



PARALLEL ALGORITHM FOR IMPLEMENTING LOGICAL OPERATIONS ON SETS OF ORTHOGONAL POLYGONS

N. Starostin, A. Shtanyuk, M. Godovitsyn, J. Zhivchikova

Lobachevsky State University,
603950, Nizhni Novgorod, Russia

DOI: 10.24412/2073-0667-2022-1-55-65

The IC designing main task is obtaining a workable crystal layout, which will be used to create a template for fabrication. It is important to perform a verification cycle of the obtained layout before transferring the designed solutions to production. The first stage of verification (DRC) consists of design rules layout according to the check. These rules are described by the system of norms, restrictions, rules and procedures regulating permissible mutual arrangement of topological elements and topological structures, taking into account design features and possibilities of technological process. There are two phases in almost any verification procedure: the first one is the search of topological elements and specific areas, the second one is the check for design rules compliance directly.

It is necessary to use logical operations on orthogonal rectangles that make up the topology of an integrated circuit. Such operations as union, intersection, subtraction are performed over layers that contain orthogonal rectangles (polygons). These operations are subject to stringent execution time requirements. The traditional bitmap rectangle representation does not allow for a quasi-linear time dependence on the processed data and requires development of new algorithms and approaches to polygon representation.

It is proposed to describe the polygon by the coordinates of the polyline nodes (boundary nodes). Such a representation will be called contour representation, which is de facto standard representation for the layout at the verification stage. In contour representation any topological layer is described by enumeration of all contours that form polygon boundaries. Each contour begins with an arbitrary starting node and ends with a finite node, which always coincides with the starting node. Such a representation is compact - the memory cost depends linearly on the number of nodes or edges of the layer contours. Problem of that kind of representation is that logical operations require information not about the boundaries, but the inner regions of polygons.

To resolve that issue, the „sweeping line“ scheme is used. The idea is to exploit three properties of a polygon. First, any edge of a polygon boundary always separates the interior region of the polygon from the exterior. Second, any edge of an orthogonal polygon is either vertical or horizontal. Third, if the plane is dissected by straight lines according to the vertical and horizontal edges into rectangular fragments, then any such fragment will belong entirely to either the interior or the exterior region of the polygon.

The procedures for input of polygon contour representations, which are divided into sets of vertical and horizontal edges, are described. As a result of performing logical operations, polygon edges of the resulting layer are formed, which, in turn, are converted into contour representations. If we take into account that we can use algorithm implementations based on the classical interval tree as functions providing work with half-intervals, then the computational complexity of this procedure in memory and in time is estimated as $O(N \log N)$, where N – is the number of horizontal edges of layer polygons.

A parallel implementation based on geometric decomposition of input data has been developed to function effectively in a high-performance computing environment. The theoretical study showed

linear scalability of this implementation on systems with distributed memory, but implementation as a multithreaded application revealed competition for shared resources. The paper presents the results of a computational experiment and outlines the ways to eliminate the competition effect.

The implementation of algorithms and procedures presented in this work were included in the plugin library of functions for working with topological layers, which, in its turn, is a part of the design rules checking system being developed. This library is implemented in C++, using C++17 standard. Implementation of parallel schemes to implement logical operations is done using the OpenMP library.

The results of a computational experiment confirming the nature of the time dependences determined theoretically are presented.

We propose the structure of a software system for DRC, built with the use of programming languages C++ and Lua.

Key words: CAD, DRC, logical operations, orthogonal rectangles, polygons, sweeping line modified algorithm, IC, parallel algorithm, OpenMP.

