1. ЛАБОРАТОРНАЯ РАБОТА № 1

ПОМЕХИ И ИХ ВОЗДЕЙСТВИЕ НА БЛОКОВЫЕ КОДЫ

Цель работы: Исследовать влияние помех на передачу информации с применением корректирующих блоковых кодов.

Теоретическая часть

1.1. Общая модель телекоммуникационной системы

Телекоммуникационная система [7] соединяет источник данных с их получателем посредством канала связи, который может быть радиоканалом или проводным каналом. Структурная схема телекоммуникационной системы приведена на рис. 1.1.

Информационные последовательности источника данных непосредственно поступают на кодер источника, который производит наиболее оптимальное их представление, осуществляя так называемое "сжатие" информации. В кодере канала обеспечивается внесение дополнительных символов в кодовую последовательность источника, т.е. создается необходимая избыточность для возможного в дальнейшем устранения ошибок. Затем модулятор преобразует поступающие на него последовательности в соответствующие высокочастотные сигналы, хорошо распространяющиеся в физическом канале, где на них могут воздействовать помехи, чаще всего в виде высокочастотных колебаний. Демодулятор формирует на выходе символы принятого кодового слова канала, которые отличаются от своих значений на входе модулятора в виду наличия помех в физическом канале. Декодер канала осуществляет за счет введенной в сообщение избыточности исправление возникших ошибок и образует на своем выходе передаваемые последовательности источника без искажений. К получателю данных информационный блок

поступает, пройдя декодер источника, где осуществляется "расширение" ранее "сжатых" информационных последовательностей.

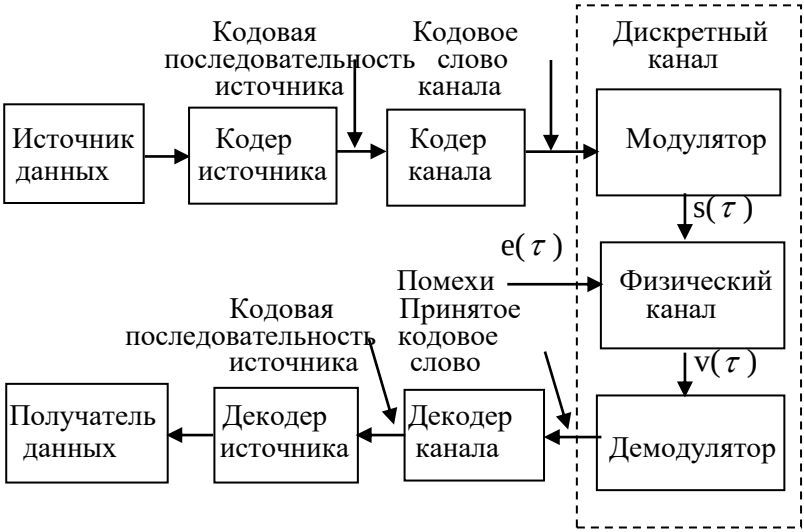


Рис. 1.1. Структурная схема телекоммуникационной системы

Далее нас будут интересовать, прежде всего, процессы, осуществляемые в кодере и декодере канала, позволяющие устранять ошибки, возникающие на выходе демодулятора, который принимает решения по каждому из принятых символов передаваемых последовательностей в условиях воздействия помех на физический канал. На каждом интервале передачи символа демодулятор принимает решение относительно его значения и подает на декодер канала выходную канальную последовательность. Если на демодуляторе выходная канальная последовательность квантуется на два уровня, принимающих значения 1 и 0, то говорят, что демодулятор принимает «жесткое» решение. Если выход демодулятора квантуется более чем на два уровня – демодулятор принимает «мягкое» решение.

1.2. Ошибки в дискретном канале связи

Под дискретным каналом связи понимают совокупность физического канала с модулятором и демодулятором (рис. 1.1), где входные и выходные сигналы являются дискретными случайными последовательностями.

При воздействии помех $e(\tau)$ по времени τ на физический канал демодулятор принимает неправильные решения и на его выходе в символах кодовых комбинаций появляются ошибки. В общем случае действие помех $e(\tau)$ можно описать с помощью оператора W, такого, что $v(\tau) = W[s(\tau), e(\tau)]$,

где $v(\tau)$ – сигнал на входе демодулятора, а $s(\tau)$ – сигнал на выходе модулятора.

В физическом канале связи помеха $e(\tau)$, взаимодействуя с сигналом $s(\tau)$, образует сигнал, пораженный помехой. В частном случае, когда $v(\tau) = s(\tau) + e(\tau)$ и $e(\tau)$ не зависит от $s(\tau)$, то помеха называется аддитивной [7].

Если оператор W представляется в виде произведения, так что $v(\tau) = \mu(\tau)\, s(\tau)$, где $\mu(\tau)$ – некоторая случайная функция, то помеха называется мультипликативной. Когда $\mu(\tau)$- медленный (по сравнению с $s(\tau)$), процесс, то влияние, вызываемое мультипликативной помехой, обуславливает замирание сигнала, которое приводит к группированию ошибок. В случае аддитивной помехи ошибки - независимые. В реальных каналах связи могут действовать как аддитивная, так и мультипликативная помехи и, когда $v(\tau) = \mu(\tau)\, s(\tau) + e(\tau)$, то ошибки, вызываемые таким воздействием на сигнал, можно назвать смешанными.

С учетом воздействия помех дискретный канал может быть без памяти и с памятью [8].

Если в любой момент времени вероятность появления символа на выходе дискретного канала зависит только от символа на входе канала для всех пар символов на входе и

выходе, то такой канал называется каналом без памяти или каналом с независимыми ошибками. Канал, в котором каждый символ выходной последовательности зависит как от соответствующего символа на входе, так и от прошлых входных и выходных символов, называется каналом с памятью или каналом с группирующимися ошибками.

Если действующими значениями на входе и выходе дискретного канала являются символы 0 и 1, а вероятности перехода 0 в 1 и 1 в 0 одинаковы, то такой канал называется двоичным симметричным каналом (ДСК). Он полностью характеризуется по выходу вероятностью ошибки на символ p и вероятностью появления символа без ошибки равной 1-p.Символы, применяемые для представления данных в двоичной системе счисления (например, 0 или 1), называют битами.

Диаграмма переходов в канале представлена на рис. 1.2.

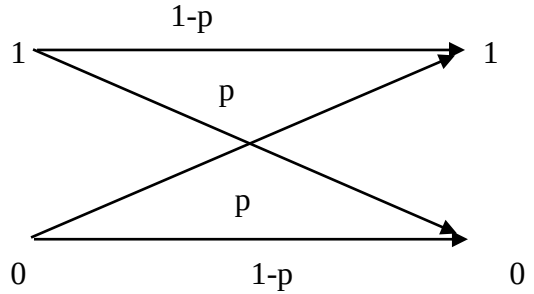


Рис. 1.2. Диаграмма переходов в двоичном симметричном канале

Двоичный симметричный канал можно представить при моделировании как канал, к которому подключен источник ошибок (рис. 1.3).

Этот источник выдаёт случайную последовательность ошибок (..., E_{i-1}, E_i, E_{i+1}, ...). Каждое значение E_i складывается с соответствующим значением кодового слова V_i в двоичном канале по модулю 2. Там, где в последовательности ошибок

{E_i} стоит 1, передаваемый символ изменяется на обратный, т.е. в выходной последовательности {V_i^*} будет ошибка. Например, если V_i=1, E_i=1, то $V_i^* = V_i \oplus E_i = 1 \oplus 1 = 0$, т. е. канал полностью описывается статистической последовательностью ошибок E_i, где $E_i \in \{0,1\}$.

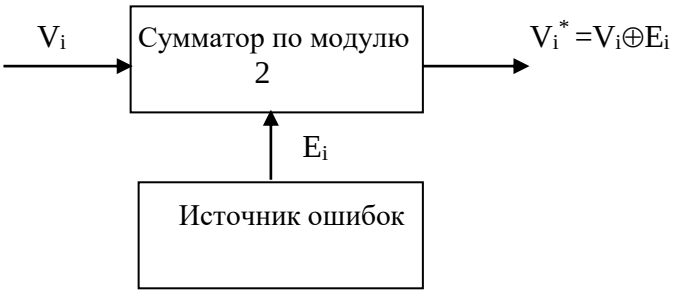


Рис. 1.3. Эквивалентная схема двоичного симметричного канала

Если ошибки при приёме отдельных символов кодовой комбинации происходят независимо друг от друга с вероятностью p, то вероятность безошибочного приёма кодовой комбинации из n элементов будет равна $(1-p)^n$, а вероятность ошибочного её приёма равна значению $1-(1-p)^n$. Вероятность одновременного числа ошибок t в информационной последовательности определяется значением p^t. Вероятность того, что в комбинации, имеющей n разрядов, n-t элементов не искажены, составляет $(1-p)^{n-t}$. Следовательно, вероятность одного заданного сочетания ошибок t в n-разрядной последовательности составляет величину $p^t(1-p)^{n-t}$.

Общее количество возможных комбинаций t-кратных ошибок в n-разрядной комбинации составляет

$$N_t = C_n^t = \frac{n!}{t!(n-t)!}.$$

Отсюда вероятность наличия в комбинации из n разрядов t ошибок в любом сочетании равна

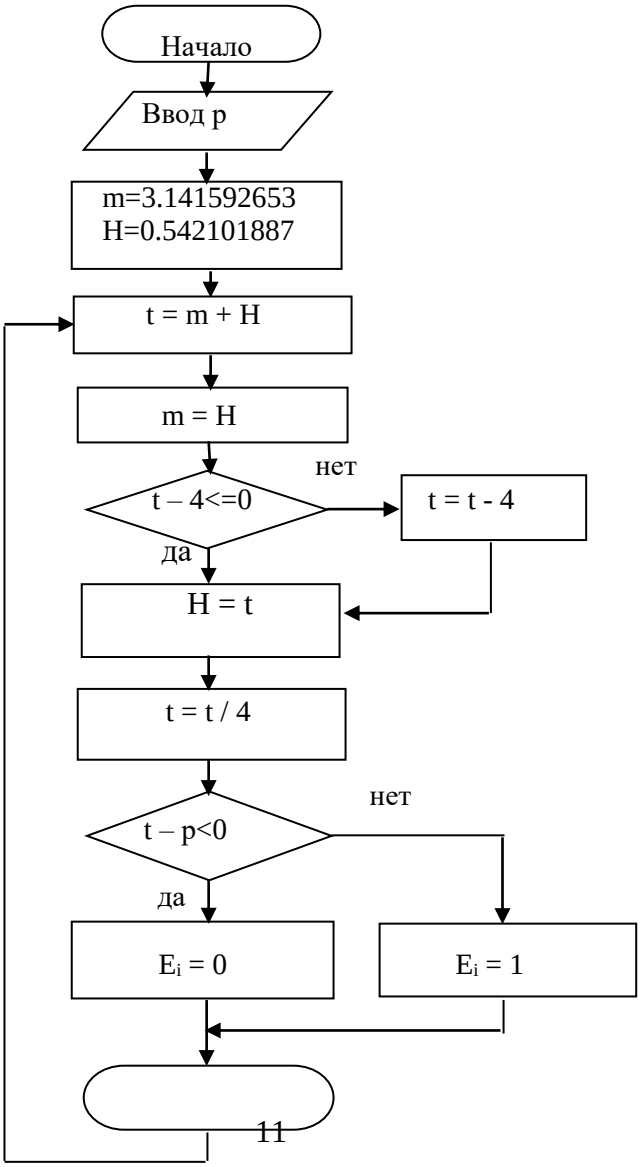
$$P_t = C_n^t p^t (1-p)^{n-t}.$$

При независимом распределении ошибок свойства канала по выходу как источника ошибок $\{E_i\}$ задаются единственным параметром – вероятностью ошибки p на символ.

В лабораторной работе для моделирования на ЭВМ последовательности ошибок $\{E_i\}$ используется специальная программа, структурная схема которой показана на рис. 1.4. Последовательность ошибок здесь получается на основе генератора случайных чисел с равномерным законом распределения, выбор которых происходит от задаваемой величины p.

Кроме независимых ошибок в канале могут возникать группирующиеся ошибки. Они образуются в каналах с памятью. Одной из основных причин таких ошибок являются перерывы, возникающие из-за плавного снижения уровня сигнала ниже порога чувствительности приёмника, когда приём сигнала фактически прекращается. Перерывы могут быть различной длительности, и некоторые из них способны даже вызывать прекращение сеанса связи. Кроме этого, причиной перерывов могут быть неисправности в аппаратуре, несовершенство её эксплуатации, проведение измерений и др. Перерывы и импульсные помехи являются основной причиной появления группирующихся ошибок при передаче дискретных сообщений по различным видам каналов связи. Импульсные помехи – это помехи, сосредоточенные по времени. Они представляют собой случайную последовательность импульсов, имеющих случайные амплитуды и следующих друг за другом через случайные интервалы времени, причём вызванные ими переходные процессы, не перекрываются во времени. Причины появления этих помех: коммутация соединений в электронной аппаратуре, наводки с высоковольтных линий, грозовые разряды, прием отраженных сигналов и т.п.

Для проведения лабораторных исследований группирующиеся ошибки в двоичных последовательностях можно получить на основе аппарата цепей Маркова (посимвольное описание) [2].



Конец

Рис. 1.4. Структурная схема моделирования последовательности ошибок $\{E_i\}$

Простейшей моделью, основанной на применении математического аппарата марковских цепей, является модель источника, предложенная Гильбертом [8]. В такой модели для частного случая двоичного симметричного канала предполагается, что канал может находиться в одном из двух состояний – хорошем (0) или плохом (1). В хорошем состоянии вероятность ошибки $P_0=0$, а в плохом вероятность ошибки $P_1 = \text{const}$.

Если при передаче символа сообщения на (i-1)-й позиции канал находится в хорошем состоянии, то на следующей позиции i он останется в том же хорошем состоянии с вероятностью P_{00} или же перейдёт в плохое состояние с вероятностью $P_{01} = 1 - P_{00}$. Наоборот, если на (i – 1)-й позиции канал находится в плохом состоянии, то на i-й позиции он останется с вероятностью P_{11} в том же состоянии, а с вероятностью $P_{10}=1-P_{11}$ перейдёт в хорошее состояние.

В такой модели последовательность ошибок полностью определяется матрицей переходных вероятностей

$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}. \qquad (1.1)$$

Приняты названия её элементов:

P_{00} – вероятность следования нуля за нулём,

P_{01} – вероятность появления единицы после нуля,

P_{11} – вероятность следования единицы за единицей,

P_{10} – вероятность появления нуля после единицы.

Значения вероятностей для матрицы (1.1) могут быть получены из графа марковского процесса (рис. 1.5).

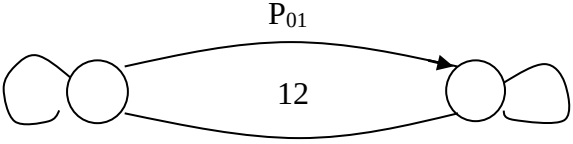


Рис. 1.5. Граф марковского процесса

Направления стрелок на рис. 1.5 обозначают переходы из одного состояния в другое, а состояния канала изображены в виде кружков с 0 и 1 внутри. Вероятность появления того или иного состояния определяется из системы уравнений, вытекающих из следования переходов на графе (рис. 1.5), т. е.

$$\begin{cases} P_0+P_1=1 \\ P_1\cdot P_{10}+P_0\cdot P_{00}=P_0 \\ P_1\cdot P_{11}+P_0\cdot P_{01}=P_1. \end{cases} \qquad (1.2)$$

Из системы уравнений (1.2) и упомянутых выше выражений, где $P_{01}=1-P_{00}$, а $P_{10}=1-P_{11}$, получаются значения для P_1 и P_0 , т.е.

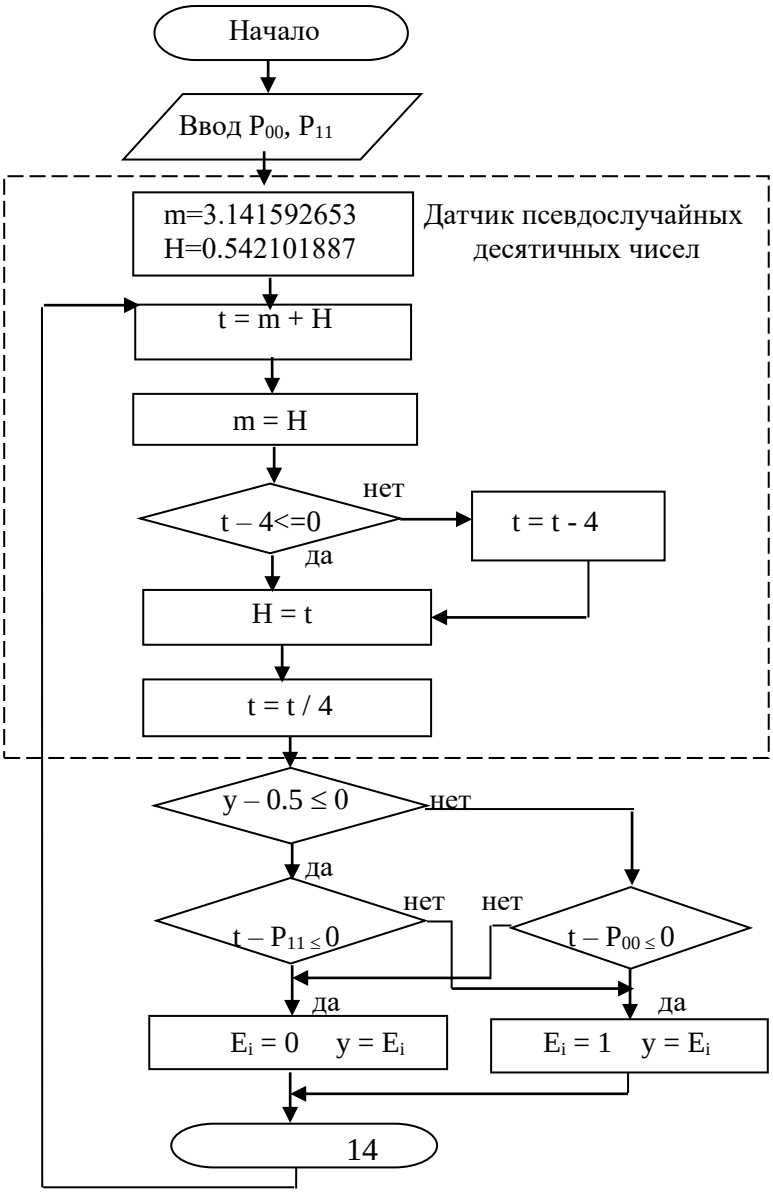
$$P_1 = \frac{P_{01}}{P_{01}+P_{10}}, \qquad P_0 = \frac{P_{10}}{P_{01}+P_{10}}.$$

Задаваясь вероятностью ошибки в канале $p=P_1$ и вероятностью P_{11}, отражающей значение группирования ошибок, из (1.2) можно найти величину P_{00} и по вероятностям P_{11} и P_{00} осуществить моделирование ошибок на ЭВМ. Для этого случая P_{00} будет определяться формулой

$$P_{00} = \frac{1-P_1(2-P_{11})}{1-P_1}. \qquad (1.3)$$

Моделирование последовательности группирующихся ошибок в лабораторной работе осуществляется по специальной программе, структурная схема которой представлена на рис. 1.6. В основе модели лежит генератор псевдослучайной последовательности десятичных чисел, которая затем в соответствии с матрицей переходных вероятностей (1.1) преобразуется в последовательность группирующихся ошибок $\{E_i\}$ в виде единиц, а неискажённые

позиции отражаются в виде нулей. Для того, чтобы последовательность начиналась с нулей, необходимо задаться начальным значением y = 0. При начальном значении y =1 моделируемая последовательность будет начинаться с единиц.



Конец

Рис. 1.6. Структурная схема моделирования последовательности группирующихся ошибок

1.3. Классификация помехоустойчивых кодов

Известно большое число помехоустойчивых кодов, которые способны обнаруживать ошибки, исправлять их или одновременно часть ошибок исправлять, а другие обнаруживать.

Коды способные исправлять ошибки называются корректирующими кодами и подразделяются на два больших класса: блоковые и непрерывные, которые в свою очередь делятся на другие виды [2,9,12,19], как показано на рис.1.7. Здесь приведена примерная схема классификации корректирующих кодов, которая только в основном отражает их многообразие.

В случае блокового кодирования последовательность информационного сообщения источника разбивается на части и каждой части ставится в соответствие определенная последовательность (блок) кодовых символов, которую принято называть разрешенной кодовой комбинацией или по-другому, кодовым словом. Непрерывные коды, формируемые на основе определенных рекуррентных соотношений, и получившие название сверточных кодов, характеризуются тем, что операции кодирования и декодирования осуществляются над непрерывной последовательностью символов без разбиения ее на блоки.

Существуют разделимые и неразделимые коды. При кодировании разделимыми кодами выходные последовательности состоят из символов, роль которых может быть отчетливо разграничена. Блоковые разделимые коды обозначаются как (n, k), где n- длина кода, k- число информационных символов, а величина r = n- k – число проверочных символов. При кодировании неразделимыми

кодами разделить символы выходной последовательности на информационные и проверочные невозможно без их предварительной обработки. Непрерывные коды также могут быть разделимыми и неразделимыми.

Наиболее обширный класс среди разделимых блоковых кодов образуют систематические или линейные коды, являющиеся групповыми. Основным отличием систематических кодов является то, что их проверочные символы представляют собой различные линейные комбинации информационных символов.

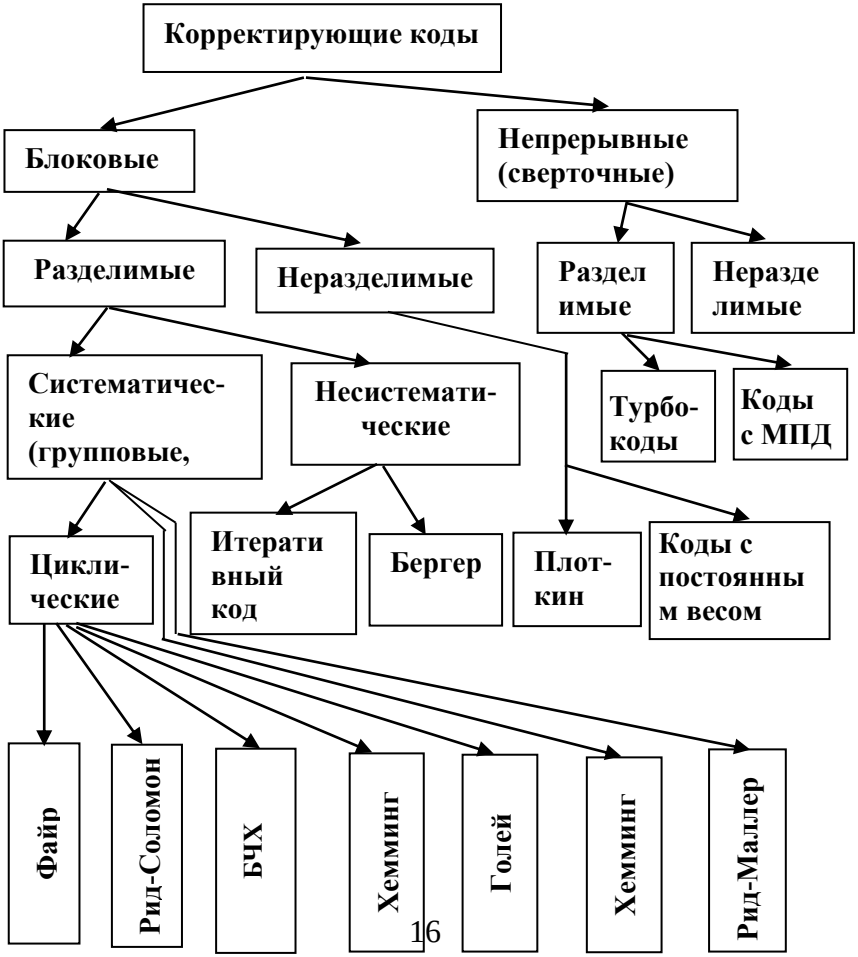


Рис. 1.7. Примерная классификация корректирующих кодов

Название "групповой код" обусловлено тем, что код является группой по отношению к операции сложения по модулю 2, т. к. сумма по модулю 2 любых двух кодовых слов также является кодовым словом, принадлежащим этой группе. К линейным кодам можно отнести код Рида-Маллера, допускающий мажоритарное декодирование со специально сформированной для этой цели порождающей матрицей.

Систематические коды могут быть циклическими, когда циклический сдвиг любого их кодового слова в регистре сдвига длиной n также является кодовым словом. Циклические коды, в свою очередь, подразделяются на простые и алгебраические. К алгебраическим относятся коды БЧХ, декодирование которых осуществляется методами линейной алгебры. Сюда также относятся коды Рида-Соломона, исправляющие пачки ошибок. Циклическими кодами, исправляющими пачки ошибок, являются коды Файра. Наиболее часто в практике корректирующего кодирования применяются коды: БЧХ, Рида-Соломона, Хемминга, сверточные, каскадные (формируются, как правило, из кодов Рида-Соломона и Хемминга). Кроме этого, в последнее время, используются турбокоды, получаемые из каскадирования двух и более сверточных кодов (чаще всего разделимых), а также коды с многопороговым декодированием (МПД) [9].

1.4. Принцип построения корректирующих кодов

В системах передачи данных, не использующих корректирующее кодирование, применяются, как правило, безызбыточные коды. Например, если необходимо передавать в цифровом виде буквы алфавита, то можно использовать

пятизначный код. Этот код содержит 2^5 значений, и ими можно пронумеровать перечень букв. Сочетание букв, например А, Б, В, в этом случае представляется в виде числовой последовательности из единиц и нулей, т. е. 00001, 00010, 00011.

В случае воздействия помех искажение символов в кодовых последовательностях приведет к переходам одних букв в другие, и правильное восприятие текста будет невозможно.

Для обеспечения обнаружения или исправления ошибок требуется расширить кодовое пространство, увеличив длину кодовой последовательности, которой кодируются буквы, т.е. ввести дополнительную избыточность.

После этого можно говорить о так называемых «разрешенных» и «запрещенных» кодовых комбинациях. Так, если увеличить длину кодовой последовательности с 5 бит до 8 бит, то вместо 32 значений получается $2^8 = 256$ значений. В результате такой процедуры будем иметь те же 32 разрешенные комбинации кода (по числу букв) из общих 256. Среди них - 224 запрещенные (ими буквы не передаются). Разрешенные кодовые комбинации (кодовые слова) должны быть все разные и обладать определенным количеством единиц и нулей, чтобы появилась возможность обнаруживать или исправлять возникающие в них ошибки при передаче ими информации по каналу связи. Чем больше происходит увеличение длины кодовой последовательности, тем выше исправляющие свойства разрешенных кодовых комбинаций. Набор разрешенных кодовых комбинаций будет определять так называемый корректирующий код, способный обнаруживать или исправлять t ошибок. Этот код будет обладать так называемым кодовым расстоянием d между разрешенными кодовыми комбинациями. В рассмотрении этого дадим определение расстоянию по Хеммингу между двумя кодовыми комбинациями. Таким расстоянием называется суммарное количество единиц, получаемое при

поразрядном сложении двух комбинаций по модулю 2. Так, например, две комбинации

$$\begin{array}{r} 10100110 \\ \oplus\underline{01010110} \\ 11110000 \end{array}$$

имеют расстояние по Хеммингу, равное 4.

При сравнении между собой разрешенных кодовых комбинаций хеммингово расстояние между ними будет получаться различным. Минимальное хеммингово расстояние, получаемое при таком сравнении, называется кодовым расстоянием корректирующего кода.

Рассмотрим теперь, как связано кодовое расстояние с возможностями кода обнаруживать или исправлять ошибки в разрешенных кодовых комбинациях.

В начале определимся с процедурой обнаружения ошибок в разрешенных кодовых комбинациях, прошедших канал связи, которые под воздействием помех переходят в запрещённые кодовые комбинации.

Для обнаружения t ошибок достаточно, чтобы было замечено различие хотя бы в одном символе при сравнении (сложение по модулю два) полученной запрещённой комбинации со всеми разрешёнными, хранящимися на приемной стороне. Отсутствие различия с одной из них (хеммингово расстояние равно нулю) будет свидетельствовать, что передавалась именно эта разрешенная кодовая комбинация, и ошибки в канале связи не исказили ее.

Таким образом, корректирующий код (набор разрешенных кодовых комбинаций), обнаруживающий t ошибок, должен иметь кодовое расстояние, определяемое формулой:

$$d \geq t+1. \qquad (1.4)$$

Для исправления ошибок кодовое расстояние d должно быть другим. Его значение может быть определено, если процедуру обнаружения ошибок, введенную выше, дополнить

их исправлением за счёт применения попыток инвертирования (в различных вариантах) t символов в принятой разрешенной комбинации, пораженной помехой. При удачной попытке t ошибок будут исправлены. В самом неудачном варианте получаем увеличение ошибок на максимальную величину t, так что всего искаженных символов станет 2t. Теперь необходимо выполнять условие, чтобы у разрешенных комбинаций число единиц было больше или равно величине 2t+1, которое обеспечит, что запрещенная комбинация с числом ошибок 2t будет отличаться от всех разрешенных комбинаций при проведении процедуры сравнения с ними.

Таким образом, код, исправляющий t ошибок, должен иметь кодовое расстояние, определяемое формулой

$$d \geq 2t+1. \qquad (1.5)$$

Далее, зная требуемое число исправляемых ошибок t, можно к информационным символам k (буквы) добавить по определенным правилам некоторое число дополнительных проверочных символов r, чтобы образовались необходимые разрешенные кодовые комбинации. Такая процедура называется построением корректирующего блокового кода (n, k) с заданным кодовым расстоянием d. Она будет рассмотрена при описании теоретической части второй лабораторной работы.

1.5. Помехоустойчивость блоковых кодов при независимых ошибках

Под помехоустойчивостью обычно понимают способность системы передачи информации противостоять вредному влиянию помех на передачу сообщения. Так как действие помехи проявляется в том, что принятое сообщение отличается от переданного, то количественно помехоустойчивость при заданных ошибках можно характеризовать степенью соответствия принятого и

переданного сообщений. Обычно эту величину называют общим термином – достоверность. При передаче дискретной последовательности символов влияние помехи на нее проявляется в том, что вместо фактически переданного символа может быть принят какой-либо другой. Такое событие называется ошибкой. В случае передачи данных, представляемых двоичной последовательностью символов (0 или 1),называемых битами, ошибка в бите (символе) данных означает переход 1 в 0 или 0 в 1.

Для уменьшения числа ошибок при передаче сообщений используют блоковые (n, k) корректирующие коды. Они повышают достоверность принимаемых данных за счет исправления возникающих в блоке длиной n (кодовом слове) ошибок, используя процедуру декодирования.

Для определения количественной меры помехоустойчивости, прежде всего, необходимо использовать основные соотношения, имеющие место для канала с независимыми ошибками, воздействующими с вероятностью p на символы блокового (n, k) кода.

Для этого случая вероятность i ошибок в блоке длиной n находится через биноминальный закон

$$P_i = C_n^i \, p^i (1-p)^{n-i},$$

где

$$C_n^i = \frac{n!}{i!\,(n-i)!}.$$

Если блоковый код исправляет t ошибок, то вероятность того, что блок длиной n будет декодирован неправильно при наличии в нем числа ошибок больше чем t, называется вероятностью ошибочного декодирования кодового слова корректирующего кода [1,19] , обозначаемой как $P_{од}$, и рассчитываемой по формуле

$$P_{од} \approx \sum_{i=t+1}^{n} C_n^i p^i (1-p)^{n-i}. \qquad (1.6)$$

С такой вероятностью будет декодировано с ошибкой кодовое слово (n,k) корректирующего кода длиной n, содержащее k информационных и r проверочных символов.

Используя далее выражение (1.6), следует рассчитать значение вероятности ошибки $P_{ош}$ в декодированном двоичном символе (бите) на выходе декодера канала, которое рекомендуется применять в качестве количественной меры помехоустойчивости [19,1,2]. При выполнении таких расчетов необходимо знать структуру корректирующего кода, в частности набор расстояний от передаваемого кодового слова до всех разрешенных кодовых слов. К сожалению, такие данные для блоковых кодов обычно не приводятся и для расчетов $P_{ош}$ ограничиваются оценками ее величины, полагая, что если в кодовом слове происходит t +1 или более ошибок, то из-за ошибочного его декодирования в блоке корректирующего кода длиной n дополнительно возникает еще t ошибок, случайно поражающих в равной степени как информационные, так и проверочные символы. Так, если в процессе декодирования в кодовом слове длиной n находилась t +1 ошибка только в проверочных символах, то в них часть символов останется пораженной канальными ошибками, а также появятся дополнительные ошибки в информационных символах за счет неправильного декодирования. Оценку значения вероятности $P_{ош}$ определяют по формуле [19,1]

$$P_{ош} \approx \frac{1}{n} \sum_{i=t+1}^{n} i \cdot C_n^i p^i (1-p)^{n-i}. \qquad (1.7)$$

Для ориентировочной оценки вероятности $P_{ош}$ с учетом величины $P_{од}$ (1.6) при значениях $p \le 10^{-2}$ можно использовать формулу, предлагаемую в [24] , т.е.

$$P_{ош} \le \frac{d}{n} P_{од},$$

где d- кодовое расстояние блокового корректирующего кода, а n- его длина (число символов в кодовом слове).

1.6. Помехоустойчивость блоковых кодов при группирующихся ошибках

В случае группирующихся ошибок число пораженных символов кода резко возрастает по сравнению с действием независимых помех. Блоковый код, предназначенный для исправления t независимых ошибок, в этом случае не обеспечивает необходимой помехоустойчивости.

Все расчеты при этом должны вестись для конкретно выбранной модели канала.

Для случая Марковской модели, приведенной в п.1.2, средняя длина серии элементов, соответствующих плохому состоянию канала (1), определяется по формуле [8]

$$\bar{\ell}_{(1)} = \sum_{i=1}^{\infty} iP_1(i) = P_{10} + 2P_{11}P_{10} + 3P_{11}^3P_{10} + \ldots = \frac{1}{P_{10}},$$

где $P_1(i) = P_{11}^{i-1}P_{10}$ - вероятность того, что возникшее плохое состояние канала будет распространяться на i переданных элементов.

Для выбора кодов необходимо знать вероятность появления t ошибок в кодовой комбинации длиной n элементов, которая может быть найдена по формуле

$$P_n(t) = \sum_i P^{(1)}(i, n)P_n(t/i),$$

где $P^{(1)}(i, n)$ - вероятность того, что число элементов в кодовом слове, переданных за время плохого состояния канала (1), равно i, а

$$P_n(t/i) = C_n^t P_1^t (1 - P_1)^{i-t} -$$

вероятность появления t ошибок в кодовой последовательности при условии, что число элементов в ней, переданных за время плохого состояния канала, также равно i.

Полагая, что в кодовой комбинации длиной n элементов возможно появление только одного пакета ошибок, получим

$$P^{(1)}(1, n) = P_1 P_{10} P_{00}^{n-3} [2P_{00} + P_{10}(n-1)];$$

$$P^{(1)}(i, n) = P_1 P_{10} P_{11}^{i-1} P_{00}^{n-i-2} [2P_{00} + (n-i-1)P_{00}]; 2 \le i \le n-2;$$

$$P^{(1)}(n-1, n) = 2P_1 P_{10} P_{11}^{n-2};$$

$$P^{(1)}(n, n) = P_1 P_{11}^{n-1}.$$

Кроме марковской модели ошибок используют также модель Пуртова [11], для описания которой достаточно двух параметров: p-вероятность ошибок на символ в канале и г - коэффициент группирования ошибок. Тогда вероятность того, что в комбинации длиной n будет одна или более ошибок, определяется формулой

$$P_n(t \ge 1) = p \cdot n^{1-\Gamma}, \quad 0 < \Gamma < 1.$$

Для канала без памяти г =0; при г =1 все ошибки сосредоточены в виде одного пакета. Например, характерные значения для некоторых проводных выделенных каналов p = $=2.82 \cdot 10^{-4}$ и г = 0.77, а для проводных коммутируемых каналов $p=2 \cdot 10^{-3}$ и г = 0.34. Наиболее полные сведения о реальных каналах приведены в [1,2,8] и других специальных изданиях.

Вероятность появления в комбинации ошибок кратности t и более может быть приближенно определена по формуле

$$P_n(t) \approx (n/t)^{1-\Gamma} p, \qquad t < 0.3n.$$

Модель Пуртова можно реализовать через марковскую модель, если при заданном $p=P_1$ принять $P_{11}=$ г, а затем найти P_{00}, используя формулу (1.3). По P_{00} и P_{11} можно осуществить моделирование группирующихся ошибок, используя рис. 1.6. Например, возьмем значение $p = P_1 = 2 \cdot 10^{-3}$ и г = 0.34, соответствующее проводному каналу связи. Получаем $P_{11}=$ г =0.34 и вычисляем по (1.3) значение P_{00} , т. е.

$$P_{00} = \frac{1 - P_1(2 - P_{11})}{1 - P_1} = \frac{1 - 2 \cdot 10^{-3}(2 - 0.34)}{1 - 2 \cdot 10^{-3}} = 0.9987.$$

Используя полученные значения для P_{00}=0.9987 и P_{11}=0.34, можно промоделировать группирующиеся ошибки по рис. 1.6.

Корректирующий код для группирующихся ошибок выбирают, зная вероятность появления в нём определённого их числа. Чаще всего используют код Рида–Соломона или код Файра [12]. Однако, эффективность применения таких кодов резко падает при появлении длинных пакетов ошибок. В связи с этим на практике более предпочтительным вариантом их исправления является применение корректирующего кода для независимых искажений совместно с декорреляцией группирующихся ошибок. После проведения декорреляции ошибки становятся практически независимыми. Для этого используются различные устройства, обеспечивающие так называемое перемежение кодовых символов [1], одно из них-блоковое.

Блоковое устройство перемежения работает следующим образом. Кодовые последовательности предварительно записываются в передающий буфер (блок) памяти, построенный в виде матрицы, имеющей B столбцов и N строк. Запись осуществляется по столбцам B, длина которых равна длине n корректирующего кода и определяет значение N. Перемежение (перестановка) состоит в том, что для передачи по каналу связи символы кодовых последовательностей

считываются из, блока памяти по строкам. На приемной стороне имеется устройство восстановления после перемежения в виде буфера (блока) аналогичного передающему, куда символы кодовых последовательностей, поступающие из канала, записываются по строкам N, а считываются в декодер по столбцам B. В результате таких процедур группирующиеся ошибки накрывают не длину корректирующего кода n, а символы, расположенные в строках матрицы памяти. Таким образом ошибки распределяются на приемной стороне по B столбцам в ограниченном количестве, которое корректирующий код способен исправить.

Наиболее важные свойства при таком способе перемежения состоят в следующем:

1. Любой пакет ошибок $c \leq B$ переходит на выходе устройства восстановления в одиночные ошибки, каждые две из которых разделены не менее N символами.
2. Любой пакет ошибок длиной $c = a \cdot B$ $(a>1)$ переходит в пакет ошибок длиной, не больше a символов, каждые два из которых разделены не менее чем N-c символами.
3. Устройство требует наличие памяти ёмкостью $N \cdot B$ символов, и задержка в их выдаче составляет $2 \cdot N \cdot B$ символов.

Для блоковых кодов значение N выбирают, как правило, равным длине кода n. Значение B берется в зависимости от длины пакета ошибок и обычно равно ему. В противном случае потребуется корректирующий код, исправляющий большее число ошибок. Так, если $B = c/2$, то необходим код, исправляющий две независимые ошибки на длине n.

Устройство перемежения необходимо включать после кодера канала, а устройство восстановления перед декодером канала на приёмной стороне (рис. 1.1).

1.7. Определение объёма выборки данных при исследовании помехоустойчивости систем передачи информации

При моделировании систем передачи информации приходится оценивать неизвестную вероятность ошибки на символ p по её частоте P^* после переданных M бит данных. Причём P^* определяется как отношение числа символов с ошибками ко всем переданным символам.

В связи с этим возникает вопрос о точности и надёжности такой оценки, т.е. о построении доверительного интервала для вероятности p при числе M переданных бит данных или, другими словами, каким должен быть объём выборки M при моделировании процесса с планируемым значением вероятности p, чтобы полученное значение P^* незначительно отличалось от p.

Величина $\delta = |P^*-p|/p$ называется относительным доверительным интервалом.

Границы доверительного интервала обозначаются: нижняя через P^*_1 и верхняя через P^*_2 и рассчитываются по формулам [13]

$$P^*_1 = P^* - t_\beta \sqrt{\frac{P^*(1-P^*)}{M}}, \tag{1.8}$$

$$P^*_2 = P^* + t_\beta \sqrt{\frac{P^*(1-P^*)}{M}}, \tag{1.9}$$

так что величина P^* попадает в этот интервал с доверительной вероятностью β, а t_β- коэффициент, определяемый по формуле:

$$t_\beta = \arg\Phi^*\left(\frac{1+\beta}{2}\right),$$

где arg Φ^* - функция обратная нормальной функции распределения Φ^*.

Определяют t_β, как правило, из заранее рассчитанных таблиц [13]. Например, для наиболее часто встречаемых значений $\beta = 0.9$, $\beta = 0.95$ и $\beta = 0.99$ значения t_β получаются соответственно 1.643, 1.960 и 2.576.

Формулами (1.8) и (1.9) можно пользоваться, когда величины $M \cdot p$ и $M \cdot (1-p)$ порядка 10 и более [13].

Используя формулу (1.8) для нижней доверительной границы, определяют выражение для относительного доверительного интервала, приняв $P^*_1 = p$, т.е.,

$$\delta = \frac{| P^* - p |}{p} = \frac{t_\beta \sqrt{\frac{P^*(1-P^*)}{M}}}{P^* - t_\beta \sqrt{\frac{P^*(1-P^*)}{M}}}. \quad (1.10)$$

После преобразований выражения (1.10) получается формула для требуемого объема выборки M

$$M = \frac{t_\beta^2 (1+\delta)^2}{\delta^2} \cdot \frac{1-P^*}{P^*}, \quad (1.11)$$

при которой ожидаемая вероятность ошибки P^* отличается от расчетного (планируемого) значения p так, что она входит в относительный доверительный интервал δ с доверительной вероятностью β.

Например, если моделируется процесс с планируемым значением вероятности $p = 10^{-3}$ при $\delta = 0.1$ и $\beta = 0.95$, то величина $t_\beta = 1.96$, и тогда, заменяя в (1.11) P^* на p, получим

$$M = \frac{t_\beta^2 (1+\delta)^2}{\delta^2} \cdot \frac{1-P^*}{P^*} = \frac{1.96^2 (1+0.1)^2}{0.1^2} \cdot \frac{1-10^{-3}}{10^{-3}} = 464400 \ .$$

Если изменить величину β до 0.9, то при соответствующем этому значению t_β=1.643 получим

$$M = \frac{t_\beta^2 (1+\delta)^2}{\delta^2} \cdot \frac{1-P^*}{P^*} = \frac{1.643^2 (1+0.1)^2}{0.1^2} \cdot \frac{1-10^{-3}}{10^{-3}} = 326300 ,$$

т. е. для данного варианта требуется меньший объем выборки, чем в первом случае.

Проверяя допустимость применения для полученных результатов формулы (1.11), исходя из требований, что M·p>=10 и M· (1-p)>=10, получаем

$M \cdot p = 464400 \cdot 10^{-3} = 464,$

$M \cdot (1-p) = 464400 \cdot (1-10^{-3}) = 463935,$

т. е. оба условия выполняются.

При моделировании систем передачи информации с корректирующим кодированием для определения объема выборки M, представленной последовательностью блоков n корректирующего кода, в формулу (1.11) необходимо подставлять вместо P^* планируемое значение в эксперименте вероятности p.

Экспериментальное значение $P_{ош}$ определяется как отношение числа ошибок в символах выборки M, после декодирования блоков корректирующего кода, к общему числу символов, содержащихся в ее объеме. Если при таком моделировании системы с корректирующим кодированием при рассчитанном значении объёма выборки M вероятность $P_{ош}$ получается равной нулю, то это свидетельствует о том, что величину M необходимо увеличить. Значение $P_{ош}$ до увеличения M следует вычислить как отношение одной ошибки ко всему числу символов в выборке.

Предположим, в качестве примера, что после моделирования системы передачи данных с объёмом выборки M= =150000 символов, все возникшие ошибки будут исправлены и вероятность $P_{ош}$, окажется равной нулю. Это свидетельствует о том, что до увеличения объёма выборки M значение $P_{ош}$ необходимо рассчитать по формуле

$$P_{ош} \le \frac{1}{150000} = 6.6 \cdot 10^{-6}.$$

Для более точной оценки $P_{ош}$ следует увеличить объем выборки M. Установив, например, число символов в выборке M, на порядок большим, т. е. 1500000, получим, что число ошибок в символах станет равным 3 вместо нуля, тогда уточненное значение вероятности ошибки $P_{ош}$ определится как

$$P_{ош} = \frac{3}{1500000} = 2 \cdot 10^{-6}.$$

1.8. Лабораторное задание

1. Осуществить генерацию независимых ошибок различной интенсивности с выводом их на экран монитора. Для этого выполнить следующие пункты:

1.1. Включить ЭВМ и войти в диск Z. Выбрать в диске Z файл LABRNEW и открыть его. Затем открыть файл LAB1Win и далее файл Lab1 [14] (рис.1.8).

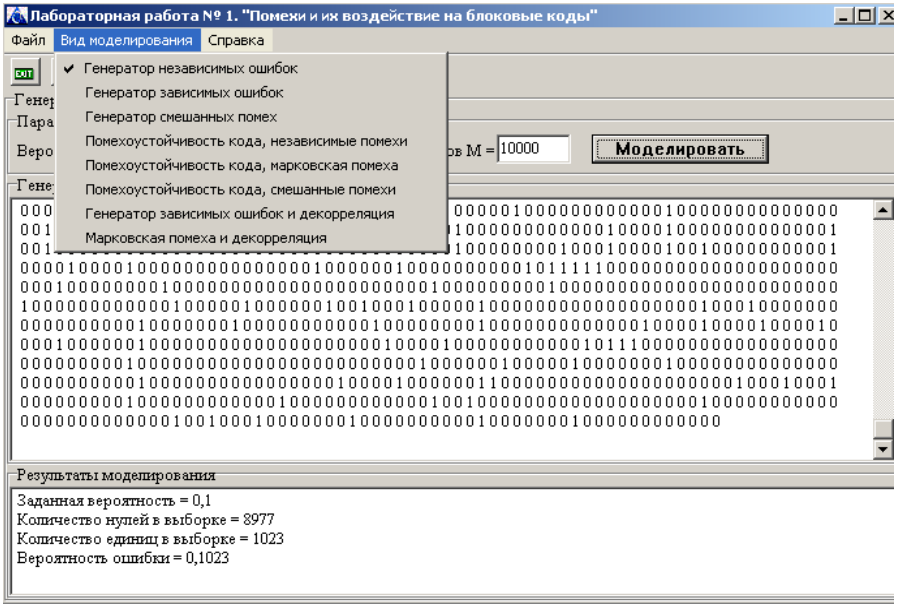


Рис. 1.8. Окно заставки для проведения лабораторных исследований с режимами работы и результатами моделирования

1.2. Для моделирования независимых ошибок рассчитать необходимые объемы выборок M (см. стр. 26-29) для обеспечения того, чтобы величины вероятностей ошибок p (0,4; 0,3; 0,2; 0,1; 0,05) попадали в доверительный интервал $\delta = 0{,}01$ с доверительной вероятностью $\beta = 0{,}99$ (Контрольные значения: 101500; 157900; 270800; 609200; 1286000).

