УДК 681.3

К ПРОГРАММНОЙ РЕАЛИЗАЦИИ АЛГОРИТМА СЖАТИЯ ОБЛАСТИ ОПРЕДЕЛЕНИЯ ЛОГИЧЕСКИХ ФУНКЦИЙ

Е.Н. Коробкова[1], А.С. Шпак[2]
([1]Белгородский государственный технологический университет им. В.Г. Шухова, Россия, [2]Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков)

Рассмотрен алгоритм сжатия области определения логических функций, позволяющий представить их в форме обобщенных функций (ОФ) с зависимыми параметрами, отличительной чертой которых является, то, что значение их в точках области определения равно не только значению 0 и 1, но и любой из функций от других переменных. Разработаны программные средства реализации алгоритма.

алгоритм сжатия, область определения логических функций

Постановка проблемы. В связи с появлением и бурным развитием нового класса микросхем – программируемых логических интегральных схем (ПЛИС), внедрением их в практику проектирования и производства цифровых устройств вновь стали актуальными вопросы, связанные с синтезом структур [1], контролем и диагностикой их функционирования [2]. При решении как первого, а также второго класса задач, возникает необходимость представления области определения логических функций, число точек которых резко возрастает с увеличением числа переменных, определяющих эти функции, что ведет к усложнению алгоритмов решения задач в целом. Возникает проблема сжатия области определения функций, обеспечивающая возможность представления их в области с меньшим числом элементов.

Анализ исследований и публикаций. Число работ, посвященных синтезу цифровых устройств, контролю и диагностике их так велико, что даже простое перечисление их представляет собой далеко не тривиальную задачу [1, 2]. Поэтому мы ограничимся только лишь замечанием, что в большинстве работ основным представлением функций в точках области определения являются константы ноль и единица. Некоторые работы посвящены представлению функции в многозначном структурном алфавите [1], но опять в виде констант его. И только лишь в работах, посвященных синтезу цифровых устройств на мультиплексорах [3], вводится понятие мультиплексных функций, которые можно трактовать как разновидность обобщенных логических функций. В работах [4 – 6] представлено и исследовано несколько версий алгоритма сжатия области определения традици-

онных логических функций, позволивших представить эти функции в форме обобщенных с последующей многоверсионной минимизацией [6 – 8], обеспечивающей не только упрощение процедуры, но и автоматический контроль достоверности получаемых результатов, а также упрощение алгоритма нахождения булевых производных [9 – 11], лежащих в основе одного из перспективных способов контроля и диагностики [2]. Однако, в отмеченных работах алгоритм не доведен до программной его реализации.

Цель работы и постановка задачи исследования. Цель работы – продолжить исследования, начатые в [4 – 11], и решить задачу разработки программных средств сжатия области определения логических функций, обеспечивающих возможность автоматизировать представление их в форме обобщенных.

Решение задачи. *Описание алгоритма и его блок-схема.* Предложенный и исследованный в [4 – 6] алгоритм сжатия области определения функций от переменных множества X, основанный на разбиении этого множества на два подмножества X1 Є X и X2 = X\X1, был реализован в среде разработки Delphi 7 [12, 13].

Каждое n-разрядное двоичное число, соответствующее единичному набору, можно рассматривать как два двоичных числа, одно из которых образовано битами переменных подмножества X1 (блок 1 на рис. 1), а второе – битами переменных подмножества X2 (блок 2 на рис. 1). Если в подмножество X1 включены k переменных с младшими индексами (блок 3 на рис.1), то для нахождения координат точек сжатой области определения и значения функции в них достаточно проставить точку между разрядами подмножества переменных X1 и X2, что соответствует делению исходного набора на 2^k (блок 6 на рис.1). При этом двоичное число, образованное n – k старшими переменными, соответствующее частному (целому), дает номер точки в сжатой области определения, а двоичное число, образованное k младшими разрядами переменных, соответствующее остатку от деления, дает индекс минтерма, образуемого переменными подмножестве X1, определяющего значение функции в этой точке (блок 7 на рис. 1).

Если в подмножество X1 включены k переменных со старшими индексами, то для нахождения координат точек сжатой области и значения функции в них достаточно n-разрядное двоичное число разделить на 2^{n-k} (блок 4 на рис. 1). Остаток от деления, образованный n – k младшими разрядами дает номер точки в сжатой области, а частное, образованное k старшими разрядами, дает индекс минтерма, определяющего значение функции в этой точке (блок 5 на рис. 1).

