УДК 519.725

И.А. ЗАВАДСКИЙ

МЕТОД ДЕКОДИРОВАНИЯ ПОМЕХОУСТОЙЧИВОГО КОДА ПЕРЕМЕННОЙ ДЛИНЫ НА ОСНОВЕ КОНЕЧНЫХ АВТОМАТОВ

Аннотация. Рассмотрен алгоритм декодирования для кода, основанного на обработке информационных сообщений конечными автоматами и использовании двухбазисной системы исчисления, а также оценена его эффективность. Кроме того, описан общий алгоритм кодирования. Как кодирование, так и декодирование осуществляется с помощью двухуровневой системы: на внутреннем уровне входное сообщение представлено в виде нижнего (2,3)-кода, а на внешнем помехоустойчивые свойства этого кода усиливаются путем его преобразования конечным автоматом специального вида. При декодировании ошибки улавливаются, прежде всего, на внешнем уровне, однако если этого не происходит, результат «подчищается» на внутреннем уровне. Исследована взаимосвязь внешнего уровня рассматриваемой системы со сверточными кодами и показаны преимущества предложенного метода.

Ключевые слова: конечный автомат, помехоустойчивый код, (2,3)-код, сверточный код.

ВВЕДЕНИЕ

Принципы генерирования помехоустойчивых кодов с помощью конечных автоматов, а также два таких автомата, работа которых основана на различных математических подходах, описаны в [1]. В данной статье рассмотрена двухуровневая система помехоустойчивого кодирования, в которой используются оба автомата, и приведен метод эффективного декодирования генерируемого этой системой кода. Мощные помехоустойчивые свойства предложенной системы обеспечиваются благодаря сочетанию на ее внешнем и внутреннем уровнях разнородных подходов, состоящих в генерировании символов кода на основе нескольких входных двоичных символов и выполнении арифметических операций над большим блоком символов входного кода как над целым числом. Эффективность исправления ошибок данной системой существенно выше показателей известных простых кодов той же скорости, особенно при высоком уровне шума в каналах связи (отношение полезного сигнала к шуму Eb/No меньше 2 дБ). Так, при значении величины Eb/No, равном 1 дБ, пакетная помехоустойчивость предложенного кода превышает аналогичный показатель широко используемого в системах мобильной связи и хранения данных кода NASA (171,133) более чем в 12 раз.

ОБЩИЙ АЛГОРИТМ КОДИРОВАНИЯ

Входящую двоичную последовательность будем делить на L-битовые блоки. К каждому блоку дописывается контрольная сумма длиной c бит, использующаяся при декодировании как критерий его правильности. Затем блок преобразуется в число из множества $\mathbf{N}_{2,3}$ (множество чисел, взаимно простых с 2 и 3), для которого генерируется помехоустойчивый (2,3)-код автоматом из [1, рис. 1], и к этому коду применяется автомат из [1, рис. 2].

Опишем работу алгоритма.

1. Делим входную последовательность битов на блоки длиной L.

2. Дописываем к каждому блоку c-битовую контрольную сумму, i-й бит которой равен сумме по модулю 2 всех битов блока с номерами $i(\mathrm{mod}\, c)$.

3. Полученную $(L+c)$-битовую последовательность преобразуем в число из множества $\mathbf{N}_{2,3}$ по описанному ниже алгоритму.

4. Для этого числа строим нижний (2,3)-код, переписываем последовательность его (Δ, k)-групп справа налево и по ней генерируем помехоустойчивый (2,3)-код с помощью внутреннего автомата [1, рис. 1].

5. К полученному на предыдущем шаге коду применяем автомат из [1, рис. 2].

6. Помехоустойчивые коды блоков (кодовые слова) конкатенируем без добавления разделителей.

Заметим, что при использовании пакетного принципа передачи сообщений, применяемого во всех системах мобильной связи, величины L и c можно подобрать так, чтобы кодовое слово укладывалось в пакет (дисперсия длин кодовых слов невелика). Тогда на шаге 6 описанного алгоритма следует не конкатенировать кодовые слова, а дополнять их нулями до размера пакета. Если биты передаются потоком, то для повышения помехоустойчивости в некоторые фиксированные позиции в коде нужно вставлять маркеры конца кодового слова, определяющие, насколько отклоняется длина кодового слова от позиции самого маркера. Однако использование маркеров детально рассматривать не будем, так как преимущества данного метода кодирования проявляются именно при пакетной передаче данных.

