УДК 681.391

МЕТОДЫ И АППАРАТУРА ДЕКОДИРОВАНИЯ БЛОКОВЫХ КВАЗИЦИКЛИЧЕСКИХ НИЗКОПЛОТНОСТНЫХ КОДОВ

Кравченко А.Н., *к.т.н., проектировщик систем Ubiso GmbH , Germany, e-mail: alexander.kravtchenko@ubiso.com*

Ключевые слова: низкоплотностные коды, декодирование, BQC-LDPC коды, архитектура декодера.

Преимуществами структурированных квазициклических низкоплотностных (QC-LDPC) кодов по сравнению со случайными LDPC кодами являются: линейное время кодирования, простая структура аппаратуры декодера, основанная на циклических сдвигающих регистрах, низкие требования к памяти для хранения проверочных матриц и хорошие показатели коррекции ошибок. В настоящее время QC-LDPC коды широко используются в стандартах цифрового вещания - DVB-S2/T2/C2, в стандартах систем связи и коммуникаций - WiMax(IEEE 802.16e), WLAN(IEEE 802.11n), WPAN (IEEE 802.15.3c), ITU-G.9960, MBWA (IEEE 802.20), WRAN (IEEE802.22). Для беспроводных систем связи QC-LDPC коды привлекают значительное внимание, поскольку облегчают аппаратурную реализацию проверочных матриц и тем самым позволяют легко регулировать длину кода и скорость кодирования.

BQC-LDPC коды: основные определения

Во всех выше перечисленных стандартах используются блоковые квазициклические низкоплотностные (BQC-LDPC) коды. BQC-LDPC код определяется базовой матрицей $\mathbf{H}_\mathbf{b}$ размерности $\mathbf{M}_\mathbf{b} \times \mathbf{N}_\mathbf{b}$, где $\mathbf{N}_\mathbf{b}$ – число блок столбцов и $\mathbf{M}_\mathbf{b}$ – число блок строк с $\mathbf{M}_\mathbf{b} = M / z$ и $\mathbf{N}_\mathbf{b} = N / z$, где M определяет число проверочных уравнений, N равно длине кодового слова. Параметр z определяет размерность квадратной перестановочной матрицы $\mathbf{A}_{ij}$, имеющий вес 1, как для строк так и для столбцов. Перестановочная матрица получается вследствие сдвига единичной матрицы, в которой каждая последующая строка является циклическим сдвигом вправо на одно место предыдущей строки. Индекс сдвига определяет позицию «1» в первой строке матрицы. Вся базовая матрица состоит из перестановочных матриц с различными индексами сдвига и нулевых матриц. Каждая нулевая матрица также есть квадратная матрица размерности $z \times z$. Табл. 1 для примера иллюстрирует проверочную матрицу LDPC кода с кодовой скоростью R = 7/8 для стандарта WPAN (IEEE 802.15.3c), где индекс -1 определяет нулевую перестановочную матрицу (матрица состоит из одних нулей), индекс 0 определяет единичную матрицу, все остальные матрицы, входящие в базовую матрицу, являются перестановочными матрицами.

Базовую матрицу можно также представить как матрицу, содержащую $\mathbf{M}_\mathbf{b}$ групп проверочных узлов (CNG) и $\mathbf{N}_\mathbf{b}$ групп символьных узлов (SNG). Каждая перестановочная матрица занимает определенное положение в базовой матрице. Например, перестановичная матрица $\mathbf{A}_{3,31}$ находится в третьей группе проверочных узлов и занимает позицию 3 в 31 группе символьных узлов с соответствующим индексом сдвига, равном 10. Фиксированное положение перестановочных матриц и их индексов существенно упрощает аппаратуру управления декодера, а циклическая структура кода значительно снижает сложность декодирования BQC-LDPC кода.

Алгоритмы декодирования BQC-LDPC кодов и архитектуры декодеров

LDPC коды могут декодироваться различными методами, а именно: мажоритарным методом (majority - logic), методом с перевертыванием бита (bit - flipping), взвешенным методом с перевертыванием бита, а также итеративными вероятностными методами, обладающими наиболее высокой эффективностью декодирования (ЭД) по сравнению с другими методами. Широко в практике декодирования LDPC кодов используются двухфазный алгоритм декодирования (TPMPA) [1], его модификации [2-4] и метод послойного декодирования. В литературе этот метод имеет различные названия: layered decoding (LDA), staggered decoding (SDA), turbo-decoding message-passing decoding алгоритм (TDMPA). Основными преимуществами LDA алгоритма декодирования LDPC кодов по сравнению с TPMPA алгоритмом являются следующие:

– декодер, спроектированный при использовании этого алгоритма, требует меньшего объема памяти чем стандартный декодер, реализующий стандартный двухфазный метод декодирования;

Таблица 1. Проверочная матрица LDPC кода с кодовой скоростью R = 7/8

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0	18	6	5	7	18	16	0	10	2	3	6	10	16	9	0	20	7	9	5	4	12	4	4	4	10	19	5	10	-1	-1	-1
5	0	18	6	0	7	18	16	6	10	2	3	0	10	16	9	5	20	7	9	4	4	12	4	5	4	10	19	19	10	-1	-1
6	5	0	18	16	0	7	18	3	6	10	2	9	0	10	16	9	5	20	7	4	4	4	12	19	5	4	10	17	19	10	-1
18	6	5	0	18	16	0	7	2	3	6	10	16	9	0	10	7	9	5	20	12	4	4	4	10	19	5	4	7	17	19	10

– эффективность декодирования и скорость сходимости алгоритма существенно выше, чем у TPMPA (алгоритм выполняет меньше итераций для достижения равной эффективности декодирования).

Основы метода были заложены в [5] и далее получили развитие в работах [6-7]. Модифицируем данные методы для декодирования BQC-LDPC кодов и исследуем эффективность декодирования и скорость сходимости модифицированных алгоритмов.

TPMDA алгоритм декодирования BQC-LDPC и архитектура декодера

Алгоритм декодирования

Для декодирования BQC-LDPC кода используем TPMDA алгоритм, модифицированный следующим образом.

1. Инициализировать матрицу сообщений $\mathbf{E}$ канальной информацией $\boldsymbol{\lambda}$. Алгоритм инициализации будет пояснен ниже.

2. Для k - ой итерации

$for\ m = 1, 2, \ldots, \mathbf{N}_b$ do {Первая фаза}

$\vec{S} = \vec{\lambda}_m$

$for\ n = 1, 2, \ldots, dv$ do {параметр dv определяет число ненулевых перестановочных матриц в блоке столбцов, $n = p_q$, $q = 1, 2, \ldots, \mathbf{M}_b$, p_q определяет положение перестановочной матрицы в текущем блоке}

$\vec{R}_n = \vec{E}_{m,n}^{(k-1)}$ {$\rightarrow$ обозначает вектор сообщений размерности z }

$\vec{S} = \vec{S} + \vec{R}_n$

$end\ for$

$for\ j = 1, 2, \ldots, z$ do

$if\ (S_j > 0)\quad c_{m,j} = 1\ else\ c_{m,j} = 0$ {жесткое решение для двоичного вектора }

$for\ n = 1, 2, \ldots, dv$ do

$Q_{n,j} = S_j - R_{n,j}$ {вычисление внутренних (intrinsic) сообщений}

$end\ for$

$end\ for$

$for\ n = 1, 2, \ldots, dv$ do

$\vec{E}_{m,n}^{(k)} = \vec{Q}_n$ {обновление сообщений}

$end\ for$

$end\ for$ {конец первой фазы }

$for\ l = 1, 2, \ldots, \mathbf{M}_b$ do {Вторая фаза}

$for\ n = 1, 2, \ldots, dc$ do {параметр dc определяет число ненулевых перестановочных матриц в блоке строк, $n = p_q$, $q = 1, 2, \ldots, \mathbf{N}_b$, p_q определяет положение перестановочной матрицы в текущем блоке}

$\vec{Q}_n = [\vec{E}_{l,n}^{(k-1)}]^{S(n)}$ {S(n) есть индекс сдвига для n -ой перестановочной матрицы в блоке l, $\rightarrow$ обозначает вектор сообщений размерности z }

$end\ for$

$for\ j = 1, 2, \ldots, z$ do

$for\ n = 1, 2, \ldots, dc$ do

$R_{n,j} = f(Q_{n,j})$ {функция $f()$ определяет один из возможных методов вычисления (extrinsic) внешних сообщений}

$end\ for$

$end\ for$

$for\ n = 1, 2, \ldots, dc$ do

$\vec{E}_{l,n}^{(k)} = [\vec{R}_n]^{S(n)}$ {обновление сообщений}

$end\ for$

$end\ for$ do {конец второй фазы }

3. Проверить кодовое слово на условие – $\hat{\mathbf{c}}\mathbf{H}^{\mathbf{T}} = 0$. Если условие удовлетворяется или число итераций рав-но максимальному лимиту, то декодирование пре-кра-щается, иначе $k = k + 1$ и переход к шагу 2.

