ISSN 1816-0301 (Print)
ISSN 2617-6963 (Online)
УДК 519.714; 519.7

Поступила в редакцию 27.11.2018
Received 27.11.2018

Принята к публикации 08.01.2019
Accepted 08.01.2019

Логическая минимизация булевых сетей с использованием разложения Шеннона

П. Н. Бибило✉, Ю. Ю. Ланкевич

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

Аннотация. Синтез логических схем, реализующих комбинационные блоки сверхбольших интегральных схем, – одна из важнейших задач компьютерного проектирования, так как размерность задач проектирования увеличивается, возрастает также время выполнения этапов синтеза логических схем. Особенно трудоемкой является глобальная технологическая независимая оптимизация – первый этап синтеза логической схемы. Суть второго этапа заключается в технологическом отображении оптимизированных логических представлений функций на логические элементы технологической библиотеки. Основные характеристики логической схемы, такие как площадь, быстродействие, энергопотребление, зависят от эффективности первого этапа. Эволюция методов глобальной логической оптимизации показала эффективность разложения Шеннона при оптимизации многоуровневых представлений систем полностью определенных булевых функций. Разработано множество методов и программ, использующих графические представления разложений Шеннона – BDD-представления. Большинство разработанных методов оптимизации BDD-представлений используют задания исходных систем булевых функций в виде дизъюнктивных нормальных форм (ДНФ).

Предлагается алгоритм минимизации числа вершин булевой сети, являющейся многоуровневым представлением системы полностью определенных булевых функций. Минимизация осуществляется на основе разложения Шеннона и поиска вершин сети, реализующих одинаковые и взаимно инверсные функции. Предложенный алгоритм логической оптимизации реализован в виде программы. Эксперименты показали, что данный алгоритм и полученную программу целесообразно использовать в случае, когда исходное многоуровневое представление функций невозможно представить (за приемлемое время работы компьютерной программы) в виде системы ДНФ либо когда система ДНФ, полученная из многоуровневого представления, содержит большое число (десятки и сотни тысяч) элементарных конъюнкций.

Ключевые слова: булева функция, булева сеть, разложение Шеннона, дизъюнктивная нормальная форма, синтез логических схем, BDD

Благодарности. Исследование выполнено при финансовой поддержке БРФФИ в рамках проекта № Ф19-023.

Для цитирования. Бибило, П. Н. Логическая минимизация булевых сетей с использованием разложения Шеннона / П. Н. Бибило, Ю. Ю. Ланкевич // Информатика. – 2019. – Т. 16, № 2. – С. 73–89.

Logical optimization of Boolean nets using Shannon expansion

Petr N. Bibilo✉, Yury Y. Lankevich

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

Abstract. A synthesis of logical circuits, comprising functional combination blocks of very large scale integration circuits, is one of the most important tasks of computer-aided design. As the data size of design tasks increases, the execution time of synthesis of logic circuits also increases. The global technological independent

optimization as the first stage of synthesis of logical circuit is especially labor-consuming. The second stage is technological mapping of optimized logical representations of functions to the logical elements of technological library. The main features of logical circuit, such as area, performance, power consumption, depend on the efficiency of the first stage – global logical optimization. The evolution of methods of global logical optimization has revealed the efficiency of Shannon expansion in case of optimization of multi-level representations of the systems of fully defined Boolean function. A number of methods and programs were developed using graphical representations of Shannon expansions – BDD representations. Most of the developed methods of optimization of BDD-representations use the initial representations of functions systems in the form of disjunctive normal form (DNF).

In the article an algorithm of minimization of nodes number of Boolean net, which is a multi-level representation of the system of fully defined Boolean function, is proposed. Minimization is based on Shannon expansion and a search of equal (with accuracy up to inversion) nodes in Boolean net. Such algorithm of logical optimization was implemented as application. The experiments have shown that this algorithm and the application is reasonable to use in cases when the initial multi-level representation of functions is impossible to define as DNF system, or when DNF system contains a large number of elementary conjunctions.

Keywords: Boolean function, Boolean net, Shannon expansion, disjunctive normal form, synthesis of logical circuits, BDD

Acknowledgements. The study was carried out with the financial support of the BRFFR in the framework of project No. F19-023.

For citation. Bibilo P. N., Lankevich Y. Y. Logical optimization of Boolean nets using Shannon expansion. *Informatics*, 2019, vol. 16, no. 2, pp. 73–89 (in Russian).

Введение. Синтез логических схем, реализующих функциональные комбинационные блоки цифровых заказных сверхбольших интегральных схем (СБИС), по-прежнему остается одной из важных задач автоматизированного проектирования, так как возрастает размерность задач проектирования и, соответственно, растет время выполнения этапов синтеза. Особенно трудоемкой является глобальная технологически независимая оптимизация, являющаяся первым этапом синтеза схемы [1, 2]. Суть второго этапа синтеза – технологического отображения (technology mapping) – заключается в «покрытии» оптимизированных логических представлений функций библиотечными логическими элементами. Основные характеристики логической схемы, такие как площадь (часто выражаемая в числе транзисторов), быстродействие и энергопотребление, зависят во многом от эффективности выполнения первого этапа – глобальной логической оптимизации. На втором этапе при покрытии оптимизированных логических уравнений функциональными описаниями библиотечных логических элементов выполняется локальная оптимизация с учетом особенностей логических элементов соответствующей библиотеки. Элементы библиотеки могут реализовать симметричные либо несимметричные булевы функции, покрываемые фрагменты могут иметь несущественные (фиктивные) переменные и т. д.

Как указано в фундаментальном обзоре [3], в первых системах автоматизированного проектирования основными методами технологически независимой оптимизации были методы минимизации (совместной или раздельной с учетом инверсирования) в классе ДНФ [4], после чего осуществлялась алгебраическая факторизация [5, 6].

Алгебраическая факторизация – выделение общих частей алгебраических представлений функций – осуществлялась на уровне представлений функций в виде булевых сетей. Булевы сети (графы) используют «мелкозернистые» функциональные описания вершин (узлов), узлы при покрытии объединяются в кластеры, а каждый кластер реализуется при этом библиотечным логическим элементом. «Мелкие» логические выражения удобны для формирования кластеров, поэтому они и нашли применение в программах синтеза [7, 8].

Развитие методов глобальной логической оптимизации показало эффективность разложения Шеннона при оптимизации многоуровневых представлений. Было разработано много методов [9–12] и программ, использующих графические представления разложений Шеннона булевых функций – так называемые BDD-представления (Binary Decision Diagram – диаграмма двоичного выбора). В русскоязычной литературе BDD называют также диаграммами двоичных решений, бинарными диаграммами решений, двоичными решающими диаграммами и т. д. Наибольшее развитие получили методы и программы оптимизации с помощью BDD для исходных систем функций, заданных в виде ДНФ и представленных парой матриц (троичная мат-

рица задает элементарные конъюнкции, булева матрица – вхождения элементарных конъюнкций в ДНФ функций системы). Однако получение систем ДНФ в современных синтезаторах не предусматривается. Получение исходных для логической оптимизации представлений в современных синтезаторах логических схем осуществляется после выполнения высокоуровневого синтеза – локальной замены алгоритмических конструкций языков VHDL и Verilog логическими выражениями (формулами), задающими многоуровневые представления систем булевых функций. Например, получение булевых сетей в синтезаторе LeonardoSpectrum [13] осуществляется командой unmap, в результате выполнения которой синтезированная комбинационная схема представляется, по сути, в виде булевой сети. Получаемые многоуровневые представления близки к булевым сетям: кроме операций инверсии (отрицания) и логических двухвходовых операций дизъюнкции и конъюнкции в таких формулах имеется еще операция «исключающее ИЛИ», часто называемая «суммой по модулю два». Такое представление функций может быть оптимизировано с помощью методов глобальной оптимизации, что позволяет достаточно часто при повторном синтезе улучшать результаты начального синтеза логической схемы [12].

