УДК 004.627

АППАРАТНАЯ РЕАЛИЗАЦИЯ АДАПТИВНОГО АЛГОРИТМА ПОСТРОЕНИЯ КОДА ХАФФМАНА

Ю.А Зубань, канд. техн. наук; В.В. Петров, магистр
Сумский государственный университет

Адаптивные алгоритмы кодирования весьма эффективны для сжатия данных. Адаптивный (динамический) метод Хаффмана позволяет достичь наиболее высокой степени сжатия и достаточно хорошей скорости обработки данных. Аппаратная реализация адаптивного алгоритма построения кода Хаффмана - основная цель этой статьи. Полученная модель устройства и результаты исследования могут использоваться для аппаратной реализации адаптивного алгоритма кодирования Хаффмана.

ВВЕДЕНИЕ

С каждым годом растут объемы информации, которые необходимо передавать, обрабатывать и хранить для ее своевременного использования при решении производственных, научных, экономических и других задач. Для снижения затрат на обработку информации, ее хранение и поиск, а также для уменьшения емкости памяти, занятой в электронно-вычислительных машинах (ЭВМ), применяется сжатие информации. Так как объемы сжимаемых данных постоянно увеличиваются, то растут и требования к программам и устройствам сжатия. Поэтому разработка новых методов сжатия, алгоритмов и устройств, их реализующих, является востребованной задачей.

Под сжатием информации понимают операцию, вследствие которой определенному коду или сообщению ставится в соответствие код или сообщение меньшей длины. Одним из способов сжатия информации без потерь, который широко применяется, является кодирование Хаффмана [1].

В кодировании Хаффмана входным символам ставятся в соответствие двоичные кодовые комбинации переменной длины. Длина кода каждого символа обратно пропорциональна двоичному логарифму его вероятности. Это кодирование является префиксным, что позволяет легко декодировать, так как при префиксном кодировании код любого символа не является префиксом кода никакого другого символа.

Метод Хаффмана дает достаточно высокую скорость и хорошее качество сжатия. Широко применяется как в программных (всевозможные компрессоры, архиваторы и программы резервного копирования файлов и дисков), так и в аппаратных (системы сжатия, "прошитые" в модемы и факсы, сканеры) реализациях.

Как правило, кодовые комбинации синтезируются программно. Для аппаратной реализации обычно используется табличный метод. Его суть в том, что кодирование входных символов осуществляется с помощью таблицы (размещенной в ROM, ПЛМ и других носителях данных,

имеющих адресный доступ и соответствующую разрядность), которая содержит в себе информацию о кодовом дереве или о нескольких деревьях, структура и количество которых выбираются в зависимости от характеристик источника информации. По сути, в данном случае таблица представляет собой преобразователь кода, который пришедшей кодовой комбинации ставит в соответствие одну из немногих комбинаций кода Хаффмана согласно изменению одного (или группы) параметров источника информации [2].

ПОСТАНОВКА ЗАДАЧИ

Синтез универсального устройства для построения кода Хаффмана – актуальная задача, решение которой позволяет построить более адекватную модель источника информации и соответственно получить более оптимальный с точки зрения средней длины код для любого распределения вероятностей кодируемого алфавита. Целью данной работы является разработка универсального электронного устройства для аппаратной реализации адаптивного алгоритма построения кода Хаффмана.

Если распределение вероятностей символов алфавита входного потока известно, то можно построить модель оптимального кодирования. Однако если распределение вероятностей заранее неизвестно, то задача значительно усложняется. В этом случае применяется метод адаптивного кодирования. Его общий принцип состоит в том, чтобы менять схему кодирования в зависимости от характера изменений входного потока. Такой подход имеет однопроходный алгоритм и не требует сохранения информации об использованном кодировании в явном виде. Адаптивное кодирование может дать большую степень сжатия по сравнению со статическим, поскольку более полно учитываются изменения частот входного потока. Данный метод известен как динамическое кодирование Хаффмана.

