УДК 519.725

**В. П. Семеренко**, канд. техн. наук
Винницкий национальный технический университет
(Украина, 21021, Винница, Хмельницкое шоссе, 95,
тел. (0432) 439002, E-mail: sm@vinnitsa.com)

# Декодирование кодов Рида—Соломона на основе графовой и автоматной моделей

Рассмотрены способ аналитического описания и многоуровневая графовая модель кодов Рида—Соломона (РС) на основе теории линейной последовательностной схемы. Предложены алгоритмы исправления независимых ошибок по графовой и автоматной моделях кодов РС. Выделен подкласс легкодекодируемых кодов РС. Проанализирована сложность алгоритмов при их последовательной и параллельной реализациях.

Розглянуто спосіб аналітичного опису і багаторівнева графова модель кодів Ріда—Соломона на основі теорії лінійної послідовнісної схеми. Запропоновано алгоритми виправлення незалежних помилок на основі графової та автоматної моделей кодів Ріда—Соломона. Виділено підклас легкодекодованих кодів РС. Проведено аналіз складності алгоритмів при їхній послідовній та паралельній реалізаціях.

*Ключевые слова: коды Рида—Соломона, автомат, линейная последовательностная схема, поля Галуа, независимые ошибки.*

Коды Рида—Соломона (РС) имеют хорошие корректирующие возможности и широко применяются в различных сферах — от систем передачи информации до защиты оптических дисков.

Наиболее известным алгоритмом декодирования кодов РС является алгоритм Берлекэмпа—Месси [1, 2], предложенный более 40 лет назад. Альтернативой жестким алгебраическим методам декодирования полиномиальной сложности, к которым относится алгоритм Берлекэмпа—Месси, является «мягкое» декодирование, позволяющее увеличить выигрыш от кодирования в результате использования дополнительной информации от демодулятора. Однако недавно было доказано, что один из основных методов мягкого декодирования, по максимуму правдоподобия, относится к числу *NP*-полных задач, что делает его непригодным для длинных кодов [3].

Предложенный в работе [4] алгоритм списочного декодирования кодов РС с полиномиальным временем создан на основе алгоритма Берлекэмпа—Уэлча, работающего за пределами классической корректирующей способ-

ности. Однако алгоритм эффективен только для низкоскоростных кодов. Позднее появилась его улучшенная версия для высокоскоростных кодов [5], и исследования в этом направлении продолжаются [6].

В последние годы много внимания уделяется алгебраическому мягкому декодированию, в котором объединены преимущества жесткого и мягкого декодирования [3, 7]. Следует заметить, что критерии оценки различных методов декодирования остаются неизменными. Выполняется только подсчет требуемых арифметических действий без учета сложности формализации отдельных этапов алгоритма, степени пригодности к программно-аппаратной реализации и даже сложности использования алгоритма в инженерной практике.

Стремительный прогресс в микро- и наноэлектронике выдвигает новые критерии и новые требования к вычислительным алгоритмам. В частности, важнейшей является задача их эффективного отображения на архитектуру высокопроизводительных вычислительных систем, использующих принципы параллельной обработки информации.

**Способы аналитического описания кодов Рида—Соломона.** Поскольку код РС принадлежит классу циклических кодов, он может иметь все виды представлений, присущих циклическим кодам. Выбор способа описания любого помехоустойчивого кода определяется в первую очередь используемым алгоритмом его декодирования.

Поскольку наиболее известными методами декодирования кодов РС являются алгебраические методы, чаще всего используется их полиномиальное представление, согласно которому код определяется через его порождающий многочлен. В этом случае под кодом РС, позволяющим исправлять $\tau_{\min}$ ошибок, понимают циклический $(n, k)$-код длины $n = q - 1$ над полем Галуа $GF(q)$, где $q = 2^m$ $(m = 2, 3, 4, \ldots)$, чей порождающий многочлен задается через корни последовательности степеней $\alpha^i$ примитивного элемента $\alpha$ поля $GF(q)$:

$$g(x) = (x - \alpha)(x - \alpha^2) \ldots (x - \alpha^d), \tag{1}$$

где $d$ — минимальное кодовое расстояние, $d = n - k = 2\tau_{\min}$.

Рассматриваемый метод декодирования кодов РС требует иного представления этих кодов, которое основано на использовании специального класса конечных автоматов — линейных последовательностных схем (ЛПС). Согласно [8] ЛПС над полем $GF(q)$ задается функцией состояний (переходов)

$$S(t+1) = A \times S(t) + B \times U(t) \tag{2}$$

и функцией выходов

$$Y(t) = C \times S(t) + D \times U(t), \tag{3}$$

где $t$ — дискретное время; $S$ — вектор состояния, $S = |s_i|_r$; $U$ и $Y$ — входной и выходной векторы, $U = |u_i|_l$, $Y = |y_i|_w$; $A = |a_{ij}|_{r\times r}$, $B = |b_{ij}|_{r\times l}$, $C = |c_{ij}|_{w\times r}$, $D = |d_{ij}|_{w\times l}$ — характеристические матрицы ЛПС.

Будем использовать ЛПС с одним входом и одним выходом ($l = 1$, $w = 1$), для которой функция выхода совпадает с функцией состояния: $Y(t) = S(t)$. Размерности матриц ЛПС и параметры $(n, k)$-кода РС связаны коэффициентом $r$, который для кода равен числу контрольных разрядов кодового вектора: $r = n - k$, $r = d$.

Аппаратной реализацией ЛПС над полем $GF(q)$ является устройство, содержащее элементы задержки, а также сумматоры и умножители, выполняющие соответственно операции сложения и умножения по правилам этого поля. Матрицы $A$, $B$, $C$ и $D$ определяют структуру взаимосвязей составных частей ЛПС. Над полем $GF(2)$ ЛПС представляет собой обычный регистр сдвига с линейными обратными связями. Часто ЛПС называют фильтром.

Если порождающий полином (1) преобразовать к виду

$$g(x) = \alpha^i_0 + \alpha^i_1 x + \alpha^i_2 x^2 + ... + \alpha^i_{d-1} x^{d-1} + x^d, \qquad (4)$$

то матрицы ЛПС над полем $GF(q)$ можно записать в виде

$$A = \begin{vmatrix} 0 & 0 & \cdots & 0 & \alpha^i_0 \\ \alpha^0 & 0 & \cdots & 0 & \alpha^i_1 \\ 0 & \alpha^0 & \cdots & 0 & \alpha^i_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \alpha^0 & \alpha^i_{d-1} \end{vmatrix}, \quad B = \begin{vmatrix} \alpha^0 \\ 0 \\ 0 \\ \cdots \\ 0 \end{vmatrix}, \quad i = 0, 1, 2, ..., n-1, \qquad (5)$$

где элементы последнего столбца матрицы $A$ представляют собой коэффициенты порождающего многочлена (4).