1.3. Выбрать режим «Генератор независимых ошибок» (рис. 1.8). Ввести в окно заставки лабораторной работы вероятность ошибки p = 0,4 и величину M, рассчитанную по условиям п. 1.2 (контрольные значения для M в скобках). Осуществить моделирование. Наблюдать получившееся распределение. Зафиксировать результаты моделирования.

Последовательно вводить вероятности ошибок 0,3; 0,2; 0,1; 0,05 с соответствующими величинами М из п. 1.2.

Осуществить моделирование. Зафиксировать полученные результаты. Отметить особенности получающихся распределений нулей и единиц.

1.4. Выполнить требования пункта п. 1.3, но с одинаковой величиной выборки $M = 1000$. Зафиксировать полученные результаты моделирования.

1.5. Осуществить сравнительный анализ результатов моделирования по пунктам 1.3 и 1.4 в виде таблицы. Сделать выводы.

2. Осуществить генерацию группирующихся ошибок различной интенсивности. Для этого выполнить следующие пункты:

2.1. Выбрать вид моделирования «Генератор зависимых ошибок». Ввести в окно заставки лабораторной работы значения P_1=0.05 при P_{11}: 0.2; 0.3; 0.4; 0.5; 0.6; 0.7 и M=650, осуществляя каждый раз моделирование. Зафиксировать полученные длины пакетов ошибок (число единиц в пакете).

2.2. Осуществить сравнительный анализ распределений независимых и группирующихся ошибок. Указать, в чем состоит их существенное отличие.

3. Осуществить генерацию «смешанных» ошибок различной интенсивности. Для этого выполнить следующие пункты:

3.1. Выбрать вид моделирования «Генератор смешанных помех», отражающий появление как независимых, так и группирующихся ошибок. Ввести в окна заставки лабораторной работы значения P_1 = 0,05, P_{11}=0,81 , M = 1000, p = 0,035. Осуществить моделирование. Зафиксировать результаты.

3.2. Установить значение вероятности P_1=0. Осуществить моделирование. Зафиксировать результаты.

3.3. Провести сравнительный анализ результатов моделирования. Отметить разницу распределений из пунктов 3.1 и 3.2.

4. Осуществить моделирование влияния на корректирующий блоковый код независимых ошибок при различной исправляющей его способности. В лабораторном исследовании применяется код (256,k) с разной длиной k, формируемой программой в зависимости от числа исправляемых ошибок t. Выполнить следующие пункты:

4.1. Выбрать вид моделирования «Помехоустойчивость кода, независимые помехи», отражающий появление независимых ошибок.

4.2. Для моделирования помехоустойчивости корректирующих кодов определить число блоков последовательности длиной 1024 символа (число для контроля - 6543) для обеспечения того, чтобы величина p=0.01 попала в доверительный интервал $\delta = 0{,}01$ с доверительной вероятностью $\beta = 0{,}99$.

4.3. Ввести в окна заставки лабораторной работы число блоков определенное в пункте 4.2, длину блока 1024, вероятность независимой ошибки p = 0,01, число ошибок, корректируемое кодом равное 1, длину кода равную 256. Помеху на экран не выводить. Осуществить моделирование.

Зафиксировать полученную вероятность $P_{ош}$ и вероятность появления хотя бы одной строки длиной 1024 элемента с неисправленными ошибками $P_{строк}$.

4.4. Поступая аналогично пункту 4.3, получить значения $P_{ош}$ и $P_{строк}$ для кода 256, но исправляющего соответственно 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 ошибок.

Результаты занести в таблицу с учетом пункта 4.3.

5. Исследовать помехоустойчивость передачи информации при воздействии на корректирующий код группирующихся ошибок. Для этого выполнить следующие пункты:

5.1. Выбрать вид моделирования «Помехоустойчивость кода, марковская помеха», отражающий группирование ошибок в канале передачи данных.

Ввести вероятности $P_1 = 0{,}01$ и $P_{11} = 0{,}81$. Принять длину блока равную 1024, число блоков взять из п. 4.2, длину кода 256, а число ошибок, корректируемое кодом, равное 1. Осуществить моделирование. Зафиксировать полученные вероятности ошибок $P_{ош}$ и $P_{строк}$.

5.2. Поступая аналогично п. 5.1, получить $P_{ош}$ и $P_{строк}$ для кода 256, но исправляющего соответственно 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 ошибок. Результаты занести в таблицу с учетом п. 5.1.

6. Исследовать помехоустойчивость передачи информации блоковым корректирующим кодом при «смешанных» ошибках. Для этого выполнить следующие пункты:

6.1. Выбрать вид моделирования «Помехоустойчивость кода, смешанные помехи», отражающий смесь независимых и группирующихся ошибок.

Ввести вероятности $P_1 = 0{,}005$, $P_{11} = 0{,}81$, число блоков взять из п. 4.2, длину блока 1024, вероятность ошибки $p=0{,}005$, длину кода 256, число ошибок, корректируемое кодом 1, помеху на экран не выводить. Осуществить моделирование. Зафиксировать полученные вероятности ошибок $P_{ош}$ и $P_{строк}$.

6.2. Поступая аналогично п. 6.1, получить $P_{ош}$ и $P_{строк}$ для кода 256, но исправляющего соответственно 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 ошибок. Результаты занести в таблицу с учетом п. 6.1.

7. Используя данные таблиц из пунктов 4, 5, 6, построить два графика (три зависимости на каждом графике), отображающих изменения величин $P_{ош}$ и $P_{строк}$ от кратности исправляемых ошибок. По горизонтали откладывать число ошибок, а вниз по вертикали значения $P_{ош}$ и $P_{строк}$ в логарифмическом масштабе. Сравнить результаты. Сделать выводы.

8. Исследовать влияние декорреляции на распределение группирующихся ошибок в кодовых последовательностях. Для этого выполнить следующие пункты:

8.1. Выбрать вид моделирования «Генератор зависимых ошибок и декорреляция». Ввести $P_1 = 0{,}05$, $P_{11} = 0{,}81$, M= 512. Осуществить моделирование. Наблюдать распределение группирующихся ошибок без декорреляции. Зафиксировать длины групп ошибок (число единиц в группе).

8.2. Наблюдать влияние изменения параметров декорреляции на распределение ошибок в выборке M, т.е. числа единиц в пакетах ошибок. Для чего ввести в окна параметров декорреляции величины 2 и 256, определяющие соответственно число столбцов B и число строк N матриц памяти блокового устройства перемежения (см. стр.25) и осуществить моделирование. Зафиксировать происшедшие изменения в выборке, а также длины групп ошибок.

8.3. Повторить моделирование для параметров декорреляции 4 128, 8 64, 16 32. Отразить в отчете сопутствующие изменения распределения ошибок в выборке M от параметров декорреляции в виде длин пакетов ошибок.

9. Исследовать влияние декорреляции на помехоустойчивость корректирующего кода при группирующихся ошибках. Для этого выполнить следующие пункты:

9.1. Выбрать вид моделирования «Марковская помеха и декорреляция». Ввести $P_1 = 0{,}05$, $P_{11} = 0{,}81$, число блоков 16. Длина блока равна 1024, параметры декорреляции 1 и 1 (отсутствие декорреляции), длина кода равна 32, число ошибок, корректируемое кодом равно 3, включить вывод помехи на экран. Осуществить моделирование. Зафиксировать значения $P_{ош}$ и $P_{строк}$, которые получаются в случае параметров декорреляции 1 и 1, т.е. ее отсутствия.

9.2. В п. 9.1 ввести параметры декорреляции 256 и 32, оставив остальные параметры прежними. Осуществить моделирование. Зафиксировать значения $P_{ош}$ и $P_{строк}$ при

значениях декорреляции 256 и 32. Сравнить две величины $P_{ош}$ из п. 9.1 и п. 9.2, сделать выводы.

10. Удостовериться, что действие декорреляции при группирующихся ошибках приводит к значению $P_{ош}$, приближающемуся к случаю действия независимых ошибок. Для этого выполнить следующие пункты:

10.1. Выбрать вид моделирования «Помехоустойчивость кода, независимые помехи». Ввести длину блока двоичной последовательности - 1024, выбрать число таких блоков-16, ввести длину корректирующего кода-32, число ошибок, исправляемое кодом-3, вероятность независимой ошибки p=0.05. Включить вывод помехи на экран. Осуществить моделирование. Зафиксировать в отчет полученное значение $P_{ош}$.

10.2. Сравнить полученное значение $P_{ош}$ из п. 10.1 и п. 9.2. Сделать выводы.

Содержание отчета

1. Количественные показатели по выполнению конкретных пунктов с их комментариями и выводами.
2. Графики и таблицы по проведенным измерениям.
3. Краткие выводы по лабораторной работе.

Контрольные вопросы

1. Дайте определение независимым, группирующимся и смешанным ошибкам.
2. Для каких целей используется корректирующее кодирование в системах передачи информации? Нарисуйте структурную схему системы передачи информации.
3. Как определяются основные вероятностные показатели, характеризующие корректирующий код?
4. Что такое кодовое расстояние кода и как оно определяется?

5. Какая существует зависимость между кодовым расстоянием и корректирующей способностью кода?
6. Дайте определение модели источника ошибок, предложенной Гильбертом.
7. Запишите матрицу переходов из одного состояния в другое для модели Гильберта. Охарактеризуйте входящие в нее величины.
8. Как с помощью марковской цепи можно описать группирование ошибок?
9. Как можно создать эквивалентный канал с группирующимися ошибками по известным параметрам канала с независимыми ошибками?
10. Что такое декорреляция группирующихся ошибок и как она осуществляется?
11. Назовите, в каких случаях применяется декорреляция и какие ее основные параметры.
12. Как правильно выбрать параметры декорреляции?

2. ЛАБОРАТОРНАЯ РАБОТА № 2

БЛОКОВЫЕ КОРРЕКТИРУЮЩИЕ КОДЫ

Цель работы: изучить построение, а также кодирование и декодирование информации блоковыми корректирующими кодами.

Теоретическая часть

2.1. Структура блоковых кодов и их формирование

Наиболее часто в практике помехоустойчивого кодирования используются блоковые коды. Здесь будут рассматриваться разделимые блоковые коды, называемые систематическими (линейными) блоковыми кодами и обозначаемые как (n, k), где n – длина кода, k – число информационных символов, а величина n-k = r определяет число проверочных символов, обеспечивающих необходимую избыточность кода [4]. Основным отличием систематических кодов является то, что их проверочные символы представляют собой различные линейные комбинации информационных символов.

Блоковый код формируется из множества $M = 2^n$ кодовых комбинаций, куда включается и нулевая комбинация. Всё это множество кодовых комбинаций разделяют на разрешённые и запрещённые. Передача информации осуществляется разрешёнными кодовыми комбинациями (кодовыми словами) с кодовым расстоянием d, число которых определяется величиной 2^k. Ошибки, возникающие в канале связи, переводят их в запрещённые. На приёмной стороне эти искажения могут быть обнаружены, исправлены или одновременно некоторые исправлены, а другие обнаружены после осуществления определённых операций с разрешенными кодовыми комбинациями.

Вместо термина кодовое слово иногда используют понятие “кодовый вектор”, который может быть нулевым или содержать определенное число единиц, называемое весом и определяющее его расстояние от нулевого кодового слова.

Корректирующий код будет задан, если определены и зафиксированы все его разрешённые кодовые комбинации (кодовые слова), например, записаны в постоянное запоминающее устройство. Однако такой способ непригоден при большом числе кодовых слов, и поэтому наиболее часто используется матричная запись (n,k) кода. При этом матрица, именуемая как G_{nk} (рис. 2.1), должна быть такой, чтобы по ней можно было получить все разрешенные кодовые комбинации. Применяемые при этом преобразования должны быть несложными и базироваться, как правило, на использовании простых логических функций, например, сложение по модулю 2.

Для возможности формирования информационной части кодового слова применяется часть G_{nk} - единичная матрица G_1 длиной k. Складывая строки G_1 в определенном порядке, можно создать информационные составляющие всех разрешенных кодовых комбинаций, кроме нулевой.

Для образования вторых частей разрешенных кодовых комбинаций, называемых проверочными, используется вторая часть матрицы G_{nk} размером $r \times k$, имеющая нерегулярную структуру (в отличие от G_1). Таким образом, матрица G_{nk} , состоящая из двух частей, называется производящей (образующей) и по ней можно построить все разрешенные кодовые комбинации (n,k) кода (рис.2.1). Эта матрица используется для задания систематических блоковых корректирующих кодов.

Ввиду того, что структура второй части G_{nk} заранее неизвестна, однозначного правила для построения производящей матрицы блокового (n, k) кода нет, но есть общие положения, которыми можно руководствоваться при ее получении.

Прежде всего, каждая строка G_{nk}, исходя из требований к разрешенным кодовым комбинациям, должна иметь количество единиц $d \geq 2t+1$. Такое же количество единиц должно получаться и при образовании разрешенных кодовых слов, когда осуществляется сложение в любых сочетаниях двух строк, трех строк и т. д. до 2t строк включительно. При таком сложении в правой части кодового слова должно обеспечиваться получение недостающего числа единиц, дополняющего результат до значения $d \geq 2t+1$. В других вариантах формирования разрешенных кодовых комбинаций таких требований не предъявляется, так как необходимое значение $d \geq 2t+1$ будет уже обеспечиваться сложением достаточного количества единиц из структуры единичной матрицы G_1.

n столбцов

	k элементов							r элементов						
	1	0	0	0	0	⋯	0	1	1	0	1	0	⋯	1
	0	1	0	0	0	⋯	0	1	0	1	0	1	⋯	0
	0	0	1	0	0	⋯	0	0	1	0	1	0	⋯	1
k строк	0	0	0	1	0	⋯	0	0	1	1	1	0	⋯	0
	0	0	0	0	1	⋯	0	0	0	1	1	1	⋯	1
	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	0	0	0	⋯	1	0	0	0	1	1	⋯	1

Рис. 2.1. Образующая матрица блокового кода $G_{n,k}$

Таким образом, матрица $G_{n,k}$ состоит из k линейно независимых кодовых слов. Остальные 2^k-k-1 разрешенных кодовых слова получаются как комбинации ее строк в определенных сочетаниях. Нулевое кодовое слово также

является разрешенным. Таким образом, условия формирования правой части производящей матрицы, исходя из выше изложенных требований, можно получить на основе следующего. Вследствие того, что каждая ее строка является разрешенной кодовой комбинацией, число единиц в любой строке правой части $G_{n,k}$, в соответствии с определением кодового расстояния, должно составлять не менее чем d-1, т. к. одна единица всегда находится в левой части по определению единичной матрицы. Ввиду этого, сумма двух любых строк правой части $G_{n,k}$ должна содержать число единиц большее или равное $d - 2$, а трех любых строк большее или равное $d - 3$ и т. д. до суммы в любом сочетании 2t строк, когда число единиц должно быть больше или равно величине $d – 2t$.

Эти условия являются определяющими для различных алгоритмов, по которым можно построить правую часть производящей матрицы $G_{n,k}$.

Наиболее просто это осуществляется для блоковых кодов Хемминга с $d = 3$, исправляющих одну ошибку ($t = 1$) [7].

Например, построим правую часть матрицы $G_{7,4}$ для кода Хемминга (7,4) с $r = 3$. Для этого возьмем из общего количества комбинаций $2^r = 2^3$ (001, 010, 011, 100, 101, 110, 111,000) те, которые содержат больше одной единицы, т. е. 011, 101, 110, 111. Количество единиц здесь больше или равно 2, и они могут быть использованы в качестве строк правой части $G_{7,4}$. Проверка показывает, что сумма двух комбинаций в различных вариантах содержит число единиц большее или равное $d – 2t$, что соответствует требованиям для формирования $G_{7,4}$.

Таким образом, полная производящая матрица кода Хемминга (7,4) может быть записана в следующем виде:

$$G_{7,4} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \qquad (2.1)$$

Аналогично можно сформировать производящие матрицы и для кодов Хемминга с увеличенным значением r. Построение $G_{n,k}$ для других кодов с кодовым расстоянием d большим трех может быть осуществлено при выборе определенных значений r и d. Общий подход заключается в следующем. Зная r, необходимо вычислить и зафиксировать все $2^r - 1$ комбинации. Далее среди них выбрать те, которые имеют количество единиц, соответствующее заданному d и образовать, например, массив M(1). Теперь, используя вышеизложенные условия существования правой части производящей матрицы с заданным d, осуществить их проверку для всех возможных вариантов различных наборов k комбинаций из массива M(1). Тот набор, который удовлетворяет этим условиям, может быть рекомендован в качестве правой части производящей матрицы блокового кода. Этот вариант может быть использован только при малом значении количества проверочных символов r. В других случаях такой подход требует значительного времени для вычислений $G_{n,k}$ и является непрактичным.

Сократить продолжительность обработки можно, если использовать другие алгоритмы, позволяющие уменьшить вычислительные процедуры. Структурная схема одного из таких алгоритмов показана на рис. 2.2.

Процедура формирования правой части производящей матрицы осуществляется в два этапа. Вначале получается ее промежуточное значение, которое является неполным. Затем для увеличения количества строк матрицы проводится повторное проведение операций, когда к полученному первоначальному массиву чисел образующей матрицы добавляются по очереди числа из M(1). После проведения всех переборов образуется полное число строк правой части $G_{n,k}$. Аналогичных матриц с одинаковыми свойствами достаточно много. Их количество определяется массивом M(1). В

принципе, можно получить столько матриц, сколько в нем чисел, в то же время количество строк $G_{n,k}$ зависит от значения числа из массива M(1), с которого начинается перебор. Изменяя его и отслеживая количество полученных строк, добиваются максимального их значения. Например, в [17] была получена одна из производящих матриц для кода с r = 8 и d = 5. При таком кодовом расстоянии возможно исправление до двух независимых ошибок, что следует из выражения $d \geq 2t+1$, приведенного в разд. 1.

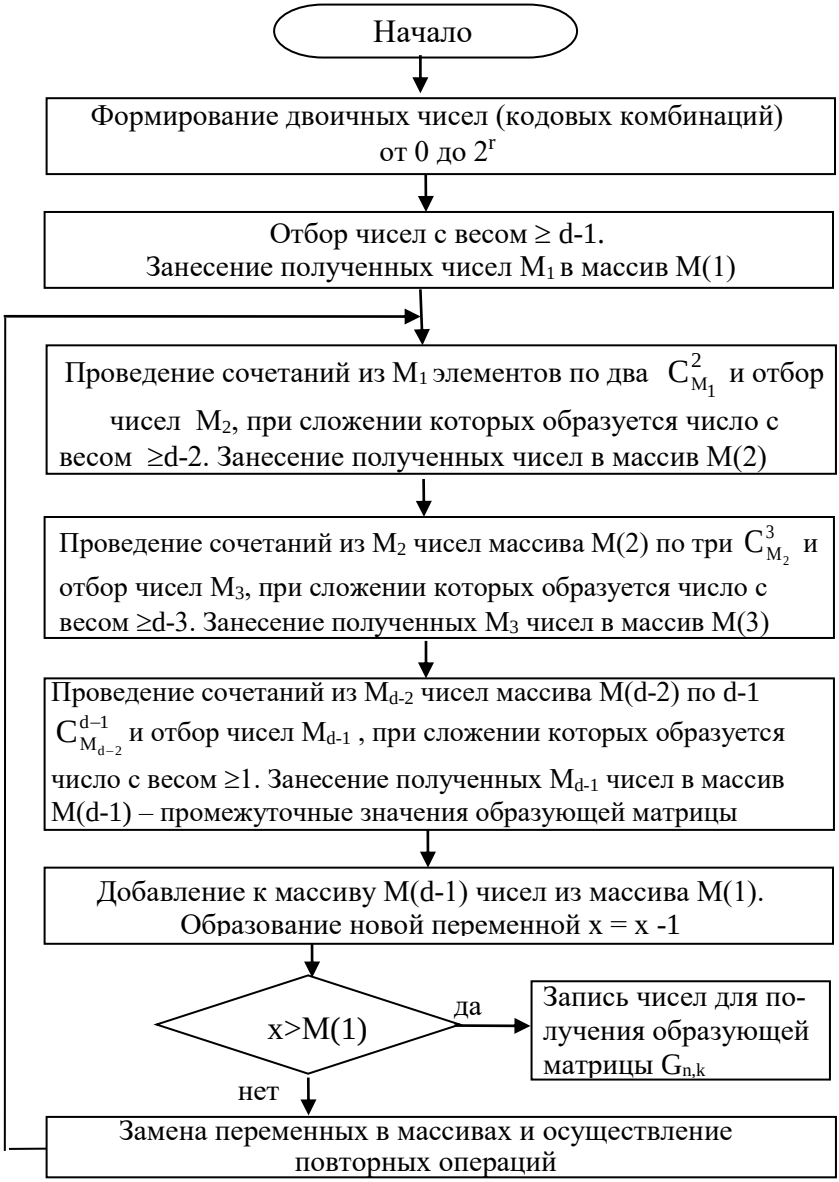


Рис.2.2. Структурная схема алгоритма получения правой части образующей матрицы блокового кода

Всего строк $G_{n,k}$ получается 9, а промежуточное значение равно 6. Полная образующая матрица блокового кода (16,8) после вычеркивания одной из строк представлена на рис. 2.3.

$$G_{16,8} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Рис. 2.3. Образующая матрица блокового кода (16,8) с d=5

Производящие матрицы могут быть сформированы и для других значений d и r. Например, для d = 5 и r = 6 получаем следующую производящую матрицу:

$$G_{8,2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}, \qquad (2.2)$$

по которой может быть получен блоковый код (8,2) с r = 6, исправляющий до двух независимых ошибок.

2.2. Качество блокового кода

Сформировав образующую матрицу корректирующего кода, следует оценить его эффективность.

Общепринято, что основными характеристиками корректирующего кода являются относительная скорость кода R= k/n и его кодовое расстояние d [7]. Кроме этого одной из характеристик кода является его избыточность R_n, указывающая степень удлинения кодовой комбинации для достижения определенной корректирующей способности. Избыточность кода может быть выражена как $R_n = (n - k)/n = r/n$, а связь с относительной скоростью кода определится соотношением $R=1-R_n= =1- r/n =k/n$. Например, для кода Хемминга (7,4) с матрицей (2.1) $R_n = 0.43$, для кода (16,8) (рис. 2.3) $R_n =0,5$, а для кода (8,2) (2.2) $R_n =0,75$.

Естественно, желательно иметь код с меньшей избыточностью при большей исправляющей его способности. Здесь возникает вопрос о числе проверочных символов для такого кода. Однозначно этот вопрос не решается, а существуют лишь границы для числа проверочных символов r при заданном кодовом расстоянии d, которые необходимо использовать для выяснения того, насколько построенный код близок к оптимальному коду. Для этой цели наиболее часто используют верхнюю границу Хемминга и нижнюю границу Варшамова-Гилберта [9].

Границу Хемминга можно получить из следующих соображений. Пусть задана кодовая комбинация с блоковой длиной n, и в ней возникают независимые ошибки кратностью до t. Общее число ошибок в комбинации складывается из однократных, двукратных, трехкратных и т.д. до t-кратных ошибок и может быть определено из равенства

$$N_{ош} = C^{1}_{n} + C^{2}_{n} + C^{3}_{n} + ...+ C^{t}_{n}.$$

Это число ошибок должно найти соответствие в количестве проверочных символов r и определяться через них из соотношения

$$2^{r} - 1 \geq C^{1}_{n} + C^{2}_{n} + C^{3}_{n} + ...+ C^{t}_{n}$$

или

$$2^r \geq 1 + C^1_n + C^2_n + C^3_n + ... + C^t_n .$$

Из последнего выражения получается граница Хемминга для блокового кода

$$r \geq \log_2 \sum_{i=0}^{t} C_n^i , \qquad (2.3)$$

которую можно интерпретировать как границу существования кода при минимальной избыточности, необходимой при заданных k и t.

Граница Варшамова-Гилберта, в отличие от границы Хемминга, является достаточным условием существования кода с кодовым расстоянием не меньшим d и записывается в виде

$$r \leq \log_2 \sum_{i=0}^{d-2} C_n^i \quad . \qquad (2.4)$$

Например, оценим полученное значение r для кода, сформированного по матрице представленной на рис. 2.3.

Применяя границу Хемминга (2.3), получим

$$r \geq \log_2[1 + C^1_{16} + C^2_{16}].$$

Отсюда после вычисления логарифма получим, что $r \geq 7$, т.е. наименьшее значение числа проверочных символов может быть равно 7.

Используя границу Варшамова-Гилберта (2.4) и учитывая, что d= 5, получаем выражение

$$r \leq \log_2[1 + C^1_{16} + C^2_{16} + C^3_{16}]. \qquad (2.5)$$

Логарифмируя (2.5), находим, что $r \leq 11$, т.е. граница Варшамова - Гилберта гарантирует при таком условии

существование кода длиной n = 16, исправляющего две ошибки.

Отсюда можно сделать вывод, что полученный код (16,8) является достаточно хорошим и может использоваться в классе кодов с d =5.

2.3. Кодирование блоковыми кодами

Кодирование блоковыми кодами заключается во введении в кодовую информационную последовательность y_1, y_2, y_3, ..., y_k дополнительной избыточности, необходимой для коррекции ошибок. Значение этой избыточности зависит от числа исправляемых ошибок в кодовом векторе и выражается, в конечном счете, в конкретной величине кодового расстояния между разрешенными комбинациями (кодовыми словами), используемыми при передаче сообщений. Результатом кодирования является получение кодового слова на основе вычисления проверочных символов b_1, b_2, b_3, ...,b_r, представляющих собой линейную комбинацию информационных символов, осуществленную по определенной процедуре.

Такая процедура выполняется на основе производящей матрицы $G_{n,k}$, которую с учетом вышеприведенного обозначения проверочных символов можно записать в виде

$$G_{n,k} = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & b_{11} & b_{12} & \dots & b_{1r} \\ 0 & 1 & 0 & 0 & \dots & 0 & b_{21} & b_{22} & \dots & b_{2r} \\ 0 & 0 & 1 & 0 & \dots & 0 & b_{31} & b_{32} & \dots & b_{3r} \\ 0 & 0 & 0 & 1 & \dots & 0 & b_{41} & b_{42} & \dots & b_{4r} \\ 0 & 0 & 0 & 0 & \dots & 1 & b_{k1} & b_{k2} & \dots & b_{kr} \end{bmatrix}, \quad (2.6)$$

где b_{11}, b_{12}, ..., b_{kr} принимают значения 1 или 0.

Строки матрицы $G_{n,k}$ (2.6) представляют собой часть разрешенных кодовых слов из их общего количества 2^k. Все остальные слова можно сформировать, складывая в определенном порядке строки $G_{n,k}$, с учетом структуры единичной матрицы. Получение проверочных символов для информационного слова, состоящего из одних единиц, предполагает сложение друг с другом по модулю 2 всех строк правой части $G_{n,k}$. В том случае, когда единицы находятся в первой и последней позициях информационной последовательности, придется складывать соответственно первую и последнюю строки правой части $G_{n,k}$ и т.п.

Процедура получения кодового слова $\vec{V}$, с точки зрения математики [12], реализуется произведением информационного вектора $\vec{Y} = y_1, y_2, y_3,...,y_k$ на производящую матрицу $G_{n,k}$, т.е.

$$\vec{V} = \vec{Y} \cdot G_{n,k}, \qquad (2.7)$$

где $\vec{V} = y_1\ y_2\ y_3\ ...y_k\ b_1\ b_2\ b_3\ ...\ b_r$ – кодовое слово (разрешенная кодовая комбинация).

Например, необходимо осуществить кодирование информационной последовательности 1101 с помощью кода Хемминга (7,4).

Воспользуемся для этого производящей матрицей (2.1), т.е.

$$G_{7,4} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} = \tag{2.8}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & b_{11} & b_{12} & b_{13} \\ 0 & 1 & 0 & 0 & b_{21} & b_{22} & b_{23} \\ 0 & 0 & 1 & 0 & b_{31} & b_{32} & b_{33} \\ 0 & 0 & 0 & 1 & b_{41} & b_{42} & b_{43} \end{bmatrix}.$$

В соответствии с вышеизложенной процедурой в (2.8) необходимо сложить по модулю 2 первую, вторую и четвертую строки правой части $G_{7,4}$, получая при этом проверочную часть

$$001 = 011 \oplus 101 \oplus 111.$$

Вся разрешенная кодовая комбинация $y_1\ y_2\ y_3\ y_4\ b_1\ b_2\ b_3$ формируется в виде 1101001 и может быть передана в канал связи.

Аналогичная процедура получения проверочных символов может быть осуществлена также через проверочные уравнения, составленные с учетом расположения единиц в столбцах правой части $G_{7,4}$ (2.8), т.е.

$$\begin{aligned} b_1 &= y_2 \oplus y_3 \oplus y_4 = 1 \oplus 0 \oplus 1 = 0; \\ b_2 &= y_1 \oplus y_3 \oplus y_4 = 1 \oplus 0 \oplus 1 = 0; \\ b_3 &= y_1 \oplus y_2 \oplus y_4 = 1 \oplus 1 \oplus 1 = 1. \end{aligned} \tag{2.9}$$

Следует отметить, что соотношения (2.9) удобнее составлять с помощью, так называемой проверочной матрицы H, содержащей r строк и n столбцов. Она строится на основе производящей матрицы (2.6) и записывается как

$$H_{n,k} = \begin{bmatrix} b_{11} & b_{21} \ldots b_{k1} & 1 & 0 \ldots 0 \\ b_{12} & b_{22} \ldots b_{k2} & 0 & 1 \ldots 0 \\ . & . & . & . \\ b_{1r} & b_{2r} \ldots b_{kr} & 0 & 0 \ldots 1 \end{bmatrix}. \qquad (2.10)$$

Непосредственно для кода Хемминга (7,4) запись (2.10) с учетом (2.8) принимает вид

$$H_{7,4} = \begin{bmatrix} b_{11} & b_{21} & b_{31} & b_{41} & 1 & 0 & 0 \\ b_{12} & b_{22} & b_{32} & b_{42} & 0 & 1 & 0 \\ b_{13} & b_{23} & b_{33} & b_{43} & 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} b_1 \\ b_2 \\ b_3 \end{matrix}. \qquad (2.11)$$

$$y_1 \; y_2 \; y_3 \; y_4$$

Обозначив буквами b_1, b_2, b_3 и y_1, y_2, y_3, y_4 соответственно строки и столбцы у (2.11), можно образовать ранее полученные уравнения (2.9), сложив конкретные номера информационных элементов кодовой комбинации, соответствующие единицам в строках левой части матрицы (2.11).

Для декодирования кодов используют транспонированную проверочную матрицу H^T, которую формируют из

(2.10) таким образом, чтобы ее столбцы стали строками H^T, что позволяет применять ее в процедуре декодирования, основанной на математическом аппарате для матриц, т.е.

$$H^T = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & b_{22} & \cdots & b_{2r} \\ . & . & . & . \\ b_{k1} & b_{k2} & \cdots & b_{kr} \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ . & . & . & . \\ 0 & 0 & \cdots & 1 \end{bmatrix}. \qquad (2.12)$$

Для кода Хемминга (7,4) H^T записывается следующим образом:

$$H^T_{7,4} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \qquad (2.13)$$

Для производящей матрицы $G_{n,k}$ и матрицы $H^T_{n,k}$ справедливо соотношение

$$G_{n,k} \cdot H^T_{n,k} = 0, \qquad (2.14)$$

в чем легко можно удостовериться, используя правило перемножения матриц [13]. Используя (2.14) также проверяют

принадлежность кодового слова к разрешенным кодовым комбинациям кода, когда $\vec{V} \cdot H^T = 0$.

2.4. Декодирование блоковых корректирующих кодов

Декодирование блокового корректирующего кода заключается в обнаружении ошибок в разрешенной кодовой комбинации и дальнейшем их устранении за счет специальных вычислительных процедур.

Существует несколько методов декодирования блоковых кодов: синдромный метод, метод максимального правдоподобия, мажоритарный метод [10]. Наиболее распространенными из них являются синдромный метод и метод максимального правдоподобия при малом значении k. Мажоритарный метод применим только к ограниченному числу кодов, и является менее эффективным. В связи с этим в предлагаемой лабораторной работе будет рассматриваться синдромный метод, основанный на полном переборном алгоритме декодирования, а также метод максимального правдоподобия.

2.4.1. Полный переборный алгоритм декодирования блоковых кодов

Будем считать, что после кодирования информационной последовательности $\vec{Y}$ (y_1 y_2 y_3 ... y_k) получено кодовое слово $\vec{V}$ (y_1 y_2 y_3 ... y_k b_1 b_2 $b_{3....}$ b_r) , которое передается в канал связи, где могут возникнуть ошибки из-за воздействия помех.

Первой составляющей процедуры декодирования является вычисление так называемого синдрома $\vec{S}$, который включает в себя информацию об ошибках. Если в принятом кодовом слове $\vec{V}$ отсутствуют ошибки, то величина $\vec{S}$, называемая синдромом, с учетом (2.14), равна 0, т.е.

$$\vec{S} = \vec{V} \cdot H^T = \vec{Y} \cdot G \cdot H^T = \vec{Y} \cdot 0 = 0,$$

где $\vec{Y}$ - информационная последовательность длиной k,

$\vec{V}$ - кодовое слово, полученное в результате кодирования $\vec{Y}$.

При наличии числа ошибок $\vec{E}$ в кодовом слове, не превышающем корректирующей способности кода, получаем, что

$$\vec{S} = (\vec{V} \oplus \vec{E}) \cdot H^T = \vec{V} \cdot H^T \oplus \vec{E} \cdot H^T = 0 \oplus \vec{E} \cdot H^T = \vec{E} \cdot H^T \qquad (2.15)$$

и синдром $\vec{S}$ не равен нулю.

Используя свойства синдрома [5], вытекающие из (2.15), можно определить позиции ошибок в кодовом слове. Так, если имеются ошибки на позициях с номерами а, в, с..., так что $\vec{E} = 0 \ldots 01 \ldots 1 \ldots 1 \ldots 0$, то из (2.15) получаем, что

а в с

$$\vec{S} = \sum_i \vec{E}_i \cdot H_i^T,$$

где H_i - i-я строка H^T.

Из этого выражения с учетом номеров (а, в, с) позиций ошибок в $\vec{E}$ следует, что

$$\vec{S} = H_a \oplus H_b \oplus H_c.$$

Таким образом, синдром при наличии ошибок в кодовом слове равен сумме по модулю 2 тех строк транспонированной проверочной матрицы H^T, номера которых совпадают с номерами ошибок в кодовом слове. Это является основным правилом синдромного декодирования.

Иначе, определив номера строк H^T, составляющих синдром, можно указать позиции ошибок и далее, изменив на этих местах символы кодового слова на противоположные, - исправить их. Проблема декодирования блокового группового кода при таком подходе заключается в минимизации процедуры поиска номеров строк H^T, что обеспечивает максимальную скорость обработки кодовых слов.

Приведем пример исправления одиночных ошибок кодом Хемминга (7,4) с d = 3. Пусть передается кодовое слово

$\vec{V}(1101001)$. В результате воздействия помех в канале связи произошла ошибка $\vec{E}(0100000)$, так что $\tilde{\vec{V}} = \vec{V} \oplus \vec{E} = 1001001$, т.е. ошибка имеет место во втором символе кодового слова $\vec{V}$. Осуществляем вычисление синдрома $\vec{S} = \tilde{\vec{V}} \cdot H^T$. Для этого производим умножение вектора $\tilde{\vec{V}}$ на H^T (2.13), т.е.

$$\vec{S} = (1001001)\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = 011 \oplus 111 \oplus 001 = 101.$$

Таким образом, синдром не равен нулю и в кодовом слове есть ошибки. Так как значение синдрома 101 совпадает со второй строкой H^T (2.13), то ошибка произошла во втором символе кодового слова, и она может быть исправлена.

Если в коде (7,4) произошло две ошибки, то правильное декодирование невозможно. Для устранения такого количества ошибок потребуется код с кодовым расстоянием d = 5.

Воспользуемся производящей матрицей $G_{(8,2)}$ (2.2) для кода с d = 5 и построим по (2.12) транспонированную матрицу $H^T_{(8,2)}$, т.е.

$$\mathbf{H}^{\mathrm{T}}_{(8,2)} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Пусть теперь передается полученное с помощью $G_{8,2}$ (2.2) кодовое слово $\vec{V} = 11111100$, где два первых символа информационные, а последующие шесть проверочные. В результате помех в канале связи сформировался вектор ошибок $\vec{E} = =11000000$, что приводит к $\vec{V} = \vec{V} \oplus \vec{E} =$ 00111100, т.е. ошибки находятся в первом и во втором символах кодового слова $\vec{V}$.

Осуществляем вычисление синдрома $\vec{S} = \vec{V} \cdot H^T$. Для этого производим умножение вектора $\vec{V}$ с ошибками на H^T (2.15), т.е.

$$\vec{S} = (00111100) \cdot \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \tag{2.16}$$

$$= 100000 \oplus 010000 \oplus 001000 \oplus 000100 = 111100.$$

Полученный синдром (2.16) не равен нулю, значит, в кодовом слове есть ошибки. Число единиц в синдроме в

общем случае не определяет количество искаженных позиций кодового слова за исключением варианта, когда ошибки находятся в проверочных символах. Сравнивая значение синдрома 111100 со всеми строками матрицы H^T (2.15), устанавливаем, что совпадений нет. Это говорит о том, что одиночные ошибки отсутствуют и в наличии искажения более высокой кратности, т.е. t = 2. Далее производим суммирование в различных сочетаниях двух строк H^T и сравниваем результат с синдромом (2.16).

Итог сравнений показывает, что с (2.16) совпадает лишь сумма первой и второй строк H^T, т.е. $001111 \oplus 110011 = 111100$.

Такой результат свидетельствует, что ошибки находятся в первой и второй позициях кодового слова $\bar{V}$ и далее могут быть исправлены.

Общая структурная схема коррекции ошибок, реализующая этот метод, показана на рис. 2.4.

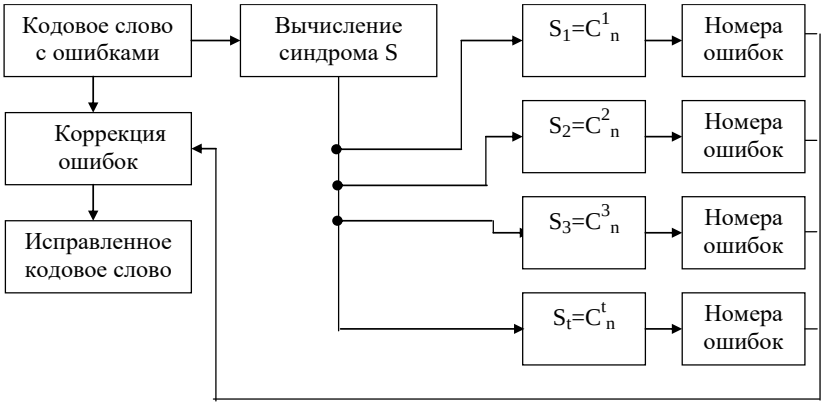


Рис. 2.4. Структурная схема переборного алгоритма коррекции ошибок блоковыми кодами

После вычисления синдрома S начинается процедура его сравнения с синдромами ошибок S_1, S_2 ... S_t. Синдромы ошибок предварительно получают путем различных вариантов

сочетаний строк транспонированной проверочной матрицы кода $H^T_{n,k}$ от одной до t и хранят в долговременной памяти устройства. Каждому синдрому ошибок соответствуют номера позиций символов кодового слова, подлежащих исправлению. Эта информация поступает в блок коррекции ошибок, где искаженные символы в кодовом слове инвертируются, и тем самым осуществляется их исправление. Исправленное кодовое слово после завершения декодирования поступает к получателю информации.

Недостатком такого переборного метода исправления ошибок является необходимость иметь достаточно большой объем долговременной памяти, обеспечивающий исправление многократных искажений в символах кодового слова за счет длительной процедуры сравнений.

2.4.2. Декодирование блоковых кодов по методу максимального правдоподобия

Название "метод максимального правдоподобия" связано с тем, что разрешенная кодовая комбинация, принятая с ошибкой, сравнивается здесь со всеми разрешенными кодовыми комбинациями, и среди них отыскивается "максимально правдоподобная" принятой по минимуму различия между ними.

Как упоминалось в п. 2.1, при сложении двух разрешенных кодовых комбинаций не содержащих ошибок число единиц в результирующей комбинации должно быть $\geq d$. При том же сложении, но при наличии в одном из слов t ошибок результирующее число единиц будет $\geq d - t$. Когда происходит сложение разрешенной кодовой комбинации, имеющей искажения, со своей копией без ошибок, число единиц получается равным t.

В том случае, когда для принятого кодового слова величина $d - t > t$, удается при полном сравнении со всеми кодовыми разрешенными комбинациями найти по

минимуму числа единиц ту, которая ранее передавалась без ошибок, и выдать ее как результат декодирования.

Алгоритм декодирования по методу максимального правдоподобия на основе такого подхода может быть построен следующим образом:

1. После приема n-элементной кодовой разрешенной комбинации $\tilde{V}$ представляющей собой одну из множества $M_{раз}$ разрешенных кодовых комбинаций кода, но с ошибками, декодирующее устройство последовательно складывает ее по модулю 2 со всеми кодовыми словами $\bar{V}_i$ из $M_{раз}$ и вычисляет векторы ошибок $\vec{E}_i = \tilde{V} \oplus \bar{V}_i$

2. Декодирующее устройство подсчитывает число единиц в каждой из последовательностей $\vec{E}_i$.

3. Декодирующее устройство принимает решение, что та из множества разрешенных кодовых комбинаций при сложении с которой число единиц в $\vec{E}_i$ было минимально, соответствует разрешенной кодовой комбинации без ошибок $\bar{V}_i$.

Как видно, данный алгоритм предусматривает хранение в памяти декодирующего устройства всех разрешенных кодовых комбинаций и проведение их перебора при вычислении векторов ошибок. При большой длине информационного слова это требует значительных объемов памяти и больших затрат времени на процедуру сравнения.

В связи с этим декодирование по методу максимального правдоподобия не имеет широкого распространения в практике коррекции ошибок. Его используют для блоковых кодов с ограниченным значением n.

2.5 Лабораторное задание

1. Изучить принцип построения образующей матрицы блокового кода. Для этого выполнить следующие пункты:

1.1. Включить ЭВМ и войти в диск Z. Выбрать файл LABRNEW и открыть его. Затем открыть файл LAB2Win и далее файл Lab2 [17,18] (рис. 2.5).

1.2. Выбрать вид моделирования «Построение образующей матрицы». Ввести кодовое расстояние d = 5, соответствующее t = 2 и число проверочных символов r = 7.

1.3. Число из массива M(1), с которого нужно осуществлять построение образующей матрицы, взять равным 15. Нажимая на кнопку «Далее», получить образующую матрицу. Отразить в отчете порядок ее построения, используя комментарии с экрана через полосу прокрутки. Определить параметры полученного кода.

1.4. Повторить построение с d = 5 и r = 5. Объяснить в отчете, почему невозможно получение образующей матрицы.

1.5. Повторить построение с d = 5 и r = 6. Объяснить в отчете причину получения только двух строк образующей матрицы.

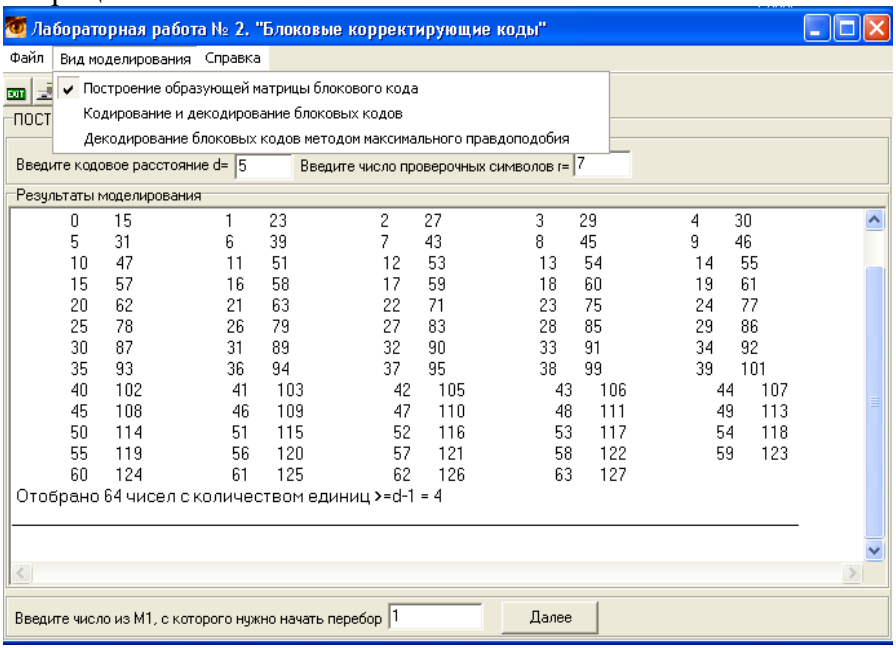


Рис. 2.5. Окно заставки для проведения лабораторных исследований с режимами работы и результатами моделирования

1.6. Повторить построение с d = 5 и r = 8. Зафиксировать в отчете основные фрагменты получения образующей матрицы и записать ее в отчет.

1.7. Построить образующую матрицу для d = 7 и r = 11. Число из M(1), с которого нужно осуществлять перебор, принять равным 127. Записать матрицу в отчет. Изменить начальное число на 1884 и построить матрицу. Ввести начальное число 2001 и повторить построение. Отразить в отчете полученные изменения. Сделать выводы.

1.8. Построить образующую матрицу для d = 7 и r = 16. Число из M(1), с которого нужно начинать перебор, взять равным 2001. Отразить в отчете параметры полученного кода.

2. Изучить принцип кодирования и декодирования блоковыми кодами. Для этого выполнить следующие пункты:

2.1. Выбрать вид моделирования «Кодирование и декодирование блоковыми кодами».

2.2. Ввести d = 3, затем r = 3. Это соответствует коду Хемминга (7,4), исправляющему однократные ошибки. Нажимать на кнопку «моделирование». Записать образующую матрицу в отчет.

2.3. Ввести информационные символы, например, число 10 и нажать на кнопку «Далее». По введенным информационным символам программа осуществит получение проверочных символов. Отразить в отчете порядок их получения по известным информационным символам.

2.4. Ввести одну ошибку в разрешенную кодовую комбинацию, набрав в окне число 1. Уточните через ввод во второе окно, в каком символе будет ошибка, например, в 5. Нажимать на кнопку «Далее». Зафиксировать изменения в кодовой комбинации, а также проверочную и

транспонированную матрицу кода, которые потребуются для процедуры декодирования.

2.5. По транспонированной проверочной матрице получен синдром. Отразить в отчете порядок его получения.