АНАЛИЗ И АППАРАТНАЯ РЕАЛИЗАЦИЯ АДАПТИВНОГО АЛГОРИТМА ПОСТРОЕНИЯ КОДА ХАФФМАНА

Как и всех адаптивных систем, работу устройства сжатия, построенного на основе кода Хаффмана, можно представить в виде такой структуры (рис. 1), в которую входят моделирование источника информации и кодирование.

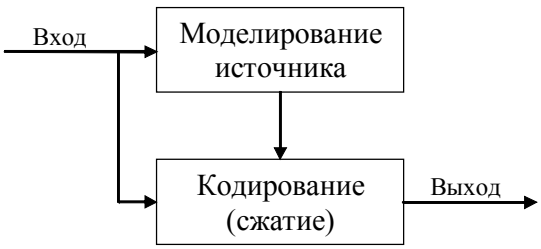

Рисунок 1 – Общая структура адаптивной системы сжатия информации

Моделирование источника системой определяет степень адаптивности последней к кодируемому информационному потоку. Чем более адекватна модель источника в данный момент времени самому источнику, тем больший коэффициент сжатия можно получить при кодировании. При поступлении очередного символа система изменяет модель и потом на основании этой модели кодирует (сжимает) поступивший символ.

Код Хаффмана строится по следующему алгоритму:

1) выписываются в ряд все символы алфавита в порядке возрастания

или убывания вероятности их появления в тексте;

2) последовательно объединяются два символа с наименьшими вероятностями появления в новый составной узел, вероятность которого равна сумме вероятностей составляющих его элементов;

3) в результате получится дерево, каждый узел (кроме листьев) которого имеет суммарную вероятность всех узлов, находящихся ниже него;

4) прослеживается путь к каждому листу дерева, помечая значением бита направление к каждому узлу (например, направо – 1, налево – 0).

При аппаратной реализации алгоритма формирования кода Хаффмана возникает задача представления и хранения в удобном для изменения виде информации о кодовом дереве. Для этого целесообразно применять динамически связанные списки– набор связанных между собой элементов. Каждый элемент списка состоит из нескольких полей, одним из которых является адрес, так называемая ссылка (link) на следующий элемент списка. Это дает возможность однозначно определить порядок следования элементов, т.е. структуру кодового дерева (рис. 2). Списки называются динамически связанными, потому что для изменения порядка не нужно перемещать сами элементы, изменяются только ссылки – адреса переходов.