Поскольку в одно из подмножеств могут входить любые переменные множества X и остальные в другое, то для большинства вариантов разбиения переменных на вторичные и первичные биты двоичного числа, определяю-

щего точку сжатой области, в исходной области будут “перемешаны” (чередоваться) с битами двоичного числа, определяющего индекс минтерма. Для удобства работы с исходными n-разрядными и полученными k- и (n – k)-разрядными числами рекомендуется эти двоичные числа представлять в таблице (блок 8 на рис. 1). Таблица содержит (n – k) столбцов для представления двоичного числа, определяющего координату точки в сжатой области определения и k столбцов для представления двоичного числа, определяющего минтерм, образуемый переменными подмножества X1. Если (n – k)- и k-разрядные двоичные числа представить их десятичными эквивалентами, то таблицу можно упростить.

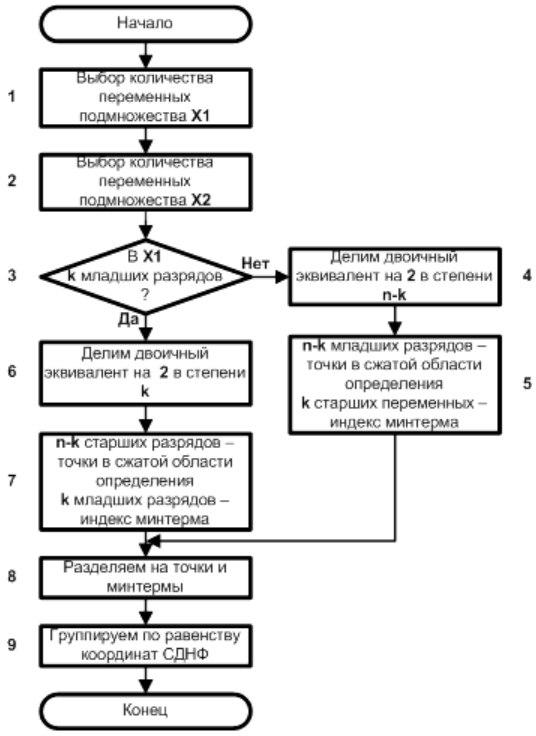

Рис. 1. Блок-схема алгоритма

При таком представлении первое десятичное число дает номер (координату) ненулевой точки в сжатой области определения, а второе – индекс минтерма, определяющего значение функции в ней. Полученные таким образом пары десятичных чисел объединяют в группы, с одинаковым значением первого числа, которое указывает на номер (координату) ненулевой точки в сжатой области, а вторые числа в этой группе дают индексы минтермов, логическая сумма которых есть СДНФ (блок 9 на рис. 1) функции, определяемой переменными подмножества X1 в этой точке сжатой области определения.

Блок-схема алгоритма программной реализации и ее описание. Работа алгоритма начинается с выбора из списка количества переменных базового множества X (блок 1 на рис. 2). После выбора число помещается в переменную Razr. Следует заметить, что при формировании списка нужно учитывать тот факт, что при слишком большом числе переменных множества X, координаты точек области определения могут не поместиться в объявленную переменную. Для решения этой проблемы нужно формировать список с возможными вариантами количества переменных, либо, если органи-

зован какой-либо другой способ ввода этой величины, нужно выполнять проверку на ввод корректного значения. Также можно объявить переменную, хранящую количество точек области определения, большим объемом, но в этом случае также остается актуальной проблема переполнения.

