шифр « Красота и здоровье »

Научная работа на конкурс за направлением:

Информатика и кибернетика

на тему: « Разработка метода сжатия сообщений на основе многозначных биномиальных чисел »

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Еще вчера казалось, что диск размером в один гигабайт — это так много, что даже неясно, чем его заполнить, и уж конечно, каждый про себя думал: был бы у меня гигабайт памяти, я бы перестал «жадничать» и сжимать свою информацию какими-то архиваторами. Но, видимо, мир так устроен, что «свято место пусто не бывает», и как только у нас появляется лишний гигабайт — тут же находится чем его заполнить. Да и сами программы, как известно, становятся все более объемными. Так что, видимо, с терабайтами и экзабайтами будет то же самое.

Поэтому, как бы ни росли объемы памяти диска, упаковывать информацию, похоже, не перестанут. Наоборот, по мере того как «места в компьютере» становится все больше, число новых архиваторов увеличивается, при этом их разработчики не просто соревнуются в удобстве интерфейсов, а в первую очередь стремятся упаковать информацию все плотнее и плотнее.

Однако очевидно, что процесс этот не бесконечен. Где лежит этот предел, какие архиваторы доступны сегодня, по каким параметрам они конкурируют между собой, где найти свежий архиватор.

В этой работе вашему вниманию предлагается еще один метод сжатия информации, который основан на нумерационном кодироание и реализован цифровых элементах.

1 ОСНОВНЫЕ МЕТОДЫ СЖАТИЯ

Полагаю, что многие из вас неоднократно пользовались архиваторм. Целью архивации файлов является экономия места на жестком или гибком магнитном диске. Кому не приходилось время от времени задумываться над тем, войдет ли данный файл на дискету? Существует большое число программ-архиваторов, имеются и специальные системные программные средства типа Stacker или Doublespace и т.д., решающие эту проблему.

Первые теоретические разработки в области сжатия информации относятся к концу 40-х годов. В конце семидесятых появились работы Шеннона, Фано и Хафмана. К этому времени относится и создание алгоритма FGK (Faller, Gallager, Knuth), где используется идея "сродства", а получатель и отправитель динамически меняют дерево.

Пропускная способность каналов связи более дорогостоящий ресурс, чем дисковое пространство, по этой причине сжатие данных до или во время их передачи еще более актуально. Здесь целью сжатия информации является экономия пропускной способности и в конечном итоге ее увеличение. Все известные алгоритмы сжатия сводятся к шифрованию входной информации, а принимающая сторона выполняет дешифровку принятых данных.

Существуют методы, которые предполагают некоторые потери исходных данных, другие алгоритмы позволяют преобразовать информацию без потерь. Сжатие с потерями используется при передаче звуковой или графической информации, при этом учитывается несовершенство органов слуха и зрения, которые не замечают некоторого ухудшения качества, связанного с этими потерями.

Сжатие информации без потерь осуществляется статистическим кодированием или на основе предварительно созданного словаря. Статистические алгоритмы (напр., схема кодирования Хафмана) присваивают каждому входному символу определенный код. При этом наиболее часто используемому символу присваивается наиболее короткий код, а наиболее редкому - более длинный. Таблицы кодирования создаются заранее и имеют

ограниченный размер. Этот алгоритм обеспечивает наибольшее быстродействие и наименьшие задержки. Для получения высоких коэффициентов сжатия статистический метод требует больших объемов памяти[6].

Сжатие основано на устранении избыточности, содержащейся в исходных данных. Простейшим примером избыточности является повторение в тексте фрагментов (например, слов естественного или машинного языка). Подобная избыточность обычно устраняется заменой повторяющейся последовательности ссылкой на уже закодированный фрагмент с указанием его длины. Другой вид избыточности связан с тем, что некоторые значения в сжимаемых данных встречаются чаще других. Сокращение объёма данных достигается за счёт замены часто встречающихся данных короткими кодовыми словами, а редких — длинными (энтропийное кодирование). Сжатие данных, не обладающих свойством избыточности (например, случайный сигнал или шум, зашифрованные сообщения), принципиально невозможно без потерь.

В основе любого способа сжатия лежит модель источника данных, или, точнее, модель избыточности. Иными словами, для сжатия данных используются некоторые априорные сведения о том, какого рода данные сжимаются. Не обладая такими сведениями об источнике, невозможно сделать никаких предположений о преобразовании, которое позволило бы уменьшить объём сообщения. Модель избыточности может быть статической, неизменной для всего сжимаемого сообщения, либо строиться или параметризоваться на этапе сжатия (и восстановления). Методы, позволяющие на основе входных данных изменять модель избыточности информации, называются адаптивными. Неадаптивными являются обычно узкоспециализированные алгоритмы, применяемые для работы с данными, обладающими хорошо определёнными и неизменными характеристиками. Подавляющая часть достаточно универсальных алгоритмов являются в той или иной мере адаптивными.

Все методы сжатия данных делятся на два основных класса:

Сжатие без потерь

Сжатие с потерями

При использовании сжатия без потерь возможно полное восстановление исходных данных, сжатие с потерями позволяет восстановить данные с искажениями, обычно несущественными с точки зрения дальнейшего использования восстановленных данных. Сжатие без потерь обычно используется для передачи и хранения текстовых данных, компьютерных программ, реже — для сокращения объёма аудио- и видеоданных, цифровых фотографий и т. п., в случаях, когда искажения недопустимы или нежелательны. Сжатие с потерями, обладающее значительно большей, чем сжатие без потерь, эффективностью, обычно применяется для сокращения объёма аудио- и видеоданных и цифровых фотографий в тех случаях, когда такое сокращение является приоритетным, а полное соответствие исходных и восстановленных данных не требуется.

Коэффициент сжатия — основная характеристика алгоритма сжатия. Она определяется как отношение объёма исходных несжатых данных к объёму сжатых, то есть:

$$k = So/Sc \qquad (1.1)$$

где k — коэффициент сжатия, So — объём исходных данных, а Sc — объём сжатых. Таким образом, чем выше коэффициент сжатия, тем алгоритм эффективнее. Следует отметить:

если $k = 1$, то алгоритм не производит сжатия, то есть выходное сообщение оказывается по объёму равным входному;

если $k < 1$, то алгоритм порождает сообщение большего размера, нежели несжатое, то есть, совершает «вредную» работу.

