр

$$G_\delta(f) = T \sum_{n\in\mathbb{Z}} \delta_n e^{-j2\pi nfT} = T \quad , \tag{42}$$

что позволяет отделить влияние носителя на спектр последовательности чисел. Их совмещение произойдет позже.

Последовательность чисел, посаженная на дельта-носитель, — дискретный сигнал, значит его спектр Фурье будет периодическим с периодом $1/T$. Оказывается, что спектр мощности пропорционален квадрату модуля обычного спектра. Это вытекает из того, что спектральная плотность мощности связана с функцией корреляции $R(\tau)$ преобразованием Фурье[19]

$$\Phi(f) = \int_{-\infty}^{\infty} R(\tau) e^{-j2\pi f\tau} d\tau \quad , Вт/Гц. \tag{43}$$

Подставляя в (43) выражение для функции корреляции детерминированного вещественного импульса $g(t)$ длительностью T

19 Для стационарных по отношению к функции корреляции случайных процессов

$$R(\tau)=\frac{1}{T}\int_{-\infty}^{\infty} g(t)\cdot g(t-\tau)\,dt \quad , Bm, \tag{44}$$

последовательно получим

$$\begin{gathered}\Phi(f)=\frac{1}{T}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(t)\cdot g(t-\tau)\,dt\,e^{-j2\pi f\tau}d\tau \quad , \\ \Phi(f)=\frac{1}{T}\int_{-\infty}^{\infty} g(t)\int_{-\infty}^{\infty} g(t-\tau)e^{-j2\pi f\tau}d\tau\,dt \quad , \quad t-\tau=p \quad , \\ \Phi(f)=\frac{1}{T}\int_{-\infty}^{\infty} g(t)\int_{-\infty}^{\infty} g(p)e^{j2\pi fp}dp\,e^{-j2\pi ft}dt \quad , \\ \Phi(f)=\frac{1}{T}G^{*}(f)\int_{-\infty}^{\infty} g(t)e^{-j2\pi ft}dt=\frac{1}{T}G^{*}(f)G(f)=\frac{1}{T}|G(f)|^{2} \quad .\end{gathered} \tag{45}$$

Для последовательности чисел функция корреляции $\phi(i)$ будет дискретной, поэтому спектр мощности найдется через сумму

$$\Phi_{digit}(f)=T\sum_{i\in\mathbb{Z}}\phi(i)e^{-j2\pi ifT} \quad , Bm/Гц, \tag{46}$$

где T — интервал следования чисел, определяющий период спектра мощности по частоте как $f_T=1/T$.

А) Рассмотрим последовательность некоррелированных чисел b_n , $n\in\mathbb{Z}$, с некоторым средним значением μ_b и дисперсией D_b

$$\begin{aligned}&\mu_b=M[b_n],\\ &D_b=M\left[(b_n-\mu_b)^2\right],\\ &r=\frac{M\left[(b_n-\mu_b)(b_{n-i}-\mu_b)\right]}{D_b}=\begin{cases}1, & i=0;\\ 0, & i\neq 0.\end{cases}\end{aligned} \quad , \tag{47}$$

где $M[.]$ — оператор нахождения среднего, r — коэффициент корреляции чисел b_n и b_{n-i} , зависящий только от индекса i.

Тогда функция корреляции данной последовательности будет равна

$$\phi_{bb}(i)=M[b_nb_{n-i}]=M\left[(b_n-\mu_b)(b_{n-i}-\mu_b)\right]+\mu_b{}^2=\begin{cases}D_b+\mu_b{}^2, & i=0;\\ \mu_b{}^2, & i\neq 0.\end{cases} \quad . \tag{48}$$

Подставляя (48) в (46), получим спектр мощности

$$\Phi_{bb}(f)=D_bT+\mu_b{}^2T\sum_{i\in\mathbb{Z}}e^{-j2\pi ifT} \quad . \tag{49}$$

Используя знакомую «продвинутым» замену суммы экспонент из (35), можно записать

$$\Phi_{bb}(f)=D_b T+\mu_b{}^2\sum_{i\in\mathbb{Z}}\delta(f-i/T)\ . \qquad (50)$$

Из формулы (50) следует, что спектр последовательности некоррелированных чисел состоит из двух слагаемых: первое возникает из-за их некоррелированности и говорит о присутствии всех частот, а второе — из-за ненулевого среднего значения, что говорит о паразитной корреляции по постоянной составляющей. Такие игольчатые выбросы при передаче чисел-символов вредны (лишняя трата энергии), поэтому проектировщики систем связи делают все возможное, чтобы снизить среднее значение потока чисел.

Б) Рассмотрим последовательность коррелированных чисел I_n, где корреляция вызвана преобразованием некоррелированных чисел b_n с нулевым средним и дисперсией D_b

$$I_n=b_n+b_{n-1}\ ,\quad \mu_b=M[b_n]=0\ ,\quad D_b=M\left[(b_n-\mu_b)^2\right]\ . \qquad (51)$$

Тогда функция корреляции чисел (51) будет равна

$$\begin{aligned}\phi_{ii}(i)&=M[I_n I_{n-i}]=M\left[(b_n+b_{n-1})(b_{n-i}+b_{n-i-1})\right]=\\&=M[b_n b_{n-i}+b_n b_{n-i-1}+b_{n-1}b_{n-i}+b_{n-1}b_{n-i-1}]=\\&=\begin{cases}2D_b, & i=0;\\ D_b, & i=\pm1;\\ 0, & |i|>1.\end{cases}\end{aligned}\ . \qquad (52)$$

Три слагаемых в (46) дадут искомый спектр мощности

$$\Phi_{ii}(f)=2D_b T\left[1+\cos(2\pi f T)\right]\ . \qquad (53)$$

Функция корреляции — четная, поэтому у каждой экспоненты найдется комплексно-сопряженная пара, значит спектр мощности — это неотрицательная вещественная функция, что, в общем-то, следует и из квадрата модуля. Площадь фигуры, ограниченной кривой спектра и осью частот за период, определяет полную среднюю мощность сигнала; в данном случае она равна $2D_b$, что говорит о повышении мощности в два раза после преобразования (51), так как площадь фигуры для последовательности b_n равна D_b.

Введение корреляции (51) погасит частоты, кратные нечетным $1/2T$, а введение корреляции типа $I_n=b_n-b_{n-1}$ погасит частоты кратные $1/T$, в том

числе и нулевую частоту. Вычисление разности — это, по сути, дифференцирование сигнала, которое закономерно удаляет постоянную составляющую.

Введение более сложных корреляций позволяет сформировать спектр, состоящий из большого количества косинусных гармоник кратных частот. Это можно использовать для формирования более компактного спектра, так же как это делается при выборе окна в дискретном преобразовании Фурье.

2.3 Спектральная плотность мощности цифрового сигнала

Цифровой сигнал с линейной модуляцией может быть записан как* [1]

$$v(t)=\sum_n I_n g(t-nT) \quad , \tag{54}$$

где T — длительность импульсного носителя $g(t)$;

I_n — последовательность чисел, несущих информацию.

**Принято считать, что если в сумме не указаны пределы суммирования, то индекс пробегает весь диапазон целых чисел.*

Вычислим функцию корреляции случайного процесса (54) [1]

$$\phi_{vv}(t+\tau;t)=M\left[v(t)v(t+\tau)\right]=\sum_n\sum_m M\left[I_n I_m\right]g(t-nT)\,g(t+\tau-mT) \quad . \tag{55}$$

Так как в (55) индекс суммирования m пробегает все целые числа, то заменим его на $m+n$ и предположим, что последовательность I_n стационарна по отношению к функции корреляции

$$M\left[I_n I_m\right]=M\left[I_n I_{n+m}\right]=\phi_{ii}(m) \quad . \tag{56}$$

Тогда подставив (56) в (55) и поменяв суммы местами, получим

$$\phi_{vv}(t+\tau;t)=\sum_m \phi_{ii}(m)\sum_n g(t-nT)\,g(t+\tau-nT-mT) \quad . \tag{57}$$

Из (57) следует, что функция корреляции процесса $v(t)$ — периодическая функция времени, так как индекс n в сумме пробегает весь ряд целых чисел и сколько раз T ни прибавляй к аргументу $g(t)$, суммы не изменишь.

Обозначая среднее значение последовательности I_n как $\mu_i=M[I_n]$, найдем среднее значение исходного сигнала

$$M[v(t)]=\mu_i\sum_n g(t-nT) \quad ,$$

которое также является периодической функцией. Значит процесс $v(t)$ является *периодически стационарным* по отношению к функции корреляции. Чтобы избавиться от периодичности, функцию можно усреднить за период [1]

$$\phi_{vv}(\tau)=\frac{1}{T}\int_{-T/2}^{T/2}\phi_{vv}(t+\tau;t)\,dt= \\ =\sum_m \phi_{ii}(m)\sum_n \frac{1}{T}\int_{-T/2-nT}^{T/2-nT} g(t)g(t+\tau-mT)\,dt \quad . \tag{58}$$

Интеграл в (58) вместе с суммой по n является функцией корреляции сигнала $g(t)$

$$\phi_{gg}(\tau)=\frac{1}{T}\int_{-\infty}^{\infty} g(t)g(t+\tau)\,dt \quad , \tag{59}$$

Поэтому подстановка (59) в (58) дает

$$\phi_{vv}(\tau)=\sum_m \phi_{ii}(m)\phi_{gg}(\tau-mT) \quad . \tag{60}$$

Преобразование Фурье (60) дает спектральную плотность мощности процесса $v(t)$

$$\Phi_{vv}(f)=\frac{1}{T^2}|G(f)|^2\Phi_{ii}(f) \quad , Вт/Гц, \tag{61}$$

где $G(f)$ — преобразование Фурье импульса $g(t)$;

$\Phi_{ii}(f)=T\sum_m \phi_{ii}(m)e^{-2\pi jfmT}$ — спектр мощности последовательности I_n .

Из выражения (61) следует, что на спектр сигнала с линейной модуляцией влияет спектр импульса-носителя, а также корреляционные (спектральные) характеристики информационной последовательности. Это является важным фактом при понимании разных кодов, используемых на самом нижнем физическом уровне, таких как RZ, NRZ, NRZ-I, **Манчестер,** MLT-3 и тому подобных. Эти коды, вкупе с логическими кодами (4B/5B, 8B/10B, 8B/6T...), позволяют сформировать желательное для данной среды передачи распределение мощности по частотному диапазону, но и не только это.

3. Описание лабораторного макета

Макет (рис. 33) представляет собой исполняемый файл для операционных систем семейства **Windows.** Код написан на C++ с использованием открытых библиотек **Qt** [2], что позволяет собрать бинарный файл под операционные системы с ядром **Linux.**

На панели **Параметры импульса-носителя** задаются:

- Форма носителя: прямоугольная, треугольная, косинус на пьедестале и Манчестер;
- Длительность импульса-носителя (в процентах от длительности бита, равной 1 с);
- Количество импульсов (битов);
- Амплитуда U+ бита **1** (положительный уровень) и U− бита **0** (отрицательный уровень).

