

Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра радиотехнических систем

П. Г. Семашко

КОДИРОВАНИЕ И ДЕКОДИРОВАНИЕ ЛИНЕЙНЫХ ГРУППОВЫХ КОДОВ

МЕТОДИЧЕСКОЕ ПОСОБИЕ
к лабораторной работе по дисциплинам
**ТЕОРИЯ КОДИРОВАНИЯ И ЗАЩИТА ИНФОРМАЦИИ,
МЕТОДЫ КОДИРОВАНИЯ И ЗАЩИТА ИНФОРМАЦИИ,
ПРИКЛАДНАЯ ТЕОРИЯ КОДИРОВАНИЯ И ЦИФРОВАЯ
ОБРАБОТКА СИГНАЛОВ**
для студентов радиотехнических специальностей
всех форм обучения

Минск 2006

УДК 621.391.251 (075.8)

ББК 32.811.4 я 73

С 30

Семашко П.Г.

С 30 Кодирование и декодирование линейных групповых кодов: Метод. пособие к лаб. работе по дисц. «Теория кодирования и защита информации», «Методы кодирования и защита информации», «Прикладная теория кодирования и цифровая обработка сигналов» для студ. радиотехнических спец. всех форм обуч./ П.Г. Семашко. – Мн.: БГУИР, 2006.–24 с.: ил. ISBN 985-444-977-7

В пособии рассматриваются принципы помехоустойчивой передачи информации с использованием блоковых линейных кодов, способы задания, параметры и корректирующая способность кодов, принципы работы и синтез схем кодирующих и декодирующих устройств. В работе экспериментально исследуется способность кода корректировать различные виды ошибок.

УДК 621.391.251 (075.8)

ББК 32.811.4 я 73

ISBN 985-444-977-7

© Семашко П. Г., 2006

© БГУИР, 2006

СОДЕРЖАНИЕ

1. ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ
2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ
 - 2.1. ПРИНЦИП ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ
 - 2.2. ПАРАМЕТРЫ КОДОВ
 - 2.3. ЛИНЕЙНЫЕ ГРУППОВЫЕ КОДЫ
 - 2.4. ПОРОЖДАЮЩАЯ И ПРОВЕРОЧНАЯ МАТРИЦЫ КОДА
 - 2.5. КОРРЕКТИРУЮЩАЯ СПОСОБНОСТЬ КОДА
 - 2.6. СИНТЕЗ КОДЕРА
 - 2.7. СИНТЕЗ СИНДРОМНОГО ДЕКОДЕРА
 - 2.8. СИНТЕЗ МАЖОРИТАРНОГО ДЕКОДЕРА
3. МОДЕЛЬ ЛАБОРАТОРНОЙ УСТАНОВКИ
4. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ
 - 4.1. ПРЕДВАРИТЕЛЬНОЕ ЗАДАНИЕ
 - 4.2. ЛАБОРАТОРНОЕ ЗАДАНИЕ
5. СОДЕРЖАНИЕ ОТЧЕТА
6. КОНТРОЛЬНЫЕ ВОПРОСЫ
- ЛИТЕРАТУРА

1. ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ

1. Изучить методы помехоустойчивого кодирования и декодирования информации с помощью линейных групповых кодов.
2. Изучить принципы построения и функционирования кодирующих и декодирующих устройств этих кодов.
3. Исследовать основные характеристики линейных кодов.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. Принцип помехоустойчивого кодирования

Методы передачи цифровой информации по различным физическим каналам (проводным, оптическим, радио) получили в настоящее время широчайшее распространение, начиная от специальных систем до массовых телекоммуникаций. Даже такие традиционно аналоговые службы как наземное телевизионное и радиовещание в диапазонах от длинных до ультракоротких волн начали осваивать цифровой формат.

При проектировании современных систем ставится задача передать с максимальной скоростью и минимальными потерями (ошибками) информацию по каналу с ограниченной полосой частот при воздействии помех. Для этого используется многоуровневое кодирование в сочетании со сложными методами модуляции. Одним из средств уменьшения потерь информации, вызванных помехами, является применение *корректирующих кодов (помехоустойчивое кодирование)*. Упрощенно систему передачи информации с помехоустойчивым кодированием можно представить следующим образом (рис. 1).

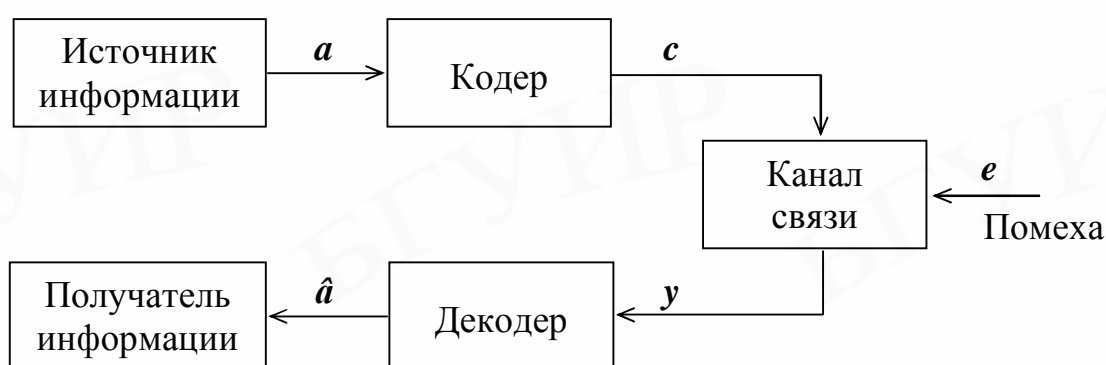


Рис. 1. Система передачи информации

Источник формирует цифровое *информационное сообщение* a , которое подвергается предварительной обработке в кодере, и полученное *кодированное сообщение* c передается по каналу, состоящему из передатчика, физической среды

и приемника. Под воздействием помехи e сообщение c изменяется (в нем возникают *ошибки*). Принятое кодовое сообщение $y \neq c$ обрабатывается декодером с целью получить исходную информацию $\hat{a} = a$. Однако, если в канале произошло слишком много ошибок (для данного кода), то получим $\hat{a} \neq a$. Поэтому на практике выбирают код в соответствии с уровнем помех.

Возможность исправления ошибок объясняется тем, что при кодировании в исходную информацию вносится *избыточность*. Пусть, например, каждый информационный разряд сообщения $a = 1011\dots$ передается по три раза, то есть $c = 111\ 000\ 111\ 111\dots$ (трехкратная избыточность). Если при передаче произошли ошибки в 3, 4 и 5 разрядах, то говорят, что *вектор ошибки* равен $e = 001\ 110\ 000\ 000\dots$ (местоположение единиц показывает, в каких разрядах кодового слова c произошли ошибки). Понятие вектора ошибки позволяет связать c и y простым выражением

$$y = c + e, \quad (1)$$

где «+» обозначает сложение по модулю 2. Таким образом, из канала связи будет принято кодовое сообщение $y = 110\ 110\ 111\ 111$. Очевидно, алгоритм декодирования должен заключаться в сравнении между собой разрядов в каждой тройке. Их несовпадение (например, 110) указывает на то, что произошли ошибки, и наоборот, совпадение разрядов (например, 111) говорит об отсутствии ошибок. Исправление ошибок может быть основано на предположении, что большинство принятых разрядов все-таки правильные, то есть если принято 110, то передавалось 111, а не 000. Это значит, что будет исправлена ошибка только в 3 разряде, а 4 и 5 останутся ошибочными и декодер выдаст $\hat{a} = 1111 \neq a$. Рассмотрев другие примеры, нетрудно убедиться, что код с трехкратным повторением способен исправить только одну ошибку в пределах тройки и обнаружить не более двух ошибок в пределах тройки.