В настоящей работе предлагается алгоритм минимизации числа вершин в булевой сети, являющейся многоуровневым представлением системы полностью определенных булевых функций, которая получается после этапа высокоуровневого синтеза либо при повторном синтезе схемы. Минимизация осуществляется на основе разложения Шеннона и поиска вершин сети, реализующих одинаковые булевы выражения (функции), а также поиска вершин, реализующих взаимно инверсные булевы выражения. Такой алгоритм глобальной технологически независимой логической оптимизации программно реализован и экспериментально исследован. Эксперименты показали, что его целесообразно применять в тех случаях, когда исходное многоуровневое представление функций невозможно представить (за приемлемое время (часы) работы компьютерной программы) в виде системы ДНФ либо когда система ДНФ, полученная из многоуровневого представления, содержит многие десятки и сотни тысяч элементарных конъюнкций.

Представления систем булевых функций в виде булевых сетей. Булева сеть – это ориентированный ациклический граф [3]. Вершины, обладающие нулевой полустепенью исхода, помечаются как выходы сети, а вершины, обладающие нулевой полустепенью захода, – как входы. Каждой вершине графа соответствует некоторая булева переменная. Каждую вершину, не являющуюся входом сети, представляет булева функция. Вершина i называется входной вершиной для вершины j, если в сети есть дуга, ведущая из i в j. Переменные, соответствующие выходам сети, называются выходными; переменные, относящиеся к входам сети, – входными, а переменные, которые соответствуют остальным вершинам сети, – промежуточными. В статье рассматриваются булевы сети, представляющими функциями вершин которых могут быть двухоперандные логические операции И (*, конъюнкция), ИЛИ (+, $\vee$, дизъюнкция), а также однооперандная операция НЕ (^, инверсия). Чтобы построить булеву сеть по выражению, задающему булеву функцию, необходимо для каждого знака логической операции этого выражения построить вершину сети и поставить ей в соответствие данную операцию и некоторую промежуточную переменную. Далее необходимо добавить входные вершины и правильным образом соединить вершины дугами. На рис. 1 изображена булева сеть, построенная по формуле $f = \psi_0 \vee \psi_1 = \bar{x}_i \varphi_0 \vee x_i \varphi_1$.

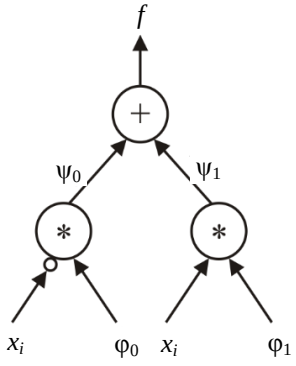

Рис. 1. Булева сеть из трех узлов

Сложность многоуровневого представления системы функций в булевом базисе И, ИЛИ, НЕ будем оценивать суммарным числом двухвходовых логических операторов в соответствующей булевой сети. Инверсии переменных при подсчете сложности представления не принимаются во внимание. Такая оценка сложности хорошо согласуется с известной в литературе [3] оценкой сложности алгебраических представлений булевых функций по общему числу литералов булевых переменных.

Рассмотрим пример булевой сети (рис. 2), имеющей четыре входные переменные *x*0, *x*1, *x*2, *x*3 и две выходные переменные *y*1, *y*2, функции семи узлов которой задаются формулами

$$\begin{gathered} y1 = x0 + \mathrm{tmp\_0}, \\ \mathrm{tmp\_0} = x1 + \mathrm{tmp\_2}, \\ \mathrm{tmp\_2} = \wedge x2 * x3, \\ y2 = x0 + \mathrm{tmp\_3}, \\ \mathrm{tmp\_3} = \mathrm{tmp\_4} + \mathrm{tmp\_5}, \\ \mathrm{tmp\_4} = \wedge x1 * x2, \\ \mathrm{tmp\_5} = \wedge x1 * \wedge x3. \end{gathered} \tag{1}$$

Сложность булевой сети на рис. 2 равна семи: три узла реализуют операцию * конъюнкции, четыре узла – операцию + дизъюнкции.

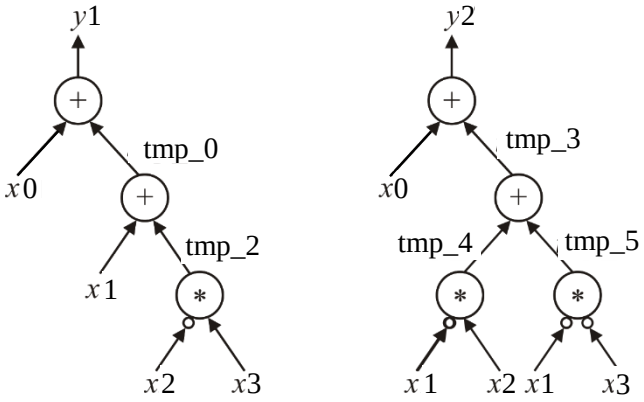

Рис. 2. Булева сеть из семи узлов для формул (1)

Функциональные разложения, используемые при оптимизации многоуровневых представлений систем булевых функций, чаще всего базируются на разложении Шеннона. Формула разложения Шеннона для одной булевой функции $f(x) = f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$ имеет вид

$$f(x) = \bar{x}_i f(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n) \vee x_i f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n). \tag{2}$$

Разложение Давио использует коэффициенты (подфункции) $f(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n)$, $f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n)$, на его базе строятся функциональные диаграммы решений (Functional Decision Diagram, FDD).

Положительное разложение Давио имеет вид

$$\begin{gathered} f(x) = f(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n) \oplus x_i (f(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n) \oplus \\ \oplus f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n)), \end{gathered} \tag{3}$$

отрицательное разложение Давио –

$$\begin{gathered} f(x) = f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n) \oplus \bar{x}_i (f(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n) \oplus \\ \oplus f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n)). \end{gathered} \tag{4}$$

Кронекеровские функциональные диаграммы решений (Kronecker FDD, KFDD) используют как разложение Шеннона, так и разложение Давио, однако на каждом шаге разложения, т. е. при разложении по очередной переменной, используется только один из видов разложений (2)–(4).

Краткий обзор функциональных разложений приведен в работе [14], обобщения для неполностью определенных (частичных) булевых функций предложены в работе [15].

Одним из новых функциональных разложений по двум переменным является разложение вида

$$f(x) = f(x_1, ..., x_i, ..., x_j, ..., x_n) = \\ = (x_i \oplus x_j) f(x_1, ..., \bar{x}_j, ..., x_j, ..., x_n) \vee (x_i \sim x_j) f(x_1, ..., x_j, ..., x_j, ..., x_n), \quad (5)$$

подробно изученное в работе [14] и названное biconditional expansion. В выражении (5) через $\oplus$ обозначена логическая операция «исключающее ИЛИ», через $\sim$ – логическая операция «эквивалентность».

Постановка задачи. Задано многоуровневое представление системы $f(x) = (f^1(\boldsymbol{x}), \ldots, f^m(\boldsymbol{x}))$, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, полностью определенных булевых функций в виде взаимосвязанных логических формул. Каждая из формул имеет вид ДНФ. Требуется минимизировать сложность булевой сети, представляющей систему функций $f(x) = (f^1(\boldsymbol{x}), \ldots, f^m(\boldsymbol{x}))$.

В настоящей работе предлагается проводить минимизацию булевых сетей на основе разложения Шеннона с поиском одинаковых (и взаимно инверсных) подвыражений, реализуемых узлами булевой сети.

