УДК 004.023

Методы разбиения логических схем для оптимизации решения задач проверки эквивалентности и функциональной коррекции схем

Г.В. Антюфеев[1], В.В. Жуков[1], Е.Ю. Зенин[2], М.С. Шуплецов[1]

[1]Московский государственный университет им. М.В. Ломоносова – факультет вычислительной математики и кибернетики, grigoriy.rus@gmail.com, shupletsov@cs.msu.ru

[2]ИНЕУМ им. И.С. Брука, euzenin@gmail.com

***Аннотация* — Международное соревнование «ICCAD CAD Contest» является одним из ключевых в области разработки алгоритмов автоматизации проектирования интегральных схем, которое проводится под эгидой международной конференции International conference on computer-aided design (ICCAD) и при поддержке министерства образования Тайваня. В рамках соревнования 2015 года компанией Cadence design systems, Inc была предложена задача поиска таких разбиений логического описания интегральной схемы, которые позволяют одновременно оптимизировать процесс проверки эквивалентности схем и их функциональной коррекции. В данной работе авторы представляют решение поставленной задачи, которое заняло первое место по результатам соревнования, а также некоторые его дополнительные усовершенствования.**

***Ключевые слова* — схемы из функциональных элементов, логический синтез, разбиение схем, проверка эквивалентности, функциональная коррекция, поиск функциональных соответствий.**

I. Введение

Разработка эффективных алгоритмов проверки эквивалентности [1] и функциональной коррекции [2] для схем большого размера является одной из ключевых технологий для современных маршрутов проектирования сверхбольших интегральных схем (СБИС). Описания современных интегральных схем обладают очень большим размером и сложностью, поэтому анализ указанных описаний, рассматриваемых как единое целое, требует значительных вычислительных ресурсов. Одним из способов решения указанной проблемы может служить разбиение исходных схем на подсхемы меньшего размера. Такое разбиение позволяет снизить сложность рассматриваемых задач анализа схем.

Задачи проверки эквивалентности и функциональной коррекции предполагают анализ различий в логической структуре и функционировании пары схем, первая из которых является исходным описанием схемы, а вторая получается в результате решения задачи логического синтеза.

При решении задачи проверки эквивалентности схем нахождение соответствующих друг другу точек в исходном описании и синтезированной схеме позволяет выделить в рассматриваемых схемах группы функционально эквивалентных подсхем, которые можно использовать для разбиения исходных описаний схем. В результате такого разбиения рассматриваемая задача проверки эквивалентности может быть разделена на несколько задач проверки эквивалентности для схем меньшего размера, что существенно снижает её сложность. Тем не менее, нужно учитывать тот факт, что различные оптимизации, проводимые при синтезе схемы, могут существенно осложнить поиск указанных выше точек соответствия в силу возникающих структурных различий между исходным описанием и синтезированной схемой.

В свою очередь, при решении задачи функциональной коррекции нахождение таких точек соответствия и выделение эквивалентных подсхем помогает локализовать фрагменты схем, содержащие функциональные различия, что также упрощает решаемую задачу и позволяет минимизировать набор требуемых функциональных изменений. Для современных маршрутов проектирования интегральных схем, когда время от начала проектирования устройства и до его выхода на рынок является ключевым фактором, поздние модификации в структуре схемы, связанные с изменением спецификации или при обнаружении логических ошибок в схеме, реализуются при помощи формирования небольшой подсхемы-заплатки (англ. Engineering Change Order, ECO) [2]. Внедрение такой подсхемы в уже синтезированную схему позволяет исправить все обнаруженные логические несоответствия, и при этом не требуется заново полностью повторять все этапы логического и физического синтеза интегральной схемы, что существенно сокращает время проектирования схемы. Необходимо учитывать, что для больших схем обнаружение небольших неэквивалентных фрагментов схем является очень трудоемкой задачей. Возможность локализовать и изолировать такие фрагменты за счет

разбиения исходных схем не только позволяет снизить сложность решаемой задачи, но и улучшить качество получаемых подсхем-заплаток.

Таким образом, решение задачи проверки эквивалентности и функциональной коррекции может быть упрощено за счет предварительного разбиения схем на соответствующие друг другу подсхемы меньшего размера. При этом нужно учитывать, что критерии, которыми нужно руководствоваться при выделении подсхем для указанных задач, различные. В случае решении задачи проверки эквивалентности ключевым фактором является сохранение эквивалентности получаемых соответствующих подсхем и минимизация их размеров. Для задачи функциональной коррекции основную роль играет выделение эквивалентных подсхем, которые покрывают максимальную часть исходных схем и тем самым позволяют минимизировать размер оставшихся неэквивалентных подсхем.

