УДК621.382.82+ 621.391

ПРИМЕНЕНИЕ МЕТОДОВ МИНИМИЗАЦИИ БУЛЕВЫХ ФУНКЦИЙ ДЛЯ ОПТИМИЗАЦИИ ЦИФРОВЫХ УСТРОЙСТВ

© 2009 А. А. Смагин, А. В. Шиготаров

Ульяновский государственный университет

Поступила в редакцию 18.09.2009

В статье рассмотрены вопросы применения алгоритмов минимизации булевых функций для оптимизации цифровых устройств. Предлагаются точный и приближенный методы минимизации, основанные на использовании двоичных диаграмм решения и хешировании общих префиксов термов, соответственно. Результаты компьютерных экспериментов показывают преимущество предложенных алгоритмов (в смысле скорости работы), по сравнению с методами, приводимыми в литературе.

Ключевые слова: алгоритмы минимизации, булевы функции, цифровые устройства,

Булевы функции традиционно используются в качестве математических моделей цифровых устройств. Процедура минимизации булевых функций, как правило, применяется на этапе логического синтеза для получения экономичного представления проектируемого устройства. Так, например, площадь кристалла, занимаемого программируемой логической матрицей (ПЛМ), напрямую зависит от количества конъюнкций в ДНФ реализуемой булевой функции. Данной проблеме посвящено большое количество исследований, однако тенденция к автоматизированному проектированию цифровых устройств, а также постоянный рост степени интеграции обусловливают потребность в разработке новых, более эффективных схем минимизации.

В рамках данной работы были разработаны две схемы минимизации булевых функций в классе дизъюнктивных нормальных форм. Предлагаемые методы могут быть использованы в системах автоматизированного проектирования интегральных схем, а также для сжатия таблиц маршрутизации (ТМ). Последнее направление является особенно актуальным, вследствие постоянного увеличения размера ТМ. При этом, разработка эффективных схем хранения и поиска записей в таблицах маршрутизации является важнейшим моментом при построении быстродействующих сетей. Базовым методом Интернет-маршрутизации является сопоставление по наиболее длинному префиксу. Каждая запись таблицы маршрутизации содержит префикс узла назначения и соответствующий ему узел, на который необходимо отправлять трафик. Одному адресу назначения может соответствовать несколько записей в таблице – выбирается же из них та, которая имеет наиболее длинную маску подсети.

В настоящее время существует ряд программных и аппаратных методик повышения скорости поиска в таблицах маршрутизации. Общим недостатком алгоритмических подходов является то, что они требуют нескольких обращений к памяти. Требования к производительности современных сетей являются очень высокими, в то время как время отклика современных архитектур памяти ограничивают возможное количество обращений к памяти. Среди аппаратных средств можно выделить два основных решения – это специализированные интегральные схемы (ASIC) и контентно-адресуемая память (CAM)[6]. При этом наибольшее распространение получила именно CAM.

Главной особенностью CAM является то, что адресация осуществляется на основе содержания данных, поиск нужной записи, при этом, может быть осуществлен за одно обращение к памяти. В каждой позиции CAM может быть только 0 или 1. CAM поддерживают только сравнения образцов с фиксированной длиной и, следовательно, в явном виде не подходят для поиска префиксов. Для построения таблиц маршрутизации, как правило, используется тернарная контентно-адресуемая память (TCAM). Помимо индекса, TCAM хранят также отдельную маску для каждой записи. Маска определяет, какие из битов индекса являются активными. Основным недостатком TCAM является высокая стоимость и большое энергопотребление. При этом потребляемая мощность пропорциональна количеству записей, хранимых в ТМ.

Сжатие таблиц маршрутизации позволяет повысить скорость поиска нужной записи, а также использовать память меньшего размера. Ме-

Смагин Алексей Аркадьевич, доктор технических наук, профессор, зав. кафедрой "Телекоммуникационные технологии и сети". E-mail: smagin@sv.ulsu.ru.
Шиготаров Андрей Владимирович, аспирант.
E-mail: andreyshig@yandex.ru.

