УДК 517.98
DOI: 10.15827/0236-235X.131.449-463
Дата подачи статьи: 22.01.20
2020. Т. 33. № 3. С. 449–463

Экспериментальное сравнение эффективности алгоритмов оптимизации BDD-представлений систем булевых функций

П.Н. Бибило [1], *д.т.н., профессор, зав. лабораторией, bibilo@newman.bas-net.by*
Ю.Ю. Ланкевич [1], *младший научный сотрудник, yurafreedom18@gmail.com*

[1] *Объединенный институт проблем информатики Национальной академии наук Беларуси, г. Минск, 220012, Беларусь*

В статье описываются результаты экспериментального сравнения программ технологически независимой минимизации сложности многоуровневых представлений систем полностью определенных функций на основе разложения Шеннона. Графической формой таких представлений являются Binary Decision Diagrams (BDD) – диаграммы двоичного выбора. Порядок переменных, по которым ведется разложение Шеннона, влияет на размер BDD, поэтому задача минимизации сложности формы представления сводится к поиску оптимального порядка переменных разложения, при котором количество вершин графа BDD будет минимальным.

Для поиска оптимального порядка переменных разложения используются эвристические алгоритмы, поскольку полный перебор для задач большой размерности сложно либо невозможно осуществить за приемлемое время с использованием современных вычислительных средств.

Сравниваются результаты, полученные с помощью новых и ранее предложенных эвристик. После получения минимизированных по числу вершин графов BDD, заданных в виде совокупности взаимосвязанных формул разложения Шеннона, выполняется синтез логических схем в одной и той же библиотеке проектирования заказных цифровых КМОП сверхбольших интегральных схем, результаты сравниваются по площади кристалла и по быстродействию (временной задержке).

Дополнительного сокращения сложности логических описаний и улучшения результатов синтеза схем во многих случаях можно добиться, выполняя дополнительную логическую минимизацию на основе булевых сетей. Критерием оптимизации в данном случае является число вершин булевой сети без учета инверсий булевых переменных, что хорошо согласуется с критерием «число литералов» при оптимизации многоуровневых логических схем.

Ключевые слова: *система булевых функций, дизъюнктивная нормальная форма, Binary Decision Diagram (BDD), синтез логической схемы, VHDL, СБИС, КМОП-технология.*

Математический аппарат *диаграммы двоичного выбора* (BDD) в настоящее время используется в различных областях науки [1, 2]. В системах проектирования цифровых *сверхбольших интегральных схем* (СБИС) графовый аппарат BDD применяется при верификации цифровых систем [3], программы минимизации BDD используются на этапе технологически независимой оптимизации [4] при синтезе логических схем.

Основное внимание в литературе уделялось поиску перестановки переменных, по которой производится разложение функций исходной системы и подфункций (cofactors), получаемых в процессе разложения с целью минимизации сложности BDD. Под сложностью понимается число вершин графа BDD, в этом графе каждой вершине соответствует полная либо редуцированная формула разложения Шеннона.

Известны также подходы, ориентированные на заключительный этап синтеза – технологическое отображение (technology mapping) на основе покрытия BDD (формул разложения Шеннона) описаниями логических элементов, в результате чего получаются структурные описания – нетлисты логических схем в заданном технологическом базисе, часто называемом целевой библиотекой логического синтеза.

В отечественных программных пакетах САПР [5, 6] используются несколько программ минимизации BDD-представлений систем булевых функций, реализующих различные алгоритмы. Целью настоящей работы является изучение эффективности применения данных алгоритмов и реализующих их программ при синтезе комбинационных блоков заказных КМОП СБИС.

Задача BDD-минимизации

BDD-минимизация булевых функций основана на разложении Шеннона. Разложением Шеннона булевой функции $f(x) = f(x_1, \ldots, x_n)$ по

переменной x_i является представление $f(x)$ в виде $f(x) = \overline{x}_i f_0 \vee x_i f_1$.

Каждый из коэффициентов (cofactors) $f_0 = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$, $f_1 = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$ может быть разложен по одной из переменных из множества $\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$. Процесс разложения коэффициентов заканчивается, когда все n переменных будут использованы для разложения. В процессе разложения либо на его последнем шаге некоторые коэффициенты могут вырождаться до констант 0, 1. На каждом шаге разложения осуществляется сравнение полученных коэффициентов на равенство и из множества равных коэффициентов оставляется один. Под BDD 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 (перестановке) переменных, по которым проводятся разложения. Функциональная вершина, соответствующая функции f, называется корнем, листовым вершинам соответствуют константы 0, 1. Далее для системы булевых функций $f(x) = (f^1(x), \ldots, f^m(x))$, $x = (x_1, \ldots, x_n)$, будут рассматриваться BDD, построенные по общей для всех компонентных функций $f^i(x)$ перестановке переменных. Иногда в литературе BDD такого вида называются совместными. Если исходная система функций представлена в виде системы *дизъюнктивных нормальных форм* (ДНФ), заданной на общем множестве элементарных конъюнкций, то в результате BDD-оптимизации матричная форма исходной системы функций заменяется графом BDD, каждой вершине которого соответствует полная либо сокращенная (редуцированная) формула разложения Шеннона. Пример графа BDD, построенного для системы (ДНФ)

$$f^1 = x_1 x_2 x_3 \overline{x}_4 \vee \overline{x}_2 \overline{x}_3 \overline{x}_4 \vee \overline{x}_1 x_2 x_4;$$

$$f^2 = \overline{x}_1 \overline{x}_2 \overline{x}_3 x_4 \vee x_1 \overline{x}_2 \overline{x}_3 \overline{x}_4 \vee x_1 x_2 \overline{x}_3 x_4 \vee \vee \overline{x}_1 x_2 x_4 \vee x_3 x_4 \vee x_2 x_3;$$

$$f^3 = \overline{x}_1 \overline{x}_2 \overline{x}_3 x_4 \vee x_1 \overline{x}_2 \overline{x}_3 \overline{x}_4 \vee x_1 x_2 \overline{x}_3 x_4 \vee \vee x_1 x_2 x_3 \overline{x}_4 \vee \overline{x}_1 x_2 \overline{x}_4$$

булевых функций ($n = 4$, $m = 3$), приведен на рисунке 1. Матричная форма $<T^x, B^f>$ (табл. 1) рассматриваемой системы ДНФ булевых функций состоит из троичной матрицы T^x задания элементарных конъюнкций в виде троичных векторов и булевой матрицы B^f вхождений конъюнкций в ДНФ компонентных функций.

Данному графу BDD (рис. 1), построенному по перестановке $<x_3, x_2, x_1, x_4>$ переменных, соответствуют 12 формул разложения Шеннона (^ обозначает инверсию, * – конъюнкцию, + – дизъюнкцию в языке SF, являющемся внутренним языком пакетов [5, 6]):

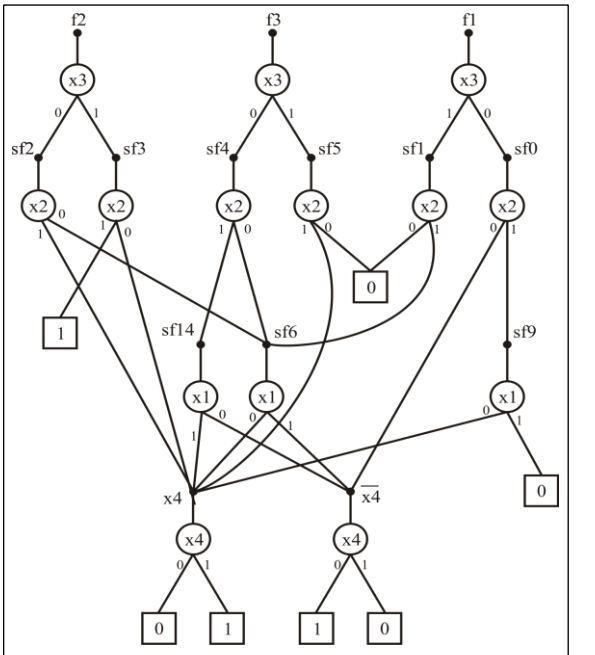

Рис. 1. Граф BDD

Fig. 1. BDD Graph

*f*1 = ^*x*3**sf*0 + *x*3**sf*1; *f*2 = ^*x*3**sf*2 + *x*3**sf*3;
*f*3 = ^*x*3**sf*4 + *x*3**sf*5; *sf*0 =^*x*2*^*x*4 + *x*2**sf*9;
*sf*4 = ^*x*2**sf*6 + *x*2**sf*14; *sf*2 = ^*x*2**sf*6 + *x*2**x*4;
*sf*3 = ^*x*2**x*4 + *x*2; *sf*5 = *x*2*^*x*4;
*sf*1 = *x*2**sf*6; *sf*14 = ^*x*1*^*x*4 + *x*1**x*4;
*sf*9 = ^*x*1**x*4; *sf*6 = ^*x*1**x*4 + *x*1*^*x*4.