Ситуация с $k < 1$ вполне возможна при сжатии. Принципиально невозможно получить алгоритм сжатия без потерь, который при любых данных образовывал бы на выходе данные меньшей или равной длины. Обоснование этого факта заключается в том, что поскольку число различных сообщений длиной n бит составляет ровно 2n, число различных сообщений с длиной меньшей или равной n (при наличии хотя бы одного сообщения меньшей длины)

будет меньше 2n. Это значит, что невозможно однозначно сопоставить все исходные сообщения сжатым: либо некоторые исходные сообщения не будут иметь сжатого представления, либо нескольким исходным сообщениям будет соответствовать одно и то же сжатое, а значит их нельзя отличить.

Коэффициент сжатия может быть как постоянным (некоторые алгоритмы сжатия звука, изображения и т. п., например A-закон, μ-закон, ADPCM, усечённое блочное кодирование), так и переменным. Во втором случае он может быть определён либо для каждого конкретного сообщения, либо оценён по некоторым критериям:

средний (обычно по некоторому тестовому набору данных);

максимальный (случай наилучшего сжатия);

минимальный (случай наихудшего сжатия);

или каким-либо другим. Коэффициент сжатия с потерями при этом сильно зависит от допустимой погрешности сжатия или качества, которое обычно выступает как параметр алгоритма. В общем случае постоянный коэффициент сжатия способны обеспечить только методы сжатия данных с потерями.

Основным критерием различия между алгоритмами сжатия является описанное выше наличие или отсутствие потерь. В общем случае алгоритмы сжатия без потерь универсальны в том смысле, что их применение безусловно возможно для данных любого типа, в то время как возможность применения сжатия с потерями должна быть обоснована. Для некоторых типов данных искажения не допустимы в принципе. В их числе

символические данные, изменение которых неминуемо приводит к изменению их семантики: программы и их исходные тексты, двоичные массивы и т. п.;

жизненно важные данные, изменения в которых могут привести к критическим ошибкам: например, получаемые с медицинской измерительной аппаратуры или контрольных приборов летательных, космических аппаратов и т. п.;

многократно подвергаемые сжатию и восстановлению промежуточные данные при многоэтапной обработке графических, звуковых и видеоданных.

Различные алгоритмы могут требовать различного количества ресурсов вычислительной системы, на которых они реализованы:

оперативной памяти (под промежуточные данные);

постоянной памяти (под код программы и константы);

процессорного времени.

В целом, эти требования зависят от сложности и «интеллектуальности» алгоритма. Общая тенденция такова: чем эффективнее и универсальнее алгоритм, тем большие требования к вычислительным ресурсам он предъявляет. Тем не менее, в специфических случаях простые и компактные алгоритмы могут работать не хуже сложных и универсальных. Системные требования определяют их потребительские качества: чем менее требователен алгоритм, тем на более простой, а следовательно, компактной, надёжной и дешёвой системе он может быть реализован.

Так как алгоритмы сжатия и восстановления работают в паре, имеет значение соотношение системных требований к ним. Нередко можно усложнив один алгоритм значительно упростить другой. Таким образом, возможны три варианта:

Алгоритм сжатия требует больших вычислительных ресурсов, нежели алгоритм восстановления.

Это наиболее распространённое соотношение, характерное для случаев, когда однократно сжатые данные будут использоваться многократно. В качестве примера можно привести цифровые аудио- и видеопроигрыватели.

Алгоритмы сжатия и восстановления требуют приблизительно равных вычислительных ресурсов.

Наиболее приемлемый вариант для линий связи, когда сжатие и восстановление происходит однократно на двух её концах (например, в цифровой телефонии).

Алгоритм сжатия существенно менее требователен, чем алгоритм восстановления.

Такая ситуация характерна для случаев, когда процедура сжатия реализуется простым, часто портативным устройством, для которого объём доступных ресурсов весьма критичен, например, космический аппарат или большая распределённая сеть датчиков. Это могут быть также данные, распаковка которых требуется в очень малом проценте случаев, например запись камер видеонаблюдения [6].

Разработано большое количество разнообразных методов, их модификаций и подвидов для сжатия данных. Современные архиваторы, как правило, одновременно используют несколько методов одновременно. Можно выделить некоторые основные [3].

Running - Это самый простой из методов упаковки информации . Предположите что Вы имеете строку текста, и в конце строки стоит 40 пробелов. налицо явная избыточность имеющейся информации. Проблема сжатия этой строки решается очень просто - эти 40 пробелов (40 байт) сжимаются в 3 байта с помощью упаковки их по методу повторяющихся символов (running). Первый байт, стоящий вместо 40 пробелов в сжатой строке, фактически будет являються пробелом (последовательность была из пробелов) . Второй байт - специальный байт "флажка" который указывает что мы должны развернуть предыдущий в строке байт в последовательность при восстановлении строки . Третий байт - байт счета (в нашем случае это будет 40). Как Вы сами можете видеть, достаточно чтобы любой раз, когда мы имеем последовательность из более 3-х одинаковых символов, заменять их выше описанной последовательностью, чтобы на выходе получить блок информации меньший по размеру, но допускающий восстановление информации в исходном виде.

Оставляя все сказанное выше истинным , добавлю лишь то, что в данном методе основной проблемой является выбор того самого байта "флажка", так как в реальных блоках информации как правило используются все 256 вариантов байта и нет возможности иметь 257 вариант - "флажок".

LZW

LZW - История этого алгоритма начинается с опубликования в мае 1977 г. Дж. Зивом (J. Ziv) и А. Лемпелем (A. Lempel) статьи в журнале "

Информационные теории " под названием " IEEE Trans ". В последствии этот алгоритм был доработан Терри А. Велчем (Terry A. Welch) и в окончательном варианте отражен в статье " IEEE Compute " в июне 1984 . В этой статье описывались подробности алгоритма и некоторые общие проблемы с которыми можно столкнуться при его реализации. Позже этот алгоритм получил название - LZW (Lempel - Ziv - Welch).

Алгоритм LZW представляет собой алгоритм кодирования последовательностей неодинаковых символов. Возьмем для примера строку " Объект TSortedCollection порожден от TCollection.". Анализируя эту строку мы можем видеть, что слово "Collection" повторяется дважды. В этом слове 10 символов - 80 бит. И если мы сможем заменить это слово в выходном файле, во втором его включении, на ссылку на первое включение, то получим сжатие информации. Если рассматривать входной блок информации размером не более 64К и ограничится длинной кодируемой строки в 256 символов, то учитывая байт "флаг" получим, что строка из 80 бит заменяется 8+16+8 = 32 бита. Алгоритм LZW как-бы "обучается" в процессе сжатия файла. Если существуют повторяющиеся строки в файле , то они будут закодированны в таблицу. Очевидным преимуществом алгоритма является то, что нет необходимости включать таблицу кодировки в сжатый файл. Другой важной особенностью является то, что сжатие по алгоритму LZW является однопроходной операцией в противоположность алгоритму Хаффмана (Huffman), которому требуется два прохода.

