УДК 519.725

И.А. Завадский

Повышение эффективности сверточных помехоустойчивых кодов с помощью конечных автоматов

Предложен метод повышения эффективности сверточных помехоустойчивых кодов без увеличения вычислительных затрат на кодирование и декодирование. Сверточные коды рассматриваются как частный случай более общей схемы, где кодирование и декодирование выполняются конечным автоматом определенной структуры. Метод позволяет почти вдвое повысить уровень помехоустойчивости лучшего из известных сверточных кодов.

The method of the convolutional codes empowering without increasing the computational complexity is presented. The convolutional code is considered as a partial case of more general schema where decoding and encoding are performed by a finite automaton of the certain structure. The proposed method outperforms the best convolutional code of the same graph size almost twice, having the same computational complexity.

Запропоновано метод підвищення ефективності згорткових завадостійких кодів без збільшення обчислювальних витрат на кодування та декодування. Згорткові коди розглядаються як окремий випадок більш загальної схеми, де кодування та декодування виконуються скінченним автоматом певної структури. Метод дає змогу майже вдвічі покращити показники завадостійкості найкращого згорткового коду.

Введение. Сверточные коды – это один из наиболее изученных и широко используемых методов помехоустойчивого кодирования. Коды просты в реализации, но эффективность даже самых лучших кодов этого класса далека от теоретического *предела Шеннона* [1]. Поэтому в последнее десятилетие во многих практических приложениях предпочтение отдается более мощным кодам с низкой плотностью проверок на четность (*LDPC*-кодам) [2] и турбо-кодам [3]. Однако высокая эффективность упомянутых помехоустойчивых кодов достигается не «бесплатно»: *LDPC*-коды эффективны только при достаточно больших длинах блоков входных данных, а турбо-коды представляют собой многоуровневую схему, на каждом уровне которой реализован некий базовый, более простой код (который может быть в частности сверточным кодом), а значит, и вычислительная сложность кодирования и декодирования для турбо-кодов будет существенно выше, чем для базовых сверточных кодов.

В данной статье приведен метод повышения эффективности сверточных кодов, не требующий дополнительных вычислительных затрат. Как известно, любой сверточный код может быть сгенерирован конечным автоматом специального вида, удовлетворяющим определенным ограничениям на количество состояний, количество переходов из каждого состояния и пр. Идея метода состоит в том, чтобы рассматривать в качестве генераторов кода более широкий класс конечных автоматов, что, возможно, позволит получить код с лучшими помехоустойчивыми свойствами. При этом, как показано далее, вычислительная сложность кодирования и декодирования не повышается.

Постановка задачи

Исследуем свойства конечных автоматов, позволяющих генерировать эффективные помехоустойчивые коды, т.е. коды, существенно снижающие вероятность ошибки на бит при передаче сообщений по каналу связи с помехами. С этой целью следует описать метод ко-

Ключевые слова: конечный автомат, помехоустойчивый код, сверточный код.

дирования посредством конечных автоматов общего вида, метод декодирования, алгоритм построения эффективных кодирующих автоматов, а также методы определения важнейших характеристик кодов, генерируемых конечными автоматами.

Помехоустойчивое кодирование с использованием конечных автоматов

Рассмотрим конечный автомат, из каждого состояния которого существует 2^k переходов в другие состояния. Предположим, что переходы выбираются исходя из значения k двоичных символов входной последовательности битов и на каждом переходе автомат генерирует t битов кодового слова, где $k < t$. Таким образом код, генерируемый автоматом, будет иметь скорость k / t. Например, на рис. 1 изображен автомат с четырьмя состояниями, генерирующий код скорости которого ½. Входные символы записаны над стрелками переходов перед косой чертой, а генерируемые – после косой. Предположим также, что состояния автомата пронумерованы и он начинает работу в состоянии ноль.