References

1. Batishchev D. I., Starostin N. V., Filimonov A. V. Mnogourovnevyyj algoritm resheniya zadachi komponovki integral'nyh skhem // Sistemy upravleniya i informacionnye tekhnologii. 2007. N 3 29. S. 48-52 (in Russian).
2. Starostin N. V., Filimonov A. V., Balashov V. V. Reshenie zadachi razmeshcheniya elementov specjalizirovannyh bol'shih integral'nyh skhem na osnove bazovyh matrichnyh kristallov // Sistemy upravleniya i informacionnye tekhnologii. 2009. N 2-1 36. S. 189-194 (in Russian).
3. Starostin N. V., Balashov V. V. Ispol'zovanie gipergrafov dlya resheniya zadachi ortogonal'noj trassirovki bol'shih integral'nyh skhem s neregulyarnoj strukturoj // Radiotekhnika i elektronika. 2008. T. 53. N 5. S. 618–623 (in Russian).
4. Vlasov S. E., Godovicyn M. M., Starostin N. V. Koncepciya mnogourovnevoj trassirovki cepej integral'nyh skhem s ispol'zovaniem virtual'nyh kanalov // Uspekhi kibernetiki. 2020. T. 1. N 1. S. 8–16 (in Russian).
5. Afrajmovich L. G., Vlasov V. S., Kulikov M. S., Priluckij M. H., Starostin N. V., Filimonov A. V. Planirovaniye i operativnoe upravlenie processom izgotovleniya slozhnyh izdelij // V sbornike: XII vserossijskoe soveshchanie po problemam upravleniya VSPU-2014. Institut problem upravleniya im. V. A. Trapeznikova RAN. 2014. S. 5138–5149 (in Russian).
6. Afrajmovich L. G., Vlasov V. S., Priluckij M. H., Sedakov D. V., Starostin N. V., Filimonov A. V., Kulikov M. S. Zadachi planirovaniya i operativnogo upravleniya processom izgotovleniya integral'nyh skhem s mikronnymi i submikronnymi topologicheskimi normami // Avtomatizaciya v promyshlennosti. 2014. N 8. S. 17–21 (in Russian).
7. Shtanyuk A. A., Semenov A. O. Problema analiza topologii integral'nyh skhem na osnove GDSII fajlov. Inzhenernye i informacionnye tekhnologii, ekonomika i menedzhment v promyshlennosti // Sbornik nauchnyh statej po itogam vtoroj mezhdunarodnoj nauchnoj konferencii. Volgograd, 2020. S. 334–336 (in Russian).
8. Galashov D. A., Shtanyuk A. A. Postanovka zadachi issledovaniya algoritmov proverki izomorfizma gipergrafov pri analize topologii integral'nyh skhem na osnove GDSII i DEF fajlov // Informacionnye sistemy i tekhnologii IST-2020: sbornik trudov Mezhdunarodnoj nauchno-tehnicheskoy konferencii. Nizhnij Novgorod, 2020. S.743–749 (in Russian).
9. Godovitsyn Maxim, Zhivchikova Julia, Starostin Nickolay, Shtanyuk Anton. Algorithm for Implementing Logical Operations on Sets of Orthogonal Polygons. Proceedings of the 31st International Conference on Computer Graphics and Vision (GraphiCon 2021) Nizhny Novgorod, Russia, September 27–30, 2021 // CEUR Workshop Proceedings, 2021, 3027, S. 1088–1097. DOI: 10.20948/graphicon-2021-3027-1088-1097.



ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА ЛОГИЧЕСКИХ ОПЕРАЦИЙ НАД МНОЖЕСТВАМИ ОРТОГОНАЛЬНЫХ МНОГОУГОЛЬНИКОВ

Н. В. Старостин, А. А. Штанюк, М. М. ГодовицЫн, Ю. А. Живчикова

Нижегородский государственный университет им. Н. И. Лобачевского
603950, Нижний Новгород, Россия

УДК 519.175.1, 621.3.049.771.14

DOI: 10.24412/2073-0667-2022-1-55-65

В рамках разработки отечественных САПР для верификации норм конструкторско-технологических ограничений (КТО) необходима разработка библиотеки выполнения логических операций над ортогональными многоугольниками, составляющими топологию интегральной схемы. Функции библиотеки выполняют операции над слоями. Под слоем понимается множество ортогональных прямоугольников (полигонов). К этим операциям предъявляются жесткие требования по времени выполнения. Существует реализация, построенная на основе алгоритма заметающей прямой и позволяющая добиться времени работы алгоритма порядка $O(N \log N)$.

Для эффективного функционирования в высокопроизводительной вычислительной среде разработана параллельная реализация, основанная на геометрической декомпозиции исходных данных. Проведенное теоретическое исследование показало линейную масштабированность данной реализации на системах с распределенной памятью, но реализация в виде многопоточного приложения выявила конкуренцию за разделяемый ресурс. В работе приводятся результаты вычислительного эксперимента и намечаются пути устранения эффекта конкуренции. Представленные в работе реализации алгоритмов и процедур вошли в состав подключаемой библиотеки функций работы со слоями топологического описания, которая, в свою очередь, является частью разрабатываемой системы верификации норм КТО. Данная библиотека реализована на языке C++ с использованием стандарта C++17. Имплементация параллельных схем реализации логических операций выполнена с использованием библиотеки OpenMP.

Ключевые слова: САПР, проверка конструктивно-технологических ограничений, логические операции, полигоны, параллельная реализация алгоритма заметающей прямой, OpenMP.

