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 КОМПЬЮТЕРНЫХ ЛАБОРАТОРНЫХ РАБОТ ПО СИСТЕМАМ СВЯЗИ

Учебно-методическое пособие для проведения 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

2018

Новиков А.В.
Сборник компьютерных лабораторных работ по системам связи: учебно-методическое пособие для проведения 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, 2018. — 151 с.

Настоящий сборник состоит из семи компьютерных лабораторных работ по системам связи.

В первых двух работах изучаются двоичные циклические коды Хемминга. Подробно разбираются пространственные свойства циклического кода (7, 4), а также принципы и правила кодирования и декодирования с помощью регистров сдвига с линейными обратными связями.

В третьей работе изучаются сверточные коды со скоростью кодирования ½. Рассматриваются три способа кодирования и два способа декодирования: по алгоритму Витерби и пороговое.

Четвертая работа посвящена методике цифрового формирования двоичного частотно-манипулированного (ЧМн) сигнала с непрерывной фазой и методике цифровой некогерентной демодуляции такого сигнала. Предполагается ручная установка фазы тактовых импульсов в цепи тактовой синхронизации демодулятора.

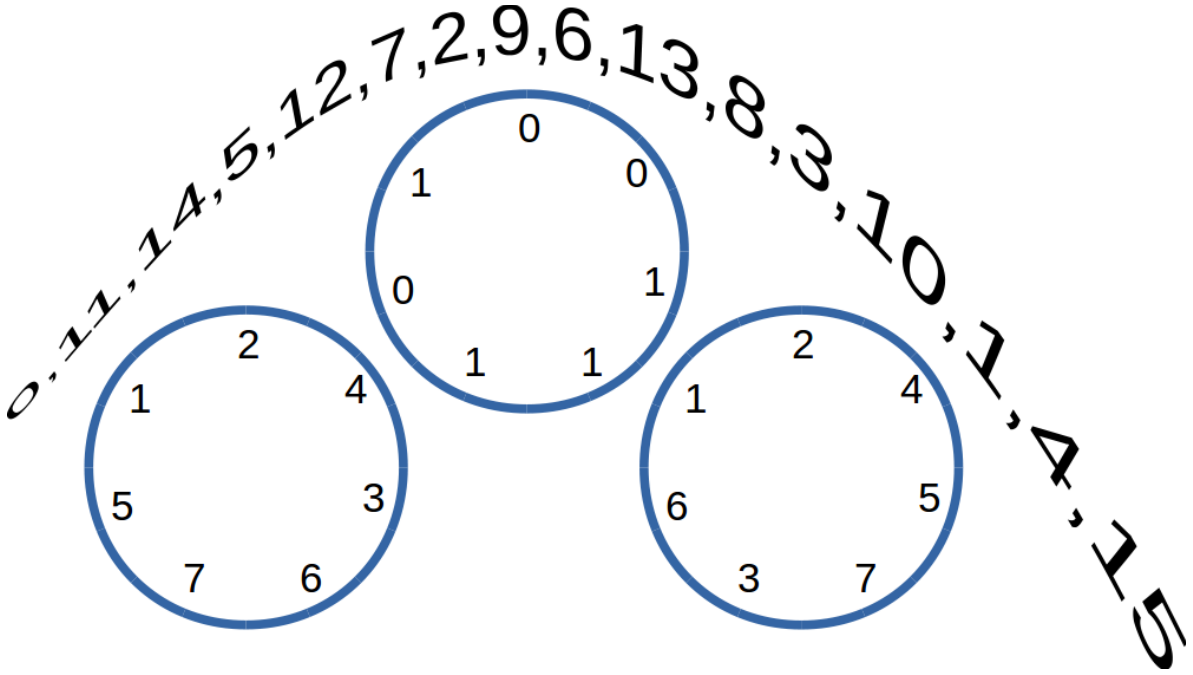
В пятой работе изучаются особенности спектров цифровых сигналов с линейной модуляцией. Акцент делается на влияние автокорреляционной функции дискретной случайной последовательности на спектр линейно-модулированного сигнала. В частности, исследуется код с чередованием полярности и код MLT-3.

Шестая работа посвящена дельта-модуляции. Разъясняется принцип формирования выходного сигнала модулятора, а также способ его демодуляции. Иллюстрируется применение z-преобразования для расчета остаточного напряжения на выходе цифрового фильтра нижних частот в режиме молчания.

Седьмая работа посвящена принципам и правилам двоичного кодирования случайной последовательности. Иллюстрируется противостояние методов последовательного счета и поразрядного взвешивания. С использованием макета изучается кодирование поразрядным взвешиванием числами Фибоначчи. Показывается возможность обнаружения и исправления некоторых битовых ошибок, возникающих непосредственно в процессе кодирования.

ОГЛАВЛЕНИЕ

ЦИКЛИЧЕСКИЕ КОДЫ-1



1. Введение

Современная цифровая система передачи информации напичкана множеством разных кодов. Среди этого множества можно выделить коды канала и коды источника (*шифрование* как специфический вид кодирования отдадим «раздутой» криптографии).

Коды канала (помехоустойчивые коды) созданы для борьбы с вредным влиянием помех. Боевым механизмом является разумно введенная *избыточность*. Избыточность (при прочих равных условиях) снижает скорость передачи полезной (пользовательской) информации, потому что на передачу избыточной информации (контрольной суммы) требуется дополнительное время. Количество символов n после избыточного кодирования всегда больше количества символов k до кодирования, $n>k$.

Коды источника (экономные коды) уменьшают всякую избыточность, и созданы для увеличения скорости передачи полезной информации. Использование таких кодов (при прочих равных условиях) снижает помехозащищенность системы передачи информации в целом, так как информация после экономного кодирования упакована плотно, и искажение кодированного символа приведет при декодировании к искажению множества символов исходного сообщения. Количество символов после экономного кодирования, как правило, меньше количества символов до кодирования. Как правило, потому что не всегда удается получить выигрыш от кодирования; попробуйте, допустим, сжать архиватором видеофайл.

В системах передачи информации экономные коды и коды канала используются вместе. Это объясняется тем, что сперва всякая (то есть неконтролируемая) избыточность должна быть по возможности устранена, а затем, по мере надобности, должна быть введена такая избыточность (контролируемая), которая будет направлена на борьбу с ошибками. Можно сказать, что коды канала вносят полезную избыточность.

Избыточность содержится во многих источниках информации, и ее, как правило, нельзя использовать для борьбы с ошибками, потому что такая избыточность для технического устройства является случайной. Вспомните запись денежной суммы прописью — эта избыточность вводится для человека, она «бухгалтерская» и больше привязана к русскому языку, чем к математике. Из-за «случайности» естественной избыточности, ее долю с помощью экономных кодов стараются уменьшить.

К кодам канала можно отнести все коды, которые вычисляют контрольные суммы и предназначены для обнаружения (и/или исправления) ошибок. Это, например, циклические суммы **CRC,** хэш-суммы **MD5** и **SHA** и т. д. Перечисленные коды предназначены лишь для обнаружения ошибок, а исправление в этом случае делается путем повторного запроса блока данных и повторной проверки контрольной суммы. Для «настоящего» исправления ошибок используются, например, сверточные коды и коды Рида-Соломона.

К кодам источника относятся все архиваторы и мультимедиа кодеки, причем некоторые архиваторы также поддерживают контрольные суммы и могут обнаруживать ошибки при повреждении архива.

Блоки кодирования и декодирования, собранные вместе, называют **кодеком** (так же как и модулятор и демодулятор называют модемом).

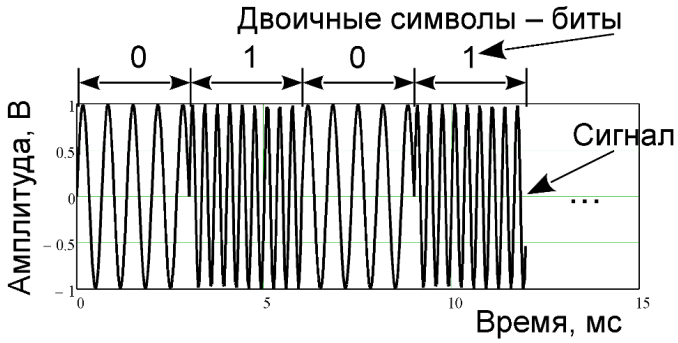
В этой лабораторной работе изучаются пространственные свойства циклического кода Хемминга (7, 4), а также методы кодирования и декодирования этого кода. Логическим продолжением является работа **Циклические коды-2,** в которой изучаются принципы реализации кодеков на основе регистров сдвига с линейными обратными связями.

Отчет по работе должен состоять из следующих пунктов:

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

2. Сведения из теории

Циклические коды относятся к *помехоустойчивым кодам* и предназначены для защиты сигналов от помех, которые могут приводить к ошибкам демодуляции (распознавания) принимаемого сигнала.



Помехоустойчивое кодирование принято называть *кодированием канала.* Неизменной особенностью помехоустойчивых кодов является избыточность, введенная по заранее определенным правилам, проверка которых на приемной стороне позволяет обнаруживать и/или исправлять некоторые ошибки.

Самый простой способ помехоустойчивого кодирования (и декодирования) — табличный, определяемый кодовой таблицей. Таблица состоит из двух колонок, причем в первой перечислены все возможные информационные слова, а во второй — им соответствующие разрешенные (табл. 1).

Табл. 1. Пример помехоустойчивого (6, 2)*-кода, заданного таблично*

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

К табличному коду предъявляются два требования: **однозначность декодирования** и наличие **избыточности.** Разрешенные кодовые слова желательно выбирать так, чтобы они в меньшей степени походили друг на друга (были бы подальше удалены друг от друга).

Примером табличного кодирования является код 4B/5B, используемый для целей тактовой синхронизации на физическом уровне сетевой технологии Ethernet 100Мбит/с (100Base-TX). Этот код гарантирует в

выходном потоке не более трех нулевых битов подряд. Кодовая таблица состоит из 16 строк и не накладывает серьезных ограничений на объем памяти кодека. Избыточность кода 4B/5B проявляется в том, что $2^4=16$ разрешенных слов кодируются пятью битами. Этот пример показывает, что избыточность можно использовать не только для борьбы с ошибками, но и, например, как средство, помогающее восстановить тактовые синхроимпульсы в приемнике.

Если оценить размер кодовой таблицы кода с информационными словами, допустим, из 100 битов, то выйдет $2^{100} \approx 10^{30}$ строк! Очевидно, что никакое электронное устройство не имеет такого объема памяти. Однако, коды таких размеров используются, например, в современном стандарте цифрового телевидения **DVB-T2** [1].

Катастрофическое разрастание кодовой таблицы было обыграно переходом от табличных кодов к *линейным* (это один из примеров практической пользы математики). Главной особенностью линейных кодов является линейность операций кодирования и декодирования. Это позволяет вместо кодовой таблицы пользоваться **порождающей матрицей,** содержащей k строк и n столбцов. С помощью такой матрицы можно «породить» (вычислить) 2^k двоичных кодовых слов (и, вообще, m^k кодовых слов, состоящих из m -значных символов).

Требование однозначности декодирования табличных кодов приводит к необходимости **линейной независимости строк** порождающей матрицы, так как разрешенные кодовые слова (длиной n) являются значениями всех возможных линейных комбинаций строк этой матрицы.

Стремление к компактности привело к тому, что среди линейных кодов выделили класс циклических, фундаментом которых является следующая аксиома: *циклическая перестановка* любого разрешенного кодового слова дает разрешенное кодовое слово. При этом доказано [2], что циклический код можно задать всего лишь одним вектором (одной строкой порождающей матрицы). Если теперь элементы этого вектора рассмотреть как коэффициенты

некоторого полинома, то этот полином будет **порождающим полиномом** циклического кода.

Таким образом, циклические коды имеют порождающий полином, следующую из него порождающую матрицу, и, естественно, кодовую таблицу, то есть данные коды — это весьма интересное и не совсем простое для изучения подмножество линейных кодов (не всякий линейный код будет циклическим, но всякий циклический является линейным).

В данной работе изучаются два циклических кода Хемминга (7, 4) со следующими порождающими полиномами [1]

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

Пара чисел (7, 4) определяет размеры кодовых слов: на вход кодирующего устройства подаются $k=4$ бита, а с выхода считываются $n=7$, причем три из них являются проверочными битами, $r=n-k=3$. Коэффициент избыточности $R=r/n$ здесь равен $3/7$.

Пусть соответствие элементов произвольного кодового слова коэффициентам некоторого полинома строится по следующему правилу

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

тогда двум полиномам (1) будут соответствовать два кодовых слова

$$\begin{aligned} g^{(1)} &= (1101000) \\ g^{(2)} &= (1011000) \end{aligned} . \qquad (3)$$

Циклические коды являются *блочными,* так как каждое кодовое слово кодируется и декодируется независимо от других кодовых слов (говорят, что *кодовые слова независимы*).

Элементы кодовых слов (и, само собой, коэффициенты полиномов) для **двоичных** кодов суммируются по модулю **два**

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

Это правило сформировано на основе обычного суммирования с последующим вычислением остатка от деления на 2. Вычисление остатка

дает гарантию того, что результат будет принадлежать алфавиту из символов {0, 1}.

Циклические коды являются блочными кодами, поэтому степени всех полиномов должны быть меньше n, то есть меньше длины кодового слова. Цикличность при этом заключается в том, что

$$x^n = x^0 = 1 \ ,$$

(это равенство, несмотря на его простоту, очень содержательно)

то есть n-я степень x эквивалентна нулевой степени. Величина n является своеобразным модулем (периодом цикла) при операциях со степенями.

Поясним цикличность на примере следующего полинома

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

Полиному (5) при периоде цикла $n = 7$ по правилу (2) соответствует следующее кодовое слово

$$(1100100) \ .$$

Полиному, умноженному на x, соответствует кодовое слово, сдвинутое вправо на один бит, при этом за счет равенства $x^n = 1$ бит из крайней правой позиции должен быть перемещен в крайнюю левую (рис. 1).

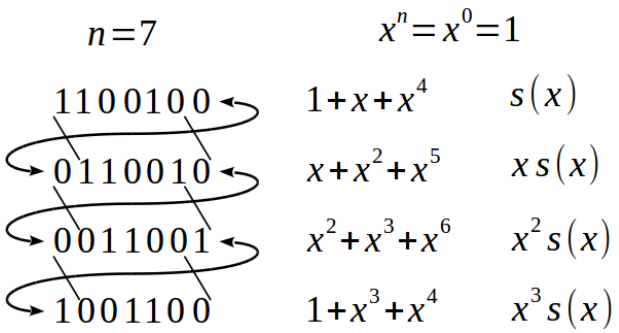


Рисунок 1 Последовательные циклические сдвиги кодового слова на один, два и три бита вправо

Кодирование циклическим кодом осуществляется достаточно просто: входное кодовое слово преобразуется в выходное с помощью умножения входного полинома $a(x)$ на порождающий

$$s(x) = a(x)g(x) \ , \qquad (6)$$

при этом доказано [2], что степень порождающего полинома должна быть равна $r = n - k$.

Что любопытно, фундаментальное равенство $x^n = 1$, если его переписать в виде $x^n - 1 = 0$ и разложить на множители двучлен $x^n - 1$,

позволяет определить все возможные порождающие полиномы путем комбинирования входящих многочленов.

Изучаемым здесь кодам соответствует следующая факторизация

$$x^7+1=(x+1)(x^3+x+1)(x^3+x^2+1)\ ,$$

которая проверяется непосредственно (естественно, что коэффициенты полиномов при этом следует приводить по модулю два).

Более того, если известен порождающий полином $g(x)$, то по двучлену x^n-1 можно найти проверочный полином $h(x)=(x^n-1)/g(x)$ (остаток от деления при этом обязан быть равен нулю).

Коэффициенты при g_0 и g_r должны отличаться от нуля, иначе код снизит свою помехоустойчивость[1]. Для двоичных кодов единственным ненулевым элементом является единица, поэтому для них всегда

$$g_0=g_r=1.$$

При способе кодирования (6) разрешенное слово (то, которое на выходе кодера) не обязано состоять из слова вида [информационное слово, проверочное слово]*.

---

*Непосредственной подстановкой можно убедиться в том, что, например, для кода (7, 4) с полиномом $g(x)=x^3+x+1$ и входного слова $a=(a_0a_1a_2a_3)$ выходное слово равно $s=(a_0, a_0+a_1, a_1+a_2, a_0+a_2+a_3, a_1+a_3, a_2, a_3)$. Из этого следует, что на последних двух позициях и на первой стоят информационные символы. Информационный символ a_1 «размазан» по второму, третьему и пятому символам выходного слова.

Однако, оказывается, что разрешенные слова циклического кода всегда можно представить в таком удобном «факторизованном» виде. Для этого информационные символы сдвигают на r позиций вправо, чему, как мы уже знаем, соответствует произведение $a(x)x^r$. После такой операции резервируется r нулевых позиций под проверочные (контрольные) символы, которые необходимо вычислить. Их вычисление и добавление гарантирует

1 Желающие могут в этом убедиться непосредственным сравнением кодов.

принадлежность сформировавшегося слова кодовой таблице, а значит оно будет **разрешенным.**

Чтобы некоторое кодовое слово было разрешенным, из правила кодирования (6) следует, что соответствующий полином должен делиться без остатка на порождающий[2]

$$res(x)=s(x) \bmod g(x)\equiv 0. \qquad (7)$$

Поэтому при кодировании таким «факторизованным» кодом вычисляют остаток от деления «сдвинутого» информационного полинома на порождающий

$$[a(x)x^r] \bmod g(x). \qquad (8)$$

Чтобы полином $a(x)x^r$ делился без остатка, необходимо вычесть из него найденный остаток (8). Для двоичных кодов (и только для них) вычитание эквивалентно сложению по модулю два

$$1-1=1+1=0 \ , \ 0-1=0+1=1 \ ,$$

поэтому «факторизованному» кодированию соответствует правило

$$s(x)=a(x)x^r+[a(x)x^r] \bmod g(x). \qquad (9)$$

Выражение (9) определяет *кодирование в систематической форме* [1], исследованию которого и посвящена данная лабораторная работа, при этом правило (6) определяет *кодирование в НЕсистематической форме.*

Правило декодирования принятого кодового слова v , которому соответствует полином $v(x)$, такое:

А) **вычисляется остаток** от деления полинома $v(x)$ на порождающий полином $g(x)$;

В) **определяется номер ошибочного символа,** который в случае двоичных кодов исправляется на обратный.

Пункт **В** определяет действия декодера в режиме исправления однократных ошибок (так, как это реализовано в лабораторном макете). Декодер, в принципе, может работать в режиме обнаружения ошибок, когда вычисленный остаток сравнивается с нулем и принимается решение о

2 Функция остатка обозначается символами mod, полином остаток как res(x)

повторном запросе (также декодер может работать в режиме восстановления стертых символов, а также в смешанных режимах).

Если остаток от деления отличен от нуля, то принятое кодовое слово является запрещенным (ошибка при этом обнаружена). Внимание! Если остаток равен нулю, то это говорит лишь о том, что принятое кодовое слово является разрешенным; в этом случае либо ошибки нет, либо она такая, что перевела одно разрешенное слово в другое — не обнаруживаемая ошибка.

Благодаря линейности циклического кода остаток от деления определяется лишь вектором ошибки и не зависит от переданного слова

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

где $e(x)$ — полином, соответствующий *вектору ошибок* $\mathbf{e}$.

Для удобства вводят такое понятие как *кратность ошибки* q , которая численно равна количеству ненулевых элементов в векторе ошибок $\mathbf{e}$, или, что эквивалентно, его весу $q=w(\mathbf{e})$.

Подытоживая, скажем, что декодирование — это анализ принятых символов и принятие решения из заранее составленного набора решений.

3. Примеры кодирования и декодирования

Выберем один из двух порождающих полиномов кода Хемминга (7, 4)

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

Зададим входное (информационное) слово $a=(0111)$, выпишем ему соответствующий полином

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

Разберем оба способа кодирования, а также пример декодирования систематического кода.

Кодирование в систематической форме

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

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

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

который соответствует кодовому слову $s=(\mathbf{010}0111)$, при этом жирным шрифтом выделены проверочные символы, а красным — информационные.

При делении в столбик не забываем про приведение коэффициентов полиномов *по модулю два*

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

Кодирование в НЕсистематической форме

Оно алгебраически тривиально

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

и дает следующее разрешенное слово $s=(\mathbf{01}10001)$.

Задание: **показать** (алгебраически), что первые два символа и последний в найденном разрешенном слове (они выделены жирным шрифтом) являются информационными символами a_0 , a_1 и a_3 , соответственно.

Декодирование систематического кода

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

Разделим в столбик полином $s(x)=x^6+x^5+x^4+x$, соответствующий разрешенному слову **систематического** кода, на порождающий полином. Получим нулевой остаток, что говорит о том, что принятое слово является разрешенным, то есть оно принадлежит кодовой таблице. Так как код систематический, то декодируемым словом будет $\hat{a}=(0111)$, которое совпадает с исходным.

Рассмотрим процесс декодирования при наличии однократной ошибки

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

Запишем принятый полином

$$v(x)=s(x)+e(x)=\left(x^6+x^5+x^4+x\right)+x^5=x^6+x^4+x \quad .$$

Деление полинома $v(x)$ даст ненулевой полином-остаток $x+1$, которому соответствует вектор (110) в двоичном представлении (младший бит слева).

Если все семь остатков от деления $e(x)=x^i,\ i=0...6$ на $g(x)$ занести в таблицу, то окажется, что все остатки получаются **разными** (проверьте это!), а остатку $x+1$ будет соответствовать полином ошибок $e(x)=x^5$. Таким образом, принятое слово (0100101) корректируется в (0100111) и декодируется как $\hat{a}=(0111)$. Ошибка была исправлена.

4. Описание лабораторного макета

Лабораторный макет представляет из себя приложение **cyclic,** написанное на языке C++ с использованием библиотек **Qt** [3] (рис. 2).

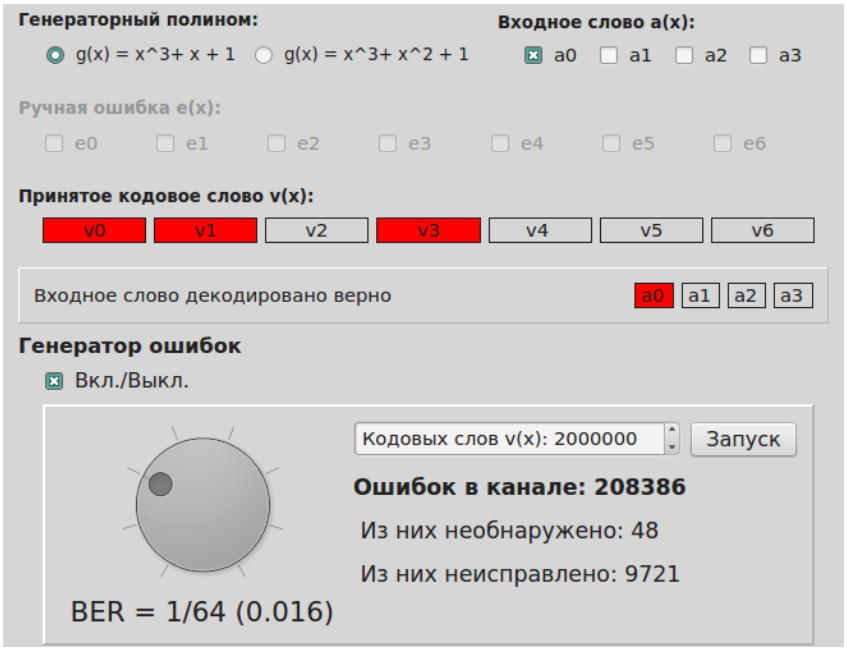


Рисунок 2 Программный макет для изучения циклического кода Хемминга (7, 4)

В макете реализовано кодирование и декодирование систематического циклического кода (7, 4).

Порождающие (генераторные) полиномы $g(x)$ выбираются соответствующими кнопками; входной полином $a(x)$ и полином ручной ошибки $e(x)$ задаются флажками, причем поднятый флажок означает бит **1,** а опущенный — бит **0.** Принятое из канала кодовое слово декодируется автоматически, причем красный цвет означает бит **1**, а цвет фона окна — бит **0.** Результат декодирования выводится на панель, расположенную непосредственно под

панелью Принятое кодовое слово v(x). На рис. 2 результат декодирования показан 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 статистических испытаний, которые заключаются в генерации случайных ошибок согласно модели *двоичного симметричного канала с независимыми ошибками* (п. 5.3). *Вероятность битовой ошибки* задается регулятором BER. Объем статистических испытаний задается числом в поле Кодовых слов v(x).

Статистические испытания начинаются сразу после нажатия кнопки Запуск. По окончании испытаний в текстовые поля выводятся числовые результаты. При включении генератора ошибок, режим ручной ошибки автоматически выключается.