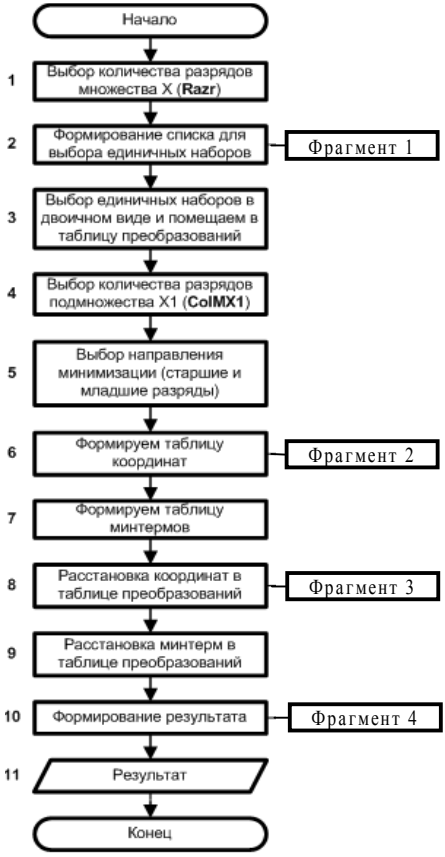

Рис. 2. Блок-схема алгоритма для программной реализации

Вычисление количества координат (Rows) производится по формуле 2^{Razr}. Вычисление этой величины начинается с присвоения переменной Rows значения 1. Следующим шагом является организация цикла с количеством проходов от 1 до Razr – 1, в теле которого выполняется операция Rows= Rows*2. Таким образом, после работы цикла, в переменной Rows находится искомое значение.

Следующий шаг представляет собой выбор единичных наборов (блок 3 на рис. 2) из списка. В списке наборы представлены в двоичном виде. При их выборе они удаляются из списка и переносятся в таблицу преобразований для дальнейшей работы с ними. По сути, все наборы в таблице преобразований представляют собой ячейки карты Карно или Вейча с единичным значением.

Процесс формирования списка для выбора единичных наборов (блок 2 на рис. 2) начинается с расстановки десятичных значений координат наборов (рис. 3). Это действие осуществляется в цикле с количеством проходов от 0 до Rows – 1 и в теле цикла производится помещение в список текущего значения счетчика цикла.

После этого начинается часть, формирующая двоичный эквивалент координат наборов и состоящая из двух циклов. Цикл 1 (цикл 1 на рис. 3) производит перебор переменных от младших разрядов к старшим. Помимо этого, в теле этого цикла вычисляется шаг смены значения k, т.е. то значение, при достижении которого будет меняться вводимое значение с 0 на 1 и на-

оборот. Шаг смены значения равен 2^L, где L – это индекс текущей переменной. Цикл 2 (цикл 2 на рис. 3) производит непосредственное заполнение списка текущим значением переменной (s). Ввод выполняется до тех пор, пока не будет введено количество значений, предусмотренное шагом смены значений (k). Для этого выполняется проверка делимости нацело текущей позиции цикла 2 на k. Если это условие выполняется, то следующим действием будет проверка на вводимый символ, в зависимости от результатов которой будет выполняться смена вводимого в список значения. Если же условие делимости не выполняется, то производится ввод текущего значения s в список.