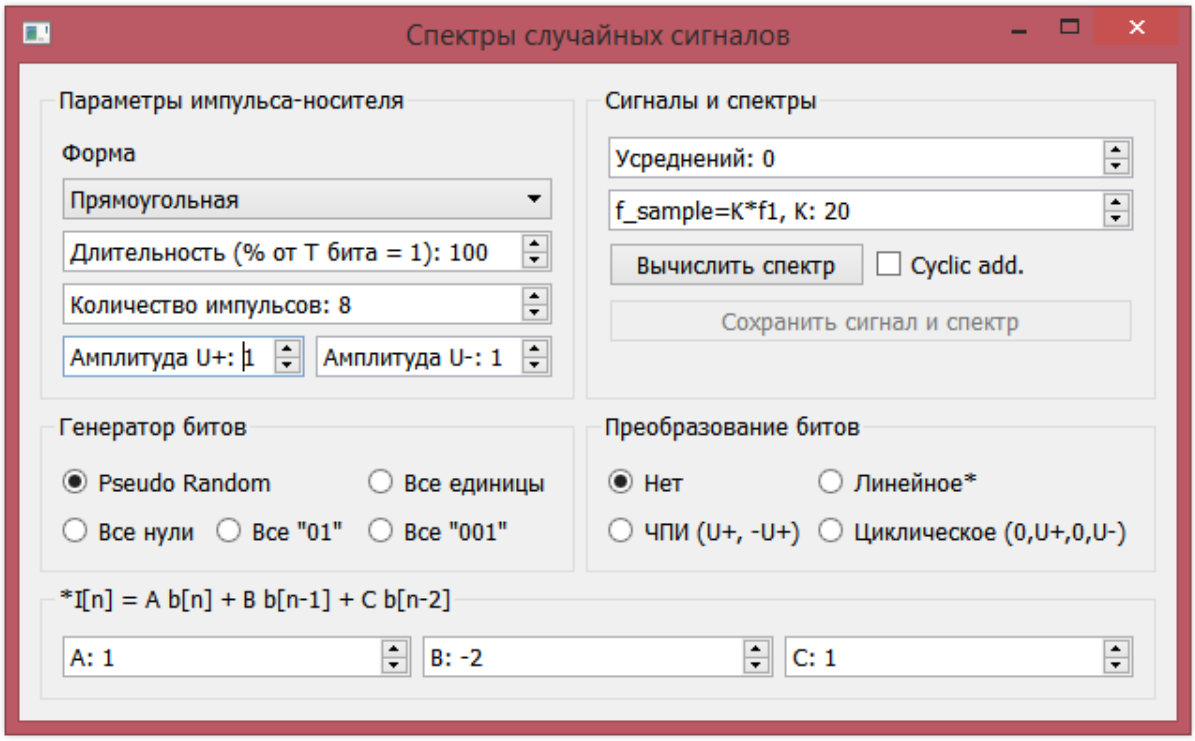


Рисунок 33: Главное окно лабораторного макета "Спектры случайных сигналов"

Панель **Генератор битов** определяет шаблон генерации битов. Например, **Pseudo Random** означает выдачу равновероятных независимых битов. Остальные кнопки не нуждаются в комментариях и задают неслучайный сигнал.

Панель **Преобразование битов** задает способ вычисления «новой» последовательности чисел по «старой».

Пункт **Линейное** означает преобразование

$$I_n = A\,b_n + B\,b_{n-1} + C\,b_{n-2} \quad , \qquad (62)$$

параметры которого задаются на панели ниже. При вычислении (62) считается, что биты периодически продолжены (период — количество импульсов).

Пункт **ЧПИ (U+,−U+)** задает кодирование «Чередование полярности импульса»: если передается бит **1,** то текущая полярность изменяется на противоположную, если **0,** то уровень равен нулю. Первый бит передается уровнями U+ или 0 для битов **1** и **0** соответственно.

Последняя кнопка **Циклическое…** означает кодирование кольцом уровней (0,U+,0,U−): при единичном бите уровень изменяется на следующий

по кольцу, при нулевом — повторяет предыдущий. Первый бит передается уровнями U+ и U− для **1** и **0** соответственно.

На панели **Сигналы и спектры** задаются:

- Число усреднений квадрата модуля спектра; ноль означает отсутствие усреднения;
- Число точек на длительность импульса.

Нажатие на кнопку **Вычислить спектр** открывает окно со спектром плотности мощности (дБВт/Гц). Децибелы считаются как $10\lg\left[\Phi(f_k)\right]$, где $f_k=\frac{k}{N\,T_d}$ — дискретная частота, $N\,T_d$ — длительность всего сигнала, состоящего из *N* отсчетов. Спектральная плотность мощности сигнала g_i, измеряемого в вольтах, вычисляется через дискретное преобразование Фурье

$$\Phi(f_k)=\frac{T_d}{N}|c_k|^2 \quad , \quad c_k=\sum_{0\le i<N} g_i e^{-j\frac{2\pi ik}{N}}$$ (мощность на 1 Ом).

Ось частот пронумерована в герцах. Спектр пересчитывается только при нажатии кнопки **Вычислить спектр;** если при этом активна кнопка **Pseudo Random,** то обновляется также и сигнал.

Флажок **Cyclic add.** включает режим добавления спектров на график для удобства их сравнения. Максимальное количество добавляемых кривых равно трем. Номер добавляемой кривой при каждом нажатии кнопки **Вычислить спектр** меняется циклически по кольцу (1, 2, 3).

Кнопка **Сохранить сигнал и спектр** позволяет сохранить текущие отсчеты сигнала и спектра в текстовые файлы **signal** и **spectr** в формате

$$[t_i \quad s(t_i)] \quad \text{и} \quad [f_i \quad \Phi(f_i)] \ .$$

Это удобно использовать при построении графиков в сторонней программе **(Excel, Mathcad, gnuplot,...)** для отправки их в отчет. Для этого в папке с программой-макетом лежит файл **reader.mcd.**

4. Порядок выполнения работы

1. Вычислить аналитически спектр Фурье и его квадрат модуля для треугольного импульса длительностью T и амплитудой A. Сравнить скорости убывания энергетических спектров в зависимости от частоты $1/f^k$ для трех сигналов: прямоугольный, треугольный и косинус на пьедестале.
2. Запустить программу **spm.** Выставить 10000 усреднений, 16 импульсов. Остальные параметры не менять. Нажать **Вычислить спектр.** Сохранить графики сигнала и спектра в файлы **signal** и **spectr,** осмысленно переименовав их.
3. Подобрать аналитически амплитуды **треугольного** и **косинусного** импульсов так, чтобы площади, ограниченные этими импульсами и осью времени, совпали с площадью прямоугольного импульса. Выставить найденные амплитуды и последовательно построить с помощью программы два сигнала и два спектра, сохранив их в файлы (не забыв переименовать).
4. Извлечь из сохраненных файлов спектры и построить их на одном графике в сторонней программе. Провести на спектрах линии убывания мощности в зависимости от частоты $1/f^k$ (асимптоты). Частоты удобно откладывать в логарифмическом масштабе.
5. На основании (43) и (44) (или только (45)) определить зависимость спектральной плотности **энергии** на нулевой частоте $\Phi(0)$ от площади фигуры, ограниченной сигналом $g(t)$ и осью времени.
6. Показать на построенных спектрах то, что площади трех импульсов равны. Экспериментально определить зависимость спектральной плотности **мощности** при $f=0$ от длительности прямоугольного импульса, взяв, например, $T=T_{бит}$ (100%), $T=T_{бит}/2$ (50%), $T=T_{бит}/4$ (25%).

$$\Phi(0)\ [\text{дБВт/Гц}] = A \lg \frac{T}{T_{\text{бит}}} + B \ , \quad A = ? \ , \quad B = ? \ .$$

7. Вычислить спектральную плотность мощности на нулевой частоте $\Phi(0)$ для импульса типа **Манчестер,** у которого половина импульса амплитудой U+, другая половина — минус U−, так что U+ = U−. Построить спектр с помощью программы (16 импульсов, 10000 усреднений, амплитуды по единице). Показать на спектре то, что данный код обладает хорошей самосинхронизацией.
8. Аналитически вычислить спектр мощности для преобразования чисел $I_n = Ab_n + Bb_{n-1} + Cb_{n-2}$, подставив для $A = C = 1$, $B = -2$. Числа b_n обладают теми же корреляционными свойствами, что и в (51). Построить два спектра — с преобразованием и без него — и проверить правильность вычисленного преобразования, взяв прямоугольный импульс-носитель. Проверку сделать ниже вставки графика в отчет, прокомментировав рисунок (видим то, видим это…).
9. Выбрать преобразование **Циклическое…** и построить три спектра (для всех импульсов, кроме **Манчестер**). На одном графике сравнить пары типа <Спектр с циклическим преобразованием; Спектр без него>. Прокомментировать графики.
10. Измерить частоту основной (первой) гармоники циклического преобразования (импульс брать прямоугольный), отметив для этого **Все единицы.** При измерении частоты пользоваться лупой (**zoom**). Гармонику измерять в долях $1/T$. Получить точный результат графически, нарисовав от руки основную (первую) гармонику **на графике сигнала.**
11. Выбрать преобразование **ЧПИ...,** импульс прямоугольный, 16 равновероятных битов (амплитуда U− автоматически установится на ноль). Построить **одну реализацию сигнала** и спектр (10000 усреднений). Графически доказать отсутствие постоянной составляющей для шаб-

лонов **Все единицы, Все 01, Все 101, Все 100,** нарисовав вручную соответствующие сигналы (два-три периода).

12. Будут ли эквивалентными шаблоны **Все 110, Все 011 и Все 101?** Сколько разных шаблонов длиной 4 бита можно набрать, так, чтобы они не переходили в эквивалентные шаблоны? Перечислите эти шаблоны.

5. Вопросы

1. Почему истинно случайная последовательность прямоугольных импульсов амплитудой $+1$ и -1 имеет постоянную составляющую?
2. Почему последовательность из предыдущего вопроса не имеет частот кратных $1/T$, где T — длительность импульсов.
3. Принадлежит ли истинно случайной последовательности отрезок из 10 единиц? Из 100? Из 1000?
4. Какой отрезок не принадлежит истинно случайной последовательности?
5. Как в зависимости от числа слагаемых частичного ряда Фурье растет крутизна меандра в точках перепада его уровня?

6. Литература

1. Прокис Джон. Цифровая связь.
2. Qt | Cross-platform application & UI development framework, http://www.qt.io/.

Для заметок

ДЕЛЬТА-МОДУЛЯЦИЯ

1. Введение

Дельта-модуляция является одной из многих техник приближенного представления аналогового сигнала последовательностью импульсов, и по праву относится к цифровой связи. «Цифра» проявляется в том, что каждому импульсу сопоставляется символ, а импульсы следуют друг за другом через равные интервалы времени — **такты.** Потребность такого представления возникла при попытке передачи речевых сигналов на дальние расстояния, когда сигнал, проходя усилитель за усилителем, постепенно теряется в шумах.

Количество символов в алфавите при дельта-модуляции равно двум, значит такое устройство должно уметь генерировать два различных по форме импульса. Дельта-модуляция по своей сути является **однобитовым аналого-цифровым преобразователем** (АЦП), однако в чистом виде она, как правило, не используется из-за присущих ей недостатков.

Импульсно-кодовая модуляция (ИКМ), появившаяся раньше дельта-модуляции, является другой техникой представления аналогового сигнала последовательностью импульсов. Она отличается от дельта-модуляции тем, что рядом стоящие импульсы объединяются в группы, которым сопоставляются кодовые слова (многобитовый АЦП), и каждый бит в слове даёт разный вклад в восстанавливаемый в приемнике аналоговый сигнал. Однако ИКМ и дельта-модуляция отличаются не только количеством битов на отсчет сигнала.

Дело в том, что для многих физически регистрируемых сигналов их основная мощность сосредоточена в относительно узкой частотной области, и эти сигналы оцифровываются с частотой, выбираемой по теореме отсчетов с некоторым запасом. Такая избыточная дискретизация дает гарантию того, что рядом стоящие отсчеты сигнала с большой вероятностью будут отличаться незначительно, что наталкивает на мысль о кодировании не самих отсчетов, а их разностей. Суть дельта-модуляции состоит в кодировании не самой

разности, а **лишь ее знака.** Логика **больше-меньше** естественным образом отображается в двоичный алфавит **{1, 0}.**

В данной работе изучается система связи с дельта-модуляцией, состоящая из генератора сигналов, модулятора, линии передачи, демодулятора и осциллографа-вольтметра.

