УДК 541.186

ИНФРАСТРУКТУРА ДЛЯ ИССЛЕДОВАНИЯ НЕКОТОРЫХ АЛГОРИТМОВ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ СБИС

© 2010 г. ***А.В. Живодеров, А.М. Камаев, К.В. Корняков, И.Б. Мееров***

Нижегородский госуниверситет им. Н.И. Лобачевского

meerov@vmk.unn.ru

Поступила в редакцию 23.03.2010

Рассматривается исследовательская инфраструктура, предоставляющая средства разработки, отладки и апробации алгоритмов физического проектирования сверхбольших интегральных схем (СБИС), таких как размещение, трассировка, вставка буферов и т.д. Анализируется состояние области, описываются предлагаемая инфраструктура, ключевые идеи и детали реализации. Возможности разработанной инфраструктуры иллюстрируются примерами применения для проведения исследований в области САПР СБИС.

Ключевые слова: САПР, СБИС, физическое проектирование, маршрут проектирования СБИС, инфраструктура.

Введение

В настоящее время в области автоматизированного проектирования СБИС известно множество актуальных задач, требующих решения. По мере уменьшения размеров элементной базы (технологического процесса) все большее значение приобретает разработка и совершенствование алгоритмов физического проектирования. Вызвано это как ростом размерности решаемых задач, так и необходимостью учитывать дополнительные физические эффекты при ориентации на глубоко субмикронные технологии [1].

Маршрут проектирования определяет этапы проектных процедур, используемых на всех стадиях разработки СБИС. Проектирование СБИС представляет собой сложный многостадийный процесс, на каждом этапе которого применяется большое количество моделей, методов и алгоритмов.

В настоящее время существует множество реализаций маршрутов проектирования СБИС, варьирующихся от полнофункциональных промышленных САПР до свободно доступных (иногда в исходных кодах) реализаций отдельных алгоритмов проектирования [2]. Так, каждая крупная компания, занимающаяся производством СБИС (Intel, AMD и т.д.), имеет собственную реализацию маршрута проектирования. Чаще всего подобные САПР сохраняются в секрете и недоступны для приобретения и использования. Наряду с закрытыми промышленными САПР существуют системы, доступные для приобретения. Широко известны системы компаний Cadence и Synopsys, однако стоимость полного комплекта ПО может достигать нескольких миллионов долларов, что делает подобные разработки недоступными для использования в академических и исследовательских организациях. Кроме коммерческих, существует некоторое количество свободно доступных САПР и отдельных утилит, которые пригодны для промышленного проектирования СБИС, стоит отметить такие ресурсы, как Open EDA Tools, Open Circuit Design и VLSI CAD Bookshelf [1]. Свободно доступные инструменты обладают разной степенью зрелости и стабильности, но даже несмотря на это, подобные разработки чаще всего неудобны для проведения НИР. Некоторые из них закрыты для модификации или крайне сложны для освоения и последующей поддержки. В других случаях требуется наличие выделенного сервера со специализированной БД, хранящей состояние СБИС, и исследователи оказываются в большой степени привязаны к нему. Как следствие, затрудняется распределенная разработка, и, кроме того, сервер становится узким местом при массовом проведении экспериментов. Эти обстоятельства обусловили то, что исследователи отдельных алгоритмов проектирования создают обособленные утилиты, решающие одну конкретную задачу. Последний подход также имеет недостатки, и прежде всего тот, что при исследовании нескольких алгоритмов приходится повторно решать задачи хранения данных и реализации базовых алгоритмов их обработки.

Таким образом, задача создания исследовательской инфраструктуры для разработки, апробации и настройки отдельных алгоритмов и экспериментальных маршрутов проектирования СБИС является достаточно актуальной. Приме-

нение такой инфраструктуры позволит исследователю комплексно опробовать новые идеи, реализованные в виде программного кода. Например, при разработке нового алгоритма, используемого на одном из начальных этапов маршрута проектирования, можно будет оценить его влияние на конечный результат, подключив его к уже имеющимся в инфраструктуре компонентам. Основной эффект ожидается от экономии времени на интеграции разрозненных утилит в целостный маршрут проектирования за счет использования инфраструктуры с уже готовым решением таких общих вопросов, как хранение исходных, промежуточных и результирующих данных, реализация базовых операций, а также предоставление средств автоматизации процесса отладки и проведения экспериментов.

В данной работе описывается реализованная авторами исследовательская инфраструктура и примеры маршрутов проектирования, созданных с ее использованием. Приведены результаты некоторых вычислительных экспериментов.