Архитектура декодера

Архитектура декодера (рис. 1) включает следующие основные компоненты:

1. Постоянную память для хранения параметра dc, определяющего число ненулевых перестановочных матриц в слое (блок строке), номера слоя в базовой матрице, индексов сдвига перестановочных матриц, а также их местоположение в слое. Для примера определим размерность памяти для хранения параметров базовой матрицы LDPC кода, имеющего кодовую скорость, равную R = 7/8 (табл. 1). Для кодирования параметра dc требуется 6 бит, для кодирования слоя 2 бита, для кодирования индекса сдвига 5 бит и для кодирования местоположения перестановочной матрицы в слое – 5 бит. Все выше перечисленные параметры кодируются одной строкой в памяти, содержащей 18 бит. Параметры первого слоя содержатся в 29 строках. В общей сложности для хранения базовой матрицы требуется 29+30+31+32=122 строки. Аналогичным образом рассчитываются потребности в памяти для других базовых матриц кода. Постоянная память состоит из трех сегментов (три кодовых скорости для рассматриваемого стандарта), каждый из которых содержит сведения о базовой матрице. Каждый сегмент имеет определенный стартовый адрес.

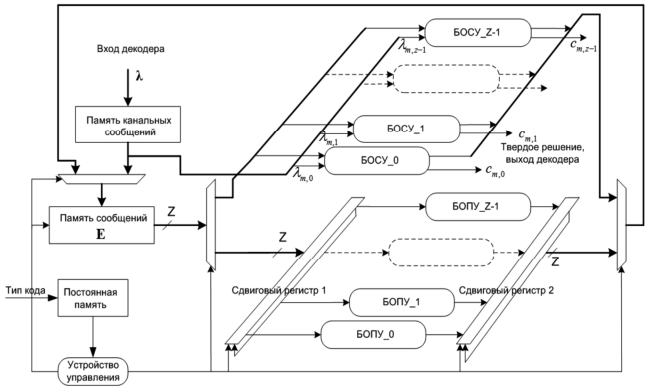

Рис. 1. Архитектура декодера с двухфазным алгоритмом декодирования

2. Оперативную память для хранения, считывания и записи сообщений во время декодирования (E память). Размерность памяти равна числу ненулевых перестановочных матриц, содержащихся в базовой матрице, умноженному на ширину LLR_WORD слова. Например, для кода с кодовой скоростью R = 7/8 (табл. 1) требуемый объем памяти равен 122*LLR_WORD слов, где LLR_WORD=Z*LLR. Каждое LLR значение (размерность одного сообщения) представлено 6 битами. Во время процесса инициализации памяти, канальные сообщения записываются способом, поясненным на рис. 2. Канальные сообщения записываются символьными (SNG) группами, в зависимости от параметра dv, который определяет степень символьной группы. E память связанна с блоками обработки проверочных (БОПУ) и символьных узлов (БОСУ) через соответствующий мультиплексер/демультиплексер.

3. Память канальных сообщений. Размерность памяти равна $\mathbf{N_b}$ * LLR_WORD слов.

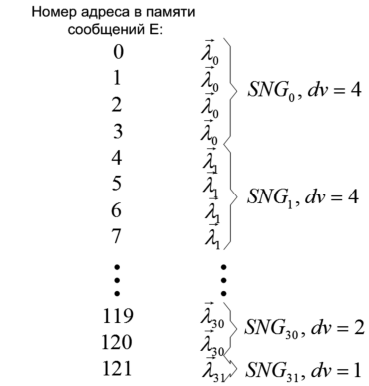

Рис. 2. Процесс инициализации E памяти канальной информацией

4. Блок обработки проверочных узлов (БОПУ). В состав декодера входят z БОПУ блоков. Все блоки обрабатывают сообщения параллельно во времени. Вычисление внешних сообщений производится следующим образом. В соответствии с группой проверочных узлов базовой матрицы $\mathbf{H_b}$ из E памяти считывается последовательно dc LLR_WORD слов. Например, для первой проверочной группы, в соответствии с базовой матрицей, считываются сообщения с адресами – 0,4,8,…, 108, 112. Каждое LLR_WORD слово, считываемое из памяти, сдвигается в сдвиговом регистре 1 в соответствии с его индексом сдвига, считываемым из постоянной памяти. После считывания dc сообщений в каждом БОПУ блоке находится по 29 LLR значений. Затем каждый блок обрабатывает полученные LLR значений в соответствии с определенной методикой вычисления внешних сообщений (в данной работе для вычислений используется метод [8]). Вычисленные $29 * z$ внешних (обновленных) сообщений в виде LLR_WORD слов последовательно сдвигаются в соответствии с их индексами сдвига и записываются в E память в соответствии с адресами, равными адресам считывания. На этом процесс вычисления внешних сообщений и записи обновленных LLR значений в E память заканчивается.