Структура отчета должна соответствовать следующему:

- ✔ Титульный лист;
- ✔ Ход работы;
- ✔ Ответы на вопросы;
- ✔ Выводы.

Принимаются форматы файлов **doc, docx, odt** и **pdf.**

2. Сведения из теории

Дельта-модуляция [1] была изобретена в 40-х годах XX века независимо во Франции (Derjavitch B., Deloraine E.M., Van Mierlo S., 1946 г), США (de Jager F., 1947 г) и СССР (инженер Коробков Л.А., 1948 г). Она явилась развитием импульсно-кодовой модуляции (ИКМ), запатентованной английским исследователем Alec Reeves в 1938 г, и предложившим ее как результат исследований на тему уменьшения накопления шумов при передаче речевых сигналов на дальние расстояния [2]. В результате внедрения ИКМ появился **регенератор** — устройство, восстанавливающее битовый поток из «грязных» входных импульсов, и заново генерирующее «чистые» выходные, что проблему накопления шумов в каскаде усилителей переводит в такие возникающие битовые ошибки, которые не так страшны, как накопление шумов.

Несмотря на достоинства ИКМ перед аналоговой передачей, технические устройства ИКМ были громоздкими вплоть до появления транзисторов и в дальнейшем интегральных микросхем. Технически дельта-модулятор и демодулятор намного проще устройств ИКМ, правда для обеспечения равного качества передачи сигнала для системы связи с дельта-модуляцией требуется более высокая частота взятия отсчетов, а значит и полоса пропускания. По сути, все проблемы аналоговой связи, в той или иной степени устраненные дельта-модуляцией (накопление шумов, громоздкость техники ИКМ, влияние импульсных помех), вылились в требование более широкой полосы пропускания канала.

На основе дельта-модуляции была разработана **дельта-сигма модуляция,** требующая при равных условиях меньшей полосы пропускания канала и меньшего количества функциональных блоков. Однако данная работа является учебной, поэтому придется смириться с изучением детской основы.

Простейший дельта-модулятор можно составить из тактового генератора (**генератор тактовых импульсов,** ГТИ), компаратора и интегратора (рис. 34).

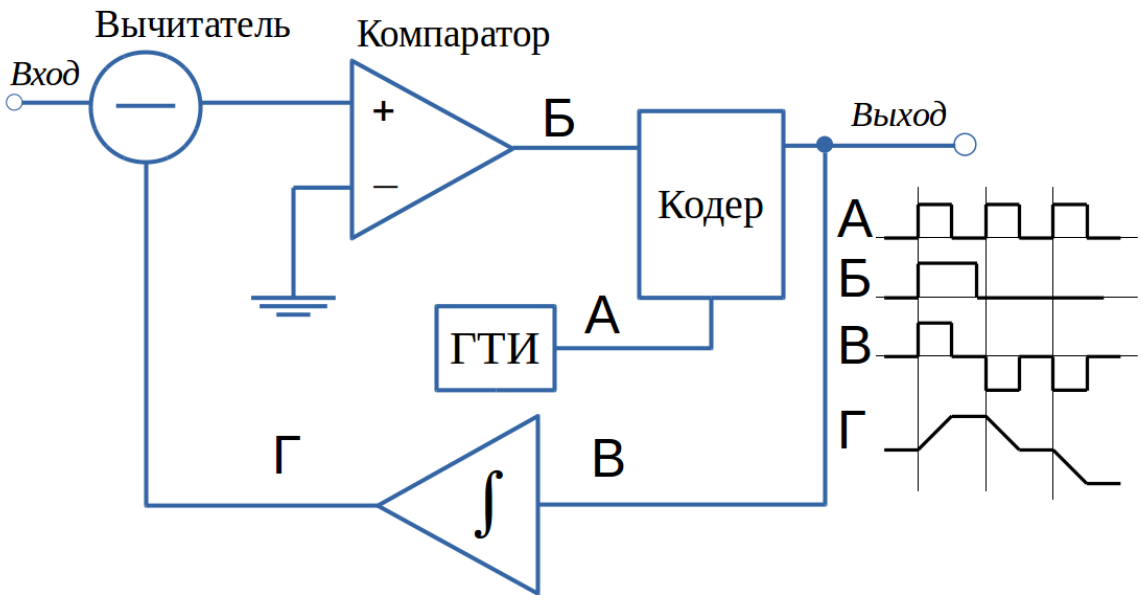


Рисунок 34: Функциональная схема простейшего дельта-модулятора

Практически же данная схема содержит **кодер** — преобразователь логических униполярных уровней **Б** в сигнальные биполярные **В.** В те моменты времени, когда на выходе ГТИ есть импульс, кодер выдает или $+U$, или $-U$ в зависимости от уровня напряжения на выходе компаратора. В моменты пауз (когда на выходе ГТИ нулевой уровень) кодер выдает нулевое напряжение.

Так называемый **вычитатель** строится на **операционном усилителе,** равно как интегратор и компаратор. Тактовый генератор выдает меандр **А,** в котором наличие импульса означает активную половину такта, а отсутствие — пассивную (пауза).

Анализ схемы на рис. 34 можно начать с предположения, что напряжение на входе модулятора постоянно (0,5 В) и больше напряжения на выходе интегратора (0,3 В). Тогда выход вычитателя будет положительным 0,5−0,3 = 0,2 В, что даст логическую единицу на выходе компаратора. В активную половину такта эта единица появится на выходе кодера в виде положительного импульса и одновременно пойдет на вход интегратора. Такое

напряжение вызовет положительный линейный рост напряжения на выходе интегратора на некоторую фиксированную величину Δu, называемую шагом квантования модулятора. Во время паузы напряжение на выходе идеального интегратора не меняется (в реальном интеграторе немного спадает).

Допустим, шаг квантования равен 0,15 В, тогда текущая разность будет равна 0,5−0,45 = 0,05 В, что меньше предыдущей. Активная половина следующего такта приведет к росту напряжения на выходе интегратора с 0,45 В до 0,6 В, что даст отрицательную разность и дальнейшее спадание до 0,45 вольт. В итоге мы можем убедиться, что при постоянном входном напряжении 0,5 В и шаге интегратора 0,15 В на выходе модулятора будет чередующаяся последовательность нулей и единиц.

На самом деле чередование будет при любом шаге интегратора, превышающем порог чувствительности компаратора. В итоге любой постоянный уровень напряжения кодируется битами **...0101010...,** что говорит о невозможности передачи абсолютного значения напряжения и о том, что дельта-модулятор стремится передать лишь форму сигнала — его переменную составляющую. Такая ситуация в связи встречается сплошь и рядом: в аналоговом телевидении невозможно передать уровень черного, из-за чего применяют схемы его восстановления; все проводные линии связи, где используются разделительные трансформаторы, в принципе не могут передать постоянную составляющую сигнала; речевые и музыкальные сигналы по своей природе ее не имеют (а ухо человека слышит до частоты 16 Гц).

Демодулятор строится по еще более простой схеме (рис. 35), в которую входят лишь аналогичный интегратор и фильтр нижних частот (ФНЧ).

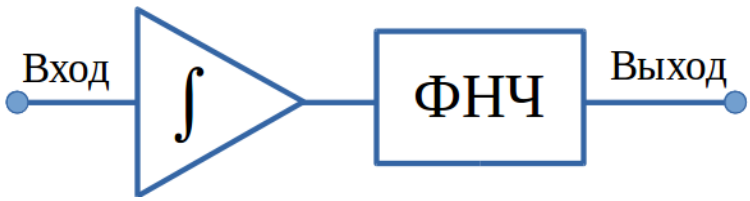


Рисунок 35: Функциональная схема простейшего демодулятора

ФНЧ предназначен для сглаживания изломов напряжения с выхода интегратора. Полоса пропускания ФНЧ подбирается исходя из ширины спектра полезного сигнала, поступающего на вход модулятора. Для упрощения схемотехники ФНЧ его, как правило, делают пропускающим постоянную составляющую, которую дельта-модулятор все равно не способен передать.

Чтобы наглядно показать процесс модуляции-демодуляции, ниже приведены несколько осциллограмм при кодировании синусоидального сигнала с частотой 40 кГц и амплитудой 1,5 В.

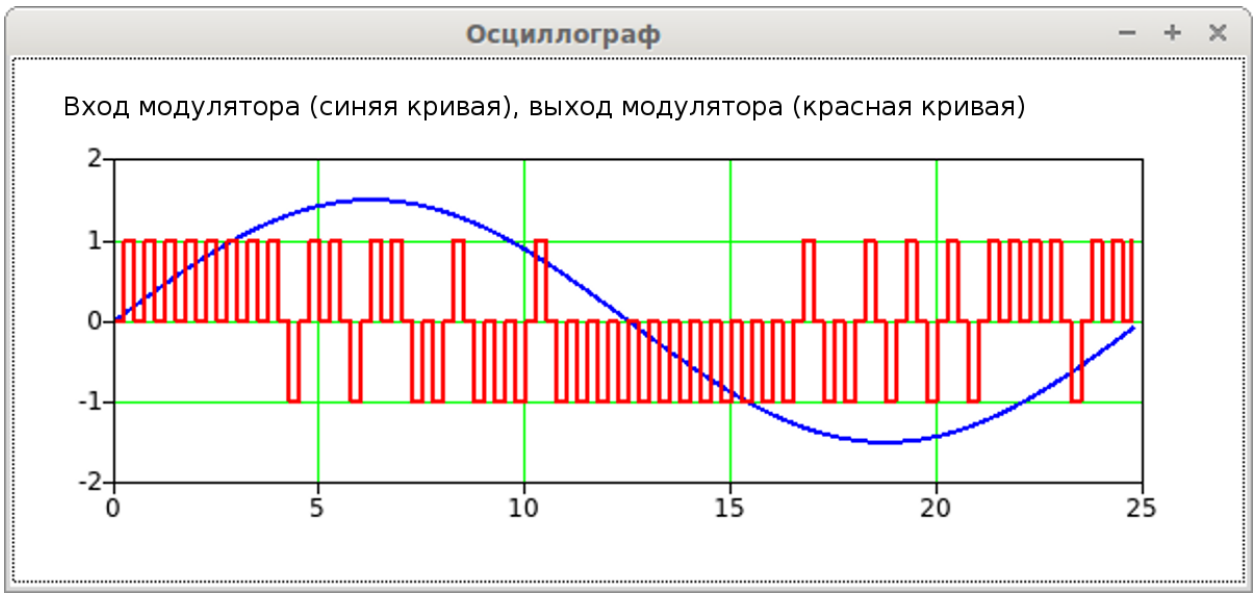
Единицы деления для всех осциллограмм: 5 мкс, 1 В.