2.6. Осуществить декодирование “вручную”. Прочитать внимательно порядок исправления одиночных и кратных ошибок. Нажимая на кнопку «Далее», следовать инструкции и произвести по таблице сравнение синдрома со строками транспонированной проверочной матрицы. Выбрать в графе N номер символа с ошибкой в случае совпадения числа отражающего синдром с числом, отражающим строку транспонированной проверочной матрицы. В конкретном случае это будет 5. Введите число 5 в окно заставки и нажмите «Далее». Обратить внимание на исправление ошибки.

2.7. Повторить моделирование. Ввести d = 5 и r = 6 для кода, исправляющего двукратные ошибки, т.е. t = 2. Отразить в отчете порядок исправления любой конфигурации двукратной ошибки на конкретном примере.

2.8. Повторить моделирование. Ввести d = 5 и r = 8, соответствующие коду с t = 2. Ввести в качестве информационного символа число “0”. Отразить в отчете порядок исправления двукратной ошибки.

2.9. Повторить моделирование. Ввести d = 7 и r = 10. Записать полученный код и кратность исправляемых им ошибок. Ввести информационные символы, соответствующие числу “0”, и осуществить исправление произвольной конфигурации трехкратных ошибок в режиме “вручную”.

2.10. Повторить моделирование. Ввести d = 9 и r = 14. Записать код и кратность исправляемых им ошибок. Ввести информационные символы, соответствующие числу “0”, и осуществить исправление произвольной конфигурации четырехкратных ошибок в режиме “вручную”.

3. Изучить принцип декодирования блоковых кодов методом максимального правдоподобия. Для этого выполнить следующие пункты:

3.1. Выбрать вид моделирования «Декодирование блоковых кодов методом максимального правдоподобия».

3.2. Ввести кодовое расстояние d = 3 и r = 3 и нажать «Моделирование». Затем ввести информационные символы, например число 14 и нажать «Далее».

Ввести одну ошибку числом 1. Номер символа с ошибкой - 3. Нажать «Далее». Осуществить декодирование в режиме “вручную”.

3.3. В представленной на экране таблице выбрать минимальное расстояние Хемминга между полученной и одной из разрешенных кодовых комбинаций, определяя соответствующий этому номер N. В данном случае N = 14. Ввести номер N кодовой комбинации и нажать «Далее». Зафиксировать результат декодирования.

3.4. Повторить моделирование для d = 5 и r = 7, образующих код с t = 2. Ввести произвольное расположение двух ошибок, осуществить декодирование. Зафиксировать в отчете порядок декодирования и получаемые разрешенные кодовые комбинации. Осуществить то же самое с d = 9, r = 16 при t = 4.

Содержание отчета

1. Количественные показатели по выполнению конкретных пунктов с их комментариями и выводами.
2. Графики и таблицы по проведенным измерениям.
3. Краткие выводы по лабораторной работе.

Контрольные вопросы

1. Что такое блоковый корректирующий код? Какими основными параметрами характеризуется этот код?

2. Дайте определение кодового расстояния d блокового кода. Как величина d может быть определена из набора разрешенных комбинаций кода?

3. Что такое производящая (образующая) матрица блокового кода и как она строится?

4. Что такое проверочная матрица блокового кода и как она строится?

5. Как строится транспонированная проверочная матрица блокового кода?

6. Какое существует соотношение между производящей и транспонированной матрицами блокового кода?

7. Как определить синдром блокового кода?

8. Что характеризует синдром и какое его основное свойство используется при декодировании блокового кода?

9. Дайте определение границ Хемминга и Варшамова-Гилберта для блокового кода, что они означают?

10. Как осуществляется кодирование блоковыми корректирующими кодами?

11. Как осуществляется декодирование блоковых корректирующих кодов синдромным методом?

12. Как осуществляется декодирование блоковых кодов методом максимального правдоподобия?

3. ЛАБОРАТОРНАЯ РАБОТА №3
ЦИКЛИЧЕСКИЕ КОРРЕКТИРУЮЩИЕ КОДЫ

Цель работы: изучить построение, а также методы кодирования и декодирования информации циклическими корректирующими кодами.

Теоретическая часть

3.1. Введение в циклическое кодирование

Как было показано (п. 2.1), наиболее трудоемкими задачами при использовании блокового кода являются построение его производящей матрицы и организация процедур декодирования, требующие выполнения большого числа сравнений кодовых комбинаций и значительного обьема памяти для их хранения и обработки. Сократить время декодирования и изменить правила построения корректирующих кодов возможно, если кодовые комбинации представить в виде двоичных многочленов, для которых доступно осуществление арифметических операций (подобно десятичным числам) и использовать неприводимые многочлены (аналог простых десятичных чисел для многочленов). Процедура кодирования (получение кодового слова) в этом случае может быть представлена, например, через деление кодовой комбинации на неприводимый многочлен и получение остатка от деления, используемого далее в виде проверочных символов. Декодирование осуществляется на приемной стороне как деление принятого кодового слова на известный неприводимый многочлен и проведение анализа полученного остатка. В том случае, когда остаток равен нулю - ошибок нет. При их наличии по конкретному значению остатка может быть определенно положение ошибок и осуществленно их исправление.

Корректирующий код, использующий аналогичные процедуры имеет ту особенность, что циклический сдвиг кодового слова приводит к получению другого кодового слова, принадлежащего этому же коду, и поэтому называется циклическим блоковым кодом. Такая особенность позволяет также представить циклический код по одной кодовой комбинации неприводимого многочлена (образующей кодовой комбинации), что намного удобнее принятого представления блокового кода через образующую матрицу.

Для лучшего понимания теоретических и практических вопросов циклических кодов, рассмотрим, прежде всего, понятие о двоичных многочленах и действиях над ними.

3.2. Двоичные многочлены и действия над ними

При описании циклических кодов возможна запись любого n-разрядного двоичного числа в виде многочлена степени n-1, содержащего фиктивную переменную x [20].

Показатели степени у x соответствуют номерам разрядов, а коэффициентами при x являются цифры 0 или 1. При этом наименьшему разряду числа соответствует фиктивная переменная $x^0 = 1$. Запишем, например, в виде многочлена пятиразрядную кодовую комбинацию 10101, т.е.

$$G(x) = 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x + 1 \cdot x^0 .$$

Поскольку члены с нулевыми коэффициентами при записи многочлена опускаются, окончательно получим

$$G(x) = x^4 + x^2 + 1.$$

Теперь действия с кодовыми комбинациями можно свести к действиям над многочленами.

Умножение одного многочлена на другой состоит из двух этапов:

1) умножение одного многочлена на другой по правилам алгебры,

2) операций сложения по модулю 2.

Например, перемножим два многочлена

$$x^4 + x^2 + x + 1 \rightarrow 10111$$
$$\times \quad x \rightarrow 10$$
$$00000$$
$$10111$$
$$= x^5 + x^3 + x^2 + x \rightarrow 101110.$$

Как видно, в случае наличия единицы в старшем разряде многочлена происходит повышение степени у переменной x.

Так как кодовая комбинация циклического кода определяется длиной регистра или количеством его ячеек n, замкнутых в кольцо, то повышение степени здесь недопустимо. В данном случае принято производить символическое умножение кодовой комбинации на x через циклический сдвиг без увеличения максимальной (n-1)-й степени многочлена.

Пусть, например, циклически сдвигается (влево) кодовая комбинация 10111 с единицей в старшем разряде, т.е.:

$$1\ 0\ 1\ 1\ 1$$
$$0\ 1\ 1\ 1\ 1.$$

Такая операция соответствует умножению многочлена степени n-1 на x с одновременным вычитанием из результата умножения многочлена $x^n + 1$, т.е. с приведением многочлена по модулю $x^n + 1$.

Пример такой операции выглядит следующим образом:

$$1\ 0\ 1\ 1\ 1 \quad \rightarrow \quad x^4 + x^2 + x + 1,$$

$$x^4 + x^2 + x + 1 \qquad\qquad x^5 + x^3 + x^2 + x$$

$$\times \underline{\qquad\qquad x \qquad}, \quad \underline{\oplus\ x^5 + 1}$$

$$x^5 + x^3 + x^2 + x \qquad\qquad x^3 + x^2 + x + 1 \rightarrow 01111,$$

т.е. для регистра сдвига, замкнутого в кольцо, это соответствует переносу единицы из старшего разряда ячейки в младший разряд с одновременным сдвигом остальной части кодовой комбинации на одну ячейку влево.

В том случае, когда разрядность регистра превышает разрядность вводимого в него числа, умножению на x будет соответствовать его сдвиг на один разряд влево. Так умножим, например, многочлен $x^3 + x^2 + 1$, отражающий в регистре комбинацию 0001101, на x. В результате получим многочлен $x^4 + x^3 + x$, соответствующий комбинации 0011010, а сравнение с предыдущим случаем показывает наличие сдвига на один элемент влево. Умножение многочлена, находящегося в регистре при тех же условиях, на многочлен x^k следует осуществить циклическим сдвигом его на k разрядов влево.

Сложение и вычитание многочленов равносильны и производятся через сложение по модулю 2, обозначаемому $\oplus$ или иногда для упрощения через +.

Например, сложим две комбинации

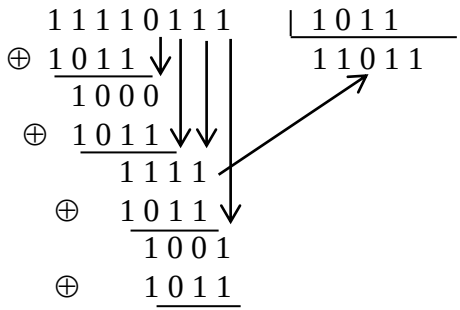
$$\begin{array}{l} x^7 + x^5 + x^3 + x^2 + 1 \rightarrow 10101101 \\ \underline{\oplus \qquad\qquad x^5 + x + 1 \rightarrow \ 00100011} \\ \qquad x^7 + x^3 + x^2 + x \rightarrow \ 10001110\,. \end{array}$$

Деление одного многочлена на другой производится по правилам алгебры за исключением входящих в процедуру операций вычитания, которые здесь равносильны сложению и производятся по модулю 2.

Деление многочлена на многочлен осуществляется до тех пор, пока степень остатка не станет меньше степени делителя, что соответствует числу разрядов остатка, меньшему числа разрядов делителя. Например, возьмем два многочлена

$x^7 + x^6 + x^5 + x^4 + x^2 + x + 1 \rightarrow 1\,1\,1\,1\,0111,$

$x^3 + x + 1 \rightarrow 1\,0\,1\,1,$



Таким образом, операции сложения, умножения и деления многочленов могут быть использованы для различных вычислений и процедурных преобразований применительно к циклическим кодам. В таких действиях также используются образующие многочлены, которые являются неприводимыми.

Неприводимыми многочленами называются такие, которые не могут быть представлены в виде произведения многочленов низших степеней, что подобно делению без остатка только на себя и единицу для простых чисел, используемых при проведении арифметических вычислений. Неприводимые многочлены получают из общего множества многочленов путем отбора их через основное свойство неприводимости.

Например, имеется только два многочлена первой степни, т.е. x и x +1, которые по определению оба неприводимы. Неприводимым многочленом второй степени будет являтся многочлен $x^2 + x + 1$. Он выбран из трех возможных многочленов второй степени: $x^2 + x = x(x + 1)$, $x^2 +1 = (x +1)(x +1)$ и $x^2 +x +1$, так как последний не может быть

представлен в виде произведений многочленов первой степени x и x +1. Аналогичным образом могут быть получены неприводимые многочлены третьей степени, которых будет два: $x^3 + x^2 + 1$ и $x^3 + x + 1$. В [5] предложен способ нахождения неприводимых многочленов, который наиболее эффектино осуществлять на ЭВМ. Таблицу неприводимых многочленов до 34-й степени включительно можно найти в [20]. Некоторые из них представлены в таблице.

Неприводимый многочлен	Десятичное представление
$x + 1 \rightarrow 11$	3
$x^2 + x + 1 \rightarrow 111$	7
$x^3 + x + 1 \rightarrow 1011$	11
$x^3 + x^2 + 1 \rightarrow 1101$	13
$x^4 + x + 1 \rightarrow 10011$	19
$x^4 + x^3 + 1 \rightarrow 11001$	25
$x^4 + x^3 + x^2 + x + 1 \rightarrow 11111$	31
$x^5 + x^2 + 1 \rightarrow 100101$	37
$x^5 + x^3 + 1 \rightarrow 101001$	41
$x^5 + x^3 + x^2 + x + 1 \rightarrow 101111$	47
$x^5 + x^4 + x^2 + x + 1 \rightarrow 110111$	55
и т.д.	

По аналогии с десятичными простыми числами, где имеются зависимости

$2^3 - 1 = 7$, $2^4 - 1 = 3 \times 5$, $2^5 - 1 = 31$,

$2^6 - 1 = 3 \times 3 \times 7$, $2^7 - 1 = 127$, $2^8 - 1 = 3 \times 5 \times 17$ и т.д.,

показано [5], что двучлен $x^n + 1$, где $n = 2^r - 1$ может быть представлен также набором произведений неприводимых многочленов, степени которых являются делителями числа r (от единицы до r включительно).

Так, для $n = 15 = 2^4 - 1$ степени неприводимых многочленов для их произведений будут 1, 2, 4, т.е.

$x^{15} + 1 = (x + 1)(x^2 + x + 1)(x^4 + x + 1)(x^4 + x^3 + 1)(x^4 + x^3 + x^2 + x + 1)$.

Неприводимые многочлены из таблицы могут быть использованы для построения образующих матриц $G_{n,k}$ циклических кодов, которые применяются для кодирования, т.е. получения кодовых слов. Причем в этом случае неприводимый многочлен называется образующим многочленом (полиномом) циклического кода и обозначается через P(x). Например, для циклического кода (7, 4), исправляющего одну ошибку, может быть использован полином $P(x) = x^3 + x + 1 \rightarrow 1011$, и по нему построена образующая матрица $G_{7,4}$.

Чтобы построить такую матрицу, необходимо, найти ее правую часть, которая получается путем деления кодовых комбинаций единичной матрицы, дополненных тремя нулями справа для получения длин всех строк $G_{7,4}$ n = 7, на образующий полином и определения остатков. Полученные остатки будут являться правой частью производящей матрицы $G_{n,k}$.

Рассмотрим это на примере кода (7,4) с

$$P(x) = x^3 + x + 1 \rightarrow 1011.$$

Кодовые комбинации, подлежащие делению, являются строками единичной матрицы, т.е.: 1000; 0100; 0010; 0001.

При их делении на $P(x) \rightarrow 1011$ необходимо предварительно к каждой комбинации записать три дополнительных нуля справа, что эквивалентно также умножению их на x^3, выполняемому сдвигом на три элемента влево. Начнем с комбинации 0001, тогда

```
1 0 0 0   | 1011
1 0 1 1     1
  0 1 1 →   остаток.
```

Для остальных комбинаций имеем

$$
\begin{array}{llll}
10000 & \underline{1011} & 100000 & \underline{1011} \\
\underline{1011} & 1 & \underline{1011} & 1 \\
110 \rightarrow \text{остаток} & & 1100 & \\
 & & \underline{1011} & \\
 & & 111 \rightarrow \text{остаток} &
\end{array}
$$

$$
\begin{array}{ll}
1000000 & \underline{1011} \\
\underline{1011} & 11 \\
1100 & \\
\underline{1011} & \\
1110 & \\
\underline{1011} & \\
101 \rightarrow \text{остаток.} &
\end{array}
$$

Теперь можно построить образующую матрицу $G_{7,4}$, используя полученные при делении остатки, т.е.

$$
G_{7,4} = \left| \begin{array}{ll}
1000 & 101 \\
0100 & 111 \\
0010 & 110 \\
0001 & 011
\end{array} \right| .
$$

Эту же процедуру можно представить в ином виде, осуществляя сдвиги комбинации 011 и приведение ее по модулю P (x) $\rightarrow$ 1011, там, где это необходимо, т.е. определение остатка от деления. Построение начинается с записи нижней строки матрицы $G_{7,4}$, отражающей P (x), т.е. 0001011. Затем осуществляют циклический сдвиг элементов комбинации 011, с записью слева к ней второй строки единичной матрицы, т.е.

0010 110
0001 011.

Третий сдвиг для комбинации 110 осуществлять нельзя, т. к. левый крайний символ равен единице и перемещение такой комбинации приведет к увеличению разрядности слова за счет умножения на x $\rightarrow$10, т.к. 110×10=1100. Здесь необходимо провести деление числа на P (x) и записать результат в виде остатка в матрицу, осуществляя дальнейший сдвиг, если крайний слева символ не единица, т.е.

$$
\begin{array}{ll}
1100 & \underline{1011} \\
+ \underline{1011} &
\end{array}
$$

1 1 1→ остаток.

В результате получим, что три строки матрицы будут иметь вид

0 1 0 0 1 1 1
0 0 1 0 1 1 0
0 0 0 1 0 1 1.

Комбинацию 111 тоже нельзя сдвигать. Осуществляем ее деление на P (x), умножив предварительно на x →10, т.е.

1 1 1 0 | 1 0 1 1
1 0 1 1
1 0 1 → остаток.

В результате получаем производящую матрицу циклического кода $G_{7,4}$ с t =1

$$\left|\begin{array}{cc} 1\,0\,0\,0 & 1\,0\,1 \\ 0\,1\,0\,0 & 1\,1\,1 \\ 0\,0\,1\,0 & 1\,1\,0 \\ 0\,0\,0\,1 & 0\,1\,1 \end{array}\right|.$$

3.3. Кодирование информации циклическими кодами

Ввиду того, что производящая матрица $G_{n,k}$ циклического кода аналогична по построению блоковому коду, то и кодирование в принципе может осуществляться так же, как и для блоковых кодов.

Однако у циклических кодов более целесообразно производить кодирование, используя свойство цикличности, реализуя его на регистрах сдвига [20].

Вначале заданный многочлен сообщения $Y_и(x)$, соответствующий k- разрядной информационной последовательности, умножают на x^r, где r = n- k – число

проверочных символов, равное степени образующего многочлена. Это равносильно добавлению к информационной комбинации r нулей справа или продвижению комбинации в регистре сдвига на r ячеек. Второй этап кодирования заключается в делении произведения $Y_и(x) \cdot x^r$ на образующий полином кода P(x) и получении остатка, который и будет представлять проверочные символы, добавляемые затем к информационной последовательности $Y_и(x)$.

Приведем пример кодирования для кода (7, 4) , когда $Y_и(x) \rightarrow 1101$ и $P(x) \rightarrow 1011$.

Умножаем $Y_и(x)$ на x^r, т.е. 1101×1000, или в многочленах: $(x^3 + x^2 + 1)x^3 = x^6 + x^5 + x^3 \rightarrow 1101000$. Разделим далее полученное произведение $Y_и(x) \cdot x^r$ на P(x), т.е.

```
1   (1 1 0)1 0 0 0      |1 0 1 1
    1 0 1 1               1 1 1 1
2    (1 1 0)0
      1 0 1 1
3      (1 1 1)0                          (3.1)
        1 0 1 1
4        (1 0 1)0
          1 0 1 1
5          (0 0 1)  → остаток.
```

В результате кодовая комбинация будет иметь следующий вид: 1101001. В начале полученного кодового слова располагается k информационных символов $Y_и(x) \rightarrow 1101$, а затем r проверочных символов 001.

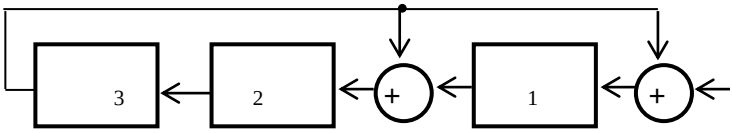
Технически представленная выше процедура осуществляется в регистре сдвига с числом ячеек r (высшая степень полинома). Операция умножения реализуется сдвигом информационной последовательности на r ячеек в регистре, а деление произведения $Y_и(x) \cdot x^r$ производится за счет обратных связей, включенных через сумматоры по модулю 2. Количество таких сумматоров равно числу отличных от нуля

членов производящего полинома P(x) без учета его старшего разряда. Последнее условие вызвано тем, что сумма по модулю 2 старших разрядов многочлена сообщения $Y_и(x)$ и P(x) всегда равна нулю при делении “столбиком”, например, как в (3.1).

Следует отметить, что организация работы регистра сдвига с обратными связями и сумматорами по модулю 2 полностью реализует алгоритм деления “столбиком”.

Сумматоры включаются перед ячейками регистра, совпадающими при его построении со значащими слагаемыми (не нуль) производящего полинома, представленного, например, в виде единиц и нулей, кроме слагаемого, отражающего высшую степень полинома.

Приведем пример (рис. 3.1) схемы деления информационного многочлена $Y_и(x) \rightarrow 1\ 1\ 0\ 1$ на многочлен $P(x) = x^3 + x + 1 \rightarrow 1\ 0\ 1\ 1$.



$Y_и(x) \rightarrow 1101$

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

Рис. 3.1. Схема деления информационного многочлена при подаче его справа

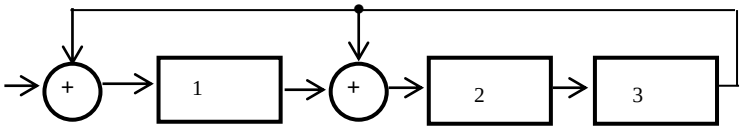
Здесь число ячеек регистра берется равным высшей степени полинома при x, т.е. 3. Перед входом стрелки в ячейки 1 и 2, совпадающие по обозначению со значащими слагаемыми полинома $P(x) \rightarrow 1011$, ставятся сумматоры по

модулю 2, на которые заводятся обратные связи с выхода регистра.

После ввода в регистр трех информационных символов $Y_и(x)$ состояние регистра будет 110 (рис. 3.1). После поступления еще одного информационного символа (1) состояние регистра за счет действия обратных связей станет 110. Это действие равносильно вычитанию многочлена делителя из многочлена делимого (см. строку 2 в (3.1)).

Таким образом, сложение по модулю 2 с P(x) происходит только тогда, когда в крайней ячейке на выходе регистра появляется единица и она по обратной связи, установленной в соответствии с P(x), поступает в другие ячейки регистра.

После этого в течение трех тактов на вход регистра подаются символы 0 и формируются проверочные символы (остаток), что соответствует трем нулям делимого в (3.1), т.е. образуются такт за тактом последовательности: 111, 101, 001. Число 001 является результатом деления и соответствует проверочным символам.



$Y_и(x) \rightarrow 1011$

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

Рис. 3.2. Схема деления информационного многочлена при подаче его слева

Регистр, изображенный на рис. 3.2, формируется из рис. 3.1, когда информационная последовательность вводится слева.

Как видно из результата деления, остаток кодируется одинаково с предыдущей схемой (рис. 3.1), только расположение символов в остатке строго наоборот, т.е. 100 вместо 001.

Деление “столбиком” в этом случае осуществляется в виде

```
1011000  |1101
1101      11
 1100
 1101
   0100 → остаток,
```

где образующий полином записывается как 1101 вместо 1011.

Техническая реализация процедуры кодирования должна учитывать, чтобы после информационных символов сразу следовали проверочные. Данные, подлежащие кодированию, вначале поступают в канал, затем следующие элементы должны “притормаживаться” в буфере памяти на время формирования проверочных символов и их дальнейшей передачи в канале.

Схема кодирующего устройства на регистрах сдвига с обратными связями для кода (7,4) показана на рис. 3.3.

Исходное положение переключателей : П3 – замкнут, П2 в 1, П1 – замкнут. За три такта регистр деления и буферный регистр заполняются данными. На четвертом такте начинается деление в регистре и одновременно вывод k символов из буферного регистра. После четвертого такта П3 размыкается и за три следующих такта буферный регистр освобождается, а также заканчивается деление в регистре. Затем П2 переводится в положение 2, П1 размыкается, П3 замыкается и за три такта выводятся проверочные символы (остаток) из регистра
деления в канал с одновременным его заполнением новыми данными из памяти, которые также поступают в буферный регистр.

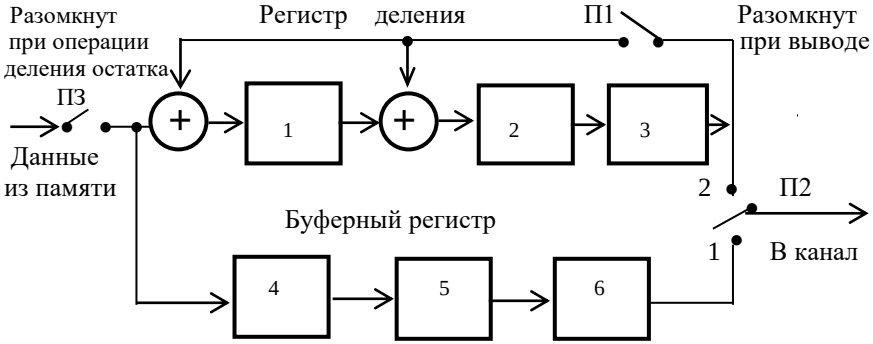


Рис. 3.3. Кодирующее устройство для циклического кода (7,4) с $P(x) = x^3 + x + 1$

Далее П1 замыкается, П2- 1, и за следующий такт поступает еще один символ данных из памяти. Потом П3 размыкается и начинает осуществляться вывод в канал символов из буферного регистра, а также деление в регистре в течение 3 тактов, т.е. процедура кодирования продолжается до выбора всех символов из памяти данных.

Таким образом, информационные символы циклического кода поступают в кодирующее устройство с перерывами на время вставки после них трех проверочных элементов.

Схему рис. 3.3 можно модернизировать, убрав буферный регистр. Это можно осуществить за счет переноса сумматора по модулю 2 с крайней левой ячейки к крайней правой. Теперь поступающие символы входной последовательности будут сразу идти в цепь обратной связи регистра деления и отпадает необходимость ждать его заполнения, как в предыдущем
кодирующем устройстве с буферным регистром.

С учетом изменений кодирующее устройство для кода (7,4) будет иметь вид, представленный на рис. 3.4.

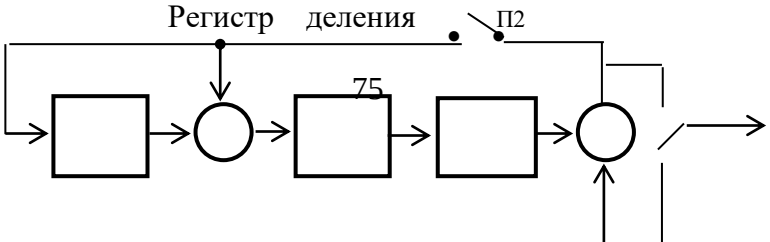


Рис. 3.4. Модернизированный вариант кодирующего устройства для кода (7,4)

Информационные символы k поступают из памяти данных через замкнутый ключ П1, в это время П3 находится в положении 1, а П2 замкнут. После четырех символов k П1 размыкается, П3 переводится в положение 2, а П2 размыкается и из регистра деления, где перед этим был вычислен остаток, в канал поступает три проверочных символа (r = 3). Как и в предыдущем случае, данные из памяти также берутся с перерывом для вставки в них проверочных символов.

3.4. Декодирование циклических кодов

3.4.1. Обнаружение ошибок циклическими кодами

Декодирование кодов с обнаружением ошибок заключается в делении принятой кодовой комбинации с ошибками на образующий полином кода P(x). Если ошибок нет, то остаток равен нулю. При наличии искажений в кодовой комбинации остаток принимает какое-то конкретное значение. Так, возьмем кодовое слово 1101001, состоящее из 4 инфор-мационных и 3 проверочных символов, полученное в резуль- тате кодирования (3.1).

Проведем теперь деление кодового слова без ошибок

1101001 на образующий полином P (x) = 1011. Деление можно осуществить, например, столбиком, т.е.

```
1 1 0 1 0 0 1  |1 0 1 1
1 0 1 1        | 1 1 1 1
  1 1 0 0
  1 0 1 1
    1 1 1 0                                   (3.2)
    1 0 1 1
      1 0 1 1
      1 0 1 1
        0 0 0  →  остаток.
```

Полученный остаток, называемый синдромом S, в этом случае равен нулю, что свидетельствует об отсутствии ошибок в кодовом слове.

При наличии одиночной ошибки, например, в третьем символе кодового слова из (3.2) 1 1 (1) 1 0 0 1, где под воздействием помех произошел переход 0 в 1, получаем после деления

```
1 1 1 1 0 0 1    | 1 0 1 1
1 0 1 1
  1 0 0 0
  1 0 1 1
      1 1 0 1                                 (3.3)
      1 0 1 1
        1 1 0 ,
```

что синдром (остаток) S = 110 и это свидетельствует о наличии ошибки в кодовом слове.

Техническую реализацию процедур декодирования с обнаружением ошибок можно реализовать на регистрах сдвига с обратными связями, например, по рис. 3.1 или по рис. 3.2.

Процедура, аналогичная приведенной выше, используется в модемах при передаче информации между

ЭВМ [3]. Здесь может использоваться, например, циклический код (1024, 16) с обнаружением ошибок и полиномом $P(x) = x^{16} + x^{12} + x^5 + 1$. Схема деления в этом случае будет иметь вид, представленный на рис. 3.5.

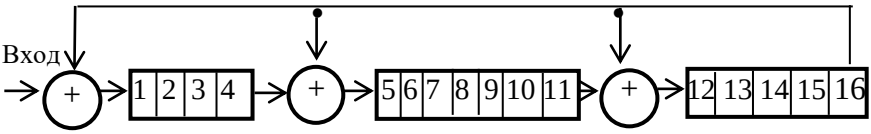


Рис. 3.5. Деление входной последовательности на полином $P(x) = x^{16} + x^{12} + x^5 + 1$

Применение кода с полиномом $P(x) = x^{16} + x^{12} + x^5 + 1$ гарантирует обнаружение практически всех одиночных и двойных ошибок, всех нечетных ошибок, всех пакетов ошибок длиной до 16 элементов, а также всех 17- битовых ошибок с вероятностью обнаружения $P_{обн} = 0{,}999969$ и более длинных пакетов с $P_{обн} = 0{,}999984$ [3].

3.4.2. Декодирование циклических кодов методом “вылавливания” ошибок

Прежде чем рассмотреть метод “вылавливания” ошибок, укажем общий подход к декодированию циклических кодов.

Исправление ошибок здесь можно осуществлять по виду остатка (синдрома) от деления разрешенной кодовой комбинации (кодового слова) на образующий полином кода, которое выполняется обычно на регистре сдвига с обратными связями.

Поскольку циклический код является подмножеством линейного блокового кода, то для него можно построить производящую матрицу $G_{n,k}$, а также проверочную $H_{n,k}$ и транспонированную матрицу $H^T_{n,k}$ и, следовательно, возможно применять алгоритмы декодирования, аналогичные для блоковых кодов.

Транспонированная проверочная матрица для циклического кода (7,4) может быть получена в виде

$$H^T_{7,4} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} \qquad (3.4)$$

подобно блоковому коду Хемминга (7,4).

Теперь, если по кодовому слову 1111001, имеющему ошибку в третьем символе слева, вычислить синдром методом деления (3.3), то его значение S = 110 будет совпадать с третьей строкой $H^T_{7,4}$ (3.4).

Это указывает на ошибку в символе и после его инвертирования, кодовая комбинация считается исправленной.

Циклическая структура кода (7,4) позволяет упростить эту процедуру сравнения через циклические сдвиги кодовой комбинации в регистре и последующие вычисления синдромов, определяя те, которые соответствуют ошибкам в проверочных разрядах r.

Дело в том, что только эти синдромы имеют вес (число единиц) $W \leq t$, что в конкретном примере видно из (3.4) по номерам 5, 6, 7 H^T.

Таким образом, сдвинув все ошибки в кодовой комбинации в проверочные разряды, легко определить их положение,

так как они совпадают с позициями единиц синдрома, вес которого $W \leq t$.

Проблема в таком методе декодирования состоит в том, что сдвиг всех ошибочных разрядов в кодовой комбинации на позиции проверочных разрядов осуществляется только в том случае, когда $t < (n / k)$ [20].

Это условие выполняется, например, для кода (15,7), где $t = 2$, но для кода (15,5), где $t = 3$, оно не выполняется, и часть ошибок останется неисправленной. Например, как показали исследования, проведенные на базе программного модуля [21], код (15,5) с проверочным полиномом $P(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$ будет исправлять все однократные и двукратные ошибки, но среди трехкратных ошибок часть останется неисправленной и общее их число составляет около одного процента. Схема алгоритма декодирования методом вылавливания ошибок представлена на рис. 3.6.

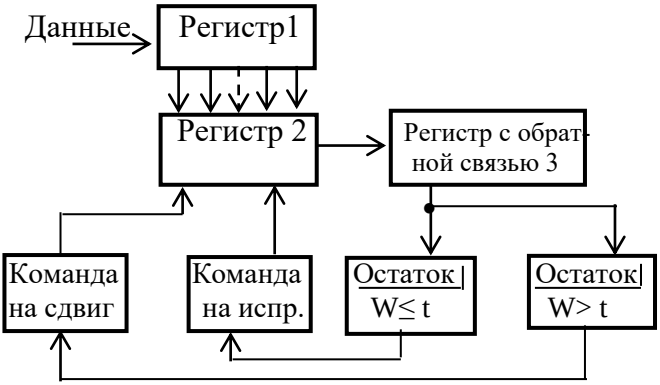


Рис. 3.6. Структурная схема алгоритма декодирования циклического кода

В регистр 1 осуществляется запись кодового слова длиной n и далее перезапись его в регистр 2 для дальнейшего декодирования, пока поступает новое слово в регистр 1.

В регистре 3 вычисляется синдром S как результат деления кодового слова на образующий полином. Затем происходит определение веса синдрома W. При $W > t$ подается команда на циклический сдвиг кодового слова, записанного в регистре 2, и снова вычисляется S. Такая процедура происходит до тех пор, пока не выполнится условие $W \leq t$.

В этом случае подается команда на исправление, осуществляемая как инвертирование символов в элементах кодового слова, соответствующих единицам в синдроме S, и затем следует сдвиг кодового слова в регистре 2 в обратную сторону на число тактов, предшествующих выполнению условия $W \leq t$.

Ошибки в кодовом слове исправлены, и оно выводится из регистра 2 для выделения информационных символов, а из регистра 1 в регистр 2 поступает следующее кодовое слово.

Рассмотрим пример декодирования, используя циклический код (15,5) с $t = 3$ и

$$P(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1 \rightarrow 10100110111.$$

Образующая матрица кода (15,5) строится обычным образом по P (x), как в п. 3.2 и имеет вид

$$G_{15,5} = \begin{bmatrix} 1\,0\,0\,0\,0 & 1\,0\,1\,0\,0\,1\,1\,0\,1\,1 & 1 \\ 0\,1\,0\,0\,0 & 1\,1\,1\,1\,0\,1\,0\,1\,1\,0 & 2 \\ 0\,0\,1\,0\,0 & 0\,1\,1\,1\,1\,0\,1\,0\,1\,1 & 3 \\ 0\,0\,0\,1\,0 & 1\,0\,0\,1\,1\,0\,1\,1\,1\,0 & 4 \\ 0\,0\,0\,0\,1 & 0\,1\,0\,0\,1\,1\,0\,1\,1\,1 & 5. \end{bmatrix} \qquad (3.5)$$

По (3.5) строится транспонированная проверочная матрица H^T, у которой строками являются элементы правой

части образующей матрицы, а затем она добавляется матрицей G_1 (единичной матрицей). В таком виде H^T может использоваться для декодирования

$$H^T_{15,5} = \begin{bmatrix} 1\,0\,1\,0\,0\,1\,1\,0\,1\,1 & 1 \\ 1\,1\,1\,1\,0\,1\,0\,1\,1\,0 & 2 \\ 0\,1\,1\,1\,1\,0\,1\,0\,1\,1 & 3 \\ 1\,0\,0\,1\,1\,0\,1\,1\,1\,0 & 4 \\ 0\,1\,0\,0\,1\,1\,0\,1\,1\,1 & 5 \\ 1\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 6 \\ 0\,1\,0\,0\,0\,0\,0\,0\,0\,0 & 7 \\ 0\,0\,1\,0\,0\,0\,0\,0\,0\,0 & 8 \\ 0\,0\,0\,1\,0\,0\,0\,0\,0\,0 & 9 \\ 0\,0\,0\,0\,1\,0\,0\,0\,0\,0 & 10 \\ 0\,0\,0\,0\,0\,1\,0\,0\,0\,0 & 11 \\ 0\,0\,0\,0\,0\,0\,1\,0\,0\,0 & 12 \\ 0\,0\,0\,0\,0\,0\,0\,1\,0\,0 & 13 \\ 0\,0\,0\,0\,0\,0\,0\,0\,1\,0 & 14 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,1 & 15 \end{bmatrix} . \qquad (3.6)$$

Будем использовать матрицы (3.5.) и (3.6.) как вспомогательные для объяснения процедуры декодирования циклического кода методом “вылавливания” ошибок, чтобы не использовать регистры сдвига с обратными связями, как менее наглядные для этой цели.

Возьмем информационную комбинацию $Y_и$ = 11000 и осуществим кодирование по (3.5), что эквивалентно кодированию в регистре по рис. 3.3 или рис. 3.4. Результат кодирования будет иметь вид 110000101001101, где десять проверочных символов r получены сложением по модулю 2 позиций 1 и 2 правой части строк матрицы (3.5).

Пусть после воздействия помехи возникли ошибки в третьем, четвертом и пятом символах кодового слова, т.е. 111110101001101.

Определим синдром S методом деления в столбик

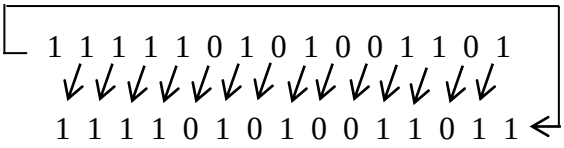
```
111110101001101   |10100110111
10100110111
 10111000111
 10100110111
    11110000101
    10100110111                                   (3.7)
     1010110010  → S,   W > t  = 5.
```

Такой же результат получается по (3.6), если сложить строки 3,4,5, т.е.

```
     0111101011
⊕    1001101110                                   (3.8)
     0100110111
     1010110010  → S,   W > t  = 5.
```

Как видно из (3.7) и (3.8), вес синдрома (W = 5) больше числа ошибок t = 3, исправляемых кодом и, следовательно, необходимо осуществлять сдвиг кодового слова, вычисляя новый синдром S с последующим определением его веса W и сравнением с t.

Сдвиг кодового слова с ошибками 111110101001101 осуществим в регистре сдвига, т.е.



(3.9)

По сдвинутому кодовому слову (3.9) вычисляем синдром S через (3.6), что эквивалентно делению в регистре и определяется сложением строк 1, 2, 3, 4, 6, 8, 11, 12, 14, 15, т.е.

$$
\begin{array}{ll}
 & 1\,0\,1\,0\,0\,1\,1\,0\,1\,1 \\
 & 1\,1\,1\,1\,0\,1\,0\,1\,1\,0 \\
 & 0\,1\,1\,1\,1\,0\,1\,0\,1\,1 \\
 & 1\,0\,0\,1\,1\,0\,1\,1\,1\,0 \\
\oplus & 1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\
 & 0\,0\,1\,0\,0\,0\,0\,0\,0\,0 \\
 & 0\,0\,0\,0\,0\,1\,0\,0\,0\,0 \\
 & 0\,0\,0\,0\,0\,0\,1\,0\,0\,0 \\
 & 0\,0\,0\,0\,0\,0\,0\,0\,1\,0 \\
 & \underline{0\,0\,0\,0\,0\,0\,0\,0\,0\,1} \\
 & 0\,0\,0\,1\,0\,1\,0\,0\,1\,1 \rightarrow \mathrm{S},\quad W > t\ = 4.
\end{array}
\qquad (3.10)
$$

Ввиду того, что снова в (3.10) $W > t = 4$, осуществляем следующий сдвиг кодового слова из (3.9) с последующим вычислением синдрома и определением его веса W по (3.6). Как показывает аналогичная процедура и последующие ей, только на четвертом сдвиге будет выполняться условие, когда вес синдрома $W = t$, т.е.

$$
\begin{array}{l}
1\,1\,1\,1\,0\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1 \\
1\,1\,1\,0\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,1 \\
1\,1\,0\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,1\,1 \\
1\,0\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,1\,1\,1 \\
0\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,1\,1\,1\,1\,.
\end{array}
\qquad (3.11)
$$

Деление последнего кодового слова из (3.11) на образующий полином дает следующий результат:

$$
\begin{array}{l}
\quad k \qquad\quad r \\
01010\bullet 0110111111 \ \big|\ \underline{10100110111} \\
\underline{1010\bullet 0110111} \\
0000\bullet 0000000111 \rightarrow \mathrm{S},\quad W = t\ = 3.
\end{array}
\qquad (3.12)
$$

Как видно из (3.12), вес синдрома W равен числу исправляемых ошибок $t = 3$ и, следовательно, позиции ошибок

определены. Они, как следует из синдрома S = 0000000111, находятся на трех последних позициях кодового слова. Изменив их на противоположные в нижней строке (3.11), получаем исправленное кодовое слово 010100110111000. Теперь, сдвинув его обратно вправо на число сдвигов, осуществленных ранее влево (3.11), т.е. пять – получим исходное переданное кодовое слово 110000101001101, что видно из следующих сдвигов:

010100110111000
001010011011100
000101001101110
000010100110111
100001010011011
110000101001101 → исходное кодовое слово.

Декодирование методом вылавливания ошибок в приведенном случае успешно завершено.

К сожалению, этот метод исправляет все ошибки до t включительно, если выполняется условие $t < (n / k)$, т. е. когда удается сдвинуть все комбинации ошибок в проверочные разряды r и произвести далее их исправление. В других случаях некоторое количество ошибок не исправляется.

Чтобы исключить такой вариант, применяют комбинированный метод, когда алгоритм исправления ошибок путем их “вылавливания” сочетается с синдромным декодированием, как у блоковых кодов. В этом случае синдромным методом исправляется только небольшая часть ошибок, а все остальные – методом вылавливания ошибок.

3.4.3. Декодер Меггита для циклических кодов

Декодер Меггита [12] использует метод “вылавливания” ошибок и основан на следующих свойствах циклического кода:

1) существует взаимно однозначное соответствие между множеством всех исправляемых ошибок и множеством всех синдромов;

2) если комбинация ошибок в кодовом слове циклически сдвинута на один символ, то для получения нового синдрома следует сдвинуть содержимое регистра сдвига с обратными связями, содержащего синдром S, также на один символ.

Поясним свойство 2 на примере циклического кода (15,5), использованного в п. 3.4 для декодирования методом “вылавливания” ошибок.

Здесь (рис. 3.6) после сдвига кодового слова в регистре 2 новый синдром вычислялся путем деления в регистре с обратными связями 3, что иллюстрируется на примерах (3.9) и (3.10).

Аналогичную процедуру можно осуществить, используя свойство 2 циклического кода, т.е. после сдвига кодового слова не вычислять вновь синдром в регистре 3 (рис. 3.6), а циклически сдвинуть его содержимое на один такт, тогда будет получено новое значение S, как в (3.10).

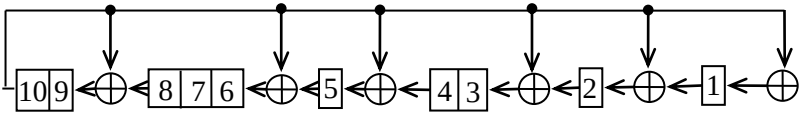
Покажем это на примере регистра с обратными связями (рис. 3.7), образованного по полиному кода (15, 5)

$$P(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1.$$

Из рис. 3.7 видно, что после одного сдвига синдрома S=1010110010(1) образуется новый синдром S=0001010011(2), соответствующий процедуре (3.10). После пяти сдвигов первоначального синдрома получается вариант синдрома с тремя единицами, как в (3.12).

Таким образом, структурную схему алгоритма декодирования по рис. 3.6 можно упростить, используя вышеизложенное свойство циклического кода, т.е. отпадает необходимость

каждый раз вычислять синдром после очередного сдвига кодового слова в регистре 2.



1 0	1 0 1	1	0 0	1	0 (1)
0 0	0 1 0	1	0 0	1	1 (2)
0 0	1 0 1	0	0 1	1	0 (3)
0 1	0 1 0	0	1 1	0	0 (4)
1 0	1 0 0	1	1 0	0	0 (5)
0 0	0 0 0	0	0 1	1	1 (6)

Рис. 3.7. Вычисление синдромов в регистре

Структурная схема процедуры декодирования по алгоритму Меггита для случая, когда $t < (n / k)$, показана на рис. 3.8.

Здесь принятое кодовое слово записывается в информационный регистр и одновременно осуществляется вычисление синдрома в регистре деления на P(x). Затем проверяется условие $W \leq t$. Если оно не выполняется, то переключатель П1 устанавливается в положение 1, и начинают синхронно осуществляться сдвиги в информационном регистре и регистре деления. После выполнения условия $W \leq t$ подается команда на суммирование по модулю 2 полученного синдрома с позициями r в информационном регистре и ошибки оказываются исправленными. Потом подается команда на обратный сдвиг кодового слова в информационном регистре на число тактов, предшествующих выполнению условия $W \leq t$. После этого переключатель П1 ставится в положение 2 и информация из регистра поступает к пользователю.

В том случае, когда у циклического кода $t > (n / k)$, схема рис. 3.8 изменяется. Теперь необходимо определять не условие

W ≤ t, а осуществлять распознавание синдромов, соответствующих расположению одной из ошибок в крайней правой
ячейке информационного регистра.

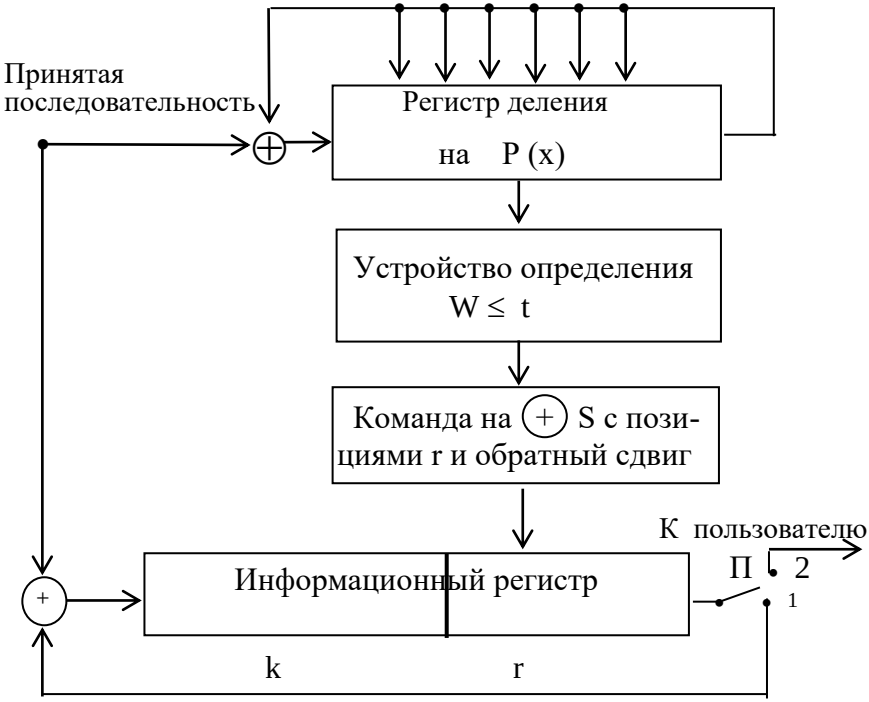


Рис. 3.8. Декодер Меггита для случая t <(n / k)

Для этих целей требуется устройство, аналогичное используемому при декодировании блоковых кодов, но значительно с меньшим объемом памяти, т. к. теперь не надо запоминать все комбинации синдромов ошибок.