Декодер при включенном генераторе ошибок параллельно работает в двух режимах (рис. 3):

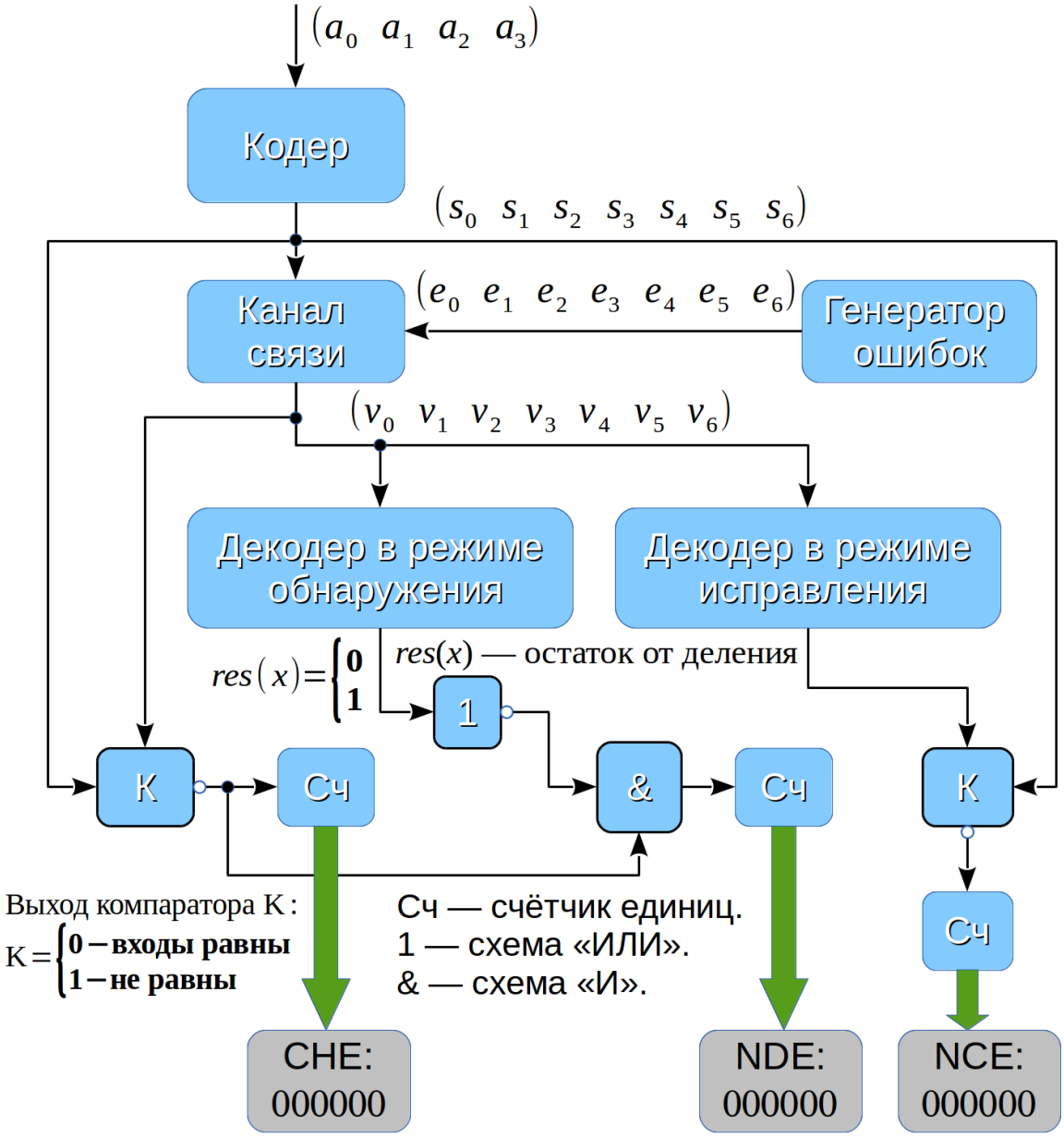
- В режиме обнаружения ошибок;
- В режиме исправления ошибок.

Изучаемый код Хемминга (7, 4) способен гарантированно обнаружить все ошибки вплоть до кратности два включительно, а гарантированно исправить — лишь однократные.

Если декодер работает в режиме обнаружения, то он после вычисления остатка выносит одно из двух возможных решений: либо ошибка обнаружена, либо — нет. Это делается благодаря сравнению остатка от деления с нулем. Приложение при этом подсчитывает (счетчик **NDE, No Detected Errors,** рис. 3) количество кодовых слов с не обнаруженными ошибками (при условии, что ошибка произошла, то есть при $e(x) \neq 0$). Заметим, что не будут обнаружены такие, и только такие, ошибки, которым соответствует полином $e(x) \neq 0$, **делящийся без остатка** на порождающий.

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 навредит (вводя в макете ошибку вручную, это следует пронаблюдать). Приложение при этом подсчитывает (счетчик **NCE, No Corrected Errors,** рис. 3) количество кодовых слов с неисправленными ошибками.

Счетчик **CHE** (**Channel Errors,** рис. 3) показывает количество кодовых слов с **хотя бы** одной канальной ошибкой.



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

5. Порядок выполнения работы

1. Сделать расчетное задание;
2. Ответить на вопросы;
3. Выполнить экспериментальную часть работы.

5.1 Расчетное задание

Методом деления в столбик отыскать все разрешенные слова двух систематических кодов с полиномами $g_1(x)$ и $g_2(x)$. Результаты занести в таблицы А и В. Также вычислить веса w разрешенных слов.

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

№	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, взяв зеркальный полином

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

Проверить правильность заполнения кодовых таблиц с помощью приложения **cyclic.**

Взять наугад из таблиц А и В по одному разрешенному слову, сопоставить этим словам полиномы и найти остатки от деления этих полиномов на соответствующие порождающие. Убедиться, что найденные остатки равны нулю.

Таблицу С заполнить полиномами-остатками от деления x^i на $g_1(x)$ и $g_2(x)$.

Таблица С. Остатки от деления целых степеней x на порождающие полиномы

Порождающий полином $g_1(x)$		Порождающий полином $g_2(x)$	
Делимое	Остаток	Делимое	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 выполнения расчетного задания ответить на вопросы. Ответы **записать** в отчет.

5.2 Анализ результатов выполнения расчетного задания

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

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

2. Определить веса, по которым сразу определяются запрещенные 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 слово?

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

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

6. Если к любому разрешенному слову прибавить любое, имеющее вес два (то есть ввести двукратную ошибку), 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 какова вероятность 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 ситуаций использовать таблицу С.

5.3 Экспериментальная часть

В лабораторном макете имеются два режима ввода ошибок:

1. Режим ручной ошибки, когда оператор поднимает/опускает флажки на панели **Ручная ошибка *e*(*x*)** (рис. 2). Если флажок поднят, то соответствующий бит в принятом кодовом слове инвертируется (**0** заменяется на **1**, **1** на **0**).
2. Режим генерации ошибок, когда оператор задает объем испытаний и после нажатия кнопки **Запуск** программа случайным образом вводит ошибки в фиксированное разрешенное слово.

Режим ручной ошибки

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

- Все 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 **занести** в отчет.

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

Режим генерации ошибок

В лабораторной макете реализована модель двоичного симметричного канала с независимыми ошибками (рис. 4). Ошибка в каждом бите возникает с вероятностью p (вероятность битовой ошибки, BER, *Bit Error Rate*). Вероятность q-кратной ошибки определяется *формулой Бернулли* (см. ниже).

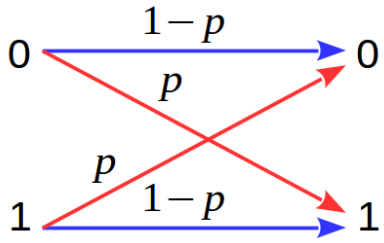


Рисунок 4 Модель двоичного симметричного канала с независимыми ошибками

Предварительно сделав расчетное задание (п. 5.1) и ответив на вопросы (п. 5.2), **для двух** изучаемых кодов (7, 4) **получить** точные формулы, позволяющие вычислить следующие вероятности:

- Вероятность P_1 того, что кодовое слово из семи бит, прошедшее через рассматриваемый канал, **будет содержать ошибку;**
- Вероятность P_2 того, что **имеющаяся** в принятом (канальном) кодовом слове ошибка **не будет обнаружена** декодером (декодер при этом работает в режиме обнаружения);
- Вероятность P_3 того, что имеющаяся ошибка **не будет исправлена** декодером (декодер при этом работает в режиме исправления);

При выводе формул в качестве базовой использовать формулу Бернулли, которая позволяет вычислить вероятность $P(q)$ того, что в слове из n битов произойдет ровно q битовых ошибок

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

где p — вероятность битовой ошибки;

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

Например, $q=2$ означает то, что в кодовом слове из семи битов произошла ошибка в двух битах. Число разных двукратных ошибок в нашем случае составляет $C_7^2=21$. Числа сочетаний также могут быть вычислены с помощью треугольника Паскаля.

При выводе формул **учесть** результаты ответов на вопросы из п. 5.2.

По найденным формулам для вероятностей битовых ошибок

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

рассчитать $\{P_1,P_2,P_3\}$ и занести результаты в таблицу.

В приложении **cyclic** включить генератор ошибок и **для обоих** порождающих полиномов провести статистические испытания. Результаты **занести** в таблицу. Входное слово при этом может быть любым (кстати, почему?). Число испытаний выбрать из диапазона 2’000’000–5’000’000 и больше **не менять.**

По результатам испытаний найти оценки $\{\hat{P}_1,\hat{P}_2,\hat{P}_3\}$ и **сравнить** их с вычисленными теоретическими значениями $\{P_1,P_2,P_3\}$.

Перенести содержимое файлов **decoder_error_distrib_inv_p_X,** созданных приложением, в таблицу. Параметр **X** в именах файлов равен обратной вероятности битовой ошибки в канале, то есть $1/p$. В файлах записано экспериментальное распределение кратности битовых ошибок **после декодирования.** По заполненной таблице численно оценить вероятность битовой ошибки на выходе декодера $\hat{p}_{\text{out}}$ в режиме исправления и построить график зависимости $\hat{p}_{\text{out}}(p)$. При этом по обеим осям должен быть логарифмический масштаб.

Дозаполнить таблицу D, переключив приложение **cyclic** в режим ручной ошибки.

Таблица D. Теоретическое распределение кратностей ошибок на входе и выходе декодера циклического кода Хемминга (7, 4)

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 и формулы Бернулли **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 $p_{\text{out}}(p)$ на одном графике с экспериментальной зависимостью $\hat{p}_{\text{out}}(p)$.

Ответить на вопросы:

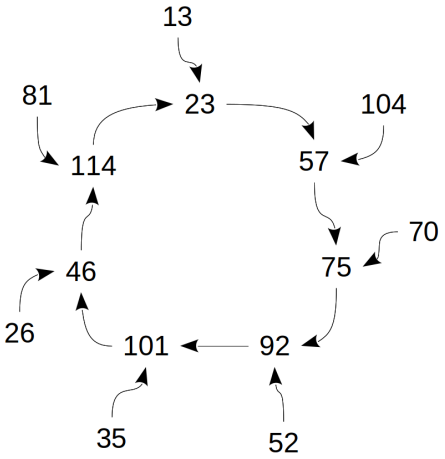
1. Чем отличаются два рассмотренных кода? В чем совпадают?
2. Какой из рассмотренных кодов более помехоустойчивый и, если это так, то почему?

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. Сагалович Ю.Л. Введение в алгебраические коды. Учебное пособие. — 2-е изд., перераб. и доп. — М.: ИППИ РАН, 2010. — 302 с.
3. Qt | Cross-platform application & UI development framework, http://www.qt.io/.

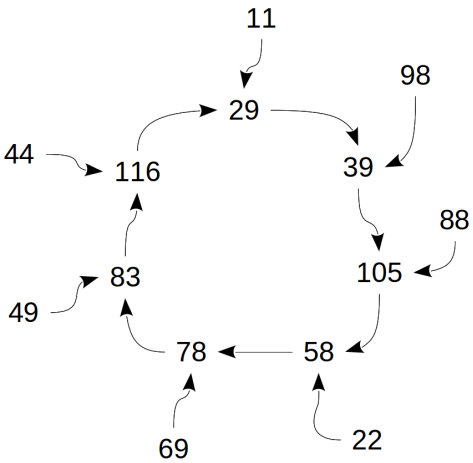
Для заметок

ЦИКЛИЧЕСКИЕ КОДЫ-2



23=0010111

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



29=0011101

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

1. Введение

Перед выполнением данной работы рекомендуется сделать работу **Циклические коды-1.**

В данной лабораторной работе изучаются регистры сдвига с линейными обратными связями, на которых реализуются кодеки циклических кодов.

Рассматриваются циклические коды Хемминга (7, 4) и (15, 11).

Отчет должен состоять из:

- ✔ Титульного листа;
- ✔ Хода работы;
- ✔ Ответов на вопросы;
- ✔ Выводов.

2. Сведения из теории

Так как циклические коды полностью задаются полиномом (порождающим или проверочным), то кодеки циклических кодов являются **самыми экономными** среди кодеков линейных блочных кодов и включают в себя **один** (меньше не бывает) *регистр сдвига*.

Регистр сдвига — это цифровое устройство, состоящее из последовательно соединенных ячеек памяти, и позволяющее синхронно сдвигать все символы из одной ячейки в соседнюю (рис. 5).

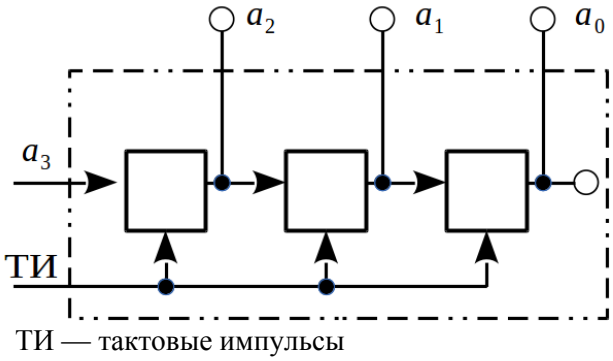


Рисунок 5 Пример трехразрядного регистра сдвига

Исторически сложилось так, что для электронного представления цифровой информации используется двоичный алфавит, который состоит из двух символов — битов $A=\{0,1\}$. Примером тому являются все широко распространенные модули оперативной памяти компьютеров, реализованные в транзисторной логике. Транзистор имеет два четко различимых режима работы: режим открыт (большое потребление тока) и режим закрыт (малое потребление тока). По этой причине транзисторы удобны для оперативного хранения битов, правда для этого требуется как минимум два транзистора — *триггер* [1].

Если попытаться с помощью того же количества транзисторов хранить трехзначные символы (триты), то транзистор потребуется вводить в «средний» (линейный) режим работы. Это приведет к усложнению схемотехники и, кроме того, повысится вероятность ошибки чтения данных. Однако, в настоящее время имеются карты памяти и твердотельные диски с тремя [2] и даже четырьмя [3] битами на ячейку! Это связано с желанием повысить плотность записи в накопителях (а в конечном итоге — их объем в гигабайтах), при этом

добиться приемлемой вероятности ошибки помогают помехоустойчивые коды, в особенности циклические и притом с основанием кода $m>2$.

Символы, хранящиеся в ячейках регистра сдвига (рис. 5), после прихода *тактового импульса* (ТИ) переходят в следующие ячейки согласно стрелкам. Индексом i в символе a_i обозначен *дискретный момент времени.* Во время действия очередного ТИ, символ со входа текущей ячейки перемещается на вход следующей. После ослабления переходных процессов появляется интервал времени, когда с регистра можно считывать данные. Этот интервал времени можно обозначить индексом.

Пусть, например, состояние $a_2 a_1 a_0$ регистра на рис. 5 соответствует моменту времени с индексом 0, тогда 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 либо потеряется, либо пойдет на вход следующего устройства.

Бывает полезно (для вывода формул) вообразить, что **новые** (подходящие к регистру) символы хранятся в неограниченно длинном виртуальном регистре, расположенном слева от входа реального регистра, и такт за тактом поступают на его вход. Аналогичная ситуация и с потерявшимися символами: можно представить, что они уходят в неограниченно длинный виртуальный регистр, находящийся справа от выхода реального регистра.

Так как в этой лабораторной работе рассматриваются двоичные коды, то символы a_i могут принимать всего лишь два значения, которые условно обозначаются как **0** и **1.**

По своей сути регистр сдвига является буфером, предназначенным для хранения и преобразования цифровой информации, последовательно поступающей на его вход. На базе таких регистров строятся цифровые фильтры и, в частности, кодеки циклических кодов (для программистов **регистр сдвига** можно заменить **очередью**).

Поставим в соответствие некоторой точке регистра сдвига полином согласно ниже идущим рассуждениям.

Любая ячейка памяти имеет вход и выход. Направление передвижения символов указывается стрелкой. В любой момент времени на входе и выходе ячейки имеются символы. В зависимости от коэффициента передачи ячейки, логически можно различать два типа ячеек памяти: **повышающая и понижающая,** хотя физически они ничем не отличаются (рис. 6).

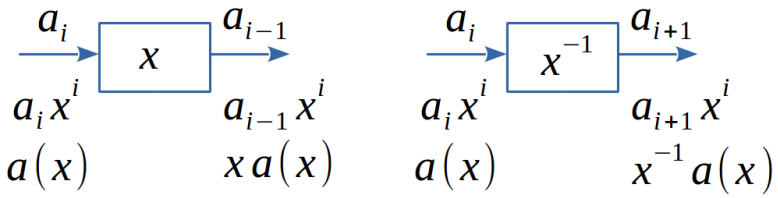


Рисунок 6 Ячейки памяти двух логических типов: повышающая и понижающая

Состояние ячеек на рис. 6 «сфотографировано» в момент времени с индексом i, чему символически 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. У левой ячейки коэффициент передачи x (она повышает степень), у правой $1/x$ (она понижает степень).

Количество слагаемых в полиномах удобно ничем не ограничивать (вспоминаем виртуальный регистр), то есть всегда полагать, что

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

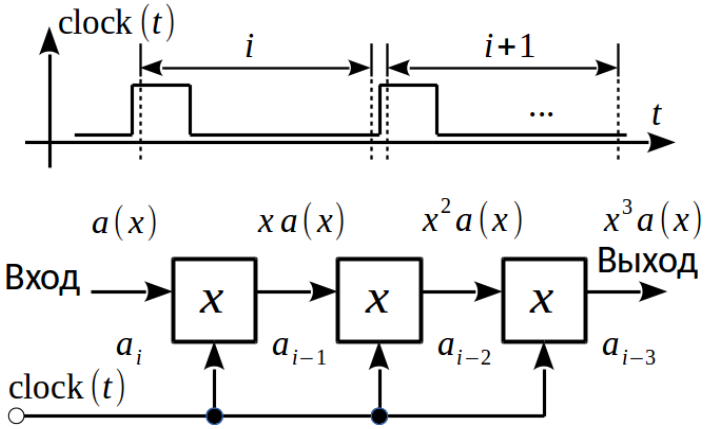


Рисунок 7 Трехразрядный регистр на повышающих ячейках

Это дает право не думать про согласование индексов на границах при разного рода преобразованиях: умножении двух полиномов, сложении, делении.

Рассмотрим для примера регистр сдвига, изображенный на рис. 7. Он меняет свое состояние с каждым приходом ТИ, реагируя на перепад напряжения снизу-вверх (на передний фронт). В течение времени с индексом i можно считывать данные.

Глядя на рис. 7, можно вычислить коэффициент передачи данного регистра, равный отношению выходного полинома ко входному

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

Такой коэффициент передачи говорит о том, что рассматриваемый фильтр выполняет задержку входных символов на три такта.

Простейшим способом кодирования с помощью циклического кода является умножение информационного полинома $a(x)$ на порождающий $g(x)$. Из только что сделанных выводов можно заключить, что для такого кодирования необходим цифровой фильтр, коэффициент передачи которого совпадает с порождающим полиномом кода.

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

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

В нем присутствуют три слагаемых, откуда следует, что для кодирования необходимы *сумматоры*.

Считается, что **сумматоры** являются безынерционными элементами, то есть складывающими мгновенно. Однако, на самом деле (схемотехнически) им требуется время, но это время много меньше длительности такта и этим фактором 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. Вся числовая ось из целых чисел отображается по модулю два в числовую ось из чередующихся нулей и единиц ...0,1,0,1,0,1,0,1..., что соответствует разделению всех целых чисел по признаку четное нечетное.

Рассмотрим, например, алфавит из трех символов $\{-1,0,1\}$, где модуль равен трем. Запишем произвольную сумму и приведем ее по модулю три

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

Выражение

$$2\equiv -1 \bmod 3$$

читается так: число 2 **сравнимо** с числом –1 по модулю 3.

Таким образом выяснено, что схема несистематического кодера двоичного циклического кода Хемминга (7, 4) может иметь следующий вид (рис. 8).

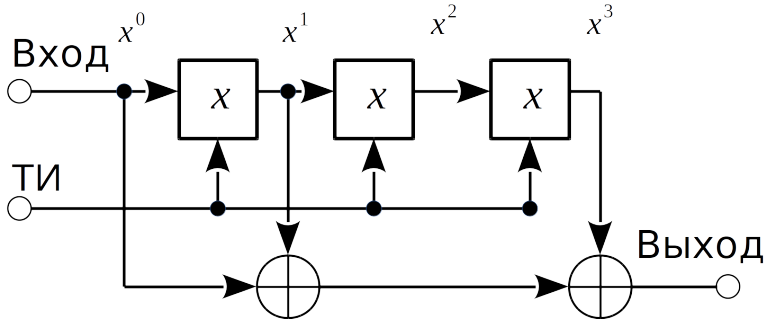


Рисунок 8 Схема несистематического кодера двоичного циклического кода Хемминга (7, 4) *с порождающим полиномом* x^3+x+1

Как видно из рис. 8, ТИ на сумматоры не подаются, так как они в них не нуждаются.

Проверим корректность работы предложенной схемы.

Для этого зададим входное кодовое слово $a=(0011)$, которому соответствует полином

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

и 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 это же кодовое слово с помощью предложенной схемы. Для этого входное слово младшим битом вперед подадим на вход кодера и, соответственно, по тактам распишем этапы кодирования (табл. 1). Каждая строка таблицы соответствует некоторому дискретному моменту времени. В начальный момент времени регистр обнулен.

Табл. 1. Результат потактовой работы несистематического кодера циклического кода Хемминга (7, 4) *(рис. 8)*

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

Биты столбца x^0 совпадают с битами на входе кодера. Биты столбца s_i вычисляются как сумма *по модулю два* битов в столбцах x^0, x^1 и x^3, то есть в точном согласии с порождающим полиномом. Коэффициент при старшей степени выходного полинома определится на седьмом такте.

Итак, детально разобран **принцип умножения двух полиномов** с помощью регистра сдвига и набора сумматоров. Для завершения теоретического обзора необходимо разобрать **принцип деления двух полиномов,** что заметно сложнее из-за появления такого объекта как *остаток от деления...*

Для возможности деления в регистр сдвига с помощью сумматоров необходимо ввести линейные обратные связи.

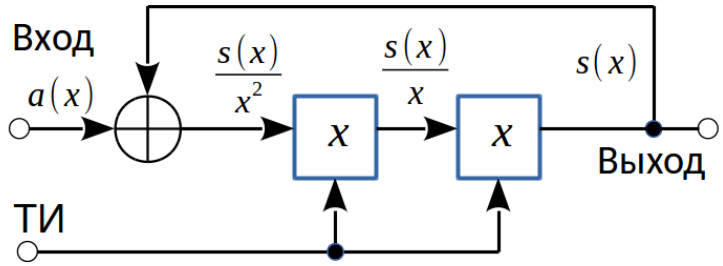


Рисунок 9 Двухразрядный регистр сдвига с обратной связью — пример схемы деления двух полиномов

Рассмотрим для начала регистр из двух ячеек памяти с одной обратной связью (рис. 9).

Так как схема содержит один сумматор, то для определения коэффициента передачи потребуется лишь одно уравнение

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

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

На выходе сумматора получается опереженный на два такта выходной полином $s(x)$, что полностью согласуется с правилом умножения или деления полинома на степени переменной x (рис. 6). Также следует учесть, что знак ⊕ в рассматриваемом примере является суммой по модулю два, поэтому слагаемые переносятся за знак равенства **без изменения знака.**

Замечаем, что 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. 10).

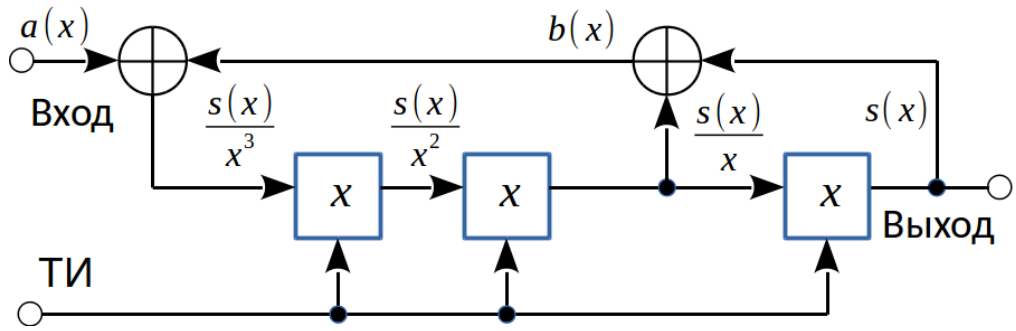


Рисунок 10 Трехразрядный регистр сдвига с обратными связями

В трехразрядном регистре появляется возможность несимметричного включения сумматоров, что позволяет увидеть закономерность формирования знаменателя коэффициента передачи.

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

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

и исключить вспомогательный полином $b(x)$

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

Найденное выражение показывает, что степени слагаемых полинома делителя растут слева-направо, то есть от входа регистра к его выходу. Запоминать это не следует, так как дальше будут приведены модифицированные схемы, которые изменят этот порядок, хотя, записать на листочке шаблоны (их будет четыре) весьма полезно.

Чтобы осмыслить найденное выражение коэффициента передачи, разложим его в ряд Тейлора по степеням переменной x . Для этого очень удобно (экономит время) использовать программы компьютерной алгебры (Maxima, Mathcad, SymPy (Python) и другие)

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

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

Знание коэффициента передачи любой схемы, состоящей из регистра сдвига и набора сумматоров, позволяет напрямую записать выходную последовательность, если на вход при этом подается 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 найденного разложения, 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 возникают благодаря числителю x^3 и соответствуют задержке на три такта. После трех нулей идет некоторая последовательность битов, которая неизбежно должна повториться (период выделен **жирным** шрифтом), так как состояние регистра (рис. 10) полностью определяется состоянием трех ячеек (а число разных состояний конечно).

С другой стороны, если отключить вход и принудительно задать хотя бы одну единицу в любой точке регистра, то такая схема начнет генерировать периодическую последовательность символов. При этом период будет ограничен сверху количеством различных ненулевых кодовых слов длиной, равной количеству ячеек памяти регистра. В данном случае у нас три ячейки, поэтому максимально возможный период равен $2^3-1=7$. Нулевую комбинацию исключают, так как если регистр обнулить и подавать на него ТИ, то он не изменит своего состояния (нуль-петля).

