МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

МАТЕРИАЛЫ

III Всероссийской молодежной научной конференции

«МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННЫХ, ТЕХНИЧЕСКИХ И ЭКОНОМИЧЕСКИХ СИСТЕМ»

Томск, 22–23 мая 2015 г.

Под общей редакцией
кандидата технических наук И.С. Шмырина

Томск
Издательский Дом Томского государственного университета
2015

Таблица 1

Результаты исследования безызбыточных систем ДНФ, полученных из STG описаний поведения реальных дискретных устройств

Название схемы	I	O	P	Общее количество путей	Количество ложных путей	Доля ложных путей,%
BBARA_b_k_2k_s_min	10	9	120	597	60	9.95
TRAIN11_b_k_2k_s_min	8	8	42	100	0	0
LION9_b_k_2k_s_min	8	8	64	228	25	9.12
LOG_b_k_2k_s_min	15	36	117	354	0	0
OPUS_b_k_2k_s_min	11	15	102	353	0	0
BEECOUNT_b_k_2k_s_min	9	10	67	241	28	8.6
SHIFTREG_b_k_2k_s_min	7	8	43	124	16	7.75
MODULO12_b_k_2k_s_min	7	7	31	80	0	0

Обозначения в таблице: I – количество входов схемы. O – количество выходов схемы. P – количество продукций в PLA описании.

Заключение

Разработан алгоритм поиска ложных путей с учетом дополнительной информации о схеме, а именно, с использованием STG описания рабочей области функционирования устройства с памятью.

Реализация алгоритма требует больших вычислительных затрат, поскольку для каждой из константных неисправностей литеры необходимо находить все множество тестовых наборов, представляя его в виде сокращенной ДНФ. Такая ДНФ, как известно, может оказаться большой длины. Кроме того, многократно приходится перебирать пары переходов рабочей области функционирования устройства с памятью.

Из сказанного следует, что поиск ложных путей в рамках предлагаемого подхода является достаточно ресурсоемкой процедурой, однако такой подход обеспечивает нахождение всего множества ложных путей.

ЛИТЕРАТУРА

1. *Matrosova A., Lipsky V., Melnikov A., Singh V.* Path delay faults and ENF // Proceedings of IEEE East-West Design and Test Symposium. 2010. P. 164–167.

2. *Матросова А.Ю., Кудин Д.В., Николаева Е.А.* Обнаружение ложных путей в комбинационной схеме. Томск: Изд-во Том. ун-та, 2011.

3. *Седов Ю.В.* Обеспечение работоспособности систем с произвольным доступом и самопроверяемости логических схем: диссертация ... кандидата технических наук: 05.13.01. – Томск, 2004. – 117 с. ил.

4. *Николаева Е.А.* Алгоритмы синтеза легко тестируемых комбинационных схем и тестов для кратных константных неисправностей и неисправностей задержек путей: диссертация ... кандидата технических наук: 05.13.01. – Томск, 2011. – 135 с. ил.

АЛГОРИТМ ПОИСКА МИНИМАЛЬНОГО BDD-ГРАФА, ПРЕДСТАВЛЯЮЩЕГО РЕАЛИЗАЦИЮ ЧАСТИЧНО ОПРЕДЕЛЕННОЙ БУЛЕВОЙ ФУНКЦИИ

И. Е. Кириенко

Томский государственный университет

E-mail: irina.kirienko@sibmail.com

Введение

Существует множество практических задач, где нужно найти минимальную, в смысле количества узлов BDD, реализацию частично определенной булевой функции,

например, логический синтез в базисе FPGA [2], синтез маскирующих подсхем для повышения надежности частично программируемых схем [3].

В общем случае задача поиска одной из реализаций частично определенной булевой функции, представленной BDD с минимальным числом узлов, является NP-полной [5], в связи с этим разработано множество эвристических алгоритмов, решающих эту задачу за приемлемое время [5, 6]. Для некоторых приложений точность решения данной задачи эвристическими алгоритмами не достаточна, нужно использовать алгоритмы, дающие точное решение.