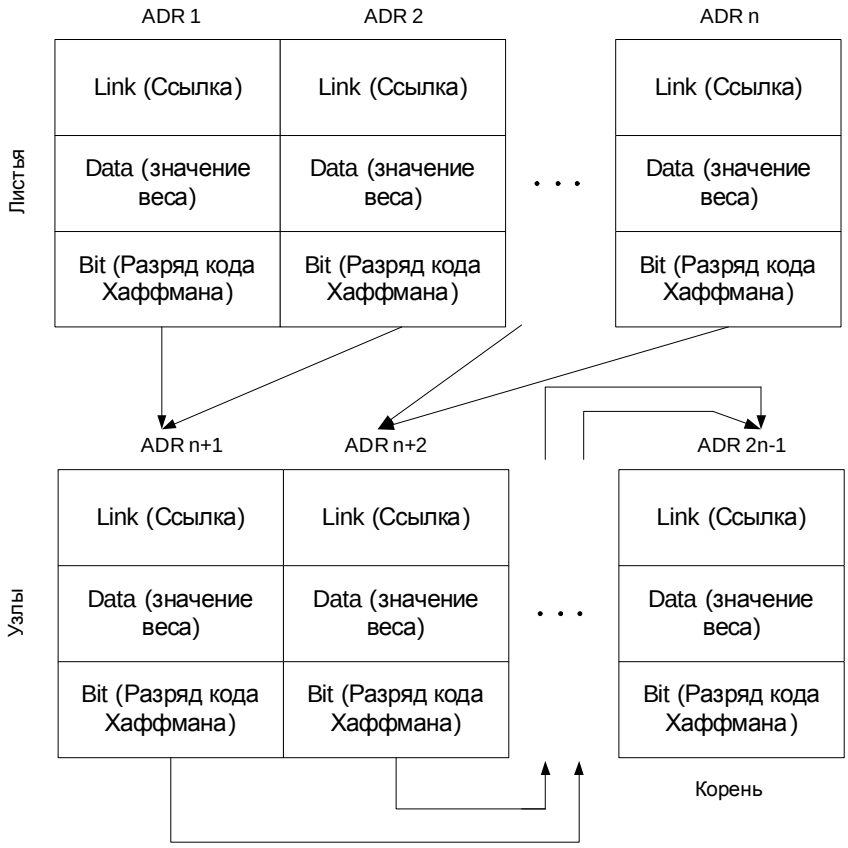

Рисунок 2 – Структура кодового дерева Хаффмана в памяти с использованием динамически связанных списков

Согласно порядку формирования кода Хаффмана с учетом применения списков алгоритм функционирования устройства будет иметь такой вид:

1) создается список листьев дерева Хаффмана — элементов множества

M = {m_1, m_2,..., m_n}, где n – количество букв входного алфавита, в которых записаны вес и ссылка на следующий элемент. Список формируется непрерывным, т.е. последний элемент ссылается на первый;

2) двум вспомогательным элементам (min1 и min2) присваиваются максимальные значения вероятностей. Эти элементы не являются частью списка и нужны лишь для поиска и хранения двух наименьших значений;

3) проводится сравнение каждого элемента динамически связанного списка со значениями min1 и min2 по такому принципу: если значение i-го элемента (i = n...1) меньше или равно min1, то содержимое min1 переписывается в min2, а в min1 заносится значение i-го элемента и его адрес (link). Если предыдущее условие не выполняется, то происходит сравнение с min2 — когда значение i-го элемента связанного списка больше или равно min2, то в таком случае в последний заносятся значение i-го элемента и ссылка (link) на него. Сравнение происходит только в том случае, если значение списка больше нуля, в противном случае берется следующий элемент. Сигналом окончания сравнения будет служить адрес последнего элемента;

4) создается новый элемент списка (узел) с адресом n+M-1 (где M – некоторое вспомогательное значение, первоначально равное n), в который заносится суммарное значение двух наименьших элементов, и формируется ссылка на ячейку, в которой находится link на наименьший элемент;

5) в поле Bit самого наименьшего элемента (min1) заносится 1, а в min2 – 0. Также их ссылки меняются на значение адреса вновь созданного элемента – узла дерева. В остальных элементах связанного списка ссылки на два наименьших элемента min1 и min2 заменяются ссылкой на созданный узел. Значение M уменьшается на 1;

6) если M больше единицы, то повторяется операция нахождения двух элементов с наименьшими значениями, создание нового узла и модификация списка, иначе – формирование дерева Хаффмана считается оконченным;

7) код кодируемого символа, принадлежащий входному алфавиту A = {a_1, a_2, ..., a_n}, является адресом листа дерева, с которого начинается формирование кода Хаффмана. При переходах по ссылкам считываются ранее сформированные значения полей Bit (нуль или единица), с которых и формируется код Хаффмана для кодируемого символа. Окончанием формирования кода символа будет служить link со значением 2n-1.

Устройство формирования кода должно выполнять две основные функции: формировать и хранить дерево Хаффмана (моделирование) и преобразовывать входные сообщения в двоичном натуральном коде в соответствующие выходные двоичные последовательности кода Хаффмана (кодирование).