Таблица 1. Таблица префиксов в TCAM

Запись	Префикс	Маска	Следующий узел
1	10100000	11110000	1
2	10110000	11110000	1
3	11011000	11111000	2
4	10001000	11111000	3
5	10111000	11111000	2

тод уменьшения размера таблиц маршрутизации, рассматриваемый в данной работе основан на агрегации записей [4,7]. Так, например, записи 1 и 2 из табл. 1 отличаются только четвертым битом и могут быть скомбинированы (10100000, 11100000). Такой подход естественным образом сводится к задаче минимизации ДНФ – префиксам из таблицы маршрутизации мы сопоставляем элементарные конъюнкции, а каждому множеству записей, имеющих одинаковые длины префиксов k и следующий узел u, ставится в соответствие отдельная булева функция. Так, например, записям из таблицы 1, имеющим следующий узел 1 будет соответствовать функция $f_{4,1} = x_1\bar{x}_2x_3\bar{x}_4 \vee x_1\bar{x}_2x_3x_4$. Операции же объединения записей осуществляются посредством запуска процедуры минимизации ДНФ (относительно количества конъюнкций) для каждой функции $f_{k,u}$. Такое преобразование, очевидно, не влияет на процесс маршрутизации, то есть выбор узла, на который необходимо отправлять трафик.

Введем далее ряд основных определений, используемых в работе. Пусть задан алфавит $\{x_1,...,x_n\}$. Элементарной конъюнкцией называется логическое произведение вида $K = x_{i_1}^{\sigma_1} \& ... \& x_{i_r}^{\sigma_r}$ ($i_\nu \neq i_\mu$ при $\nu \neq \mu$, $\sigma_i \in \{0,1\}$, $x^0 = \bar{x}$, $x^1 = x$). Символ $x_{i_j}^{\sigma_j}$ называется буквой или простым сомножителем K. Под булевой функцией будем понимать функцию $f(x_1, x_2, ..., x_n)$ с областью определения $\{0,1\}^n$ и областью значений $\{0,1,*\}$, где * соответствует безразличному состоянию функции. Дизъюнктивной нормальной формой (ДНФ) называется дизъюнкция элементарных конъюнкций $D = K_1 \vee K_2 \vee ... \vee K_m$, в которой все K_j являются различными. Любая булева функция может быть представлена в виде ДНФ. Импликантой булевой функции $f(x_1, x_2, ..., x_n)$ называется элементарная конъюнкция $K = x_{i_1}^{\sigma_1} \& ... \& x_{i_r}^{\sigma_r}$, такая, что на любом наборе значений переменных $\{x_1,...,x_n\}$, на котором K принимает значение 1, функция f равна 1 или *. Импликанта называется простой, если при удалении из нее любого символа она перестает быть импликантой. В геометрической интерпретации набору значений переменных соответствует вершина n-мерного единичного куба E, а импликанте функции f – грань E такая, что на любой ее точке f принимает значение 1 или *.

Под проблемой минимизации булевых функций понимается задача построения для произвольной булевой функции ее минимальной, относительно заданного критерия, ДНФ. В качестве такого критерия наиболее часто используют количество конъюнкций, либо букв переменных в ДНФ. В контексте данной работы, нас, прежде всего, будет интересовать построение кратчайших ДНФ – т. е. ДНФ с наименьшим числом конъюнкций.

Проблеме минимизации булевых функций в классе ДНФ было посвящено множество исследований. Тем не менее не были найдены, а возможно и не существуют (в предположении, что $P \neq NP$) полиномиальные алгоритмы ее решения.

Основными факторами, затрудняющими построение минимальной ДНФ функции, являются большие размерности задачи – количество точек, в которых функция принимает значение 1 и количество всех простых импликант; первое, очевидно, не превосходит 2^n. Можно также показать, что верхняя граница количества простых импликант функции равна $3^n/n$ [1].

В зависимости от способа представления булевых функций и покрытий, существующие алгоритмы можно разбить на две группы – явные и неявные методы. Первые обрабатывают дискретные объекты последовательно, храня каждый из них в отдельной области памяти. При этом количество таких объектов может достигать значительных величин, что ограничивает сферу применения таких алгоритмов задачами небольшой размерности. Неявные же алгоритмы используют разделяемое представление для дискретных объектов, размер которого не зависит линейно от их количества. Традиционно в качестве такого представления используются двоичные диаграммы решения (BDD). В данной работе были использованы две разновидности BDD – упорядоченные двоичные диаграммы решения (OBDD) и диаграммы с отбрасыванием незначащих нулей (ZDD) – первые применяются для представления булевых функций, вторые – для представления покрытий.