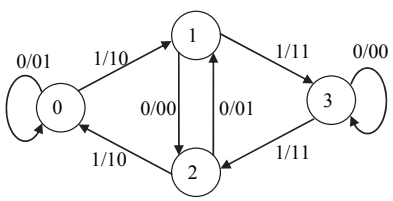

Рис. 1. Автомат, генерирующий код скорости которого ½

Одна из существенных характеристик качества кода это $d_{\min}$ – минимальное расстояния Хэмминга между кодовыми словами. Опишем метод вычисления величины $d_{\min}$ для кода, генерируемого конечным автоматом. Примем, что есть конечный автомат с n вершинами, а F – квадратная матрица, строки и столбцы которой пронумерованы парами индексов ij: $0 \le i < n$, $i \le j < n$. Рассмотрим два экземпляра автомата, один из которых находится в состоянии i, а другой – в состоянии j. Допустим, что через одну итерацию автоматы перейдут в состояния p и k, причем какой из экземпляров автомата в какое именно перейдет состояние – несущественно. Элементу $f_{ij,kp}$ матрицы F присвоим минимальное расстояние Хэмминга между словами, которые могут быть сгенерированы автоматами при таких переходах. Если из состояний i и j в состояния p и k перейти невозможно или если $i = j$ и $p = k$, будем считать, что элемент $f_{ij,kp}$ равен некоторому достаточно большому числу M. Так, для изображенного на рис. 1 автомата матрица F показана в табл. 1 (сначала записаны строки и столбцы ij с различными значениями i и j).

Таблица 1. Матрица минимальных расстояний Хэмминга между кодовыми словами, генерируемыми конечным автоматом

	01	02	03	12	13	23	00	11	22	33
01	*M*	1	1	1	1	*M*	*M*	*M*	*M*	*M*
02	0	*M*	*M*	*M*	*M*	*M*	*M*	2	*M*	*M*
03	*M*	1	1	1	1	*M*	*M*	*M*	*M*	*M*
12	*M*	*M*	*M*	1	1	*M*	*M*	*M*	*M*	*M*
13	*M*	*M*	*M*	*M*	*M*	0	*M*	*M*	2	2
23	*M*	*M*	*M*	1	1	*M*	*M*	*M*	*M*	*M*
00	2	*M*	*M*	*M*	*M*	*M*	*M*	*M*	*M*	*M*
11	*M*	*M*	*M*	*M*	*M*	2	*M*	*M*	*M*	*M*
22	*M*	*M*	*M*	*M*	*M*	*M*	*M*	*M*	*M*	*M*
33	*M*	*M*	*M*	*M*	*M*	2	*M*	*M*	*M*	*M*

Рассмотрим над множеством целых чисел поле, в котором операцией « + » будем считать поиск минимума, а операцией « * » – сложение, причем предположим, что для любого числа a выполняется равенство $M*a = M$. Над матрицами, элементы которых принадлежат этому полю, определена операция умножения и, как можно отметить, в матрице F^d элемент с индексами ij,kp будет равен минимально возможному расстоянию Хэмминга между словами, генерируемыми за d переходов двумя экземплярами автомата, которые начинают работу в состояниях i и j и заканчивают в состояниях k и p.

Очевидно, что минимальное расстояние Хэмминга между генерируемыми автоматом кодовыми словами равно минимальному, по всем возможным комбинациями i и j, расстоянию Хэмминга между различными словами, которые могут быть сгенерированы за одно и то же количество итераций двумя экземплярами автомата, начинающими работу в одном и том же состоянии i и заканчивающими ее в одном и том же состоянии j, т.е. эта величина равна минимальному среди элементов $f_{ii,jj}$ в матрицах F, F^2, F^3 и т.д. Так, для изображенного на рис. 1 автомата, матрица F которого

приведена в табл. 1, все элементы матриц F и F^2 с индексами ii,jj равны M, в матрице F^3 они равны пяти, в матрице F^4 – шести, а начиная с матрицы F^4 – семи и более. Поэтому величина $d_{\min}$ для кода, генерируемого таким автоматом, составляет пять.