**Пример 1.** (7, 3)-код РС над полем $GF(8)$ с порождающим многочленом $g(x) = \alpha^3 + \alpha^1 x + \alpha^0 x^2 + \alpha^3 x^3 + x^4$ с помощью матриц ЛПС имеет следующий вид:

$$A = \begin{vmatrix} 0 & 0 & 0 & \alpha^3 \\ \alpha^0 & 0 & 0 & \alpha^1 \\ 0 & \alpha^0 & 0 & \alpha^0 \\ 0 & 0 & \alpha^0 & \alpha^3 \end{vmatrix}, \quad B = \begin{vmatrix} \alpha^0 \\ 0 \\ 0 \\ 0 \end{vmatrix}. \qquad (6)$$

С помощью ЛПС удобно выполнять вычисления элементов поля $GF(q)$, если эти элементы представлены в виде отдельных многочленов степени $m-1$. В этом случае понадобится отдельная ЛПС размерности

($m-1$), которая будет задана функциями, аналогичными (2), (3), но над полем $GF(2)$. Матрица $A$ этой ЛПС имеет такую же структуру, как матрица $A$ в (5), только в последнем столбце она содержит коэффициенты примитивного многочлена

$$h(x)=1+h_1x+h_2x^2+...+h_{m-2}x^{m-2}+h_{m-1}x^{m-1}+x^m,$$

использованного при построении поля $GF(2)$. Для краткости ЛПС над полем $GF(q)$, $q>2$, назовем символьной, а ЛПС над полем $GF(2)$ — битовой.

**Пример 2.** Многочлену $h(x)=1+x+x^3$, который может быть использован для построения поля $GF(8)$, соответствуют следующие матрицы битовой ЛПС:

$$A=\begin{vmatrix}0&0&1\\1&0&1\\0&1&0\end{vmatrix},\quad B=\begin{vmatrix}1\\0\\0\end{vmatrix}.$$

Выбрав в качестве начального состояния битовой ЛПС состояние, соответствующее примитивному элементу $\alpha^0$ поля $GF(q)$, можно получить все последующие элементы поля расширения: $\alpha^i=A\times\alpha^{i-1}$, $i=1, 2, ..., n-1$. Нулевому елементу поля $GF(q)$ соответствует вектор нулевого состояния S (0) битовой ЛПС.

**Графовые модели кодов Рида—Соломона.** Поскольку коды РС могут быть описаны с помощью ЛПС, являющейся конечным автоматом, в качестве графовой модели таких кодов можно выбрать диаграмму переходов (ДП) автомата. Для $r$-мерной символьной ЛПС над полем $GF(q)$ (далее будем говорить только о символьных ЛПС) ДП представляет собой ориентированный граф $G_{FA}(V_{FA}, E_{FA})$, в котором $q^r$ вершин из множества вершин $V_{FA}$ соответствуют $q^r$ внутренним состояниям автомата, а дуги из множества дуг $E_{FA}$ показывают направления переходов между внутренними состояниями.

В общем случае из вершины $v_j$ может выходить нулевая дуга $e^{\text{out}}_{\text{null}}$, соответствующая нулевому елементу поля $GF(q)$, и $n$ дуг $e^{\text{out}}_0$, ..., $e^{\text{out}}_{n-1}$ (будем называть их ненулевыми), которые соответствуют степеням $\alpha^0$, ..., $\alpha^{n-1}$ примитивного элемента $\alpha$ поля $GF(q)$. В вершину $v_j$ может входить нулевая дуга $e^{\text{in}}_{\text{null}}$, соответствующая нулевому елементу поля $GF(q)$, и $n$ ненулевых дуг $e^{\text{in}}_0$, ..., $e^{\text{in}}_{n-1}$, которые соответствуют степеням $\alpha^0$, ..., $\alpha^{n-1}$ примитивного элемента $\alpha$ поля $GF(q)$ ($v_j\in V_{FA}, e^{\text{in}}_{\text{null}}, e^{\text{in}}_i, e^{\text{out}}_{\text{null}}, e^{\text{out}}_i\in E_{FA}, i=0\div n-1$, $j=1\div q^r$).

Поскольку коды РС принадлежат классу циклических кодов, вершины графа $G_{FA}$ образуют многочисленные циклы. Упорядочим эти циклы на основе нулевых дуг.

Всегда имеется одна вершина $v_{\text{null}}$, для которой входящая $e^{\text{in}}_{\text{null}}$ и выходящая $e^{\text{out}}_{\text{null}}$ нулевые дуги объединяются и образуют петлю. Используя терминологию [9, 10], полагаем, что вершина $v_{\text{null}}$ с дугами $e^{\text{in}}_{\text{null}}$ и $e^{\text{out}}_{\text{null}}$ образует тривиальный нулевой цикл (ТНЦ). Остальные вершины графа $G_{FA}$ с помощью нулевых дуг образуют циклы длины не более $n$, которые можно расположить по уровням так.

На первом уровне располагаются $n$ основных нулевых циклов (ОНЦ) длины $n$, и $i$-й ОНЦ связан с ТНЦ с помощью пары противоположно направленных ненулевых дуг соответственно $e^{\text{in}}_{i}$ и $e^{\text{out}}_{i}$ .

Все остальные нулевые циклы (НЦ), которые будем называть периферийными (ПНЦ), распределяются так.

На втором уровне располагаются те ПНЦ, каждый из которых связан с одним или двумя ОНЦ парами противоположно направленных ненулевых дуг соответственно $e^{\text{in}}_{i,j}$ и $e^{\text{out}}_{i,j}$ $(j=1,2)$.

На $(\tau+1)$-м уровне каждый ПНЦ имеет ненулевые дуги $e^{\text{in}}_{i,j}$ и $e^{\text{out}}_{i,j}$ с НЦ $\tau$-го уровня и не имеет ненулевых дуг $e^{\text{in}}_{i,j}$ и $e^{\text{out}}_{i,j}$ с НЦ уровней $(\tau-1)$ и менее $(\tau=2,3,\ldots, j=1\div\tau+1)$. Такие дуги назовем вертикальными. При этом вертикальную дугу от НЦ $\tau$-го уровня к НЦ $(\tau+1)$-го уровня назовем прямой, а от НЦ $(\tau+1)$-го уровня к НЦ $\tau$-го уровня — обратной. Для $(n, k)$-кода РС над полем $GF(q)$ число НЦ на $\tau$-м уровне составляет

$$N_q^{(\tau)} = (q-1)^{\tau} \times \Psi_n^{\tau} / n, \qquad (7)$$

где $\Psi_n^{\tau}$ — число сочетаний из $\tau$ по $n$.