Изучению рассматриваемой задачи был посвящен целый ряд научных работ. Так, например, в [3] был предложен метод поиска разрезов на основе двоичных решающих диаграмм (BDD), которые порождают функционально эквивалентные соответствующие подсхемы, а в работе [4] описана система для эффективного поиска ECO для логических схем среднего размера. В свою очередь, в работе [5] методы поиска булевых соответствий были обобщены для поиска неточных соответствий и применены для повышения эффективности поиска решений задачи ECO. Наконец, в работе [6] представлена система, которая использует решение задачи выполнимости булевых формул (SAT) для выявления логически эквивалентных фрагментов схем, которые используются для последующей оптимизации решения задачи проверки эквивалентности исходных схем.

Поиск комплексного решения рассматриваемой задачи разбиения схем, направленного на повышение эффективности процедур проверки эквивалентности и функциональной коррекции, был положен в основу одной из задач [7] международного соревнования по разработке алгоритмов автоматизации проектирования СБИС «CAD Contest at ICCAD 2015» [8], предложенной компанией Cadence Design Systems, Inc. Данная статья посвящена описанию алгоритмов и соответствующей системы для решения сформулированной в рамках соревнования задачи разбиения схем. По результатам соревнования разработанная авторами система заняла первое место.

Статья имеет следующую структуру. В параграфе II приводится формальное описание задачи, сформулированной в рамках соревнования, а также даётся подробное описание тестового набора схем, использованного для оценки его результатов. Последующие параграфы описывают алгоритмы и общую структуру разработанной для решения задачи соревнования системы. В заключительном параграфе VI приводятся результаты тестирования предложенного решения задачи.

II. Соревнование ICCAD CAD Contest 2015

A. *Формальная постановка задачи*

Дадим формальное описание задачи разбиения схем [7], сформулированной в рамках международного соревнования [8]. В качестве входа задачи заданы две схемы из функциональных элементов (СФЭ) Σ' и Σ'', реализующие системы функций алгебры логики (ФАЛ) $F' = (f_1', ..., f_k')$ и $F'' = (f_1'', ..., f_k'')$, соответственно. При этом каждому входу Σ' взаимно однозначно сопоставлен вход Σ'', а каждому выходу Σ' взаимно однозначно сопоставлен выход Σ''. Указанные подмножества сопоставленных друг другу входов рассматриваемых схем будем называть *множествами отождествления (МО)*. В свою очередь, произвольное подмножество вершин СФЭ Σ' и Σ'' будем называть *множеством сравнения* (*МС*). При этом такое множество будем считать *эквивалентным*, если оно содержит не менее двух вершин и ФАЛ, реализуемые в вершинах этого множества, являются попарно равными (с учетом существующего сопоставления входов схем Σ' и Σ''). Считается, что изначально заданы k МС $\{f_i', f_i''\}$, $i = 1,\ldots,k$, которые содержат соответствующие выходы СФЭ Σ' и Σ'' (каждое из указанных МС может быть как эквивалентным, если соответствующие выходы СФЭ Σ' и Σ'' были эквивалентными, так и не эквивалентными в противном случае).

Пусть задана СФЭ Σ, а множество её вершин и ребер обозначены через V и E, соответственно. *Разрезом* C_e ориентированного ребра $e = (u, v)$, $e \in E$, СФЭ Σ будем называть пару вершин $C_e = (u', v')$ специального вида, которая получается в результате удаления из СФЭ Σ ребра e и добавления ребер (u, u') и (v', v). При этом разрез C_e в схеме Σ с одной стороны порождает новую выходную вершину u' (считается, что вершина u' реализует ту же ФАЛ, что и вершина u), а с другой стороны – новую входную вершину v', которой приписана новая входная переменная или ее отрицание (в этом случае разрез будем называть *инверсным*). Отметим, что ребра, введённые после разреза ребра СФЭ, также могут разрезаться.

Предполагается, что в результате решения рассматриваемой задачи часть ребер Σ' и Σ'' подвергается разрезам, множество которых затем разбивается на пересекающиеся подмножества (классы). При этом множество входов (выходов), полученных в результате разбиения СФЭ $\hat{\Sigma}'$ и $\hat{\Sigma}''$, распадается на порожденные введенными классами МО (соответственно МС). Множество всех МС, полученных таким образом для СФЭ $\hat{\Sigma}'$ и $\hat{\Sigma}''$, будем называть их *разрезающим множеством* (*РМ*).

Будем считать, что вершина v, $v \in V$ достижима из вершины u, $u \in V$, если в СФЭ Σ существует ориентированный (u,v) -путь. Для вершины v, $v \in V$ и заданного подмножества вершин $\overline{V}$, $\overline{V} \subseteq V$, из которых достижима вершина v, под *конусом* $K_v(\overline{V})$ будем понимать множество всех вершин СФЭ Σ (исключая вершины множества $\overline{V}$), которые лежат на ориентированных путях, соединяющих вершины множества $\overline{V}$ с вершиной v. Если множество $\overline{V}$ состоит из всех тех входных вершин, из которых вершина достижима v, то соответствующий конус будем называть *максимальным*. При этом мощность конуса $K_v(\overline{V})$ будем называть его *весом* а мощность множества $\overline{V}$ - *шириной* данного конуса. Отметим, что выходные вершины разрезов учитываются при подсчете весов конусов, в которые они попадают.

