УДК 519.725.2

И. Ю. Сысоев, Э. М. Габидулин

Московский физико-технический институт (государственный университет)

Декодирование ранговых кодов с использованием слабоортогонального базиса

Процедура вычисления синдрома является важной частью операции декодирования рангового кода. В работе показано, что, используя слабый самоортогональный базис, можно уменьшить сложность этой операции. Оценка сложности приблизительно определяется величиной $n(\log n)^2$. Оценка меньше, чем оценка сложности операции при использовании только стандартного базиса и известного алгоритма Карацубы–Офмана.

Ключевые слова: помехоустойчивое кодирование, ранговые коды, коды Габидулина, синдром, слабый самоортогональный базис, быстрые вычисления, алгоритм Карацубы–Офмана, ПЛИС, реализация, оптимизация.

1. Введение

Ранговые коды представляют интерес для многих областей применения: для коммуникационных технологий, криптографии, пространственно-временного кодирования, случайного сетевого кодирования. Ранговое декодирование является достаточно сложной процедурой и характеризуется большим количеством умножений в расширенном поле Галуа. Поэтому разработчику необходимо решать много проблем при разработке системы, осуществляющей декодирование ранговых кодов. К таким проблемам относятся борьба с большими задержками, высокое потребление мощности и ресурсов кристалла интегральной схемы.

Одним из эффективных способов решения перечисленных проблем является уменьшение сложности алгоритма. Этап вычисления синдрома в процессе декодирования занимает значительную долю во всей процедуре декодирования. В работе будет показано, как можно уменьшить сложность вычисления синдрома, используя слабоортогональный базис.

Основная часть данной работы организована следующим образом. В разделе 2 приводится обзор ранговой метрики и ранговых кодов. В разделе 3 выводится оценка сложности вычисления синдрома рангового кода стандартным методом и вклад этой операции в общую сложность процедуры декодирования. Слабый самоортогональный базис вводится в разделе 4. В разделе 5 рассматриваются используемые быстрые алгоритмы умножения. В разделе 6 обсуждается особенности реализации блока вычисления синдрома с использованием слабого самоортогонального базиса. В разделе 7 обсуждается эффективность нового алгоритма.

2. Ранговые метрика и ранговые коды

На практике в большинстве случаев используется метрика Хэмминга. Однако представляют интерес другие метрики, в частности *ранговая метрика*.

Ранговая норма матрицы X, определяемая в виде $N_R k(X)$, может быть вычислена как количество линейно-независимых строк матрицы. *Ранговое расстояние* между матрицами X и Y определено как ранг разности этих матриц:

$$d_{Rk}(X, Y) = Rk(X - Y). \tag{1}$$

Кодирование с использованием ранговой нормы, или ранговое кодирование, подходит для тех каналов передачи данных, в которых одна и та же ошибка распространяется на различные символы кодового слова, например в сетевом кодировании [16]. В качестве примера стоит привести способ передачи данных под названием «сетевое кодирование» [1].

Введём необходимые обозначения. Обозначим через $GF(q)$ базовое конечное поле из q элементов, где q – степень простого числа. $GF(q^N)$ – это расширение базового поля степени N. Если α – примитивный элемент $GF(q^N)$, то каждый ненулевой элемент β из $GF(q^N)$ есть некоторая степень α, то есть $\beta = \alpha^s$. Более того, для неравных целых r и s существует целое k, такое, что $\alpha^r - \alpha^s = \alpha^k$.

Введём понятие рангового кода [9]. Пусть X^n – n-мерное векторное пространство над полем $GF(q^N)$. Пусть $u_1, u_2, \ldots, u^N$ – некоторый фиксированный базис поля $GF(q^N)$, рассматриваемого как векторное пространство над $GF(q)$. Любой элемент $x_i \in GF(q^N)$ однозначно представляется в виде $x_i = a_{1i}u_1 + a_{2i}u_2 + \ldots + a_{Ni}u_N$. Пусть A_N^n означает совокупность всех $(N \times n)$-матриц с элементами из $GF(q)$. Можно задать биекцию $A : X^n \to A_N^n$ правилом: для любого вектора $x = (x_1, x_2, \ldots, x_n)$ можно найти соответствующую матрицу

$$A(x) = \begin{Vmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{N1} & a_{N2} & \ldots & a_{Nn} \end{Vmatrix}. \tag{2}$$

Рангом вектора x над $GF(q)$ называется ранг матрицы $A(x)$. То есть ранг вектора – это максимальное число его координат, линейно независимых над $GF(q)$. Ранг вектора x над $GF(q)$ обозначается как $r(x; q)$. Аналогично рангом $(r \times n)$-матрицы H с элементами из $GF(q^N)$ над $GF(q)$ называется максимальное число столбцов, линейно независимых над $GF(q)$. Обозначим ранг матрицы, используя $r(H; q)$. Обозначим ранг $r(H; q)$. Очевидно, что $r(H; q) \geq r(H; q^N)$.

Представим класс кодов максимального рангового расстояния для длин $n \leq N$. Эти коды являются аналогами обобщённых кодов Рида–Соломона [17]. Для удобства записи введём обозначение $[i] = q^i$, $i = 0, \pm 1, \ldots$

Пусть $h_i \in GF(q^N)$, $i = 1, 2, \ldots, n$, и пусть эти элементы линейно независимы над $GF(q)$. Зададим целое $d \leq n$. Образуем матрицу

$$H = \begin{Vmatrix} h_1 & h_2 & \ldots & h_n \\ h_1^{[1]} & h_2^{[1]} & \ldots & h_n^{[1]} \\ \ldots & \ldots & \ldots & \ldots \\ h_1^{[d-2]} & h_2^{[d-2]} & \ldots & h_n^{[d-2]} \end{Vmatrix}. \tag{3}$$