По виду найденной выше выходной последовательности можно определить, что ее период равен семи и совпадает с максимальным. Подтвердим этот факт прогоном схемы по тактам (табл. 2). В начальный момент времени регистр, как всегда, обнулен.

Табл. 2. Прогон регистра по тактам (рис. 10) 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 опорных точек, определяющих состояние регистра, взяты входы ячеек памяти. Этим точкам соответствуют третий, четвертый и пятый столбцы таблицы (выделены серым фоном; это — независимые столбцы). Остальные столбцы являются зависимыми (вспомните два уравнения при выводе коэффициента передачи), и приведены для удобства и наглядности.

Из табл. 2 следует, что биты последнего столбца совпадают с коэффициентами разложения $K(x)$ в ряд Тейлора.

К сведению: если обратные связи в r -разрядном регистре сдвига введены так, что регистр при подаче на его вход дельта-последовательности выдает последовательность с максимальным периодом, то такое устройство становится генератором *псевдослучайной последовательности (M-последовательности, последовательности максимальной длины)* с периодом $M=2^r-1$ (двойка в основании является следствием выбора двоичного кода; в общем случае максимальный период равен m^r-1).

Во всех выше рассмотренных вариантах деления полиномов последовательности подавались **младшим** коэффициентом вперед, поэтому на выходе получалась последовательность, соответствующая разложению коэффициента передачи в ряд Тейлора по степеням x (как вы уже, наверное, догадались, в противоположном случае будет соответствие с разложением по степеням

x^{-1}). Однако для кодирования **систематическим** циклическим кодом необходим **остаток от деления** полинома $x^r a(x)$ на производящий полином $g(x)$

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

Известно, что остаток от деления можно найти путем деления в столбик. Эта процедура начинается со **старшей** степени делимого, при этом делимое шаг за шагом уменьшает свою степень до тех пор, пока не станет строго меньше степени делителя.

Пойдем от частного к общему и рассмотрим остатки от деления степеней x^i для некоторых $i=0,1,2,3,4$ на порождающий полином $g(x)$. Остатки обозначим как $res_i(x)$:

$g(x)=x^3+x^2+1$, (под суммой как всегда понимаем сумму по модулю два)

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

$res_3(x)=x^3 \bmod g(x)=x^2+1$ 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.

Замечаем **рекурсию**

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

Остатки рано или поздно начнут повторяться с некоторого n , поэтому $res_n(x)$ — это периодическая функция дискретного аргумента n .

Известно, что остаток от деления суммы равен сумме остатков, поэтому остаток от деления любого полинома выражается в виде линейной комбинации тривиальных остатков $res_i(x)$. Более того, остатки $res_i(x)$ выражаются в виде линейной комбинации базисных

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

Значит, чтобы с помощью регистра сдвига вычислить остаток от деления, входную последовательность необходимо подавать **старшим коэффициентом вперед.**

Вывод: порядок следования входных и выходных коэффициентов задается исходя из назначения схемы, и именно этим порядком определяется коэффициент передачи ячейки памяти.

Преобразуем схему на рис. 10 так, чтобы старшие коэффициенты шли впереди, но чтобы знаменатель коэффициента передачи при этом не изменился (рис. 11)

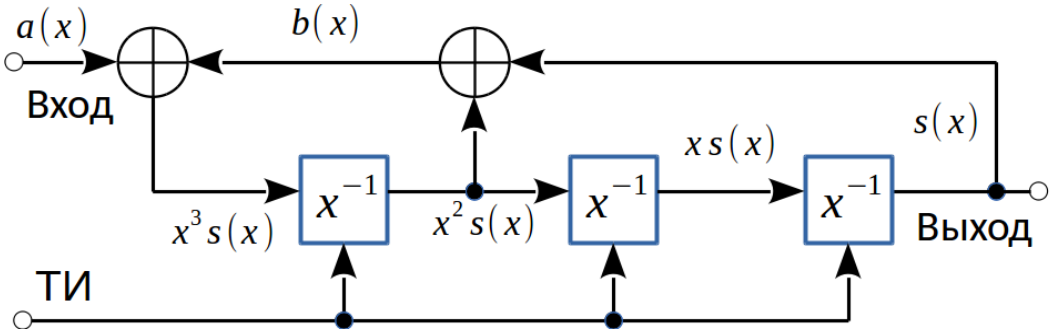


Рисунок 11 Трехразрядный регистр сдвига с обратными связями

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

Теперь задержка в регистре на три такта, поверьте, будет означать гарантированное равенство нулю частных от деления полиномов $1, x$ и x^2 на полином-знаменатель.

Если на рассматриваемую схему подать дельта-последовательность $a(x)=1$, то на выходе будет последовательность, отвечающая за частные от деления 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. Подтвердим этот факт таблицей (табл. 3). Результаты деления степеней x на полином x^3+x^2+1 в столбик содержатся в табл. 4.

Поставим соответствие между двумя таблицами.

Например, седьмой строке табл. 4 (выделена серым) соответствует частное 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 в рассматриваемом частном не будет.

Табл. 3. Результат потактовой работы регистра, изображенного на рис. 11

$a(x)$	$b(x)$	$x^3 s(x)$	$x^2 s(x)$	$x\,s(x)$	$s(x)$
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. Результаты ручного деления 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 — результат деления в столбик и результат деления с помощью регистра — совпадают, чего не скажешь про остатки от деления, которые требуются при кодировании и декодировании циклических кодов: сравните третий, четвертый и пятый столбцы табл. 3 с остатками табл. 4.

Остатки от деления должны храниться в ячейках памяти регистра. Схема на рис. 11 не дает нам остатки в чистом виде, так как обратная связь организована не только с выхода, но и с промежуточных ячеек. Рекурсивная процедура нахождения остатков (12) требует подачи обратной связи **только с выхода.**

Формула $res_n = res_{n-1} + res_{n-3}$ применительно к регистру сдвига означает то, что последний бит (крайний правый, с индексом n) переходит в предпоследний (с индексом $n-1$) и в бит с индексом $n-3$.

Степень порождающего полинома x^3+x^2+1 равна трем, поэтому регистр будет состоять из трех ячеек, входные биты которых можно интерпретировать как коэффициенты полинома-остатка от деления.

Согласно рекурсии, порождаемой полиномом x^3+x^2+1, бит на выходе последней ячейки должен перейти в третий и первый. Например, регистр с состоянием $001(0)$ — что соответствует остатку деления x^2 на $g(x)$ — на следующем такте перейдет в устойчивое состояние $101(1)$ — 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 получается из-за наличия четвертой единицы $000(1) \rightarrow 101(1)$. В итоге, остаток от деления x^4 будет равен

$$010+101=111 .$$

Преобразуем регистр на рис. 11 так, чтобы сумматоры располагались непосредственно между ячейками памяти (рис. 12).

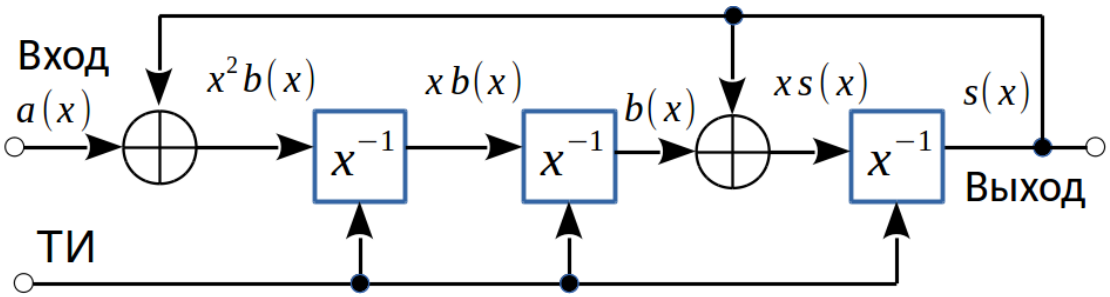


Рисунок 12 Трехразрядный регистр сдвига с обратными связями. Явно вычисляет остатки от деления входного полинома на порождающий полином.

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

Результат потактовой работы рассматриваемого регистра сведен в 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. Анализируя «ручные» остатки табл. 4 и три столбца табл. 5, выделенные серым фоном, отмечаем совпадение. Старшие коэффициенты остатков в табл. 5 находятся справа, так как старшие согласно нашего выбора идут впереди.

Табл. 5. Результат потактовой работы регистра на рис. 12 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

Вывод: схема, пригодная для явного вычисления остатка от деления, содержит сумматоры непосредственно между ячейками памяти. Старшие коэффициенты входных и выходных полиномов располагаются правее

(поступают на вход и выход первыми), младшие — левее (входят и выходят последними).

Подводя итог, составим схему систематического кодера, основываясь на схеме рис. 12 и правиле кодирования (11), которое для выбранного кода (7, 4) будет записано в виде

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

Слагаемые полинома $a(x)$ могут иметь степени 0, 1, 2 и 3, так как число входных символов равно четырем. Тогда степени слагаемых полинома $x^3a(x)$ могут быть 3, 4, 5 и 6, поэтому для кодирования необходимо знать остатки от деления лишь следующих степеней: 3, 4, 5 и 6. Чтобы сэкономить три такта, входные биты сразу подаются на выход регистра.

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 (рис. 13).

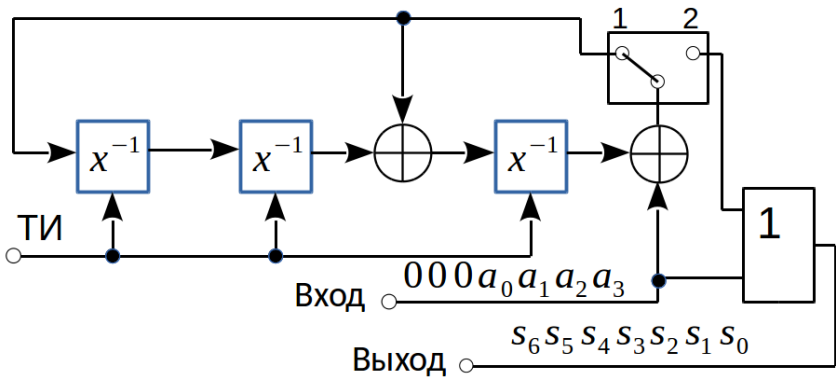


Рисунок 13 Схема систематического кодера для кода с порождающим полиномом $g(x)=x^3+x^2+1$

Информационные биты одновременно участвуют как в вычислении остатка, так и в поступлении на выход кодера через логическую схему **ИЛИ** (которая на схеме обозначена символом **1**).

Для вывода найденного остатка (контрольной суммы) переключатель переводится в положение 2. Вывод остатка происходит за три такта, поэтому на весь процесс кодирования отводится семь тактов. Три нуля, стоящие слева

от входных символов, введены для того, чтобы не исказить вычисленный остаток от деления (ноль на сумму не влияет).

3. Порядок выполнения работы

1. Определить полином двоичного кода, дающий генератор последовательности максимальной длины (*M*-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 доказательство, подобно рис. 10 и табл. 2. Обозначить период.

2. Составить схему **несистематического** кодера двоичного циклического кода для определенного в п.1 полинома. Привести таблицу и рисунок, подобно табл. 1 и рис. 8, соответственно. Входное слово a задать случайным ненулевым.

3. Найти **все** остатки от деления целых степеней $x^i,\ i=0\ldots14,$ на найденный в п.1 порождающий полином. Остатки найти методом деления в столбик, приведя таблицу, подобную табл. 4, а также непосредственно по схеме деления, подобно рис. 12 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 определенного в п.1 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. Основываясь на схеме рис. 13, привести схему систематического кодера для определенного в п.1 полинома. Составить таблицу кодирования случайно выбранного входного слова с весом $w>2$. Подтвердить правильность кодирования алгебраически по (11).

6. Вычислить с помощью схемы и методом деления в столбик остаток для найденного в п.5 разрешенного слова систематического кода. Должен ли он быть равен нулю? Если да, то что это означает?

4. Вопросы

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

2. Чему равен период последовательности, генерируемой троичным* трехразрядным регистром сдвига с линейными 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 использовать программу компьютерной алгебры. Выписать найденную последовательность, соответствующую одному периоду.

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

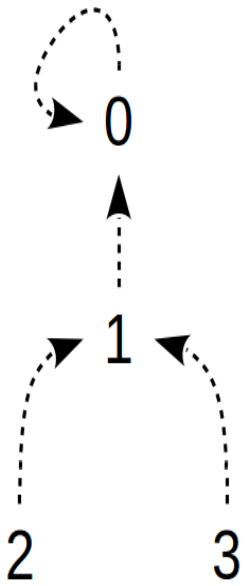
3. Приведите пример порождающего полинома третьей степени с троичным алфавитом, дающего максимальный период.

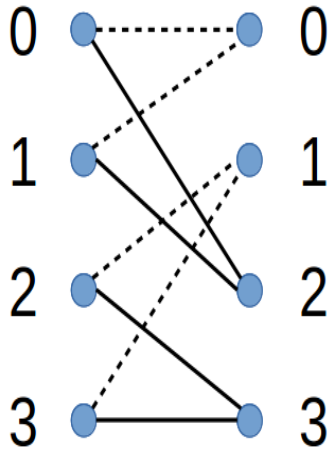
5. Литература

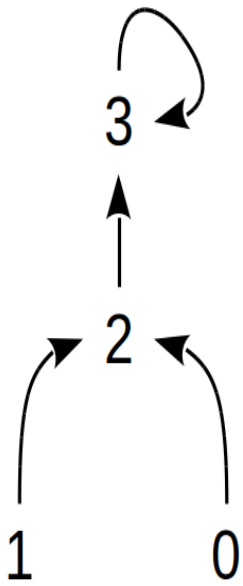
1. Триггер, http://ru.wikipedia.org/wiki/%D2%F0%E8%E3%E3%E5%F0.
2. 3BIT/CELL MLC NAND FLASH, http://www.samsung.com/global/business/semiconductor/minisite/SSD/uk/html/about/MlcNandFlash.html.
3. SANDISK SHIPS WORLD'S FIRST FLASH MEMORY CARDS WITH 64 Gigabit X4 (4-BITS-PER-CELL) NAND FLASH TECHNOLOGY, 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

СВЕРТОЧНЫЕ КОДЫ







1. Введение

Сверточные коды нашли широкое применение в современных цифровых системах радиосвязи и радиовещания [1]. Акцент на *радио* сделан не зря. Качество радиоканалов зачастую уступает качеству проводных каналов (и, тем более, оптических), поэтому в цифровых радиосистемах передачи и приема информации применяются мощные коды с большой избыточностью, в частности, изучаемые здесь сверточные коды.

Любое помехоустойчивое кодирование (канальное кодирование) вносит *избыточность* в передаваемое сообщение. В качестве примера можно привести принцип записи денежных сумм прописью. В этом случае вероятность сделать ошибку меньше, чем при считывании и распознавании цифр, так как буквы связаны между собой смысловым контекстом, а цифры — нет. При этом качество записи всех символов (цифр, букв) должно быть одинаковым, иначе сравнение кодов по помехоустойчивости будет некорректным.

Избыточность, введенная по заранее известным правилам, позволяет обнаруживать и исправлять некоторые ошибки.

Целью помехоустойчивого кодирования является снижение вероятности ошибки после декодирования. Вероятность ошибки также можно снизить, например, увеличивая мощность передатчика и/или увеличивая длительность импульса. Однако, такой грубый подход, в итоге, снизит общую эффективность системы связи. Зачастую введение кодирования дает больший эффект, и, например, такого же снижения вероятности ошибки можно добиться гораздо меньшим увеличением мощности, чем при грубом подходе (плата за кодирование — дополнительные вычислительные ресурсы, которые сейчас гораздо дешевле такого ресурса как, например, мощность СВЧ-передатчика, сводимая к стоимости мощного выходного транзистора).

Процесс кодирования сверточными кодами очень простой, в отличие от процесса декодирования. Циклические коды против сверточных в этом отношении являются симметричными.

Сверточное кодирование выполняется с помощью *регистра сдвига* (последовательно соединенных ячеек памяти) и набора *сумматоров* (логических схем). В данной работе изучаются двоичные сверточные коды со скоростью кодирования ½ в связи с их базовостью.

Сверточные коды имеют одну особенность — **память,** которая приводит к тому, что перед декодированием текущего бита необходимо дождаться приема нескольких следующих. Образно говоря, декодер должен заглядывать в будущее. Такой интересный способ декодирования является следствием наличия памяти, которая приводит также к тому, что сверточные коды полностью задаются диаграммой состояний, которая показывает все разрешенные переходы. Запрещенные же переходы позволяют бороться с некоторыми ошибками.

Существование разрешенных и запрещеных переходов влечет за собой существование *разрешенных* и *запрещенных* кодовых последовательностей. В отличие от кодовых слов линейных блочных кодов, кодовые последовательности идут непрерывным потоком взаимосвязанных между собой символов. В связи с этим, сверточные коды не являются блочными.

Отчет должен включать:

- ✔ Титульный лист;
- ✔ Цель работы;
- ✔ Ход работы;
- ✔ Выводы.

2. Сведения из теории

2.1 Кодирование

Сверточные коды относятся к помехоустойчивым линейным кодам. Избыточность вносится так, что одному входному биту a соответствует n выходных битов

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

*Выходное кодовое слово $b^{(1)} b^{(2)} b^{(3)} \dots b^{(n)}$ удобно интерпретировать в виде числа на отрезке $\left[0, 2^{n-1}\right]$.

Так как сверточные коды являются линейными, то для формирования выходного слова подходят лишь **линейные** по отношению к входному биту операции.

В простейшем (вырожденном) случае входной бит копируется на все позиции выходного слова (такой код памяти не имеет). Однако, если в линейных операциях учесть некоторое количество предыдущих входных битов (ввести память), то вариантов кодирования становится больше: разным позициям выходного слова могут соответствовать *значения разных линейных комбинаций* битов, хранящихся в регистре памяти.

Правило кодирования сверточным кодом задается набором *порождающих полиномов,* каждый из которых отвечает за конкретный выходной бит.

Рассмотрим сверточный код с соответствием $a \to b^{(1)} b^{(2)}$, согласно которому на один входной бит приходится два выходных. В связи с этим, *скорость кодирования* такого кода равна ½. Возьмем, например, два следующих порождающих полинома

$$G_1(x)=1 \quad , \quad G_2(x)=1+x \quad . \tag{14}$$

Последовательности входных битов a_i поставим в соответствие полином[3]

3 Математически удобно индекс суммирования ничем не ограничивать

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

где символом $\mathbb{Z}$ обозначено множество целых чисел.

Полиномы, соответствующие выходным последовательностям, определим как результат умножения входного полинома на соответствующие порождающие

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

На основании (15) определим правило работы кодирующего устройства (кодера) во временной области

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

Известно, что коэффициенты полинома, равного произведению двух полиномов, вычисляются через *свертку* коэффициентов перемножаемых полиномов[4]. Для доказательства этого факта требуется лишь знание правил перемножения двух полиномов и приведения подобных слагаемых. Также данный факт известен тем, кто знаком с азами теории z-преобразования.

Если для изучаемого сверточного кода сконструировать *порождающую матрицу,* которая есть у любого *линейного блочного кода,* то окажется, что она будет бесконечной

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

поэтому сверточные коды не являются блочными (они являются поточными).

Функциональная схема кодера, реализующего кодирование по (15), изображена на рис. 14. Блок **MUX** является мультиплексором, выходные биты которого являются результатом чередования входных. Блок x — это триггер (ячейка памяти), хранящий на своем выходе предыдущий входной бит

4 Отсюда и название изучаемых кодов — сверточные

a_{i-1} . Сумматор по модулю два обозначен окружностью со знаком +. Он имеет два входа и один выход. Таблица сложения по модулю два простая и по сути является операцией вычисления остатка от деления на два

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

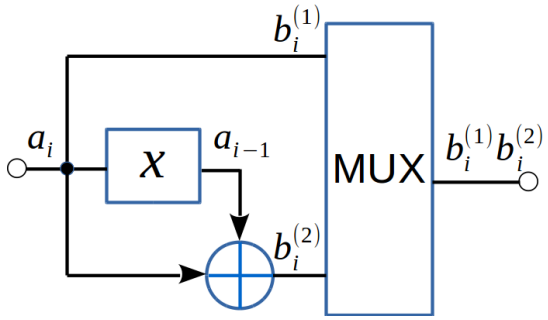
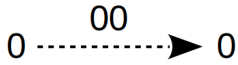


Рисунок 14: Схема сверточного кодера ½

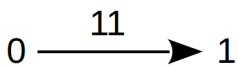
Выходное слово кодера однозначно определяется текущим входным битом и набором предыдущих, хранящихся в регистре сдвига. Предыдущие биты образуют *вектор состояния* кодера, который удобно записывать в виде соответствующего десятичного числа. В рассматриваемом примере вектор состояния состоит из единственного бита a_{i-1} , которому можно поставить в соответствие два числа — 0 и 1.

Поясним принцип построения *диаграммы состояний* на примере рассматриваемого сверточного кода.

Общепринято, что в начальный момент времени кодер находится в нулевом состоянии. Если при этом на вход поступил бит **0,** то на выходе будет кодовое слово **00.** В момент прихода следующего бита кодер перейдет в состояние **0,** так как до этого на входе был **0.** Данный расклад можно символически изобразить веткой, в которой штриховая линия обозначает приход нулевого входного бита, число слева обозначает текущее состояние кодера (состояние в момент выдачи выходного слова **00,** которое указано над линией). Число на конце стрелки указывает на будущее состояние кодера.



Определим ветку, если на вход поступил бит **1,** а кодер при этом находится в состоянии **0.** Очевидно, что выходное слово будет равно **11,** а будущее состояние — **1.** В данном случае будущее состояние определяется только текущим входным битом, так как ячейка



памяти одна; в общем случае будущее состояние зависит и от некоторых предыдущих битов. Здесь линия сплошная, так как на входе 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.**

Отображая все возможные состояния кодера в виде прямоугольных блоков с надписями внутри, и соединяя их линиями со стрелками, можно увидеть *диаграмму состояний* (рис. 15).

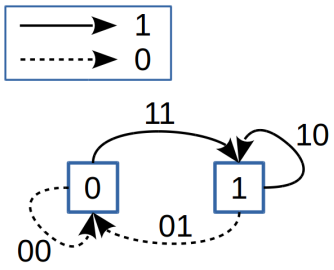


Рисунок 15: Диаграмма состояний сверточного кодера ½

Если кодер переходит в исходное состояние, например, из **1** в **1,** то получается *петля* (*цикл*). К такому циклу приводит кодирование последовательности из всех единиц, что в установившемся режиме даст периодическую выходную последовательность кодовых слов (иногда кодовые слова называют *кодовыми символами*)

...10 10 10....

Процесс кодирования по диаграмме состояний очень прост и нагляден.

Для начала подадим на вход кодера дельта-последовательность

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

отклик на которую является *импульсной характеристикой* (ИХ)

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

Два первых ненулевых слова (**11** и **01**) ИХ говорят о наличии у кода памяти (один входной бит дает два кодовых символа). Индекс n в ИХ h_n отвечает **за пару битов (за кодовое слово).** Избыточность кода равна 50%, так как на один входной бит приходится два выходных.

Теперь рассмотрим более сложную входную последовательность

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

которую согласно диаграмме состояний можно закодировать следующей последовательностью кодовых слов

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

Этот же результат можно получить иным способом, учтя, что отклик кодера на сумму входных последовательностей равен сумме откликов[5], и что любую входную последовательность можно выразить через сумму дельта-последовательностей, отличающихся лишь задержкой

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

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

Не забываем про суммирование коэффициентов по модулю два.

- Изобразить схему сверточного кодера с генераторными полиномами $G_1(x)=1$, $G_2(x)=1+x+x^2$, а также диаграмму состояний кода.
- Изучив параграф 2.2 Декодирование по Витерби, на основании диаграммы состояний построить решетку кода до момента ее стабилизации.

5 Это следует из равенства: $A(x)[B(x)+C(x)]=A(x)B(x)+B(x)C(x)$

2.2 Декодирование по Витерби

Логика декодирования сверточных кодов гораздо интереснее логики кодирования.

При декодировании удобно использовать решетку кода (рис. 16), которая является результатом развертки во времени его диаграммы состояний.

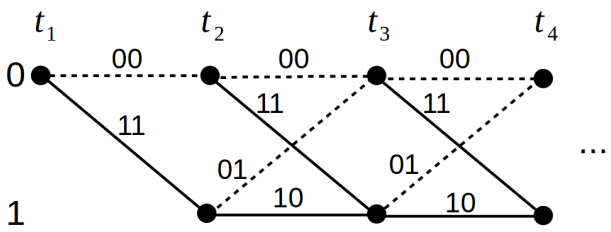


Рисунок 16: Решетка сверточного кода ½

Процесс построения решетки начинается с записи в столбец всех возможных состояний кодера. В нашем случае состояний два — это **0** и **1.**

Кодер (и декодер) стартует с нулевого состояния, поэтому от нулевого состояния отводятся две линии согласно диаграмме состояний, в результате чего получается второй столбец (момент времени t_2).

От тех состояний второго столбца, которые задействованы входящими линиями предыдущего шага, согласно диаграмме состояний отводятся по две выходящие линии. В результате получается третий столбец (момент времени t_3). Данный процесс повторяется до тех пор, пока не будут задействованы все состояния. Последний построенный элемент (со всеми задействованными состояниями) затем будет повторяться (копироваться).

Число отводимых линий на каждом шаге удваивается вплоть до 2^{r+1} , где r — число ячеек памяти кодера (в нашем случае ячейка одна).

Решетка отображает эволюцию сверточного кодера во времени, показывая все разрешенные переходы. Значит, по решетке кода можно выписать все разрешенные кодовые последовательности (последовательности кодовых символов).

