ДЕКОДИРОВАНИЕ БЛОЧНОГО ТУРБО-КОДА С КОМПОНЕНТНЫМИ РАСШИРЕННЫМИ КОДАМИ ХЭММИНГА

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Якубенко П.Н.

Саломатин С.Б. – к.т.н., доцент

На сегодняшний день практически ни одна система связи не обходится без помехоустойчивого кодирования. Необходимость кодирования убедительно показана в широко известной работе К. Шеннона. С момента публикации этой работы было разработано множество различных кодов. Одним из таких кодов является блочный турбо-код (turbo-product code, TPC), в наиболее общем виде предложенный в 1993 году учеными К. Берроу, А. Главье и П. Ситимашимой. Ими было показано, что использование этого кода в канале с шумами позволяет очень сильно приблизиться к границе Шеннона.

Блочный турбо-код оказался полезным для различных каналов связи. Его использование можно обнаружить в самых разных системах, от спутниковой связи до сетей широкополосного доступа. Особенно хорошо использовать блочные турбо коды при высоких требованиях к пропускной способности, задержке, спектральной эффективности и помехоустойчивости.

Как и для большинства помехоустойчивых кодов, самой сложной задачей для применения блочного турбо-кода на практике является эффективное декодирование. Алгоритмы максимального правдоподобия имеют огромную вычислительную сложность, поэтому необходимы алгоритмы, имеющие оптимальное соотношение сложности и эффективности.

Целью данной статьи является представление одного из таких алгоритмов, который позволяет декодировать блочный турбо-код в случае, когда компонентным кодом является расширенный код Хэмминга. Алгоритм разработан для реализации на логической схеме на основе известного более общего алгоритма. Использование предлагаемой модификации позволяет достичь высокой пропускной способности по сравнению с известным алгоритмом. Также уменьшается количество ошибок при относительно высоком отношении сигнал-шум

В общем случае TPC строится следующим образом: выбирается два линейных кода в систематической форме с параметрами (n_1,k_1) и (n_2,k_2) соответственно. Из информационных бит формируется матрица размера $k_2 \times k_1$. Сначала строки этой матрицы кодируются первым кодом, и полученные проверочные биты приписываются к начальной матрице, образуя в итоге матрицу размера $k_2 \times n_1$. Затем все столбцы полученной матрицы кодируются вторым кодом, и в итоге получается матрица размером $n_2 \times n_1$:

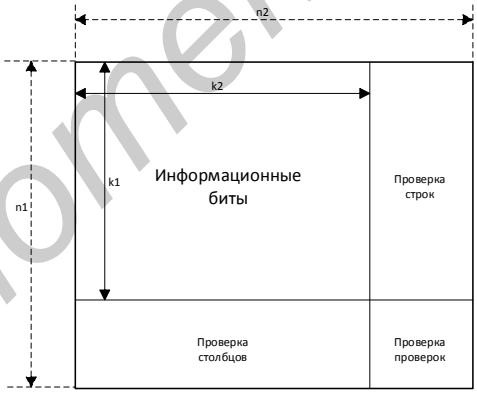

Рисунок 1 – матрица блока турбо-кода

Биты в части «проверка проверок» выражаются через суммы бит подматриц информационной матрицы и не зависят от порядка суммирования. Другими словами, строки с номерами, большими k_1, будут являться кодовыми словами первого кода как и остальные строки.

На вход декодера поступает матрица логарифмических отношений правдоподобия (LLR) размером $n_2 \times n_1$. Будем обозначать ее $\boldsymbol{R}$.

На верхнем уровне алгоритм декодирование разбивается на итерации, каждая из которых состоит из двух полуитераций. Первая полуитерация декодирует строки матрицы SISO алгоритмом, т.е. декодером с мягким входом и мягким выходом, вторая – столбцы. Внутри полуитераций декодирование может происходить последовательно или параллельно. Выходом полуитерации будет являться добавочная информация $\boldsymbol{W}(i-1) = \boldsymbol{R}' - \boldsymbol{R}(i-1)$, где $\boldsymbol{R}'$ – матрица из выходов SISO декодеров. На вход следующей полуитерации поступает матрица $\boldsymbol{R}(\boldsymbol{i})$, полученная по формуле:

$$\boldsymbol{R}(i) = \boldsymbol{R} + \alpha(i)\boldsymbol{W}(i)$$

На каждой полуитерации также генерируется флаг остановки, принимающий значение 1 тогда и только тогда, когда синдромы всех строк (столбцов) на входе равнялись 0. Если флаг остановки принимает значение 1 на двух полуитерациях подряд, это значит, что алгоритм сошелся и дальнейшего изменения бит не будет. В этой ситуации алгоритм прерывается, и выдает жесткое решение для бит информационной части матрицы. Если этого не происходит через некоторое число итераций, принятое максимальным, алгоритм выдает информацию.

Опишем работу SISO декодера для расширенного кода Хэмминга.

На вход декодера подается вектор $\boldsymbol{r}$, являющийся строкой или столбцом матрицы $\boldsymbol{R}(i)$. Генерируем вектор максимального правдоподобия $\boldsymbol{h}$, по принципу $r_i \geq 0 \Rightarrow h_i = 0$; $r_i < 0 \Rightarrow h_i = 1$.