Из данного примера можно сделать важные выводы:

- 1) для контроля ошибок (обнаружения, исправления) требуется избыточность;
- 2) платой за возможность контроля ошибок является увеличение объема передаваемого сообщения (скорость передачи полезной информации по каналу связи ниже его пропускной способности);
- 3) *корректирующая способность кода* (количество обнаруживаемых и исправляемых ошибок) ограничена и зависит от способа кодирования.

2.2. Параметры кодов

Коды подразделяются на *непрерывные (сверточные)* и *блоковые*. При использовании блоковых кодов информационный поток, формируемый источни-

ком, разбивается на блоки фиксированной длины (по k разрядов). Каждый блок кодируется и декодируется независимо от других блоков.

Пусть *информационный блок* (*информационное слово*, *информационный вектор*) состоит из k разрядов:

$$\mathbf{a} = [a_0, a_1, a_2, \dots, a_{k-1}].$$

В результате кодирования будет получено *кодое слово* (*кодоеый вектор*), состоящий из n разрядов (если код избыточный, то $n > k$):

$$\mathbf{c} = [c_0, c_1, c_2, \dots, c_{n-1}].$$

Величина n называется *длиной* или *значностью* кода. Для кодов используется обозначение вида (n, k) , например $(7, 4)$.

Отношение k/n называется *скоростью кода* и характеризует снижение скорости передачи информации, обусловленное кодированием.

Значения разрядов (символов) a_i, c_i выбираются из некоторого множества \mathcal{Q} , состоящего из q элементов, где q называется *основанием* кода. Например, для двоичных кодов $a_i, c_i \in \mathcal{Q} = \{0, 1\}$, $q = 2$.

Количество различных значений информационного вектора \mathbf{a} длиной k

$$M = q^k.$$

Поскольку каждому информационному вектору \mathbf{a} соответствует свой кодовый вектор \mathbf{c} , то количество различных значений \mathbf{c} также равно M . Величина M называется *мощностью* кода. Для избыточных кодов $M < q^n$, так как $k < n$. То есть для передачи информации используются не все возможные n -разрядные вектора. Используемые для передачи кодовые вектора \mathbf{c} называются *разрешенными*, а все остальные – *запрещенными*.

2.3. Линейные групповые коды

Множество всех n -разрядных векторов образует *группу* с операцией n -разрядного сложения по модулю q . В алгебре группой называют некоторое множество, для элементов которого определена одна основная операция – сложение или умножение. Любой группе присущи следующие свойства:

1) *Замкнутость*. Результатом операции сложения (умножения) двух любых элементов группы является некоторый элемент этой же группы.

2) *Ассоциативный закон*. Для любых трех элементов группы A, B и C

$$(A+B)+C = A+(B+C) \text{ или } (A \cdot B) \cdot C = A \cdot (B \cdot C).$$

3) Один из элементов группы является *нейтральным* (E). В группе с операцией сложения $A+E = A$, то есть $E = 0$. В группе с операцией умножения $A \cdot E = A$, то есть $E = 1$.

4) Для каждого элемента A группы существует такой *обратный элемент* \bar{A} , принадлежащий этой же группе, что

$$A + \bar{A} = 0 \text{ или } A \cdot \bar{A} = 1.$$

Если в группе выделить подмножество, для которого будут справедливы все указанные свойства, то такое подмножество называется *подгруппой*.

Код называется *групповым*, если множество всех разрешенных кодовых слов образует подгруппу в группе n -разрядных векторов.

Код называется *линейным*, если каждый разряд кодового слова c связан линейной зависимостью f с разрядами информационного слова a :

$$c_j = f_j(a_0, a_1, \dots, a_{k-1}) = \sum_{i=0}^{k-1} g_{i,j} a_i; g_{i,j} \in Q; j = 0, 1, \dots, (n-1). \quad (2)$$

где $g_{i,j}$ – константы. Например, для линейного кода (7,4) могут быть заданы следующие уравнения:

$$\begin{aligned} c_0 &= a_0, \\ c_1 &= a_1, \\ c_2 &= a_2, \\ c_3 &= a_3, \\ c_4 &= a_0 + a_1 + a_2, \\ c_5 &= a_0 + a_1 + a_3, \\ c_6 &= a_0 + a_2 + a_3. \end{aligned} \quad (3)$$

При этом множество всех M кодовых слов образует линейное k -мерное векторное пространство, то есть любое кодовое слово (вектор) может быть получено путем линейной комбинации k базисных векторов. Базисными векторами могут быть любые k линейно независимых кодовых векторов.

Любой групповой код является линейным, поскольку векторному пространству присущи свойства группы.

Система линейных уравнений вида (2) определяет способ кодирования и от нее зависят свойства кода, поэтому говорят, что она *задает* код. Код может задаваться также с помощью *порождающей* или *проверочной матрицы*.

2.4. Порождающая и проверочная матрицы кода

Любая система линейных уравнений, как известно, может быть записана в матричном виде. Тогда для системы (2) получим

$$c = a G, \quad (4)$$

где $G = [g_{i,j}]$ – матрица констант, $i = 0, 1, \dots, (k-1)$ – номер строки, $j = 0, 1, \dots, (n-1)$ – номер столбца.

Матрицу G называют порождающей. Например, для системы (3)

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}. \quad (5)$$

В качестве порождающей может быть задана любая матрица, содержащая в качестве строк k линейно независимых n -разрядных q -ичных векторов. Эти вектора и будут являться базисом для векторного пространства кода (см. раздел 2.3). Можно составить несколько различных матриц с размерами $k \times n$ и линейно независимыми строками, но от содержимого этих матриц будет зависеть корректирующая способность кода. Для получения кода с наибольшей корректирующей способностью при составлении матрицы G нужно стремиться к максимальному количеству ненулевых элементов в строках и, одновременно, к максимальному различию всех строк между собой (см. раздел 2.5).

Для порождающей матрицы допускаются перестановки строк и столбцов, линейные операции со строками. В результате получаются *эквивалентные коды*. Применяя данные операции, любую порождающую матрицу можно привести к *каноническому виду (приведенно-ступенчатой форме)*

$$G = [I_k \ G^*], \quad (6)$$

где I_k – единичная матрица порядка k ;

G^* – проверочная часть порождающей матрицы размером $k \times (n-k)$.

Приведенная выше матрица G имеет канонический вид.

Код, заданный порождающей матрицей канонического вида является *систематическим*, поскольку в кодовом слове первые k разрядов будут равны информационным (см. уравнения (3)). Остальные (не информационные) разряды кодового слова называются *проверочными*. Количество проверочных разрядов обозначают $r = n-k$.