Результат работы алгоритма оценивается на основе сформированного РМ. Если оно содержит только эквивалентные МС, то такое РМ называется *эквивалентным*. Вес такого множества задается упорядоченным по убыванию вектором весов максимальных конусов, порожденных вершинами всех входящих в него МС. Если РМ не является эквивалентным, то его вес задается суммой весов максимальных конусов, порожденных всеми вершинами, входящими в неэквивалентные МС (при этом веса максимальных конусов вершин, которые попали в эквивалентные МС, не учитываются). Если РМ содержит только эквивалентные МС, то такое множество всегда оценивается выше любого РМ, содержащего хотя бы одно неэквивалентное МС.

B. Пример построения разбиения схем

Разберем несколько примеров построения разбиения схем для иллюстрации задачи [7], сформулированной в рамках соревнования. Рассмотрим СФЭ Σ и Σ', представленные на рис. 1 и заданные логическими формулами: $(ab \oplus (a \oplus b)) \oplus (b \mid c)$ и $(ab \vee (a \oplus b)) \sim bc$, соответственно (сопоставление входов и выходов схем определяется именами переменных).

Сначала рассмотрим случай, когда добавленные разрезы формируют только эквивалентные МС. На рис. 1 в СФЭ Σ и Σ' произведены разрезы ребер f, g, f' и g', при этом разрез ребра g' во второй схеме является инверсным. Указанные группы разрезов порождают новые эквивалентные МС (f, f') и (g, g') веса (4,4) и (2,2), соответственно. При этом не нарушается эквивалентность МС (o, o'), а его вес становится равным (1,1).

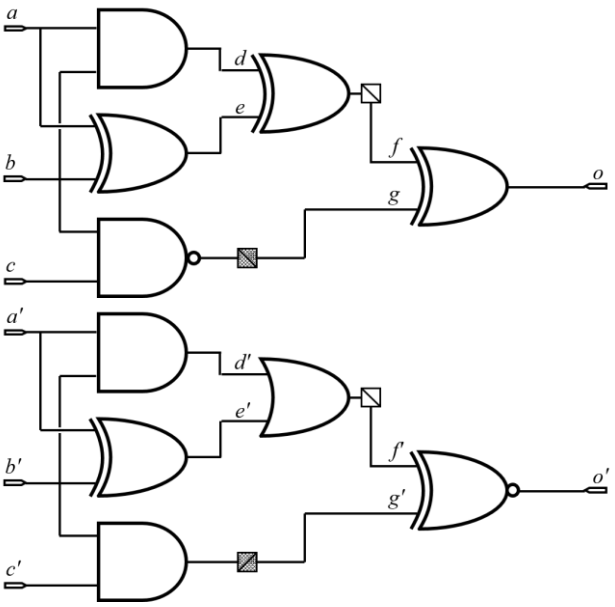

Рис. 1. Пример вставки разрезов, порождающих только эквивалентные МС

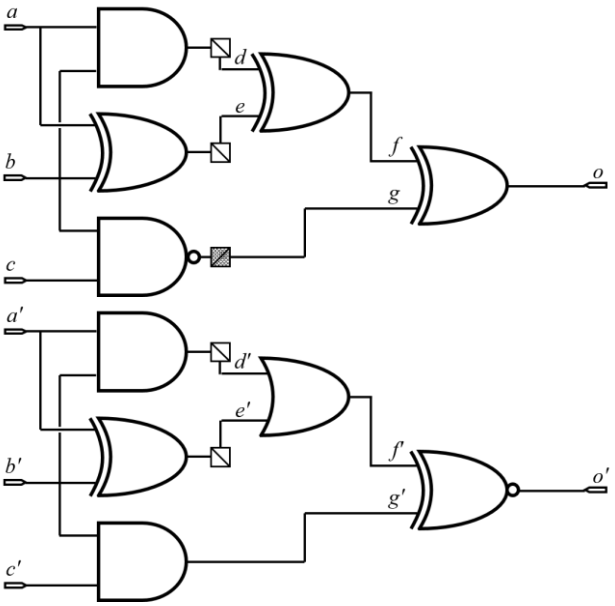

Рис. 2. Пример вставки разрезов, порождающих неэквивалентное МС

Теперь рассмотрим случай, когда после введения в схемы разрезов возникают неэквивалентные МС. Соответствующий пример приведен на рис. 2. В этом случае в первую схему вводятся разрезы d, e и g, а во вторую – d' и e'. При этом возникают новые МС (d, e, d', e') и (g) веса 8 и 2, соответственно, а МС (o, o') становится неэквивалентным множеством веса 5. Таким образом, решение, представленное на рис. 1, будет оцениваться выше, чем решение, представленное на рис. 2.

