Секция: Технические науки

Сайлаукызы Жулдыз

докторант кафедры «Вычислительная техника»

Евразийского национального университета

имени Л. Н. Гумилева

г. Астана, Республика Казахстан

ОБЕСПЕЧЕНИЕ НАДЁЖНОГО ХРАНЕНИЯ ДАННЫХ ВО FLASH-ПАМЯТИ

Введение. Flash-память благодаря компактности, дешевизне, механической прочности, большому объёму, скорости работы и низкому энергопотреблению, широко используется в цифровых портативных устройствах и носителях информации. Повышение плотности записи, достигаемое за счет уменьшающегося физического размера ячейки наряду с возрастающим количеством используемых состояний, приводит к снижению надежности хранения данных, что требует использования помехоустойчивого кодирования.

Обеспечение надёжного хранения данных в больших объёмах флеш-памяти также становится всё более актуальной. При быстром увеличении объёма таких блоков хранения данных и значительным росте требований к целостности хранимой в них информации методы решения этой проблемы до не давнего времени практически не развивались. Но эта задача становится всё более актуальной, поскольку даже редкие ошибки при передаче критически важной информации во многих случаях вообще недопустимы.

К настоящему времени предложено довольно много вариантов помехоустойчивого кодирования, предназначенного для многоуровневой NAND flash-памяти: коды БЧХ, коды Рида - Соломона, низкоплотностные коды (LDPC), решетчатые коды [1, с. 241-249; 2, с. 429-439; 3, с. 900-908].

Очень важно, что требуемая вероятность ошибки на бит в данном случае должна быть очень малой, порядка $10^{-12} \div 10^{-15}$.

Коды с низкой плотностью проверок на четность (LDPC) показывают хорошие практические результаты при использовании вероятностного декодирования [4, с. 144].

LDPC-код задается своей проверочной матрицей H, обладающей свойством разреженности, т.е. строки и столбцы матрицы содержат мало ненулевых позиций по сравнению с ее размерностью.

LDPC код представляется графом Таннера [5, с. 721-727; 6, с. 766-775] (рис.1). Узлы в графе Таннера называются информационными и проверочными узлами, которые мы обозначим как C и F, соответственно. Можно отметить, что граф будет содержать f проверочных узлов F, по одному для каждого проверочного уравнения, и C информационных узлов c, по одному для каждого символа кодового слова.

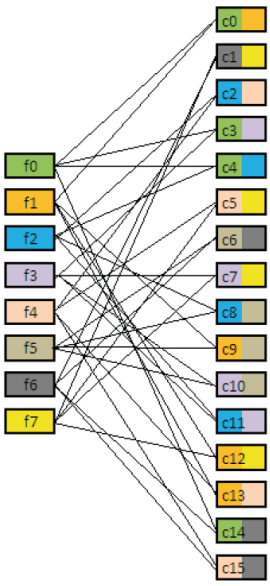

Рис. 1. Граф Таннера для линейного блокового кода (8, 16)

С целью анализа эффективности реализации рассмотрим основные алгоритмы декодирования LDPC кодов. LDPC код – это частный случай линейного блочного кода. В этом кодировании мы будем использовать две матрицы: одна является генераторной матрицей G на кодировщике и

матрица проверки на четность H на декодере. Кодирование линейного блочного (n, k) – кода задается генераторной матрицей $\boldsymbol{G}_{n,k}$, которая определяется как массив размером $k \times n$.

Генерация кодового слова $\boldsymbol{U}$ в матричной записи равна произведению:

$$C = uG \tag{1}$$

(8, 16) - коде генераторная матрица (G) и проверочная матрица (H) имеют следующий вид:

$$G_{16,8}=\begin{pmatrix}1000000000000111\\0100000000001101\\0010000001110100\\0001000001100111\\0000100001110111\\0000010000001100\\0000001001000101\\0000000101101000\end{pmatrix} \qquad H_{16,8}=\begin{pmatrix}0000000010000000\\0011101101000000\\0011100100100000\\0010100000010000\\0100010100001000\\1111111000000100\\1001100000000010\\1101101000000001\end{pmatrix}$$

Рассмотрим кодирование информационных символов [u=01001110]

1) Перемножив информационные символы [u] и матрицу G, получаем кодовое слово: 01001110 00110011
2) Полученное кодовое слово должно удовлетворять условию.

$$U*H=0 \tag{2}$$

Декодирование. Bit-flipping алгоритм является жестким методом декодирования сообщений для LDPC-кодов. В канал связи с шумами поступает сообщение с кодовым словом y=01001**0**10 00110011, если проверить его на принадлежность к кодовым словам то результат отличен от нуля, а следовательно вектор «y» не является кодовым словом, а это значит, что во время передачи возникли ошибки. Производится инициализация и на бит-узлы поступают значения с входа декодера. На первом шаге проверочные узлы считывают значения с бит-узлов и формируют сообщения для второго шага. На втором шаге проверочные узлы отправляют значения на бит-узлы для сравнения.

