УДК 517.98 DOI: 10.31114/2078-7707-2020-1-33-39

Экспериментальное исследование эффективности программ минимизации BDD-представлений систем булевых функций при синтезе комбинационных схем из библиотечных КМОП элементов

П.Н. Бибило, Ю.Ю. Ланкевич

Объединенный институт проблем информатики НАН Беларуси,

bibilo@newman.bas-net.by, yurafreedom18@gmail.com

***Аннотация* — 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. 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 СБИС, результаты сравниваются по площади кристалла и по быстродействию (временной задержке). Дополнительного сокращения сложности логических описаний и улучшения результатов синтеза схем можно добиться во многих случаях, выполняя дополнительную логическую минимизацию на основе булевых сетей.**

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

I. Введение

Математический аппарат BDD в настоящее время используется в различных областях науки [1]. В области автоматизации проектирования цифровых систем применение BDD позволило получить значительные успехи в формальной верификации алгоритмических описаний цифровых систем [2, 3]. Проблематика BDD связывается в настоящее время также с решением SAT-проблем [4, 5]. Однако использование BDD в проектировании цифровых систем не ограничивается только верификацией. Системы проектирования цифровых СБИС используют программы минимизации BDD на этапе технологически независимой оптимизации [6]. Оптимизации BDD-представлений систем полностью определенных булевых функций посвящено много работ, их краткие обзоры содержатся в [2, 6, 7]. Основное внимание уделялось поиску перестановки переменных, по которой производится разложение функций исходной системы и подфункций (cofactors), получаемых в процессе разложения с целью минимизации сложности BDD, под которой понимается число вершин графа BDD, а каждой вершине соответствует полная либо редуцированная формула разложения Шеннона. Известны также подходы [8, 9], 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, который также часто называют целевой библиотекой логического синтеза.

В отечественных САПР [10] и системах логической оптимизации используются несколько программ минимизации BDD-представлений систем булевых функций, реализующих различные алгоритмы. Целью данной работы является изучение эффективности применения этих программ при синтезе комбинационных схем из библиотечных КМОП-элементов.

II. Задача BDD-минимизации

BDD-минимизация булевых функций основана на разложении Шеннона. *Разложением Шеннона* булевой функции $f(\boldsymbol{x}) = f(x_1,\ldots,x_n)$ по переменной x_i называется представление $f(\boldsymbol{x})$ в виде

$$f(\boldsymbol{x}) = \bar{x}_i f_0 \vee x_i f_1 . \qquad (1)$$

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 последнем шаге разложения коэффициенты вырождаются до констант 0, 1. На каждом шаге разложения выполняется сравнение на равенство полученных коэффициентов и из множества равных коэффициентов оставляется один. Под диаграммой двоичного выбора (BDD – Binary Decision Diagram) 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 $\boldsymbol{f}(\boldsymbol{x}) = (f^1(\boldsymbol{x}), \ldots, f^m(\boldsymbol{x}))$ будут рассматриваться BDD, которые построены по *общей* для всех компонентных функций $f^i(\boldsymbol{x})$ перестановке переменных. В результате BDD-оптимизации матричная форма исходной системы функций заменяется графом BDD, каждой вершине которого соответствует полная либо сокращенная (редуцированная) формула (1) разложения Шеннона. Пример графа BDD, построенного для матричного задания системы дизъюнктивных нормальных форм (ДНФ)

$$f^1 = x_1x_2x_3\bar{x}_4 \vee \bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2x_4;$$
$$f^2 = \bar{x}_1\bar{x}_2\bar{x}_3x_4 \vee x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1x_2\bar{x}_3x_4 \vee \bar{x}_1x_2x_4 \vee x_3x_4 \vee x_2x_3;$$
$$f^3 = \bar{x}_1\bar{x}_2\bar{x}_3x_4 \vee x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1x_2\bar{x}_3x_4 \vee x_1x_2x_3\bar{x}_4 \vee \bar{x}_1x_2\bar{x}_4$$

булевых функций (n=4, m=3), приведен на рис. 1. Матричная форма $<T^x, B^f>$ (табл. 1) 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.

