ФУНДАМЕНТАЛЬНЫЕ ОСНОВЫ ПРОБЛЕМ НАДЕЖНОСТИ И КАЧЕСТВА

FUNDAMENTALS OF RELIABILITY AND QUALITY ISSUES

УДК 004.421.6
doi: 10.21685/2307-4205-2024-2-1

МЕТОДЫ ПОВЫШЕНИЯ БЫСТРОДЕЙСТВИЯ ДЕКОДИРОВАНИЯ ПОТОКОВЫХ ДАННЫХ НА ОСНОВЕ КОДА РИДА – СОЛОМОНА

М. Ю. Звездочкин[1], В. В. Миронов[2]

[1] Филиал акционерного общества «Ракетно-космический центр "Прогресс" – особое конструкторское бюро "Спектр"», Рязань, Россия
[2] Рязанский государственный радиотехнический университет имени В. Ф. Уткина, Рязань, Россия
[2] Научно-исследовательский институт обработки космических изображений «Фотон», Рязань, Россия
[1] mikhail@zvyozdochkin.ru, [2] mironov1vv@mail.ru

Аннотация. *Актуальность и цели.* Декодирование информации с использованием помехоустойчивых кодов Рида – Соломона является весьма ресурсоемкой задачей в части организации вычислений. Особенно эта проблема актуальна при декодировании измерительной информации и видеопотоков на лету, в режиме времени, близком к реальному. *Материалы и методы.* Рассматриваются два метода оптимальной организации декодирования Рида – Соломона. Первый метод – организация распараллеливания, декодирование в несколько вычислительных потоков. Второй метод – искусственное снижение количества ошибок, исправляемых кодом Рида – Соломона, он позволяет уменьшить объем вычислений. Рассматривается экспериментальная проверка указанных методов. *Результаты.* Показано, что рассмотренные методы позволяют более эффективно организовать вычисления, даются рекомендации по их применению, как раздельному, так и совместному. *Выводы.* Методы можно применять в системе обработки видеопотоков. В зависимости от режима работы система обработки должна позволять применять как оба метода, так и один выбранный.

Ключевые слова: видеосигнал, потоковое видео, искажения цифрового видеосигнала, помехоустойчивое кодирование

Для цитирования: Звездочкин М. Ю., Миронов В. В. Методы повышения быстродействия декодирования потоковых данных на основе кода Рида – Соломона // Надежность и качество сложных систем. 2024. № 2. С. 5–14. doi: 10.21685/2307-4205-2024-2-1

METHODS FOR IMPROVING THE PERFORMANCE OF DECODING STREAMING DATA BASED ON THE REED – SOLOMON CODE

M.Yu. Zvyozdochkin[1], V.V. Mironov[2]

[1] Branch of the Progress Rocket and Space Center Joint Stock Company – Specter Special Design Bureau, Ryazan, Russia
[2] Ryazan State Radio Engineering University named after V.F. Utkin, Ryazan, Russia
[2] Photon Research Institute for Space Image Processing, Ryazan, Russia
[1] mikhail@zvyozdochkin.ru, [2] mironov1vv@mail.ru

Abstract. *Background.* Decoding information using noise-tolerant Reed–Solomon codes is a very resource-intensive task in part of calculation imlpementation. This problem is especially relevant when video streams or telemetry decoding on the fly, in a time mode close to real time. *Matherials and methods.* The article discusses two methods of optimal organization of Reed – Solomon decoding. First method is a decoding into several computational streams, some processor threads using. Second method is an artificially reducing the number of errors corrected by the Reed–Solomon code to reduce calculation time. *Results.* The experimental verification of these methods is considered. Recommendations for their application, both separate and joint, are given. *Conclusions.* Methods can be used in video streaming processing systems. Processing system must allow use either both method or one selected by operator.

Keywords: video signal, video streaming, digital video signal distortion, error control codes

For citation: Zvyozdochkin M.Yu., Mironov V.V. Methods for improving the performance of decoding streaming data based on the Reed – Solomon code. *Nadezhnost' i kachestvo slozhnykh sistem = Reliability and quality of complex systems.* 2024;(2):5–14. (In Russ.). doi: 10.21685/2307-4205-2024-2-1

Введение

Коды Рида – Соломона (англ. *Reed – Solomon codes*, в честь сотрудников Массачусетского технологического института Ирвинга Рида и Густава Соломона, опубликовавших в 1960 г. этот способ кодирования данных), напомним, – это недвоичные циклические коды, позволяющие исправлять ошибки в блоках данных. При этом элементами кодового вектора являются не биты, а группы битов (или блоки битов).