Декодируем по решетке отрезок безошибочной кодовой последовательности

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

Так как ошибок заведомо нет, то декодирование осуществляется очень просто. Начинаем с нулевого состояния решетки. На вход декодера пришло слово **11.** По какому пути пошел кодер, когда выдал это слово? Естественно, по пути из **0** в **1.** Линия на решетке сплошная, значит первый декодированный бит будет **1.** Дальше принимаем слово **10,** что соответствует переходу из **1** в **1** в моменты t_2 и t_3 на решетке. Декодируем второй бит как **1.** И так далее, что даст отрезок 110100, совпадающий с исходным.

Если нет уверенности в том, что входная последовательность безошибочная (а помехоустойчивые коды нет смысла использовать в каналах без ошибок), то зачастую используется метод декодирования *по Витерби* [2], дающий максимально правдоподобную оценку исходной последовательности.

Декодируем по Витерби отрезок кодовой последовательности с ошибкой в первом бите третьего слова (такой отрезок последовательности не ложится на решетку и поэтому будет запрещенным)

$$b = 11\ 10\ \mathbf{1}1\ 11\ 01\ 00\ 00\ .$$

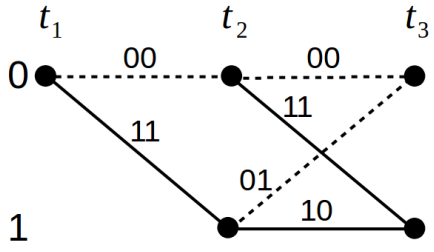


Рисунок 17: Первый шаг декодирования — нашли две пары путей

Начинаем, как всегда, с нулевого состояния. Строим решетку до тех пор, пока в каждое состояние не будет входить два пути (рис. 17). Так как сначала решетка не установилась, то сперва потребуется несколько шагов для формирования всех пар входящих путей. Затем для достраивания решетки будет достаточно одного шага.

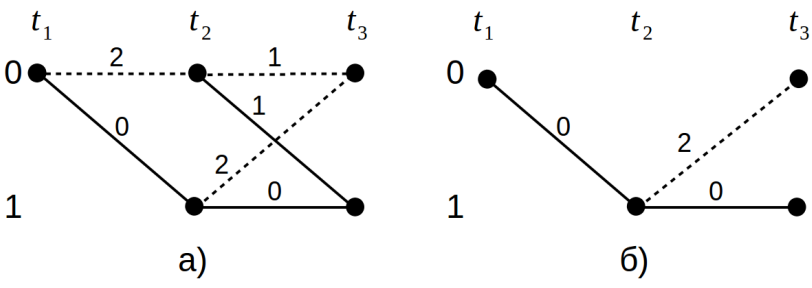


Рисунок 18: Первый шаг декодирования — исключили из каждой пары путей пути с наибольшей суммарной метрикой

Теперь, когда имеются по два пути, входящих в состояния **0** и **1** (рис. 18а), необходимо оставить

ровно по одному, предварительно пометив эти пути числами, равными расстоянию между принятым кодовым словом и соответствующим словом решетки.

Первый путь из **0** в **0** (верхняя линия) соответствует выходной последовательности кодера 0000, которая отличается от принятой последовательности 1110 тремя битами, поэтому *метрика* первого пути равна трем (2+1).

Второй путь из **0** в **0** (V-образная линия) имеет метрику два (0+2), что меньше метрики первого пути. Значит первый путь, как менее вероятный, вычеркиваем, а V-образный — оставляем.

Для путей, входящих в состояние **1**, метрики равны 0 и 3 (рис. 18а). Оставляем L-образный путь, **объединяя** его с V-образным (рис. 18б).

В результате осталась решетка (рис. 18б), в которой имеется только одна линия между моментами времени t_1 и t_2, что позволяет однозначно декодировать бит как **1.** Теперь элемент решетки, соответствующий моментам t_1 и t_2, можно удалить, так как он никак не влияет на процесс дальнейшего декодирования.

Вывод: основная цель декодирования по Витерби состоит в определении **наиболее вероятного** пути.

Чтобы декодировать второй бит, к оставшейся решетке следует добавить **эталонный элемент** (тот, который будет постоянно одним и тем же, рис. 17, между t_2 и t_3), вычислить метрики всех пар входящих путей и для каждого состояния оставить по одному пути с наименьшей метрикой. Если после всего этого между моментами t_2 и t_3 останется более одной линии, то текущий бит декодировать нельзя. В этом случае следует добавить еще один эталонный элемент и опять проделать операцию исключения менее вероятных путей. Добавление каждого элемента решетки требует приема следующего кодового слова. Весь этот процесс продолжается до тех пор, пока в решетке между моментами t_2 и t_3 не останется единственной

линии. Если кодовые слова по каким-то причинам заканчиваются*, то из всего множества путей выбирают единственный путь с минимальной суммарной метрикой. *При практических реализациях декодера Витерби ставят ограничитель на длину решетки, при достижении которого выбирается единственный путь, а решетка сбрасывается. Также может закончится логический кадр, так как сверточные коды искусственно ограничивают блоками сравнительно большой длины, чтобы была возможность независимого декодирования этих блоков — это дополнительная степень свободы.

Казалось бы, а почему сразу не декодировать второй бит, выбрав нижний путь на рис. 18б, ведь у него метрика нулевая, в отличии от диагонального пути с метрикой два? Если мы так сделаем, то не используем весь потенциал кода (не высосем кость до конца). За счет памяти кода следующее (третье) слово несет информацию о втором бите, и это надо использовать.

В итоге, к решетке добавляется эталонный элемент, что гарантирует наличие двух сливающихся путей для каждого из (крайних правых) состояний (рис. 19а). Дальше из каждой пары путей исключается путь с наибольшей метрикой (рис. 19б), и второй бит однозначно декодируется как **1**.

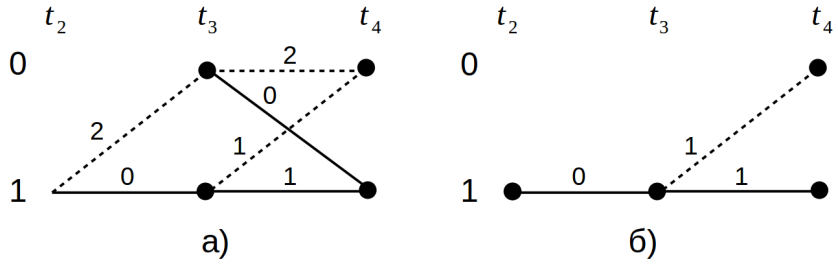


Рисунок 19: Второй шаг декодирования — декодировали второй бит

Между моментами t_3 и t_4 нет пути с нулевой метрикой (рис. 19б). Это напрямую связано с наличием ошибки в третьем принятом слове (в результате чего слово **01** заменилось на **11**).

На третьем шаге декодирования принимаем **четвертое** слово **11.** За счет памяти кода оно несет информацию о третьем бите. Рассмотрим путь из **1** в **1** (рис. 20а).

Λ-образный путь соответствует разрешенной последовательности 0111, что дает метрику $1+0=1$, а горизонтальный путь — последовательности 1010 и метрику $1+1=2$. Очевидна победа первого

пути. Путь из **1** в **0** предпочтительнее пройти как $1 \to 1 \to 0$ с метрикой 2, что меньше метрики 3 пути $1 \to 0 \to 0$. Таким образом, неопределенность относительно третьего бита не разрешилась.

Достраиваем решетку до момента времени t_6 (рис. 20б) по **пятому** принятому слову **01.**

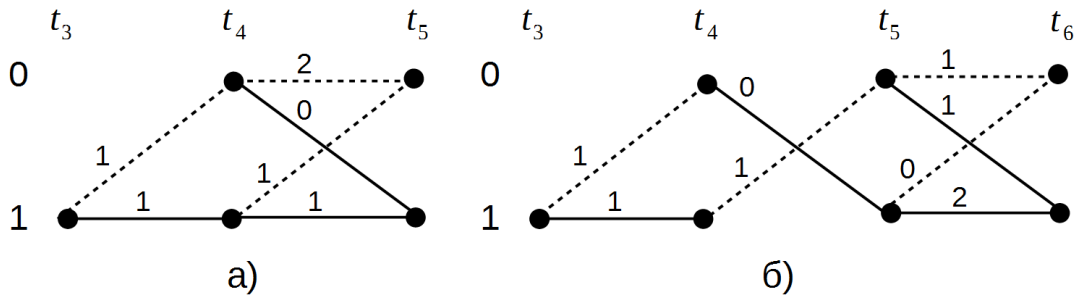


Рисунок 20: Третий шаг декодирования — третий бит не однозначен

Анализ рис. 20б дает интересный результат: победителем пути в состояние **0** является *N*-образный путь (с метрикой $1+0+0=1$), поэтому между моментами t_3 и t_4 остается штриховая линия, которая обозначает бит **0**, но еще есть путь в **1** — он также даст свой вес, который игнорировать нельзя. Метрики обоих путей в состояние **1** оказались одинаковыми

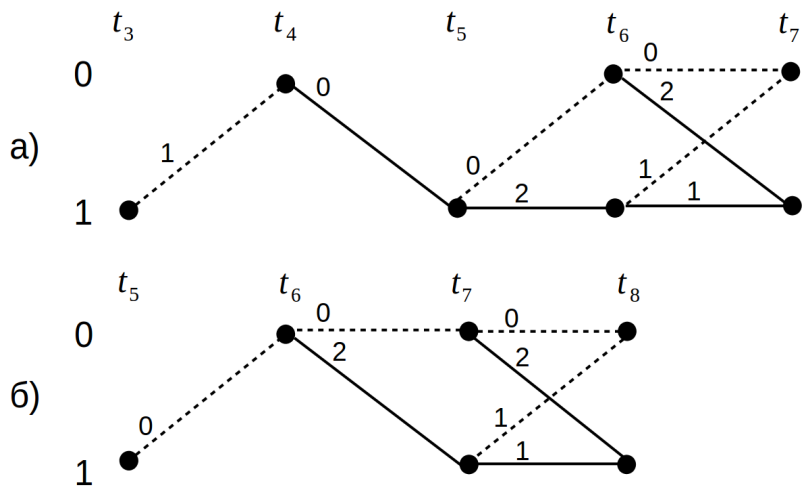


Рисунок 21: Третий шаг декодирования — декодированы третий и четвертый биты

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

поэтому выбираем любой путь. Выгоднее выбрать тот, который исключает все еще имеющуюся неопределенность относительно третьего бита — это путь $1 \to 0 \to 1 \to 1$ с метрикой $1+0+2=3$. В результате, неопределенность разрешилась сразу относительно двух битов: третьего **(0)** и четвертого **(1)** (рис. 21а).

Достраиваем решетку между t_6 и t_7 на основании предпоследнего принятого слова **00.** Исключаем лишние пути и достраиваем решетку по последнему слову **00** (рис. 21б). Декодируем пятый бит как **0.** После дальнейшего исключения лишних путей декодируем шестой бит как **0.**

Таким образом, декодированная последовательность совпала с исходной: ошибка была исправлена.

Важно! Если взять неограниченно длинную решетку рассматриваемого кода и по ней выписать все разрешенные последовательности, то, как ни крути, между собой они будут отличаться как минимум тремя битами. Это говорит о том, что свободное расстояние данного кода равно трем, что, в свою очередь, говорит о возможности исправления всех однократных ошибок.

2.3 Пороговое декодирование

Метод порогового декодирования гораздо проще метода декодирования по алгоритму Витерби. Рассмотрим метод порогового декодирования на примере систематического сверточного кода (14).

Так как код систематический, то выходные биты кодера попарно чередуются ИП ИП ИП…, где символом И обозначен информационный бит, а П — проверочный, равный линейной комбинации некоторых информационных.

Пороговый декодер, аналогично кодеру, вычисляет проверочную последовательность на основании принятой из канала информационной.

Обозначим информационный бит как a_i, а соответствующий ему проверочный как b_i. Поместим часть этих битов в табл. 1. 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 пояснению порогового декодирования сверточного кода

Жирным шрифтом выделены информационный бит a_3 и зависящие от него проверочные

$$b_4 = a_4 + a_3 \quad , \quad b_3 = a_3 + a_2 \quad .$$

Если ошибки в канале отсутствуют, то биты b_3 и $b_3^{(d)}$, b_4 и $b_4^{(d)}$ попарно совпадут (обратное в общем случае неверно). Декодер в зависимости от количества несовпадений либо корректирует текущий информационный бит, либо нет. Если декодер сделал коррекцию бита, то проверочные символы, которые от него зависят, пересчитываются.

Как определить порог, то есть то минимальное количество несовпадений, при превышении которого бит корректируется?

Как правило, порог определяется исходя из минимума вероятности ошибки после декодирования при малой вероятности ошибки в канале (системы связи при больших вероятностях ошибки не работают). Для рассматриваемого кода количество несовпадений является дискретной случайной величиной на множестве значений $\{0, 1, 2\}$.

Так как сверточный код имеет память, то, например, ошибка декодирования бита a_2 повлияет на вероятность правильного декодирования бита a_3. Этот факт является очень важным и вскрывает неблочную структуру сверточных кодов (в блочных кодах результат декодирования текущего кодового слова не влияет на результат декодирования следующего).

В данной работе считается, что каждый бит (**0** и **1**) может быть распознан демодулятором (искажен каналом) ошибочно с вероятностью, не зависящей от результата распознавания предыдущих битов. Такой канал называется *двоичным симметричным каналом с независимыми ошибками.* Вероятность q-кратной ошибки в таком канале в блоке из n битов вычисляется по биномиальной формуле

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

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

p — вероятность битовой ошибки в канале.

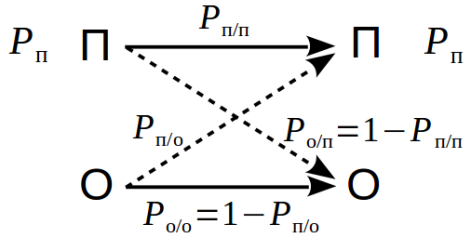


Рисунок 22: К определению вероятности правильного декодирования

Определим вероятность правильного декодирования $P_{п}$ бита пороговым декодером рассматриваемого сверточного кода. Для этого абстрагируемся от каких бы то ни было методов декодирования и представим себе следующее.

Пусть процесс декодирования характеризуется различными переходами между правильным декодированием и ошибочным (рис. 22), где буквой **П** обозначено событие правильное декодирование, а буквой **О** — ошибочное декодирование. Так как итоговая вероятность правильного декодирования теоретически определяется по бесконечному количеству принятых символов, то можно показать, что слева и справа на рис. 22 в пределе стоит одинаковая величина — искомая вероятность правильного декодирования $P_{п}$, что, в итоге, дает следующее

$$P_{п}=\frac{P_{п/о}}{P_{п/о}-P_{п/п}+1}\quad . \tag{17}$$

После детального анализа всех возможных состояний порогового декодера 😡 можно 😭 найти формулы для переходных вероятностей $P_{п/о}$ и $P_{п/п}$ для канала (16), и из (17) получить следующий результат

$$P_{п}(p)=\frac{(2p^3-2p^2+1)(4p^4-8p^3+8p^2-5p+2)}{16p^6-52p^5+72p^4-50p^3+19p^2-5p+2}\quad . \tag{18}$$

Графически удобнее анализировать вероятность ошибки

$$P_{о}(p)=1-P_{п}(p)=\frac{-p^2(2p^2-4p+3)(4p^3-12p^2+12p-5)}{16p^6-52p^5+72p^4-50p^3+19p^2-5p+2}\approx\frac{15}{2}p^2\quad ,$$

построив ее график в логарифмическом масштабе (рис. 23). Приближение сделано для $p\ll 1$.

Анализ найденной формулы показывает, что если вероятность ошибки до декодирования, допустим, была $p=0{,}001$, то после декодирования частота ошибок уменьшится в 100 раз, что говорит о наличии выигрыша, даваемого сверточным кодированием. Величина выигрыша, к сожалению, зависит от качества канала: например, для $p=0{,}01$ частота ошибок после декодирования уменьшится примерно в 10 раз, а не в 100.

Выигрыш обеспечивается за счет разумно введенной избыточности. Для изучаемого кода избыточность равна 50%. Вопрос: как регулировать избыточность в сторону менее 50%?

Для этого применяется *перфорирование* (**puncturing,** удаление некоторых

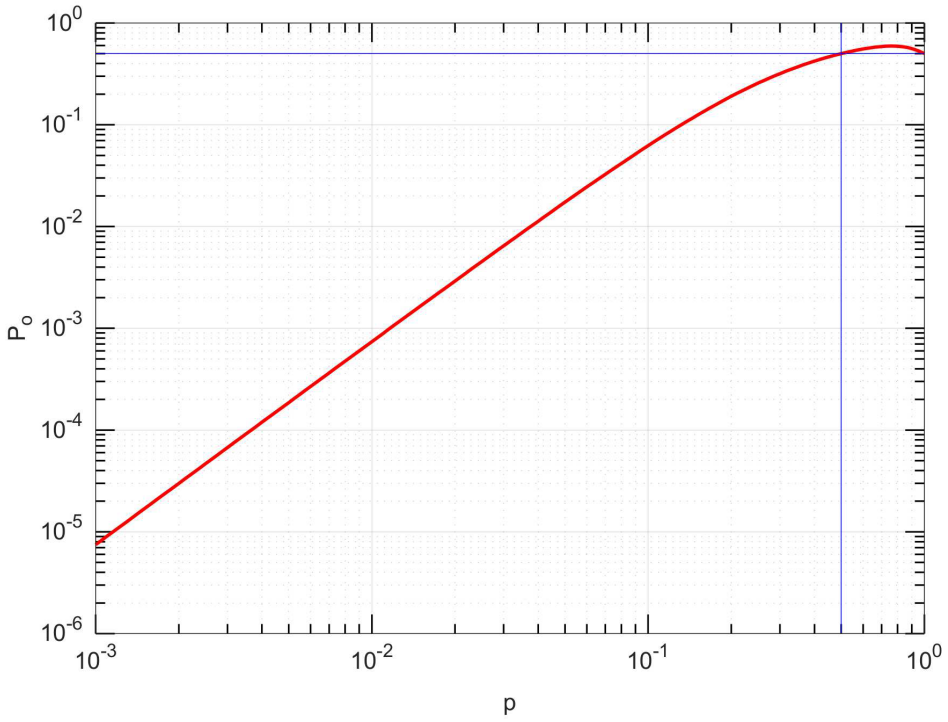


Рисунок 23: Частота битовых ошибок P_o после декодирования сверточного кода ½ в зависимости от частоты битовых ошибок p в двоичном симметричном канале связи с независимыми ошибками.

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 выходит за рамки данной работы.

Вообще, надо сказать, что коды со скоростью кодирования ½ являются базовыми и, в итоге, после вставки перфорированных битов в приемнике,

декодируются именно они. Причина этому — унификация декодеров (ядро декодера настроено на декодирование кодов ½).

3. Описание лабораторного макета

Лабораторный макет выполнен в виде консольного приложения (рис. 24) и предназначен для анализа сверточного кодирования и двух методов декодирования: порогового и по алгоритму Витерби. Приложение собрано с использованием библиотек **Qt** [3].

После запуска приложения **thr_svert** в консоль выводится заголовок и перечисляются коэффициенты полиномов сверточного кода, по умолчанию совпадающие с коэффициентами полиномов изучаемого кода (14). Если требуется иной набор полиномов, то следует отредактировать файл **polynoms.txt.**

Дальше по ходу вывода приложения предлагается ввести число входных битов кодера и вероятность битовой ошибки (в канале связи). Если выбран режим **Выводить битовые последовательности на экран,** то в консоль будут выведены последовательности во всех основных контрольных точках: на входах и выходах кодера и декодера. Такой режим имеет смысл включать при сравнительно небольшой длине отрезков последовательностей. Наконец, предлагается ввести величину порога для работы порогового декодера.

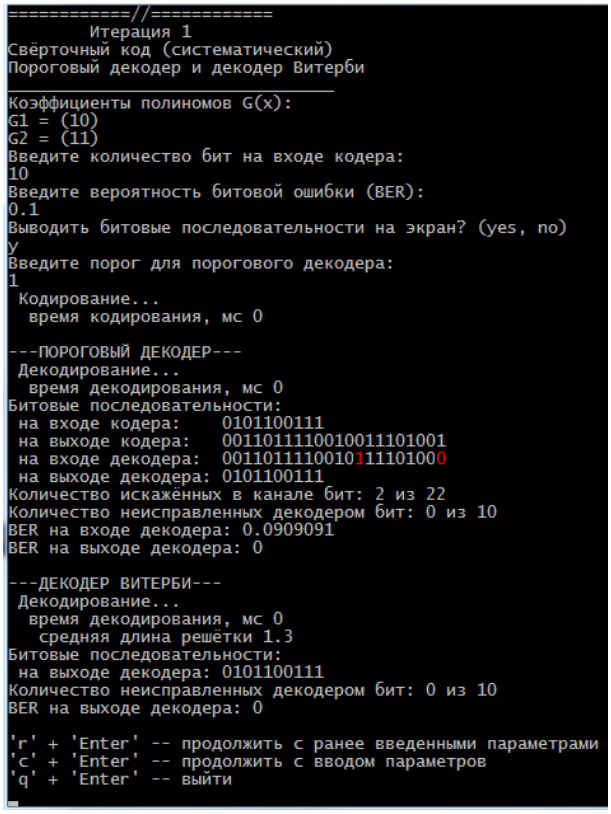


Рисунок 24: Консольная программа кодирования и декодирования систематического сверточного кода со скоростью кодирования ½

Входные биты задаются датчиком случайных чисел (равновероятно). Канальные ошибки вносятся датчиком случайных чисел с заданной пользователем вероятностью. При небольшом числе канальных битов (менее 20–50) заданная пользователем вероятность обеспечивается грубо, поэтому в любом случае в программе выводятся текущие оценки этих вероятностей (до декодирования и после). Для удобства ошибочные биты выделяются красным.

Пороговое декодирование в программе выполняется согласно рассмотренному ранее алгоритму. В частности, при превышении порога текущий бит корректируется.

В самый конец 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 могут существовать несколько наиболее коротких путей, из которых программа случайным образом выбирает один.

4. Порядок выполнения работы

1. Закодировать сверточным кодом (рис. 14) отрезок последовательности битов из приложения Б (согласно своего варианта).
2. Запустить программу **thr_svert** и ввести следующие параметры:

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

Последовательным нажатием клавиш `r` и `Enter`, добиться случая двух-трех канальных ошибок (двух-трех красных битов), которые исправляются **пороговым** декодером (чтобы на выходе декодера красных битов не было). Все отрезки битовых последовательностей при этом выписать в тетрадь (в отчет), **в том числе и для декодера Витерби** (вне зависимости от результата декодирования). Закодировать входную последовательность методом импульсных характеристик или по диаграмме состояний. Декодировать канальную последовательность, следуя Приложению А.

3. Выполнить предыдущий пункт при условии, что останется **хотя бы один** ошибочный бит при пороговом декодировании.
4. Декодировать по алгоритму Витерби канальные последовательности из пп. 2 и 3. Привести необходимые решетки декодирования. Сравнить ручные результаты с программными.
5. Сравнить методы порогового декодирования и по алгоритму Витерби при следующих порождающих полиномах (полином $G_1(x)$ всегда равен единице, так как код систематический)

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

*Пороги подобраны по минимуму вероятности ошибки после декодирования для малой вероятности ошибки в канале связи.

Вероятности битовых ошибок в канале 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 решетки* (то есть число элементов в ней; например, на рис. 20б длина решетки равна трем; элемент решетки — весь набор путей из текущего состояния в следующее).

Построить графики для вероятности ошибки после декодирования в зависимости от вероятности ошибки до. На всех графиках дополнительно привести теоретическую зависимость вероятности битовой ошибки после декодирования для кода Хемминга (7, 4)[6]

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

у которого избыточность 3/7. Все зависимости отобразить в логарифмическом масштабе.

6. Рассмотреть избыточный код с дублированием информационного бита $a_1 a_1 a_2 a_2 a_3 a_3 a_4 a_4 \ldots$, избыточность которого также, как и у рассматриваемого сверточного кода, равна 50%. Найти порождающие полиномы данного кода (это вырожденный случай сверточного кода). Сделать вывод об эффективности данного кода по вероятности ошибки после декодирования. Вероятность ошибки измерить с помощью программы **thr_svert,** откорректировав файл с коэффициентами полиномов. Канальные вероятности ошибок задавать по п. 5, порог задать равным нулю.

5. Контрольные вопросы

1. Является ли сверточный код блочным? Почему?
2. Что из себя представляет порождающая матрица сверточного кода?
3. Разумно ли используется избыточность 50%[7] в коде с дублированием входного бита?