Требования к инфраструктуре

Разработанная инфраструктура ориентирована на стадии физического проектирования СБИС. Ее назначением является предоставление базовых средств для разработки и исследования методов повышения производительности СБИС при использовании сотен тысяч стандартных элементов и современного технологического процесса (авторами использовались 45- и 180-нанометровые библиотеки элементов).

Изначально к инфраструктуре предъявлялись следующие требования:

1. *Поддержка современных индустриальных форматов представления СБИС.* Это требование связано с тем, что наибольший интерес представляют интегральные схемы с большим числом элементов и современные технологические процессы.

2. *Возможность добавления новых алгоритмов.* Инфраструктура должна быть открыта для реализации новых алгоритмов физического проектирования и вставки их в маршрут проектирования.

3. *Возможность компоновки различных маршрутов проектирования.* Инфраструктура должна позволять проводить настройку алгоритмов, выполняющихся на конкретных стадиях, задание набора стадий и порядка их следования.

4. *Возможность автоматизированного проведения большого числа вычислительных экспериментов и обработки их результатов.* Как правило, после разработки и реализации алгоритма для определения оптимальных значений его параметров необходимо провести серию экспериментов. Всю полученную информацию необходимо представить исследователю в виде,

Рис. 1. Архитектура инфраструктуры

удобном для анализа и последующего принятия решений.

Архитектура

На рис. 1 представлена общая архитектура инфраструктуры.

Условно можно выделить четыре уровня:

1. *Уровень представления данных.* Нижний уровень системы отвечает за хранение данных, используемых в процессе проектирования (информация о соединениях, элементах, их физических характеристиках и т. д.).

2. *Интерфейсный уровень.* Код, относящийся к данному уровню, работает непосредственно с низкоуровневым представлением данных. Он представляет собой набор объектно-ориентированных интерфейсов для работы с данными СБИС. Также к этому уровню относится набор базовых алгоритмов САПР СБИС, например таких как подсчет длины соединений (half-perimeter wire length, HPWL) и вычисление суммарной емкостной нагрузки, оптимизированных для максимально эффективного использования особенностей хранения данных.

3. *Уровень алгоритмов проектирования.* Основной уровень инфраструктуры, состоящий из реализаций прикладных алгоритмов. Включает в себя парсеры входных форматов, примитивные алгоритмы проверки легальности размещения и целостности данных, а также непосредственно алгоритмы физического проектирования, такие как размещение, легализация, трассировка, статический временной анализ (СВА) и другие. На данном уровне выполняется расширение инфраструктуры новыми алгоритмами.

4. *Уровень компоновки маршрутов проектирования.* Инструментарий данного уровня служит для того, чтобы выстраивать прикладные алгоритмы предыдущего уровня в соответствии с исследуемым маршрутом проектирования. Отличительной особенностью разработанной инфраструктуры является то, что маршруты проектирования выстраиваются в рамках единого исполняемого модуля, что освобождает разработчиков от необходимости написания вспомогательных процедур импорта/экспорта данных для взаимодействия между составляющими маршрут алгоритмами.

Отдельно от четырех уровней следует рассмотреть четыре служебные подсистемы, которые облегчают процесс разработки, отладки и исследования алгоритмов проектирования:

• *Система конфигурирования* – предоставляет гибкий механизм задания параметров маршрута проектирования. Позволяет производить полную настройку как отдельных алгоритмов, так и совокупного маршрута.

• *Система визуализации* – набор методов визуализации, используемых для наглядного графического отображения хода работы алгоритмов.

• *Система логирования* – предоставляет набор методов для сохранения на жесткий диск промежуточных результатов, отладочной информации и других технических деталей. Позволяет сохранять информацию трех типов: текстовая информация, состояние СБИС в DEF-формате и графическая информация в виде изображений и видео.

• *Система проведения экспериментов* – наиболее новая из всех систем, на момент написания статьи представляет собой набор сценариев на языке программирования Python, позволяющих автоматически производить массовые запуски вычислительных экспериментов, первичную обработку результатов и получение удобных для анализа таблиц по результатам многих запусков.