Если из вершины $v^{\beta\to\gamma}_{\text{beg}}$, принадлежащей $\beta$-му НЦ $\tau$-го уровня, выходит прямая вертикальная дуга к вершине $v^{\beta\to\gamma}_{\text{end}}$, принадлежащей $\gamma$-му НЦ $(\tau+1)$-го уровня, то вершину $v^{\beta\to\gamma}_{\text{beg}}$ называем начальной «вертикальной» связывающей вершиной (ВСВ) $\beta$-го НЦ относительно $\gamma$-го НЦ, вершину $v^{\beta\to\gamma}_{\text{end}}$ — конечной ВСВ $\gamma$-го НЦ относительно $\beta$-го НЦ. Аналогично обратная вертикальная дуга связывает начальную ВСВ $v^{\gamma\to\beta}_{\text{beg}}$, принадлежащую $\gamma$-му НЦ $(\tau+1)$-го уровня, с конечной ВСВ $v^{\gamma\to\beta}_{\text{end}}$, принадлежащей $\beta$-му НЦ $\tau$-го уровня. В некоторых случаях одна и та же вершина может одновременно быть начальной и конечной ВСВ относительно НЦ различных уровней. Число конечных ВСВ в одном НЦ $\tau$-го уровня $(\tau \le \tau_{\min})$ равно $\tau$, а в НЦ $(\tau_{\min}+1)$-го уровня (если он имеется) оно всегда меньше, чем $(\tau_{\min}+1)$.

Последовательность векторов внутренних состояний ЛПС, соответствующих вершинам одного цикла в графе $G_{FA}$, также образуют цикл. Поскольку совокупность циклов из векторов состояний имеет такую же

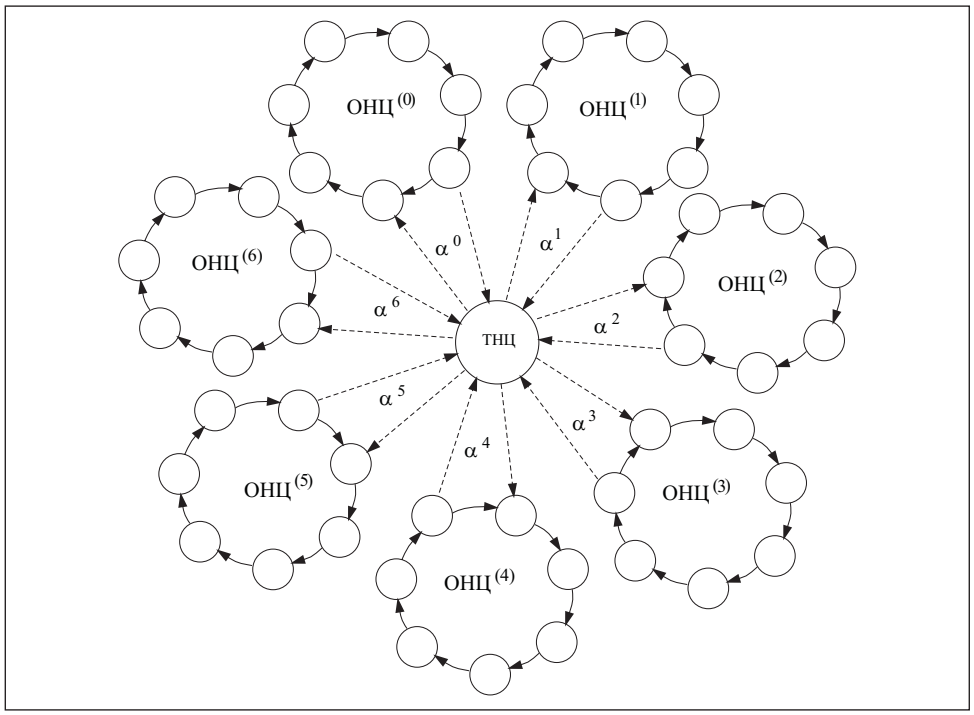


*Рис. 1.* ТНЦ и ОНЦ для (7,3)-кода РС, задаваемого порождающим многочленом $g(x) = \alpha^3 + \alpha^1 x + \alpha^0 x^2 + \alpha^3 x^3 + x^4$: сплошные линии — нулевые дуги; штриховые линии — ненулевые дуги

структуру, как и совокупность циклов из вершин, для характеристики циклов из векторов состояний будем использовать те же термины: ТНЦ, ОНЦ и ПНЦ.

Для связи между НЦ, образованных векторами состояний ЛПС, введем следующие обозначения: начальное «вертикальное» связывающее состояние (ВСС) $S_{\text{beg}}^{\beta\to\gamma}(t)$ β-го НЦ τ-го уровня относительно γ-го НЦ $(\tau+1)$-го уровня (соответствует вершине $v_{\text{beg}}^{\beta\to\gamma}$), и конечное ВСС $S_{\text{end}}^{\beta\to\gamma}(t)$ γ-го НЦ $(\tau+1)$-го уровня относительно β-го НЦ τ-го уровня (соответствует вершине $v_{\text{end}}^{\beta\to\gamma}$). Аналогично обозначим ВСС $S_{\text{beg}}^{\gamma\to\beta}(t)$ и $S_{\text{end}}^{\gamma\to\beta}(t)$, соответствующие вершинам $v_{\text{beg}}^{\gamma\to\beta}$ и $v_{\text{end}}^{\gamma\to\beta}$.

Далее в графовых моделях подразумеваем графовые НЦ, т.е. образованные вершинами графа, а при математических преобразованиях — автоматные НЦ, т.е. образованные векторами состояний ЛПС.

**Пример 3.** Граф $G_{FA}$ для (7,3)-кода РС, задаваемого матрицами (6), содержит ТНЦ, семь ОНЦ длины семь, 147 ПНЦ 2-го уровня длины семь и

431 ПНЦ 3-го уровня длины семь. На рис. 1 показаны ТНЦ и все ОНЦ для этого кода.

**Интерпретация независимых ошибок на основе графовой и автоматной моделей кода Рида—Соломона.** При передаче данных по каналу связи некоторые разряды исходного кодового вектора $Z$ случайно могут быть искажены, т.е. будет получен кодовый вектор $Z_{\text{err}}^{(\tau)}$ с ошибками кратности $\tau$. Взаимосвязь между кодовыми векторами $Z$ и $Z_{\text{err}}^{(\tau)}$ выражается через вектор ошибки $R_{\text{err}}^{(\tau)} = Z + Z_{\text{err}}^{(\tau)}$, $GF(q)$.