Введение. Основная задача проектирования интегральной схемы [1–4] заключается в получении топологии работоспособного кристалла, по которой будет создаваться шаблон для фабричного изготовления. Перед передачей спроектированных решений в производство [5, 6] важно выполнить цикл верификации полученной топологии [7, 8]. Начальный этап верификации заключается в проверке проекта топологии интегральной схемы на соответствие нормам конструктивно-технологических ограничений (КТО), которые описываются системой норм, ограничений, правил и процедур, регламентирующих допустимое взаимное расположение топологических элементов и топологических структур с учетом конструктивных особенностей и возможностей технологического процесса. Структура процессов верификации топологии включает две основные фазы: на первой

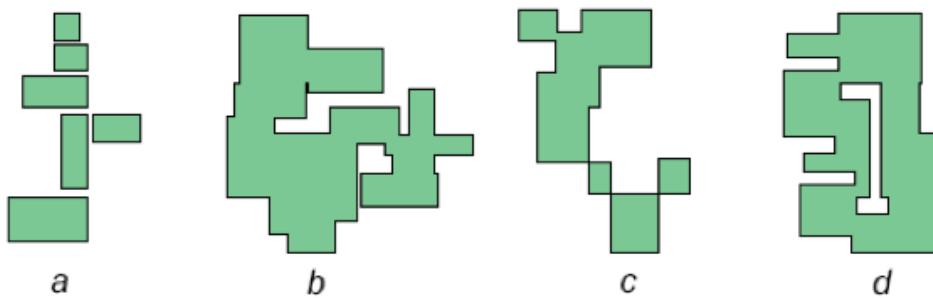


Рис. 1. Примеры ортогональных полигонов

происходит поиск элементов топологии и специфических областей, на второй осуществляется непосредственно проверка соответствия правилам проектирования. Подавляющее большинство поисковых и проверочных процедур формулируются в терминах логических операций (объединение, пересечение, вычитание) над множествами, содержащими ортогональные многоугольники (полигоны). К алгоритмам, реализующим логические операции, предъявляются жесткие требования по вычислительным издержкам, которые по времени и по памяти не должны превышать квазилинейные оценки.

На практике размеров входных данных исчисляется в десятках и сотнях миллионов полигонов и более. При данных порядках времени работы алгоритмов в рамках проверочных процедур значительно даже с учетом квазилинейных издержек. Следует отметить, что, если в результате автоматической проверки найдены проблемы, это означает необходимость внесения изменений в синтезированную топологию и предполагает цикл разработки с повторной проверкой до тех пор, пока не будет получено корректное решение. В результате возрастают значимость требований по сокращению временных издержек к программно-техническим средствам верификации, что обуславливает использование высокопроизводительной вычислительной базы и эффективных параллельных реализаций алгоритмов на данном этапе проектирования изделий микроэлектроники.

1. Слои интегральной схемы и последовательная схема реализации логических операций. Рассмотрим последовательную схему работы алгоритмов, формирующих результат работы логических операций над слоями интегральной схемы [9]. В качестве входных данных для логических операций выступают топологические слои. Под топологическим слоем понимается множество попарно непересекающихся плоских ортогональных многоугольников (полигонов). Каждый полигон характеризуется внутренней областью со своей границей, которая представляется в виде ортогональной ломаной, состоящей только из прямых отрезков, параллельных одной из осей плоскости кристалла (контуровое представление). На рис. 1 приведены примеры ортогональных полигонов с границами и внутренними областями.

В качестве базовых логических операций выделяют операции: объединение (обозначают OR), пересечение (обозначают AND), вычитание (обозначают NOT) и исключающее „ИЛИ“ (обозначают XOR). Результатом операции объединения является новый слой, который включает в себя полигоны, описывающие области, относящиеся к полигонам первого и/или второго слоев. Результатом операции пересечения является новый слой, который включает в себя полигоны, описывающие области, относящиеся одновременно к полигонам первого и второго слоев. Результатом операции вычитания является новый слой, который

