УДК 658.512.2.011

СЕТЕВЫЕ ТЕХНОЛОГИИ КАК ИНСТРУМЕНТ РАСПРЕДЕЛЕННОГО ПРОЕКТИРОВАНИЯ ЭЛЕКТРОННЫХ СХЕМ

В.М. Глушань, д.т.н., профессор; П.В. Лаврик, аспирант (Таганрогский технологический институт Южного федерального университета, Некрасовский пер., 4, г. Таганрог, 347928, Россия, gluval07@rambler.ru, levarto@mail.ru)

Аннотация. Дается хронологический экспресс-анализ подходов к построению быстродействующих САПР электронных схем. Приводятся разработанная авторами структура и результаты имитационного моделирования распределенной САПР, ставших обоснованием целесообразности построения реальной подсистемы конструкторского проектирования электронных схем. Описаны основные модули подсистемы, их функционирование и результаты экспериментальных исследований разработанной подсистемы. Подсистема позволяет уменьшить время проектирования до трех раз.

Ключевые слова: *GRID-система, распределенная САПР, имитационное моделирование, клиент-серверная архитектура.*

Важнейшей причиной, обусловившей все расширяющиеся сферы использования САПР, является перманентное увеличение сложности проектируемых объектов. Ручное проектирование сложных объектов становится неэффективным (по многим показателям: временным, стоимостным и др.) или даже невозможным. Поэтому первостепенной задачей, которую призваны решать САПР и ради которой они создаются, является задача сокращения сроков проектирования. Она решается как совершенствованием всех видов обеспечения, входящих в САПР, так и применением новой организационно-технической парадигмы.

Проектирование с помощью САПР предполагает ведение интерактивного диалога человека с ЭВМ. Поэтому такое проектирование является автоматизированным, а не автоматическим. Оно сочетает скоростные возможности средств вычислительной техники в переборах вариантов решений и творческие способности человека при оценке и принятии решений.

Интерактивный диалог является эффективным, если он происходит в *режиме реального времени* (РРВ). В САПР под РРВ понимается такое взаимодействие пользователя и ЭВМ, когда ответы на запросы пользователя поступают из ЭВМ со скоростью, комфортной для пользователя. Известно [1], что для реализации РРВ необходимо, чтобы ЭВМ выдавала ответы не более чем через 2–3 секунды после поступления запроса пользователя. В противном случае человек становится самым ненадежным звеном в цепи «человек-ЭВМ», внося существенную долю ошибок, вызванных длительным ожиданием ответа. При этом теряется всякий смысл в использовании режима диалога.

В [2] выделены три основных направления в обеспечении интерактивности САПР. Начало первого направления в 70–80-е годы прошлого века связано с бурным расцветом *многопроцессорных вычислительных систем* (МВС). Основная идея увеличения быстродействия этих систем состояла в распараллеливании обработки информации на некоторое множество процессоров. При этом для обеспечения параллелизма обработки информации предлагались самые различные подходы структурной организации процессоров в МВС: векторная, матричная, систолическая и т.п.

Развитие второго направления обусловлено ростом сложности решаемых задач, при котором универсальная ЭВМ стала как бы одинаково неэффективной для каждой из них. Использование *пакета прикладных программ* (ППП) пользователя в универсальной ЭВМ по сути представляет собой настройку общезначимых вычислительных средств на решение узкого круга задач. Поэтому универсальную ЭВМ совместно с ППП в любой момент времени можно рассматривать как специализированную систему, ориентированную на решение определенной задачи. Большая часть ресурсов аппаратных и программных средств такой системы оказывается неиспользованной. Обладая способностью реализовывать любой алгоритм, универсальная ЭВМ не может конкурировать со специализированными процессорами при решении определенных классов задач.

Указанные обстоятельства инициировали разработку специализированных процессоров, максимально учитывающих специфику и структуру решаемых задач, что позволило создавать высокопроизводительные и эффективные программно-аппаратные ускорители. Своего расцвета исследования в области построения аппаратных ускорителей достигли в середине 80-х годов прошлого века. Систематическое изложение вопросов

теоретического и практического характера построения аппаратных акселераторов для решения задач автоматизированного конструирования приведено в [3].