Данные формулы содержат 9 операций дизъюнкции и 20 операций конъюнкции. Функциональные вершины нижнего (некон-

Таблица 1

Пример системы ДНФ трех булевых функций

Table 1

The example of a DNF system of three Boolean functions

T^x	B^f
$x_1\ x_2\ x_3\ x_4$	$f^1\ f^2\ f^3$
0 0 0 1	0 1 1
1 0 0 0	0 1 1
1 1 0 1	0 1 1
1 1 1 0	1 0 1
- 0 0 0	1 0 0
0 1 - 0	0 0 1
0 1 - 1	1 1 0
- - 1 1	0 1 0
- 1 1 -	0 1 0

стантного) уровня BDD равны переменной либо ее инверсии, в примере это функциональные вершины x_4, $\bar{x}_4$. В виде уравнений такие функции не будут записываться, поэтому число уравнений многоуровневого представления обычно меньше сложности BDD на одну либо две функциональные вершины. Под сложностью S_{BDD} в литературе обычно понимается число функциональных вершин BDD.

BDD задает в виде графа последовательность разложений Шеннона исходной функции и получаемых коэффициентов разложения по некоторой перестановке переменных. Минимизация сложности BDD основана на том, что в процессе разложения могут появляться одинаковые коэффициенты разложения не только у одной, но и у нескольких (либо даже у всех) компонентных функций. Хорошо известно, что от перестановки переменных, по которым ведется разложение, зависит сложность (число вершин) BDD, поэтому основной задачей компактного представления булевой функции (либо системы функций) в виде BDD является нахождение перестановки, дающей минимальное число вершин графа BDD. Далее под BDD-минимизацией будет пониматься оптимизация многоуровневых представлений систем булевых функций, соответствующих сокращенным упорядоченным BDD, то есть ROBDD (Reduced Ordered BDD). Подробное описание ROBDD дано в [1]. Далее под BDD будут пониматься ROBDD.

Задача 1 (BDD-минимизации). Для заданной системы ДНФ булевых функций найти такую перестановку $< x_{i_1}, x_{i_2}, ..., x_{i_n} >$ переменных, по которой будет получена BDD минимальной сложности.

Задача BDD-минимизации имеет достаточно богатую историю, аппарат BDD в настоящее время широко используется при решении SAT-проблем [7] многими исследователями. Оптимизации BDD-представлений систем полностью определенных булевых функций посвящено много работ, их обзоры и библиографию можно найти в [2, 8–10]. Не претендуя на полноту, сделаем краткий обзор.

Предшествующие работы. Появлению BDD предшествовало создание бинарных программ для представления булевых функций [11] и граф-схем решений [12]. Бинарные программы [13] содержат только два оператора: присвоения значений 0, 1 и условного перехода «если … то». Каждому оператору присваивалась метка, при выполнении программы для вычисления значения булевой функции переходы осуществляются на метки. По сути такая модель задает ориентированный граф BDD в другой форме. BDD является ориентированным графом, который может интерпретироваться как орграф с помеченными вершинами и дугами, пометки дуг могут отсутствовать, но порядок (сначала младший потомок, затем старший) может быть установлен по умолчанию. Для целей проектирования логических схем каждую вершину BDD удобно интерпретировать как мультиплексор; корневая вершина в данном случае соответствует выходному полюсу логической схемы, на котором реализуется функция, представленная BDD. Так как каждая вершина BDD может быть реализована мультиплексором с одним управляющим входом, синтез каскадных схем в базисе таких мультиплексоров по существу может быть отнесен к проблематике BDD, в таких методах также возникала проблема выбора нахождения лучшей перестановки переменных. Для мультиплексоров с несколькими управляющими входами это была проблема разбиения входных переменных на непересекающиеся блоки [14].

BDD-структуры данных. Термин BDD ввел Ш. Акерс и показал, что путям BDD от корневых вершин к листовым соответствуют ортогональные элементарные конъюнкции – непересекающиеся кубы [15]. Огромный интерес к BDD возник в связи с работой [16], в которой предложены эффективные алгоритмы для сокращения BDD и выполнения логических (двухместных) операций над BDD. Статья оказалась настолько актуальной, что стала наиболее цитируемой в информатике, поскольку революционизировала структуры данных, используемые для представления булевых функций [1, с. 303]. Суть этих алгоритмов сводится к выполнению сначала операции слияния BDD (получению совместной BDD-системы функций), а затем логических операций в листовых вершинах, после чего выполнялось сокращение BDD [4]. Подробное описание алгоритмов над структурами BDD дано в [1], там же приведены различные прикладные задачи, где может использоваться аппарат BDD, в том числе для задач астрономической сложности.

По сути эффективность представления систем функций в виде совместных BDD основана на том, что BDD в компактной графовой форме задают ортогонализованные представления систем функций, а именно, описание графа гораздо компактнее перечисления всех

путей на графе, так как каждому пути от корневой вершины до листовой соответствует элементарная конъюнкция ортогонализованной ДНФ. Эффективность оперирования BDD-представлениями систем функций заключается в том, что они задают ортогонализованные формы функций, операции над которыми требуют меньше времени, чем над произвольными ДНФ, задающими функции системы.

BDD в формальной верификации цифровых систем. Области применения BDD расширились, этот аппарат явился вычислительной основой решения задач верификации параллельных систем с конечным числом состояний на основе метода model checking, что позволило увеличить сложность анализируемых систем в миллионы миллионов раз [2, с. 295]. В работе [2] приводится много примеров применения BDD, там же указываются и некоторые из модификаций аппарата BDD. В [3] аппарат BDD применяется для целей верификации и тестирования цифровых систем.

BDD и SAT-проблемы. В настоящее время проблематика BDD связана также с решением SAT-проблем [7, 17]. Аппарат BDD был обобщен на случай конечнозначных функций, появились разнообразные обобщения классических BDD [18]. Перспективным признается применение BDD при решении SAT-задачи (задачи «выполнимость») [2, с. 329], а также идей BDD для решения логических уравнений [19].

BDD в синтезе логических схем. В области автоматизации проектирования цифровых систем применение BDD позволило достичь значительных успехов в формальной верификации алгоритмических описаний цифровых систем. Однако использование BDD в проектировании цифровых систем не ограничивается только верификацией. Компактность и удобство оперирования BDD привлекли их в качестве структур данных в системах автоматизированного проектирования цифровых систем на СБИС. Минимизация BDD стала одним из основных видов технологически независимой оптимизации в промышленных синтезаторах логических схем [3]. Она стала заменять классическую логическую минимизацию [20, 21] двухуровневых (И-ИЛИ) представлений дизъюнктивных нормальных форм, обычно представляемых в виде троичных и булевых матриц. Многоуровневые скобочные представления комбинационной логики – булевы сети [22], вершины которых соответствуют логическим функциям и операциям, успешно стали заменяться BDD-представлениями. Основные усилия были направлены на различные алгоритмы (например, [23–26] и др.) нахождения перестановок переменных, по которым строятся BDD, то есть на решение задачи BDD-минимизации. Кроме технологически независимой оптимизации, BDD стали использоваться и при технологическом отображении при синтезе структур FPGA [27] и синтезе схем из библиотечных элементов. В работе [28] рассматриваются задачи поиска на BDD дизъюнктивного, конъюнктивного и «сумма по модулю 2» разложений подфункций, а также задачи факторизации (выделения одинаковых подвыражений). В работе [29] предложено оптимизировать булевы сети, находя не только одинаковые подфункции при разложении Шеннона скобочного представления системы функций, а не матричной формы системы ДНФ булевых функций, но и одинаковые узлы.