Автомат можно представить в виде таких функциональных блоков (рис. 3):

1) буфер – производит “слежение” и сопряжение входного потока информации, синхронизуя последний по времени с самой системой;

2) блок статистики – ведет статистику: хранит и модифицирует частоты повторений (вероятностей) символов входного алфавита;

3) блок хранения дерева Хаффмана – содержит в себе информацию о кодовом дереве, связях между узлами и значениях поля Bit выходного кода, которые считываются при перемещении по дереву;

4) блок сравнения – находит два наименьших узла дерева;

5) блок формирования переходов и создания нового узла – выполняет функции создания и модификации списка;

6) кодер – реализует распознавание листа дерева Хаффмана, переходы между узлами и выдачу последовательности битов кода Хаффмана для

определенного символа входного алфавита;

7) блок управления – отвечает за правильную и синхронную работу всех блоков устройства как единой системы.

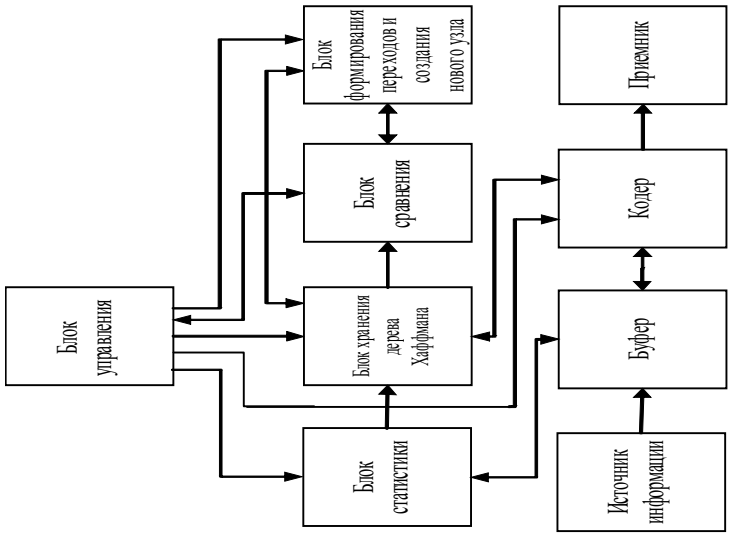

Рисунок 3 – Структурная схема устройства адаптивного построения кода Хаффмана

Блок статистики выполняет функции сбора, хранения и модификации информации о частоте повторения символов входного алфавита. Он реализуется в виде массива данных (оперативной памяти), сумматора и промежуточного регистра временного хранения результата.

Например, для кодирования символов, представленных в коде КОИ-8 в качестве хранилища динамически изменяющейся статистической информации, рационально использовать статическую оперативную память размером 256x8, что соответствует объему и разрядности кодовой таблицы символов. То есть на вход этой схемы должны поступать однобайтные слова, соответствующие символам входного алфавита. К частоте повторения символа с помощью сумматора SM прибавляется единица (рис. 4). Получившееся значение запоминается в регистре, после чего записывается по текущему адресу в RAM. Таким образом, частота повторения поступившего символа увеличивается на 1.

Блок статистики передаёт хранящиеся в нём данные в блок хранения дерева Хаффмана (оперативное запоминающее устройство).

Функцию формирования дерева выполняет преобразователь, состоящий из устройства сравнения и блока формирования переходов и создания нового узла. Устройство сравнения предназначено для поиска двух наименьших значений вероятностей (min1 и min2), проверки условий окончания просмотра списка и формирования дерева, изменения начального и конечного адресов поиска наименьших значений путем сравнения. То есть этот блок получает значения, записанные в ОЗУ, проводит сравнение, чтобы определить элементы связанного списка с двумя наименьшими весами, и по окончании посылает полученные данные в блок формирования переходов для создания нового узла. Последний принимает данные от устройства сравнения, создаёт новый узел дерева Хаффмана, ссылку на созданный узел, изменяет ссылки других узлов и листьев и записывает их в ОЗУ.