5. Блок обработки символьных узлов (БОСУ) (рис. 3). В состав декодера входят z БОСУ блоков. Все блоки обрабатывают сообщения параллельно во времени. Вычисление внутренних сообщений производится следующим образом. В соответствии с группой символьных узлов базовой матрицы из E памяти считывается последовательно dv LLR_WORD слов. Например, для первой символьной группы, в соответствии с базовой матрицей, считываются слова с адресами – 0,1,2,3. После считывания сообщений в каждом БОСУ блоке находится по 4 LLR значения. Затем каждый блок обрабатывает полученные LLR значений (процесс обработки поясняет схема блока, изображенного на рис. 3). Вычисленные сооб-

щения в виде «LLR_WORD» слов записываются в E память в соответствии с адресами, равными адресам считывания. На этом процесс вычисления внутренних сообщений и записи обновленных LLR значений в E память заканчивается.

В зависимости от типа кода, декодируемого декодером, из постоянной памяти считывается сегмент управляющих сигналов, предназначенных для управления всех блоков, входящих в декодер.

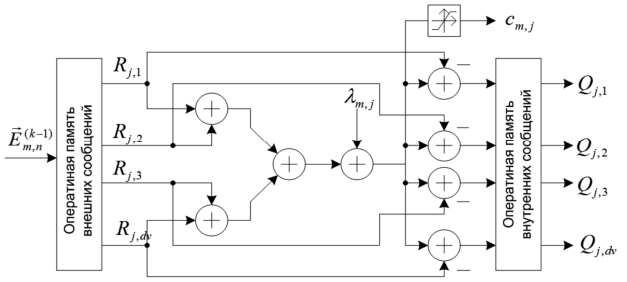

Рис. 3 Блок обработки символьных узлов (БОСУ)

LDA алгоритм декодирования BQC-LDPC и архитектура декодера

Алгоритм декодирования

Для декодирования BQC-LDPC кода используем LDA алгоритм [6-7], модифицированный следующим образом. В LDA алгоритме проверочную матрицу BQC-LDPC кода можно рассматривать как каскадное соединение нескольких блок строк или слоёв, число которых равно $\mathbf{M_b}$. Основная особенность LDA алгоритма заключается в том, что после того как закончится вычисление внутренних сообщений $\mathbf{Q}$ для текущего слоя, эти сообщения немедленно используются для вычисления внешних сообщений $\mathbf{R}$ и обновления апостериорных сообщений $\mathbf{\Lambda}$. Обновленные $\mathbf{\Lambda}$ сообщения используются для вычисления внутренних сообщений следующего слоя и т.д.

Каждая итерация декодирования в алгоритме состоит из $\mathbf{M_b}$ числа подитераций. В начале процесса декодирования вектор $\mathbf{\Lambda}$ сообщений инициализируется вектором канальных сообщений $\boldsymbol{\lambda}$ и используется далее для обработки проверочных узлов первого ряда блоков (первой группы проверочных узлов). В процессе вычисления внешних сообщений $\mathbf{R}$ вычисленные внутренние сообщения $\mathbf{Q}$ используются «на лету», без запоминания в памяти. После завершения обработки первого блока строк апостериорные сообщения $\mathbf{\Lambda}$ обновляются на основе внутренних и внешних сообщений. На этом заканчивается первая подитерация. Во время обработки второй группы проверочных узлов (второго блока строк) сразу же используются апостериорные сообщения от предыдущей подитерации (апостериорные сообщения, вычисленные для первого блока строк). Аналогичным образом обрабатываются все группы проверочных узлов, входящих в базовую матрицу. С завершением обработки последнего блока строк/слоя (последней подитерации) проверочной матрицы завершается первая итерация итеративного декодирования.

Послойный алгоритм декодирования для BQC-LDPC кодов включает следующие основные шаги.

1. Инициализация

Для итерации $k = 0$ положить все значения $\vec{R}_{l,n} = 0$ и $\vec{\Lambda}_q^{(0)} = 2 * \vec{\lambda}_q / \delta^2$ для $q = 1, 2, \ldots, \mathbf{N}_b$

2. Для k - ой итерации

$for\ l = 1, 2, \ldots, \mathbf{M}_b$ do {цикл для подитераци}

$for\ n = 1, 2, \ldots, dc$ do {параметр dc определяет число ненулевых перестановочных матриц в текущем слое, $n = p_q$, $q = 1, 2, \ldots, \mathbf{N}_b$, p_q определяет положение перестановочной матрицы в текущем слое}

$\vec{Q}_{l,n}^{(k)} = [\vec{\Lambda}_{l,n}^{(k-1)}]^{S(n)} - \vec{R}_{l,n}^{(k-1)}$ {$S(n)$ есть индекс сдвига для n - ой перестановочной матрицы в слое l, $\rightarrow$ обозначает вектор сообщения размерности z }