Еще одним направлением применения BDD является схемная реализация BDD, при которой решаются проблемы сокращения энергопотребления [30, 31]. Обычно здесь учитывается вероятность появления единичных значений входных сигналов, что влияет на перестановку переменных при построении BDD [4]. Получение коэффициентов разложения Шеннона позволяет записать и другие виды разложения по одной переменной. Использование коэффициентов разложения по двум переменным и соответствующее представление, названное BBDD (Biconditional Binary Decision Diagrams, BBDD – диаграмма двоичного выбора с двумя условиями), изучаются в [18]. Поиск лучшей перестановки для схемной реализации зависит от формы представления исходной системы функций, поэтому различаются алгоритмы для матричных форм и алгоритмы для многоуровневых представлений, которые получаются записью уравнений по нетлистам схем. В этом случае говорят о поиске перестановки для описания «с учетом структуры схемы». Речь идет о том, что трудно сравнить на равенство многоуровневые представления коэффициентов, ведь для сравнения их подфункций целесообразно получать какую-либо каноническую форму, например, полином Жегалкина, как это сделано в [32].

Современные синтезаторы логических схем, например LeonardoSpectrum, имеют в своем составе BDD-оптимизацию (об этом кратко сообщается в технической документации), однако информация о реализованных методах и алгоритмах не приводится. Там же отмечается, что для схем умножителей

BDD не может быть построена. Однако в книге [1, с. 271] приводятся цифры (более 136 миллионов вершин) размерности BDD для умножителей, выполняющих перемножение чисел, представленных 16-разрядными двоичными кодами.

Основными методами нахождения лучшей перестановки, предложенными в литературе, являются методы ветвей и границ [17], методы генерирования случайных перестановок, дополненные различными эвристиками [4], и генетические алгоритмы, библиография по которым дана в [30].

Алгоритмы и программы минимизации BDD-представлений систем булевых функций

Описшем алгоритмы и реализующие их программы, строящие оптимизированные BDD-представления по исходным матричным формам систем ДНФ на языке SF – это известные программы BDD_OPT [4, с. 195], BDD_Energy [4, с. 206] и новая программа BDD_Builder.

Алгоритм и программа BDD_Builder

Эвристический алгоритм решения задачи 1, реализованный в программе BDD_Builder, является итерационным и выполняется до тех пор, пока текущая система функций (коэффициентов разложений) не выродится до констант 0, 1. На первой итерации в качестве текущей берется исходная система функций, для которой выполняется переход к полиномиальному представлению каждой функции в виде полинома Жегалкина [32].

На каждой итерации j (j = 1, …, n) выполняются следующие шаги.

Шаг 1. Выбор переменной x_j, по которой производится разложение Шеннона всех функций текущей системы.

1.1. Построение разложений Шеннона по каждой из переменных x_1, x_2, …, x_n каждой из ДНФ текущей системы функций системы, то есть получение ДНФ коэффициентов, не равных 0. Исключение из рассмотрения коэффициентов, равных 0.

1.2. Проверка полученных полиномов Жегалкина на равенство, в результате чего из всего множества ДНФ, представляющих коэффициенты, формируется множество M_i^j попарно различных подфункций – коэффициентов разложения.

1.3. Исключение из дальнейшего рассмотрения коэффициентов, равных константе 1.

1.4. Оценка переменных, от которых зависят функции текущей системы. Каждая переменная x_i текущей системы оценивается числом S_i^j – мощностью $\left|M_i^j\right| = S_i^j$ множества M_i^j различных коэффициентов, полученных при разложении всех функций текущей системы по переменной x_i.

1.5. В качестве переменной x_j для разложения Шеннона на итерации j выбирается переменная x_i, оцениваемая минимальным числом S_i^j. Если таких переменных несколько, выбирается первая по порядку. Переход на шаг 2.

Шаг 2. Построение разложений Шеннона текущей системы функций и формирование системы функций для итерации j + 1.

2.1. Построение разложений Шеннона функций текущей системы (на итерации j) по выбранной переменной x_i, то есть получение ДНФ коэффициентов, не равных 0. Исключение из рассмотрения коэффициентов, равных 0.

2.2. Проверка полученных подфункций на равенство и формирование множества M_i^j попарно различных коэффициентов.

2.4. Исключение из множества M_i^j (то есть из дальнейшего рассмотрения) коэффициентов, равных 1.

2.5. Проверка, является ли множество M_i^j пустым, то есть остались ли в нем коэффициенты, не равные константе 1. Если M_i^j оказалось пустым, осуществляется переход на шаг 3. В противном случае множество подфункций из M_i^j является текущей системой функций для итерации j + 1.

Шаг 3. Конец.

Программа BDD_Builder позволяет работать с внутренними данными (подфункциями), представленными как полиномами Жегалкина, так и в виде ДНФ, использует распараллеливание вычислений на ядра процессора как на трудоемком этапе 1.4 алгоритма, так и на начальном этапе получения полиномов Жегалкина по ДНФ каждой из функций.

Алгоритм и программа OPT_BDD

Алгоритм выбора перестановки аргументов подробно описан в [4] и состоит в последовательном применении трех эвристик. Каждая эвристика применяется итерационно до тех пор, пока очередная итерация не даст уменьшения сложно-

сти BDD. Для каждой из рассматриваемых перестановок подсчитывается сложность BDD.

Эвристика 1 (перестановка одного аргумента x_i с правым соседом). Исходной является начальная перестановка $<x_1, \ldots, x_n>$. Аргументы $x_1, \ldots, x_n$ рассматриваются поочередно. Сначала аргумент x_1 меняется позицией с правым аргументом x_2 (осуществляется транспозиция элементов перестановки), затем x_1 меняется позицией с правым соседом x_3 и т.д., пока x_1 не окажется на последнем месте. Среди пройденных выбирается лучшая перестановка (лучшей перестановке соответствует BDD меньшей сложности). Она является начальной для последующего движения аргумента x_2, при этом x_2 помещается на первую позицию и двигается путем перестановки с правым соседом на последнюю позицию и так далее для оставшихся аргументов $x_3, \ldots, x_n$. Поочередное движение всех аргументов составляет одну итерацию применения эвристики 1. Если итерация применения эвристики 1 уменьшает сложность BDD, то выполняется следующая итерация. Если же итерация не приводит к уменьшению сложности BDD, то осуществляется переход к применению эвристики 2.

Эвристика 2 (попарная перестановка). Берется лучшая перестановка., полученная в результате применения эвристики 1. Очередная рассматриваемая перестановка отличается от предыдущей переменой мест только двух аргументов, пока не будет найдена новая лучшая перестановка. Если найдена лучшая перестановка, процесс попарной перестановки повторяется уже для новой лучшей перестановки. Эвристика 2 прекращает работу, если после рассмотрения всех попарных перестановок сложность BDD не уменьшается.

Эвристика 3 (оконная перестановка). Исходной является лучшая перестановка X_{best2}, полученная в результате применения эвристики 2. Окно представляет собой четыре последовательно расположенных аргумента $\langle x_i, x_{i+1}, x_{i+2}, x_{i+3}\rangle$, внутри которого производятся все 4! = 24 перестановки четырех аргументов.

Затем окно перемещается на один аргумент вправо, получается новое окно $\langle x_{i+1}, x_{i+2}, x_{i+3}, x_{i+4}\rangle$, внутри которого опять осуществляются все перестановки. Сдвиг окна до последней позиции и перебор внутри него всех перестановок свидетельствуют об окончании одной итерации применения эвристики 3. Лучшая перестановка является начальной для следующей итерации применения эвристики 3.