Таблица 1

Пример системы ДНФ трех булевых функций

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

Данному графу BDD, построенному по перестановке $< x_3, x_2, x_1, x_4 >$ переменных, соответствуют 12 формул разложения Шеннона (^ обозначает инверсию, * - конъюнкцию, + - дизъюнкцию в языке SF [10]):

```
f1=^x3*sf0+x3*sf1; f2=^x3*sf2+x3*sf3;
f3=^x3*sf4+x3*sf5; sf0=^x2*^x4+x2*sf9;
sf4=^x2*sf6+x2*sf14; sf2=^x2*sf6+x2*x4;
sf3=^x2*x4+x2; sf5=x2*^x4;
sf1=x2*sf6; sf14=^x1*^x4+x1*x4;
sf9=^x1*x4; sf6=^x1*x4+x1*^x4.
```

Данные формулы содержат 9 операций дизъюнкции и 20 операций конъюнкции. Функциональные вершины нижнего (неконстантного) уровня BDD равны переменной либо ее инверсии, в примере это функциональные вершины x_4, $\bar{x}_4$. В виде вырожденных уравнений такие функции не будут записываться, поэтому число уравнений многоуровневого представления обычно меньше сложности BDD на одну либо две функциональные вершины. Под *сложностью (размером) BDD* обычно понимается в литературе число функциональных вершин BDD.

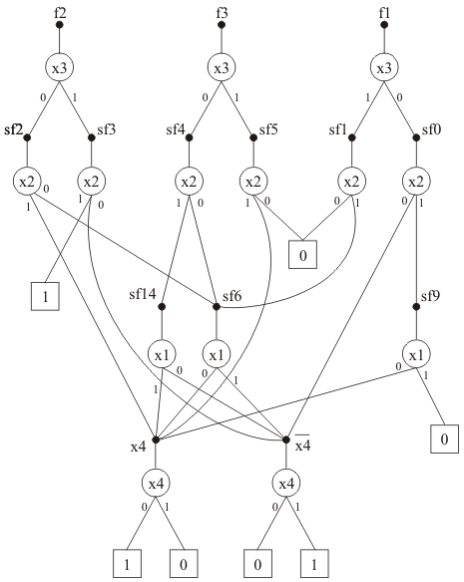

Рис. 1. Граф BDD

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, т. е. ROBDD (ROBDD - Reduced Ordered BDD). Подробное описание ROBDD дано в [1]. Далее под BDD будут всегда пониматься ROBDD.

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

Основными предложенными в литературе методами нахождения лучшей перестановки являются методы «ветвей и границ» [4, 11], методы генерирования

случайных перестановок, дополненные различными эвристиками [6], и генетические алгоритмы.

III. Алгоритмы и программы минимизации BDD-представлений систем булевых функций

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. Выполняется переход к полиномиальному представлению каждой функции в виде полинома Жегалкина.

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, т. е. 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, то из них 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, т. е. 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 (т. е. из дальнейшего рассмотрения) коэффициентов, равных 1.

2.5. Проверка, является ли множество M_i^j пустым, т. е. остались ли в множестве M_i^j такие коэффициенты, которые не равны константе 1. Если M_i^j оказалось пустым, то 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. Две другие программы BDD-минимизации BDD_OPT [6, с. 195], BDD_Energy [6, с. 206], как и программа BDD_Builder, обрабатывают исходные матричные описания систем ДНФ, представленные на языке SF – внутреннем языке САПР [10].

IV. 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. Такая оценка сложности булевой сети хорошо согласуется с такой известной в литературе [12] оценкой сложности алгебраического представления булевой функции, как общее число литералов булевых переменных, содержащихся в нем.

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

В работе [13] 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 данную программу 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; T11=T25+^T14; T25=x2*T40;
T14=x2+x4; T19=x2*^x4; 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. При BDD-минимизации проводится поиск одинаковых подфункций - коэффициентов разложения Шеннона, а при минимизации булевых сетей – поиск одинаковых подвыражений, соответствующих узлам сети. Заметим, что при применении программы BoolNet_Opt к исходной системе ДНФ, заданной в табл. 1, получаем булеву сеть другого вида, но той же сложности. Данный пример демонстрирует тот факт, что число двухвходовых логических операций может быть сокращено применением дополнительной оптимизации к формулам, полученным в результате BDD-минимизации, что и было доказано в результате эксперимента на потоке примеров практической размерности.