Двоичные диаграммы решения сочетают в себе два полезных качества – приемлемый расход памяти и, как правило, быстрое выполнение операций над реализуемыми функциями. Упорядоченная двоичная диаграмма решений (OBDD) для заданных функции $f(x_1,...x_n)$ и отношения порядка $\pi : V \to \{0,1,...,n\}$, где $V = \{x_1,...,x_n\}$ – это направленный ацикличный граф, состоящий из нетерминальных вершин, отмеченных переменными из V и терминальными вершинами, отмеченными булевыми констан-

тами 1 и 0. Каждая нетерминальная вершина имеет два исходящих ребра – 1-ребро и 0-ребро. Начальный узел OBDD называется корнем. Для вычисления значения функции на заданном наборе значений переменных мы формируем путь от корня к терминальной вершине следующим образом: если переменная соответствующая текущей вершине принимает значение 1, то выбирается 1-ребро, в противном случае – 0-ребро. В сформированном пути мы можем встретить каждую переменную не более одного раза. На пути от корня к терминальной вершине, переменные встречаются в соответствии с заданным отношением порядка π.

За исключением таблиц истинности, OBDD-представление булевых функций является единственной структурой данных, которая имеет полиномиальные алгоритмы для всех базовых операций[5]. В связи с тем, что таблицы истинности всегда имеют экспоненциальный размер от количества переменных, на практике они применимы лишь для небольшого количества переменных. OBDD же может быть экспоненциально меньше таблицы истинности для реализуемой функции.

Любую булеву функцию можно представить в виде OBDD, однако такое представление, в общем случае, не является уникальным и это сильно усложняет такие алгоритмы, как, например, проверка эквивалентности. Однако с помощью очень простого механизма упрощения, произвольное OBDD-представление функции можно привести к стандартной, или канонической форме. Очевидно, что OBDD-представление функции после применения следующих двух правил упрощения по-прежнему реализует исходную функцию:

1. Если 1- и 0-ребра вершины v указывают на одну и ту же вершину u, тогда удаляем v и перенаправляем все входящие ребра v к u.
2. Все терминальные вершины с заданным значением объединяются в одну вершину, входящие ребра перенаправляются к этой вершине. Если две нетерминальные вершины u и v, отмечены одной и той же переменной и их 0- и 1-ребра указывают на одни и те же вершины соответственно, то удаляем одну из них и перенаправляем входящие ребра к оставшейся вершине.

OBDD называется упрощенной, если ни одно из перечисленных правил упрощения к ней неприменимо. Следующее основополагающее свойство каноничности [5], определяет важнейшие алгоритмические свойства упрощенных OBDD: для заданных булевой функции f и порядка переменных π, существует единственная упрощенная OBDD. Таким образом, не имеет значения, каким образом мы изначально сконструировали OBDD, так как с помощью простого повторения применения правил упрощения мы можем привести ее к канонической форме (относительно заданного порядка переменных).

Кроме универсальности и каноничности, упрощенная OBDD имеет еще одно фундаментальное свойство, которое делает ее удачным решением при выборе структуры данных для представления булевых функций – речь идет о высокой эффективности алгоритмов работы с OBDD. Тем не менее, опыт использования BDD показывает, что их применение не является оптимальным решением для многих задач дискретной оптимизации. В некоторых случаях число узлов в BDD-представлении становится слишком большим для эффективного выполнения требующихся операций. В частности, такая ситуация может иметь место для задач, оперирующих с разреженными множествами, представленными характеристическими функциями. В 1992 была предложена новая разновидность двоичных диаграмм решения, позволяющая эффективно работать с такими множествами – двоичные диаграммы решения с отбрасыванием незначащих нулей (ZDD).

В случае двоичных диаграмм с отбрасыванием незначащих нулей (ZDD), второе из описанных правил упрощения сохраняется, а вместо удаления вершин, исходящие ветви которых указывают на один и тот же узел, удаляются узлы с единичным выходом, указывающим на 0. Такая модификация правила упрощения позволяет существенно повысить эффективность работы с разреженными множествами [5].