Легко показать, что для определения $d_{\min}$ для генерирующего код автомата с n вершинами достаточно рассмотреть матрицы F^d, где $d \le n^2$. Действительно, обозначим через $H(a, b,\ldots;\ x, y,\ldots)$ расстояние Хэмминга между кодовыми словами, генерируемыми двумя экземплярами автомата, один из которых проходит через состояния a, b, …, а другой – через состояния x, y, …, и предположим, что $d_{\min} = H(i_1, \ldots,i_d\ ;\ j_1, \ldots, j_d)$, где $d > n^2$, $i_1 = j_1$, $i_d = j_d$. Поскольку существует всего n возможных значений i_t и j_t, то существует не более n^2 различных пар $(i_t\ , j_t)$, а значит, для некоторых значений k и t, таких, что $1 \le k < t < d$, будет выполняться равенство $(i_k\ , j_k) = (i_t\ , j_t)$ и существуют пути через состояния $i_1, \ldots, i_k, i_{t+1}, \ldots, i_d$ и $j_1, \ldots, j_k, j_{t+1}, \ldots, j_d$, длина которых меньше d. Но, так как $H(i_k, \ldots, i_t\ ;\ j_k, \ldots, j_t) \ge 0$, то $H(i_1, \ldots, i_k, i_{t+1}, \ldots,\ i_d\ ;\ j_1, \ldots, j_k, j_{t+1}, \ldots, j_d) \le H(i_1, \ldots, i_d\ ;\ j_1, \ldots, j_d) = d_{\min}$, а значит, $d_{\min}$ достигается на путях, которые короче d.

Декодирование

Для декодирования сгенерированного автоматом кода можно применять те же методы, что и для сверточных кодов, наиболее известный из которых – метод Витерби. Он предполагает осуществление двух проходов: прямого и обратного. На i-й итерации прямого прохода каждому из состояний j, в которых может пребывать автомат после i-го перехода, приписывается *метрика* m_i^j. Рассматривая двоичную фазовую модуляцию в канале с аддитивным гауссовым белым шумом (АГБШ) [4], и кодируя нулевой бит как сигнал 1, а единичный бит – как сигнал –1, будем вычислять метрику по формуле

$$m_i^j = \min_{k \in K_j} \left(m_{i-1}^k + \delta\left(s_{kj}, r_i\right)\right). \qquad (1)$$

Через K_j обозначено множество состояний, из которых существуют переходы в состояние j, s_{kj} – строка из t символов, записываемых автоматом в кодовое слово на переходе из k-го состояния в j-е, r_i – набор из t действительных чисел – сигналов на входе декодера, соответствующих i-му переходу, а

$$\delta\left(s_{kj}, r_i\right) = \sum_{p=1}^{t} \delta_p\left(s_{kj}^p, r_i^p\right), \text{ где}$$

$$\delta_t\left(s_{kj}^p, r_i^p\right) = \begin{cases} \left(-1 - r_i^p\right)^2, & \text{если } s_{kj}^p = '0'; \\ \left(r_i^p - 1\right)^2, & \text{если } s_{kj}^p = '1'. \end{cases}$$

Предположим, что при декодировании, как и при кодировании, автомат начинает работу в состоянии ноль, и будем считать, что $m_0^0 = 0$.

На обратном проходе реконструируется путь из состояний автомата, от последнего к первому. Сначала выбираем состояние с минимальной метрикой, полученной на последней итерации прямого прохода, а затем придерживаемся следующего принципа: если на шаге i было реконструировано состояние j, то на шаге $i-1$ выбираем состояние k, минимизирующее метрику m_i^j в формуле (1).

Построение эффективных кодирующих автоматов

В дальнейшем будем рассматривать только автоматы, генерирующие код скорости ½, однако все изложенные результаты могут быть легко обобщены и для других разновидностей автоматов, генерирующих помехоустойчивые коды.

Рассмотрим влияние структуры графа конечного автомата на качество генерируемого им кода. Как отмечено ранее, одна из важнейших характеристик качества кода – $d_{\min}$ – минимальное расстояние Хэмминга между кодовыми словами, которое можно определить как минимальное расстояние Хэмминга между словами, генерируемыми автоматом при проходе различными путями одинаковой длины из некоторого состояния a в некоторое состояние b. Если в ориентированном графе автомата существует два различных пути одинаковой длины m из состояния a в состояние b, будем называть их *двойным путем* длины m. Докажем следующий факт.

Лемма 1. Граф автомата, генерирующего код скорости ½ и имеющего n состояний, содержит

по крайней мере один двойной путь длины $\lfloor \log_2 n \rfloor + 1$.