Если итерация не приводит к уменьшению сложности BDD, полученная в результате применения эвристики 3 перестановка является результирующей, именно по ней и строится BDD.

Алгоритм и программа BDD_Energy

Алгоритм, реализованный в программе BDD_Energy, ориентирован на уменьшение энергопотребления логической схемы, построенной по минимизированному BDD-описанию, и основан на переборе случайных перестановок, для каждой из которых строится BDD и оценивается ее сложность S_{BDD}. При этом учитываются вероятности единичных значений сигналов входных переменных. Подробно алгоритм описан в [4]. Во всех экспериментах вероятность появления единичных значений входных переменных полагалась равной 0,5, чтобы оценить эффективность алгоритмов перебора случайных перестановок (без учета «энергетического качества» входных переменных) на сложность S_{BDD} и на сложность синтезируемых логических схем.

Задача минимизации сложности булевой сети

Рассматриваются булевы сети, представляющими функциями вершин которых могут быть бинарные логические операции И (*, конъюнкция), ИЛИ (+, ∨, дизъюнкция), а также унарная операция НЕ (^, инверсия). Сложность булевой сети оценивается суммарным числом бинарных логических операторов, содержащихся в ней. Инверсии переменных при оценке сложности булевой сети не принимаются во внимание. Данная оценка сложности булевой сети хорошо согласуется с такой известной в литературе оценкой сложности алгебраического представления булевой функции, как общее число литералов булевых переменных, содержащихся в нем [22].

Задача 2. Задано многоуровневое представление системы $f(x) = (f^1(x), \ldots, f^m(x))$, $x = (x_1, x_2, \ldots, x_n)$ полностью определенных булевых функций в виде взаимосвязанных логических формул. Каждая из формул имеет вид ДНФ. Требуется минимизировать сложность булевой сети, представляющей систему функций $f(x)$.

В работе [29] предложено проводить минимизацию булевых сетей (решать задачу 2) на основе разложения Шеннона и на нахождении одинаковых (и взаимно инверсных) подвыражений, реализуемых узлами булевой сети, для

чего была разработана соответствующая программа BoolNet_Opt. Применяя ее к формулам разложения Шеннона, соответствующим графу BDD, можно получить следующие формулы минимизированной булевой сети:

```
f1 = t7 + t8; t7 = x3*t9; t8 = ^x3*t11;
f2 = t12 + t13; t12 = x3*t14; t13 = ^x3*t16;
f3 = t17 + t18; t17 = x3*t19; t18 = ^x3*t21;
t9 = x2*t22; t14 = x2 + x4; t19 = x2*^x4;
t11 = t25 + ^t14; t25 = x2*t40; t16 = t30 + t31;
t30 = x2*x4; t31 = ^x2*t22; t21 = t35 + t31;
t35 = x2*t37; t22 = t40 + t41; t40 = x4*^x1;
t41 = ^x4*x1; t37 = t46 + t47; t46 = x4*x1;
t47 = ^x4*^x1;
```

Формулы содержат 9 операций дизъюнкции и 16 операций конъюнкции и получены для перестановки $<x_3, x_2, x_4, x_1>$ переменных. Булева сеть, соответствующая данным формулам, изображена на рисунке 2.

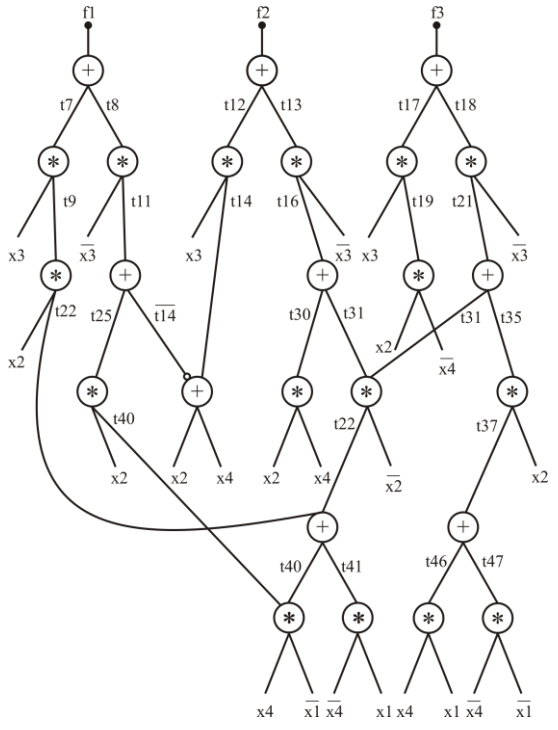

Рис. 2. Булева сеть, реализующая систему ДНФ булевых функций

Fig. 2. Boolean network implementing the DNF system of Boolean functions

При BDD-минимизации проводится поиск одинаковых подфункций – коэффициентов разложения Шеннона, а при минимизации булевых сетей – одинаковых подвыражений, соответствующих узлам сети. Заметим, что, применив программу BoolNet_Opt к исходной системе ДНФ, заданной в таблице 1, получаем булеву сеть другой структуры, но той же сложности. Данный пример демонстрирует, что число двухвходовых логических операций может быть сокращено путем применения дополнительной оптимизации к формулам, полученным в результате BDD-минимизации, что и было доказано в результате эксперимента на потоке примеров практической размерности.

Результаты экспериментов

Для оценки эффективности программ BDD-минимизации при синтезе схем из библиотечных КМОП-элементов были проведены вычислительные эксперименты.

Эксперимент 1. Сравнение эффективности программ BDD-оптимизации на потоке промышленных примеров.

Исходными описаниями блоков комбинационной логики являлись системы ДНФ булевых функций на языке SF [6], взятые из библиотеки примеров схем [33]. Для каждой из систем выполнялась BDD-минимизация с помощью трех программ – BDD_OPT, BDD_Energy, BDD_Builder. Программа BDD_Energy выполнялась для вероятностей 0,5 появления единичных значений для каждой из входных переменных системы ДНФ. Минимизированные описания конвертировались в VHDL-описания и подавались на вход синтезатора Leonardo-Spectrum. Для каждого из примеров синтез осуществлялся с одними и теми же опциями управления синтезом и для одной и той же целевой библиотеки синтеза. Целевой являлась библиотека проектирования заказных цифровых КМОП СБИС, состав библиотеки приведен в работе [4]. Этапы эксперимента 1 показаны на рисунке 3.

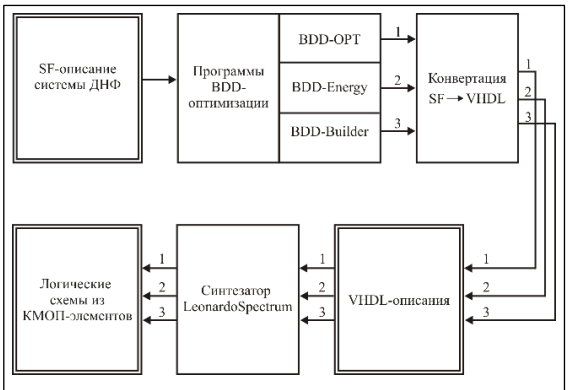

Рис. 3. Этапы эксперимента 1

Fig. 3. Phases of experiment 1

Результаты эксперимента 1 на потоке промышленных примеров приведены в таблицах 2 и 3 для программ BDD_Energy и BDD_Builder.