Код с проверочной матрицей вида (3) является кодом с максимальным ранговым расстоянием длины n с расстоянием d. При этом порождающая матрица кода имеет вид

$$G = \begin{Vmatrix} g_1 & g_2 & \ldots & g_n \\ g_1^{[1]} & g_2^{[2]} & \ldots & g_n^{[1]} \\ \ldots & \ldots & \ldots & \ldots \\ g_1^{[k-1]} & g_2^{[k-1]} & \ldots & g_n^{[k-1]} \end{Vmatrix}, \tag{4}$$

где $k = n - d + 1$, а элементы $g_1, g_2, \ldots, g_n$ линейно независимы над $GF(q)$. Для проверочной и порождающей матриц должно выполняться условие

$$GH^T = 0. \tag{5}$$

Кодирование ранговых кодов во многих случаях сводится к вычислению значений линеаризованных многочленов в поле $GF(q^N)$. Пусть порождающая матрица имеет вид (4). Тогда кодовое слово имеет вид

$$g = (F(g_1), F(g_2), \ldots, F(G_n)), \tag{6}$$

где

$$F(z) = \sum_{i=0}^{k-1} u_i z^{[i]}, \tag{7}$$

причём $u_0, \ldots, u_{k-1}$ являются информационными символами.

Коды с проверочной матрицей вида (3) допускают декодирование с помощью алгоритма, который подобен алгоритму декодирования обобщённых кодов Рида-Соломона.

Пусть $g = (g_1, \ldots, g_n)$ – кодовый вектор, $e = (e_1, \ldots, e_n)$ – вектор ошибки $y = g + e$ – принятый вектор. После получения вектора y из канала связи инеобходимо вычислить синдром:

$$s = (s_0, s_1, \ldots, s_{d-2}) = yH^T = eH^T. \tag{8}$$

Задача декодера – по известному вектору синдрома s найти вектор ошибки e. Пусть ранговая норма вектора ошибки равна m. Тогда ошибку e можно представить в виде

$$e = EY = (E_1, \ldots, E_m)Y, \tag{9}$$

где величины $E_1, \ldots, E_m$ линейно независимы над $GF(q)$, а $Y = (Y_{ij})$ – $(m \times n)$-матрица ранга m с элементами из $GF(q)$. Тогда вместо (8) можно записать

$$s = EYH^T = EX, \tag{10}$$

где матрица $X = YH^T$ имеет вид

$$X = \begin{Vmatrix} x_1 & x_1^{[1]} & \ldots & x_1^{[d-2]} \\ x_2 & x_2^{[1]} & \ldots & x_2^{[d-2]} \\ \ldots & \ldots & \ldots & \ldots \\ x_m & x_m^{[1]} & \ldots & x_m^{[d-2]} \end{Vmatrix}, \tag{11}$$

причём величины

$$x_p = \sum_{j-1}^{n} Y_{pj} h_j, p = 1, \ldots, m, \tag{12}$$

линейно незавимимы над $GF(q)$. Равенство (9) эквивалентно системе уравнений относительно неизвестных $E_1, \ldots, E_m, x_1, \ldots, x_m$:

$$\sum_{i=1} E_i x_i^{[p]} = s_p, p = 0, 1, \ldots, d-2. \tag{13}$$

Пусть найдено решение этой системы. Тогда из (12) определяется матрица Y, а из (9) – вектор ошибки e. Отметим, что система (13) при заданном m имеет много решений, однако при $m \leq (d-1)/2$ все решения приводят к одному и тому же вектору e.

3. Сложность вычисления синдрома

Проанализируем один из этапов процедуры декодирования, а именно систему вычисления синдрома и поиска вектора ошибок рангового кода (n, k, d). Здесь n – длина кодового вектора и в нашем варианте равняется размерности расширенного поля. Примем, что размерность поля N равняется длине кодового вектора n, k – количество информационных векторов, d – кодовое расстояние, определяющее максимальный ранг ошибки. Интересным представляется случай $n = 2k$, поэтому примем $n \propto k$.

Заметим, что операция вычисления синдрома выполняется всегда для каждого принятого кодового вектора, в то время как операция поиска вектора ошибок выполняется только в случае ненулевого значения синдрома, полученного на первом этапе.

Поскольку операция вычисления синдрома есть умножение вектора на матрицу, то сложность алгоритма равняется сложности вычисления результатов от n^2k умножений. В этом случае асимптотически сложность алгортма вычисления синдрома «простым» способом оценивается как $\mathcal{O}(n^2k)$ или как $\mathcal{O}(n^3)$ умножений.

В работе [1] сложность алгоритма поиска вектора ошибок оценивается величиной, равной $\frac{5}{2}n^2 - \frac{3}{2}k^2 + \frac{n-k}{2}$ или $\mathcal{O}(n^2) + \mathcal{O}(n)$ умножениям (см., например, [2]). Из приведёных

оценок следует, что одно из возможных направлений эффективных оптимизаций – это уменьшение сложности вычисления синдрома. Далее будет показан способ оптимизации операции вычисления синдрома с помощью такого математического объекта, как слабоортогональный базис.

4. Слабоортогональный базис

Выберем n линейно-независимых элементов расширенного поля

$$g_0(n) = (g_1, g_2, \ldots, g_n), g_j \in GF(q^n). \tag{14}$$

Этот набор (14) является базисом из $GF(q^n)$ над $GF(q)$. Для любого такого набора векторов можно всегда ввести матрицу специального вида

$$G_n = \left\| \begin{array}{cccc} g_1 & g_2 & \ldots & g_n \\ g_1^q & g_2^q & \ldots & g_n^q \\ g_1^{q^2} & g_2^{q^2} & \ldots & g_n^{q^2} \\ \ldots & \ldots & \ldots & \ldots \\ g_1^{q^{n-1}} & g_2^{q^{n-1}} & \ldots & g_n^{q^{n-1}} \end{array} \right\|. \tag{15}$$

Определение 1. Базис $g_0(n) = (g_1, g_2, \ldots, g_n)$ называется слабым самоортогональным базисом, если для матрицы (15) выполняется условие

$$G_n G_n^T = D, \tag{16}$$

где D – это диагональная матрица в $GF(q^n)$, которая не является результатом умножения на единичную матрицу I_n.