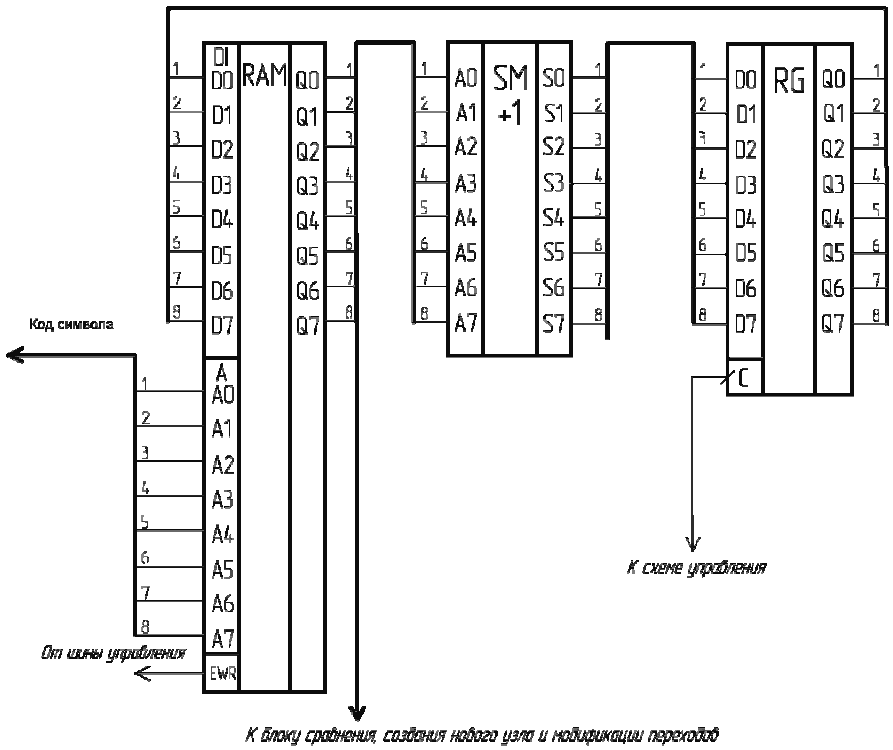

Рисунок 4 – Блок статистики

Функцию кодирования информации, поступающей с блока хранения, выполняет кодер. Он сигнализирует блоку хранения о готовности принять код символа, после чего получает данные. Путем выборки конкретного листа и переходов от него до корня дерева по адресам ссылок между элементами списка происходит формирование кода Хаффмана для конкретного символа.

Для формирования необходимой последовательности битов кода Хаффмана, полученных при переходах по дереву от листа к корню, применяется стековая память. С ее помощью изменяется порядок следования битов на обратный. Она состоит из универсального регистра сдвига RG, логических элементов, RS – триггера и двоичного реверсивного счетчика CT. Схема приведена на рис. 5

Узел работает таким образом: при включении питания происходит обнуление RS-триггера и реверсивного счетчика CT, который настраивается на режим суммирования.

При поступлении кода очередного символа происходят изменение статистики и построение нового кода символа по кодовому дереву. Синхронно с переходами от листа дерева до корня счетчик увеличивает значение на 1 с каждым переходом, а значения поля Bit записываются последовательно со сдвигом в RG. Это происходит до момента достижения адреса корня кодового дерева. Признаком достижения корня является сигнал на выходе дешифратора DC, который переводит триггер в единичное состояние и тем самым изменяет направление сдвига регистра RG и режим работы счетчика с суммирующего на вычитающий. Это обеспечивает выдачу кодовой последовательности Хаффмана для кодируемого символа в нужном порядке.