Опишем подробно, как выполняется шаг 3 алгоритма (преобразование произвольной последовательности длиной M бит в целое число, взаимно простое с 2 и 3):

1) если последовательность начинается с символа 1 и представляет собой взаимно простое с 2 и 3 двоичное число, преобразование выполнять не нужно;

2) добавляем символ 1 к последовательности слева; если получено взаимно простое с 2 и 3 число, то это и есть результат; иначе переходим к шагу 3;

3) добавляем символ 1 к последовательности справа; если получено взаимно простое с 2 и 3 число, то это и есть результат; иначе переходим к шагу 4;

4) добавляем символ 1 к последовательности справа.

Обратная процедура преобразует число из множества $\mathbf{N}_{2,3}$ во входную последовательность битов. Она может завершиться ошибкой, которая означает, что это число нельзя получить в результате прямого преобразования M-битовой последовательности.

Опишем обратную процедуру пошагово:

1) если битовая длина числа меньше M или больше $M+3$, завершаем процедуру с ошибкой;

2) если битовая длина числа равна $M+3$, удаляем наименее значимый бит; если полученное число взаимно простое с 2 и 3, завершаем процедуру с ошибкой;

3) если битовая длина полученного на предыдущем шаге числа равна $M+2$, удаляем наименее значимый бит; если полученное число взаимно простое с 2 и 3, завершаем процедуру с ошибкой;

4) если битовая длина полученного на предыдущем шаге числа равна $M+1$, удаляем наиболее значимый бит.

АЛГОРИТМ ДЕКОДИРОВАНИЯ

Декодирующий алгоритм включает два этапа: преобразование кода, получаемого на выходе внешнего автомата [1, рис. 2], в сжатый (2,3)-код; получение из помехоустойчивого (2,3)-кода сообщения, близкого к исходному, с помощью внутреннего автомата [1, рис. 1]. Однако эти этапы выполняются не последовательно: сжатый (2,3)-код декодируется по мере его генерирования на первом этапе, с некоторым запаздыванием.

Суть метода исправления ошибок состоит в следующем. Если среди трех битов с номерами $3m+1$, $3m+2$, $3m+3$ есть один ошибочный, при обработке внешним автоматом этой триады возникнет ошибка, поскольку, как показано в [1], автомат будет ожидать триаду из множества кодовых слов A, а считает три-

аду из множества кодовых слов B или наоборот. Будем последовательно предполагать, что ошибочным является бит $3m+1, 3m+2$ и $3m+3$, инвертировать его и запускать внешний автомат дальше, а к генерируемому им (2,3)-коду применять внутренний автомат. Если предположение об ошибочном бите верно, внешний (а за ним — и внутренний) автомат будет работать успешно вплоть до следующего ошибочного бита. В противном случае вместо одной ошибки в триаде битов образуется две, и в соответствии с доказанными в [1, разд. 3] свойствами сбой в работе внешнего автомата произойдет в одной из двух последующих триад. Все три варианта исправления ошибок включим в список потенциальных месторасположений ошибок, однако вариант, в котором автомат при том же количестве исправлений пройдет дальше, будем считать предпочтительным и подлежащим дальнейшей обработке в первую очередь. Кроме того, рассмотрим и включим в список потенциальных месторасположений ошибок триады с двумя инвертированными битами, расположенные на одну или две триады до места сбоя при работе внешнего автомата, и триады с тремя инвертированными битами в месте сбоя. Если в силу высокой плотности и особой конфигурации ошибок сбоя внешнего автомата не произойдет, высока вероятность возникновения ошибки при работе внутреннего автомата.