Мы будем использовать следствия из следующей леммы.

Лемма 1. *Пусть $\beta \in GF(q^{2n})$ будет элементом, таким что $\beta^{1+q^n} = -1$. Если $g_0(n) = (g_1, g_2, \ldots, g_n)$ является слабым самоортогональным базисом поля F_{q^n}, тогда $g_0(2n) = (g_1, g_2, \ldots, g_n, \beta g_1, \beta g_2, \ldots, \beta g_n)$ есть слабый самоортогональный базис расширенного поля $GF(q^2 n)$.*

Доказательство см. в [3]. Далее в тексте будут рассматриваться только конечные поля, для которых $q = 2$.

Стоит напомнить, что существует *сильный самоортогональный базис*, для которого выполняется условие

$$G_n G_n^T = I. \tag{17}$$

В отличие от слабого самоортогонального базиса, самоортогональный базис не позволяет рекурсивно строить пространства с сохранением свойства самоортогональности.

5. Алгоритмы умножения во встроенных системах

В настоящее время оптимальными для практической реализации считаются алгоритмы, производные от алгоритма умножения по методу Карацубы–Офмана [4], [5], [6], имеющие сложность порядка $\mathcal{O}(n^{\log_2 3})$. При $n \to \infty$ наименьшую асимптотическую сложность имеет алгоритм умножения Шёнхаге–Штрассена [7], имеющий оценку $\mathcal{O}(n \cdot \log n \cdot \log\log n)$. Однако для случая ранговых кодов он не подходит, так как на практике эффективное использование алгоритма Шёнхаге–Штрассена допустимо при $n \geqslant \mathcal{O}(2^{15})$ [8], [9]. По этой причине в данной работе предложено использовать схему умножения Карацубы–Офмана совместно со слабосамоортогональным базисом.

5.1. Вычисление синдрома с использованием слабых самоортогональных базисов

Рассмотрим самый простейший вариант базиса, состоящий из двух векторов

$$b_2 = (1, f_1), \tag{18}$$

где

$$f_1^3 = 1 \tag{19}$$

или

$$f_1^3 = f^1. \tag{20}$$

Этот базис является слабосамоортогональным. Нетрудно убедиться, что

$$\begin{Vmatrix} 1 & f_1^1 \\ 1 & f_1^2 \end{Vmatrix} \cdot \begin{Vmatrix} 1 & 1 \\ f_1^1 & f_1^2 \end{Vmatrix} = \begin{Vmatrix} 1+f_1^2 & 1+f_1^3 \\ 1+f_1^3 & 1+f_1^4 \end{Vmatrix} = \begin{Vmatrix} 1+f_1^2 & 0 \\ 0 & 1+f_1 \end{Vmatrix}. \tag{21}$$

Используя базис (18) и базисный элемент f_2

$$f_2^{2^{2^1}+1} = f_2^5 = 1, f_2 \in GF(2^2), \tag{22}$$

построим базис пространства вдвое большей размерности:

$$b_4 = (1, f_1, f_2, f_2 f_1). \tag{23}$$

В качестве примера оптимизации рассмотрим $GF(2^4)$-линейный ранговый код $(4, 2, 3)$, $d = n-k+1$ с максимальным ранговым расстоянием [10]. Для задания проверочной матрицы H мы будем использовать базис, в котором базисный элемент f_1 задаётся выражением

$$f_1^{2^{2^0}+1} = f_1^3 = 1, f_1 \in GF(2^2). \tag{24}$$

Базис $b_1 = (1, f_1)$ является слабым самоортогональным базисом в $GF(2^2)$, поскольку для него выполняется условие (16):

$$\begin{Vmatrix} 1 & f_1 \\ 1 & f_1^2 \end{Vmatrix} \begin{Vmatrix} 1 & 1 \\ f_1 & f_1^2 \end{Vmatrix} = \begin{Vmatrix} 1+f_1^2 & 0 \\ 0 & 1+f_1^4 \end{Vmatrix} = \hat{D}. \tag{25}$$

Используя базис (18), вектор из (22) и выражение (25), из леммы 1 получаем, что b является слабым самоортогональным базисом в $GF(2^4)$. Тогда на основе базиса b можно построить проверочную матрицу рангового кода

$$H_2 = \begin{Vmatrix} 1 & f_1 & f_2 & f_1 f_2 \\ 1 & f_1^2 & f_2^2 & f_1^2 f_2^2 \end{Vmatrix}. \tag{26}$$

Аналогично, для размерности $n = 8$ проверочная матрица рангового кода может быть записана в виде

$$H_4 = \begin{Vmatrix} 1 & f_1 & f_2 & f_1 f_2 & f_3 & f_3 f_1 & f_3 f_2 & f_3 f_2 f_1 \\ 1 & f_1^{[1]} & f_2^{[1]} & (f_2 f_1)^{[1]} & f_3^{[1]} & (f_3 f_1)^{[1]} & (f_3 f_2)^{[1]} & (f_3 f_2 f_1)^{[1]} \\ 1 & f_1 & f_2^{[2]} & (f_2 f_1)^{[2]} & f_3^{[2]} & (f_3 f_1)^{[2]} & (f_3 f_2)^{[2]} & (f_3 f_2 f_1)^{[2]} \\ 1 & f_1^{[1]} & f_2^{[3]} & (f_2 f_1)^{[3]} & f_3^{[3]} & (f_3 f_1)^{[3]} & (f_3 f_2)^{[3]} & (f_3 f_2 f_1)^{[3]} \end{Vmatrix}. \tag{27}$$