Структурная схема декодера Меггита для этого случая показана на рис. 3.9.

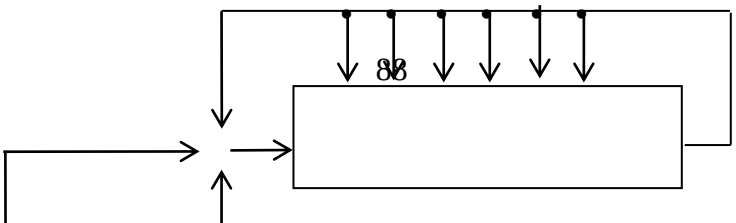


Рис. 3.9. Структурная схема декодера Меггита для случая $t > (n / k)$

Схема функционирует следующим образом. Вначале принятое кодовое слово записывается в информационный регистр и одновременно с этим в регистре деления на P(x) вычисляется начальный синдром S(x). Переключатель П1 ставится в положение 1. Затем содержимое регистра деления циклически сдвигается один или несколько раз с одновременным сдвигом комбинации в информационном регистре. Получаемые новые синдромы из регистра деления подаются на устройство распознавания. В случае соответствия синдрома положению ошибки в крайнем правом разряде информационного регистра выдается сигнал Ê (x) на ее исправление с одновременным удалением ошибки из регистра деления путем добавления Ê (x) =1 на его вход. Продолжая далее сдвиги в регистре деления и информационном регистре,

получают за счет дальнейших совпадений синдромов, соответствующих положениям ошибок в крайнем правом разряде информационного регистра, последующие их исправления и удаление из регистра деления. После исправления ошибок содержимое регистра синдрома станет нулевым. Затем осуществляется обратный сдвиг кодового слова без ошибок в информационном регистре. Декодирование законченно. Далее переключатель П1 устанавливается в положение 2 и кодовое слово направляется к пользователю.

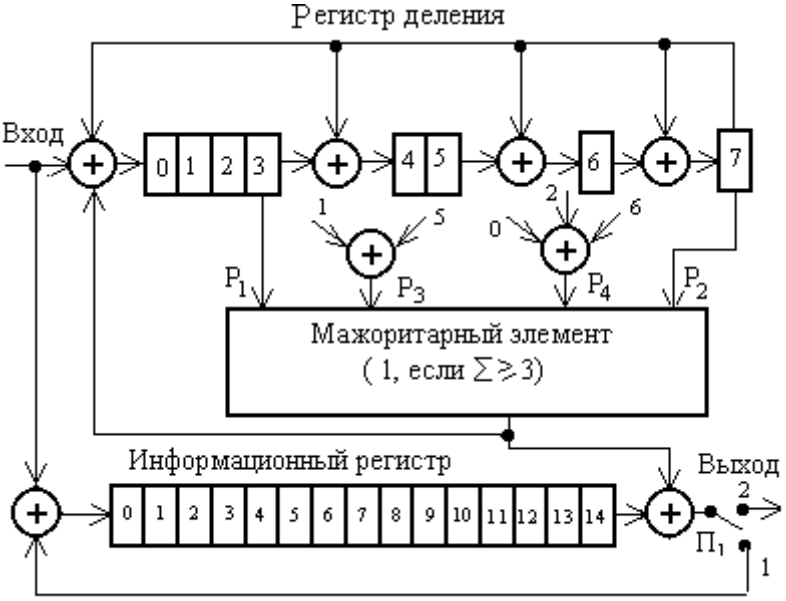
3.4.4. Пороговое декодирование циклических кодов

Пороговое декодирование циклических кодов базируется на мажоритарных правилах (выбор решений по большинству), когда «порогом» является принятое заранее условие относительно суммарного числа единиц на входах мажоритарного элемента, используемого в пороговом декодере.

Структура порогового декодера для циклического кода (15,7) с полиномом $P(x) = x^8 + x^7 + x^6 + x^4 + 1$, способного исправлять до двух ошибок (d=5), показана на рис. 3.10. Этот декодер функционирует аналогично декодеру Меггита для циклического кода (рис. 3.9), использующего при исправлении ошибок устройство (память) для распознавания синдромов, образующихся на выходах регистра деления.

В пороговом декодере вместо устройства распознавания используется мажоритарный элемент, входы которого определенным образом соеденины с регистром деления и выбран порог для исправления ошибок (1, если $\sum \geq 3$). Этот порог означает, если суммарное число единиц на входах мажоритарного элемента больше или равно трем, то на его выходе образуется сигнал 1, при не выполнении этого условия сигнал равен нулю.

Рис. 3.10. Пороговый декодер для циклического кода



После записи с входа символов кодового слова в информационный регистр в регистре деления на P(x) вычисляется первый синдром и переключатель $П_1$ ставится в положение 1. Далее содержимое регистра деления циклически сдвигается с появлением различных синдромов на выходах его ячеек и одновременно перемещается кодовое слово в информационном регистре.

В случае выполнения условий для мажоритарного элемента (1, если $\sum \geq 3$) с его выхода подается сигнал 1, корректирующий ошибку в позиции 14 информационного регистра через сумматор по модулю 2.

Коррекция ошибок происходит за один полный цикл сдвига в информационном регистре. Затем осуществляют в нем обратный сдвиг. Переключатель $П_1$ ставится в положение 2 и исправленное кодовое слово идет на выход декодера.

Определение связей входов мажоритарного элемента с ячейками регистра деления осуществляют на основе проверочной матрицы циклического кода, которую строят, используя регистр деления (рис.3.11).

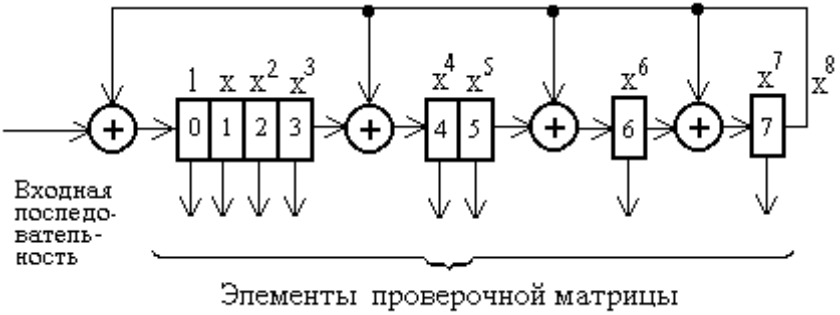


Рис. 3.11. Построение матрицы для составления проверочных уравнений

Рассмотрим основные положения по выбору связей мажоритарного элемента на примере циклического кода (15, 7), с d = 5 и образующим полиномом $P(x) = x^8 + x^7 + x^6 + x^4 + 1$. По этому полиному построим проверочную матрицу, используя регистр деления (рис. 3.11).

Для этого нужно на вход регистра (рис. 3.11), построенного по полиному кода (15,7), подать комбинацию 100000000000000 и, осуществив полный цикл сдвигов, получить проверочную матрицу (рис. 3.12).

Строки матрицы представляют собой синдромы одиночных ошибок. Эти ошибки могут иметь место в разрядах кодового слова циклического кода, направляемого при декодировании в информационный регистр декодера, а столбцы матрицы пронумерованы и соответствуют номерам ячеек регистра деления (рис. 3.10).

. Для исправления ошибок входы мажоритарного элемента (рис. 3.10) должны быть определенным образом соединены с ячейками регистра деления в соответствии с

положением единиц в строках и столбцах проверочной матрицы.

	0 1 2 3 4 5 6 7
0	1 0 0 0 0 0 0 0
1	0 1 0 0 0 0 0 0
2	0 0 1 0 0 0 0 0
3	0 0 0 1 0 0 0 0
4	0 0 0 0 1 0 0 0
5	0 0 0 0 0 1 0 0
6	0 0 0 0 0 0 1 0
7	0 0 0 0 0 0 0 1
8	1 0 0 0 1 0 1 1
9	1 1 0 0 1 1 1 0
10	0 1 1 0 0 1 1 1
11	1 0 1 1 1 0 0 0
12	0 1 0 1 1 1 0 0
13	0 0 1 0 1 1 1 0
14	0 0 0 1 0 1 1 1

Рис. 3.12. Проверочная матрица кода (15, 7)

Базовое соединение образуется по14 строке матрицы, расположение и число единиц в которой по номерам столбцов (3,5,6,7) определяет 4 входа мажоритарного элемента, связанных соответственно с базовыми ячейками регистра деления 3,5,6 и 7 (рис.3.10). Такие связи в дополнение с другими, обьединенными через сумматоры по модулю два с некоторыми базовыми, должны обеспечивать, чтобы синдромы любых одиночных ошибок (кроме 14) и любые синдромы двойных ошибок (без учета 14) не вызывали «подделки» под синдром 14 и выполнение условий для появления 1 на выходе мажоритарного элемента. Подбор таких связей может быть осуществлен программно на ЭВМ через задание вышеуказанных условий для синдромов или путем анализа выбора соединений через запись так называемых

проверочных уравнений, составляемых путем выбора определенных столбцов матрицы (рис.3.12) так, чтобы эти уравнения оказались ортогональными по последнему r_{14} элементу кодовой последовательности.

Выбрав столбцы 3, 7 и осуществив связи столбца $5 \oplus 1$ и столбцов $6 \oplus 0 \oplus 2$, получим следующие проверочные уравнения относительно входов P_1, P_2, P_3, P_4 мажоритарного элемента (рис. 3.10) по расположению единиц в столбцах 3,7 и в измененных стобцах 5 и 6:

$$\left.\begin{aligned} P_1 &= r_3 \oplus r_{11} \oplus r_{12} \oplus r_{14}, \\ P_2 &= r_7 \oplus r_8 \oplus r_{10} \oplus r_{14}, \\ P_3 &= r_1 \oplus r_5 \oplus r_{13} \oplus r_{14}, \\ P_4 &= r_0 \oplus r_2 \oplus r_6 \oplus r_{14}. \end{aligned}\right\} \quad . \qquad (3.13)$$

Так как уравнения (3.13) проверочные, то они зависят только от расположения ошибок E_i в кодововом слове, т.е.

$$\left.\begin{aligned} P_1 &= E_3 \oplus E_{11} \oplus E_{12} \oplus E_{14}, \\ P_2 &= E_7 \oplus E_8 \oplus E_{10} \oplus E_{14}, \\ P_3 &= E_1 \oplus E_5 \oplus E_{13} \oplus E_{14}, \\ P_4 &= E_0 \oplus E_2 \oplus E_6 \oplus E_{14} \end{aligned}\right\}. \qquad (3.14)$$

Каждое значение E_i входит как компонент только в одно уравнение (3.14), кроме E_{14}, входящего во все четыре уравнения. В связи с этим уравнения (3.14), отражающие такую особенность, называются ортогональными по E_{14}. Следует отметить, что одиночная ошибка, отличная от E_{14}, приводит к нарушению не более одного из уравнений (3.14), в то время как ошибка E_{14} – к нарушению всех четырех уравнений. Также любая комбинация двух ошибок, не содержащая E_{14}, приводит к нарушению не более двух уравнений, а содержащая E_{14} – не менее трех. Следовательно, используя уравнения (3.14), можно исправлять все одиночные и двойные ошибки, устанавливая соответствующие связи (рис.3.10) входов P_i мажоритарного элемента с номерами ячеек регистра деления, т.е. P_1-3, P_2-7, P_3- $5 \oplus 1$, P_4- $6 \oplus 0 \oplus 2$.

Непосредственная коррекция их в декодере осуществляется в сумматоре по модулю 2, подключенном к последней 14 ячейке информационного регистра (рис.3.10).

3.5 Лабораторное задание

1. Изучить процедуры кодирования и декодирования циклических кодов на основе регистров сдвига с обратными связями. Для этого выполнить следующие пункты:

1.1. Открыть в диске Z файл LABRNEW и далее файл LAB3Win, в котором открыть файл “Regictr”(рис. 3.13) [21,22,23].

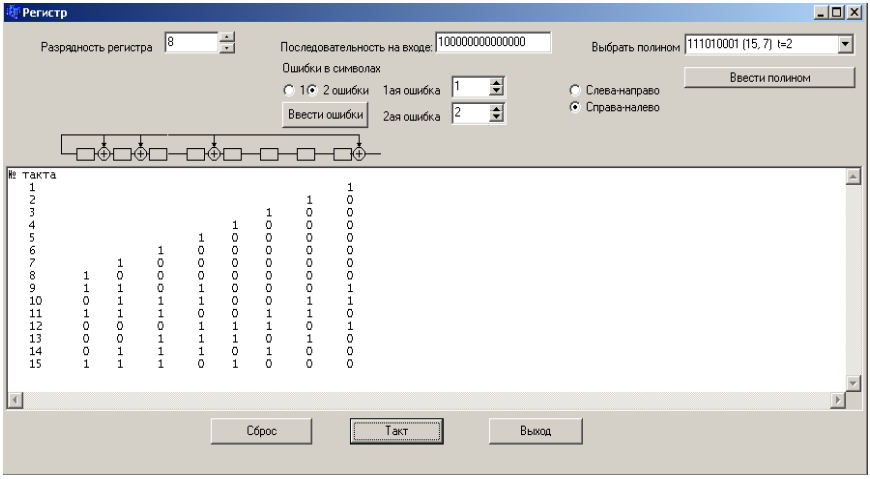


Рис. 3.13. Окно заставки для исследования свойств регистров с обратными связями на основе образующих полиномов

1.2. Открыть окно « Выбрать полином» (кнопка справа) и ввести P(x) → 1011 для циклического кода (7,4). Установить ввод справа-налево. Нажать на кнопку «Ввести полином».

1.3. Осуществить процедуру деления в регистре входных последовательностей на полином P(x) →1011 кода (7,4) с числом тактов, равным 7. Вводить в окно

«Последовательность на входе» последовательности (числа): 0001000; 0010000; 0100000; 1000000. После установки каждого числа нажимать на кнопку такт. По результатам деления четырех последовательностей - чисел (значения в регистре на 7-м такте) записать в отчет производящую матрицу для циклического кода (7,4), а затем и транспонированную матрицу (см. стр.70, 78).

1.4. Осуществить аналогичные процедуры, как и в п. 1.3, но для ввода кодовой последовательности “слева-направо”. Изменив ввод, нажать на кнопку «Ввести полином». Записать в отчет производящую и транспонированную матрицы.

1.5. Установить входную последовательность в виде 1000000 и осуществить ввод её в регистр для кода (7,4) сначала “справа-налево”, а затем “слева-направо” с числом тактов 7. Записать полученные числа на тактах 4, 5, 6, 7. Сравнить их с набором чисел (матрицами), полученными в п. 1.3 и п. 1.4. Записать в отчет предложения по построению образующих матриц циклических кодов.

1.6. Используя опыт по п.п. 1.2-1.5, построить образующую и транспонированную проверочную матрицы для циклического кода с $P(x) \rightarrow 111010001$ (15,7) $t = 2$ для ввода “справа-налево” и “слева-направо”. Последовательность на входе установить 100000000000000. Нажимать «Такт» 15 раз. Записать образующие матрицы (строки 9 - 15).

1.7. Осуществить кодирование информации на основе циклического кода $P(x) \rightarrow 1011$ (7,4) $t = 1$ с вводом последовательности “справа-налево” за 7 тактов на основе примера (3.1) из теоретической части (см. стр. 71 – 73). Приняв за информационное слово $Y_и(x) \rightarrow 1101000$ (последовательность на входе), получить проверочные символы и записать кодовое слово с проверочными символами $V(x)$. Повторить аналогичные вычисления для любых трёх информационных слов $Y_и(x)$. Записать результаты в отчет.

1.8. Подавать по очереди на вход кодовые слова V(x), полученные в п. 1.7, и для каждого из них вычислить синдром S (см. стр. 76), нажимая «Такт» 7 раз. Объяснить результат.

1.9. Осуществить вычисление синдрома для любого кодового слова из п. 1.8 после ввода ошибок в один из его символов. Сравнить полученный результат вычисления синдромов с номерами строк транспонированной проверочной матрицы из п. 1.3. Сделать выводы по свойствам синдромов.

1.10. Провести исследования, аналогичные пп. 1.7-1.9 для кода с P(x) $\rightarrow$111010001 (15,7) t = 2 для ввода “справа-налево”. Подать на вход комбинацию из 15 нулей и осуществить ввод двух ошибок. Синдром сравнить с суммой $\oplus$ строк матрицы из п. 1.6, с номерами, как и номера у ошибок (см. стр.54).

1.11. Исследовать свойство регистра Меггита (см. стр. 85), когда сдвиг синдрома в нём на такт приводит к новому синдрому, соответствующему сдвигу на такт ошибки в кодовом слове. Для этого в нулевое кодовое слово кода (7,4), t = 1, P(x) $\rightarrow$ 1011, (ввод справа-налево) ввести ошибку в четвёртый символ, т.е. 0001000 и осуществить вычисление синдрома за 7 тактов. Продолжить продвижение синдрома в регистре до 10-го такта, и записать новые синдромы. Определить их соответствие строкам ранее зафиксированной транспонированной матрицы кода (7,4) в п. 1.3, номера которых соответствуют номерам символов с ошибками в кодовом слове.

1.12. Осуществить аналогичные исследования для кода (15,7), с полиномом P(x) $\rightarrow$111010001, t = 2 при сдвиге на 21 такт и кодовым словом в окне 000000100000000. Зафиксировать результаты, начиная с 15 по 21 такт. Сделать выводы.

1.13. Проанализировать процедуру исправления ошибок декодером циклического кода на основе регистра Меггита при t <(n / k) . Для этой цели открыть файл labr3 (рис. 3.14). В открывшейся заставке выбрать код (7,4). Нажать на клавишу «Перейти к кодированию». Используя заставку (рис. 3.15),

нажать кнопку «Построить». Построить регистр для P(x) →1011, следуя указаниям (см. стр. 71-72). Ввести информационную последовательность $Y_и(x)$ →1101, соответствующую в окне десятичному числу 13 и нажать кнопку «Кодирование». Сравнить полученные результаты с данными из п. 1.7.

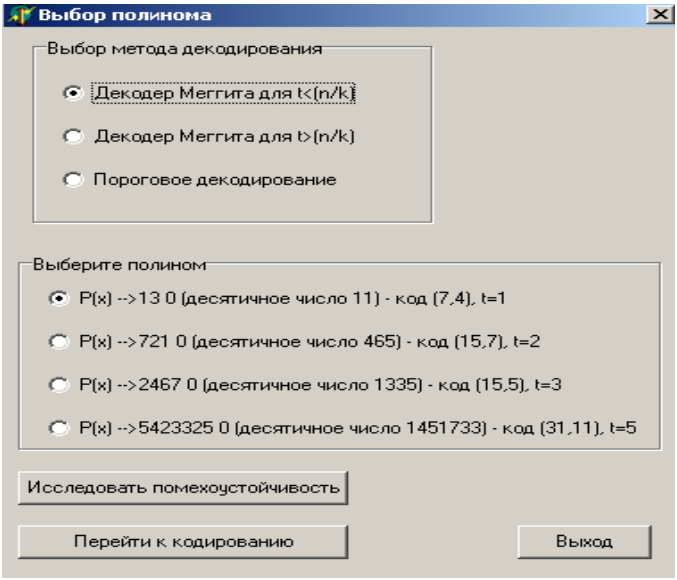


Рис. 3.14. Окно заставки для выбора процедур кодирования и декодирования циклических кодов

1.14. Далее ввести ошибку в кодовое слово (см. п. 1.9) и подтвердить его ввод.

Нажать на кнопку «Перейти к декодированию». Используя появившуюся заставку (рис. 3.16), нажать на кнопку «Начать декодирование». Зафиксировать значения синдромов и соответствующих им кодовых последовательностей в информационном регистре, нажимая на кнопку «Такт». Проследить процедуру исправления ошибок.

Нажать на кнопку «Закрыть». Осуществить аналогичные действия для остальных информационных слов из п. 1.8. Зафиксировать процедуры при исправлении ошибок. Нажать кнопку «Закрыть».

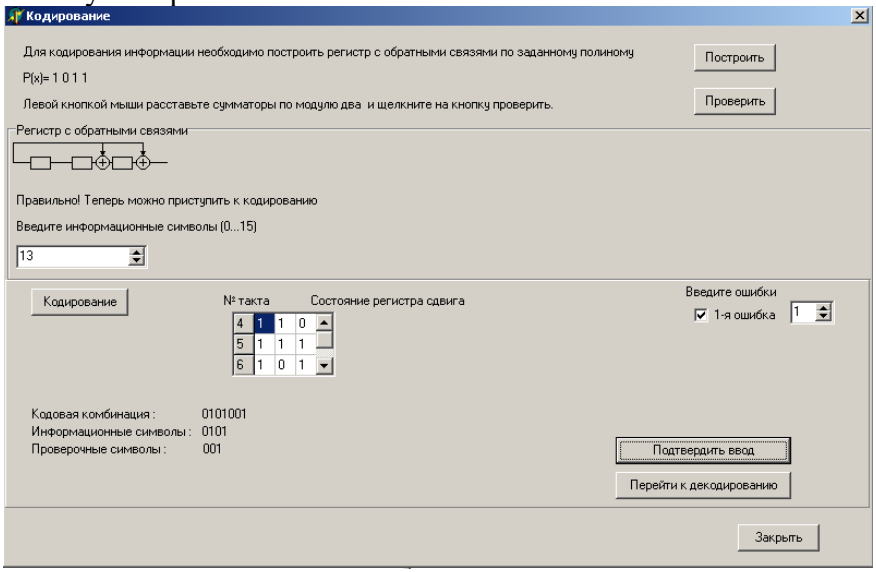


Рис. 3.15. Окно заставки для осуществления процедур кодирования и ввода ошибок в кодовые комбинации

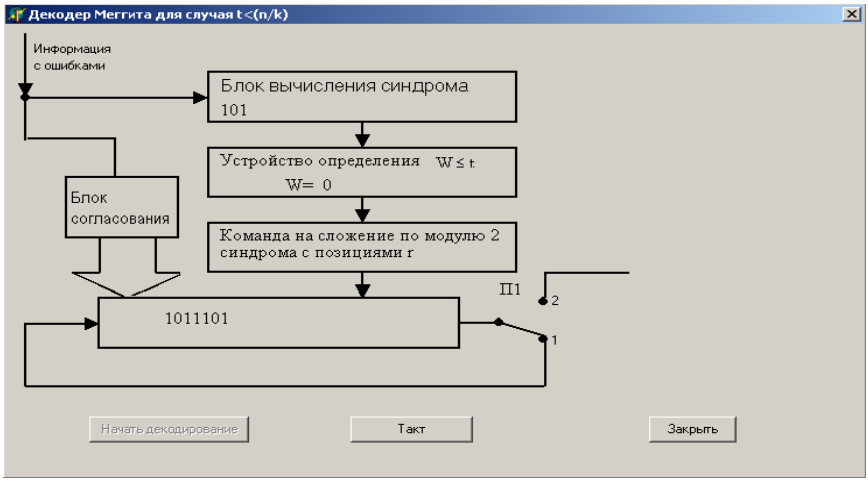


Рис. 3.16. Окно для осуществления процедур декодирования циклического корректирующего кода при t <(n / k)

1.15. Нажать на кнопку «Закрыть» в заставке «кодирование» (рис. 3.15) и перейти к полиному P(x) для кода (15,7) с t = 2 (рис. 3.14). Осуществить процедуру кодирования (рис. 3.15) и декодирования (рис. 3.16). Вводить информационные символы и ошибки произвольно. Зафиксировать результаты.

1.16. Перейти к полиному для кода (15,5) с t = 3 и осуществить аналогичные действия как в п. 1.15 для нескольких кодовых слов.

1.17. Перейти к полиному для кода (31,11) с t = 5 и осуществить аналогичные действия как в п. 1.16 для нескольких кодовых слов. Закрыть все заставки.

1.18. Определить число исправляемых искажений в кодовых комбинациях при использовании метода вылавливания ошибок. Для этой цели открыть файл «Model» (рис. 3.17).

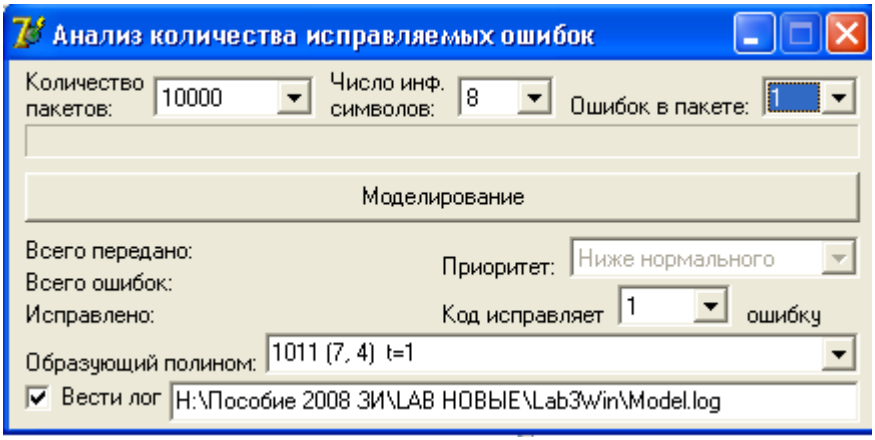


Рис. 3.17. Окно для исследования числа исправляемых ошибок в кодовых комбинациях циклического кода

1.19. Установить количество пакетов 10000. Открыть вкладку «Образующий полином» и провести исследования для всех полиномов кроме CRC-32. Для этой цели выбрать полином, установить число информационных символов, указанное в скобках кода (вторая цифра), установить число ошибок равным t, ввести число исправляемых ошибок кодом также равным t . Затем осуществить моделирование и зафиксировать полученные результаты. Объяснить в отчете, почему только код (7,4) при t = 1 и код (15,7) при t = 2 способны исправлять все ошибки, когда используется декодирование методом их вылавливания. Закрыть панель.

1.20. Осуществить процедуру исправления ошибок декодером Меггита при t > (n / k). Для этой цели открыть файл labr3 (рис. 3.14) и выбрать закладку «декодер Меггита для t > (n / k)». Выбрать полином для кода (31,16) с t =3.

1.21. Перейти к кодированию. В открывшейся заставке (рис. 3.15) построить регистр с обратными связями. Ввести информационные символы (десятичное число), совпадающие с номером студента по списку группы. Осуществить

кодирование. Ввести ошибки в символы (произвольно), соответствующие началу, середине и концу кодового слова и подтвердить ввод. Перейти к декодированию (рис. 3.18).

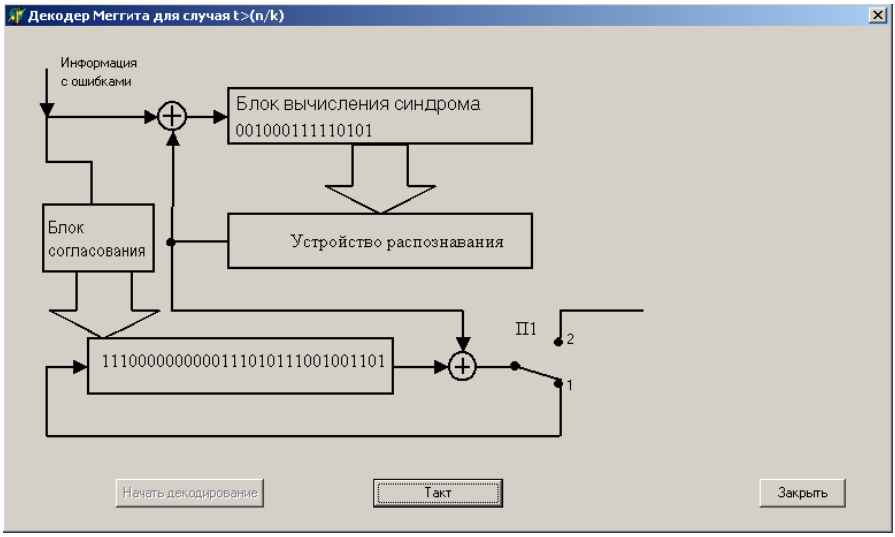


Рис.3.18. Окно для декодирования при t > (n / k)

1.22. Начать декодирование через использование кнопки «Такт». Записать значения синдромов при распознавании ошибок. Какую особенность имеют эти синдромы? Выполнить обратный сдвиг и записать результат декодирования. Зафиксировать в отчет процедуру исправления ошибок. Закрыть панель.

1.23. Открыть файл labr3, выбрать вкладку «Пороговое декодирование» и перейти к кодированию. Построить регистр с обратными связями. Ввести информационные символы в виде номера студента по списку в группе. Осуществить кодирование и ввести ошибки в символы (произвольно). Зафиксировать в отчете полученное кодовое слово. Перейти к декодированию (рис. 3.19).

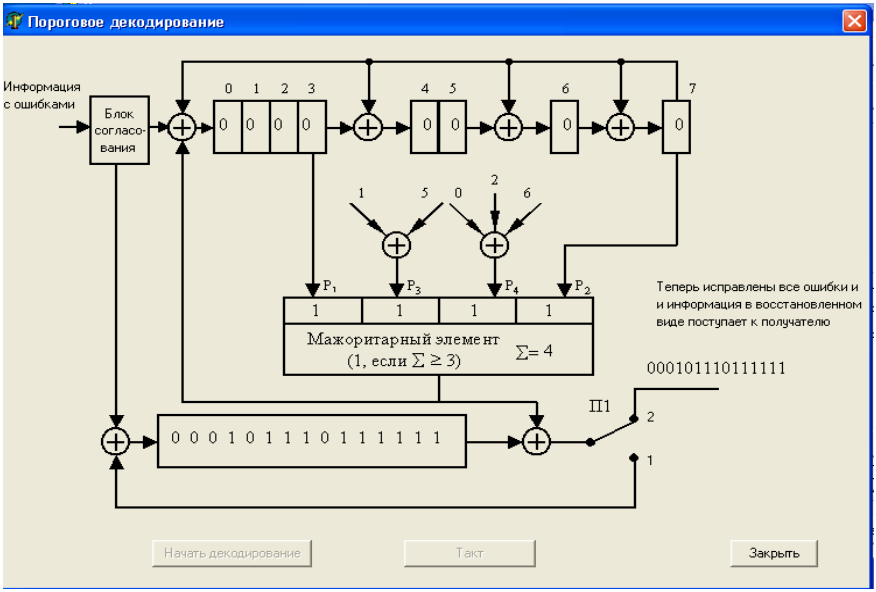


Рис. 3.19. Окно для порогового декодирования

1.24. Начать декодирование. Зафиксировать первоначальный синдром. Нажимая на кнопку «Такт», фиксировать значения синдромов и мажоритарного элемента. Осуществить обратный сдвиг, зафиксировать декодированное кодовое слово. Закрыть все вкладки.

1.25. Принимая во внимание результаты по п. 1.24, зафиксировать в отчет принципы и порядок исправления ошибок для порогового декодирования.

1.26. В открытом файле labr3 провести исследование помехоустойчивости декодеров при независимых ошибках.

Выбрать декодер с соответствующим полиномом и исследовать его помехоустойчивость при разных вероятностях независимых ошибок p(0,01- 0,1). Число кодовых комбинаций принимать для кода с n = 7→100000, для кодов с n = 15 → 50000, для кодов с n = 31→ 25000. Произвести моделирование и зафиксировать его результаты. Данные занести в таблицу. Построить три графика. Сделать выводы.

1.27. Исследовать влияние группирования ошибок на помехоустойчивость порогового декодера.

Для этой цели при вероятности P_1=0.05 изменять значение P_{11} от 0.1 до 0.8.

1.28. Исследовать влияние смешанной помехи на помехоустойчивость порогового декодера. Принять p=0.025, P_1=0.025. Вероятность следования единицы за единицей P_{11} устанавливать как в п. 1.27. Провести моделирование. Зафиксировать результаты. Сделать выводы.

Содержание отчета

1. Количественные показатели по выполнению конкретных пунктов с их комментариями и выводами.
2. Графики и таблицы по проведенным измерениям.
3. Краткие выводы по лабораторной работе.

Контрольные вопросы

1. Какими основными свойствами обладает циклический код?
2. Что такое двоичные многочлены, и какие математические операции осуществляются над ними?
3. Дайте определение неприводимого многочлена. Какие основные свойства он имеет?
4. Для чего используются неприводимые многочлены?
5. Как с помощью образующего полинома циклического кода можно построить его производящую матрицу?

6. Приведите пример деления «столбиком» кодовой комбинации на образующий полином. Для каких целей используется это деление?
7. Как технически осуществляют деление кодовой комбинации на образующий полином?
8. Назовите основные правила построения регистра с обратными связями по образующему полиному. Для каких целей используют этот регистр?
9. Как осуществляется кодирование информационного кодового слова циклическим кодом?
10. Как осуществляется процедура обнаружения ошибок циклическим кодом?
11. Как происходит декодирование циклических кодов методом «вылавливания ошибок»?
12. При каком условии возможно полное декодирование методом «вылавливания ошибок»?
13. Что такое декодер Меггита для циклического кода, и какие его основные свойства?
14. Используя структурную схему декодера Меггита для случая $t<(n/k)$, запишите последовательность процедур его функционирования. Какие основные недостатки имеет такой декодер?
15. В чем особенности построения декодера Меггита для случая $t>(n/k)$?
16. На чем основано пороговое декодирование циклического кода?
17. По структурной схеме порогового декодера объясните основные последовательности процедур, применяемые при декодировании?

4. ЛАБОРАТОРНАЯ РАБОТА №4
ДЕКОДИРОВАНИЕ КОДОВ БЧХ ПО ФОРМУЛАМ

Цель работы: изучить построение, а также кодирование и декодирование информации кодами БЧХ по формулам

Теоретическая часть

4.1. Введение в коды БЧХ

Коды БЧХ независимо открыли Хоквингем и Боуз, а также Рой-Чоудхури, в честь которых они и названы [19,24]. Эти ученые занимались разработкой теории циклических кодов, способных исправлять многократные ошибки. Задачей исследования было получение новых регулярных процедур, обеспечивающих коррекцию заранее принятого числа изменений в символах кодового слова, произошедших под воздействием помех. В процессе этой работы было показано, что ранее применявшиеся преобразования для циклических кодов (сравнение, сдвиг, деление и т.д.) недостаточны для декодирования многократных ошибок (две, три и более) без больших затрат памяти устройств сравнения синдромов, хранящихся там в виде простых наборов единиц и нулей.

Как было установлено, дальнейшее увеличение числа исправляемых ошибок необходимо искать в возможности осуществления определенных алгебраических преобразований над синдромами через решения систем уравнений, составленных на их основе. Такая процедура возможна лишь в том случае, когда синдромы будут элементами некоторого конечного поля Галуа (Э. Галуа - выдающийся математик), в котором можно осуществлять все необходимые вычисления.

Таким условиям будут удовлетворять циклические коды БЧХ, порождающие полиномы которых выбираются с учетом того, что они представляют собой наименьшее общее кратное

(произведение) так называемых минимальных многочленов, число которых выбирается равным количеству ошибок, исправляемых кодом БЧХ (см. п. 4.5).

4.2. Основные понятия полей Галуа

Прежде чем рассмотреть понятие полей Галуа, напомним понятие поля вообще. Под полем [16] понимают множество математических объектов, которые можно “складывать”, “умножать”, “вычитать”, “делить”. Образцом таких множеств являются, например, вещественные и комплексные числа. Если для множества объектов разрешено только “сложение” и “вычитание”, то оно образует не поле, а группу. Понятие кольцо – это множество математических объектов, которые можно “складывать”, “вычитать” и “умножать”. Названия этих операций взяты в кавычки потому, что, вообще-то говоря, они не являются принятыми арифметическими операциями, а лишь имеют с ними сходство. Поле Галуа в отличие от бесконечных полей вещественных, комплексных и рациональных чисел называется конечным полем с q элементами и в общем виде обозначается как GF(q). Такие поля существуют не при любом числе элементов, а только в том случае, если их количество является простым числом или результатом возведения его в степень. В первом случае поле называется простым, во втором – расширением простого поля.

Все конечные поля обладают следующими свойствами:

существуют две операции, используемые для комбинирования элементов: умножение и сложение;

результатом умножения или сложения двух элементов поля является третий элемент, лежащий в этом поле;

поле всегда содержит мультипликативную единицу 1 и аддитивную единицу 0; таким образом, что $a+0=a$ и $a \cdot 1=a$ для любого элемента a;

для любого элемента а существует обратный элемент по сложению (-а) и обратный элемент по умножению $а^{-1}$ (если $а \neq 0$) такие, что $а + (-а) = 0$ и $а \cdot а^{-1} = 1$;

выполняются правила ассоциативности [а + (в + с) = =а+ в + с, а · (в · с) = (а · в) · с], коммутативности [а + в = в + а ,
а · в = в · а] и дистрибутивности [а · (в + с) = а · в + а · с].

Простейшим из всех возможных полей Галуа является поле, состоящее из двух элементов 0 и 1. Такое поле обозначается как GF (2), где q = 2. Здесь справедливы операции сложения: 0 + 0 = 0; 0 + 1 = 1; 1 + 0 = 1; 1 + 1 = 0. Действуют также операции умножения: 0 · 0 = 0; 0 ·1 = 0; 1 · 0 = 0; 1 · 1 = 1.

Более сложные поля Галуа получаются, если q является результатом возведения в степень простого числа, например, $q = b^m$. В этом случае элементами поля являются все многочлены (полиномы) степени m-1 или менее, коэффициенты которых лежат в простом поле GF (b). Так, если b=2, а m=3, то получаем поле $GF(2^3)$= GF (8), где многочлены имеют степени 2;1; 0 и их коэффициенты принадлежат простому полю GF (2). Такими многочленами тут являются записи чисел от 1 до 7 в двоичном виде или через представление с переменной х, а также нулевой многочлен (000), обозначаемый в пособии буквой н (нуль), что связано с особеностями ввода формул в текст и применения процедур моделирования на ЭВМ.

Правила умножения и сложения для многочленов получаются из общепринятых с последующим приведением результата по модулю некоторого специального многочлена P(x) степени m. Так, например, для поля $GF(2^3)$

$$P(x) = x^3 + x + 1.$$

Этот многочлен и аналогичные ему обладают тем свойством, что их нельзя представить в виде произведения многочленов низших степеней с коэффициентами из GF (2), поэтому они называются неприводимыми. Приведем [1]

некоторые неприводимые многочлены до восьмой степени включительно. Здесь слева приводится многочлен в алгебраической форме, а справа - в десятичном виде, т. е.

$$x^2 + x + 1 \rightarrow 7, \quad (4.1)$$

$$x^3 + x + 1 \rightarrow 11, \quad (4.2)$$

$$x^4 + x + 1 \rightarrow 19, \quad (4.3)$$

$$x^5 + x^2 + 1 \rightarrow 37, \quad (4.4)$$

$$x^6 + x + 1 \rightarrow 67, \quad (4.5)$$

$$x^7 + x^3 + 1 \rightarrow 137, \quad (4.6)$$

$$x^8 + x^4 + x^3 + x^2 + 1 \rightarrow 285. \quad (4.7)$$

По многочленам (4.1) – (4.7) можно построить поля Галуа различной размерности. Так, например, по многочлену (4.2) $P(x) = x^3 + x + 1 \rightarrow 1011$ можно построить поле Галуа $GF(2^3)$, которое является расширением подполя GF(2). Кроме элементов подполя GF(2), в поле расширения существуют однозначные элементы, которые представляются новым символом α. Каждый ненулевой элемент в $GF(2^3)$ можно представить как степень α. Множество элементов поля $GF(2^3)$ образуется из стартового множества {н, 1, α} и дополняется другими элементами путем последовательного умножения последней записи на α [1], т. е.

$GF(2^3) = \{н, 1, \alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$.

В двоичной записи многочленов стартовое множество элементов поля $GF(2^3)$ отображается: н $\rightarrow$ 000, $1 \rightarrow \alpha^0 \rightarrow$ 001, $\alpha^1 \rightarrow$ 010. Элемент α^2 образуется в результате умножения α^1 на α^1 или в двоичной записи $010 \cdot 010 = 100$, что эквивалентно сдвигу единицы в элементе $\alpha^1 \rightarrow$ 010 влево на один разряд, т.е. $\alpha^2 \rightarrow 100$. Элемент поля $\alpha^3 = \alpha^2 \cdot \alpha^1$ или $100 \cdot 010 = 1000$. Далее следует осуществить приведение результата умножения по модулю $P(x) \rightarrow 1011$. Приведение многочлена 1000 по модулю P(x) эквивалентно делению его на P(x) и взятию остатка.

Приведение по модулю $P(x) \rightarrow 1011$ осуществляем делением 1000 на 1011 с фиксацией остатка, т.е.

$$\begin{array}{l} 1000 \mid \underline{1011} \\ \underline{1011} \\ \quad 011 \rightarrow \text{остаток.} \end{array}$$

Таким образом, элемент поля α^3 соответствует многочлену 011. Следующий элемент поля $\alpha^4 = \alpha^3 \cdot \alpha^1$ реализуется в двоичном представлении как сдвиг влево единиц элемента 011, т.е. 110. Получение элемента $\alpha^5 = \alpha^4 \cdot \alpha^1$ или в двоичной записи $110 \cdot 010 = 1100$ требует вычисления остатка через деление 1100 на $P(x) \rightarrow 1011$, в результате которого будет иметь

$$\begin{array}{l} 1100 \mid \underline{1011} \\ \underline{1011} \\ \quad 111 \rightarrow \text{остаток.} \end{array}$$

В итоге $\alpha^5 \rightarrow 111$. Аналогично для $\alpha^6 = \alpha^5 \cdot \alpha^1$ или $111 \cdot 010 = 1110$ находим остаток, т.е.

$$\begin{array}{l} 1110 \mid \underline{1011} \\ \underline{1011} \\ \quad 101 \rightarrow \text{остаток} = \alpha^6. \end{array}$$

Результат получения поля $GF(2^3)$ можно отразить следующими последовательностями:

$$\begin{array}{ll} & н \rightarrow 000, \\ 0. & \alpha^0 = 1 \rightarrow 001, \\ 1. & \alpha^1 \rightarrow 010, \\ 2. & \alpha^2 \rightarrow 100, \\ 3. & \alpha^3 \rightarrow 011, \\ 4. & \alpha^4 \rightarrow 110, \\ 5. & \alpha^5 \rightarrow 111, \\ 6. & \alpha^6 \rightarrow 101. \end{array} \qquad (4.8)$$

При таком же вычислении элемента α^7, получаем, что $\alpha^7 = \alpha^0 \rightarrow$ $\rightarrow 001$ и это свидетельствует о замыкании множества элементов поля по отношению к операции умножения [1].

Аналогично можно построить поле Галуа $GF(2^4)$ на основе примитивного многочлена $P(x)= x^4+ x + 1 \rightarrow 10011$. Результат получения поля $GF(2^4)$ отражается следующими последовательностями:

$$
\begin{array}{ll}
\text{н} \rightarrow 0000, & \\
0.\ \alpha^0 \rightarrow 0001, & 1.\ \alpha^1 \rightarrow 0010, \\
2.\ \alpha^2 \rightarrow 0100, & 3.\ \alpha^3 \rightarrow 1000, \\
4.\ \alpha^4 \rightarrow 0011, & 5.\ \alpha^5 \rightarrow 0110, \\
6.\ \alpha^6 \rightarrow 1100, & 7.\ \alpha^7 \rightarrow 1011, \\
8.\ \alpha^8 \rightarrow 0101, & 9.\ \alpha^9 \rightarrow 1010, \\
10.\ \alpha^{10} \rightarrow 0111, & 11.\ \alpha^{11} \rightarrow 1110, \\
12.\ \alpha^{12} \rightarrow 1111, & 13.\ \alpha^{13} \rightarrow 1101, \\
14.\ \alpha^{14} \rightarrow 1001. &
\end{array}
\qquad (4.9)
$$

Как и для поля $GF(2^3)$ получается, что $\alpha^{15} = \alpha^0 \rightarrow 0001$. В степенной форме представления поля индекс степени элемента совпадает с его номером. Элемент поля, состоящий из всех нулей, обозначается как «н» (нуль), например, $0000 \rightarrow$ н.

Операция умножения элементов поля выполняется обычным методом, с приведением результата по модулю P(x), если она осуществляется с двоичными многочленами. В случае перемножения элементов поля из (4.8) как $\alpha^4 \cdot \alpha^5$ процедура будет иметь вид, с учетом того, что $\alpha^7 = \alpha^0$

$$\alpha^4 \cdot \alpha^5 = \alpha^9 = \alpha^7 \cdot \alpha^2 = \alpha^0 \cdot \alpha^2 = \alpha^2. \qquad (4.10)$$

Деление элементов поля друг на друга осуществляется по выражению

$$A/B = A \cdot B^{-1}. \qquad (4.11)$$

Вычисление обратного элемента поля может быть получено, например, следующим образом: $(\alpha^4)^{-1} = \alpha^0 \cdot \alpha^{-4} = \alpha^7 \cdot \alpha^{-4} = \alpha^{7-4} = \alpha^3$.

Деление двух элементов поля α^4 на α^6 осуществляется в виде

$$\frac{\alpha^4}{\alpha^6} = \alpha^4 \cdot \alpha^0 \cdot \alpha^{-6} = \alpha^4 \cdot \alpha^7 \cdot \alpha^{-6} = \alpha^4 \cdot \alpha^{7-6} = \alpha^4 \cdot \alpha^1 = \alpha^5 .$$

Таким образом, деление находится через умножение с обратным элементом и приведением результата по модулю P(x). Например, осуществим деление $\alpha^4 = 110$ на $\alpha^2 = 100$. Обратный элемент для α^2 получим как

$$\alpha^{-2} = \alpha^0 \cdot \alpha^{-2} = \alpha^7 \cdot \alpha^{-2} = \alpha^{7-2} = \alpha^5 \rightarrow 111.$$

Теперь произведем умножение столбиком с дальнейшим приведением результата по $P(x) \rightarrow 1011$, т.е.

$$\begin{array}{rl} & 110 \rightarrow \alpha^4 \\ \times & \underline{111} \rightarrow \alpha^5 \\ & 110 \\ & 110 \\ & \underline{110} \\ & 10010 \;\big|\; \underline{1011} \rightarrow P(x) \\ & \underline{1011} \\ & 100 \rightarrow \alpha^2 . \end{array}$$

Таким образом, получаем, что $\alpha^4 / \alpha^2 = \alpha^2$.

Деление α^2 на α^4 дает следующий результат после вычисления α^{-4}, т.е.

$$\alpha^{-4} = \alpha^7 \; \alpha^{-4} = \alpha^3 \rightarrow 011,$$

$$\begin{array}{rl} & 100 \rightarrow \alpha^2 \\ \times & \underline{011} \rightarrow \alpha^3 \\ & 100 \\ & \underline{100} \\ & 1100 \;\big|\; \underline{1011} \rightarrow P(x) \\ & \underline{1011} \\ & 111 \rightarrow \alpha^5 \text{ и } \alpha^2 / \alpha^4 = \alpha^5 . \end{array}$$

Сложение элементов поля Галуа осуществляется по следующей процедуре.