В данной работе рассматривается точный алгоритм построения минимального BDD-графа, описанного в работе [7]. Алгоритм основан на использовании графа совместимости с последующим выделением клик графа с использованием дополнительных условий. Во втором разделе работы рассматриваются необходимые определения. В третьем разделе приводится определение графа совместимости и алгоритм его построения. В четвертом разделе формулируются дополнительные ограничения при поиске замкнутого покрытия кликами графа совместимости. В пятом разделе предлагается модификация алгоритма построения минимального BDD-графа, являющегося реализацией частично определенной булевой функции. В шестом разделе приведены экспериментальные результаты.

1. Основные определения

BDD-граф – это направленный ациклический граф, каждый узел которого помечен переменной x_i, и где каждый нетерминальный узел n_i имеет исходящие 0- и 1-дуги, соединяющие узел n_i с дочерними узлами n_i^z и n_i^o, соответственно. Терминальные узлы обозначим n_z и n_o. Далее 0-дуга будет соответствовать левой исходящей дуге из узла n_i, 1-дуга – правой.

BDD-граф соответствует полностью определенной булевой функции $f(x_1,...,x_n)$. Множество f_{on} содержит наборы, на которых функция $f(x_1,...,x_n)$ принимает значение 1, множество f_{off} содержит наборы, на которых функция $f(x_1,...,x_n)$ принимает значение 0.

BDD-граф называется сокращенным (ROBDD), если из любых двух узлов не существует одного и того же пути в терминальный узел. Так как ROBDD-граф с заданным порядком переменных является каноническим представлением полностью определенной булевой функции *f*, будем обозначать n_i как узел ROBDD-графа и как переменную булевой функции *f*.

Уровнем $L(n_i)$ узла n_i назовем индекс переменной в порядке разложения. Пусть корень BDD-графа имеет уровень 0. Если *N* – количество переменных, то *N* – уровень терминальных узлов BDD-графа. Максимальный уровень $L_{\max}(n_i)$ множества *s* узлов – это максимальный уровень среди узлов множества *s*. Уровнем $L(h)$ функции *h* называется уровень переменной BDD-графа, реализующего функцию *h*. Если n_i – узел BDD-графа, *m* – набор значений переменных, то $n_i(m)$ – терминальный узел, который достигается из вершины n_i при *m* (или значение функции n_i на наборе *m*). Поэтому, если *m* – набор значений переменных, *F* – BDD-граф для функции *f*, то $n_0(m)$ – значение функции *f* на наборе *m*.

BDD-граф называется полным, если каждое ребро BDD-графа соединяет узел уровня *i* с узлом уровня *i*+1.

Трехтерминальный BDD-граф (3TBDD) определяется аналогично BDD-графу с тем исключением, что 3TBDD имеет 3 терминальные вершины n_z, n_o и n_x. 3TBDD-граф соответствует частично определенной булевой функции *f*, которая имеет множества наборов f_{on}, f_{off} и f_{dc}, на которых функция *f* принимает значения n_o, n_z и n_x, соответственно.

2. Граф совместимости

В данной работе будем использовать 3TBDDF, который соответствует частично определенной булевой функции *f*. Будем полагать, что граф *F* полный (при необходимости можно ввести дополнительные узлы, 0- и 1-дуги которых ведут в тот же узел).

Введем определения совместимых узлов и узлов, совместимых в обобщенном смысле.

Определение 1. Два узла n_i и n_j графа *F* будем называть совместимыми ($n_i \sim n_j$), если и только если не существует такого набора *m*, для которого справедливы $n_i(m) = n_z \vee n_j(m) = n_o$ или $n_i(m) = n_o \vee n_j(m) = n_z$.