Запишем разложение Шеннона (2) в виде $f = \bar{x}_i\varphi_0 \vee x_i\varphi_1 = \psi_0 \vee \psi_1$. Формула (2) представляется логической сетью с тремя узлами (вершинами) (см. рис. 1), выходная вершина $f = \psi_0 \vee \psi_1$ реализует дизъюнкцию, две входные ψ_0, ψ_1 вершины сети – конъюнкции: $\psi_0 = \bar{x}_i\varphi_0$, $\psi_1 = x_i\varphi_1$, т. е. каждая формула разложения Шеннона заменяется своей «малой» булевой сетью, состоящей из трех вершин. В классических алгоритмах минимизации многоуровневых представлений на основе разложения Шеннона (BDD-оптимизации) находятся одинаковые коэффициенты (подфункции) φ_0, φ_1 в разложениях различных функций системы. В работе [16] введено понятие BDDI и находятся одинаковые и взаимно инверсные коэффициенты. Под BDDI (Binary Decision Diagram with Inverse cofactors – диаграмма двоичных решений с инверсными коэффициентами) понимается ориентированный ациклический граф, задающий последовательные разложения Шеннона булевой функции $f(x_1, \ldots, x_n)$ по всем ее переменным $x_1, \ldots, x_n$ при заданном порядке (последовательности, перестановке) переменных, по которым проводятся разложения. Граф BDDI одной полностью определенной булевой функции содержит функциональные вершины, соответствующие разлагаемым функциям и подфункциям (и их инверсиям), вершины-переменные и листовые вершины, соответствующие константам 0, 1. Функциональная вершина BDDI реализует одну функцию φ (подфункцию) либо две функции (подфункцию φ и ее инверсию $\bar{\varphi}$). Эксперименты [16] показали, что нахождение инверсных коэффициентов позволяет получать более компактные формулы и менее сложные схемы при последующем синтезе схем из библиотечных КМОП-элементов, образующих библиотеку проектирования отечественных заказных СБИС.

В отличие от минимизации BDD- и BDDI-представлений, основанных на поиске одинаковых (и взаимно инверсных) коэффициентов φ_0, φ_1, в булевых сетях после выполнения очередного разложения Шеннона находятся одинаковые и взаимно инверсные булевы выражения (функции узлов булевой сети). Такие выражения определяются на основе законов булевой алгебры [17], примеры взаимно инверсных булевых выражений приведены в табл. 1.

Таблица 1
Взаимно инверсные булевы выражения – функции узлов булевой сети

Выражение (функция узла)	Инверсное выражение
$\psi_0 + \psi_1$	$\overline{\psi}_0 * \overline{\psi}_1$
$\overline{x}_i * \varphi_0$	$x_i + \overline{\varphi}_0$
$x_i * \varphi_1$	$\overline{x}_i + \overline{\varphi}_1$

Алгоритм минимизации булевой сети. Алгоритм включает три этапа:

Этап 1. Строится булева сеть по исходному заданию системы булевых функций, зависящих от n переменных. На данном этапе инверсные подвыражения не находятся. Замена многооперандных операций дизъюнкции и конъюнкции осуществляется каскадными формулами (бинарными деревьями) из соответствующих двухвходовых операций. Например, формула многооперандной дизъюнкции $D = k1 + k2 + k3 + k4 + k5 + k6 + k7$ заменяется совокупностью формул:

$$D = t1 + t2 \text{ (уровень 3)};$$
$$t1 = k1 + t3,\ t2 = t4 + t5 \text{ (уровень 2)};$$
$$t3 = k2 + k3,\ t4 = k4 + k5,\ t5 = k6 + k7 \text{ (уровень 1)}.$$

Аналогично можно записать формулы булевой подсети для многооперандной конъюнкции.

Этап 2. Находится первая перестановка входных переменных булевой сети, по которой строится первое оптимизированное представление системы функций в виде булевой сети.

Алгоритм второго этапа является локально-оптимальным и состоит из следующих шагов:

Шаг 1 (итеративный). Выбирается первая переменная, по которой будет проведено разложение Шеннона.

Для каждой из входных переменных x_i множества $X = \{x_1, x_2, \ldots, x_n\}$ булевой сети проводится построение разложения Шеннона по переменной x_i, полученные булевы подсети редуцируются по законам булевой алгебры. Первая остаточная подсеть (остаточные уравнения) получаются при подстановке $x_i = 0$, вторая подсеть – при подстановке $x_i = 1$. Редукция каждой из остаточных подсетей производится с учетом нахождения одинаковых и взаимно инверсных подвыражений, являющихся функциями вершин остаточных подсетей. После этого подсети объединяются – добавляются вершины, соответствующие формулам разложения Шеннона для каждой из выходных переменных, затем булева сеть повторно упрощается за счет нахождения одинаковых и взаимно инверсных узлов. Для полученной булевой сети проводится еще одно дополнительное сокращение путем нахождения вырожденных формул разложения Шеннона. Так, формула разложения Шеннона $f = \overline{x_i}\varphi_0 \vee x_i\varphi_1$, когда $\varphi_1 = 1$, заменяется более простой формулой $f = \varphi_0 \vee x_i$, так как $\overline{x}_i\varphi_0 \vee x_i = \varphi_0 \vee x_i$. Аналогично, если формула разложения Шеннона для какой-то из функций имеет вид $\varphi_0 = 1$, то она заменяется формулой $f = \overline{x}_i \vee \varphi_1$. После этого дополнительного сокращения производится оценка сложности булевой сети, построенной по переменной x_i разложения Шеннона.

После оценки разложений Шеннона по всем переменным множества $X = \{x_1, x_2, \ldots, x_n\}$ выбирается переменная x_{j_1}, по которой редуцированная булева сеть будет иметь наименьшую сложность. Эта переменная будет первой переменной в искомой перестановке $< x_{j_1}, x_{j_2}, \ldots, x_{j_n} >$ переменных для построения многоуровневого представления.

При программной реализации были испытаны три эвристики оценки сложности булевой сети.

Шаг 2 (итеративный). Выбираются остальные $n - 1$ переменные в искомой перестановке переменных для разложения Шеннона.

По выбранной на шаге 1 переменной x_{j_1} строится разложение Шеннона для булевой сети, полученная сеть будет исходной для выбора следующей переменной x_{j_2} разложения, которая выбирается аналогичным образом из множества оставшихся переменных, т. е. тех переменных, по которым еще не проведено разложение Шеннона функций, реализуемых булевой сетью.

Этап 3 (итеративный). Перебираются перестановки $< x_{j_1}, x_{j_2}, ..., x_{j_n} >$ для уменьшения сложности булевой сети.

Перебор является итеративным и осуществляется с помощью алгоритма второго этапа. Исходным представлением будет булева сеть, полученная в результате выполнения второго этапа. После нахождения очередной перестановки она запоминается, как и сложность соответствующей булевой сети. Итерации (перебор перестановок) прекращаются, если сгенерирована такая перестановка, которая была уже рассмотрена ранее, либо время вычислений превысило заданное.

Пример работы алгоритма на сети, заданной уравнениями (1).

Этап 1. Каждое из уравнений может быть представлено одной вершиной булевой сети (см. рис. 2).

Этап 2. *Шаг 1.* Нахождение первой перестановки переменных, по которым строятся разложения Шеннона.

Итерация 1. Оценка сложности остаточной сети, полученной при разложении Шеннона по переменной *x*0.