В начале 90-х годов Интернет объединил большинство существовавших тогда сетей, и это событие привлекло пристальное внимание исследователей и специалистов, занимавшихся поисками путей совершенствования САПР. Третье направление опирается на достижения сетевых вычислительных технологий. Но центральная идея – распараллеливание обработки информации – заимствована из старой концепции, но состоит она теперь в том, чтобы вместо спецпроцессоров МВС, представляющую по существу сосредоточенную структуру, использовать компьютеры вычислительной сети – локальной или глобальной, представляющей распределенную систему [4].

Обзор и анализ возможности распараллеливания вычислительных процессов средствами сетевых технологий приведен в [5], где отмечается, что современные условия способствовали развитию трех направлений в системах распределенной обработки информации – это CAD/CAM/CAE/PDM, Beowulf- и GRID-системы. Первые направлены на создание единого информационного пространства, в котором могут работать одновременно специалисты различной направленности. Вторые призваны создать альтернативу дорогим суперкомпьютерам для обеспечения больших вычислительных мощностей в образовательных и научно-исследовательских организациях. GRID-системы сходны с кластерными системами в своем составе и цели создания. Но они отличаются по организации самого процесса вычислений и применяемыми при этом программными решениями.

В настоящей статье приводятся некоторые результаты исследований по созданию программной подсистемы автоматизированного конструкторского проектирования электронных схем.

С точки зрения системной архитектуры и организации вычислительного процесса разработанная подсистема наиболее близка интенсивно развивающейся GRID.

Имитационное моделирование распределенной подсистемы

Первый шаг в создании подсистемы состоит в анализе и обосновании ее эффективности. В качестве критерия эффективности по понятным соображениям выбрано время проектирования. Исследование эффективности было проведено путем имитационного моделирования по специально разработанной структурной схеме, приведенной на рисунке 1.

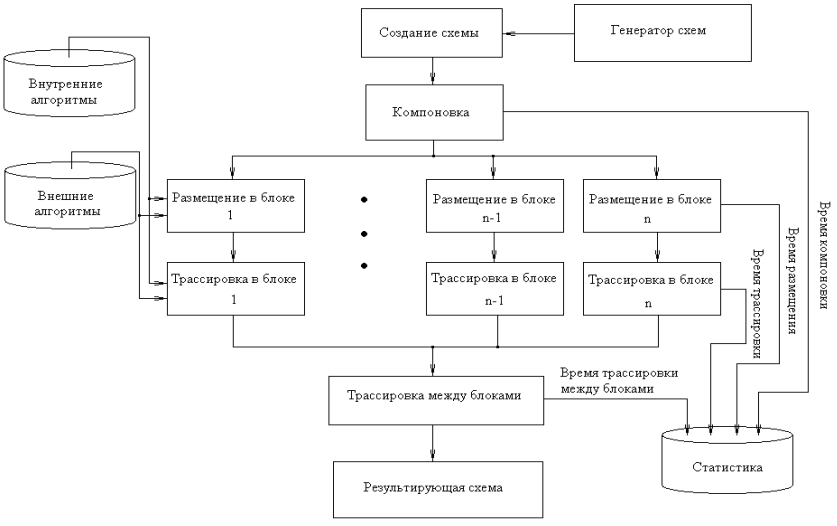

Рис. 1. Структурная схема имитационного моделирования

Очевидно, что время проектирования будет зависеть от числа частей (подсхем) исходной принципиальной схемы при условии, что все части будут обрабатываться одновременно. По окончании процесса обработки каждой части должна осуществляться сборка всех частей. При такой организации процесса проектирования должны оптимальным образом сочетаться две противоречивые процедуры. Первая – это обработка каждой из

подсхем. Чем больше подсхем, тем меньше их сложность, а поэтому и время обработки каждой подсхемы будет меньше. Вторая процедура – сборка всех частей в единую схему. Она протекает тем дольше, чем больше число частей на которые разбита исходная схема, поскольку тогда при сборке приходится обрабатывать большее число связей между подсхемами. Очевидно, что должно существовать оптимальное число подсхем, на которое целесообразно разбивать проектируемую схему.