Коды Рида – Соломона широко применяются для помехоустойчивого кодирования (кодирования, исправляющего ошибки) в системах хранения и передачи данных (см., например, [1]).

Коды Рида – Соломона подразделяются на систематические и несистематические коды [2]. При систематическом кодировании изменения в исходную информацию не вносятся. Закодированный сигнал чередуется блоками контрольных символов.

При несистематическом кодировании исходные слова подвергаются умножению информационного слова на порожденный полином, закодированный сигнал не содержит исходного в явном виде. Преимуществом систематического кодирования является тот факт, что при отсутствии искажений сигнала декодирование не требуется.

В данной статье рассматривается код, в котором применяется систематическое кодирование.

С практической точки зрения весьма важным и даже особым случаем является кодирование и декодирование информации, что называется, «на лету», в режиме времени, достаточно близком к реальному, например, при передаче измерительной и видеоинформации с летательных или космических аппаратов. При этом предъявляются повышенные требования к эффективности алгоритмов кодирования и декодирования информации. Особенно это относится к алгоритму декодирования, поскольку декодирование данных по Риду – Соломону является чрезвычайно ресурсоемкой задачей, в значительно большей степени, чем кодирование.

Для достижения приемлемого быстродействия в предыдущие годы был предложен ряд известных алгоритмов (алгоритм Берлекэмпа – Месси, алгоритм Форни), однако объем вычислений все равно остается значительным. Особенно это критично для задач обработки закодированной информации в реальном времени, когда необходимо исключить риск потери принимаемых данных.

В данной статье для достижения практически оптимального быстродействия декодирования данных будут рассмотрены два метода, позволяющие ускорить процедуру декодирования «на лету» и снизить риск потери данных при декодировании.

Постановка проблемы

Итак, в работе рассматривается задача декодирования «на лету» сигнала блочной структуры с применением разновидности кода Рида – Соломона с 16-разрядными информационными словами. Это более сложный код, чем описанный в стандарте ECMA-130, где количество бит на символ $m = 8$[1].

Информация принимается в виде циклов с частотой следования Fc циклов в секунду.

Каждый цикл включает в себя Nb блоков, каждый из которых подвергается кодированию Рида – Соломона. Кодирование – систематическое. Это означает, что информационный блок передается в неизменном виде и сопровождается контрольными словами. Таким образом, при отсутствии ошибок информационный блок может не подвергаться декодированию.

[1] Standard ECMA-130. Data Interchange on Readonly 120 mm Optical Data Disks (CD-ROM). 2nd edition (June 1996). URL: https://www.ecma– international.org/publications/standards/Ecnia-130.htm

Параметры кода: m = 16 – количество бит на символ (степень *RS*-полинома), n – количество символов в кодовом блоке, k – количество информационных символов в кодовом блоке, $r = n - k$ – количество контрольных символов в кодовом блоке (теоретически позволяет исправить до $r/2$ ошибочных информационных слов). Код является укороченным, т.е. длина кода меньше длины конечного поля Галуа [3].

Перед началом сеанса обработки строится поле Галуа, которое используется для операций над полиномами. С точки зрения программной реализации оно представляет собой массив 16-разрядных чисел.

При декодировании для каждого блока проверяется наличие ошибок, для чего находится так называемый синдром:

$$S_j = C\left(\alpha^j\right),\ j = 1...r. \tag{1}$$

Если все компоненты синдрома равны нулю, код не содержит ошибок. Это быстрая операция, не затрудняющая дальнейшую обработку и передачу видеоинформации:

$$\forall j \in [1, r] S_j = 0. \tag{2}$$

Если же ошибки найдены, т.е. условие (2) не выполняется, производится поиск ошибок по алгоритму Берлекэмпа – Месси [1]. По этому алгоритму производится вычисление полинома $\Lambda(x)$ локаторов ошибок. Если полином не найден, ошибки исправить невозможно. Если найден, то ищутся корни уравнения $\Lambda(x) = 0$. Количество корней должно быть равным степени полинома, иначе ошибки неисправимы. Из корней полинома выделяются локаторы ошибок. Вычисляется полином величин ошибок, производная полинома локаторов, предполагаемый полином локаторов и находится исправленный полином кадров.