Для большей ясности построения разложений Шеннона перепишем функции выходных переменных в виде

$$y1 = x0 + tmp\_0,$$
$$y2 = x0 + tmp\_3$$

и получим

$$y1 = \hat{}x0 * tn\_0 + x0 * tn\_2,$$
$$y2 = \hat{}x0 * tn\_1 + x0 * tn\_3.$$

Подставим в формулы (1) значение *x*0 = 0, при этом для каждой вершины сети (см. рис. 2) сформируем инверсное выражение. Получим остаточные уравнения, приведенные в табл. 2.

Сравним остаточные уравнения (в том числе инверсные) на равенство. Одинаковых уравнений нет.

Таблица 2

Результаты подстановки *x*0 = 0 в формулы (1)

Остаточные уравнения	Инверсии остаточных уравнений
tn_0 = tmp_0	^tn_0 = ^tmp_0
tmp_0 = *x*1 + tmp_2	^tmp_0 = ^*x*1 * ^tmp_2
tmp_2 = ^*x*2 * *x*3	^tmp_2 = *x*2 + ^*x*3
tn_1 = tmp_3	^tn_1 = ^tmp_3
tmp_3 = tmp_4 * tmp_5	^tmp_3 = ^tmp_4 + ^tmp_5
tmp_4 = ^*x*1 * *x*2	^tmp_4 = *x*1 + ^*x*2
tmp_5 = ^*x*1 * ^*x*3	^tmp_5 = *x*1 + *x*3

Подставим в формулы (1) значение *x*0 = 1, при этом для каждой неконстантной вершины сети сформируем инверсное выражение.

Остаточными уравнениями при *x*0 = 1 будут уравнения tn_2 = 1 и tn_3 = 1. Сравним остаточные уравнения (в том числе инверсные) на равенство. Одинаковых уравнений нет.

Объединим подсети, полученные на шагах 2, 4, и получим остаточную булеву сеть для оценки по переменной *x*0:

$$\begin{gathered}
\text{tmp_0} = x1 + \text{tmp_2}, \\
\text{tmp_2} = \hat{}x2 * x3, \\
\text{tmp_3} = \text{tmp_4}*\text{tmp_5}, \\
\text{tmp_4} = \hat{}x1 * x2, \\
\text{tmp_5} = \hat{}x1 * \hat{}x, \\
y1 = \hat{}x0 * \text{tmp_0} + x0, \\
y2 = \hat{}x0 * \text{tmp_3} + x0.
\end{gathered} \tag{6}$$

При записи уравнений в $y1 = \hat{}x0 * \text{tn\_0} + x0 * \text{tn\_2}$ переменную tn_0 заменили на tmp_0, tn_2 = 1, в уравнении $y2 = \hat{}x0 * \text{tn\_1} + x0 * \text{tn\_3}$ переменную tn_1 заменили на tmp_3, tn_3 = 1.

Сложность остаточной сети, полученной разложением Шеннона по переменной $x0$, равна 5.

Итерация 2. Оценка сложности остаточной сети, полученной при разложении Шеннона по переменной $x1$.

Подставим в формулы (1) значение $x1 = 0$, при этом для каждой вершины сети (см. рис. 2) сформируем инверсное выражение (табл. 3).

Таблица 3

Результаты подстановки $x1 = 0$ в формулы (1)

Остаточные уравнения	Инверсии остаточных уравнений
$\text{tn\_6} = x0 + \text{tmp\_2}$	$\hat{}\text{tn\_6} = \hat{}x0 * \hat{}\text{tmp\_2}$
$\text{tmp\_2} = \hat{}x2 * x3$	$\hat{}\text{tmp\_2} = x2 + \hat{}x3$
$\text{tn\_7} = x0 + \text{tn\_5}$	$\hat{}\text{tn\_7} = \hat{}x0 * \hat{}\text{tn\_5}$
$\text{tn\_5} = x2 + \hat{}x3$	$\hat{}\text{tn\_5} = \hat{}x2 * x3$

Сравним остаточные уравнения (в том числе инверсные) на равенство. Найдено одно совпадение:

$$\text{tn\_5} = \hat{}\text{tmp\_2}.$$

Заменим tn_5 на ^tmp_2 и получим следующие уравнения:

$$\begin{gathered}
\text{tn_6} = x0 + \text{tmp_2}\ (\hat{}\text{tn_6} = \hat{}x0 * \hat{}\text{tmp_2}), \\
\text{tn_7} = x0 + \hat{}\text{tmp_2}\ (\hat{}\text{tn_7} = \hat{}x0 * \text{tmp_2}), \\
\text{tmp_2} = \hat{}x2 * x3\ (\hat{}\text{tmp_2} = x2 + \hat{}x3).
\end{gathered}$$

Подставив в формулы (6) значение $x1 = 1$, получим $\text{tn\_8} = 1$ и $\text{tn\_9} = x0$. Сравним остаточные уравнения (в том числе инверсные) на равенство. Одинаковых уравнений нет. Объединим подсети, тогда остаточная булева сеть для оценки по переменной $x1$ имеет вид

$$\begin{gathered}
\text{tn_6} = x0 + \text{tmp_2}, \\
\text{tn_7} = x0 + \hat{}\text{tmp_2}, \\
\text{tmp_2} = \hat{}x2 * x3, \\
\text{tn_9} = x0.
\end{gathered} \tag{7}$$

В результате получим выражения

$$\begin{gathered}
y1 = \text{tn_6} + x1, \\
y2 = \hat{}x1 * \text{tn_7} + x1 * x0.
\end{gathered}$$

В уравнении $y1 = \hat{}x1 * \text{tn\_6} + x1 * \text{tn\_8}$ переменная $\text{tn\_8} = 1$, в уравнении $y1 = \hat{}x1 * \text{tn\_7} + x1 * \text{tn\_9}$ промежуточная переменная $\text{tn\_9} = x0$. Остаточная сеть по переменной $x1$ имеет сложность 3. Итерация 3 (для переменной $x2$) и итерация 4 (для переменной $x3$) первого этапа выполняются аналогично.

Шаг 2 (итеративный). Поиск первой перестановки переменных разложения Шеннона.

Остаточная сеть по переменной разложения $x1$ имеет наименьшую сложность 3, поэтому переменная $x1$ является первой переменной в искомой перестановке $< x_{j_1}, x_{j_2}, ..., x_{j_n} >$, а формулы (7) будут исходными для выбора следующей переменной для проведения следующего разложения Шеннона. Оценив сложности остаточных уравнений для переменных $x0$, $x2$, $x3$, получим, что следующей переменной разложения является переменная $x0$ (один узел в остаточной сети). Для оставшихся двух переменных $x2$, $x3$ остаточная сеть имеет вид tmp_2 = ^$x2$ * $x3$ со сложностью 1.

Результатом выполнения шага 2 (и этапа 2) является перестановка <$x1$, $x0$, $x2$, $x3$> переменных, по которым проведены разложения Шеннона. Полученной перестановке соответствует совокупность формул

$$\begin{gathered} y1 = \text{tn\_6} + x1, \\ y2 = \text{\^{}}x1 * \text{tn\_7} + x1 * x0, \\ \text{tn\_6} = \text{tmp\_2} + x0, \\ \text{tn\_7} = \text{\^{}tmp\_2} + x0, \\ \text{tmp\_2} = \text{\^{}}x2 * x3. \end{gathered} \tag{8}$$

Сложность соответствующей булевой сети равна 7.

Этап 3. Перебор перестановок для уменьшения сложности булевой сети.