**Определение 1.** Независимой символьной ошибкой кратности $\tau$ называется совокупность $\tau$ искаженных символов кодового вектора $Z_{\text{err}}^{(\tau)}$, которые могут быть распределены по всей длине кодового вектора.

**Определение 2.** $\tau$-разреженным циклическим символьным пакетом ошибок длины $r$ называется совокупность $\tau$ искаженных символов кодового вектора $Z_{\text{err}}^{(\tau)}$, которые могут быть распределены только внутри циклической последовательности символов длины $r$.

В графе $G_{FA}$ последовательность из $n$ нулевых и ненулевых дуг $e_0, e_1, ..., e_{n-1} (e_i \in E_{FA}, i = 0 \div n-1)$, которая начинается и заканчивается в вершине $v_{\text{null}}$ (т.е. в ТНЦ), представляет собой кодовый путь $\eta$ и соответствует кодовому вектору $Z$, а множество всех возможных кодовых путей образует $(n, k)$-код РС. Кодовому вектору $Z_{\text{err}}^{(\tau)}$, содержащему $\tau$ независимых ошибок, будет соответствовать прямой кодовый путь ошибки $\eta_{\text{for}}$, который начинается в вершине $v_{\text{null}}$ и заканчивается в некоторой вершине ошибки $v_{\text{err}}$.

Кодовый путь, который начинается в вершине ошибки $v_{\text{err}}$ и заканчивается в вершине $v_{\text{null}}$, будем называть обратным кодовым путем ошибки $\eta_{\text{rev}}$. Нулевой цикл, который содержит вершину $v_{\text{err}}$, будем далее называть НЦ ошибки. Также, как и в графовых моделях над полем $GF(2)$ [9], НЦ ошибки находится на расстоянии $\tau$ от ТНЦ, т.е. расположен на $\tau$-м уровне графа $G_{FA}$. Между вершинами $v_{\text{null}}$ и $v_{\text{err}}$ может быть два прямых кодовых пути, $\eta_{\text{for1}}$ и $\eta_{\text{for2}}$ (которые соответствуют векторам $Z_{\text{err}}^{(\tau)}$ и $R_{\text{err}}^{(\tau)}$), и $\tau$ обратных кодовых путей $\eta_{\text{rev}\,1}, ..., \eta_{\text{rev}\,\tau}$.

Теперь дадим интерпретацию независимых ошибок с позиций теории ЛПС. Под воздействием входного вектора $Z$ ЛПС из нулевого начального состояния $S(0)$ перейдет в состояние $S(n)$, совпадающее с исходным состоянием, т.е. будет получен нулевой синдром: $S(n) = S(0)$. Под воздействием входного вектора $Z_{\text{err}}^{(\tau)}$ ЛПС из состояния $S(0)$ перейдет в некоторое ненулевое состояние $S_{\text{err}}^{(\tau)}(n)$, которое назовем синдромом ошибки кратности $\tau$. Нетрудно показать, что вершина $v_{\text{null}}$ графа $G_{FA}$ соответствует состоянию $S(0)$ ЛПС, а вершина $v_{\text{err}}$ — состоянию $S_{\text{err}}^{(\tau)}(n)$ ЛПС.

Следует заметить, что можно оценить корректирующие способности кода РС за границей Синглтона [2] на основе анализа структуры его графовой модели: код РС позволяет обнаруживать и исправлять некоторое количество независимых ошибок кратности $(\tau_{\min}+1)$, если его граф $G_{FA}$ имеет соответствующее количество ПНЦ на $(\tau_{\min}+1)$-м уровне.

**Алгоритмы поиска независимых ошибок по регулярным состояниям.** Декодирование кода РС начинается с $n$-кратного рекурсивного вычисления по формуле (2) при использовании кодового вектора в качестве входного вектора $U(t)$. При наличии $\tau$ ошибок в кодовом векторе $Z_{\text{err}}^{(\tau)}$ будет получен синдром ошибки $S_{\text{err}}^{(\tau)}(n)$. Возможны две основные стратегии исправления возникшей ошибки.

Первая основана на использования одного из традиционных методов поиска ошибок, например алгебраического алгоритма Берлекэмпа—Месси [1], или списочного декодирования Судана [4], и требует сложных математических преобразований. Вторая стратегия заключается в сравнении синдрома ошибки с хранимыми синдромами всех возможных независимых ошибок в пределах корректирующей способности кода, поэтому требуется большой объем памяти для хранения всех синдромов ошибок. Оптимальным решением было бы использование достоинств обеих стратегий при минимизации их недостатков.

Для нахождения символьных ошибок достаточно построить обратный кодовый путь ошибки $\eta_{\text{rev}}$ между вершинами $v_{\text{err}}$ и $v_{\text{null}}$ в терминах графовой модели или найти цепочку переходов от состояния $S_{\text{err}}^{(\tau)}(n)$ к состоянию $S(0)$ в терминах автоматной модели. При наличии графа $G_{FA}$ решение такой задачи не составит труда, однако огромные размеры этого графа не позволяют использовать такой подход на практике. На самом деле наличие полного графа не нужно, достаточно иметь заранее лишь специальные опорные вершины (начальные или конечные ВСВ), через которые всегда проходит прямой и обратный кодовые пути ошибки.

Далее будем использовать только конечные ВСВ. Каждая такая вершина внутри своего НЦ однозначно определяет все остальные вершины этого НЦ, поскольку связана с ними нулевыми дугами и максимальная длина пути не превышает $n$. Если сформировать множество конечных ВСВ $M_{\text{BCB}}=\{v_{\text{end}}^{\beta\to\gamma}\}$, которое будет включать по одному конечному ВСВ из всех НЦ, то такое множество однозначно определит все вершины графа $G_{FA}$ и при этом его мощность будет в $n$ раз меньше мощности множества вершин $V_{FA}$ этого графа.

Каждой конечной ВСВ из множества $M_{\text{BCB}}$ можно поставить в соответствие обратный кодовый путь от этой вершины к вершине $v_{\text{null}}$, который назовем базисным кодовым путем $\eta_{\text{base}}$. Множество базисных путей

$M_{\text{path}} = \{\eta_{\text{base}}\}$ также имеет мощность в $n$ раз меньше мощности множества всех обратных кодовых путей в указанном графе.

