ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ
LOGICAL DESIGN

УДК 519.717
https://doi.org/10.37661/1816-0301-2022-19-2-26-55

Оригинальная статья
Original Paper

Экспериментальное сравнение эффективности программ минимизации систем булевых функций в классе дизъюнктивных нормальных форм

П. Н. Бибило✉, И. П. Логинова

Объединенный институт проблем информатики
Национальной академии наук Беларуси,
ул. Сурганова, 6, Минск, 220012, Беларусь
✉*E-mail: bibilo@newman.bas-net.by*

Аннотация

Цели. Методы, алгоритмы и программы решения задач минимизации дизъюнктивных нормальных форм (ДНФ) представлений булевых функций широко используются при проектировании цифровых систем для уменьшения сложности (площади кристаллов) функциональных комбинационных блоков, размещаемых в составе цифровых СБИС.

Целью работы является сравнение программ, входящих в отечественную систему FLC-2 логической оптимизации, с двумя широко известными и свободно распространяемыми зарубежными программами минимизации Espresso IIC и ABC.

Методы. Для сравнения программ использованы четыре набора примеров входных данных: широко известные примеры, на которых проверялась эффективность программы Espresso IIC, псевдослучайные системы ДНФ и два набора промышленных примеров из практики проектирования логических схем. Предложены программные средства для применения программ совместной минимизации при раздельной минимизации функций. Разработаны алгоритмы и программы распараллеливания вычислений при раздельной минимизации функций в классе ДНФ.

Результаты. Выявлены области предпочтительного использования и время работы программ для исходных (минимизируемых) систем функций, характеризуемых большими значениями параметров (десятками аргументов и функций, десятками тысяч элементарных конъюнкций) и различными формами задания входных данных. Изучена эффективность применения программ минимизации для различных форм задания входных данных: ДНФ, ортогонализованных ДНФ, BDD-представлений систем функций, таблиц истинности и систем совершенных ДНФ.

Заключение. Результаты экспериментов показывают эффективность параллельных программ. Они позволяют сокращать время вычислений и увеличивать размерности решаемых задач раздельной минимизации систем булевых функций.

Ключевые слова: система булевых функций, дизъюнктивная нормальная форма (ДНФ), кратчайшая ДНФ, раздельная минимизация функций, кратчайшая система ДНФ, совместная минимизация функций, BDD-представления, стандарт OpenMP

Для цитирования. Бибило, П. Н. Экспериментальное сравнение эффективности программ минимизации систем булевых функций в классе дизъюнктивных нормальных форм / П. Н. Бибило, И. П. Логинова // Информатика. – 2022. − Т. 19, № 2. – С. 26–55. https://doi.org/10.37661/1816-0301-2022-19-2-26-55

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Поступила в редакцию | Received 20.01.2022
Подписана в печать | Accepted 28.02.2022
Опубликована | Published 29.06.2022

Experimental comparison of the effectiveness of programs for minimizing systems of Boolean functions in the class of disjunctive normal forms

Petr N. Bibilo✉, Irina P. Loginova

The United Institute of Informatics Problems
of the National Academy of Sciences of Belarus,
st. Surganova, 6, Minsk, 220012, Belarus
✉*E-mail: bibilo@newman.bas-net.by*

Abstract

Objectives. Methods, algorithms and programs for solving problems of minimizing the DNF representations of Boolean functions are widely used in the design of digital systems to reduce the complexity (crystal area) of functional combinational blocks of digital systems placed into digital VLSI.

The objective of the work is experimental comparison of domestic programs for minimizing Boolean functions in the DNF class included in the FLC-2 with two well-known foreign freely distributed programs for minimizing DNF known as Espresso IIC and ABC.

Methods. Four sets sample of input data were used to compare the programs – there are widely known examples on which the effectiveness of the Espresso IIC program was tested and two sets of industrial examples from the practice of designing the logic circuits. Algorithms and programs for parallelization of calculations when separate functions of minimizing have been developed. Software tools for the application of joint minimization programs with separate minimization of functions are proposed.

Results. The areas of preferred use and the execution time of programs for the source systems of functions (for minimization) characterized by large parameter values of dozens of arguments and functions, tens of thousands of elementary conjunctions are revealed. The efficiency of application of minimization programs for various forms of input data assignment is investigated – DNF, orthogonalized DNF, BDD (Binary Decision Diagrams) representations for systems of functions, truth tables and perfect DNF systems.

Conclusion. The experimental results show the effectiveness of parallel programs – reducing the calculation time and increasing the dimensions of solved problems of separate minimization of Boolean function systems.

Keywords: system of Boolean functions, disjunctive normal form (DNF), shortest DNF, separate minimization of functions, the shortest DNF system, joint minimization of functions, Binary Decision Diagram (BDD), OpenMP standard

For citation. Bibilo P. N., Loginova I. P. *Experimental comparison of the effectiveness of programs for minimizing systems of Boolean functions in the class of disjunctive normal forms*. Informatika [*Informatics*], 2022, vol. 19, no. 2, pp. 26−55 (In Russ.). https://doi.org/10.37661/1816-0301-2022-19-2-26-55

Conflict of interest. The authors declare of no conflict of interest.

Введение. Задача минимизации представлений булевых функций и систем в классе ДНФ является классической задачей теории булевых функций, начало исследований положено в работах Квайна [1] и МакКласки [2]. Как утверждается в статье [3], число публикаций по данной проблематике превышает тысячу. Например, в обзоре [4] процитировано 217 работ.

Задачу минимизации ДНФ-представлений в учебной литературе часто называют задачей минимизации булевых функций. Эта задача имеет многочисленные практические применения, среди которых, пожалуй, важнейшее место занимает ее использование в качестве средства технологически независимой логической оптимизации при синтезе логических схем [5]. Минимизация ДНФ одной булевой функции традиционно применялась (и применяется в настоящее время) в качестве метода глобальной оптимизации – начального этапа синтеза одновыходных комбинационных схем. В случае многовыходных комбинационных схем решается задача совместной либо раздельной минимизации системы булевых функций. Критериями оптимизации выступает число элементарных конъюнкций либо число литералов в минимизированных системах ДНФ. Появление программируемых логических матриц (ПЛМ) – функциональных блоков заказных СБИС – потребовало развития методов и программ совместной минимизации систем булевых функций, применяемых для сокращения площади ПЛМ [6]. Появление схем FPGA (от англ. Field-Programmable Gate Array, программируемая пользователем вентильная матрица) привело к развитию методов декомпозиции и оптимизации многоуровневых BDD-представлений (BDD – Binary Decision Diagram, бинарная диаграмма решений, диаграмма двоичного выбора) на основе разложения Шеннона [7]. Размерности задач проектирования комбинационной логики заказных сверхбольших интегральных схем (СБИС) и систем-на-кристалле возрастают так, что при логической оптимизации исходное функциональное описание часто требуется разбивать на блоки, подвергаемые оптимизации различными программами. Эксперименты [8] показали, что при синтезе комбинационных блоков заказных СБИС в базисе библиотечных элементов для некоторых систем функций в качестве технологически независимой оптимизации более эффективна двухуровневая минимизация в классе ДНФ, для других – многоуровневая BDD-оптимизация. Минимизация используется для сокращения сложности функциональных описаний подсистем функций при их декомпозиции [8], а также как начальный этап оптимизации многоуровневых представлений систем функций, после которого следует применение факторизационных методов, позволяющих получать алгебраические скобочные представления [5].

В настоящей работе проводится экспериментальное сравнение отечественных программ минимизации булевых функций в классе ДНФ, входящих в систему логической оптимизации FLC-2 [9], с зарубежными, свободно распространяемыми программами Espresso IIC и ABC. Предлагаются средства распараллеливания как исходных данных, так и программ, что в целом позволяет обеспечивать сокращение времени решения задач минимизации функций и увеличивать размерности решаемых задач.

Основные определения и постановки задач. *Булевыми* называются двоичные (0, 1) функции $f(\boldsymbol{x}) = f(x_1, \ldots, x_n)$ двоичных (булевых) переменных $x_1, x_2, \ldots, x_n$. Пусть V^x – булево пространство, построенное над переменными булева вектора $\boldsymbol{x} = (x_1, \ldots, x_n)$. Элементами этого пространства являются n-компонентные наборы (векторы) $\boldsymbol{x}^*$ нулей и единиц. Булева функция, значения 0, 1 которой определены на всех элементах $\boldsymbol{x}^* \in V^x$, называется *полностью определенной*. Любая полностью определенная булева функция (кроме константы 0) может быть задана в виде ДНФ, т. е. в форме дизъюнкции элементарных конъюнкций. *Элементарная конъюнкция* – это конъюнкция литералов (булевых переменных x_i либо их инверсий $\overline{x}_i$). Литерал x_i называется *положительным*, литерал $\overline{x}_i$ – *отрицательным*. Если в ДНФ каждая элементарная конъюнкция является полной, т. е. содержит литералы всех переменных, то такая ДНФ называется *совершенной* (СДНФ). Систему булевых функций будем обозначать $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$, значениями системы функций на элементах $\boldsymbol{x}^*$ булева пространства являются m-компонентные булевы векторы $\boldsymbol{f}(\boldsymbol{x}^*)$. Рассмотрим матричные формы задания ДНФ и СДНФ систем булевых функций.

В матричной форме система ДНФ, представляющая систему полностью определенных функций $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), f_3(\boldsymbol{x}), f_4(\boldsymbol{x}))$, $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$, задается парой матриц (табл. 1). Строки троичной матрицы T^x представляют элементарные конъюнкции (троичные векторы – интервалы булева пространства V^x), а единичные значения элементов в булевой матрице B^f отмечают вхождения соответствующих конъюнкций в ДНФ функций:

$$D^1_{f_1} = \bar{x}_2 x_3 \vee x_4;\ \ D^1_{f_2} = \bar{x}_1 x_2 x_3 \vee x_1 \bar{x}_2 x_3 \vee \bar{x}_1 x_4;\ D^1_{f_3} = \bar{x}_1 x_2 x_3 \vee x_1 x_2 \bar{x}_3 \vee x_2 x_3 x_4 \vee \bar{x}_3 \bar{x}_4;$$
$$D^1_{f_4} = x_1 x_2 \bar{x}_3 \vee x_2 \bar{x}_4 \vee \bar{x}_3 \bar{x}_4.$$

Таблица 1
Матричная форма системы ДНФ

Table 1
Matrix form of the DNF system

T^x	B^f
$x_1\ x_2\ x_3\ x_4$	$f_1\ f_2\ f_3\ f_4$
– 0 1 –	1 0 0 0
0 1 1 –	0 1 1 0
– – – 1	1 0 0 0
1 0 1 –	0 1 0 0
1 1 0 –	0 0 1 1
– 1 – 0	0 0 0 1
– 1 1 1	0 0 1 0
0 – – 1	0 1 0 0
– – 0 0	0 0 1 1

Далее будут использоваться следующие обозначения:

n – число аргументов (число компонент булева вектора $\boldsymbol{x} = (x_1, \ldots, x_n)$);

m – число функций системы $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$;

k – число общих элементарных конъюнкций, на которых заданы ДНФ всех функций системы $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$;

k_i – число элементарных конъюнкций в задании компонентной функции f_i системы;

$\sum_{i=1}^{m} k_i$ – суммарное число элементарных конъюнкций в ДНФ всех функций системы.

Для системы ДНФ из табл. 1 заданы следующие значения параметров: $n = 4$, $m = 4$, $k = 9$, $k_1 = 2$, $k_2 = 3$, $k_3 = 4$, $k_4 = 3$, $\sum_{i=1}^{m} k_i = 2 + 3 + 4 + 3 = 12$ (число единиц в булевой матрице B^f).

Матричная форма системы СДНФ той же системы функций $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), f_3(\boldsymbol{x}), f_4(\boldsymbol{x}))$ представлена в табл. 2 и состоит из пары булевых матриц. Матрица B^x задает полные элементарные конъюнкции, матрица B^f – их вхождения в СДНФ компонентных функций системы $\boldsymbol{f}(\boldsymbol{x})$. Для многих программ минимизации в целях более компактного представления исходных данных наборы, на которых $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$, обычно не указываются. *Таблица истинности* отличается от матричной формы системы СДНФ тем, что в матрице B^x содержатся все 2^n полных элементарных конъюнкций, поэтому в матрице B^f могут оказаться нулевые строки. Заметим, что таких строк в табл. 2 нет. Кроме того, она является таблицей истинности – содержит все 2^4 двоичных наборов, соответствующих всем 16 полным элементарным конъюнкциям от четырех переменных.

Кратчайшей ДНФ D_f булевой функции f называется ДНФ минимальной длины, т. е. ДНФ с минимальным числом элементарных конъюнкций. Под длиной ДНФ булевой функции понимается число ее элементарных конъюнкций.

В табл. 3 приведены кратчайшие ДНФ функций из табл. 2.

Таблица 2
Таблица истинности системы полностью определенных булевых функций

Table 2
Truth table of a system of fully defined Boolean functions

x_1 x_2 x_3 x_4	f_1 f_2 f_3 f_4
0 0 0 0	0 0 1 1
0 0 0 1	1 1 0 0
0 0 1 0	1 0 0 0
0 0 1 1	1 1 0 0
0 1 0 0	0 0 1 1
0 1 0 1	1 1 0 0
0 1 1 0	0 1 1 1
0 1 1 1	1 1 1 0
1 0 0 0	0 0 1 1
1 0 0 1	1 0 0 0
1 0 1 0	1 1 0 0
1 0 1 1	1 1 0 0
1 1 0 0	0 0 1 1
1 1 0 1	1 0 1 1
1 1 1 0	0 0 0 1
1 1 1 1	1 0 1 0

Таблица 3
Раздельно минимизированные ДНФ системы функций (табл. 2)

Table 3
Separately minimized DNF systems of functions (table 2)

x_1 x_2 x_3 x_4	f_1 f_2 f_3 f_4
1 1 0 –	0 0 0 1
– 1 – 0	0 0 0 1
– – 0 0	0 0 0 1
1 1 – 1	0 0 1 0
– – 0 0	0 0 1 0
0 1 1 –	0 0 1 0
1 0 1 –	0 1 0 0
0 1 1 –	0 1 0 0
0 – – 1	0 1 0 0
– – – 1	1 0 0 0
– 0 1 –	1 0 0 0

Конъюнкции из ДНФ различных компонентных функций могут совпадать. В табл. 2 это конъюнкции, представленные троичными векторами (– – 0 0), (0 1 1 –). Для системы из табл. 3 $k_1 = 2$, $k_2 = 3$, $k_3 = 3$, $k_4 = 3$, $\sum_{i=1}^{m} k_i = 11$.