По определению терминальные узлы n_z и n_o не совместимы между собой, терминальный узел n_x совместим с любой вершиной 3TBDD.

Определение 2. Два узла n_i и n_j графа *F* будем называть совместимыми в обобщенном смысле ($n_i \approx n_j$), если и только если существует полностью определенная функция *h* такая, что $h \sim n_i$ и $h \sim n_j$ и $L(h) \geq \max(L(n_i), L(n_j))$.

По определению терминальные узлы n_z и n_o между собой не совместимы в обобщенном смысле, терминальный узел n_x совместим в обобщенном смысле с любой вершиной 3TBDD.

Важно отметить, что узел, соответствующий полностью определенной функции *h*, необязательно является некоторым узлом 3TBDD. В большинстве случаев это действительно так, поскольку многие узлы 3TBDD соответствуют частично определенным функциям.

Следующая лемма устанавливает отношение между совместимостью и совместимостью в обобщенном смысле.

Лемма 1. Если $n_i \approx n_j$, то $n_i \sim n_j$.

Лемма 2 формулирует условие, при котором верно и обратное утверждение.

Лемма 2. Если $L(n_i) = L(n_j)$, то $n_i \sim n_j$, следовательно $n_i \approx n_j$.

Совместимость в обобщенном смысле можно определить для множества узлов.

Определение 3. Узлы множества $s_i = (n_1, n_2, ..., n_s)$ совместимы в обобщенном смысле, если и только если существует полностью определенная булева функция *h* такая, что $h \sim n_i$, $\forall i \in \{1, ..., s\}$, и $L(h) \geq L_{\max}(s_i)$.

Определение 4. Множество узлов, совместимых в обобщенном смысле, будем называть совместимым множеством или совместимостью.

Множество узлов является совместимостью, если любые два узла этого множества попарно совместимы в обобщенном смысле. В общем случае обратное утверждение неверно. Однако справедлива следующая лемма.

Лемма 3. Пусть s_i – это множество узлов, принадлежащих одному уровню в графе. Тогда s_i является совместимым множеством, если и только если все узлы в s_i попарно совместимы в обобщенном смысле.

Введем определение графа совместимости.

Определение 5. Граф совместимости $G=(V,E)$ – это не ориентированный граф, который показывает, какие узлы в 3TBDD-графе F могут быть совмещены. Кроме терминального узла n_x, каждому узлу графа F соответствует один узел в графе G с тем же значением уровня. Узлам n_i^z и n_i^o графа F соответствуют узлы g_i^z и g_i^o графа G.

В графе G два узла g_i и g_j соединены ребром, если узлы n_i и n_j графа F совместимы в обобщенном смысле. Каждое ребро в графе G имеет метки. Метка – это множество узлов. Существует три типа меток: типа e, t и l.

Следующие две леммы необходимы для построения графа совместимости.

Лемма 4. Если $L(n_i)=L(n_j)$ и $n_i \approx n_j$, то $n_i^o \approx n_j^o \vee n_i^z \approx n_j^z$.

Лемма 5. Если $L(n_i)<L(n_j)$ и $n_i \approx n_j$, то $n_i^o \approx n_j \vee n_i^z \approx n_j \vee n_i^o \approx n_i^z$.

Алгоритм построения графа совместимости.

1. Пусть граф G – полный. Удаляем одно ребро (g_z, g_o).
2. Если $L(g_i)=L(g_j)$, тогда согласно лемме 4 ребро (g_i, g_j) имеет две метки: $e:(g_i^0, g_j^0)$ и $t:(g_i^1, g_j^1)$.
3. Если $L(g_i)<L(g_j)$, тогда согласно лемме 5 ребро (g_i, g_j) имеет метку $l:(g_i^0, g_i^1, g_j)$.
4. Для всех пар узлов (g_i, g_j) проверяем, если ребро между узлами g_i и g_j имеет метку, которая содержит пару (g_a, g_b), при этом между вершинами g_a и g_b нет ребра, то удаляем ребро (g_i, g_j). Повторяем шаг 4 до тех пор, пока граф G не перестанет изменяться.