Таблица 1

Алгоритм инверсии битов (bit-flipping algorithm)

Узел	y_i получил	Сообщения от контрольных узлов						Решение
c_0	0	$f_5 \to 1$	$f_6 \to 0$	$f_7 \to 0$				0
c_1	1	$f_4 \to 0$	$f_5 \to 0$	$f_7 \to 1$				1
c_2	0	$f_1 \to 0$	$f_2 \to 0$	$f_3 \to 0$	$f_5 \to 1$			0
c_3	0	$f_1 \to 0$	$f_2 \to 0$	$f_5 \to 1$	$f_6 \to 0$	$f_7 \to 0$		0
c_4	1	$f_1 \to 1$	$f_2 \to 1$	$f_3 \to 1$	$f_5 \to 0$	$f_6 \to 1$	$f_7 \to 1$	1
c_5	**0**	$f_4 \to 1$	$f_5 \to 1$					**1**
c_6	1	$f_1 \to 1$	$f_5 \to 0$	$f_7 \to 1$				1
c_7	0	$f_1 \to 0$	$f_2 \to 0$	$f_4 \to 1$				0
c_8	0	$f_0 \to 0$						0
c_9	0	$f_1 \to 0$						0
c_{10}	1	$f_2 \to 1$						1
c_{11}	1	$f_3 \to 1$						1
c_{12}	0	$f_4 \to 0$						0
c_{13}	0	$f_5 \to 0$						0
c_{14}	1	$f_6 \to 1$						1
c_{15}	1	$f_7 \to 1$						1

Последний шаг запускает проверку нового сообщения на соответствие проверочным уравнениям матрицы *H*. Для 0-7 узла.

$f_0 = y_8 = 0$
$f_1 = y_2 \oplus y_3 \oplus y_4 \oplus y_6 \oplus y_7 \oplus y_9 = 0$
$f_2 = y_2 \oplus y_3 \oplus y_4 \oplus y_7 \oplus y_{10} = 0$
$f_3 = y_2 \oplus y_4 \oplus y_{11} = 0$
$f_4 = y_1 \oplus y_5 \oplus y_7 \oplus y_{12} = 0$
$f_5 = y_0 \oplus y_1 \oplus y_2 \oplus y_3 \oplus y_4 \oplus y_5 \oplus y_6 \oplus y_{13} = 0 = 0$
$f_6 = y_0 \oplus y_3 \oplus y_4 \oplus y_{14} = 0$
$f_7 = y_0 \oplus y_1 \oplus y_3 \oplus y_4 \oplus y_6 \oplus y_{15} = 0$

В результате проверки не было найдено несоответствий, поэтому алгоритм останавливается и возвращает. 01001110 00110011.

Реализация программного модуля кодирования и декодирования на основе математических моделей и алгоритмов. Языком описания был выбран Verilog HDL, наиболее часто используемый в проектировании, верификации и реализации аналоговых, цифровых и смешанных

электронных систем на различных уровнях абстракции. Была выбрана среда проектирования Vivado, поддерживающая разработку программного обеспечения.

Структурная схема разработанной системы, включающей кодер и декодер приведена на рисунке 2. Кодер включает в себя: сдвигающий регистр и логические элементы сложения по модулю два, осуществляющие формирование проверочных символов.

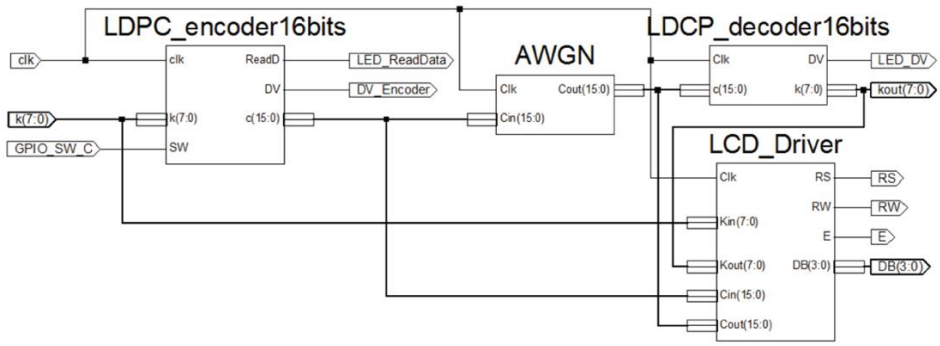

Рис. 2. Структурная схема аппаратного блока включающей кодер и декодер

Таким образом, применение корректирующих кодов позволяет обнаружить, и по возможности исправить искаженные данные, причем ранее записанные данные и их структура останутся в неприкосновенности.

Литература

1. Sun F., Devarajan S., Rose K., Zhang T. Design of On-Chip Error Correction Systems for Multilevel NOR and NAND Flash Memories / IET Circuits Devices and Systems. - 2007. - Vol. 1. - N 3. - P. 241–249.
2. Dong G., Xie N., Zhang T. On the Use of Soft-Decision Error-Correction Codes in NAND Flash Memory / IEEE Transactions on Circuits and Systems I: Regular Papers. - 2011. - Vol. 58. - N 2. - P. 429–439.
3. Kurkoski B. M. Coded Modulation Using Lattices and Reed-Solomon Codes, with Applications to Flash Memories. IEEE Transactions on Selected Areas in Communications, 2014, vol. 32, no. 5, pp. 900–908.
4. Галлагер Р. Дж. Коды с малой плотностью проверок на четность. М.:

Мир, 1966. - 144 с.

5. L. Wei "Several properties of short LDPC codes," IEEE Trans. Commun., vol. 52, pp. 721-727, May 2004.
6. H. Zhong and T. Zhang «Block-LDPC: A practical LDPC coding system design approach» IEEE Trans. Circuits Syst. I, vol. 52, no. 4, pp. 766-775, Apr. 2005.