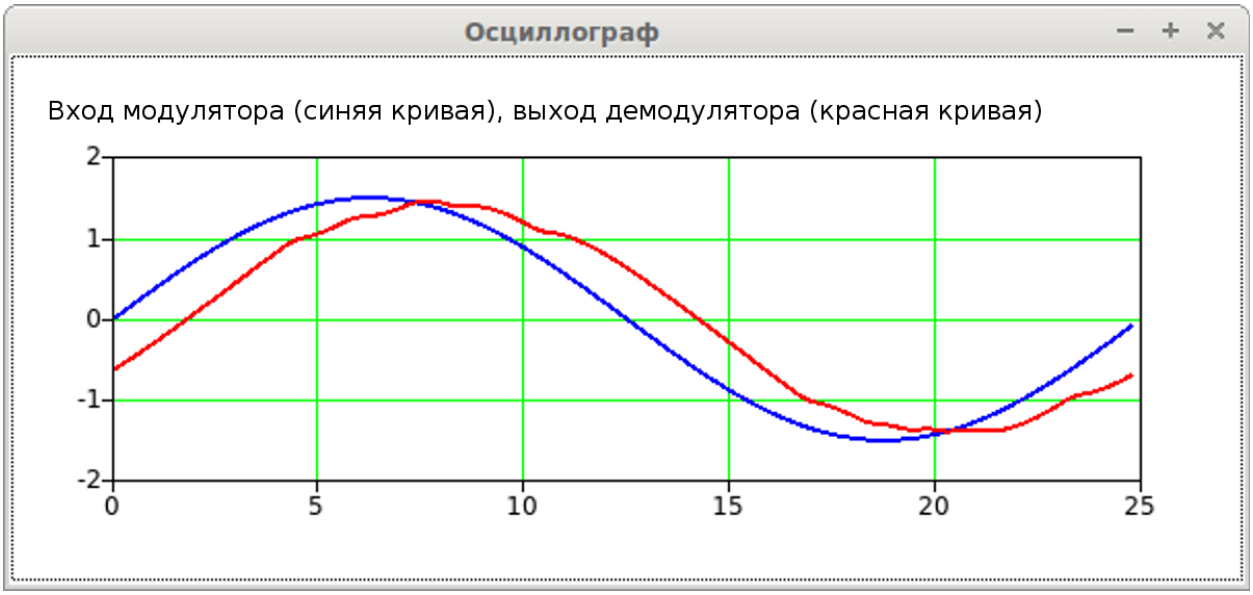
Из последней осциллограммы **(выход демодулятора)** следует, что восстановленный сигнал несколько задержан относительно исходного. При моделировании никаких задержек не вносилось, поэтому задержка произошла за счет фазо-частотной характеристики ФНЧ.

Осциллограммы измерены в установившемся режиме работы модулятора и демодулятора.

Выход модулятора кодируется тремя уровнями (0, ±1) В, что удобно для интеграторов, реагирующих на положительные и отрицательные скачки напряжения и не реагирующих на нулевой уровень.


Осциллограф
Вход модулятора (синяя кривая), выход интегратора модулятора (красная кривая)
2
1
0
-1
-2
0
5
10
15
20
25


Осциллограф
Вход модулятора (синяя кривая), выход модулятора (красная кривая)
2
1
0
-1
-2
0
5
10
15
20
25


Осциллограф
Вход модулятора (синяя кривая), выход демодулятора (красная кривая)
2
1
0
-1
-2
0
5
10
15
20
25

3. Описание лабораторного макета

Макет представляет собой исполняемый файл для операционных систем семейства **Windows.** Код написан на C++ с использованием открытых библиотек **Qt** [3], что позволяет собрать бинарный файл в операционной системе с ядром **Linux.**

После запуска файла **delta_mod** появляется окно (рис. 36).

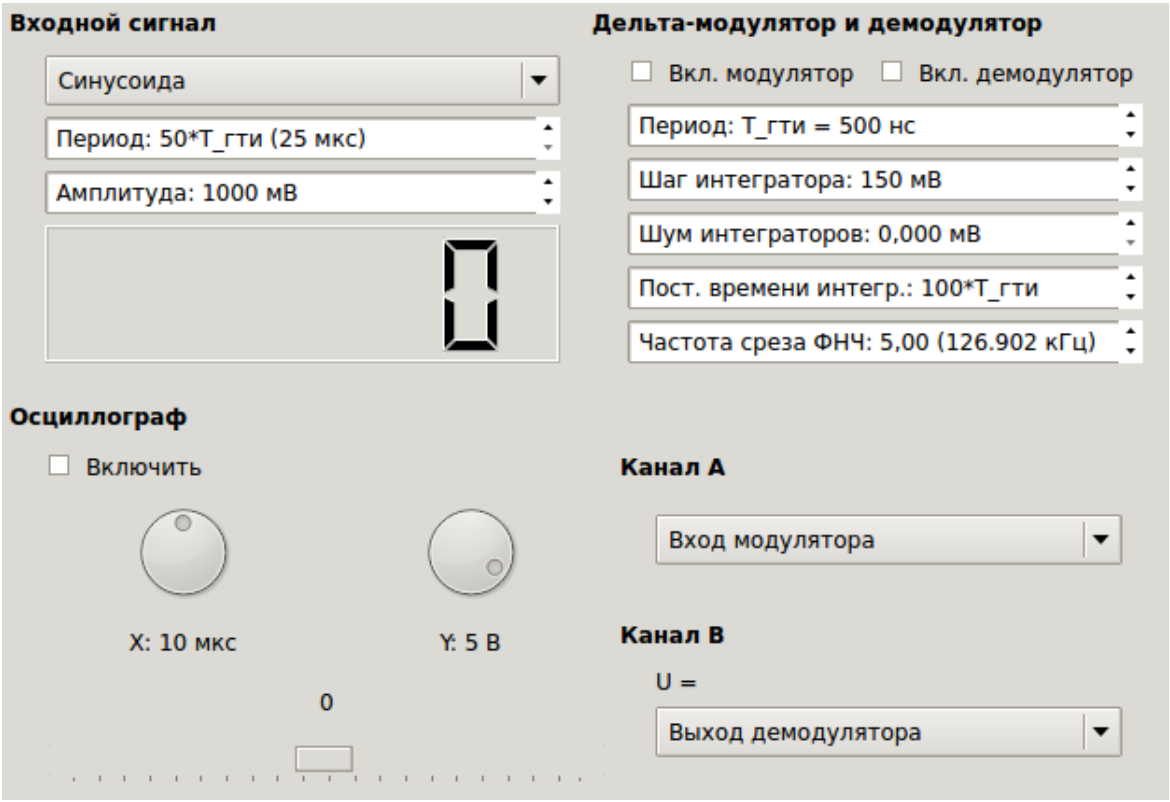


Рисунок 36: Главное окно лабораторного макета «Дельта-модуляция»

В главном окне задаются:

- Параметры и тип входного сигнала;
- Параметры модулятора-демодулятора;
- Параметры осциллографа.

Поддерживаются следующие типы входного сигнала:

- Синусоида;
- Треугольник;
- Пила;
- Меандр.

Для входного сигнала задается амплитуда и период, который выбирается кратным периоду тактового генератора.

Параметрами модулятора-демодулятора являются:

- Период $T_{\text{гти}}$ импульсов тактового генератора;
- Шаг квантования Δe интегратора;
- Уровень шума U_{noise} и постоянная времени интегратора;
- Частота среза ФНЧ f_c .

Шум интегратора моделируется случайной величиной, равномерно распределенной в диапазоне от $-U_{\text{noise}}$ до $+U_{\text{noise}}$.

ФНЧ первого порядка (RC-цепочка) моделируется методом билинейного z-преобразования [4]. Частота среза ФНЧ f_c по уровню 3 дБ задается коэффициентом *k*, равным отношению постоянной времени τ аналогового ФНЧ-прототипа к периоду следования отсчетов $T_s = 1/f_s$ [20]

$$f_c = \frac{1}{\pi T_s} \arctan\left(\frac{T_s}{2\tau}\right) = \frac{1}{\pi T_s} \arctan\left(\frac{1}{2k}\right) \text{ , Гц.} \qquad (63)$$

Модулятор и демодулятор можно по отдельности включать и выключать флажками **Вкл. модулятор** и **Вкл. демодулятор.**

Осциллограф содержит:

- Ручки для выбора единиц деления по осям **X** и **Y;**
- Гнезда для подключения контрольных точек схемы к каналам **A** и **B;**
- Бегунок типа **slider** для смещении осциллограмм вправо-влево по оси **X** (фаза фазовой синхронизации).

В канале **B** автоматически измеряется уровень действующего напряжения за период сигнала (корень из суммы квадратов отсчетов, деленной на их количество).

К каналам **A** и **B** осциллографа можно подключить:

- Вход или выход модулятора;

---

20 Шаг дискретизации T_s выбран так, что на период импульсов ГТИ укладывается два отсчета

- Вход или выход демодулятора;
- Выход интегратора модулятора или демодулятора.

Семисегментный индикатор в группе **Входной сигнал** показывает «схемное» время (в миллисекундах), прошедшее с момента запуска осциллографа. Во время включения осциллографа запускается таймер с периодом 40 мс (25 Гц), по звонку которого производится последовательный расчет всех напряжений в модуляторе и демодуляторе для всего отрезка времени, отображаемого на осциллографе.

Развертка по оси **X** рассчитана на **пять** делений. Например, если единица деления по оси **X** равна 5 мкс, то осциллограф будет принимать блок данных для отрезка в 5·5 = 25 мкс через каждые 40 мс (по срабатыванию таймера). В этом случае реальное время будет больше схемного в $40 \cdot 10^{-3} / 25 \cdot 10^{-6} = 1600$ раз! При этом показание индикатора увеличится на 1 мс через 1,6 с.

Получается, что с помощью данного макета можно «тормозить время» и наблюдать медленные процессы. Расчет и графическое отображение всей этой «катавасии» в реальном времени средствами центрального процессора практически затруднен (можно сказать «невозможен» или «финансово неоправдан»).

4. Порядок выполнения работы

Аналитическая часть §§5–6 — z-преобразование и вычеты — отнимает немало времени, поэтому её рекомендуется сделать дома заранее.

1. Запустить макет. Включить модулятор и демодулятор. Включить осциллограф. Входной сигнал выставить на **Выключить.** В канал **B** осциллографа подсоединить **Выход интегратора модулятора.** Развертку по **X** выставить на 2 мкс/дел., развертку по **Y** — на 0,2 В/дел.
2. Зарисовать (сделать скриншот) осциллограмму на выходе интегратора модулятора и на входе модулятора. Указать стрелками с выносками период следования тактовых импульсов и величину шага квантования по

уровню (отредактировать скриншот). Сравнить с установленными программно (визуально убедиться в правильности работы схемы).

3. Вычислить действующее значение напряжения на выходе интегратора аналитически (за период сигнала, равный двум тактам ГТИ), считая что форма этого напряжения — трапеция (рис. 37). Сравнить результат вычисления с показаниями вольтметра канала **B.** Объяснить различие, зная что вольтметр цифровой и измеряет лишь два раза за такт в моменты времени $(n/2)T_{\text{гти}}$. Нарисовать форму непрерывного сигнала, которая даст действующее значение напряжения, совпадающее с показанием цифрового вольтметра (соединить жирные точки на рис. 37).

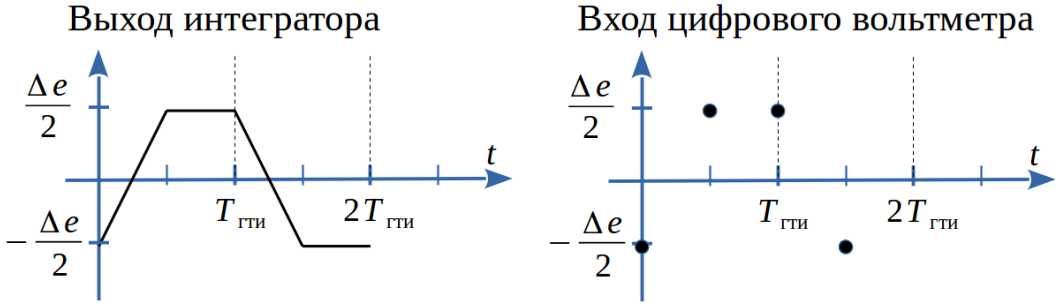


Рисунок 37: К измерению действующего значения напряжения цифровым вольтметром

4. Переключить канал **A** на **Выход модулятора.** Развертку по **Y** выставить на удобную (1 В/дел.). Сделать скриншот осциллограммы и словесно в отчете пояснить процесс интегрирования знакочередующихся прямоугольных импульсов.
5. Переключить канал **A** на **Выход интегратора демодулятора,** а канал **B** на **Выход демодулятора.** Развертку по **Y** выставить на 100 мВ/дел. Вычислить частоту взятия отсчетов f_s , записать из программы частоту среза ФНЧ f_c . Зная, что при цифровом моделировании отсчеты на выходе интегратора идут в виде периодической последовательности

$$\frac{\Delta e}{2}(1,1,-1,-1,1,1,-1,-1,\ldots) \ ,$$

определить ее z-образ $I(z)$.