Можно переписать матрицу (26) иначе. Для этого введём обозначения

$$F_{1,2} = \begin{Vmatrix} 1 & f_1 \\ 1 & f_1^{[1]} \end{Vmatrix}, \tag{28}$$

$$F_{2,2} = \begin{Vmatrix} f_2^{[0]} & 0 \\ 0 & f_2^{[1]} \end{Vmatrix}. \tag{29}$$

Тогда (26) преобразуется следующим образом:

$$H_2 = \begin{Vmatrix} F_{1,2} & F_{2,2} \cdot F_{1,2} \end{Vmatrix}. \tag{30}$$

Для $n = 8$ проверочная матрица также может быть представлена компактным образом. Новый вектор f_3, используемый для генерации пространства элементов в $GF(2^8)$ из элементов пространства $GF(2^4)$, приведёт к матрице

$$F_{3,4} = \begin{Vmatrix} f_3^{[0]} & 0 & 0 & 0 \\ 0 & f_3^{[1]} & 0 & 0 \\ 0 & 0 & f_3^{[2]} & 0 \\ 0 & 0 & 0 & f_3^{[3]} \end{Vmatrix}. \tag{31}$$

Сконструируем из (29) матрицу размерности 4:

$$F_{2,4} = \begin{Vmatrix} F_{2,2}^{[0]} & 0 \\ 0 & F_{2,2}^{[2]} \end{Vmatrix}, \tag{32}$$

а из (28) матрицу той же размерности:

$$F_{1,4} = \begin{Vmatrix} F_{1,2} & F_{1,2} \\ F_{1,2} & F_{1,2} \end{Vmatrix} = \begin{Vmatrix} I & I \\ I & I \end{Vmatrix} F_{1,2}. \tag{33}$$

Введём обозначение

$$K_4 = \begin{Vmatrix} F_{1,2} & F_{2,2} \cdot F_{1,2} \\ F_{1,2} & F_{2,2}^{[2]} \cdot F_{1,2} \end{Vmatrix} = \begin{Vmatrix} I & F_{2,2} \\ I & F_{2,2}^{[2]} \end{Vmatrix} F_{1,2}. \tag{34}$$

Тогда проверочная матрица из пространства $GF(2^8)$ примет вид

$$H_4 = \begin{Vmatrix} K_4 & F_{3,4} \cdot K_4 \end{Vmatrix} = \begin{Vmatrix} I & F_{3,4} \end{Vmatrix} K_4. \tag{35}$$

Таким образом, если проверочная матрица строится с использованием слабого самоортогонального базиса, то это построение носит *рекурсивный* характер. Рекурсивная структура позволяет по-новому подойти к операции декодирования рангового кода.

В общем случае при $n = 4$ декодер рангового кода $(4, 2, 3)$ получает вектор $y = (y_0, y_1, y_2, y_3)$ и, используя матрицу (26), вычисляет синдром

$$S = \begin{Vmatrix} s_0 & s_1 \end{Vmatrix} = yH^T = \begin{Vmatrix} y_0 + y_1 f_1 + f_2(y_2 + y_3 f_1) \\ y_0 + y_1 f_1^2 + f_2^2(y_2 + y_3 f_1^2) \end{Vmatrix}^T. \tag{36}$$

Заметим, что каждый элемент $y_i \in GF(2^4)$, $i = 0, 1, 2, 3$, может быть представлен в форме

$$y_i = a_i + f_2 b_i, \tag{37}$$

где $a_i \in GF(2^2)$, $b_i \in GF(2^2)$, а $f_2 \in GF(2^2)$.

Элемент f_2^2 может так же быть представлен по правилу (37), как

$$f_2^2 = \alpha + f_2\beta, \alpha \in GF(2^2), \beta \in GF(2^2). \tag{38}$$

При определённом выборе f_2 можно добиться того, что в (38) $\alpha = 1$, но этот факт принципиально не изменит дальнейшие рассуждения. Тогда

$$\begin{aligned} f_2(y_2 + y_3 f_1) &= f_2(a_2 + f_2 b_2 + (a_3 + f_2 b_3) f_1) = \\ &= f_2 a_2 + \alpha b_2 + f_2 \beta b_2 + f_2(a_3 f_1) + \alpha b_3 f_1 + f_2 \beta b_3 f_1 = \\ &= \alpha(b_2 + b_3 f_1) + f_2(a_2 + a_3 f_1 + \beta(b_2 + b_3 f_1)) = \\ &= r + f_2 t, r \in GF(2^2), t \in GF(2^2). \end{aligned} \tag{39}$$

Заметим, что для вычисления r и t мы должны выполнить 3 сложения по модулю 2 в $GF(2^2)$ и 4 умножения в $GF(2^2)$. Если $C_{GF(2^{2^2})}$ обозначить как сложность предыдущей операции (39), тогда

$$C_{GF(2^{2^2})} = 3ADD_{GF(2^2)} + 4MULT_{GF(2^2)} = C_2. \tag{40}$$

5.2. Обобщение на случай больших полей

В общем случае, если базис является слабым самоортогональным и $n = 2^M$, $M = 0, 1, 2, 3, \ldots$, то, обобщив (40), мы получим

$$C_i = 3ADD_{GF(2^{2^{i-1}})} + 2MULT_{GF(2^{2^{i-1}})} + 2C_{i-1} \tag{41}$$

при $i = 1, 2, 3, \ldots$ ($C_0 = 0$). $MULT_{GF(2^{2^{i-1}})}$ – сложность произведения двух небазисных векторов. В случае $M = 2$ из (41) получаем (40). Поэтому сложность нахождения элемента синдрома $s_i \in GF(2^{2^k})$ есть

$$C^s = \sum_{i=1}^{M} \left(C_i + ADD_{GF\left(2^{2^M}\right)} \right). \tag{42}$$