В рамках данной работы были синтезированы два алгоритма минимизации ДНФ – точный и приближенный. Точный алгоритм минимизации использует классическую схему работы, состоящую из двух этапов:

1. Генерация всех простых импликант функции.
2. Поиск минимального покрытия множества точек, в которых функция принимает значение 1, множеством простых импликант.

Генерация простых импликант, а также удаление доминируемых строк и столбцов осуществляются с помощью методов, предложенных в [2].

Далее задача построения минимальной ДНФ рассматривается как задача о покрытии. Пусть M обозначает множество всех точек $\{m_1, m_2, ..., m_l\}$, на которых заданная функция f принимает значение 1, а $P = \{p_1, p_2, ..., p_k\}$ – множество всех простых импликант f. Матрицей покрытия будем называть бинарную матрицу A размерности $l \times k$ такую, что $A_{ij}=1$ (при этом будем говорить, что j-й столбец покрывает i-ю строку), $1 \le i \le l$, $1 \le j \le k$, если $m_i \in p_j$, в противном случае $A_{ij}=0$. Требуется найти минимальное количество столбцов, покрывающих все строки матрицы A.

Для нахождения минимального покрытия применяется алгоритм ветвей и границ, основанный на выборе столбца матрицы покрытия и разбиении множества решений задачи на два непересекающихся подмножества – одно из которых, содержит данный столбец, а другое нет. Затем данная процедура рекурсивно запускается для двух сгенерированных подзадач. Такое ветвление применяется до тех пор, пока мы не достигнем терминальной вершины, либо не определим, что данная ветвь не содержит оптимальное решение. С каждой вершиной дерева связывается нижняя граница стоимости покрытия, представленного данной вершиной. Если нижняя граница, соответствующая текущей вершине, равна $M \geq m$, где m – стоимость лучшего найденного покрытия, то мы можем прекратить ветвление. За счет отсечения ветвей, которые не содержат оптимального решения, в основном, и достигается экономия в вычислениях. Основными факторами, влияющими на эффективность метода ветвей и границ, являются: выбор столбца для ветвления, способ получения нижних и верхних границ.

В предлагаемом алгоритме для ветвления используется столбец [3]

$$y = \arg\max \sum_{x \in y} \frac{1}{|\{y \in Y \mid x \in y\}|}.$$

Большинство методов минимизации ДНФ (в том числе все неявные алгоритмы) используют для вычисления нижней границы подход, основанный на поиске максимального множества строк L таких, что ни один столбец не покрывает больше, чем одну строку из L. Матрице покрытия можно сопоставить граф, в котором вершины соответствуют строкам, причем две вершины соединены дугой тогда и, только тогда, когда существует столбец, покрывающий обе эти строки. Множеству L при этом соответствует максимальное независимое множество вершин. Пусть задана матрица покрытия, обозначим через X множество всех строк, через X', $X' \subseteq X$ – независимое множество строк. Для того, чтобы покрыть X мы должны покрыть X', следовательно, нижняя граница C может быть определена, как $C = |X'|$.

Задача построения максимального независимого множества является NP-сложной, поэтому на практике используются эвристические алгоритмы построения независимого множества, близкого к оптимальному.

Для вычисления нижних границ мы рассматриваем задачу построения минимального покрытия, как задачу целочисленного линейного программирования:

$$\begin{aligned} &\min \sum_{j=1}^{m} c_j x_j \ ; \\ &\sum_{j=1}^{m} a_{ij} x_j >= 1, i = 1..n \ ; \\ &x_j \in \{0,1\}. \end{aligned} \qquad (1)$$

При этом очевидно, что решение релаксированной задачи

$$\min\{cx \mid Ax >= b\} \ (2),$$

где b – единичный вектор, дает нижнюю границу для (1).

Для решения релаксированной задачи был использован двойственный симплекс-метод. При этом, необходимо заметить, что в процессе работы алгоритма ветвей и границ, нам не нужно решать (2) для каждой новой вершины заново. Вместо этого можно добавить к симплекс-таблице, соответствующей родительскому узлу ограничение x_j=1, либо x_j=0 и применить к ней двойственный симплекс-метод снова.