Доказательство. Если автомат генерирует код скорости ½, то в каждом состоянии он считывает один бит исходного кода и, в зависимости от его значения, записывает два бита в кодовое слово и переходит в одно из двух других состояний. Таким образом, из каждой вершины графа этого автомата исходит две дуги. Предположим, что все вершины графа пронумерованы и обозначим через a_m список из 2^m номеров вершин (возможно, повторяющихся), в которые из вершины a ведут пути длины m. Если некоторый номер встречается в списке a_m более одного раза, то это означает, что граф содержит двойной путь длины m. Список $a_{\lfloor \log_2 n \rfloor + 1}$ содержит более n элементов и, так как всего в рассматриваемом графе n вершин, то элементы этого списка будут повторяться, из чего и следует утверждение леммы.

Чем выше значение $d_{\min}$, тем большее количество ошибок в кодовых словах заданной длины можно исправлять. Поскольку при каждом переходе рассматриваемые автоматы записывают в код на выходе два бита, то при наличии в графе автомата двойного пути длины m величина $d_{\min}$ генерируемого им кода не может превышать $2m$. Поэтому, если рассматривать только структуру графа, из двух автоматов лучшим можно считать тот, в графе которого длина самого короткого двойного пути будет большей. С учетом леммы 1 среди автоматов с n состояниями наилучшими будут те, графы которых не содержат двойных путей длиной $\lfloor \log_2 n \rfloor$ или более коротких. В лемме 2 сформулируем принцип построения графа с 2^m вершинами, удовлетворяющего этому свойству, т.е. не содержащего двойных путей длиной m или более коротких.

Лемма 2. Предположим, что граф имеет 2^m вершин, пронумерованных от нуля до $2^m - 1$ и из каждой вершины выходят две дуги, причем если $0 \le i \le 2^{m-1} - 1$, то дуги ведут в вершины $2i$ и $2i + 1$, а если $2^{m-1} \le i < 2^m$ – то в вершины $2i - 2^m$ и $2i - 2^m + 1$. Тогда граф не содержит двойных путей длиной меньше $m + 1$.

Доказательство. Возьмем произвольную вершину с номером i и рассмотрим представление этого номера в двоичном виде. Если $i \le 2^{m-1} - 1$, то из этой вершины за один шаг можно перейти к вершинам $2i$ и $2i + 1$, $2i - 2^m$ или $2i - 2^m + 1$. Переход к вершине $2i$ означает сдвиг двоичного представления номера i влево на один бит, а переход к вершине $2i + 1$ – сдвиг влево на один бит с последующей инверсией самого младшего бита. Если же $i \ge 2^{m-1}$, то за один шаг можно перейти к вершине $2i - 2^m$, что означает сдвиг номера вершины влево на один бит и инверсию самого старшего бита либо к вершине $2i - 2^m + 1$, что эквивалентно сдвигу влево на один бит с последующей инверсией самого старшего и самого младшего битов. Таким образом, при любом переходе самый младший бит в двоичном представлении номера вершины i сместится влево на одну позицию, но его значение при этом не изменится. Более того, его значение не изменится и на $m - 1$ в последующих переходах, пока он не станет самым старшим, т.е. $m + 1$. Теперь заметим, что самые младшие биты номеров вершин, в которые ведут дуги из некоторой вершины i, всегда различны. Поэтому будут различны и вторые биты номеров вершин, к которым существуют различные пути из вершины i длиной два, третьи биты номеров вершин, к которым существуют различные пути из вершины i длиной три и т.д. Таким образом, из произвольно выбранной вершины i невозможно перейти в одну и ту же другую вершину j различными путями, длина которых меньше $m + 1$. Лемма доказана.

Заметим, что граф автомата с четырьмя вершинами, изображенный на рис. 1, удовлетворяет сформулированному в лемме 2 принципу. Легко проверить, что он не содержит двойных путей длины меньше трех.

Лемма 3 устанавливает взаимосвязь между графами описанной в лемме 2 структуры и диаграммами состояний сверточных кодов.

Лемма 3. Вершины графа диаграммы состояний любого несистематического сверточного кода скорости ½, обладающего памятью m, могут быть пронумерованы так, чтобы выполнялось условие леммы 2.