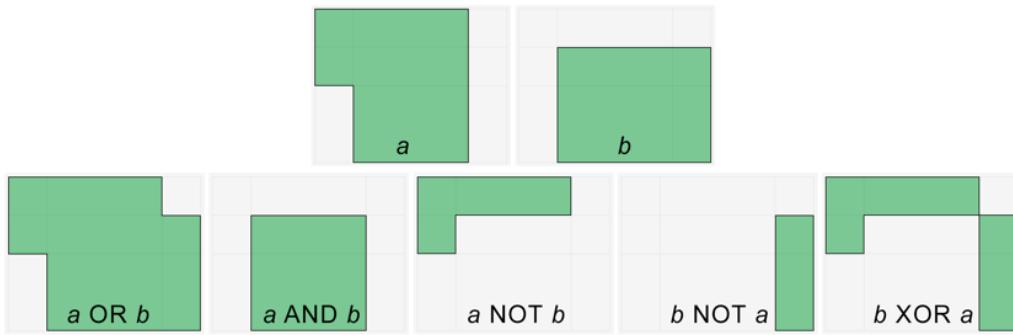


Рис. 2. Примеры результатов работы логических операций над слоями

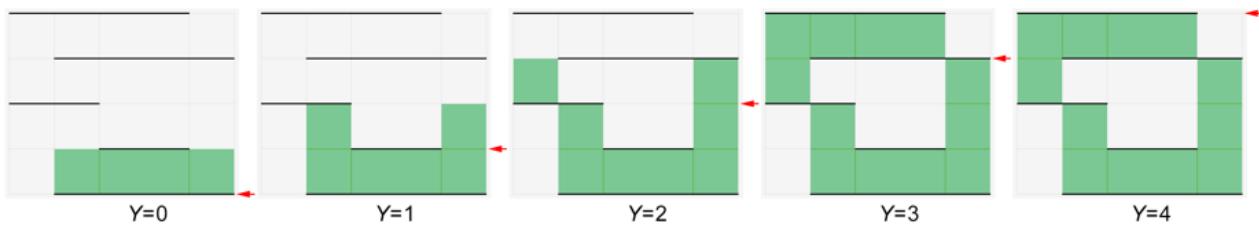


Рис. 3. Схема работы алгоритма вычисления внутренних областей слоя

включает в себя полигоны, описывающие области полигона первого слоя, за исключением совпадающих областей второго слоя. Результатом операции исключающего „ИЛИ“ является новый слой, который включает в себя полигоны, описывающие области полигона первого или второго слоев, за исключением совпадающих областей первого и второго слоя. На рис. 2 приведены примеры результатов работы логических операций над слоями.

Процесс вычисления логических операций основан на схеме вычисления внутренних областей полигонов слоя по контурному представлению. Данный процесс выглядит следующим образом. Рассмотрим только горизонтальные ребра полигонов одного слоя. Вычислим ограничивающий прямоугольник, включающий все горизонтальные ребра. Разрежем ограничивающий прямоугольник на горизонтальные полосы так, чтобы разрезы проходили строго по всем горизонтальным ребрам. Упорядочим (пронумеруем) разрезы по порядку их размещения на плоскости „снизу вверх“. Так как любое ребро границы полигона всегда отделяет внутреннюю область многоугольника от внешней, то смена внутренних областей полигона на внешние и, наоборот, внешних на внутренние происходит строго на разрезах. Тогда схема вычисления внутренних областей фактически заключается в последовательном переборе разрезов „снизу вверх“. Для каждого i -го разреза происходит вычисление результата операции $I_i = B_i \text{ XOR } I_{i-1}$, где I_i — интервалы, определяющие внутренние области полигонов слоя над i -м разрезом; B_i — интервалы, соответствующие горизонтальным ребрам, попавшие в i -й разрез. На рис. 3 приведена иллюстрация работы алгоритма.

Вычислительная сложность представленной схемы оценивается как $O(N \log N)$ как по времени работы, так и по пиковому потреблению памяти.

2. Общая схема работы алгоритма логических операций над слоями. Схема вычисления результата логической операции над слоями может быть описана в виде следующего алгоритма [9]:

1. Исходные данные: контурное представление слоя 1 и слоя 2;
2. Извлечь H_1 и H_2 — множества горизонтальных ребер слоя 1 и слоя 2;
3. Упорядочить элементы множеств H_1 и H_2 по значению координат Y ;
4. Вычислить внутренние области полигонов по вертикальной координате, получить множество горизонтальных ребер H ;
5. Извлечь V_1 и V_2 — множества вертикальных ребер слоя 1 и слоя 2;
6. Упорядочить элементы множеств V_1 и V_2 по значению координат X ;
7. Вычислить внутренние области полигонов по горизонтальной координате, получить множество вертикальных ребер V ;
8. Построить реберный граф по элементам из H и V ;
9. Найти все простые циклы обходом в глубину реберного графа;
10. Сохранить все найденные циклы в контурное представление итогового слоя.

Аккумулируя информацию по сложности работы отдельных шагов работы алгоритма, можно отметить, что общая вычислительная сложность представленной схемы расчета результата логических операций над слоями в контурном представлении зависит только от числа ребер полигонов слоя и оценивается как $O(N \log N)$ по времени работы и по потреблению памяти.

3. Параллельная схема реализации логических операций. Представленная выше схема вычисления внутренних областей полигонов слоя основана на последовательном переборе разрезов строго „снизу вверх“ (или „слева направо“) и не позволяет начать вычисление внутренних областей с произвольного разреза. Дело в том, что для того, чтобы корректно вычислить внутренние области над i -м разрезом I_i , необходимо знать интервалы размещения внутренних областей под i -м разрезом, что эквивалентно интервалам размещения внутренних областей над $(i-1)$ -м разрезом I_{i-1} .

Для вычисления внутренних областей над $(i-1)$ -м разрезом воспользуемся свойством границы полигона, которая всегда отделяет внутреннюю область многоугольника от внешней. Однако, при этом будем рассматривать только вертикальные ребра, пересекающие горизонтальную прямую, расположенную между $(i-1)$ -м и i -м разрезами (см. рис. 4). Если упорядочить (пронумеровать, начиная с 1) все такие вертикальные ребра по порядку их размещения на плоскости „слева направо“ (по возрастанию координаты X), то каждое нечетное ребро служит индикатором „переключения“ внешних областей полигона на внутренние, а каждое четное ребро отмечает факт „переключения“ внутренних областей на внешние, что дает возможность построить простую схему вычисления искомых интервалов размещения внутренних областей над $(i-1)$ -м разрезом I_{i-1} .