Схема разработанного приближенного алгоритма заключается в следующем. Обозначим X – множество всех точек, в которых минимизируемая функция f принимает значение 1. На первом шаге выбирается некоторая точка, и генерируются все простые импликанты, содержащие ее. Затем из них выбирается импликанта p, которая покрывает максимальное количество еще не покрытых точек; все точки, покрываемые p удаляются из X. Данный процесс продолжается до тех пор, пока в X не останется ни одного элемента. Для хранения точек, в которых f принимает значение 1 была использована двухуровневая структура данных, основанная на выделении у термов префиксов и последующей группировке по ним. Для реализации быстрого поиска термов применялось хеширование. Данный алгоритм минимизации ориентирован на применение к задаче сжатия таблиц маршрутизации и наиболее эффективен для функций с большим количеством конъюнкций в исходном ДНФ представлении.

Программная реализация рассмотренных алгоритмов была выполнена на языке C++, проект был скомпилирован с помощью gcc 3.2. При разработке использовались пакеты CUDD[14], extra[11] и lp_solve[12]. Предлагаемый точный алгоритм (MINDNF_E) сравнивался с двумя известными программами минимизации ДНФ – Espresso[8] и Rondo[15]. Эксперименты проводились на компьютере с процессором Intel Pentium 2.6 Ггц, и оперативной памятью 1 Гб. В качестве входных данных использовались набор MCNC Benchmarks и данные, сгенерированные на основе таблиц маршрутизации, полученных в рамках проекта [13].

Таблица 2. Сравнение алгоритмов минимизации ДНФ на наборе MCNC Benchmarks

Name	I	O	P	Rondo		MINDNF_E	
				TCC	TCP	TCC	TCP2
ex5*	8	63	2532	6.3	T	9.3	90.94
ibm	48	17	1047948792	0.53	0	0.72	0
jbp	36	57	2496809	3	0.08	2.17	0.06
mainpla	27	54	87692	83.9	0	176.41	0
max1024*	10	6	1278	0.7	T	0.84	400.2
misg	56	23	6499491840	0.27	0	0.34	0
misj	35	14	139103	0.145	0	0.17	0
pdc	16	40	28231	4.15	2.1	4.14	2.58
prom2*	9	21	2635	14.5	T	17.72	54
shift	19	16	165133	1.54	0	0.53	0
signet	39	8	78735	6.48	0	6.41	0
soar*	83	94	330477287437440	30.8	T	15.2	58.73
ti	47	72	836287	3.2	0.4	3.83	0.33
ts10	22	16	524281	0.5	0	0.58	0
x7dn	66	15	566698632	19.2	2.3	8.55	4.63
xparc	41	73	15039	5.3	0.01	4.2	0.09

I – число входных переменных; O – число выходных переменных; P – количество простых импликант; TCC – время построения упрощенной матрицы (сек); TCP – время поиска минимального покрытия (сек)

Результаты экспериментов для задач из набора MCNC Benchmarks приведены в таблице 2. При этом для систем булевых функций использовалось представление в виде характеристической булевой функции. При этом было установлено ограничение на время работы – 1 час. С помощью espresso за данное время не удалось найти решение ни для одной из задач. Программа Rondo не нашла решения для 4 задач, отмеченных *. С помощью MINDNF удалось найти решения для всех приведенных задач.

На рис. 1 показаны результаты сравнения времени работы предлагаемых методов MINDNF_E(точный), MINDNF_A (приближенный) и метода espresso-exact, использовавшегося для оптимизации таблиц маршрутизации в [4]. При этом, суммарное время минимизации при использовании MINDNF_A вменьше в 4.35 раз меньше, по сравнению с MINDNF_E, и в 91.2 - по сравнению с espresso-exact. Стоимость решений, получаемых с помощью MINDNF_A на 3-5% больше стоимости оптимальных решений (рис. 2).

ЗАКЛЮЧЕНИЕ

В данной работе рассмотрены методы оптимизации цифровых устройств, на основе применения алгоритмов минимизации булевых функций. Разработанные алгоритмы могут быть использованы в системах автоматизаированного проектирования интегральных схем, а также для сжатия таблиц маршрутизации большой размерности, реализованных на основе тернарной контентно-адресуемой памяти. Экспериментальные результаты показывают, что время работы предлагаемого приближенного метода минимизации значительно меньше, чем у алгоритмов, приводимых в литературе. Качество получаемых решений, при этом, отличается на 3-5% от точных решений. Точный алгоритм минимизации может быть использован для оценки качества решений, получаемых с помощью приближенных методов.