Таблица 2

Результаты исследования программы BDD_Energy (эксперимент 1)

Table 2

Analysis of BDD_Energy program (experiment 1)

Пример	n	m	k	α	S_{BDD}	V + &	t	S_{ASIC}	τ
Rd73	7	3	147	20 000	43	111	1 м 20 с	***15 925***	***3,94***
Dc2	8	7	58	20 000	62	123	2 м	22 889	3,33
Dist	8	5	256	20 000	146	374	1,4 с	***68 601***	5,51
M2	8	16	96	20 000	126	320	1 м 1 с	58 562	4,69
M3	8	16	128	20 000	141	338	1 м 54 с	59 929	***5,18***
Radd	8	5	120	20 000	29	63	1 м 43 с	***8 465***	4,44
Root	8	5	256	20 000	75	148	1 м 41 с	***26 717***	4,81
Z9sym	9	1	420	20 000	33	79	1 м 2 с	***15 909***	***4,77***
Tial	14	8	640	20 000	828	2 270	10 м 21 с	321 575	9,51
Intb	15	7	664	20 000	617	1 594	8 м 45 с	***229 433***	***8,27***
B2	16	17	110	20 000	560	1 352	5 м 57 с	***172 043***	9,02
Sin_16	16	16	65 536	20	17 761	51 678	4 м 5 с	***9 068 945***	***16,28***
Syst_4	17	12	370	20 000	296	761	10 м	129 618	6,77
Syst_8	25	20	45 548	20	29 368	87 762	5 м	14 453 628	21,50
				200	17 171	50 959	30 м	8 719 844	21,05
Vtx1	27	6	110	20 000	157	322	2 м 57 с	***25 015***	4,95
X9dn	27	6	120	20 000	198	418	3 м 43 с	25 897	5,73
Too_large_matr	38	3	1 027	2 000	1 817	4 958	9 м 53 с	631 304	***15,88***
X1_matr	51	35	274	20 000	464	1 101		92 879	***5,00***
Dalu_matr	75	16	194	20 000	253	724	40 м	***81 116***	7,64
Soar_matr	83	94	529	2 000	960	2 104	5 м 30 с	191 600	6,41
				20 000	868	1 865	1 ч 0 м 28 с	175 558	6,73
X3_matr	135	99	915	2 000	1 590	3 741	28 м 13 с	318 953	9,84
Frg2_matr	143	139	3 090	200	11 544	32 132	18 м	2 195 384	18,60

Для программы BDD_OPT задавалось фиксированное время – 10 минут, результаты выполнения программы BDD_OPT оказались неконкурентоспособными, поэтому не приводятся.

Примеры с окончанием «_matr» – это системы ДНФ, полученные из соответствующих исходных многоуровневых представлений систем функций; Sin_16 – это задание в виде таблицы истинности 16-битного приближения тригонометрической функции $y = \sin x$; примеры Syst4, Syst8 взяты из практики промышленного проектирования цифровых схем.

В таблицах 2 и 3 (и далее) n – число аргументов; m – число функций системы ДНФ, заданной на k общих элементарных конъюнкциях; S_{BDD} – сложность BDD; V + & – число операций конъюнкции и дизъюнкции в формулах разложений Шеннона; t – время выполнения программы на компьютере с процессором Intel(R) Core i5-2320 и тактовой частотой 3 GHz; S_{ASIC} – суммарная площадь элементов схемы в условных единицах; τ – задержка схемы (нс); α – число случайно генерируемых перестановок входных переменных для программы BDD_Energy. В таблице 3 символом «П» помечены три примера, оптимизированные программой BDD_Builder в режиме распараллеливания вычислений.

Эксперимент 2. Сравнение эффективности программ BDD-оптимизации на потоке псевдослучайных примеров.

Были проведены аналогичные эксперименты на потоке псевдослучайных примеров систем ДНФ, характеризуемых различными средними значениями числа конъюнкций, числа литералов в конъюнкциях и средних значений числа вхождений конъюнкций в ДНФ функций системы. Результаты эксперимента 2 приведены в таблицах 4 и 5, где n_1 – среднее число литералов в элементарной конъюнкции;

Таблица 3

Результаты исследования программы BDD_Builder (эксперимент 1)

Table 3

Analysis of BDD_Builder program (experiment 1)

Пример	n	m	k	S_{BDD}	V + &	t	S_{ASIC}	τ
Rd73	7	3	147	43	111	0,25 с	***15 925***	***3,94***
Dc2	8	7	58	63	124	1 с	***21 505***	3,99
Dist	8	5	256	146	374	1 с	69 560	5,09
M2	8	16	96	137	278	0,3 с	***47 536***	***4,25***
M3	8	16	128	153	321	0,24 с	***58 813***	5,20
Radd	8	5	120	35	87	1,02 с	10 117	***3,34***
Root	8	5	256	75	148	0,27 с	***26 717***	4,81
Z9sym	9	1	420	33	79	0,9 с	***15 909***	***4,77***
Tial	14	8	640	720	1 856	1,85 с	294 830	***7,71***
Intb	15	7	664	822	2 118	2,4 с	327 401	9,21
B2	16	17	110	773	13 153	2,4 с	183 638	***7,84***
Sin_16	16	16	65 536	17 761	51 679	47,6 с **П**	9 105 109	17,27
Syst_4	17	12	370	366	839	0,8 с	***115 791***	7,09
Syst_8	25	20	45 548	17 440	50 090	24,3 с **П**	***7 961 142***	***18,49***
Vtx1	27	6	110	103	199	1,2 с	26 354	6,58
X9dn	27	6	120	104	204	0,7 с	24 876	5,55
Too_large_matr	38	3	1 027	1 347	3 602	57 с	***511 759***	16,69
X1_matr	51	35	274	491	1 083	10,6 с	***79 565***	6,40
Dalu_matr	75	16	194	245	712	25,8 с	82 316	***7,17***
Soar	83	94	529	530	904	30 с	***135 298***	5,67
X3_matr	135	99	915	858	2 134	4 м 57 с	***242 406***	***7,91***
Frg2_matr	143	139	3 090	1 539	4 028	2 м 50 с **П**	***507 841***	***10,94***

Таблица 4

Результаты исследования программ BDD_Energy на псевдослучайных примерах (эксперимент 2)

Table 4

Analysis of BDD_Energy program on the random-looking example (experiment 2)

Пример	n	n_1	m	m_1	k	α	S_{BDD}	V + &	t	S_{ASIC}	τ
GenP_1	15	10	7	3	664	2 000	11 472	32 306	55 м 01 с	***4 843 250***	16,27
GenP_2	20	15	10	3	400	200	314 953	86 337	10 м 2 с	12 275 866	17,95
GenP_3	30	20	10	3	100	2 000	4 257	9 912	4 м 26 с	***1 086 621***	***11,66***
GenP_4	40	30	10	3	100	2 000	4 229	8 327	4 м 43 с	945 866	12,15
GenP_5	20	15	10	3	100	2 000	3 097	7 424	2 м 29 с	***915 706***	***11,12***
GenP_6	20	10	10	3	100	2 000	6 448	17 597	6 м 14 с	***2 486 699***	***13,87***
GenP_7	20	5	10	3	100	2 000	8 553	23 884	18 м 7 с	***3 720 610***	***16,04***
GenP_8	20	20	10	3	100	2 000	1 137	1 591	20 с	235 867	8,73

Таблица 5

Результаты исследования программ BDD_Builder на псевдослучайных примерах (эксперимент 2)

Table 5

Analysis of BDD_Builder program on the random-looking example (experiment 2)

Пример	n	n_1	m	m_1	k	S_{BDD}	V + &	t	S_{ASIC}	τ
GenP_1	15	10	7	3	664	12 319	34 919	27,5 с	5 009 769	15,89
GenP_2	20	15	10	3	400	29 175	79 634	39,6 с	10 647 047	16,92
GenP_3	30	20	10	3	100	4 667	11 098	8,7 с	1 224 894	13,60
GenP_4	40	30	10	3	100	4 361	8 601	8,7 с	939 923	12,08
GenP_5	20	15	10	3	100	3 931	9 768	4,6 с	1 260 494	12,51
GenP_6	20	10	10	3	100	6 929	18 964	10,9 с	2 672 558	14,92
GenP_7	20	5	10	3	100	10 996	30 670	48,7 с	4 640 769	18,18
GenP_8	20	20	10	3	100	1 161	1 617	0,9 с	233 785	7,37

m_1 – среднее число вхождений конъюнкции в ДНФ функций системы. Для первого псевдослучайного примера GenP_1 число n столбцов троичной матрицы T^x, задающей конъюнкции, равно 15, число k строк матриц T^x, B^f равно 664, среднее число n_1 определенных (0, 1) элемен-

тов в строке матрицы T^x равно 10, среднее число единичных значений в строке матрицы B^f равно 3.