Кратчайшей системой ДНФ D_f для системы булевых функций $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$ называется такая система ДНФ, которая содержит минимальное число общих элементарных конъюнкций, на которых заданы ДНФ D_{f_i}, $i = 1, \ldots, m$, всех функций f_i системы $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$.

Задача 1. Совместная минимизация системы булевых функций. Для заданной системы $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$ булевых функций найти кратчайшую систему ДНФ D_f.

Задача 2. Раздельная минимизация системы булевых функций. Найти кратчайшие ДНФ D_{f_i} для каждой компонентной функции $f_i(\boldsymbol{x})$, $i = 1, \ldots, m$, заданной системы $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$ булевых функций.

Заметим, что в совместно минимизированной системе ДНФ (табл. 1) $\sum_{i=1}^{m} k_i = 12$. Конъюнкции, представленные троичными векторами (– – 0 0), (0 1 1 –), (1 1 0 –) (см. табл. 1), входят в ДНФ пар компонентных функций.

При решении задачи 1 минимизируется число k общих элементарных конъюнкций, при решении задачи 2 минимизируются числа k_i. Для задач 1 и 2 существенную роль играют понятия импликанты и простой импликанты. *Импликантой* булевой функции $f(x_1, \ldots, x_n)$ называется такая элементарная конъюнкция K литералов x_i, $\bar{x}_i$ булевых переменных x_i, что на любом наборе значений переменных $x_1, x_2, \ldots, x_n$, на котором K равна единице, значение функции $f(x_1, \ldots, x_n)$ также равно единице. *Простой импликантой* называется та, которая перестает быть импликантой при удалении из нее любого символа. Понятие простой импликанты одной функции обобщается на случай простой импликанты системы функций [10]. Если для некоторого множества

функций системы существует простая импликанта K_{pr}, то это не означает, что она же будет являться простой импликантой для каждой функции в отдельности. Элементарная конъюнкция K_{pr} будет простой импликантой для системы функций F тогда и только тогда, когда выполняются следующие условия:

– найдется такая подсистема функций $F^* \subseteq F$, $F^* \neq \varnothing$, что K_{pr} является импликантой для каждой функции подсистемы F^*;

– K_{pr} не будет импликантой хотя бы для одной функции подсистемы F^{**}, где $F^{**} = F^* \cup f^*$, $f^* \in F \setminus F^*$;

– импликанта K_{pr} не будет импликантой хотя бы для одной функции подсистемы F^*, если из K_{pr} удалить любой символ.

Программы минимизации. Программа Espresso [11] является самой известной программой минимизации и предназначена для совместной и раздельной минимизации систем полностью определенных и частичных булевых функций (и систем многозначных функций) в классе ДНФ по различным критериям – числу элементарных конъюнкций и числу литералов. В табл. 1 представлена кратчайшая система ДНФ для системы функций из табл. 2, полученная программой Espresso IIC (далее – Espresso).

Существуют различные варианты и модификации данной программы: Espresso-Exact (программа точной минимизации), Espresso-MV [12] (программа минимизации многозначных функций), Espresso-Signature [13]. Программе Espresso посвящена монография [11], на которую имеется большое число ссылок в научной литературе. В программе Espresso в процессе минимизации итеративно выполняются следующие операции: ESSENTIAL – выделение существенных импликант, EXPAND – расширение, REDUCE – сокращение, RESHAPE – модификация, IRREDUNDANT – безызбыточное покрытие. Выполнение процедуры ESSENTIAL заключается в выделении существенных импликант, которые входят в любое безызбыточное покрытие, состоящее из простых импликант. Эти импликанты переносятся в искомую безызбыточную ДНФ в качестве ее обязательной части и далее в поиске интервального покрытия исходных булевых функций не участвуют, что сокращает время поиска. Выполнение процедур REDUCE, EXPAND и IRREDUNDANT циклически повторяется до тех пор, пока решение не стабилизируется, т. е. пока на очередной итерации цикла не получится результат с таким же числом импликант (или литералов), что и на предыдущей итерации. Это означает, что решение достигло локального минимума относительно процедур REDUCE, EXPAND и IRREDUNDANT. Для сбрасывания полученного решения с минимума в ESPRESSO-MV используются более изощренные, чем REDUCE, процедуры LAST_GASP или SUPER_GASP [12] и NEW_GASP [14]. В экспериментах, описание которых приводится далее, использовалась программа Espresso (без ключей) для совместной минимизации системы полностью определенных функций; критерий оптимизации – число общих элементарных конъюнкций.

Программа ABC [15] для раздельной минимизации основывается на графовых моделях многоуровневых BDD-представлений систем функций. Сначала осуществляется построение совместной (shared) BDD для системы функций, затем используется модель бинарной диаграммы решений с подавлением нулей (Zero-suppresed BDD, ZDD). ZDD представляют структуры данных для решения комбинаторных задач, в которых необходима работа с множествами подмножеств. Такие структуры данных эффективны для «разреженных» объектов, в которых много дуг ведет к листовой вершине 0. Быстрое получение ZDD оказалось эффективным для раздельной минимизации ДНФ булевых функций большой размерности и используется в программе ABC. О модели ZDD можно найти информацию в работе [16, с. 358].

В программе Tie [10] путем совместной минимизации систем булевых функций, где критерий оптимизации – число общих элементарных конъюнкций, реализуется алгоритм, состоящий из следующих шагов: 1) поиск всех простых импликант исходной системы ДНФ булевых функций; 2) построение булевой матрицы A покрытия (матрицы Квайна) и ее сокращение; 3) нахождение кратчайшего столбцового покрытия матрицы A. Подробное описание алгоритма дано в работе [10]. Его особенностями являются оригинальная процедура нахождения импликант и процедура нахождения покрытия матрицы A из работы [17].

Программа Minim [18] предназначена для совместной и раздельной минимизации систем как полностью определенных, так и частичных булевых функций, заданных в интервальной форме. Критериями оптимизации являются число общих элементарных конъюнкций при совместной минимизации и число конъюнкций в кратчайшей ДНФ каждой компонентной функции системы при раздельной минимизации. Программа реализует обобщение алгоритма из работы [19], написана на языке C++ и использует специальные структуры данных для векторно-матричных объектов (булевых и троичных векторов и матриц) [20, 21]. Подробное описание форм представления данных, реализованных в программе алгоритмов и результатов экспериментов дано в работе [18]. В табл. 3 приведены кратчайшие ДНФ функций из табл. 2, полученные в результате раздельной минимизации программой Minim. Далее в экспериментах исследовался реализованный в программе Minim итерационный алгоритм как для совместной, так и для раздельной минимизации с заданием режима, ориентированного на получение решения лучшего качества [19].

Программы Tie и Minim включены в систему FLC-2 [9] и используют в качестве входных и выходных данных текстовый матричный формат SDF языка SF иерархического функционально-структурного описания комбинационной логики [22].

Примеры систем функций для экспериментов. Исследования были проведены для четырех наборов примеров систем булевых функций.

Первый набор примеров был представлен системами функций в формате PLA из библиотеки Berkeley PLA Test Set (URL: http://www1.cs.columbia.edu/~cs6861/sis/espresso-examples/ex). Данные системы ДНФ булевых функций переводились из формата PLA в матричные описания на языке SF. В экспериментах исходными описаниями являлись матричные формы систем ДНФ, СДНФ, таблиц истинности. Если в параметрах исходного задания было указано значение $k = 2^n$, то в качестве исходных выступали таблицы истинности соответствующей системы булевых функций. Было установлено, что примеры GARY и IN0 задают одну и ту же систему функций в виде различных систем ДНФ, т. е. ДНФ с различными множествами элементарных конъюнкций. Примеры Too_large_matr, X3_matr систем ДНФ были получены из многоуровневых описаний систем функций в виде суперпозиций ДНФ.

Второй набор примеров составляли системы булевых функций, взятых из практики проектирования управляющей логики заказных СБИС. Примеры VERG1, VERG2 – системы СДНФ функций, задающих таблицы для хранения микропрограмм команд микропроцессоров; примеры Sist4, Sistem8 – системы ДНФ, задающие блоки управляющей логики в заказных СБИС. Примеры S12, …, S16 – таблицы истинности, задающие SF-описания систем функций (n = 12, ..., 16, m = 12, ..., 16, $k = 2^{12}$, ..., 2^{16}). Данные SF-описания были получены следующим образом: двоичные строки в матрице B^f являются значениями с заданной точностью (в двоичной системе счисления) функции $y = \sin x$ на интервале $[0, \pi/2]$, матрицу B^x образуют все полные элементарные конъюнкции, состоящие из n литералов. Для примера S12 (n = 12, m = 12, $k = 2^{12}$) первая строка матрицы B^x состоит из 12 нулей, последняя – из 12 единиц, матрица B^f – из 12 столбцов. Строго говоря, эти примеры не задают функциональных описаний математической функции $y = \sin x$ на интервале $[0, \pi/2]$, так как в левой части таблиц истинности перечислены не значения аргумента x в двоичном коде, а 2^n полных конъюнкций.

Третий набор примеров был сформирован из псевдослучайных систем ДНФ (табл. 4), сгенерированных программой [23], троичные строки в матрице T^x содержали случайное число $n - p$ значений «–», булевы строки в матрице B^f содержали случайное число c единичных значений. При переходе от систем ДНФ к таблицам истинности в булевых матрицах B^f оказывалось много единичных элементов. Например, для примера Psevdo_1 в таблице истинности матрица B^f оказалась на 83 % заполнена единицами, так как сгенерированные элементарные конъюнкции в матрице T^x часто оказывались неортогональными. В практике экспериментальных исследований было установлено, что данные примеры имеют большую сложность при схемной реализации в FPGA, поэтому представлялась целесообразной проверка эффективности логической минимизации для этих примеров.

Таблица 4
Параметры псевдослучайных систем ДНФ

Table 4
Parameters of pseudorandom DNF systems

Пример *Example*	*n*	*m*	*k*	Среднее число литералов в конъюнкции, *p* *Average number literals in conjunction, p*	Среднее число вхождений конъюнкции в ДНФ функций, *c* *Average number occurrences conjunctions in DNF functions, c*	Число неопределенных элементов «–» в матрице T^x, % *Number uncertain elements «–» in matrix T^x, %*
Psevdo1	10	20	1 000	9	2	33
Psevdo2	15	20	5 000	14	2	32
Psevdo3	20	20	10 000	19	3	30
Psevdo4	25	20	25 000	24	4	31
Psevdo5	30	20	30 000	29	5	31

Четвертый набор примеров составляли системы функций, описывающие функционирование умножителей (устройств перемножения) и устройств возведения в квадрат целых неотрицательных чисел. Матричные описания систем ДНФ таких устройств были получены в результате перехода от структурных описаний логических схем, синтезированных в библиотечном базисе, путем элиминации (устранения) промежуточных переменных. Особенностью этих примеров является то, что они «частично» минимизированы, т. е. в процессе перехода от задания функций в виде взаимосвязанных логических уравнений к ДНФ были выполнены операции обобщенного склеивания и поглощения [6]. Подобные примеры систем функций появляются, например, при решении задач перепроектирования логических схем из одного базиса логических элементов в другой базис.

Эксперименты без распараллеливания вычислений. Эксперименты проводились на персональном компьютере Lenovo в 64-разрядной ОС Windows7 PC с процессором Intel Core i5-2320 и тактовой частотой 3,00 ГГц, ОЗУ 4 Гб. Программной средой для выполнения экспериментов была система FLC-2, языком описания данных в этой системе является язык SF функционально-структурного описания комбинационных логических схем. В языке SF допускаются иерархические описания, при этом для листовых описаний возможны как матричные описания систем ДНФ, СДНФ, таблиц истинности, так и логические уравнения в булевом базисе.

Эксперимент 1. Сравнение эффективности программ совместной минимизации Espresso, Tie и Minim для решения задачи 1. Исходными данными для большинства примеров в этом эксперименте являлись системы ДНФ, а для некоторых примеров – их частные случаи: системы СДНФ и таблицы истинности. Программа Minim выполнялась с установкой опций (совместной минимизации, итерационного метода).

Далее в таблицах с результатами экспериментов через $k_{\min}$ обозначено число элементарных конъюнкций в кратчайшей системе ДНФ. Лучшие решения помечены символом «*». Символ «–» означает, что для соответствующего примера программа минимизации не выполнялась. Например, для примера VTX1 (табл. 5) программа Tie не выполнялась, так как она имеет ограничение $n \le 22$ для исходных данных. Для некоторых примеров указано время работы программы и пометка «Нет решения», которая означает, что за указанное в таблице время программа минимизации не получила решения.

По результатам эксперимента 1 (табл. 5) можно сделать следующие выводы. При размерностях задачи совместной минимизации $n \le 20$ лучшие решения получает программа Tie, хотя и затрачивает немного больше времени на решение. При $n > 20$ и большом числе конъюнкций в исходных ДНФ (либо таблицах истинности) лучшие решения получает программа Espresso. Однако при больших размерностях задачи на потоке псевдослучайных примеров конкурентоспособной оказывается программа Minim, которая, например, получает решение для примера Psevdo5 за практически приемлемое время, а программе Espresso за 1,5 ч (5 400 с) не удается

обработать данный пример. Эксперимент 1 позволил установить, что псевдослучайные примеры из третьего набора примеров не только имеют большую сложность при схемной реализации в FGPA, но и являются трудоемкими при совместной минимизации, так как требуют много времени для решения задачи 1, а полученные в результате решения ДНФ содержат много элементарных конъюнкций.