СПИСОК ЛИТЕРАТУРЫ

1. *Chandra A.K., Markowsky G.* On the number of prime implicants.– Discrete Mathematics 24(1978), pp 7-11.
2. *Coudert O., Madre J.C.* Implicit and incre-mental

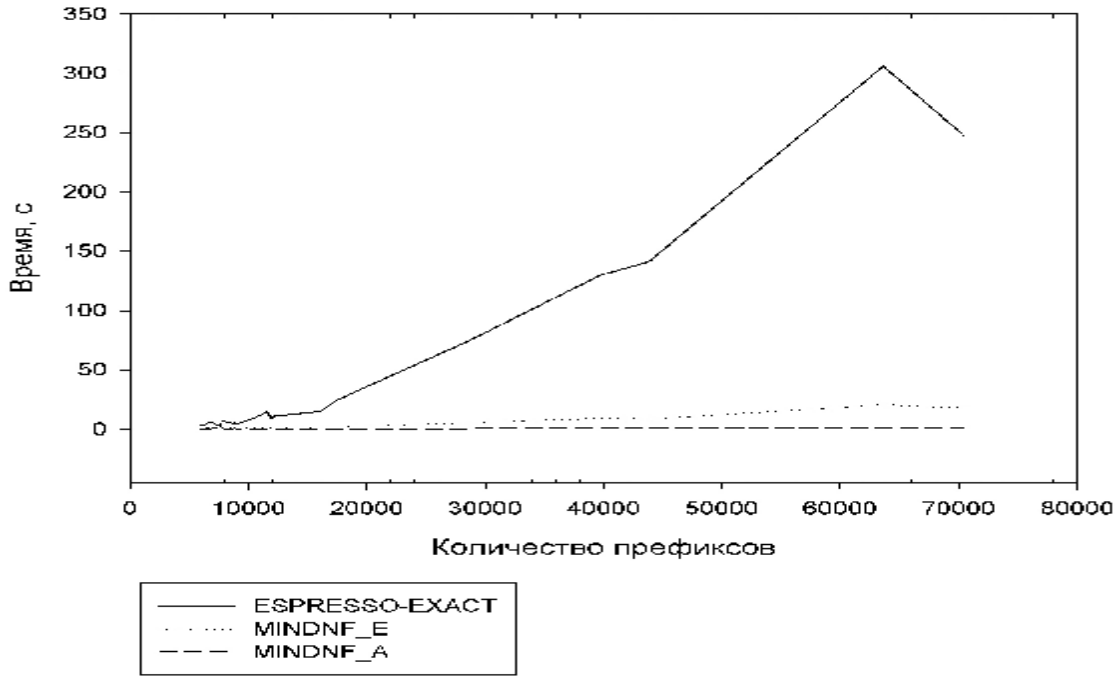

Рис. 1. Зависимость времени минимизации от количества префиксов

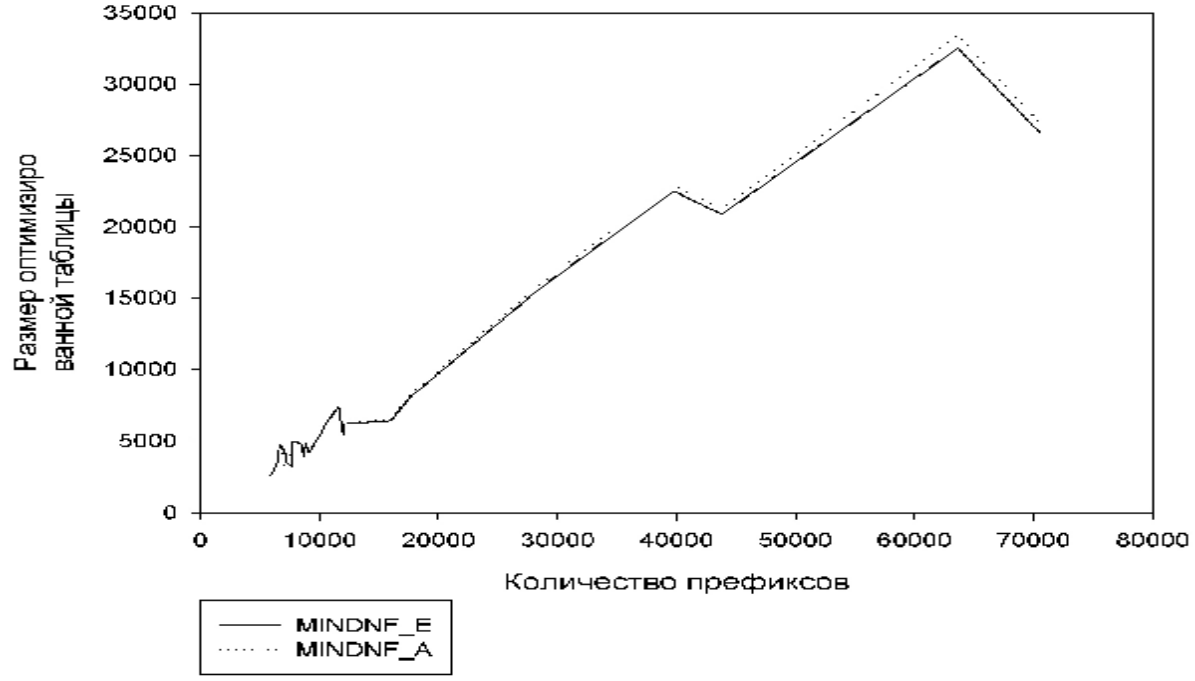

Рис. 2. Зависимость размера сжатой таблицы маршрутизации от размера исходной

computation of primes and essential primes of Boolean functions // Proc. of the Design Automation Conf. (Anaheim, CA, 1992), pp 36-39.

3. *Coudert O., Madre J.C.* New ideas for solv-ing covering problems // Proc. of the Design Automation Conference, 1995, pp 641-645.
4. *Liu H.* Routing Table Compaction in Ter-nary-CAM // IEEE Micro, Jan/Feb 2002, pp. 58-64.
5. *Meinel C., Theobald T.* Algorithms and data structures in VLSI design, Springer-Verlag NY, 1998. – 267 p.
6. *McAuley A., Francis P.* Fast Routing Table Lookup Using CAMs // Proc. IEEE Infocom, vol. 3, IEEE CS Press, Los Alamitos, Calif., 1993, pp. 1382-1391.
7. *Ravikumar V.C., Bhuyan L.N., Mahapatra R.N.,* EaseCAM: An Energy and Storage Efficient TCAM-Based Router Architecture for IP Lookup // IEEE Transactions on Computers, vol. 54, 2005, 521-533.
8. *Rudell R.* Multiple-valued minimization for PLA synthesis. UCB technical report M86/65, 1986.