---

6 Это помехоустойчивый линейный код без памяти (линейный блочный код)

7 В двоичном симметричном канале с независимыми ошибками

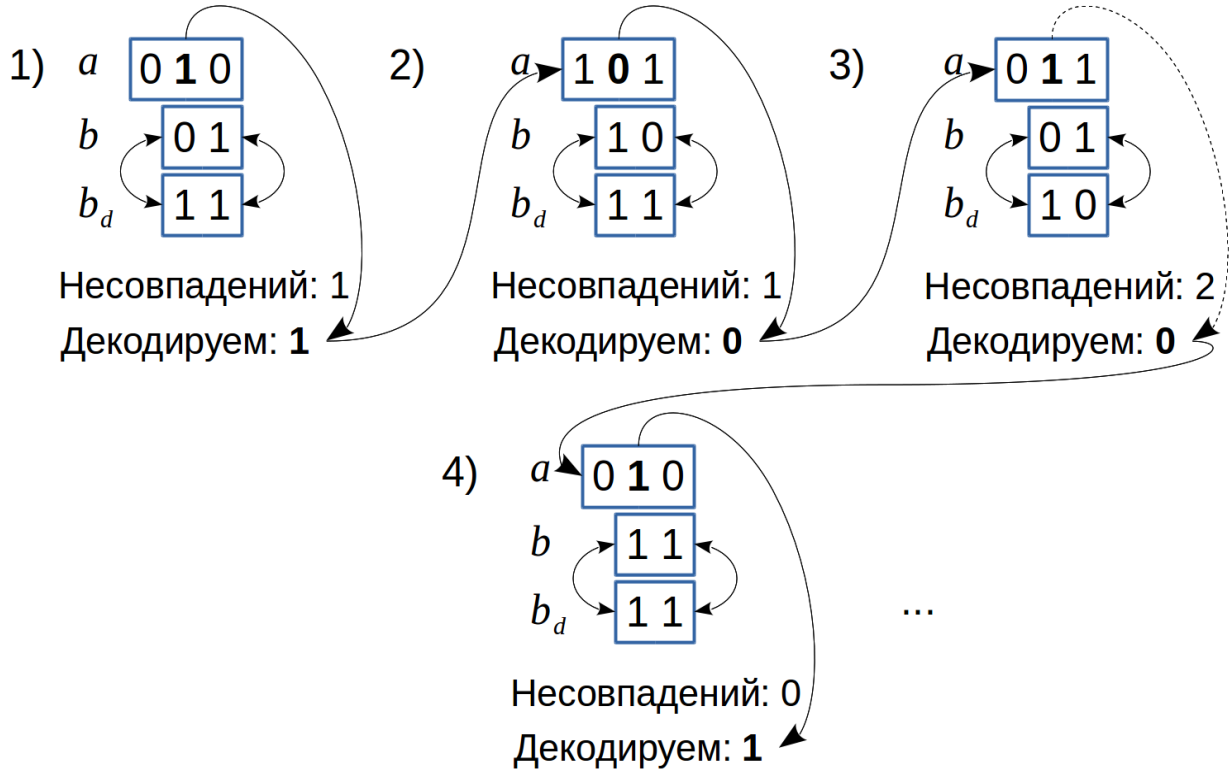
4. Какой декодер лучше по вероятности ошибки: пороговый или Витерби? В каких случаях? Какой декодер быстрее по скорости декодирования?
5. Какие коды лучше по вероятности ошибки: Хемминга или сверточные? В каких случаях?

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 сверточного 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 словах, причем в первом слове искажен проверочный бит, а в третьем и восьмом — информационные биты.

Приложение Б Отрезки последовательностей для сверточного кодирования

Номер варианта	Отрезок битовой последовательности									
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 ЧАСТОТНО-МАНИПУЛИРОВАННОГО СИГНАЛА

1. Введение

Частотная манипуляция (ЧМн) является нелинейным методом модуляции и содержит в себе много интересных моментов. В частности, имеется возможность формирования ЧМн-сигнала с непрерывной фазой, что дает право на когерентную демодуляцию с использованием решетки фаз. Использование такой решетки позволяет провести аналогию между декодированием сверточных кодов, например, по алгоритму Витерби. Также ЧМн-сигнал с непрерывной фазой имеет более компактную спектральную плотность[8] по сравнению, например, со спектральной плотностью сигнала с амплитудно-фазовой манипуляцией.

В данной лабораторной работе изучается некогерентный демодулятор ЧМн-сигнала, основанный на корреляторе. 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**. Программу можно (но необязательно) взять с сайта

http://www.tatsuromatsuoka.com/octave/Eng/Win/.

Также имеется возможность запустить макет в программе **Simulink** не ниже 2015b.

2. Основные сведения из теории

Частотной манипуляцией называют кодирование потока символов гармоническим сигналом с соответствующим переключением частоты (ЧМн-сигналом), при этом амплитуда сигнала (теоретически) не меняется, а начальная фаза — не определена (может быть любой).

Слово *кодирование* здесь понимается буквально, так как в настоящее время для формирования сигналов используется техника прямого цифрового синтеза **(DDS, Direct Digital Synthesis),** когда заданный сигнал формируется в

8 Под компактностью подразумевается не столько ширина спектра, сколько мера сосредоточенности мощности сигнала в заданном диапазоне частот. Это определяется скоростью уменьшения спектральной плотности при отклонении частоты от центральной.

цифровом виде (в коде) и поступает на цифро-аналоговый преобразователь (ЦАП).

При цифровой реализации формирователя не составляет проблем сформировать фазу ЧМн-сигнала по принципу непрерывности (отсутствия скачков). При таком «гладком» формировании снижаются требования к величине подавления внеполосного излучения полосовыми фильтрами. Непрерывность фазы ЧМн-сигнала сохраняет гладкость сигнала при **скачкообразных** переключениях частоты (рис. 25).

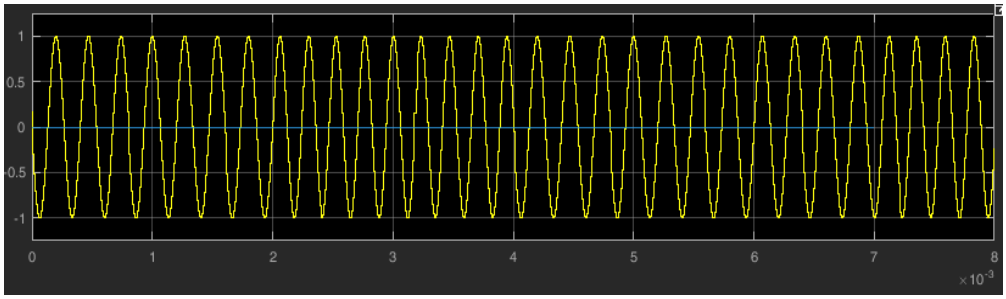


Рисунок 25 ЧМн-сигнал без разрыва фазы; частоты: 3,75 и 4,25 кГц; Длительность реализации — 8 мс

Длительность символа (бита) на рис. 25 равна 1 мс. Первый импульс содержит 3,75 периодов на символ[9], то есть имеет частоту 3,75 кГц (допустим, этому соответствует бит **0**). Второй импульс — аналогично, но начальная фаза у него будет другая[10]. Третий радиоимпульс имеет более высокую частоту: 4,25 кГц (это бит **1**). Таким образом, ЧМн-сигнал декодируется в отрезок **00110001.**

Из рис. 25 следует, что начальная фаза импульса (относительно момента его начала) зависит от вида предыдущего импульса, что говорит о наличии памяти в таком формирователе ЧМн-сигнала. Декодирование такого сигнала с учетом памяти возможно лишь при когерентной реализации приемника (то есть при наличии петли ФАПЧ[11]). Некогерентный приемник для повышения достоверности приема наличие памяти использовать не может, так как при таком приеме теряется зависимость результата от начальной фазы.

---

9 В реальности количество периодов на символ составляет несколько десятков
10 Проследите это!
11 Фазовой автоподстройки частоты

Математической моделью ЧМн-сигнала с непрерывной фазой и центральной несущей частотой f_0 является следующая конструкция

$$s_{FM}(t)=A\cos\left[2\pi\left(f_0 t+f_d\int_{-\infty}^{t}s_m(t)\,dt\right)+\varphi\right]\ , \qquad (19)$$

где f_d — частота девиации (у сигнала на рис. 25 девиация равна 250 Гц),

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

модулирующий сигнал[12], a_n — передаваемые биты в биполярном коде ±1, $g(t)\ge 0$ — формирующий импульс, отличный от нуля на интервале $0\le t<T$. Этот импульс влияет на спектральную плотность ЧМн-сигнала (распределение его средней мощности по частотам).

При скачкообразном изменении частоты формирующий импульс $g(t)$ является прямоугольным, а импульсы, определяющие мгновенную фазу ЧМн-сигнала, являются треугольными (линейно-ломаными). Поэтому спектральная плотность ЧМн-сигнала, сформированного без разрыва фазы, будет определяться сигналом типа косинус, у которого фаза изменяется по линейно-ломанному закону.

В системе сотовой связи GSM формирующий импульс сглаживается гауссовским фильтром, поэтому в GSM частота изменяется не скачкообразно, но достаточно быстро относительно длительности радиоимпульса. Сглаживание позволяет более простым способом выдержать частотную маску, диктуемую стандартами радиосвязи.

ЧМн-сигнал, сгенерированный согласно (19), является полосовым **(band-pass).** Практически же сначала генерируются лишь эквивалентный низкочастотный сигнал **(baseband),** полная фаза которого получается при приравнивании центральной частоты к нулю

$$\Theta(t)=2\pi f_d\int_{-\infty}^{t}\sum_{n\in\mathbb{Z}}a_n g(t-nT)\,dt+\varphi\ . \qquad (20)$$

Отсюда определим правило формирования непрерывной фазы на символьном интервале с целочисленным индексом k

12 Это сигнал с линейной модуляцией; ЧМн-сигнал является сигналом с **не**линейной модуляцией.

$$\Theta_k(kT+t)=2\pi f_d a_k \int_0^t g(t)\,dt+\varphi_{k-1}\ ,\quad 0\le t<T\ , \tag{21}$$

где $\varphi_{k-1}=\Theta_{k-1}(kT)$ — конечная фаза на предыдущем символьном интервале. Именно относительно нее и формируется фаза в текущем интервале, что гарантирует отсутствие разрыва фазы и, соответственно, косинуса фазы, то есть уровня результирующего ЧМн-сигнала.

Функциональная схема формирователя ЧМн-сигнала без разрыва фазы приведена на рис. 26.

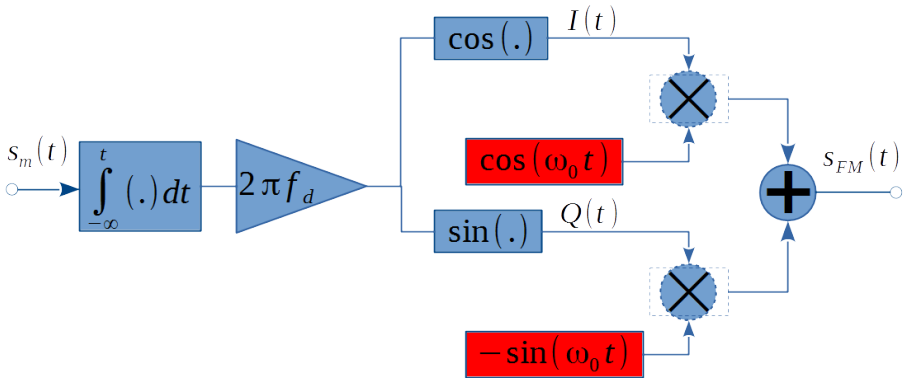


Рисунок 26 Функциональная схема формирователя ЧМн-сигнала без разрыва фазы

Осциллограммы квадратурных компонентов, соответствующих кодированию отрезка битов **00110001,** показаны на рис. 27.

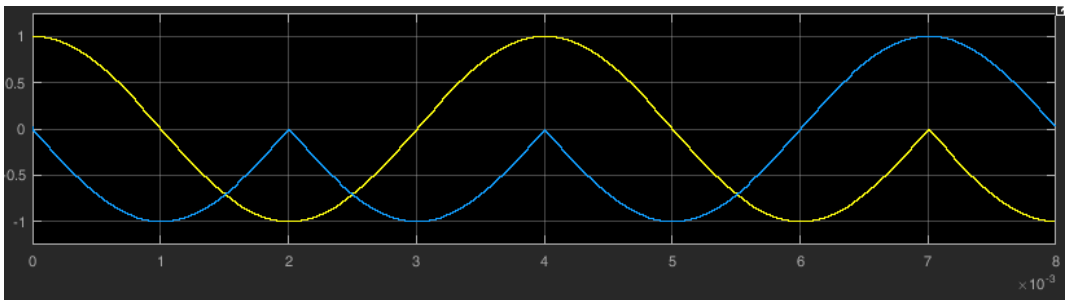


Рисунок 27 Развертка IQ-квадратур ЧМн-сигнала во времени. Соответствует рис. 25

На рис. 28 показан относительный угловой ход вектора при кодировании рассматриваемого отрезка битов. В качестве начального взят вектор с декартовыми координатами $(I,Q)=(1,0)$.

Вектор с центральной частотой 4 кГц (он называется **опорным**) вращается против часовой стрелки (так принято в радиотехнике). На рис. 28 показано

вращение текущего вектора **относительно** опорного. Относительность проявляется в том, что частоте 3,75 кГц соответствует вращение по часовой стрелке (отрицательная скорость, синий сектор), а частоте 4,25 кГц — против часовой стрелки (положительная скорость, красный сектор); частоте же 4 кГц при этом соответствует нулевая скорость.

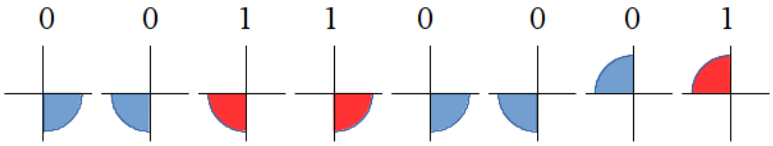


Рисунок 28 Угловой ход вектора с координатами (I, Q) при кодировании отрезка битов ЧМн-сигналом без разрыва фазы

На рис. 27 I -квадратура показана желтым цветом, Q -квадратура — синим. После прихода первого бита конец вектора из точки $(I,Q)=(1,0)$ переходит в точку $(I,Q)=(0,-1)$. Отследите на рассматриваемых рисунках весь процесс кодирования.

Наконец, определим **индекс модуляции** m как отношение величины разноса частот к символьной скорости передачи данных. В рассматриваемом примере индекс равен

$$m=\frac{(4{,}25-3{,}75)\ \text{кГц}}{1\ \text{кГц}}=1/2\ .$$

Оценка спектральной плотности такого ЧМн-сигнала показана на рис. 29, из которого следует, что уровень первого бокового лепестка равен −23 дБ. Это достигается за счет непрерывности фазы; для сигнала со случайным разрывом фазы уровень первого бокового равен −13 дБ.

Измеренная по спектру асимптотическая скорость спада мощности составила 24 дБ/октаву[13]. Это соответствует асимптотике спектральной плотности $1/f^8$, так как

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

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

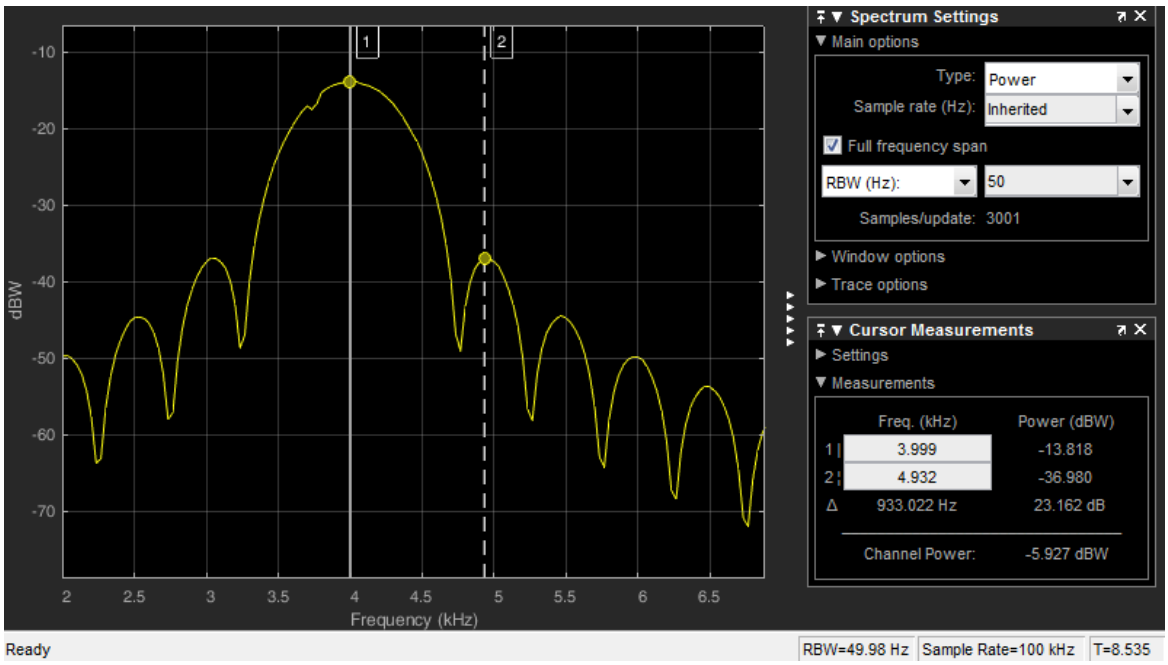


Рисунок 29 Спектральная плотность ЧМн-сигнала с непрерывной фазой, $m=1/2$

ЧМн-сигнал со случайным разрывом фазы имеет асимптотику спектральной плотности $1/f^2$.

Полоса частот ЧМн-сигнала, определенная по первым нулям, равна

$$2B_0=3m/T \quad . \tag{22}$$

Минимальный индекс модуляции, при котором можно обеспечить ортогональность разнесенных по частоте сигналов, равен ½. Почему важна ортогональность, т. е. равенство нулю скалярного произведения двух сигналов?

Дело в том, что демодулятор ЧМн-сигнала содержит два канала: один настроен на нижнюю частоту, другой — на верхнюю. Если на вход демодулятора поступает импульс с нижней частотой, то отклик в нижнем канале будет наибольшим, в то время как отклик в верхнем канале будет определяться величиной взаимной корреляции, и чем она будет ниже, тем «дальше» будут друг от друга отклики разных каналов, а, значит, ниже будет вероятность ошибки. Для ЧМн-сигнала возможно так подобрать разнос по частоте, что взаимная корреляция будет равна нулю, и для рассматриваемого

вида модуляции это будет наилучшим (с точки зрения вероятности ошибки) вариантом[14].

Для иллюстрации ортогональности рассмотрим взаимный корреляционный интеграл (нормированный к единице)

$$r_I = \frac{2}{T}\int_0^T \cos\left(2\pi(f_0 - f_d)t\right)\cos\left(2\pi(f_0 + f_d)t + \varphi\right)dt = \approx \frac{\sin(2\pi m - \varphi) + \sin\varphi}{2\pi m} \quad . \tag{23}$$

Интеграл для Q -канала получается дифференцированием (23) по фазе φ

$$r_Q = \frac{d\,r_I}{d\,\varphi} \approx \frac{-\cos(2\pi m - \varphi) + \cos\varphi}{2\pi m} \quad . \tag{24}$$

Приближение сделано из-за практического выполнения условия $f_0 \gg f_d$. При некогерентной демодуляции играет роль модуль

$$r = \sqrt{r_I^{\,2} + r_Q^{\,2}} \quad . \tag{25}$$

Зависимость уровня кросс-корреляции r от индекса модуляции отражена на рис. 30. Заметим, что здесь показана зависимость, полученная при интегрировании на одном символьном интервале. Практически же можно реализовать интегрирование (суммирование) на интервале, несколько отличном от символьного, что может привести к снижению уровня кросс-корреляции для дробных индексов модуляции и к соответствующему уменьшению частоты символьных ошибок.

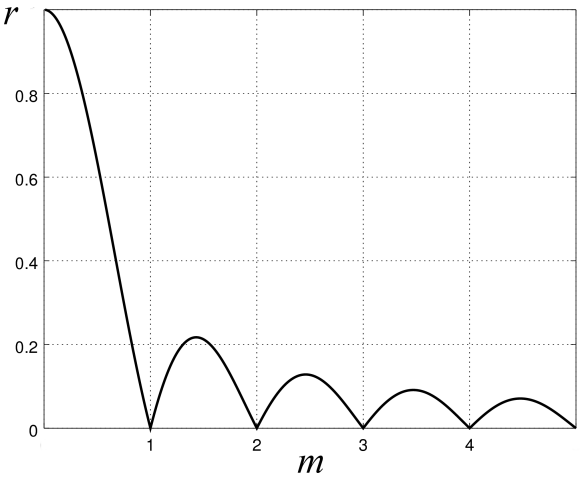


Рисунок 30 Зависимость уровня кросс-корреляции от индекса модуляции на выходе детектора огибающей ЧМн-демодулятора

14 Для сигналов с фазовой манипуляцией, ФМн-сигналов, взаимный корреляционный интеграл может быть отрицательным по величине, поэтому системы с ФМн более помехоустойчивы, чем системы с ЧМн.

Спектральная плотность ЧМн-сигнала с индексом модуляции $m=1$ содержит особенности в виде игл на частотах модуляции (4500 и 3500 Гц, рис. 31). Поясним природу этих игл.

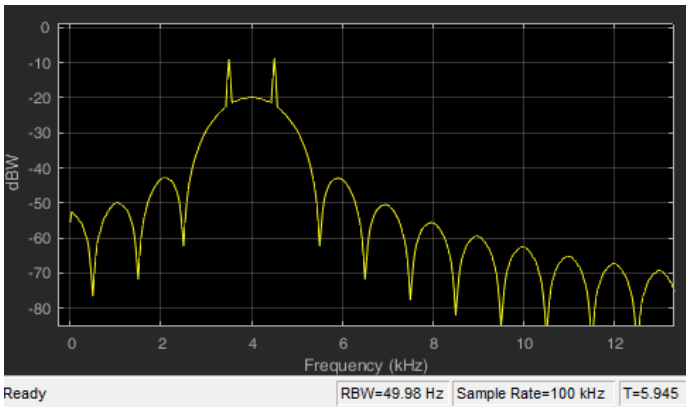


Рисунок 31 Спектральная плотность ЧМн-сигнала без разрыва фазы, $m=1$

Набег фазы за символьный интервал при $m=1$ равен 180 градусам (в общем случае $\Delta\varphi=180^{\circ}m$). Знак набега фазы зависит от бита. Косинусная квадратура I нечувствительна к знаку набега фазы, так как косинус функция четная, поэтому как бы биты ни сочетались, эта квадратура будет формироваться так, как будто биты не изменяются, что и приведет к ее «чистоте». Синусная же квадратура при смене бита будет иметь излом. «Чистота» косинусной квадратуры и приводит к иглам (практическому наблюдению дельта-функции) на частотах 4500 и 3500 Гц (период косинусной квадратуры равен 2 мс).

Чтобы исключить иглы, индекс модуляции в данной работе рекомендуется брать равным 0,625. В этом случае набег фазы за длительность символа будет 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

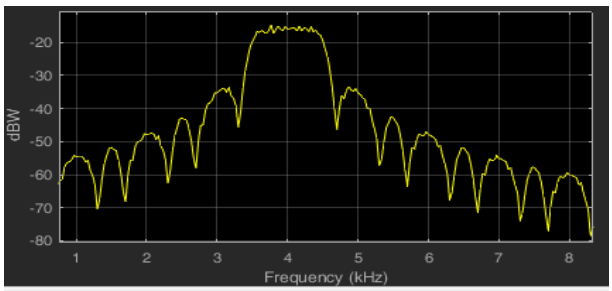


Рисунок 32 Спектральная плотность ЧМн-сигнала без разрыва фазы, $m=0{,}625$

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

Это способствует прямоугольности спектральной плотности (рис. 32). Однако, уровень первого бокового лепестка при этом равен −18 дБ, а скорость спада мощности внеполосного излучения — 20 дБ/октаву; пример развертки квадратур во времени показан на рис. 33.

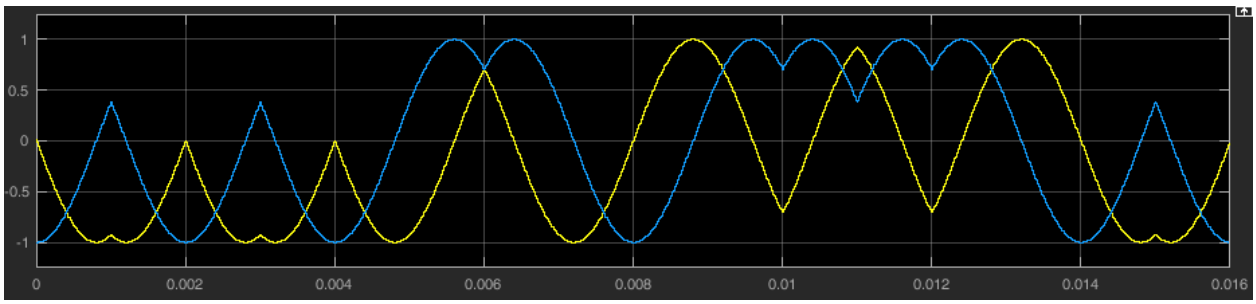


Рисунок 33 Развертка IQ-квадратур ЧМн-сигнала во времени, $m=0{,}625$, 16 битов

Схема некогерентного ЧМн-демодулятора показана на рис. 34.

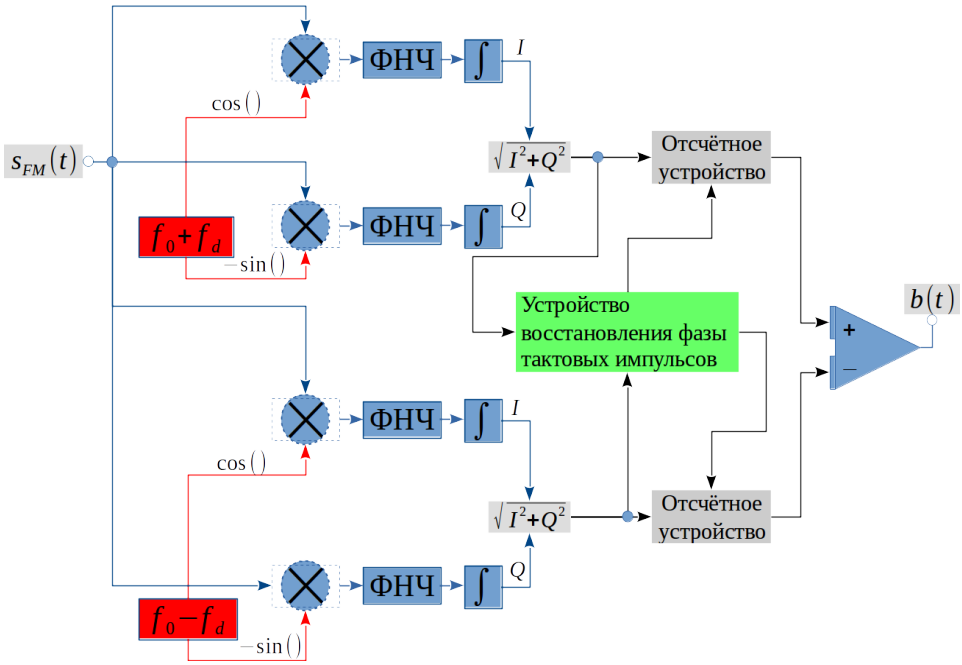


Рисунок 34 Функциональная схема некогерентного демодулятора ЧМн-сигнала

ЧМн-сигнал $s_{FM}(t)$ поступает на первые входы четырех смесителей (умножителей), на вторые входы которых поступают опорные сигналы. Верхний канал, включающий два смесителя, настроен на верхнюю частоту f_0+f_d , нижний — на нижнюю f_0-f_d . Начальные фазы обоих опорных генераторов должны поддерживаться одинаковыми.