Таблица 5
Результаты эксперимента 1. Сравнение программ совместной минимизации, исходные данные – системы ДНФ

Table 5
Results of experiment 1. Comparison of joint minimization programs, initial data – DNF system

Исходные данные *Initial data*				Совместная минимизация *Joint minimization*					
				Espresso		Tie		Minim	
Пример *Example*	*n*	*m*	*k*	k_{min}	Время, с *Time, s*	k_{min}	Время, с *Time, s*	k_{min}	Время, с *Time, s*
Первый набор примеров									
Z4	7	4	128	*59	0,10	*59	*0,02	*59	0,05
Z5XP1	7	10	128	76	*0,12	*63	4,94	66	0,62
RADD	8	5	120	*75	0,11	*75	*0,02	*75	0,06
ROOT	8	5	256	*57	0,11	*57	*0,02	62	0,07
MLP4	8	8	256	133	*0,13	*121	4,99	135	0,16
MAX512	9	6	512	141	*0.13	*133	4,99	153	0,49
SYM10	10	1	837	*210	*0.14	*210	5,01	*210	0,91
MAX1024	10	6	1 024	276	*0,20	*260	4,99	297	3,43
ADD6	12	7	1 092	*355	*0,16	*355	1,48	*355	1,93
TIAL	14	8	640	581	*0,38	*575	12,0	636	5,89
MP2D	14	14	123	31	*0,13	*30	14,84	36	0,02
INTB	15	7	664	631	*0,48	*629	13,27	635	19,46
B12	15	9	431	42	0,14	*41	19,4	48	*0,09
M181	15	9	430	42	0,14	*41	16,04	48	*0,03
IN0	15	11	138	*107	0,14	*107	6,26	109	*0,13
GARY	15	11	442	*107	*0,17	*107	6,29	128	0,26
B9	16	5	123	*119	0,12	*119	20,01	123	*0,11
IN1	16	17	110	*104	0,15	*104	24,87	105	*0,12
IN2	19	10	137	135	*0,15	*134	489,98	137	0,20
VTX1	27	6	110	*110	0,17	–	–	*110	*0,13
X9DN	27	7	120	*120	0,15	–	–	*120	*0,13
jbp	36	57	166	*122	0,39	–	–	166	*0,13
Too_large_matr	38	3	1 027	*1 027	*0,97	–	–	*1 027	18,21
ibm	48	17	173	*173	1,05	–	–	*173	*0,58
SOAR	83	94	529	*353	*0,83	–	–	427	5,42
X3_matr	135	99	915	*915	*1,26	–	–	*915	18,58
Второй набор примеров									
S12	12	12	4 096	2 523	*7,36	*2 215	4,98	2 784	79,44
S13	13	13	8 192	4 929	*33,15	*4 275	20,01	5 836	279,59
S14	14	14	16 384	*9 672	197,06	10 148	*91,5	11 596	1 253,77
S15	15	15	32 768	*18 768	947,31	19 068	*780,49	Нет решения	5 400,00
S16	16	16	65 536	*36 509	*5 609,26	–	–		
Sist4	17	12	370	266	*0,33	*260	419,94	282	1,45
Verg1	17	61	2 003	482	*1,01	*469	4,95	546	17,72
Verg2	18	63	2 129	528	*1,06	*508	9,95	607	32,39
Sistem8	25	20	45 548	28 668	*2 239,40	–	–	Нет решения	5 400,00
Третий набор примеров									
Psevdo1	10	20	1 000	*509	*2,42	893	5,01	704	13,88
Psevdo2	15	20	5 000	*4 754	*20,13	9 571	653,36	4 770	181,11
Psevdo3	20	20	10 000	*7 639	*376,05	Нет решения	5 400,00	*7 639	666,25
Psevdo4	25	20	25 000	*6 522	*453,84			*6 522	488,31
Psevdo5	30	20	30 000	Нет решения	5 400,00			*9 132	*869,39

Эксперимент 2. Сравнение программ Espresso, Tie и Minim совместной минимизации. Исходными данными являлись ортогонализованные системы ДНФ булевых функций. Ортогонализация выполнялась с помощью программы [24]. В ортогонализованной системе ДНФ каждая пара элементарных конъюнкций K_i, K_j является ортогональной: $K_i\&K_j = 0$. Результаты эксперимента приведены в табл. 6, где через k^{ort} обозначено число элементарных конъюнкций в ортогонализованной системе ДНФ.

Таблица 6
Результаты эксперимента 2. Сравнение программ совместной минимизации, исходные данные – ортогонализованные системы ДНФ

Table 6
Results of experiment 2. Comparison of joint minimization programs, initial data – orthogonalized DNF systems

Исходные данные *Initial data*				Ортогонализация *Orthogonalization*	Совместная минимизация *Joint minimization*					
					Espresso		Tie		Minim	
Пример *Example*	*n*	*m*	*k*	k^{ort}	$k_{\min}$	Время, с *Time, s*	$k_{\min}$	Время, с *Time, s*	$k_{\min}$	Время, с *Time, s*
RADD	8	5	120	255	*75	0,12	*75	*0,02	*75	0,07
TIAL	14	8	640	11 299	581	*1,28	*575	11,38	801	106,39
B12	15	9	431	1 606	*41	*0,16	*41	19,31	46	0,85
M181	15	9	430	2 239	*41	*0,17	*41	20,57	47	1,95
IN0	15	11	138	138	*107	*0,15	*107	6,29	109	0,16
GARY	15	11	442	282	*107	0,14	*107	6,29	112	*0,12
B9	16	5	123	12 793	*119	*0,41	*119	19,97	169	111,20
IN1	16	17	110	185	*104	*0,15	*104	25,04	109	*0,15
IN2	19	10	137	1 772	137	*0,17	*134	544,99	148	1,54

Эксперимент 3. Сравнение программ совместной минимизации. В качестве исходных данных использовались таблицы истинности систем булевых функций. Для эквивалентного перехода от систем ДНФ к таблицам истинности в системе FLC-2 имеется соответствующая программа. Результаты приведены в табл. 7.

Таблица 7
Результаты эксперимента 3. Сравнение программ совместной минимизации, исходные данные – таблицы истинности

Table 7
Results of experiment 3. Comparison of joint minimization programs, source data – truth tables

Исходные данные *Initial data*				Таблица истинности *Truth table*	Совместная минимизация *Joint minimization*					
					Espresso		Tie		Minim	
Пример *Example*	*n*	*m*	*k*	Число наборов *Number of sets*	$k_{\min}$	Время, с *Time, s*	$k_{\min}$	Время, с *Time, s*	$k_{\min}$	Время, с *Time, s*
RADD	8	5	120	256	*75	0,10	*75	0,02	*75	*0,09
TIAL	14	8	640	16 384	582	*1,32	*575	11,42	800	97,74
B12	15	9	431	32 768	*41	*1,25	*41	19,25	47	38,57
M181	15	9	430	32 768	*41	*1,21	*41	20,70	47	41,39
IN0	15	11	138	32 768	*107	*0,59	*107	6,30	120	38,18
GARY	15	11	442	32 768	*107	*0,60	*107	6,29	120	38,18
B9	16	5	123	65 536	*119	*1,39	*119	20,0	254	2 156,8
IN1	16	17	110	65 536	*104	*1,41	*104	25,09	107	14,46
IN2	19	10	137	524 288	137	*19,73	*134	559,53	–	–

Эксперименты 2 и 3 показали, что для программ Espresso и Tie форма исходного задания (ДНФ, ортогонализованные ДНФ, таблицы истинности) не влияет на качество и время реше-

ния. Для программы Minim более предпочтительной формой исходного задания являются ДНФ, а не таблицы истинности.

Эксперимент 4. Сравнение программ раздельной минимизации. Исходными данными являются системы ДНФ булевых функций. В качестве одного из параметров для исходных данных указывается $\sum_{i=1}^{m} k_i$ – суммарное число элементарных конъюнкций в исходной (неминимизированной) системе ДНФ, через $\sum_{i=1}^{m} k_{\min}^{i}$ (в табл. 8 и далее) обозначено суммарное число элементарных конъюнкций в раздельно минимизированных ДНФ.

Программа Minim выполнялась с установкой опций (раздельной минимизации, итерационного метода). Программа ABC в результате своей работы выдавала значение параметра Node (число узлов в совместной BDD функций). Применение программы Tie совместной минимизации для выполнения раздельной минимизации осуществлялось следующим образом. По исходному SF-описанию системы функций строилась логическая сеть, блоками которой являлись компонентные функции системы. Затем последовательно выполнялась минимизация каждого блока по отдельности с помощью программы Tie и подсчитывалось суммарное время на обработку всех блоков сети. Последовательная обработка листовых описаний логической сети проводилась с помощью стратегии «Минимизировать листовые SF-описания» [22]. Результаты эксперимента приведены в табл. 8.

Эксперимент 4 позволил установить тот факт, что программа ABC раздельной минимизации является наиболее быстродействующей, она проиграла программе Minim по быстродействию только в одном случае (см. пример Psevdo5, табл. 8). Однако программа ABC проигрывает по качеству решений программам Tie и Minim. Как и ожидалось, для диапазона $n \le 20$ лучшие решения получила программа Tie совместной минимизации, используемая в данном эксперименте для нахождения кратчайших ДНФ отдельных функций.

Эксперимент 5. Проверка эффективности программы совместной минимизации Espresso после выполнения программы ABC раздельной минимизации. Результаты эксперимента приведены в табл. 9.

Основываясь на данных табл. 9, можно утверждать, что качество решений задачи совместной минимизации программой Espresso чаще всего можно улучшить, если предварительно провести раздельную минимизацию программой ABC. Тем не менее время вычислений может существенно возрасти для псевдослучайных примеров Psevdo3, Psevdo4 (табл. 9), хотя качество решений, получаемых в данных случаях программой Espresso, не меняется. Вместе с тем закономерность такого значительного увеличения времени выявить не удалось.

Эксперимент 6. Сравнение эффективности применения программы ABC для компонентных функций по отдельности и для системы функций в целом. Эксперимент проводился на примере Psevdo5, обработка которого заняла значительное время (табл. 9). Результаты эксперимента показали, что разбиение исходной системы функций на подсистемы, содержащие по одной функции, позволяет значительно сократить время работы программы ABC. Обработка компонентных функций по отдельности привела к получению суммарного числа узлов раздельных BDD для 20 функций системы – 3 934 961 (для системы в целом в совместной BDD было 3 357 956 узлов). Суммарное время минимизации 20 подсистем составило 233,36 с, время обработки системы в целом – 4 669,62 с (табл. 9). В итоге получили $\sum_{i=1}^{m} k_i$ = 45 809. Это тот же результат, что и при минимизации системы в целом. Разбиение системы на «одиночные» функции позволило сократить время в 20 раз – по числу функций системы. Таким образом, если система функций содержит достаточно много компонентных функций и размерность задачи раздельной минимизации велика: $n > 30$, $k > 30\,000$, то целесообразно разбить исходную систему на подсистемы (например, содержащие по одной компонентной функции) и выполнить раздельную минимизацию программой ABC для подсистем.

Таблица 8
Результаты эксперимента 4. Сравнение программ раздельной минимизации

Table 8
Results of experiment 4. Comparison of separate minimization programs

Исходные данные *Initial data*					Раздельная минимизация *Separate minimization*							
					ABC			Minim			Разбиение системы функций на блоки и Tie *Splitting the system of functions into blocks and Tie*	
					Моноблок – система функций *Monoblock – function system*			Моноблок – система функций *Monoblock – function system*			Сеть компонентных функций *Network of Component Functions*	
Пример *Example*	n	m	k	$\sum_{i=1}^{m} k_i$	$\sum_{i=1}^{m} k_{\min}^{i}$	Node	Время, с *Time, s*	$k^{\text{общ}}$	$\sum_{i=1}^{m} k_{\min}^{i}$	Время, с *Time, s*	$\sum_{i=1}^{m} k_{\min}^{i}$	Время, с *Time, s*
Первый набор примеров												
Z4	7	4	128	256	*59	26	*0,01	59	*59	0,03	*59	1,01
Z5XP1	7	10	128	576	*74	50	*0,01	71	*74	0,04	*74	2,44
RADD	8	5	120	120	*75	34	*0,01	75	*75	0,04	*75	1,01
ROOT	8	5	256	615	75	68	*0,01	71	*71	0,04	*71	2,10
MLP4	8	8	256	678	145	142	*0,01	143	*143	0,06	*143	1,81
MAX512	9	6	512	1 616	183	155	*0,01	171	172	0,13	*164	6,38
SYM10	10	1	837	837	240	43	*0,006	210	*210	0,67	*210	5,24
MAX1024	10	6	1 024	3 232	364	254	*0,01	328	*330	0,73	629	16,66
ADD6	12	7	1 092	1 092	*355	65	*0,01	355	*355	1,26	*355	1,53
TIAL	14	8	640	644	653	867	*0,01	639	643	0,76	*631	5,66
MP2D	14	14	123	204	*76	122	*0,005	38	*76	0,05	*76	6,51
INTB	15	7	664	664	635	633	*0,01	635	635	1,32	*629	3,65
B12	15	9	431	454	*53	81	*0,005	48	*53	0,03	*53	4,78
M181	15	9	430	453	*53	81	*0,01	48	*53	0,02	*53	4,87
IN0	15	11	138	487	203	391	*0,01	166	193	0,08	*191	7,61
GARY	15	11	442	442	203	391	*0,01	164	*191	0,05	*191	7,54
B9	16	5	123	123	156	85	*0,01	119	*119	0,06	*119	7,21
IN1	16	17	110	1 074	727	567	*0,01	273	*698	0,42	*698	17,59
IN2	19	10	137	310	239	290	*0,01	201	*231	0,10	*230	166,14
VTX1	27	6	110	110	*110	182	*0,01	110	*110	0,04	–	–
X9DN	27	7	120	120	*120	182	*0,01	120	*120	0,05	–	–