Таким образом, для обеспечения геометрической декомпозиции исходных данных на заданное количество частей для параллельного независимого расчета необходимо: 1) определить разрезы, по которым будет осуществляться разрезание всей геометрии на части; 2) осуществить декомпозицию горизонтальных ребер по частям; 3) идентифицировать списки вертикальных ребер, пересекающих соответствующие разрезы. Определять места разрезов предлагается по геометрическому принципу — прямоугольник, ограничивающий полигоны слоя, разрезается на заданное число полос равной ширины.

В основу процедур декомпозиции горизонтальных ребер по частям предлагается положить параллельную схему — каждый процесс независимо обрабатывает свою часть массива вертикальных ребер и распределяет их по независимым спискам, которые проассоциированы соответствующим частям декомпозиции. Далее, после фазы параллельного расчета, списки перераспределяются по „своим“ частям данных. По аналогии с процессом

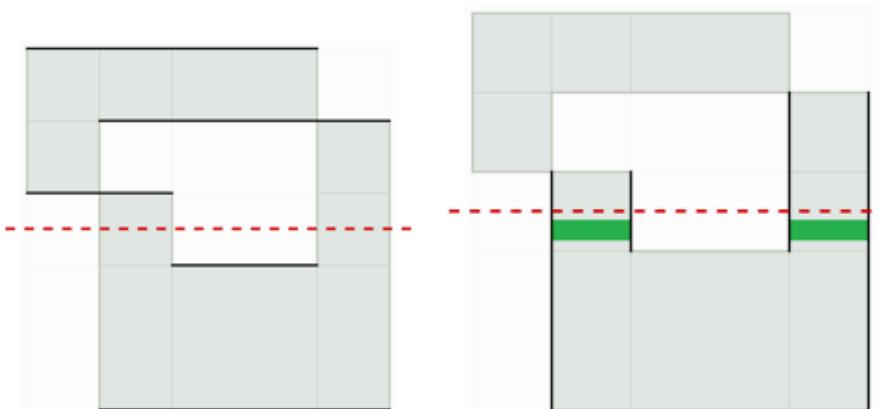


Рис. 4. Распределение горизонтальных ребер по двум частям (слева), определение вертикальных ребер и вычисление внутренних областей под разрезом (справа)

декомпозиции может быть организована параллельная схема вычисления списков вертикальных ребер, пересекающих соответствующие разрезы.

В результате описанной выше процедуры будут подготовлены данные для независимого параллельного вычисления внутренних областей в рамках каждой из частей. В таком случае оценка ускорения предложенной параллельной схемы вычисления данных с учетом фазы подготовки данных и фазы вычисления оценивается как N/P , где N — число ребер; P — число процессов/процессоров.

4. Программное обеспечение системы верификации КТО. Представленные в работе реализации алгоритмов и процедур вошли в состав подключаемой библиотеки функций работы со слоями топологического описания, которая, в свою очередь, является частью разрабатываемой системы верификации норм КТО [9]. Данная библиотека реализована на языке C++, с использованием стандарта C++17. Имплементация параллельных схем реализации логических операций выполнена с использованием библиотеки OpenMP.

Общая архитектура системы верификации КТО топологии микросхем приведена на рис. 5. В соответствии с архитектурой, система верификации КТО включает следующие компоненты: интерпретатор языка Lua — позволяет выполнять программу автоматической проверки правил КТО; библиотека функций для работы со слоями в составе содержит функции выполнения логических операций; блок работы с файлами GDSII; блок работы с базой данных слоев в составе; блок работы с отчетами.

Следует отметить, что входными данными системы верификации КТО являются GDSII файл топологии интегральной схемы и скрипт верификации. Выходными данными являются база данных результирующих слоев и отчеты работы алгоритмов верификации. Таким образом, для скрипта верификации обеспечиваются следующие возможности: чтение слоев из файла GDSII; обработка слоев при помощи функций из библиотеки функций для работы со слоями; сохранение слоев в базе данных слоев; сохранение логов и отладочной информации в хранилище отчетов работы алгоритмов верификации.