Для проведения экспериментальных исследований в соответствии с рисунком 1 была разработана программа имитационного моделирования. Она состоит из трех основных модулей: модуля формирования случайных схем с заданными топологическими свойствами, проектирующего модуля и модуля статистической обработки. Назначение первого модуля – генерация множества случайных схем. Второй модуль осуществляет непосредственное конструкторское проектирование, третий реализует удобный исследовательский интерфейс для задания необходимых схем, а также выводит на экран компьютера нужные графические зависимости и спроектированную схему [6].

Проектирующий модуль разбивает исходную схему на подсхемы последовательным алгоритмом, размещает элементы внутри подсхем алгоритмом парных перестановок и выполняет трассировку соединений внутри подсхем и между подсхемами (другими словами – внутриблочную и межблочную) волновым алгоритмом. На рисунке 2 изображен график зависимости временного выигрыша (в разах) при проектировании схемы на распределенной САПР по сравнению с сосредоточенной в зависимости от числа частей разбиения схемы. Результаты получены для случайно сгенерированной схемы со следующими параметрами: число элементов N = 300, число контактов у каждого элемента – 12, число цепей – 300, разветвленность цепи – 8, коэффициент площади – 1,3.

На рисунке 2 наглядно приведены результаты для небольшой размерности схемы. Эксперименты проводились для различных по сложности схем. Во всех случаях тенденция, связанная с наличием оптимального числа частей схемы, сохранялась.

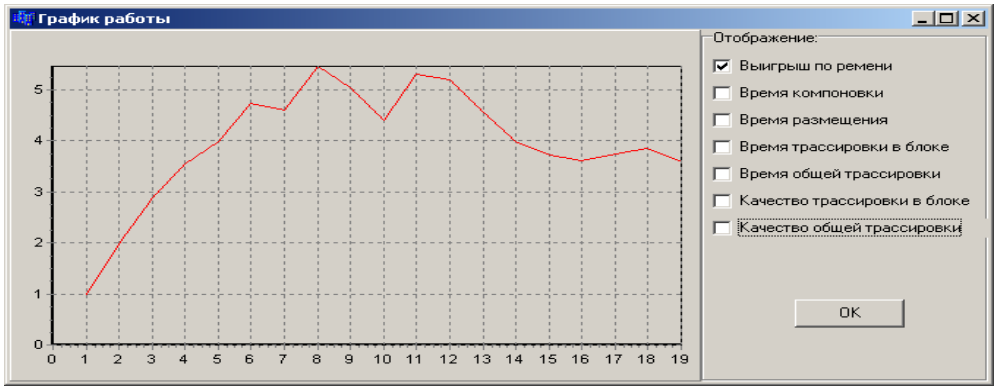

Рис. 2. Временной выигрыш в зависимсти от числа частей разбиения схемы

Разработка реальной распределенной подсистемы

Имитационное моделирование вместе с ранее полученными теоретическими результатами, изложенными в [7], стимулировали разработку реальной распределенной подсистемы конструкторского проектирования электронных схем. Для построения распределенной подсистемы была выбрана клиент-серверная архитектура, поскольку она является относительно несложной в реализации и не требует специальных настроек операционных систем используемых машин. Особенность реализации подсистемы состоит в том, что сервер является активной управляющей компонентой, а клиенты – пассивными поставщиками вычислительных ресурсов. Общая схема распределенной подсистемы конструкторского проектирования представлена на рисунке 3.