Эксперимент 3. Изучение возможности улучшения результатов BDD-оптимизации с помощью оптимизации булевых сетей.

После проведения синтеза в экспериментах 1 и 2 выбирались те BDD-описания, которые приводили к схемам меньшей площади, и для этих BDD-описаний выполнялась программа BoolNet_Opt оптимизации булевых сетей, после чего осуществлялся повторный синтез схем. Результаты этого эксперимента приведены в таблице 6, где символом «*» помечены несколько лучших решений, полученных программой BDD_OPT.

Таблица 6

Улучшение BDD-решений программой BoolNet_Opt минимизации булевых сетей (эксперимент 3)

Table 6

Improvement of BDD solutions by the Boolean network minimization program BoolNet_Opt (experiment 3)

Пример	**Синтез по исходным описаниям**		**Лучшее схемное решение, полученное программами BDD-минимизации BDD_OPT, BDD_Energy, BDD_Builder**		**Улучшение решений BDD-минимизации программой BoolNet_Opt**	
	S_{ASIC}	τ	**Наименьшая площадь, S_{ASIC}**	**Наименьшая задержка, τ**	S_{ASIC}	τ
Rd73	21 684	***3,67***	15 925	3,94	16 428	3,89
Dc2	26 745	4,17	***21 505***	***3,33***	22 136	3,49
Dist	83 136	7,03	68 601	5,09*	***64 549***	***5,06***
M2	62 161	7,31	47 536	4,25	***45 583***	4,76
M3	82 227	7,09	58 813	5,18	***52 982***	5,73
Radd	12 092	2,89	8 465	3,34	***7 399***	3,99
Root	55 750	6,47	26 717	4,53*	***26 538***	***4,37***
Z9sym	59 896	9,90	15 909	4,77	***15 367***	4,95
Tial	303 100	8,01	260 636*	***7,71***	294 858	9,80
Intb	382 844	9,26	***229 433***	8,27	297 888	***3,99***
B2	***159 834***	11,36	172 043	***7,84***	162 350	8,52
Sin_16	-	-	9 068 945*	***16,28****	***9 049 560***	16,30
Syst_4	185 206	7,64	115 791	***6,74****	125 879	7,17
Syst_8	-	-	8 719 844	***21,05***	***7 934 408***	17,26
Vtx1	***18 386***	4,08	25 015	4,37*	33 407	6,26
X9dn	***19 195***	5,24	24 312*	5,00	20 473	5,75
Too_large_matr	429 459	12,98	511 759	15,88	***416 575***	***12,55***
X1_matr	***67 312***	3,99	79 565	5,00	98 917	5,70
Dalu_matr	***49 673***	4,22	81 116	7,17	51 715	4,44
Soar	150 995	5,94	135 298	***5,67***	152 992	6,39
X3_matr	206 957	***6,69***	242 406	***7,91***	***205 428***	6,97
Frg2_matr	***406 062***	10,83	507 841	10,94	507 841	10,94
Псевдослучайные примеры						
GenP_1	***978 905***	***11,73***	4 843 250	15,89	5 038 377	14,84
GenP_2	***801 857***	***10,43***	10 647 047	16,92	10 697 379	18,08
GenP_3	***278 130***	***7,24***	1 086 621	11,66	1 126 016	11,90
GenP_4	***378 597***	***7,60***	939 923	12,08	854 638	12,88
GenP_5	***218 786***	***7,53***	915 706	11,12	945 883	12,40
GenP_6	***190 434***	***6,53***	2 486 699	13,87	2 514 460	13,87
GenP_7	***126 499***	***5,88***	3 720 610	16,04	3 673 303	16,55
GenP_8	***235 671***	***7,32***	233 785	7,37	240 509	7,96

Таблица 7

Использование минимизации систем функций в классе ДНФ (эксперимент 4)

Table 7

Using minimization of function systems in the DNF class (experiment 4)

Пример	Синтез по исходным описаниям		Синтез после раздельной минимизации функций		Синтез после совместной минимизации функций	
	S_{ASIC}	τ	S_{ASIC}	τ	S_{ASIC}	τ
GenP_1	978 905	11,73	1 001 108	***9,83***	***776 636***	11,33
GenP_2	***801 857***	10,43	818 971	10,86	803 230	***9,55***
GenP_3	278 130	7,24	279 530	7,14	***277 929***	***6,92***
GenP_4	***378 597***	7,60	381 577	7,48	385 316	***7,37***
GenP_5	***218 786***	7,53	261 454	***6,95***	219 244	7,43
GenP_6	190 434	***6,53***	***189 463***	7,17	190 468	6,67
GenP_7	126 499	5,88	119 635	***5,28***	***116 477***	6,45
GenP_8	235 671	***7,32***	***233 010***	7,42	235 286	8,11

Для псевдослучайных примеров значительно лучшие результаты синтеза показывает синтезатор LeonardoSpectrum при использовании своих встроенных программ технологически независимой оптимизации. BDD-минимизация для псевдослучайных примеров неэффективна, если синтез вести в LeonardoSpectrum.

Эксперимент 4. Исследование эффективности раздельной и совместной минимизации в классе ДНФ для псевдослучайных примеров [21, 22].

Результаты эксперимента 4, представленные в таблице 7, показывают, что для псевдослучайных примеров более эффективна минимизация в классе ДНФ (раздельная либо совместная) при синтезе схем из библиотечных элементов.

Эксперимент 5. Сравнение сложности решений, полученных программой BDD_Builder, с известными решениями из [25].

Результаты эксперимента 5 на потоке промышленных примеров приведены в таблице 8. Они свидетельствуют о том, что программа BDD_Builder конкурентоспособна с программой из [25].

Проведенные экспериментальные исследования показали, что в практике проектирования целесообразно использовать две взаимно дополняющие друг друга программы – BDD_Energy и BDD_Builder. Программа BDD_Builder является быстродействующей, позволяет распараллеливать вычисления и за короткое время получать приемлемое решение. Если задать большое число случайных перестановок, то не исключено, что программа BDD_Energy может получить лучшее решение, чем программа BDD_Builder, однако для этого может понадобиться много времени, так как время вычислений для программы BDD_Energy определяется размерностью задачи и числом заданных (генерируемых случайным образом) перестановок. Дополнительное улучшение решений (уменьшение площади и задержки схемы) достаточно часто можно получить, применяя программу BoolNet_Opt к минимизированным BDD-описаниям систем функций.

Таблица 8

Результаты сравнения программ оптимизации BDD (эксперимент 5)

Table 8

The comparison results of BDD optimization programs (experiment 5)

Пример	*n*	*m*	*k*	S_{BDD}	
				Программа из работы [25]	Программа BDD_Builder
ADD6	12	7	1 092	***47***	56
ADR4	8	5	256	***29***	***29***
ALU1	12	8	19	20	***18***
DC1	4	7	15	23	24
DC2	8	7	58	64	***63***
DIST	8	5	256	152	***146***
F51m	8	8	256	67	***64***
MLP4	8	8	256	***141***	166
RADD	8	5	120	***29***	33
RD53	5	3	32	***23***	***23***
RD73	7	3	147	***43***	***43***
ROOT	8	5	256	***75***	***75***
SEX	9	14	23	***47***	59
SQN	7	3	96	53	***50***
Sqr6	12	8	152	***71***	***71***
Z4	14	8	640	***26***	***26***
Z5XP1	7	10	128	68	***67***
Z9SYM	9	1	420	***33***	***33***
Число лучших решений				11	13

Заключение

Разработанные программы технологически независимой оптимизации являются эффективными, прошли экспериментальную проверку на примерах схем практической размерности и включены в отечественные системы автоматизированного проектирования. Использование мощных отечественных программ логической оптимизации позволяет улучшить результаты синтеза логических КМОП-схем в промышленном синтезаторе LeonardoSpectrum.