Если при максимально возможном количестве ошибок поиск не увенчался успехом, блок считается неисправимым.

Проблема описанного алгоритма заключается в том, что время декодирования достаточно велико и сопоставимо с временем, отведенным на весь цикл обработки.

Эксперимент с замером времени декодирования 40 000 блоков, принадлежащих одному закодированному видеопотоку, показывает, что по мере увеличения количества ошибок в одном блоке время, необходимое для вычислений, возрастает по зависимости, близкой к линейной (рис. 1).

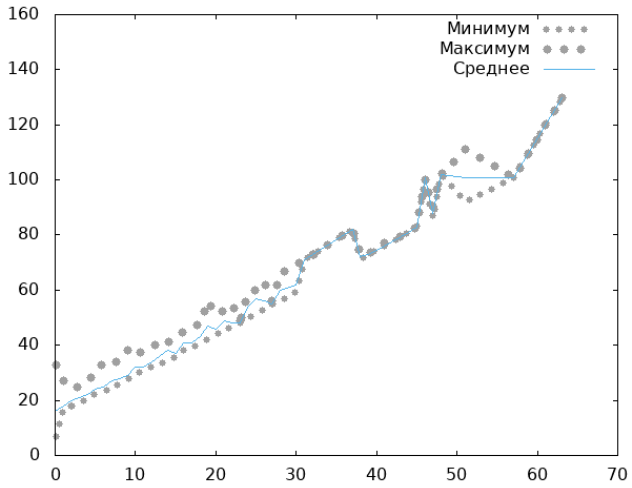

Рис. 1. Зависимость времени вычислений от ожидаемого числа ошибок в блоке

По оси абсцисс находится ожидаемое число ошибок в блоке, по оси ординат – измеренное время (в миллисекундах) декодирования одного блока для указанного числа ошибок (минимальное, усредненное, максимальное). Заметим, что разброс значений при количестве ошибок, превышающем 30, обусловлен малым количеством таких ошибок.

Эксперимент показал, что часть вычислений приходится на неисправимые блоки, т.е. вычисления проводятся впустую. Следует отметить также, что большая часть исправимых ошибок является

одиночными, а их число в блоке относительно невелико – в пределах 10–20 ошибок на блок при размере блока 10 кБайт.

В частности, при $N_b = 8$ и $F_c = 4$ время декодирования цикла из 8 сильно зашумленных блоков даже на современных вычислительных средствах может превысить 0,25 с (период следования циклов) и таким образом привести к потере следующего обрабатываемого цикла. В ряде случаев потери циклов могут оказаться более критичными, чем ошибки, исправленные кодом Рида – Соломона.

В статье рассмотрены два способа преодоления описанной проблемы:

– декодирование в несколько вычислительных потоков;

– ограничение требований к эффективности декодирования (искусственное снижение количества ошибок, исправляемых кодом Рида – Соломона).

По мнению авторов, возможны и другие способы решения указанной проблемы. К примеру, на основе работы [1] можно построить аппаратный декодер, использующий регистры сдвига. Рассмотрение этого способа выходит за рамки нашей статьи.

Декодирование в несколько потоков

Распараллеливание вычислений операции декодирования на несколько вычислительных потоков – наиболее очевидное средство борьбы с нехваткой вычислительных ресурсов. Его применению в описываемом случае способствует то, что каждый из 8 блоков в цикле кодируется независимо, т.е. задача хорошо распараллеливается.

После получения цикла закодированной видеоинформации декодирование каждого блока помещается в отдельный поток (thread). Управляющий поток ожидает завершения декодирования всех 8 блоков, после чего объединяет результирующие массивы данных и передает объединенный массив декодированных блоков на дальнейшую обработку. Выдача потокам заданий на декодирование и определение, какие потоки завершили вычисление, производится обменом сообщениями между классами.

Расчеты и эксперимент показывают, что снижение времени декодирования по описанной схеме прямо пропорционально количеству ядер процессора (с поправкой на накладные расходы на организацию многопоточности) при условии, что оно не менее количества блоков в цикле. Идеальной является ситуация, при которой количество ядер процессора по крайней мере на единицу больше, чем количество блоков в цикле.