Метод кодирования Хаффмана (энтропийный метод)

Сначала кажется что создание файла меньших размеров из исходного без кодировки последовательностей или исключения повтора байтов будет невозможной задачей. Но давайте мы заставим себя сделать несколько умственных усилий и понять алгоритм Хаффмана (Huffman). Потеряв не так много времени мы приобретем знания и дополнительное место на дисках.

Сжимая файл по алгоритму Хаффмана первое что мы должны сделать - это необходимо прочитать файл полностью и подсчитать сколько раз встречается каждый символ из расширенного набора ASCII. Если мы будем учитывать все

256 символов, то для нас не будет разницы в сжатии текстового и EXE файла. После подсчета частоты вхождения каждого символа, необходимо просмотреть таблицу кодов ASCII и сформировать мнимую компоновку между кодами по убыванию. То есть не меняя местонахождение каждого символа из таблицы в памяти отсортировать таблицу ссылок на них по убыванию. Каждую ссылку из последней таблицы назовем "узлом". В дальнейшем (в дереве) мы будем позже размещать указатели которые будут указывает на этот "узел". Просчитав дерево, мы можем кодировать файл . Мы должны всегда начинать из корня. Кодируя первый символ . Мы прослеживаем вверх по дереву все повороты ветвей и если мы делаем левый поворот, то запоминаем 0-й бит, и аналогично 1-й бит для правого поворота. Каждый символ изначально представлялся 8-ю битами (один байт), и так как мы уменьшили число битов необходимых для представления каждого символа, мы следовательно уменьшили размер выходного файла [4].

Метод Шеннона-Фано

Алгоритм Шеннона — Фано — один из первых алгоритмов сжатия, который впервые сформулировали американские учёные Шеннон и Фано (англ. Fano). Данный метод сжатия имеет большое сходство с алгоритмом Хаффмана, который появился на несколько лет позже. Коды Шеннона — Фано префиксные, то есть никакое кодовое слово не является префиксом любого другого. Это свойство позволяет однозначно декодировать любую последовательность кодовых слов.

Подобно алгоритму Хаффмана, алгоритм Шеннона — Фано использует избыточность сообщения, заключённую в неоднородном распределении частот символов его (первичного) алфавита, то есть заменяет коды более частых символов короткими двоичными последовательностями, а коды более редких символов — более длинными двоичными последовательностями.

Алгоритм был независимо друг от друга разработан Шенноном (публикация «Математическая теория связи», 1948 год) и, позже, Фано (опубликовано как технический отчёт).

Основные этапы

Символы первичного алфавита выписывают в порядке убывания вероятностей.

Символы полученного алфавита делят на две части, суммарные вероятности символов которых максимально близки друг другу.

В префиксном коде для первой части алфавита присваивается двоичная цифра «0», второй части — «1».

Полученные части рекурсивно делятся и их частям назначаются соответствующие двоичные цифры в префиксном коде.

Когда размер подалфавита становится равен нулю или единице, то дальнейшего удлинения префиксного кода для соответствующих ему символов первичного алфавита не происходит, таким образом, алгоритм присваивает различным символам префиксные коды разной длины. На шаге деления алфавита существует неоднозначность, так как разность суммарных вероятностей может быть одинакова для двух вариантов разделения (учитывая, что все символы первичного алфавита имеют вероятность больше нуля).

Код Шеннона — Фано строится с помощью дерева. Построение этого дерева начинается от корня. Всё множество кодируемых элементов соответствует корню дерева (вершине первого уровня). Оно разбивается на два подмножества с примерно одинаковыми суммарными вероятностями. Эти подмножества соответствуют двум вершинам второго уровня, которые соединяются с корнем. Далее каждое из этих подмножеств разбивается на два подмножества с примерно одинаковыми суммарными вероятностями. Им соответствуют вершины третьего уровня. Если подмножество содержит единственный элемент, то ему соответствует концевая вершина кодового дерева; такое подмножество разбиению не подлежит. Подобным образом поступаем до тех пор, пока не получим все концевые вершины. Ветви кодового дерева размечаем символами 1 и 0, как в случае кода Хаффмана.

При построении кода Шеннона — Фано разбиение множества элементов может быть произведено, вообще говоря, несколькими способами. Выбор разбиения на уровне n может ухудшить варианты разбиения на следующем уровне (n + 1) и привести к неоптимальности кода в целом. Другими словами,

оптимальное поведение на каждом шаге пути ещё не гарантирует оптимальности всей совокупности действий. Поэтому код Шеннона — Фано не является оптимальным в общем смысле, хотя и дает оптимальные результаты при некоторых распределениях вероятностей. Для одного и того же распределения вероятностей можно построить, вообще говоря, несколько кодов Шеннона — Фано, и все они могут дать различные результаты. Если построить все возможные коды Шеннона — Фано для данного распределения вероятностей, то среди них будут находиться и все коды Хаффмана, то есть оптимальные коды[5].

Метод контекстного моделирования (CM)

сокращение от context modeling - контекстное моделирование.

В этом методе строится модель исходных данных. При сжатии очередного элемента данных эта модель выдает свое предсказание или вероятность. Согласно этой вероятности, элемент данных кодируется энтропийным методом. Чем точнее модель будет соответствовать исходным данным, тем точнее она будет выдавать предсказания, и тем короче будут кодироваться элементы данных.

Для построения эффективной модели требуется много памяти. При распаковке приходится строить точно такую же модель. Поэтому скорость и требования к объему оперативной памяти для упаковки и распаковки почти одинаковы. В данный момент методы контекстного моделирования позволяют получить наилучшую степень сжатия, но отличаются чрезвычайно низкой скоростью.

PPM (Prediction by Partial Matching) – предсказание по частичному совпадению.

Это особый подвид контекстного моделирования. Предсказание выполняется на основании определенного количества предыдущих элементов данных. Основным параметром является порядок модели, который задает это количество элементов. Чем больше порядок модели, тем выше степень сжатия, но требуется больше оперативной памяти для хранения данных модели. Если оперативной памяти недостаточно, то такая модель с большим порядком

показывает низкие результаты. Метод PPM особенно эффективен для сжатия текстовых данных.

Предварительные преобразования или фильтрация

Данные методы служат не для сжатия, а для представления информации в удобном для дальнейшего сжатия виде. Например, для несжатых мультимедиа данных характерны плавные изменения уровня сигнала. Поэтому для них применяют дельта-преобразование, когда вместо абсолютного значения берется относительное. Существуют фильтры для текста, исполняемых файлов, баз данных и другие.