Доказательство. Устройство, генерирующее указанный в условии леммы сверточный код, на каждой итерации считывает из входящего сообщения m двоичных символов, которые обозначим $x_t, \ldots, x_{t+m-1}$, и вычисляет значения двух линейных функций:

$$y_1 = x_{i_1} \oplus x_{i_2} \oplus \ldots \oplus x_{i_k},$$
$$y_2 = x_{j_1} \oplus x_{j_2} \oplus \ldots \oplus x_{j_l}.$$

Здесь $t \le i_q \le t + m - 1$, $t \le j_q \le t + m - 1$. Биты y_1 и y_2 записываются в код на выходе. На следующей итерации кодирующее устройство считывает символы $x_{t+1}, \ldots, x_{t+m}$, вновь вычисляет по ним эти же линейные функции (с соответствующим сдвигом индексов аргументов), записывает два бита в код на выходе и т.д. Если интерпретировать биты $x_t, \ldots, x_{t+m-1}$ как двоичное представление номера i состояния конечного автомата, то граф этого автомата будет иметь 2^m вершин, а следующее двоичное число $x_{t+1}, \ldots, x_{t+m}$, совпадающее с номером следующего состояния автомата, будет равно:

$2i$, если $x_t = 0$ и $x_{t+m} = 0$;
$2i + 1$, если $x_t = 0$ и $x_{t+m} = 1$;
$2i - 2^m$, если $x_t = 1$ и $x_{t+m} = 0$;
$2i - 2^m + 1$, если $x_t = 1$ и $x_{t+m} = 1$.

Полученные выражения полностью совпадают с указанными в условиях леммы 2. Лемма доказана.

Маркировкой графа кодирующего автомата назовем способ, которым дугам графа сопоставляются символы, записываемые автоматом в код на выходе при соответствующих переходах (способ сопоставления дугам графа битов входной последовательности несущественен, если считать, что значения каждого такого бита равновероятны). Хотя, как было показано, графы сверточных кодов имеют в определенном смысле оптимальную структуру, нет оснований полагать, что маркировка графов сверточных кодов, даже самых эффективных, также оптимальна. Как отмечено ранее, в сверточном коде каждый из символов кодового слова может быть вычислен как результат линейной булевой функции, зависящей от номера состояния, в котором пребывает автомат, и считанного бита исходного сообщения. Представлять выходные символы как функции, зависящие от номера состояния кодирующего автомата и исходного бита, можно и в общем случае, рассматривая произвольные автоматы, однако эти функции не обязательно будут линейными. Логично предположить, что отказ от линейности может привести к повышению помехоустойчивости кода. Чтобы проверить эту гипотезу, будем маркировать граф оптимальной структуры с помощью алгоритма 1, в котором недетерминированная составляющая сочетается с некоторыми правилами маркировки.

Алгоритм 1 маркировки графа генерирующего код автомата оптимальной структуры.

1. $b \leftarrow 0$.

2. Выбираем произвольно одну из четырех двухбитовых строк: 00, 01, 10 или 11 и маркируем ею любую из дуг, выходящих из вершины b. Другую дугу, выходящую из b, маркируем противоположной строкой (для 00 противоположной будет строка 11, для 10 – 01 и наоборот).

3. Пусть a – любая из двух вершин, таких, что существует дуга $a \rightarrow b$, а c – вершина, в которую ведет другая дуга из вершины a. Тогда любой из двух дуг, выходящих из c, приписываем одну из двух строк, не используемых при маркировке дуг, выходящих из b, а другую дугу, выходящую из c, маркируем противоположной строкой.

4. Для всех вершин, из двух входящих дуг в которые маркирована только одна (строкой xy), маркируем другую входящую дугу строкой, противоположной к xy.

5. Присваиваем переменной b номер любой такой вершины, что входящие в нее дуги маркированы, а выходящие – нет, и переходим на шаг 2. Если таких вершин в графе нет, маркировка завершена.

Корректность этого алгоритма, а также свойства конструируемой им маркировки устанавливает лемма 4.