C. *Структура и особенности тестового набора схем*

Для проверки решений, представленных в рамках соревнования, его организаторами был разработан специальный набор из 22 пар тестовых схем. Каждая пара содержит описание части интегральной схемы, полученное на некоторых последовательных этапах логического синтеза.

Каждая схема представляет одномодульное описание комбинационной подсхемы некоторого цифрового устройства или его части на языке Verilog с использованием только примитивных функциональных элементов (ФЭ). Указанные описания были сгенерированы при помощи индустриального комплекса программ Genus Synthesizer компании Cadence Design Systems, Inc [9] и их размер варьируется от 7422 до 213304 ФЭ. При этом спецификации проектируемых устройств, которые использовались для генерации тестовых пар, были взяты из следующих источников: набор тестовых схем ITC 99 [10], блоки интеллектуальной собственности OpenCores [11] и различные модификации операционных автоматов (datapath), спроектированные организаторами соревнования.

Сам тестовый набор имеет следующую структуру: половина тестовых пар содержит схемы, которые задают эквивалентные МС, а вторая половина пар – схемы, которые содержат хотя бы одну пару неэквивалентных сопоставленных выходов. При этом тестовые пары из второй группы разделены на две подгруппы. В первой подгруппе неэквивалентность возникает за счет того, что в описание одной из схем была введена локальная ошибка (замена небольшой подсхемы). Во второй подгруппе описание второй схемы в каждой паре было получено в результате логического синтеза с использованием измененной спецификации проектируемого устройства. Заметим, что часть схем использовалась для скрытого тестирования и была обнародована только после объявления результатов соревнования.

III. Алгоритмы предварительного глобального разбиения схем

Если исходные схемы имеют большой размер и содержат неэквивалентные МС, то система производит предварительное разбиение схем на подсхемы, анализируя только структуру исходных схем. Исходя из текущих настроек системы, этот этап может быть пропущен и система переходит к разбиению исходных схем на основе алгоритмов, описанных в параграфе V.

Максимальным деревом для вершины v, $v \in V$ СФЭ Σ будем называть максимальный по весу конус, все вершины которого имеют исходящую степень, равную 1. В свою очередь, максимальный по весу конус $K_v(\overline{V})$ вершины v, все исходящие ребра вершин которого, идут в вершины, которые принадлежат $K_v(\overline{V})$, называется максимальным замкнутым конусом (англ. maximal fanout-free cone, MFFC). Разбиения СФЭ Σ на максимальные деревья и на максимальные замкнутые конусы являются единственными [12].

Разработанная система позволяет пользователю произвести разбиение исходных схем на максимальные деревья и на максимальные замкнутые конусы. При этом, как правило, исходные схемы разбиваются на большое число схем небольшого размера: основная часть подсхем имеет в среднем 4 или 5 входов и лишь небольшая часть подсхем имеет 20 и более входов.

В случае, когда предварительно удалось установить эквивалентность всех исходных МС заданных схем, разбиение не производится, так как это может нарушить эквивалентность МС.

IV. Алгоритмы поиска функциональных соответствий

На вход алгоритмов, описываемых в данном параграфе, подается набор подсхем, полученных в результате предварительного разбиения, в котором каждая подсхема порождает по определению свое неэквивалентное МС. Основная задача рассматриваемых алгоритмов выявить как можно больше эквивалентных подсхем. При этом, эквивалентными называются подсхемы, которые реализуют одинаковые наборы ФАЛ с точностью до перестановки и инвертирования входов, а также инвертирования самих ФАЛ (отметим, что возможность инвертирования входов и выходов ФАЛ обеспечивается за счет использования инверсных разрезов при разбиении схем). Задачу проверки пары схем на эквивалентность представленного типа будем называть задачей поиска функционального соответствия (англ. Boolean matching).

Указанная задача достаточно широко представлена в зарубежной литературе. При этом существуют эффективные алгоритмы, которые позволяют находить соответствия как для больших схем [13], так и для схем небольшого размера [14]. Опишем, как указанные алгоритмы используются в разработанной системе. Сначала все подсхемы разбиваются на группы согласно количеству их входов. Таким образом, поиск соответствий производится только между подсхемами, которые попали в одну группу. Для хранения информации о найденных эквивалентных подсхемах в рамках одной группы используется структура данных для непересекающихся множеств.

Если количество входов у подсхемы небольшое (не более 5), то используется алгоритм на основе подхода, предложенного в [14]. В данном случае, для каждого класса эквивалентности ФАЛ относительно рассматриваемых операций перестановки и инвертирования входных переменных ФАЛ, а также инвертирования значения самой ФАЛ, можно определить канонического представителя, вычисление

которого позволит автоматически определить класс, в который попадает ФАЛ, реализуемая выбранной подсхемой. В качестве такого представителя можно использовать лексикографически наименьший при фиксированном порядке следования переменных столбец значения среди всех ФАЛ из заданного класса эквивалентности. Эффективный алгоритм вычисления такого канонического представителя представлен в [14] и интегрирован в программу ABC [15]. Таким образом, алгоритм поиска соответствий в данном случае состоит из следующих этапов: по подсхеме вычисляется ФАЛ, реализуемая этой подсхемой, далее по полученной ФАЛ вычисляется канонический представитель соответствующего класса эквивалентности, в который попадает рассматриваемая ФАЛ. Если указанный представитель ранее не встречался, то создается новое множество в системе непересекающихся множеств и к нему привязывается рассматриваемая подсхема, в противном случае эта подсхема добавляется к уже существующему множеству.