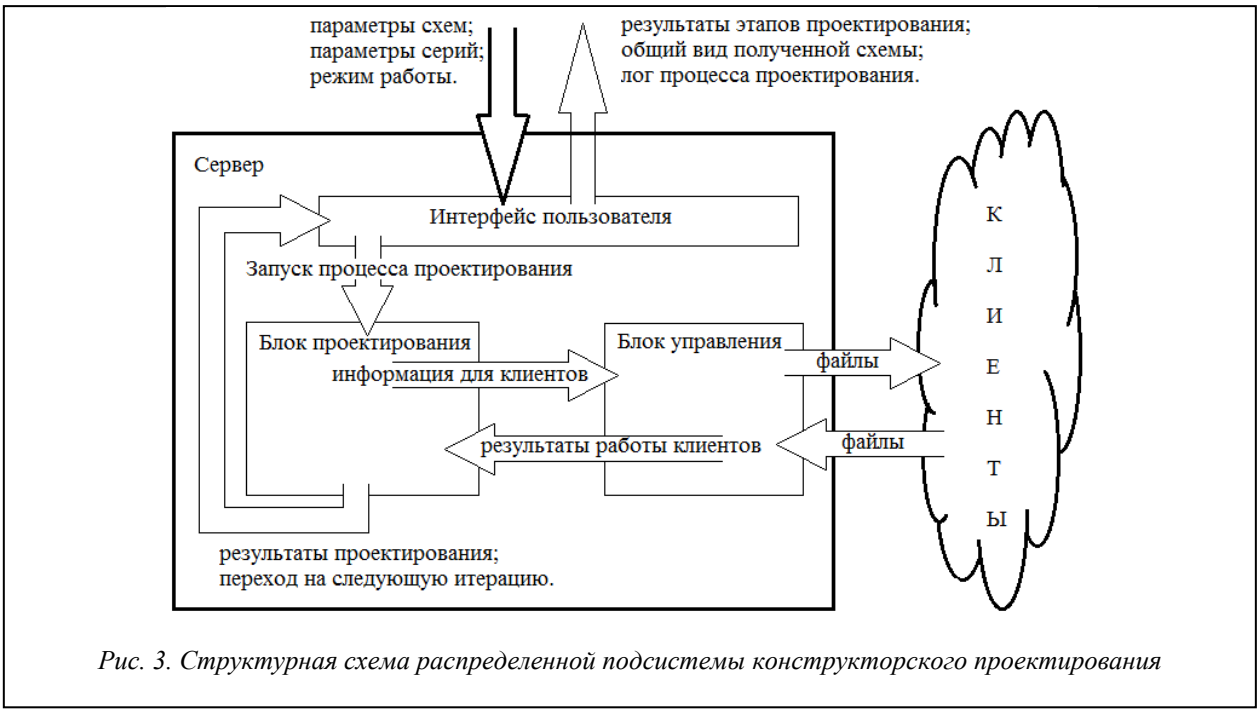

Рис. 3. Структурная схема распределенной подсистемы конструкторского проектирования

На серверную часть подсистемы возложены следующие функции: управление процессом проектирования, синхронизация потоков данных, удаленное управление клиентами, генерация моделей схем по заданным параметрам, ввод-вывод информации из файлов, прием/передача данных по сети, проведение серий экспериментов по заданным параметрам, последовательный запуск различных процедур проектирования, осуществление взаимодействия с пользователем, отображение информации о результатах экспериментов. В связи с этим серверную часть можно условно разделить на блоки интерфейса, проектирования и блок управления. Блок проектирования, в свою очередь, подразделяется на платформу, в которую входят структуры данных и генераторы этих структур, и надстройку, включающую внутренние реализации алгоритмов проектирования, блок взаимодействия с подключаемыми библиотеками и блок набора и обработки статистических данных. Схема серверной части подсистемы представлена на рисунке 4.

Блок управления (рис. 5) содержит TCP-сервер, реализованный стандартными средствами Borland C++ Builder 6.0, процедуры управления и сохранения информации о схеме в файл и извлечения ее из файла. Основная работа блока связана с организацией совместной работы сервера и клиентов. По завершении сервером начальных проектных процедур блоком управления создаются выходные файлы, содержащие информацию о результатах компоновки и расположения подсхем друг относительно друга. Они сохраняются в директорию "../Uploads_server". Далее производится последовательная передача файлов на клиентские машины. Блок управления ожидает сообщения от клиентов об окончании работы. Как только сигналы будут получены от всех клиентов, на которые рассылались файлы, блок управления последовательно дает разрешение каждому клиенту на передачу файла результатов работы, принимает и сохраняет файлы в директорию "../Downloads_server". После приема всех файлов клиентам посылаются сообщения об удачном приеме и команды отключения от сервера.