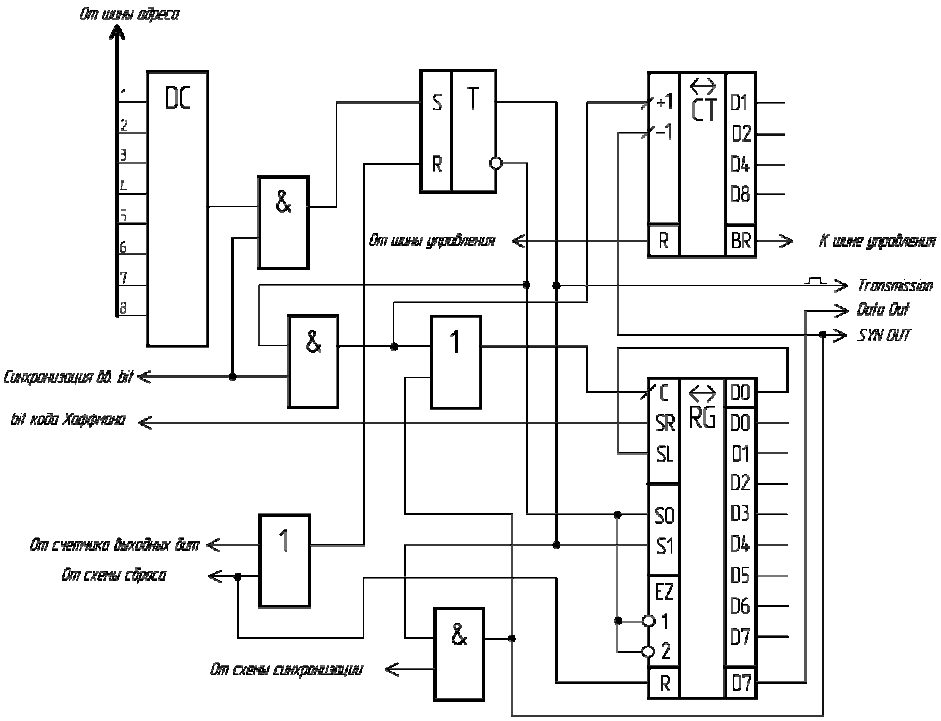

Рисунок 5 – Схема стековой памяти

Сигал окончания передачи выработается счетчиком CT при достижении нуля (выход BR). Он поступает на вход сброса RS-триггера и переводит регистр в режим приема и сдвига вправо.

Для синхронной и правильной работы проектируемого устройства нужно обеспечить управление всеми его блоками. Эту задачу выполняет схема управления. Она обеспечивает:

1) начальную установку и сброс;
2) начальную запись и считывание данных с ОЗУ;
3) подключение устройства сравнения;
4) подключение кодера и схемы изменения порядка.

Метод Хаффмана требует определенного времени для изменения дерева. Построение кодового дерева Хаффмана происходит при приходе каждого символа, тем самым изменяется модель источника. Это занимает определенное время, которое зависит от количества символов исходного алфавита (время увеличивается с увеличением символов). Уменьшить это время можно, если перестраивать дерево не при приходе каждого символа, а при выполнении определенных условий.

При анализе [3] было показано, что при приходе некоторого числа символов дерево Хаффмана может не изменяться. Даже при изменении кодового дерева длина кода может оставаться прежней. Поэтому критерием изменения модели должно быть уменьшение или увеличение длины кода Хаффмана.

Признаком этого, исходя из [3], может служить сравнение максимальных и минимальных элементов на разных уровнях кодового дерева. По сути перестраивать дерево (модифицировать код) нужно, если частота повторения символа, максимального на уровне L, станет больше частоты символа, минимального на L-1 уровне [4]:

$$N(a_{maxL}) > N(a_{minL\text{-}1}), L = 1,2,3,\ldots\ , \qquad (1)$$

где N – частота повторения символа;

L – номер уровня в дереве для поступившей буквы ($L = 0$ – корень кодового дерева).

Смысл формулы (1) заключается в том, что перестраивать дерево (модифицировать код) нужно, если вес символа a_{max} , максимального на уровне L, станет больше веса символа a_{min} , минимального на L-1 уровне.

Таким образом, используя указанный выше критерий можно значительно сократить время, необходимое для реализации адаптивного алгоритма построения кода Хаффмана.