Лемма 4. Для любого кодирующего автомата, граф которого удовлетворяет условиям леммы 2, алгоритм 1 корректен, а получаемая в результате его применения маркировка удовлетворяет следующим свойствам:

• дуги, выходящие из одной вершины, маркируются противоположными строками: 00 и 11 либо 01 и 10, т.е. строками, расстояние Хэмминга между которыми составляет два;

• дуги, входящие в одну вершину, маркируются противоположными строками;

• если существуют дуги $a \to b$ и $a \to c$, то переходы из вершин b и c маркируются разнотипными маркировками, т.е. если переходы из одной из этих вершин маркируются как 00 и 11, то переходы из другой вершины должны маркироваться как 01 и 10.

Доказательство. Чтобы доказать корректность работы алгоритма, следует установить два факта: во-первых, что после маркировки на шаге 4 не образуется вершин, одна из выходящих дуг из которых маркирована, а другая – нет; во-вторых, что перед шагом 5 не существует вершины, в которую входит одна маркированная дуга и одна немаркированная, либо из нее выходит одна маркированная дуга и одна немаркированная. Из построения графа, описанного в лемме 2, следует, что в вершины с номерами $2i$ и $2i + 1$ ведут дуги из вершин i и $i + 2^{m-1}$. Это означает, что если существуют дуги $a \to b$ и $a \to c$, то другие две дуги в вершины b и c также ведут из некоторой одной и той же вершины d (рис. 2,*а*). Заметим также, что если $i = 0$, то $i = 2i$ и вершины a и b совпадают (рис. 2,*б*), а если $i = 2^{m-1} - 1$, то $2i + 1 = i + 2^{m-1} = 2^m - 1$ и совпадают вершины c и d (рис. 2,*в*). Иначе говоря, на шагах 2 – 4 каждой итерации алгоритма выполняется маркировка всех дуг двух структур, подобных изображенным на одном из рис. 2*а – в*, и никаких других дуг, из чего и следуют два факта, позволяющие доказать корректность алгоритма, а также то, что каковы бы ни были вершины b и c, если существует некоторая вершина e, из которой дуги ведут в b и c, то дуги, исходящие из b и c, маркируются на одной итерации алгоритма. Из последнего факта, а также из способа маркировки дуг на шаге 3 следует сформулированное в условии леммы свойство 3. Свойство 1 из условия леммы следует непосредственно из способа маркировки дуг на шагах 2 и 3 алгоритма, а свойство 2 – из способа маркировки дуг на шаге 4. Лемма доказана.

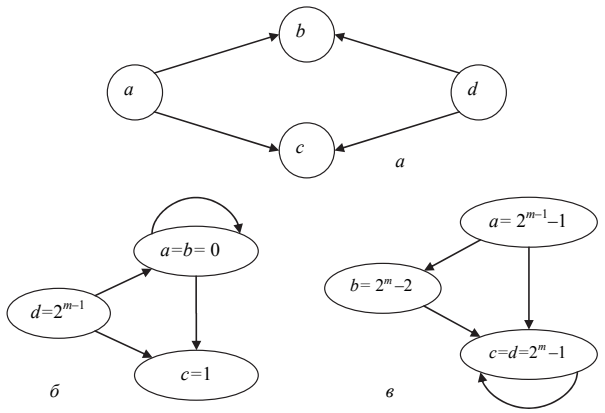

Рис. 2. Конструкции из четырех дуг в графе кодирующего автомата оптимальной структуры: $a - 0 < a < 2^{m-1}$, *б* – $a = 0$, *в* – $a = 2^{m-1} - 1$

Выполнение указанных в условии леммы 4 свойств гарантирует, что расстояние Хэмминга между ветками любого двойного пути будет не меньше пяти, а также, что среднее расстояние Хэмминга между ветками всех двойных путей в графе кодирующего автомата будет достаточно большим. Последнее особенно существенно, так как помехоустойчивость кода зависит не столько от величины $d_{\min}$, сколько от упомянутого среднего расстояния.