Так как вычисление канонического представителя является наиболее трудоемкой задачей, то для обеспечения эффективности программы результаты этих вычислений сохраняются в специальной таблице. С учетом того, что число входов рассматриваемых схем ограничено сверху, можно считать, что на анализ одной такой подсхемы система тратит константное время. Таким образом, алгоритм поиска соответствий для данных классов будет работать линейно относительно числа подсхем, которые в них попали.

Для всех остальных классов описанный выше подход является слишком трудоемким из-за экспоненциального роста сложности вычисления канонического представителя и невозможности построить столбец значений для ФАЛ, реализуемой подсхемой. В этом случае применяются алгоритмы, представленные в работе [13] и реализованные в программе ABC [15], которые позволяют установить функциональное соответствие между ФАЛ, реализуемыми двумя подсхемами, если оно существует. При этом изначально считается, что каждая подсхема из заданного класса формирует свое отдельное множество в системе непересекающихся множеств и выбирается в качестве канонического представителя этого множества. В процессе работы алгоритм последовательно выбирает множество из системы непересекающихся множеств и проверяет его канонического представителя на существование функционального соответствия с каноническими представителями других множеств в системе непересекающихся множеств. При обнаружении соответствия найденные множества объединяются, и рассматриваемая подсхема выбирается в качестве канонического представителя их объединения. Алгоритм совершает поиск соответствий до тех пор, пока происходят изменения в структуре системы непересекающихся множеств.

V. Алгоритмы локального разбиения схем

После первоначального этапа глобального разбиения схем и поиска функциональных соответствий для полученных подсхем проводится локальный поиск разрезов, который позволяет, с одной стороны, уменьшить размер конусов, входящих в эквивалентные МС, с другой – уменьшить суммарный вес конусов вершин, которые попали в неэквивалентные МС. Для этих целей в системе реализовано два различных алгоритма, которые отличаются временем работы и качеством получаемых разрезов. При этом один из предложенных алгоритмов гарантирует сохранение эквивалентности получаемых МС, если исходные схемы задавали эквивалентные МС. Стоит также отметить, что в зависимости от размерности задачи указанные алгоритмы можно применять как к исходно заданным схемам (не производя предварительного разбиения схемы), так и к подсхемам, полученным после этапа предварительного разбиения. В последнем случае последовательно выбираются пары подсхем, к которым применяется один из предложенных далее алгоритмов в зависимости от структуры самих подсхем и их МС.

A. Алгоритм разбиения на отдельные элементы

На вход алгоритма подаются две СФЭ Σ_1 и Σ_2, выходы которых задают некоторый набор МС. При этом считается, что указанный набор в основном содержит неэквивалентные МС. На первом шаге алгоритма СФЭ Σ_1 разбивается на отдельные ФЭ. Фактически разрезы вставляются во все ребра СФЭ Σ_1. На следующем шаге для каждого элемента СФЭ Σ_1 производится поиск соответствующего ФЭ в схеме Σ_2 и введение соответствующих разрезов для входящих и исходящих ребер найденного элемента. При этом ребра найденного ФЭ дополнительно анализируются на наличие конфликтов разрезов (если этот элемент или его соседние элементы уже были сопоставлены и их входящие или исходящие ребра были разрезаны) для того, чтобы избежать введения лишних фиктивных разрезов (не содержащих ФЭ). Для обеспечения эффективности поиска таких ФЭ используется специальная структура данных, которая позволяет хранить информацию об одноэлементных конусах всех внутренних вершин СФЭ Σ_2. ФЭ СФЭ Σ_1, для которых не удается найти соответствующий ФЭ в СФЭ Σ_2, формируют одноэлементные неэквивалентные МС. Кроме того, за счет введения разрезов во входящих ребрах ФЭ СФЭ Σ_2 могут возникнуть дополнительные неэквивалентные одноэлементные МС.

Предложенный алгоритм имеет линейное относительно числа вершин в СФЭ Σ_1 и Σ_2 время работы, но при этом не гарантирует нахождение оптимального набора разрезов. Стоит также отметить,

что возникающие в процессе работы алгоритма одноэлементные МС могут быть дополнительно проанализированы с точки зрения их возможного сопоставления.