V. Результаты экспериментов

Для оценки эффективности программ BDD-минимизации при синтезе схем из библиотечных КМОП элементов были проведены вычислительные эксперименты. Исходные описания – системы ДНФ булевых функций представлялись на языке SF [10], 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 LeonardoSpectrum. 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 [6]. Результаты синтеза на потоке промышленных примеров из [14] приведены в табл. 2, 3 для программ BDD_Energy, BDD_Builder. 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. Для программы BDD_OPT задавалось фиксированное время – 10 мин., результаты выполнения программы BDD_OPT оказались неконкурентоспособными, поэтому не приводятся.

В таблицах 2 – 4: n – число аргументов; m – число функций системы ДНФ, заданной на k общих элементарных конъюнкциях; S_{BDD} - сложность BDD; $V+\&$ - суммарное 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 табл. 3 буквой «П» помечены три примера, оптимизированные программой BDD_Builder в режиме распараллеливания вычислений. После проведения синтеза 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 синтез схем. Результаты этого эксперимента приведены в табл. 4, где символом «*» помечается решение, полученное программой BDD_OPT. 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 за короткое время получать приемлемое решение. Если же задать большое число случайных перестановок, то не исключено, что программа BDD_Energy получит лучшее решение, чем BDD_Builder. Дополнительное улучшение решений (уменьшение площади и задержки схемы) достаточно часто можно получить, применяя программу BoolNet_Opt к минимизированным BDD-описаниям систем функций.

VI. Заключение

Разработанные программы технологически независимой оптимизации являются эффективными, прошли экспериментальную проверку на примерах схем практической размерности и включены в отечественные системы автоматизированного проектирования. Их использование позволяет во многих случаях улучшать результаты синтеза логических КМОП схем в промышленном синтезаторе LeonardoSpectrum.

Поддержка

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

Таблица 2

Результаты исследования программы Tie_Energy

Пример	*n*	*m*	*k*	α	S_{BDD}	*V+&*	*t*	S_{ASIC}	τ
Rd73	7	3	147	20 000	43	111	1м20с	***15 925***	***3.94***
Dc2	8	7	58	20 000	62	123	2м	22 889	3.33
Dist	8	5	256	20 000	146	374	1,4с	***68 601***	5.51
M2	8	16	96	20 000	126	320	1м1с	58 562	4.69
M3	8	16	128	20 000	141	338	1м54с	59 929	***5.18***
Radd	8	5	120	20 000	29	63	1м43с	***8 465***	4.44
Root	8	5	256	20 000	75	148	1м41с	***26 717***	4.81
Z9sym	9	1	420	20 000	33	79	1м2с	***15 909***	***4.77***
Tial	14	8	640	20 000	828	2270	10м21с	321 575	9.51
Intb	15	7	664	20 000	617	1594	8м45с	***229 433***	***8.27***
B2	16	17	110	20 000	560	1352	5м57с	***172 043***	9.02
Sin_16	16	16	65 536	20	17761	51678	4м5с	***9 068 945***	***16.28***
Syst_4	17	12	370	20 000	296	761	10м	129 618	6.77
Syst_8	25	20	45 548	20	29 368	87762	5м	14 453 628	21.50
				200	17 171	50959	30м	8 719 844	21.05
Vtx1	27	6	110	20 000	157	322	2м57с	***25 015***	4.95
X9dn	27	6	120	20 000	198	418	3м43с	25 897	5.73
Too_large_matr	38	3	1027	2 000	1 817	4958	9м53с	631 304	***15.88***
X1_matr	51	35	274	20 000	464	1101		92 879	***5.00***
Dalu_matr	75	16	194	20 000	253	724	40м	***81 116***	7.64
Soar_matr	83	94	529	2 000	960	2104	5м30с	191 600	6.41
				20 000	868	1865	1ч0м28с	175 558	6.73
X3_matr	135	99	915	2 000	1 590	3741	28м13с	318 953	9.84
Frg2_matr	143	139	3090	200	11 544	32132	18м	2 195 384	18.60