Код может быть задан *проверочной матрицей H* , которая связана с порождающей матрицей *основным уравнением кодирования*:

$$G H^T = 0,$$

где знак T обозначает транспонирование, а 0 – нулевая матрица порядка k .

Проверочная матрица H имеет размеры $r \times n$.

Если порождающая матрица задана в каноническом виде (6), то проверочная матрица может быть найдена путем транспонирования подматрицы G^* и добавления справа единичной матрицы порядка r :

$$G = [I_k \ G^*] \Rightarrow H = [G^{*T} \ I_r].$$

Например, для порождающей матрицы (5) получим

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \quad (7)$$

2.5. Корректирующая способность кода

Для определения способности кода противостоять ошибкам необходимо ввести следующие понятия.

Весом Хэмминга $wt(c)$ вектора c называется количество его ненулевых разрядов. Например, $wt(1001011) = 4$, $wt(0000000) = 0$.

Расстоянием Хэмминга $d(c_1, c_2)$ между двумя векторами c_1 и c_2 называется вес их разности $d(c_1, c_2) = wt(c_1 - c_2)$. Например, $d(1001, 1011) = wt(0010) = 1$. Видно, что расстояние Хэмминга равно количеству попарно несовпадающих разрядов векторов.

Из выражения, определяющего изменение кодового слова под воздействием ошибки $y = c + e$ видно, что расстояние между y и c равно весу вектора ошибки $d(y, c) = wt(y - c) = wt(e)$. Отсюда могут быть сделаны следующие выводы:

1) Гарантировать *обнаружение* ошибки e можно только тогда, когда ее вес меньше расстояния между передаваемым словом c_1 и всеми другими словами кода. В этом случае разрешенный вектор c_1 под воздействием ошибки превращается в запрещенный (рис. 2, а – кодовые вектора изображены как точки в некотором пространстве). В противном случае разрешенный вектор c_1 может превратиться в другой разрешенный вектор этого же кода ($c_2 = c_1 + e$) и обнаружение такой ошибки будет невозможным (рис. 2, б). Поскольку c_1 и c_2 являются элементами одной и той же группы, то из равенства $c_1 + e = c_2$ и свойства замкнутости группы следует, что e тоже является элементом этой группы. Иными словами, если вектор ошибки совпадает с одним из разрешенных кодовых слов, то такую ошибку обнаружить не возможно.

а) ошибка будет обнаружена

б) ошибка не будет обнаружена

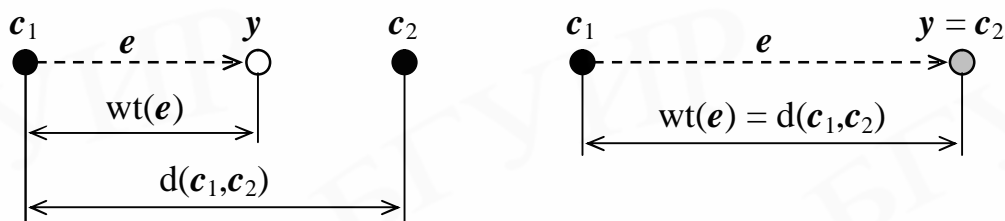


Рис. 2. Геометрическая интерпретация процесса обнаружения ошибок

2) Гарантировать *исправление* ошибки e можно только тогда, когда ее вес меньше половины расстояния между передаваемым словом c_1 и всеми другими словами кода. При декодировании принятого вектора y , по сути, происходит отождествление его с ближайшим (в смысле расстояния Хэмминга) разрешенным вектором c . Если под воздействием ошибки кодовый вектор y окажется ближе к другому разрешенному вектору c_2 , то он будет отождествлен с c_2 , а не с c_1 , который передавался (рис. 3).

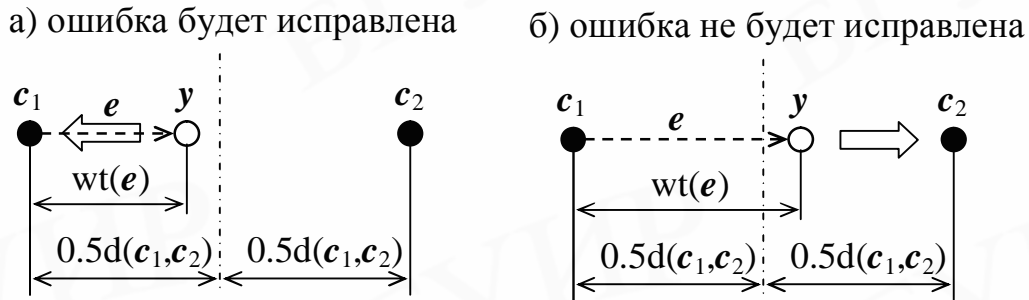


Рис. 3. Геометрическая интерпретация процесса исправления ошибок

Из сказанного ясно, что корректирующая способность кода в целом определяется расстоянием Хэмминга между двумя ближайшими разрешенными кодовыми словами. Это минимальное расстояние называется *кодовым расстоянием* d_{\min} .

Кодовое расстояние можно найти: а) по множеству кодовых слов; б) по порождающей матрице G ; в) по проверочной матрице H .

Если известно множество кодовых слов $\{c_1, c_2, \dots, c_M\}$, то кодовое расстояние

$$d_{\min} = \min_{i \neq j} \{d(c_i, c_j)\} = \min_{i \neq j} \{wt(c_i - c_j)\}, i, j = 1, 2, \dots, M.$$

В силу замкнутости группы кодовых слов, разность $c_i - c_j$ при $i \neq j$ равна некоторому ненулевому кодовому слову этого же кода. Следовательно, кодовое расстояние равно минимальному весу ненулевых кодовых слов

$$d_{\min} = \min_i \{wt(c_i)\}, i = 1, 2, \dots, M.$$

Если задана порождающая матрица кода G , то для нахождения кодового расстояния необходимо:

- 1) найти минимальный вес строк G : $d_1 = \min \{wt(c_i)\}, i = 1, 2, \dots, k$;
- 2) найти минимальное расстояние Хэмминга между всеми строками G : $d_2 = \min \{d(c_i, c_j)\}, i, j = 1, 2, \dots, k, i \neq j$;
- 3) кодовое расстояние будет равно $d_{\min} = \min \{d_1, d_2\}$.

Если задана проверочная матрица кода \mathbf{H} , то кодовое расстояние будет равно минимальному количеству столбцов \mathbf{H} , сумма которых равна нулевому вектору. Например, в проверочной матрице (7) нет ни одного нулевого столбца, следовательно, $d_{\min} \neq 1$. В ней также нет двух одинаковых столбцов (сумма одинаковых векторов по модулю 2 всегда равна нулю), следовательно, $d_{\min} \neq 2$. Зато, если сложить, например, 1, 2 и 7 столбцы, то получим нуль

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

следовательно, $d_{\min} = 3$.

Знание кодового расстояния позволяет вычислить максимальный вес векторов ошибок, которые будут гарантированно обнаруживаться или исправляться данным кодом. Вес вектора ошибки называют *кратностью ошибки*.