Замечание. Необходимо обратить внимание, что (рабочее) поле Галуа является общим массивом для всех объектов-декодеров. Для уменьшения времени подготовки вычислений (исключение дублирующих вычислений) и потребляемого объема оперативной памяти все 8 объектов-декодеров могут использовать единое поле Галуа, которое вполне можно построить перед началом сеанса обработки.

При однопоточном декодировании суммарное время декодирования T_{dc} всех N_b блоков цикла определяется как

$$T_{dc} = \sum_{i=1}^{N_b} T_{dbi}, \tag{3}$$

здесь T_{dbi} – время декодирования каждого блока, зависящее в первую очередь от количества ошибок в блоке.

При многопоточном декодировании время T_{dc} оценивается менее однозначно:

$$T_{db\max} < T_{dc} \leqslant \sum_{i=1}^{N_b} T_{dbi}, \tag{4}$$

здесь $T_{db\max}$ – максимальное время декодирования одного из входящих в цикл блоков.

В работе авторы исходят из предположения (и в следующем разделе будет проведена экспериментальная проверка этой гипотезы), что если декодирование каждого блока выполняется в отдельном потоке и количество блоков N_b превышает или равно количеству ядер процессора N_c, то время вычисления уменьшается почти прямо пропорционально увеличению значения N_c:

$$T_{dc} \approx \frac{\sum_{i=1}^{N_b} T_{dbi}}{N_c - 1} \text{ при } N_b \geq N_c. \tag{5}$$

Вычитание одного ядра учитывает работу других процессов, в том числе системных. Работа других ресурсоемких процессов (например, антивирусных программ) может нарушить эту оценку.

Дальнейшее увеличение значения N_c не должно влиять на быстродействие декодирования, и время декодирования цикла стремится к наибольшему времени декодирования одного из входящих в его состав блоков, с поправкой на накладные расходы для организации многопоточности:

$$T_{dc} \approx T_{db\max} \text{ при } N_c > N_b . \quad (6)$$

Замечание. Недостатком декодирования в несколько потоков является усложнение отладки процедуры декодирования по сравнению с линейной вычислительной последовательностью. Кроме того, возможно проявление проблем многозадачности, таких как взаимные блокировки [4].

Также необходимо учитывать, что распараллеливание вычислений уменьшает время на декодирование, но увеличивает суммарную нагрузку на процессор, время которого занимают и другие процессы, в том числе системные. При низком объеме вычислений многопоточное декодирование может оказаться даже менее эффективным, чем однопоточное (см., к примеру, [5]), а большое количество одновременно выполняемых потоков может привести к падению пропускной способности [6].

Таким образом, целесообразность (в естественном ее понимании) данного способа должна быть проверена на конкретной, заданной аппаратно-программной конфигурации еще до проведения вычислений.

Вызывает также интерес использование рассмотренных подходов к проблеме повышения эффективности сеансов связи космических аппаратов дистанционного зондирования Земли с наземными пунктами приема информации в современных высокоскоростных радио- и телесистемах [7].

Экспериментальная проверка многопоточного декодирования

Экспериментальная проверка предложенного метода производилась на ПЭВМ, оснащенной 4-ядерным процессором Intel Core 2 Quad Q8200 под управлением 64-разрядной операционной системы Debian GNU/Linux.

Параметры кода:

– $F_c = 4$;

– $N_b = 8$;

– кодирование – систематическое;

– $m = 16$;

– $r = 128$ (теоретически позволяет исправить до $r/2 = 64$ слов).

На вход декодирующей программы подавался сигнал блочной структуры, длительностью почти 20 минут. Для каждого цикла замерялось суммарное время декодирования T_{dc} всех N_b блоков отдельно для однопоточного и многопоточного режима.

Результаты проверки приведены в табл. 1 (все интервалы измеряются в миллисекундах).

Таблица 1

Время декодирования и число потоков

T_{dc}	Один поток	Восемь потоков
Мин	54	16
Макс	432	184
Средний	133	49
СКО	19	18

Как видно из анализа, быстродействие увеличивается приблизительно в $N_c - 1$ раз, где N_c – количество ядер процессора. В рассматриваемом случае, при $N_c = 4$, быстродействие увеличивается почти в 3 раза. Таким образом, для максимально быстрого декодирования 8 блоков, если количество ошибок в канале связи на блок приближается к максимальному теоретически исправимому ($r/2$), целесообразно (при естественном определении цели) применить 10-ядерный процессор.