B. Жадный алгоритм разбиения с сохранением эквивалентности

На вход алгоритма подаются две СФЭ Σ_1 и Σ_2, выходы которых задают некоторый набор эквивалентных МС. Алгоритм производит поиск функционально эквивалентных подсхем, отрезание которых заведомо не нарушает эквивалентности существующих МС. Для этого используется следующее наблюдение, которое легко доказывается от противного. Пусть $\hat{\Sigma}_1$ и $\hat{\Sigma}_2$ - найденные эквивалентные подсхемы в СФЭ Σ_1 и Σ_2, соответственно, и входы указанных подсхем образуют подмножество входов исходных схем, а сами подсхемы реализуют систему ФАЛ $F = (f_1,..., f_k)$. Тогда СФЭ $\breve{\Sigma}_1$ и $\breve{\Sigma}_2$, получающиеся из Σ_1 и Σ_2 после удаления подсхем $\hat{\Sigma}_1$ и $\hat{\Sigma}_2$, будут эквивалентными, если любой набор значений $\tilde{\alpha} = (\alpha_1,..., \alpha_k)$, $\tilde{\alpha} \in \{0,1\}^k$ на выходах удаленных подсхем $\hat{\Sigma}_1$ и $\hat{\Sigma}_2$ реализуется при подаче некоторого набора значений на их входы.

Таким образом, алгоритм состоит из следующих этапов. Сначала алгоритм строит топологический порядок на множестве вершин каждой из СФЭ Σ_1 и Σ_2. Далее, используя построенные топологические порядки, алгоритм ищет пару эквивалентных подсхем, которая удовлетворяет необходимому условию сохранения эквивалентности (эквивалентность подсхем устанавливается при помощи алгоритмов, описанных в параграфе V). Если такая пара найдена, то она отрезается от исходных схем и алгоритм переходит к поиску следующей пары. Так как размер подсхем очень сильно влияет на скорость проверки реализуемости всех возможных наборов на их выходах, то размер подсхем при их поиске ограничен сверху некоторой положительной константой.

Предложенный алгоритм также имеет малое время работы, но при этом выделяет лишь небольшое число эквивалентных подсхем малого размера. Это связано с тем, что предложенное необходимое условие – сохранение эквивалентности на каждом этапе разбиения схем и отрезание подсхем только со стороны входов исходных схем – сильно ограничивает множество выделяемых подсхем.

VI. Результаты тестирования

Для проверки качества получаемого в результате добавления разрезов разбиения исходных схем была разработана специальная программа-валидатор, которая позволяет установить является ли заданное МС эквивалентным или нет, используя средства программы ABC [15], а также рассчитать стоимость получаемого решения. Кроме того, программа проверяет синтаксическую корректность Verilog описания полученного решения.

Таблица 1

Результаты тестирования для пар схем, содержащих неэквивалентные МС

Имя	Число входов	Число выходов	Число ФЭ в Σ_1	Число ФЭ в Σ_2	Суммарный вес неэквивалентных МС					Относительное сокращение веса неэквивалентных МС			
					Б.р.	A	B	C	D	A	B	C	D
ut2	249	914	13876	10064	503826	318661	470050	41550	33819	36,6%	6,7%	91,8%	93,3%
ut4	1364	5397	78169	52219	1064521	832430	988829	230280	167985	21,8%	7,1%	78,4%	84,2%
ut8	2282	2726	51439	91236	14582834	3557143	14577666	267054	125682	75.6%	0,0%	98,2%	99.1%
ut9	650	2474	59886	48061	1363736	400296	1174440	188662	111966	70,75%	13,9%	86,4%	91,8%
ut11	56	129	14409	14600	1524769	86813	1513138	57660	20581	94,3%	0,8%	96,2%	98,7%
ut13	99	128	28993	28052	2590770	271694	2483103	113618	57010	89,6%	4,2%	95,6%	97,8%
ut15	99	128	7323	17572	1516255	1246862	1516188	49356	26566	17,8%	0,0%	96,7%	98,3%
ut17	128	256	82408	46395	9029081	3021783	8856532	256658	91325	66,5%	1,9%	97,2%	99,0%
ut19	160	385	147784	74618	NA	60868791	24821951	NA	199749	NA	NA	NA	NA
ut21	80	288	213224	200217	48463859	3051084	48463162	48463859	194477	93,7%	0,0%	0,0%	99,6%
ut23	80	192	94707	101754	NA	476021	NA	NA	100934	NA	NA	NA	NA
									Среднее значение по всем тестам	63,0%	3,8%	82,2%	95,8%

Таблица 2

Результаты тестирования для пар схем, содержащих только эквивалентные МС

Имя	Число входов	Число выходов	Число ФЭ в Σ_1	Число ФЭ в Σ_2	Максимальный вес максимального входного конуса		
					A	B	C
ut1	249	914	13877	10064	4675	4124	3240
ut3	1364	5397	78169	52219	2247	1996	2004
ut5	650	2474	59886	48061	7494	7043	7361
ut7	2282	2726	51439	91236	8134	8020	8134
ut10	56	129	10083	14409	13607	13610	13611
ut12	99	128	17498	28993	27748	27053	27748
ut14	99	128	7323	16643	15794	15793	15794
ut16	128	256	41474	82480	76931	76910	76931
ut18	160	385	74889	147784	108158	NA	108158
ut20	80	288	78181	213224	208487	208486	208488
ut22	80	192	87239	94707	90673	90672	90673