Кратность обнаруживаемых кодом ошибок $t_{\text{обн}} \leq d_{\min} - 1$. Кратность исправляемых кодом ошибок $t_{\text{исп}} \leq (d_{\min} - 1)/2$. Например, если $d_{\min} = 3$, то код обнаружит все ошибки кратностью 1 и 2, исправит все ошибки кратностью 1.

Для обнаружения и исправления ошибок используется синдромный или мажоритарный методы декодирования.

2.6. Синтез кодера

Кодер умножает входной информационный вектор на заданную порождающую матрицу $\mathbf{c} = \mathbf{aG}$. Используя приведенную ранее матрицу \mathbf{G} кода (7,4), можно записать

$$[c_0 \ c_1 \ c_2 \ c_3 \ c_4 \ c_5 \ c_6] = [a_0 \ a_1 \ a_2 \ a_3] \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix},$$

откуда по правилам матричного умножения получаем способ вычисления каждого разряда кодового слова:

$$c_0 = 1 \cdot a_0 + 0 \cdot a_1 + 0 \cdot a_2 + 0 \cdot a_3 = a_0,$$

$$c_1 = a_1,$$

$$c_2 = a_2,$$

$$c_3 = a_3,$$

$$c_4 = a_0 + a_1 + a_2,$$

$$c_5 = a_0 + a_1 + a_3,$$

$$c_6 = a_0 + a_2 + a_3,$$

и схему кодера (рис. 4) на элементах «исключающее ИЛИ».

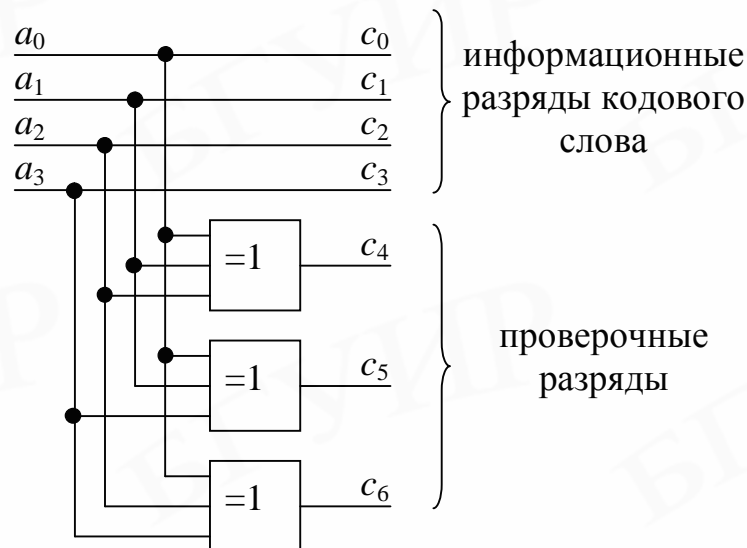


Рис. 4. Кодер линейного кода

2.7. Синтез синдромного декодера

Декодирование принятого вектора y по синдрому заключается в вычислении синдрома s , дешифрации синдрома (определении оценки вектора ошибки \hat{e}) и исправлении ошибки (рис. 5).

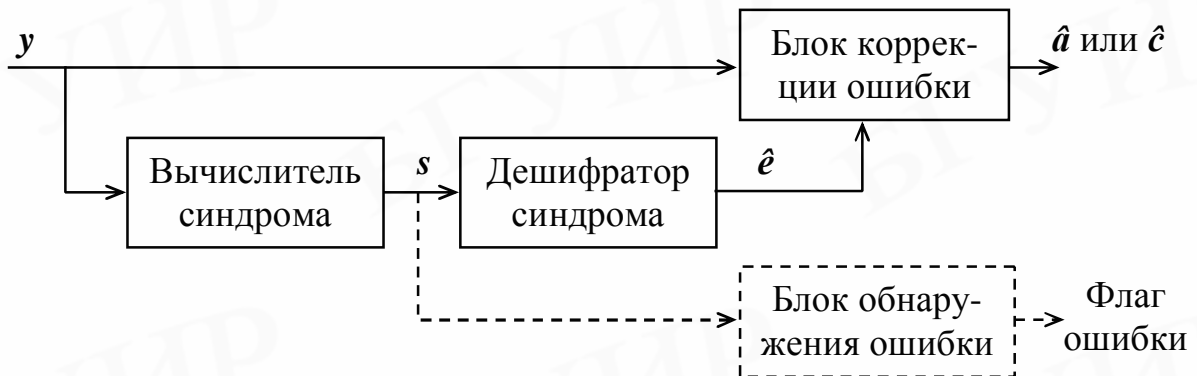


Рис. 5. Структура синдромного декодера

Синдромом ошибки называется r -разрядный вектор

$$s = yH^T, \quad (8)$$

значение которого зависит от ошибки и не зависит от передаваемого кодового слова. Действительно, используя (1) и (4), получим $s = yH^T = (c+e)H^T = cH^T + eH^T = aGH^T + eH^T$. Так как $GH^T = 0$, то

$$s = eH^T. \quad (9)$$

Используя полученную ранее матрицу \mathbf{H} кода (7,4), выражение (8) можно записать в виде

$$[s_0 \quad s_1 \quad s_2] = [y_0 \quad y_1 \quad y_2 \quad y_3 \quad y_4 \quad y_5 \quad y_6] \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

откуда получаем способ вычисления каждого разряда синдрома:

$$s_0 = 1 \cdot y_0 + 1 \cdot y_1 + 1 \cdot y_2 + 0 \cdot y_3 + 1 \cdot y_4 + 0 \cdot y_5 + 0 \cdot y_6 = y_0 + y_1 + y_2 + y_4,$$

$$s_1 = y_0 + y_1 + y_3 + y_5,$$

$$s_2 = y_0 + y_2 + y_3 + y_6,$$

и схему вычислителя синдрома (рис. 4) на элементах «исключающее ИЛИ».

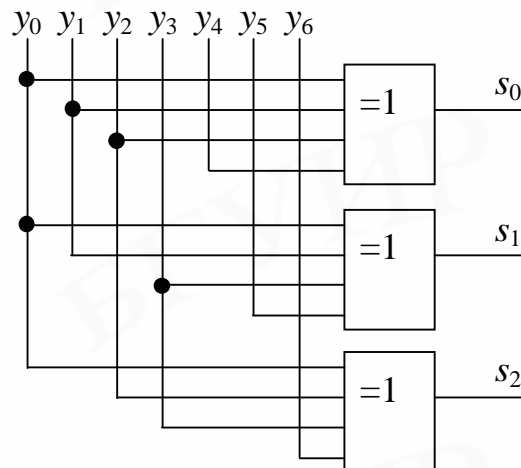


Рис. 6. Вычислитель синдрома

Дешифрация синдрома основана на соотношении $s = e\mathbf{H}^T$, из которого можно найти значение синдрома для каждого вектора ошибки, который может быть исправлен с помощью данного кода.

Например, для кода, заданного проверочной матрицей (7), было определено, что он гарантирует исправление любой одиночной ошибки ($t_{\text{испр}} = 1$). Ошибке $e = [1000000]$ соответствует синдром $s = e\mathbf{H}^T = [111]$ и т.д. (рис. 7, а).