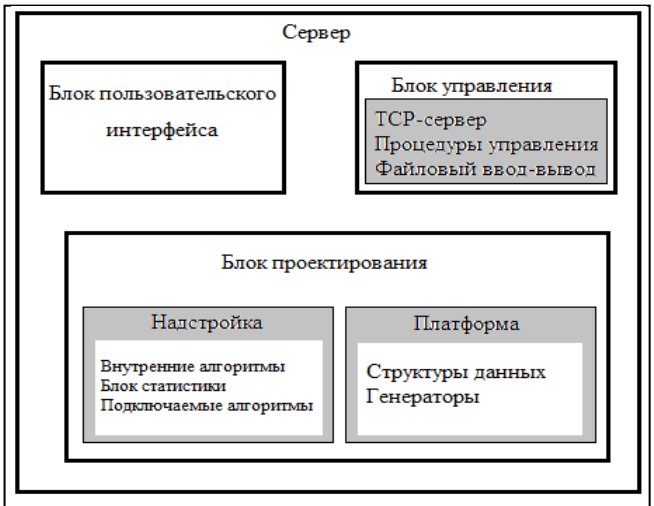

Рис.4. Схема серверной части подсистемы

Далее из файлов последовательно извлекается информация о частях схемы, вызывается генератор *дискретного рабочего поля* (ДРП), результаты проектирования на клиентах переносятся на сформированную структуру, и процесс проектирования возобновляется.