Окончание табл. 8

End of table 8

Исходные данные *Initial data*					Раздельная минимизация *Separate minimization*							
					ABC			Minim			Разбиение системы функций на блоки и Tie *Splitting the system of functions into blocks and Tie*	
					Моноблок – система функций *Monoblock – function system*			Моноблок – система функций *Monoblock – function system*			Сеть компонентных функций *Network of Component Functions*	
Пример *Example*	n	m	k	$\sum_{i=1}^{m} k_i$	$\sum_{i=1}^{m} k_{\min}^{i}$	Node	Время, с *Time, s*	$k^{\text{общ}}$	$\sum_{i=1}^{m} k_{\min}^{i}$	Время, с *Time, s*	$\sum_{i=1}^{m} k_{\min}^{i}$	Время, с *Time, s*
jbp	36	57	166	189	190	437	*0,01	166	*189	0,02	–	–
Too_large_matr	38	3	1 027	1 056	*1 056	1 050	*0,05	1 027	*1 056	7,79	–	–
ibm	48	17	173	173	*173	471	*0,02	173	*173	0,07	–	–
SOAR	83	94	529	529	500	551	*0,02	433	*486	0,32	–	–
X3_matr	135	99	915	997	*997	851	*0,03	915	*997	2,76	–	–
Второй набор примеров												
S12	12	12	4 096	27 238	3 421	1 758	*0,02	2916	2 944	72,01	*2 770	43,34
S13	13	13	8 192	58 616	6 604	3 033	*0,06	5 540	5 584	278,94	*5 192	44,02
S14	14	14	16 384	125 497	12 859	5 136	*0,10	11 236	11 300	490,88	*9 859	50,63
S15	15	15	32 768	267 551	25 186	8 862	*0,24	25 671	25 792	1228,56	*18 830	63,06
Sin16	16	16	65 536	568 043	49 382	15 005	*0,48	52 853	53 034	4 897,58	*36 329	89,26
Sist4	17	12	370	373	326	407	*0,01	284	285	0,21	*284	19,07
Verg1	17	61	2 003	97 127	9 334	1 685	*0,01	1 044	8 686	25,36	*8 646	66,38
Verg2	18	63	2 129	107 281	7 652	1 680	*0,01	1 031	7 548	22,09	*7 334	180,98
Sistem8	25	20	45 548	45 947	37 035	14 435	*0,40	36 870	*36 883	2979,90	–	–
Третий набор примеров												
Psevdo1	10	20	1 000	2 103	1 248	1 629	*0,01	965	1 059	1,35	*977	81,32
Psevdo2	15	20	5 000	10 676	15 284	45 857	*0,57	6 724	9 604	100,58	*9 571	103,70
Psevdo3	20	20	10 000	26 657	20 775	534 629	*77,38	8 408	*20 320	476,15	*20 320	1 788,42
Psevdo4	25	20	25 000	92 300	23 939	1 278 016	*341,90	6 606	*23 938	566,25	–	–
Psevdo5	30	20	30 000	150 620	*45 809	3 357 956	4 669,62	9 140	45 899	*1 641,27	–	–

Таблица 9
Результаты эксперимента 5. Совместная минимизация программой Espresso после выполнения раздельной минимизации программой ABC

Table 9
Results of experiment 5. Joint minimization by Espresso program after performing separate minimization by ABC program

Пример *Example*	Исходное задание *Initial task*	Раздельная минимизация *Separate minimization*		Совместная минимизация *Joint minimization*			
		ABC		Espresso после ABC *Espresso after ABC*		Espresso от исходных описаний *Espresso from original descriptions*	
	$\sum_{i=1}^{m} k_i$	$\sum_{i=1}^{m} k_i^{ABC}$	Время, с *Time, s*	$\sum_{i=1}^{m} k_i^{Esp}$	Время, с *Time, s*	$\sum_{i=1}^{m} k_i^{Esp}$	Время, с *Time, s*
S12	27 238	3 421	0,01	*2 442	*5,39	2 523	7,36
S13	58 616	6 604	0,06	*4 667	37,07	4 929	*33,15
S14	125 497	12 859	0,10	*9 080	*125,40	9 672	197,06
S15	267 551	25 186	0,24	*17 441	*621,56	18 768	947,31
S16	568 043	49 382	0,48	*33 666	*3 060,86	36 509	5 609,26
Sistem8	45 947	37 035	0,40	*28 575	2 246,04	28 668	*2 239,40
Psevdo1	2 103	1 248	0,01	522	*1,44	*509	2,42
Psevdo2	10 676	15 284	0,57	4 806	59,16	*4 754	*20,13
Psevdo3	26 657	20 775	77,38	*7 639	1 621,76	*7 639	*376,05
Psevdo4	92 300	23 939	341,90	*6 522	18 572,60	*6 522	*453,84
Psevdo5	150 620	45 809	4 669,62	–	–	–	–

Эксперимент 7. Сравнение длин ДНФ, найденных с использованием BDD, с длинами кратчайших ДНФ функций. Результаты эксперимента приведены в табл. 10. По BDD, полученным программой OPT_BDD [24], строилась система ДНФ с помощью программы элиминации промежуточных переменных [25], которая входит в систему FLC-2. В данной программе в процессе получения ДНФ функций выполняются операции поглощения конъюнкций, а для результирующей ДНФ – операции обобщенного склеивания и поглощения конъюнкций.

В итоге строится матричная форма системы ДНФ булевых функций, которая содержит k^{BDD} общих элементарных конъюнкций. Суммарное число конъюнкций в ДНФ, построенных по BDD, обозначено через $\sum_{i=1}^{m} k_i^{BDD}$, где k_i^{BDD} – число элементарных конъюнкций в ДНФ i-й функции системы, m – число функций системы. Исходное задание системы функций ранее минимизировалось программами раздельной минимизации. Результаты минимизации приведены в табл. 9, их можно сравнить с данными из табл. 10.

По эксперименту 7 можно сделать следующие выводы: переход от BDD к ДНФ меняет форму задания функций, подвергаемых минимизации. Поэтому в результате могут быть получены как лучшие, так и худшие решения по сравнению с исходными системам ДНФ. Если у пользователя нет программы минимизации функций в классе ДНФ, то он может воспользоваться программой минимизации BDD, после чего перейти к ДНФ. Это будет целесообразно, если $k^{BDD} \le k$. Однако в общем случае такой подход неэффективен, так как зачастую k^{BDD} будет значительно больше числа k конъюнкций в исходном задании функций. Такая ситуация объясняется тем, что BDD задают ортогонализованные формы компонентных функций, а использование при элиминации простых преобразований ДНФ (склеивания и поглощения конъюнкций) не позволяет провести качественную минимизацию в классе ДНФ ортогонализованных форм, получаемых по BDD-представлениям.

Таблица 10
Результаты эксперимента 7. Раздельная минимизация по BDD-представлению системы функций

Table 10
Results of experiment 7. Separate minimization by BDD-representation of a system of functions

Исходные данные *Initial data*					BDD-минимизация и элиминация *BDD-minimization and elimination*	
Пример *Example*	n	m	k	$\sum_{i=1}^{m} k_i$	k^{BDD}	$\sum_{i=1}^{m} k_i^{BDD}$
Z4	7	4	128	256	59	*59
Z5XP1	7	10	128	576	79	*82
RADD	8	5	120	120	75	*75
ROOT	8	5	256	615	69	*79
MLP4	8	8	256	678	195	*226
MAX512	9	6	512	1 616	221	*236
SYM10	10	1	837	837	240	*240
MAX1024	10	6	1 024	3 232	432	*480
ADD6	12	7	1 092	1 092	651	*651
TIAL	14	8	640	*644	2 702	2 798
MP2D	14	14	123	204	48	*85
INTB	15	7	664	*664	1 785	1 791
B12	15	9	431	454	57	*63
M181	15	9	430	453	58	*64
IN0	15	11	138	487	206	*262
GARY	15	11	442	442	206	*262
B9	16	5	123	*123	249	249
IN1	16	17	110	*1 074	537	1 791
IN2	19	10	137	*310	566	659
VTX1	27	6	110	*110	612	612
X9DN	27	7	120	*120	1 818	1 818
jbp	36	57	166	*189	574	638
ibm	48	17	173	*173	1 380	1 380
SOAR	83	94	529	*529	1 112	1 182
X3_matr	135	99	915	*997	17 205	17 312
S12	12	12	4 096	27 238	5 035	*6 048
Sist4	17	12	370	*373	946	975
Verg1	17	61	2 003	97 127	1 837	*16 206
Verg2	18	63	2 129	107 281	2 473	*17 527

Многопоточное программирование и стандарт OpenMP. Разработка параллельных алгоритмов [26] и проектирование могопоточных программ представляют собой довольно сложные процессы и требуют использования стандартов и технологий параллельного программирования, которые позволяют упростить работу по организации параллельных вычислений. В настоящее время для организации параллельных вычислений был использован стандарт OpenMP. Этот стандарт реализован в языках программирования C/C++ и Fortran, работает как в Unix-, так и в Windows-системах. Технология OpenMP представляет собой набор особых директив компилятора, процедур и переменных окружения (URL: http: //www.openmp.org/) [27, 28]. Процесс использования OpenMP сводится к вставке директив в места программы, где код может быть распараллелен [29].

Проектирование параллельной программы сводится к поэтапному выполнению таких действий, как разделение общих задач на подзадачи, выявление информационных зависимостей у подзадач, масштабирование подзадач и распределение по вычислительным элементам. (Для систем с общей памятью роль вычислительных элементов выполняют ядра процессора.)

В общем случае параллельные программы – программы с несколькими исполняемыми потоками или процессами – могут работать как на одном компьютере (многоядерной системе), так и на мультипроцессорной системе (на кластере или по сети). В последних двух случаях для синхронизации используется передача сообщений.

Поскольку выполняемая программа образует процесс, то возможна реализация параллельной программы с использованием двух подходов:

1. Программа, использующая несколько потоков в рамках одного процесса, может быть выполнена быстрее за счет параллельного выполнения ее отдельных частей (распределенных по потокам). Общаются потоки либо с помощью общей памяти, либо обмениваясь сообщениями.

2. Программа, использующая нескольких потоков в рамках нескольких процессов, тоже может быть реализована быстрее за счет параллельного выполнения отдельных ее процессов. Такая организация процесса вычислений эффективна, если потокам почти не надо общаться, когда много потоков выполняют операции параллельно и сводят свои результаты в один ответ только тогда, когда все потоки посчитали свои части.

Первый способ распараллеливания вычислений при раздельной минимизации булевых функций. Распараллеливание алгоритма раздельной минимизации системы функций, когда каждая из функций системы минимизируется независимо от других, может быть сравнительно просто реализовано на системе с общей памятью [27]. При параллельной реализации алгоритма раздельной минимизации систем булевых функций использовались современные многопоточные расширения C++ [26] и директивы OpenMP. Основным объектом распараллеливания (по числу функций m) является программный цикл, поскольку на циклических участках программы сосредоточены большие объемы вычислений. В частности, выполняется минимизация компонентной булевой функции f_i.

Таким образом, достаточно разбить исходную систему ДНФ булевых функций на m подсистем, содержащих по одной функции, после чего распределить подсистемы между m параллельными регионами и получить от них минимизированные ДНФ D_{f_i}. При этом возможно близкое (с учетом закона Амдала) к m-кратному увеличению быстродействие параллельной реализации алгоритма без потери качества на вычислительной системе по сравнению с последовательной реализацией. Основная часть алгоритма минимизации для компонентной функции, реализованного в работе [18], не меняется. Распараллеливание проводится на уровне обхода m подсистем для компонентных функций, на которые разбита исходная система ДНФ и для которых по отдельности выполняется алгоритм минимизации для компонентных функций.

Исходная система ДНФ полностью определенных булевых функций задается объектом Sop1. Минимизированная система ДНФ представляется объектом ResSop. Основой алгоритма представленной в листинге 1 функции MiOi (Limit, Crit, Tm1, Tm) является проведение минимизации в классе ДНФ одной полностью определенной булевой функции, заданной объектом Tm1 (троичная матрица). Результат минимизации (троичная матрица) сохраняется в объекте Tm. Результат минимизации (троичная матрица) сохраняется в объекте ResTm. В функции MiRi качество решения управляется параметрами Limit и Crit, задание которых стабилизирует качество получаемой системы ДНФ.

Работа параллельной реализации последовательного алгоритма [18] начинается с единственного потока – основного. После размещения OpenMP-директивы (строка 6) образуются параллельные регионы, входя в которые, основной поток создает группы потоков (включающие основной поток). В конце параллельного региона группы потоков останавливаются, а выполнение основного потока продолжается (строка 26). Далее внутри отдельного параллельного региона посредством размещения OpenMP-директивы task (строка 8) в текущем потоке выделяется отдельная независимая задача (так называемый легковесный поток – нить), с которой ассоциирован блок операторов. Таким образом, задача (task) помещается внутрь параллельной области и задает блок операторов, который может выполняться в отдельном потоке. Однако задача (task) не образует новый поток, а помещается в пул, из которого ее может взять для выполнения один из свободных потоков. Задача может выполняться немедленно после создания или быть отложенной на неопределенное время и выполняться по частям. Размер таких частей

определяется размером фрагмента блока операторов, а порядок выполнения частей разных текущих или отложенных задач определяется системным планировщиком задач и условиями синхронизации. Динамическое распределение задач по потокам осуществляется алгоритмами планирования типа work stealing [28].