Для машинной реализации алгоритма поиска ошибок целесообразно перейти к автоматной модели, в которой базисному кодовому пути $\eta_{\text{base}}$ будет соответствовать базисный вектор ошибки $R^{(\tau)}_{\text{base}}$, под воздействием которого ЛПС перейдет в одно из конечных ВСС $S^{\beta\to\gamma}_{\text{end}}(t)$. Следовательно, множество конечных ВСС $M_{\text{BCC}} = \{S^{\beta\to\gamma}_{\text{end}}(t)\}$ и множество базисных векторов $M_{\text{err}} = \{R^{(\tau)}_{\text{base}}\}$ в автоматной модели будут эквивалентны соответственно множеству конечных ВСВ $M_{\text{BCB}} = \{v^{\beta\to\gamma}_{\text{end}}\}$ и множеству базисных путей ошибки $M_{\text{path}} = \{\eta_{\text{base}}\}$ в графовой модели.

Множества $M_{\text{BCC}}$ и $M_{\text{err}}$ можно сформировать один раз и затем многократно использовать в качестве своеобразной базы данных для нахождения на их основе вектора ошибки для каждой конкретной ошибки. Однако указанные множества вследствие большого количества НЦ кода РС также являются очень громоздкими. Поэтому рассмотрим возможность сокращения объемов хранимой информации, а в некоторых случаях и полного отказа от ее использования.

**Определение 3.** Конечное ВСС $S^{\beta\to\gamma}_{\text{end}}(t)$ называется $\tau$-регулярным, если оно представляет собой $(n-k)$-разрядный вектор, в котором среди нулевых символов имеется $\tau$ ненулевых символов $\alpha^i$, при этом один из них расположен в младшем (первом) разряде ($i = 0 \div n-1, \tau = 1 \div \tau_{\min}$).

Нулевой цикл, содержащий $\tau$-регулярное конечное ВСС, также будем называть регулярным.

В алгоритмах поиска ошибок на основе $\tau$-регулярных конечных ВСС использованы следующие их свойства.

1. В одном регулярном НЦ, расположенном на $\tau$-м уровне, среди $\tau$ конечных ВСС может быть только одно $\tau$-регулярное конечное ВСС.

2. На каждом уровне всегда имеется несколько регулярных НЦ, их число на $\tau$-м уровне можно определить с помощью известной формулы для нахождения числа сочетаний из $\tau - 1$ элементов по $n - k - 1$:

$$N^{(\tau)}_{\text{reg}} = (q-1)^{\tau} \times \Psi^{\tau-1}_{n-k-1}. \qquad (8)$$

Снова перейдем к графовой модели кода РС и назовем $\tau$-регулярными те конечные ВСВ, которые соответствуют $\tau$-регулярным конечным ВСС в автоматной модели кода РС. Эти вершины однозначно идентифицируют номер уровня графа $G_{FA}$ и могут быть своеобразными маркерами при построении обратного кодового пути от вершины $v_{\text{err}}$ ошибки к вершине $v_{\text{null}}$.

Рассмотрим алгоритм исправления всех независимых ошибок кратности $1, 2, ..., \tau_{min}$ в полученном кодовом векторе $Z_{err}^{(\tau)}$ для $(n,k)$-кода при попадании синдрома ошибки $S_{err}^{(\tau)}(n)$ в $\tau$-регулярный НЦ ошибки.

АЛГОРИТМ 1.

1. Для $i$ от 0 до $n-1$ при нулевом начальном состоянии $S(0)$ выполнить следующее:

$$S(i+1) = A \times S(i) + B \times Z_{err}^{(\tau)}[i], \qquad (9)$$

где $Z_{err}^{(\tau)}[i]$ — $i$-й разряд кодового вектора $Z_{err}^{(\tau)}$.

2. Получить синдром ошибки $S_{err}^{(\tau)}(n)$: $S_{err}^{(\tau)}(n) = S(n)$.

3. Задать позицию ошибки кратности $(\tau_{min}+1)$: pos $(\tau_{min}+1) = 0$, и исходное состояние ЛПС: $S(0) = S_{err}^{(\tau)}(n)$.

4. Для $j$ от $\tau_{min}$ до 1 выполнить следующее:

4.1. Для $i$ от 0 до $n-1$ выполнить следующее:

4.1.1. Если вектор $S(i)$ является $j$-регулярным, то определить позицию pos $(j)$ и значение val $(j)$ $j$-й ошибки:

$$\text{pos}(j) = \text{pos}(j+1) + i, \ \text{val}(j) = S(i)[0], \ GF(n),$$

где $S(i)[0]$ — нулевой разряд вектора $S(i)$. Перейти к п. 4.2.

4.1.2. Определить вектор нового состояния ЛПС $j$-го уровня:

$$S(i+1) = A \times S(i), \ GF(q).$$

4.2. Определить вектор первого состояния ЛПС $(j-1)$-го уровня:

$$S(0) = A \times S(i) + B \times S(i)[0], \ GF(q).$$

5. При нахождении $j$-регулярных векторов состояний $S(i)$ исправить кодовый вектор $Z_{err}^{(\tau)}$ согласно вычисленным параметрам ошибок и получить кодовый вектор $Z$.

6. Конец.

Вследствие циклической природы кодов РС существует взаимнооднозначное соответствие между регулярными состояниями и векторами ошибок.

**Определение 4.** Базисный вектор ошибки $(n,k)$-кода РС называется $\tau$-базисным вектором ошибки $R_{base}^{(\tau)}$, если он содержит $\tau$ ненулевых символов, которые распределены только внутри циклической последовательности символов длины $r$, и один из них находится в старшей, $n$-й позиции.

**Теорема 1.** Если синдром ошибки $S_{err}^{(\tau)}(n)$ является $\tau$-регулярным конечным ВСС $(S_{err}^{(\tau)}(n) = S_{end}^{\beta \to \gamma}(t))$, то ему соответствует $\tau$-разреженный циклический символьный пакет ошибок длины $r$ и $\tau$-базисный вектор ошибки $R_{base}^{(\tau)}$.

**Теорема 2.** Если синдром ошибки $S_{\text{err}}^{(\tau)}(n)$ принадлежит τ-регулярному НЦ, но не является τ-регулярным конечным ВСС ($S_{\text{err}}^{(\tau)}(n) \neq S_{\text{end}}^{\beta\to\gamma}(t)$), то справедливы следующие выражения:

$$S_{\text{end}}^{\beta\to\gamma}(t) = A^{\rho} \times S_{\text{err}}^{(\tau)}(t), \tag{10}$$

$$S_{\text{err}}^{(\tau)}(t) = A^{n-\rho} \times S_{\text{end}}^{\beta\to\gamma}(t). \tag{11}$$

Здесь $A^i$ — $i$-я степень характеристической матрицы $A$ ЛПС, и вектор ошибки $R_{\text{err}}^{(\tau)}$ соответствует сдвинутому вправо на ρ позиций или влево на $(n-\rho)$ позиций τ-базисного вектора ошибки $R_{\text{base}}^{(\tau)}$.