Метод сортировки блока данных (BWT) – сокращение от Burrows Wheeler Transform - по имени авторов.

Это особый вид или группа преобразований, в основе которых лежит сортировка. Такому преобразованию можно подвергать почти любые данные. Сортировка производится над блоками, поэтому данные предварительно разбиваются на части. Основным параметром является размер блока, который подвергается сортировке. Для распаковки данных необходимо проделать почти те же действия, что и при упаковке. Поэтому скорость и требования к оперативной памяти почти одинаковы. Архиваторы, которые используют данный метод, обычно показывают высокую скорость и степень сжатия для текстовых данных.

Непрерывные блоки или непрерывный режим (Solid mode)

Во многих методах сжатия начальный участок данных или файла кодируется плохо. Например, в словарном методе словарь пуст. В методе контекстного моделирования модель не построена. Когда количество файлов большое, а их размер маленький, общая степень сжатия значительно ухудшается за счет этих начальных участков. Чтобы этого не происходило при переходе на следующий файл, используется информация, полученная исходя из предыдущих файлов. Аналогичного эффекта можно добиться простым представлением исходных файлов в виде одного непрерывного файла.

Этот метод используется во многих архиваторах и имеет существенный недостаток. Для распаковки произвольного файла необходимо распаковать и

файлы, которые оказались в начале архива. Это необходимо для правильного заполнения словаря или построения модели. Существует и промежуточный вариант, когда используются непрерывные блоки фиксированного размера. Потери сжатия получаются минимальными, но для извлечения одного файла, который находится в конце большого архива, необходимо распаковать только один непрерывный блок, а не весь архив.

Сегментирование

Во всех методах сжатия при изменении типа данных собственно сам переход кодируется очень плохо. Словарь становится не актуальным, модель настроена на другие данные. В этих случаях применяется сегментирование. Это предварительная разбивка на однородные части. Затем эти части кодируются по отдельности или группами [3].

Нумерационное кодирование

Нумерационное кодирование – представление последовательности сообщений источника в виде последовательности целых чисел.

2.1 РАЗРАБОТКА АЛГОРИТМА МЕТОДА СЖАТИЯ ДВОИЧНЫХ СООБЩЕНИЙ

2.1 Разработка алгоритма метода сжатия двоичных сообщений на основе многозначных биномиальных чисел

Под сжатием информации понимают операцию, в результате которой данному коду или сообщению ставится в соответствие более короткий код или сообщение.

Актуальность проблемы сжатия различных сигналов в настоящее время очевидна. Особенно важно сжимать изображения. Простые расчеты показывают, что сжатие изображений является фактически обязательным для уменьшения требований к емкости массовой памяти и скоростям передачи для систем, связанных с хранением и обработкой неподвижных и видеоизображений. Для цифрового представления 35-мм негатива, сканированного с разрешением 12мкм, требуется около 144 Мбит (2048*3072 ЭИ при24 бит/ЭИ). Цифры становятся еще более впечатляющими, если говорить об истинных видеофильмах с движущимися изображениями. Для хранения одной секунды несжатого цветного видеофильма при скорости 30 кадров в секунду требуется 512*480 ЭИ/кадр при 24 бит/ЭИ, или 180 Мбит.

Следует отметить, что методы и устройства сжатия различных сигналов (в том числе и телевизионных) интенсивно развивались с 70-х годов, но только в последнее время на базе новых электронных технологий достигнуты значительные результаты. Достаточно эффективным оказалось нумерационное кодирование данных.

Для сжатия и восстановления изображений можно использовать как преимущественно аппаратные средства, так и программный способ реализации алгоритма. Выбор способа определяется тремя основными факторами: размером изображения, скоростью передачи данных и сложностью алгоритма. Программный способ реализации предпочтителен, как правило, для алгоритмов сжатия и восстановления небольших неподвижных изображений, особенно если

скорость не имеет первостепенного значения. Программные средства можно использовать также для реализации алгоритмов восстановления движущихся изображений, поскольку эти алгоритмы требуют менее интенсивных вычислений, чем алгоритмы сжатия данных. Системы, осуществляющие сжатие / восстановление изображений аппаратными средствами, сейчас почти в 30 раз превосходят по быстродействию аналогичные программные системы с применением самых высокоскоростных компьютеров.

В данной работе к рассмотрению предлагается аппаратная реализация алгоритма сжатия нумерацией двоичных последовательностей на основе многозначной биномиальной системы счисления.

Известные методы нумерации осуществляют процедуру сжатия в течение одной процедуры. Однако эти методы сложны и трудно реализуемы аппаратно и при этом недостаточно универсальны.

Устранить эти недостатки можно путем использования структурных систем счисления, в частности, многозначной биномиальной системы счисления. При использовании структурных систем счисления нумерация осуществляется в два этапа: сначала организовывается переход к числу в биномиальной системе счисления, а затем происходит переход от этого числа к его номеру.

Используемая в рассмотренном методе сжатия специальная система счисления с неоднородной структурой - многозначная q-ичная биномиальная система счисления - характеризуется тем, что:

а) ее диапазон и весовые значения разрядов задаются биномиальными коэффициентами;

б) максимальное число разрядов в биномиальных числах равно k;

в) количество r информационных разрядов для различных биномиальных чисел является переменным;

г) алфавит используемых цифр с учетом нуля содержит m-k цифр, где m-параметр системы счисления, влияющий на ее диапазон;

д) вес разряда зависит от его местоположения в числе, стоящей в нем цифры и предшествующих цифр;

е) содержит контрольное число $q=m-k$, превышение которого в биномиальном числе приводит к появлению в нем ошибки.

Числовая функция, представляющая многозначную биномиальную систему счисления, имеет следующий вид:

$$F = \sum_{i=0}^{X_{r-1}-1} C_{m-i-1-X_{r-0}}^{k-1} + \sum_{i=0}^{X_{r-2}-1} C_{m-i-2-X_{r-0}-X_{r-1}}^{k-2} + \dots + \sum_{i=0}^{X_{r-j}-1} C_{m-i-j-q_j}^{k-j} + \dots + \sum_{i=0}^{X_0-1} C_{m-i-r-q_r}^{k-r} = \sum_{j=0}^{k-1} A_j, \qquad (2.1.1)$$

где $A_j = \sum_{j=0}^{X_{r-j}-1} C_{m-i-j-q_j}^{k-j}$, $q_j = \sum_{p=0}^{j-1} X_{r-p}$, $X_{r-0} = 0$,

X_{r-j}-цифра *(r-j)*-го разряда, *j=1,2,...,r.*

при следующих ограничениях:

$$0 \le X_0 \le n - p - \sum_{\gamma=0}^{k-1} X_k - \gamma;$$

$$X_{k-0} = 0; \qquad (2.1.2)$$

$$p-k \ge 0$$

Если q - контрольное число, то $q = n-p$.