Если

$$ADD_{GF\left(2^{2^i}\right)} = \mathcal{O}(2^i) \tag{43}$$

и

$$MULT_{GF\left(2^{2^i}\right)} = \mathcal{O}\left(2^{i\log_2 3}\right) \text{ (см. [5])}, \tag{44}$$

тогда из (41) следует

$$\begin{aligned}
C_i &= 3\mathcal{O}(2^{i-1}) + 2\mathcal{O}(2^{(i-1)\log_2 3}) + 2C_{i-1} = \\
&= 3\sum_{j=1}^{i-1} \left(2^{i-1-j}\mathcal{O}(2^j)\right) + 2\sum_{j=1}^{i-1} \left(2^{i-1-j}\mathcal{O}(2^{j\cdot\log_2 3})\right) = \\
&= 3\sum_{j=1}^{i-1} \left(2^{i-1}\mathcal{O}(1)\right) + 2\sum_{j=1}^{i-1} \left(2^{i-1}\mathcal{O}(2^{j\cdot\log_2 3-1})\right) = \\
&= 3(i-1)2^{i-1}\mathcal{O}(1) + 2^i \sum_{j=1}^{i-1} \mathcal{O}\left(3\left(\frac{3}{2}\right)^{j-1}\right) \leqslant \\
&\leqslant 3(i-1)2^{i-1}\mathcal{O}(1) + \mathcal{O}(2^{i\cdot\log_2 3}),
\end{aligned}$$

или

$$C_i \approx \mathcal{O}(i2^i) + \mathcal{O}(2^{i\cdot\log_2 3}). \tag{45}$$

Используя (43), (44) и (45), упростим (42):

$$\begin{aligned}
C^s &= \sum_{i=1}^{M} \left(C_i + ADD_{GF\left(2^{2^M}\right)} \right) = \\
&= \sum_{i=1}^{M} \left(\mathcal{O}(i2^i) + \mathcal{O}(2^{i\cdot\log_2 3}) + \mathcal{O}(2^M) \right) \leqslant \\
&\leqslant \mathcal{O}(k^2 2^M) + \mathcal{O}(2^{M\cdot\log_2 3}).
\end{aligned} \tag{46}$$

Так как $M = \log_2 n$, то из (46) получим

$$C^s \approx \mathcal{O}\left((\log n)^2 n\right) + \mathcal{O}(n^{\log_2 3}). \tag{47}$$

Используя только алгоритм Карацубы–Офмана, выполним оценку сложности вычисления элемента синдрома

$$C_K^s = (n-1)\mathcal{O}(n^{\log 3}) + (n-1)O(n) \approx \mathcal{O}(n^{\log 3+1}) + \mathcal{O}(n^2). \tag{48}$$

Сравнивая (47) и (48), замечаем, что предложенный метод имеет значительную меньшую вычислительную сложность, чем метод, основанный только на использовании алгоритма Карацубы–Офмана.

6. Реализация блока вычисления синдрома

Опишем реализацию блока вычисления синдрома для случая $(8, 4, 5)$. Согласно (23), слабый самоортогональный базис для этого случая будет основываться на базисе b_4 и иметь вид

$$b_8 = (1, f_1, f_2, f_2f_1, f_3, f_3f_1, f_3f_2, f_3f_2f_1). \tag{49}$$

Для того чтобы выполнить умножение в слабом самоортогональном базисе, необходимо определить матрицы перехода из исходного (стандартного) базиса в слабый самоортогональный (14). Степени Фробениуса ($\alpha^{[i]} = \alpha^{2^i}$) векторов этого базиса составляют элементы проверочной матрицы. Подобно (26), для кода, имеющего скорость $1/2$, проверочная матрица будет иметь вид

$$H = \begin{Vmatrix} 1 & f_1 & f_2 & f_2f_1 & f_3 & f_3f_1 & f_3f_2 & f_3f_2f_1 \\ 1 & f_1^{[1]} & f_2^{[1]} & (f_2f_1)^{[1]} & f_3^{[1]} & (f_3f_1)^{[1]} & (f_3f_2)^{[1]} & (f_3f_2f_1)^{[1]} \\ 1 & f_1^{[2]} & f_2^{[2]} & (f_2f_1)^{[2]} & f_3^{[2]} & (f_3f_1)^{[2]} & (f_3f_2)^{[2]} & (f_3f_2f_1)^{[2]} \\ 1 & f_1^{[3]} & f_2^{[3]} & (f_2f_1)^{[3]} & f_3^{[3]} & (f_3f_1)^{[3]} & (f_3f_2)^{[3]} & (f_3f_2f_1)^{[3]} \end{Vmatrix}. \tag{50}$$

Далее подробно будет рассматриваться схема вычисления элемента синдрома s_0, аналогичная (36),

$$s_0 = y_0 + f_1y_1 + f_2y_2 + f_1f_2y_3 + f_3y_4 + f_3f_1y_5 + f_3f_2y_6 + f_3f_2f_1y_7. \tag{51}$$

Схемы вычисления других элементов синдрома строятся аналогичным образом и имеют схожий с (51) вид. Для них изменятся только базисные векторы f_i, согласно формуле

$$f_i^{2^{[i]}+1} = 1, f_i \in GF(2^{[i]}). \tag{52}$$

Заметим, что числа вида $F_n = 2^{[n]} + 1$ называются числами Ферма [15]. Из (52) следуют формулы (23) и (24).