Если код систематический, то не обязательно производить полную дешифрацию синдрома. Достаточно вычислить только те разряды вектора ошибки, которые соответствуют информационным разрядам кодового слова. Действительно, чтобы получить переданную информацию a , достаточно восстано-

вить лишь информационные разряды, а избыточные (проверочные) разряды восстанавливать не обязательно. Тогда таблицу дешифрации можно сократить (рис. 7, б), что упростит и схему.

а)

Вектор ошибки							Синдром		
e_0	e_1	e_2	e_3	e_4	e_5	e_6	s_0	s_1	s_2
1	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	1	1
0	0	0	0	1	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0
0	0	0	0	0	0	1	0	0	1

→

б)

s_0	s_1	s_2	e_0	e_1	e_2	e_3
1	1	1	1	0	0	0
1	1	0	0	1	0	0
1	0	1	0	0	1	0
0	1	1	0	0	0	1

Рис. 7. Полная (а) и достаточная (б) таблицы дешифрации синдрома

Дешифратор синдрома может быть реализован, например, с использованием селекторов кода «НЕ-И» (рис. 8). Каждый селектор формирует свой разряд e_i и выдает логическую «1» только при определенном значении синдрома.

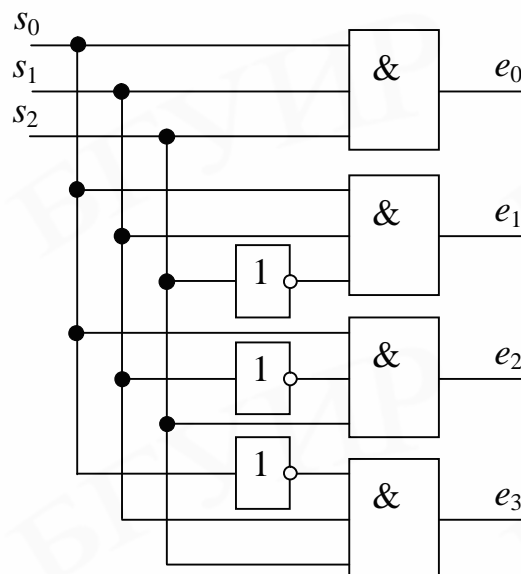


Рис. 8. Дешифратор синдрома

Коррекция ошибок основана на выражении (1), из которого можно выразить передаваемое кодовое слово $c = y - e$. Для двоичных кодов ($q = 2$) вычитание эквивалентно сложению, поэтому блок коррекции ошибок строится на элементах «исключающее ИЛИ».

Для рассматриваемого систематического кода, заданного порождающей матрицей (5), достаточно исправить только информационные разряды. Соответствующая схема приведена на рис. 9.

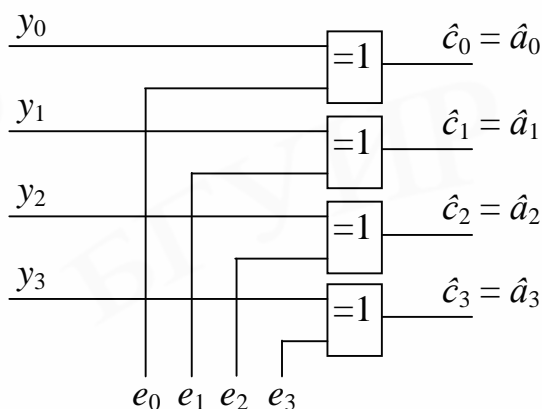


Рис. 9. Блок коррекции ошибок

Если используется несистематический код, то необходима полная дешифрация синдрома, коррекция всех разрядов кодового слова, а затем операция обратная кодированию для получения информационного вектора.

Синдромный декодер может работать в режиме обнаружения ошибок. Из формулы (9) видно, что при отсутствии ошибок $s = \mathbf{0}$. Таким образом, флаг наличия ошибки можно сформировать с помощью схемы «ИЛИ» (рис. 10), которая будет выдавать логическую «1» при $s \neq \mathbf{0}$.

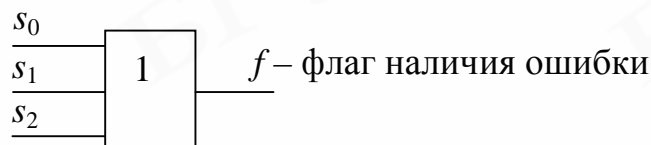


Рис. 10. Блок обнаружения ошибки

2.8. Синтез мажоритарного декодера

Мажоритарный метод (метод большинства) основан на том, что, в силу избыточности кодового слова, одна и та же информация может быть извлечена из разных его частей (разрядов). Если ошибками поражена некоторая часть кодового слова, то информация может быть извлечена из других, не поврежденных частей.

Декодирование каждого информационного разряда производится отдельно, независимо от других. В качестве примера рассмотрим декодирование разряда a_0 для кода, заданного системой уравнений (3). Из этих уравнений можно

выразить a_0 четырьмя способами, и, учитывая эквивалентность сложения и вычитания по модулю 2, записать:

$$\left\{ \begin{array}{l} a_0 = c_0 \\ a_0 = c_4 + a_1 + a_2 \\ a_0 = c_5 + a_1 + a_3 \\ a_0 = c_6 + a_2 + a_3 \end{array} \right. \xrightarrow{a_1 = c_1; a_2 = c_2; a_3 = c_3} \left\{ \begin{array}{l} a_0 = c_0 \\ a_0 = c_1 + c_2 + c_4 \\ a_0 = c_1 + c_3 + c_5 \\ a_0 = c_2 + c_3 + c_6 \end{array} \right.$$

При отсутствии ошибок все эти равенства будут выполняться и для принятого кодового слова y . Таким образом в декодере формируются предварительные оценки a_0 :

$$\left\{ \begin{array}{l} \hat{a}_0 = y_0 \\ \hat{a}_0 = y_1 + y_2 + y_4 \\ \hat{a}_0 = y_1 + y_3 + y_5 \\ \hat{a}_0 = y_2 + y_3 + y_6 \end{array} \right. \quad (10)$$

Если ошибка произошла в разряде y_0 , то первая оценка будет неверной, а три остальных – верными. Если ошибка произошла в разряде y_1 , то две из четырех оценок будут неверными (2 и 3 уравнения), и т.д. Поскольку заранее не известно значение a_0 , то решение принимается по большинству (мажоритарный принцип). Например, если получены оценки 1, 0, 1 и 1, то декодер выдаст $\hat{a}_0 = 1$.

Из уравнений (10) видно, что при одиночной ошибке в разряде y_1 , y_2 или y_3 невозможно определить большинство, так как из 4-х оценок две неверные. В то же время известно, что код исправляет любую одиночную ошибку ($t_{\text{испр}} = 1$). Значит, уравнения оценок выбраны неудачно. Можно получить другие уравнения оценок, используя подстановки переменных, полученные из исходной системы (3). Другой способ получения новых уравнений состоит в сложении между собой имеющихся уравнений. Например, если сложить три последних уравнения из системы (10), то получим $\hat{a}_0 = y_4 + y_5 + y_6$ (так как $q = 2$ и сложение выполняется по модулю 2, то $\hat{a}_0 + \hat{a}_0 = 0$, $y_1 + y_1 = 0$, $y_2 + y_2 = 0$, $y_3 + y_3 = 0$).