Следующая лемма устанавливает связь между существованием ребра в графе совместимости и понятиями совместимости и совместимости в обобщенном смысле.

Лемма 6. Если $n_i \approx n_j$, то в графе совместимости существует такое ребро e, что $e=(g_i, g_j)$. Следовательно, $n_i \sim n_j$.

Важно отметить, что в общем случае обратное утверждение неверно, то есть существование ребра между двумя узлами графа G не означает, что эти узлы совместимы в обобщенном смысле.

Рассмотрим 3TBDD, представленный на рис. 1. Для данного 3TBDD построен граф совместимости с помощью приведенного выше алгоритма.

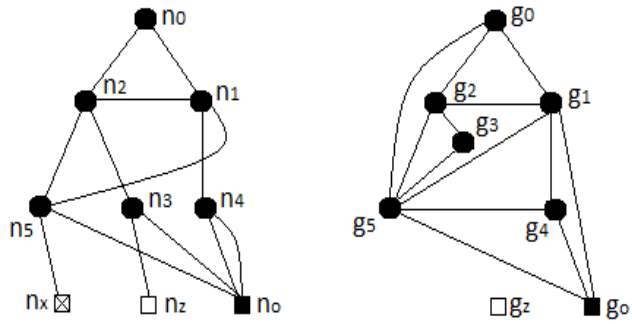

Рис. 1. 3TBDD и соответствующий граф совместимости

3. Замкнутое покрытие кликами

Кликой графа G называется его подграф, являющийся полным графом. Любому множеству s узлов, образующему клику графа G, соответствует класс множеств. Пусть $s_i = (g_{i_1},...,g_{i_w})$ – это множество узлов, которые образуют клику графа G. Для данного множества s_i определим e, t и l классы.

Определение 6. e-класс множества s_i (обозначим как $C_e(s_i)$) – это множество узлов из e-метки ребра между узлами g_j и g_k из s_i таких, что $L(n_k) = L(n_j) = L_{\max}(s_i)$.

Определение 7. t-класс множества s_i (обозначим как $C_t(s_i)$) – это множество узлов из t-метки ребра между узлами g_j и g_k из s_i таких, что $L(n_k) = L(n_j) = L_{\max}(s_i)$.

Определение 8. l-класс множества s_i (обозначим как $C_l(s_i)$) – это множество узлов из l-метки ребра между узлами g_j и g_k из s_i таких, что $L(g_j) \neq L(g_k)$.

Лемма 7. Если множество узлов s_i образует клику графа G и $C_l(s_i) \subseteq s_i$, то s_i является совместимым множеством.

Важно отметить, что некоторая клика графа G, не удовлетворяющая условию леммы 7, необязательно будет совместимым множеством.

Алгоритм поиска минимальной BDD, совместимой с исходной частично определенной функцией, основан на выборе таких узлов графа G, которые могут быть заменены одним узлом в финальной BDD. Если множество s узлов графа G могут быть объединены в один узел, то s должно быть совместимым множеством. Следовательно, множество s должно быть кликой графа G, удовлетворяя условию определения 7. Таким образом, необходимо найти такое множество клик, что каждый узел графа G будет покрыт по крайней мере одной кликой. Данный поиск требует дополнительных ограничений.

Определение 9. Множество $S = \{s_1,...,s_n\}$ множеств узлов графа G называется замкнутым покрытием кликами для графа G, если справедливы следующие условия:

1. s является покрытием G: $\forall g_i \in G \; \exists \, s_j \in S : g_i \in s_j$.
2. Все s_k являются кликами графа G:
$$\forall g_i, g_j \in s_k : (g_i, g_j) \in edges(G).$$
3. s замкнуто относительно меток типа e и t:
$$\forall s_i \in S \exists s_j \in S : C_e(s_i) \subseteq s_j \wedge \; \forall s_i \in S \exists s_j \in S : C_t(s_i) \subseteq s_j .$$
4. Все множества из s замкнуты относительно метки типа l: $\forall s_i \in S : C_l(s_i) \subseteq s_i$.