6.1. Построение матрицы перехода

Для преобразования векторов в слабый самоортогональный базис и обратно необходимо найти матрицу перехода. Пусть кодовые векторы представлены в стандартном базисе. Чтобы построить матрицу перехода из слабо самоортогонального базиса в исходный стандартный базис, надо определить, каким векторам в новом базисе соответствуют векторы матрицы H (50). Примем, что $f_{\min}$ соответствует вектор, равный примитивному элементу поля в степени $n/2$ (для рассматриваемого базиса (49) это будет вектор α^4), такой, что единичный вектор преобразуется в единичный:

$$00000001_s \to 000000001_w. \tag{53}$$

С одной стороны, остальные векторы, кроме единичного (53), представлены как комбинация произведений векторов f_i ($f_i \neq f_{\min}$). Назовём его подмножество $\mathcal{A}$. С другой стороны, оставшиеся векторы являются комбинациями из подмножества $\mathcal{A}$, умноженные на

вектор $f_{\min}$. Назовём это подмножество $\mathcal{B}$. Можно показать, что подпространству, образуемому каждым вектором f_i ($f_i \neq f_{\min}$), можно поставить во взаимно однозначное соответствие подпространство, образуемое векторами y_i с такими же условиями (52), но из поля $GF(2^{n/2})$. Тогда базисные векторы из подмножества $\mathcal{A}$ будут поставлены в соответствие векторам y_i, записанным в младших $n/2$ разрядах, а векторы из подмножества $\mathcal{B}$ будут поставлены в соответствие тем же векторам y_i, но записанным в старших $n/2$ разрядах. Операции умножения преобразуются так, чтобы в слабо самоортогональном базисе выполнялись следующие правила [10]:

$$f_i \cdot f_j \in \mathcal{A}; \tag{54}$$

$$(f_{mult} \cdot f_i) \cdot f_j \in \mathcal{B}; \tag{55}$$

$$f_{\min} \cdot f_{\min} = \alpha + \beta \cdot f_{\min}, \beta \cdot f_{\min} \in \mathcal{B}, \alpha \in \mathcal{A}. \tag{56}$$

6.2. Пример вычисления матрицы перехода

Пусть имеется поле $GF(2^8)$ или $GF(2^{[3]})$ и элементы в поле задаются неприводимым многочленом [11]

$$p_8(x) = x^8 + x^4 + x^3 + x^2 + 1. \tag{57}$$

Для определения подмножеств нам необходимо выбрать полином из поля $GF(2^{[3-1]})$:

$$p_4(x) = x^4 + x + 1. \tag{58}$$

Примитивный элемент обозначим как δ. Используя условия (52), вычислим значения векторов f_i, на основе которых строятся следующие базисные векторы:

$$f_1 = \delta^{([8]-1)/(2^{[1-1]}+1)} = \delta^{255/3} = \delta^{85} = 11010110; \tag{59}$$

$$f_2 = \delta^{([8]-1)/(2^{[2-1]}+1)} = \delta^{255/5} = \delta^{51} = 00001010; \tag{60}$$

$$f_3 = \delta^{([8]-1)/(2^{[3-1]}+1)} = \delta^{255/17} = \delta^{15} = 00100110. \tag{61}$$

Отметим, что $f_{\min} = f_3$. Элемент в новом базисе, соответствующий элементу ν в старом базисе, обозначим $\tilde{\nu}$. Тогда, поскольку в новом базисе образ элемента $f_{\min}$ есть примитивный элемент поля степени $N/2$,

$$\tilde{f}_3 = \tilde{f}_{\min} = 00010000. \tag{62}$$

Вектору f_1 поставим в соответствие вектор из поля, образованного неприводимым многочленом (57), и запишем значение получившегося младшего элемента в младшие разряды образа в новом базисе. Найдём образы векторов f_1 и f_2 в базисе размерности $N/2$. Для нахождения элементов поля воспользуемся полиномом (58):

$$g_1^3 = 1 \Rightarrow g_1 = \gamma^{15/3} = \gamma^5 = 0110, \tag{63}$$

$$g_2^5 = 1 \Rightarrow g_2 = \gamma^{15/5} = \gamma^3 = 1000, \tag{64}$$

$$g_1 \cdot g_2 = \gamma^8 = 0101, \tag{65}$$

где γ – примитивный элемент поля $GF(2^{[1]})$. Согласно принятому соглашению, образы векторов f_1 и f_2 будут равны

$$\tilde{f}_1 = 0000, g_1 = 00000100, \tag{66}$$

$$\tilde{f}_2 = 0000, g_2 = 00001000, \tag{67}$$

$$\tilde{f}_1 \cdot \tilde{f}_2 = 0000, g_1 \cdot g_2 = 00000101. \tag{68}$$

Здесь мы определили образы подмножества $\mathcal{A}$. Поскольку элементы подмножества $\mathcal{B}$ есть элементы подмножества $\mathcal{A}$, умноженные на вектор $f_{\min}$ то, если воспользоваться правилом

умножения (55), можно легко найти образы базисных векторов этого подмножества в новом базисе:

$$f_{\min} \cdot f_1 \to g_{\min} \cdot g_1 = 01100000, \tag{69}$$
$$f_{\min} \cdot f_2 \to g_{\min} \cdot g_2 = 10000000, \tag{70}$$
$$f_{\min} \cdot f_1 \cdot f_2 \to g_{\min} \cdot g_1 \cdot g_2 = 01010000. \tag{71}$$

Учитывая правила (53), (62), (66), (67), (68), (69), (70) и (71), можем записать в виде матрицы набор векторов в стандартном базисе F_{std}. В этом случае исходным векторам будет соответствовать столбец:

$$F_{std} = \left\| f_{\min} f_2 f_1 \quad f_{\min} f_2 \quad f_{\min} f_1 \quad f_{\min} \quad f_2 f_1 \quad f_2 \quad f_1 \quad 1 \right\|, \tag{72}$$
$$F_{std} = \left\| \alpha^{151} \quad \alpha^{100} \quad \alpha^{66} \quad \alpha^{15} \quad \alpha^{136} \quad \alpha^{51} \quad \alpha^{85} \quad \alpha^{0} \right\| \tag{73}$$