Исходной является булева сеть, реализующая формулы (8). Применив к ней шаги этапа 2 алгоритма, получим новую перестановку <$x0$, $x1$, $x2$, $x3$> и соответствующую булеву сеть, реализующую формулы (6):

$$\begin{gathered} y1 = \text{\^{}tn\_57} + x0, \\ y2 = \text{tn\_57} + x0, \\ \text{tn\_57} = \text{\^{}tn\_88} * \text{\^{}}x1, \\ \text{tn\_88} = \text{\^{}}x2 * x3. \end{gathered} \tag{9}$$

Применив к формулам (9) шаги этапа 2 алгоритма, получим перестановку <$x0$, $x1$, $x2$, $x3$>, которая была найдена ранее. Поэтому формулы (9) являются итогом работы алгоритма минимизации для исходной сети, заданной уравнениями (1). В результате получено многоуровневое представление (рис. 3) с двумя конъюнкциями и двумя дизъюнкциями (сложность 4), при этом изначально сеть состояла из трех конъюнкций и четырех дизъюнкций (сложность 7).

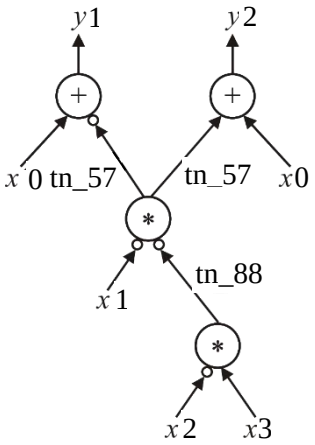

Рис. 3. Булева сеть из четырех узлов для формул (9)

Заметим, что представление булевых функций *y*1, *y*2 в виде BDD

$$\begin{gathered} y1 = \hat{}x0 * sf1 + x0, \\ y2 = \hat{}x0 * sf3 + x0, \\ sf1 = \hat{}x1 * sf5 + x1, \\ sf3 = \hat{}x1 * sf6, \\ sf5 = \hat{}x2 * x3, \\ sf6 = \hat{}x2 * \hat{}x3 + x2 \end{gathered} \tag{10}$$

имеет сложность 10, а представление в виде BDDI

$$\begin{gathered} y2 = \hat{}x0 * \hat{}sf0 + x0, \\ y1 = \hat{}x0 * sf0 + x0, \\ sf0 = \hat{}x1 * sf2 + x1, \\ sf2 = \hat{}x2 * x3 \end{gathered} \tag{11}$$

имеет сложность 7.

Экспериментальные исследования. Предложенный в настоящей работе алгоритм был программно реализован. В разработанной программе BDD_Builder2 исследованы три эвристики:

1. Для проведения разложения Шеннона выбирается переменная x_i, для которой сложность остаточной сети является наименьшей. Сложность подсети, задающей формулы разложений Шеннона, не учитывается.

2. Для проведения разложения Шеннона выбирается переменная x_i, для которой суммарная сложность остаточной сети и подсети, реализующей формулы разложения Шеннона, является наименьшей.

3. Для проведения разложения Шеннона выбирается переменная x_i, для которой суммарная сложность остаточной сети и подсети, реализующей формулы разложения Шеннона, является наименьшей, при этом учитывается только одна десятая сложности подсети для формул разложения Шеннона. Данная эвристика выражает компромисс между эвристиками 1 и 2, при котором сложность остаточной сети имеет бóльший приоритет.

Цель экспериментов состояла в том, чтобы сравнить результаты синтеза по неоптимизированным и оптимизированным функциональным описаниям многовыходных комбинационных схем. Исходные описания представлялись на языке SF в системе FLC логической оптимизации [18], затем выполнялась программа BDD_Builder2. Полученные оптимизированные функциональные описания конвертировались в VHDL-описания и подавались на вход синтезатора LeonardoSpectrum [13], который выполнял построение логических схем. Все схемы строились в одной и той же библиотеке *s3lib* КМОП-элементов (табл. 4). Сравнивались результаты синтеза по площади схем и их быстродействию. Каждое выполнение программы BDD_Builder2 сопровождалось формальной верификацией исходного и полученного оптимизированного описаний.

Первый поток из девяти примеров для проведения экспериментов (add6, b2, Intb, root, sist_4, tial, tms, z5xp1, z9sym) включал функциональные описания в виде взаимосвязанных логических уравнений, которые являются формулами, полученными командой unmap в синтезаторе LeonardoSpectrum [13] после выполнения синтеза. Особенностью таких формул является то, что каждая из них содержит только одну логическую операцию над парой булевых переменных либо операцию инверсирования одной булевой переменной. Заметим, что исходный синтез для данных примеров был проведен в расширенной библиотеке, приведенной в работе [18], а повторный синтез осуществлялся в библиотеке *s3lib* элементов (табл. 4). Исходные описания примеров систем ДНФ находятся в библиотеке примеров схем [19].

Таблица 4

Библиотека *s3lib* логических КМОП-элементов проектирования заказных СБИС

Элемент	Функция	Число транзисторов
GND	$y = 0$	1
VCC	$y = 1$	1
N	$y = \overline{A}$, инвертор	2
NX2	$y = \overline{A}$, двухкратный инвертор	4
NX4	$y = \overline{A}$, четырехкратный инвертор	8
NA	$y = \overline{AB}$	4
NO	$y = \overline{A \vee B}$	4
NAO	$y = \overline{(A \vee B)C}$	6
NOA	$y = \overline{(AB) \vee C}$	6
NA3O	$y = \overline{(A \vee B)CD}$	8
NO3A	$y = \overline{(AB) \vee C \vee D}$	8
NA3	$y = \overline{ABC}$	6
NO3	$y = \overline{A \vee B \vee C}$	6

Второй поток – это восемь широко известных примеров (Apex6, C8, Cht, Count, Dalu, Frg2, Too_large, X3) многоуровневых описаний комбинационной логики. Для этих описаний осуществлялся синтез схем без выполнения программ предварительной оптимизации и с помощью предварительной оптимизации булевых сетей, предложенной в настоящей работе.

Эксперимент 1. Сравнивались эффективности применения трех эвристик оптимизации булевых сетей без использования упрощений

$$f = \overline{x}_i\varphi_0 \vee x_i = \varphi_0 \vee x_i, \quad f = \overline{x}_i \vee x_i\varphi_1 = \overline{x}_i \vee \varphi_1. \tag{12}$$

Эксперимент 2. Сравнивались эффективности применения трех эвристик оптимизации булевых сетей с использованием упрощений (12).

Эксперимент 3. Сравнивались предложенный алгоритм и его программная реализация (три эвристики) с известными в литературе алгоритмами (программами) оптимизации многоуровневых представлений, которые предназначаются для использования в качестве предварительного этапа при синтезе логических схем. Целью таких алгоритмов оптимизации является сокращение сложности функциональных описаний, что благоприятно сказывается на дальнейшей минимизации площади схем при последующем их синтезе.

Для сравнения были выбраны:

– алгоритм, основанный на оптимизации BDDI [16], использующий три эвристики и применимый для исходного матричного задания систем булевых функций в виде ДНФ;

– программа минимизации числа вершин графа BBDD (Biconditional Binary Decision Diagrams – диаграмма двоичного выбора с двумя условиями), задающего последовательные разложения (5) (URL: http://lsi.epfl.ch/BBDD).

Для программы минимизации BBDD исходными данными являются булевы сети, функциями вершин которых могут быть двухоперандные логические операции: конъюнкция, дизъюнкция, исключающее ИЛИ, а также однооперандная операция – инверсия.

В качестве библиотеки синтеза в эксперименте 3 использовалась та же библиотека (power) КМОП-элементов, что и в работе [16].