Анализ распределения значений T_{dc} показывает, что плотность распределения в общем случае имеет три пика:

– первый пик соответствует случаям, когда ни в одном из блоков цикла ошибок не обнаружено. В этом случае все расчетное время уходит на проверку отсутствия ошибок. Этот пик близок ко второму, а в многопоточном режиме они практически сливаются;

– второй пик – самый крупный и самый размытый – соответствует случаям, когда в одном или нескольких блоках цикла обнаружено небольшое количество исправимых ошибок;

– третий пик соответствует наличию хотя бы в одном блоке цикла большого количества ошибок, как правило, неисправимых. Этот пик самый низкий, и в количественном выражении им можно было бы пренебречь. Однако именно он порождает недопустимые потери быстродействия и сопутствующие им потери, о которых уже говорилось в статье.

На рис. 2 третий пик искусственно выделен (для наглядности). По оси абсцисс приведено время декодирования цикла в миллисекундах.

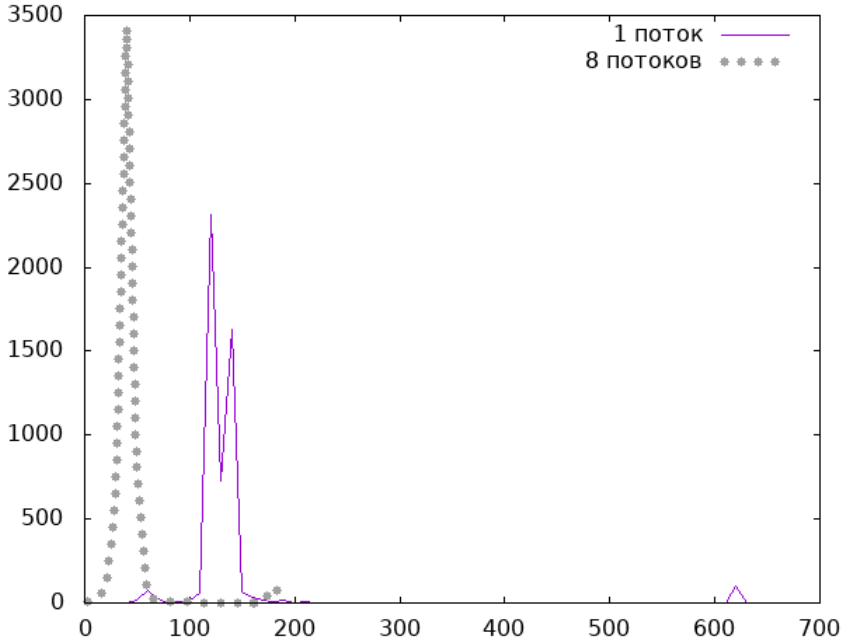

Рис. 2. Распределение времени декодирования

В данном эксперименте измерения проводились как для однопоточного, так и многопоточного режимов. В однопоточном режиме первый пик находится в области 50–80 мс, второй (основной) – в области 120–150 мс. Наиболее существенно, что третий пик размыт и находится в области от 270 мс и далее, а отдельные случаи попадают на область 620–640 мс. Это недопустимая ситуация, ибо она приводит к потерям данных.

В многопоточном режиме первый и второй пик сливаются и перемещаются в область 20–70 мс. Верхняя граница третьего пика находится в районе 180–200 мс, что уже позволяет штатно завершить обработку цикла.

Отмеченные пики имели бы более четко выраженный вид, если вместо T_{dc} замерялось время декодирования каждого блока T_{db}, которое явно зависит от числа ошибок в блоке. Но в многопоточном режиме связь между T_{db} и T_{dc} неоднозначна (на само значение T_{db} многопоточность практически не влияет), а для общей оценки быстродействия практическое значение имеет именно T_{dc}.

Ограничение требований к эффективности декодирования

Описанный далее метод борьбы с недостатком вычислительных ресурсов основан на особенности построения декодера Рида – Соломона, уже отмеченной в начале статьи. Дело в том, что применение данного кода позволяет исправлять до $r/2$ ошибочных информационных слов на блок. Вместе с тем наибольшие расходы вычислительных ресурсов приходятся на блоки с большим количеством ошибок, а часть этих расходов приходится на попытки исправления неисправимых блоков. Как показывает опыт, исправимое количество ошибок, приближающееся к числу $r/2$, встречается достаточно редко ($r = n?k$ – количество контрольных символов в кодовом блоке).