Листинг 1. *Псевдокод параллельного алгоритма раздельной минимизации (первый способ распараллеливания)*

```
1   MiRi(Limit, Crit, CSOP &Sop1, CSOP & ResSop )
2   {
3     n=Sop1.foo1(), m=Sop1.foo2(), k=Sop1.foo3();
      // n – число входов, m – число выходов, k – число конъюнкций
4     CSOP SopR(….);        // создать объект для ДНФ D_fi
5     num_core= omp_get_num_procs ();  // определить число ядер
6   #pragma omp parallel for shared(Limit, Crit, Sop1, SopR, n, m)
      num_threads(num_core)
7     for (i = 0 ; i < m ; i++) {
8   #pragma omp task untied firstprivate(i) shared(Sop1, k, SopR, n, m)
9     {
10      CTM Tm, Tm1;        // создать два объекта: Tm, Tm1 – троичные матрицы
11      CSOP SopRR(….);     // в каждой task объект для ДНФ D_fi
12        ......
13      Tm1=foo4(…..) ;
14      …….
15      if(Tm1.foo5())
16      { // минимизация одной полностью определенной функции
17        MiOi (Limit, Crit, Tm1, Tm);
18        for (j = 0; j < Tm.foo4(); j++) {
19        {SopRR.foo5(Tm.foo6(j),….);}
20  #pragma omp critical
21        {
22          SopR.foo7(…SopRR);
23        }     // конец критической секции
24      }       // конец проверки условия в 15-й строке
25    }         // конец кода в блоке task
26   }          // конец кода в блоке parallel
27   ResSop.foo8(…SopR); // результирующая система ДНФ
28  }
```

Введение дополнительной опции untied для OpenMP-директивы task означает, что отложенная задача может быть продолжена любым потоком, выполняющимся в параллельной области. Если дополнительная опция не указана, то задача может быть продолжена только породившим ее потоком. С помощью OpenMP-директивы critical (строка 20) оформляется критическая секция программы, в которой собираются минимизированные ДНФ компонентной функции, полученные отдельными задачами. В каждый момент времени в критической секции может находиться не более одной задачи. Все другие задачи будут заблокированы, пока вошедшая задача не закончит выполнение. Как только вошедший в критическую секцию поток выйдет из нее, один из заблокированных потоков туда войдет. Если на входе стоит несколько потоков, то случайным образом выбирается один из них, а остальные продолжают ожидание. Естественно, введение критических секций снижает степень параллелизма. Задачи (tasks) являются альтернативой традиционным потокам (threads), позволяющей реализовать лучший баланс нагрузки, эффективное использование имеющихся ресурсов и высокий уровень абстракции, благодаря которому можно не заботиться о непосредственном управлении параллельным исполнением [28]. Вхождение той или иной задачи в критическую секцию, где формируется объединение ДНФ текущих решений, определяется скоростью нахождения минимальной ДНФ i-й компонентной функции.

Программа MinimPar раздельной минимизации является модификацией программы Minim и использует параллельные вычисления в рамках первого подхода. Исходными данными для программы служит система ДНФ, представленная одним блоком в формате SDF языка SF.

Второй способ распараллеливания вычислений при раздельной минимизации булевых функций. Суть организации параллельного выполнения, основанного на использовании средств современных ОС, состоит в следующем. Процесс – это экземпляр программы, загруженной в память компьютера для выполнения. Когда программа производит запуск другой программы, ОС всегда создает новый процесс, т. е. процесс создает дочерний процесс. Средства создания нового процесса полезны для использования функциональных возможностей программ, не имеющих графического интерфейса и работающих с командной строкой.

Для среды разработки Qt [29] процессы можно создавать с помощью класса QProcess, который определен в заголовочном файле QProcess. Объекты этого класса в состоянии считывать информацию, выводимую запущенными процессами, и даже подтверждать их запросы на ввод информации. Этот класс содержит методы для манипулирования системными переменными процесса. Работа с объектами класса QProcess осуществляется в асинхронном режиме, что позволяет сохранять работоспособность графического интерфейса программы-источника в моменты, когда запущенные процессы функционируют. При появлении данных или других событий объекты класса QProcess высылают сигналы. Например, при возникновении ошибок объект процесса вышлет сигнал error() с кодом этой ошибки.

Для создания процесса его нужно запустить. Запуск процесса выполняется методом start(), в который необходимо передать имя команды и список ее аргументов либо все вместе – команду и аргументы одной строкой. В поле аргументов метода start() может быть введена любая команда, соответствующая ОС. Как только процесс будет запущен, высылается сигнал started(), а после завершения его работы – сигнал finished(). Вместе с сигналом finished() высылается код и статус завершения работы процесса. Для чтения данных запущенного процесса класс QProcess предоставляет два разделенных канала: канал стандартного вывода (stdout) и канал ошибок (stderr). Считывать и записывать данные в процесс можно с помощью методов класса QIODevice::write(), read(), readLine() и getChar(). Также для чтения можно воспользоваться методами, привязанными к конкретным каналам: readAllStandardOutput() и readAllStandardError().

В рамках второго подхода параллельные вычисления реализует программа StartParBlock (листинг 2).

Листинг 2. *Псевдокод алгоритма раздельной минимизации, реализующего блочно-параллельных подход (второй способ распараллеливания)*

```
    // n – число входов, num – число выходов, k – число конъюнкций,
    // num_core – число ядер
1   #pragma omp parallel for firstprivate (…) shared(…,num)
    num_threads(num_core)
2   for(i=1; i<=num; i++) // цикл по числу блоков (num)
3   {
4     #pragma omp task untied firstprivate(i, …) shared (……,num)
5     {
6      ……
7       QProcess* m_process = new QProcess();
        // образование объекта – нового Qt-процесса
8       m_process->setProcessChannelMode(QProcess::MergedChannels);
       // образование канала для вывода протоколов i-го процесса
9       m_process->setStandardOutputFile("Qt_protocol.log");
       // Qt_protocol.log – протокол i-го блока
10      QString strCommand = NameProg;
       // NameProg – имя программы минимизации
       // основная часть командной строки для запуска дочернего i-го процесса
11      QString strCommand1;
       // объект для основной части команды запуска программы
```

```
12          strCommand1+="-i"+W_in.data(); // исходный sf-файл i-го блока
13          strCommand1+="-o"+W_out.data();
          // sf-файл - результат минимизации i-го блока;
14          strCommand+=strCommand1;
          // основная часть строки запуска процесса минимизации
15          .....
          // формирование командной строки для запуска программ минимизации
16        if(NameProg=="Minim")  // для программы Minim
17        {
18          strCommand +="-r"+ config.ini";
19        }
20        if(NameProg=="MinimPar")  // для программы MinimPar
21        {
22          strCommand +="-r"+ config.ini";
23        }
24        if(NameProg=="Espresso")  // для программы Espresso
25        {
26          ...// подготовка дополнительных параметров программы Espresso
27          strCommand +="-f"+"Probe.prb";
28        }
29        m_process->start(strCommand);  // запуск i-го процесса
30        m_process->waitForFinished(-1);
31        rc = m_process->exitCode();
32        }  // конец omp-директивы task
33    }      // конец цикла по числу блоков (Qt-процессов)
34    foo1(.....); // собрать время всех Qt-процессов в один протокол
35    foo2(.....);
      // собрать сеть с результатами минимизации для всех Qt-процессов
```

Программа StartParBlock организует запуск в параллельном режиме процессов, являющихся контейнерами для исполняемых модулей различных программ минимизации. В листинге 2 приведены команды запуска для программ минимизации Minim, MinimPar и Espresso. Описанием входных данных для программы StartParBlock является многоуровневое представление системы булевых функций в виде иерархического двухуровневого SF-описания в формате 2-CONNECT [22]. Каждый листовой блок представляет отдельную выделенную подсистему булевых функций. Программа StartParBlock подготавливает элементы исходного описания в отдельные блоки с автономными SF-описаниями в формате SDF. Именно с SDF-форматом работают исполняемые модули программ Minim, MininPar, Esspresso и др. Программа StartParBlock подготавливает также командную строку для запуска соответствующей программы минимизации и организует параллельную работу процессов-контейнеров.

Реализация программ MinimPar и StartParBlock осуществлена на языке программирования C++(11.0), реализация директив и библиотек OpenMP (версии 3.1) – в среде Qt [29] с компилятором Mingw (версии 5.7.0). Эксперименты для программ минимизации с распараллеливанием вычислений выполнялись в 64-разрядной операционной системе Windows7 на четырехъядерных процессорах Intel Core 7-4820K с тактовой частотой 3,7 ГГц, ОЗУ 32 ГБ с включением режима гипертрейдинга.

Эксперименты с распараллеливанием вычислений

Эксперимент 8. Распараллеливание вычислений при раздельной минимизации с использованием программы Espresso. По исходному SF-описанию системы функций строилась логическая сеть в формате 2-CONNECT, блоками которой являлись компонентные функции системы. Данная сеть подавалась на вход программы StartParBlock, которая запускала программу Espresso для каждого листового блока в двух режимах: последовательной и параллельной обработки листовых описаний. Результаты эксперимента 8 представлены в табл. 11.

Таблица 11
Результаты эксперимента 8. Распараллеливание вычислений
при раздельной минимизации с использованием программы Espresso

Table 11
Results of experiment 8. Parallelization of calculations
with separate minimization using the Espresso program

Исходные данные *Initial data*					Раздельная минимизация *Separate minimization*			
					Сеть компонентных функций *Network of Component Functions*		Сеть компонентных функций *Network of Component Functions*	
					Последовательная обработка *m* блоков сети программой Espresso *Sequential processing of m network blocks by the Espresso program*		Параллельная обработка *m* блоков сети программой Espresso *Parallel processing of m network blocks by Espresso program*	
Пример *Example*	*n*	*m*	*k*	$\sum_{i=1}^{m} k_i$	$\sum_{i=1}^{m} k_{\min}^{i}$	Время, с *Time, s*	$\sum_{i=1}^{m} k_{\min}^{i}$	Время, с *Time, s*
Второй набор примеров								
S12	12	12	4 096	27 238	2 921	2,34	2 921	*0,69
S13	13	13	8 192	58 616	5 548	4,14	5 548	*1,14
S14	14	14	16 384	125 497	10 572	11,65	10 572	*2,78
S15	15	15	32 768	267 551	20 150	45,46	20 150	*11,01
S16	16	16	65 536	568 043	38 636	188,46	38 636	*44,11
System8	25	20	45 548	45 947	32 378	143,64	32 378	*39,37
Третий набор примеров								
Psevdo1	10	20	1 000	2 103	1 011	2,20	1 011	*0,69
Psevdo2	15	20	5 000	10 676	9 574	4,57	9 574	*1,24
Psevdo3	20	20	10 000	26 657	20 320	33,52	20 320	*8,37
Psevdo4	25	20	25 000	92 300	23 938	198,15	23 938	*56,36
Psevdo5	30	20	30 000	150 620	45 809	941,62	45 809	*286,00
Четвертый набор примеров								
mult_7_*DNF*	14	14	13 060	13 252	8 133	7,76	8 133	*1,79
mult_7_*TABL*	14	14	16 384	93 334	8 102	10,57	8 102	*2,18
mult_8_*DNF*	16	16	52 810	53 621	30 600	102,59	30 600	*29,26
mult_8_*TABL*	16	16	65 536	434 660	30 538	169,64	30 538	*39,69
mult_9_*DNF*	18	18	208 598	212 047	113 128	2 999,20	113 128	*1 183,30
mult_9_*TABL*	18	18	262 144	1 302 835	112 850	4 692,82	112 850	*1 514,90
square_12_*DNF*	12	24	5 612	5 902	3 350	3,40	3 350	*0,71
square_12_*TABL*	12	24	4 096	39 962	3 326	3,69	3 326	*0,84
square_13_*DNF*	13	26	11 259	11 904	6 501	5,32	6 501	*1,11
square_13_*TABL*	13	26	8 191	87 928	6 455	6,28	6 455	*1,67
square_14_*DNF*	14	28	22 867	24 151	12 452	10,91	12 452	*2,71
square_14_*TABL*	14	28	16 384	192 100	12 401	15,25	12 401	*3,68
square_15_*DNF*	15	30	45 568	48 045	23 841	31,91	23 841	*8,44
square_15_*TABL*	15	30	32 768	416 811	23 711	52,62	23 711	*12,27

Эксперимент 9. Распараллеливание вычислений при раздельной минимизации с использованием программ Minim, MinimPar. Результаты эксперимента представлены в табл. 12. Суммарные длины ДНФ для минимизированных систем функций часто различаются, однако все результаты минимизации являются корректными решениями. Все решения были проверены, и установлена их функциональная эквивалентность с исходными заданиями на минимизацию. Различия в решениях (суммарных длинах ДНФ) объясняются тем, что программа Minim реализует макроалгоритм [18], который в зависимости от сочетания различных параметров может выполнять процедуры минимизации в разной последовательности из-за того, что исходные задания для программ (табл. 12) различаются. Для моноблока – это пара матриц, для сети – иерархические описания, в которых описания отдельных функций являются листовыми.

Эффективность реализации распараллеливания первым и вторым способами примерно одинакова. Так, второй способ распараллеливания немного выигрывает по быстродействию у первого. Вместе с тем результаты раздельной минимизации, получаемые первым способом, лучше. Естественно, что по сравнению с Minim параллельный вариант MinimPar является более быстродействующим: время работы этой программы примерно в пять раз меньше.

Эксперимент 10. Сравнение эффективности и трудоемкости программ совместной минимизации Espresso и Minim. Результаты эксперимента представлены в табл. 13.

Эксперимент 10 показывает значительное преимущество программы Espresso по сравнению с программой Minim при решении задачи совместной минимизации как по качеству, так и по времени решения. Только в одном случае (пример Psevdo5) программа Espresso не смогла получить решение, проработав 7,5 ч (27 000 с). Обе сравниваемые в табл. 13 программы быстрее обрабатывают исходные системы ДНФ (*_DNF*), чем таблицы истинности (*_TABL*).

Отметим также то, что предварительная совместная минимизация функций в классе ДНФ позволяет сокращать время BDD-оптимизации. Например, выполняя BDD-оптимизацию с помощью программы BDD-Builder, входящей в систему FLC-2, время получения BDD-представления для примера mult_9*_DNF* сокращается в шесть раз по сравнению с получением того же BDD-представления, когда в качестве исходных данных выступает таблица истинности mult_9_*TABL*.