9. *Stergiou S., Jain J.* Optimizing Routing Tables on Systems-on-Chip with Content-Addressable Memories //*Proc. IEEE International System-on-Chip Symposium*,

2008, pp 1-6.
10. *Umans C., Villa T.* Sangiovanni-Vincentelli A. Complexity of two-level logic minimization. IEEE Transactions on Computer-Aided design, 2006, pp 1230-1246.
11. http://www.ee.pdx.edu/~alanmi/research/ex-tra.htm.
12. lpsolve.sourceforge.net.
13. http://www.routeviews.org/.
14. vlsi.colorado.edu/~fabio/.
15. http://web.cecs.pdx.edu/~alanmi/research/min/rondo.exe.

LOGIC MINIMIZATION BASED APPROACH FOR OPTIMIZATION OF DIGITAL DEVICES

© 2009 A.A. Smagin, A.V. Shigotarov

Ulyanovsk State University

This article deals with the using of logic minimization algorithms for optimization of digital devices. We propose exact and approximate minimization algorithms, based on the using of binary decision diagrams and hashing of common prefixes of the terms, respectively. Experimental results show that our methods outperform well-known methods appeared in literature in terms of runtime.

Key words: logic minimization algorithms, Boolean functions, digital devices.

Alex Smagin, Doctor of Technics, Professor, Head at the Telecommunication Technologies and Networks Department.
E-mail: smagin@sv.ulsu.ru.
Andrew Shigotarov, Graduate Student.
E-mail: andreyshig@yandex.ru.