$end\ for$

$for\ j = 1, 2, \ldots, z$ do {вычисление внешних сообщений}

$for\ n = 1, 2, \ldots, dc$ do

$\vec{R}_{n,j}^{(k)} = f(\vec{Q}_{n,j}^{(k)})$ {функция $f()$ определяет один из возможных методов вычисления внешних сообщений}

$end\ for$

$end\ for$

$for\ n = 1, 2, \ldots, dc$ do {обновление Λ_j (a posteriori) сообщений }

$\vec{\Lambda}_n^{(k)} = [\vec{Q}_{l,n}^{(k)} + \vec{R}_{l,n}^{(k)}]^{S(n)}$ {Первая операция сложения, потом циклический сдвиг результата в соответствии с индексом сдвига $S(n)$}

$end\ for$

$end\ for$ do {конец цикла для подитераций}

3. Проверить кодовое слово $\hat{c} = [\hat{c}_0, .., \hat{c}_{n-1}]$ $for\ j = 0, \ldots, N_B * z - 1$ где $\hat{c}_j = 1$ если $\Lambda_j^{(k)} > 0$, иначе $\hat{c}_j = 0$.

Если $\hat{\mathbf{c}}\mathbf{H}^{\mathbf{T}} = 0$ или число итераций равно максимальному лимиту то стоп, иначе $k = k + 1$ и переход к шагу 2.

Архитектура декодера

Архитектура декодера (рис. 4) включает следующие основные компоненты.

1. Постоянную память. Структура и объем постоянной памяти LDA декодера аналогичны структуре и объему постоянной памяти TPMDA декодера.

2. Оперативную память для хранения, считывания и записи апостериорных сообщений ($\mathbf{\Lambda}$ память). Размерность памяти равна $\mathbf{N}_b$ *LLR_WORD слов. Твердое решение декодера (двоич ный вектор декодированного кода) выполняется при обработке апостериорной информации, считываемой из $\mathbf{\Lambda}$ памяти, и является выходом декодера только тогда, когда выполняется условие $\hat{\mathbf{c}}\mathbf{H}^{\mathbf{T}} = 0$.

3. Декодер, включающий в себя z вычислительных блоков для вычисления внешних сообщений $\mathbf{R}$. Структуру вычислительного блока поясняет рис. 5. Предыдущие внешние сообщения считываются из памяти $\mathbf{R}$, вычитаются из предыдущих апостериорных сообщений и, как новые внутренние сообщения $\mathbf{Q}$,вводятся в элемент FIFO для определенной задержки, пока новые внешние сообщения не будут вычислены в блоке вычисления функции $f(\vec{Q}_{n,j}^{(k)})$. Новые внешние $\mathbf{R}$ сообщения и задержанные новые внутренние сообщения $\mathbf{Q}$ используются для вычисления новых апостериорных сообщений. Вычисленные внешние сообщения запоминаются в памяти внешних сообщений. Для вычисления внешних сообщений в блоке вычисления функции $f(\vec{Q}_{n,j}^{(k)})$ могут использоваться различные методы. Выбор метода зависит от требуемой эффективности декодирования и ограничений на аппаратурные затраты [4].

4. Оперативную память для хранения, считывания и записи внешних сообщений ($\mathbf{R}$ память). Память делится на z автономных модулей, каждый из которых связан со своим вычислительным блоком. Размерность модуля памяти зависит от метода вычисления сообщений. При прямом методе вычисления сообщений (без упрощений) для запоминания вычисленных значений требуется $\mathbf{M}_B * dc * LLR$ элементов памяти (включая знак каждого сообщения). При упрощенном вычислении внешних значений, например при использовании метода Min-Sum [4], требования к памяти уменьшаются. Так, вместо запоминания $dc * LLR$ элементов для одного слоя, требуется запомнить dc знаков для каждого сообщения, входящего в слой, два значения LLR с минимальными значениями и положение первого минимума в слое.

5. Сдвиговые регистры, предназначенные для сдвига апостериорных сообщений в соответствии с индексами сдвига, считываемыми с постоянной памяти. В зависимости от типа кода, декодируемого декодером, из постоянной памяти считывается сегмент управляющих сигналов, предназначенных для управления всех блоков, входящих в декодер.