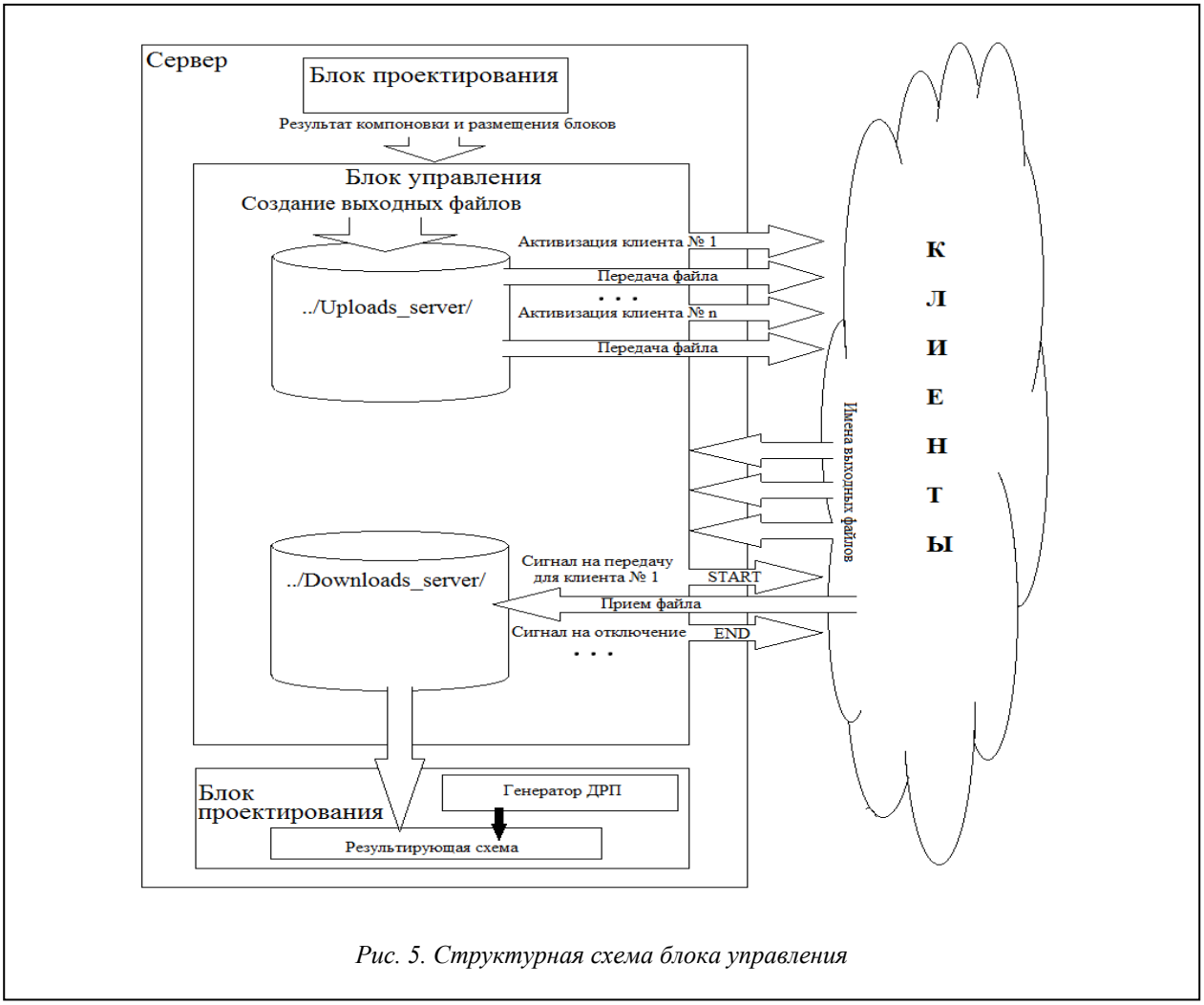

Рис. 5. Структурная схема блока управления

Блок проектирования, схема которого представлена на рисунке 6, содержит

- базу встроенных реализаций алгоритмов конструкторского проектирования;
- интерфейс для подключения сторонних реализаций различных процедур проектирования;
- реализацию процесса проектирования с возможностью циклического повтора и автоматической работы по набору статистических данных;
- генераторы графовых моделей схем, ДРП, схем с заданными топологическими свойствами;
- структуры данных, представляющие схему.

Под сторонними реализациями проектных процедур понимаются генераторы схем, средства ручного ввода схем, средства конвертации схем из других форматов, реализации алгоритмов конструкторского проектирования, отличающихся от встроенных принципами решения задач. Сторонние реализации оформляются в виде dll-библиотек, и должны поддерживать внутренние структуры данных подсистемы.

В блоке проектирования применяются следующие встроенные алгоритмы: конструктивный алгоритм компоновки, итерационный алгоритм парных перестановок, двухэтапный итерационно-последовательный алгоритм размещения пограничных элементов в подсхемах с использованием принципов факторизации и силовой релаксации, размещение элементов алгоритмом групповых перестановок, трассировка соединений волновым алгоритмом.

Поскольку решение, полученное в результате работы последовательного алгоритма, имеет невысокое качество, на следующем этапе оно улучшается с помощью алгоритма парных перестановок. Для предложенного алгоритма входными данными служит полученное ранее начальное распределение вершин по подграфам. Далее на каждой итерации для каждой пары подграфов производятся попытки обмена вершинами. Если при обмене вершинами общее число внешних связей уменьшается, перестановка фиксируется, в противном случае – отменяется.