Исключить заведомо непроизводительные расходы можно, заранее ограничив сверху количество предполагаемых ошибок $N_{e,\max}$:

$$1 \le N_{e,\max} \le \frac{r}{2}. \quad (7)$$

Если оцениваемое количество ошибок блока на какой-либо итерации превышает число $N_{e,\max}$, то декодирование блока прекращается и в качестве выходного массива возвращается неисправленный блок данных. Поскольку код Рида – Соломона, применяемый в данном случае, является систематическим, из таких блоков можно корректно извлечь нужную информацию (частично зашумленную).

Если же число $N_{e,\max}$ не превышено, то вычисления продолжаются до тех пор, пока все ошибочные слова не будут исправлены. Учитывая зависимость объема вычислений от количества ошибок, оптимизация по времени получается весьма существенной.

Максимальным теоретическим ограничителем величины $N_{e,\max}$ является число $r/2$, равное половине исправимых кодом Рида – Соломона сопутствующих контрольных символов. Выбор же слишком низкого значения $N_{e,\max}$, как показывают теоретические соображения и опыты, снижает эффективность помехоустойчивого кодирования.

В работе предлагается выбирать величину $N_{e,\max}$ на основе априорной информации, полученной из предыдущих сеансов обработки (разумеется, при наличии таковых).

Как показывает опыт, ошибки при передаче закодированной видеоинформации классифицируются в две принципиально различные группы:

– одиночные ошибки – когда количество ошибок сравнительно невелико и они распределены по отдельным информационным словам в блоке;

– групповые ошибки – когда искажены продолжительные фрагменты блока, блок целиком или даже несколько подряд следующих блоков.

Исправимые ошибки, как правило, являются одиночными, и абсолютное их количество не превышает 20–40 ошибок на один блок. Если оценка количества ошибок предсказывает большее значение, то с вероятностью, близкой к максимальной 1, блок является неисправимым.

Ограничение требований к эффективности может быть рекомендовано для декодирования информации, что называется, «на лету», т.е. в режиме времени, близком к реальному. При отложенной обработке преимущества этого способа утрачиваются, и следует проверять все возможные значения количества исправимых ошибок, вплоть до $r/2$.

Для экспериментальной проверки, приводимой далее, выбрано значение $N_{e,\max}$, равное 10. Правильный выбор значения $N_{e,\max}$ является, как можно заметить, ключевым фактором, определяющим корректность описанного способа оптимизации. Поэтому в работе предлагается проводить анализ и уточнять его значение по мере накопления информации.

Экспериментальная проверка ограничения требований к эффективности

Экспериментальная проверка метода ограничения требований к эффективности проводилась с такими же параметрами кода Рида – Соломона, как и предыдущая проверка. В первом случае поток декодировался без ограничений (что эквивалентно $N_{e,\max} = 64$), во втором случае было введено ограничение $N_{e,\max} = 10$.

Результаты проверки приведены в табл. 2 (все интервалы, как и в предыдущем случае, приведены в миллисекундах).

Таблица 2

Время декодирования и максимальное число ошибок

T_{dc}	$N_{e,\max} = 64$	$N_{e,\max} = 10$
Мин	53	53
Макс	637	228
Средний	133	131
СКО	21	15

Результаты проверочного теста показывают, что минимальный и средний замеренный интервал практически не изменились. Это объясняется тем, что они получаются при отсутствии ошибок и низком количестве ошибок соответственно. Вместе с тем максимальный интервал значительно отличается, поскольку он в первом случае соответствует попытке исправить неисправимый блок. Другими словами, первый и второй пики распределения при введении $N_{e,\max}$ не изменяют свое местонахождение, а третий пик сдвигается в область 200–220 мс. Как и на предыдущем графике, значение крайнего правого пика на рис. 3 искусственно выделено.