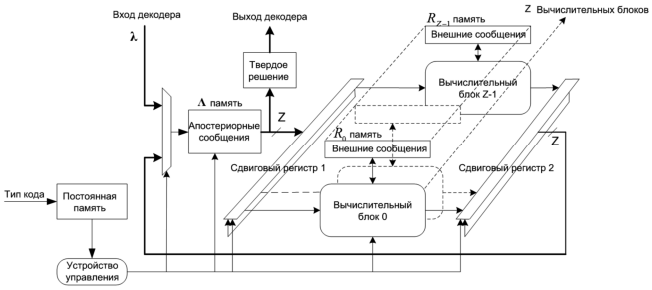

Рис. 4. Архитектура декодера с послойным алгоритмом декодирования

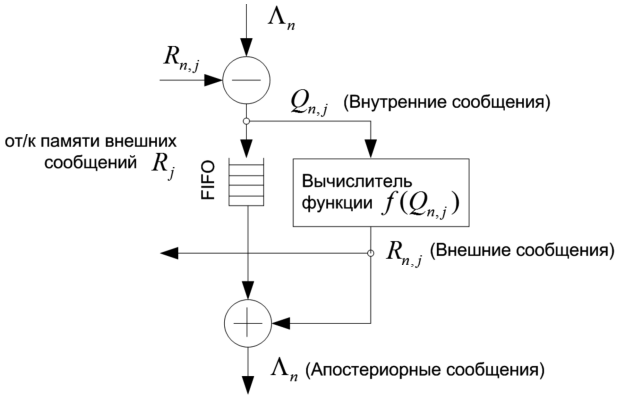

Рис. 5. Структура вычислительного блока

Результаты моделирования эффективности декодирования и скорости сходимости TPMDA и LDA декодирующих алгоритмов

В настоящей работе представлены результаты моделирования LDPC кодов стандарта «WPAN» с кодовыми скоростями R = 1/2, 3/4 и 7/8. Данные коды являются нерегулярными, блоковыми квази-циклическими низкоплотностными кодами с длиной кода равной 672 бит. При моделиривании декодеров использовалась арифметика с фиксированной запятой. Для представления мягких решений (LLR значений) демодулятора и сообщений внутри декодера использовалася прямой код с числом разрядов b = 6. При вычисления внешних сообщений в функции $f(\vec{Q}_{n,j}^{(k)})$ использовался алгоритм [8].

На рис. рис. 6, 8 и 10 представлены результаты исследования эффективности декодирования декодеров при различных скоростях кода. Следует отметить, что LDA декодер показывает улучшенный выигрыш от кодирования (coding gain) по сравнению с TPMDA декодером в области высокого отношения сигнал-шум при низком значении итераций, выполняемых декодером (5 итераций). Данная особенность декодера может быть использована в высокопроизводительных декодерах, когда число итераций должно быть минимальным. Наиболее высокий выигрыш от кодирования, при различном числе итераций, показывает LDA декодер для кода с кодовой скоростью R = 1/2. Следует также отметить, что с увеличением числа итераций выигрыш от кодирования умень-

шается для обоих декодеров (увеличение числа итераций незначительно изменяет эффективность декодирования), особенно для кодов с высокой кодовой скоростью.

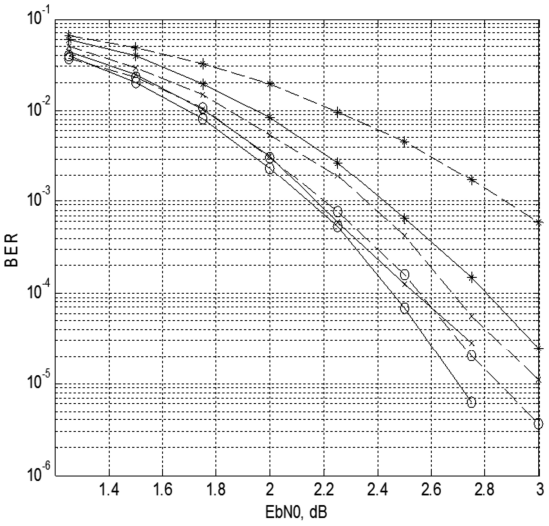

*Рис. 6. Эффективность TPMDA ('- -' линия) и LDA ('—' линия) декодеров для кодовой скорости R = 1/2 при числе ите-раций: * - 5 итераций; x - 10 итераций; o – 15 итераций*

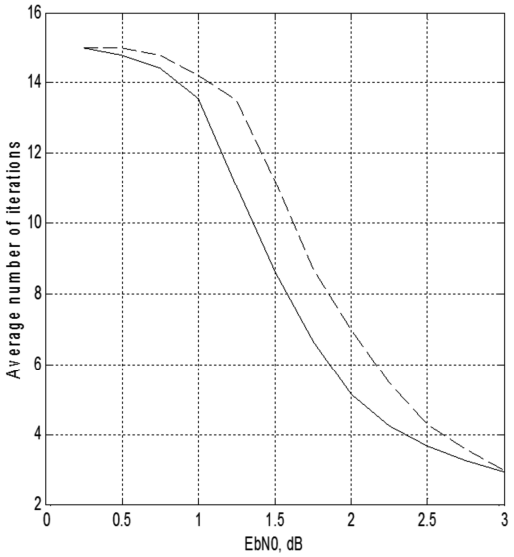

Рис. 7. Скорости сходимости деко -дирования TPMDA ('- -' линия) и LDA ('—' линия) алгоритмов для кодовой скорости R = 1/2 при числе итераций равном 15