6. Зная системную функцию цифрового ФНЧ первого порядка [4]

$K(z)=\dfrac{1+z^{-1}}{1+2k+(1-2k)z^{-1}}$, где по (63) $k=\tau/T_s$,

определить все значения напряжения на выходе цифрового ФНЧ в установившемся режиме (их будет четыре*). Сравнить с результатом измерения по осциллографу. Установившийся режим определять как предел h_n при $n\to\infty$, где h_n — отсчеты на выходе ФНЧ, найденные через обратное z-преобразование от произведения $K(z)I(z)$. Для выполнения данного задания рекомендуется использовать программы символьных вычислений **(Mathcad, SymPy, ...).** Ответ доказать программой или на отдельном листочке-приложении к отчету.

***Ответ:** $h_\infty=\pm\dfrac{\Delta e}{2}\dfrac{2k\pm1}{4k^2+1}$.

7. Развертку по **X** выставить на 5 мкс/дел., а по **Y** — на 0,5 В/дел. Подать на вход меандр амплитудой 1 В. Канал **A** выставить на **Вход модулятора,** а канал **B** — на **Выход демодулятора.** Сделать скриншот осциллограммы и по ней оценить наибольшую крутизну (В/мкс) сигнала с выхода демодулятора. Сравнить ее с расчетной. Влияет ли на крутизну сигнала наличие ФНЧ? Для ответа на этот вопрос посмотреть осциллограммы выходного сигнала для разных частот среза ФНЧ.
8. Переключить входной сигнал на **Синусоида,** развертку по **Y** на 1 В/дел., а развёртку по **X** — на 20 мкс/дел. Частоту синусоиды выставить на 10 кГц, а частоту среза ФНЧ — на 31,50 (20,2085 кГц). Снять с помощью вольтметра амплитудно-частотную характеристику системы связи, уменьшая период синусоиды с шагом 10 мкс до предела. Чему теоретически должно быть равно действующее значение напряжения, если частота синусоиды равна частоте среза ФНЧ, а ее амплитуда — 1 В? Какова максимально допустимая частота синусоиды, не допускающая перегрузку по скорости? Расчет проверить наблюдением осциллограмм (выставить расчетную частоту, плавно ее увеличить и удостовериться в начале перегрузки).

9. Выключить входной сигнал, шаг интегратора уменьшить до 20 мВ, развертка по **X** — 10 мкс/дел., по **Y** — 50 мВ/дел. Канал **A** настроить на **Вход модулятора,** канал **B** — на **Выход интегратора демодулятора.** Увеличивая уровень шума интеграторов U_{noise} от нуля до 40 мВ с шагом 2 мВ, измерить действующее напряжение. Построить график.
10. Выставить частоту среза ФНЧ на 10,00 (63,609 кГц). Сделать предыдущий пункт, но канал **B** настроить на **Выход демодулятора.**
11. Наблюдая шумовой сигнал на выходе интегратора при уровне шума 30 мВ и шаге интегратора 20 мВ, визуально оценить интервал напряжений, за пределы которого шум практически не выходит. Учесть, что шум интеграторов — равномерно распределенная в диапазоне от $-U_{\text{noise}}$ до $+U_{\text{noise}}$ случайная величина. Аналитически оценить интервал действия шума на выходе интегратора по правилу **трех сигм,** считая, что общая дисперсия шума равна сумме дисперсий неслучайного шума (чередование уровней $-\Delta e/2$ и $\Delta e/2$) и случайного шума.

5. Вопросы

1. Что такое перегрузка по скорости при дельта-модуляции? Перегрузка по амплитуде при использовании реального интегратора?
2. Можно ли с помощью дельта-модуляции передать постоянную составляющую?
3. Как зависит от частоты среза ФНЧ максимальное значение шума молчания* на выходе ФНЧ, если постоянная времени τ ФНЧ много больше периода следования тактовых импульсов $T_{\text{гти}}$?

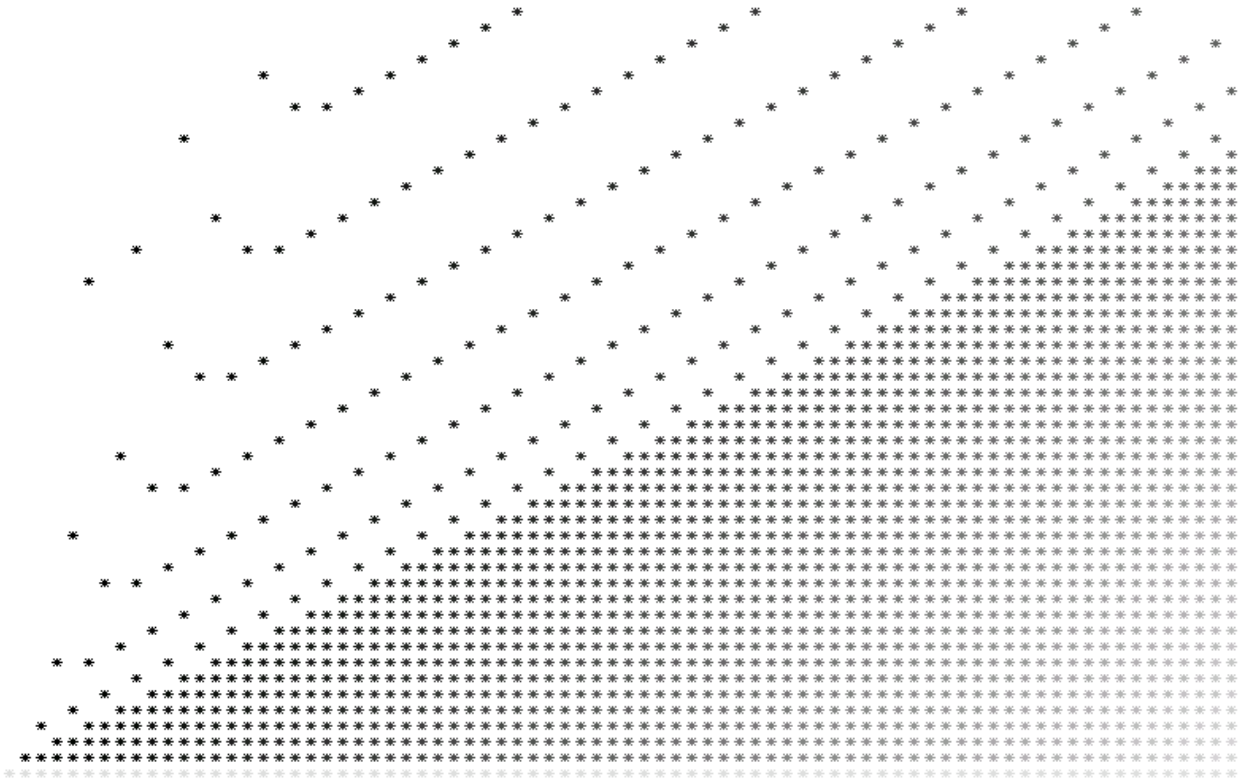
   *Режим молчания — это когда на входе модулятора нулевое напряжение.
4. Что можно сказать о влиянии случайного шума интеграторов на выходное напряжение демодулятора? Какова при этом роль ФНЧ?

6. Литература

1. Raymond Steele. Принципы дельта-модуляции. Пер. с англ./ Под ред. В.В. Маркова. М.: Связь, 1979. — 368 с.
2. Alec H. Reeves, The past, present and future of PCM, http://tkhf.adaxas.net/cd1/Reeves2.pdf
3. Qt | Cross-platform application & UI development framework. http://www.qt.io/
4. Каратаева Н.А. Радиотехнические цепи и сигналы. Часть 2 Дискретная обработка сигналов и цифровая фильтрация: Учебное пособие. https://edu.tusur.ru/training/publications/2799

Для заметок

МЕТОДИКА АНАЛОГО-ЦИФРОВОГО ПРЕОБРАЗОВАНИЯ



Наука начинается с тех пор, как начинают измерять. Точная наука немыслима без меры.

Д.И. Менделеев.

1. Введение

Аналого-цифровое преобразование (АЦП) — это измерительная процедура, которая несмотря на свое современное и технологичное название является результатом практической потребности человека в **счете.**

Испокон веков счет яблок, грибов, шкур животных и других предметов начинается с **ноля.** По этой причине числа 0, 1, 2, 3, … называются натуральными (они определяют количество **цельных** предметов). Эти числа можно назвать и **атомными,** потому что атом по определению не делится. Дальше будет видно, что в процедуре АЦП операция деления отсутствует[21].

Яблоки бывают разными по весу, форме, вкусу и тому подобным параметрам, но одно яблоко оно и есть одно яблоко (предмет) — это основа аналого-цифрового преобразования. Результат измерения — это набор чисел, причем всегда можно обойтись только лишь натуральными числами (номерами).

Естественно, что число само по себе — это абстрактный объект, который не может быть напрямую записан на физический носитель, поэтому любое число кодируется с помощью сигналов (электрических, оптических, механических и других).

Например, цифра **1** может быть нарисована разными способами: разными шрифтами, разными цветами и так далее, но для абстрактного восприятия единица останется единицей. Правда при этом не стоит абстрагироваться абсолютно: ведь **1** означает не просто «единицу в вакууме», а единицу измерения, которой соответствует, например, определенный электрический ток.

---

21 Деление как таковое в природе отсутствует. Клетки, например, не делятся, а множатся!

В данной лабораторной работе изучаются две методики (АЦП): *поразрядного взвешивания* и *последовательного счета;* причем упор делается на логику измерений, а не на схемотехнику.

Две только что названные методики АЦП восходят к древним практикам взвешиваний грузов и измерений длин, несмотря на их современную микроминиатюрную реализацию в виде полупроводниковых схем.

Структура отчета должна соответствовать следующему:

- ✔ Титульный лист;
- ✔ Ход работы;
- ✔ Ответы на вопросы;
- ✔ Выводы.

Принимаются форматы файлов **doc, docx, odt** и **pdf.**

2. Сведения из теории

2.1 АЦП последовательного счета

К процедуре последовательного счета приводит задача измерения длины некоторого отрезка. Здесь возникают два вопроса: **в чем измерять** и **как измерять.**

Ответ на первый вопрос дает такое понятие как *единица измерения,* которая задается изначально.

Чем точнее требуется результат, тем меньшей должна быть эта единица и тем медленнее будет идти процесс измерения. Ошибка измерения никогда не превысит выбранной единицы.

Ответ на второй вопрос (как измерять) приводит к понятию *метода измерения.*

Рассмотрим метод *последовательного счета.*

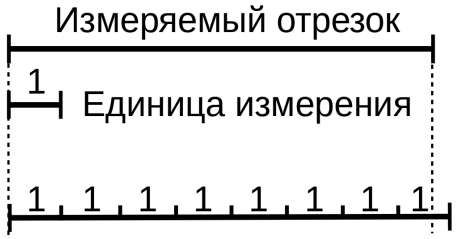


Рисунок 38. Принцип последовательного счета

Если имеется некоторый отрезок, длину которого необходимо измерить, и выбрана единица измерения, то процедуру последовательного счета можно изобразить в виде рис. 38. Счет здесь идет слева направо, а результат измерения шаг за шагом выдается в виде набора из **одних** единиц 1111…1.

В рассматриваемом примере результатом измерения будет набор из восьми единиц 11111111, так как на восьмом такте длина суммарного отрезка стала больше длины измеряемого. В зависимости от соглашения, можно оставить семь единиц 1111111, при этом ошибка измерения все равно не превысит единицы измерения (допускается округление в большую или меньшую сторону).

Таким образом, даже при простейшем измерении необходимо сравнивающее устройство — *компаратор* (от англ. *compare* — сравнение).