Сравнение результатов экспериментальных исследований. Результаты экспериментов для различных модификаций программы Espresso (-II, -IIC, -MV, -Exact) опубликованы в работе [12]. Однако в этой работе, а также в работе [18] приведено лишь общее время решения всех 134 примеров и не представлено время обработки для отдельных примеров. Указывается, что программа Espresso-Exact нашла точное решение для 104 примеров.

В публикации [30] описывается метод минимизации булевых функций с большим числом импликант (более миллиона), использующий представление множества импликант в виде троичного дерева, на котором выполняются операции слияния листьев. Процедура слияния листьев соответствует слиянию двоичных наборов из таблицы истинности, которые отличаются только одной последней переменной, чем достигается ускорение выполнения операций. Предложенный алгоритм реализован в программе TT-Min. Приводятся результаты экспериментальных исследований на стандартных примерах Benchmark в сравнении с программой Espresso. Показано, что программа TT-Min проигрывает программе Espresso по качеству получаемого решения, но если ее использовать для предварительной обработки функции, то совместное время (TT-Min+Espresso) может оказаться лучше, чем время, затраченное только программой Espresso. Целесообразно также использовать программу TT-Min для функций, порождающих большое число импликант.

Программа BOOM [31, 32] эффективна для функций с несколькими сотнями входных переменных, значения которых определены лишь на небольшой части булева пространства. Эксперименты показали, что она дает лучшие результаты для больших схем, чем программа Espresso. Программа BOOM-II [33] объединяет две программы минимизации: BOOM и FC-Min [34]. Для программы BOOM-II проведено экспериментальное сравнение с программой Espresso.

Таблица 12

Результаты эксперимента 9. Распараллеливание вычислений при раздельной минимизации с использованием программ Minim и MinimPar

Table 12

Results of experiment 9. Parallelization of calculations with separate minimization using Minim and MinimPar programs

Исходные данные *Initial data*					Моноблок *Monoblock*		Сеть *Net*			
					Первый способ распараллеливания MinimPar *The first way to parallelize MinimPar*		Последовательная обработка *m* блоков сети программой Minim *Sequential processing of m network blocks by the Minim program*		Второй способ распараллеливания. Параллельная обработка *m* блоков сети программой MinimPar *The second way of parallelization. Parallel processing of m network blocks by the MinimPar program*	
Пример *Example*	*n*	*m*	*k*	$\sum_{i=1}^{m} k_i$	$\sum_{i=1}^{m} k_{\min}^{i}$	Время, с *Time, s*	$\sum_{i=1}^{m} k_{\min}^{i}$	Время, с *Time, s*	$\sum_{i=1}^{m} k_{\min}^{i}$	Время, с *Time, s*
Второй набор примеров										
S12	12	12	4 096	27 238	*2 916	19,9	2 944	67,57	2 944	*19,17
S13	13	13	8 192	58 616	*5 613	39,47	5 653	167,65	5 655	*36,51
S14	14	14	16 384	125 497	12 737	81,16	*11 896	340,63	12 767	*58,31
S15	15	15	32 768	267 551	*26 694	334,20	26 737	1 064,68	26 818	*274,52
S16	16	16	65 536	568 043	53 521	1 819,80	*53 420	4 449,4	53 694	*1 023,59
System8	25	20	45 548	45 947	37 070	759,30	*36 949	2 664,47	37 083	*700,19
Третий набор примеров										
Psevdo1	10	20	1 000	2 103	1 059	*0,40	1 059	1,42	1 059	0,43
Psevdo2	15	20	5 000	10 676	9 604	22,44	9 604	97,83	9 604	*17,41
Psevdo3	20	20	10 000	26 657	20 320	100,26	20 320	420,75	20 320	*79,87
Psevdo4	25	20	25 000	92 300	23 938	100,70	23 938	447,22	23 938	*81,50
Psevdo5	30	20	30 000	150 620	45 809	244,92	45 809	1 517,54	45 809	*201,13
Четвертый набор примеров										
mult_7_*DNF*	14	14	13 060	13 252	8 268	84,21	8 298	348,113	8 268	*78,84
mult_7_*TABL*	14	14	16 384	93 334	8 289	94,63	8 255	324,145	8 312	*72,07
mult_8_*DNF*	16	16	52 810	53 621	36 005	691,33	36 018	2 362,42	36 018	*623,49
mult_8_*TABL*	16	16	65 536	434 660	38 057	937,68	37 800	2 315,10	38 170	*711,59
square_12_*DNF*	12	24	5 612	5 902	3 373	11,05	3 380	32,90	3 380	*10,27
square_12_*TABL*	12	24	4 096	39 962	3 359	12,57	3 368	36,48	3 368	*11,85
square_13_*DNF*	13	26	11 259	11 904	6 570	69,85	6 576	216,57	6 578	*65,04
square_13_*TABL*	13	26	8 191	87 928	6 555	57,64	6 561	174,05	6 561	*54,79
square_14_*DNF*	14	28	22 867	24 151	12 960	95,71	12 794	442,90	13 058	*87,37
square_14_*TABL*	14	28	16 384	192 100	12 941	118,00	12 762	476,20	12 980	*94,17
square_15_*DNF*	15	30	45 568	48 045	27 601	285,467	27 493	907,76	27 636	*218,01
square_15_*TABL*	15	30	32 768	416 811	28 464	334,685	28 094	1 038,20	28 497	*270,45

Таблица 13
Результаты эксперимента 10. Сравнение программ Espresso и Minim совместной минимизации

Table 13
Results of experiment 10. Comparison of Espresso and Minim joint minimization programs

Пример *Example*				Совместная минимизация *Joint minimization*			
				Espresso		Minim	
Пример *Example*	*n*	*m*	*k*	k_{min}	Время, с *Time, s*	k_{min}	Время, с *Time, s*
Второй набор примеров							
S12	12	12	4 096	*2 523	*6,67	2 685	89,96
S13	13	13	8 192	*4 929	*28,77	5 836	239,38
S14	14	14	16 384	*9 672	*176,32	11 596	1 142,07
S15	15	15	32 768	*18 768	*940,39	22 902	6 564,36
S16	16	16	65 536	*36 509	*5 462,30	–	–
System8	25	20	45 548	*28 668	*2 113,57	37 048	19 336,82
Третий набор примеров							
Psevdo1	10	20	1 000	*509	*1,83	704	12,98
Psevdo2	15	20	5 000	*4 754	*16,08	4 770	168,67
Psevdo3	20	20	10 000	*7 639	*333,96	*7 639	556,92
Psevdo4	25	20	25 000	*6 522	1 464,04	*6 522	*409,83
Psevdo5	30	20	30 000	Нет решения	27 000,00	*9 132	*822,08
Четвертый набор примеров							
mult_7_*DNF*	14	14	13 060	*7 224	*47,058	9 312	691,366
mult_7_*TABL*	14	14	16 384	*7 795	*147,727	9 827	852,310
mult_8_*DNF*	16	16	52 810	*27 161	*1 009,221	36 001	11 109,94
mult_8_*TABL*	16	16	65 536	*29 363	*2 433,198	Нет решения	14 400,00
mult_9_*DNF*	18	18	208 598	Нет решения	6 800,00	–	–
square_12_*DNF*	12	24	5 612	*2 801	*7,310	3 482	83,04
square_12_*TABL*	12	24	4 096	3 393	*13,565	*3 304	95,59
square_13_*DNF*	13	26	11 259	*5 483	*26,457	7 370	295,25
square_13_*TABL*	13	26	8 191	*6 847	*86,345	7 333	255,55
square_14_*DNF*	14	28	22 867	*10 681	*102,202	14 471	555,18
square_14_*TABL*	14	28	16 384	*13 932	*438,661	14 671	5 759,00
square_15_*DNF*	15	30	45 568	*20 664	*425,612	28 151	3 959,00
square_15_*TABL*	15	30	32 768	*28 407	*1 848,140	29 458	4 661,07

В алгоритме Scherzo [35] булевы функции представлены в виде двух графов BDD, причем размер этих графов напрямую не связан с числом элементов, которые они описывают. Это позволяет сократить размер используемой памяти и уменьшить время работы программы. Разработано новое циклическое ядро для работы с этими двумя графами. Алгоритм применяет новую нижнюю границу в методе ветвей и границ, основанную на отсечении возможных решений, что позволяет сократить число рекурсивных вызовов процедуры поиска. Представлены результаты экспериментальных исследований описанного алгоритма, реализованного в программе Scherzo, на примерах серии MCNC Benchmark в сравнении с программами Espresso-Signature и Espresso-Exact. Показано [35], что программа Scherzo выигрывает по времени у программ Espresso-Signature и Espresso-Exact на большинстве примеров, однако не улучшает качества получаемых решений.

В работе [36] для минимизации используются два подхода: BDD-оптимизации и BDD-дизъюнктивной декомпозиции, программа получила название MUSASHI. На стандартных примерах проведено экспериментальное сравнение этих программ минимизации.

Применение программ выполнимости конъюнктивной нормальной формы (SAT-solvers) и процедур программы Espresso позволило создать мощный логический минимизатор SAT-Espresso. В работе [37] утверждается что SAT-Espresso работает в 5–20 раз быстрее, чем Espresso, и в 3–5 раз быстрее, чем минимизатор BOOM на множестве больших примеров. Про-

грамма Espresso используется не только для минимизации, но и для интеллектуального анализа данных [38]. В статье [39] проведено сравнение разработанной для решения задач маршрутизации в сетях компьютеров программы минимизации с программами Espresso и Rondo на 16 стандартных примерах. Для примеров подсчитаны значения числа простых импликант, что показывает большую комбинаторную сложность нахождения точных решений задачи совместной минимизации. Утверждается, что полученные приближенным алгоритмом решения отличаются на 3–5 % от точных решений, однако точные решения не приводятся.

В лаборатории логического проектирования Объединенного института проблем информатики НАН Беларуси традиционно развивались методы решения задач минимизации систем булевых функций в классе ДНФ. В монографии [6] описаны точные и приближенные методы минимизации частичных и слабо определенных булевых функций. Предложены методы, усовершенствующие алгоритм Квайна – МакКласки. Для совместной минимизации систем булевых функций предложен метод конкурирующих интервалов – последовательное построение простых импликант, включаемых в решение, путем расширения интервала, представляющего импликанту, за счет включения в него непокрытых элементарных конъюнкций. В работе [3] описан метод минимизации частичных булевых функций. Частичная булева функция представляется в виде двух 2^n-компонентных булевых векторов. В основе метода лежит технология [40] параллельных операций над соседними элементами в булевом пространстве многих переменных, позволяющая быстро находить в множестве единичных значений функции элементы с малым числом соседей и определяемые ими импликанты минимизируемой булевой функции. Алгоритм ориентирован на функции с числом переменных до 24. В работах [41, 42] развивается подход к минимизации систем функций на основе аппарата покрытий троичных матриц, используемого также при решении задач декомпозиции систем функций. Опубликованные в статье [41] результаты экспериментов по совместной минимизации, проведенных на 10 примерах из первого набора примеров (табл. 14), вполне согласуются с результатами, представленными в настоящей статье. Программа из работы [41] выигрывает у Minim, получает одинаковые решения с программой Tie и лучшие решения, чем у Espresso. Преимуществом программы из статьи [41] перед Tie является то, что она может обрабатывать примеры с числом переменных $n > 22$, в то время как программа Tie обрабатывает примеры, для которых $n \le 22$.

Таблица 14
Сравнение программ совместной минимизации

Table 14
Comparison of joint minimization programs

Пример *Example*	*n*	*m*	*k*	Программа [41] k_{min} *Program [41]* k_{min}	Minim k_{min}	Espresso k_{min}	Tie k_{min}
Max512	9	6	512	*133	153	141	*133
Alu1	12	8	19	*19	*19	*19	*19
Mp2d	14	14	123	*30	36	31	*30
B12	15	9	431	*41	48	42	*41
Gary	15	11	442	*107	128	*107	*107
B2	16	17	110	*104	105	*104	*104
In2	19	10	137	*134	135	137	*134
In5	24	14	62	*62	*62	*62	–
In7	26	10	84	*54	55	*54	–
Chkn	29	7	153	*140	*140	141	–

Программные эксперименты, проведенные в работе [18], касались сравнения программы Minim с программой Espresso MV на 12 примерах, среди которых z5xp1, Radd, Root и др. Были получены одинаковые результаты совместной минимизации при решении задачи 1. Однако сравнение проводилось на примерах небольшой размерности ($n \le 10$), был только один пример с числом входных переменных $n = 25$. Другие эксперименты касались исследования эффективности программы Minim для различных режимов ее работы на потоках псевдослучайных примеров. В работе [43] проведено детальное сравнение программ Espresso и Minim на двух при-

мерах: test2 (35 функций от 11 переменных) и pdc (40 функций от 16 переменных) – по числу импликант, размеру матрицы покрытия Квайна и по результатам минимизации, а также по сравнению техники программирования, использованной при создании упомянутых программ. В работе [10] дано сравнение программы Tie с программой Espresso и показана ее конкурентоспособность для примеров $n \le 10$ как по качеству, так и по времени нахождения решений.

В работе [44] описан подход к распараллеливанию вычислений при программной реализации метода Квайна – МакКласки, однако приведено мало экспериментальных данных. Для учебных целей разработано много программ минимизации, описания некоторых из них представлены в методических работах [45–47].

По итогам сравнения результатов экспериментов можно сделать следующие выводы. Программа ABC является быстродействующей программой раздельной минимизации, но если на вход подается система ДНФ большой размерности, целесообразно для сокращения времени вычислений разбить данную систему на подсистемы, содержащие по несколько либо по одной функции.