Так, если необходимо сложить α^5 и α^6, то следует перейти к их представлению в виде наборов элементов длиной m, т.е. $\alpha^5 = 111$ и $\alpha^6 = 101$, а затем осуществить сложение по модулю 2, т. е. $111 \oplus 101 = 010$. Далее, необходимо перейти к обозначениям элементов поля через α^i, используя полученное значение 010, т.е. $010 = \alpha^1$. В результате имеем $\alpha^5 \oplus \alpha^6 = \alpha^1$.

4.3. Понятие корней многочленов

Для декодирования кодов БЧХ, представленных в виде многочленов, необходимо вычислять их корни. Для понимания сути корней приведем пример определения их у многочлена $f(x) = x^3 + x + 1$, который неприводим над полем GF(2). Корнем этого многочлена будет элемент поля Галуа $GF(2^3)$ α, т.к. после его подстановки в f (x) вместо x, f (α) становится равным нулевому элементу поля 000→ н, т.е.

$$f(\alpha) = \alpha^3 + \alpha + 1 \rightarrow 011 + 010 + 001 = 000 .$$

Кроме α, многочлен $f(x) = x^3 + x + 1$ будет иметь еще некоторые корни. Дело в том, что полином f (x) с элементами x из некоторого конечного поля называется неприводимым, если его нельзя разложить на множители, используя лишь элементы этого поля. Однако этот многочлен всегда можно разложить на множители, используя элементы из некоторого расширения поля. Так, если f (x) - неприводимый многочлен с коэффициентами из GF(b) и α - его корень, то α^b, α^{b^2}, α^{b^3} … также будут его корнями [15]. В данном примере при b=2 для f (x) и α^2 и α^4 также будут являться его корнями, как и α, что легко проверяется их подстановкой в f (x). В самом деле:

$$f(\alpha^2) = 1 + \alpha^2 + \alpha^6 \rightarrow 001 + 100 + 101 = 000,$$

$$f(\alpha^4) = 1 + \alpha^4 + \alpha^{12} = 1 + \alpha^4 + \alpha^5 \rightarrow 001 + 110 + 111 = 000.$$

Аналогично тому, как каждый многочлен с вещественными коэффициентами можно разложить на множители, допустив введение комплексных чисел, каждый полином с коэффициентами из конечного поля можно представить в виде составляющих с помощью элементов из

некоторого расширения этого поля. Связь между сомножителями и корнями многочленов с коэффициентами из конечного поля полностью аналогична связи между корнями и сомножителями многочленов с вещественными коэффициентами. Например, некоторый многочлен может быть представлен в виде

$$f(x) = (x - \beta_1)(x - \beta_2) \ldots (x - \beta_j),$$

где j значений $\beta_1, \ldots, \beta_j$ являются корнями многочлена $f(x)$. Знак минус здесь поставлен по аналогии с алгеброй. При действиях с многочленами в поле Галуа сложение и вычитание эквивалентно. Так, если α, α^2 и α^4 являются корнями многочлена $f(x) = x^3 + x + 1$,то его можно записать, используя значение элементов поля из (4.8) в виде

$$f(x)=(x+\alpha)(x+\alpha^2)(x+\alpha^4)=x^3+x^2(\alpha+\alpha^2+\alpha^4)+x(\alpha^3+\alpha^6+\alpha^5)+\alpha^7=x^3+x+1.$$

4.4. Формирование проверочной матрицы кодов БЧХ

Проверочная матрица кодов БЧХ была получена на основе расширения свойств кодов Хемминга, исправляющих одиночные ошибки [5].

Рассмотрим вначале принципы построения кода БЧХ, увеличивающие возможность кода Хемминга до коррекции двух ошибок вместо одной.

Возьмем в качестве примера код Хемминга (15,11) с числом проверочных символов $r = 4$ и $d = 3$. Такой код имеет проверочную матрицу H, которую можно построить по рекомендациям из п. 2.3 как

$$H = \begin{bmatrix} 0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,0\,0\,0 \\ 0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,0\,1\,0\,0 \\ 1\,0\,1\,1\,0\,1\,0\,0\,1\,1\,0\,0\,1\,0 \\ 1\,1\,0\,1\,1\,1\,0\,1\,0\,1\,0\,0\,0\,1 \end{bmatrix}. \qquad (4.12)$$

Перестроим матрицу (4.12) таким образом, чтобы рассматриваемый код длины $n = 2^m - 1 = 15$ при $m = r = 4$ имел матрицу, где ее столбцы соответствовали порядку

расположения элементов, образующих поле Галуа $GF(2^4)$ из (4.9), т.е.

$$H = \begin{bmatrix} 0\,0\,0\,1\,0\,0\,1\,1\,0\,1\,0\,1\,1\,1\,1 \\ 0\,0\,1\,0\,0\,1\,1\,0\,1\,0\,1\,1\,1\,1\,0 \\ 0\,1\,0\,0\,1\,1\,0\,1\,0\,1\,1\,1\,1\,0\,0 \\ 1\,0\,0\,0\,1\,0\,0\,1\,1\,0\,1\,0\,1\,1\,1 \end{bmatrix}$$

или иначе

$$H = \left[\alpha^0\alpha^1\alpha^2\alpha^3\alpha^4\alpha^5\alpha^6\alpha^7\alpha^8\alpha^9\alpha^{10}\alpha^{11}\alpha^{12}\alpha^{13}\ \alpha^{14}\right]. \qquad (4.13)$$

Мы собираемся добавить к H (4.13) еще дополнительную строку для возможности коррекции двух ошибок. Размерность этой строки должна полностью совпадать с предыдущей строкой. Поэтому будем присоединять к матрице (4.13) эту строку таким способом, когда для удобства дальнейшего ее формирования элементы поля условно обозначим через индексы степеней α, значения которых при этом совпадут с номерами символов кодового слова. Через эти же индексы возможно также указание на расположение ошибок в кодовом слове. В результате получим, что

$$H^1 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & \ldots\ldots 14 \\ f(0) & f(1) & f(2) & f(3) & f(4) & \ldots\ldots f(14) \end{bmatrix}. \qquad (4.14)$$

При таком обозначении значениями функции f(i) второй строки также будут являться элементы поля $GF(2^4)$. Поэтому i-й столбец матрицы H^1 (4.14) можно представить как

$$H^1_i = \begin{pmatrix} i \\ f(i) \end{pmatrix},$$

который представляет собой столбец из двух элементов поля с общим размером 8.

Предположим теперь, что произошло две ошибки на позициях i и j. Тогда по свойствам синдрома его значение

$$\mathbf{S} = \mathbf{H}_i + \mathbf{H}_j = \begin{pmatrix} \mathbf{i} + \mathbf{j} \\ \mathbf{f(i)} + \mathbf{f(j)} \end{pmatrix} = \begin{pmatrix} \mathbf{S_1} \\ \mathbf{S_3} \end{pmatrix},$$

где S_1 – первая часть синдрома S, а S_3 – его вторая часть, а выбор именно такой их нумерации будет ясен позже. Далее необходимо выбрать f (i) и f (j) так, чтобы декодер по S_1 и S_3 нашел элементы поля i и j и по ним мог указать на позиции ошибок в кодовом слове, т.е. чтобы он осуществил решение уравнения

$$\begin{cases} \mathbf{i} + \mathbf{j} = \mathbf{S_1} \\ \mathbf{f(i)} + \mathbf{f(j)} = \mathbf{S_3} \end{cases} \qquad (4.15)$$

относительно i и j при заданных синдромах S_1 и S_3.

Примером плохого выбора является функция f(i) = c · i, где c - постоянная величина, т.к. в этом случае система (4.15) выглядит как

$$\begin{cases} \mathbf{i} + \mathbf{j} = \mathbf{S_1} \\ \mathbf{c(i + j)} = \mathbf{S_3} \end{cases}$$

и поэтому не может быть решена с позиций математики [16].

Другим примером такого же плохого выбора является функция $f(i) = i^2$, т. к.

$$i^2 + j^2 = (i + j)^2 = i^2 + 2ij + j^2 = i^2 + ij + ij + j^2 = i^2 + 0 + j^2 = i^2 + j^2,$$

где действия осуществляются в поле $GF(2^4)$ и система (4.15) приобретает вид

$$\begin{cases} \mathbf{i} + \mathbf{j} = \mathbf{S_1} \\ \mathbf{(i + j)^2} = \mathbf{S_3} \end{cases}$$

и также не имеет решения, т. к. второе уравнение получается как квадрат первого.

Выбор f(i) = i^3 приводит к хорошему результату, так как теперь система (4.15) имеет вид

$$\begin{cases} \mathbf{i} + \mathbf{j} = \mathbf{S_1} \\ \mathbf{i^3} + \mathbf{j^3} = \mathbf{S_3} \end{cases} \qquad (4.16)$$

и поэтому может быть решена (см. п. 4.6), т.к. уравнения оказываются независимыми.

Таким образом, проверочная матрица (4.14), расширяющая возможности кода Хемминга, может быть записана с учетом выбора f (i) = i^3. После удаления введенных условных обозначений в (4.14) она представляется в виде

$$H^{\Gamma} = \begin{bmatrix} 1\alpha^1\alpha^2\alpha^3\alpha^4\alpha^5\alpha^6\alpha^7\alpha^8\alpha^9\alpha^{10}\alpha^{11}\alpha^{12}\alpha^{13}\alpha^{14} \\ 1\alpha^3\alpha^6\alpha^9\alpha^{12}1\alpha^3\alpha^6\alpha^9\alpha^{12}\,1\;\alpha^3\;\alpha^6\;\alpha^9\;\alpha^{12} \end{bmatrix}. \qquad (4.17)$$

Здесь предполагается, что элементы второй строки получены в результате возведения в третью степень элементов первой строки с использованием основных правил для поля $GF(2^3)$ из п. 4.2. Для произвольной длины n кода БЧХ матрица (4.17) записывается как

$$H^{\Gamma}_{n,k} = \begin{bmatrix} 1\alpha^1\alpha^2\alpha^3....\alpha^{n-1} \\ 1\alpha^3\alpha^6\alpha^9....\alpha^{3(n-1)} \end{bmatrix}. \qquad (4.18)$$

Анализ, проведенный в [5], позволил определить правила построения проверочных матриц для кодов БЧХ, корректирующих t ошибок.

Количество строк проверочной матрицы (4.18) должно быть равно числу исправляемых ошибок кодом БЧХ, а сама строка строится по форме в виде

$$1\alpha^{2t-1}\alpha^{2(2t-1)}\alpha^{3(2t-1)}...\alpha^{(n-1)(2t-1)}. \qquad (4.19)$$

Для кода с t =1 имеем одну строку, для t =2 – две строки, для t =3 – три строки и т.д., построенных согласно (4.19).

В соответствии с (4.19) следует, что для получения второй строки кода с t =2 элементы первой строки проверочной матрицы кода Хемминга (4.13) возводятся в третью степень. Для формирования третьей строки кода БЧХ с t =3 элементы первой строки должны возводиться в пятую степень, а для получения четвертой строки при t = 4 – в седьмую степень и т.д., где необходимое значение степени определяется как d-2.

Таким образом, синдром, вычисленный по проверочной матрице кода БЧХ, разделяется на компоненты S_1, S_3, S_5, S_7 и т.д. в зависимости от числа исправляемых ошибок t. Значения компонет синдрома выражаются через элементы поля Галуа, например для удобства, при моделировании процедур декодирования кодов БЧХ, в виде индексов степеней α. Кроме нечетных номеров синдромов в процедуре декодирования используются также четные номера, т.е. S_2, S_4, S_6, S_8 и т.д. Они определяются по нечетным номерам синдромов как $S_2 = S_1^2$, $S_4 = S_1^4$, $S_6 = S_3^2$ и т.д.

4.5. Определение кодов БЧХ через многочлены

Кроме проверочной матрицы кода БЧХ, рассмотренной в разделе 4, для полной процедуры кодирования и декодирования требуется наличие образующего полинома, выраженного в виде многочлена. Образующий полином кода БЧХ определяется по длине кода и заданному кодовому расстоянию. Длина кода БЧХ, как было выяснено в п. 4.4, определяется из выражения $n = 2^m - 1$, аналогичного и для кодов Хемминга, где m – любое целое число, начиная с 3, а количество проверочных разрядов кода r задается выражением

$$r = m\,t = m\,(d - 1) / 2. \qquad (4.20)$$

Это следует из процедуры построения проверочной матрицы кода БЧХ. Здесь за основу берется число проверочных разрядов для кода Хемминга, определяемое величиной m, и далее оно кратно увеличивается на величину t, значение которой определяется из формулы $d = 2t + 1$. Отсюда следует, что коды БЧХ строятся по заданным n и t. Значение числа информационных символов k неизвестно, пока не найден образующий полином кода БЧХ P(x).

Построение образующего полинома кода БЧХ напрямую связано со структурой его проверочной матрицы. Для кодов БЧХ он является наименьшим общим кратным (НОК) так называемых минимальных полиномов m_i (x), порядок которых равен d – 2, а i = 1, 3, 5 …, т.е.

$$P(x) = НОК\{m_1(x), m_3(x)...m_{d-2}(x)\} \qquad (4.21)$$

Число минимальных полиномов равно числу исправляемых ошибок t кодом БЧХ. Под порядком d-2 минимального полинома понимается соответствующая степень элемента поля Галуа, в которую необходимо возвести первую строку H (4.13) для получения ее последней строки с обеспечением требуемого d. Так для формирования кода БЧХ, исправляющего две ошибки (d = 5), это 3 ($\alpha^{3)}$, а для трех ошибок (d = 7) это 5 (α^5) и т.д.

Минимальные многочлены m_i (x) - это те неприводимые многочлены (п. 3.2) , корнями которых являются элементы α, α^3, α^5, α^7 и т.д. до степени d-2. Подстановка корня в минимальный многочлен вместо x обращает его в нуль.

В таблице представлены минимальные многочлены для поля GF (2^4), которые расставлены в ней в соответствии с неприводимыми многочленами, взятыми из п. 3.1.

Порождающий многочлен P(x) для кода БЧХ длины n=15, исправляющего две ошибки, получается по (4.21) исходя из вышеизложенного следующим образом:

$$P(x) = НОК\{m_1(x), m_3(x)\} =$$
$$= НОК\{x^4 + x + 1, x^4 + x^3 + x^2 + x + 1\}$$
$$= (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1) = \quad (4.22)$$
$$= x^8 + x^7 + x^6 + x^4 + 1.$$

Поскольку в (4.22) степень P(x) равна 8, то n-k=8. Отсюда получаем, что k=7, и мы определили все необходимые параметры кода БЧХ (15,7)

Минимальные многочлены поля $GF(2^4)$

$GF(2^4)$, в степенной форме	Элементы поля в двоичном виде	Минимальные многочлены $m_i(x)$
α^0	0 0 0 1	$x + 1$
α^1	0 0 1 0	$x^4 + x + 1$
α^2	0 1 0 0	$x^4 + x + 1$
α^3	1 0 0 0	$x^4 + x^3 + x^2 + x + 1$
α^4	0 0 1 1	$x^4 + x + 1$
α^5	0 1 1 0	$x^2 + x + 1$
α^6	1 1 0 0	$x^4 + x^3 + x^2 + x + 1$
α^7	1 0 1 1	$x^4 + x^3 + 1$
α^8	0 1 0 1	$x^4 + x + 1$
α^9	1 0 1 0	$x^4 + x^3 + x^2 + x + 1$
α^{10}	0 1 1 1	$x^2 + x + 1$
α^{11}	1 1 1 0	$x^4 + x^3 + 1$
α^{12}	1 1 1 1	$x^4 + x^3 + x^2 + x + 1$
α^{13}	1 1 0 1	$x^4 + x^3 + 1$
α^{14}	1 0 0 1	$x^4 + x^3 + 1$

Таким же способом строится порождающий многочлен и для другого кода БЧХ длины 15 с t = 3:

$$
\begin{aligned}
&P(x) = НОК\{m_1(x),\ m_3(x),\ m_5(x)\} = \\
&= НОК\{(x^4 + x + 1), (x^4 + x^3 + x^2 + x + 1), (x^2 + x + 1)\} = \\
&= (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1) = \\
&= x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1.
\end{aligned}
\qquad (4.23)
$$

В результате процедуры (4.23) получился порождающий многочлен для кода БЧХ (15,5), исправляющего три ошибки. По порождающим многочленам P(x) можно построить образующие матрицы кодов БЧХ, используя, например, процедуру сдвигов, изложенную в п. 3.2.

Так для кода БЧХ (15,5) удается сформировать образующую матрицу, если использовать представление P (x) $\rightarrow$10100110111, т.е.

$$
G_{15,5} = \begin{array}{c}
\begin{matrix} 14 & 13 & 12 & 11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{matrix} \\
\left[\begin{matrix}
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1
\end{matrix}\right]
\begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5. \end{matrix}
\end{array}
\qquad (4.24)
$$

В верхней части (4.24) обозначены номера столбцов образующей матрицы $G_{15,5}$. Матрица (4.24) с приведенной нумерацией столбцов может применяться для процедуры кодирования кодов БЧХ, которая аналогична и для циклических кодов.

4.6. Решение систем уравнений в поле Галуа

Вернемся теперь к решению системы из двух уравнений (4.16), составленных в предположении, что при этом используется проверочная матрица (4.17) [12].

Система уравнений имеет вид

$$\begin{cases} i + j = S_1 \\ i^3 + j^3 = S_3 \end{cases}$$

и по известным S_1 и S_3 необходимо найти элементы поля i и j и по их степеням определить положение ошибок в кодовом слове.

Для этой системы имеем

$$S^3 = i^3 + j^3 = (i + j)(i^2 + ij + j^2) = (i + j)(i^2 + j^2 + ij) =$$

$$= (i + j)[(i + j)^2 + ij] = S_1 (S_1^2 + ij),$$

откуда

$$ij = \frac{S_3}{S_1} + S_1^2 . \qquad (4.25)$$

Из (4.25) и зависимости $i + j = S_1$ можно установить, что i и j являются корнями квадратного уравнения. Это следует из элементарных математических зависимостей, при которых их сумма равна коэффициенту при неизвестном в первой степени, а произведение корней равно свободному члену, т.е.

$$x^2 + S_1 x + \left(\frac{S_3}{S_1} + S_1^2\right) = 0. \qquad (4.26)$$

Уравнение (4.26) можно переписать в общем виде

$$x^2 + \sigma_{21} x + \sigma_{22} = 0, \qquad (4.27)$$

где

$$\sigma_{21} = S_1, \sigma_{22} = \frac{S_3}{S_1} + S_1^2 . \qquad (4.28)$$

Зная величины σ_{21} и σ_{22}, можно найти корни уравнения (4.27) в виде элементов поля, и их порядковый номер будет определять положение ошибок в кодовом слове. Определение корней уравнения (4.27) может осуществляться одной из самых простых процедур - последовательной подстановкой в него элементов поля Галуа (процедура Ченя) [5]. Корнями уравнения будут те элементы поля Галуа, которые после их подстановки в него дают в левой части нулевой результат (н).

Решение двух уравнений (4.16) позволяет определять положение двух произвольных ошибок в кодовом слове. Для случая трех ошибок можно составить в соответствии с общими правилами построения проверочной матрицы для кодов БЧХ три уравнения

$$\begin{cases} i + j + \ell = S_1 \\ i^3 + j^3 + \ell^3 = S_3 \\ i^5 + j^5 + \ell^5 = S_5 \end{cases}. \qquad (4.29)$$

Из (4.29) следует предположить, что i, j, ℓ являются корнями кубического уравнения

$$x^3 + \sigma_{31} x^2 + \sigma_{32} x + \sigma_{33} = н. \qquad (4.30)$$

Коэффициенты, входящие в (4.30), могут быть вычислены, например, обычными методами подстановки, как в [20], выполняемыми для системы уравнений (4.29). Не задерживаясь на громоздких преобразованиях, приведем конечный результат

$$\sigma_{31} = S_1, \sigma_{32} = \frac{S_1^2 \cdot S_3 + S_5}{S_1^3 + S_3}, \sigma_{33} = \frac{S_2^3 + S_3^2 + S_1 \cdot \left(S_1^2 \cdot S_3 + S_5\right)}{S_1^3 + S_3}. \qquad (4.31)$$

Приведем здесь же результат для случая четырех ошибок в кодовом слове, отражающий решение уравнения четвертой степени

$$x^4 + \sigma_{41} \cdot x^3 + \sigma_{42} \cdot x^2 + \sigma_{43} \cdot x + \sigma_{44} = н.$$

(4.32)

Для уравнения (4.32) получаются следующие значения коэффициентов:

$$\sigma_{41} = S_1, \sigma_{42} = [S_1 \cdot (S_1^7 + S_7) + S_3 \cdot (S_1^5 + S_5)]/A;$$
$$\sigma_{43} = [S_1 \cdot (S_1^3 \cdot S_5 + S_1 \cdot S_7) + S_3 \cdot (S_1^6 + S_3^2)]/A; \qquad (4.33)$$
$$\sigma_{44} = [S_1^3 \cdot (S_1^7 + S_7) + S_3 \cdot (S_1^7 + S_1 \cdot S_3^2 + S_7) +$$
$$+ S_5 \cdot (S_1^5 + S_1^2 \cdot S_3 + S_5)]/A,$$

где

$$A = S_1 \cdot (S_1^5 + S_5) + S_3 \cdot (S_1^3 + S_3).$$

Как правило, в специальной литературе ограничиваются формулами (4.28), (4.31), (4.33), ввиду дальнейшей громоздкости аналогичных выражений с увеличением кратности исправляемых ошибок t.

4.7. Декодирование кодов БЧХ

4.7.1. Алгоритм декодирования кодов БЧХ по формулам

В случае применения при декодировании кодов БЧХ формул для коэффициентов уравнений (4.28), (4.31), (4.33) необходимо знать число ошибок, которое произошло в кодовом слове (разрешенной кодовой комбинации). Только после этого можно в соответствующее уравнение, например (4.27) или (4.30), подставлять вместо x элементы поля Галуа и определять положение ошибок, как было рекомендовано в п. 4.6.

Для определения количества ошибок t в кодовом слове используют зависимости, существующие между частями синдрома $S_1, S_3, S_5, \ldots, S_{2t-1}$ [12].

Из частей синдрома, вычисленного по проверочной матрице $H^{г}$ кода БЧХ, составляется матрица синдромов в виде

$$\begin{bmatrix} S_1 S_2 S_3 \ldots\ldots S_{t-1} S_t \\ S_2 S_3 S_4 \ldots\ldots S_t S_{t+1} \\ S_3 S_4 S_5 \ldots\ldots S_{t+1} S_{t+2} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ S_t S_{t+1} S_{t+2} \ldots\ldots S_{2t-2} S_{2t-1} \end{bmatrix} . \qquad (4.34)$$

Дальнейшее определение числа ошибок в кодовом слове связано с выражением (4.34). Если оно составлено для конкретного значения t ошибок и в кодовой комбинации произошло именно такое их количество, то матрица (4.34) не вырождена, т.е. ее определитель $\Delta \neq$ н (нулевой элемент поля). Если же ошибок было меньше t, то $\Delta =$ н и матрица вырождена.

Приведем пример для кода БЧХ (15,5) с t = 3, проверочная матрица которого может быть представлена в виде

$$H^{\Gamma}_{15,5} = \left[\begin{array}{cccccccccccccccc} \overline{0} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ \hline 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \qquad (4.35)$$

$$\begin{array}{cccccccccccccc} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ \underline{1} & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \underline{1} \end{array}$$

Синдромная матрица (4.34) для этого случая записывается как

$$M = \begin{bmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{bmatrix}. \qquad (4.36)$$

Пусть в кодовом слове произошло 3 ошибки, например, в символах, соответствующих столбцам матрицы (4.35) 12, 13, 14. Синдром в этом случае будет равен сумме по модулю 2 этих трех столбцов, т.е.

$$S = \underset{S_1}{1011}\ \ \underset{S_3}{1001}\ \ \underset{S_5}{0000}.$$

Для S имеем: $S_1 = 1011 = \alpha^7$, $S_2 = S_1^2 = (\alpha^7)^2 = \alpha^{14}$, $S_3 = 1001 = \alpha^{14}$, $S_4 = S_2^2 = (\alpha^{14})^2 = \alpha^{28} = \alpha^{28-15} = \alpha^{13}$, $S_5 = $ н .

Вычислим теперь определитель Δ для матрицы (4.36) одним из методов [16], т.е.

$$\begin{aligned} &\Delta = S_1 \cdot [S_3 \cdot S_5 + S_4 \cdot S_4] + S_2 \cdot [S_2 \cdot S_5 + S_3 \cdot S_4] + \\ &+ S_3 \cdot [S_2 \cdot S_4 + S_3 \cdot S_3] = \alpha^7 \cdot [\alpha^4 \cdot н + \alpha^{13} \cdot \alpha^{13}] + \\ &+ \alpha^{14} \cdot [\alpha^{14} \cdot н + \alpha^{14} \cdot \alpha^{13}] + \alpha^{14} \cdot [\alpha^{13}\alpha^{13} \cdot + \alpha^{14} \cdot \alpha^{14}] = \\ &= \alpha^3 + \alpha^{11} + \alpha^{11} + \alpha^{12} = \alpha^{10}. \end{aligned} \qquad (4.37)$$

Как видно из (4.37), определитель не равен н (нулю), а равен α^{10} или через индекс степени α равен 10, из чего следует, что в кодовом слове произошло три ошибки.

При двух ошибках в символах 13 и 14 кодового слова, синдром, вычисленный по матрице (4.35), отличается от предыдущего синдрома, т.е.

$$S = \underset{S_1}{0100}\ \underset{S_3}{0101}\ \underset{S_5}{0001}. \qquad (4.38)$$

Для синдрома (4.38): $S_1 = 1011 = \alpha^2$, $S_2 = S_1 = \alpha^4 = \alpha^{14}$, $S_3 = 0101 = \alpha^3$, $S_4 = S_2^2 = (\alpha^4)^2 = \alpha^8$, $S_3 = 0001 = 1$.

Определитель в этом случае вычисляется также по формуле (4.37) и будет равен

$$\Delta = \alpha^2 [\alpha^8 \cdot 1 + \alpha] + \alpha^4 [\alpha^4 \cdot 1 + \alpha^8 \cdot \alpha^8] + \alpha^8 \cdot [\alpha^4 \cdot \alpha^8 + \alpha^{16}] =$$
$$= \alpha^{10} + \alpha^3 + \alpha^8 + \alpha^5 + \alpha^5 + \alpha^5 + \alpha^9 = н.$$

Следовательно, в кодовом слове для этого случая, произошло не 3 ошибки.

Таким образом, алгоритм декодирования кодов БЧХ по формулам реализуется в следующей последовательности:

1. По принятому кодовому слову вычисляется синдром S ($S_1, S_3, S_5, \ldots, S_{2t-1}$).

2. Если значение S= н, то в кодовом слове ошибок нет.

3. Если значение $S \neq$ н, но равно S_1, то произошла одна ошибка и ее можно определить по уравнению $x + \sigma_{11} =$ н, где $\sigma_{11} = S_1$.

4. Если значение $S \neq$ н и $S \neq S_1$, то произошло больше одной ошибки и следует вычислить определитель Δ.

5. Если определитель $\Delta \neq$ н, то произошло ровно t ошибок, для которых составляется матрица (4.34), и берется соответствующее t ошибкам уравнение с формулами для его коэффициентов. По этим формулам вычисляются их конкретные значения и записываются в уравнение. Далее, подставляя элементы поля в уравнение вместо x, определяется положение ошибок.

6. Если значение определителя Δ = н, то следует перестроить матрицу (4.34) для меньшего числа ошибок, т.е. t-1, и вернуться к вычислению определителя, и если теперь Δ

≠н, то выбрать уравнение для меньшего числа ошибок и определить их положение, как в п. 5. Если и при втором вычислении снова Δ = н, то следует вновь уменьшить размер матрицы (4.34) и повторить вычисление определителя Δ.

7. По вычисленным положениям ошибок в кодовом слове необходимо инвертировать соответствующие им символы и закончить декодирование.

Приведем теперь пример декодирования на основе вычисления по формулам для кода БЧХ (15,5), исправляющего до трех независимых ошибок включительно.

Пусть нам необходимо передать по каналу связи информационные символы 11000. Предварительно до их передачи необходимо осуществить получение кодового слова (кодирование), используя образующую матрицу кода $G_{15,5}$ (4.24), для чего потребуется сложить первую и вторую ее строки. В результате получим кодовое слово, где присутствуют информационные k и проверочные символы r, т. е

14	13	12	11	10		9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	.	0	1	0	1	0	0	1	1	0	1
		к								r					

. (4.39)

Нумерация элементов комбинации (4.39) осуществляется по индексам степеней элементов α поля $GF(2^4)$, где первому элементу поля (α^0) соответствует крайний правый символ кодового слова.

Когда в кодовом слове ошибки отсутствуют, синдром S равен нулевому элементу поля (н).

Осуществим вычисление S по матрице $H_{15,5}$ (4.35), сложив по модулю 2 те ее столбцы, номера которых совпадают с номерами позиций единиц в кодовом слове, что соответствует процедуре его умножения на матрицу H^r. В результате получим значение синдрома S, т.е.

$$0001\ \ 0001\ \ 0001 \rightarrow 0$$

0100 1100 0111 $\rightarrow$ 2
1000 1010 0001 $\rightarrow$ 3 столбцы
1100 1000 0001 $\rightarrow$ 6 H^r(4.35)
0101 1010 0111 $\rightarrow$ 8
1101 1010 0110 $\rightarrow$ 13
1001 1111 0111 $\rightarrow$ 14
0000 0000 0000 $\Rightarrow$ S .

Пусть теперь произошло 2 ошибки в символах с номерами 0 и 1 кодового слова (4.39), что соответствует изменению их значений на противоположные, т.е.

14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
1 1 0 0 0 .. 0 1 0 1 0 0 1 1 1 0 . (4.40)

Далее вычислим синдром для последовательности (4.40) аналогично предыдущему случаю, т.е.

0010 1000 0110 $\rightarrow$ 1
0100 1100 0111 $\rightarrow$ 2
1000 1010 0001 $\rightarrow$ 3 столбцы
1100 1000 0001 $\rightarrow$ 6 H^r (4.35)
0101 1010 0111 $\rightarrow$ 8
1101 1010 0110 $\rightarrow$ 13
1001 1111 0111 $\rightarrow$ 14
0011 1001 0111 $\rightarrow$ S .

Так как ошибок только две, что может быть выяснено по полученному синдрому S=0011 1001 0111 и матрице (4.36), то необходимо использовать для декодирования уравнение второй степени (4.27)

$$x^2 + \sigma_{21}x + \sigma_{22} = н,$$

где $\sigma_{21} = S_1$ и $\sigma_{22} = \frac{S_3}{S_1} + S_1^2$.

В полученном синдроме S=0011 1001 0111 будем использовать только S_1=0011= α^4 и S_3 =1001 = α^{14}.

Вычислим σ_{22}, используя значение для S_1 и S_3, т.е.

$$\sigma_{22} = \frac{S_3}{S_1} + S_1^2 = \frac{\alpha^{14}}{\alpha^4} + (\alpha^4)^2 = \alpha^{10} + \alpha^8 = 0111 + 0101 = \alpha.$$

После подстановки вычисленных значений σ в уравнение (4.27) получаем его в виде

$$x^2 + \alpha^4 x + \alpha = н. \qquad (4.41)$$

Определение корней уравнения (4.41) осуществим последовательной подстановкой элементов поля GF (2^4) вместо x.

В результате этой процедуры при $x = \alpha^0 \rightarrow 0001$ получим, что левая часть (4.41) равна нулевому элементу поля н, т.е.

$$(\alpha^0)^2 + \alpha^4 \cdot \alpha^0 + \alpha = \alpha^0 + \alpha^4 + \alpha \rightarrow 0001 + 0011 + 0010 = 0000.$$

Это означает, что ошибка находится в символе кодового слова (4.39) с номером 0, что соответствует первому элементу поля α^0, имеющему индекс степени 0 .

После подстановки второго элемента поля α в (4.41) получим, что

$$\alpha^2 + \alpha^4 \cdot \alpha + \alpha = \alpha^2 + \alpha^5 + \alpha \rightarrow 0100 + 0110 + 0010 = 0000,$$

и значит, вторая ошибка в кодовом слове (4.39) будет находиться в символе с номером 1, что соответствует второму элементу поля α, имеющему индекс степени, равный единице.

Таким образом, позиции ошибок определены и следует далее изменить символы с номерами 0 и 1 в кодовом слове

(4.40) на противоположные. В результате получим исправленное кодовое слово

11000 . 010100 1101.

4.8. Лабораторное задание

1. Включить ЭВМ и войти в диск Z. Выбрать файл LABRNEW и открыть его. Затем открыть файл LAB4Win и далее файл labr4 (рис. 4.1) [25,26].

2. Построить поле Галуа GF(2^4) от α^0 до α^{14} по полиному $P(x) = x^4+x+1$, который представляется в двоичном виде как $P(x) \rightarrow 10011$ и в десятичном виде как $P(x) \rightarrow 19$. Для этого выполнить следующие пункты:

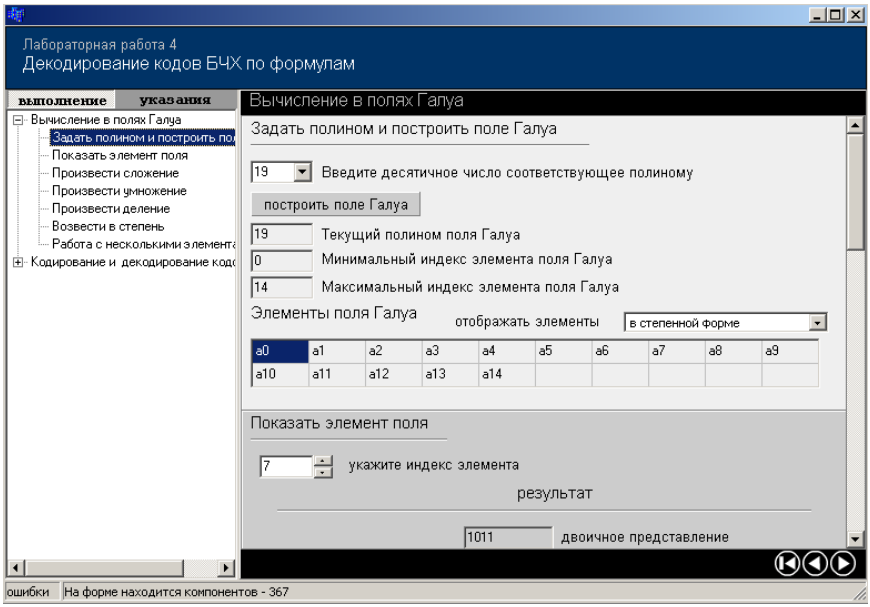


Рис. 4.1. Заставка «Декодирование кодов БЧХ по формулам» для выполнения операций с элементами поля Галуа

2.1. В заставке «Декодирование кодов БЧХ по формулам» (рис. 4.1) выбрать строку «Задать полином и построить поле Галуа», а затем нажать на кнопку «построить поле Галуа». Зафиксировать в отчете все элементы поля $GF(2^4)$ в двоичном виде.

2.2. Задать полином P(x) →37и построить поле Галуа. Используя строку «Показать элемент поля», вывести несколько элементов поля из начала, середины и конца таблицы поля, указав индексы элементов поля. Элементы поля представить в степенной форме элементов поля и в двоичном виде. Осуществить аналогичные процедуры для полинома P(x) →137.

2.3. Задать полином P(x) →19 и построить поле $GF(2^4)$, используя соответствующие надписи в заставке «Вычисление в полях Галуа». Осуществить неколько сложений, умножений, делений и возведений в степень различных двух элементов поля $GF(2^4)$.

3. Записать проверочную матрицу H^{Γ} для кода с d=7, способного исправлять одну, две или три ошибки, используя поле Галуа $GF(2^4)$. Для этого выполнить следующие пункты:

3.1. Используя запись строки матрицы H^{Γ} в виде

$$\alpha^0;\ \alpha^{2t-1};\ \alpha^{2(2t-1)};\ \alpha^{3(2t-1)};\ \ldots\ \alpha^{(n-1)(2t-1)},$$

представить H^{Γ} в виде трёх строк для кода длины n=15, где для первой строки t = 1, для второй строки t = 2, для третьей строки t = 3. Зафиксировать полученную матрицу H^{Γ} в отчет со строками, представленными в виде элементов поля $GF(2^4)$, представленных в показательной форме (см. стр.115 пособия).

4. Используя построенную матрицу H^{Γ} из п. 3.1, осуществить процедуру исправления двух и трёх ошибок, введённых в разрешенную кодовую комбинацию (кодовое слово) длиной n. Для этого выполнить следующие пункты:

4.1. Для ввода двух ошибок в кодовое слово длиной n=15 осуществить сложение двух произвольных столбцов (номера указываются преподавателем) матрицы H^{Γ} из п. 3.1. Проделать то же самое, но для трёх столбцов H^{Γ} (номера указываются преподавателем). Этим будет осуществлен ввод ошибок и вычисление синдромов соответственно S_1, S_3 для двух ошибок и S_1, S_3, S_5 для трёх ошибок, где номера символов с ошибками совпадают с номерами столбцов H^{Γ}. Зафиксировать полученные результаты, а также вычислить $S_2=S_1^2$ и $S_4=S_2^2$. Записать соответствующие вычисления синдромов в отчет. Для сложения воспользоваться процедурой «Вычисление в полях Галуа». Производя сложение, предварительно убедиться, что задано поле $GF(2^4)$ с P(x) →19 и оно построено. Для возведения в степень воспользоваться строкой «Возвести в степень».

4.2. Определить коэффициенты σ (сигма) для уравнений

$$x^2+\sigma_{21}x+\sigma_{22} = н \text{ и } x^3+\sigma_{31}x^2+ \sigma_{32}x+\sigma_{33} = н,$$

воспользовавшись значениями синдромов из п. 4.1 и формулами:

$$\sigma_{21} = s_1, \qquad \sigma_{22} = \frac{s_1^3 + s_3}{s_1}, \qquad \sigma_{31} = s_1, \qquad \sigma_{32} = \frac{s_1^2 \cdot s_3 + s_5}{s_1^3 + s_3},$$

$$\sigma_{33} = \frac{s_2^3 + s_3^2 + s_1(s_1^2 \cdot s_3 + s_5)}{s_1^3 + s_3}.$$

Для этой цели воспользоваться строкой «Работа с несколькими элементами» и ввести значения переменных S_1, S_2, S_3, S_4, S_5, вычисленных в п. 4.1 (нулевое значение синдромов (н) вводится кнопкой как (-1)). Далее открыть список окна «Введите выражение» через кнопку расположенную под надписью справа, и последовательно открывая формулы для σ_{22}, σ_{32}, σ_{33}, вычислить их значения, нажимая на клавишу «Результат». Результаты зафиксировать в отчет и ввести в список переменных.

4.3. По полученным значениям σ_{21} и σ_{22} (п. 4.2) и уравнению $x^2 + \sigma_{21}x + \sigma_{22} = н$ определить положение ошибок в кодовом слове. Для этого ввести из заставки «Введите выражение» формулу уравнения для двух ошибок $x^2 + G21x + G22 = н$, где $G21=\sigma_{21}$, $G22=\sigma_{22}$. Последовательно вводить в название переменной x из списка переменных значения индексов элементов поля: 0, 1, 2, 3, 4, …14, каждый раз нажимая на клавишу «Результат». Зафиксировать результаты вычислений. Определить положение ошибок как значения индексов, приводящие к нулевым (н) значениям результатов вычислений по уравнению. Полученные величины зафиксировать в отчет.

4.4. По полученным значениям $\sigma_{31}=G31$, $\sigma_{32}=G32$ и $\sigma_{33}=G33$ (п. 4.2) и уравнению $x^3 + G31x^2 + G32x + G33 = н$ определить положение трёх ошибок, используя

последовательность вычислений п. 4.3. Результаты зафиксировать в отчет.

4.5. Осуществить декодирование кодов БЧХ в режиме вычислений. Для этого открыть заставку «Кодирование и декодирование кодов БЧХ» (рис. 4.2).

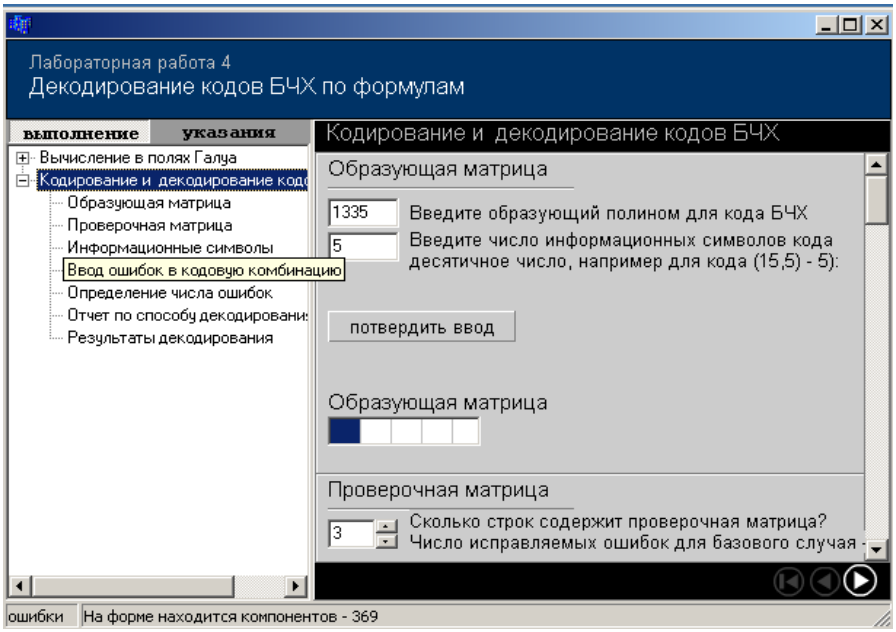


Рис. 4.2. Заставка «Декодирование кодов БЧХ по формулам» для выполнения операций кодирования и декодирования

4.6. Для кода БЧХ(15,5) с полиномом равным 1335 и числом информационных символов 5 построить образующую матрицу кода (15,5) через заставку «Образующая матрица» и подтвердить ввод. Получить проверочную матрицу кода (15,5), аналогично образующей матрице. Сравнить полученную матрицу с проверочной матрицей H^{Γ} из п. 3.

4.7. Ввести информационные символы числом от 0 до 31 (номер студента по списку группы) и подтвердить ввод. Зафиксировать в отчет полученные кодовые последовательности.

4.8. Ввести три ошибки в позиции символов кодового слова в соответствии с номерами столбцов из п. 4.1 через строку «Ввод ошибок в кодовую комбинацию» и подтвердить ввод. Записать полученные кодовые последовательности.

4.9. Открыть строку «Определение числа ошибок». Вычислить компоненты синдрома (значения элементов поля Галуа). Зафиксировать компоненты синдрома в отчет и сравнить их с величинами, полученными в п. 4.1. Записать в отчет также матрицу синдромов и значение её определителя. Отразить в отчете процедуру определения числа ошибок.

4.10. Открыть строку «Отчет по способу декодирования ошибок» и переместить движок окна вверх. Зафиксировать из окна порядок декодирования. Сравнить величины индексов коэффициентов сигма со значениями из п. 4.2.

4.11. Открыть строку «Результат декодирования» и зафиксировать полученные кодовые комбинации после исправления ошибок.

4.12. Перейти к началу заставки «Кодирование и декодирование кодов БЧХ» (рис. 4.2). Открыть строку «Ввод ошибок в кодовую комбинацию». Установить количество ошибок равное 2 и ввести их номера, из п. 4.1. Подтвердить ввод.

4.13. Осуществить декодирование двух ошибок, используя последующие строки. Вычислить вновь компоненты синдрома. Зафиксировать в отчет изменения в процедуре исправления двух ошибок для кода БЧХ(15,5) с d=7 по сравнению с исправлением им трех ошибок.

4.14. Поступая анологично процедуре п. 4.12 и п. 4.13, осуществить декодирование одной ошибки по номеру одного из выбранных столбцов из п.4.1. Зафиксировать результаты и отличия от декодирования для трех и двух ошибок.

4.15. Осуществить кодирование и декодирование информации, применив теперь тоже поле Галуа $GF(2^4)$, представленное полиномом $P(x) \rightarrow 19$, но используя полином кода БЧХ(15,7) $P(x) \rightarrow 465$ и число информационных символов k=7. Для этого при открытой заставке «Кодирование и декодирование кодов БЧХ» (рис. 4.1) нажать «Образующая матрица» и ввести в окна полином $P(x) \rightarrow 465$ и k=7.

После подтверждения ввода перейти к проверочной матрице и ввести число строк 2 с подтверждением ввода.

4.16. Используя далее порядок ввода информационных символов и ошибок в кодовую комбинацию, осуществить процедуру декодирования, применяемую ранее для кода БЧХ(15,5) с $P(x) \rightarrow 1335$. Зафиксировать результаты декодирования в отчет.

Содержание отчета

1. Количественные показатели по выполнению конкретных пунктов с их комментариями и выводами.
2. Краткие выводы по лабораторной работе.

Контрольные вопросы

1. Что понимают под математическими терминами поле вообще и поле Галуа?
2. Что такое простое поле и расширение простого поля Галуа?
3. Какими свойствами обладают конечные поля Галуа?
4. Какие многочлены используются при построении полей Галуа?
5. Как строится поле Галуа по известному полиному $P(x)$?
6. Какие основные математические действия существуют в полях Галуа и как они выполняются?

7. Дайте понятие корней многочленов. Какие особенности корней используются при разложении многочленов на сомножители?
8. Как осуществляется построение проверочной матрицы H^{Γ} для кодов БЧХ с различным числом исправляемых ошибок?
9. Как определяется число проверочных символов кода БЧХ?
10. Что необходимо знать для определения числа информационных символов кода БЧХ?
11. Как определяется образующий полином кода БЧХ?
12. Для каких целей используется образующий полином кода БЧХ и как это осуществляется?
13. Как решаются системы уравнений в полях Галуа?
14. Дайте определение коэффициентам σ (сигма). Как формируются формулы для их вычислений?
15. Как определяется число ошибок в кодовой комбинации кода БЧХ?
16. Что такое процедура Ченя и как она реализуется для кодов БЧХ?
17. Как осуществляется декодирование кодов БЧХ по формулам.

5. ЛАБОРАТОРНАЯ РАБОТА № 5

ДЕКОДИРОВАНИЕ КОДОВ БЧХ АЛГОРИТМОМ ПГЦ

Цель работы: изучить процедуру декодирования кодов БЧХ алгоритмом Питерсона-Горенстейна-Цирлера.