или

$$F_{std} = \begin{Vmatrix} \mathbf{1} & 0 & \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{1} & 0 \\ \mathbf{0} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{0} & 0 \\ \mathbf{1} & 0 & \mathbf{1} & 1 & \mathbf{0} & 0 & \mathbf{0} & 0 \\ \mathbf{0} & 1 & \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{1} & 0 \\ \mathbf{1} & 0 & \mathbf{0} & 0 & \mathbf{1} & 1 & \mathbf{0} & 0 \\ \mathbf{0} & 0 & \mathbf{0} & 1 & \mathbf{1} & 0 & \mathbf{1} & 0 \\ \mathbf{1} & 0 & \mathbf{0} & 1 & \mathbf{1} & 1 & \mathbf{1} & 0 \\ \mathbf{0} & 1 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{0} & 1 \end{Vmatrix}. \tag{74}$$

Соответствующий им набор векторов в слабом самоортогональном базисе F_{weak} есть

$$F_{weak} = \left\| \tilde{f}_{\min} \tilde{f}_2 \tilde{f}_1 \quad \tilde{f}_{\min} \tilde{f}_2 \quad \tilde{f}_{\min} \tilde{f}_1 \quad \tilde{f}_{\min} \quad \tilde{f}_2 \tilde{f}_1 \quad \tilde{f}_2 \quad \tilde{f}_1 \quad 1 \right\| \tag{75}$$

или

$$F_{weak} = \begin{Vmatrix} \mathbf{0} & 1 & \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{0} & 0 \\ \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{0} & 0 & \mathbf{0} & 0 \\ \mathbf{0} & 0 & \mathbf{1} & 0 & \mathbf{0} & 0 & \mathbf{0} & 0 \\ \mathbf{1} & 0 & \mathbf{0} & 1 & \mathbf{0} & 0 & \mathbf{0} & 0 \\ \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{0} & 1 & \mathbf{0} & 0 \\ \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{1} & 0 \\ \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{1} & 0 \\ \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{1} & 0 & \mathbf{0} & 1 \end{Vmatrix}. \tag{76}$$

Используя операции «сложение столбцов», преобразуем (76) в единичную матрицу. Применим такие же операции к (74). После этих операций преобразованная матрица (74) будет равна матрице перехода из слабо самоортогонального базиса в первоначальный, то есть стандартный базис. В рассматриваемом случае

$$T_{std} = \begin{Vmatrix} \mathbf{0} & 1 & \mathbf{1} & 0 & \mathbf{0} & 0 & \mathbf{1} & 0 \\ \mathbf{1} & 0 & \mathbf{0} & 0 & \mathbf{0} & 1 & \mathbf{0} & 0 \\ \mathbf{1} & 0 & \mathbf{0} & 1 & \mathbf{0} & 0 & \mathbf{0} & 0 \\ \mathbf{0} & 0 & \mathbf{1} & 0 & \mathbf{0} & 0 & \mathbf{1} & 0 \\ \mathbf{0} & 1 & \mathbf{1} & 1 & \mathbf{0} & 1 & \mathbf{1} & 0 \\ \mathbf{0} & 1 & \mathbf{1} & 1 & \mathbf{0} & 1 & \mathbf{0} & 0 \\ \mathbf{0} & 0 & \mathbf{0} & 1 & \mathbf{1} & 1 & \mathbf{0} & 0 \\ \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{0} & 0 & \mathbf{0} & 1 \end{Vmatrix}. \tag{77}$$

Матрица перехода из стандартного базиса в слабый самоортогональный определяется как обратная к матрице T_{std}:

$$T_{weak} = \begin{Vmatrix} \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \\ \mathbf{1} & 0 & \mathbf{0} & 1 & \mathbf{0} & 0 & \mathbf{0} & 0 \\ \mathbf{1} & 1 & \mathbf{1} & 1 & \mathbf{0} & 1 & \mathbf{0} & 0 \\ \mathbf{1} & 0 & \mathbf{0} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \\ \mathbf{0} & 1 & \mathbf{1} & 0 & \mathbf{0} & 0 & \mathbf{1} & 0 \\ \mathbf{1} & 1 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \\ \mathbf{1} & 1 & \mathbf{1} & 0 & \mathbf{0} & 1 & \mathbf{0} & 0 \\ \mathbf{0} & 1 & \mathbf{0} & 1 & \mathbf{1} & 1 & \mathbf{1} & 1 \end{Vmatrix}. \tag{78}$$

Для данных значений $T_{std}T_{weak} = I$.

6.3. Блок-схема устройства вычисления синдрома

В рассматриваемом случае для реализации математических операций в блоке вычисления синдрома рангового кода потребуются следующие элементы:

- блок преобразования векторов из стандартного базиса в слабый самоортогональный;
- блок умножения в поле размерности $n/2$ на вектор f_1;
- блок умножения в поле размерности $n/2$ на вектор f_2;
- блок умножения в поле размерности $n/2$ на вектор β (вектор α принимаем равным *единичному*).

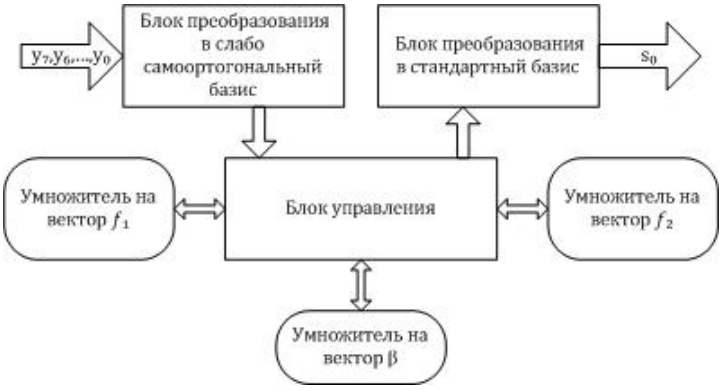

Рис. 1. Структурная схема устройства вычисления синдрома

На рис. 1 показано взаимодействие между элементами устройства. Данные поступают на вход системы в порядке убывания индекса $(y_7, y_6, \ldots, y_0)$. Наиболее частое обращение происходит к умножителю на f_1 (8 обращений). К блоку, реализующему умножение на f_2, происходит 4 обращения. Меньше всего обращений (1 раз) выполняется по отношению к умножителю на β. На выход системы после определённого количества тактов выдаётся значение элемента синдрома s_i. Для скорости, равной 1/2, потребуется 4 блока ($n = 8$). Заметим, что поскольку в проверочной матрице последующая строчка получается возведением в определённую степень Фробениуса первой строчки, то имеется возможность оптимизировать вычисления за счёт учёта результатов, найденных в соседних блоках. Результаты в соседних блоках получены в процессе определения других элементов синдрома.