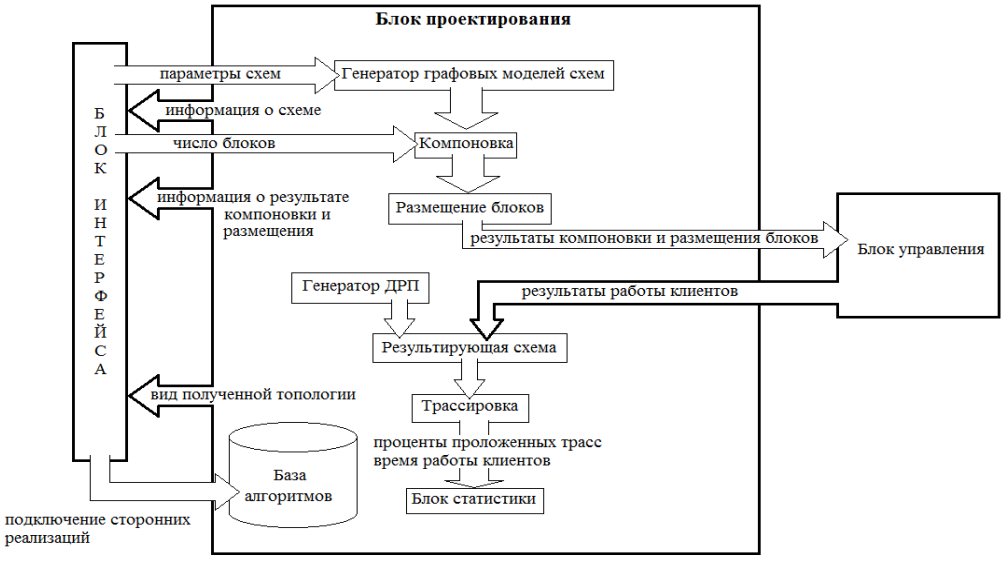

Рис. 6. Структурная схема блока проектирования

Двухэтапный итерационно-последовательный алгоритм размещения пограничных элементов в подсхемах с использованием принципов факторизации и силовой релаксации заключается в следующем. На первом этапе по результатам компоновки формируется взвешенный граф, каждая вершина которого соответствует одной подсхеме разбиения, ребра между вершинами показывают наличие внешних связей между конкретными подсхемами, а вес ребра соответствует числу внешних связей, то есть производится процедура факторизации графа, представляющего схему. Мнемоническая схема формирования взвешенного графа показана на рисунке 7. Далее производится размещение подсхем на основе взвешенного графа с использованием парных перестановок.

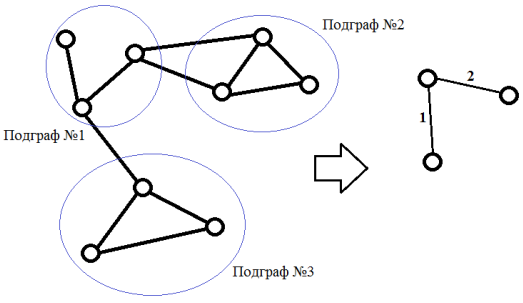

Рис. 7. Мнемоническая схема формирования взвешенного графа

При этом целевая функция имеет вид

$$f = \sum_{i=1}^{n} l_i * p_i,$$

где l_i – длина i-го ребра, p_i – вес i-го ребра. При оптимизации размещения ищется минимальная суммарная взвешенная длина связей. Данная целевая функция предполагает размещение на ДРП наиболее сильно связанных подсхем максимально близко друг от друга. После получения окончательного варианта взаимного расположения подсхем результаты компоновки дополняются информацией о размещении. При этом каждой вершине присваивается вес, равный числу внешних связей, инцидентных ей. Для каждой вершины задается также направление силы растяжения. Принцип задания весов и направлений показан на рисунке 8. Второй этап начинается с извлечения клиентом информации из полученного файла. Создается начальное размещение элементов, при этом используется сила растяжения, модуль которой равен весу вер-

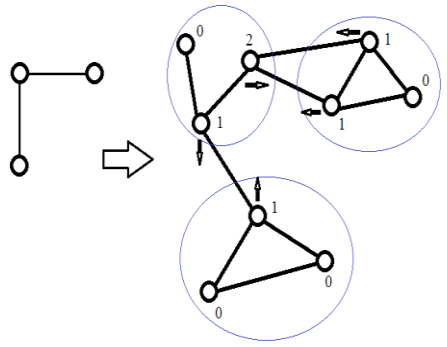

Рис. 8. Принцип задания весов и направлений

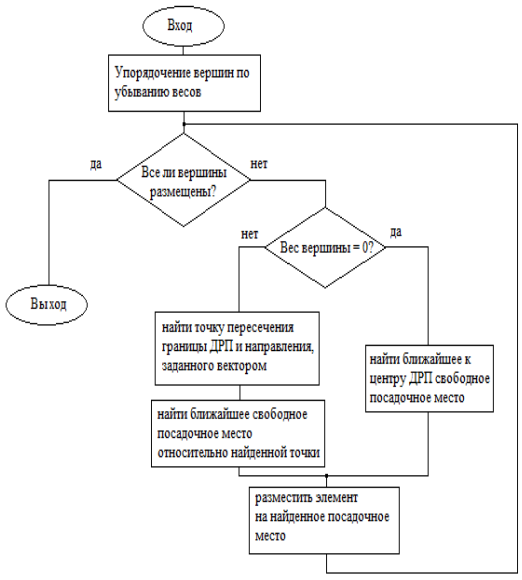

Рис. 9. Итерационно-последовательный алгоритм размещения пограничных элементов

шины, а направление задается векторами, поставленными в соответствие каждой вершине на первом этапе. Блок-схема второго этапа применяемого алгоритма приведена на рисунке 9.