Заметим, что алгоритм декодирования применим и к жесткой, и к мягкой реализации декодера. В первом случае предполагается, что на вход декодера поступают биты, а во втором — амплитуда сигнала, кодирующего каждый бит на входе декодера. Она является непрерывной величиной, получаемой в результате наложения на уровень передаваемого сигнала (1 — для единичного значения бита и -1 — для нулевого) аддитивного гауссового белого шума в канале связи. При мягкой реализации положительную амплитуду сигнала отождествляем с единичным битом, отрицательную — с нулевым, а тот факт, что ее значения могут не быть целочисленными, используется только в описанной ниже операции сравнения корректирующих конфигураций, что повышает помехоустойчивые свойства кода при использовании мягкого декодера.

Предположим, что в некоторые биты кодового слова внесены ошибки, т.е. эти биты инвертированы. Множество позиций битов, которые считаем ошибочными, назовем маской ошибок. Пусть $q = 3m+1$ — номер первого бита некоторой триады кодового слова, $\upsilon = (\upsilon_1, \ldots, \upsilon_t)$, $\upsilon_1 < \ldots < \upsilon_t$, — маска ошибок, w — результат декодирования внешним автоматом части кодового слова, начинающейся с первого (наиболее значимого) бита и заканчивающейся битом q, в случае инвертирования битов, принадлежащих маске υ. Кроме того, обозначим s состояние внешнего автомата, соответствующее положению головки q, т.е. информацию как о группе состояний, так и о номере состояния внутри группы. Внешней корректирующей конфигурацией назовем четверку (υ, w, q, s), внутренней — четверку (x_w, q_1, a, t_{out}), где $t_{out} = (\upsilon, w, q, s)$ — некоторая внешняя корректирующая конфигурация, $q_1 < q$ — кратное трем число, x_w — результат декодирования внутренним автоматом части слова w, начинающейся с первого бита и заканчивающейся битом q_1, a — состояние внутреннего автомата, соответствующее положению головки q_1.

Для описания алгоритма декодирования необходимо определить несколько вспомогательных операций над корректирующими конфигурациями.

Для внешней и внутренней корректирующих конфигураций определим операции инкремента. Пусть $t_{out} = (\upsilon, w, q, s)$ — некоторая внешняя конфигурация, а внешний автомат, начинающий с нее работу, останавливается с ошибкой на бите $q' \geq q$ в состоянии s', генерируя при этом последовательность w'. Тогда результатом инкремента $t_{out}{+}{+}$ будем считать конфигурацию $t'_{out} = (\upsilon, w', q', s')$. Если автомат достигнет конца всего кода без ошибки, полагаем величину q' равной номеру последнего бита кода.

Результатом инкремента внутренней конфигурации $t_{in} = (x_w, q_1, a, t_{out})$ считаем внутреннюю конфигурацию $t_{in}{+}{+} = (x_{w'}, q_1, a', t_{out}{+}{+})$, получаемую в результате обработки внутренним автоматом, находящимся в состоянии a, части слова w', включающей биты с номерами от q_1 до q_2. Здесь $(\upsilon, w', q', s') = t_{out}{+}{+} = (\upsilon, w, q, s){+}{+}$, $q_2 = q' - 3d$, где d — некоторая константа, а $x_{w'}$ и a' — соответствующие q_2 декодированная часть слова и состояние внутреннего автомата (рис. 1, *а*). Принцип выбора величины d поясняется ниже. Инкремент внутренней конфигурации может завершиться ошибкой до того, как автомат перейдет к обработке триады, начинающейся с бита q_2.

Корректирующие конфигурации могут считаться более или менее успешными: чем большее количество битов удалось обработать и чем меньше битов для этого пришлось инвертировать (чем короче маска), тем более вероятно, что элементы маски соответствуют действительному положению ошибок. Поэтому введем операцию сравнения внешних конфигураций p и m, предположив, что в соответствующих им словах обработано q_p и q_m битов (результат сравнения внутренних конфигураций совпадает с результатом сравнения входящих в их состав внешних конфигураций). Введем также метрику (t_p и t_m), которая по-разному вычисляется в случае жесткой и мягкой реализации декодера. При жесткой реализации величина t_p равна количеству битов в маске конфигурации p, а при мягкой — сумме модулей или квадратов амплитуд, соответствующих битам маски. В обоих случаях, чем меньше метрика, тем более правдоподобна корректирующая конфигурация.