4. Построение минимального BDD-графа

Сформулируем модификацию алгоритма построения минимального BDD-графа R.

1. Строим замкнутое покрытие кликами $S = \{s_1,...,s_n\}$ для графа G. На каждой итерации проверяем замкнутость относительно класса C_l, если покрытие не замкнуто, переходим к другому подмножеству.
2. Для каждого $s_i \in S$ создаем узел r_i нового графа R, при чем уровень этого узла $L(r_i) = L_{\max}(s_i)$.
3. Узел r_i, соответствующий множеству $s_i \in S$, будет терминальным в графе R, если множество s_i содержит узел, который соответствует терминальному узлу в графе F.

4. 0-ребро узла r_i соединяет его с узлом r_j, соответствующим множеству $s_j \in S$, если $C_e(s_i) \subseteq s_j$.

5. 1-ребро узла r_i соединяет его с узлом r_j, соответствующим множеству $s_j \in S$, если $C_t(s_i) \subseteq s_j$.

Лемма 8. Построенный граф *R* является OBDD-графом, совместимым с исходным графом *F*.

Отметим полученный результат. Пусть $\Re$ – это множество всех BDD-графов, представляющих функции, каждая из которых совместима с исходной частично определенной булевой функцией *f*. Тогда справедлива следующая лемма.

Лемма 9. BDD-граф, построенный на основе минимального замкнутого покрытия для *G*, является BDD-графом из $\Re$ с минимальным числом узлов. [7]

5. Экспериментальные результаты

В ходе работы проведен ряд экспериментов для произвольных частично определенных булевых функций, для каждой из которых осуществляется поиск такой полностью определенной булевой функции, что, во-первых, она является реализацией данной частично определенной булевой функции, во-вторых, ее BDD-представление имеет минимальный размер. Результаты экспериментов приведены в таблице 1.

Таблица 1

Экспериментальные результаты

Кол-во переменных	Степень неопределенности								
	20%			50%			80%		
	Кол-во узлов в финал. BDD	Время, мс		Кол-во узлов в финал. BDD	Время, мс		Кол-во узлов в финал. BDD	Время, мс	
		Полный перебор	Модификация		Полный перебор	Модификация		Полный перебор	Модификация
4	9	3.0663	2.1236	8	3.2194	3.5443	5	5.8666	4.1948
4	8	2.7533	1.77	6	2.3573	2.4319	6	18.8894	9.0928
4	10	2.7383	0.9935	6	7.0954	4.9602	6	2.6759	1.7072
5	12	9.5365	3.5001	12	1.8039	1.4064	9	42.723	20.3136
5	13	14.953	12.8933	9	11.2859	6.5554	9	164.0122	43.9712
5	15	7.515	5.0214	14	18.4323	11.0688	7	109.6831	467.8115

Таблица содержит информацию о времени выполнения поиска клик графа совместимости, которые замкнуты относительно класса *L*. Указанное время рассчитывалось для алгоритма, основанного на полном переборе, и для алгоритма с использованием модификации. Полученные экспериментальные данные подтверждают тот факт, что модификация алгоритма позволяет сократить количество операций при вычислении клик графа без потери точности решения.

Заключение

Проведенный анализ показал, что модифицированный алгоритм требует меньше времени для построения минимального BDD-графа по сравнению с базовым алгоритмом.

ЛИТЕРАТУРА

1. *Bryant R.* Graph Based Algorithm for Boolean Function Manipulation // IEEE Trans. Computers. – 1986. – V. 35. – No.8. – P. 667–691.