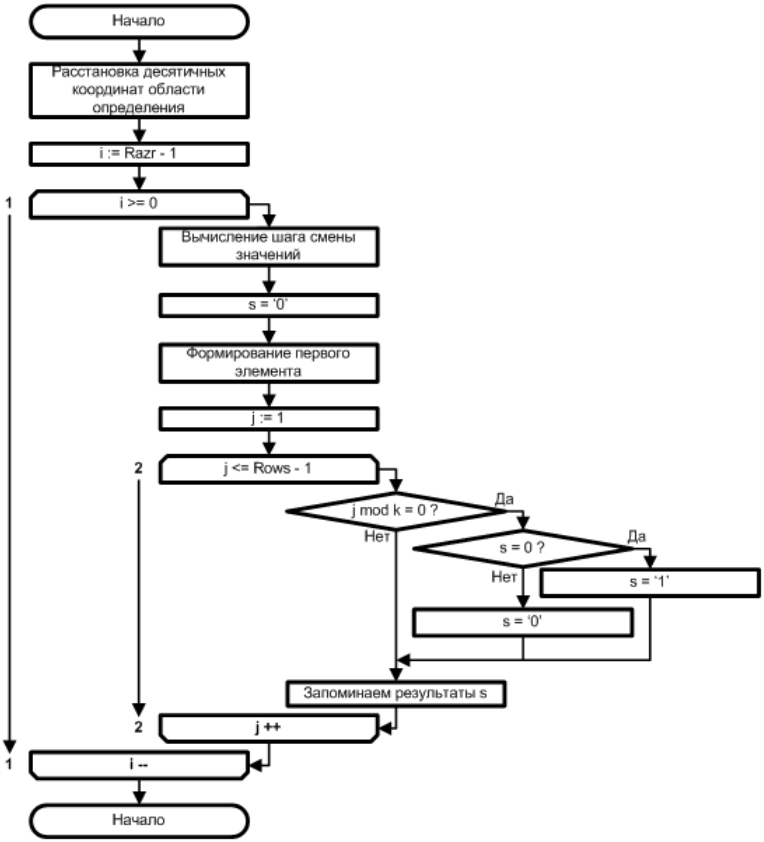

Рис. 3. Блок-схема формирования списка для выбора единичных наборов

Сам процесс формирования списка для выбора единичных наборов не является сложным, но при количестве переменных больше чем 10 начинает значительно замедлять работу программы. Решение этой проблемы также является простым, для этого необходимо заменить список строкой ввода (в этом случае исчезнет из алгоритма блок 2 на рис. 2),

поскольку алгоритм направлен на работу только с единичными наборами, а в списке большинство наборов остаются невостребованными. Как следствие из вышеописанного, следует вводить только те значения, которые задействованы в алгоритме. Для этого при работе со строкой ввода описывается фрагмент для отсеивания всех символов, кроме цифр. При считывании сформированного набора производится проверка на вхождение в диапазон от 0 до Rows – 1 и набор помещается в таблицу преобразований, если это условие выполняется.

В таблице преобразований наборы находятся в десятичном виде, следовательно, их нужно преобразовать в двоичный. Это действие выполняется при помощи деления исходного числа на 2. Деление производится нацело с запоминанием остатка после каждой операции. Таким образом мы выполняем это действие до тех пор, пока исходное число не станет равным 1. Затем считываем все остатки в обратном порядке с учетом последней единицы. Полученный набор нулей и единиц будет двоичным эквивалентом десятичного числа. После выбора единичных наборов осуществляется выбор количества переменных (блок 4 на рис. 2) подмножества X1 (ColMX1) и вычисляется количество значений (RowMX1), которые могут быть образованы переменными данного подмножества. Значение переменной RowMX1 рассчитывается по формуле 2^{ColMX1}.

На этом же шаге выполняется выбор направления минимизации (блок 5 на рис. 2), т.е. в зависимости от выбранного направления в подмножество X1 включаются ColMX1 младших переменных, либо ColMX1 старших. В зависимости от этого выбора производятся вычисления:

ColMX1 := считанное значение;

ColMX2 := Razr – ColMX1, при младших разрядах;

ColMX2 := считанное значение;

ColMX1 := Razr – ColMX2, при старших разрядах.

Следующим шагом является формирование таблицы координат (блок 6 на рис. 2). Весь фрагмент представляет собой несколько действий (рис. 4). Первым действием выполняется расстановка переменных таблицы в цикле с количеством проходов от ColMX1 до 1. В зависимости от выбранного направления вычисляются индексы переменных, лежащие в основе формирования таблицы. Затем следует расстановка десятичной нумерации в этой же таблице. Происходит это также в обратном цикле от RowMX1 до 1. После завершения работы цикла строки таблицы будут пронумерованы, начиная с 0 до значения RowMX1 – 1.

После внешнего оформления таблицы выполняется непосредственное заполнение таблицы значениями нуля либо единицы. Этот шаг состоит из двух циклов. В цикле 1 (цикл 1 на рис. 4) выполняется перебор переменных таблицы. Первое действие в теле цикла – вычисление шага смены значений в

зависимости от позиции, на которой находится данная переменная. Затем вступает в работу вложенный цикл 2 (цикл 2 на рис. 4), осуществляющий передвижение по ячейкам столбца с текущей переменной и расстановку значения переменной на данном наборе. Следует заметить, что это действие является идентичным, описанному на рис. 3, за исключением того факта, что вышеописанный фрагмент работает со списком, а текущий – с таблицей.