Компаратор (рис. 1) каждый измерительный такт должен выдавать **один из двух** сигналов: либо **больше,** либо **меньше.** В логических схемах этим сигналам соответствуют логические уровни, условно обозначаемые как **0** и **1.** Теоретически, если сигналы на входе компаратора окажутся равными, потребуется сигнал **равно.** Практически же равенства никогда не бывает, поэтому в те моменты времени, когда два сигнала близки друг к другу с заданной точностью, компаратор остается в предыдущем состоянии (*гистерезис* компаратора).

Результат измерения выдается постепенно неделимыми порциями. Под порцией понимается один из двух импульсов с выхода компаратора.

Таким образом делаем вывод: методу последовательного счета соответствует некоторый способ записи чисел, то есть *система счисления.*

Другая запись чисел (система счисления) — другой способ измерения, и наоборот.

$N = \underbrace{1\ 1\ 1\ \ldots\ 1}_{N\text{ раз}}$ Методу последовательного счета соответствует представление натурального числа N по Евклиду. По сути, это сумма из N единиц.

Чем точнее мы хотим измерить некоторый отрезок, тем меньшей должна быть единица измерения и тем больше цифр потребуется для записи результата, что, в общем-то, естественно и является платой за точность. Если длительность измерительного такта является постоянной, то более точные измерения будут проходить дольше менее точных.

Ясно, что последовательный счет очень затратный по времени. Это выражается в большом количестве единиц при записи натурального числа. Представьте, что требуется записать число 141 данным способом... Потребуется нарисовать сто сорок одну единицу и потратить на это минуты времени, вместо каких-то единиц секунд для записи всего лишь трех цифр 141!

Спрашивается, а раз есть более компактная десятичная система счисления, то нельзя ли на её основе разработать другой метод измерения? Ответ: можно, но технически сложнее, ведь для этого надо уметь различить десять

уровней, поставив набор компараторов с десятиуровневым выходным сигналом.

Ситуация становится технически проще, если ограничиться двоичными системами счисления. В рассмотренном примере последовательного счета использовалась именно двоичная система счисления, но **непозиционная,** где каждая единица имеет единичный вес. Концу измерения при последовательном счёте соответствуют нули, которые нет смысла рисовать, так как они никогда не заканчиваются и не несут информации.

Получается, что метод последовательного счета правильнее сравнивать с теми методами, которым соответствуют системы счисления, использующие код с основанием два, то есть с алфавитом из двух символов $\{0,1\}$.

Мы знаем как минимум две таких системы: позиционную двоичную систему счисления, когда вес следующего разряда удваивается, и непозиционную, когда все веса равнозначны. Если последняя система очень некомпактна, то спрашивается, а двоичная максимально ли компактна? Если да, то есть ли между ними промежуточные системы счисления, также использующие *бинарный* алфавит?..

Все эти вопросы относятся к АЦП *поразрядного взвешивания.*

2.2 АЦП поразрядного взвешивания

Рассмотрим двоичную систему счисления, когда вес каждого разряда равен степеням двойки

$$N = a_{M-1}2^{M-1} + a_{M-2}2^{M-2} + \ldots + a_1 2^1 + a_0 2^0 \ , \qquad (64)$$

где a_i — числа 0 или 1, соответствующие символам двоичного алфавита **0** и **1** соответственно;

M — число разрядов (*разрядность* АЦП).

Как уже было отмечено, результат работы любого АЦП можно представить в виде натуральных чисел (и ноль там же!), поэтому сознательно не рассматриваются отрицательные степени двоек (считайте, что они неконструктивные).

Оказывается, что процедура (64) **при ограничении двоичной логикой** позволяет представить наибольшее количество чисел при фиксированной разрядности *M*. Это количество (всё множество) равно 2^M . **Троичная логика** с взвешивающими числами $\{-1,0,+1\}$ даст множество из 3^M чисел и так далее.

Идя от обратного, можно утверждать, что двоичный АЦП при фиксированном количестве кодов (натуральных чисел от 0 до *N*) будет обладать минимальной разрядностью $M=\lfloor \log_2 N \rfloor$, где знаком $\lfloor . \rfloor$ обозначено округление «вниз» (*floor* — от англ. *пол*).

Как известно, две крайности — последовательный счет и поразрядное взвешивание — это не совсем хорошо; или, хотя бы, это наталкивает на мысль о возможном существовании промежуточных вариантов.

«Нехорошесть» последовательного счета — большое число знаков, что приводит к длительным измерениям, зато пропуски (сбои) единиц не так страшны, так как нам важно знать только начало кодового слова и конец (количество тактов), а что между ними — и так ясно, что должны быть все единицы. Значит «хорошесть» этого метода заключается в повышенной помехоустойчивости.

«Плохость» двоичного поразрядного взвешивания — минимальное число знаков, что приводит к **неразличимости любых случайных сбоев.** Поясним последний факт особо.

Пусть, например, имеется двоичное АЦП с тремя разрядами, тогда каждому кодовому слову будут соответствовать цифры от 0 до 7 включительно. Если на вход подать уровень **5,** то в результате случайного сбоя в триггере старшего разряда АЦП выдаст код **0**01, а не **1**01, и вместо истинного значения **5** мы получим значение **1**, что может привести к печальным последствиям.

Для двоичной системы счисления все кодовые слова являются разрешенными: в этом и заключается её «плохость».

Рассмотренный факт хорошо известен как компромисс между скоростью передачи и надежностью сообщения. Поэтому после обычного (с нулевой избыточностью) АЦП используют разные корректирующие коды.

Оказывается, имеются процедуры двоичного **АЦП с уже встроенной избыточностью,** степень которой может регулироваться [1, 2]. В этом случае дополнительные корректирующие коды могут и не потребоваться.

Но для начала давайте докажем, что обычное двоичное АЦП (64) дает минимальное количество знаков $\lfloor \log_2 N \rfloor$ при заданном диапазоне чисел от 0 до *N*. Для этого необходимо вычислить все веса так называемых **гирь.**

Первая гиря — единица, так как это минимальное ненулевое число. С помощью этой гири можно взвешивать веса от 0 до 1 с точностью 1.

Вес второй гири должен быть выбран как можно большим, но так, чтобы уравновесить вес два, являющийся минимальным весом, который первая гиря не уравновесит. Отсюда следует, что вес второй гири — это два.

Третья гиря выбирается по тому же принципу: как можно большей по весу, но так, чтобы не потерять суммарный вес всех предыдущих гирь плюс один, то есть число $(1+2)+1=4$. Значит вес третьей гири — это четыре.

Четвертая гиря — сумма весов всех предыдущих плюс один, то есть $(1+2+4)+1=8$, и так далее, что приводит к степеням двойки.

Таким образом, двоичная система счисления дает кодирование максимального диапазона натуральных чисел при фиксированной разрядности, или обеспечивает минимальное количество разрядов при заданном диапазоне натуральных чисел (включая ноль). Такую систему счисления (или такой способ АЦП) выгодно использовать в каналах без помех, где отсутствуют случайные сбои.

Казалось бы, последовательный счет — вот способ для каналов с помехами! Да, но он, как правило, очень избыточный, потому что, например, тратить на цифровой звук качества **Audio CD** около 65536 импульсов-битов при шестнадцатибитовом кодировании затратно: в этом случае частота дис-

кретизации должна увеличиться с 44,1 кГц до 180 МГц!.. А ведь это всего лишь для звука с полосой частот не более 20 кГц… Это и есть следствие двух крайностей: **все или ничего.**

В настоящее время используются корректирующие коды, которые дополняют двоичное поразрядное АЦП, но это, как правило, относится к системам связи и носителям информации (жесткие диски, **SSD** диски, **flash**-память). А если взять, допустим, центральный процессор персонального компьютера?.. Вот он-то никак не контролирует собственные сбои, так как если у него есть слово регистра из 32 бит, то он и располагает этими 32 битами и все слова, в принципе, одинаково возможны... В случае сбоя компьютер просто-напросто зависает и ему нужен аппаратный сброс. А если произойдет сбой в процессоре спутника или ракетоносителя?..

Городить в архитектуру процессора корректирующие коды — дурной тон, поэтому, видимо, в процессорах этого и нет, да и в модулях оперативной памяти — для профессионалов. Гораздо выгоднее использовать АЦП с естественной избыточностью, которая может использоваться для обнаружения сбоев на лету без значительного снижения эффективности работы процессора.

Если контролирующая схема обнаружит сбой при выполнении некоторой инструкции, то она заставит процессор повторить её еще раз. При действительно случайном сбое, а не отказе, вторая попытка с большой вероятностью должна пройти без ошибок.

В процедуре поразрядного взвешивания есть один малозаметный параметр, в котором и кроется возможность **постепенного** перехода от нольизбыточного АЦП поразрядного взвешивания к АЦП последовательного счета. Этим параметром является *асимметрия процесса измерения* [1, 2].

Чтобы понять асимметрию процедуры взвешивания, представим ее механическую аналогию — весы с набором гирь.

Допустим, у нас есть пять гирь весом **1**, **2**, **4**, **8** и **16.** На левую чашу весов поставили вес **11.** С какой гири мы должны начать взвешивание? Верно, с максимальной по весу.

Ставим на правую чашу вес **16** и наш индикатор (компаратор) показывает **много** (бит **0**), поэтому мы должны убрать эту гирю, **затратив на это дополнительное время,** и поставить следующую. Теперь на правой чаше вес **8,** индикатор говорит **мало** (бит **1**), значит убирать ничего не надо и **дополнительное время не требуется.** Это и ведет к *асимметрии процесса взвешивания* [1].

Далее ставим **4 — много** (бит **0**); убираем **4.** Ставим **2 — мало** (бит **1**); ставим **1** — чётко (бит **1**).

АЦП — цифровое устройство, которое работает по тактам, поэтому его инерционность (время, за которое убирается гиря), в принципе, может быть любой, но кратной такту. Эталонные веса (гирьки) подставляются формирующим устройством также по тактам (… идет своеобразный конвейер из гирь ...), поэтому при убирании гири, фактически, мы будем пропускать несколько следующих гирь («поезд ушел»). Сколько? Зависит от инерционности: пропустим одну, если инерционность один такт, и так далее, а так как в обычной двоичной системе никакие гири пропускать нельзя (иначе не все числа будут представлены с точностью в одну единицу), то такую систему можно применять только в безынерционных АЦП, коими электронные и являются, так как длительность убирания гири там много меньше одного такта.

Если инерционность равна нулю, то оптимальной (максимально компактной по разрядности) процедурой АЦП с двоичной логикой (типа **много-мало**) является та, которая выражается в обычной двоичной системе счисления. Если инерционность неограниченно большая, то снятие гири при сигнале **много** фактически соответствует пропуску всех гирь и останову процесса измерения, что полностью соответствует технике последовательного счета **мало-мало-мало- … - много.**

Оказывается, что если взять инерционность снятия гири равной одному такту, то оптимальной системой гирь будет следующий ряд чисел [1, 2]

$$1,1,2,3,5,8,13,21\ldots\ , \qquad (65)$$

который известен как последовательность чисел Фибоначчи. Покажем это.

Вес первой гири, понятное дело, равен единице (без неё никуда).

Попробуем взять вторую гирю весом два, подобно двоичным гирям. Тогда **не обеспечится** измерение числа **1,** так как положив на правую чашу гирю весом **2**, получим перебор (компаратор скажет **много**). Уберем эту гирю, **затрачивая на это один такт.** За это время уйдет первая гиря и процесс закончится, так как гири всего две и измеритель обязан затратить не более двух тактов. Если бы измерялось число **3,** то двух тактов как раз бы хватило: **2 — мало,** убирать не надо, поэтому следующая гиря не убежит, что даст искомую сумму $3=2+1$.