Полагаем, что если $p > m$, то конфигурация p хуже конфигурации m и из них двух в первую очередь дальнейшей обработке подлежит конфигурация m. Итак, операция $p > m$ выполняется следующим образом (порядок выполнения сравнений существен):

1) если $q_p < q_m$ и $t_p > t_m$, результат истинный;

2) если $q_p > q_m$ и $t_p < t_m$, результат ложный;

3) для жесткой реализации: если $q_p < q_m - 12(t_m - t_p) - 4(t_m - t_p)^2$, результат истинный; для мягкой реализации: если $t_p < t_m$ и $q_p < q_m - 20(t_m - t_p) - 8(t_m - t_p)^2$, результат истинный;

4) для жесткой реализации: если $q_m < q_p - 12(t_p - t_m) - 4(t_p - t_m)^2$, результат ложный; для мягкой реализации: если $t_p > t_m$ и $q_m < q_p - 20(t_p - t_m) - 8(t_p - t_m)^2$, результат ложный;

5) если $t_p > t_m$, результат истинный;

6) если $t_p < t_m$, результат ложный;

7) если ни одно из условий 1)–6) не выполняется, результат ложный.

Таким образом, прежде всего проверяется явное преимущество одной конфигурации над другой: и метрика меньше, и количество обработанных битов больше. Если же, например, метрика в конфигурации p меньше, однако и количество обработанных битов меньше, то результат зависит от значения эвристической функции, использованной в пп. 3) и 4). Если в конфигурациях равны и метрики, и количества обработанных битов, то конфигурации равноценны и для определенности считаем результат операции их сравнения ложным.

Пусть t — некоторая конфигурация, а P — список корректирующих конфигураций. Введем операцию пополнения списка $P + t$, в результате которой конфигурация t вставляется в список P перед такой ближайшей к началу списка конфигурацией p, что $p > t$, а если такой конфигурации нет — то в конец списка. Формирование списка через операции пополнения гарантирует его упорядоченность по возрастанию, т.е. в вершине списка всегда будет находиться «наилучшая» конфигурация, подлежащая дальнейшей обработке в первую очередь.

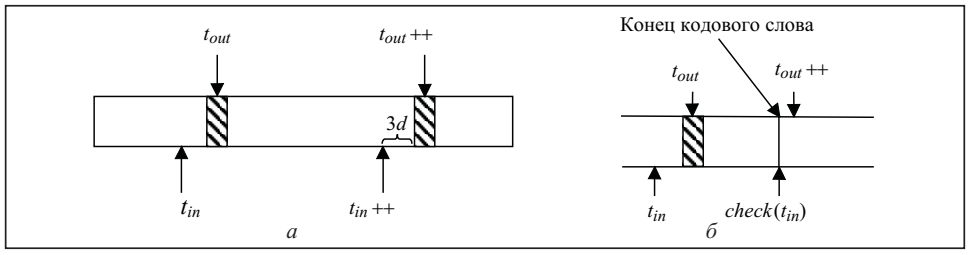

Рис. 1. Положение головки автомата до и после операций инкремента (*а*) и проверки (*б*); заштрихованы триады битов, в которых происходит сбой

Для внутренней конфигурации t_{in} введем операцию проверки $check(t_{in})$, определяющую, генерирует ли внутренний автомат, начинающий работу в этой конфигурации, число x^*, которое проходит все проверки на соответствие входящей битовой последовательности, а именно:

— число x^* без ошибки преобразуется из множества $\mathbf{N}_{2,3}$ во входную последовательность длиной $L+c$ битов;

— контрольная сумма, вычисленная на первых L битах полученной последовательности, совпадает с последними c битами.

Результат операции $check(t_{in})$ является истинным, если удовлетворяющее таким условиям число x^* получено до обработки бита q' (т.е. позиции из соответствующей внешней конфигурации $t_{out}++=(\upsilon, w', q', s')$) и до возникновения ошибки в работе внутреннего автомата (рис. 1, *б*).