На рис. рис. 7, 9 и 11 представлены результаты исследования скорости сходимости декодирующих TPMDA и LDA алгоритмов. Под скоростью сходимости алгоритма понимается среднее число итераций, необходимых для достижения определенной эффективности декодирования. Скорость сходимости декодирующих алгоритмов моделировалась при условии, что все коды декодировались с максимальным числом итераций, равном 15.

Анализ результатов моделирования показывает, что скорость сходимости LDA алгоритма значительно лучше, чем TPMDA алгоритма при низких кодовых скоростях. Скорость сходимости LDA алгоритма уменьшается по мере увеличения кодовой скорости кода и практически сравнима со скоростью сходимости TPMDA алгоритма при высоких кодовых скоростях.

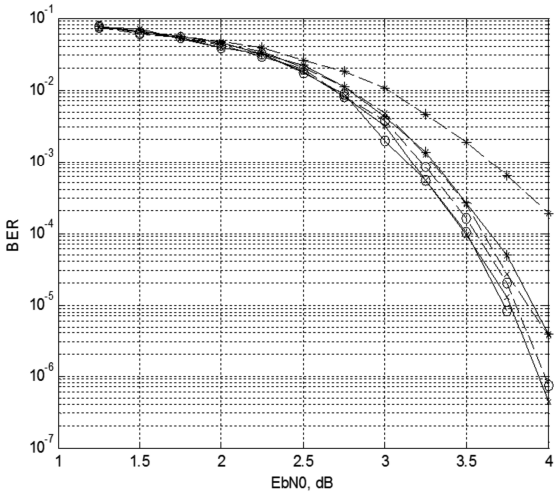

*Рис. 8. Эффективность TPMDA ('- -' линия) и LDA ('—' линия) декодеров для кодовой скорости R = 3/4 при числе итераций: * - 5 итераций; x - 10 итераций; o – 15 итераций*

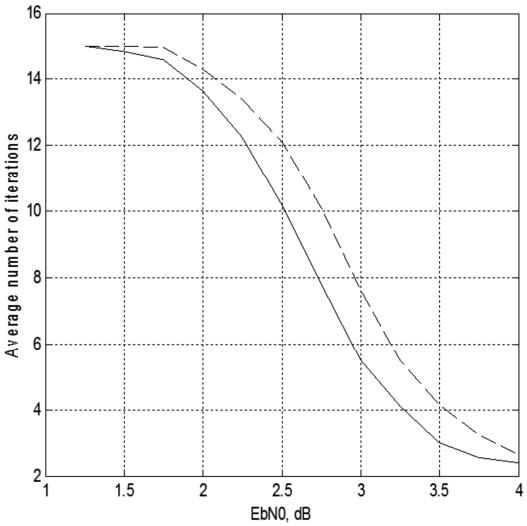

Рис. 9. Скорости сходимости деко-дирования TPMDA ('- -' линия) и LDA ('—' линия) алгоритмов для кодовой скорости R = 3/4 при числе итераций равном 15

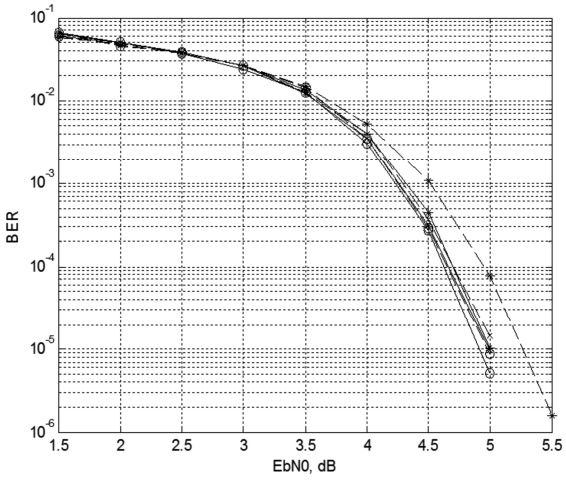

*Рис. 10. Эффективность TPMDA (' - -' линия) и LDA ('—'линия) декодеров для кодовой скорости R = 7/8 при числе итераций: * - 5 итераций; x - 10 итераций; o – 15 итераций*