При решении задачи нумерации сигналов двоичные последовательности рассматриваются как равновесные кодовые комбинации. Это объясняется тем, что подвергать процедуре сжатия путем нумерации можно только кодовые комбинации, характеризующиеся одинаковой вероятностью их генерирования. А равновесные кодовые комбинации как раз и характеризуются одинаковым количеством единиц и, как следствие, одинаковой вероятностью генерирования двоичных слов. Это свойство позволяет отнести равновесные коды к комбинаторным конфигурациям, которые наиболее удобно нумеровать с помощью биномиальных систем счисления.

Задача нумерации равновесных кодов разбивается на два этапа и решается следующим образом: вначале осуществляется переход от двоичного кодового слова с постоянным весом к сочетанию, а затем к биномиальному многозначному слову и, наконец, в соответствии с выражением (2.1.1) - к степенной системе счисления, т.е. к номеру.

Рассмотрим преобразования более подробно.

Исходное двоичное слово характеризуется следующими параметрами: длиной слова - m и количеством единиц - k. Эти параметры являются исходными для определения параметра многозначной биномиальной системы счисления, который определяется согласно соотношению $q=m-k$.

На следующем этапе формируют комбинаторную конфигурацию типа сочетания, затем осуществляется переход к биномиальному числу, далее производится подсчет количественного эквивалента биномиального числа.

Данный алгоритм может быть представлен с помощью укрупненной блок-схемы, приведенной на рисунке 2.1.1.

Рассмотрим более подробно каждую операцию данного преобразования.

Подсчет количества единиц в кодовых комбинациях может быть реализован согласно блок-схеме алгоритма, приведенной на рисунке 2.1.2.

При этом последовательно перебираются разряды заданного двоичного числа, начиная с нулевого, и в случае равенства единице анализируемого разряда устройство, фиксирующее количество единиц, должно увеличить на единицу свое состояние. Процесс прекращается, когда проанализирован последний $(m-1)$-й разряд двоичного числа.

На следующем этапе определяется параметр биномиальной системы счисления, определяемый из соотношения $q=m-k$. Значения m и k чаще всего определяются характеристиками исходной двоичной кодовой комбинации.

Обычно в качестве значения m фигурирует исходная длина двоичной кодовой комбинации. Длина комбинации определяется с помощью управляемого регистра. Значение k – количество единиц - подсчитывается. Далее определяется их разность. Чаще всего параметр многозначной

биномиальной системы счисления задается заказчиком. Это обусловлено тем, что от значения параметра зависит диапазон переводимых чисел и, как следствие, характеристика помехоустойчивости получаемого кода.

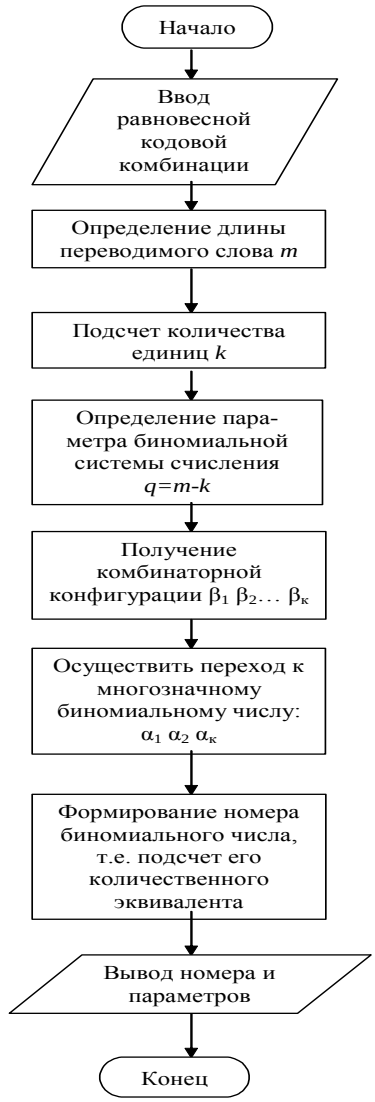

Рисунок 2.1.1 – Укрупненная блок-схема алгоритма функционирования

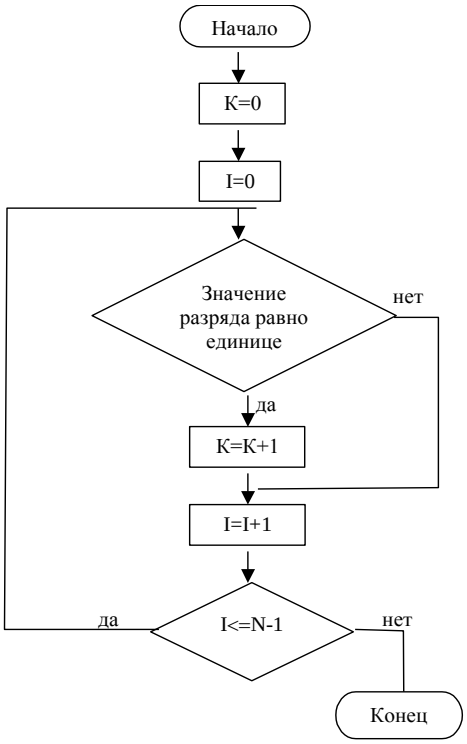

Рисунок 2.1.2 – Блок – схема алгоритма подсчета количества единиц в равновесной кодовой комбинации

При последующем преобразовании равновесного кодового слова необходимо последовательно в порядке возрастания записать адреса (номера разрядов) единиц в кодовых комбинациях. Полученная запись представляет собой комбинаторную конфигурацию типа сочетание. Например, равновесному кодовому слову 01100101 соответствует “сочетание” 1367. Это преобразование

может быть реализовано с помощью блок-схемы алгоритма, приведенной на рисунке 2.1.3.

Преобразование начинается с младшего разряда, которому при формировании сочетаний присваивается единичный номер, так как сформированная в результате преобразований конфигурация типа сочетания не может содержать нуль. Анализируется младший разряд. Если он равен единице, то адрес этого разряда записывается, иначе – теряется.

Переход от сочетания к многозначному биномиальному числу организуем с помощью алгоритма, построенного на основе утверждения.