Таблица 3

Результаты исследования программы BDD_Builder

Пример	*n*	*m*	*k*	S_{BDD}	*V+&*	*t*	S_{ASIC}	τ
Rd73	7	3	147	43	111	0,25с	***15 925***	***3.94***
Dc2	8	7	58	63	124	1с	***21 505***	3.99
Dist	8	5	256	146	374	1с	69 560	5.09
M2	8	16	96	137	278	0,3с	***47 536***	***4.25***
M3	8	16	128	153	321	0,24с	***58 813***	5.20
Radd	8	5	120	35	87	1,02с	10 117	***3.34***
Root	8	5	256	75	148	0,27с	***26 717***	4.81
Z9sym	9	1	420	33	79	0,9с	***15 909***	***4.77***
Tial	14	8	640	720	1856	1,85с	294 830	***7.71***
Intb	15	7	664	822	2118	2.4с	327 401	9.21
B2	16	17	110	773	13153	2.4с	183 638	***7.84***
Sin_16	16	16	65 536	17 761	51679	47,6с **П**	9 105 109	17.27
Syst_4	17	12	370	366	839	0,8с	***115 791***	7.09
Syst_8	25	20	45 548	17 440	50090	24,3с **П**	***7 961 142***	***18.49***
Vtx1	27	6	110	103	199	1,2С	26 354	6.58
X9dn	27	6	120	104	204	0,7С	24 876	5.55
Too_large_matr	38	3	1 027	1 347	3602	57с	***511 759***	16.69
X1_matr	51	35	274	491	1083	10,6с	***79 565***	6.40
Dalu_matr	75	16	194	245	712	25,8с	82 316	***7.17***
Soar	83	94	529	530	904	30с	***135 298***	5.67
X3_matr	135	99	915	858	2134	4м57с	***242 406***	***7.91***
Frg2_matr	143	139	3 090	1 539	4028	2м50с**П**	***507 841***	***10.94***

Таблица 4

Улучшение BDD-решений программой BoolNet_Opt минимизации булевых сетей

Пример	Синтез по исходным описаниям		Лучшее схемное решение, полученное программами BDD-минимизации BDD_Builder, Tie_Energy, BDD_OPT		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

ЛИТЕРАТУРА

[1] Кнут Д.Э. Искусство программирования. Том 4, А. Комбинаторные алгоритмы. Часть 1: Пер. с англ.: ООО «И.Д. Вильямс», 2013. – 960 с.

[2] Карпов Ю.Г. MODEL CHECKING. Верификация параллельных и распределенных программных систем. СПб.: БХВ-Петербург, 2010. – 560 с.

[3] Валидация на системном уровне. Высокоуровневое моделирование и управление тестированием / М. Чэнь [и др.]– М. : Техносфера, 2014. – 296 с.

[4] Ebendt R., Fey G., Drechsler R. Advanced BDD optimization Springer, 2005. – 222p.

[5] Игнатьев А.С., Семенов А.А. Алгоритмы работы с ROBDD как с базами булевых ограничений // Прикладная дискретная математика. – 2010. – N 1. – С. 86 – 104.

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

[7] Amaru L.G. New Data Structures and Algorithms for Logic Synthesis and Verification. Springer, 2017. – 156 p.

[8] Dutta A., Baishnab K.L., Chaudhary S. A New Evolutionary Algorithm based BDD Optimization for Area and Power // International Journal of Electronic and Electrical Engineering. – 2010. – Vol. 3, № 3. - P. 147-160.

[9] Lindgren P., Kerttu M., Thornton M., Drechsler R. Low Power Optimization Technique for BDD Mapped Circuits // Proceedings of the ASP-DAC 2001 Asia and South Pacific Design Automation Conference 2001 (ASPDAC-01). – P. 615-621.

[10] 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. – 2017. – Т. 46. – № 1. – С.72–88.

[11] 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, № 12. – P. 1657–1663.