Литература

1. Кнут Д.Э. Комбинаторные алгоритмы; в кн. Искусство программирования; [пер. с англ.]. М.: Вильямс, 2013. Т. 4А. Ч. 1. 960 с.

2. Карпов Ю.Г. Model Checking. Верификация параллельных и распределенных программных систем. СПб: БХВ-Петербург, 2010. 560 с.

3. Чэнь М., Цинь К., Ку Х.-М., Мишра П. Валидация на системном уровне. Высокоуровневое моделирование и управление тестированием; [пер. с англ.]. М.: Техносфера, 2014. 296 с. DOI: 10.1007/978-1-4614-1359-2.

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

5. Бибило П.Н., Авдеев Н.А., Кардаш С.Н., Кириенко Н.А., Ланкевич Ю.Ю., Логинова И.П., Романов В.И., Черемисинов Д.И., Черемисинова Л.Д. Система логического проектирования функциональных блоков заказных КМОП СБИС с пониженным энергопотреблением // Микроэлектроника. 2018. Т. 47. № 1. С. 72–88. DOI: 10.7868/S0544126918010076.

6. Бибило П.Н., Романов В.И. Логическое проектирование дискретных устройств с использованием продукционно-фреймовой модели представления знаний. Минск: Беларус. навука, 2011. 280 с.

7. Biere A., Heule M., van Maaren H., Walsh T., eds. Frontiers in Artificifl Intelligents and Applications. Handbook of Satisfiability. IOS Press, 2009, 980 p. DOI: 10.3233/978-1-58603-929-5-3.

8. Drechsler R., Becker B. Binary Decision Diagrams: Theory and Implementation. Springer, 1998, 210 p.

9. Bryant R.E., Meinel C. Ordered Binary Decision Diagrams. Logic Synthesis and Verification. Kluwer Acad. Publ., 2002, pp. 285–307. DOI: 10.1007/978-1-4615-0817-5_11.

10. Meinel C., Theobald T. Algorithms and Data Structures in VLSI Design: OBDD – Foundations and Applications. Springer, 1998, 267 p. DOI: 10.1016/s0898-1221(99)90407-8.

11. Lee C.Y. Representation of switching circuits by binary-decision programs. Bell Systems Technology J., 1959, vol. 38, pp. 985–999.

12. Блох А.Ш. Граф-схемы и их применение. Минск: Вышэйш. школа, 1975. 302 с.

13. Кузнецов О.П. О программной реализации логических функций и автоматов. Ч. I. Анализ и синтез бинарных программ // Автоматика и телемеханика. 1977. № 7. С. 163–174.

14. Бутов А.А. Синтез многовыходных комбинационных схем с использованием мультиплексоров // Кибернетика. 1980. № 3. С. 63–70.

15. Akers S.B. Binary decision diagrams. IEEE Transactions on Computer, 1978, vol. 27, no. 6, pp. 509–516. DOI: 10.1109/TC.1978.1675141.

16. Bryant R.E. Graph-based algorithms for boolean functions manipulation. IEEE Transactions on Computer, 1986, vol. 35, no. 8, pp. 677–691. DOI: 10.1109/TC.1986.1676819.

17. Ebendt R., Fey G., Drechsler R. Advanced BDD Optimization. Springer, 2005, 222 p. DOI: 10.1007/b107399.

18. Amaru L.G. New Data Structures and Algorithms for Logic Synthesis and Verification. Springer, 2017, 156 p. DOI: 10.1007/978-3-319-43174-1_5.

19. Игнатьев А.С., Семенов А.А. Алгоритмы работы с ROBDD как с базами булевых ограничений // Прикладная дискретная математика. 2010. Т. 7. № 1. С. 86–104.

20. Brayton K.R., Hactel G.D., McMullen C.T., Sangiovanni-Vincentelli A.L. Logic Minimization Algorithm for VLSI Synthesis. Kluwer Academic Publishers, 1984, 194 p.

21. Торопов Н.Р. Минимизация систем булевых функций в классе ДНФ // Логическое проектирование: сб. науч. тр. 1999. Вып. 4. С. 4–19.

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

23. Rudell R. Dynamic variable ordering for ordered binary decision diagrams. Proc. IEEE/ACM ICCAD, Santa Clara, 1993, pp. 42–47. DOI: 10.1109/ICCAD.1993.580029.

24. Ebendt R., Gunther W., Drechsler R. An improved branch and bound algorithm for exact BDD

minimization. Computer-Aided Design of Integrated Circuits and System, 2003, vol. 22, no. 12, pp. 1657–1663. DOI: 10.1109/TCAD.2003.819427.

25. Ishiura N., Sawada H., Yajima S. Minimization of binary decision diagrams based on exchanges of variables. Proc. IEEE Int. Conf. on Computer-Aided Design, 1991, pp. 472–475. DOI: 10.1109/ICCAD.1991.185307.

26. Bollig B., Lobbing M., Wegener I. Simulated annealing to improve variable orderings for OBDDs. Proc. IWLS, pp. 1–10.

27. Chang S.-C., Marek-Sadowska M., Hwang T. Technology mapping for TLU FPGA's based on decomposition of binary decision diagrams. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1996, vol. 15, no. 10, pp. 1226–1235. DOI: 10.1109/43.541442.

28. Yang C., Ciesielski M. BDS: A BDD-based logic optimization system. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2002, vol. 7, no. 21, pp. 866–876. DOI: 10.1109/TCAD.2002.1013899.

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

30. Dutta A., Baishnab K.L., Chaudhary S. A new evolutionary algorithm based BDD optimization for area and power. IJEEE, 2010, vol. 3, no. 3, pp. 147–160.

31. Lindgren P., Kerttu M., Thornton M., Drechsler R. Low power optimization technique for BDD mapped circuits. Proc. Conf. ASPDAC-01, 2001, pp. 615–621. DOI: 10.1109/ASPDAC.2001.913377.

32. Бибило П.Н., Ланкевич Ю.Ю. Использование полиномов Жегалкина при минимизации многоуровневых представлений систем булевых функций на основе разложения Шеннона // Программная инженерия. 2017. Т. 8. № 8. С. 369–384. DOI: 10.17587/prin.8.369-384.

33. The Tests in the Monograph «Logic Minimization Algorithms for VLSI Synthesis». URL: http://www1.cs.columbia.edu/~cs6861/sis/espresso-examples/ex (дата обращения: 20.01.2020).

Software & Systems Received 22.01.20
DOI: 10.15827/0236-235X.131.449-463 2020, vol. 33, no. 3, pp. 449–463

Experimental investigation of effectiveness of algorithms for minimizing BDD representations of Boolean function systems

P.N. Bibilo [1], *Dr.Sc. (Engineering), Professor, Head of Laboratory, bibilo@newman.bas-net.by*
Yu.Yu. Lankevich [1], *Junior Researche, yurafreedom18@gmail.com*

[1] *United Institute of Informatics Problems of the National Academy of Sciences of Belarus (UIIP NASB), Minsk, 220012, Belarus*

Abstract. The mathematical apparatus of BDD is used in various fields of science. In the computer-aided design field, BDD allowed to obtain significant success in a formal verification of algorithmic descriptions of digital circuits. Design systems of digital VLSI use programs of BDD minimization at the stage of technologically independent optimization. Many articles consider optimization of BDD representations of systems of completely defined Boolean functions. Main attention was paid to finding an arrangement of variables for minimizing the BDD complexity. The variable arrangement is used to decompose the initial functions of the system and sub-functions (cofactors), which are obtained in the process of decomposition. The complexity of a BDD is the number of nodes in it. Each node of the BDD corresponds to a complete or reduced form of Shannon expansion.

Domestic CAD and logic optimization systems use several programs for minimization of BDD representation of Boolean function systems that implement various algorithms. The purpose of this paper is to study the efficiency of these programs for synthesis of combinational circuits from library CMOS elements. After obtaining BDD minimized as for the number of graph nodes and defined as a set of interconnected formulas of Shannon expansion, the synthesis of a logic circuit is performed in the same design library of digital CMOS VLSI; the results are compared by square and delay. In many cases, it is possible to achieve additional reduction of logic description complexity by performing additional logic minimization based on Boolean nets. In this case, the optimization criterion is the number of nodes in the Boolean net, without considering inversion of Boolean variables. It is agreed with "the number of literals" criterion in optimization of multi-level logic circuits. The results of experiments on standard examples are presented.