Теоретическая часть

5.1. Декодирование кодов БЧХ по алгоритму Питерсона – Горенстейна - Цирлера (ПГЦ)

Декодирование кодов БЧХ на основе формул имеет ограничение, связанное с количеством исправляемых ошибок, когда при $t \geq 4$ из-за громоздкости вычислений оно становится непригодным для практического использования. Устранить недостатки такого декодирования возможно на основе алгоритма Питерсона – Горенстейна - Цирлера (ПГЦ), основанного на свойствах матриц, используемых для решения систем уравнений [12]. Как и в предыдущем методе, декодирование начинается с вычисления синдрома S по проверочной матрице кода H^r, равного сумме тех ее столбцов, которые соответствуют позициям ошибок в кодовом слове. Таким образом, можно записать систему уравнений, включающую 2t составляющих S, выраженных через элементы поля Галуа x_i^j, где i изменяется от 1 до ν и ($\nu \leq t$), а j от 1 до 2t, т.е.

$$\begin{cases} S_1 = x_1 + x_2 + \ldots + x_\nu \\ S_2 = x_1^2 + x_2^2 + \ldots + x_\nu^2 \\ S_3 = x_1^3 + x_2^3 + \ldots + x_\nu^3 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ S_{2t} = x_1^{2t} + x_2^{2t} + \ldots + x_\nu^{2t}. \end{cases} \qquad (5.1)$$

Решение системы уравнений (5.1) затруднено, т. к. она является нелинейной. В связи с этим предлагается процедура, позволяющая найти это решение через вспомогательные преобразования [5].

Вводится в рассмотрение дополнительный многочлен от x

$$\lambda(x) = \lambda_\nu x^\nu + \lambda_{\nu-1} x^{\nu-1} + \ldots + \lambda_1 x + 1, \qquad (5.2)$$

известный под названием многочлена локаторов позиций ошибок и определяемый как многочлен, корнями которого являются обратные к локаторам (позициям) ошибок элементы X_ℓ для $\ell = 1, \ldots, \nu$. С учетом определения корней многочлен $\lambda(x)$ можно записать в другой форме, т.е.

$$\lambda(x) = (1 + xX_1) \cdot (1 + xX_2) \ldots (1 + xX_\upsilon). \qquad (5.3)$$

Используя два вида представления (5.2) и (5.3) для $\lambda(x)$, запишем следующее выражение:

$$(1+xX_1) \cdot (1+xX_2) \ldots (1+xX_\nu) = \lambda_\nu x^\nu + \lambda_{\nu-1} x^{\nu-1} + \ldots + \lambda_1 x + 1. \qquad (5.4)$$

Умножим обе части равенства (5.4), определяющего многочлен, на $X_\ell^{j+\upsilon}$ и положим $x = X_\ell^{-1}$. Тогда левая часть (5.4) обратится в нулевой элемент (н), и мы получим

$$\text{н} = X_\ell^{j+\nu} (1 + \lambda_1 X_\ell^{-1} + \lambda_2 X_\ell^{-2} + \ldots + \lambda_{\nu-1} X_\ell^{-(\nu-1)} + \lambda_\nu X_\ell^{-\nu}),$$

или

$$X_\ell^{j+\nu} + \lambda_1 X_\ell^{j+\nu-1} + \ldots + \lambda_\nu X_\ell^{j} = \text{н}. \qquad (5.5)$$

Равенство (5.5) выполняется при каждом ℓ и при каждом j. Просуммируем эти равенства по ℓ от 1 до ν. Для каждого j это дает

$$\sum_{\ell=1}^{\nu} (X_\ell^{j+\nu} + \lambda_1 X_\ell^{j+\nu-1} + \ldots + \lambda_1 X_\ell^{j}) = \text{н},$$

или

$$\sum_{\ell=1}^{\nu}(X_{\ell}^{j+\nu}+\lambda_1\sum_{\ell=1}^{\nu}X_{\ell}^{j+\nu-1}+\ldots+\lambda_1 X_{\ell}^{j})=\text{н}. \qquad (5.6)$$

Каждая сумма в левой части равенства (5.6) является компонентой синдрома S, так что уравнение приводится к виду

$$S_{j+\nu}+\lambda_1 S_{j+\nu-1}+\lambda_2 S_{j+\nu-2}+\ldots+\lambda_\nu S_j=\text{н}. \qquad (5.7)$$

Так как $\nu \le t$, то для j в интервале $1 \le j \le \nu$ все индексы задают компоненты синдрома. С учетом этого, перенеся предварительно $S_{j+\nu}$ вправо, получаем систему уравнений

$$\lambda_1 S_{j+\nu-1}+\lambda_2 S_{j+\nu-2}+\ldots+\lambda_\nu S_j=S_{j+\nu},\ \ j=1,\ \ldots,\ \nu, \qquad (5.8)$$

которая связывает компоненты синдрома с коэффициентами многочлена $\lambda(x)$. Выражение (5.8) можно записать в матричном виде

$$\begin{bmatrix} S_1 & S_2 & S_3 & \ldots & S_{\nu-1} & S_\nu \\ S_2 & S_3 & S_4 & \ldots & S_\nu & S_{\nu+1} \\ S_3 & S_4 & S_5 & \ldots & S_{\nu+1} & S_{\nu+2} \\ \cdot & & & & & \\ \cdot & & & & & \\ \cdot & & & & & \\ S_\nu & S_{\nu+1} & S_{\nu+2} & \ldots & S_{2\nu-2} & S_{2\nu-1} \end{bmatrix} \cdot \begin{bmatrix} \lambda_\nu \\ \lambda_{\nu-1} \\ \lambda_{\nu-2} \\ \cdot \\ \cdot \\ \cdot \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} S_{\nu+1} \\ S_{\nu+2} \\ S_{\nu+3} \\ \cdot \\ \cdot \\ \cdot \\ S_{2\nu} \end{bmatrix}. \qquad (5.9)$$

Представим систему (5.9), например, для случая $\nu = t = 3$, когда код БЧХ способен корректировать три и менее ошибок

$$\begin{bmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{bmatrix} \cdot \begin{bmatrix} \lambda_3 \\ \lambda_2 \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} S_4 \\ S_5 \\ S_6 \end{bmatrix}, \qquad (5.10)$$

где многочлен локаторов ошибок (5.2) для тех же значений будет иметь вид

$$\lambda(x) = \lambda_3 x^3 + \lambda_2 x^2 + \lambda_1 x + 1. \qquad (5.11)$$

Теперь стоит задача вычисления значений коэффициентов λj из системы (5.9), например, методом обращения матрицы и последующего их использования в многочлене локаторов ошибок (5.2), (5.11).

Дальнейшее определение положения ошибок может осуществляться, например, последовательной подстановкой элементов поля Галуа в многочлен локаторов $\lambda(x)$ (5.11), называемой методом Ченя [5]. В том случае, когда $\lambda(x)$ становится равным нулевому элементу поля (н**)**, считается что положение одного из искажений определено через его корень и т.д. до значения исправляющей способности кода БЧХ $\nu = t$.

Таким образом, методом подстановки элементов поля в $\lambda(x)$ определяются его корни, а местоположения ошибок в кодовом слове находятся как обратные величины от их значений.

Все это можно осуществить, если матрица (5.9) не вырождена, что имеет место в том случае, когда в кодовом слове произошло ровно $\nu = t$ ошибок, равных исправляющей способности кода, или матрица сокращена до числа ошибок, действительно произошедших в кодовом слове, что аналогично ранее описанной процедуре для декодирования по формулам.

В связи с вышеизложенным алгоритм декодирования Питерсона-Горенстейна-Цирлера для двоичных кодов БЧХ можно отобразить структурной схемой, представленной на рис. 5.1.

Первоначально вычисляются компоненты синдрома S_1, S_2 , S_3 , …, S_{2t}.

Далее определяется число ошибок, произошедших в кодовом слове. На первом шаге допускается, что их значение ν = = t, т.е. максимальной величине, реализуемой кодом. Если это так, то определитель матрицы M, составленной из

компонент синдрома S, не равен н. Если det(M) = н, то необходимо уменьшить значение числа ошибок ν на единицу и вернуться к вычислению определителя матрицы M с синдромами для ν-1.

Эта процедура должна продолжаться до тех пор, пока не будет выполняться условие det(M) ≠ н.

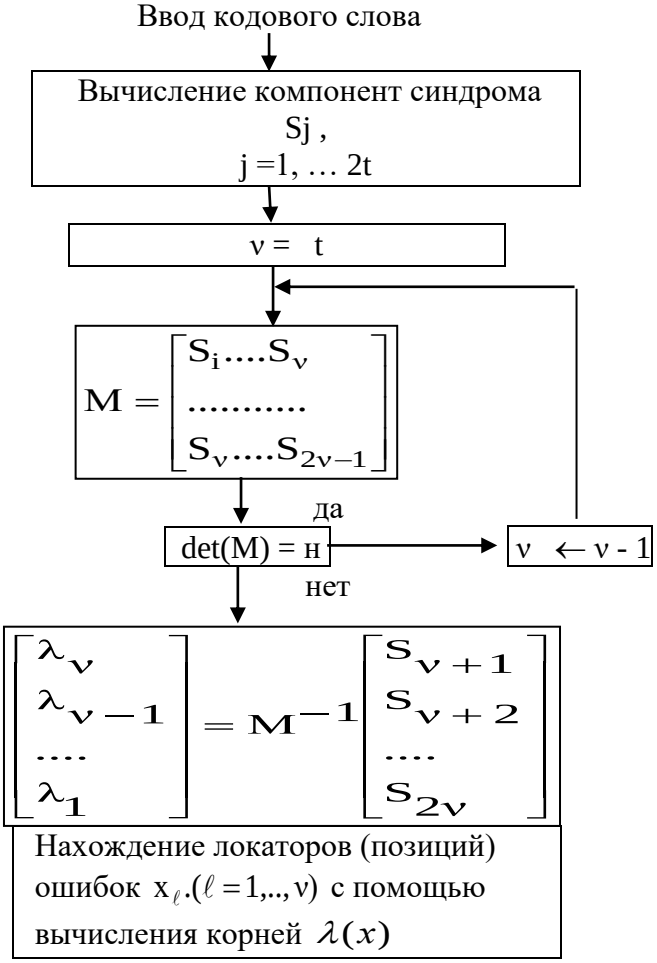


Рис. 5.1. Структурная схема декодера ПГЦ

После определения количества ошибок ν, произошедших в кодовом слове, переходят к определению коэффициентов λ_ν многочлена локаторов ошибок

$$\lambda(x) = \lambda_\nu x^\nu + \lambda_{\nu-1} x^{\nu-1} + \ldots + \lambda_1 x + 1.$$

Их вычисление осуществляют методом обращения матрицы M.

Далее используют процедуру последовательной подстановки элементов поля Галуа в многочлен $\lambda(x)$ локаторов ошибок и находят локаторы ошибок x_ℓ (ℓ=1,…,ν) с помощью вычисления его корней, с последующим определением их обратного значения.

Приведем пример декодирования кода БЧХ (15,5) с t =3 процедурой Питерсона-Горенстейна-Цирлера.

Код БЧХ (15,5) имеет $P(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + 1$, и пусть ошибки произошли в символах 12 и 7 кодового слова, состоящего из одних нулей, т.е.

14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 → нумерация элементов
k r поля
0 0 1 0 0. 0 0 1 0 0 0 0 0 0 0 → кодовое слово (5.12)
14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 → номера символов кодового слова

Номера символов кодового слова в (5.12) выбраны таким образом, чтобы они соответствовали представлению ошибок в виде многочлена, т.е. для данного случая, $E(x) = x^{12} + x^7 \rightarrow$ $\rightarrow$1000010000000, что отражает правила построения многочленов из п. 3.2 и соответствует (5.12).

Для вычисления синдрома S используется кодовое слово (5.12), по положению единиц в котором суммируется 12–й и 7-й столбцы проверочной матрицы H^r(4.35), т.е.

$$\begin{array}{rl} (12) & 1111 \quad 1100 \quad 0001 \\ (7) + & \underline{1011 \quad 1100 \quad 0110} \\ & 0100 \quad 0000 \quad 0111 \end{array} \qquad (5.13)$$

$$S_1 = \alpha^2,\ S_3 = н,\ S_5 = \alpha^{10}.$$

Дополнительно к S_1, S_3 и S_5 вычисляем $S_2 = (S_1)^2 = (\alpha^2)^2$ и $(S_2)^2 = (\alpha^4)^2 = \alpha^8$.

Вначале количество ошибок в кодовом слове на приёмной стороне неизвестно, и оно предполагается $\nu=3$. Для этого случая матрица синдромов (5.9) будет иметь вид

$$M = \begin{bmatrix} S_1 S_2 S_3 \\ S_2 S_3 S_4 \\ S_3 S_4 S_5 \end{bmatrix} = \begin{bmatrix} 2\ 4\ н \\ 4\ н\ 8 \\ н\ 8\ 10 \end{bmatrix} \qquad (5.14)$$

Определитель матрицы (5.14) равен нулю (н), т.к. в диагонали стоят нулевые элементы поля н. В соответствии с алгоритмом полагаем $\nu=2$ и переходим к новой матрице M, т.е.

$$M = \begin{bmatrix} S_1 S_2 \\ S_2 S_3 \end{bmatrix} = \begin{bmatrix} 2\ 4 \\ 4\ н \end{bmatrix} . \qquad (5.15)$$

Вычисляем определитель матрицы (5.15)

$$\det M = \alpha^2 \cdot н + \alpha^4 \cdot \alpha^4 = \alpha^8,$$

т.е. он не равен нулю и, действительно, в кодовом слове произошло две ошибки.

Далее, в соответствии с алгоритмом вычисляем обратную матрицу

$$M^{-1} = \frac{1}{\det M} \cdot \begin{bmatrix} \Delta_{11} \Delta_{21} \\ \Delta_{12} \Delta_{22} \end{bmatrix} , \qquad (5.16)$$

где через Δ_{11} , Δ_{21} , Δ_{12}, Δ_{22} обозначены алгебраические дополнения элементов матрицы M (5.15).

В результате после подстановки вычисленных элементов для (5.16) получаем

$$M^{-1} = \frac{1}{\alpha^8}\begin{bmatrix} н & 4 \\ 4 & 2 \end{bmatrix} = \begin{bmatrix} н & 11 \\ 11 & 9 \end{bmatrix} .$$

Вычисляем теперь коэффициенты многочлена локаторов ошибок, т.е.

$$\begin{bmatrix} \lambda_2 \\ \lambda_1 \end{bmatrix} = M^{-1}\begin{bmatrix} S_3 \\ S_4 \end{bmatrix} = \begin{bmatrix} н & 11 \\ 11 & 9 \end{bmatrix} \cdot \begin{bmatrix} н \\ 8 \end{bmatrix} = \begin{bmatrix} н \cdot н + 11 \cdot 8 \\ 11 \cdot н + 9 \cdot 8 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}.$$

Таким образом, имеем, что $\lambda_1=2$, $\lambda_2=4$. Используя эти значения, получаем

$$\lambda(x) = \alpha^4 \cdot x^2 + \alpha^2 \cdot x + 1. \qquad (5.17)$$

Далее осуществляем подстановку элементов поля GF (2^4) вместо x в (5.17) и определяем корни многочлена локаторов ошибок.

Для многочлена (5.17) получаем, что он равен нулевому элементу поля н при
$x = \alpha^8$ и $x = \alpha^3$, т.е.

$$\lambda(x) = \alpha^4(\alpha^8)^2 + \alpha^2 \cdot \alpha^8 + \alpha^0 = н$$

$$\lambda(x) = \alpha^4(\alpha^3)^2 + \alpha^2 \cdot \alpha^3 + \alpha^0 = н.$$

Теперь определяем по полученным корням α^8 и α^3 локаторы ошибок для кодового слова как

$$X_{\ell(1)} = 1/\alpha^8 = \alpha^{15}/\alpha^8 = \alpha^7, \qquad X_{\ell(2)} = 1/\alpha^3 = \alpha^{15}/\alpha^3 = \alpha^{12},$$

т.е. ошибки произошли в седьмом и двенадцатом символах кодового слова (5.12).

Декодирование на этом заканчивается.

5.2. Лабораторное задание

1. Включить ЭВМ и войти в диск Z. Выбрать файл LABR-NEW и открыть его. Затем открыть файл LAB5Win и далее labr5 (рис. 5.2) [27].

2. Осуществить кодирование и декодирование информации кодами БЧХ на основе алгоритма ПГЦ. Для этого выполнить следующие пункты:

2.1. В заставке «Кодирование и декодирование кодов БЧХ алгоритмом ПГЦ» выбрать название «Образующая матрица» (рис. 5.2) и в выделенном фрагменте подтвердить ввод кода БЧХ (15,5) с образующим полиномом P(x) $\rightarrow$1335 с числом информационных символов 5 и полиномом поля Галуа $GF(2^4)$ P(x) $\rightarrow$ 19. Зафиксировать в отчете образующую матрицу кода (15,5).

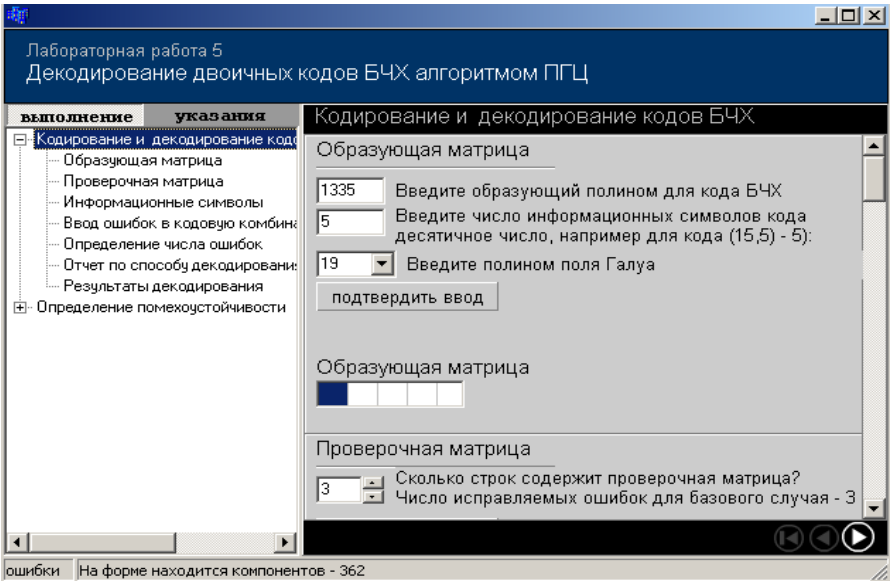


Рис. 5.2. Заставка для изучения процедур декодирования информации корректирующими кодами БЧХ алгоритмом ПГЦ

2.2. Перейти к названию «Проверочная матрица» и в выделенном фрагменте с числом строк 3 подтвердить ее ввод. Зафиксировать в отчете проверочную матрицу кода (15,5).

2.3. Открыть название «Информационные символы» и ввести десятичное число, соответствующее номеру студента по списку с подтверждением ввода. Записать полученные кодовые комбинации.

2.4. Перейти к заставке «ввод ошибок в кодовую комбинацию». Ввести три ошибки в позиции символов кодовой комбинации по указанию преподавателя. Подтвердить ввод. Записать в отчет полученные кодовые комбинации с ошибками.

2.5. Осуществить определение числа ошибок в кодовом слове через строку «Определение числа ошибок». Провести вычисление индексов компонент синдрома. Отразить также в отчете их значение совместно с матрицей синдромов. Зафиксировать в отчете последовательность уменьшения размерности матрицы синдромов.

2.6. Зафиксировать в тетрадь отчет по способу декодирования ошибок алгоритмом ПГЦ для кода (15,5) с t = 3, открыв строки «Отчет по способу декодирования» и «Результаты декодирования».

2.7. Осуществить декодирование двух ошибок кодом (15,5) с t = 3, начиная со строки «Ввод ошибок в кодовую комбинацию». Провести необходимые изменения, подтверждения и вычисления новых вводов. Отразить произошедшие изменения.

2.8. Осуществить процедуру декодирования одной ошибки кодом (15,5) с t = 3, используя процедуру предыдущего пункта. Отразить новые изменения в процедуре декодирования.

2.9. Перейти к строке «Образующая матрица». Ввести образующий полином кода P(x) $\rightarrow$ 465, число

информационных символов к = 7, соответствующее коду (15,7) с t = 2. Значение полинома поля Галуа P(x) $\rightarrow$ 19. Подтвердить ввод.

2.10. Для выполнения декодирования кода (15,7) использовать процедуры пунктов 2.2 – 2.6. Зафиксировать в отчет всю необходимую информацию.

2.11. Перейти к строке «Образующая матрица». Ввести образующий полином кода P(x) $\rightarrow$36783, число информационных символов к = 16, соответствующие коду (31,16) с числом исправляемых ошибок t = 3. Значение полинома поля Галуа установить равным числу 37. Подтвердить ввод.

Определить минимальное и максимальное количество единиц в строках полученной образующей матрицы. Как оно соотносится с исправляющей способностью кода? Отразить это в отчете.

2.12. Перейти к строке «Проверочная матрица» и ввести число строк матрицы равное исправляющей способности кода (31,16), т.е. 3. Подтвердить ввод.

2.13. Осуществить далее процедуры кодирования и декодирования для числа ошибок t = 3, используя строки расположенные в заголовке ниже. Осуществить то же самое при t=2. Отразить в отчете основные процедурные вычисления и многочлены локаторов ошибок.

2.14. Поступая аналогично пунктам 2.11-2.12, провести исследования для кода (31,11) с t = 4, имеющего образующий полином P(x) $\rightarrow$ 1451733 и число информационных символов к = =11. Значение полинома поля Галуа установить равным числу 37. Отразить в отчете основные процедурные вычисления для t = 4, t = 3 и t = 2, как в пункте 2.13.

3. Исследовать помехоустойчивость кодов БЧХ.

3.1. Выбрать строку «Определение помехоустойчивости» и далее «Моделирование» (рис. 5.3).

3.2. Для представленных в открывшейся заставке полиномов определить вероятность ошибки $P_{ош}$ на выходе декодера при разных видах помех.

3.3. Для исследования влияния независимой помехи на декодер принимать величину вероятности p независимых ошибок на символ в канале равной: 0,01; 0,02; 0,03; 0,04; 0,05; 0,06; 0,07; 0,08; 0,09; 0,1. Число блоков для полиномов брать равным: 1 – 200000; 2 – 114290; 3 – 160000; 4 – 72728.

Результаты исследований занести в таблицу. По данным таблицы построить зависимости $P_{ош}$ от вероятности p для разных номеров полиномов. Масштаб по осям принять логарифмическим.

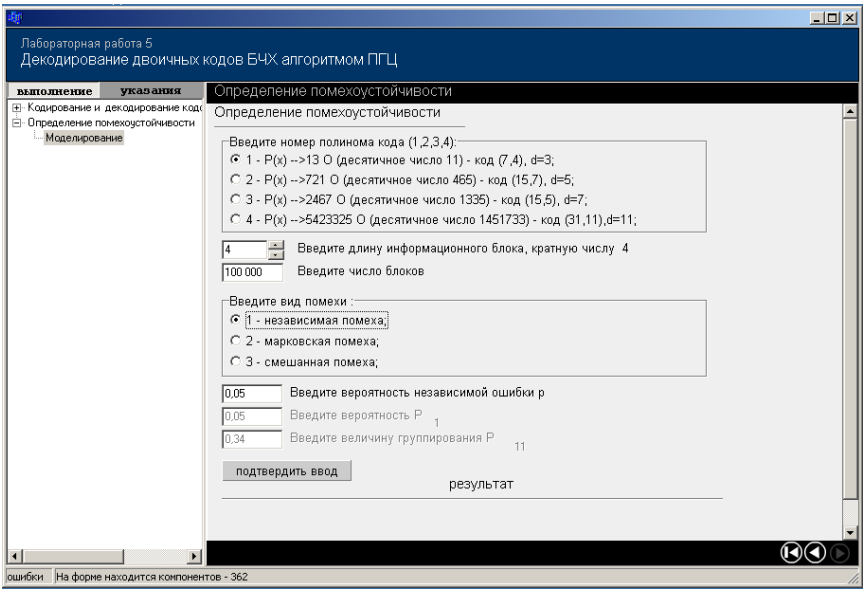


Рис. 5.3. Заставка для исследования помехоустойчивости корректирующих кодов БЧХ при различных полиномах P(x)

3.4. Для исследования влияния марковской помехи на декодер выбрать полином 1. Ввести число блоков 200000. При значениях P_1 = p, взятых из п. 3.3, провести исследования для величин P_{11} = 0,1 и P_{11} = 0,8. Результаты измерений занести в таблицу совместно с данными из п. 3.3 для полинома 1 при независимой помехе. Сделать выводы.

3.5. Исследовать воздействие смешанной помехи на декодер с полиномом 2. Ввести число блоков равным 100000 и установить вид помехи 3- смешанная помеха. Установить вероятность независимой ошибки p=0,0 (помеха отсутствует). Вероятность P_1 принять равной 0,01, а величину P_{11} установить равной 0,34. После подтверждения ввода записать в отчет результаты исследования. Ввести вероятность p = 0,01, а остальные вероятности оставить прежними. Полученный результат сравнить с предыдущим. Осуществить аналогичные исследования для величин P_{11} = 0,1 и P_{11} = 0,8. Результаты занести в таблицу. Сделать выводы.

Содержание отчета

1. Количественные показатели по выполнению конкретных пунктов с их комментариями и выводами.
2. Графики и таблицы по проведенным исследованиям.
3. Краткие выводы по лабораторной работе.

Контрольные вопросы

1. Как строится поле Галуа $GF(2^m)$ по известному полиному P(x)?
2. Какие основные математические действия существуют в поле Галуа и как они выполняются?
3. Как осуществляется построение проверочной матрицы $H^Г$ для кодов БЧХ с различным числом исправляемых ошибок t?

4. Как осуществляется кодирование информации кодами БЧХ?
5. Что такое процедура Ченя и как она реализуется для кодов БЧХ?
6. Чем ограничено применение процедуры декодирования кодов БЧХ по формулам?
7. Каким образом решается проблема формул в алгоритме ПГЦ?
8. На чем основана процедура решения системы уравнений в алгоритме ПГЦ?
9. Что представляет собой многочлен локаторов ошибок?
10. Какие имеются особенности корней многочлена локатора ошибок?
11. Какие основные этапы процедуры декодирования кодов БЧХ входят в алгоритм ПГЦ?
12. Как определяется число ошибок, произошедших в кодовом слове БЧХ?
13. Как осуществляется вычисление компонент синдрома по проверочной матрице H^{Γ} при известных положениях ошибок в кодовом слове?
14. С помощью, какой процедуры и как осуществляется вычисление коэффициентов λ_v для многочлена локаторов ошибок $\lambda(x)$?

6. ЛАБОРАТОРНАЯ РАБОТА № 6

КОДЫ РИДА-СОЛОМОНА

Цель работы: изучить построение, а также методы кодирования и декодирования информации кодами Рида-Соломона.

Теоретическая часть

6.1. Введение в коды Рида-Соломона

У кодов Рида-Соломона (РС) символы кода представляют собой m-битовые последовательности, где m- положительное число большее 1, поэтому они называются недвоичными, в отличие от двоичных кодов, где символы кода принимают только значения 0 или 1 [1]. Наибольшее применение нашли коды РС, у которых символы взяты из элементов поля Галуа GF(q), где $q=2^m$ и $m \geq 3$. При этом получаются коды РС, по сути являющиеся m –ичными ($m \geq 3$) кодами БЧХ длины $2^m-1 = q-1$. В качестве символа теперь в них выступает не двоичный символ (бит), а элемент поля Галуа (несколько битов). Например, произвольное кодовое слово для кода Рида-Соломона с m=3 и длиной $2^3-1=7$, записывается как набор из элементов поля $GF(2^3)$, т. е.

$$\begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{array} \rightarrow \text{позиции символов кода РС}, \qquad (6.1)$$

где на позициях символов кода РС располагаются элементы поля $GF(2^3)$, т.е. $\alpha^1\alpha^1\alpha^1\alpha^1\alpha^5\alpha^0\alpha^4$.

При таких обозначениях в коде РС (6.1) говорят об ошибках расположения (ошибках в позициях символов кодового слова) и об ошибках значения в элементах поля $GF(2^3)$ [1]. Длина кода Рида-Соломона обозначается большой буквой N в отличие от n для других кодов и $N = q - 1$, где N измеряется в числе позиций символов кодового слова.

Порождающий многочлен кода РС длины $N = q-1$ может быть записан в виде [1]

$$P(x) = (x + \alpha)(x + \alpha^2)\ldots(x + \alpha^{2t}). \qquad (6.2)$$

Заметим, что степень $P(x)$ в (6.2) равна $2t$, так что для исправления t ошибок в позициях кодового слова требуется только 2t проверочных символов, т.е. $R = 2t$.

Число информационных символов кода РС обозначается также как и его длина N большой буквой K, т.е.

$$K = N - R = N - 2t = N - (2t + 1 - 1) =$$
$$= N - D + 1, \qquad (6.3)$$

где $D = 2t + 1$ - кодовое расстояние по обычному определению для двоичных кодов.

Например, найдем образующий полином $P(x)$ кода РС с $t = 1$ для поля $GF(2^3)$. Исходя из (6.2) запишем, что

$$P(x) = (x + \alpha)(x + \alpha^2) = x^2 + \alpha^2 x + \alpha x + \alpha^3 = x^2 + (\alpha^2 + \alpha)x + \alpha^3 =$$
$$= x^2 + \alpha^4 x + \alpha^3. \qquad (6.4)$$

Код РС, построенный по (6.4), будет иметь $N = 7$, а $K = N - D + 1 = 7 - 3 + 1 = 5$ в соответствии с (5.3) и одно из его возможных кодовых слов (6.1) было показано ранее. Число проверочных символов здесь составит $R = N - K = 7 - 5 = 2$..

В поле $GF(2^3)$ можно также построить код РС с $t = 2$ и $K = 3$, который будет иметь полином в виде

$$P(x) = (x + \alpha)(x + \alpha^2)(x + \alpha^3)(x + \alpha^4) =$$
$$= x^4 + \alpha^3 x^3 + \alpha^0 x^2 + \alpha^1 x + \alpha^3. \qquad (6.5)$$

Для поля GF(2^4) число ошибок, исправляемых кодом РС, может быть увеличено, так приведем примеры полиномов соответственно для t = 2, t = 3 и t = 4, т.е.

$$P(x) = x^4 + \alpha^{13}x^3 + \alpha^6x^2 + \alpha^3x + \alpha^{10}, \quad (6.6)$$

$$P(x) = x^6 + \alpha^{10}x^5 + \alpha^{14}x^4 + \alpha^4x^3 + \alpha^6x^2 + \alpha^9x + \alpha^6, \quad (6.7)$$

$$P(x) = x^8 + \alpha^{14}x^7 + \alpha^2x^6 + \alpha^4x^5 + \alpha^2x^4 + \alpha^{13}x^3 + \\ + \alpha^5x^2 + \alpha^{11}x + \alpha^6 . \quad (6.8)$$

Для (6.5) N =7 и К = 3, для (6.6) N = 15 и К = 11, для (6.7) N = 15, К = 9 и для (6.8) имеем, что N = 15, а К = 7.

6.2. Образующая и проверочная матрицы кода Рида – Соломона

Так как коды РС являются циклическими, то с целью обеспечения кодирования ими информации могут применяться соответствующие регистры сдвига. Рассмотрим здесь наряду с построением регистров также правила получения образующих матриц, которые будут далее использоваться совместно с проверочными матрицами для обеспечения процедуры декодирования кодов РС более наглядной и понятной без привлечения сложного математического аппарата.

Образующая матрица кода Рида–Соломона строится по образующему полиному P(x). Количество ее строк равно числу информационных символов К. Для ее формирования можно на вход регистра с обратными связями, предназначенного для кодирования, подать соответствующую последовательность символов, как и для двоичных циклических кодов. Отличием является то, что сдвиги для обычных кодов осуществляются в поле GF(2), а для кодов Рида–Соломона требуется производить их в поле GF(2^m), где m – размерность поля Галуа. Ввиду этого регистры сдвига, сумматоры и устройства умножения, стоящие в цепи обратной связи, должны обеспечивать вычисления с учетом этой особенности и

соответственно строиться при технической реализации как «многоярусные».

Для кода РС (7,5) с $P(x) = x^2 + \alpha^4 x + \alpha^3$ регистр с элементами в цепи обратной связи 1, α^4, α^3, обеспечивающий кодирование, а также формирование образующей матрицы, имеет вид (рис. 6.1), где 1 и 2 триггерные «многоярусные» ячейки.

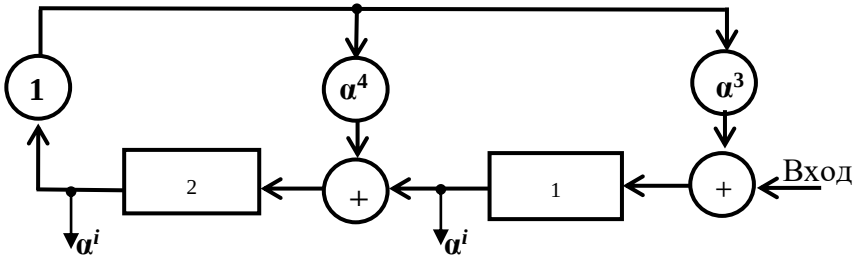


Рис. 6.1. Регистр сдвига с обратными связями для кода РС(7,5)

Для получения образующей матрицы на вход регистра (рис. 6.1) необходимо подать последовательность α^0нннннн, где α^0 = 1- элемент поля $GF(2^3)$, а н - нулевой элемент поля – 000. С учетом этого иногда удобнее представить входную последовательность в виде 1нннннн.

После записи в результате сдвигов единицы входной последовательности в крайнюю левую ячейку регистра (рис. 6.1) начинается построение элементов α^i образующей матрицы в течение последующих пяти тактов. Запись элементов поля в кружках означает умножение на них входящей последовательности. В результате будет получена правая часть образующей матрицы в виде

$$\begin{matrix} 4 \\ 3 \\ 2 \\ 1 \\ 0 \end{matrix} \begin{bmatrix} \alpha^4 & \alpha^3 \\ \alpha^0 & \alpha^0 \\ \alpha^5 & \alpha^3 \\ \alpha^5 & \alpha^1 \\ \alpha^4 & \alpha^1 \end{bmatrix}. \qquad (6.9)$$

С учетом присоединения к (6.9) единичной матрицы, получается полная образующая матрица кода РС (7,5), т.е.

$$G_{7,5} = \begin{bmatrix} \text{нннн1}\alpha^4\,\alpha^3 \\ \text{ннн1н}\alpha^0\,\alpha^0 \\ \text{нн1нн}\alpha^5\,\alpha^3 \\ \text{н1ннн}\alpha^5\,\alpha^1 \\ \text{1нннн}\alpha^4\,\alpha^1 \end{bmatrix} \begin{matrix} 4 \\ 3 \\ 2 \\ 1 \\ 0 \end{matrix}. \qquad (6.10)$$

Матрицу (6.10) также можно получить, осуществляя вычисления, аналогичные процедуре в регистре (рис. 6.1), т.е.

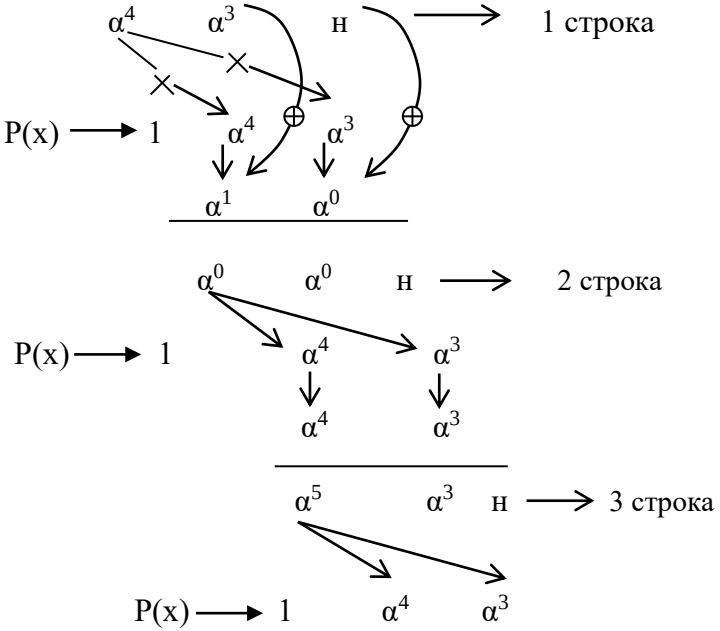
для получения второй строки образующей матрицы по рис. 6.2, где первая строка записана по коэффициентам полинома, коэффициент первой строки α^4 умножается на коэффициенты полинома α^4 и α^3, и один из результатов α^1 суммируется $\oplus$ с α^3, а итог α^0 переносится во вторую строку. Второй результат α^0 суммируется $\oplus$ с нулевым элементом поля н и итог также переносится во вторую строку. Аналогичные вычисления осуществляются и для других строк образующей матрицы.

Проверочная матрица кода Рида-Соломона строится по известной длине кода N и числу проверочных символов R, которое задает количество исправляемых кодом ошибок (6.3) и может быть представлена в виде [15]

N-1 N-2 N-3 … 1 0 $\rightarrow$ нумерация столбцов H^{Γ}

$$\mathbf{H}^{\Gamma} = \begin{bmatrix} \alpha^0 & \alpha & \alpha^2 & \dots & \alpha^{N-2} & \alpha^{N-1} \\ \alpha^0 & \alpha^2 & \alpha^4 & \dots & \alpha^{2(N-2)} & \alpha^{2(N-1)} \\ \alpha^0 & \alpha^3 & \alpha^6 & \dots & \alpha^{3(N-2)} & \alpha^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha^0 & \alpha^R & \alpha^{2R} & \dots & \alpha^{R(N-2)} & \alpha^{R(N-1)} \end{bmatrix}. \qquad (6.11)$$

Построение матрицы (6.11) напрямую связано с корнями минимальных многочленов $M_i(x)$ кодов Рида-Соломона [12]. Если для двоичных кодов БЧХ элементы первой строки проверочной матрицы возводилась в степени корней минимальных многочленов $m_i(x) = \alpha^3, \alpha^5, \alpha^7, \dots, \alpha^{d-2}$ (п. 4.5), то для кодов РС эти же элементы должны теперь возводится, в степени корней минимальных многочленов $M_i(x) = \alpha^2, \alpha^3, \alpha^4, \dots, \alpha^{2t=R}$. Таким образом, для кода, исправляющего одну ошибку, H^{Γ} (6.11) будет содержать две строки, для t = 2 – четыре строки и т.д.



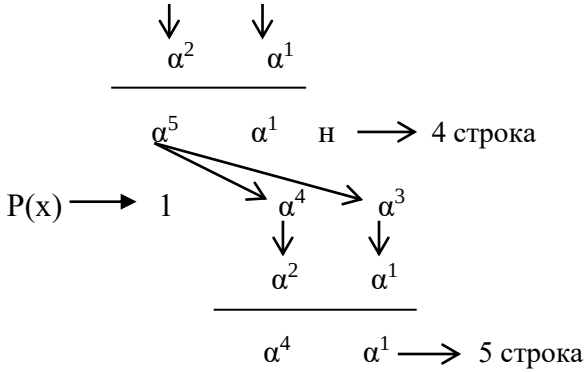


Рис. 6.2. Вычисления для получения образующей матрицы кода РС(7,5) с $P(x) = x^2 + \alpha^4 x + \alpha^3$

Для правильной процедуры кодирования и декодирования строки образующей и столбцы проверочной матрицы должны соответствующим образом нумероваться и быть согласованы с номерами позиций кодового слова.

Нумерация строк образующей матрицы должна быть такой, как в (6.10). Нумерация столбцов проверочной матрицы должна производиться как в (6.11). Номера позиций в кодовом слове указываются слева - направо, т.е.

0, 1 , 2, 3, …, N-2, N-1. (6.12)

6.3. Кодирование информации кодами Рида–Соломона

Обычно проведение кодирования осуществляют в регистре сдвига с обратными связями, построенном по соответствующему образующему полиному [1]. Так для проведения процедуры кодирования на вход регистра (рис.6.1) необходимо подать информационное слово, например 1нннн умноженное на x^2, т.е. 1нннннн и далее осуществлять запись. После записи единицы входной последовательности в крайнюю левую ячейку регистра в результате трех сдвигов, начнется формирование проверочных символов на последующих тактах. На седьмом такте зафиксируется

результат α^4 α^1 (6.9), который и будет являться проверочными символами, т. е. итогом деления информационного слова на образующий полином $P(x) = x^2 + \alpha^4 x + +\alpha^3$. Эту же процедуру можно осуществить, умножив информационное слово на образующую матрицу кода Рида – Соломона, осуществляя это по правилам алгебры [16]. Получение таким способом проверочных символов потребуется для объяснения далее процедур реализации декодирования кодом РС.

Приведем пример кодирования для кода РС (7,5) с t = 1, имеющего К = 5 с ипользованием образующей матрицы. В качестве образца информационного слова возьмем последовательность α^0 α^0 α^0 α^0 α^5 и умножим ее на образующую матрицу (6.10), представив их произведение как

$$\begin{matrix} 0 & 1 & 2 & 3 & 4 \\ \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^5 \end{matrix} \times \begin{bmatrix} \alpha^4 & \alpha^3 \\ \alpha^0 & \alpha^0 \\ \alpha^5 & \alpha^3 \\ \alpha^5 & \alpha^1 \\ \alpha^4 & \alpha^1 \end{bmatrix} \begin{matrix} 4 \\ 3 \\ 2 \\ 1 \\ 0. \end{matrix}$$

Результат умножения записывается в виде

$$\begin{matrix} \alpha^5 \cdot \alpha^4 & \alpha^5 \cdot \alpha^3 \\ \alpha^0 \cdot \alpha^0 & \alpha^0 \cdot \alpha^0 \\ \alpha^0 \cdot \alpha^5 & \alpha^0 \cdot \alpha^3 \\ \alpha^0 \cdot \alpha^5 & \alpha^0 \cdot \alpha^1 \\ \alpha^0 \cdot \alpha^4 & \alpha^0 \cdot \alpha^1 \end{matrix} \quad \text{или} \quad \begin{matrix} \alpha^2 & \alpha^1 \\ \alpha^0 & \alpha^0 \\ \alpha^5 & \alpha^3 \\ \alpha^5 & \alpha^1 \\ \alpha^4 & \alpha^1 \end{matrix}.$$

Далее осуществляем сложение последних столбцов по модулю 2, т.е.

$$\begin{array}{l} \alpha^2 \to 100 \\ \alpha^0 \to 001 \\ \oplus\ \alpha^5 \to 111 \\ \alpha^5 \to 111 \\ \underline{\alpha^4 \to 110} \\ \alpha^3 \to 011 \end{array}, \qquad \begin{array}{l} \alpha^1 \to 010 \\ \alpha^0 \to 001 \\ \oplus\ \alpha^3 \to 011 \\ \alpha^1 \to 010 \\ \underline{\alpha^1 \to 010} \\ \text{н} \to 000 \end{array}.$$

Полученные проверочные символы α^3 и н в дальнейшем добавляются к информационным символам и в результате образуется кодовое слово, направляемое в канал связи, т.е.

$$\underbrace{\alpha^0 \quad \alpha^0 \quad \alpha^0 \quad \alpha^0 \quad \alpha^5}_{\text{информационные символы}} \ \underbrace{\alpha^3 \quad \text{н}}_{\text{проверочные символы}}. \qquad (6.13)$$

Аналогичные процедуры по кодированию могут быть применены и к другим кодам Рида–Соломона с более высоким кодовым расстоянием.

6.4. Декодирование кодов Рида–Соломона

В отличие от двоичных кодов БЧХ у кодов Рида–Соломона символ кодового слова отображается не двумя возможными значениями 0 или 1, а значениями элементов поля Галуа. В представлении позиций символов кода РС говорят об ошибках расположения по символам кодового слова и об ошибках значения (искажения) в элементах поля Галуа.

Учитывая эту особенность, многочлен ошибок для кода РС может быть записан в виде [12]

$$E(x) = e_{n-1} \cdot x^{n-1} + e_{n-2} \cdot x^{n-2} + \ldots + e_1 \cdot x + e_0 \cdot 1. \qquad (6.14)$$

В (6.14) составляющие многочлена отражают не более t величин $e_{n-1},\ldots e_1$, e_0 отличных от нуля, фиксирующих искажения (ошибки значения) в символах кодового слова,

представленных элементами поля Галуа. Предположим теперь, что на самом деле произошло ν ошибок, $0 \le \nu \le t$, и что этим ошибкам соответствуют неизвестные позиции $i_1, i_2, \ldots i_\nu$. С учетом этого многочлен ошибок можно записать в виде

$$E(x) = e_{i_1} x^{i_1} + e_{i_2} x^{i_2} + \ldots + e_{i_\nu} x^{i_\nu}, \qquad (6.15)$$

где e_i – величина ошибки значения (искажения) элемента поля на позициях $i_1, i_2, \ldots i_\nu$ (в двоичном случае $e_i = 1$) .

Перед декодированием мы не знаем ни $i_1, \ldots i_\nu$, ни $e_{i1}, \ldots e_{i\nu}$, и для исправления ошибок нужно вычислить все этисоставляющие. Для получения компоненты синдрома S_1 надо найти значения многочлена (6.15) при $x = \alpha$, т.е.

$$S_1 = e_{i_1}\alpha^{i_1} + e_{i_2}\alpha^{i_2} + \ldots + e_{i_\nu}\alpha^{i_\nu} . \qquad (6.16)$$

Заменим теперь в (6.16) $e_{i1}, \ldots e_{i\nu}$, на величины $Y_1, Y_2, \ldots Y_\nu$, отражающие искажения, произошедшие в элементах поля Галуа, а $\alpha^{i1}, \alpha^{i2}, \alpha^{i3}, \ldots, \alpha^{i\nu}$ на $X_1, X_2, \ldots, X_\nu$, обозначив через них элементы поля Галуа, связанные с положением этих искажений.

В этих обозначениях S_1 запишется в виде

$$S_1 = Y_1X_1 + Y_2X_2 + \ldots + Y_\nu X_\nu .$$

Аналогично можно вычислить значения многочлена (6.15) при всех степенях α, входящих в P(x) кода РС (6.2), и тогда получим следующую систему из 2t уравнений относительно ν неизвестных локаторов (позиций) $X_1, \ldots, X_\nu$ и ν неизвестных искажений поля $Y_1, \ldots Y_\nu$, т.е.

$$\begin{cases} S_1 = Y_1X_1 + Y_2X_2 + \ldots + Y_\nu X_\nu, \\ S_2 = Y_1X_1^2 + Y_2X_2^2 + \ldots + Y_\nu X_\nu^2, \\ \vdots \\ S_{2t} = Y_1X_1^{2t} + Y_2X_2^{2t} + \ldots + Y_\nu X_\nu^{2t}. \end{cases} \qquad (6.17)$$

Система (6.17) отличается от аналогичной для двоичных кодов БЧХ (4.42) тем, что перед локаторами ошибок $X_1,\ldots, X_\nu$ стоят величины $Y_1,\ldots Y_\nu$, т.е. синдромы включают в себя теперь две составляющие ошибок.

Полученная система уравнений (6.17) является нелинейной, как и (5.1), и может быть решена тем же способом, что и для двоичных кодов БЧХ, т.е. через вспомогательный многочлен $\lambda(x)$, где обе части равенства (5.4) должны быть теперь умножены не на $X_\ell^{i+\nu}$, а на величину $Y_\ell \cdot X_\ell^{i+\nu}$.