6.4. Моделирование и синтез

Для моделирования был использован пакет Active-HDL 7.2 Student Edition от компании Aldec [12]. В качестве языка описания аппаратной части использовался VHDL [13]. Проект создавался для ПЛИС XC3S700AN-4FGG484C [14]. Синтез (synthesize) и алгоритм создания описания для упаковки в память ПЛИС (tranlation, mapping, placement и routing) выполнялись с настройками по умолчанию. Результаты приведены в табл. 1.

Таблица 1

Результаты моделирования и синтеза

Тип схемы	Задержка, такты	Элементарные блоки (slice), шт. (%)	4-портовая память (LUT), шт. (%)	Макс. рабочая частота, МГц
Предложенный алгоритм	23	1355 (23)	1549 (13)	173
Алгоритм на базе умножителей Карацубы–Офмана	106	751 (12)	1124 (9)	173

Отметим, что задержка в тактах получена при моделировании, однако точно такая же задержка будет и в физическом устройстве. Блок на основе предложенного алгоритма оказывается примерно в 4 раза быстрее. Из-за большего количества элементов (несколько типов умножителей и блоков преобразования векторов в требуемый базис) в физическом устройстве предложенный блок занимает больше ресурсов микросхемы.

7. Заключение

Предложенный способ позволяет для случая декодирования ранговых кодов $(n/2)$ сократить время вычисления синдрома по сравнению с методом, основанным на умножителях Карацубы–Офмана. Проверочное моделирование и синтез показал, что для реализации нового метода требуется в 1,5 раза больше ресурсов кристалла. Вычисление синдрома, основанного на слабо самоортогональном базисе, предпочтительно для систем, в которых время декодирования рангового кода является существенным ограничением.

Литература

1. *Ahlswede R., Cai N.* [*et al.*]. Network Information Flow // IEEE Transactions on Information Theory. — V. 46, N 4. — 2000. — P. 1204–1216.
2. *Loidreau P.* A Welch-Berlekamp like algorithm for decoding Gabidulin codes // Proc. 4^{th} Int. Workshop on Coding and Cryptography. — 2005. — P. 36–45.
3. *Gabidulin E. M., Pilipchuk N. I.* Symmetric matrices and codes correcting rank errors beyond the $\lfloor (d-1)/2 \rfloor$ bound // Discrete Applied Mathematics. — 2006. — N 154. — P. 305–312.
4. *Карацуба А. А., Офман Ю. П.* Умножение многозначных чисел на автоматах // Доклады Академии наук СССР. — 1962. — Т. 145, № 2. — С. 293–294.
5. *Афанасьев В. Б., Габидулин Э. М.* Кодирование в радиоэлектронике. — М. : Радио и связь, 1986. — 176 с.
6. *Bednara M., Gathen J., Grabbe C., Shokrollahi J., Teich J.* FPGA designs of parallel high performance $GF(2^{333})$ multipliers // Proc. of the 2003 International Symposium on IEEE «Circuits and Systems (ISCAS03)». — 2003. — V. II. — P. 268–271.
7. *Schönhage A., Strassen V.* Schnele Multiplikation großer Zahlen // Computing. — 1971. — I. 7. — P. 281–292.

8. *Coronado Garcia L.C.* Can Schönhage multiplication speed up the RSA encryption or decryption? // University of Technology, Darmstadt. — 2005.

9. *Габидулин Э.М.* Теория кодов с максимальным ранговым расстоянием // Проблемы передачи информации. — 1985. — Т. 21, № 1. — С. 1–12.

10. *Gabidulin E.M., Sysoev I.Y.* Rank codes using weak self-orthogonal bases // Proc. 2010 IEEE Region 8 International Conference on Computational Technologies in Electrical and Electronics Engineering. — 2010. — P. 70–71.

11. *Лидл Р., Нидеррайтер Г.* Конечные поля. Т. 2. — М. : Мир, 1988. — 822 с.

12. *Aldec inc.* Active-HDL Manual / aldec.com: web-site of Aldec inc. 2012. URL: http://aldec.com/resources/manuals/Active-HDL/index.htm (дата обращения 22.01.2012).

13. *Xilinx inc.* ISE 13.1 Documentation / xilinx.com: web-site of Xilinx inc. 2012 URL: http://www.xilinx.com/support/documentation/dt_ise13-1.htm (дата обращения 22.01.2012).

14. *Xilinx inc.* Spartan-3AN FPGA User Guides / xilinx.com: web-site of Xilinx inc. 2012 URL: http://www.xilinx.com/support/documentation/spartan-3an_user_guides.htm (дата обращения 22.01.2012).

15. *Luigi Morelli.* Distributed search for Fermat Number Divisors / fermatsearch.org: http://www.fermatsearch.org/index.html (дата обращения 04.05.2013).

16. *Silva D., Kschischang F.R., Koetter R.* A Rank-Metric Approach to Error Control in Random Network Coding // IEEE Transactions on Information Theory. — 2008. — V. 54, I. 9. — P. 3951–3967.

17. *Сагалович Ю.Л.* Введение в алгебраические коды: учебное пособие. — М. : МФТИ, 2007. — 262 с.

Поступила в редакцию 06.05.2013.