Пользуемся фактом, что последний бит расширенного кода Хэмминга является суммой всех остальных. Находим сумму всех бит $\boldsymbol{h}$ по модулю два, а также синдром обычного (не расширенного) кода Хэмминга для вектора без последнего бита. Тогда $\boldsymbol{h}$ будет являться кодовым словом в том и только в том случае, когда сумма равна 0 и все компоненты синдрома равны 0. Далее выделяем из входной мягкой информации вектор надежности $\boldsymbol{m}$, получаемый взятием модулей компонент. Затем $\boldsymbol{h}$ корректируется так, чтобы синдром был ненулевым, а четность равнялась нулю. Для этого в случае ненулевой четности один бит меняет значение на противоположное, при этом выбирается либо бит с минимальной надежностью, либо второй минимальный в случае, когда изменение бита с наименьшей надежностью приводит к получению кодового слова. В случае когда $\boldsymbol{h}$ изначально является кодовым словом, два бита с минимальной надежностью изменяют значения на противоположные. Вместе с изменением битов в $\boldsymbol{h}$ изменяется также и $\boldsymbol{m}$: на тех позициях, где поменялся бит, надежность изменяется на отрицательную. Также модифицируется синдром, чтобы он соответствовал новому значению $\boldsymbol{h}$.

После модификации $\boldsymbol{h}$ и $\boldsymbol{m}$ список кандидатов формируется следующим образом. На сфере с центром в точке $\boldsymbol{h}$ и радиусом 2 (в метрике Хэмминга) выбираются все кодовые слова. Для этого все биты разбиваются на пары так, чтобы изменение двух бит в любой паре приводило к получению кодового слова. Чтобы получить пару для бита на некоторой позиции, нужно взять соответствующий столбец проверочной матрицы, вычесть из него синдром, и найти номер столбца проверочной матрицы, в которой стоит полученный вектор.

По итогу такого разбиения получаем список из $n/2$ кодовых слов. Аналогично с оригинальным алгоритмом [1], выбираем решение $\boldsymbol{d}$ по критерию минимальности нормы Евклида. При этом сами нормы можно не вычислять, если воспользоваться следующим преобразованием (полученным с учетом, что векторы $\boldsymbol{d}$ и $\boldsymbol{c}$ состоят из 1 и -1):

$$\frac{||\boldsymbol{c}-\boldsymbol{r}||^2 - ||\boldsymbol{d}-\boldsymbol{r}||^2}{4} = \sum_{c_j \neq d_j} d_j r_j = \sum_{c_j \neq d_j} d_j h_j m_j = \sum_{\substack{c_j \neq d_j \\ d_j = h_j}} m_j - \sum_{\substack{c_i \neq d_i \\ d_i \neq h_i}} m_i \quad (1)$$

В нашем случае последнее выражение имеет простой смысл и легко вычисляется. Векторы $\boldsymbol{d}$ и $\boldsymbol{c}$ различаются в четырех компонентах, так как оба получены из вектора $\boldsymbol{h}$ изменением некоторой пары. При этом в первую сумму войдут те компоненты, которые менялись для получения вектора $\boldsymbol{c}$, а во вторую те, которые менялись для получения вектора $\boldsymbol{d}$. Значит, для нахождения вектора с минимальной нормой достаточно вычислить суммы в каждой паре и найти минимальную. Каждый бит входит в какую-то из пар, значит оценка надежности каждого бита может быть произведена по формуле (1). В случае, когда бит соответствует решению, вычитается минимальная сумма из второй минимальной.

Ниже приведены графики, которые получены в моделях с количеством итераций декодирования равно 8.

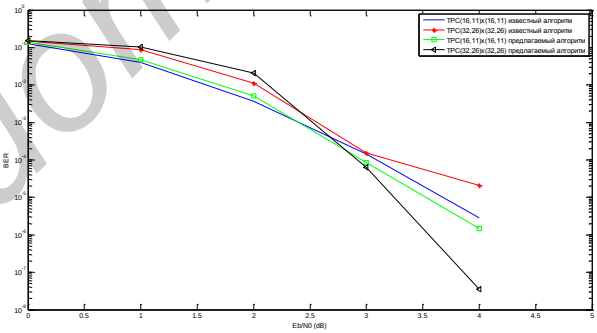

Рисунок 2 – сравнение BER оригинального и модифицированного алгоритмов.

Таблица 1 – сравнение сложности алгоритмов по количеству операций.

	Произведений	Сумм	Сравнений
Известный (16,11)	256	512	320
Предлагаемый (16,11)	0	160	280
Известный (32,26)	512	1040	700
Предлагаемый (32,26)	0	320	600

Список использованных источников:

1. R.Pyndiah, A.Glaviex, A.Picart, S.Jacq, ”Near Optimum Decoding of Product codes”, IEEE Globecom’94

2. C.Berrow, A.Glavieux, P.Thitimajshima, “Near Shannon limit error-correcting coding and decoding: Turbo-codes (1)”, IEEE Int. Conf. on Comm. ICC’93.