Справедливость теорем 1 и 2 можно легко проверить нахождением по формуле (2) состояния ЛПС при подаче на ее вход τ-базисного вектора ошибки $R_{\text{base}}^{(\tau)}$. В графовой модели ЛПС выражение (10) соответствует переходу внутри τ-регулярного НЦ ошибки от вершины ошибки $v_{\text{err}}$ по ρ нулевым дугам к τ-регулярной конечной ВСВ, а выражение (11) — переходу внутри τ-регулярного НЦ ошибки от этой же τ-регулярной конечной ВСВ по другим $(n-\rho)$ нулевым дугам к вершине ошибки $v_{\text{err}}$.

Теорема 1 позволяет по синдрому ошибки $S_{\text{err}}^{(\tau)}(n)$ определить ее параметры, а теорема 2 определяет способ поиска независимых ошибок в кодах РС на основе автоматной модели, т.е. нахождение ближайшего к синдрому ошибки $S_{\text{err}}^{(\tau)}(n)$ τ-регулярного конечного ВСС $S_{\text{end}}^{\alpha\to\beta}(t)$ с последующей коррекцией τ-базисного вектора ошибки $R_{\text{base}}^{(\tau)}$. Практической реализацией этого способа является следующий алгоритм.

АЛГОРИТМ 2.

1. Определить синдром ошибки $S_{\text{err}}^{(\tau)}(n)$ (как в алгоритме 1).

2. Положить исходные значения $S(0) = S_{\text{err}}^{(\tau)}(n)$, $R_{\text{err}}^{(\tau)}[i] = 0$ для всех $i = 0 \div n-1$.

3. Для $i$ от 0 до $n-1$ выполнить следующее:

3.1 Если вектор состояния $S(i)$ является $j$-регулярным ($j = 1 \div \tau_{\min}$), то присвоить $\rho = i$ и перейти к п. 5.

3.2 Определить вектор состояния $S(i+1)$:

$$S(i+1) = A \times S(i),\ GF(q). \tag{12}$$

4. Перейти к п.8.

5. Сформировать $j$-базисный вектор ошибки:

$$R_{\text{base}}^{(\tau)}[n-j-1] = S(i)[j],\ GF(n),\ j = 0 \div r-1,$$

где $S(i)[j]$ и $R_{\text{base}}^{(\tau)}[n-j-1]$ — $j$-й и $(n-j-1)$-й разряды векторов $S(i)$ и $R_{\text{base}}^{(\tau)}$.

6. Сформировать фактический вектор ошибки:

6.1 Для $i$ от 0 до $n-1$ выполнить следующее:

$$R_{\text{err}}^{(\tau)}[i]=R_{\text{base}}^{(\tau)}[i+\rho],\ GF(n).$$

7. Исправить кодовый вектор $Z_{\text{err}}^{(\tau)}$ и получить кодовый вектор $Z$.

8. Конец.

Таким образом, алгоритм 2, в отличие от алгоритма 1, позволяет определять параметры ошибок без построения обратного кодового пути ошибки $\eta_{\text{rev}}$.

**Пример 4.** Пусть для (7,3)-кода РС, задаваемого матрицами (6), получен кодовый вектор $Z_{err}^{(\tau)}=\alpha^5x^6+\alpha^1x^5+\alpha^6x^3+\alpha^4x^2+\alpha^2x+\alpha^0$, или в сокращенной форме, $Z_{\text{err}}^{(\tau)}=\{\alpha^5\alpha^1 0\alpha^6\alpha^4\alpha^2\alpha^0\}$. Для нахождения синдрома ошибки определим последовательность состояний ЛПС согласно (9). Операции над матрицами $A$ и $B$ легко заменить операциями сдвига и сложения над полем $GF$ (8) векторов состояний ЛПС, которые для удобства представим в транспонированном виде:

$$S(1)=\{\alpha^5 000\},\ S(2)=\{\alpha^1\ \alpha^5\ 00\},\ S(3)=\{0\alpha^1\ \alpha^5\ 0\},\ S(4)=\{\alpha^6 0\alpha^1\ \alpha^5\}.$$

Для нахождения последующих состояний необходимо использовать последний столбец матрицы $A$ из (6):

$$S(5)=\{\alpha^4\alpha^6 0\alpha^1\}+\{\alpha^3\alpha^1\ \alpha^0\ \alpha^3\}\times\alpha^5=\{\alpha^2 0\alpha^5\ 0\},\ GF(8),$$

$$S(6)=\{\alpha^2\alpha^2 0\alpha^5\},$$

$$S(7)=\{\alpha^0\alpha^2\alpha^2 0\}+\{\alpha^3\alpha^1\ \alpha^0\ \alpha^3\}\times\alpha^5=\{\alpha^3\alpha^0\ \alpha^3\ \alpha^1\},\ GF(8).$$

В итоге получим синдром ошибки $S_{\text{err}}^{(\tau)}(n)=\{\alpha^3\alpha^0\ \alpha^3\ \alpha^1\}$, свидетельствующий о наличии ошибок в кодовом векторе. Поскольку данный код РС может исправить не более двух символьных ошибок и синдром ошибки не является 1- или 2-регулярным конечным ВСС, нельзя сразу определить параметры возникшей ошибки. В терминах автоматной модели далее необходимо построить цепочку переходов от состояния $S_{\text{err}}^{(\tau)}(n)$ до ближайшего $\tau$-регулярного конечного ВСС при нулевом входном воздействии. Поэтому, приняв в качестве исходного состояния $S(0)$ синдром ошибки, согласно (12) вычисляем:

$$S(1)=\{0\alpha^3\alpha^0\alpha^3\}+\{\alpha^3\alpha^1\ \alpha^0\ \alpha^3\}\times\alpha^1=\{\alpha^4\alpha^5\ \alpha^3\ \alpha^6\},$$

$$S(2)=\{0\alpha^4\alpha^5\alpha^3\}+\{\alpha^3\alpha^1\ \alpha^0\ \alpha^3\}\times\alpha^6=\{\alpha^2\alpha^5\ \alpha^1\ \alpha^5\},$$

$$S(3)=\{0\alpha^2\alpha^5\alpha^1\}+\{\alpha^3\alpha^1\ \alpha^0\ \alpha^3\}\times\alpha^5=\{\alpha^1\alpha^0\ 00\}.$$

Для получения 2-регулярного состояния $S(3)$ трижды использована формула (12), т.е. $\rho=3$. На основе состояния $S(3)$ формируем 2-базисный вектор ошибок $R_{\text{base}}^{(2)}=\{00000\alpha^0\alpha^1\}$. Далее циклически сдвигаем его на $\rho$