Результаты тестирования пар тестовых схем, содержащих хотя бы одно неэквивалентное МС, представлены в таблице 1. В свою очередь, результаты тестов для схем, которые содержат только эквивалентные множества сравнений, представлены в таблице 2. Столбец «A» в указанных таблицах представляет авторское решение, которое заняло первое место по результатам соревнования, столбцы «B» и «C» представляют результаты команд [8], которые заняли второе и третье места, соответственно, а столбец «D» представляет результаты авторского решения задачи, полученное в результате исследований, проведенных после завершения соревнования. Отметим, что в силу сложности решения поставленной задачи и строгих временных ограничений на время, отведенное на получение решения, не во всех случаях участникам соревнования удавалось получить решение. Такие ситуации в таблице отмечены пометкой «NA». Рассматриваемые таблицы также содержат более подробную информацию о структуре тестовых схем (число входов, число выходов, количество ФЭ). Кроме того, в таблице 1 представлена стоимость решения без введения дополнительных разрезов (столбец «**Б.р.**»), которая использовалась для того, чтобы посчитать относительное сокращение суммарного веса всех неэквивалентных разрезов, получаемое в результате решения задачи (для тестов **ut19** и **ut21** это значение установить не удалось из-за сложности решения задачи проверки эквивалентности схем).

Из представленных результатов можно сделать вывод, что в случае, когда схемы содержат только эквивалентные МС, лучший результат показала команда, занявшая второе место. В случае, когда исходные схемы содержат неэквивалентные МС, в рамках соревнования лучший результат показала команда, занявшая третье место. В свою очередь, авторское решение заняло первое место, так как показало более стабильные результаты по всей совокупности тестов. Кроме того, решение, полученное с использованием доработанного авторами алгоритма (столбец «**D**»), показывает наилучшие результаты для схем, содержащих неэквивалентные МС, и улучшает результаты, представленные в статье [5].

Благодарности

Авторы статьи выражают благодарность Игорю Леонидовичу Маркову, Сергею Андреевичу Ложкину и Дмитрию Сергеевичу Романову за проявленный интерес к работе, ценные замечания и предложения во время ее подготовки.

Литература

[1] Kuehlmann A., Krohm F. Equivalence checking using cuts and heaps // DAC. 1997. P. 263–268.

[2] Lin C.-C., Chen K.-C., Chang S.-C., Marek-Sadowska M., Cheng K.-T. Logic synthesis for engineering change // DAC. 1995. P. 647–652.

[3] Khasidashvili Z., Moondanos J., Kaiss D., Hanna Z. An enhanced cut-points algorithm in formal equivalence verification // HLDVT. 2001. P. 171–176.

[4] Krishnaswamy S., Ren H., Modi N., Puri R. DeltaSyn: An efficient logic-difference optimizer for ECO synthesis // ICCAD. 2009. P. 789–796.

[5] Huang S.-L., Lin W.-H., (Ric) Huang C.-Y. Match and replace: a functional ECO engine for multi-error circuit rectification // ICCAD. 2011. P. 383–388.

[6] Long J., Brayton R.K., Case M.L. LEC: Learning driven data-path equivalence checking. DIFTS @FMCAD. 2013.

[7] Chih-Jen (Jacky) Hsu Large-Scale Equivalence Checking and Function Correction. URL: http://cad-contest.el.cycu.edu.tw/problem_B/default.htm (дата обращения: 04.04.2016).

[8] 2015 CAD Contest at ICCAD. URL: http://cad-contest.el.cycu.edu.tw/CAD-contest-at-ICCAD2015/index.html (дата обращения: 04.04.2016).

[9] Genus Synthesis Solution. URL: http://www.cadence.com/products/ld/genus (дата обращения: 04.04.2016).

[10] Corno F., Reorda M.S., Squillero G., RT-level ITC'99 benchmarks and first ATPG results // IEEE Design & Test of Computers. 2000. V. 17, № 3, P. 44–53.

[11] OpenCores. URL: http://opencores.org/ (дата обращения: 04.04.2016).

[12] Cong J., Ding Y. Combinational logic synthesis for LUT based field programmable gate arrays // ACM Trans. Des. Autom. Electron. Syst. 1, 2. 1996. P. 145–204.

[13] Katebi H., Markov I.L. Large-scale Boolean matching // Design, Automation & Test in Europe Conference & Exhibition (DATE). 2010. P. 771–776.

[14] Huang Z., Wang L., Nasikovskiy Y., Mishchenko A. Fast Boolean matching based on NPN classification // ICFPT, 2013. P. 310–313.