Введем также операцию пополнения маски ошибок, которую обозначим символом +. Ее левым операндом будет внутренняя конфигурация, а правым — номер бита, добавляемого к маске ошибок: $t'_{in} \leftarrow t_{in}+k$. Значения параметров конфигурации t'_{in} совпадают со значениями параметров конфигурации t_{in}, за тем исключением, что маска ошибок входящей в ее состав внешней конфигурации дополняется битом k.

Кроме того, введем операцию отката внутренней конфигурации на кратное трем количество битов: $t'_{in} \leftarrow rollback(t_{in}, 3d)$. Если $t_{in}=(x_w, q_1, a, t_{out})$, то в конфигурации $t'_{in}=(x', q_1-3d, a', t'_{out})$ головка внутреннего автомата находится в позиции q_1-3d, состояние внутреннего автомата a' и результирующее число x' соответствуют этому положению головки, а все параметры внешней конфигурации t'_{out} — позиции q_1-3d головки внешнего автомата.

Опишем алгоритм декодирования, исправляющий ошибки.

1. Формируем начальные конфигурации: внешнюю $t_{out} \leftarrow (\{\}, "", 1, \{1,1\})$ и внутреннюю $t_{in} \leftarrow ("", 1, 1, t_{out})$. Во внешней конфигурации множество ошибок и результирующая строка пустые, головка автомата расположена на первом (самом старшем) бите, а сам автомат находится в первом состоянии первой группы; во внутренней — входная строка пустая, головка расположена на первом бите, а автомат находится в первом состоянии. Формируем также список допустимых внутренних конфигураций P_{in}, состоящий сначала из одной начальной конфигурации: $P_{in} \leftarrow \{t_{in}\}$.

2. Пусть h — первый элемент списка допустимых внутренних конфигураций. Выполняем присваивание $t_{in} \leftarrow h$ и исключаем h из списка.

3. Осуществим операцию инкремента внутренней конфигурации: $t'_{in} \leftarrow t_{in}++=$ $=(x_{w'}, q_1-3d, a', t_{out}++)=(x', q_2, a', t'_{out})$. Сначала выполняется инкремент соответствующей внешней конфигурации $t_{out}++$, в результате которого внешний автомат останавливается в позиции q_1, генерируя на выходе слово w', затем оно обрабатывается внутренним автоматом. Ниже объясняется принцип выбора ве-

личины d. Заметим, что ее оптимальным значением является 2, т.е. внутренний автомат должен останавливаться на две триады раньше той позиции, где остановился внешний автомат. Если операция $t_{in}{+}{+}$ завершилась ошибкой до достижения головкой внутреннего автомата позиции $q_1 - 3d$ и список P_{in} не пустой, возвращаемся к шагу 2, иначе переходим к шагу 4.

4. Выполняем проверку внутренней конфигурации, полученной в результате инкремента: $check(t'_{in})$. Если эта операция возвращает истинное значение, генерируя число x^*, то x^* считаем результатом декодирования кодового слова. Остается применить к этому числу обратное преобразование из множества $\mathbf{N}_{2,3}$ и отбросить биты контрольной суммы. Если результат $check(t'_{in})$ не является истинным, переходим к шагу 5.

5. Предполагаем, что сбой в работе внешнего автомата вызван единичной ошибкой в триаде битов с номерами q_1, q_1+1, q_1+2. Для значений $i = 0, 1, 2$ выполняем операцию добавления бита к маске: $t''_{in} \leftarrow t'_{in} + (q_1 + i)$ и пополняем список допустимых внутренних конфигураций: $P_{in} \leftarrow P_{in} + t''_{in}$.

6. Предполагаем, что сбой в работе внешнего автомата в триаде, начинающейся с позиции q_1, вызван двойной ошибкой в одной из двух предшествующих триад. Выполняем откат конфигурации t'_{in} на шесть битов и к маске полученной конфигурации добавляем все возможные пары битов i и k, принадлежащих одной триаде битов и находящихся в диапазоне от $q_1 - 6$ до $q_1 - 1$ включительно: $t''_{in} \leftarrow rollback(t'_{in}, 6) + i + k$. Пополняем список допустимых внутренних конфигураций конфигурацией t''_{in}: $P_{in} \leftarrow P_{in} + t''_{in}$.