В результате через решение системы уравнений, аналогичной (5.9), могут быть получены коэффициенты λ_1 ,… λ_ν и далее определены положения ошибок (локаторов) $X_1,\ldots, X_\nu$.

После подстановки полученных значений $X_1, X_2 ,\ldots, X_\nu$ в (6.17) образуется система линейных уравнений [12], где неизвестными величинами являются $Y_1,Y_2,\ldots Y_\nu$, и которая может быть записана в матричной форме, т.е.

$$\begin{bmatrix} X_1 & \ldots & X_\nu \\ \vdots & & \vdots \\ X_1^\nu & \ldots & X_\nu^\nu \end{bmatrix} \cdot \begin{bmatrix} Y_1 \\ \vdots \\ Y_\nu \end{bmatrix} = \begin{bmatrix} S_1 \\ \vdots \\ S_\nu \end{bmatrix}. \qquad (6.18)$$

Решение (6.18) может быть получено через обращение матрицы, т.е.

$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_\nu \end{bmatrix} = \begin{bmatrix} X_1 & \dots & X_\nu \\ \vdots & & \vdots \\ X_1^\nu & \dots & X_\nu^\nu \end{bmatrix}^{-1} \cdot \begin{bmatrix} S_1 \\ \vdots \\ S_\nu \end{bmatrix}. \qquad (6.19)$$

Для окончательного исправления ошибок в кодовом слове кода Рида–Соломона необходимо полученные значения $Y_1,\dots Y_\nu$ сложить по модулю 2 с теми его символами, которые соответствуют ранее определенным локаторам ошибок $X_1,\dots, X_\nu$.

Таким образом, код Рида–Соломона может быть декодирован аналогично двоичному коду БЧХ по алгоритму Питерсона–Горенстейна–Цирлера, представленного структурной схемой на рис. 5.1, где дополнительно должна быть вставлена еще одна процедура (6.19) после блока нахождения локаторов ошибок.

Приведем пример декодирования кода Рида-Соломона с образующим полиномом (6.4), исправляющего одну ошибку.

Возьмем в качестве кодового слова последовательность (6.13), т.е.

$$\underbrace{\alpha^0 \quad \alpha^0 \quad \alpha^0 \quad \alpha^0 \quad \alpha^5}_{\text{информационные символы}} \ \underbrace{\alpha^3 \quad \text{н}}_{\text{проверочные символы}},$$

которая направляется в канал связи, где она может быть искажена под воздействием помех.

Первым этапом процедуры декодирования в соответствии с рис. 5.1 является вычисление синдрома S. В случае отсутствия ошибок в кодовом слове синдром равен нулю, а при их наличии – отличен от нуля.

Будем считать, что в кодовом слове не возникло ошибок, и вычислим синдром. Предварительно запишем для проведения вычислений перечень элементов поля $GF(2^3)$ и проверочную матрицу H^Γ.

Перечень элементов поля $GF(2^3)$ с учетом нулевого элемента поля 000(н) будет иметь вид

$$
\begin{aligned}
\alpha^0 &\to 001\\
\alpha &\to 010\\
\alpha^2 &\to 100\\
\alpha^3 &\to 011\\
\alpha^4 &\to 110\\
\alpha^5 &\to 111\\
\alpha^6 &\to 101,
\end{aligned}
\qquad (6.20)
$$

где $\alpha^0 \to 001 \to 1 \to \alpha^7$, а проверочная матрица H^{Γ}, вычисленная по (6.11), записывается как

$$
H^{\Gamma}_{7,5} = \begin{bmatrix} \alpha^0 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \alpha^0 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \end{bmatrix}. \qquad (6.21)
$$

Для вычисления синдрома осуществим умножение кодового слова на проверочную матрицу H^{Γ} (6.21).

Расположим кодовое слово относительно проверочной матрицы в соответствии с указаниями (6.11) и (6.12) следующим образом:

$$
\begin{array}{ccccccc|c}
6 & 5 & 4 & 3 & 2 & 1 & 0 & \to \text{ позиции символов кодового слова} \\
\text{н} & \alpha^3 & \alpha^5 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \to \text{ кодовое слово}
\end{array}
$$

$$
H^{\Gamma}_{7,5} = \begin{bmatrix} \alpha^0 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \alpha^0 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \end{bmatrix} \to \; H^{\Gamma} \qquad (6.22)
$$

и произведем действия в соответствии с правилами [12] умножения вектора-столбца на матрицу, где фигурная скобка со знаком $\oplus$ относится к результатам осуществленных умножений, т. е.

$$\left.\begin{aligned} \text{н}\cdot\alpha^0 &= 000\\ \alpha^3\cdot\alpha &= \alpha^4 = 110\\ \alpha^5\cdot\alpha^2 &= \alpha^0 = 001\\ \alpha^0\cdot\alpha^3 &= \alpha^3 = 011\\ \alpha^0\cdot\alpha^4 &= \alpha^4 = 110\\ \alpha^0\cdot\alpha^5 &= \alpha^5 = 111\\ \alpha^0\cdot\alpha^6 &= \alpha^6 = 101 \end{aligned}\right\}\oplus \qquad S_1 = 000,$$

$$\left.\begin{aligned} \text{н}\cdot\alpha^0 &= 000\\ \alpha^3\cdot\alpha^2 &= \alpha^5 = 111\\ \alpha^5\cdot\alpha^4 &= \alpha^2 = 100\\ \alpha^0\cdot\alpha^6 &= \alpha^6 = 101\\ \alpha^0\cdot\alpha &= \alpha = 010\\ \alpha^0\cdot\alpha^3 &= \alpha^3 = 011\\ \alpha^0\cdot\alpha^5 &= \alpha^5 = 111 \end{aligned}\right\}\oplus \qquad S_2 = 000.$$

В результате получен нулевой синдром, что свидетельствует об отсутствии ошибок в кодовом слове, и оно может быть направлено потребителю.

При наличии ошибок в кодовом слове синдром S не будет равен нулю.

Будем считать, что ошибка произошла во второй позиции кодового слова, т.е.

$$\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & \rightarrow \text{позиции символов кодового слова}\\ \hline \alpha^0 & \alpha^0 & \alpha^6 & \alpha^0 & \alpha^5 & \alpha^3 & \text{н} & \rightarrow \text{кодовое слово} \end{array} \qquad (6.23)$$

В итоге, вместо элемента поля $\alpha^0 = 001$ в кодовом слове появится элемент $\alpha^6 = 101$.

Осуществим вновь вычисление синдрома S как умножение (6.23) на H^{Γ}, т.е.

$$\begin{array}{cccccccc} 6 & 5 & 4 & 3 & 2 & 1 & 0 & \rightarrow \text{позиции символов кодового слова}\\ \hline \text{н} & \alpha^3 & \alpha^5 & \alpha^0 & \alpha^6 & \alpha^0 & \alpha^0 & \rightarrow \text{кодовое слово с ошибкой} \end{array}$$

$$H^{\Gamma}_{7,5} = \begin{bmatrix} \alpha^0 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6\\ \alpha^0 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \end{bmatrix} \rightarrow H^{\Gamma}$$

$$\left.\begin{aligned} \text{н}\cdot\alpha^0 &= 000 \\ \alpha^3\cdot\alpha &= \alpha^4 = 110 \\ \alpha^5\cdot\alpha^2 &= \alpha^0 = 001 \\ \alpha^0\cdot\alpha^3 &= \alpha^3 = 011 \\ \alpha^6\cdot\alpha^4 &= \alpha^3 = 011 \\ \alpha^0\cdot\alpha^5 &= \alpha^5 = 111 \\ \alpha^0\cdot\alpha^6 &= \alpha^6 = 101 \end{aligned}\right\}\oplus \qquad \left.\begin{aligned} \text{н}\cdot\alpha^0 &= 000 \\ \alpha^3\cdot\alpha^2 &= \alpha^5 = 111 \\ \alpha^5\cdot\alpha^4 &= \alpha^2 = 100 \\ \alpha^0\cdot\alpha^6 &= \alpha^6 = 101 \\ \alpha^6\cdot\alpha &= \alpha^0 = 001 \\ \alpha^0\cdot\alpha^3 &= \alpha^3 = 011 \\ \alpha^0\cdot\alpha^5 &= \alpha^5 = 111 \end{aligned}\right\}\oplus .$$

$$S_1 = 101 = \alpha^6 \qquad S_2 = 011 = \alpha^3$$

Таким образом, в результате воздействия ошибки на кодовое слово, синдром получается не равным н, т.е. $S = \alpha^6\,\alpha^3$, где $S_1 = \alpha^6$ и $S_2 = \alpha^3$.

Здесь необходимо отметить тот факт, что полученные значения S_1 и S_2 не связаны зависимостью характерной для двоичных кодов БЧХ, когда $S_2 = S_1^2$.

Для устранения возникшей ошибки далее необходимо осуществить декодирование в соответствии со структурной схемой (рис. 5.1), учитывая особенности построения кодов Рида–Соломона. Приняв ν = t =1, находим детерминант матрицы синдромов M, построенный из определенных ранее величин синдромов. Для рассматриваемого кода РС (7,5), $M=[\alpha^6]$ и $\det(M)=6$, следовательно, необходимо осуществить следующий

этап декодирования – определение коэффициентов многочлена локаторов ошибок λ(x).

Применяемая для этого случая обратная матрица

$$M^{-1} = \left[\alpha^6\right]^{-1} = \frac{1}{\alpha^6} = \alpha^1,$$

а значение коэффициента $\lambda_1 = M^{-1}[S_2] = \alpha^1\alpha^3 = \alpha^4$.

С учетом этого многочлен локаторов ошибок записывается в виде

$$\lambda(x)=\lambda_1 \cdot x+1=\alpha^4 x+1. \qquad (6.24)$$

Корень выражения (6.24) определяется подстановкой в него вместо x элементов поля $GF(2^3)$ из (6.20) и равен α^3.

Местоположение ошибки в кодовом слове определяется через обратную величину от найденного корня, т.е. $X_1= 1/\alpha^3 = =\alpha^4$ и далее необходимо обратиться к (6.22). По значению $X_1=\alpha^4$ выбирается соответствующий элемент первой строки H^{Γ}, по относительному расположению которого со строкой номеров позиций символов кодового слова определяется, что ошибка находится во втором символе.

Теперь следует исправить в соответствующем символе возникшие искажения Y, которые находятся из (6.19), т.е.

$$Y_1=[x_1]^{-1}\cdot[S_1]=[\alpha^4]^{-1}\cdot[\alpha^6]=\frac{1}{\alpha^4}\alpha^6=\alpha^2\rightarrow 100.$$

Для коррекции кодового слова осуществляют сложение по модулю 2 пораженного ошибкой его символа с Y_1 и тогда

$$\alpha^6 \oplus \alpha^2 \rightarrow 101 \oplus 100 = 001 \rightarrow \alpha^0,$$

т.е. искажения исправлены и кодовое слово $\alpha^0\ \alpha^0\ \alpha^0\ \alpha^0\ \alpha^5\ \alpha^3$ н после исключения проверочных символов может быть направлено потребителю.

Приведем теперь пример декодирования кодом Рида–Соломона (7,3) с $P(x)=x^4+\alpha^3x^3+\alpha^0x^2+\alpha^1x+\alpha^3$ двух ошибок $(t=2)$.

Правая часть образующей матрицы кода, вычисленная по процедуре (рис. 6.2), будет иметь следующий вид

$$\begin{bmatrix} \alpha^3 & \alpha^0 & \alpha^1 & \alpha^3 \\ \alpha^2 & \alpha^0 & \alpha^6 & \alpha^6 \\ \alpha^4 & \alpha^0 & \alpha^4 & \alpha^5 \end{bmatrix}. \qquad (6.25)$$

Проверочная матрица H^{Γ} кода строится по правилу (6.11), т. е.

$$H^{\Gamma}_{(7,3)} = \begin{bmatrix} \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \alpha^0 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 & \alpha^5 \\ \alpha^0 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha^1 & \alpha^4 \\ \alpha^0 & \alpha^4 & \alpha^1 & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 \end{bmatrix}. \qquad (6.26)$$

В качестве информационной комбинации возьмем последовательность: α^1 , α^2, α^3.

После умножения ее на (6.25) получается следующее кодовое слово:

$$\frac{0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6}{\alpha^1 \quad \alpha^2 \quad \alpha^3 \quad \alpha^2 \quad \alpha^6 \quad \alpha^3 \quad \alpha^1}, \qquad (6.27)$$

где символы $\alpha^2\ \alpha^6\ \alpha^3\ \alpha^1$ – проверочные.

Введем две ошибки в кодовое слово (6.27). Первая ошибка в первой позиции символов (1), где элемент α^2 под воздействием помехи исказился в элемент α^6, а вторая ошибка в пятой позиции символов (5) с искажением помехой элемента α^3 в α^2. В результате последовательность (6.27) примет вид

$$\alpha^1\ \overset{*}{\alpha^6}\ \alpha^3\alpha^2\alpha^6\ \overset{*}{\alpha^2}\ \alpha^1, \qquad (6.28)$$

где знаком * помечены позиции символов с ошибками.

Проведем теперь декодирование в соответствии с алгоритмом Питерсона – Горенстейна–Цирлера (рис. 4.1).

Вычисление индексов компонент синдрома S осуществим как умножение (6.28) на (6.26), т.е.

$$\begin{array}{ccccccc} 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{array} \rightarrow \text{позиции символов}$$

$$\begin{array}{ccccccc} \alpha^1 & \alpha^{2^*} & \alpha^6 & \alpha^2 & \alpha^3 & \alpha^{6^*} & \alpha^1 \end{array} \rightarrow \text{кодовое слово}$$

$$\begin{bmatrix} \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \alpha^0 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 & \alpha^5 \\ \alpha^0 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha^1 & \alpha^4 \\ \alpha^0 & \alpha^4 & \alpha^1 & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 \end{bmatrix} \rightarrow H^{\Gamma} . \quad (6.29)$$

Осуществляя аналогичные процедуры как в (6.22), получим следующие составляющие синдрома: $S_1 = \alpha^1$, $S_2 = \alpha^1$, $S_3 = $ =н, $S_4 = \alpha^0$, или

$$S = \alpha^1\alpha^1 н\alpha^0. \quad (6.30)$$

Матрица синдромов, построенная по (6.30), будет иметь вид

$$M = \begin{bmatrix} \alpha^1 & \alpha^1 \\ \alpha^1 & н \end{bmatrix} \quad (6.31)$$

и $\det(M) = \alpha^1 \cdot н + \alpha^1 \cdot \alpha^1 = \alpha^2$.

Обратная матрица M^{-1} вычисляется по правилу (4.57), т.е.

$$M^{-1} = \frac{1}{\det(M)}\begin{bmatrix}\Delta_{11} & \Delta_{21}\\ \Delta_{12} & \Delta_{22}\end{bmatrix} = \frac{1}{\alpha^2}\begin{bmatrix}н & \alpha^1\\ \alpha^1 & \alpha^1\end{bmatrix} = \\ = \begin{bmatrix}н & \alpha^6\\ \alpha^6 & \alpha^6\end{bmatrix}. \qquad (6.32)$$

С учетом (5.32) находим коэффициенты многочлена локаторов ошибок в соответствии с рис. 4.1, т.е.

$$\begin{bmatrix}\lambda_2\\ \lambda_1\end{bmatrix} = M^{-1}\begin{bmatrix}S_3\\ S_4\end{bmatrix} = \begin{bmatrix}н & \alpha^6\\ \alpha^6 & \alpha^6\end{bmatrix}\cdot\begin{bmatrix}н\\ \alpha^0\end{bmatrix} = \\ = \begin{bmatrix}н\cdot н + \alpha^0\cdot\alpha^6\\ \alpha^6\cdot н + \alpha^0\alpha^6\end{bmatrix} = \begin{bmatrix}\alpha^6\\ \alpha^6\end{bmatrix}. \qquad (6.33)$$

Таким образом, получаем в соответствии с (6.33) $\lambda_1=\alpha^6$, $\lambda_2=\alpha^6$.

Многочлен локаторов ошибок для t = 2 записывается с учетом полученных коэффициентов как

$$\lambda(x) = \alpha^6\cdot x^2 + \alpha^6\cdot x + 1. \qquad (6.34)$$

Корни многочлена (6.34) после последовательной подстановки элементов поля $GF(2^3)$ получаются следующие: α^6, α^2.

Положение ошибок в позициях символов определяется как значения обратных величин от найденных корней, т.е.

$$X_1 = 1/\alpha^6 = \alpha^1, \qquad X_2 = 1/\alpha^2 = \alpha^5.$$

По значениям корней и с учетом (6.29) находим, что ошибки находятся в позициях символов кодового слова 1 и 5.

Искажения в символах 1 и 5 кодового слова находятся в соответствии с выражением (6.19) в виде

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 \\ X_1^2 & X_2^2 \end{bmatrix}^{-1} \cdot \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} =$$

$$= \begin{bmatrix} \alpha^1 & \alpha^5 \\ \alpha^2 & \alpha^3 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \alpha^1 \\ \alpha^1 \end{bmatrix} = \begin{bmatrix} \alpha^0 \\ \alpha^5 \end{bmatrix},$$

т.е. $Y_1=\alpha^0$ и $Y_2=\alpha^5$.

Для окончательного исправления ошибок в кодовой комбинации

$$\alpha^1\ \alpha^{6*}\alpha^3\alpha^2\alpha^6\alpha^{2^*}\alpha^1$$

необходимо сложить полученные искажения Y_1 и Y_2 с элементами α^{6*}и α^{2*}, т.е.

$$\alpha^{6^*} \oplus \alpha^0 = \alpha^2,$$

$$\alpha^{2^*} \oplus \alpha^5 = \alpha^3.$$

Полученные значения элементов поля α^2 и α^3 соответствуют позициям символов 1 и 5 кодового слова (6.27) до его передачи в канал связи. На этом декодирование кода Рида–Соломона (7,3) завершается.

6.5 Лабораторное задание

1. Осуществить кодирование информации кодами Рида-Соломона. Для чего выполнить следующие пункты:

1.1. Включить ЭВМ и войти в диск Z. Выбрать файл LABRNEW и открыть его. Затем открыть файл LAB6Win и далее файл labr6 [29] (рис. 6.3).

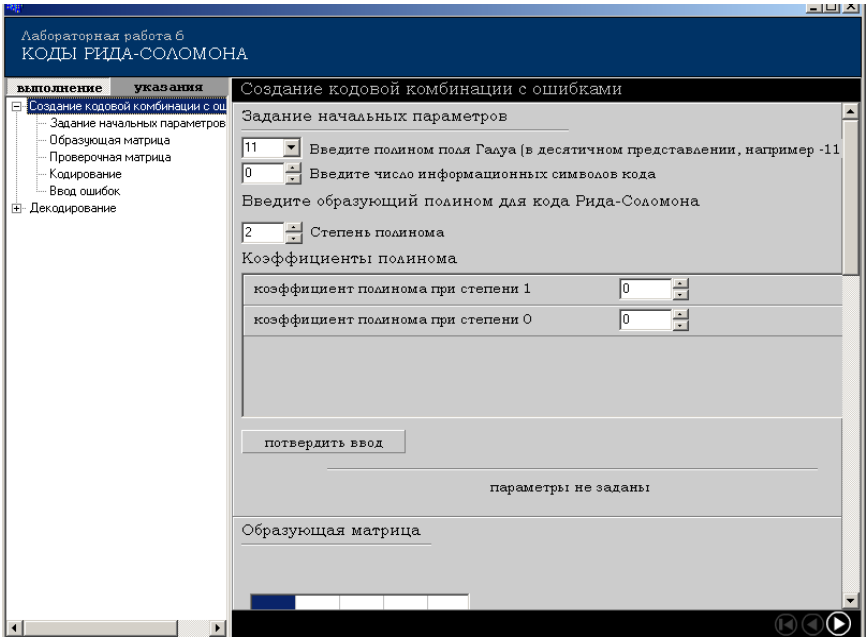


Рис. 6.3. Заставка для исследования кодов Рида-Соломона

1.2. Из заголовка «Создание кодовой комбинации с ошибками» (рис. 6.3) выбрать надпись «Задание начальных параметров». Ввести полином поля Галуа $GF\left(2^3\right)$ в десятичном представлении, т. е. P(x) $\rightarrow$ 11, что соответствует P(x) = =x^3 + x +1. Для кода Рида-Соломона (7,5) с N = 7 и K = 5, исправляющего одну ошибку (t = 1), ввести число информационных символов K=5 и образующий полином $P(x) = x^2 + \alpha^4 x + \alpha^3$, где степень полинома 2, коэффициент полинома при степени 1 равен α^4 (ввести 4) и коэффициент полинома при степени 0 равен α^3(ввести 3). В дальнейшем, будем условно обозначать для удобства работы с программой коэффициенты полиномов через их степени. Подтвердить ввод, удостоверившись по надписи «Параметры заданы».

1.3. Открыть из заголовка надпись «Образующая матрица» и записать в отчет образующую матрицу кода РС (7,5), проставив нумерацию ее строк. Отразить в отчете порядок ее построения.

1.4.Аналогично открыть надпись «Проверочная матрица» и записать в отчет проверочную матрицу для кода РС (7,5), проставив нумерацию ее столбцов. Отразить в отчете порядок ее построения.

1.5. Открыть надпись «Кодирование» и осуществить кодирование информации кодом РС (7,5). Для подготовки проведения процедуры кодирования записать в отчет элементы поля Галуа $GF\left(2^3\right)$ в виде индексов степеней α и двоичных чисел (см. стр.161). Записать в тетрадь, выбранную Вами произвольную информационную комбинацию в виде единиц и нулей, соответствующую пяти символам кода РС (7,5), т.е. 15 двоичным символам. Разбить выбранную последовательность на элементы поля $GF\left(2^3\right)$ (3 символа) и представить полученные двоичные числа в виде индексов степеней элементов поля $GF\left(2^3\right)$, нулевой элемент поля в виде трех нулей записывать числом (-1), которое при вводе переобозначается программой в букву «н». Ввести полученные индексы элементов поля, соответствующие информационной части кода РС (7,5), в заставку «Кодирование» и подтвердить ввод. Записать результат вычисления кодовой комбинации кода РС (7,5) в виде индексов элементов поля Галуа и соответствующих им двоичных чисел.

Кодирование осуществляется как умножение кодовой комбинации информационных символов на образующую матрицу (см. теоретическую часть лабораторной работы стр.157). Расписать эту процедуру для выбранных ранее информационных символов при оформлении отчета.

1.6. Открыть заставку «Ввод ошибок» и ввести одну ошибку в произвольную позицию символа кода РС (7,5) с

произвольным искажением символа значением элемента поля Галуа $GF\left(2^3\right)$. Подтвердить ввод. Сравнить полученную кодовую комбинацию с комбинацией из п. 1.5.

2. Осуществить декодирование кода РС (7,5) с введенной ошибкой в кодовую комбинацию. Для этого выполнить следующие пункты:

2.1. Для вычисления компонент синдрома открыть надпись «Декодирование» и далее надпись «Вычисление компонент синдрома». Их вычисление осуществляется как произведение кодовой комбинации с ошибками в позициях символов кодового слова и искажениями в них на образующую матрицу кода Рида-Соломона.

Расписать при оформлении отчета порядок образования компонент синдрома S_1 и S_2, используя кодовую комбинацию с ошибками и полученную ранее проверочную матрицу (см. стр. 163). Сравнить результаты вычислений.

2.2. Для построения матрицы синдромов открыть надпись «Матрица синдромов». Матрица синдромов строится по образцу (см. стр.: 138, 164, 167). Записать матрицу в отчет и зафиксировать порядок ее построения.

2.3. Для вычисления определителя матрицы открыть надпись «Определитель матрицы синдромов». Зафиксировать результат вычислений и число ошибок в кодовой комбинации.

2.4. Открыть надпись «Отчет по способу декодирования». Зафиксировать в отчет порядок определения номера позиции символа с ошибкой и искажением в символе, представленном соответствующим элементом поля Галуа.

Отразить в отчете вычисление коэффициентов многочлена локатора ошибок $\lambda(x)$ (см. стр. 137, 164, 167).

Зафиксировать в отчет порядок вычисления ошибок в позициях символов кодового слова и определения искажений Y_i.

Записать результаты декодирования.

2.5. Используя п. 1 и п. 2 осуществить кодирование и декодирование информации кодом Рида-Соломона (7,3) с полиномом $P_x = x^4 + a^3x^3 + \alpha^0x^2 + \alpha^1x + \alpha^3$, исправляющим две ошибки t = 2, и числом информационных символов К=3 в поле $GF\left(2^3\right)$, образованном P(x) $\rightarrow$ 11 (десятичное представление полинома). Три информационных символа взять из п. 1.5.

Зафиксировать в отчете образующую матрицу, проверочную матрицу, кодовое слово с проверочными символами, число ошибок и их расположение в кодовом слове, значения синдромов, матрицу синдромов, значения определителя, обратную матрицу, многочлен локаторов ошибок $\lambda(x)$, номера символов с ошибками и соответствующими искажениями в них.

Используя пример исправления ошибок кодом РС (7,3) из теоретической части (см. стр. 165), отразить в отчете процедуру вычислений при кодировании и декодировании с конкретными значениями ошибок. Сравнить результаты с процедурой, осуществленной ранее.

2.6. Используя п. 1 и п. 2 осуществить кодирование и декодирование информации в поле $GF\left(2^4\right)$, образованном полиномом P(x) $\rightarrow$19 (десятичное представление полинома) для кодов Рида-Соломона с образующими полиномами:

$\mathrm{P}(x) = x^4 + \alpha^{13}x^3 + \alpha^6x^2 + \alpha^3x + \alpha^{10}$ с К= 11, N = 15 и t = 2;

$\mathrm{P}(x) = x^6 + \alpha^{10}x^5 + \alpha^{14}x^4 + \alpha^4x^3 + \alpha^6x^2 + \alpha^9x + \alpha^6$

с К = 9. N = 15 и t = 3;

$P(x) = x^8 + \alpha^{14}x^7 + \alpha^2x^6 + \alpha^4x^5 + \alpha^2x^4 + \alpha^{13}x^3 + \alpha^5x^2 + \alpha^{11}x + \alpha^6$

с К = 7, N = 15, t = 4.

Информационные символы формировать аналогично п. 1.5.

Зафиксировать в отчете образующие матрицы, проверочные матрицы, кодовые комбинации с проверочными

символами, число ошибок и их расположение в кодовом слове, значения синдромов, матрицы синдромов, значения определителей, обратные матрицы, многочлены локаторов ошибок $\lambda(x)$, номера позиций символов с ошибками и соответствующими искажениями в них.

Содержание отчета

1. Количественные показатели по выполнению конкретных пунктов с их комментариями и выводами.
2. Графики и таблицы по проведенным измерениям.
3. Краткие выводы по лабораторной работе.

Контрольные вопросы

1. Чем отличается код Рида-Соломона от кода БЧХ?
2. Как рассчитывается образующая матрица кода Рида-Соломона?
3. Как технически осуществляется построение образующей матрицы кода Рида-Соломона?
4. Приведите пример получения проверочных символов кода Рида-Соломона. Как осуществляется эта процедура?
5. Как осуществляется построение проверочной матрицы кода Рида-Соломона и для чего она используется?
6. Как определяется общая длина кода Рида-Соломона в битах?
7. Приведите пример образования компонент синдрома кода Рида-Соломона, как это осуществляется?
8. Чем отличается алгоритм декодирования ПГЦ для кодов Рида-Соломона?

7. ЛАБОРАТОРНАЯ РАБОТА № 7

СВЕРТОЧНЫЕ КОДЫ

Цель работы: Изучение кодирования и декодирования информации сверточными кодами

Теоретическая часть

7.1. Введение в сверточное кодирование

Название «сверточные» коды получили по причине того, что процесс кодирования ими входной информационной последовательности, т.е. получения выходных символов, с точки зрения математики, отражается процедурой, называемой сверткой [31].

В отличие от блоковых кодов, где закодированные символы становятся последовательностью независимых кодовых слов одинаковой длины, при использовании сверточных кодов ситуация другая. Дополнительные символы для таких кодов зависят от ряда предшествующих информационных символов, и выходная последовательность становится одним полубесконечным кодовым словом. Это накладывает свои особенности на технические решения получения сверточных кодов и разработку методов их кодирования и декодирования.

Сверточные коды, как правило, получают методом моделирования на ЭВМ, путем просмотра большого числа образующих их многочленов и подбора лучших из них, обеспечивающих получение сверточных кодов с хорошими характеристиками.

Первоначально сверточные коды появились как альтернатива блоковым кодам для упрощения процедур кодирования и декодирования с технических позиций при

исправлении одиночных ошибок [20]. Однако в дальнейшем ученые обратили внимание на возможность получения на основе сверточных кодов более высоких показателей по помехоустойчивости, чем для блоковых кодов. Это связано с тем, что для сверточных кодов, применяемых с алгоритмом декодирования Витерби [30,31], можно достаточно просто создать декодер в расчете на использование демодулятора с мягким решением, увеличивающим выигрыш примерно на 2 дБ по сравнению с демодулятором с жестким решением, который в основном применяется в случае блоковых кодов. Таким образом, введение сверточной структуры наделяет код рядом дополнительных свойств, которые облегчают его декодирование и улучшают характеристики.

7.2. Кодирование информации сверточными кодами

Кодирование информации сверточным кодом является более простой процедурой, чем для блоковых кодов, и выполняется в регистре сдвига, у которого отдельные ячейки соединены через сумматоры по модулю 2.

Изучение сверточных кодов начнем с рассмотрения кодера, представленного на рис. 7.1.

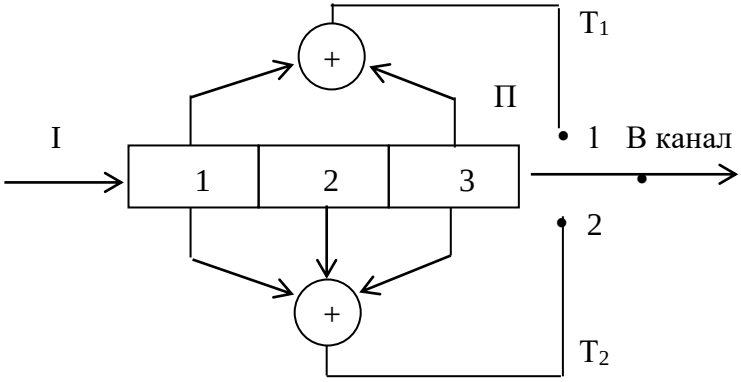


Рис. 7.1. Структурная схема кодера сверточного кода

Входная двоичная информационная последовательность I поступает на вход регистра сдвига с ячейками 1,2,3 и на выходе сумматоров формируются двоичные последовательности T_1 и T_2. Сначала переключатель П находится в положении 1, а затем переводится в положение 2.

Поступление данных в канал при такой коммутации удваивается, т.е. на один входной информационный символ кодер формирует на выходе два кодовых символа.

Так, например, пусть на вход кодера подается последовательность 100. В соответствии со связями между ячейками регистра на рис. 7.1, выходная двоичная последовательность кодера складывается из двоичных последовательностей символов T_1 и T_2, поступающих с выходов сумматоров на ключ П и образуется в виде последовательности 110111, которая состоит из шести символов вместо трех входных. Структура выходной последовательности определяется связями регистра с сумматорами. Так для связей (рис. 7.2) выходная последовательность при той же входной 100 будет 100111, и будет содержать меньшее количество единиц, чем для кодера по рис. 7.1.

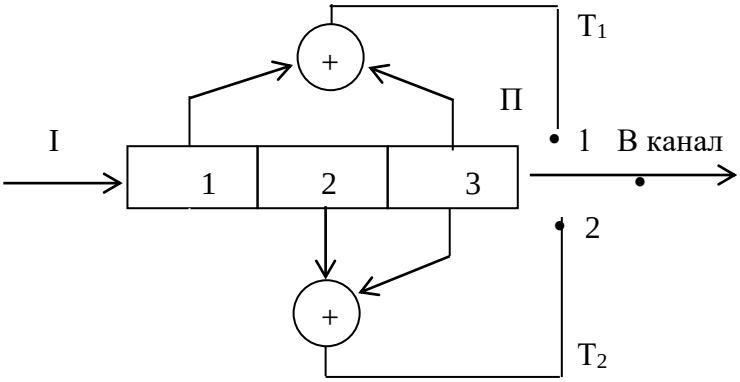


Рис. 7.2. Сверточный кодер

Роль соединений между ячейками регистра и сумматорами будет выяснена далее. Стоит только сказать, что эти связи имеют существенное значение при построении сверточных кодов.

В выходной последовательности сверточного кодера (рис. 7.1), информационные символы в явном виде не содержатся, т. к. ни T_1, ни T_2 не совпадают с I, и такое кодирование называется несистематическим. Таким же несистематическим будет и кодирование для схемы кодера представленного на рис. 7.4. Для их выделения из общего потока данных на приемной стороне должно стоять устройство, обеспечивающее оценку информационных символов. Таким устройством обычно является декодер сверточного кода, например декодер Витерби, в котором после определенных преобразований осуществляется оценивание информационных символов.

Для получения систематического сверточного кода необходимо изменить подключение одного из сумматоров (рис.7.1), чтобы информационная последовательность была частью выходной последовательности, что эквивалентно отключению одного из сумматоров, как это показано на рис. 7.3.

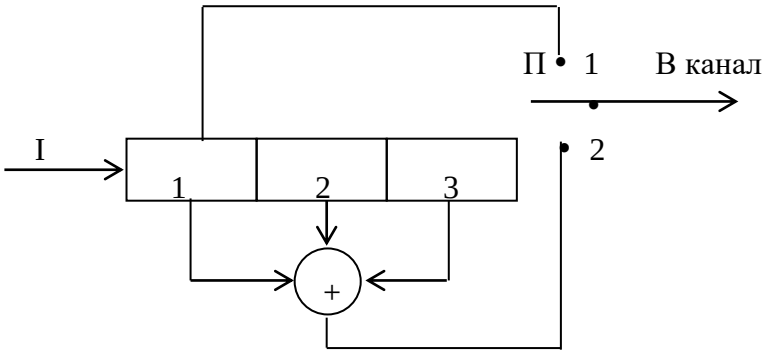


Рис. 7.3. Кодер для систематического кодирования

В сверточных кодах вводится понятие кодового ограничения K_o, под которым понимается число ячеек памяти регистра сдвига плюс его входная ячейка, которая в регистре не является ячейкой памяти, а только интерпретатором очередного значения символа из набора k_o входных информационных символов I. Относительная скорость кода определяется формулой $R=k_o/n_o$, где n_o- набор выходных символов кодера, поступающих с выходов его сумматоров в канал при входном наборе символов k_o. Для схемы (рис. 7.1) $K_o=3$, а $R = 1/2$, при наличии в ней одного трехразрядного регистра сдвига и двух сумматоров. При двух регистрах сдвига и трех сумматорах в схеме кодера, представленного на рис. 7.4, для каждых двух информационных символов образуется три символа канала по одному для T_1, T_2 и T_3. В таком кодере относительная скорость $R = 2/3$, а $K_o= 4$.

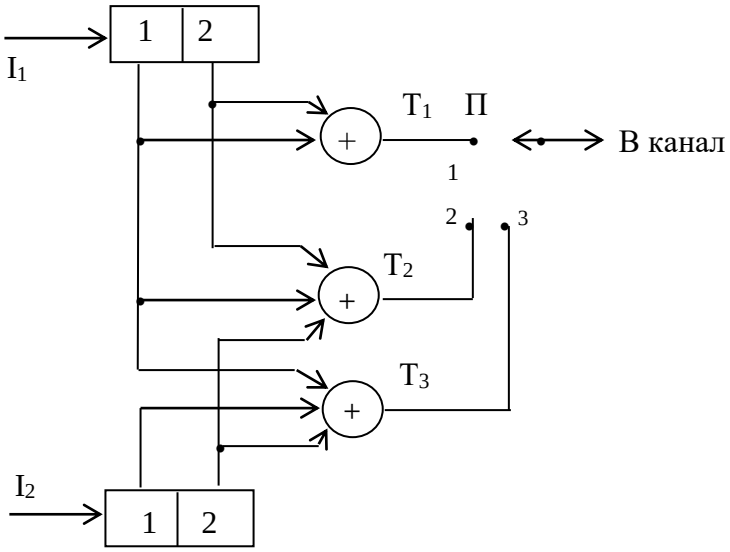


Рис. 7.4. Кодер сверточного кода для скорости R = 2/3

Как видно из представленных схем кодеров сверточных кодов, их схемная реализация не вызывает затруднений и намного проще чем для блоковых кодов.

Связь между ячейками регистров сдвига и сумматорами удобно описывать порождающими многочленами (полиномами связи) и ими же задавать сверточный код. Это одна из форм задания сверточного кода [1]. Другие формы, такие как древовидная и решетчатая диаграммы будут рассмотрены далее применительно к пояснению алгоритма Витерби.

Например, для рис. 7.1 верхний и нижний сумматоры с их связями можно отразить соответственно многочленами $q_1(x)= =1 + x^2$ и $q_2(x) =1 + x + x^2$, где степени растут слева направо, так что самая левая ячейка соответствует свободному члену, равному 1. Информационную двоичную последовательность также можно представить в виде степенного ряда

$I(x) = i_0 + i_1 x + i_2 x^2 + i_3 x^3 + \ldots$, где i_j – j - й информационный символ (равный 0 или 1).

Исходя из этого, двоичные последовательности символов на выходах сумматоров (рис. 7.1) T_1и T_2 могут быть представлены в виде $T_1(x) = I(x) \cdot q_1(x)$, $T_2(x) = I(x) \cdot q_2(x)$. Объединение этих последовательностей образует выходную последовательность кодера.

Если на вход кодера (рис. 7.1) подать последовательность, у которой один символ 1, за которым следуют символы 0 (I(x) = 1), то на его выходе образуется последовательность 1101110000…, называемая импульсной характеристикой кодера. Значащая часть импульсной характеристики ограничена и формируется прохождением символа 1 через регистр кодера. Выходная последовательность, соответствующая произвольной входной последовательности получается сложением по модулю 2 сдвигов импульсной характеристики [1].

С учетом этого образующая (порождающая) матрица сверточного кода может быть представлена в виде соответствующих сдвигов импульсной характеристики, т.е. для кодера по рис. 7.1 получаем, что

$$G = \begin{bmatrix} 1\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\ldots \\ 0\,0\,1\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\ldots \\ 0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\ldots \end{bmatrix}. \qquad (7.1)$$

Сдвиги импульсной характеристики осуществляются на число символов, равное числу сумматоров, с которых снимается выходная последовательность кодера, т.е. в данном случае (7.1) для кодера (рис. 7.1) на 2 символа.

Число строк образующей матрицы сверточного кода не ограничивается, но для работы с ней необходимо хранить только первую строку, которая является импульсной характеристикой кодера. При этом выходная последовательность, соответствующая произвольной входной последовательности I(x), может быть получена путем линейного сложения строк матрицы G (7.1), номера которых соответствуют номерам символов 1 во входной последовательности кодера. Например, в схеме кодера (рис. 7.1) при входной последовательности

$$I(x) \rightarrow 1010000000\ldots\ldots \qquad (7.2)$$

выходная последовательность получается с учетом того, что первый и третий символы (7.2) это единицы. В связи с этим необходимо взять первую и третью строки образующей матрицы (7.1) и сложить их по модулю 2, т.е.

$$\begin{array}{ll} & 1\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\ldots \\ \oplus & \underline{0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,0\,0\ldots} \\ & 1\,1\,0\,1\,0\,0\,0\,1\,1\,1\,0\,0\ldots. \end{array}$$

Сверточный код строят исходя из заданных порождающих многочленов (полиномов связи). Значения этих многочленов приводятся в специальной литературе, например

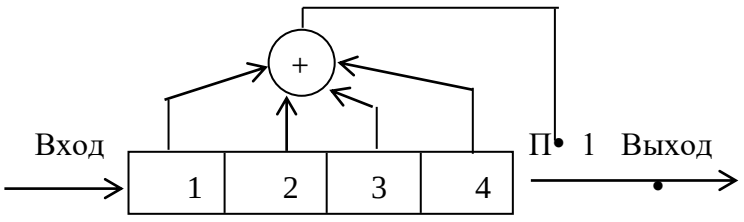
[9,31]. Здесь, как правило, многочлены (числа) записываются в так называемой восьмеричной форме и приводятся в таблицах.

Так, например, для первой строки из таблицы записываем число 5 в виде 101, а число 7 в виде 111 и соответственно получаем, что $q_1(x) = 1+x^2$ и $q_2(x) = 1+x+x^2$.

Порождающие многочлены сверточных кодов

Кодовое ограничение K_o	Порождающий многочлен q (x)	Свободное кодовое расстояние $d_{св}$
3	5; 7 (101; 111)	5
4	17; 13 (1111; 1011)	6
5	27; 31 (10111;11001)	7

Для второй строки таблицы имеем, что представление числа 17→ 001.111, где правая триада соответствует десятичному числу 7. Аналогично для второго числа 13 → 001.011 имеем две триады, по которым, как и для первого числа, осуществляется запись многочленов. По порождающим многочленам q(x) строят кодирующие схемы для сверточных кодов. Сумматоры со связями по $q_1(x)$ и q_2 (x) располагают обычно для наглядности сверху и снизу регистра. Очередность их расположения роли не играет, но должна быть учтена при декодировании. Так, например, для первой строки из таблицы кодер будет соответствовать рис. 7.1, а для второй строки схема будет иметь вид, представленный на рис. 7.5.



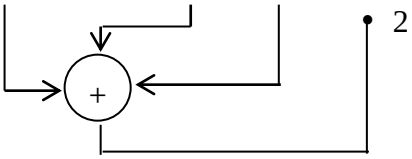


Рис. 7.5. Кодер для порождающих многочленов (17; 13)

Корректирующая способность сверточного кода зависит от так называемого свободного кодового расстояния $d_{св}$, которое по существу содержит ту же информацию о коде, что и кодовое расстояние для блоковых кодов. Это расстояние можно определить, используя различные виды представления сверточного кода [1]. При полиномиальном представлении значение свободного кодового расстояния $d_{св}$ находится сразу по его импульсной характеристике. Такая характеристика определяется как выходная последовательность кодера при подаче на его вход последовательности, у которой один символ 1, за которым следуют символы 0. Суммарное количество единиц в импульсной характеристике и определяет $d_{св}$.

Так, для образующей матрицы (7.1) сверточного кода с кодовым ограничением $K_o = 3$ получается, что суммарное количество единиц в ее первой строке как и в других строках, представляющих собой сдвиги импульсной характеристики, равно пяти, то и $d_{св} = 5$. Сверточный кодер с $d_{св} = 7$ можно построить по полиномам $q_1(x) \rightarrow 10111$ и $q_2(x) \rightarrow 11001$. Схема кодера для этого случая представлена на рис. 7.6.

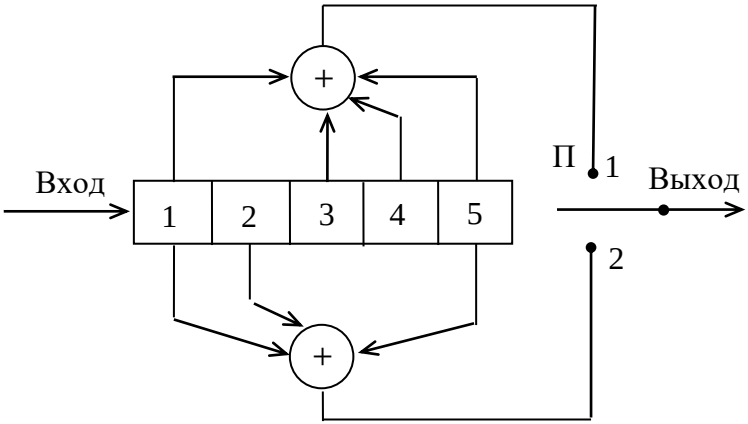


Рис. 7.6. Кодер для $q_1(x) \rightarrow 10111$, $q_2(x) \rightarrow 11001$

Образующая матрица для этого кода может быть построена аналогично матрице G (7.1) и будет иметь следующий вид:

$$G = \begin{bmatrix} 1\,1\,0\,1\,1\,0\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,\ldots \\ 0\,0\,1\,1\,0\,1\,1\,0\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,\ldots \\ 0\,0\,0\,0\,1\,1\,0\,1\,1\,0\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,\ldots \\ 0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,0\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,\ldots \\ 0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,0\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,\ldots \end{bmatrix} \quad (7.3)$$

В матрице G (7.3) в первой строке, которая также представляет собой импульсную характеристику сверточного кода, суммарное количество единиц равно семи и отсюда $d_{св} = 7$.

7.3. Декодирование сверточных кодов

7.3.1. Алгоритм Витерби

А.Д. Витерби [31] был разработан и проанализирован алгоритм, декодирования сверточных кодов, в котором сверточная структура кода была использована для построения декодера максимального правдоподобия с пониженной сложностью и улучшенными характеристиками, по сравнению с классическими декодерами максимального правдоподобия, применявшимися до этого для блоковых и сверточных кодов. Алгоритм Витерби основан, прежде всего, на представлении сверточного кода в виде так называемой «решетки» сверточного кода, построение которой осуществляется по его древовидной диаграмме (дереву), которая строится, в свою очередь, в соответствии со схемой кодера сверточного кода [19].

Построим древовидную диаграмму (рис. 7.7) для сверточного кода, образованного полиномами $q_1(x)=1+x^2$ и $q_2(x)= =1+x+x^2$, соответствующая схема кодера на основе этих полиномов была представлена на рис. 7.1.

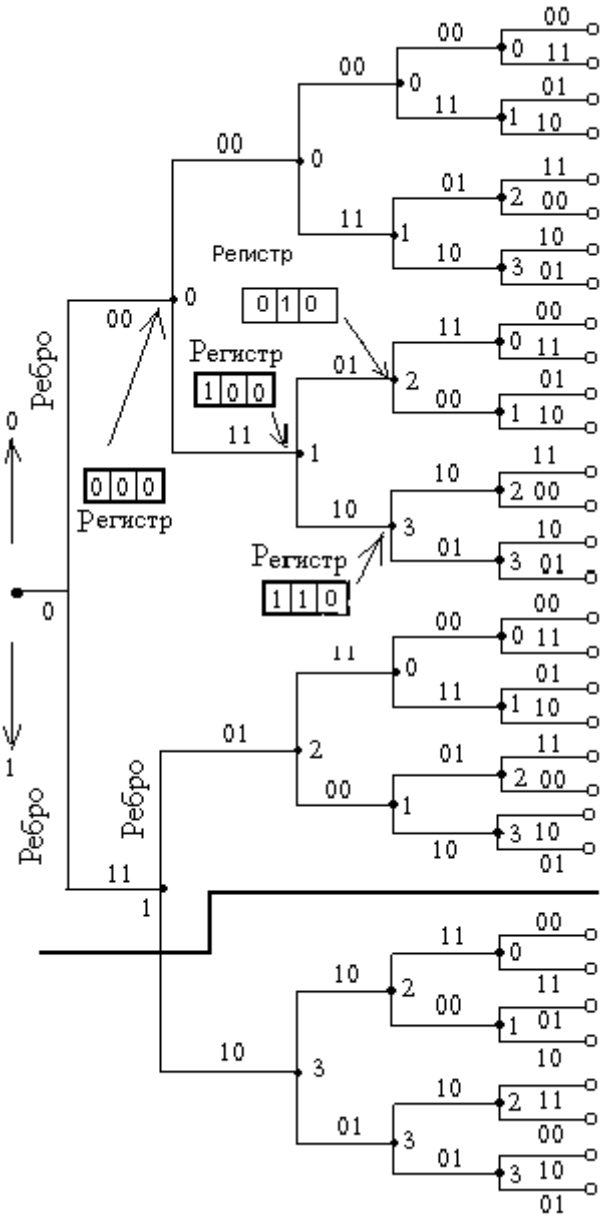


Рис. 7.7. Древовидная диаграмма для сверточного кода с кодовым ограничением $K_o = 3$

При построении дерева каждому его ребру соответствует некоторый входной символ кодера. При этом входной символ 0 соответствует верхнему ребру, а 1 - нижнему.