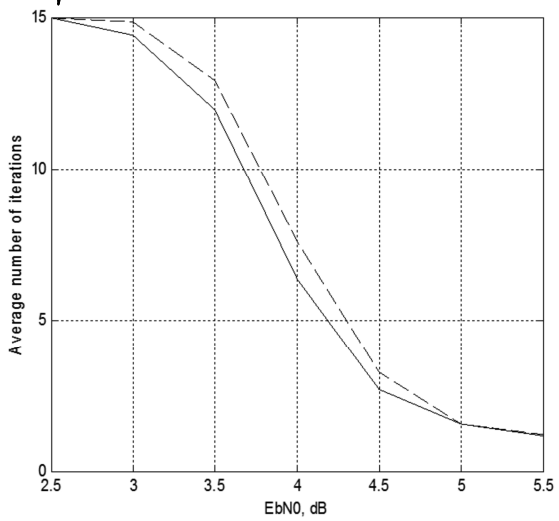

Рис. 11 Скорости сходимости деко-дирования TPMDA ('- -' линия) и LDA ('—'линия) алгоритмов для кодовой скорости R = 7/8 при числе итераций равном 15

В заключение оценим объемы оперативной памяти, необходимые при реализации и декодеров.

TPMDA декодер

1. $\boldsymbol{E}$ память, необходимая для записи, хранения и считывания сообщений равна E*LLR_WORD слов, где E параметр определяет число ненулевых перестановочных матриц в базовой проверочной матрице. Например, для кода с кодовой скоростью 7/8 объем равен 122*LLR_WORD слов.

2. Память канальных сообщений равна $\mathbf{N_b}$ *LLR_WORD слов.

3. Каждый блок обработки проверочных узлов (БОПУ) требует $dc_{\max}$ *LLR элементов памяти и каждый блок обработки символьных узлов (БОСУ) требует $(dv_{\max}+1)$ *LLR элементов памяти, где $dc_{\max}$ есть максимальная степень группы проверочных узлов и $dv_{\max}$ есть максимальная степень группы символьных узлов в базовой матрице.

LDA декодер

1. Память для записи, хранения и считывания внешних сообщений R состоит из z модулей. Каждый модуль включает $dc_{\max} * \mathbf{M}_B * LLR$ элементов памяти. В общем все модули включают $dc_{\max} * \mathbf{M}_B * LLR * z = dc_{\max} * \mathbf{M}_B * LLR\_WORD$ элементов памяти. Например для кода с кодовой скоростью 7/8 объем равен 32*4=128 LLR_WORD слов.

Размерности $\boldsymbol{E}$ и $\boldsymbol{R}$ памятей практически соизмеримы.

2. Память апостериорных сообщений $\boldsymbol{\lambda}$ равна $\mathbf{N_b}$ *LLR_WORD слов.

3. Каждый вычислительный блок требует $dc_{\max}$ *LLR элементов памяти и один элемент задержки (FIFO) с глубиной равной $dc_{\max}$.

Литература

1. D.J.C. MacKay and R.M. Neal, «Near Shannon limit performance of low density parity check codes», Electronics Letters, vol. 33, no. 6, pp. 457-458, 13 March 1997.

2. X-Y Hu, E. Eleftherior, D-M. Arnold, and A. Dholaki. Efficient implementation of the Sum-product algorithm for decoding LDPC codes, Proc. 2001 IEEE Globe Com Conf., pp. 1036-1036E, Nov. 2001.

3. J. Chen, A. Dholakia, E. Eleftherior, M.P.C. Fossorier, and Xiao-Y. Hu. Reduced-Complexity Decoding of LDPC Codes. IEEE Trans. On Communications, vol. 53, pp. 1288-1298, August 2005

4. Кравченко А.Н. Снижение сложности декодирования низкоплотностного кода. «Цифровая обработка сигналов». 2010, № 2, С.35-41.

5. E. Yeo et al., High throughput low-density parity-check decoder architectures, in Proc. IEEE GLOBECOM, San Antonio, TX, Nov. 2001, pp. 3019–3024.

6. Hocevar D.E. A reduced complexity decoder architecture via layered decoding of LDPC codes, Signal Processing Systems, 2004. SISP 2004. IEEE Workshop on pp. 107-112, 13-15 Oct. 2004.

7. M. M. Mansour and N. R. Shanbhag. High-Throughput LDPC Decoders. IEEE Trans. On Very Large Integrated Systems, vol. 11, No. 6, pp. 976-996, December 2003.

8. F. Guilloud, E. Boutillon, and J.-L. Danger. Lambda-min decoding algorithm of regular and irregular ldpc codes. In Proceedings of 3rd International Symposium on Turbo Codes & Related Topics, 1-5 Sept. 2003.