Для декодирования адаптивного кода вводится служебная информация об изменении дерева кода Хаффмана. При этом возможна ситуация, что при условии изменения модели источника сжатие не происходит, а, наоборот, возрастет избыточность за счет служебной информации. Для того чтобы выполнялось сжатие, длина выходной последовательности должна удовлетворять следующему выражению:

$$\sum_{i=1}^{k} (N(a_i) \times n_{вх}(a_i))\Big|_{k=1,2,3,\ldots}^{вх} > \sum_{i=1}^{k} (N(a_i) \times n_{вых}(a_i))\Big|_{k=1,2,3,\ldots}^{вых} + l_{служ}\ , \qquad (2)$$

где $N(a_i)$ – вес a_i символа;

$n_{вх}(a_i)$ – разрядность входной кодовой комбинации,

$n_{вх}(a_1) = n_{вх}(a_2) = \ldots = n_{вх}(a_k) = n$ – для равномерного входного кода;

$n_{вых}(a_i)$ – разрядность выходной кодовой комбинации Хаффмана для символа a_i;

$l_{служ}$ – длина служебной части, необходимой для декодирования.

Условие (2) говорит о том, что при сжатии информации между идущими последовательно двумя модификациями дерева суммарное количество битов, пришедшее на вход системы сжатия, должно быть больше объема сжатых данных и служебной информации об изменении модели.

Следствием условия (2) является то, что чем больше вес пришедшего символа, тем меньше времени затрачивается на кодирование и корректировку дерева Хаффмана.

Условия (1) и (2) применяются в следующей последовательности: при поступлении очередного кода символа на вход системы происходит модификация статистики, которая может привести к тому, что условие (1) станет верно. Условие (1) сигнализирует о начале перестройки дерева Хаффмана. При этом генерируется служебная часть, что может внести избыточность, которая не компенсируется устраненной избыточностью. Для того чтобы этого не случилось, проверяется условие (2): символ, вызвавший модификацию, представляется в виде кодовой комбинации Хаффмана, далее по собранным данным вычисляется правая и левая части неравенства и производится сравнение. Если условие (2) выполняется, то происходит модификация модели, если нет – вводится следующий символ и повторяется проверка условий (1) и (2).

ВЫВОДЫ

Таким образом, применение рассмотренных критериев построения системы сжатия данных на основе адаптивного алгоритма построения кода Хаффмана позволяет оптимизировать и упростить реализацию данного устройства на практике. Полученные результаты могут быть

наиболее востребованы для именно адаптивных систем и позволяют понизить алгоритмическую избыточность при кодировании и повысить качество сжатия, существенно улучшив скорость и степень сжатия.

Рассмотренный в статье вариант реализации устройства построения кода Хаффмана может применяться в любых системах построения неравномерных кодов, где есть необходимость постоянно изменять кодовое отображение в соответствии с источником информации. Предложенный вариант синтеза данного устройства возможен как в виде специализированного устройства, так и на основе ПЛИС.

SUMMARY

The adaptive encoding algorithms are essentially effective for data compression. The adaptive (dynamic) Huffman method allows to achieve the most high compression rate and sufficiently good processing speed. Hardware representation of adaptive Huffman algorithm is a main goal of this article. Received model and research results can be used for hardware implementation of the adaptive Huffman encoding algorithm.

СПИСОК ЛИТЕРАТУРЫ

1. Жураковський Ю.П., Полторак В.П. Теорія інформації та кодування: Підручник. – К.: Вища шк., 2001. – 255 с.:іл..
2. Ватолин Д., Ратушняк и др. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М.: ДИАЛОГ-МИФИ, 2002. – 348 с.
3. Зубань Ю.А., Петров В.В. Цифровой автомат адаптивного построения кода Хаффмана //Вісник СумДУ. Технічні науки. – 2005. - №9(81). – С.100-106.
4. Зубань Ю.А., Петров В.В. Адаптивная система сжатия на основе кода Хаффмана /Тези науково- технічної конференції викладачів, співробітників, аспірантів і студентів фізико-технічного факультету. - Суми: Вид-во СумДУ, 2006. – 150 с.

Поступила в редакцию 12 мая 2006 г.