2. *Matrosova A., Loukovnikova E., Ostanin S., Zinchuk A., Nikoleva E.* Test Generation for Single and Multiple Stuck-at Faults of a Combinational Circuit Designed by Covering Shared ROBDD with CLBs // Proc.IEEE Intl.Symp. on Defect and Fault-Tolerance in VLSI Systems (DFT 2007). – 2007. – P. 206–214.

3. *Matrosova A., Ostanin S., Kirienko I.* Increasing Manufacturing Yield Using Partially Programmable Circuits with CLB implementation of Incompletely Specified Boolean Function of the Corresponding Sub-circuit // Proc. IEEE Intl. Symp. on Design and Diagnostics of Electronic Circuits and Systems (DDECS 2015). – 2015.

4. *Lavagno L., McGeer P., Saldanha A., Sangiovanni-Vincentelli A.* Timed Shannon Circuits: A Power-Efficient Design Style and Synthesis Tool// Proc. 32nd Design Automation Conf. – 1995. – P. 254–260.

5. *Shiple T., Hojati R., Sangiovanni-Vincentelli A., Bryaton R.* Heuristic Minimization of BDDs Using Don't Cares// Proc. Design Automation Conf. – 1994. – P. 225–231.

6. *Chang S., Cheng D., Marek-Sadowska M.* Minimizing ROBDD Size of Incompletely Specified Multiple Output Functions // Proc. European Design and Test Conf. – 1994. – P. 620–624.

7. *Oliveira A., Carloni L., Villa T., Sangiovanni-Vincentelli A.* Exact minimization of Binary Decision Diagrams Using Implicit Techniques// IEEE Trans. Computers. – 1998. – V. 47. – No. 11. – P. 1282–1296.

ИССЛЕДОВАНИЕ ДЕКАРТОВОГО ДЕРЕВА И НАПИСАНИЕ ОБУЧАЮЩЕЙ ПРОГРАММЫ ПО ЕГО ПОСТРОЕНИЮ

Т. С. Овчинникова, В. А. Сибирякова
Томский государственный университет
E-mail: tanyalastochkina@mail.ru, val349@mail.ru

Введение

В настоящее время роль информационных технологий очень важна. Поэтому возникает необходимость создания обучающих систем. Но в нынешнем разнообразии систем нет эффективных программ для изучения деревьев поиска. А эти структуры данных очень важны для хранения и быстрого поиска информации в базах данных и других поисковых системах.

Поэтому целью данной работы было создание обучающей системы для изучения декартового дерева поиска. Данная система может быть использована не только студентами, но и преподавателями для проверки контрольных работ, а также любыми желающими с целью изучить декартовы деревья и, в дальнейшем, применять их. Большой плюс обучающей программы в том, что желающий может работать с ней в любой день, в любое время суток и продолжительность обучения не ограничена.

1. Постановка задачи

Требуется изучить свойства декартового дерева и написать обучающую программу. Обучающая программа должна:

1) демонстрировать алгоритм построения дерева;
2) проверять знания пользователя по построению дерева.

2. Определение декартового дерева

Декартово дерево – это двоичное дерево поиска, в узлах которого хранятся:

1) ссылки на правое и левое поддерево;
2) ключи x и y, которые являются двоичным деревом поиска по ключу x и двоичной кучей по ключу y; а именно, для любого узла дерева n:
   a) ключи x узлов правого (левого) поддерева больше (меньше либо равны) ключа x узла n;
   b) ключи y узлов правого и левого детей больше либо равны ключу y узла n.

В англоязычной литературе очень популярно название treap, которое наглядно показывает суть структуры: tree + heap. В русскоязычной же иногда можно встретить составленные по такому же принципу: дерамида (дерево + пирамида) или дуча (дерево + куча).

Впервые дерамиды были предложены в статье Seidel, Raimund; Aragon, Cecilia R. (1996), «Randomized Search Trees» [1].