Программная реализация

Разработка ведется на ОС Windows, в среде Microsoft Visual Studio 2005, на языке программирования C++. Реализованная инфраструктура сочетает в себе объектно-ориентированный программный интерфейс и использование массивов для представления данных на самом нижнем уровне. Такой подход обеспечивает удобство разработки алгоритмов проектирования и в то же время позволяет эффективно проводить эксперименты со СБИС, содержащими сотни тысяч элементов. Для организации вычислительных экспериментов и последующего сбора и анализа результатов используются сценарии на языке программирования Python версии 3.1. На момент написания данной статьи общий объем кода инфраструктуры составляет 209 000 строк, из которых 29 000 строк собственного программного кода и 180 000 строк стороннего.

Основной исполняемый файл представляет собой консольное приложение, которое реализует требуемый маршрут проектирования, получая на вход данные о СБИС и сохраняя все результаты в файлы требуемого формата. На консольное окно во время работы выводится статистическая и отладочная информация о работе каждой стадии маршрута проектирования. Также при необходимости создается дополнительное окно, на которое в графической форме в реальном времени выводится информация о текущем состоянии проектируемой СБИС.

Реализована поддержка следующих форматов представления информации о СБИС:

• Cadence LEF/DEF Exchange Format. Design Exchange Format (DEF) – открытая спецификация для представления физического состояния интегральной схемы. Описывает список соединений и размещение схемы. Используется совместно с Library Exchange Format (LEF)[2].

• Liberty Library Modeling Format. Используется для представления параметров библиотеки элементов: определение функций элементов, их временных характеристик и параметров питания (NLDM-модель)[3].

В процессе построения исследовательской инфраструктуры активно использовались программные модули сторонних разработчиков. На данный момент в инфраструктуре использованы следующие компоненты:

• Системные библиотеки: STL**,** libconfig, fastdelegate, OpenCV[4].

• Математические библиотеки: TAO (включая PETSc, BLAS), GLPK[5].

• Компоненты САПР СБИС: парсеры LEF/DEF и Liberty.

Отдельные системы инфраструктуры

Система конфигурирования

Для реализации была использована программная библиотека libconfig. Конфигурационные файлы представляют собой текстовые файлы с С-подобным синтаксисом представления настроек. Это, с одной стороны, позволяет разработчикам на основе единого файла быстро сконфигурировать инфраструктуру, а с другой стороны, существует возможность задания файла настроек сценариями на языке программирования Python, что необходимо для проведения экспериментов. Настройки конфигурационного файла могут быть переопределены через параметры командной строки, что также используется во время проведения массовых запусков при поиске оптимальных параметров работы алгоритмов.

Система позволяет на уровне конфигурационного файла задать различные параметры отдельных алгоритмов в зависимости от того, в каком контексте он используется. Так, например, легализация после стадии глобального размещения и во время его выполнения (это может использоваться для оценки качества) может иметь различные конфигурационные параметры.

Система визуализации

Визуализация работы алгоритмов позволяет уже на ранних стадиях разработки выявить проблемы, которые сложно обнаружить на основании числовых характеристик. Особенно актуально это для процесса размещения, где иногда лишь наблюдение за перемещением элементов может дать информацию о недостатках алгоритма.

Визуализация в реальном времени процесса проектирования СБИС, содержащих тысячи элементов, – весьма трудоемкая процедура, оказывающая существенную нагрузку на ресурсы компьютера. В реализованной инфраструктуре используется OpenCV – библиотека алгоритмов компьютерного зрения, обладающая достаточно высокой производительностью отображения графических примитивов.

Система логирования

Все экспортируемые файлы (текстовые с отладочной и статистической информацией, DEF-файлы с описанием состояния СБИС, изображения и видео) маркируются временем запуска вычислительного эксперимента и сохраняются в специальные директории. Это позволяет впоследствии идентифицировать и отобрать для анализа все материалы, относящиеся к конкретному эксперименту.

Также реализована возможность сохранения изображений системы визуализации в форме как отдельных изображений (для сохранения состояния СБИС после работы ключевых стадий), так и видеофайлов (например, для последующей демонстрации динамики процесса размещения). Эта функциональность полезна во время проведения массовых экспериментов, когда исследователь не имеет возможности отслеживать процесс проектирования в реальном времени. Так, например, иногда необходимо просмотреть видеофайлы после серии ночных экспериментов, особенно если алгоритмы дали сбой и требуется произвести отладку.

Система проведения экспериментов

Для автоматизации проведения вычислительных экспериментов и обработки результатов выбран язык программирования Python. Это решение является в достаточной степени устоявшейся практикой среди исследователей.