Сигналы с выхода смесителей поступают на ФНЧ, задача которого — ослабить компоненты с частотами $2f_0+2f_d$ и $2f_0$.

После ФНЧ идет фильтр, приближенно реализующий интегратор со скользящим интервалом интегрирования

$$y(t)=\int_{t}^{t+T} x(t)\,dt \quad .$$

В макете такой интегратор реализован в виде фильтра с конечной импульсной характеристикой (КИХ), равной единице на символьном интервале (попросту, это сумматор). Задача интегратора — ослабить влияние теплового шума (за счет эффекта усреднения), а также ослабить компоненту с разностной частотой $2f_d$, образующейся при поступлении «чужого» сигнала. При этом при поступлении «своего» сигнала образуется постоянная составляющая, интегрирование которой дает отсчет, содержащий в себе максимум информации[15] о переданном бите.

В принципе, ФНЧ в схеме не обязателен (зависит от реализации), но интегратор — принципиально нужен, так как в противном случае вероятность ошибки резко возрастает (мешает компонента с удвоенной частотой девиации).

После формирования квадратур, в каждом канале формируется огибающая (это и есть ключевая операция некогерентного демодулятора), которая поступает на отсчетное устройство, делающее выборки в тактовые моменты времени, определяемые тактовым генератором. Сигналы с выходов двух отсчетных устройств сравниваются компаратором, который в зависимости от результата сравнения устанавливает свой выход в **0** или в **1.** В идеале, если в одном канале максимум, то в другом канале должен быть ноль, и наоборот. В реальности, этому мешает собственный шум приемника и ненулевой уровень кросс-корреляции сигналов.

ФНЧ в макете спроектирован в виде цифрового КИХ-фильтра с переходной полосой (2500...5000) Гц, с подавлением в полосе заграждения 60 дБ и неравномерностью в полосе пропускания 1 дБ. Частота дискретизации ЧМн-сигнала равна 100 кГц. Важно: все эти параметры и следующие не являются догмой!

---

15 Максимум, если отсчеты шума некоррелированы и распределены по нормальному закону

В макете между ФНЧ и интегратором стоит блок **Downsample** с коэффициентом 4, поэтому интегратор имеет порядок (длину импульсной характеристики фильтра)

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

Отсчетное устройство в макете делает выборки через каждые 25 отсчетов, но начинает выборку с определенной фазы. Эта фаза **(offset)** зависит от задержки в ФНЧ, которая для рассматриваемого фильтра равна 44 отсчетам или 0,44 мс (рис. 35).

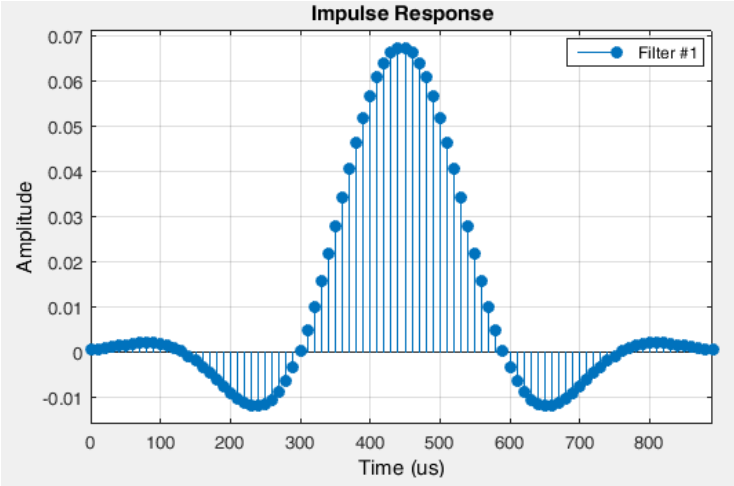


Рисунок 35 Импульсная характеристика ФНЧ некогерентного ЧМн-демодулятора

Для упрощения работы в лабораторном макете отсутствует устройство восстановления фазы тактовых импульсов. Эта фаза (параметр **offset**) выставляется в виде заранее вычисленной константы.

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` 120 дБ. Индекс модуляции выставить на `m=0,625.` Выключить режим статистических испытаний, выставив `Nstat=1`. Набрать в переменную `inputBits` битовую последовательность согласно своему варианту и Приложению А.

1. Привести в отчете осциллограммы в следующих контрольных точках формирователя ЧМн-сигнала:

- Битовую последовательность в биполярном коде: переменная `input;`

- Квадратуры *I* и *Q:* переменные `cpfskRe` и `cpfskIm;`
- Сформированный ЧМн-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, Lpf0Im-Down, Lpf1ReDown` и `Lpf1ImDown;`
- Сигналы с выходов интеграторов: 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: переменная `demodulatedBits.`

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

Для вывода графиков в качестве образца использовать приведенный в скрипте код, строящий три графика. Убедиться, что биты в результате демодуляции детектированы верно. Убедиться, что ЧМн-сигнал не имеет разрыва фазы. Привести частотные характеристики ФНЧ, воспользовавшись функцией `freqz(h).`

Повторить п. 1 для ЧМн-сигнала с индексом модуляции `m=1.`

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

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 демодуляции ортогональных сигналов [3]

$$P_s = \frac{1}{2}\exp\left(-\frac{1}{2}\frac{E_b}{N_0}\right)$$, (здесь сигнал-шум **в разах**). (26)

Построить соответствующие графики.

3. Выставить сигнал-шум на 120 дБ. Вывести график спектра мощности ЧМн-сигнала[16]: переменная `psdFM`. Измерить полосу сигнала по уровню −3 дБ. Измерить уровень боковых лепестков, а также скорость спада мощности в зависимости от частоты (измеряется в дБ/октаву).

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

5. Выставить `m=0,625, Nstat=1` и `EbNo=20дБ`. Сравнить сигналы с выходов вычислителей модуля (`out0` и `out1`) при включенном и при выключенном интеграторе.

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

4. Контрольные вопросы

1. Чем хорош ЧМн-сигнал с непрерывной фазой?

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

3. К какому виду модуляции относят частотную модуляцию: к нелинейной или линейной? Почему?

4. Почему теоретический график вероятности ошибки (характеристика верности) заметно отличается от результатов статистических испытаний? Предложите свой вариант схемы с лучшей характеристикой верности.

5. Список литературы

1. Бернард Скляр. Цифровая связь.

2. Джон Прокис. Цифровая связь.

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

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

Приложение А Последовательности для подачи на вход ЧМн-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 лабораторная работа адресована студентам, изучающим принципы цифровой связи.

В цифровых системах связи исходная информация представляется в виде последовательности символов, например, битов

...100101101010...

Символы, являющиеся абстракцией, без формирования соответствующих сигналов не могут быть переданы по линиям связи, поэтому используют отображение множества символов в множество сигналов. Этот процесс можно назвать *модуляцией.* Таким образом, в цифровых системах связи под модуляцией понимается процесс отображения потока символов в набор сигнальных импульсов (импульсов-носителей). Хотя, если докопаться, поток символов, поступающих на вход модулятора, уже представлен в виде сигналов, но других, не тех, которые пригодны для конкретной линии связи. Поэтому под модуляцией можно понимать переход от одной системы сигналов к другой, согласованной с линией связи. Хотя, если опять докопаться, можно вообразить математическое отображение «символ-сигнал». Хотя… Хотеть не вредно!

Важно! Требуется усвоение таких понятий как спектр сигнала, ряд и интеграл Фурье.

Основная особенность данной лабораторной работы заключается в наличии в цифровом сигнале такого качества как *информационная случайность.* Такая особенность делает спектр Фурье соответствующего сигнала зависимым от конкретной *реализации символов.* Обратите внимание на слово **реализация,** которое говорит о наличии *ансамбля* реализаций (*множества* реализаций).

Спектры, содержащие *случайную* составляющую, не всегда нужны, поэтому эту составляющую ослабляют с помощью операции *усреднения,* которая сохраняет *регулярную* (неслучайную) составляющую, обязательно присутствующую в спектре хотя бы из-за неслучайной формы *импульса-носителя*…

Почему последовательность символов случайная?

Дело здесь в том, что передача информации происходит лишь тогда, когда приемник различает изменение сигнала и это изменение для приемника непредсказуемо — вот в этой частичной (или полной) непредсказуемости и кроется та случайность, которую можно назвать информационной и которая является столпом любой связи. При полной предсказуемости нет смысла передавать информацию, так как она уже известна.

Поясним предыдущий абзац на житейском примере.

Допустим, у нас есть уговор с рыбаком, отплывающим в 6 часов утра на другой берег. Уговор в том, что он в 10 часов утра в случае удачного улова зажигает красный сигнальный огонь, а в случае неудачного — зеленый, но один из двух обязательно должен быть зажжен. Какой это будет цвет, нам (приемнику) заранее не известно, зато *алфавит* (все возможные цвета), разумеется, известен, иначе мы ничего не примем (не поймем).

Информация от рыбака в час **X** будет получена лишь тогда, когда будет определен цвет (этому может помешать дым костра). В случае определения цвета происходит снятие 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.

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. Однако в этой работе вредное влияние таких шумов не изучается.

В системах связи идет поток сигналов-импульсов, причем каждому импульсу соответствует один символ алфавита. Рассмотрим самый простой алфавит — *двоичный (битовый).*

Пусть биту **1** соответствует импульс прямоугольной формы, а биту **0** — отсутствие импульса (пауза). Тогда можно заметить, что для потока из одних нулей **...000...** или одних единиц **...111...** будет генерироваться постоянное напряжение, что сводит на нет возможность передачи такого сигнала через существующие 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. Отсюда ясно, что на форму спектра влияет не только длительность импульса, но и статистика символов.

Чтобы ослабить случайную составляющую спектра, усредняют **квадрат его модуля,** который пропорционален **мощности.** Такая характеристика как *средний квадрат модуля* в данной работе играет ключевую роль.

Отчет должен содержать:

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

2. Сведения из теории

2.1 Спектральная плотность импульса-носителя

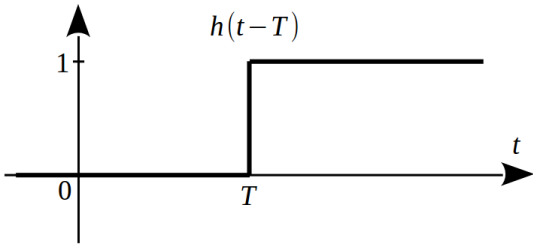


Рисунок 36: Функция Хевисайда — единичный скачок

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

Например, сигналом является импульс *прямоугольной формы,* равный разности двух функций *Хевисайда*

$h(t)$ (рис. 36, рис. 37). Физически таких объектов нет и к ним можно лишь приближаться.

Единичный скачок не ограничен по времени, а сигнал, переносящий один символ, обязан иметь конечную длительность[17] (быть финитным). По этой причине при построении прямоугольного импульса амплитуды вычитаемых скачков выбираются равными

$$\mathrm{rect}(t) = A \cdot h(t) - A \cdot h(t-T) \quad . \qquad (27)$$

Нулевой уровень не считается сигналом, поэтому прямоугольный импульс ограничен во времени. Параметр T в (27) называется *длительностью импульса,* а A *— амплитудой импульса.*

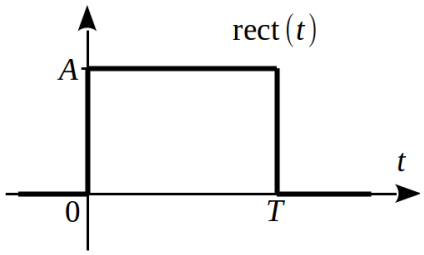


Рисунок 37: Импульс прямоугольной формы

Сигналы, соответствующие электрическим процессам, никогда не имеют строго прямоугольной формы, так как для скачкообразного изменения амплитуды от нуля до некоторой ограниченной величины требуется множество гармоник все более и более высоких частот[18]. Плюс после скачка уровня необходимо на некоторое время удерживать постоянный уровень, чего также не бывает, так как в цепях идут переходные процессы, связанные с неустранимыми емкостями и индуктивностями.

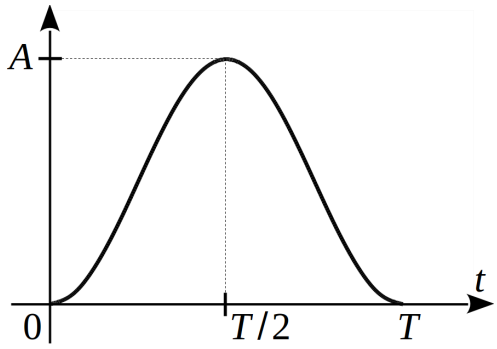


Рисунок 38: Импульс в виде приподнятого косинуса

Рассмотренный прямоугольный импульс можно считать одним из многих носителей, используемых для кодирования битов. Разные носители имеют разные плотности распределения энергии по частоте, отличающиеся, в частности, скоростью убывания энергии. Скорость убывания играет важную роль, ведь чем более *компактным* будет спектр

17 Это не догма, но для начала и так сойдет! (смотри импульсы Котельникова, например)
18 Смотри ряд для меандра (36) и его первую производную в точках перепада уровня

(компактным не означает узким), тем меньшим будет уровень внеполосного излучения (смотри понятие *спектральная маска сигнала*).

Другим примером носителя является импульс в виде *приподнятого косинуса* (рис. 38)

$$g_c(t)=\frac{A}{2}\left(1-\cos\frac{2\pi t}{T}\right) \quad . \tag{28}$$

Этот импульс также идеален, так как его спектр неограничен по частоте, зато энергия с ростом частоты убывает быстрее энергии прямоугольного импульса. Чтобы убедиться в различии спектров, вычислим спектральные функции с помощью интеграла Фурье.

Для прямоугольного импульса

$$G_{rect}(f)=\int_0^T A\,e^{-2\pi jft}\,dt=\frac{A}{-2\pi jf}\,e^{-2\pi jft}\Bigg|_0^T=AT\,\frac{\sin(\pi fT)}{\pi fT}\,e^{-\pi jfT} \quad . \tag{29}$$

Для расчета спектральной плотности (в данном случае — энергии) необходимо взять квадрат модуля от (29) (объяснение смотри в (43))

$$\Phi_{rect}(f)=|G_{rect}(f)|^2=(AT)^2\left(\frac{\sin(\pi fT)}{\pi fT}\right)^2 \quad , \quad B^2\cdot c/Гц \quad . \tag{30}$$

Если A измеряется в вольтах, то (30) — в $\text{вольт}^2\cdot\text{секунда}/\text{герц}$. Для сопротивления 1 Ом это будет $\text{джоуль}/\text{герц}$, что и говорит о плотности энергии, то есть сколько джоулей в среднем приходится на полосу один герц.

Интеграл от (30) по всем частотам даст полную энергию сигнала, которая в данном случае вычисляется намного проще через интеграл от квадрата сигнала

$$\int_{-\infty}^{\infty}(AT)^2\left(\frac{\sin(\pi fT)}{\pi fT}\right)^2 df=A^2T \quad , \tag{31}$$

что подтверждается известным в математике равенством

$$\int_{-\infty}^{\infty}\left(\frac{\sin x}{x}\right)^2 dx=\pi \quad .$$

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 импульс с постоянным уровнем.

Исходя из равенства $\Phi_{rect}(1/T) = 0$ делаем вывод, что составляющей с частотой $1/T$ в прямоугольном импульсе нет. Этой составляющей 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. 39), 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.

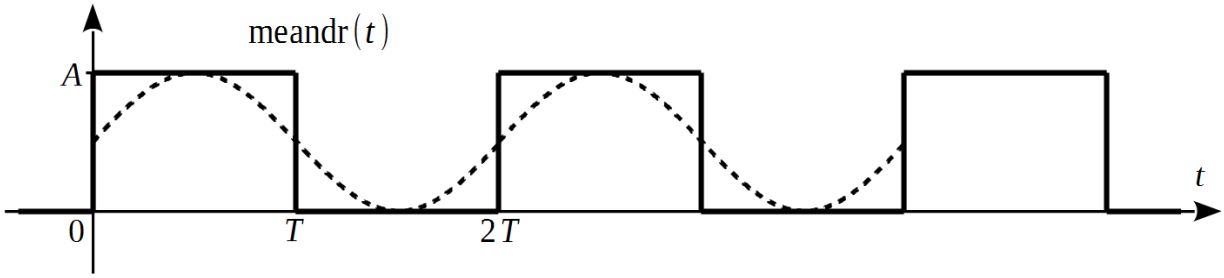
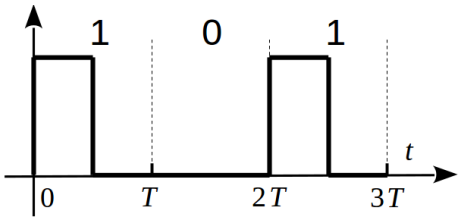


Рисунок 39: Меандр — результат переключения уровня с частотой $1/T$

В цифровом приемнике важно знать моменты, соответствующие центру импульсов, поэтому приемник должен генерировать тактовые импульсы, *выровненные по фазе* с центрами принимаемых импульсов, и,

значит, следующие с частотой $1/T$. Если в системе связи выделен канал для передачи синхроимпульсов от передатчика к приемнику, то это хорошо, а если нет, то приемник должен их выделить из информационных импульсов.

Мы только что выяснили, что энергетическая доля частотной составляющей $1/T$ равна нулю, так что никакие линейные фильтры не смогут выделить стабильных тактовых синхроимпульсов.



Одним из выходов является сложение (по модулю два) исходного сигнала и сигнала, задержанного на половину такта. Это эквивалентно повышению частоты переключения в два раза, что при псевдослучайном исходном битовом потоке даст стабильный уровень искомой составляющей с частотой $1/T$. По такому принципу может работать *схема восстановления тактовой частоты* (есть и другие принципы, например, на основе детектора Гарднера).

Вычисление спектральной функции меандра (можно пропустить)

==========

Меандр состоит из бесконечного количества прямоугольных импульсов, **смещенных** друг относительно друга на постоянную величину $2T$, поэтому спектр меандра можно вычислить как сумму спектров прямоугольного импульса (29), отличающихся **фазовым множителем,** отражающим задержку по времени соответствующего импульса

$$G_{meandr}(f)=AT\frac{\sin\pi fT}{\pi fT}\left[\ldots+e^{7\pi jfT}+e^{3\pi jfT}+e^{-\pi jfT}+e^{-5\pi jfT}+e^{-9\pi jfT}+\ldots\right] . \quad (32)$$

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

$$\begin{aligned} G_{meandr}(f)&=G_{rect}(f)\left[\ldots+e^{2\pi j2f2T}+e^{2\pi j1f2T}+1+e^{-2\pi j1f2T}+\ldots\right]= \\ &=\frac{G_{rect}(f)}{2T}\left[\ldots+\delta\left(f+\frac{1}{2T}\right)+\delta(f)+\delta\left(f-\frac{1}{2T}\right)+\delta\left(f-\frac{2}{2T}\right)+\ldots\right] . \end{aligned} \quad (33)$$

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 свойство изменения масштаба

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

Замену суммы экспонент суммой дельта-функций можно осознать, если построить график частичной суммы экспонент, взяв для начала 5 слагаемых, затем 21 и так далее (рис. 40). Суммировать следует симметрично относительно слагаемого, равного 1.

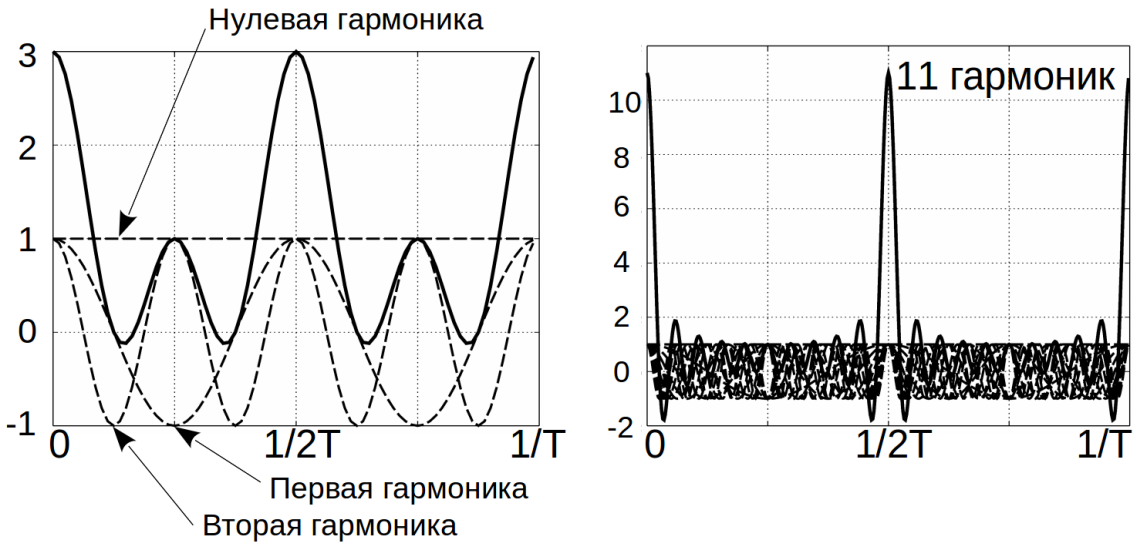


Рисунок 40: Частичные суммы косинусов кратных частот

Особенность суммируемых экспонент в том, что каждая имеет комплексно-сопряженную пару, поэтому по сути суммируются только косинусы. Из рис. 40 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 (33) за счет наличия 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 при численном расчете амплитуд спектра (33), так как $\delta(0)=\infty$. Эта проблема появилась из-за того, что энергия бесконечного количества импульсов равна бесконечности. Чтобы этого избежать, спектр делят на величину интервала интегрирования, то есть вычисляют через предел

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

Сигналы, с неограниченной энергией называют *мощностными.* Сигналы с ограниченной энергией называются *энергетическими.* Мощностной сигнал не обязан быть периодическим, так как сигналы в системах связи часто содержат случайную последовательность битов.

Замечая из (29), 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 спектр амплитуд меандра, получим

$$G_{meandr}(k)=\begin{cases}\dfrac{A}{2}, & k=0\\ 0, & k-\text{чётное}\\ -j\dfrac{A}{\pi}\dfrac{1}{k}, & k-\text{нечётное}\end{cases}. \qquad (34)$$

Дельта-функции $\delta(t)$ при вычислении спектра **мощностного** сигнала переходят в цифровые дельта-функции δ_k, принимающие значения **0** и **1.** В этом можно убедиться, если в (33) взять три слагаемых $N=3$, симметрично относительно единице

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

Три слагаемых отвечают за три прямоугольных импульса. Если разделить данный спектр на величину **отрезка** меандра $5T$ (три импульса, две паузы), а затем найти предел при $N\to\infty$ на частоте $k/(2T)$

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

то можно убедиться в искомой трансформации дельта-функций.

Из выражения (34) видно, что в спектре меандра имеется постоянная составляющая и синусные нечетные гармоники. Для подтверждения

выполненных расчетов вычислим коэффициенты ряда Фурье для некоторого сигнала с периодом $2T$

$$s_T(t)=\frac{a_0}{2}+\sum_{k=1}^{\infty} a_k\cos(\pi k t/T)+b_k\sin(\pi k t/T) \qquad (35)$$

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 результат в (35)

$$s_{meandr}(t)=\frac{A}{2}+\frac{2A}{\pi}\sum_{k=1,3,5,\ldots}\frac{1}{k}\sin(\pi k t/T) \ , \text{(период равен } 2T\text{).} \qquad (36)$$

Выражение (36) полностью совпадает с известным рядом Фурье меандра.

==========

Наконец, вычислим спектральную функцию приподнятого косинуса, (28)

$$G_c(f)=\frac{1}{2}G_{rect}(f)-\frac{A}{2}\int_0^T\cos\left(\frac{2\pi t}{T}\right)e^{-2\pi jft}dt= \\ =\frac{AT}{2}\frac{1}{1-(fT)^2}\frac{\sin(\pi fT)}{\pi fT}e^{-j\pi fT} \ . \qquad (37)$$

Расчет спектральной плотности дает

$$\Phi_c(f)=|G_c(f)|^2=\left(\frac{AT}{2}\right)^2\left[\frac{1}{1-(fT)^2}\right]^2\left[\frac{\sin(\pi fT)}{\pi fT}\right]^2 \ . \qquad (38)$$

Таким образом, спектральная плотность косинусного импульса убывает как $1/f^6$ с ростом частоты, то есть на четыре порядка быстрее плотности прямоугольного импульса. Но ширина спектра косинусного импульса, если ее определить по первому нулю, больше, так как

$\Phi_c(1/T)\neq 0$. Предлагается определить, чему равно это ненулевое значение, а также построить графики энергетических спектров обоих импульсов.

2.2 Спектр мощности дискретной случайной последовательности

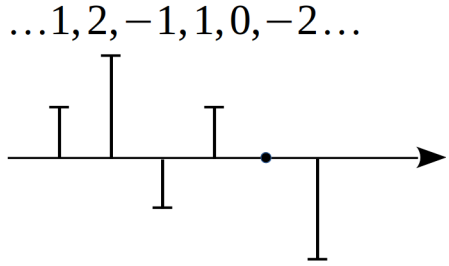


Рисунок 41: Реализация дискретной случайной последовательности

Из-за неограниченности числовой последовательности во времени (рис. 41) имеет смысл спектр мощности.