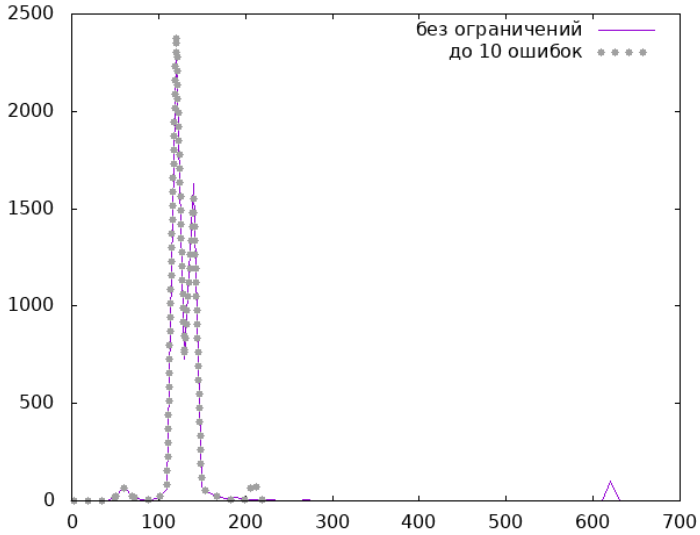

Рис. 3. Результаты проверочного теста

Обработка цикла в течение 637 мс означает превышение максимально допустимого времени декодирования и, возможно, полную потерю следующего обрабатываемого цикла. В то же время 228 мс – допустимый интервал, меньше $1/F_c$ = 0, 25с.

Замечание. Разумеется, значение $N_{e,\max} = 10$ – это низкое значение, не раскрывающее потенциал примененной разновидности кода Рида – Соломона. Рекомендуется увеличить $N_{e,\max}$, одновременно применив описанное выше многопоточное декодирование.

Заключение

Итак, в работе рассмотрены два метода повышения быстродействия декодирования потоковых данных на основе кода Рида – Соломона. Оба метода борьбы с вычислительной сложностью имеют свои ограничения.

Декодирование в несколько потоков ограничено количеством ядер процессора, накладными расходами на переключение потоков и требованием над выполнением других процессов (например, антивирусных программ). Кроме того, максимальный эффект от этого метода достигается только на многоядерных процессорах (для $N_b = 8$ рекомендуется использовать процессоры с $N_c > 8$), что увеличивает стоимость применяемых технических средств.

Ограничение требований к эффективности декодирования применимо только при условии сохранения эффективности кода Рида – Соломона.

Исходя из этого, при построении программной реализации декодера авторы рекомендуют:

– при декодировании «на лету» – сочетать оба рассмотренных подхода, при этом количество предполагаемых ошибок $N_{e,\max}$ должно настраиваться оператором и выбираться на основе априорной информации, полученной по итогам анализа предыдущих сеансов обработки;

– при отложенной обработке – применять только декодирование в несколько потоков;

– с целью сохранения универсальности программного обеспечения программная реализация декодера должна предусматривать возможность настройки включения и отключения многопоточного декодирования со стороны оператора в зависимости от применяемых технических средств и общесистемного программного обеспечения;

– вводить в программную реализацию декодера средства самодиагностики, позволяющие оценить быстродействие декодирования в однопоточном и многопоточном режимах [8, 9].

Список литературы

1. Блейхут Р. Теория и практика кодов, контролирующих ошибки : пер. с англ. М. : Мир, 1986. 576 с.
2. Золотарев В. В., Овечкин Г. В. Помехоустойчивое кодирование. Методы и алгоритмы. М. : Горячая Линия – Телеком, 2004.
3. Майстренко В. А., Привалов Д. Д., Седунов Д. П. Сравнительный анализ временных характеристик декодирования полноразмерных и укороченных кодов Рида – Соломона // Техника радиосвязи. 2018. № 1. С. 103–111.
4. Nemirovsky M., Tullsen D. M. Multithreading Architecture. Morgan and Claypool Publishers, 2013.

5. Кадомский А. А., Захаров В. А. Эффективность многопоточных приложений // Научный журнал. 2016. № 7. С. 26–28.
6. Мильковский С. И., Сорокин А. А., Королев С. П. [и др.]. Оценка производительности гибридного вычислительного кластера па базе процессоров IBM POWER8 // Программирование. 2019. № 6. С. 30–39.
7. Миронов В. В., Кащеев А. А. Повышение точности оценки длительности сеансов связи космических аппаратов дистанционного зондирования Земли с наземными пунктами приема информации для надежных высокоскоростных радиолиний // Современные проблемы дистанционного зондирования Земли из космоса. 2018. Т. 15, № 6. С. 235–244.
8. Самаров В. В. Тестирование динамического выполнения кода python программ при проведении этапа сертификационных испытаний (разработки) в системе сертификации Минобороны России // Надежность и качество сложных систем. 2023. № 2. С. 96–103.
9. Иванов А. И., Иванов А. П., Горбунов К. А. Нейросетевое преобразование биометрии в код аутентификации: дополнение энтропии хэмминга энтропией корреляционных связей между разрядами // Надежность и качество сложных систем. 2023. № 1. С. 91–98.

