ISSN 1816-0301 (Print)
ISSN 2617-6963 (Online)

УДК 681.325
https://doi.org/10.37661/1816-0301-2020-17-3-54-63

Поступила в редакцию 06.06.2020
Received 06.06.2020

Принята к публикации 13.07.2020
Accepted 13.07.2020

Алгоритмы разбиения логических схем на подсхемы

Н. А. Кириенко

Объединенный институт проблем информатики
Национальной академии наук Беларуси, Минск, Беларусь
E-mail: natalia.kirienko@tut.by

Аннотация. Рассматривается задача разбиения логической схемы на подсхемы, имеющая большое значение при выполнении оптимизационных преобразований в процессе синтеза схемы. Приводится краткий обзор методов и алгоритмов разбиения, выделяются две группы алгоритмов: конструктивные и итеративные. Представляется интерпретация логической схемы в виде графа, формулируется задача разбиения в теоретико-графовой модели и предлагается набор алгоритмов для ее решения. Функционирование логической схемы задается системой логических уравнений. Алгоритмы осуществляют разбиение системы логических уравнений на подсистемы с выполнением ограничений по числу входных и выходных переменных. Рассматриваются структуры данных, необходимых для выполнения алгоритмов. Описываются различные виды взаимосвязей уравнений, определяющих получение оптимальных решений. Исследуются вопросы применения алгоритмов разбиения для улучшения качества схемы на этапе технологически независимой оптимизации. Результаты экспериментального исследования, выполненного с помощью процедуры BDD-оптимизации функционального описания схемы и промышленного синтезатора LeonardoSpectrum подтверждают эффективность разработанных алгоритмов. Алгоритмы реализуются в виде набора процедур разбиения схемы в рамках экспериментальной системы логического проектирования FLC.

Ключевые слова: логическая схема, разбиение логической схемы, системы булевых функций, конструктивные алгоритмы, технологически независимая оптимизация, синтез логических схем

Для цитирования. Кириенко, Н. А. Алгоритмы разбиения логических схем на подсхемы / Н. А. Кириенко // Информатика. – 2020. − Т. 17, № 3. – С. 54–63. https://doi.org/10.37661/1816-0301-2020-17-3-54-63

Algorithms for partitioning logical circuits into subcircuits

Natalia A. Kirienko

The United Institute of Informatics Problems of the National Academy
of Sciences of Belarus, Minsk, Belarus
E-mail: natalia.kirienko@tut.by

Abstract. The problem of partitioning a logical circuit into subcircuits is considered. It is of great importance when performing optimization transformations in the process of circuit synthesis. The brief review of partitioning methods and algorithms is given, and two groups of algorithms are identified: constructive and iterative one. The interpretation of a logical circuit in the form of a graph is presented. The problem of partitioning in terms of a graph-theoretic model is defined and some algorithms for solving the partitioning problem are proposed. Logic circuit functions are defined by a system of logical equations. Algorithms perform the partitioning the system of logical equations into subsystems with the restrictions of the number of input and output variables. The data structures to execute the algorithms are defined. Various types of equations connections, obtaining better solutions for partitioning are described. The problems of the use of partitioning algorithms to improve the quality of the circuit at the stage of technology-independent optimization are investigated. The results of an experimental study carried out by the BDD optimization procedure for the functional description of the circuit and LeonardoSpectrum synthesis confirm the effectiveness of the developed

algorithms. The algorithms are implemented as partitioning circuit procedures in the experimental FLC system for logical design.

Keywords: logical circuit, logical circuit partitioning, systems of Boolean functions, constructive algorithms, technology-independent optimization, synthesis of logical circuits

For citation. Kirienko N. A. Algorithms for partitioning logical circuits into subcircuits. *Informatics*, 2020, vol. 17, no. 3, pp. 54−63 (in Russian). https://doi.org/10.37661/1816-0301-2020-17-3-54-63

Введение. Задача разбиения логической схемы на подсхемы заключается в преобразовании исходного описания схемы в результирующее описание, содержащее множество подсхем (блоков), связанных между собой. На описание подсхемы накладываются ограничения, например, по числу входов и выходов, по размерам подсхемы и др. Исходное и результирующее описания должны быть функционально эквивалентны. Целью разбиения логических схем на подсхемы является решение задач, возникающих при проектировании цифровых устройств.

Задача разбиения логических схем на подсхемы является важным аспектом процесса проектирования цифровых устройств [1] по многим причинам. Она встречается при размещении устройства на отдельных конструктивных компонентах [2–5], таких как печатные платы, микросхемы, и состоит в том, чтобы разбить схему на части (блоки) с учетом ограничений на число элементов в блоках, число внешних выводов блоков, суммарную площадь, занимаемую элементами и соединениями, и др. Основными критериями для данного разбиения являются число образующихся блоков, число внешних выводов на блоках, задержки в распространении сигналов, электромагнитная и тепловая совместимость элементов и т. д.

Задача разбиения часто используется при реализации алгоритмов оптимизации, синтеза, размещения в процессе проектирования устройств [6–11]. В рассматриваемом случае оценка стоимости разбиения варьируется, но часто минимизируется сложность блоков разбиения и количество связей между ними.

Микросхема может содержать десятки миллионов транзисторов, а оптимизирующие преобразования потребуют большого количества времени даже при современном уровне развития вычислительных средств. Большинство задач, возникающих при проектировании, являются NP-полными, время их выполнения резко возрастает с увеличением размера схемы. Понижение размерности схемы служит хорошим способом решения задачи сокращения времени выполнения процедур оптимизации.

Целью настоящего исследования является решение задачи понижения размерности логической схемы путем разбиения на блоки меньшей размерности, разработка алгоритмов разбиения и экспериментальная оценка их эффективности.

Задача разбиения схемы на подсхемы. Под разбиением будем понимать процесс разложения логической схемы на более мелкие подсхемы, которые называются блоками. Процедура разбиения используется в процессе технологически независимой оптимизации схемы, выполняемой перед процедурой технологического отображения в процессе синтеза схемы, и позволяет существенно сократить время процедур оптимизации, которые работают значительно быстрее на блоках меньшего размера.

Задача разбиения может быть выражена в терминах теории графов. Текущим представлением логической схемы является взвешенный ориентированный граф $G = (V, E)$, каждая вершина которого рассматривается как логический компонент, а дуги представляют связи между компонентами. Задачу разбиения можно представить как задачу распределения множества V вершин графа G в множестве непересекающихся подмножеств вершин $\{V_1, \ldots, V_p\}$ графа (блоков), таких, что $V_i \subset V$, $V_i \neq \varnothing$, $V_i \bigcap\limits_{i \neq j} V_j = \varnothing$, $\bigcup\limits_{i=1}^{p} V_i = V$ для $i, j \in \{1, \ldots, p\}$, $i \neq j$. Вершины графа имеют весовые характеристики, например сложности описания соответствующих компонентов.

Ограничения и целевые функции для задачи разбиения варьируются для каждого уровня применения задачи и этапа проектирования. Наиболее распространенной целевой функцией является минимизация числа ребер в разрезе между блоками (проблема mincut):

$$\sum_{i=1}^{p}\sum_{j=1}^{p} c_{ij} \to \min,\ i \neq j, \tag{1}$$

где c_{ij} – количество дуг между блоками V_i и V_j.

Возможны ограничения на число блоков разбиения, число входов и выходов каждого блока, задержку схемы, площадь каждого блока и др. Исходя из ограничений, для решения задачи разбиения строится соответствующая целевая функция.

Классификация алгоритмов разбиения. Алгоритмы разбиения по принципу работы делятся на две большие группы: конструктивные и итерационные. Конструктивные алгоритмы [4, 5] обычно используются для формирования некоторого начального разбиения, которое может быть улучшено с помощью других алгоритмов. Они выполняют построение разбиения по заранее заданным правилам присоединения вершин к блоку. Процесс характеризуется последовательным построением блоков. Конструктивные алгоритмы обладают высоким быстродействием.

Итерационные алгоритмы первоначально используют разбиение графа на заданное число блоков произвольным образом либо с помощью конструктивного алгоритма. Затем по определенным правилам производится перестановка вершин из одной части в другую с целью минимизации числа дуг между блоками. Оптимизация разбиения достигается парными или групповыми перестановками вершин графа между различными блоками.

Итерационные алгоритмы принимают в качестве исходных данных множество блоков и список дуг между ними и генерируют улучшенный набор блоков с измененным списком дуг. К итерационным относятся алгоритмы групповой миграции [6–9], алгоритмы моделирования отжига [3, 10], генетические алгоритмы [11] и др.

Алгоритмы групповой миграции можно отнести к наиболее ранним разработкам. Они принадлежат к классу алгоритмов итеративного улучшения. Целью таких алгоритмов является уменьшение (или увеличение) некоторой целевой функции. Алгоритмы групповой миграции начинаются с некоторых начальных разбиений, обычно генерируемых случайным образом. Затем к полученному разбиению применяются некоторые локальные изменения, чтобы уменьшить значение целевой функции. Процесс повторяется до тех пор, пока не будет достигнуто допустимое качество решения. Наиболее яркими представителями этих алгоритмов являются KL [7], FM [8], hMETIS [9].

Алгоритмы, рассматриваемые в настоящей работе, можно отнести к группе конструктивных алгоритмов. Они позволяют выполнять процедуры оптимизации исходных описаний для логических схем практической размерности.

Разбиение логической схемы на подсхемы. Среди различных способов задания функционирования логической схемы важное место занимает описание ее поведения с помощью системы логических уравнений. В работе рассматриваются системы логических уравнений, представленных в алгебраической форме.

Дана система логических уравнений

$$\begin{gathered} y_1 = f_1(x_1, x_2, ..., x_n), \\ y_2 = f_2(x_1, x_2, ..., x_n), \\ ..., \\ y_m = f_m(x_1, x_2, ..., x_n), \end{gathered} \tag{2}$$

где $x_1, x_2, ..., x_n$ – входные булевы переменные; $y_1, y_2, ..., y_m$ – выходные булевы переменные; $f_1, f_2, ..., f_m$ – булевы функции.

Требуется разбить систему (2) на подсистемы $S_1, S_2, ..., S_k, ..., S_p$ так, чтобы число входных и выходных переменных не превышало заданных ограничений n и m и число блоков разбиения p было минимальным.

Основная идея алгоритма разбиения. Подсистемы формируются последовательно в два этапа:

1. Выбор стартового уравнения для подсистемы.
2. Включение уравнений системы в подсистему с проверкой ограничений.

На первом этапе возможны различные варианты выбора стартового уравнения:

– с максимальным числом переменных;

– описывающее выходную переменную (от выхода);

– содержащее максимальное число входных переменных (от входа).

Действия на втором этапе условно можно разделить на две группы:

– выбор уравнения для включения в подсистему;

– проверка выполнения ограничений.

На каждом шаге второго этапа в подсистему включается только одно уравнение. Выбор уравнения для включения в подсистему осуществляется исходя из следующих соображений (приводятся в порядке предпочтения):

– поскольку алгоритм направлен на устранение промежуточных переменных, в первую очередь выбираются уравнения, позволяющие удалять промежуточные переменные, т. е. связанные с подсистемой по входам или выходам;

– выбираются уравнения, добавляющие минимальное число новых входных переменных в подсистему.

Выбранное уравнение включается в текущую подсистему S_k (рис. 1), которая подвергается процедуре элиминации (устранения промежуточных переменных). Если полученная подсистема удовлетворяет заданным ограничениям, выбирается очередное уравнение для подсоединения.

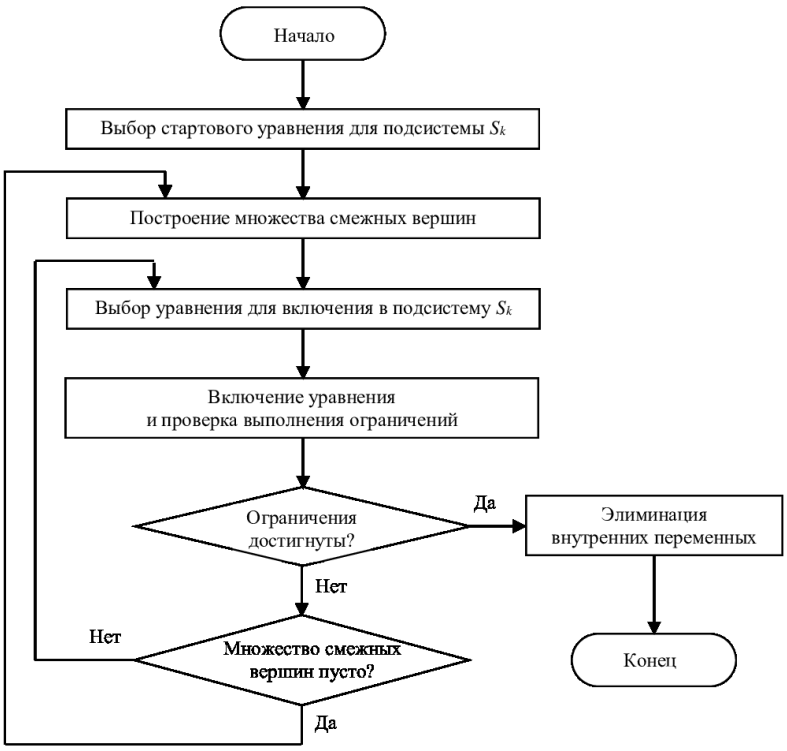

Рис. 1. Алгоритм формирования подсистемы S_k

Возможны следующие варианты процедуры элиминации для подсистемы логических уравнений:

– выполнять серию шагов по включению уравнений в подсистему, пока не будут достигнуты ограничения на число входов и выходов, затем выполнить элиминацию;

– выполнять элиминацию при включении каждого уравнения в подсистему.

Второй вариант требует больших временных затрат, но позволяет получить более качественное решение.

Если ни одно из оставшихся уравнений исходной системы не может быть включено в подсистему (без нарушения ограничений), то формирование подсистемы S_k считается законченным и выполняется переход к формированию следующей подсистемы.

Если все уравнения уже включены в подсистемы, то алгоритм заканчивает работу.

Этап включения уравнений системы в подсистему с проверкой ограничений. Для включения уравнения в подсистему строится множество смежных вершин для текущего состояния подсистемы, в которое помещаются уравнения из исходной системы, содержащие в качестве входных переменных выходные переменные подсистемы или в качестве выходных переменных – входные переменные подсистемы, т. е. вершины, связанные с подсистемой по входам или выходам. В дальнейшем изложении для обозначения уравнения используется термин «вершина» в тех случаях, когда система уравнений представлена в виде графа.

Далее выполняется перебор уравнений из множества смежных вершин и делается попытка включения каждого из них (по очереди) в подсистему. При включении каждого уравнения в подсистему осуществляется проверка ограничений. Если в результате получается подсистема, не удовлетворяющая ограничениям на число входных и выходных переменных, то уравнение не включается в подсистему и рассматривается следующее уравнение из множества смежных вершин. Если множество смежных вершин на данном шаге оказывается пустым, то строится новое множество смежных вершин для текущего состояния подсистемы S_k.

После того как будут достигнуты ограничения и закончится формирование подсистемы, выполняется процедура элиминации внутренних переменных, заключающаяся в подстановке выражений для внутренних переменных в уравнения подсистемы. При этом осуществляются действия по упрощению подсистемы, в результате которых может сократиться число уравнений и элементарных конъюнкций в подсистеме.

Построение множества смежных вершин. На каждом шаге формирования очередного блока разбиения строится множество смежных вершин для текущего состояния подсистемы. В это множество помещаются уравнения, которые позволяют эффективно выполнить процедуру элиминации промежуточных переменных подсистемы.

На рис. 2 показаны простые связи уравнения с подсистемой, когда выходная переменная подсистемы является входной переменной уравнения (рис. 2, *а*) и когда выходная переменная уравнения является входной переменной для подсистемы (рис. 2, *б*).

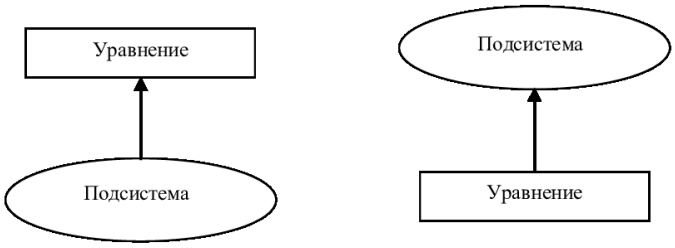

Рис. 2. Простые связи уравнения с подсистемой по выходу (*а*) и входу (*б*)

На рис. 3 изображены связи уравнения с подсистемой в виде разветвления, когда выходная переменная подсистемы имеется в качестве входной переменной в нескольких уравнениях (рис. 3, *а*) и когда выходная переменная уравнения имеется в качестве входной переменной подсистемы и нескольких уравнений исходной системы уравнений (рис. 3, *б*).

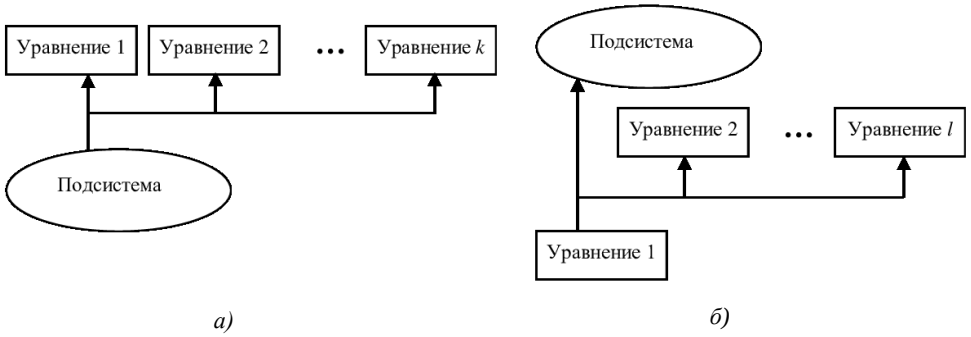

Рис. 3. Примеры разветвлений при формировании подсистем логических уравнений по выходу (*а*) и входу (*б*)

В случае разветвлений в алгоритме возникают новые критерии при включении очередного уравнения в подсистему. Необходимо включать или все уравнения разветвления, или ни одного, так как включение одного уравнения разветвления не позволит сократить число внутренних переменных подсистемы. Процедура включения очередного уравнения в подсистему является ключевой в алгоритме, от нее зависит качество получаемого разбиения.

В множестве смежных вершин различают два типа вершин. Вершину, которая связана по входу или выходу только с уравнениями подсистемы, назовем вершиной типа А (рис. 2). Она может быть выходной (рис. 2, *а*) или входной (рис. 2, *б*).

Вершину, на вход которой поступает выходная переменная подсистемы и эта же выходная переменная поступает на вход других вершин в сети (рис. 3, *а*), назовем выходной вершиной типа В. Вершину, выход которой поступает на вход подсистемы и, кроме этого, на вход других вершин в сети (рис. 3, *б*), назовем входной вершиной типа В.

Множество смежных вершин разделяется на группы, которые перечислены ниже в порядке предпочтения просмотра:

1) выходные вершины типа В на предмет включения всего разветвления с элиминацией;
2) выходные вершины типа В на предмет включения всего разветвления без элиминации;
3) выходные вершины типа А с элиминацией;
4) входные вершины типа А с элиминацией;
5) выходные вершины типа А без элиминации;
6) входные вершины типа А без элиминации;
7) выходные вершины типа В на предмет включения каждой вершины без элиминации;
8) входные вершины типа В на предмет включения каждой вершины без элиминации.

Предлагается множество смежных вершин просматривать несколько раз для выбора очередной вершины и включения ее в подсистему. В каждом просмотре выбираются вершины одной группы.

Алгоритмы модулей. Алгоритм процедуры разбиения логической схемы на подсхемы реализуется с помощью набора модулей, каждый из которых выполняет определенные функции.

Для построения множеств смежных вершин используются следующие модули:

b_out – модуль построения множества смежных вершин AB_Out. В него входят выходные вершины типа А (рис. 2, *а*) и В (рис. 3, *а*);

b_in – модуль построения двух множеств смежных вершин: A_inp, в которое входят входные вершины типа А (рис. 2, *б*), и B_inp, в которое входят входные вершины типа В (рис. 3, *б*);

b_all – модуль построения множества вершин All, не смежных с формируемой подсистемой. В него входят все уравнения исходной системы, которые еще не подключены в сформированные подсистемы и не смежные с формируемой подсистемой.

Для включения уравнений из множеств смежных вершин в формируемую подсистему используются следующие модули:

proc_out – модуль включения уравнений из множества AB_Out;

proc_in – модуль включения уравнений из множеств A_inp, pB_inp;

proc_all – модуль включения уравнений из множества All.

В множество уравнений Equations первоначально помещаются все уравнения исходной системы, при включении уравнений в формируемую подсистему они удаляются из этого множества.

Алгоритм build_k процедуры построения k-го блока разбиения. Алгоритм построения *k*-го блока разбиения (см. рис. 1) содержит шаги по построению множеств смежных вершин (уравнений) для текущего состояния *k*-й подсистемы (AB_Out, A_Inp, B_Inp, All) и шаги по включению уравнений из этих множеств в формируемую подсистему.

При описании алгоритма используется следующее обозначение: include – количество подключенных уравнений в подсистему (на очередном шаге алгоритма).

Шаги алгоритма build_k:

1. Найти в множестве Equations уравнение с максимальным числом входных переменных.
2. include = 0.
3. Выполнить модуль b_out.
4. Выполнить модуль proc_out (формируется значение include).

5. Если include ≠ 0, переход на шаг 2.
6. include = 0.
7. Выполнить модуль b_in.
8. Выполнить модуль proc_in (формируется значение include).
9. Если include ≠ 0, переход на шаг 6.
10. include = 0.
11. Выполнить модуль b_all.
12. Выполнить модуль proc_all (формируется значение include).
13. Если include ≠ 0, переход на шаг 10.
14. Элиминация внутренних переменных.
15. Конец построения *k*-го блока разбиения.

Алгоритм partopt процедуры разбиения системы уравнений на блоки с ограничением на число входов и выходов. При описании алгоритма используются следующие обозначения:

inp – ограничение на число входов подсистемы;

out – ограничение на число выходов подсистемы;

num_eq – количество уравнений в множестве Equations, еще не подключенных в уже сформированные блоки;

k – номер формируемого блока;

$Block_k$ – множество уравнений, входящих в формируемую подсистему (*k*-й блок).

Шаги алгоритма partopt:

1. Сформировать множество уравнений Equations, поместив в него все уравнения исходной системы.
2. $k = 0$.
3. Определить num_eq.
4. Если num_eq = 0, переход на шаг 9.
5. Сформировать пустое множество $Block_k$.
6. Выполнить процедуру build_k.
7. Сформировать SF-описание для блока $Block_k$.
8. $k = k + 1$, переход на шаг 3.
9. Конец.

Результаты исследования алгоритмов. Рассмотренные алгоритмы были использованы при разработке процедур разбиения логических схем на подсхемы. Процедуры включены в ряд экспериментальных систем логического проектирования FLC [12], ЭЛС [13], CMOSLD [14], разработанных в лаборатории логического проектирования ОИПИ НАН Беларуси.

Были исследованы три процедуры разбиения. Основной идеей эксперимента являлось использование для преобразования схемы сочетания процедур разбиения схемы на блоки и дальнейшей логической оптимизации блоков. В качестве средства логической оптимизации выбрана процедура BDD-оптимизации функционального описания систем полностью определенных булевых функций с минимизацией числа коэффициентов разложений Шеннона с точностью до инверсий [15]. Выполнение процедуры разбиения на блоки позволило понизить размерность задачи для оптимизационных преобразований (что хорошо сказалось на времени выполнения), а также получить функциональные описания, схемная реализация которых даст лучшие характеристики.

Исследование проведено на примерах известной серии для оценки алгоритмов (URL: http://www1.cs.columbia.edu/~cs6861/sis). Для каждого примера синтезировалась схема в базисе библиотеки проектирования КМОП-элементов с помощью промышленной системы синтеза LeonardoSpectrum [16]. После завершения синтеза система LeonardoSpectrum позволяла оценить характеристики полученной схемы: площадь кристалла (число транзисторов синтезированного описания) и количество базовых ячеек. В эксперименте рассматривалась только площадь схемы.

В процессе исследования для каждого примера были построены пять вариантов схемы с помощью синтезатора LeonardoSpectrum. Варианты различались применением различных процедур разбиения схем, выполненных перед этапами оптимизации и синтеза. Два варианта не ис-

пользовали процедуры разбиения логических схем на блоки, а три варианта использовали различные процедуры разбиения.

Результаты исследования представлены в таблице. Для каждого примера приведены параметры схемы: n – число входных переменных; m – число выходных переменных; S1, S2, S3, S4, S5 – значения площадей схемы (выраженных в числе транзисторов) по разным вариантам синтеза. Для каждой схемы определены: t_1 – время выполнения BDD-оптимизации функционального описания, не разбитого на блоки; t_2 – время выполнения BDD-оптимизации функционального описания, разбитого на блоки.

Для сравнительной оценки получены следующие варианты решений без разбиения схемы на блоки:

1. Исходная схема синтезировалась без предварительной оптимизации. Площадь полученной схемы S1.

2. Исходная схема синтезировалась с предварительной оптимизацией с помощью процедуры BDD-оптимизации. Площадь полученной схемы S2, время выполнения BDD-оптимизации t_1.

Результаты исследования алгоритмов разбиения

Пример	Параметры схемы		Синтез по исходному описанию	Синтез с оптимизацией BDD	Время оптимизации BDD, с	Синтез схемы, полученной путем разбиения на блоки и BDD-оптимизации блоков			
						Процедура parti	Процедура partopt без элиминации	Процедура partopt с элиминацией	Время оптимизации BDD, с
	n	m	S1	S2	t_1	S3	S4	S5	t_2
Apex6	135	94	1830	2110	156	**1816**	1862	1858	9
C8	28	18	312	324	1,5	**310**	320	316	1
Cht	47	36	680	670	1,5	**656**	668	668	0,7
Count	35	168	256	256	4	256	256	256	1,2
Dalu	75	16	1834	**1396**	266	2070	1876	1886	25
X3	135	99	3462	3788	2079	3446	3514	**3342**	25
term1	34	10	1044	3086	32	1034	**908**	1090	12
x4	94	71	1544	2892	23	**1424**	1528	1488	9
ttt2	24	21	698	766	20	658	666	**654**	3
example2	85	63	850	996	8	836	848	**826**	3
Кол-во лучших решений по площади			–	**1**		**8**			

Для оценки эффективности процедуры разбиения получены три варианта синтезированной схемы:

1. Процедура parti выполняла построение блоков, наполняя их уравнениями в произвольном порядке, например в порядке следования в исходном описании. Далее следовали BDD-оптимизация каждого блока, процедура устранения иерархии описания и синтез. Площадь полученной схемы S3.

2. Процедура partopt_w выполняла построение блоков, наполняя их уравнениями, связанными между собой по входам или выходам. Далее следовали BDD-оптимизация каждого блока, процедура устранения иерархии описания и синтез. Площадь полученной схемы S4.

3. Процедура partopt выполняла построение блоков, наполняя их уравнениями, связанными между собой по входам или выходам. Для каждого блока осуществлялась процедура ликвидации внутренних переменных (элиминации) в полученной системе булевых уравнений. Далее следовали BDD-оптимизация каждого блока, процедура устранения иерархии описания и синтез. Площадь полученной схемы S5.

Время t_2 BDD-оптимизации для схем, разбитых на блоки, для процедур 1–3 различается незначительно (в таблице представлено среднее значение).

Заключение. Результаты эксперимента показывают, что процедуры разбиения улучшают площадь синтезированных схем в восьми случаях из десяти. В одном случае лучшее решение получено при применении BDD-оптимизации к исходной схеме без использования процедуры

разбиения на блоки (Dalu). Один пример (Count) дает абсолютно одинаковые результаты для любых методов оптимизации. Для всех примеров получен выигрыш по времени оптимизации, разбиение на блоки для примера X3 позволило получить выигрыш по времени в 83 раза. Процедуры разбиения схемы на блоки в большинстве случаев повышают эффективность выполнения дальнейших оптимизационных преобразований схемы, сокращают время выполнения процедур оптимизации и позволяют решать оптимизационные задачи для схем большой размерности за приемлемое время. Полученные схемные решения выигрывают по качеству.

Разработанные алгоритмы были реализованы в виде набора процедур разбиения схемы в рамках экспериментальной системы логического проектирования FLC [12].

Список использованных источников

1. Рабаи, Ж. М. Цифровые интегральные схемы. Методология проектирования / Ж. М. Рабаи, А. Чандракасан, Б. Николич. – М. : Вильямс, 2007. – 912 с.

2. Partitioning-based methods / ed.: C. J. Alpert, D. P. Mehta, S. S. Sapatnekar // Handbook of Algorithms for Physical design Automation. – CRC Press, 2009. – P. 290–308.

3. Global and detailed placement / A. B. Kahng [et al.] // VLSI physical design: Graph Partitioning to Timing Closure. – Springer, 2011. – P. 95–122.

4. Bibilo, P. Block synthesis of combinational circuits / P. Bibilo, N. Kirienko // Design of Embedded Control Systems. – Springer, 2004 – P. 189–196.

5. Базилевич, Р. П. Алгоритмические и программные средства для размещения разногабаритных элементов на конструктиве / Р. П. Базилевич, И. Ф. Щербюк // Автоматизация проектирования дискретных систем : материалы Шестой Междунар. конф. – Минск : ОИПИ НАН Беларуси, 2007. – С. 157–164.

6. Breuer, M. A. Fundamental CAD algorithms / M. A. Breuer, M. Sarrafzadeh, F. Somenzi // IEEE Trans. Computer-Aided Design. – 2000. – Vol. 19, no. 12. – P. 1449–1475.

7. Kernighan, B. W. An efficient heuristic procedure for partitioning graphs / B. W. Kernighan, S. Lin. // Bell Labs Technical J. – 1970. – Vol. 49. – P. 291–307.

8. Fiduccia, C. M. A linear time heuristic for improving network partitions / C. M. Fiduccia, R. M. Mattheyes // Proc. IEEE-ACM Design Automation Conf., Las Vegas, Nevada, USA, 14–16 June 1982. – Las Vegas, 1982. – P. 175–181.

9. Karypis, G. Multilevel k-way hypergraph partitioning / G. Karypis, V. Kumar // Proc. IEEE-ACM Design Automation Conf., New Orleans, USA, 21–25 June 1999. – New Orleans, 1999. – P. 343–348.

10. Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning / D. S. Johnson [et al.] // Operations Research. – 1989. – Vol. 37. – P. 865–892.

11. Гладков, Л. А. Генетические алгоритмы / Л. А. Гладков, В. В. Курейчик, В. М. Курейчик ; под ред. В. М. Курейчика. – М. : Физматлит, 2006. – 320 с.

12. 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.

13. Автоматизация логического синтеза КМОП-схем с пониженным энергопотреблением / П. Н. Бибило [и др.] // Программная инженерия. – 2013. – № 8. – С. 35–41.

14. A system for logical design of custom CMOS VLSI functional blocks with reduced power consumption / P. N. Bibilo [et al.] // Russian Microelectronics. – 2018. – Vol. 47, no. 1. – P. 65–81.

15. Бибило, П. Н. Логическая оптимизация многоуровневых представлений систем булевых функций на основе блочного разбиения и разложения Шеннона / П. Н. Бибило, Н. А. Кириенко, Ю. Ю. Ланкевич // Информатика. – 2018. – Т. 15, № 3. – С. 56–70.

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

References

1. Rabaey Jan M. *Digital Integrated Circuits: A Design Perspective*. Prentice Hall, 1995, 702 p.

2. Alpert C. J., Mehta D. P., Sapatnekar S. S. (eds.). Partitioning-based methods. *Handbook of Algorithms for Physical design Automation.* CRC Press, 2009, pp. 290–308.

3. Kahng A. B., Liening J., Markov I. L., Hu J. Global and detailed placement. *VLSI physical design: Graph Partitioning to Timing Closure*. Springer, 2011, pp. 95–122.

4. Bibilo P., Kirienko N. Block synthesis of combinational circuits. *Design of Embedded Control Systems*. Springer, 2004, pp. 189–196.

5. Bazilevich R. P., Shcherbyuk I. F. Algoritmicheskie i programmnye sredstva dlya razmeshcheniya raznogabaritnyh elementov na konstruktive [Algorithmic and software tools for placing oversized elements on a construct]. Avtomatizaciya proektirovaniya diskretnyh sistem: materialy Shestoy Mezhdunarodnoj konferencii [*Computer-Aided Design of Discrete Devices: Proceedings of the Sixth International Conference*]. Minsk, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus, 2007, pp. 157–164 (in Russian).

6. Breuer M. A., Sarrafzadeh M., Somenzi F. Fundamental CAD algorithms. *IEEE Transactions Computer-Aided Design*, 2000, vol. 19, no. 12, pp. 1449–1475.

7. Kernighan B. W., Lin S. An efficient heuristic procedure for partitioning graphs. *Bell Labs Technical Journal*, 1970, vol. 49, pp. 291–307.

8. Fiduccia C. M., Mattheyes R. M. A linear time heuristic for improving network partitions. *Proceedings IEEE-ACM Design Automation Conference, Las Vegas, Nevada, USA, 14–16 June 1982*. Las Vegas, 1982, pp. 175–181.

9. Karypis G., Kumar V. Multilevel k-way hypergraph partitioning, *Proceedings IEEE-ACM Design Automation Conference, New Orleans, USA, 21–25 June 1999*. New Orleans, 1999, pp. 343–348.

10. Johnson D. S., Aragon C. R., Mcgeoch L. A., Schevon C. Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. *Operations Research*, 1989, vol. 37, pp. 865–892.

11. Gladkov L. A., Kurejchik V. V., Kurejchik V. M. (ed.) Geneticheskie algoritmy. *Genetic Algorithms*. Moscow, Fizmatlit, 2006, 320 p. (in Russian).

12. 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).

13. Bibilo P. N., Cheremisinova L. D., Kardash S. N., Kirienko N. A., Romanov V. I., Cheremisinov D. I. Avtomatizaciya logicheskogo sinteza KMOP-skhem s ponizhennym energopotrebleniem [Low-power logical synthesis of cmos circuits automation]. Programmnaya inzheneriya [*Software Engin*eering], 2013, no. 8, pp. 35–41 (in Russian).

14. Bibilo P. N., Avdeev N. A., Kardash S. N., Kirienko N. A., Lankevich Yu. Yu., …, Cheremisinova L. D. A system for logical design of custom CMOS VLSI functional blocks with reduced power consumption. *Russian Microelectronics*, 2018, vol. 47, no. 1, pp. 65–81.

15. Bibilo P. N., Kirienko N. A., Lankevich Yu. Yu. Logicheskaya optimizaciya mnogourovnevyh predstavlenij sistem bulevyh funkcij na osnove blochnogo razbieniya i razlozheniya Shennona [Logical optimization the multilevel representations of systems of Boolean functions based on partitioning into blocks and Shannon decomposition]. Informatika [*Informatics*], 2018, vol. 15, no. 3, pp. 56–70 (in Russian).

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

Информация об авторе

Кириенко Наталья Алексеевна, кандидат технических наук, доцент, старший научный сотрудник, Объединенный институт проблем информатики Национальной академии наук Беларуси, Минск, Беларусь.
E-mail: natalia.kirienko@tut.by

Information about the author

Natalia A. Kirienko, Cand. Sci. (Eng.), Associate Professor, Senior Researcher, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus, Minsk, Belarus.
E-mail: natalia.kirienko@tut.by