Программа Espresso по-прежнему остается эффективным инструментом минимизации при числе переменных, большем 20, и числе элементарных конъюнкций в несколько тысяч. Предварительная раздельная минимизация с помощью программы ABC часто (но не всегда) позволяет сокращать время работы программы Espresso для совместной минимизации функций. Несомненным достоинством Espresso является не только ее быстродействие, но и устойчивость получения одинаковых решений для различных форм исходного задания – систем ДНФ, ортогонализованных ДНФ, систем СДНФ (таблиц истинности).

Эффективными являются программа Tie совместной минимизации для систем функций, зависящих не более чем от 20 переменных, и программа Minim, позволяющая быстрее получать решения задачи раздельной минимизации в диапазоне большой размерностей, когда вычисления с помощью Espresso требуют много времени. Вместе с тем программа Minim чувствительна к форме исходного задания минимизируемых систем функций. Она получает лучшие решения при условии, что на ее вход подаются системы ДНФ. Если же входными данными для Minim являются системы СДНФ, то качество решений программы может незначительно ухудшиться.

Оба подхода к распараллеливанию вычислений при раздельной минимизации обеспечивают сокращение времени получения решений. Второй подход к распараллеливанию является универсальным и может применяться не только для блочной минимизации функций в классе ДНФ, но и для блочной (и раздельной – по одной функции) оптимизации подсистем функций, выполняемой на основе логической минимизации многоуровневых представлений систем булевых функций.

Можно выделить области предпочтительного использования программ. Если число переменных минимизируемой системы функций не превышает 20, то целесообразно применять программу Tie как для совместной, так и раздельной минимизации. Если же число переменных больше 20 и число конъюнкций в исходном задании не превышает 10 000, то надо применять Espresso. Если данная программа будет работать долго, то при раздельной минимизации следует применить Minim. Для обработки примеров, в которых число конъюнкций превышает 50 000, целесообразно применять ABC. Если же требуется совместная минимизация, то надо выполнить сначала программу ABC, затем Espresso (однако такой подход не всегда дает сокращение времени). Для раздельной минимизации конкуренцию ABC может составить распараллеливание работы Espresso. Могут быть случаи, когда применение MinimPar даст решение быстрее, чем Espresso (в проведенных экспериментах это были псевдослучайные системы ДНФ). В очень ответственных случаях следует учитывать диапазоны применения программ и проводить вычисления, используя все имеющиеся подходы и программы, после чего выбирать лучшее решение.

Заключение. Система FLC-2 логической оптимизации, содержащая эффективные отечественные и зарубежные свободно распространяемые программы минимизации ДНФ представлений систем булевых функций, а также программные средства, позволяющие разбивать исходные системы функций на подсистемы и обрабатывать иерархические описания логических

сетей, обеспечивает решение задач минимизации функций в классе ДНФ практической размерности за приемлемое время. Благодаря распараллеливанию вычислений появляется возможность дополнительно сокращать время вычислений и увеличивать размерности решаемых задач раздельной минимизации функций. Выявлены области предпочтительного использования программ для систем булевых функций, характеризуемых как относительно небольшими значениями параметров (не более 20 переменных, менее 1000 элементарных конъюнкций), так и большими значениями (десятки аргументов и функций, десятки тысяч элементарных конъюнкций).

Вклад авторов. *И. П. Логинова* – разработка программных средств для распараллеливания программ, выполнение соответствующих экспериментов, подготовка разделов статьи, относящихся к распараллеливанию программ. *П. Н. Бибило* – проведение экспериментов без распараллеливания программ минимизации, подготовка остальных разделов статьи.
Авторы выражают благодарность В. И. Романову за помощь в программной генерации псевдослучайных систем ДНФ и реализации программной оценки сложности результатов раздельной минимизации систем функций.

Список использованных источников

1. Quine, W. V. The problem of simplifying of truth functions / W. V. Quine // The American Mathematical Monthly. – 1952. – Vol. 59, no. 8. – P. 521–531.

2. McCluskey, E. J. Minimization of Boolean functions / E. J. McCluskey // The Bell System Technical J. – 1956. – Vol. 35, no. 6. – P. 1417–1444.

3. Закревский, А. Д. ДНФ-реализация частичных булевых функций многих переменных / А. Д. Закревский, Н. Р. Торопов, В. И. Романов // Информатика. – 2010. – № 1(25). – С. 102–111.

4. Сапоженко, А. А. Минимизация булевых функций в классе дизъюнктивных нормальных форм / А. А. Сапоженко, И. П. Чухров // Итоги науки и техники. Теория вероятностей. Математическая статистика. Теоретическая кибернетика. – 1987. – Т. 25. – С. 68–116.

5. Брейтон, Р. К. Синтез многоуровневых комбинационных логических схем / Р. К. Брейтон, Г. Д. Хэчтел, А. Л. Санджованни-Винчентелли // ТИИЭР. – 1990. – Т. 78, № 2. – С. 38–83.

6. Закревский, А. Д. Логический синтез каскадных схем / А. Д. Закревский. – М. : Наука, 1981. – 416 с.

7. Бибило, П. Н. Применение диаграмм двоичного выбора при синтезе логических схем / П. Н. Бибило. – Минск : Беларус. навука, 2014. – 231 с.

8. Авдеев, Н. А. Эффективность логической оптимизации при синтезе комбинационных схем из библиотечных элементов / Н. А. Авдеев, П. Н. Бибило // Микроэлектроника. – 2015. – Т. 44, № 5. – С. 383–399.

9. Бибило, П. Н. Система логической оптимизации функционально-структурных описаний цифровых устройств на основе продукционно-фреймовой модели представления знаний / П. Н. Бибило, В. И. Романов // Проблемы разработки перспективных микро- и наноэлектронных систем. – 2020. – Вып. 4. – С. 9–16.

10. Леончик, П. В. Минимизация систем булевых функций в классе дизъюнктивных нормальных форм / П. В. Леончик // Информатика. – 2006. – № 1(9). – С. 88–96.

11. Logic Minimization Algorithm for VLSI Synthesis / K. R. Brayton [et al.]. – Boston : Kluwer Academic Publishers, 1984. – 193 p.

12. Rudell, R. Multiple-valued minimization for PLA optimization / R. Rudell, A. L. Sangiovanni-Vincentelli // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – 1987. – Vol. CAD-6, no. 5. – P. 727–751.

13. McGeer, P. Espresso-signature: A new exact minimizer for logic functions / P. McGeer, A. L. Sangiovanni-Vincentelli // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. – 1993. – Vol. 1, no. 4. – P. 618–624.

14. Гольдберг, Е. И. Метод сбрасывания решения с локального минимума при минимизации интервального покрытия / Е. И. Гольдберг. – Минск : Ин-т техн. кибернетики АН Беларуси, 1991. – 18 с. (Препринт № 13).

15. Mishchenko, A. An Introduction to Zero-Suppressed Binary Decision Diagrams / A. Mishchenko. – Berkeley : University of California, 2014. – 15 p.

16. Карпов, Ю. Г. Model checking. Верификация параллельных и распределенных программных систем / Ю. Г. Карпов. – СПб. : БХВ-Петербург, 2010. – 560 с.

17. Леончик, П. В. Алгоритм покрытия разреженных булевых матриц / П. В. Леончик // Информатика. – 2007. – № 2(14). – С. 53–61.

18. Торопов, Н. Р. Минимизация систем булевых функций в классе ДНФ / Н. Р. Торопов // Логическое проектирование : сб. науч. тр. – Минск : Ин-т техн. кибернетики НАН Беларуси, 1999. – Вып. 4. – С. 4–19.

19. Торопов, Н. Р. Приближенный алгоритм минимизации систем слабоопределенных булевых функций / Н. Р. Торопов // Известия АН СССР. Техническая кибернетика. – 1969. – № 1. – С. 72–78.

20. Романов, В. И. Булевы векторы и матрицы в С++ / В. И. Романов, И. В. Василькова // Логическое проектирование : сб. науч. тр. – Минск : Ин-т техн. кибернетики НАН Беларуси, 1997. – Вып. 2. – С. 150–158.

21. Черемисинов, Д. И. Троичные векторы и матрицы в С++ / Д. И. Черемисинов, Л. Д. Черемисинова // Логическое проектирование : сб. науч. тр. – Минск : Ин-т техн. кибернетики НАН Беларуси, 1998. – Вып. 3. – С. 146–156.

22. Бибило, П. Н. Логическое проектирование дискретных устройств с использованием продукционно-фреймовой модели представления знаний / П. Н. Бибило, В. И. Романов. – Минск : Беларус. навука, 2011. – 279 с.

23. Закревский, А. Д. Генераторы псевдослучайных логико-комбинаторных объектов в С++ / А. Д. Закревский, Н. Р. Торопов // Логическое проектирование : сб. науч. тр. – Минск : Ин-т техн. кибернетики НАН Беларуси, 1999. – Вып. 4. – С. 49–63.

24. Кардаш, С. Н. Ортогонализация системы ДНФ булевых функций / С. Н. Кардаш // VII Междунар. науч.-практ. конф. «BIG DATA and Advanced Analytics» (BIG DATA 2021) : материалы междунар. науч. конф., Минск, Беларусь, 19–20 мая 2021 г. – Минск : БГУИР, 2021. – С. 26–30.

25. Торопов Н. Р. Преобразование многоярусной комбинационной сети в двухъярусную / Н. Р. Торопов // Логическое проектирование : сб. науч. тр. – Минск : Ин-т техн. кибернетики НАН Беларуси, 2000. – Вып. 5. – С. 4–14.

26. Williams, A. C++ Concurrency in Action: Practical Multithreading / A. Williams. – Manning Publications, 2012. – 528 p.

27. Darryl, G. Multicore Application Programming: for Windows, Linux, and Oracle Solaris / G. Darryl. – Addison-Wesley, 2010. – 480 p.

28. The design of OpenMP tasks / E. Ayguade [et al.] // IEEE Transactions on Parallel and Distributed Systems. – 2009. – Vol. 20, no. 3. – P. 404 –418.

29. Шлее, М. Qt 5.10. Профессиональное программирование на С++ / М. Шлее. – СПб. : БХВ-Петербург, 2018. – 1072 с.

30. Fišer, P. A fast SOP minimizer for logic functions described by many product terms / P. Fišer, D. Toman // Proc. of 12th Euromicro Conf. on Digital Systems Design (DSD'09), Patras, 27–29 Aug. 2009. – Patras, 2009. – P. 757–764.

31. Fišer, P. Two-level Boolean minimizer BOOM-II / P. Fišer, H. Kubatova // Proc. of the 6th Intern. Workshop on Boolean Problems (IWSBP'04), Freiberg, Germany, 23–24 Sept. 2004. – Freiberg, 2004. – P. 221–228.

32. Hlavicka, J. BOOM – a heuristic Boolean minimizer / J. Hlavicka, P. Fiser // Computers and Information. – 2003. – Vol. 22, no. 1. – P. 19–51.

33. Fišer, P. Flexible two-level Boolean minimizer BOOM-II and its applications / P. Fišer, H. Kubatova // Proc. of 9th Euromicro Conf. on Digital System Design (DSD'06), Washington, USA, 30 Aug. – 1 Sept. 2006. – Washington, 2006. – P. 369–376.

34. Fišer, P. FC-Min: A fast multi-output Boolean minimizer / P. Fišer, J. Hlavička, H. Kubatova // Proc. Euromicro Symp. on Digital Systems Design (DSD'03), Belek-Antalya, Turkey, 3–5 Sept. 2003. – Belek-Antalya, 2003. – P. 451–454.

35. Coudert, O. Doing two-level logic minimization 100 times faster / O. Coudert // Discrete Algorithms : Proc. of the Sixth Annual ACM-SIAM Symp., San Francisco, California, USA, 21 Jan. 1995. – San Francisco, 1995. – P. 112–121.

36. Mishchenko, A. Large-scale SOP minimization using decomposition and functional properties / A. Mishchenko, T. Sasao // Proc. of the 40th Design Automation Conf. (DAC 2003), Anaheim, California, 2–6 June 2003. – Anaheim, 2003. – P. 149–154.

37. Sapra, S. SAT-based algorithms for logic minimization / S. Sapra, M. Theobald, E. Clarke // Proc. 21st Intern. Conf. on Computer Design, San Jose, California, 13–15 Oct. 2003. – San Jose, 2003. – P. 510–517.

38. Ashmouni, E. F. Espresso for rule mining / E. F. Ashmouni, R. Ramadan, A. A. Rashed // The 5th Intern. Conf. on Ambient Systems, Networks and Technologies (ANT–2014), Hasselt, Belgium, 2–5 June 2014. – Hasselt, 2014. – P. 596–603.

39. Смагин, А. А. Применение методов минимизации булевых функций для оптимизации цифровых устройств / А. А. Смагин, А. В. Шиготаров // Изв. Самарск. науч. центра РАН. – 2009. – Т. 11, № 3(2). – С. 343–349.

40. Закревский, А. Д. Вычисления в многомерном булевом пространстве / А. Д. Закревский. – Минск : ОИПИ НАН Беларуси, 2011. – 106 с.

41. Поттосин, Ю. В. Метод минимизации системы полностью определенных булевых функций / Ю. В. Поттосин, Н. Р. Торопов, Е. А. Шестаков // Информатика. – 2008. – № 2(18). – С. 102–110.

42. Поттосин, Ю. В. Метод минимизации системы не полностью определенных булевых функций / Ю. В. Поттосин, Н. Р. Торопов, Е. А. Шестаков // Информатика. – 2009. – № 3(23). – С. 16–26.

43. Черемисинов, Д. И. Сравнение двух программ минимизации булевых функций / Д. И. Черемисинов // Автоматизация проектирования дискретных систем (CAD DD'10) : материалы Седьмой Междунар. конф., Минск, 16–17 нояб. 2010 г. – Минск : ОИПИ НАН Беларуси, 2010. – С. 194–200.

44. Модификация метода минимизации булевых функций для мультиядерной INTEL-архитектуры / С. В. Михтонюк [и др.] // Радиоэлектроника и информатика. – 2007. – № 3. – С. 50–55.