Вертикальные линии на дереве образуют ребра, а горизонтальные линии соответствуют выходным символам кодера (указаны над линией) и образуют путь.

Таким образом, каждая входная последовательность задает некоторый путь на дереве. В частности, входная последовательность 10110 задает выходную последовательность 1101001010. Соответствующий путь (жирная линия) показан в нижней части рис. 7.7. Таким образом, на диаграмме (рис. 7.7) можно отобразить все выходные последовательности, соответствующие 32 возможным комбинациям из пяти входных двоичных символов.

При росте длины входной последовательности число возможных путей растет экспоненциально, так что наличие такой диаграммы лишь облегчает понимание работы кодера сверточного кода.

Однако экспоненциальный рост объема можно прекратить, если заметить, что для структуры дерева сверточного кода характерна периодичность. На рис.7.7 каждому узлу кодового дерева принадлежат числа от 0 до 3 в соответствии с содержимым двух левых ячеек кодового регистра в данном узле дерева (номер младшего разряда слева). Числа в двух ячейках отражают состояние кодера. Видно, что элементы дерева, выходящие из любых двух узлов, характеризуемых одинаковым состоянием, - одинаковы. Например, верхнее и нижнее поддеревья на уровне 1. То же происходит и с любой другой парой узлов, помеченных одинаковыми цифрами. Причину этого легко установить, обратившись к структуре кодера (рис.7.1). Когда четвертый входной символ вводится в кодер, первый входной символ покидает последний элемент регистра, и поэтому в дальнейшем уже не может оказывать никакого влияния на

выходные символы кодера. Следовательно, например, информационным последовательностям 100xy… и 000xy…, где крайний левый бит, является самым ранним, соответствуют кодовые последовательности, полностью совпадающие на всех ребрах после третьего. Таким образом, узлы, помеченные на дереве одинаковыми цифрами, могут быть объединены в один. Это приводит к тому, что дерево (рис. 7.7) переходит в диаграмму, изображенную на рис. 7.8. Новая диаграмма называется решетчатой диаграммой (решеткой). Это древовидная структура с объединенными узлами 0,1,2,3, которые отмечены здесь как состояние 00, 01, 10, 11 .

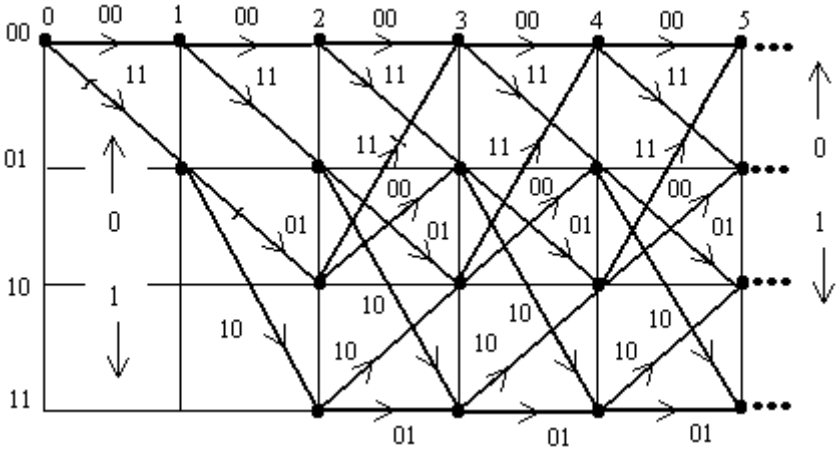


Рис. 7.8. Решетка для сверточного кода

Как и на рис. 7.7, на рис. 7.8 символ 0 соответствует верхнему ребру, выходящему из узла, а 1 – нижнему. Как и ранее, всякая входная последовательность соответствует некоторому пути на решетке. Например, входная последовательность 10110 дает выходную

последовательность 1101001010, что совпадает с результатом, отраженным на рис. 7.7.

Особенность решетчатого представления состоит в том, что с ростом числа входных символов число узлов в решетке не растет, а остается равным 2^{K_0-1}, где K_o - длина кодового ограничения. Это обусловлено тем, что избыточные части кодового дерева (рис. 7.7) отождествляются (сливаются).

По решетке (рис.7.8) можно также определить $d_{св}$, которое равно минимальному весу наиболее короткого пути (выходной последовательности), начинающегося в нулевом состоянии и заканчивающегося также в нулевом состоянии и не попадающего в это состояние в промежуточные моменты [19]. Один из таких путей в виде перечеркнутых ребер (11 01 11) показан на рис.7.8. Вес такого пути равен суммарному количеству единиц и свободное расстояние сверточного кода $d_{св} = 5$, т.е. код может исправлять двукратные ошибки.

При использовании решетки для декодирования сверточного кода вначале число путей растет экспоненциально с ростом длины входной последовательности. Однако вскоре появляется возможность исключить из рассмотрения такое число путей в каждом узле, которое в точности уравновешивает число вновь порожденных путей.

Процедура, позволяющая достичь такого эффекта за счет применения последовательности определенных вычислений, получила название алгоритма Витерби. Здесь сравниваются два пути, входящие в каждый узел, и сохраняется лишь тот из них, метрика (накопленное расстояние Хемминга) которого лучше. Другой путь исключается из рассмотрения, поскольку при любых принятых впоследствии данных его правдоподобие не сможет превзойти правдоподобия оставшегося пути. Оставшиеся пути называются выжившими. Для кода с $K_o=3$ в каждый момент будет сохраняться не более четырех путей.

Покажем, как происходит декодирование алгоритмом Витерби, на основе решетки сверточного кода (рис.7.8) [7].

Предположим, что передавалась нулевая последовательность, а принятая последовательность имеет вид

10 00 10 00 00… (7.4)

Поскольку свободное кодовое расстояние рассматриваемого кода равно 5, то все двукратные ошибки исправляются.

Берем за основу решетку кода, изображенную на рис. 7.8. В качестве принятой информации берем пары символов из последовательности (7.4). По мере поступления входных символов строим последовательно решетку с учетом расстояния Хемминга между принятой последовательностью и кодовыми символами из решетки. Сравниваем пару символов из (7.4) с парой на решетке (рис. 7.8) и в первых узлах ставим значение расстояния Хемминга. В последующих узлах ставим “накопленное” расстояние Хемминга с учетом первого и второго пути. Покажем, как формируется с учетом сказанного первая часть решетки. Обозначим уровни в решетке цифрами 0, 1, 2, 3, 4, 5, …, которые фиксируются вертикальными линиями на каждом шаге декодирования. С учетом этого начало формирования решетки будет иметь следующий вид.

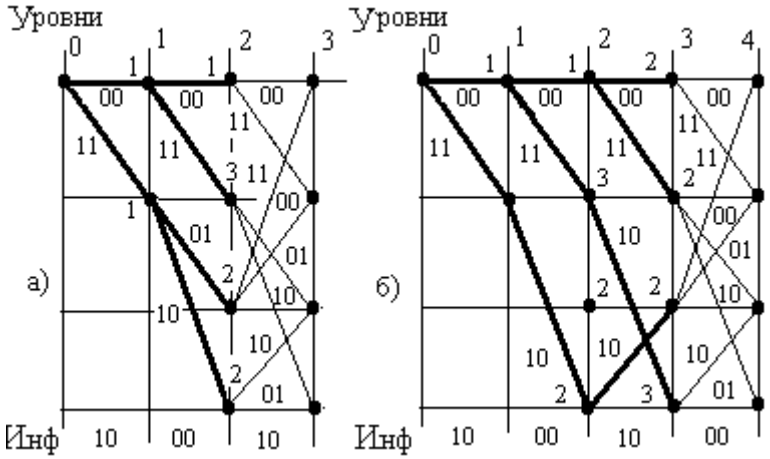


Рис. 7.9. Выбор пути с лучшей метрикой на решетке сверточного кода

Решетка (рис. 7.9) получается следующим образом. Берем пару символов (инф.) из последовательности (7.4) и сравниваем их с парами (путями) на решетке, начиная с уровня 0, т.е.

инф.(и) 10 ⊕ путь (п) 00 → 10 → 1 расст.Хемминга(р.Х.)

инф.(и) 10 ⊕ путь (п) 11 → 01 → 1 расст.Хемминга(р.Х.)

Результат сравнения расстояния Хемминга (р.Х.) проставляется в узлах решетки на уровне 1 (рис. 7.9, а).

Далее, берем вторую пару символов (инф.) из последовательности (7.4), т.е. 00 и поступаем аналогично (рис. 7.9, а). Но между уровнями 1 и 2 учитываем теперь накопленное расстояние Хемминга или накопленный путь (н.п) по первому уровню

и. 00
⊕
п. 00
00 → 0 (р.Х.) →накопленный путь (н.п.) = 1 + 0 =1
(с учетом пути 00),

и. 00
⊕
п. 11
11 → 2 (р.Х.) н.п. = 1 + 2 =3 (с учетом пути 00),

и. 00
⊕
п. 01
01 → 1 (р.Х.) н.п. = 1 + 1 = 2 (с учетом пути 11),

и. 00
⊕
п. 10
10 → 1 (р.Х.) н.п. = 1 + 1 = 2 (с учетом пути 11).

Теперь можно отбрасывать некоторые пути при построении дальнейшей решетки (третий уровень). Здесь (рис. 7.9, б) каждый из путей с уровня 2 раздваивается при общем числе путей 8. Затем сравниваются метрики (накопленное расстояние Хемминга) для пар путей, ведущих в каждый узел уровня 3, и из каждой пары сохраняется лишь лучший путь, а общее число путей вновь становится равным 4.

Так, к верхнему узлу уровня 3 ведут два пути и следует выбрать один по лучшей метрике. Исходя из этого, произведем сравнение третьей пары (инф.) из последовательности (7.4) с путями на решетке, входящими в 3-й узел:

и. 10
⊕
п. 00
10 → 1 (р.Х.) н.п. = 1 + 1 =2 (с учетом пути 00),

и. 10

⊕

п. 11

$01 \rightarrow 1$ (р.Х.) н.п. = 1 + 2 =3 (с учетом пути 11).

Из этих двух путей меньшее Хеммингово расстояние (метрику) имеет путь 00 с накоплением 2, поэтому он остается, а путь 11 исключается.

В следующий узел уровня 3 входят также два пути: 11, 00. Осуществляем выбор лучшего из них

и. 10

⊕

п. 11

$01 \rightarrow 1$ (р.Х.) н.п. = 1 + 1 =2 (с учетом пути 11),

и. 10

⊕

п. 00

$10 \rightarrow 1$ (р.Х.) н.п. = 1 + 2 =3 (с учетом пути 00).

Из этих двух путей исключается путь 00.

Следующий выбор осуществляем между путями 01 и 10, входящими в следующий узел уровня 3:

и. 10

⊕

п. 01

$11 \rightarrow 2$ (р.Х.) н.п. = 2 + 3 =5 (с учетом пути 01),

и. 10

⊕

п. 10

$00 \rightarrow 0$ (р.Х.) н.п. = 0 + 2 =2 (с учетом пути 10).

Из этих двух путей исключается путь 01.

В нижний узел уровня 3 ведут пути 10 и 01. Выбираем из них тот, который обладает лучшей метрикой

и. 10

⊕

п. 10

$00 \to 2$ (р.Х.) н.п. = 0 + 3 =3 (с учетом пути 10),

и. 10
⊕
п. 01

$11 \to 2$ (р.Х.) н.п. = 2 + 2 = 4 (с учетом пути 01).

Здесь лучшей метрикой обладает путь 10, он остается, а путь 01 исключается.

С учетом проведенных сравнений и выбора путей с лучшей метрикой решетка, представленная на рис. 7.9,а, приобретает вид, показанный на рис. 7.9,б.

Аналогичные вычисления осуществим теперь для уровня 4 решетки, представленной на рис. 7.9,б. В этих вычислениях опустим ясные теперь обозначения около путей при их сложении с очередной парой символов (инф.) 00 из (7.4)

1) 00 ⊕ 00 — 00 ⊕ 11

$00 \to 0 + 2 = 2$, $11 \to 2 + 2 = 4$,

остается путь 00;

2) 00 ⊕ 00 — 11 ⊕ 00

$11 \to 2 + 2 = 4$, $00 \to 0 + 2 = 2$,

остается путь 00;

3) 00 ⊕ 00 — 01 ⊕ 10

$01 \to 1 + 2 = 3$, $10 \to 1 + 3 = 4$,

остается путь 01;

4) 00 ⊕ 00 — 10 ⊕ 01

$10 \to 1 + 2 = 3$, $01 \to 1 + 3 = 4$,

остается путь 10.

В результате получаем решетку, показанную на рис. 7.10.

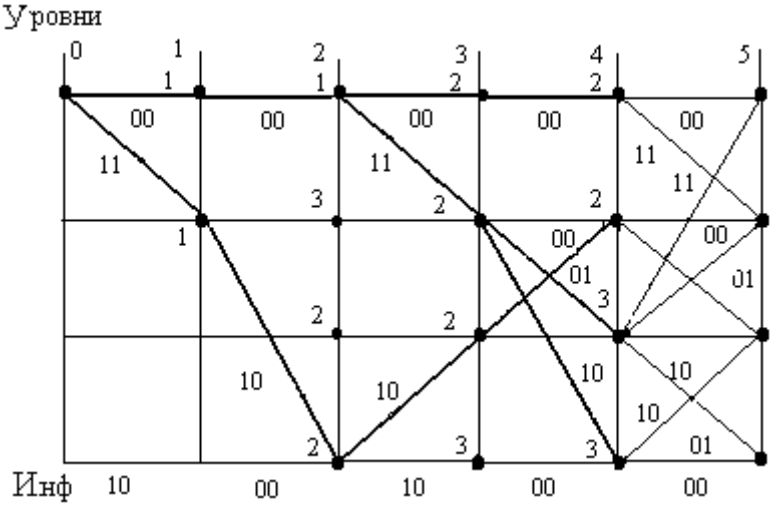


Рис. 7.10. Выбор пути с лучшей метрикой на четвертом уровне

Для уровня 5 в соответствии с рис. 7.10 выбираем лучшие пути, следующие с уровня 4, т.е.

1) 00 ⊕ 00 и 00 ⊕ 11

$00 \to 0 + 2 = 2$, $11 \to 2 + 3 = 5$,

остается путь 00;

2) 00 ⊕ 11 и 00 ⊕ 00

$11 \to 2 + 2 = 4$, $00 \to 0 + 3 = 3$,

остается путь 00;

3) 00 ⊕ 01 и 00 ⊕ 10

$01 \to 1 + 2 = 3$, $10 \to 1 + 3 = 4$,

остается путь 01;

4) 00 ⊕ 00 и 00 ⊕

$\underline{10}$ $\underline{01}$

$10 \rightarrow 1 + 2 = 3, \quad 01 \rightarrow 1 + 3 = 4,$

остается путь 10.

После устранения путей с худшей метрикой получаем решетку, показанную на рис. 7.11

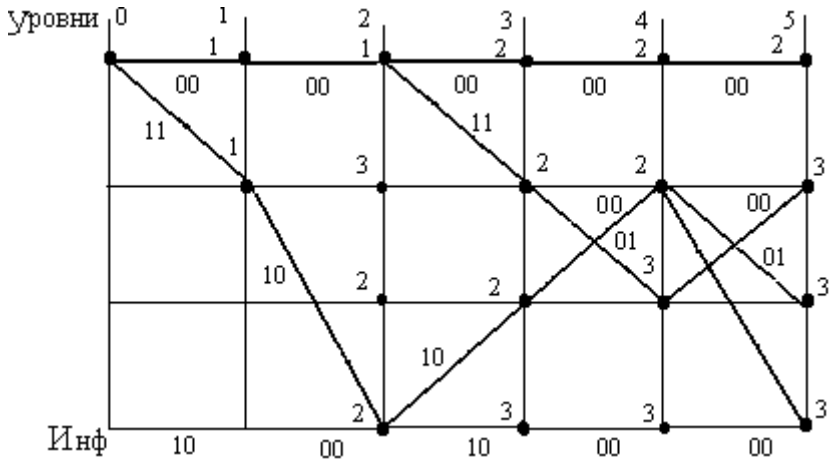


Рис. 7.11. Выбор пути с лучшей метрикой на пятом уровне

Рассматривая решетку на рис. 7.11, можно заметить, что на пятом уровне появляется отличие одного пути из четырех по лучшей метрике, т.е. минимальному накопленному расстоянию Хемминга. Здесь лучшей метрикой с величиной 2 против 3 на остальных, обладает путь 00 00 00 00 00. При включении в процедуру декодирования новых уровней через определенное их число все выжившие пути совпадут (сольются). С этого момента согласно алгоритму Витерби декодер начинает принимать решения о переданных символах, так как все выжившие пути приходят из одного узла, т. е. соответствуют одному информационному символу. Для рассматриваемого примера это произойдет на десятом уровне (такте), что можно наблюдать при выполнении лабораторного задания.

Таким образом, теория рекомендует принимать решения о переданных символах тогда, когда все выжившие пути

выходят из одного узла. Однако глубина (число уровней), на которой происходит такое слияние, не может быть вычислена заранее, так как она является случайной величиной, зависящей от количества произошедших в канале ошибок. Поэтому при практической реализации декодера не следует ждать указанного слияния, а нужно установить фиксированную глубину декодирования L. Экспериментально показано (компьютерное моделирование) [19], что при $L \geq 4K_o$ обеспечивается пренебрежимо малое ухудшение качества декодирования относительно оптимального выбора L.

Выбор конкретного значения глубины декодирования L приводит к определенной задержке начала декодирования символов принятой последовательности. Решение о значении символа (0 или 1) принимается после первых L шагов по решетке, где заранее известна метка на ребре 0 или 1 (рис. 7.8). Если ребро выжившего пути является нижним относительно второго ребра, выходящего из этого же узла, то берется значение символа 1, если верхним, то берется значение 0. Решение о следующем символе принимается после такта, когда начинается новый шаг пути по решетке и т.д.

7.4. Лабораторне задание

1. На диске Z найти и открыть файл «LABRNEW».

2. Затем открыть файл Lab7Win и далее открыть файл «Project1».

3. Через появившееся табло открыть окно ввода данных для выполнения лабораторной работы (рис. 7.12).

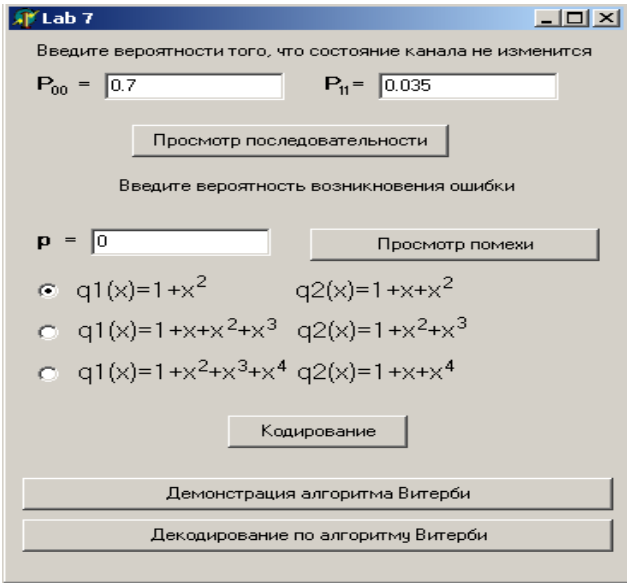


Рис. 7.12. Панель окна ввода данных

4. Изучить процедуру кодирования сверточным кодом.

4.1. Ввести в окна вероятности: P_{00}=0.7, P_{11} = 0.035, p = 0. Нажать на клавишу «Просмотр последовательности». Наблюдать последовательности из нулей и единиц, соответствующие введенным вероятностям. Их фрагмент показан на рис. 7.13. Записать в отчет одну верхнюю строку фрагмента из представленного на экране массива данных. Убрать с экрана массив, нажав на кнопку «Выход».

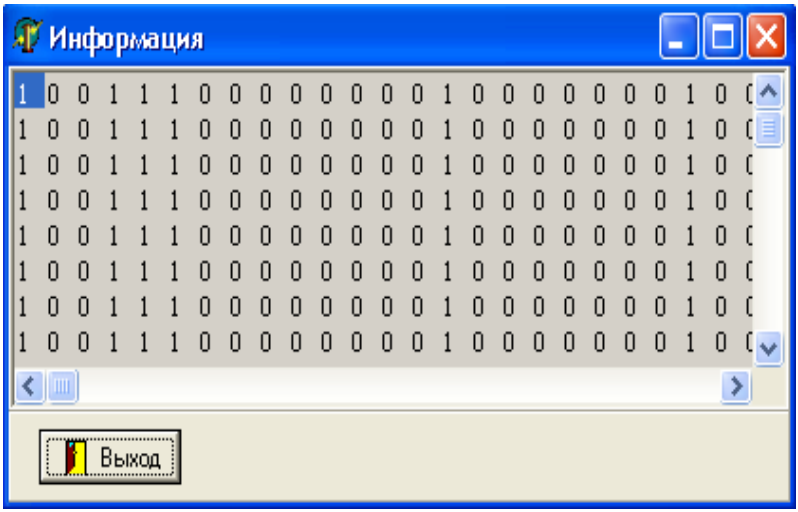


Рис. 7.13. Панель для просмотра последовательностей

4.2. Выбрать первый набор полиномов $g1(x)$ и $g2(x)$ на рис. 7.12 и осуществить кодирование массива, нажав на кнопку «Кодирование». Зарисовать схему кодера (рис. 7.14).

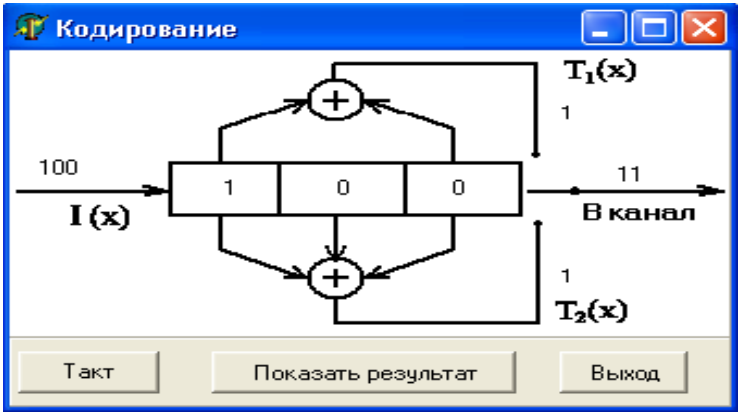


Рис. 7.14. Панель для просмотра процедуры кодирования

Зафиксировать процедуру проведенного кодирования массива данных. Обратить внимание, что в первой ячейке регистра (рис. 7.14) в начале кодирования уже находится символ 1 от первой строки входной последовательности массива данных (см. рис. 7.13), поступающей посимвольно слева на вход кодера. Нажимая поэтапно 15- 25 раз на кнопку «Такт», записывать вновь образующиеся элементы последовательности символов на выходе кодера, присоединяя их по очереди слева к уже имеющимся символам на выходе кодера. Нажать на кнопку «Показать результат» и наблюдать полностью кодированный массив данных. Сравнить полученную последовательность на выходе кодера с первой строкой кодированного массива. Убрать массив (кнопка «Выход») и далее панель «кодирование» (кнопка «Выход»).

4.2.1. Осуществить аналогичные процедуры для второй и третьей пар полиномов $g1(x)$, $g2(x)$ (рис. 7.12), используя последовательность действий из п. 4.2.

5. Осуществить демонстрацию алгоритма Витерби.

5.1. Ввести новые значения вероятностей: $P_{00} = 1$, $P_{11} = 0$ и $p = 0$ в панель окна ввода данных (рис. 7.12) и установить первую пару полиномов (верхняя строка).

5.2. Нажать на панели (рис. 7.12) клавишу «Демонстрация алгоритма Витерби» и дождаться появления надписи «Задание глубины декодирования». Установить глубину декодирования L =17 и нажать на кнопку «Начать».

5.3. В появившуюся панель «Решетка сверточного кода» с помощью «мыши» ввести в строку «помеха» кодовую комбинацию «10 00 10 00 00…» (7.4), как для примера в теоретическом разделе пособия к данной работе (см. стр. 187).

5.4. Нажать на кнопку «Такт» один раз и дождаться появления синих ребер решетки. Нажимая далее поочередно на кнопку «Такт», осуществить сравнение результатов построения решетки представленных в методических указаниях (рис. 7.9-7.11) и на экране. Наблюдать построение путей решетки для количества тактов больших пяти. Записать

в отчет число тактов, когда пути «сливаются». Перемещения решетки осуществлять движком под ней. Зафиксировать в отчет результаты декодирования кодовой комбинации с ошибками.

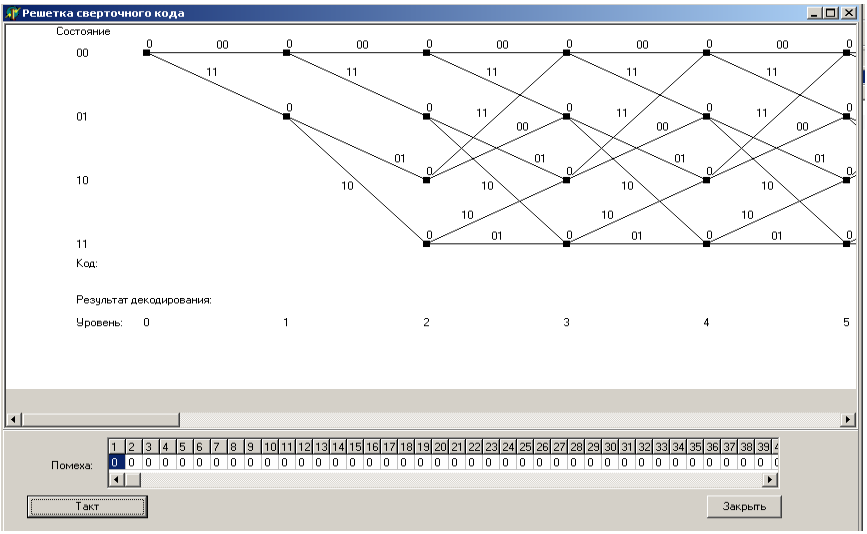


Рис. 7.15. Панель «Решетка сверточного кода» для наблюдения и фиксации процедуры декодирования алгоритмом Витерби с начальными значениями в узлах «0»

5.5. Закрыть панель «Решетка Витерби» кнопкой «Закрыть».

5.6. Нажать на панели (рис. 7.12) клавишу «Демонстрация алгоритма Витерби» и дождаться появления надписи «Задание глубины декодирования». Установить глубину декодирования 17 и нажать на кнопку «Начать».

5.7. В появившуюся панель «Решетка Витерби» с помощью «мыши» ввести в строку «Помеха» кодовую комбинацию 00010000110000 и нажать на кнопку «Такт» один раз. Дождаться появления на решетке Витерби двух первых ребер синего цвета.

5.8. Нажать на кнопку «Такт» еще раз и обратить внимание на появившиеся новые ребра решетки.

Записать в отчет последовательность процедурных вычислений накопленного расстояния у ребер решетки синего цвета (см. стр.189). Обратить внимание, что в начале отбрасывание путей пока не производилось.

5.9. Нажать на кнопку «Такт» еще раз и обратить внимание на появление «желтых» ребер вместо синих. Записать в отчет причину их появления, а также зафиксировать процедуру вычисления новых цифр в узлах решетки (см. стр.190).

5.10. Продолжая далее нажимать на кнопку «Такт», обратить внимание на появление слияния путей и зафиксировать число тактов, а также появление первого декодированного символа в строке «результат декодирования». Записать в отчет, причину появления именно этого символа (см. стр.194). Зафиксировать появление ошибок при декодировании кодовой последовательности, сдвигая для этого решетку ползунком.

5.11. Начиная с п. 5.5, повторить все пункты для комбинаций с пакетами ошибок, например, 001111111100000 и еще двух взятых произвольно, без записей процедур вычисления значений накопленных путей у узлов решетки. Отразить в отчете результаты декодирования. Сделать выводы по исправлению группирующихся ошибок и записать их в отчет. Закрыть панель «Решетка Витерби» кнопкой «Закрыть».

5.12. Исследовать помехоустойчивость сверточного кода.

Для этой цели ввести в окна панели (рис. 7.12) значения: P_{00} =1, P_{11} = 0, p = 0.01 и первую пару полиномов.

Определить вероятность ошибки $P_{ош}$ при воздействии на сверточный код независимых помех различной интенсивности с вероятностью ошибки в канале p(0.01; 0.02; и.т.д. до 0.1) при трех значениях пар полиномов. Используя клавишу «Просмотр помехи», можно наблюдать за распределением ошибок во фрагменте массива данных.

Установить первую пару полиномов. Нажать на клавишу «Декодирование по алгоритму Витерби» и дождаться появления надписи «Глубина декодирования». Установить ее значение равное 17 и нажать на кнопку «Начать». Дождаться появления значения вероятности ошибки $P_{ош}$. Осуществить тоже самое для других значений p. Результаты занести в таблицу. Провести аналогичные исследования для следующих двух пар полиномов. Сделать выводы.

Содержание отчета

1. Количественные показатели по выполнению конкретных пунктов с их комментариями и выводами.
2. Графики и таблицы по проведенным измерениям.
3. Краткие выводы по лабораторной работе.

Контрольные вопросы

1. Назовите основные отличия сверточных кодов от блоковых кодов. В чем проявляются эти отличия?
2. Как осуществляется кодирование информации сверточными кодами?
3. Как строится образующая матрица сверточного кода и какими свойствами она обладает?
4. Как строится кодер сверточного кода по порождающим многочленам?
5. Каковы основные отличия алгоритма Витерби от метода максимального правдоподобия?
6. Поясните, как строится решетка для сверточного кода и как она связана с его деревом.
7. Что такое путь по решетке и как его получают?
8. Приведите пример определения накопленного расстояния по решетке, как оно определяется?
9. Что такое выживший путь и как он образуется?

10. Объясните процедуру получения декодированных символов информации по выжившему пути на решетке.
11. Что такое глубина декодирования и как она выбирается?

8. ЛАБОРАТОРНАЯ РАБОТА № 8

КАСКАДНЫЕ КОДЫ

Цель работы: изучить построение, а также кодирование и декодирование информации каскадными кодами

Теоретическая часть

8.1. Каскадные коды

Каскадные коды используются для практической реализации кодов с большой длиной блока и весьма высокой корректирующей способностью [33,34]. Их основное предназначение – корректировать в кодовых последовательностях наборы искажений, состоящие как из группирующихся, так и независимых ошибок. Эти цели достигаются несколькими уровнями кодирования, чаще – двумя. В этих уровнях используются два кода. Одним из кодов является код Рида–Соломона (РС), а в качестве другого, чаще всего используется код Хемминга, исправляющий одиночные ошибки.

Процедура кодирования каскадным кодом сводится к следующему. Информационные символы входной последовательности разбиваются на K подблоков по m символов в каждом (рис. 8.1), что соответствует выбранной структуре кода Рида–Соломона, где K – это число информационных символов кода РС с размерностью m используемого элемента поля Галуа. Величина R - проверочные символы кода РС с такой же размерностью. Каждый из N символов кода РС представляется как двоичная кодовая комбинация длины $m=k_1$ и кодируется двоичным (n_1,k_1) кодом с r_1 проверочными символами. В результате получается двоичная кодовая комбинация длиной $n = = n_1 \cdot N$, которая и является кодовым словом каскадного кода. Для

обеспечения коррекции ошибок, возникающих в канале связи, необходимо использовать соответствующие кодеры и декодеры, как на рис. 8.2. Кодирование начинается внешним кодером кода РС, обрабатывающим внешние информационные последовательности и заканчивается внутренним кодером, использующим результаты кодирования, полученные «внутри» внешнего кодера. Кодовые последовательности на выходе канала под воздействием помех искажаются с появлением в них значительного числа ошибок (как пачек, так и одиночных). Нагружать код РС исправлением такого числа ошибок нецелесообразно по техническим причинам (сложность реализации). В связи с этим вначале работает внутренний декодер Хемминга, устраняющий все одиночные ошибки, но вносящий дополнительные искажения в те позиции кода РС, где число ошибок больше. Таким образом, общее количество искаженных позиций уменьшается, что позволяет внешнему декодеру исправить все оставшиеся искажения в позициях кода РС, среди которых и искажения от неправильного декодирования ошибок кодом Хемминга.

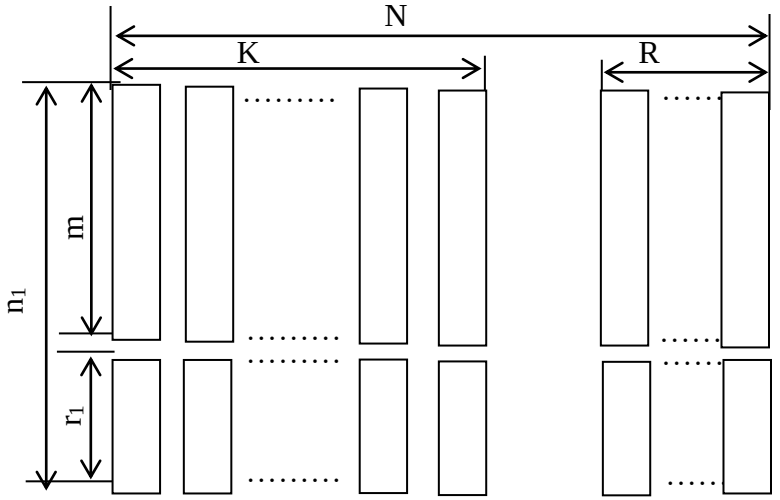


Рис. 8.1. Структура построения каскадного кода

В результате существенно снижается сложность реализации процедуры декодирования по сравнению с той, которая потребовалась бы для получения аналогичной вероятности ошибки при использовании только кода РС, естественно с кодовым расстоянием всего каскадного кода. Величина этого расстояния получается значительной и равной произведению кодовых расстояний внешнего и внутреннего кодов [1].

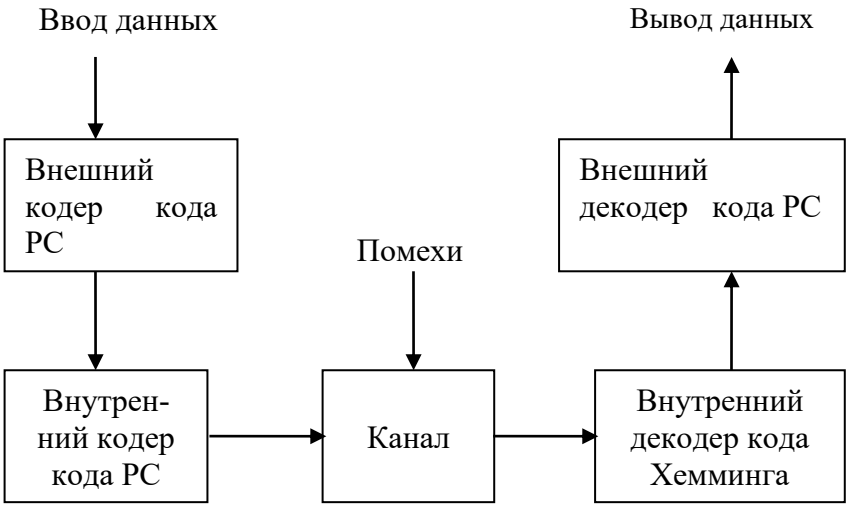


Рис. 8.2. Каскадное кодирование

8.2. Лабораторное задание

1. Просмотреть процедуру формирования, кодирования и декодирования информационных последовательностей каскадным кодом. Для этого выполнить следующие пункты:

1.1. Включить ЭВМ и войти в диск Z. Выбрать файл LABRNEW и открыть его. Затем открыть файл LAB8Win и далее файл labr8 [35] (рис. 8.3).

1.2. Выбрать и далее открыть заставку «Моделирование каскадного кода». Установить число ошибок, исправляемое внешним кодом Рида-Соломона 4, что соответствует использованию кода РС (15,7).

1.3. Использовать групповой код Хемминга (7,4) в качестве внутреннего кода, выбрав цифру 1.

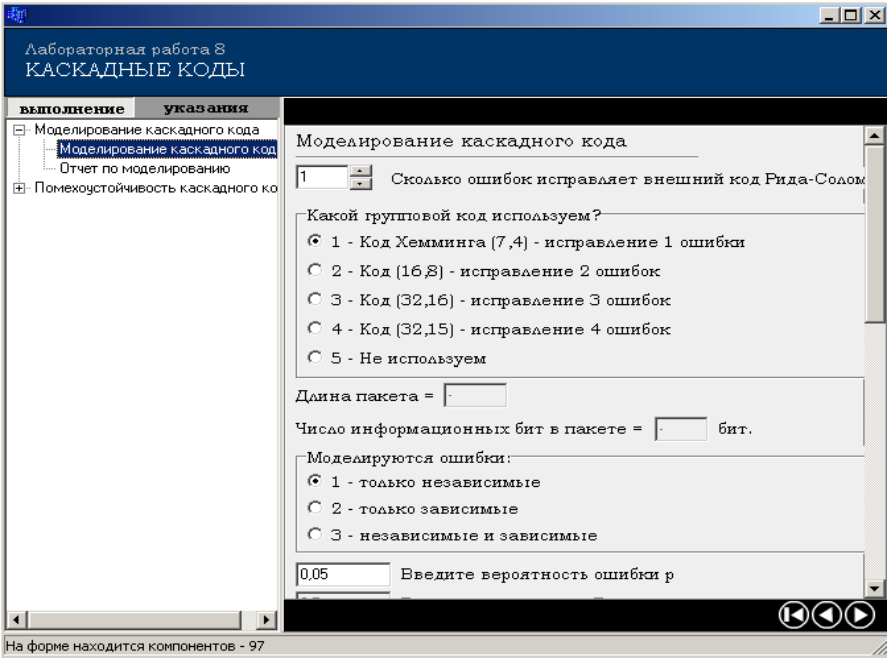


Рис. 8.3. Заставка для моделирования каскадного кода

1.4. Осуществить моделирование только независимых ошибок, выбрав цифру 1, и ввести вероятность ошибки p=0,25.

1.5. Осуществить моделирование четырёх пакетов, содержащих нулевые символы, набрав цифру 4 и подтвердить ввод.

1.6. Открыть строку «отчет по моделированию». Переместить движок окна отчета вверх.

Наблюдать пакет №1 нулевых символов с ошибками для каскадного кода, где первая позиция отражает четырёхразрядную комбинацию кода Рида-Соломона (Р) - первая буква слова Рид, а следующая затем позиция отражает трёхразрядную комбинацию проверочных символов кода Хемминга (Х) и т.д. по 15 позициям кода РС, всего 30 позиций. Цифры в позициях отражают десятичное число, полученное после воздействия ошибок. Так цифра 10 в позиции «Р» отражает двоичное представление этого числа, т.е. 1010, а цифра 0 в позиции «Х» отражает это число тремя символами, т.е. 000.

Две позиции образуют внутренний код Хемминга (7,4)-(РХ). Таким образом, длина пакета получается сложением двух произведений, первое $4 \times 15 = 60$ принадлежит коду РС, а второе $3 \times 15 = 45$ принадлежит коду Хемминга, т.е. в общей сложности 105 бит, из них 28 бит информационных (7×4), которые принадлежат коду РС(15,7). После декодирования ошибок внутренним кодом Хемминга образуется новый пакет, где часть ошибок в символах кода РС будет уже исправленной.

Записать в отчет символы кода Рида-Соломона до декодирования кодом Хемминга, и после декодирования им и далее расположить их друг под другом. Символы кода РС до декодирования кодом Хемминга брать из расположенных под буквами Р цифр, сдвигая их при необходимости через мышь и клавиатуру. Символы после декодирования взять из третьей строки цифр, которые выбраны из выше расположенного пакета. Отметить те позиции символов кода РС, в которых произошло исправление ошибок.

После исправления части ошибок в символах кода РС кодом Хемминга происходит исправление оставшихся ошибок в символах процедурой кода РС через использование вычислений в полях Галуа (см. стр. 158-160). Зафиксировать в отчет кодовые символы кода РС до исправления ошибок процедурой кода РС и после их исправления ей, записав

строки 4 и 5 цифр из окна моделирования. Отметить разницу в записи.

Зафиксировать декодирование следующих пакетов (2,3,4), перемещая движок окна вниз и используя вышеизложенные указания. Записать в отчет результаты. Сделать выводы.

1.7. Поступая аналогично, как в пп. 1.2-1.6, зафиксировать в отчет результат исправления всех ошибок в пакете, а также результаты, когда часть ошибок не исправляется. Для этого открыть заставку «моделирование каскадного кода» и установить вероятность ошибки p=0,16 вместо p=0,25 и далее поступать аналогично предыдущим исследованиям.

2. Исследовать помехоустойчивость каскадных кодов, построенных на основе кодов Рида-Соломона. Для этого выполнить следующие пункты:

2.1. Выбрать и далее открыть строку «помехоустойчивость каскадного кода» (рис. 8.4) и принять, что внешний код Рида-Соломона исправляет одну ошибку, а в качестве внутреннего кода используется код Хемминга (7,4). Зафиксировать в отчет параметры полученного каскадного кода, используя данные из теоретического материала лабораторной работы №8.

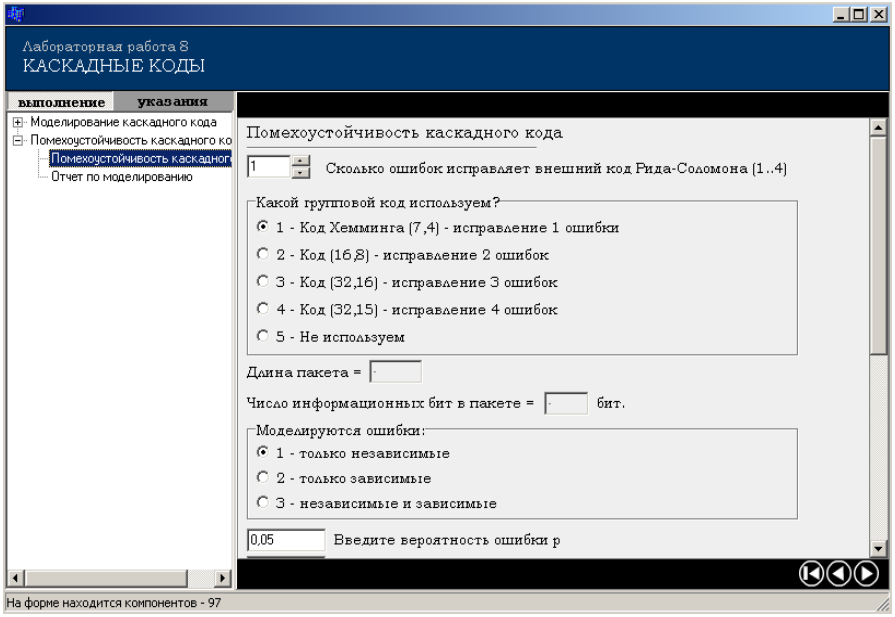


Рис. 8.4. Окно для исследования помехоустойчивости

2.2. Установить вид ошибок «только независимые», выбрав цифру 1, и ввести вероятность ошибки p=0,05. Установить число пакетов равное 10000 и подтвердить ввод. Зафиксировать результат моделирования.

2.3. Осуществить операции аналогичные п. 2.2, установив исправление внешним кодом Рида-Соломона двух ошибок с внутренним кодом Хемминга (7.4). Зафиксировать параметры полученного кода и результаты моделирования.

2.4. Осуществить операции аналогичные п. 2.3 при исправлении кодом Рида-Соломона трёх ошибок с внутренним кодом Хемминга (7.4). Зафиксировать параметры полученного кода и результаты моделирования.

2.5. Осуществить операции аналогичные п. 2.4 при исправлении кодом Рида-Соломона четырёх ошибок с внутренним кодом Хемминга (7.4). Зафиксировать параметры полученного кода и результаты моделирования.

2.6. Отразить в таблице результаты моделирования по пп. 2.1-2.5. Провести их сравнение. Сделать выводы.

2.7. Осуществить выбор лучшего группового кода из четырех для каскадного кодирования при исправлении кодом Рида-Соломона в начале трёх, а затем четырёх ошибок.

2.7.1. Реализовать процедуру п. 2.7 при независимых ошибках $p=0,05$, последовательно для всех четырех кодов.

2.7.2. Реализовать процедуру п. 2.7 для зависимых ошибок $P_1=0,05$ и $P_{11}=0,7$, последовательно для всех четырех кодов.

2.7.3. Реализовать процедуру п. 2.7 при совместных независимых и зависимых ошибках, приняв вероятность ошибки $p=0,025$, вероятность $P_1=0,025$ и вероятность $P_{11}=0.7$, последовательно для всех четырех кодов.

2.8. Исследовать помехоустойчивость кодов Рида-Соломона вне каскадного кодирования, для чего выполнить следующие пункты:

2.8.1. Выбрать надпись «Не используем»-5 и исследовать помехоустойчивость только кода Рида-Соломона при независимых ошибках, когда он последовательно исправляет число ошибок: 1, 2, 3, 4, а $p=0,05$.

2.8.2. Реализовать процедуру п. 2.8.1 для зависимых ошибок, когда $P_1=0,05$, а $P_{11}=0,7$.

2.8.3. Реализовать процедуру п. 2.8.1 для независимых и зависимых ошибок, когда вероятность ошибки $p=0,025$, вероятность $P_1=0,025$ и вероятность $P_{11}=0,7$.

2.8.4. Сравнить полученные результаты (составить таблицу) при моделировании по п. 2.8.1, п. 2.8.2 и п. 2.8.3 с результатами, полученными в п. 2.7.1, п. 2.7.2 и п. 2.7.3. Сделать выводы и отразить их в отчете.

Содержание отчета

1. Количественные показатели по выполнению конкретных пунктов с их комментариями и выводами.
2. Графики и таблицы по проведенным измерениям.

3. Краткие выводы по лабораторной работе.

Контрольные вопросы

1. Для каких целей используется каскадный код, и какие коды он включает в себя чаще всего?
2. Как осуществляется декодирование каскадных кодов?
3. Какие виды ошибок может исправлять каскадный код?
4. Что такое внутренний код и внешний код при каскадном кодировании?
5. Почему помехоустойчивость систем, применяющих каскадное кодирование, выше, чем у систем, использующих только блоковое кодирование?
6. Приведите примеры радиотехнических систем, где применяется каскадное кодирование.