Сценарии используются для проведения регулярных ночных тестовых запусков. Сценарии загружают последнюю версию исходных кодов с сервера системы контроля версий, производят сборку системы и осуществляют прогон основных алгоритмов на тестовом наборе СБИС. Полученные результаты рассылаются по электронной почте, благодаря чему разработчики могут отслеживать эффект от последних внесенных изменений.

Примеры использования

Разработанная инфраструктура была использована авторами для экспериментальной проверки нескольких алгоритмов проектирования. Ниже будут рассмотрены два примера: задача размещения элементов СБИС и задача вставки буферов. В обоих случаях были сконструированы маршруты физического проектирования СБИС, в рамках которых проверялась эффективность алгоритмов.

Первый опробованный алгоритм – многоуровневый аналитический алгоритм размещения, целью которого является минимизация длины соединений СБИС, оцениваемой с помощью метода полупериметра (HPWL-метрика). Данная разработка является собственной реализацией алгоритма APlace [3]. Ядром алгоритма является много-

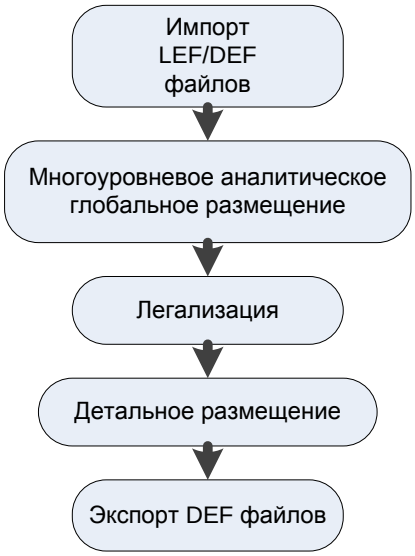

Рис. 2. Маршрут M1 проверки эффективности алгоритма размещения

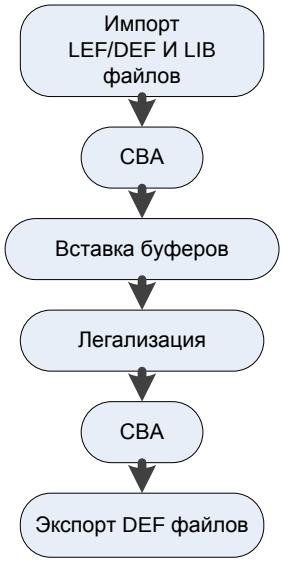

Рис. 3. Маршрут M2 проверки эффективности алгоритма вставки буферов

Таблица 1

Результаты работы алгоритма размещения элементов СБИС, маршрут M1

СБИС	Число элементов	Финальная длина соединений (HPWL-метрика)					
		M1	Dragon 3.01	Dragon 4.00	mPL 6.0	Capo 10.1	FengShui 5.1
ibm01	12506	1.63E+06	1.69E+06	-	1.62E+06	1.75E+06	1.81E+06
ibm02	19342	3.52E+06	3.65E+06	-	3.59E+06	3.71E+06	3.73E+06
ibm03	22853	4.65E+06	4.68E+06	-	4.73E+06	4.76E+06	4.71E+06
ibm04	27220	5.52E+06	5.66E+06	5.93E+06	5.80E+06	5.87E+06	5.89E+06
ibm05	28146	9.51E+06	9.67E+06	-	9.32E+06	9.77E+06	9.91E+06
ibm06	32332	4.94E+06	4.90E+06	5.25E+06	4.85E+06	5.13E+06	5.10E+06
ibm07	45639	8.18E+06	8.31E+06	8.68E+06	8.07E+06	8.78E+06	9.07E+06
ibm08	51023	8.72E+06	8.80E+06	9.60E+06	8.88E+06	9.27E+06	9.37E+06
ibm09	53110	9.21E+06	9.83E+06	-	9.21E+06	9.76E+06	9.91E+06
ibm10	68685	1.76E+07	1.75E+07	-	1.67E+07	1.83E+07	1.85E+07
ibm11	70152	1.37E+07	1.41E+07	1.48E+07	1.38E+07	1.45E+07	1.47E+07
ibm12	70439	2.18E+07	2.27E+07	-	2.17E+07	2.32E+07	2.38E+07
ibm13	83709	1.68E+07	1.75E+07	1.80E+07	1.67E+07	1.79E+07	1.81E+07
ibm14	147088	3.10E+07	3.18E+07	3.31E+07	3.04E+07	3.30E+07	3.31E+07
ibm15	161187	3.77E+07	3.95E+07	4.12E+07	3.79E+07	4.01E+07	4.12E+07
ibm16	182980	4.23E+07	4.38E+07	-	4.26E+07	4.58E+07	4.60E+07
ibm17	184752	6.25E+07	6.41E+07	-	5.92E+07	6.49E+07	6.60E+07
ibm18	210341	3.91E+07	4.16E+07	4.37E+07	4.02E+07	4.30E+07	4.27E+07