Таким образом, необходимо сформировать нечетное количество уравнений оценок и убедиться, что при любых $t_{\text{испр}}$ ошибочных разрядах в векторе y количество неверных оценок не превышает половины. В рассматриваемом примере невозможно получить три уравнения оценок, удовлетворяющих данному требованию. Поэтому для декодирования a_0 будут использоваться пять оценок:

$$\begin{cases} \hat{a}_0 = y_0 \\ \hat{a}_0 = y_1 + y_2 + y_4 \\ \hat{a}_0 = y_1 + y_3 + y_5 \\ \hat{a}_0 = y_2 + y_3 + y_6 \\ \hat{a}_0 = y_4 + y_5 + y_6 \end{cases} \quad (11)$$

Нетрудно убедиться, что при любой одиночной ошибке в слове y из пяти оценок неверными оказываются не более двух.

Схема декодирования разряда a_0 (рис. 11) будет состоять из элементов «исключающее ИЛИ», вычисляющих оценки по уравнениям (11), и *мажоритарного элемента* (Maj), формирующего окончательную оценку \hat{a}_0 по большинству.

Мажоритарный элемент может быть реализован как сумматор с пороговым элементом или методами логического синтеза. Например, логическая функция мажоритарного элемента с тремя входами имеет вид

$$y(x_1, x_2, x_3) = x_1 x_2 x_3 \vee \overline{x_1} x_2 x_3 \vee x_1 \overline{x_2} x_3 \vee x_1 x_2 \overline{x_3}.$$

Аналогичным образом разрабатываются схемы декодирования остальных информационных разрядов a_1 , a_2 и a_3 .

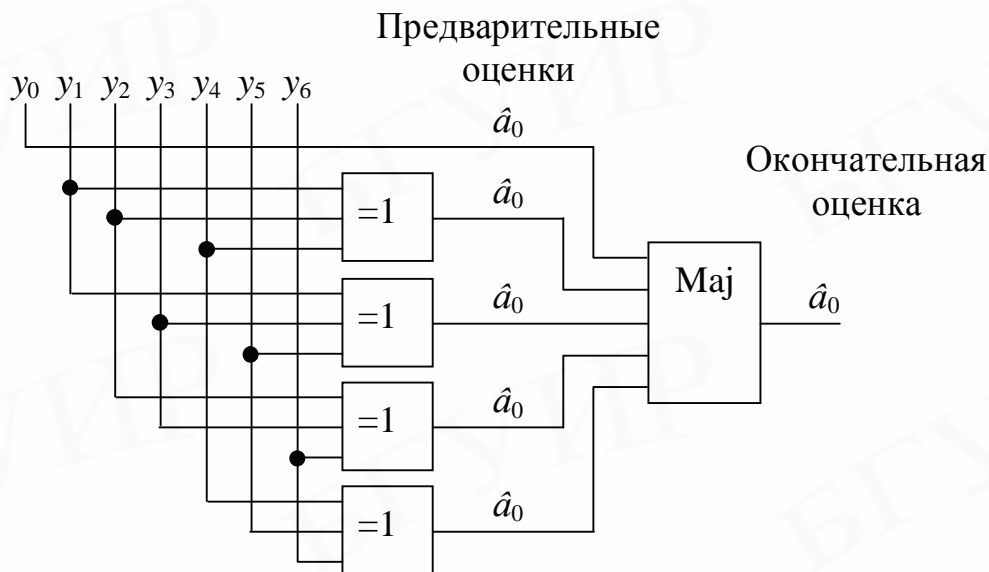


Рис. 11. Схема мажоритарного декодирования a_0

3. МОДЕЛЬ ЛАБОРАТОРНОЙ УСТАНОВКИ

Экспериментальная часть лабораторной работы выполняется на ПЭВМ в среде схемотехнического моделирования Electronics Workbench. Краткое описание этого программного обеспечения содержится в методическом пособии [7].

Модель лабораторной установки находится в файле **group_hem7_4.ewb** и содержит источник двоичного сообщения, кодер, модель канала связи, синдромный и мажоритарный декодеры, индикаторы для отображения состояний в различных точках схемы (рис. 12). Включение питания производится тумблером в верхнем правом углу окна программы. Данная схема позволяет моделировать процесс передачи информации в системе с помехоустойчивым кодированием при возникновении заданной ошибки в канале связи. Вектор ошибки задается с помощью 7-и ключей, управляемых с клавиатуры нажатием клавиш «1» – «7».

Источник двоичного сообщения состоит из 4-х ключей, управляемых с клавиатуры нажатием клавиш «Q», «W», «E», «R», и формирует 4-разрядное двоичное информационное слово.

Схемы кодера (рис. 13, а – **Coder**), синдромного декодера (рис. 13, в – **decoder**), мажоритарного декодера (**decoder2**), канала связи (**channel**) выполнены с использованием встроенных логических элементов XOR (исключающее ИЛИ), AND (И), OR (ИЛИ), NOT (НЕ). Особенностью является модульная организация схем. Так модуль **multG** (рис. 13, б), выполняющий умножение на проверочную часть порождающей матрицы G^* , используется как в кодере для вычисления проверочных разрядов, так и в декодере при вычислении синдрома. В модуле **sind_dec** собрана схема дешифрации синдрома. При реализации мажоритарного декодера можно использовать готовые модули мажоритарных элементов с 3 входами (**Maj3**) или с 5 входами (**Maj5**).

Для того чтобы изменить код, используемый в лабораторной установке, необходимо отредактировать модули **multG**, **sind_dec** и **decoder2**. Модуль открывается двойным нажатием левой кнопки «мыши», после чего возможно графическое редактирование схемы.