Утверждение. Если $\beta_1\beta_2...\beta_i...\beta_k$ есть сочетание и если от каждой, кроме первой, β_i цифры сочетания отнять предыдущую при счете слева направо цифру сочетания и единицу, а первая цифра равна старшей цифре сочетания, то полученная цифра $\alpha_i = \beta_i - \beta_{i-1} - 1$ является элементом последовательности, которая образует многозначное биномиальное число $\alpha_1\alpha_2...\alpha_i...\alpha_k$.

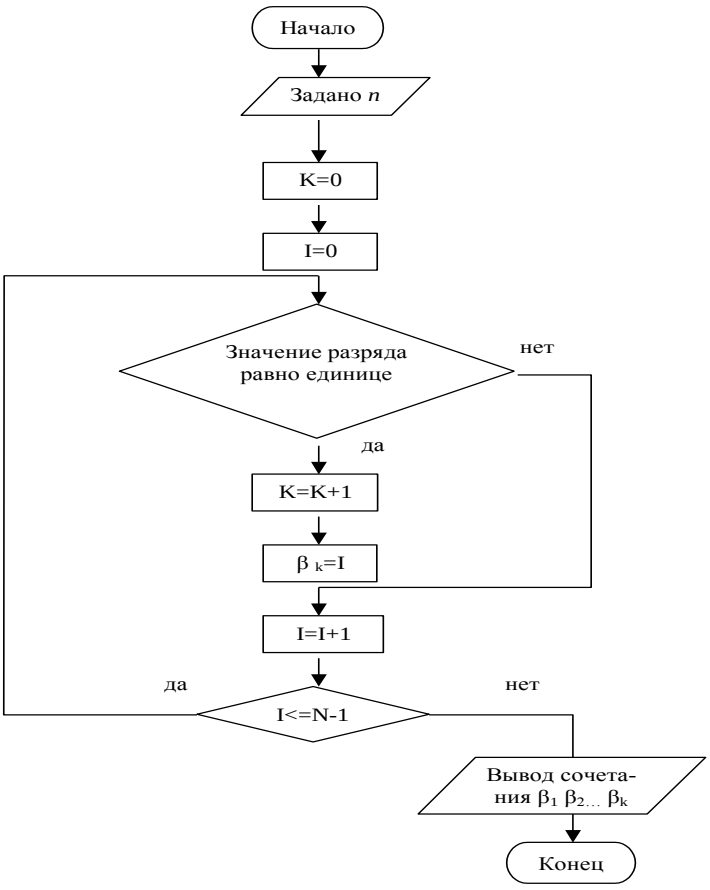

Рисунок 2.1.3 – Блок-схема алгоритма получения сочетания

Алгоритм преобразования сочетания в многозначное биномиальное число имеет следующий вид.

1 Старшая цифра многозначного биномиального числа равна первой цифре сочетания.

2 Вычисляется следующая цифра многозначного биномиального числа $\alpha_i = \beta_i - \beta_{i-1} - 1$.

3 Пункт 2 повторять, пока не будет получена младшая цифра многозначного биномиального числа.

Предложенный алгоритм реализуем с помощью блок-схемы, которая имеет вид, представленный на рисунке 4.

Рассмотрим пример получения биномиального многозначного числа из сочетания.

Пример. Преобразовать сочетание 1367 в многозначное биномиальное число.

$\alpha_1 = \beta_1 = 1$.

$\alpha_2 = \beta_2 - \beta_1 - 1 = 3-1-1 = 1$.

$\alpha_3 = \beta_3 - \beta_3 - 1 = 6-3-1 = 2$.

$\alpha_4 = \beta_4 - \beta_3 - 1 = 7-6-1 = 0$.

Получено многозначное биномиальное число 1120.

Переход от многозначного биномиального числа к номеру может быть осуществлен путем подстановки в (2.1.1) вместо X_i их значений и вычисления количественного эквивалента биномиального числа в десятичной системе счисления. Рассмотрим пример вычисления количественного эквивалента полученного биномиального числа.

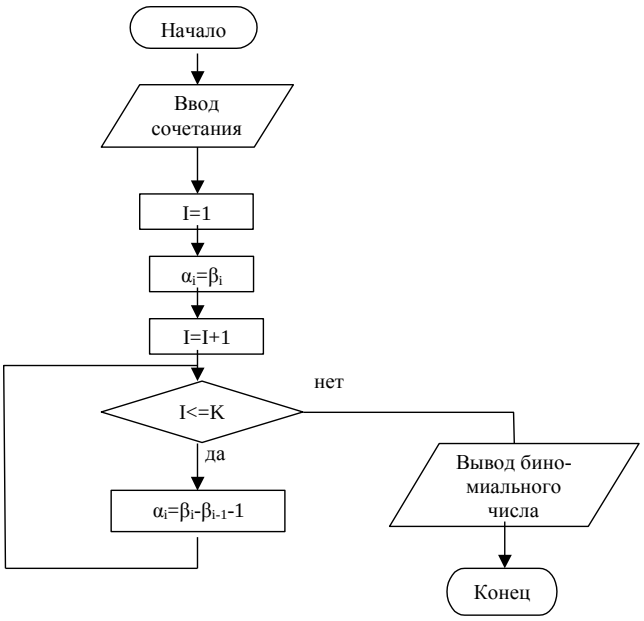

Рисунок 2.1.4 – Блок – схема алгоритма получения биномиального числа из сочетания

Зададимся исходными значениями.

Так как $k=4$, $q=m-k=4$,то

$X_3=1$, $A_3 = C_{n-1}^{k-1} = C_7^3 = 35$;

$X_2=1$, $A_2 = C_5^2 = 10$;

$X_1=2$, $A_1 = C_3^1 + C_2^1 = 3+2 = 5$;

$X_0=0$, $A_0 = 0$.

Десятичный эквивалент биномиального числа 1120 равен $A_3 + A_2 + A_1 + A_0$ = =35+10+5+0= 50. Это число является номером равновесного кодового слова 01100101. Данный алгоритм можно реализовать с помощью блок – схемы, представленной на рисунке 2.1.5.

При получении номера биномиального многозначного числа используется принцип поразрядного взвешивания. Разряды, представленные в биномиальном числе значениями, отличными от нуля, будут присутствовать в номере в качестве их десятичного эквивалента, а те разряды, которые раны нулю, – нет.

Необходимо отметить, что анализ приведенных блок – схем алгоритмов позволил сделать вывод о том, что алгоритмы под номерами 2 и 3 практически идентичны, за исключением одной операции. Это говорит о том, что при последовательном вводе двоичного числа, которое должно преобразовываться, можно практически одновременно выполнять операцию подсчета количества единиц в кодовой комбинации и операцию формирования комбинаторной конфигурации – сочетания по записи адресов тех же единиц [1].