7. Предполагаем, что все биты в триаде, начинающейся с позиции q_1, содержат ошибки: $t''_{in} \leftarrow t'_{in} + q_1 + (q_1+1) + (q_1+2)$. Дополняем список допустимых внутренних конфигураций конфигурацией t''_{in}.

8. Возвращаемся к шагу 2.

Наличие «зазора» в $3d = 6$ бит между точками окончания инкремента внешней и внутренней конфигураций (шаг 3) объясняется возможностью отката, выполняемого на шаге 6. Если остановка внешнего автомата в позиции q_1 связана с двойной ошибкой в одной из предыдущих двух триад, нельзя «требовать» от внутреннего автомата безошибочной обработки битов с номерами $q_1 - 6, \ldots, q_1 - 1$.

Далее объясним, для чего на шаге 4 алгоритма кодирования (Δ, k)-группы переписывались справа налево. Так как декодирование нижнего (2,3)-кода начинается с величины $x_t = 1$ или $x_t = 2$, (Δ, k)-группа, получаемая при кодировании последней, должна обрабатываться при декодировании первой. Если бы порядок групп не изменялся, позиция такой (Δ, k)-группы была бы неизвестной, поскольку длина кодового слова является переменной величиной.

СВЯЗЬ ВНЕШНЕГО АВТОМАТА СО СВЕРТОЧНЫМИ КОДАМИ

Сверточный код скорости l/n с памятью m бит можно представить в виде конечного автомата с 2^m состояниями и 2^l переходами из каждого состояния [3]. Переход выбирается исходя из значения l двоичных входящих символов, которые считывает автомат. При переходе автомат генерирует n символов выходного кода, определяемых n формулами, одинаковыми для всех состояний. Каждая формула представляет собой сумму по модулю 2 некоторого поднабора входных битов. Максимальная разность в этих формулах между номерами битов составляет m. Таким образом, диаграмма состояний сверточного кода — специфическая разновидность конечного автомата, которая должна удовлетворять жестким ограничениям, однако изображенный в [1, рис. 2] автомат этим ограничениям удовлетворяет. Если отождествлять его с диаграммой состояний сверточного кода, то это должен быть код скорости 2/3 с памятью 4. Тем не

менее, как показано далее, данный автомат определяет сверточный код скорости 2/3 с памятью 8. Различие между кодами с памятью 4 и 8 существенно, так как и эффективность, и сложность декодирования при использовании наиболее распространенного метода Витерби растет с увеличением памяти кода экспоненциально. Код, генерируемый рассматриваемым автоматом, обладает памятью 8, однако может быть декодирован алгоритмом с той же временной сложностью, что и код памяти 4.

Итак, выведем формулы, позволяющие интерпретировать внешний автомат как сверточный код. Пусть x_1x_0 — символы, считанные автоматом на некоторой итерации (x_3x_2, x_5x_4 и т.д. — символы, считанные на предыдущих итерациях), $y_2y_1y_0$ — символы, записываемые при последующем переходе, k_0 — бит, определяющий тип кодировки: $k_0 = 0$ для кодировки A и $k_0 = 1$ для кодировки B. Как видно из табл. 1, приведенной в [1],

$$y_2 = x_1 \oplus k_0,$$
$$y_1 = x_0 \oplus k_0, \qquad (1)$$
$$y_0 = x_0 \oplus x_1 \oplus k_0.$$

Если рассматривать номер группы состояний и номер состояния внутри группы как двоичные двухбитовые числа g_1g_0 и s_1s_0 соответственно, то, как видно из рис. 2 в [1],

$$k_0 = s_0 \oplus g_1 \text{ и в общем случае } k_{2i} = s_{2i} \oplus g_{2i+1}, \qquad (2)$$

$$g_1g_0 = s_3s_2 \text{ и в общем случае } g_{2i+1}g_{2i} = s_{2i+3}s_{2i+2}. \qquad (3)$$

Рассмотрим второй и последний столбцы табл. 1 в [1], определяющие зависимость между считанным числом (предположим, что это число x_3x_2) и состоянием автомата на следующей итерации (s_1s_0). Формула, связывающая данные числа, зависит от значения бита g_2, т.е. от четности номера группы состояний. В табл. 1, приведенной в данной работе, отображена зависимость между младшим битом g_2 номера группы состояний, считанным числом x_3x_2 и состоянием внутри группы на следующей итерации s_1s_0.