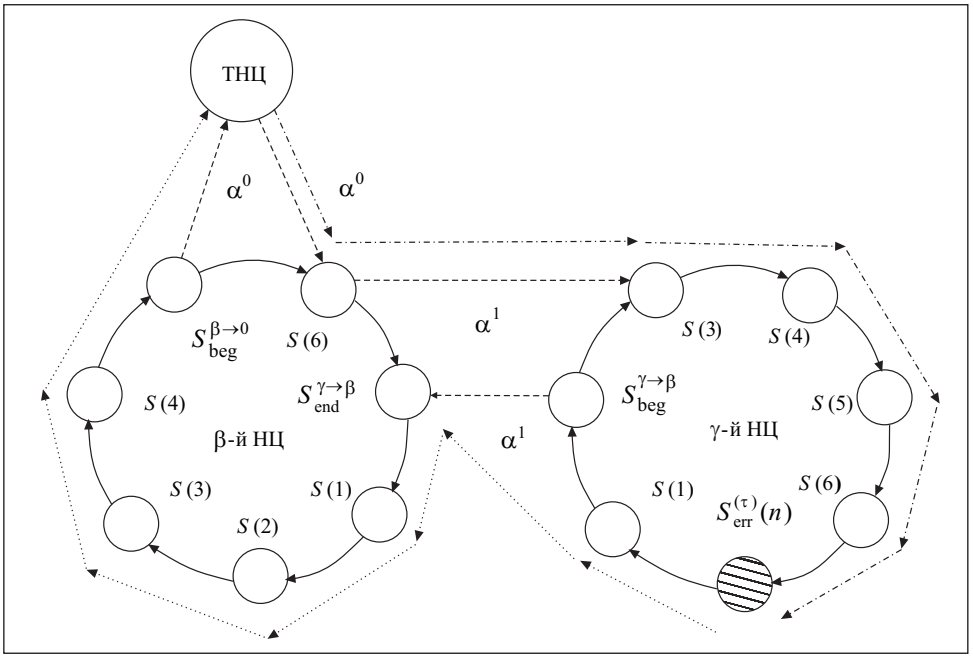


*Рис. 2.* Кодовые пути, соответствующие вектору ошибок $R^{(2)}_{\text{err}}(x)=\{0\,\alpha^0\,\alpha^1\,0\,0\,0\,0\}$: сплошные линии — нулевые дуги; штриховые — ненулевые дуги; штрих-пунктирные — прямой кодовый путь; пунктирные — обратный кодовый путь

позиций вправо и получаем искомый вектор ошибки: $R^{(2)}_{\text{err}}=\{0\ \alpha^0\alpha^1 0000\}$. На рис. 2. показаны кодовые пути, соответствующие этому вектору ошибок.

Таким образом, если синдром ошибки попадает в регулярный НЦ ошибки, то для поиска параметров ошибки понадобится не более $n$ операций (12) вычисления очередного состояния ЛПС. Однако не для всех кодов РС характерна ситуация, когда синдром ошибки будет всегда попадать в регулярный НЦ. Из (7) и (8) вытекает следующая теорема.

**Теорема 3.** В пределах корректирующей способности циклического $(n,k)$-кода все НЦ для τ-го уровня графа $G_{FA}$ будут регулярными, если выполняется условие

$$\Psi^{\tau-1}_{n-k-1}\geq\left\lfloor\frac{\Psi^{\tau}_{n}}{n}\right\rfloor, \tag{13}$$

**Определение 5.** $(n,k)$-коды РС, для которых выполняется условие (13), называются легкодекодируемыми для ошибок кратности τ и менее.

**Определение 6.** $(n,k)$-коды РС, для которых выполняется условие (13) для всех независимых ошибок в пределах корректирующей способности кода $(\tau=1,\ldots,\tau_{\min})$, называются легкодекодируемыми.

Рассмотренный в примере 4 (7,3)-код РС является легкодекодируемым.

**Общая стратегия поиска независимых ошибок.** Поскольку в основе предложенных алгоритмов лежит операция рекурсивного вычисления очередного состояния ЛПС по формуле (2), при оценке верхней границы сложности алгоритмов будем учитывать число таких рекурсий.

В течение первых $n$ тактов происходит процесс декодирования полученного кодового вектора, который заканчивается вычислением синдрома, т.е. последнего состояния ЛПС. При получении ненулевого синдрома, т.е. синдрома ошибки $S_{\text{err}}^{(\tau)}(n)$, начинается процесс определения параметров (значений и позиций) возникших ошибок. Трудоемкость поиска ошибок зависит от типа НЦ ошибки, содержащего синдром ошибки. Поэтому вначале предпринимается попытка с помощью алгоритмов 1 или 2 (последний более предпочтителен) найти в НЦ ошибки $\tau$-регулярное конечное ВСС, что будет свидетельствовать о регулярности и самого НЦ ошибки. В этом случае алгоритм 1, имеющий сложность $O((n+1)\times\tau)$, или алгоритм 2, имеющий линейную сложность $O(n)$, быстро завершат работу по локализации всех ошибок в пределах корректирующей способности кода.

Однако легкодекодируемыми являются лишь низко- или среднескоростные коды РС, т.е. коды, для которых выполняется соотношение $k/n \le 1/2$. С увеличением скорости кода уменьшается соотношение числа контрольных разрядов к общей длине кода и соответственно доля регулярных конечных ВСС среди всех конечных ВСС, особенно при возрастании кратности ошибки. При использовании высокоскоростных кодов синдром ошибки также может попасть в регулярный НЦ ошибки (например, при возникновении $\tau$-разреженного циклического символьного пакета ошибок длины не более $r$), что позволит выполнить локализацию ошибки с линейной сложностью. Поэтому необходимо всегда начинать поиск ошибок с алгоритма 1 или 2.

Только в случае отсутствия результатов работы этих алгоритмов включается итеративный алгоритм поиска (назовем его алгоритмом 3, который здесь не приводится) регулярного НЦ сначала на расстоянии одной ненулевой дуги от НЦ ошибки (первая итерация). Если среди соседних к НЦ ошибки не окажется регулярных НЦ, можно продолжать их искать на больших расстояниях (последующие итерации). Поиск реализуется поочередным вычислением состояний ЛПС и в терминах графовой модели представляет собой процесс построения пути от НЦ ошибки через конечные ВСВ к ближайшему регулярному НЦ. Каждая итерация работы этого алгоритма определяет параметры одной ошибки. После нахождении регулярного НЦ снова включается алгоритм 1 или 2, и работа быстро завершается нахождением параметров оставшихся ошибок.