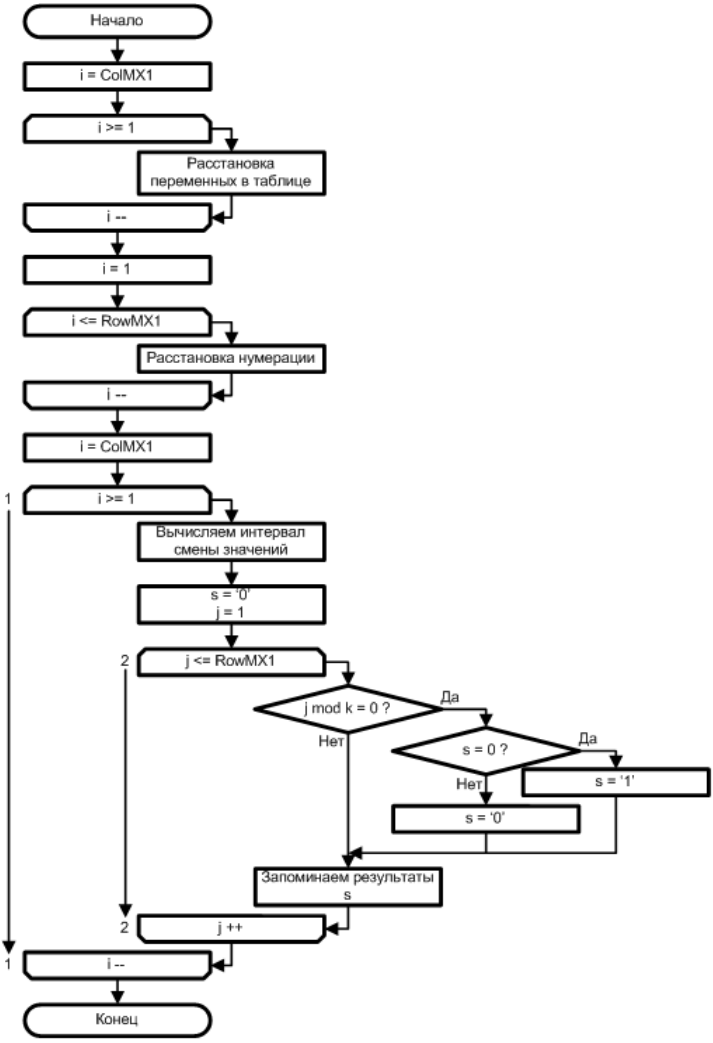

Рис. 4. Блок-схема формирования таблицы координат

Таким же образом осуществляется формирование таблицы минтерм (блок 7 на рис. 2). При этом учитывается тот факт, что для работы с

этим компонентом используется другое подмножество и другие переменные множества X.

Особенностью этого алгоритма является расстановка точек между двумя подмножествами в двоичных эквивалентах координат единичных наборов. Это действие предназначено для дальнейшего упрощения работы, поскольку возникнет потребность разделить в таблице преобразований двоичный эквивалент на минтерму и координату, а затем, при помощи таблиц минтерм и координат, поместить их десятичные значения в соответствующие ячейки таблицы. При разделении ставится точка в положении ColMX2 – это будет правильно для обоих случаев.

Теперь алгоритм приступает к работе с таблицей преобразования, а если точнее, выполнят расстановку новых координат (рис. 5).

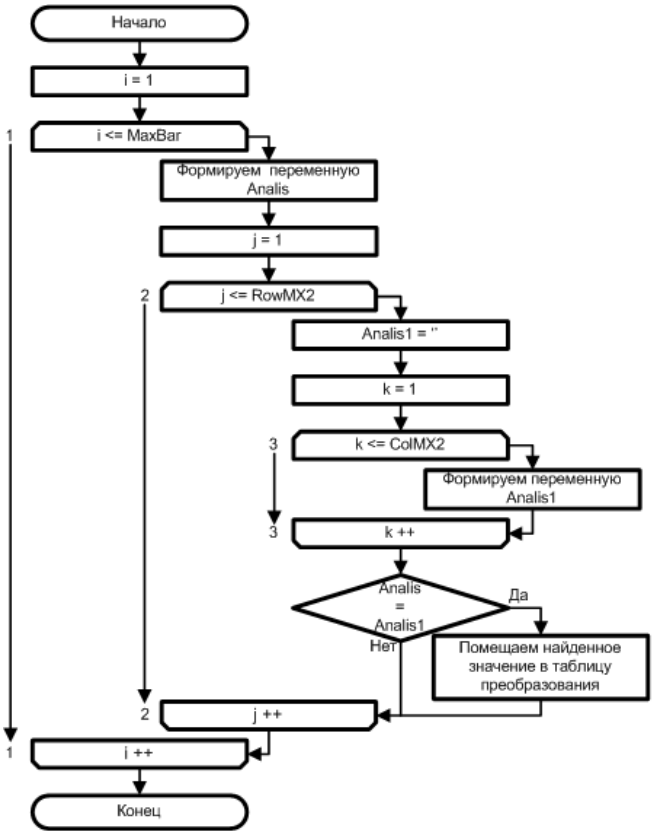

Рис. 5. Блок-схема расстановки координат в таблице преобразований

Процесс начинается с цикла 1 (цикл 1 на рис. 5), выполняющий перебор всех строк таблицы. В теле цикла формируется переменная Analis, фак-