Таким образом,

$$s_0 = x_2 \oplus 1 \text{ и в общем случае } s_{2i} = x_{2i+2} \oplus 1, \qquad (4)$$

$$s_1 = x_2 \oplus x_3 \oplus g_2 \text{ и в общем случае } s_{2i+1} = x_{2i+2} \oplus x_{2i+3} \oplus g_{2i+2}. \qquad (5)$$

Из последнего равенства, с учетом формулы (3), получаем $s_1 = x_2 \oplus x_3 \oplus s_4$. Выразив s_4 по формуле (4), получим $s_1 = x_2 \oplus x_3 \oplus x_6 \oplus 1$, а значит, $s_3 = x_4 \oplus x_5 \oplus x_8 \oplus 1$. В соответствии с формулой (3) последнее выражение равно g_1. Учитывая соотношение (4), из (2) получаем $k_0 = x_2 \oplus x_4 \oplus x_5 \oplus x_8$. Подставляя это выражение в формулы (1), имеем

$$y_2 = x_1 \oplus x_2 \oplus x_4 \oplus x_5 \oplus x_8,$$
$$y_1 = x_0 \oplus x_2 \oplus x_4 \oplus x_5 \oplus x_8,$$
$$y_0 = x_0 \oplus x_1 \oplus x_2 \oplus x_4 \oplus x_5 \oplus x_8.$$

Данные формулы определяют несистематический сверточный код скорости 2/3 с памятью 8 и порождающей матрицей

Таблица 1

g_2	x_3x_2	s_1s_0
0	00	01
	01	10
	10	11
	11	00
1	00	11
	01	00
	10	01
	11	10

$$\begin{pmatrix} 1+D^2 & D^2 & 1+D^2 \\ D+D^2+D^4 & 1+D+D^2+D^4 & 1+D+D^2+D^4 \end{pmatrix},$$

что в общепринятой для сверточных кодов записи полиномов в восьмеричной форме соответствует матрице $\begin{pmatrix} 11 & 10 & 11 \\ 26 & 27 & 27 \end{pmatrix}$.

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ЭФФЕКТИВНОСТИ КОДОВ

Рассмотренный помехоустойчивый код является двухуровневым, причем на разных уровнях реализованы принципиально различные математические подходы, сочетание которых дает возможность достичь высокой помехоустойчивости. На рис. 2 приведены графики, иллюстрирующие эффективность описанного метода по сравнению с наиболее известным сверточным кодом NASA (171,133) скорости 1/2, т.е. такой же, как и средняя скорость предложенного кода. Как видно, данный код обеспечивает существенно более низкий уровень ошибок в пакетах (Frame Error Rate — FER). Эта величина измеряется как отношение количества пакетов с ошибками к общему количеству переданных пакетов. Для моделирования выбран канал связи с аддитивным гауссовым белым шумом (AWGN), по которому передаются пакеты со средней длиной 100 бит, что соответствует 50-битовому пакету на входе канала связи при 6-битовой контрольной сумме.

Применение только внешнего кодирования позволило бы исправлять так называемый мелкий дождь ошибок, т.е. ошибки, расположенные относительно далеко одна от другой. Из описания приведенных в [1] методов исправления ошибок следует, что единичные и тройные ошибки в триадах будут исправлены с вероятностью 100 %, если после триады с ошибками следует две триады без ошибок. Также с вероятностью 100 % будут исправлены двойные ошибки в триадах, после которых следует шесть триад без ошибок.