Исходные матричные описания систем ДНФ булевых функций, используемые в работе [16] и взятые из набора [19], были переведены в булевы сети, задаваемые в виде структурных описаний на языке Verilog [20]. Программа минимизации BBDD, реализующая алгоритмы из работы [14], минимизировала число вершин в графе, задающем BBDD-представление системы функций. Минимизированные BBDD-представления задавались в виде функциональных опи-

саний на языке Verilog, которые были исходными для синтеза логических схем в библиотеке power с помощью синтезатора LeonardoSpectrum.

Для примера в левой части табл. 5 приведены логические уравнения (1) на языке Verilog, являющиеся исходными данными для программы минимизации BBDD, а в правой части – функциональное описание графа BBDD (рис. 4), полученное по описанию из левой части табл. 5 с помощью этой программы.

Таблица 5

Verilog-описания: исходные данные и результат работы программы минимизации BBDD для булевой сети, заданной уравнениями (1)

Исходные данные (структурное описание булевой сети)	Результат (функциональное описание BBDD)
module example1 (x0, x1, x2, x3, y1, y2); *input x0, x1, x2, x3;* *output y1, y2;* *wire tmp0, tmp2, tmp3, tmp4, tmp5, tmp6, tmp7, tmp8;* *orx ix3 (.a (x0), .b (tmp0), .O (y1));* *orx ix6 (.a (x0), .b (tmp3), .O (y2));* *andx ix5 (.a (tmp7), .b (x3), .O (tmp2));* *invx ix2 (.a (x2), .O (tmp7));* *andx ix8 (.a (tmp6), .b (x2), .O (tmp4));* *andx ix9 (.a (tmp6), .b (tmp8), .O (tmp5));* *invx ix10 (.a (x3), .O (tmp8));* *invx ix1 (.a (x1), .O (tmp6));* *orx ix4 (.a (x1), .b (tmp2), .O (tmp0));* *orx ix7 (.a (tmp4), .b (tmp5), .O (tmp3));* *endmodule*	*module example1 (x0, x1, x2, x3, y1, y2);* *input x0, x1, x2, x3;* *output y1, y2;* *wire one, node6, node3, node1, node5, node4, node2;* *assign node1 = (x1 ^ x0) ? one : x0;* *assign node3 = (x2 ^ x1) ? node1 : one;* *assign node6 = (x3 ^ x2) ? node3 : node1;* *assign node2 = (x1 ^ x0) ? x0 : one;* *assign node4 = (x2 ^ x1) ? node2 : x0;* *assign node5 = (x3 ^ x2) ? node4 : node2;* *assign one = 1;* *assign y1 = node6;* *assign y2 = node5;* *endmodule*

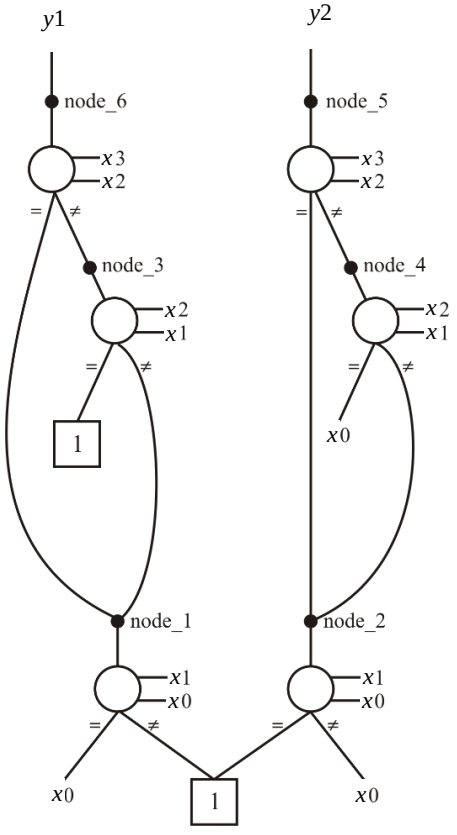

Рис. 4. Граф BBDD для примера example1

В структурном описании используются следующие логические элементы: *orx* – логическое ИЛИ (дизъюнктор), *andx* – логическое И (конъюнктор), *invx* – логическое отрицание (инвертор); в описаниях данных элементов: *a, b* – имена входов, *O* – имя выхода. В функциональном описании BBDD используется оператор *assign* присвоения значения сигналу, в правой части находятся логические операторы ^ ($\oplus$ – исключающее ИЛИ), а также тернарные (условные) операторы «?». Например, оператор

assign node6 = (x3 ^ x2) ? node3 : node1;

понимается следующим образом: если выражение $x3 \oplus x2$ истинно, то промежуточный сигнал *node6* получает значение сигнала *node3*, если же выражение $x3 \oplus x2$ ложно, то сигнал *node6* получает значение сигнала *node1*. Более подробно об операторах языка Verilog можно прочесть в работе [20], там же дано соответствие операторов языков VHDL и Verilog. Функциональное описание графа BBDD на языке VHDL для примера, приведенного в табл. 5, дано ниже:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity example1 is
  port (
    x0, x1, x2, x3 : IN std_logic ;
    y1, y2 : OUT std_logic) ;
end;
architecture beh of example1 is
signal node1, node2, node3,node4, node5, node6 : std_logic ;
begin
node1 <= (( x1 xor x0) and '1')       or  ((x1 xnor x0) and x0);
node2 <= (( x1 xor x0) and x0)        or  ((x1 xnor x0) and '1');
node3 <= (( x2 xor x1) and  node1) or  ((x2 xnor x1) and '1');
node4 <= (( x2 xor x1) and  node2) or  ((x2 xnor x1) and x0);
node5 <= (( x3 xor x2) and node4)  or  ((x3 xnor x2) and node2);
node6 <= (( x3 xor x2) and node3)  or  ((x3 xnor x2) and node1);
y1  <= node6;
y2  <= node5;
end;
```

Каждая функциональная вершина графа BBDD на рис. 4, т. е. вершина, в которую слева заходят две булевы переменные, соответствует разложению (5). Проиллюстрируем построение графа на примере функции узла node3:

$$z = {\wedge}x0 * x1 * x2 + x0 * x1 * x2 + {\wedge}x0 * {\wedge}x1 * {\wedge}x2 + x0 * {\wedge}x1 * {\wedge}x2 + \\ + {\wedge}x0 * x1 * {\wedge}x2 + x0 * {\wedge}x1 * x2 + x0 * x1 * {\wedge}x2. \qquad (13)$$

Построим разложение вида (5) функции z, полагая $x_i = x2$, $x_j = x1$:

$$z = (x2 \oplus x1) * z(x0, {\wedge}x1, x1) + (x2 \sim x1) * z(x0, x1, x). \qquad (14)$$

Для вычисления подфункции $z(x0, {\wedge}x1, x1)$ заменим в формуле (13) переменную $x2$ на $^x1$, получим функцию node1 = $^x0 * x1 + x0 * ^x1 + x0 * x1$. Для вычисления $z(x0, x1, x1)$ в (14) заменим в (13) переменную $x2$ на $x1$, получим $z(x0, x1, x1) = {\wedge}x0 * x1 + x0 * x1 + {\wedge}x0 * {\wedge}x1 + x0 * {\wedge}x1 = 1$. Аналогично поступим и для остальных функциональных вершин графа BBDD.

Булевы сети, являющиеся исходными для программы BDD_Builder [16], после замены операции $\oplus$ исключающего ИЛИ формулой ($x_i \oplus x_j = \bar{x}_i x_j \vee x_i \bar{x}_j$) переводились в VHDL-описания и подавались на вход синтезатора LeonardoSpectrum.

Результаты экспериментов 1 и 2 представлены в табл. 6, где n – число переменных, m – число функций, S_{ASIC} – площадь логической схемы (суммарное число транзисторов), τ – задержка схемы (нс). Лучшие решения выделены жирным шрифтом.

Таблица 6

Результаты экспериментов 1 и 2, синтез схем в библиотеке *s3lib*

Вид исходного задания	Схема	n	m	Синтез по исходным описаниям		Синтез по оптимизированным описаниям, программа BDD Builder2					
						Эвристика 1		Эвристика 2		Эвристика 3	
				S_{ASIC}	τ	S_{ASIC}	τ	S_{ASIC}	τ	S_{ASIC}	τ
Результаты эксперимента 1											
Unmap-описания	add6	12	7	2812	6,61	208	3,51	**206**	**3,01**	208	3,42
	b2	16	7	**2210**	6,03	2860	**4,43**	3888	5,16	2834	4,48
	intb	15	7	6848	8,01	**4982**	5,40	6008	5,93	5416	**4,44**
	root	8	5	814	3,50	**496**	**2,59**	**496**	**2,59**	**496**	**2,59**
	sist_4	17	12	**828**	3,91	2030	**3,91**	1942	4,13	2170	4,42
	tial	14	8	5184	6,93	5144	**3,82**	5564	5,10	**4904**	4,29
	tms	8	16	**636**	5,09	684	2,74	678	2,80	714	**2,46**
	z5xp1	7	10	1380	5,04	502	2,37	**460**	**1,75**	516	2,11
	z9sym	9	1	426	4,81	**226**	2,55	**226**	2,55	**226**	2,55
Исходные описания	FRG2	143	139	4224		9458	6,52			13144	7,39
	APEX6	135	94	**1832**	3,34	1992	2,95	2222	3,07	1900	**2,54**
	C8	28	18	310	1,30	312	**1,30**	322	1,40	**308**	**1,30**
	CHT	47	36	680	1,51	**658**	1,34	696	1,32	662	**1,02**
	COUNT	35	16	**256**	4,07	**256**	**4,07**	**256**	**4,07**	**256**	**4,07**
	DALU	75	16	1834	5,24	**932**	**2,61**			1038	3,07
Результаты эксперимента 2											
Unmap-описания	add6	12	7	2812	6,61	**204**	**2,96**	206	**2,96**	**204**	**2,96**
	b2	16	7	2210	6,03	2762	4,42	3842	5,20	3116	4,77
	intb	15	7	6848	8,01	6658	5,84	6270	5,70	5230	5,08
	root	8	5	814	3,50	**416**	**2,31**	**416**	**2,31**	**416**	**2,31**
	sist_4	17	12	828	3,91	2118	4,16	1700	3,87	1818	3,54
	tial	14	8	5184	6,93	5664	4,68	5002	5,11	4984	3,96
	tms	8	16	636	5,09	–	–	688	**2,43**	700	2,66
	z5xp1	7	10	1380	5,04	516	2,10	550	1,89	530	2,10
	z9sym	9	1	426	4,81	**222**	2,60	**222**	2,60	**222**	2,60
Исходные описания	FRG2	143	139	4224	–	9424	5,79	–	–	–	–
	APEX6	135	94	1832	3,34	2256	3,26	2444	3,69	1936	2,97
	C8	28	18	310	1,30	312	**1,09**	326	1,40	318	1,30
	CHT	47	36	680	1,51	660	1,07	696	1,36	662	1,04
	COUNT	35	16	256	4,07	264	4,27	256	4,07	264	4,27
	DALU	75	16	1834	5,24	**872**	2,57	1274	4,05	994	3,44

Анализируя данные табл. 6, можно сделать вывод о том, что дополнительные проверки и упрощения по формулам (12) лишь незначительно улучшают решения (либо могут даже незначительно ухудшить их, см. результаты для intb, tial, C8, CHT). В целом можно отметить, что предложенный способ многоуровневой оптимизации практически всегда позволяет при последующем синтезе уменьшать задержки схем.

Результаты эксперимента 3 представлены в табл. 7, где k – число общих элементарных конъюнкций в матричном задании системы ДНФ булевых функций, S_{ASIC} – суммарная площадь логических элементов схемы в условных единицах площади. Лучшие решения по трем эвристикам выделены жирным шрифтом.

Эксперимент 3 показал, что в одной трети случаев программа BDD_Builder2 позволяет получать лучшие решения по площади в сравнении с программой BDD_Builder, исходными данными для которой являются системы ДНФ, заданные в матричном виде. Однако переход к таким формам задания не всегда возможен для систем булевых функций, зависящих от 35 и более переменных. Разработанная программа BDD_Builder2 ориентируется на более общий способ задания минимизируемых систем булевых функций – булевы сети, и в этом заключается ее преимущество.

Кроме того, программа минимизации BBDD оказалась неконкурентоспособной для применения в качестве оптимизационной процедуры при синтезе схем из КМОП-элементов.

Таблица 7

Результаты эксперимента 3, синтез схем в библиотеке power

Схема	*n*	*m*	*k*	Минимизация BDDI, программа BDD_Builder [16], S_{ASIC}	Минимизация булевых сетей, программа BDD_Builder2, S_{ASIC}	Минимизация BBDD, программа Amaru, S_{ASIC}
add6	12	7	1092	**12 806**	15 719	41 967
b12	15	9	431	**16 137**	16 221	41 152
b2	16	17	110	164 526	**161 457**	349 777
b9	16	5	123	26 081	**22 019**	112 169
br1	12	8	34	**23 843**	25 043	42 425
br2	12	8	35	19 653	**19 234**	36 170
dist	8	5	256	**60 085**	66 887	77 752
in0	15	11	138	**91 116**	91 367	177 729
in2	19	10	137	**69 811**	84 185	170 106
intb	15	7	664	246 764	**229 717**	397 514
life	9	1	512	14 391	**14 346**	21 159
log8mod	8	5	47	23 687	**22 817**	31 192
m181	15	9	430	16 439	**15 808**	40 795
mlp4	8	8	256	**68 439**	71 441	86 127
newtpla	15	5	23	**11 316**	12 577	28 118
newtpla1	10	2	4	**3421**	3962	9977
newtpla2	10	4	9	**6640**	**6640**	140 90
p82	5	14	24	**19 988**	20 032	25 713
radd	8	5	120	**8074**	8119	11 539
rd53	5	3	32	7321	**6830**	8928
rd73	7	3	147	**15 925**	16 656	16 383
root	8	5	256	26 075	**23 732**	39 590
sex	9	14	23	**11 891**	12 354	31 672
tial	14	8	640	**261 278**	285 250	482 341

Заключение. Разработанная программа BDD_Builder2 глобальной оптимизации булевых сетей на основе разложения Шеннона позволяет обрабатывать примеры систем с десятками и сотнями переменных и функций, а также уменьшать площадь схем при повторном синтезе (на потоке unmap-описаний) в шести случаях из девяти. При этом для примеров add6, z5xp1, DALU были получены значительные выигрыши по площади. Предварительная глобальная оптимизация многоуровневой логики также часто целесообразна, однако в некоторых примерах (например, FRG2) реализация исходного многоуровневого описания приводит к схеме меньшей площади. Применение разработанной программы BDD_Builder2 оптимизации булевых сетей позволяет в двух третях случаев (примеров) улучшать результаты (площадь и быстродействие) логического синтеза в экспериментах 1 и 2. В эксперименте 3 программа BDD_Builder2 в 10 случаях из 24 позволила улучшить результаты синтеза по сравнению с программой, оптимизирующей матричные представления систем ДНФ булевых функций. Это свидетельствует о том, что переход к представлениям функций в виде булевых сетей может быть целесообразен и в тех случаях, когда имеются матричные представления систем ДНФ функций.

Программа BDD_Builder2 успешно прошла промышленную проверку и включена в систему FLC логической оптимизации.

Список использованных источников

1. Advanced Techniques in Logic Synthesis, Optimizations and Applications / ed.: S. P. Khatri, K. Gulati. – Springer, 2010. – 423 p.

2. Advanced Logic Synthesis / ed.: A. I. Reis, R. Drechsler. – Springer, 2017. – 232 p.

3. 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.

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

5. Brayton, K. R. Factoring logic functions / K. R. Brayton // IBM J. of Research & Development. – 1987. – Vol. 31, no. 2. – P. 187–198.

6. Синтез асинхронных автоматов на ЭВМ / под ред. А. Д. Закревского. – Минск : Наука и техника, 1975. – 184 с.

7. MIS: A multiple-level logic optimization systems / K. R. Brayton [et al.] // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – 1987. – Vol. 6, no. 6. – P. 1062–1081.

8. Mailhot, F. Algorithms for technology mapping based on binary decision diagrams and on Boolean operations / F. Mailhot, G. Micheli // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – 1993. – Vol. 12, no. 5. – P. 599–620.

9. Bryant, R. E. Graph-based algorithms for Boolean function manipulation / R. E. Bryant // IEEE Transactions on Computers. – 1986. – Vol. 35, no. 8. – P. 677–691.

10. Bryant, R. E. Ordered binary decision diagrams / R. E. Bryant, C. Meinel // Logic Synthesis and Verification / ed.: S. Hassoun, T. Sasao, R. K. Brayton. – Kluwer Academic Publishers, 2002. – P. 285–307.

11. Yang, S. BDS: a BDD-based logic optimization system / S. Yang, M. Ciesielski // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – 2002. – Vol. 21, no. 7. – P. 866–876.

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

13. Бибило, П. Н. Системы проектирования интегральных схем на основе языка VHDL. StateCAD, ModelSim, LeonardoSpectrum / П. Н. Бибило. – М. : СОЛОН-Пресс, 2005. – 384 с.

14. Amaru, L. G. New Data Structures and Algorithms for Logic Synthesis and Verification / L. G. Amaru. – Springer, 2017. – 156 p.

15. Прихожий, А. А. Частично определенные логические системы и алгоритмы / А. А. Прихожий. – Минск : БНТУ, 2013. – 343 с.

16. Бибило, П. Н. Использование полиномов Жегалкина при минимизации многоуровневых представлений систем булевых функций на основе разложения Шеннона / П. Н. Бибило, Ю. Ю. Ланкевич // Программная инженерия. – 2017. – № 3. – С. 369–384.

17. Закревский, А. Д. Логические основы проектирования дискретных устройств / А. Д. Закревский, Ю. В. Поттосин, Л. Д. Черемисинова. – М. : Физматлит, 2007. – 592 с.

18. 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.

19. Jeong, C. Computer-aided design of digital systems / C. Jeong [Electronic resource] // Department of Computer Scienc. – Mode of access: http://www1.cs.columbia.edu/~cs6861/sis/espresso-examples/ex. – Date of access: 20.03.2018.

20. Поляков, А. К. Языки VHDL и VERILOG в проектировании цифровой аппаратуры / А. К. Поляков. – М. : СОЛОН-Пресс, 2003. – 320 с.

References

1. Khatri S. P., Gulati K. (eds.). *Advanced Techniques in Logic Synthesis, Optimizations and Applications*. Springer, 2010, 423 p.

2. Reis A. I., Drechsler R. (eds.). *Advanced Logic Synthesis*. Springer, 2017, 232 p.

3. Brayton R. K., Hachtel G. D., Sangiovanni-Vincentelli A. L. Sintez mnogourovnevyh kombinacionnyh logicheskih skhem [Multilevel logic synthesis]. Trudy Instituta inzhenerov po jelektronike i radiotehnike [*Proceedings of the Institute of Electronics and Radio Engineering*], 1990, vol. 78, no. 2, pp. 38–83 (in Russian).

4. 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.

5. Brayton K. R. Factoring logic functions. *IBM Journal of Research & Development*, 1987, vol. 31, no. 2, pp. 187–198.

6. Zakrevskij A. D. Sintez asinhronnyh avtomatov na JeVM. *Synthesis of Asynchronous Machines on a Computer*. Minsk, Nauka i tekhnika, 1975, 184 p.

7. Brayton K. R., Rudell R., Sangiovanni-Vincentelli A. L., Wang A. R. MIS: A multiple-level logic optimization systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1987, vol. 6, no. 6, pp. 1062–1081.

8. Mailhot F., Micheli G. Algorithms for technology mapping based on binary decision diagrams and on Boolean operations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1993, vol. 12, no. 5, pp. 599–620.

9. Bryant R. E. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 1986, vol. 35, no. 8, pp. 677–691.

10. Bryant R. E., Meinel C. Ordered binary decision diagrams. *Logic Synthesis and Verification*. In Hassoun S., Sasao T., Brayton R. K. (eds.). Kluwer Academic Publishers, 2002, pp. 285–307.

11. Yang S., Ciesielski M. BDS: a BDD-based logic optimization system. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2002, vol. 21, no. 7, pp. 866–876.

12. Bibilo P. N. Primenenie diagramm dvoichnogo vybora pri sinteze logicheskih shem. *Application of Binary Decision Diagrams at Synthesis of Logical Circuits.* Minsk, Belaruskaja navuka, 2014, 231 p. (in Russian).

13. Bibilo P. N. Cistemy proektirovaniya integral'nyh skhem na osnove yazyka VHDL. StateCAD, ModelSim, LeonardoSpectrum. *Integrated Circuit Design Systems Based on the VHDL Language. StateCAD, ModelSim, LeonardoSpectrum*. Moscow, SOLON-Press, 2005, 384 p. (in Russian).

14. Amaru L. G. *New Data Structures and Algorithms for Logic Synthesis and Verification*. Springer, 2017, 156 p.

15. Prihozhij A. A. Chastichno opredelennye logicheskie sistemy i algoritmy. *Partially Defined the Logical Systems and Algorithms*. Minsk, Belorusskij nacional'nyj tehnicheskij universitet, 2013, 343 p.

16. Bibilo P. N., Lankevich Yu. Yu. Ispol'zovanie polinomov Zhegalkina pri minimizacii mnogourovnevyh predstavlenij sistem bulevyh funkcij na osnove razlozheniya Shennona [The use of Zhegalkin polynomials with minimization of multilevel representations of systems of Boolean functions on the basis of the Shannon decomposition]. Programmnaya inzheneriya [*Software Engineering*], 2017, no. 3, pp. 369–384 (in Russian).

17. Zakrevskij A. D., Pottosin Ju. V., Cheremisinova L. D. Logicheskie osnovy proektirovanija diskretnyh ustrojstv. *Logical Bases of Design of Discrete Devices*. Moscow, Fizmatlit, 2007, 592 p. (in Russian).

18. 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 Russian).

19. Jeong C. Computer-aided design of digital systems. *Department of Computer Science*. Available at : http://www1.cs.columbia.edu/~cs6861/sis/espresso-examples/ex (accessed 20.03.2018).

20. Poljakov A. K. Jazyki VHDL i VERILOG v proektirovanii cifrovoj apparatury. *VHDL and VERILOG in the design of digital equipment*. Moscow, SOLON-Press, 2003, 320 p.

Информация об авторах

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

Ланкевич Юрий Юрьевич, младший научный сотрудник, Объединенный институт проблем информатики Национальной академии наук Беларуси, Минск, Беларусь.
E-mail: yurafreedom18@gmail.com

Information about the authors

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

Yury Y. Lankevich, Junior Researcher, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus, Minsk, Belarus.
E-mail: yurafreedom18@gmail.com