Представленная архитектура разработанного программного обеспечения предоставляет широкие возможности с точки зрения расширяемости функционала системы верификации КТО, которая обеспечивается большой описательной силой языка Lua в рамках

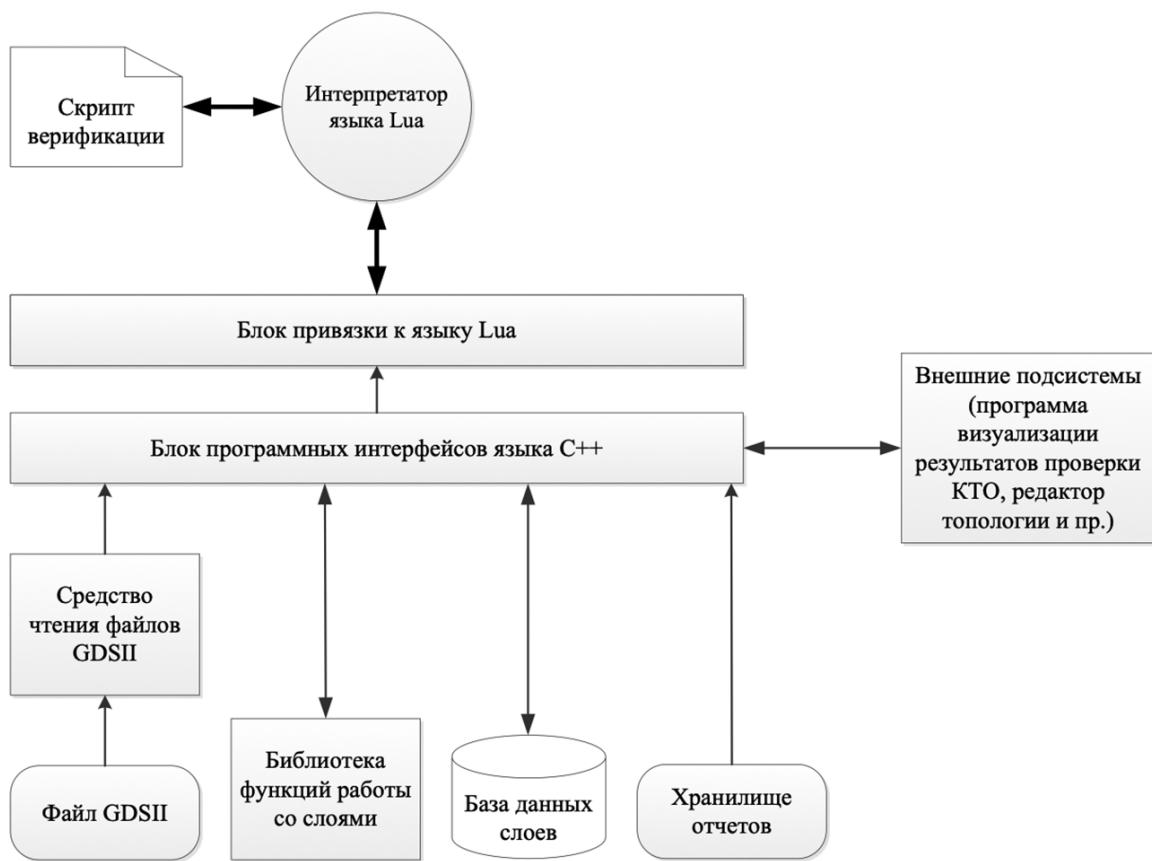


Рис. 5. Архитектура системы верификации КТО топологии микросхем

императивного подхода, а также широким спектром возможностей по наращиванию функционала динамически подключаемой библиотеки функций.

5. Результаты вычислительного эксперимента. Для апробации функций библиотеки операций над полигонами были подготовлены тестовые данные, представляющие собой набор координат вершин полигонов для двух слоев. Число полигонов в рамках одного слоя варьируется от 2000 до 50000. Над тестовыми данными выполнялись операции объединения, пересечения, вычитания и исключающего „ИЛИ“.

Для тестирования использовался персональный компьютер с 8-ядерным процессором Intel Core i7, 3.8 ГГц и объемом памяти 32 ГБ.

Результаты измерения времени для различных операций приведены на рис. 6.

Из рисунка видно сохранение квазилинейной сложности алгоритма при запуске алгоритма на нескольких процессорах, однако, отсутствует линейное ускорение алгоритма при увеличении числа потоков, а именно, наблюдается резкое снижение прироста производительности при числе потоков больше, чем 2.

Из полученных зависимостей видно, что оценка временных затрат на выполнение операций не соответствует теоретически полученным оценкам. Анализ производительности показал наличие проблемы кэш промахов (cache miss). Была предпринята попытка решения данной проблемы за счет геометрической декомпозиции области на большое количество достаточно маленьких (достаточных для попадания в кэш первого уровня) подмножеств ребер. Отчасти это улучшило общую картину — время вычисления результата