Комбинирование внешнего кодирования с внутренним позволяет успешно исправлять значительно более плотные конфигурации ошибок. Однако в этом случае определить теоретически все варианты кодовых последовательностей, исправляемых с вероятностью 100 %, не представляется возможным и при некоторых слож-

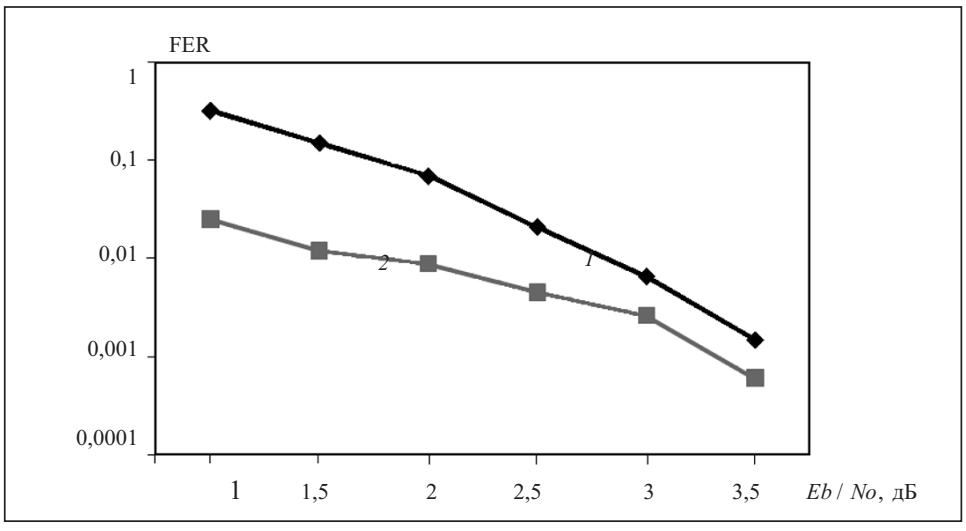

Рис. 2. Графики вероятности ошибки в пакете для помехоустойчивости кода NASA(171,133) (*1*) и предложенного кода (*2*)

ных конфигурациях ошибок алгоритм декодирования может не срабатывать. О наличии такой конфигурации ошибок свидетельствует, прежде всего, длительное время работы алгоритма. Поэтому если количество итераций превышает некоторое предопределенное, зависящее от интенсивности помех в канале связи, число (для неизвестного уровня помех полагаем его равным 200), то работу алгоритма над кодовым словом следует остановить, не внося в него никаких изменений. Это объясняется тем, что если время работы алгоритма декодирования становится недопустимо большим, он с высокой вероятностью в итоге найдет число x^*, проходящее все проверки, но отличающееся от закодированного числа, т.е. внесет дополнительные ошибки во входное сообщение.

ЗАКЛЮЧЕНИЕ

Поскольку внешний код, по сути, сверточный, в целях повышения быстродействия можно было бы проводить внешнее декодирование по методу Витерби [3], а внутренний (2,3)-код использовать для отсева конфигураций ошибок только в том случае, если результат, полученный с помощью метода Витерби, не проходит проверок на соответствие входящей битовой последовательности. Однако метод Витерби хорошо работает лишь при низкой зашумленности канала, а преимущества рассмотренного подхода проявляются, прежде всего, при высоком уровне шума. Более того, роль внутреннего кодирования при таком подходе нивелируется и помехоустойчивость кода в целом оказывается существенно ниже.

Описанный в данной статье алгоритм можно рассматривать как базовый. Его можно улучшить путем исследования дополнительных конфигураций ошибок: например, всех конфигураций, в которых две соседние триады содержат ошибки. Это несколько увеличивает время работы алгоритма, однако повышает его помехоустойчивость.

СПИСОК ЛИТЕРАТУРЫ

1. Завадский И.А. Помехоустойчивые коды переменной длины на основе конечных автоматов // Кибернетика и системный анализ. — 2015. — **51**, № 2. — С. 43–51.
2. Анисимов А.В., Завадский И.А. Помехоустойчивое префиксное кодирование на основе нижнего (2,3)-представления чисел // Кибернетика и системный анализ. — 2014. — **50**, № 2. — С. 3–14.
3. Johannesson R., Zigangirov K. Fundamentals of convolutional coding. — New York: IEEE Press, 1999. — 400 p.

Поступила 23.04.2014