[15] Berkeley Logic Synthesis and Verification Group. ABC: A System for Sequential Synthesis and Verification. http://www.eecs.berkeley.edu/~alanmi/abc/ (дата обращения: 04.04.2016).

Partitioning methods for Large-Scale Equivalence Checking and Function Correction

G.V.Antiufeev[1], V.V. Zhukov[1], E.Y. Zenin[2], M.S.Shupletsov[1]

[1]Lomonosov Moscow State University, Faculty of Computational Mathematics and Cybernetics, grigoriy.rus@gmail.com, shupletsov@cs.msu.ru

[2]INEUM, euzenin@gmail.com

***Keywords* — Boolean circuits, logic synthesis, partitioning, equivalence checking, function correction, engineering change order, Boolean matching.**

ABSTRACT

Efficient equivalence checking and functional correction on large-scale designs are crucial technologies for handling today's demanding design cycles. It is well known that partitioning the design, based on two designs' correspondence, can significantly reduce the complexity of the analysis. In the international contest «2015 CAD Contest at ICCAD» held within the scope of International Conference on Computer-Aided Design 2015 (ICCAD) the research in the area of partitioning large designs into corresponding smaller designs that are easier to analyze was inspired. Contestants were challenged to insert cut points on large designs in an effort to reduce their equivalence checking and function correction problems. Equivalence checking (EC) and functional correction (ECO) are both used to analyze the Boolean difference between design specification and its logic implementation. For two processes (solving EC and ECO) the focus was on isolating large designs into corresponding smaller instances but with two different objectives. EC divides the large instances into smaller equivalent ones and cannot tolerate any non-equivalent sub-instances. ECO isolates the large instance into root-cause of changes and minimizes the non-equivalent sub-instance. Having effective circuit's partitioning methods gives the possibility to address both the equivalence checking and functional correction problems in practice.

In this paper the authors describe an approach to design partitioning that took the first place in «2015 CAD Contest at ICCAD». Furthermore, different modifications are presented, which significantly improve the overall quality of the present approach.

ACKNOWLEDGEMENTS

Authors would like to acknowledge gratitude to I.L. Markov, S.A. Lozhkin and D.S. Romanov for insightful suggestions and valuable remarks during the course of this paper's preparation.

REFERENCES

[1] Kuehlmann A., Krohm F. Equivalence checking using cuts and heaps // DAC. 1997. P. 263–268.

[2] Lin C.-C., Chen K.-C., Chang S.-C., Marek-Sadowska M., Cheng K.-T. Logic synthesis for engineering change // DAC. 1995. P. 647–652.

[3] Khasidashvili Z., Moondanos J., Kaiss D., Hanna Z. An enhanced cut-points algorithm in formal equivalence verification // HLDVT. 2001. P. 171–176.

[4] Krishnaswamy S., Ren H., Modi N., Puri R. DeltaSyn: An efficient logic-difference optimizer for ECO synthesis // ICCAD. 2009. P. 789–796.

[5] Huang S.-L., Lin W.-H., (Ric) Huang C.-Y. Match and replace: a functional ECO engine for multi-error circuit rectification // ICCAD. 2011. P. 383–388.

[6] Long J., Brayton R.K., Case M.L. LEC: Learning driven data-path equivalence checking. DIFTS @FMCAD. 2013.

[7] Chih-Jen (Jacky) Hsu Large-Scale Equivalence Checking and Function Correction. URL: http://cad-contest.el.cycu.edu.tw/problem_B/default.htm (accessed: 04.04.2016).

[8] 2015 CAD Contest at ICCAD. URL: http://cad-contest.el.cycu.edu.tw/CAD-contest-at-ICCAD2015/index.html (accessed: 04.04.2016).

[9] Genus Synthesis Solution. URL: http://www.cadence.com/products/ld/genus (accessed: 04.04.2016).

[10] Corno F., Reorda M.S., Squillero G., RT-level ITC'99 benchmarks and first ATPG results // IEEE Design & Test of Computers. 2000. V. 17, № 3, P. 44–53.

[11] OpenCores. URL: http://opencores.org/ (accessed: 04.04.2016).

[12] Cong J., Ding Y. Combinational logic synthesis for LUT based field programmable gate arrays // ACM Trans. Des. Autom. Electron. Syst. 1, 2. 1996. P. 145–204.

[13] Katebi H., Markov I.L. Large-scale Boolean matching // Design, Automation & Test in Europe Conference & Exhibition (DATE). 2010. P. 771–776.

[14] Huang Z., Wang L., Nasikovskiy Y., Mishchenko A. Fast Boolean matching based on NPN classification // ICFPT, 2013. P. 310–313.

[15] Berkeley Logic Synthesis and Verification Group. ABC: A System for Sequential Synthesis and Verification. http://www.eecs.berkeley.edu/~alanmi/abc/ (accessed: 04.04.2016).