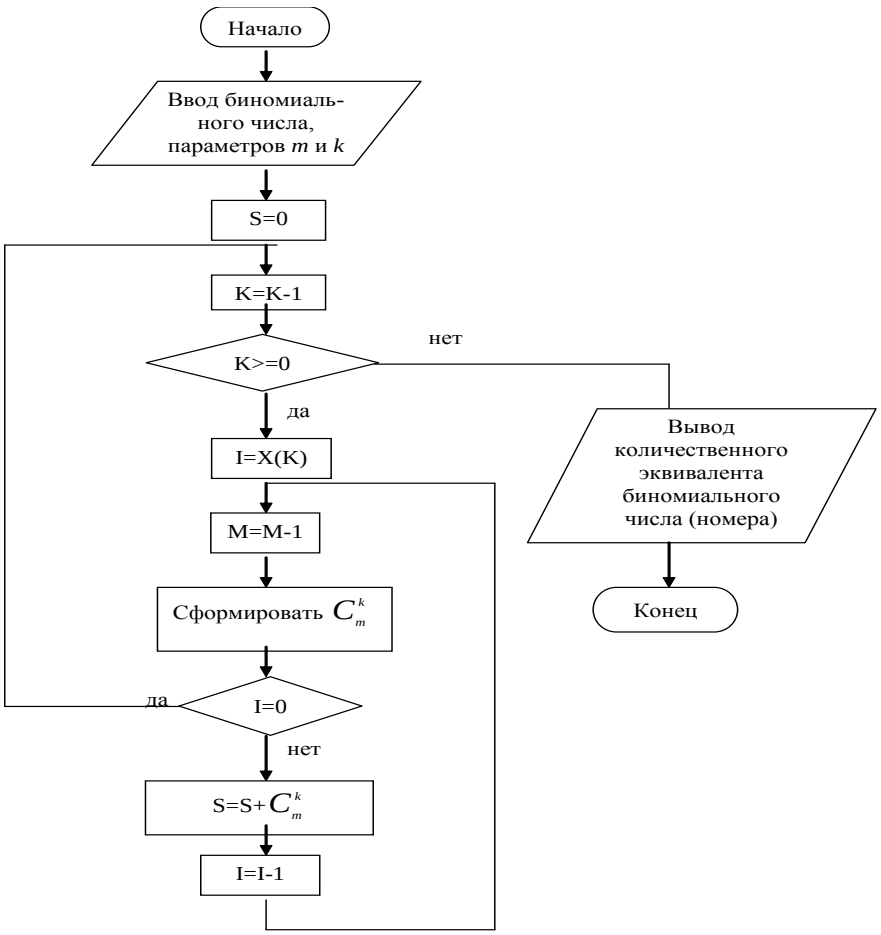

Рисунок 2.1.5 – Блок – схема алгоритма получения количественного эквивалента многозначного биномиального числа

Для того чтобы оценить коэффициент сжатия иформации выберем несколько двоичных сообщений разной длинны проезведем необходимые преобразования, найдем номер сообщения. Для определения коэффициента сжатия найдем необходимое количество бит для предоставления двоичного эквивалента номера. Коэффициент сжития определим за формулой:

$$коэф.сжатия = \frac{длинна\_исходного\_сообщения}{длинна\_выходного\_сообщения} \quad (2.1.3).$$

Преведем данные в таблице 2.1.1:

Таблица 2.1.1 – Определение коэффициента сжатия двоичных сообщений на основе многозначных биномиальных чисел

исходное сообщение	к-во бит	номер сообщения	необх. к-во бит	коэф.сжатия
10	2	2	2	1
101	3	3	2	1,5
1010	4	6	3	1,33
10101	5	10	4	1,25
11000101	8	50	6	1,33
1001000101	10	76	7	1,43
1 001 011 000 010	13	1131	11	1,18
10 110 001 010 010 100	17	25899	15	1,13
10110100010011101100	20	155518	18	1,11

Построим график на основе данных таблицы 2.1.1.

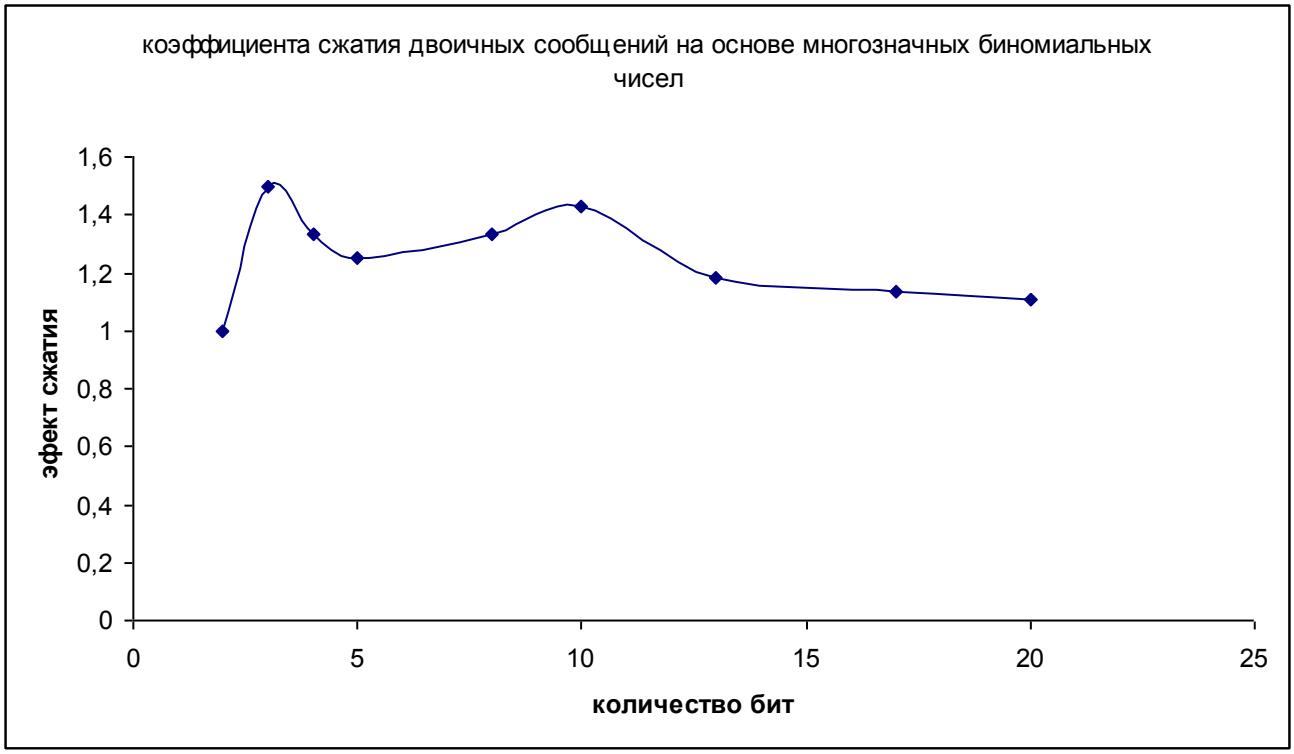

Рисунок 2.1.6 – Коэффициента сжатия двоичных сообщений на основе многозначных биномиальных чисел

Как мы видим с графика, найболее оптимальное использование данного метода при длинне двоичного сообщения от 3 до 10 бит.