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

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

[14] URL: http://www1.cs.columbia.edu/~cs6861/sis/espresso-examples/ex (access date: 25.01.2020)

Experimental Research of Effectiveness of Programs for Minimizing BDD Representations of Boolean Function Systems in Synthesis of Combinatorial CMOS Circuits

P.N. Bibilo, Yury Y. Lankevich

United Institute of Informatics Problems of NAS of Belarus

bibilo@newman.bas-net.by, yurafreedom18@gmail.com

***Abstract* — the mathematical apparatus of BDD is used in various fields of science. 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, short reviews of these articles can be found in. 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. Also there are known approaches that are oriented to a final stage of synthesis - technology mapping based on covering BDD with logical element descriptions. It results in obtaining the structure description that is a netlist of a logical circuit in a given technological basis that is also called logical synthesis library. 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 — system of Boolean functions, Disjunctive Normal Form (DNF), Binary Decision Diagram (BDD), digital logic synthesis, VHDL, VLSI, CMOS.

REFERENCES

[1] Knut D.E. Iskusstvo programmirovaniya. Tom 4, A. Kombinatornye algoritmy (The Art of Computer Programming, Vol. 4A: Combinatorial Algorithms)CHast' 1: Per. s angl.: OOO «I.D. Vil'yams», 2013. – 960 s.

[2] Karpov YU.G. MODEL CHECKING. Verifikaciya parallel'nyh i raspredelennyh programmnyh system (MODEL CHECKING. Verification of parallel and distributed software systems). SPb.: BHV-Peterburg, 2010. – 560 s.

[3] Chen M., Qin K., Ku H.-M., Mishra P. Validaciya na sistemnom urovne. Vysokourovnevoe modelirovanie i upravlenie testirovaniem (Validation at the system level. High-level simulation and testing management). M.: Tekhnosfera, 2014. 296 p. (in Russian).

[4] Ebendt R., Fey G., Drechsler R. Advanced BDD optimization Springer, 2005. – 222p.

[5] Ignat'ev A.S., Semenov A.A. Algoritmy raboty s ROBDD kak s bazami bulevyh ogranichenij (Algorithms of work with ROBDD as with bases of Boolean restrictions) // Prikladnaya diskretnaya matematika. – 2010. – N 1. – S. 86 – 104.

[6] Bibilo P.N. Primenenie diagramm dvoichnogo vybora pri sinteze logicheskih skhem (Application of Binary Decision Diagrams at Synthesis of Logical Circuits). – Minsk : Belarus. navuka, 2014. – 231 s.

[7] Amaru L.G. New Data Structures and Algorithms for Logic Synthesis and Verification. Springer, 2017. – 156 p.

[8] Dutta A., Baishnab K.L., Chaudhary S. A New Evolutionary Algorithm based BDD Optimization for Area and Power // International Journal of Electronic and Electrical Engineering. – 2010. – Vol. 3, № 3. - pp. 147-160.

[9] Lindgren P., Kerttu M., Thornton M., Drechsler R. Low Power Optimization Technique for BDD Mapped Circuits // Proceedings of the ASP-DAC 2001 Asia and South Pacific Design Automation Conference 2001 (ASPDAC-01). – pp. 615-621.

[10] Bibilo P.N. Sistema logicheskogo proektirovaniya funkcional'nyh blokov zakaznyh KMOP SBIS s ponizhennym energopotrebleniem (The system of logical design of functional blocks of custom CMOS VLSI with the lowered energy consumption) // Mikroelektronika. – 2017. – T. 46. – № 1. – S.72–88.

[11] 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, № 12. – P. 1657–1663.

[12] Brayton R.K, Hachtel G.D., Sangiovanni-Vincentelli A.L. Multilevel Logic Synthesis // Proc. of the IEEE.– 1990. – Vol. 78, no. 2. – pp. 38–83.

[13] Bibilo P.N., Lankevich YU.YU. Logicheskaya minimizaciya bulevyh setej s ispol'zovaniem razlozheniya SHennona (Logical optimization of Boolean nets using Shannon expansion) // Informatika. – 2019. – T. 16. – № 2. – S. 73 - 89.

[14] URL: http://www1.cs.columbia.edu/~cs6861/sis/espresso-examples/ex (access date: 25.01.2020).