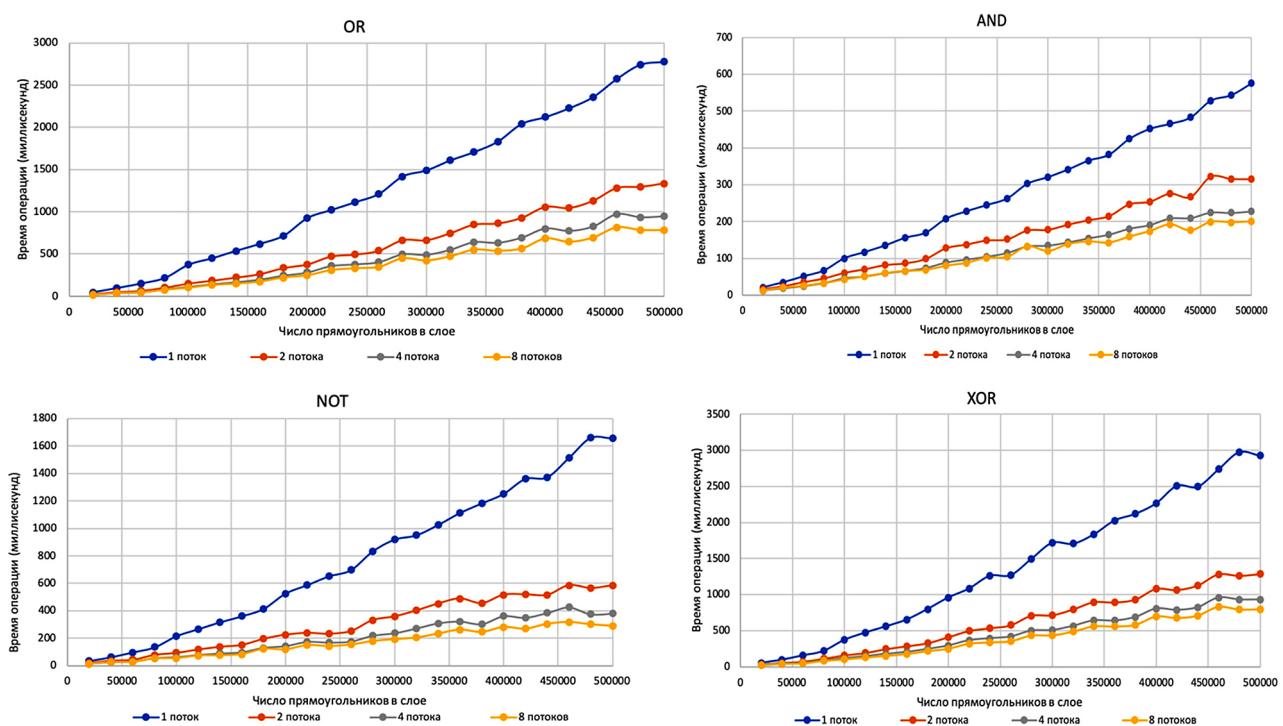


Рис. 6. Зависимости времени операций от количества полигонов в слое и числа потоков

логических операций в среднем сократилось в 3 раза, однако проблема масштабируемости в контексте параллельных вычислений на общей памяти осталась. Основная причина этого очевидно связана с конкуренцией потоковых вычислений за шину памяти. Данная проблема является характерной для всех многопроцессорных систем при работе с задачами, требующими интенсивного обмена данными по шине память-процессор. Однако стоит заметить, что предложенная схема декомпозиции ребер обеспечивает последующий независимый расчет по каждой отдельной части геометрии, что открывает широкие возможности для использования параллельных вычислений на системах с распределенной памятью.

Заключение. В рамках проекта по созданию отечественной САПР для верификации норм КТО была разработана библиотека, реализующая логические операции над слоями, образующими топологическое описание микросхемы. В основу процедур вычисления результата логических операций положена последовательная схема, обеспечивающая квазилинейную вычислительную временную сложность. Предложена технология ее распараллеливания, основанная на геометрической декомпозиции данных. Теоретический анализ показал линейную масштабируемость предложенных параллельных решений на системах с распределенной памятью. Попытка реализации данной технологии в рамках многопоточного приложения выявила типовые проблемы масштабируемости на системах с общей памятью, связанные с конкуренцией за разделяемый ресурс. Дальнейшее развитие данного направления работ предполагает реализацию предложенных решений для распределенных вычислений — ожидается, что это позволит достичь наибольшего ускорения и масштабируемости.