Если мы хотим вычислить спектр *мощности*, то последовательности чисел следует поставить в соответствие сигнал. Сигнал образуется носителем, два вида которых (прямоугольный и косинусный импульсы) были рассмотрены выше.

Для исследования спектральных свойств последовательности следует взять носитель в виде следующей функции

$$\delta_n=\begin{cases}1, \text{ если } n=0;\\ 0, \text{ если } n\neq 0.\end{cases} . \qquad (39)$$

Эта функция для радиотехники является единичным импульсным сигналом, имеющим равномерный периодический спектр (с периодом $1/T$)

$$G_\delta(f)=T\sum_{n\in\mathbb{Z}}\delta_n e^{-j2\pi nfT}=T\,\delta_0 e^{-j2\pi 0fT}=T \quad . \qquad (40)$$

Равномерность спектра позволяет отделить влияние импульса-носителя на спектр последовательности. Конструирование спектра из двух произойдет в конце работы — в этом и будет состоять ее смысл.

Известно, что спектр мощности пропорционален квадрату модуля спектральной функции. Это вытекает из того, что спектральная плотность мощности связана с функцией корреляции $R(\tau)$ преобразованием Фурье[19]

$$\Phi(f)=\int_{-\infty}^{\infty}R(\tau)e^{-j2\pi f\tau}d\tau \quad , Вт/Гц. \qquad (41)$$

Подставляя в (41) выражение для функции корреляции вещественного импульса $g(t)$ длительностью T

19 Для стационарных случайных процессов

$$R(\tau)=\frac{1}{T}\int_{-\infty}^{\infty} g(t)\cdot g(t-\tau)\,dt \quad , Bm, \tag{42}$$

последовательно получим

$$\begin{gathered}\Phi(f)=\frac{1}{T}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(t)\cdot g(t-\tau)\,dt\,e^{-j2\pi f\tau}d\tau \quad , \\ \Phi(f)=\frac{1}{T}\int_{-\infty}^{\infty} g(t)\int_{-\infty}^{\infty} g(t-\tau)e^{-j2\pi f\tau}d\tau\,dt \quad , \quad t-\tau=p \quad , \\ \Phi(f)=\frac{1}{T}\int_{-\infty}^{\infty} g(t)\int_{-\infty}^{\infty} g(p)e^{j2\pi fp}dp\,e^{-j2\pi ft}dt \quad , \\ \Phi(f)=\frac{1}{T}G^{*}(f)\int_{-\infty}^{\infty} g(t)e^{-j2\pi ft}dt=\frac{1}{T}G^{*}(f)G(f)=\frac{1}{T}|G(f)|^{2} \quad .\end{gathered} \tag{43}$$

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

$$\Phi_{digit}(f)=T\sum_{i\in\mathbb{Z}}\phi(i)e^{-j2\pi ifT} \quad , Bm/Гц, \tag{44}$$

где T — интервал следования чисел, определяющий период спектра 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

$$\begin{aligned}&\mu_b=M[b_n],\\ &D_b=M\left[(b_n-\mu_b)^2\right],\\ &r=\frac{M\left[(b_n-\mu_b)(b_{n-i}-\mu_b)\right]}{D_b}=\begin{cases}1, & i=0;\\ 0, & i\neq 0.\end{cases}\end{aligned} \quad , \tag{45}$$

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

Тогда функция корреляции данной последовательности будет равна

$$\phi_{bb}(i)=M[b_nb_{n-i}]=M\left[(b_n-\mu_b)(b_{n-i}-\mu_b)\right]+\mu_b{}^2=\begin{cases}D_b+\mu_b{}^2, & i=0;\\ \mu_b{}^2, & i\neq 0.\end{cases} \quad . \tag{46}$$

Подставляя (46) в (44), получим спектр мощности

$$\Phi_{bb}(f)=D_bT+\mu_b{}^2T\sum_{i\in\mathbb{Z}}e^{-j2\pi ifT} \quad . \tag{47}$$

Используя знакомую замену суммы экспонент из (33), можно записать

$$\Phi_{bb}(f)=D_b T+\mu_b{}^2\sum_{i\in\mathbb{Z}}\delta(f-i/T)\ . \qquad (48)$$

Из формулы (48) следует, что спектр случайной последовательности некоррелированных чисел состоит из двух слагаемых: первое возникает из-за отсутствия корреляции и говорит о равенстве мощностей всех частотных составляющих спектра, а второе — из-за отличного от нуля среднего значения μ_b и говорит о полной корреляции по постоянной составляющей. Игольчатые выбросы в спектре, связанные с ненулевым средним, вредны (лишняя трата энергии), поэтому при проектировании систем связи делают все возможное для уменьшения этого среднего.

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

$$I_n=b_n+b_{n-1}\ ,\quad \mu_b=M[b_n]=0\ ,\quad D_b=M\left[(b_n-\mu_b)^2\right]\ . \qquad (49)$$

Функция корреляции последовательности I_n будет равна

$$\begin{aligned}\phi_{ii}(i)&=M[I_n I_{n-i}]=M\left[(b_n+b_{n-1})(b_{n-i}+b_{n-i-1})\right]=\\&=M[b_n b_{n-i}+b_n b_{n-i-1}+b_{n-1}b_{n-i}+b_{n-1}b_{n-i-1}]=\\&=\begin{cases}2D_b, & i=0;\\ D_b, & i=\pm 1;\\ 0, & |i|>1.\end{cases}\end{aligned}\ . \qquad (50)$$

Три слагаемых в (44) определят искомый спектр мощности

$$\Phi_{ii}(f)=2D_b T\left[1+\cos(2\pi f T)\right]\ . \qquad (51)$$

Функция корреляции для вещественного случайного процесса — четная, поэтому у каждой экспоненты преобразования Фурье 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 мощности и осью частот за период, определяет среднюю мощность сигнала; в данном случае она равна $2D_b$, что говорит о повышении мощности в два раза после

преобразования (49), так как площадь спектральной фигуры для последовательности b_n равна D_b .

Введение корреляции (49) обнулит составляющие с частотами, кратными (2*n*+1)/(2*T*), а введение корреляции типа $I_n=b_n-b_{n-1}$ обнулит составляющие с частотами, кратными *n*/*T*, в том числе и постоянную составляющую (составляющую с нулевой частотой), где *n* — любое целое число. Вычисление разности $I_n=b_n-b_{n-1}$ по смыслу является дифференцированием, после которого естественным образом удаляется постоянная составляющая.

2.3 Спектральная плотность цифрового сигнала с линейной модуляцией

Математической моделью цифрового сигнала с линейной модуляцией является сумма * [1]

$$v(t)=\sum_n I_n g(t-nT) \quad , \tag{52}$$

где T — длительность импульса-носителя $g(t)$;

I_n — последовательность чисел, содержащая информацию.

*Принято считать, что если в сумме не указаны пределы суммирования, то индекс пробегает весь диапазон целых чисел. Отсутствие границ упрощает многие математические выкладки.

Вычислим функцию корреляции случайного процесса (52) [1]

$$\phi_{vv}(t+\tau;t)=M\left[v(t)v(t+\tau)\right]=\sum_n\sum_m M\left[I_n I_m\right]g(t-nT)g(t+\tau-mT) \quad . \tag{53}$$

Так как в (53) индекс суммирования m пробегает все множество целых чисел, то заменим его на $m+n$ и предположим, что последовательность I_n стационарна по отношению к функции корреляции

$$M\left[I_n I_m\right]=M\left[I_n I_{n+m}\right]=\phi_{ii}(m) \quad . \tag{54}$$

Тогда подставив (54) в (53) и поменяв суммы местами, получим

$$\phi_{vv}(t+\tau;t)=\sum_m \phi_{ii}(m)\sum_n g(t-nT)g(t+\tau-nT-mT) \quad . \tag{55}$$

Из (55) следует, что функция корреляции процесса $v(t)$ — периодическая функция времени с периодом T. Это можно обосновать тем, что индекс n в сумме пробегает все множество целых чисел, поэтому сколько раз T ни прибавляй к аргументу $g(t)$, результат суммирования не изменится.

Обозначая среднее значение последовательности I_n как $\mu_i = M[I_n]$, найдем среднее значение процесса $v(t)$

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

Видим, что среднее зависит от времени и является периодической функцией с периодом T. Значит процесс $v(t)$ является *периодически стационарным* по отношению к функции корреляции.

Усредним (55) за период, дабы убрать зависимость от времени t

$$\begin{aligned} \phi_{vv}(\tau) &= \frac{1}{T} \int_{-T/2}^{T/2} \phi_{vv}(t+\tau; t)\, dt = \\ &= \sum_m \phi_{ii}(m) \sum_n \frac{1}{T} \int_{-T/2-nT}^{T/2-nT} g(t)\, g(t+\tau-mT)\, dt \end{aligned} \quad . \qquad (56)$$

Интеграл в (56) вместе с суммой по n в точности является функцией корреляции сигнала $g(t)$

$$\phi_{gg}(\tau) = \frac{1}{T} \int_{-\infty}^{\infty} g(t)\, g(t+\tau)\, dt \quad , \qquad (57)$$

Поэтому подстановка (57) в (56) дает

$$\phi_{vv}(\tau) = \sum_m \phi_{ii}(m)\, \phi_{gg}(\tau - mT) \quad . \qquad (58)$$

Преобразование Фурье (58) дает спектральную плотность мощности процесса $v(t)$

$$\Phi_{vv}(f) = \frac{1}{T^2} |G(f)|^2 \Phi_{ii}(f) \quad , \text{Вт/Гц}, \qquad (59)$$

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

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

Из выражения (59) следует, что на спектр сигнала с линейной модуляцией влияют как спектр импульса-носителя, так и корреляционные (читай спектральные) характеристики цифровой последовательности. Это является важным фактом при изучении кодов, используемых на самом нижнем физическом уровне (линейных кодов, от словосочетания *линия связи*), таких как RZ, NRZ, NRZ-I, **Манчестер,** MLT-3 и тому подобных. Эти коды, вкупе с такими кодами, как, например, 4B/5B, 8B/10B, 8B/6T..., позволяют сформировать желательное для данной среды передачи распределение мощности по частотному диапазону, а также обеспечить качественную тактовую синхронизацию.

3. Описание лабораторного макета

Программный макет (рис. 42) является приложением, написанным на языке C++ с использованием библиотек **Qt** [2].

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).

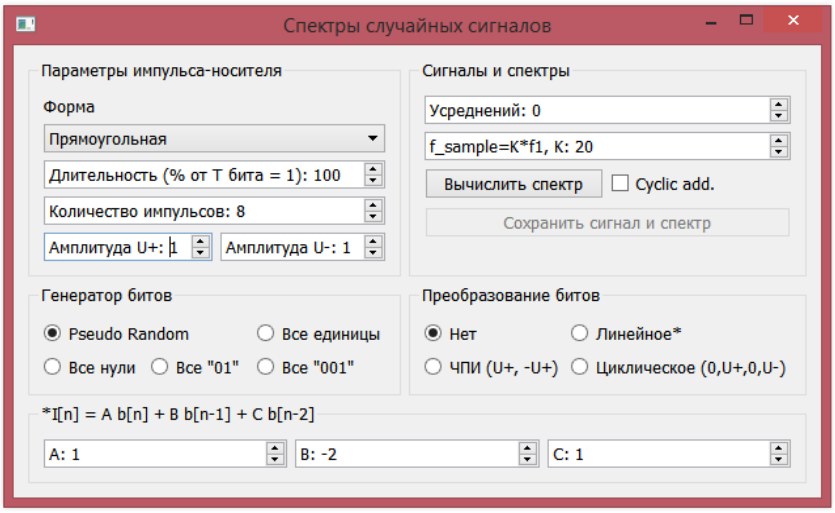


Рисунок 42: Программный макет для исследования спектров случайных сигналов

Панель **Генератор битов** определяет шаблон генерации битов. Например, шаблон **Pseudo Random** означает равновероятный и независимый поток битов. Остальные шаблоны не нуждаются в комментариях и задают неслучайный поток битов.

Панель **Преобразование битов** задает способ преобразования последовательности чисел.

Пункт **Линейное** означает следующее преобразование

$$I_n = A\,b_n + B\,b_{n-1} + C\,b_{n-2}\ , \qquad (60)$$

в котором параметры A, B и C задаются на панели ниже. При появлении в (60) отрицательных индексов считается, что биты периодически продолжены с периодом, равным количеству импульсов.

Пункт **ЧПИ (U+,−U+)** задает код с чередованием полярности импульса: если передается бит **1,** то текущая полярность импульса изменяется на противоположную, если **0,** то уровень равен нулю (импульс отсутствует). Первый бит кодируется как U+ и 0 для битов **1** и **0,** соответственно.

Пункт **Циклическое…** задает циклическое кодирование уровнями (0,U+,0,U−): при поступлении бита **1** уровень изменяется на следующий по кольцу, при поступлении **0** — повторяет текущий. Первый бит кодируется как U+ и U− для битов **1** и **0,** соответственно.

На панели **Сигналы и спектры** задаются такие параметры, как

- количество реализаций для оценки спектра мощности; ноль означает оценку по одной реализации;
- число точек на длительность импульса.

После нажатия на кнопку **Вычислить спектр** открывается окно с графиком спектра плотности мощности (дБВт/Гц). Пересчет в децибелы ведется согласно

$$10\lg\left[\Phi(f_k)\right] \ ,$$

где $f_k = \frac{k}{N T_d}$ — текущая частота, $N T_d$ — длительность реализации, состоящей из *N* отсчетов.

Спектральная плотность мощности сигнала g_i вычисляется через дискретное преобразование Фурье

$$\Phi(f_k) = \frac{T_d}{N}\left|c_k\right|^2 \ , \quad c_k = \sum_{0 \le i < N} g_i e^{-j\frac{2\pi ik}{N}}$$ (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 согласно кольцу (1, 2, 3).

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

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

Это удобно использовать при построении графиков в сторонней программе (Excel, Mathcad, gnuplot,...). Для этой цели можно использовать готовый файл **reader.mcd.**

4. Порядок выполнения работы

1. Вычислить аналитически спектральную функцию и ее квадрат модуля для импульса треугольной формы длительностью T и амплитудой A. Сравнить скорости убывания спектральной плотности (энергии) в зависимости от частоты для импульсов трех форм: прямоугольной, треугольной и косинусной (косинус на пьедестале).
2. Запустить приложение **spm.** Выставить 10000 усреднений, 16 импульсов. Остальные параметры не менять. Нажать **Вычислить спектр.** Сохранить графики сигнала и спектра в файлы **signal** и **spectr,** не забыв переименовать их (чтобы они в дальнейшем не перезаписались).
3. Подобрать аналитически амплитуды **треугольного** и **косинусного** импульсов так, чтобы площади, ограниченные этими импульсами и осью времени, совпали с площадью прямоугольного импульса (длительности всех импульсов одинаковы). Выставить найденные амплитуды в приложении и построить два сигнала и два спектра (последовательно), сохранив их в файлы (и не забыв переименовать).
4. Извлечь из сохраненных файлов спектры и построить их на одном графике в сторонней программе. Здесь же построить асимптотические линии спада мощности в зависимости от частоты $1/f^k$. Частоты отложить в логарифмическом масштабе.
5. На основании (41) и (42) (или только (43)) определить зависимость спектральной плотности **энергии** на нулевой частоте $\Phi(0)$ от площади фигуры, ограниченной сигналом $g(t)$ и осью времени.
6. Показать (пояснением к рисунку) на построенных спектрах то, что площади трех импульсов равны. Экспериментально определить зависимость спектральной плотности **мощности** на нулевой частоте от длительности прямоугольного импульса, взяв, например, $T = T_{бит}$ (100%), $T = T_{бит}/2$ (50%), $T = T_{бит}/4$ (25%).

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

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 приложения (16 импульсов, 10000 усреднений, амплитуды по единице). Показать на спектре то, что данный код обладает хорошей тактовой самосинхронизацией.
8. Аналитически вычислить спектр мощности для преобразования $I_n = Ab_n + Bb_{n-1} + Cb_{n-2}$, подставив $A = C = 1$, $B = -2$. Числа b_n обладают теми же корреляционными свойствами, что и в (49). Построить два спектра — с преобразованием и без него — и проверить правильность вычисленного преобразования, взяв прямоугольный импульс-носитель.
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).** Частоту измерять в долях $1/T$. Получить результат графически, нарисовав от руки основную (первую) гармонику **на графике сигнала.**
11. Выбрать преобразование **ЧПИ...,** импульс прямоугольный, 16 равновероятных битов (амплитуда U− автоматически установится в ноль). Построить **одну реализацию сигнала** и спектр (10000 усреднений). Графически доказать отсутствие постоянной составляющей для шаблонов **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 дает 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.

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. 43).

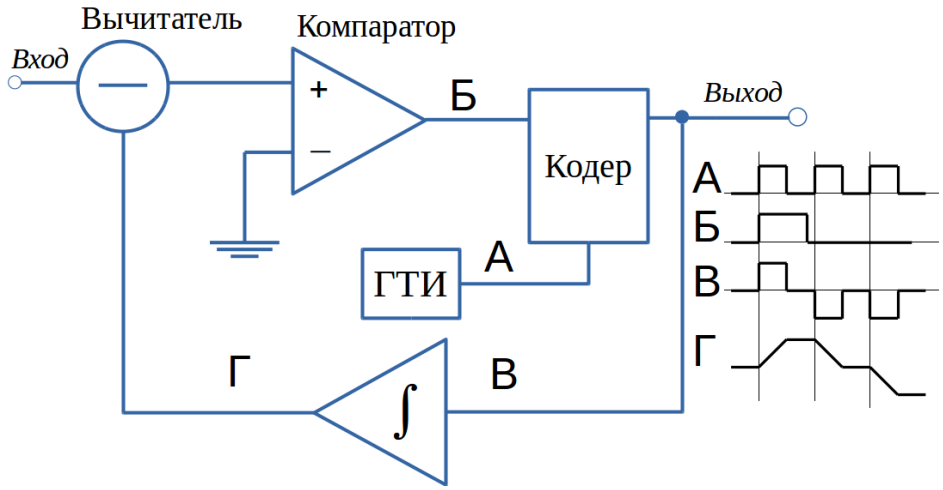


Рисунок 43: Функциональная схема простейшего дельта-модулятора

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. 43 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 несколько 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 технике передачи информации встречается сплошь и рядом: в 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; интересно, а как звучит 0 Гц?..).

Демодулятор строится по еще более простой схеме (рис. 44), в которую входят лишь аналогичный интегратор и фильтр нижних частот (ФНЧ).

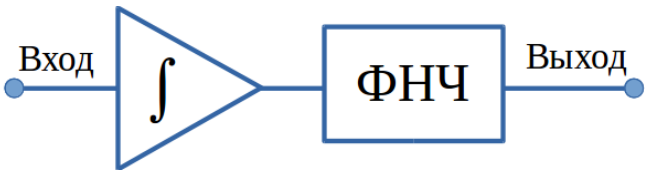


Рисунок 44: Функциональная схема простейшего демодулятора

ФНЧ предназначен для сглаживания изломов напряжения с выхода интегратора. Полоса пропускания ФНЧ подбирается исходя из ширины спектра полезного сигнала, поступающего на вход модулятора.

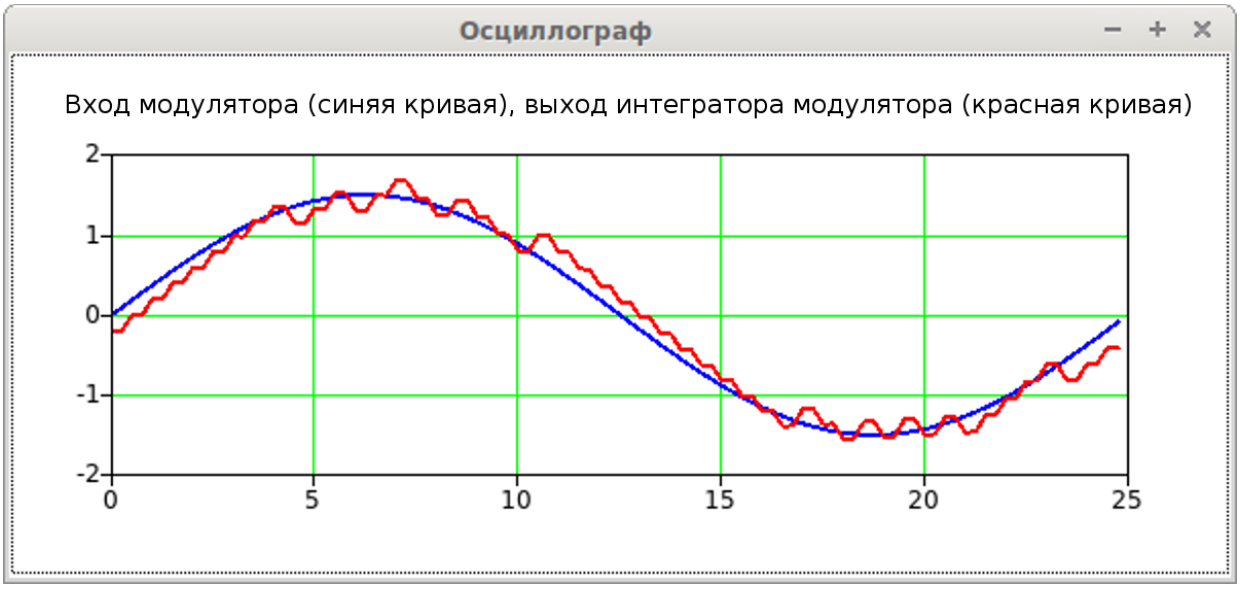
Чтобы наглядно показать процесс модуляции-демодуляции, ниже приведены несколько осциллограмм при кодировании синусоидального сигнала с частотой 40 кГц и амплитудой 1,5 В.

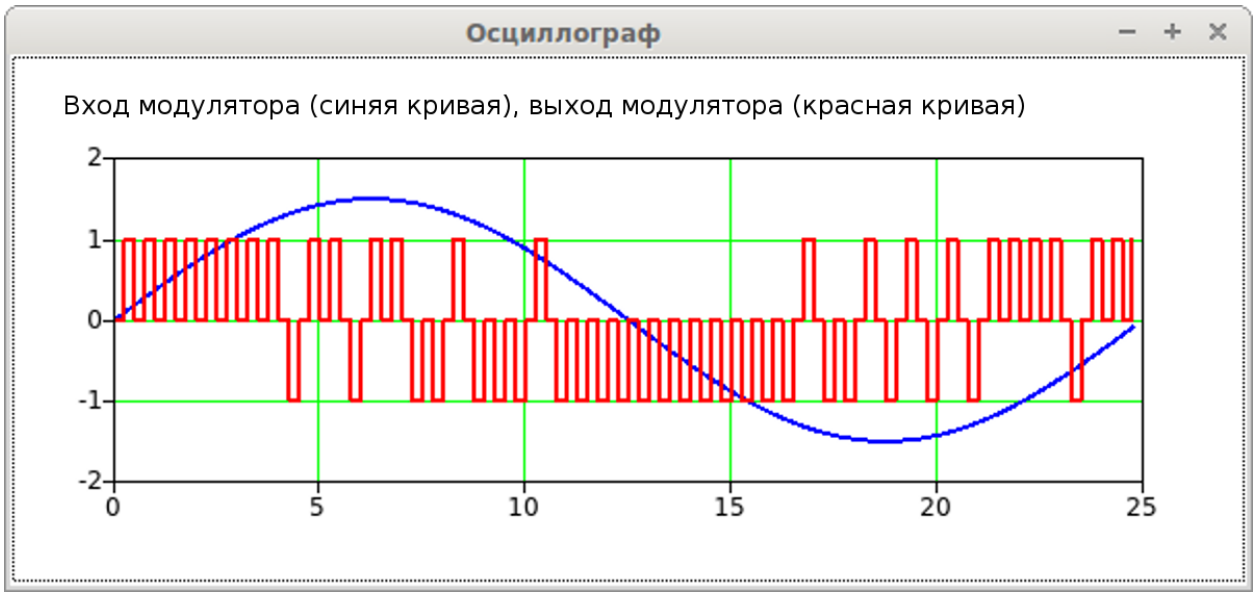
Единицы деления для всех осциллограмм: 5 мкс, 1 В.

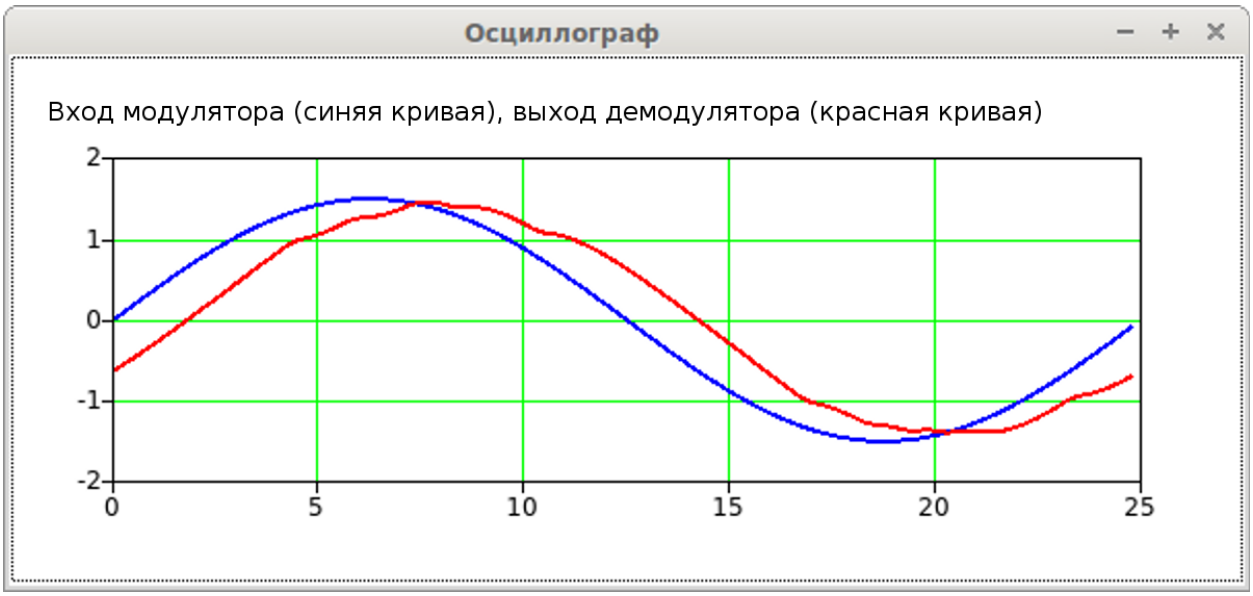
Из последней осциллограммы **(выход демодулятора)** следует, что восстановленный сигнал несколько задержан относительно исходного. При моделировании никаких задержек не вносилось, поэтому задержка произошла за счет ФНЧ.