Применяя алгоритм 1 для генерирования маркировки графов кодирующих автоматов с 64 вершинами, т.е. с таким же количеством вершин, как и в графе сверточного кода памяти 6, например кода *NASA* (171,133), который наиболее эффективен из сверточных кодов памяти 6, получено несколько кодов с более высокими помехоустойчивыми свойствами. Один из автоматов,

Таблица 2. Маркировка графа кодирующего автомата

0 00 11	8 01 10	16 01 10	24 00 11	32 11 00	40 10 01	48 10 01	56 11 00
1 10 01	9 00 11	17 11 00	25 01 10	33 01 10	41 11 00	49 00 11	57 10 01
2 01 10	10 01 10	18 01 10	26 00 11	34 10 01	42 10 01	50 10 01	58 11 00
3 11 00	11 00 11	19 00 11	27 01 10	35 00 11	43 11 00	51 11 00	59 10 01
4 00 11	12 00 11	20 01 10	28 01 10	36 11 00	44 11 00	52 10 01	60 10 01
5 01 10	13 10 01	21 00 11	29 00 11	37 10 01	45 01 10	53 11 00	61 11 00
6 00 11	14 01 10	22 01 10	30 00 11	38 11 00	46 10 01	54 10 01	62 11 00
7 01 10	15 00 11	23 00 11	31 01 10	39 10 01	47 11 00	55 11 00	63 10 01

генерирующих такой код, приведен в табл. 2. Первое число в каждой ячейке таблицы – это номер вершины. Записанная после него строка из двух двоичных символов – это маркировка дуги, ведущей из данной вершины i в вершину с четным номером, т.е. $2i$ или $2i - 2^m$, а вторая строка представляет собой маркировку дуги, ведущей из вершины i в вершину с нечетным номером, т.е. $2i + 1$ или $2i - 2^m + 1$.

Сравнение эффективности данного кодирующего автомата с эффективностью кода *NASA* (171,133) приведено в табл. 3, где показаны результаты декодирования по методу Витерби при использовании *BPSK*-модуляции и мягкого декодирования [4]. Как видно, при высоких соотношениях величины полезного сигнала и шума в канале связи ($E_b\backslash N_0$) вероятность ошибки на бит (*BER*) для предложенного кода ниже 3–8 процентов, чем у кода *NASA*, а при боле низких значениях величины $E_b\backslash N_0$ помехоустойчивость рассматриваемых кодов почти не отличается.

Таблица 3. Сравнение эффективности сверточного кода и кода, сгенерированного конечным автоматом в соответствии с условием леммы 4

E_b/N_0, дб	*BER*, Сверточный код *NASA*(171,133)	*BER*, Код, сгенерированный конечным автоматом
–1	0,302	0,293
0	0,1551	0,1432
1	0,1309	0,1309
1,5	0,015	0,0145
2	0,005	0,0049

Отметим, что упомянутое преимущество достигается не увеличением сложности кодирования или декодирования по времени. Методы кодирования и декодирования для обоих кодов одинаковы за тем исключением, что в сверточном коде значения битов строки-результата вычисляются по формулам, а в предложенном коде извлекаются непосредственно из памяти, где хранится структура и маркировка кодирующего автомата. Второй способ даже быстрее, так как не предполагает выполнения сложений по модулю 2, хотя для его реализации требуется несколько больше памяти. Однако увеличение памяти, связанное с необходимостью хранения маркировок 64 вершин, для современных вычислительных устройств несущественно.

Заключение. Помехоустойчивая эффективность сверточных кодов может быть повышена, если для генерирования символов кодовых слов вместо линейных функций использовать нелинейные. При этом временная сложность кодирования и декодирования не возрастает. Дальнейшее повышение помехоустойчивости может быть достигнуто благодаря применению различных комбинаторных алгоритмов, определяющих позиции битов, значение которых было декодировано с ошибкой. Поиск и исследование таких алгоритмов продолжается.

1. *Shannon C.E.* A mathematical theory of communication // Bell Syst. Techn. J. – July 1948. – **27**. – P. 379–423; *Shannon C.E.* // Ibid. – Oct. 1948. – P. 623–656.
2. *Analysis* of low density codes and improved designs using irregular graphs / M. Luby, M. Mitzenmacher, A. Shokrollahi et al. // IEEE Trans. Inform. Theory. – 2001. – **47**. – P. 585–598.
3. *Berrou C.*, *Glavieux A.*, *Thitimajshima P.* Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes // Proc. IEEE Int. Conf. Comm. (ICC93), Geneve, Switzerland, May 1993. – P. 1064–1070.
4. *Morelos-Zaragoza R.H.* The art of error-correcting coding. – John Wiley & Sons, 2002. – 220 p.

Поступила 02.10.2014
E-mail: *zava@ukr.net*