Список литературы

1. Батищев Д. И., Старостин Н. В., Филимонов А. В. Многоуровневый алгоритм решения задачи компоновки интегральных схем // Системы управления и информационные технологии. 2007. № 3. 29. С. 48–52.
2. Старостин Н. В., Филимонов А. В., Балашов В. В. Решение задачи размещения элементов специализированных больших интегральных схем на основе базовых матричных кристаллов // Системы управления и информационные технологии. 2009. № 2–1. 36. С. 189–194.
3. Старостин Н. В., Балашов В. В. Использование гиперграфов для решения задачи ортогональной трассировки больших интегральных схем с нерегулярной структурой // Радиотехника и электроника. 2008. Т. 53. № 5. С. 618–623.
4. Власов С. Е., Годовицын М. М., Старостин Н. В. Концепция многоуровневой трассировки цепей интегральных схем с использованием виртуальных каналов // Успехи кибернетики. 2020. Т. 1. № 1. С. 8–16.
5. Афраймович Л. Г., Власов В. С., Куликов М. С., Прилуцкий М. Х., Старостин Н. В., Филимонов А. В. Планирование и оперативное управление процессом изготовления сложных изделий // В сборнике: XII всероссийское совещание по проблемам управления ВСПУ-2014. Институт проблем управления им. В. А. Трапезникова РАН. 2014. С. 5138–5149.
6. Афраймович Л. Г., Власов В. С., Прилуцкий М. Х., Седаков Д. В., Старостин Н. В., Филимонов А. В., Куликов М. С. Задачи планирования и оперативного управления процессом изготовления интегральных схем с микронными и субмикронными топологическими нормами // Автоматизация в промышленности. 2014. № 8. С. 17–21.
7. Штанюк А. А., Семенов А. О. Проблема анализа топологии интегральных схем на основе GDSII файлов // Инженерные и информационные технологии, экономика и менеджмент в промышленности: Сборник научных статей по итогам второй международной научной конференции. Волгоград, 2020. С. 334–336.
8. Галашов Д. А., Штанюк А. А. Постановка задачи исследования алгоритмов проверки изоморфизма гиперграфов при анализе топологии интегральных схем на основе GDSII и DEF файлов // Информационные системы и технологии ИСТ-2020: сборник трудов Международной научно-технической конференции. Нижний Новгород, 2020. С. 743–749.
9. Godovitsyn Maxim, Zhivchikova Julia, Starostin Nikolay, Shtanyuk Anton. Algorithm for Implementing Logical Operations on Sets of Orthogonal Polygons. Proceedings of the 31st International Conference on Computer Graphics and Vision (GraphiCon 2021) Nizhny Novgorod, Russia, September 27–30, 2021 // CEUR Workshop Proceedings, 2021, 3027, S. 1088–1097. DOI: 10.20948/graphicon-2021-3027-1088-1097.



Старостин Николай Владимирович — д-р техн. наук, профессор кафедры информатики и автоматизации научных исследований Национального исследовательского нижегородского государственного университета им. Н. И. Лобачевского, e-mail: nvstar@iiani.unn.ru. Количество печатных работ: 80. Область научных интересов: моделирование и проектирование сложных систем, информационные технологии,

экстремальные задачи на графовых структурах, многоуровневая оптимизация.

Nikolay Starostin — Doctor of Technical Science, Professor at the Department of Informatics and Research Automation, Nizhny Novgorod State University n. a. N. I. Lobachevsky. The number of publications: 80. Research interests: modelling and design of complex systems, information technology, extremal problems on graph, multilevel optimization.

Штанюк Антон Александрович — канд. техн. наук, доцент, доцент кафедры информа-

матики и автоматизации научных исследований Национального исследовательского нижегородского государственного университета им. Н. И. Лобачевского, e-mail: anton.shtanyuk@itmm.unn.ru. Количество печатных работ: 50. Область научных интересов: моделирование и проектирование сложных систем, языки и технология программирования, обучение программированию, алгоритмы и структуры данных.



Anton Shtanyuk — PhD., Associate Professor at the Department of Informatics and Research Automation of the Institute of information technology, mathematics and mechanics at Lobachevsky State University of Nizhny Novgorod.

The number of publications: 50. Research interests: modelling and design of complex systems, programming languages and technology, programming teaching, algorithms and data structures.



Годовицын Максим Михайлович — аспирант Нижегородского государственного университета им. Н. И. Лобачевского, e-mail: maxim.godovicyn@gmail.com. Окончил Нижегородский государственный университет им. Н.И. Лобачевского в 2020 г. Количество печатных работ: 6. Область научных интересов: графовые задачи, моделирование слож-

ных систем, системный анализ, анализ данных, искусственный интеллект.

Maksim Godovitsyn — graduate student of Lobachevsky State University of Nizhny Novgorod. Graduated from Lobachevsky State University of Nizhny Novgorod in 2020. The number of publications: 6. Research interests: Graph problems, modeling of complex systems, system analysis, data analysis, artificial intelligence.



Живчикова Юлия Алексеевна — аспирант Института информационных технологий, математики и механики Нижегородского государственного университета им. Н. И. Лобачевского, e-mail: zhivchik96@mail.ru. Окончила Нижегородский государственный университет им. Н. И. Лобачевского в 2020 г. Количество печатных работ: 3. Область научных интересов: методы и технологии проектирования и производства интегральных схем, системы хранения и обработки данных.

Julia Zhivchikova — graduate student of the Institute of information technology, mathematics and mechanics at Lobachevsky State University of Nizhny Novgorod. Graduated from Lobachevsky State University of Nizhny Novgorod in 2020. The number of publications: 3. Research interests: Methods and technologies for designing and manufacturing integrated circuits, data storage and processing systems.

Дата поступления — 01.02.2022