Получается, что вторую гирю также надо взять с единичным весом (других вариантов нет). Этими гирями обеспечивается взвешивание чисел от **0** до **2.**

Третья гиря дает три шага (такта) на измерение. Попробуем взять вес третьей гири равный трем. Тогда не обеспечится измерение числа **2,** так как за три такта $3,\bar{3},1=1$ (знак $\bar{3}$ означает — убрать вес три) пропустим вторую гирю **1** (при убирании веса **3**) и не успеем уравнять **2.** А зачем тогда третья гиря, если две весом **1** и **1** прекрасно обеспечивают измерение веса **2?** Поэтому остается взять гирю весом **2.** Проверим, сколько чисел уравновесит система из трех гирь $1,1,2$.

Число **1:** $2,\bar{2},1$, число **2:** 2 , число **3:** $2,1$, число **4:** $2,1,1$. На этом все, так как сумма весов трех гирь равна четырем. Число **0** также уравновесится не более чем за три такта, потому что на третьем такте $2,\bar{2},1$ компаратор выдаст **много,** откуда ясно следует **0.**

Если числа от **0** до **4** кодировать двоичным кодом с весами **1, 1** и **2,** то можно заметить, что появляется неоднозначность (табл. 1).

Эта неоднозначность (избыточность) и является следствием ограничения на процесс измерения, так как если становятся возможными пропуски некоторых гирь, то должен быть своеобразный «запас» на гири.

Табл. 1. Избыточность двоичной системы счисления для набора весов $2,1,1$

Число	Двоичный код
0	000
1	001 или 010
2	100 или 011
3	110 или 101
4	111

Любая подстраховка — это и есть избыточность, которой можно воспользоваться **для обнаружения и исправления некоторых сбоев,** так как асимметрией процесса измерения в электронных АЦП можно пренебречь и нет **острой** необходимости в избыточном коде. Если бы инерционность была сравнима с длительностью такта (или больше), то тогда просто невозможно было бы использовать классический двоичный код и обеспечить при этом точность в одну единицу для всего диапазона измеряемых чисел.

Продолжим искать вес четвертой гири.

Возьмём гирю весом **4** и перебором отыщем промах. Он найден при измерении числа **3,** так как $4,\bar{4},1,1=2\neq 3$ (гиря весом **2** прошла мимо, пока убирали вес **4**). Остается взять вес **3.** Проверяем и его: $0:3,\bar{3},1,\bar{1}=0$, $1:3,\bar{3},1=1$, $2:3,\bar{3},1,1=2$, $3:3=3$, $4:3,2,\bar{2},1=4$, $5:3,2=5$, $6:3,2,1=6$, $7:3,2,1,1=7$.

Внимание! Обобщим результат.

Есть система гирь $1,1,2,3$. Какой вес w_k требуется добавить, чтобы отсутствовали промахи, то есть чтобы был однотактный запас на гири? Система $1,1,2,3$ обеспечивает непрерывный ряд чисел от **0** до **7** без промахов.

Если измеряется число, меньшее, чем вес гири w_k, но большее, чем вес предыдущей гири w_{k-1}, то пропускается гиря с весом w_{k-1} и требу-

ется её заменить предыдущими гирями. Максимальное измеряемое число, которое приведет к пропуску w_{k-1}, равно w_k-1, и для его уравновешивания потребуем, чтобы сумма всех оставшихся гирь была равна этому числу (вот он ключ!)

$$w_1+w_2+\ldots+w_{k-2}=w_k-1 \quad .$$

Из последнего равенства выведем рекуррентную формулу, связывающую искомые веса. Задавая последовательно $k=4,5,6\ldots$ получим цепочку равенств

$$w_1+w_2=w_4-1 \quad ,$$

$$w_1+w_2+w_3=w_5-1=w_3+w_4-1 \quad ,$$

$$w_1+w_2+w_3+w_4=w_6-1=w_4+w_5-1 \quad ,$$

$$\ldots$$

откуда следует одно равенство

$$w_k=w_{k-1}+w_{k-2} \quad .$$

Тогда пятая гиря должна весить $2+3=5$, что дает ряд Фибоначчи.

Доказательство оптимальности ряда Фибоначчи для АЦП с инерционностью в один такт закончено. Члены этого ряда с ростом номера растут не так быстро как соответствующие степени двойки.

По сути-то, степенная функция 2^n и является истинной функцией умножения, так как она определяется умножением двойки *n* раз на саму себя. А числа Фибоначчи получаются путем суммирования натуральных чисел.

Показано, что если взять инерционность *p* тактов, то оптимальным рядом гирь будут *p*-числа Фибоначчи

$$\begin{gathered} w_1=w_2=\ldots=w_{p+1}=1 \\ w_k=w_{k-1}+w_{k-1-p},\ k>p+1, \end{gathered} \qquad (66)$$

что, однако, выходит за рамки данной работы, но заинтересовавшийся читатель может узнать об этом (и не только) из первоисточника [1].

Приведем пример (табл. 2) двоичного кодирования чисел по системе из пяти гирь $5,3,2,1,1$. Максимальное кодируемое число — $5+3+2+1+1=12$.

Табл. 2. Двоичный код по системе весов 5, 3, 2, 1, 1.

Число	Двоичный код
0	**00000**
1	**00001**, 00010
2	**00100**, 00011
3	**01000**, 00110, 00101
4	**01010**, 01001, 00111
5	**10000**, 01100, 01011
6	**10010**, 10001, 01110, 01101
7	**10011**, 10100, 01111
8	**11000**, 10110, 10101
9	**11001**, 10111, 11010
10	**11100**, 11011
11	**11110**, 11101
12	**11111**

Жирным шрифтом в табл. 2 выделены кодовые слова, получающиеся в результате рассмотренной процедуры инерционного на один такт АЦП.

2.3 Способ обнаружения и исправления сбоев

Обнаружение сбоя возможно за счет равенства

$$w_k = w_{k-1} + w_{k-2} \ , \qquad (67)$$

связывающего веса гирь. Это равенство ведет к неоднозначности кодовых слов, так как две рядом стоящие единицы могут быть заменены на одну, старшую по весу, поэтому кодовые слова $0\mathbf{11}$ и $\mathbf{1}00$ будут эквивалентными.

Это связывающее равенство справедливо для всех разрядов, если продлить ряд Фибоначчи влево $w_0=0, w_{-1}=1, w_{-2}=-1, w_{-3}=2\ldots$, поэтому путем (67) любое кодовое слово можно свести к такому, у которого не будет

двух рядом стоящих единиц. Это называется приведением кода к *минимальной форме* [1].

Обособленность единиц можно использовать как **контрольный признак.**

Так как асимметрией измерения в электронных АЦП можно пренебречь, то рассмотрим **обычную** процедуру поразрядного взвешивания по системе весов из 1-чисел Фибоначчи (то есть p-чисел при $p=1$).

Возьмем АЦП из десяти разрядов с весами

$$55,\ 34,\ 21,\ 13,\ 8,\ 5,\ 3,\ 2,\ 1,\ 1\ .$$

Измерение начинается со старшего разряда.

Допустим, что на вход АЦП подан уровень **81.** Управляющее устройство включит старший триггер в единицу, чтобы подключить эталонный вес **55,** и если триггер не даст сбой типа **невключение,** то результат сравнения числа **81** с весом **55** будет положительным и соответствующий триггер останется в единице. Дальше управляющее устройство включит второй по старшинству триггер, чтобы сравнить **55**+**34** с числом **81.** Естественно, что результат сравнения будет отрицательным и компаратор выключит триггер, отвечающий за вес **34,** если, конечно, триггер не даст сбой типа **невыключение.**

Таким образом, корректный (при отсутствии сбоев) десятибитовый код числа **81** равен 1010010000 .

Всё вроде бы ничего, однако если посмотреть на процесс сравнения числа **55** с весом **55**, то за счет избыточности АЦП возможны два варианта развития событий:

- Выбрать бит **1** и обнулить оставшиеся младшие разряды;
- Выбрать бит **0**, но выставить две следующие обязательные единицы и обнулить остальные; но, по идее, последняя единица также может «превратиться» в две последующие единицы и этот процесс может продолжаться до конца разрядной сетки.

Чтобы понять многозначность процесса оцифровки числа, совпадающего с весом, рассмотрим возможные варианты кодовых слов для входного числа **55** (рис. 39).

За счет равенства (67) возможно включение в **1** не текущего триггера, а двух следующих. Это может произойти по причине того, что в реальных АЦП идеального равенства входной величины и эталонного веса не бывает: всегда бывает чуть больше или чуть меньше, и именно эти «чуть-чуть» и приведут к одному из двух исходов.

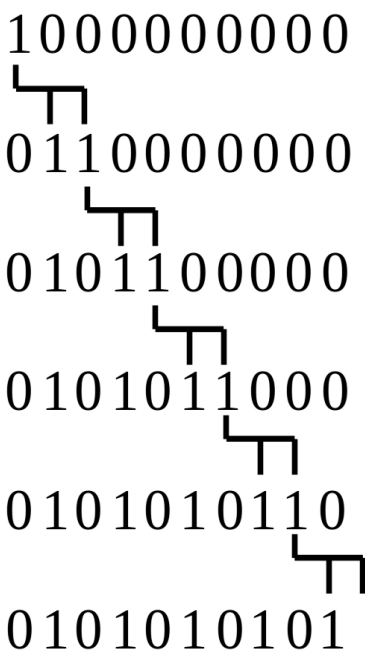


Рисунок 39: Многозначность кодирования числа 55

На выходе рассматриваемого АЦП возможны так называемые *метрологически разрешённые* кодовые слова и *метрологически запрещённые* [2]. Последние возникают при сбое триггеров.

Рассмотрим, например, два типа сбоя в старшем триггере при оцифровке числа **60.**

При сбое **невключение** будет выдано кодовое слово 0110010000 , которое является метрологически запрещенным, так как две рядом стоящие единицы допускается встретить лишь раз — после них обязаны быть нули, а у нас есть одна единица, которая возникла из-за недовеса эталоном **55.**

Хотя полученное кодовое слово и запрещенное, однако фатальной ошибки не произошло — код равен **60,** что говорит о дополнительной исправляющей способности такого АЦП. Естественно, не все сбои могут быть исправлены.

При сбое **невыключение** АЦП выдаст код 1000010000 , который будет метрологически разрешенным и верным по числовому коду, так как при оцифровке **60** старший разряд и так должен остаться включенным.

Для иллюстрации неисправленной ошибки при сбое **невыключение** в старшем триггере, определим код для числа **50:** это будет слово

1000000000 , что соответствует числу **55** и даёт ошибку **5** единиц — сбой неисправлен.

Метрологически разрешенные кодовые слова — это те, которые имеют минимальную форму, при этом допускается встретить две единицы рядом, но лишь раз, после чего идут все нули [2].

Максимально взвешиваемое число для десятиразрядного АЦП — сумма двух старших весов, то есть **89,** так как для **90** требуются слова 1100000010 или 1100000001 , которые являются метрологически запрещенными.

2.4 Вероятностная модель ошибок при АЦП

Выделим для определенности два типа ошибок [2]:

- Ошибка сравнения (случайный сбой триггера);
- Отказ триггера, то есть постоянное его **невключение** (залипание в **0**) или, наоборот, **невыключение** (залипание в **1**).

Считаем, что ошибки сравнения происходят случайно, независимо, с некоторыми вероятностями p_1 и p_0 , где 1 означает выдачу бита **1** вместо бита **0,** а 0 — выдачу бита **0** вместо бита **1.**

Отказ триггера задается принудительно, то есть неслучайно.

3. Описание лабораторного макета

Лабораторный макет — программа, запускаемая с помощью исполняемого файла **adcfib1**. Написана на языке C++ с применением открытых библиотек **Qt** [3], что дает возможность собрать бинарный файл для выполнения в операционной системе с ядром **Linux** (рис. 40).