45. Alharbi, E. Truth graph: A novel method for minimizing Boolean algebra expressions by using graphs / E. Alharbi // Proc. 11th Intern. Conf. on the Theory and Application of Diagrams (Diagrams 2020), Tallinn, Estonia, 24–28 Aug. 2020. – Tallinn, 2020. – P. 461–469.

46. Михеева, Е. А. Минимизация булевых функций геометрическим методом / Е. А. Михеева, А. Ф. Еникеева // Ученые записки УлГУ. Сер. Математика и информационные технологии. – 2018. – № 1. – С. 72–82.

47. Riznyk, V. Minimization of Boolean functions by combinatorial method / V. Riznyk, M. Solomko // Technology audit and production reserves. Information and Control Systems. – 2017. – Vol. 4, no. (36). – P. 49–64.

References

1. Quine W. V. The problem of simplifying of truth functions. *The American Mathematical Monthly*, 1952, vol. 59, no. 8, pp. 521–531.

2. McCluskey E. J. Minimization of Boolean functions. *The Bell System Technical Journal*, 1956, vol. 35, no. 6, pp. 1417–1444.

3. Zakrevskij A. D., Toropov N. R., Romanov V. I. *DNF-implementation of partial Boolean functions of many variables*. Informatika [*Informatics*], 2010, no. 1(25), pp. 102–111 (In Russ.).

4. Sapozhenko A. A., CHuhrov I. P. *Minimization of Boolean functions in the class of disjunctive normal forms*. Itogi nauki i tekhniki. Teoriya veroyatnostej. Matematicheskaya statistika. Teoreticheskaya kibernetika, [*Results of Science and Technology. Probability Theory. Mathematical Statistics. Theoretical Cybernetics*], 1987, vol. 25, pp. 68–116 (In Russ.).

5. Brayton R. K., Hachtel G. D., Sangiovanni-Vincentelli A. L. *Synthesis of multi-level combinational logic circuits.* Trudy Institute inzhenerov po jelektronike i radiotehnike [*Proceedings of the Institute of Electronics and Radio Engineering*], 1990, vol. 78, no. 2, pp. 38–83 (In Russ.).

6. Zakrevskij A. D. Logicheskij sintez kaskadnyh skhem. *Logical Synthesis of Cascading Circuit.* Moscow, Nauka, 1981, 416 p. (In Russ.).

7. Bibilo P. N. Primenenie diagram dvoichnogo vybora pri sinteze logicheskih shem. *Application of Binary Decision Diagrams in the Synthesis of Logic Circuits*. Minsk, Belaruskaja navuka, 2014, 231 p. (In Russ.).

8. Avdeev N. A., Bibilo P. N. *Logical optimization efficiency in the synthesis of combinational circuits*. Mikroelektronika [*Microelectronics*], 2015, vol. 44, no. 5, pp. 383–399 (In Russ.).

9. Bibilo P. N., Romanov V. I. *The system of logical optimization of functional structural descriptions of digital circuits based on production-frame knowledge representation model*. Problemy razrabotki perspektivnyh mikro- i nanoelektronnyh system [*Problems of Developing Promising Micro- and Nanoelectronic Systems*], 2020, iss. 4, pp. 9–16 (In Russ.).

10. Leonchik P. V. *Minimization of Boolean function systems in the class of disjunctive normal forms*. Informatika [*Informatics*], 2006, no. 1(9), pp. 88–96 (In Russ.).

11. Brayton K. R., Hachtel G. D., McMullen C., Sangiovanni-Vincentelli A. L. *Logic Minimization Algorithm for VLSI Synthesis*. Boston, Kluwer Academic Publishers, 1984, 193 p.

12. Rudell R., Sangiovanni-Vincentelli A. L. Multiple-valued minimization for PLA optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1987, vol. CAD-6, no. 5, pp. 727–751.

13. McGeer P., Sangiovanni-Vincentelli A. L. Espresso-signature: A new exact minimizer for logic functions. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 1993, vol. 1, no. 4, pp. 618–624.

14. Gol'dberg E. I. Metod sbrasyvaniya resheniya s lokal'nogo minimuma pri minimizacii interval'nogo pokrytiya. *The Method of Resetting the Solution from the Local Minimum While Minimizing the Interval Coverage*. Minsk, Institut tehnicheskoj kibernetiki Akademii nauk Belarusi, 1991, 18 p. (Preprint no. 13) (In Russ.).

15. Mishchenko A. *An Introduction to Zero-Suppressed Binary Decision Diagrams*. Berkeley, University of California, 2014, 15 p.

16. Karpov Y. G. Model checking. Verifikaciya parallel'nyh i raspredelennyh programmnyh system. *Model Checking. Verification of Parallel and Distributed Software Systems*. Saint Petersburg, BHV-Peterburg, 2010, 560 p. (In Russ.).

17. Leonchik P. V. *Algorithm of sparse Boolean matrix covering*. Informatika [*Informatics*], 2007, no. 2(14), pp. 53–61 (In Russ.).

18. Toropov N. R. *Minimization of systems of Boolean functions in the class DNF*. Logicheskoe proektirovanie [*Logical Design*]. Minsk, Institut tehnicheskoj kibernetiki Nacional'noj akademii nauk Belarusi, 1999, iss. 4, pp. 4–19 (In Russ.).

19. Toropov N. R. *An approximate algorithm for minimizing systems of weakly defined Boolean functions*. Izvestija Akademii nauk SSSR. Tekhnicheskaya kibernetika [*Proceedings of the Academy of Sciences of the USSR. Technical cybernetics*], 1969, no. 1, pp. 72–78 (In Russ.).

20. Romanov V. I., Vasil'kova I. V. *Boolean vectors and matrices in C++*. Logicheskoe proektirovanie [*Logical Design*]. Minsk, Institut tehnicheskoj kibernetiki Nacional'noj akademii nauk Belarusi, 1997, iss. 2, pp. 150–158 (In Russ.).

21. Cheremisinov D. I., Cheremisinova L. D. Ternary vectors and matrices in C++. Logicheskoe proektirovanie [*Logical Design*]. Minsk, Institut tehnicheskoj kibernetiki Nacional'noj akademii nauk Belarusi, 1998, iss. 3, pp. 146–156 (In Russ.).

22. Bibilo P. N., Romanov V. I. Logicheskoe proektirovanie diskretnyh ustrojstv s ispol'zovaniem produkcionno-frejmovoj modeli predstavlenija znanij. *Logical Design of Discrete Devices with Use of Productional and Frame Model of Representation of Knowledge*. Minsk, Belaruskaja navuka, 2011, 279 p. (In Russ.).

23. Zakrevskij A. D., Toropov N. R. *Generators of pseudorandom logical-combinatorial objects in C++*. Logicheskoe proektirovanie [*Logical Design*], Minsk, Institut tehnicheskoj kibernetiki Nacional'noj akademii nauk Belarusi, 1999, iss. 4, pp. 49–63 (In Russ.).

24. Kardash S. N. *Orthogonalization of the DNF system of Boolean functions*. VII Mezhdunarodnaya nauchno-prakticheskaya konferenciya "BIG DATA and Advanced Analytics" (BIG DATA 2021) : materialy mezhdunarodnoj nauchnoj konferencii, Minsk, Belarus, 19–20 maya 2021 g. [*VII International Scientific and Practical Conference "BIG DATA and Advanced Analytics" (BIG DATA 2021) : Materials of the International Scientific Conference, Minsk, Belarus, 19–20 May 2021*]. Minsk, Belorusskij gosudarstvennyj universitet informatiki i radiojelektroniki, 2021, pp. 26–30 (In Russ.).

25. Toropov N. R. *Transformation of a multi-level combinational network into a two-level one*. Logicheskoe proektirovanie [*Logical Design*], Minsk, Institut tehnicheskoj kibernetiki Nacional'noj akademii nauk Belarusi, 2000, iss. 5, pp. 4–14 (In Russ.).

26. Williams A. *C++ Concurrency in Action: Practical Multithreading*. Manning Publications, 2012, 528 p.

27. Darryl G. *Multicore Application Programming: for Windows, Linux, and Oracle Solaris*. Addison-Wesley, 2010, 480 p.

28. Ayguade E., Copty N., Duran A., Hoeflinger J., Lin Y., …, Zhang G. The design of OpenMP tasks. *IEEE Transactions on Parallel and Distributed Systems*, 2009, vol. 20, no. 3, pp. 404–418.

29. SHlee M. Qt 5.10. Professional'noe programmirovanie na S++. *Qt 5.10. Professional programming in C++*. Saint Petersburg, BHV-Peterburg, 2018, 1072 p.

30. Fišer P., Toman D. A fast SOP minimizer for logic functions described by many product terms. *Proceedings of 12th Euromicro Conference on Digital Systems Design, Patras, 27–29 August 2009*. Patras, 2009, pp. 757–764.

31. Fišer P., Kubatova H. Two-Level Boolean minimizer BOOM-II. *Proceedings of the 6th International Workshop on Boolean Problems (IWSBP'04), Freiberg, Germany, 23–24 September 2004*. Freiberg, 2004, pp. 221–228.

32. Hlavicka J., Fišer P. BOOM – a heuristic Boolean minimizer. *Computers and Information*, 2003, vol. 22, no. 1, pp. 19–51.

33. Fišer P., Kubatova H. Flexible two-level Boolean minimizer BOOM-II and its applications. *Proceedings of 9th Euromicro Conference on Digital System Design (DSD'06), Washington, US, 30 August – 1 September 2006*. Washington, 2006, pp. 369–376.

34. Fišer P., Hlavička J., Kubatova H. FC-Min: A fast multi-output Boolean minimizer. *Proceedings* Euromicro Symposium on Digital Systems Design (DSD'03), Belek-Antalya, Turkey, 3–5 *September* 2003. Belek-Antalya, 2003, pp. 451-454.

35. Coudert O. Doing two-level logic minimization 100 times faster. *Discrete Algorithms: Proceedings of the Sixth Annual ACM-SIAM Symposium, San Francisco, California, USA, 21 January 1995*. San Francisco, 1995, pp. 112–121.

36. Mishchenko A., Sasao T. Large-scale SOP minimization using decomposition and functional properties. *Proceedings of the 40th Design Automation Conference (DAC 2003), Anaheim, California, 2–6 June 2003.* Anaheim, 2003, pp. 149–154.

37. Sapra S., Theobald M., Clarke E. SAT-based algorithms for logic minimization. *Proceedings* 21st International Conference on Computer Design, San Jose, California, 13–15 October 2003. San Jose, 2003, pp. 510–517.

38. Ashmouni E. F., Ramadan R. A., Rashed A. A. Espresso for rule mining. *The 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014), Hasselt, Belgium, 2–5 June 2014*. Hasselt, 2014, pp. 596–603.

39. Smagin A. A., SHigotarov A. V. *Application of Boolean function minimization methods to optimize digital devices*. Izvestiya Samarskogo nauchnogo centra Rossijskoj akademii nauk [*Proceedings of the Samara Scientific Center of the Russian Academy of Sciences*], 2009, vol. 11, no. 3(2), pp. 343–349.

40. Zakrevskij A. D. Vychisleniya v mnogomernom bulevom prostranstve. *Computations in a Multidimensional Boolean Space*. Minsk, Ob"edinjonnyj institut problem informatiki Nacional'noj akademii nauk Belarusi, 2011, 106 p. (In Russ.).

41. Pottosin Y. V., Toropov N. R., Shestakov E. A. *A method for minimizing a system of completely specified Boolean functions*. Informatika [*Informatics*], 2008, no. 2(18), pp. 102–110 (In Russ.).

42. Pottosin Y. V., Toropov N. R., Shestakov E. A. *A method for minimizing the system of incompletely specified Boolean functions*. Informatika [*Informatics*], 2009, no. 3(23), pp. 16–26 (In Russ.).

43. Cheremisinov D. I. *Comparison of two Boolean function minimization programs*. Avtomatizaciya proektirovaniya diskretnyh sistem (CAD DD'10): materialy Sed'moj Mezhdunarodnoj konferencii, Minsk, 16–17 noyabrya 2010 g. [*Discrete Systems Design Automation (CAD DD'10): Proceedings of the Seventh International Conference, Minsk, 16–17 November 2010*], Minsk, Ob"edinjonnyj institut problem informatiki Nacional'noj akademii nauk Belarusi, 2010, pp. 194–200.

44. Mihtonyuk S. V., Davydov M. D., Kuznecov E. S., Parfentij A. N. *Modification of the Boolean function minimization method for multi-core INTEL architecture*. Radioelektronika i informatika [*Radio Electronics and Computer Science*], 2007, no. 3, pp. 50–55 (In Russ.).

45. Alharbi E. Truth graph: A novel method for minimizing Boolean algebra expressions by using graphs. *Proceedings 11th International Conference on the Theory and Application of Diagrams (DIAGRAMS 2020), Tallinn, Estonia, 24–28 August 2020*. Tallinn, 2020, pp. 461–469.

46. Miheeva E. A., Enikeeva A. F. *Minimization of Boolean functions by the geometric method*. Uchenye zapiski Ul'janovskogo gosudarstvennogo universiteta. Serija Matematika i informacionnye tehnologii [*Scientific notes of Ulyanovsk State University. Series Mathematics and Information Technology*], 2018, no. 1, pp. 72–82.

47. Riznyk V., Solomko M. Minimization of Boolean functions by combinatorial method. *Technology Audit and Production Reserves. Information and Control Systems*, 2017, vol. 4, no. 2(36), pp. 49–64.

Информация об авторах

Бибило Петр Николаевич, доктор технических наук, профессор, Объединенный институт проблем информатики Национальной академии наук Беларуси.
E-mail: bibilo@newman.bas-net.by

Логинова Ирина Петровна, кандидат технических наук, доцент, Объединенный институт проблем информатики Национальной академии наук Беларуси.
E-mail: irilog@mail.ru

Information about the authors

Petr N. Bibilo, D. Sc. (Eng.), Professor, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.
E-mail: bibilo@newman.bas-net.by

Irina P. Loginova, Ph. D. (Eng.), The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.
E-mail: irilog@mail.ru