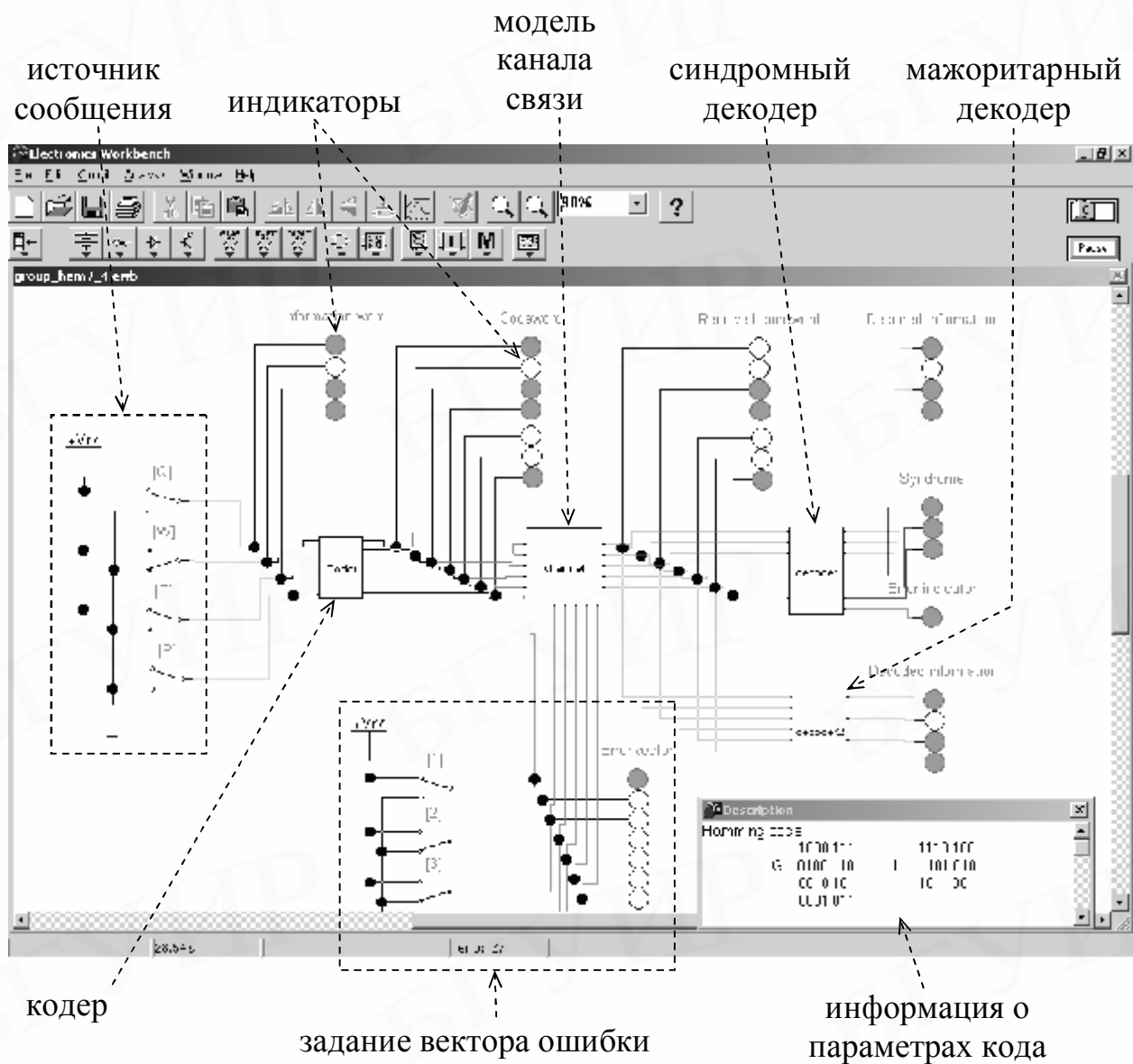


Рис. 12. Модель лабораторной установки в среде Electronics Workbench

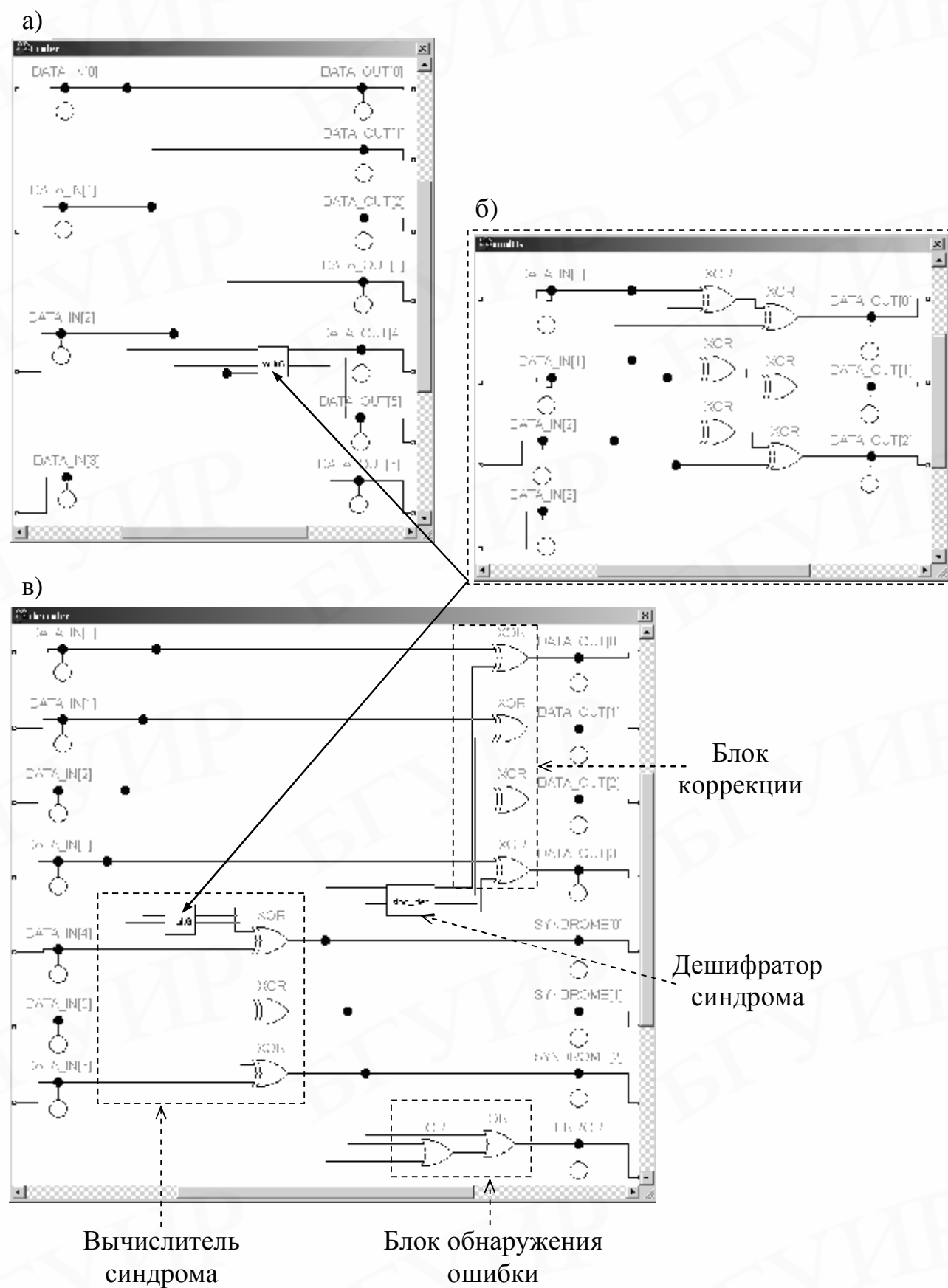


Рис. 13. Модули **Coder** (а), **multG** (б), **decoder** (в)

4. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Перед началом выполнения получить у преподавателя исходные данные – порождающую матрицу кода для своего варианта.

4.1. Предварительное задание

1. Изучить раздел «Краткие теоретические сведения» настоящего пособия.
2. Записать проверочную матрицу и определить основные параметры кода (длина, число информационных и проверочных разрядов, мощность, кодовое расстояние, максимальная кратность обнаруживаемых и исправляемых кодом ошибок).
3. Синтезировать схемы кодера, синдромного и мажоритарного декодеров.
4. Сделать заготовку отчета по лабораторной работе (см. раздел «Содержание отчета»), под таблицу эксперимента (рис. 14) отвести одну полную страницу формата А4.