2.2 Разработка структурной электрической схемы прибора

Разработанные алгоритмы необходимо представить в виде устройства, реализовывающего рассмотренные функции. Приведем структурную схему нумератора двоичных последовательностей на основе многозначных биномиальных чисел, которая представленный на рисунке 2.2.1.

Схема должна содержать следующие основные узлы:

- регистр исходного двоичного числа;
- формирователь сочетаний;
- формирователь многозначных биномиальных чисел;
- формирователь Y;
- формирователь Z;
- ПЗУ;
- формирователь номера;
- выходной регистр.

Управлять работой устройства и синхронизировать ее должна схема управления.

Так как при решении задачи нумерации двоичных слов на основе биномиальной системы счисления с многозначным алфавитом выделяется два этапа – вначале осуществляется переход от двоичного слова с постоянным весом к биномиальному многозначному слову, а затем осуществляется нумерация полученной многозначной биномиальной комбинации, то соответственно целесообразно устройство организовать таким образом, чтобы функционально можно было выделить узлы, решающие соответственно первую и вторую задачи.

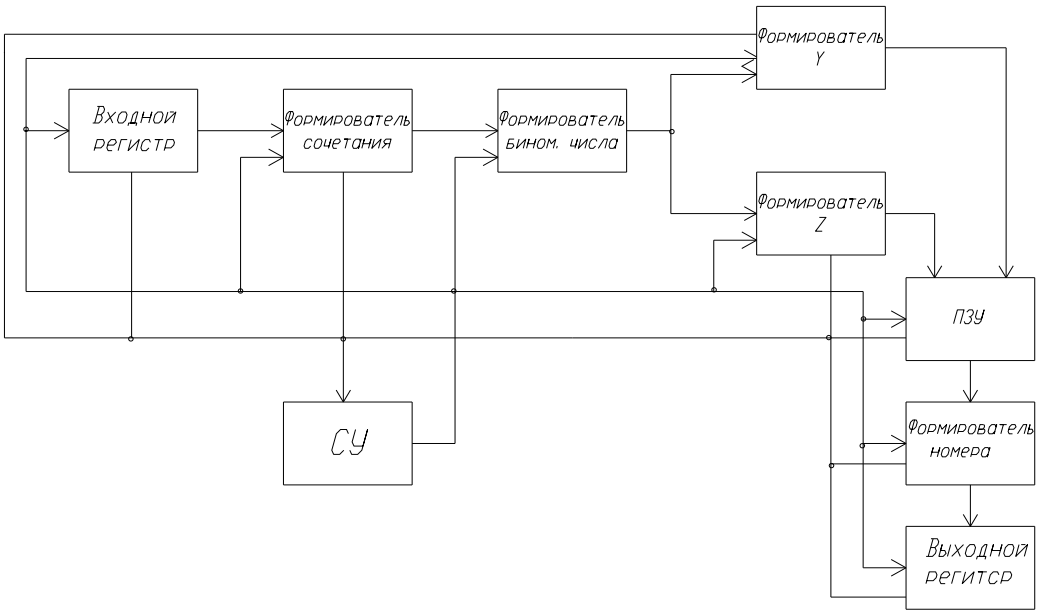

Рисунок 2.2.1 – Структурная схема устройства нумерации двоичных чисел на основе многозначных биномиальных чисел

Опишем работу структурной схемы.

На входной регистр поступает двоичная равновесная комбинация, которая преобразовывается и передается на формирователь сочетаний. После преобразования в сочетания входная комбинация переходит в следующий блок, где формируется биномиальное число. Дале операцию C_Z^Y мы разбиваем на три блока:

1) формирователь Y;

2) формирователь Z;

3) ПЗУ(постоянное запоминающее устройство).

В первых двух блоках параллельно находим значения Y и Z, которые рассчитываются за формулами:

$$\begin{aligned} Y &= k - n \\ Z &= m - i - n - X_{r-0} - X_{r-1} \end{aligned} \qquad (2.2.1)$$

После расчета Y и Z переходят в ПЗУ, которое выдает соответствующие значения C_Z^Y. В блоке формирователя номера рассчитывается номер, который выдается в канал связи с помощью выходного регистра.

ВЫВОДЫ

В ходе выполнения квалификационной работы было рассмотрено и изучено строение и принцип работы различных методов сжатия данных. Была разработана и описана работа устройства реализующего метод сжатия двоичных сообщений на основе многозначных биноминальных чисел и устройство для его реализации.

Разработанное устройство каждом сообщении присваивает номер, тем самым уменьшая объем занимаемого им памяти.

Данное устройство может выполняться на ПЛИС, что обеспечит их высокую степень интеграции. Это даст возможность применения их в бытовых и промышленных приборах.

Построение устройства сжатия информации в классическом исполнении увеличивает общую схему устройства, соответственно уменьшая его надежность и увеличивая материальные затраты. Выходом из данной ситуации является применение программируемых логических интегральных схем. Схема значительно упрощается в изготовлении, имеет значительно меньшие размеры, уменьшается количество деталей - соответственно повышается надежность.

Найболее эффективно использовать данный метод для двоичных сообщений длинной от 3 до 10 бит.

СПИСОК ЛИТЕРАТУРЫ

1. Протасова, Т.А. Метод сжатия двоичных сообщений на основе многозначных биноминальных чисел и устройство для его реализации. Вісник Сумського державного університету. Серія Технічні науки. — 2003. — №11(57). — С. 122-134.1

2. Петровский И.И. Логические ИС КР1533. КР1534. Справочник. Часть 1,2

3. http://ru.wikipedia.org/wiki/Сжатие_данных

4. http://www.firststeps.ru/theory/compr.html

5. http://ru.wikipedia.org/wiki/Алгоритм_Шеннона_—_Фано

6. http://ru.gentoo-wiki.com/wiki/Методы_сжатия_информации