Keywords: the system of Boolean functions, disjunctive normal form, Binary Decision Diagram (BDD), digital logic synthesis, VHDL, VLSI, CMOS.

References

1. Knuth D.E. Combinatorial Algorithms. In: *The Art of Computer Programming*, Pearson Education, Inc., 2011, vol. 4A, 883 p. (Russ. ed.: Moskow, 2013, 960 p.).

2. Karpov Yu.G. *Model Checking. Verification of Parallel and Distributed Software Systems*. St. Petersburg, 2010, 560 p. (in Russ.).

3. Chen M., Qin X., Koo H.-M., Mishra P. *System-Level Validation. High-Level Modeling and Directed Test Generation Techniques*. Springer, 2013, 250 p. (Russ. ed.: Moskow, 2014. 296 p.). DOI: 10.1007/978-1-4614-1359-2.

4. Bibilo P.N. *Applying Binary Selection Diagrams in the Logic Circuit Synthesis*. Minsk, 2014, 231 p. (in Russ.).

5. Bibilo P.N., Avdeev N.A., Kardash S.N., Kirienko N.A., Lankevich Yu.Yu., Loginova I.P., Romanov V.I., Cheremisinov D.I., Cheremisinova L.D. A System for logical design of custom CMOS VLSI functional blocks with reduced power consumption. *Microelectronics*, 2018, vol. 47, no. 1, pp. 72–88 (in Russ.). DOI: 10.7868/S0544126918010076.

6. Bibilo P.N., Romanov V.I. *Logical Design of Discrete Devices Using the Production-Frame Knowledge Representation Model*. Minsk, 2011, 280 p. (in Russ.).

7. Biere A., Heule M., van Maaren H., Walsh T., eds. *Frontiers in Artificifl Intelligents and Applications. Handbook of Satisfiability*. IOS Press, 2009, 980 p. DOI: 10.3233/978-1-58603-929-5-3.

8. Drechsler R., Becker B. *Binary Decision Diagrams: Theory and Implementation*. Springer, 1998, 210 p.

9. Bryant R.E., Meinel C. *Ordered Binary Decision Diagrams. Logic Synthesis and Verification*. Kluwer Acad. Publ., 2002, pp. 285–307. DOI: 10.1007/978-1-4615-0817-5_11.

10. Meinel C., Theobald T. *Algorithms and Data Structures in VLSI Design: OBDD – Foundations and Applications*. Springer, 1998, 267 p. DOI: 10.1016/s0898-1221(99)90407-8.

11. Lee C.Y. Representation of switching circuits by binary-decision programs. *Bell Systems Technology J.*, 1959, vol. 38, pp. 985–999.

12. Blokh A.Sh. *Graph Diagrams and their Application*. Minsk, 1975, 302 p. (in Russ.).

13. Kuznetsov O.P. On implementation of logical functions and automata by programs. Pt. I. Analysis and synthesis of binary-decision program. *Autom. Remote Control*, 1977, vol. 38, no. 7, pp. 1077–1087 (in Russ.).

14. Butov A.A. Synthesis of multi-output combinational circuits using multiplexers. *Kibernetika*, 1980, no. 3, pp. 63–70 (in Russ.).

15. Akers S.B. Binary decision diagrams. *IEEE Transactions on Computer*, 1978, vol. 27, no. 6, pp. 509–516. DOI: 10.1109/TC.1978.1675141.

16. Bryant R.E. Graph-based algorithms for boolean functions manipulation. *IEEE Transactions on Computer*, 1986, vol. 35, no. 8, pp. 677–691. DOI: 10.1109/TC.1986.1676819.

17. Ebendt R., Fey G., Drechsler R. *Advanced BDD Optimization*. Springer, 2005, 222 p. DOI: 10.1007/b107399.

18. Amaru L.G. *New Data Structures and Algorithms for Logic Synthesis and Verification*. Springer, 2017, 156 p. DOI: 10.1007/978-3-319-43174-1_5.

19. Ignatev A.S., Semenov A.A. The algorithms for working with ROBDD as Boolean constraint databases. *Prikladnaya Diskretnaya Matematika*, 2010, vol. 7, no. 1, pp. 86–104 (in Russ.).

20. Brayton K.R., Hactel G.D., McMullen C.T., Sangiovanni-Vincentelli A.L. *Logic Minimization Algorithm for VLSI Synthesis*. Kluwer Acad. Publ., 1984, 194 p.

21. Toropov N.R. Minimization of Boolean function systems in the DNF class. *Logicheskoe Proektirovanie Proc.*, 1999, iss. 4, pp. 4–19 (in Russ.).

22. Brayton R.K., Hachtel G.D. Multilevel logic synthesis. *Proc. IEEE*, 1990, vol. 78, no. 2, pp. 264–300. DOI: 10.1109/5.52213.

23. Rudell R. Dynamic variable ordering for ordered binary decision diagrams. *Proc. IEEE/ACM ICCAD*, 1993, pp. 42–47. DOI: 10.1109/ICCAD.1993.580029.

24. Ebendt R., Gunther W., Drechsler R. An improved branch and bound algorithm for exact BDD minimization. *Computer-Aided Design of Integrated Circuits and Systems*, 2003, vol. 22, no. 12, pp. 1657–1663. DOI: 10.1109/TCAD.2003.819427.

25. Ishiura N., Sawada H., Yajima S. Minimization of binary decision diagrams based on exchanges of variables. *Proc. IEEE Int. Conf. on Computer-Aided Design*, 1991, pp. 472–475. DOI: 10.1109/ICCAD.1991.185307.

26. Bollig B., Lobbing M., Wegener I. Simulated annealing to improve variable orderings for OBDDs. *Proc. IWLS*, 1995, pp. 1–10.

27. Chang S.-C., Marek-Sadowska M., Hwang T. Technology mapping for TLU FPGA's based on decomposition of binary decision diagrams. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1996, vol. 15, no. 10, pp. 1226–1235. DOI: 10.1109/43.541442.

28. Yang C., Ciesielski M. BDS: a BDD-based logic optimization system. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2002, vol. 7, no. 21, pp. 866–876. DOI: 10.1109/TCAD.2002.1013899.

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

30. Dutta A., Baishnab K.L., Chaudhary S. A new evolutionary algorithm based BDD optimization for area and power. *IJEEE*, 2010, vol. 3, no. 3, pp. 147–160.

31. Lindgren P., Kerttu M., Thornton M., Drechsler R. Low power optimization technique for BDD mapped circuits. *Proc. Conf. ASPDAC-01*, 2001, pp. 615–621. DOI: 10.1109/ASPDAC.2001.913377.

32. Bibilo P.N., Lankevich Yu.Yu. The use of Zhegalkin polynomials for minimization of multilevel representations of Boolean functions based on Shannon expansion. *Software Engineerin*, 2017, vol. 8, no. 8, pp. 369–384 (in Russ.). DOI: 10.17587/prin.8.369-384.

33. *The Tests in the Monograph «Logic Minimization Algorithms for VLSI Synthesis»*. Available at: http://www1.cs.columbia.edu/~cs6861/sis/espresso-examples/ex (accessed January 20, 2020).

Для цитирования

Бибило П.Н., Ланкевич Ю.Ю. Экспериментальное сравнение эффективности алгоритмов оптимизации BDD-представлений систем булевых функций // Программные продукты и системы. 2020. Т. 33. № 3. С. 449–463. DOI: 10.15827/0236-235X.131.449-463.

For citation

Bibilo P.N., Lankevich Yu.Yu. Experimental investigation of effectiveness of algorithms for minimizing BDD representations of Boolean function systems. *Software & Systems*, 2020, vol. 33, no. 3, pp. 449–463 (in Russ.). DOI: 10.15827/0236-235X.131.449-463.