уровневая кластеризация множества элементов схемы с последующим решением задачи нелинейного программирования на каждом ее уровне. Так, на последнем уровне кластеризации оптимизация происходит для всего множества элементов. Для решения задачи нелинейного программирования использован метод оптимизации Limited-Memory Variable-Metric, реализованный в библиотеке TAO. По окончании рассмотренного этапа производится легализация полученного размещения (реализован алгоритм Abacus [4]). Далее производится детальное размещение, использующее различные идеи и подходы, описанные в работах [5, 6].

В качестве набора тестовых задач (СБИС) использован пакет ISPD04[6]. Сконструированный для апробации алгоритма маршрут (M1) представлен на рис. 2. Результаты вычислительных экспериментов приводятся в таблице 1. Таблица содержит сравнение полученных результатов с результатами ведущих академических инструментов размещения. Лучшие результаты на каждой тестовой СБИС выделены серым цветом.

Отметим, что алгоритм размещения, реализованный на базе рассмотренной инфраструктуры, достиг лучших результатов на 9 из 18 тестовых СБИС. Среднее отставание от лучших результатов среди всех инструментов составляет 0.3%.

Во многом это объясняется удачной вычислительной схемой, предложенной авторами APlace, однако вклад инфраструктуры также является значимым. Так, инфраструктура позволила как сравнительно быстро выполнить реализацию, так и настроить параметры алгоритмов и проверить работоспособность идей и подходов к оптимизации.

Второй опробованный алгоритм решает задачу оптимизации временных характеристик СБИС методом вставки буферов. В качестве тестового набора для алгоритма буферизации использовались задачи из пакета IWLS2005[7]. В качестве библиотек элементов использовались 45 нм библиотека Nangate FreePDK45 и 180 нм GSCLib_3.0[8].

В качестве стартовых состояний СБИС использовались результаты работы алгоритма размещения, минимизирующего длину соединений. Перед буферизацией производилась грубая трассировка, в процессе которой для каждого соединения строилось прямоугольное дерево Штейнера. После трассировки проводился статический временной анализ. В полученные деревья вставлялись буферные элементы, позиции которых определялись классическим алгоритмом Ван Гиннекена [7]. Буферизованное размещение легализуется, после чего снова производится трассировка и СВА. Общий маршрут проектирования (M2) приведен на рис. 3.

Замерялись следующие величины: суммарная длина соединений (total wire-length, TWL) и суммарная задержка (total negative slack, TNS). Результаты вычислительных экспериментов на 180 нм библиотеке элементов приведены в таблице 2.

Результаты первых экспериментов показывают, что алгоритм буферизации сокращает суммарную задержку на 15%, но при этом увеличивает длину соединений в среднем на 29%. В будущем планируется сбалансировать влияние критериев на работу алгоритма.

Заключение

В работе описана исследовательская инфраструктура для разработки и апробации алгоритмов физического проектирования СБИС. При-

Таблица 2

Результаты работы алгоритма вставки буферов, маршрут M2

СБИС	Число элементов	TWL (нм)	TNS (наносек)	Время работы (сек)	Число вставленных буферов	ΔTWL (%)	ΔTNS (%)
s298	141	3.16E+06	9.10E+00	0.019	14	9.02	0.46
pci_spoci_ctrl	1267	4.58E+07	7.90E+01	0.104	254	40.92	6.36
tv80	7161	3.85E+08	1.31E+03	0.671	1392	47.53	-15.39
ac97_ctrl	11855	6.49E+08	2.67E+03	1.779	2284	26.19	-15.26
b21	18718	1.06E+09	2.02E+03	1.420	2055	11.42	-19.22
b22	28317	1.63E+09	2.98E+03	2.270	3225	6.11	-17.54
b17	37117	2.28E+09	5.32E+03	3.674	6510	14.94	-22.83
b18	92048	5.21E+09	1.54E+04	11.123	16697	25.03	-20.01
des_perf	98341	5.82E+09	8.94E+03	8.065	15396	19.31	-14.00
					Среднее Δ	22.27	-13.05

ведены требования к инфраструктуре, архитектура и проектные решения, сформулировано назначение основных систем, рассмотрены особенности программной реализации. Применимость инфраструктуры показана на примере решения задач размещения элементов СБИС и вставки буферов.