Кодер		Канал связи		Декодер синдромный			Декодер мажоритарный
a	c	e	y	\hat{a}	S	f	\hat{a}

Рис. 14. Таблица экспериментальных данных

4.2. Лабораторное задание

Ознакомление с моделью лабораторной установки.

1. Запустить программу **Electronics Workbench v.5.0** и открыть файл со схемой лабораторной установки.
2. Включить моделирование схемы и ознакомиться с органами управления и индикации. Убедиться, что при отсутствии помех в канале связи (нулевой вектор ошибки), а также при одиночной ошибке в любом символе информация передается без искажения. Убедиться, что при двух и более ошибках в кодовом

слове на выходах декодеров получаем искаженную информацию. Выключить моделирование схемы.

Создание схем для своего варианта кода.

3. Внести изменения в модули кодера и декодеров так, чтобы получились схемы, синтезированные в предварительном задании. Сохранить новую схему лабораторной установки в файле с другим именем.

4. Включить моделирование новой схемы. Убедиться, что при нулевом векторе ошибок информация передается без искажений. Обратить внимание на значения синдрома и флага наличия ошибки.

Кодирование.

5. Изменяя значения информационного слова на входе кодера, перебрать все возможные двоичные комбинации. Занести в таблицу эксперимента все эти информационные комбинации (a) и соответствующие им кодовые слова (c).

Экспериментальное исследование корректирующей способности кода.

6. Установить на входе кодера произвольное значение информационного вектора, которое не будет в дальнейшем изменяться. Записать в новую строку таблицы эксперимента значения информационного (a) и кодового (c) векторов.

7. Оценить способность кода обнаруживать и исправлять одиночные ошибки. Для этого нужно последовательно перебрать все возможные вектора одиночных ошибок. Для каждой ошибки (e) записать в таблицу значения кодового слова на выходе канала связи (y), информационного слова (\hat{a}), синдрома (S) и флага наличия ошибки (f) на выходе синдромного декодера, информационного слова на выходе мажоритарного декодера (\hat{a}). Об исправлении ошибки свидетельствует равенство информационных векторов на входе кодера (a) и выходах декодеров (\hat{a}). Об обнаружении ошибки свидетельствует единичный флаг наличия ошибки (f) на выходе синдромного декодера.

ВНИМАНИЕ. Если, согласно предварительному расчету, код должен обнаруживать и исправлять любую одиночную ошибку, а на практике этого не происходит, то следует найти и устранить ошибку в предварительном расчете или в схеме кодера (декодера).

8. Оценить способность кода обнаруживать и исправлять двойные ошибки. Для этого, перебирая различные значения векторов ошибок, необходимо, по возможности, зафиксировать три случая: а) ошибка обнаруживается и исправляется; б) ошибка обнаруживается, но не исправляется; в) ошибка не обнаруживается. В таблицу эксперимента занести данные ($e, y, \hat{a}, S, f, \hat{a}$) по двум примерам для каждого случая.

9. Повторить п. 8 для ошибок кратности три, четыре и так далее, до тех пор, пока не будет найдена хотя бы одна комбинация ошибок, которая не обнаруживается данным кодом.

Анализ результатов и выводы

10. По данным, полученным в п. 5, определить мощность кода и вычислить кодовое расстояние. Сравнить с результатами предварительного расчета.

11. По данным, полученным в п. 7–9, сделать вывод о максимальной кратности обнаруживаемых и исправляемых кодом ошибок. Сравнить с результатами предварительного расчета.

12. В таблице эксперимента сравнить значения синдромов для исправленных и не исправленных ошибок. Объяснить причину невозможности исправления кодом сразу всех указанных ошибок.

13. По таблице эксперимента проанализировать ошибки, которые не были обнаружены. Для этого сравнить значения векторов этих ошибок и кодовых слов на выходе канала связи с разрешенными кодовыми комбинациями. Сделать вывод о том, какие вектора ошибок не могут быть обнаружены и почему.

5. СОДЕРЖАНИЕ ОТЧЕТА

1. Формулировка цели работы.
2. Исходные данные.
3. Результаты выполнения предварительного задания, включая необходимые расчеты и схемы.
4. Таблица экспериментальных данных.
5. Анализ результатов и выводы.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Объясните, почему для исправления ошибок необходимо передавать информацию с избыточностью.
2. Назовите основные параметры блочных кодов.
3. Почему код называется групповым, и что такое группа?
4. Как задается линейный код?
5. Что такое кодовое расстояние и как определить корректирующую способность кода?
6. Как закодировать информационное сообщение?

7. Что такое синдром ошибки и как он используется для обнаружения и исправления ошибок в кодовом слове?

8. В чем суть мажоритарного метода декодирования?

ЛИТЕРАТУРА

1. Теория прикладного кодирования: Учеб. пособие. В 2 т. /Под. ред. В.К. Конопелько. – Мн.: БГУИР, 2004.

2. Кларк Дж., Кейн Дж. Кодирование с исправлением ошибок в системах цифровой связи: Пер. с англ. – М.: Радио и связь, 1987.

3. Блейхут Р. Теория и практика кодов, контролирующих ошибки: Пер. с англ. – М.: Мир, 1986.

4. Кассама Т., Токура Н., Ивадари Е. и др. Теория кодирования: Пер. с яп. – М.: Мир, 1978.

5. Питерсон У., Уэлдон Э. Коды, исправляющие ошибки: Пер. с англ. – М.: Мир, 1976.

6. Мак-Вильямс Ф. Дж., Слоэн Н. Дж. А. Теория кодов, исправляющих ошибки: Пер. с англ. – М.: Связь, 1979.

7. Бурак А. И. Моделирование импульсных и цифровых устройств в среде Electronics Workbench: Метод. пособие. – Мн.: БГУИР, 2000.

Учебное издание

Семашко Павел Геннадьевич

КОДИРОВАНИЕ И ДЕКОДИРОВАНИЕ ЛИНЕЙНЫХ ГРУППОВЫХ КОДОВ

Методическое пособие

к лабораторной работе по дисциплинам

**ТЕОРИЯ КОДИРОВАНИЯ И ЗАЩИТА ИНФОРМАЦИИ,
МЕТОДЫ КОДИРОВАНИЯ И ЗАЩИТА ИНФОРМАЦИИ,
ПРИКЛАДНАЯ ТЕОРИЯ КОДИРОВАНИЯ И ЦИФРОВАЯ
ОБРАБОТКА СИГНАЛОВ**

для студентов радиотехнических специальностей

всех форм обучения

Ответственный за выпуск П.Г. Семашко

Подписано в печать 26.05.06.
Гарнитура «Таймс».
Уч.-изд. л. 1,3.

Формат 60х84 1/16.
Печать ризографическая.
Тираж 200 экз.

Бумага офсетная.
Усл. печ. л. 1,51.
Заказ 165.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИИ №02330/0056964 от 01.04.2004. ЛП №02330/0131518 от 30.04.2004.
220013, Минск, П. Бровки, 6