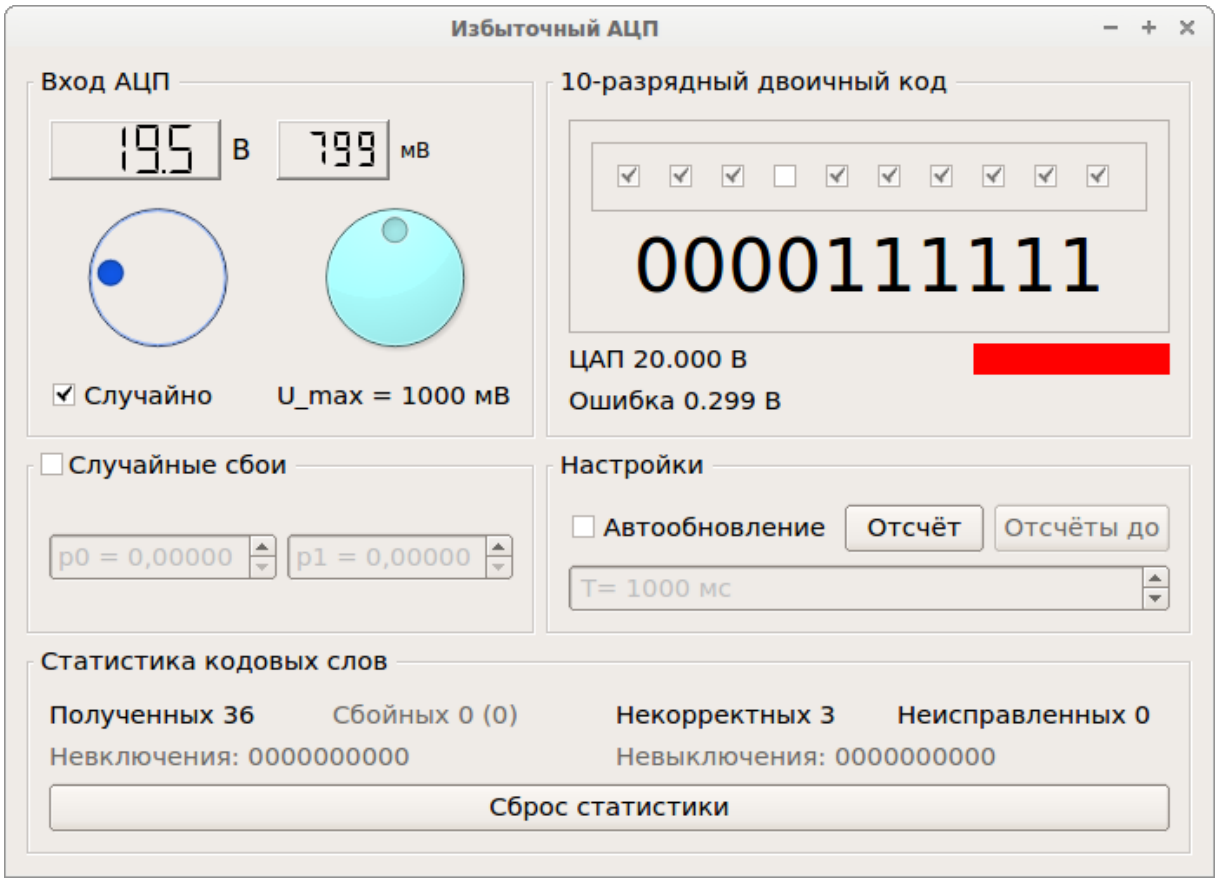


Рисунок 40: Лабораторный макет «Избыточный АЦП»

На вход АЦП поступает напряжение, задаваемое на панели **Вход АЦП** ручкой **Уровень входного напряжения** (смотри всплывающую подсказку по наведению указателя мыши). Для большей реалистичности процесса на вход АЦП подан аддитивный шум, отсчеты которого равномерно распределены на отрезке от нуля до некоторого максимального напряжения, задаваемого ручкой **Уровень U_max равномерно....** Заданное максимальное напряжение показывается в текстовом поле **U_max = … мВ.** Текущее входное напряжение и значение шума показываются на двух соответствующих LCD-индикаторах, в вольтах и милливольтах соответственно. Входное напряжение можно задавать случайным, равномерно распределенным на отрезке [0...89] В, отметив для этого флажок **Случайно.**

На панели **10-разрядный двоичный код** отображается двоичный код входного числа (с шумом), получаемый по методу поразрядного взвешивания по системе весов 55, 34, 21, 13, 8, 5, 3, 2, 1, 1 . Панель с набором из десяти флажков, расположенная над двоичным кодом, предназначена для

принудительной установки (флажок отмечен) или сброса (флажок снят) соответствующих триггеров. Серые флажки (третье состояние) соответствуют «освобождению» триггеров (нет отказа). Ниже двоичного кода показаны:

- Значение напряжения по цифровому коду на выходе идеального ЦАП (ЦАП — цифро-аналоговый преобразователь);
- Ошибка квантования, то есть разность между значением входного напряжения и значением ЦАП.

Прямоугольная цветная область, расположенная чуть ниже двоичного кода, предназначена для визуального контроля метрологически корректных (синий цвет) и некорректных (красный цвет) кодовых слов.

На панели **Случайные сбои** задаются две вероятности случайных сбоев: сбой типа **невключение** p_0 и сбой типа **невыключение** p_1. Вероятность, меньшая чем $1/32767$, считается нулевой.

Панель **Настройки** предназначена для задания ручного взятия отсчетов (кнопка **Отсчет**) и автоматического, если установлен флажок **Автообновление.** Период взятия отсчетов задается в поле **T = … мс**. Кнопка **Отсчёты до** предназначена для автоматического взятия отсчетов до появления первого **случайного** сбоя; это удобно, когда вероятность сбоя мала и требуется на нём остановиться, чтобы проанализировать.

На панели **Статистика кодовых слов** отображаются:

- Общее количество полученных кодовых слов;
- Количество сбойных кодовых слов (только случайные сбои);
- Количество метрологически некорректных кодовых слов (когда загорается красным индикатор);
- Количество неисправленных кодовых слов (когда ошибка АЦП больше или равна единице, или меньше нуля).

Текстовые поля **Невключения 0000000000** и **Невыключения 0000000000** с помощью единиц показывают биты, в которых произошли

случайные сбои соответствующих типов. В текстовом поле **Сбойных 0 (0)** число в скобках означает:

(1) — наличие сбоя,

(0) — отсутствие сбоя,

при последней текущей процедуре АЦП.

4. Порядок выполнения работы

1. Запустить программу. Выставить уровень входного напряжения 0,5 В (половина шага квантования), а уровень шума — 0 мВ (шум выключен). Период взятия отсчетов выставить на 2–3 с. Случайные сбои и неслучайные — выключить. Пронаблюдать 15–20 реализаций. Перечислить кодовые слова, которые получаются на выходе. Не меняя входного напряжения, выставить уровень шума на 1000 мВ (шаг квантования) и повторить наблюдение. В каком случае процедура АЦП проходит точнее? Какова роль шума?
2. Выключить шум. Выбрать случайное входное напряжение на интервале (0–89) В. Для этого установить флажок **Случайно.** Выключить **Автообновление** на любом ненулевом кодовом слове, и привести в отчете пошаговую процедуру поразрядного взвешивания выбранного числа. Сравнить результат ручной работы с программным.
3. Выбрать разряд (триггер), где стоит единица и принудительно выключить его (залепить в нуле). Привести в отчете процедуру взвешивания при залипшем триггере. Приведет ли невключение триггера к фатальной ошибке (то есть когда разность между истинным значением и цифровым будет больше единицы или меньше нуля)? Если не приведет, то почему.
4. Выбрать разряд (триггер), где стоит ноль и принудительно включить его (залепить в единице). Привести в отчете процедуру взвешивания при залипшем триггере. Приведет ли невыключение триггера к фатальной ошибке? Если да, то почему.

5. Включить случайные сбои, выключив неслучайные. Выставить вероятность **невключения** 0,01, вероятность **невыключения** — 0,00. Автообновление убрать. Нажимать кнопку **Отсчеты до** до тех пор, пока не появится метрологически запрещенное кодовое слово (об этом уведомит красный индикатор) плюс чтобы ошибка квантования была в интервале [0…1). Из панели **Статистика кодовых слов** выписать номер бита, где произошло невключение (если их больше одного, что маловероятно, то выписать столько, сколько есть). Привести кодовое слово, которое было бы при отсутствии сбоя (метрологически разрешенное), и кодовое слово при наличии сбоя (из программы). Логически объяснить почему сбой исправлен.
6. Сделать предыдущий пункт, поменяв местами вероятности **невключения** и **невыключения.** Кнопку **Отсчеты до** нажимать до тех пор, пока не получится неисправленный сбой. Выписать номера невыключившихся битов. Объяснить, почему сбой не исправлен.
7. Выставить случайное входное напряжение. Шум выставить на 1000 мВ. Включить случайные сбои, выключить неслучайные залипания триггеров. Период взятия отсчетов выставить минимальным (100 мс). Для ряда вероятностей p0 и p1

   а)

$$\begin{matrix} p0=\{0{,}001;0{,}005;0{,}01;0{,}05;0{,}1;0{,}3;0{,}5\} \\ p1=\{0;0;0;0;0;0;0\} \end{matrix},$$

   б)

$$\begin{matrix} p0=\{0;0;0;0;0;0;0\} \\ p1=\{0{,}001;0{,}005;0{,}01;0{,}05;0{,}1;0{,}3;0{,}5\} \end{matrix},$$

   в)

$$p0=p1=\{0{,}001;0{,}005;0{,}01;0{,}05;0{,}1;0{,}3;0{,}5\},$$

пронаблюдать 1000 реализаций, предварительно сбрасывая статистику. Останов наблюдения делается снятием флажка **Автообновления.** В итоге должно получиться 21 наблюдение. Для каждого из них в таблицу записать

количества сбойных, некорректных, неисправленных кодовых слов и общее количество полученных кодовых слов. Удобно для пунктов а), б) и в) составить три отдельные таблицы. Построить графики зависимостей количества сбойных, некорректных и неисправленных кодовых слов от вероятности сбоя. Удобно сделать три диаграммы, в каждой из которых по три линии (по горизонтальной оси откладываются те вероятности, которые изменялись).

5. Вопросы

1. Какой тип АЦП, при прочих равных условиях, более быстродействующий: поразрядного взвешивания по системе весов 2^i или последовательного счета? Во сколько раз, если оба АЦП используют двоичный код и имеют *n* разрядов?
2. Какой тип АЦП из вопроса №1 не позволяет обнаруживать сбои?
3. Чем мы платим за возможность обнаружения некоторых сбоев?
4. При одинаковом количестве разрядов какое АЦП будет обладать большей точностью: то, которое обнаруживает некоторые сбои, или то, которое не обнаруживает ни одного?
5. Дано двенадцать знаков. Как необходимо распределить эти знаки

$$\begin{pmatrix} x & x & x & x & x & x & x & x & x & x & x & x \end{pmatrix}$$ (12 разрядов),

$$\begin{pmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \end{pmatrix}, \begin{pmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{pmatrix}, \begin{pmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{pmatrix},$$

$$\begin{pmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \end{pmatrix}^T, \begin{pmatrix} x & x & x & x & x & x & x & x & x & x & x & x \end{pmatrix}^T$$ (один разряд),

чтобы поразрядный код дал максимальное количество кодируемых чисел?

6. Чему равно основание m n-разрядной системы счисления, дающей максимальный диапазон кодируемых чисел от 0 до m^n-1, если количество знаков $N=m\cdot n$ фиксировано?

6. Литература

1. А.П. Стахов. Основы математики гармонии и ее приложения, часть 2, http://www.trinitas.ru/rus/doc/0232/100a/02320067.htm
2. А.П. Стахов. Коды золотой пропорции. — 1984.
3. Qt | Cross-platform application & UI development framework, http://www.qt.io/ [Электронный ресурс], режим доступа: открытый, дата обращения: 13.12.2014.

Для заметок