Дальнейшие исследования авторов будут связаны с наполнением инфраструктуры эффективными реализациями некоторых алгоритмов физического проектирования СБИС, а также совершенствованием ее функциональности.

Благодарности

Авторы выражают благодарность Андрею Жмурину, Олегу Венгеру, Леониду Крагинскому за оказанную поддержку, консультации и обсуждения.

Работа выполнена при поддержке компании Intel в лаборатории «Информационные технологии» (ITLab) факультета ВМК ННГУ.

Примечания

1. OpenEDATools http://www.openedatools.org/; Open Circuit Design http://opencircuitdesign.com/; VLSI CAD Bookshelf http://vlsicad.eecs.umich. edu/BK/Slots/
2. Cadence LEF/DEF Exchange Format http:// www.si2.org/openeda.si2.org/projects/lefdef/
3. Liberty Library Modeling Format http:// www.opensourceliberty.org
4. Libconfig – C/C++ Configuration File Library http:// www.hyperrealm.com/libconfig/libconfig.html; C++ Delegates Implementation http:// www.codeproject.com/KB/ cpp/FastDelegate.aspx; Open Computer Vision Library http:// sourceforge.net/projects/opencvlibrary/
5. Toolkit for Advanced Optimization http:// www.mcs.anl.gov/research/projects/tao/; GLPK http://www.gnu.org/software/glpk/
6. Набор тестовых задач ISPD04 http:// www.public. iastate.edu/~nataraj/ISPD04_Bench.html
7. Набор тестовых задач IWLS05 http:// www.iwls.org/iwls2005/benchmarks.html
8 Nangate FreePDK45 – 45 нм библиотека http://www.si2.org/openeda.si2.org/project/showfiles.php?group_id=63#373; GSCLib_3.0 – 180 нм библиотека http://crete.cadence.com

Список литературы

1. Гаврилов С.В., Каграманян Э.Р., Ходош Л.С. Тенденции развития моделей библиотечных элементов для статического временного анализа цифровых СБИС // Информационные технологии. 2009. № 3. С. 20–24.
2. Kamaev A., Kornyakov K., Meyerov I. et al. Building a Research Framework for Integrated Circuit Physical Design // 6th IEEE East-West Design & Test International Symposium. Lvov, 2008. P. 251–253.
3. Kahng A.B., Wang Q. Implementation and Extensibility of an Analytic Placer // IEEE Transactions on Computer-Aided Design. 2005. Vol. 24, No. 5. P. 734–747.
4. Spindler P., Schlichtmann U., Johannes F.M. Abacus: fast legalization of standard cell circuits with minimal movement // 17th International Symposium on Physical Design (ISPD 2008), Portland, Oregon, USA, April 13–16, 2008. Proceedings. ACM, 2008. P. 47–53.
5. Wang M., Yang X., Sarrafzadeh M. Dragon2000: Standard-cell Placement Tool for Large Industry Circuits // 2000 International Conference on Computer-Aided Design (ICCAD 2000), San Jose, California, November 5–9, 2000. Proceedings. IEEE Press, 2000. P. 260–263.
6. Kornyakov K., Kurina N., Meyerov I., Zhivoderov A. An Improved Implementation of the Simulated Annealing Based Standard Cell Placement Algorithm Dragon // 5th IEEE East-West Design & Test International Symposium, Yerevan, Armenia, September 7–10, 2007. Proceedings. Kharkov National University of Radioelectronics, 2007. P. 354–359.
7. Van Ginneken L.P.P.P. Buffer placement in distributed RC-tree network for minimal Elmore delay // 1990 IEEE International Symposium Circuits and Systems (ISCAS 1990), New Orleans, LA, USA, May 1–3, 1990. Proceedings. IEEE, 1990. P. 865–868.

RESEARCH FRAMEWORK FOR VLSI CAD ALGORITHMS

A.V. Zhivoderov, A.M. Kamaev, K.V. Kornyakov, I.B. Meerov

A research framework is considered for development and testing of physical design algorithms of very-large-scale integration (VLSI) circuits (placement, routing, buffer insertion, etc.). The state-of-the-art of this area is analyzed. The proposed framework, core ideas and some implementation details are considered. Potentialities of the framework are illustrated by the examples of its implementation and use in VLSI CAD.

Keywords: VLSI CAD, physical design, VLSI design flow, framework.