тически в нее помещается та часть двоичного числа, которая представляет собой координату. После этого осуществляется вход в цикл 2 (цикл 2 на рис. 5), который отвечает за поиск значения Analis в таблице координат. При входе в цикл 2 производится подготовка цикла 3 (цикл 3 на рис. 5), который производит формирование переменной Analis1. По выходу из цикла 3 производится проверка идентичности содержимого переменных Analis и Analis1. Если они равны, то найденное десятичное значение помещается в соответствующую ячейку таблицы и алгоритм приступает к обработке следующей ячейки. Таким же образом выполняется заполнение таблицы преобразований минтермами. Следует учитывать лишь тот факт, что в переменную Analis помещается часть двоичного числа, которая является минтермом. Также поиск производится в таблице минтермов.

Теперь остается лишь сформировать окончательный результат работы алгоритма (блок 10 на рис. 2). Работа данной части алгоритма начинается с цикла 1 (цикл 1 на рис. 6), специализация которого заключается в последовательном передвижении по таблице координат. В теле цикла инициализируются начальные значения переменных Buf, Count и s. Затем наступает очередь цикла 2 (цикл 2 на рис. 2), который производит поиск в таблице преобразований всех координат, равных искомой. Если при поиске найдено нужное значение, то формируется результат в переменной s и увеличивается содержимое счетчика Count на 1. Таким образом, сделав перебор всех значений координат по таблице преобразований, выполняется коррекция результата.

Коррекция начинается со сравнения Count с нулем и RowMX2. При условии, что Count равно нулю, результату данной точки присваивается ноль. Если же Count равен RowMX2, то результату данной точки присваивается единичное значение. Если же Count находится между 0 и RowMX2, то корректируется результат, хранящийся в переменной s, и помещается в текущую точку новой области определения. В конечном итоге результат запоминается, и фрагмент переходит к поиску следующей координаты в таблице преобразований.

По окончанию формирования результата возникает потребность вывода (блок 11 на рис. 2) его на экран, либо организовывается какой-либо иной способ вывода информации.

Выводы: 1. Проведен анализ алгоритма сжатия области определения логических функций, позволяющий представить их в форме обобщенных функций с зависимыми параметрами. Разработаны программные средства реализации алгоритма.

2. Научная новизна полученных результатов, состоит в том, что впервые проведено исследование предложенного алгоритма, позволившее разработать программные средства его реализации.

3. Практическая значимость результатов исследования состоит в том, что они позволяют не только упростить процедуру последующей минимизации логических функций, но и обеспечить достоверность получаемых результатов. Кроме того, приложение полученных в работе результатов к нахождению различных типов булевых производных, используемых при тестовом контроле, упрощает процедуру нахождения.