Осциллограммы сняты в установившемся режиме работы системы связи.

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




Осциллограф
Вход модулятора (синяя кривая), выход 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

Программный макет (рис. 45) является приложением, написанным на языке C++ с использованием библиотек **Qt** [3].

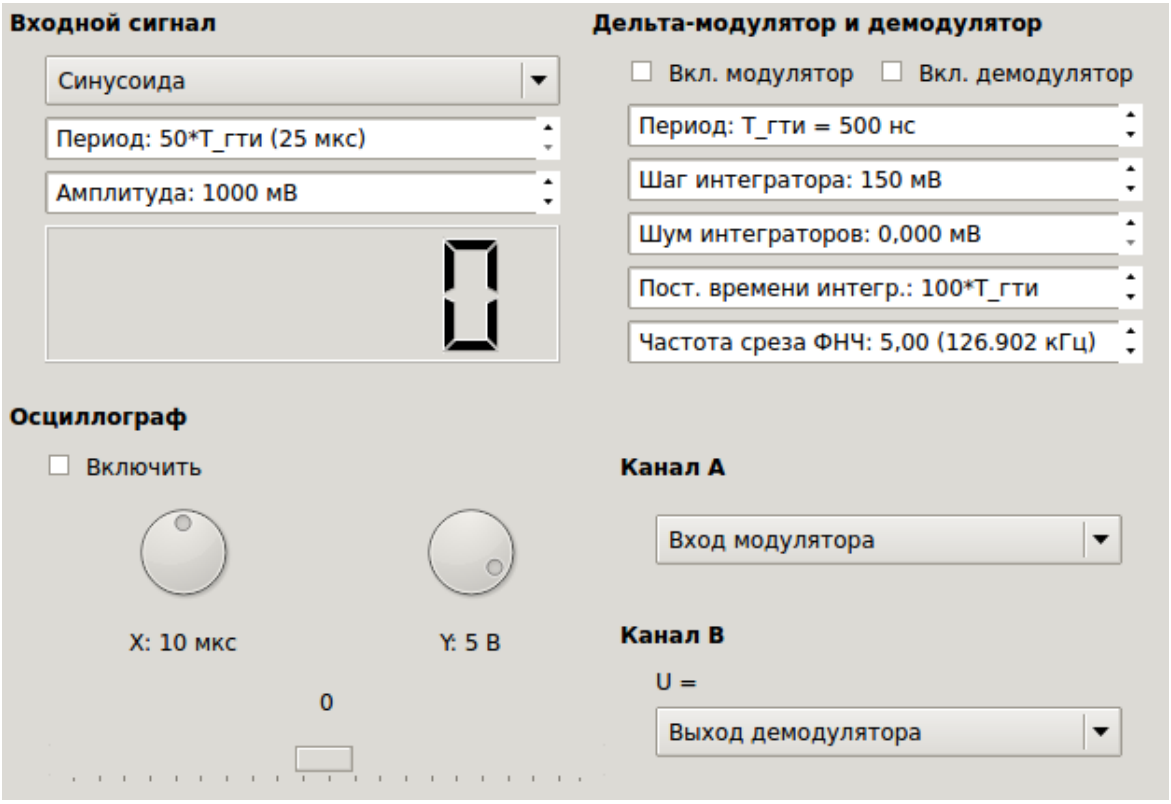


Рисунок 45: Лабораторный макет «Дельта-модуляция»

В главном окне макета задаются:

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

Поддерживаются следующие входные сигналы:

- 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 (−3 дБ).

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]

$$f_c = \frac{1}{\pi T_s} \arctan\left(\frac{T_s}{2\tau}\right) = \frac{1}{\pi T_s} \arctan\left(\frac{1}{2k}\right) \text{ , Гц.} \qquad (61)$$

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

Осциллограф имеет:

- 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 синхронизации).

В канале **B** автоматически измеряется среднее квадратическое значение (СКЗ) за период сигнала.

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

- Вход или выход модулятора;
- 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

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

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 $5 \cdot 5 = 25$ 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, демодулятор и осциллограф. Входной сигнал выключить**.** Ко входу **B** осциллографа подсоединить **Выход интегратора модулятора.** Развертку по **X** выставить на 2 мкс/дел., развертку по **Y** — на 0,2 В/дел.
2. Зарисовать осциллограмму на выходе интегратора модулятора, а также на входе модулятора. Указать период следования тактовых импульсов и величину шага квантования по уровню, и сравнить их с установленными программно (визуально для себя убедиться в правильности работы схемы).
3. Вычислить СКЗ напряжения на выходе интегратора аналитически (за период сигнала, равный двум тактам ГТИ), считая что форма этого напряжения — трапеция (рис. 46). 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. 46).

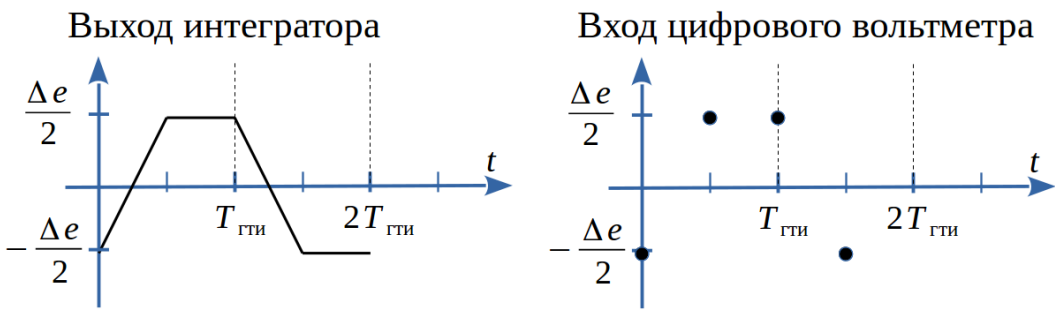


Рисунок 46: К измерению СКЗ напряжения цифровым вольтметром

4. Подать в канал **A** сигнал с выхода модулятора**.** Развертку по **Y** выставить по удобнее (1 В/дел.). Зарисовать осциллограмму и в отчете пояснить логику процесса интегрирования знакочередующихся прямоугольных импульсов.
5. Подать в канал **A** сигнла с выхода интегратора демодулятора, а в канал **B** сигнал с выхода 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 согласно (61) $k=\tau/T_s$,

определить все значения напряжения на выходе цифрового ФНЧ в установившемся режиме (их будет четыре*). Сравнить с результатом измерения осциллографом. 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 облегчения выполнения данного задания рекомендуется использовать программы символьных вычислений **(Mathcad, SymPy, ...).** Ответ доказать программой или на отдельном листочке-приложении к отчету.

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

7. Развертку по **X** выставить на 5 мкс/дел., а по **Y** — на 0,5 В/дел. Подать на вход меандр амплитудой 1 В. В канал **A** подать сигнал со входа модулятора, а в канал **B** — с выхода демодулятора. Зарисовать осциллограмму 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 пронаблюдать осциллограммы выходного сигнала при разных частотах среза ФНЧ.
8. Переключить входной сигнал на **Синусоиду,** развертку по **Y** на 1 В/дел., а развертку 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 СКЗ 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, развертку 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 СКЗ и построить график зависимости СКЗ от уровня шума.

10. Выставить частоту среза ФНЧ на 10,00 (63,609 кГц). Сделать предыдущий пункт, но канал **В** 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?

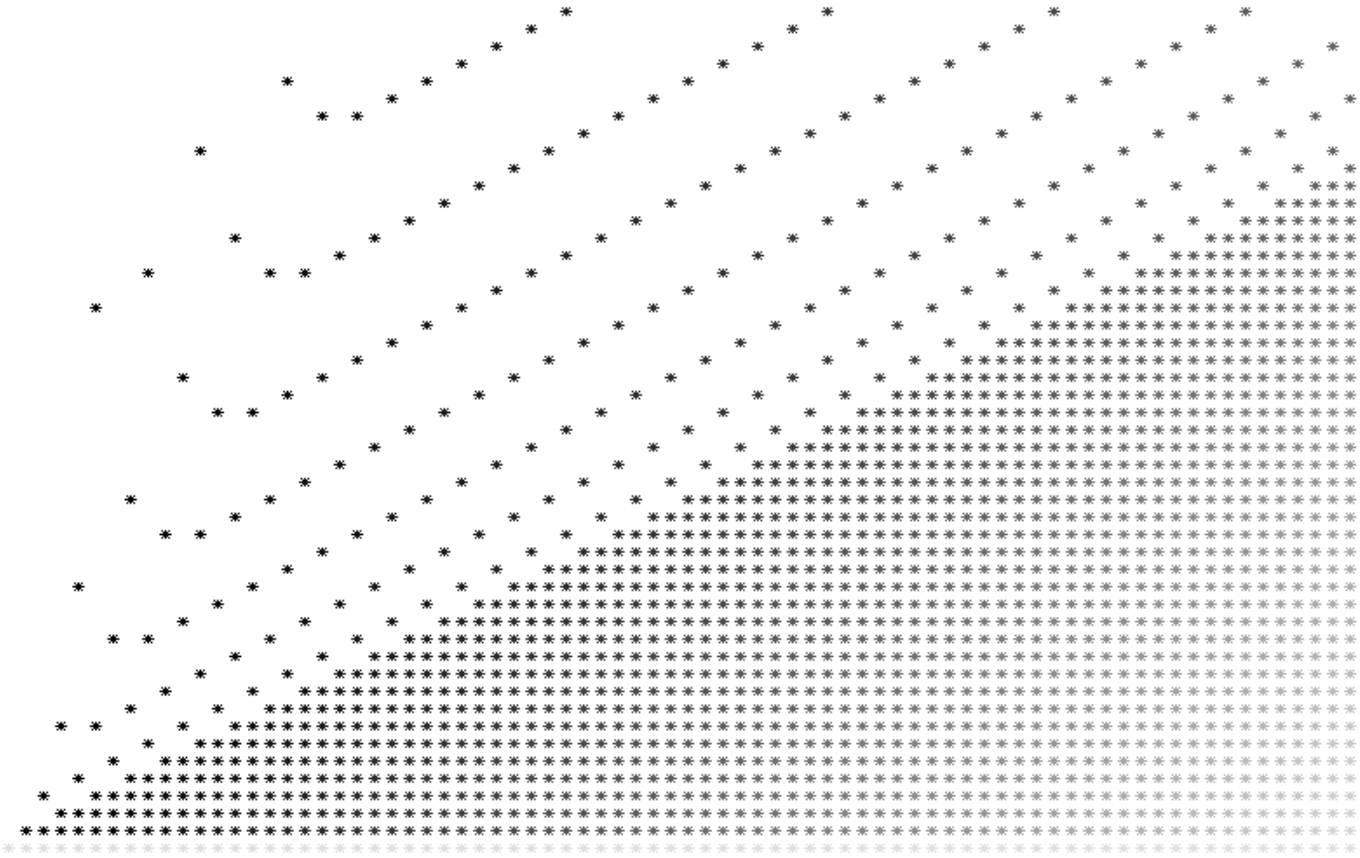
6. Литература

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 люди ведут счет яблок, грибов, шкур животных и других предметов. По этой причине числа 1, 2, 3,… называются натуральными (они определяют количество **цельных** предметов).

Если рассмотреть счет яблок, то они бывают разными по весу, форме, вкусу и тому подобным параметрам, однако 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.

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.

2. Сведения из теории

2.1 АЦП последовательного счета

К процедуре последовательного счета приводит задача измерения длины некоторого отрезка. Здесь возникают два вопроса: **в чём 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.*

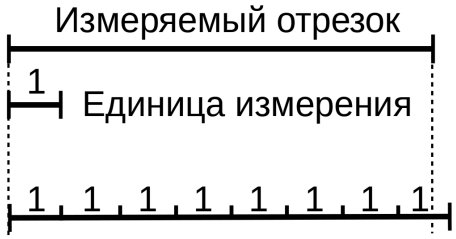


Рисунок 47. Принцип последовательного счета

Если имеется некоторый отрезок, длину которого необходимо измерить, и выбрана единица измерения, то процедуру последовательного счета можно показать рисунком 47. Счет здесь идет слева направо, а результат измерения шаг за шагом выдается в виде набора 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 каждый измерительный такт должен выдавать **один из двух** сигналов: **больше** или **меньше.** В логических схемах этим сигналам соответствуют логические уровни, условно обозначаемые **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 логический импульс на выходе 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.

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 вместо 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 различать

10 уровней, поставив набор компараторов с 10-уровневым выходным сигналом.

Технически проще 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 этом соответствуют нули, которые не имеет смысла рисовать, так как они никогда не закончатся.

Мы знаем как минимум две системы с основанием кода 2: позиционную двоичную систему счисления, когда вес следующего разряда удваивается, и непозиционную, когда все веса равнозначны. Если в последней системе длина кодовых слов максимальна, то спрашивается, а в двоичной позиционной минимальна ли? Если да, то есть ли между ними промежуточные системы счисления с таким же основанием кода?..

Все эти вопросы относятся к АЦП *поразрядного взвешивания.*

2.2 АЦП поразрядного взвешивания

Рассмотрим двоичную позиционную систему счисления, когда вес каждого разряда равен степеням двойки

$$N = a_{M-1}2^{M-1} + a_{M-2}2^{M-2} + \ldots + a_1 2^1 + a_0 2^0 \ , \qquad (62)$$

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, поэтому сознательно не рассматриваются отрицательные степени двоек (считайте, что они неконструктивные, хотя это не догма…).

Оказывается, что двоичное представление (62) позволяет вместить наибольшее количество чисел при заданном M . Это количество (всё множество) равно 2^M . **Троичная логика** с взвешивающими коэффициентами $\{-1,0,+1\}$ 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.

Минус 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 плюсом этого метода является повышенная помехоустойчивость.

Минус двоичного поразрядного взвешивания — **неразличимость 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 из-за случайного сбоя триггера старшего разряда АЦП может выдать **0**01 вместо **1**01, что может привести к печальным последствиям.

Для двоичной системы счисления все кодовые слова являются разрешенными: в этом и заключается её минус.

Обнаруженный выше факт хорошо известен как компромисс между скоростью передачи сообщения и надежностью его доставки. Поэтому после обычного АЦП (с нулевой избыточностью) используют разные корректирующие коды (коды канала).

Оказалось, что имеются процедуры двоичного **АЦП со встроенной избыточностью,** уровень которой может регулироваться [1, 2]. В этом случае дополнительные корректирующие коды могут и не потребоваться.

Но для начала давайте докажем, что обычное двоичное АЦП (62) 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,** так, чтобы количество гирь было минимальным.

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 выбирается как можно большим, но так, чтобы уравновесить 2, которую с помощью первой гири не уравновесишь. Отсюда следует, что вес второй гири равен 2.

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 $(1+2)+1=4$. Значит вес третьей гири равен 4.

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 16-битовом кодировании затратно: в этом случае частота дискретизации 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 обычные двоичные коды, 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 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, кроме профессиональных (серверных). 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 заново. При действительно случайном сбое, а не отказе, вторая попытка с большой вероятностью пройдет без ошибок.

В процедуре поразрядного взвешивания есть один малозаметный параметр, в котором и кроется возможность **постепенного** перехода от классического АЦП поразрядного взвешивания к АЦП последовательного счета. Этим параметром является *асимметрия процесса измерения* [1, 2].

Чтобы понять асимметрию процедуры взвешивания, представим ее механическую аналогию — взвешивание на весах 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** — ровно (бит **1**).

АЦП — цифровое устройство, которое работает по тактам, поэтому его инерционность (время, за которое убирается гиря), в принципе, может быть любой кратной такту. Эталонные веса (гирьки) подаются 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 (с минимальной разрядностью) процедурой АЦП с двоичной логикой (**много-мало**) является та, которая соответствует классической двоичной системе счисления. Если инерционность бесконечная, 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]

$$1, 1, 2, 3, 5, 8, 13, 21\ldots \quad , \tag{63}$$

который известен как последовательность Фибоначчи. Покажем это.

Вес первой гири, понятное дело, равен единице (куда без нее…).

Попробуем взять вторую гирю весом 2, подобно двоичным гирям. Тогда **не обеспечится** измерение числа **1,** так как положив на правую чашу гирю весом **2**, получим перебор (компаратор выдаст **много**). Убрав эту гирю,

затратим на это один такт. За это время первая гиря уйдет 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 числа **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 всех имеющихся гирь равна 4. Число **0** также уравновесится не более чем за три такта, потому что на третьем такте $2,\bar{2},1$ компаратор выдаст **много,** откуда следует, что на входе **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 введения ограничения на процесс измерения, так как если становятся возможными пропуски некоторых гирь, то должен быть своеобразный запас гирь.

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.

Возьмем гирю весом **4** и методом перебора найдем промах: это измерение 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 гирь? 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 её требуется компенсировать оставшимися гирями. Максимальное измеряемое число, которое приведет к пропуску w_{k-1}, равно w_k-1, и для его

уравновешивания требуется, 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 соответствует ряду 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

$$\begin{gathered} w_1=w_2=\ldots=w_{p+1}=1 \\ w_k=w_{k-1}+w_{k-1-p},\ k>p+1, \end{gathered} \qquad (64)$$

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 ранее рассмотренной процедуры инерционного на один такт АЦП.

2.3 Способ обнаружения и исправления сбоев

Обнаружение сбоев возможно за счет равенства

$$w_k=w_{k-1}+w_{k-2} \ , \qquad (65)$$

связывающего веса гирь. Это равенство ведет к неоднозначности кодовых слов, так как две рядом стоящие единицы могут быть заменены одной, старшей 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 (65) 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 установит 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, он не даст сбой типа **невыключение.**

Таким образом, корректный (при отсутствии сбоев) десятибитовый код числа **81** будет 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. 48).

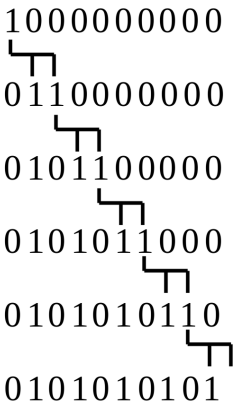


Рисунок 48: Многозначность кодирования числа 55

За счет равенства (65) 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.

На выходе рассматриваемого АЦП возможны так называемые *метрологически разрешенные* кодовые слова и *метрологически запрещенные* [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. Однако, 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 дает 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 должны идти все нули [2].

Максимально взвешиваемое число для рассматриваемого АЦП равно сумме двух старших весов, то есть **89.** При попытке закодировать **90** потребуются 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 являются случайными и независимыми и происходят с вероятностями p_1 и p_0 , где индекс 1 означает выдачу бита **1** вместо бита **0,** а индекс 0 — выдачу бита **0** вместо бита **1.**

Отказ триггера задается принудительно, то есть неслучайно.

3. Описание лабораторного макета

Лабораторный макет — приложение, написанное на языке C++ с использованием библиотек **Qt** [3], рис. 49.

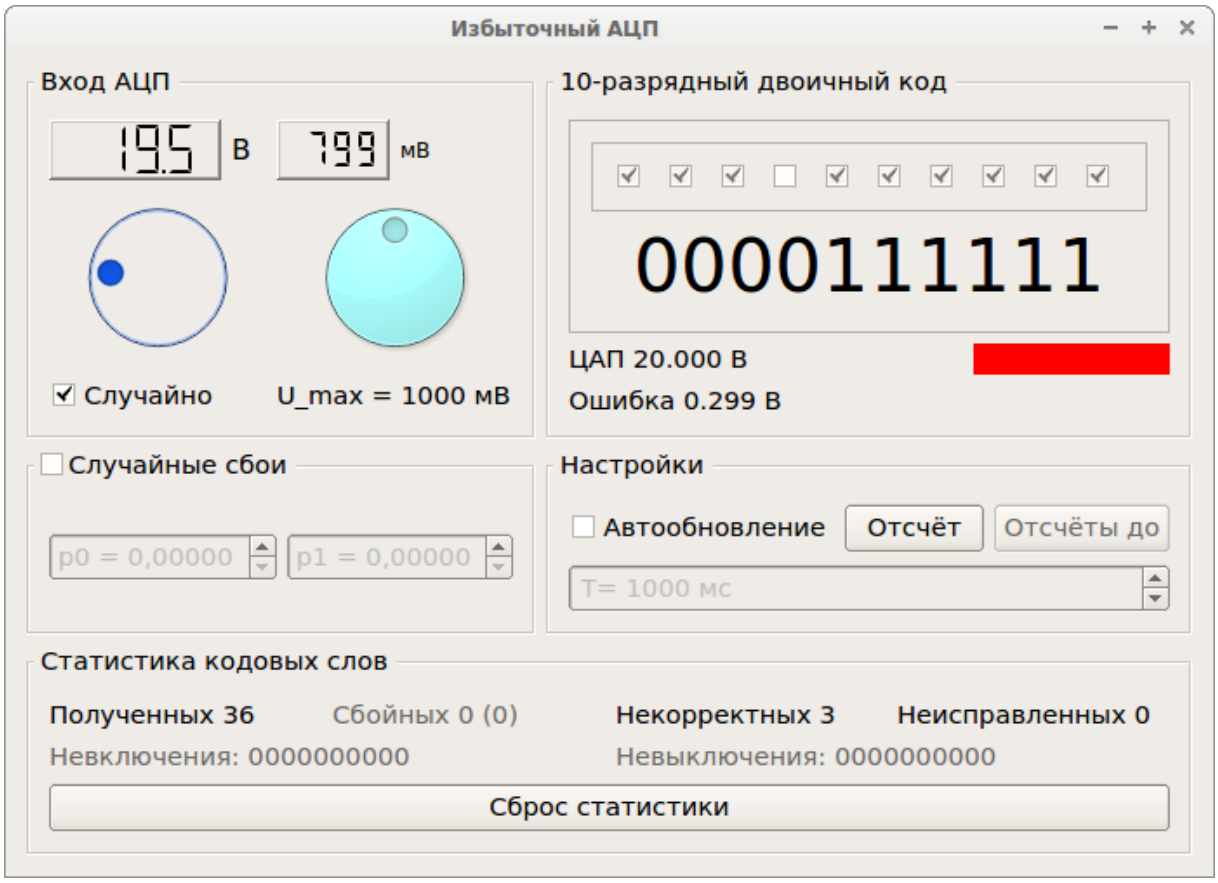


Рисунок 49: Лабораторный макет «Избыточный АЦП»

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 подается 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, подняв 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 поднят) или сброса (флажок опущен) соответствующих триггеров. Серые флажки (третье состояние) соответствуют освобождению соответствующих триггеров. Ниже двоичного кода показаны:

- Значение напряжения 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 вероятности сбоев типа **невключение** p_0 и типа **невыключение** p_1. Вероятность, меньшая $1/32767$, считается нулевой.

Панель **Настройки** предназначена для задания режимов ручного взятия отсчетов (кнопка **Отсчет**) и автоматического, если поднят 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.

На панели **Статистика кодовых слов** отображаются:

- Общее количество полученных кодовых слов;
- Количество сбойных кодовых слов (только случайные сбои);
- Количество метрологически некорректных кодовых слов (когда индикатор загорается красным);
- Количество неисправленных кодовых слов (когда ошибка АЦП больше единицы или меньше нуля).

Текстовые поля **Невключения 0000000000** и **Невыключения 0000000000** с помощью 1 показывают биты, в которых

произошли соответствующие случайные сбои. В текстовом поле **Сбойных 0 (0)** число в скобках означает:

(1) — наличие сбоя,

(0) — отсутствие сбоя,

при 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 формируются 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 в данном случае роль шума?
2. Выключить шум. Выбрать случайное входное напряжение на интервале (0–89) В. Для этого поднять флажок **Случайно.** Выключить **Автообновление** на любом ненулевом кодовом слове, и привести в отчете пошаговую процедуру поразрядного взвешивания выбранного числа. Сравнить результат ручного кодирования 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 залипание триггера в 0 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 залипание триггера в 1 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 **Отсчёты до** до тех пор, пока не появится метрологически запрещенное кодовое слово (об этом уведомит красный индикатор), плюс необходимо, чтобы при этом ошибка квантования была в интервале [0…1). По панели **Статистика кодовых слов** узнать 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 **Отсчёты до** нажимать до тех пор, пока не получится неисправленный сбой. Выписать номера не выключившихся 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 опусканием флажка **Автообновление.** 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. Для пунктов а), б) и в) удобно составить три 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

$$\begin{pmatrix} x & x & x & x & x & x & x & x & x & x & x & x \end{pmatrix} \text{ (12 разрядов)},$$

$$\begin{pmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \end{pmatrix}, \quad \begin{pmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{pmatrix}, \quad \begin{pmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{pmatrix},$$

$$\begin{pmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \end{pmatrix}^T,$$

$$\begin{pmatrix} x & x & x & x & x & x & x & x & x & x & x & x \end{pmatrix}^T \text{ (один разряд)},$$

чтобы поразрядный код дал максимальное количество кодируемых чисел? Если трудно, ответьте сначала на вопрос 6.

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