Таким образом, использование $\tau$-регулярных состояний позволяет заменить общую задачу поиска полного пути к ТНЦ задачей нахождения более короткого пути к ближайшему регулярному НЦ. Такой регулярный

НЦ будет обязательно найден, проблема заключается лишь во временных затратах. Выполнение только первой итерации алгоритма 3 имеет сложность $O(n^3 \times \tau)$. Поэтому вместо выполнения последующих итераций целесообразно использовать заранее вычисленные подмножества нерегулярных конечных ВСС $M_{\text{BCC}}^{\text{non}}$ и соответствующие им подмножества базисных векторов ошибки $M_{\text{err}}^{\text{non}}$ ($M_{\text{BCC}}^{\text{non}} \subseteq M_{\text{BCC}}$, $M_{\text{err}}^{\text{non}} \subseteq M_{\text{err}}$).

Предварительный анализ структуры графа $G_{FA}$ с помощью алгоритма 3 позволяет определить НЦ, находящиеся «далеко» от ближайших регулярных НЦ, и только для них сохранять конечные ВСС и соответствующие базисные векторы ошибок. Такой анализ выполняется только один раз для выбранного кода РС.

Наилучшим способом сокращения временных затрат в задачах поиска ошибок является переход к параллельной обработке данных. Параллелизм можно использовать как на макроуровне (например, одновременным выполнением алгоритма 3 и поиском векторов в множествах $M_{\text{BCC}}^{\text{non}}$ и $M_{\text{err}}^{\text{non}}$), так и на микроуровне (в пределах одного алгоритма).

В теории ЛПС [8] доказано, что $n$-кратное вычисление по формуле (2) может быть заменено одновременным вычислением векторов состояний $A \times S(t)$, $A^2 \times S(t)$, ..., $A^{n-1} \times S(t)$. Поэтому процесс рекурсивного вычисления очередных состояний ЛПС можно легко распараллелить на независимые потоки. При программной $n$-потоковой реализации указанных алгоритмов или аппаратной реализации с помощью $n$ процессорных элементов, выполняющих операции сравнения или сложения по модулю $q$ между $(n-k)$-разрядными векторами, сложность реализации каждого алгоритмом уменьшится в $n$ раз.

Классические алгебраические методы декодирования кодов РС на основе решения ключевого уравнения [2] при возникновении $\tau$ ошибок требуют $2\tau$ итераций и $6\tau^2$ операций умножения $(\tau \times \tau)$-разрядных матриц. При этом они содержат разнотипные и трудноформализуемые процедуры, часть которых выполняется методом проб и ошибок (процедура Ченя [2 ], предложенный Месси способ построения требуемого регистра сдвига с линейными обратными связями минимальной длины). Метод списочного декодирования кодов РС, предложенный в [5], имеет полиномиальную сложность и использует трудноформализуемые операции, связанные с построением интерполяционного многочлена и поиском его корней.

**Выводы.** Предложенная многоуровневая графовая модель кодов РС на основе математического аппарата ЛПС позволяет интерпретировать задачу исправления ошибок как задачу поиска путей по графу переходов автомата. Алгоритмы исправления ошибок на основе автоматной модели кодов РС основаны на одной операции — рекурсивном вычислении очередного состояния ЛПС, которая может быть заменена элементарными операциями сдвига и сложения $(n-k)$-разрядных векторов в поле $GF(q)$.

Поэтому эти алгоритмы пригодны для матрично-конвейерной (систолической) обработки в параллельных вычислителях либо для многопотоковой обработки при программной реализации.

Предложенный метод выделения регулярных состояний ЛПС дает возможность выделить подкласс легкодекодируемых кодов РС и простой метод исправления ошибок с линейной сложностью для низко- и среднескоростных кодов РС. В рамках общей стратегии поиска ошибок для высокоскоростных кодов РС можно исправлять с линейной сложностью также и достаточно распространенный тип ошибок в реальных каналах связи — циклические символьные пакеты ошибок длины не более $(n-k)$.

The method of the analytical description and multilevel graphical model of Reed-Solomon (RS) codes based on the theory of linear finite-state machines is considered. The algorithms of the random error correction according to suggested graphical and automatical models of RS codes are offered. The subclass of easily-correctable RS codes is selected. The analysis of complexity of algorithms is carried out at their consecutive and parallel realisations.

1. *Скляр Б.* Цифровая связь. Теоретические основы и практическое применение. Изд. 2-е, испр. — М. : Изд. дом «Вильямс», 2004. — 1104 с.
2. *Блейхут Р.* Теория и практика кодов, исправляющих ошибки. — М. : Мир, 1986. — 576 с.
3. *Jiang Gross J., Narayanan K. R.* Algebraic soft-decision decoding of Reed-Solomon Codes using Bit-level Soft Information//IEEE Trans. Inform. Theory.— 2008. — Vol. 54, №9.— P. 3907— 3928.
4. *Sudan M.* Decoding of Reed-Solomon beyond the error-correction bound// J. Complexity. — 1997. — **13,** N 1. — P. 180—193.
5. *Guruswami V., Sudan M.* Improved decoding of Reed-Solomon and algebraic-geometry codes// IEEE Trans. Inform. Theory. — 1999. — **45**, № 6. — P. 1757—1767.
6. *Wu Y.* New list decoding algorithms for Reed-Solomon and BCH Codes// Ibid. — 2008. — **54**, № 8. — P. 3611—3630.
7. *Gross W. J., Kschischang F. R., Koetter R., Gulak P. G.* Applications of algebraic decoding of Reed-Solomon and algebraic-geometry codes// Ibid. — 2003. — **49**, № 7. — P. 1224— 1234.
8. *Гилл А.* Линейные последовательностные машины. — М.: Наука, 1974. — 288 с.
9. *Семеренко В. П.* Параллельное декодирование циклических кодов Боуза—Чоудхури-Хоквингема // Электрон. моделирование. — 1998. — **20**, № 1. — С. 82—87.
10. *Semerenko V. P.* Burst-Error Correction for Cyclic Codes. //Proc. of International IEEE Conference EUROCON 2009. — S.Petersburg, Russia. — P.1646—1651.

Поступила 25.06.10;
после доработки 23.09.10

*СЕМЕРЕНКО Василий Петрович, канд. техн. наук, доцент кафедры вычислительной техники Винницкого национального технического университета. В 1976 г. окончил Винницкий политехнический ин-т. Область научных исследований — параллельная обработка данных, помехоустойчивые коды, защита информации, тестовая диагностика цифровых схем, дедуктивный вывод на знаниях.*