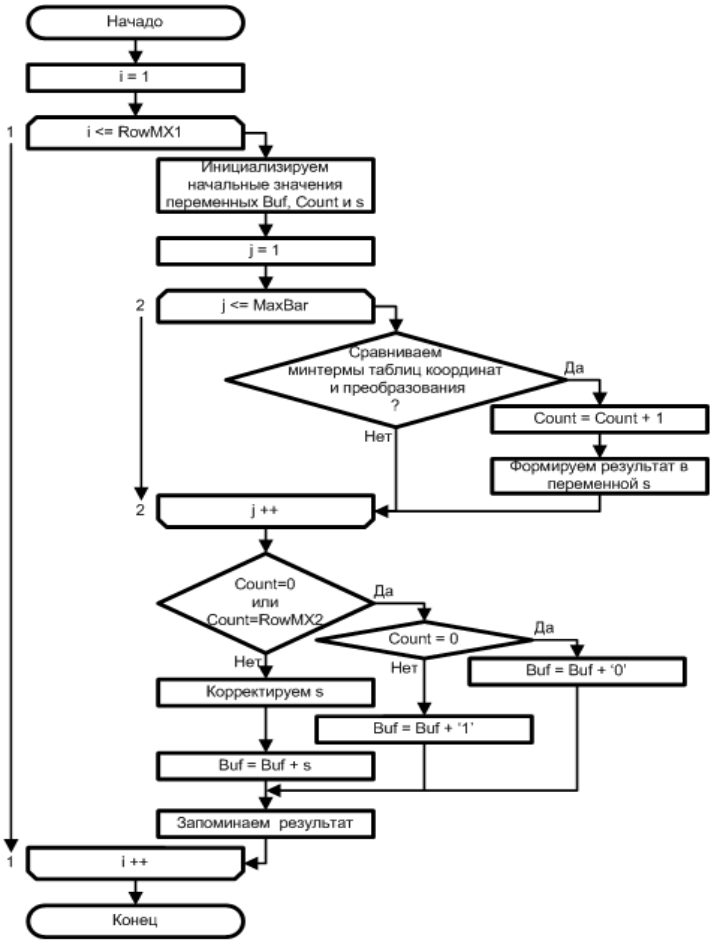

Рис. 6. Блок-схема формирования результата

4. В соответствии с приведенным описанием алгоритма была разработана и отлажена программа и решены контрольные примеры сжатия области определения функции от 4, 5, 6, 7 и 8 переменных. Результаты полностью совпадали с аналогичными, полученными «ручным» способом. Ограничение числа переменных до 8 было продиктовано только

лишь громоздкостью «ручной» реализации алгоритма. Предложенный в работе программный способ позволяет реализовать сжатие области определения функций от 30 переменных.

ЛИТЕРАТУРА

1. *Соловьев В.В. Проектирование цифровых систем на основе программируемых логических интегральных схем. – М.: Телеком, 2001. – 636 с.*
2. *Проектирование и диагностика компьютерных систем и сетей / М.Ф. Бондаренко, Г.Ф. Кривуля, В.И. Хаханов и др. – К.:НМЦВО, 2000. – 306 с.*
3. *Угрюмов Е.П. Цифровая схемотехника. – С.-Пб.: БХВ, 2000. – 528 с.*
4. *Рубанов В.Г., Коробкова Е.Н. Разработка алгоритма сжатия области логических функций // Труды современного гуманитарного университета. Белгородский филиал. – Белгород, 2000. – Вып. 18. – С. 105-112.*
5. *Коробкова Е.Н. Анализ табличного алгоритма сжатия области определения полностью определенных логических функций // Открытые информационные и компьютерные интегрированные технологии: Сб. научн. трудов. – Х.: НАКУ «ХАИ», 2003. – Вып. 18. – С. 177-186.*
6. *Коробкова Е.Н. Два способа сжатия области определения логических функций и их приложение к нахождению минимальных ДНФ // Открытые информационные и компьютерные интегрированные технологии. – Х.: НАКУ «ХАИ», 2003. – Вып. 19. – С. 245-255.*
7. *Рубанов В.Г., Коробкова Е.Н. Графо-аналитический метод нахождения минимальных дизъюнктивных нормальных форм логических функций // Системи обробки інформації. – Х.: ХВУ. – 2002. – Вып. 3 (19). – С. 46-53.*
8. *Коробкова Е.Н. Графо-аналитический метод минимизации полностью определенных логических функций в сжатых картах // Системи обробки інформації. – Х.: ХВУ. – 2002. – Вип. 6 (22). – С. 288-298.*
9. *Коробкова Е.Н. О применении метода сжатия области определения логических функций к нахождению булевых производных // Открытые информационные и компьютерные интегрированные технологии: Сб. научн. трудов. – Х.: НАКУ «ХАИ». – 2003. – Вып. 18. – С. 177-186.*
10. *Коробкова Е.Н. Об одном алгоритме нахождения векторных булевых производных // Інформаційно-керуючі системи на залізничному транспорті. – Х.: 2003. – Вып. 6. – С. 3-7.*
11. *Рубанов В.Г., Коробкова Е.Н. Об одном способе нахождения булевых функций чувствительности // Радіоелектронні і комп'ютерні системи. – Х.: НАКУ «ХАИ», 2003. – Вып. 4. – С. 139-144.*
12. *Фаронов В.В. Delphi. Программирование на языках высокого уровня. – С.-Пб: Питер, 2004. – 639 с.*
13. *Фленов М.Е. Библия Delphi. – С.-Пб: Питер, 2004. – 631 с.*

Поступила 18.01.2006

Рецензент: доктор технических наук, профессор В.С. Харченко,
Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ».