УДК 681.325

# ВЫПОЛНЕНИЕ ОПТИМИЗАЦИОННЫХ ЗАДАЧ ЛОГИЧЕСКОГО ПРОЕКТИРОВАНИЯ НА БОЛЬШИХ ДАННЫХ С ПОМОЩЬЮ ПРОЦЕДУРЫ РАЗБИЕНИЯ

***Н.А. Кириенко***
*Старший научный сотрудник ОИПИ НАНБ*
*кандидат технических наук, доцент*

*Объединенный институт проблем информатики Национальной академии наук Беларуси, Республика Беларусь.*

***Н. А. Кириенко***
*Окончила Минский радиотехнический институт. Старший научный сотрудник ОИПИ НАНБ, доцент кафедры экономической информатики БГУИР (по совместительству), кандидат технических наук, доцент. Направления научных исследований: автоматизация логического проектирования, преобразование функциональных описаний дискретных устройств.*

**Аннотация.** Функциональные описания проектируемых цифровых устройств имеют сотни входных переменных и десятки тысяч уравнений. Выполнение комбинаторных алгоритмов на данных такой размерности требует большого времени счета на компьютере (до нескольких часов), что является недопустимым в процессе проектирования. Предлагается сократить время выполнения оптимизационных преобразований путем разбиения исходного функционального описания на блоки. Приводится краткий обзор методов и алгоритмов разбиения, выделяются две группы алгоритмов: конструктивные и итерационные. Представляется интерпретация логической схемы в виде графа, формулируется задача разбиения в теоретико-графовой модели. Функционирование логической схемы задается системой логических уравнений. Описывается алгоритм разбиения системы логических уравнений на подсистемы с выполнением ограничений по числу входных и выходных переменных. Результаты экспериментального исследования, выполненного с помощью процедур разбиения и BDD-оптимизации функционального описания схемы, подтверждают значительное сокращение времени выполнения оптимизационных преобразований.

**Ключевые слова:** логическая схема, технологически независимая оптимизация, алгоритмы разбиения, системы булевых функций, синтез логических схем.

## Введение.

Процесс синтеза логических устройств [1] содержит два больших этапа: технологически независимую оптимизацию исходного представления логического устройства и технологическое отображение оптимизированного описания в схему, состоящую из элементов целевой библиотеки проектирования. Этап технологически независимой оптимизации характеризуется выполнением ряда процедур преобразования исходного описания устройства с целью сокращения значений его характеристик – числа промежуточных переменных, конъюнкций, рангов логических выражений и др.

Большинство оптимизационных задач, возникающих при проектировании дискретных устройств, 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.

Как правило, исходное функциональное описание проектируемого цифрового устройства представлено на одном из высокоуровневых языков проектирования – широко распространенных VHDL, Verilog, или специализированном SF [2], используемом в исследовательских системах

автоматизированного проектирования цифровых устройств (САПР цифровых устройств). На выходе САПР должно быть получено результирующее описание цифрового устройства в виде описания схемы из взаимосвязанных элементов целевой библиотеки проектирования. Чтобы полученная схема была оптимальна по различным параметрам (площадь кристалла, быстродействие, энергопотребление), исходное функциональное описание необходимо подвергнуть различным оптимизирующим преобразованиям. Наиболее используемое и эффективное преобразование – минимизация исходного описания по различным параметрам: числу литералов, конъюнкций, рангу конъюнкций, и др.

Однако известные алгоритмы оптимизации существенно зависят от числа входных переменных функционального описания, и требуют большого количества времени даже при современном уровне развития вычислительных средств уже при числе входных переменных n больше 20. Реальные функциональные описания могут иметь до нескольких сотен входных переменных. Одним из возможных способов понижения размерности является использование процедуры разбиения функционального описания логической схемы на подсхемы (блоки), и выполнение алгоритмов оптимизации на схемах меньшей размерности.

В настоящей работе представлено решение задачи выполнения оптимизационных преобразований функциональных описаний логических схем большой размерности путем разбиения на блоки и исследованы результаты эффективности такого подхода.

**Постановка задачи.**

Задача разбиения логической схемы на блоки заключается в представлении исходного описания схемы в виде множества взаимосвязанных описаний схем меньшей размерности (блоков).

Задача разбиения одна из фундаментальных проблем в проектировании СБИС, она широко используется при проектировании дискретных устройств для понижения размерности описаний, для решения задач компоновки и размещения конструктивных элементов на платах [3–6].

Задача разбиения может быть выражена в терминах теории графов [7]. 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 {V1, …, Vp} графа (блоков), таких, что Vi V, Vi , $V_i \cap_{i \neq j} V_j = \varnothing$ , $\sum_{i=1}^{p} V_i = V$ для i, j {1, …, p}, i 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):

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

где cij– количество дуг между блоками Vi и Vj .

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.

В настоящее время известно большое количество алгоритмов разбиения [813]. Они делятся на две группы: конструктивные и итерационные. Конструктивные алгоритмы [5, 6] 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 [8–11], алгоритмы моделирования отжига [4, 12], генетические алгоритмы [13] 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 [9], FM [10], hMETIS [11].

**Алгоритм разбиения логической схемы на блоки.**

Разработанный алгоритм [7] относится к группе конструктивных алгоритмов.

Функциональное описание логической схемы задано системой логических уравнений.

y1 = f1 (x1, x2, ..., xn),

y2 = f2 (x1, x2, ..., xn),

..., (2)

ym = fm (x1, x2, ..., xn),

где x1, x2, ..., xn – входные булевы переменные; y1, y2, ..., ym – выходные булевы переменные; f1, f2, ..., fm – булевы функции.

Требуется разбить систему (2) на подсистемы S1, S2, ..., Sk, ..., Sp так, чтобы число входных и выходных переменных не превышало заданных ограничений n и m и число блоков разбиения p было минимальным.

Подсистемы формируются последовательно и включают 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 Sk(рисунок 1), которая подвергается процедуре элиминации (устранения промежуточных переменных). Если полученная подсистема удовлетворяет заданным ограничениям, выбирается очередное уравнение для подсоединения.

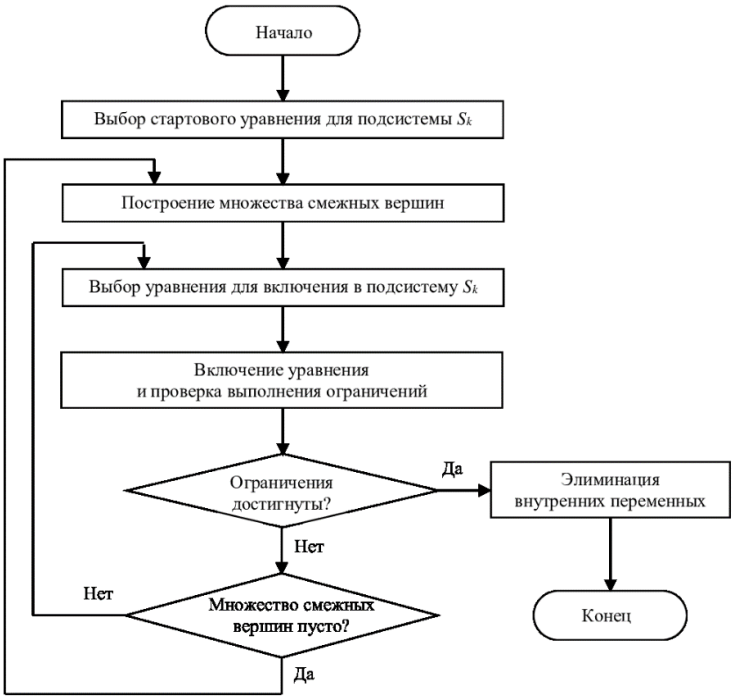


*Рисунок 1.* Алгоритм формирования подсистемы Sk

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 Sk считается законченным и выполняется переход к формированию следующей подсистемы.

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

Подробное изложение алгоритма представлено в [7].

Результаты экспериментального исследования. Разработанная процедура разбиения логических схем на блоки включена в ряд экспериментальных систем логического проектирования FLC2 [2], ЭЛС [14], CMOSLD [15], разработанных в лаборатории логического проектирования ОИПИ НАН Беларуси.

Основная задача исследования – сравнить время, затраченное на выполнение оптимизационных преобразований для исходного описания и подвергшегося разбиению. 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 [16].

Результаты исследования приведены в таблице 1. Исследование проведено на примерах известной серии для оценки алгоритмов (URL: http://www1.cs.columbia.edu/~cs6861/sis) и нескольких промышленных примерах (sistem8, timer). Имена схем представлены в столбце 1, параметры схем (n – число входов, m – число выходов) представлены в столбце 2. Для каждого примера выполнялась процедура BDD-оптимизации на исходном описании, значение затраченного времени в секундах представлено в столбце 3. Далее исходная схема подвергалась процедуре разбиения с ограничением на число входов k и число выходов l в блоке. Параметры разбиения представлены в столбце 4, количество полученных блоков в столбце 5, время в

секундах, затраченное на разбиение в столбце 6. В столбце 7 представлено время выполнения BDD-оптимизации блочного разбиения в секундах. В столбце 8 представлено значение относительного выигрыша во времени, которое вычисляется, как частное от деления времени BDD-оптимизации исходной схемы (столбец 4) на время BDD-оптимизации блочного разбиения (столбец 7).

Результаты эксперимента говорят о том, что процедура разбиения всегда приводит к сокращению времени оптимизационных преобразований, в некоторых случаях более чем в 500 раз. Хороший результат получен для схем с большим числом входов, что можно объяснить тем, что сложность оптимизационных алгоритмов существенно возрастает с ростом числа входных переменных.

*Таблица 1.* Исследование эффективности процедуры разбиения логических схем для. сокращения времени выполнения оптимизационных преобразований

| Пример | Параметры схемы *n, m* | Время выполнения BDD-оптимизации исходной схемы, сек | Параметры разбиения *k, l* | Количество блоков разбиения | Время разбиения схемы на блоки, сек | Время выполнения BDD-оптимизации блочного разбиения, сек | Относительный выигрыш |
|---|---|---|---|---|---|---|---|
| apex6 | 135, 94 | 161.00 | 30, 100 | 20 | 0.04 | 7.8 | 20.54 |
| dalu | 75, 16 | 286.00 | 30, 100 | 82 | 0.38 | 38.44 | 7.44 |
| x3 | 135, 99 | 540.00 | 30, 100 | 24 | 0.05 | 10.10 | 53.47 |
| sistem8 | 25, 20 | 168.00 | 15, 100 | 137 | 0.07 | 43.77 | 3.84 |
| frg2 | 143, 139 | 7460.00 | 40, 100 | 22 | 0.1 | 13.92 | 535.92 |
| soar | 83, 94 | 606.76 | 25, 100 | 12 | 0.03 | 14.11 | 42.99 |
| miex4 | 128, 28 | 641.12 | 20, 100 | 10 | 0.20 | 8.13 | 78.86 |
| miapex3 | 54, 50 | 55.53 | 50, 5 | 11 | 0.03 | 28.77 | 1.93 |
| timer | 84, 42 | 30.18 | 25, 100 | 6 | 0.09 | 3.30 | 9.15 |
| x4 | 94, 71 | 23.43 | 20, 100 | 18 | 0.03 | 5.95 | 3.94 |
| term1 | 34, 10 | 32.55 | 20, 100 | 20 | 0.03 | 9.76 | 3.34 |
| cordic | 23, 2 | 19.85 | 10, 100 | 19 | 0.02 | 4.25 | 4.67 |

**Заключение.**

Функциональные описания проектируемых цифровых устройств имеют сотни входных переменных и десятки тысяч уравнений. Выполнение комбинационных алгоритмов на данных такой размерности требует большого времени счета на компьютере (до нескольких часов), что является недопустимым в процессе проектирования. Процедура разбиения на блоки позволяет понизить размерность задачи для оптимизационных преобразований, что хорошо сказывается на времени выполнения. Для некоторых примеров время сокращается более чем в 500 раз.

### *Список литературы*

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

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

[3] 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.

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

[5] Bibilo, P. Block synthesis of combinational circuits / P. Bibilo, N. Kirienko // Design of Embedded Control Systems. – Springer, 2004 – P. 189-196.N. KirienkoN. Kirienko

[6] 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.

[7] Кириенко, Н.А. Алгоритмы разбиения логических схем на подсхемы / Н.А. Кириенко // Информатика. 2020. Т. 17, № 3. – С. 84-93.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] Гладков, Л. А. Генетические алгоритмы / Л. А. Гладков, В. В. Курейчик, В. М. Курейчик. – М. : Физматлит, 2006. – 320 с.

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

[15] 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.

[16] 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.

# IMPLEMENTATION OF OPTIMIZATION PROBLEMS OF LOGICAL DESIGN FOR BIG DATA USING A PARTITION PROCEDURE

***N.A. KIRIENKO,***
*Candidate of Technical Sciences,*
*Senior Researcher of UIIP of NAS of Belarus,*
*Associate Professor*

*The United Inst. of Informatics Problems of the National Academy of Sciences of Belarus, Belarus*

**Abstract.** Functional descriptions of designed digital devices have hundreds of input variables and tens of thousands of equations. The implementation of combinatorial algorithms on data of this dimension requires a large runtime on a computer (up to several hours), which is unacceptable in the design process. It is proposed to reduce the execution time of optimization by partitioning of the initial functional description into blocks. The brief review of partitioning methods and algorithms is presented, 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. A logical circuit is presented as a system of logical equations. An algorithm for partitioning a system of logical equations into subsystems with the fulfillment of restrictions on the number of input and output variables is described. The results of the experimental study carried out using the procedure of partitioning and BDD-optimization of the functional description of the circuit confirms a significant reduction in the execution time of optimization transformations.

**Keywords:** logic circuit, technologically independent optimization, partitioning algorithms, systems of Boolean functions, synthesis of logic circuits.