References

1. Bleykhut P. *Teoriya i praktika kodov, kontroliruyushchikh oshibki: per. s angl. = Theory and practice of error-controlling codes : trans. from English*. Moscow: Mir, 1986:576. (In Russ.)
2. Zolotarev V.V., Ovechkin G.V. *Pomekhoustoychivoe kodirovanie. Metody i algoritmy = Noise-resistant coding. Methods and algorithms*. Moscow: Goryachaya Liniya – Telekom, 2004. (In Russ.)
3. Maystrenko V.A., Privalov D.D., Sedunov D.P. Comparative analysis of the time characteristics of decoding full–size and shortened Reed - Solomon codes. *Tekhnika radiosvyazi = Radio communication technology*. 2018;(1): 103–111. (In Russ.)
4. Nemirovsky M., Tullsen D.M. *Multithreading Architecture*. Morgan and Claypool Publishers, 2013.
5. Kadomskiy A.A., Zakharov V.A. Efficiency of multithreaded applications. *Nauchnyy zhurnal = Scientific Journal*. 2016;(7):26–28. (In Russ.)
6. Mil'kovskiy S.I., Sorokin A.A., Korolev S.P. et al. Performance evaluation of a hybrid computing cluster based on IBM POWER8 processors. *Programmirovanie = Programming*. 2019;(6):30–39. (In Russ.)
7. Mironov V.V., Kashcheev A.A. Improving the accuracy of estimating the duration of communication sessions of Earth remote sensing spacecraft with ground-based information reception points for reliable high-speed radio lines. *Sovremennye problemy distantsionnogo zondirovaniya Zemli iz kosmosa = Modern problems of remote sensing of the Earth from space*. 2018;15(6):235–244. (In Russ.)
8. Samarov V.V. Testing the dynamic execution of python program code during the certification testing (development) stage in the certification system of the Ministry of Defense of Russia. *Nadezhnost' i kachestvo slozhnykh system = Reliability and quality of complex systems*. 2023;(2):96–103. (In Russ.)
9. Ivanov A.I., Ivanov A.P., Gorbunov K.A. Neural network transformation of biometrics into authentication code: addition of hamming entropy by entropy of correlations between bits. *Nadezhnost' i kachestvo slozhnykh system = Reliability and quality of complex systems.* 2023;(1):91–98. (In Russ.)

Информация об авторах / Information about the authors

Михаил Юрьевич Звездочкин
ведущий конструктор,
Филиал акционерного общества
«Ракетно-космический центр "Прогресс" –
особое конструкторское бюро "Спектр"»
(Россия, г. Рязань, ул. Гагарина, 59а)
E-mail: mikhail@zvyozdochkin.ru

Валентин Васильевич Миронов
доктор физико-математических наук, профессор,
профессор кафедры высшей математики,
Рязанский государственный радиотехнический
университет имени В. Ф. Уткина,
директор лаборатории системного анализа,
ведущий научный сотрудник,
Научно-исследовательский институт обработки
космических изображений «Фотон»
(Россия, г. Рязань, ул. Гагарина, 59/1)
E-mail: mironov1vv@mail.ru

Mikhail Yu. Zvyozdochkin
Leading designer,
Branch of the Progress Rocket and Space Center Joint
Stock Company – Specter Special Design Bureau
(59a Gagarina street, Ryazan, Russia)

Valentin V. Mironov
Doctor of physical and mathematical sciences, professor,
professor of the sub-department of higher mathematics,
Ryazan State Radio Engineering University
named after V.F. Utkin,
director of the laboratory of system analysis,
leading researcher,
Photon Research Institute for Space Image Processing
(59/1 Gagarina street, Ryazan, Russia)

Авторы заявляют об отсутствии конфликта интересов /
The authors declare no conflicts of interests.

Поступила в редакцию/Received 12.03.2024
Поступила после рецензирования/Revised 20.03.2024
Принята к публикации/Accepted 10.04.2024