Реализация структур данных

В подсистеме используются структуры данных, представляющие ДРП, цепи, целостное описание схемы, а также входные и выходные файлы. При реализации целостного описания схемы предъявлялись следующие требования к структуре данных

1. Для генерации исходной схемы минимальными входными данными является число элементов, число контактов элемента, число цепей, разветвленность цепи (под ней понимается число контактов, инцидентных данной цепи), дисперсия разветвленности. Могут быть также заданы размеры ДРП и расстояние между элементами. В результате должно быть получено графовое представление схемы и список цепей.
2. Для решения задачи компоновки необходимы данные о связях между элементами и количество элементов. На выходе получаем информацию о принадлежности элементов к подсхемам.
3. При решении задачи размещения элементов подсхемы необходима информация о позициях элементов на ДРП, размеры самого ДРП, число элементов в подсхеме, размеры элементов, список цепей. Результатом работы является информация о новых позициях элементов в подсхеме.
4. Входной информацией для трассировки соединений является список цепей, координаты контактов элементов, подключенных к этой цепи, информация об уже проложенных трассах и положениях элементов. После выполнения процедуры трассировки должно быть получено описание каждой трассы в виде последовательности единичных векторов.

Таким образом, разрабатываемая структура данных должна давать возможность извлечь такие данные, как: число входящих в схему элементов; элементная база; список цепей; число подсхем компоновки; принадлежность элемента к подсхеме компоновки; параметры ДРП; координаты элементов на ДРП; координаты контактов элементов; информация о проложенных и прокладываемых трассах.

Структура данных реализована с ориентацией на концепцию CALS-технологий с целью организации единого рабочего пространства на всех этапах проектирования. Она представляет класс DRP, который содержит следующие данные: размер поля в ячейках, размер элементной базы, количество цепей, подсхемы разбиения и их количество, представление ДРП в виде массива ячеек, массив элементов, список цепей, цепи в виде *минимальных связывающих деревьев* (МСД).

Взаимосвязь основных компонентов структуры данных показана на рисунке 10.

Ячейка ДРП представляет собой квадрат 3×3 позиции и выполнена в виде класса faset. В классе содержатся физические координаты центра ячейки, номер трассы, проходящей в ней, ссылки на ячейки, располагающиеся рядом, вес ячейки, флаги состояния ячейки (контакт элемента, трасса, пусто).

Элемент представлен прямоугольным посадочным местом, размеры которого определяются количеством контактов. Он оформлен в виде класса Element, в котором хранится информация о координатах точки привязки элемента (левый верхний угол), количестве контактов, принадлежности к подсхеме компоновки. Сами

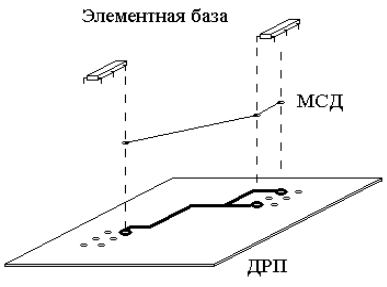

Рис. 10. Взаимосвязь компонентов структуры данных

контакты являются ссылками на ячейки ДРП. Имеется также флаг заполненности элемента (элемент считается заполненным, когда ко всем его контактам подходят цепи).

Для представления цепей используется минимальное связывающее дерево. С его помощью можно обеспечить близкую к реальной оценку длин цепей до этапа трассировки, адекватный перерасчет длин цепей при решении задачи размещения, значительное уменьшение времени и ресурсов, требующихся для трассировки. В качестве вершин для МСД используются ячейки ДРП, соответствующие выводам элементов. Данная структура применяется на всех этапах проектирования внутри системы. Для сохранения результатов трассировки в файл была разработана векторная структура трасс в связи с тем, что по завершении этапа трассировки трассы представляют собой совокупность ячеек ДРП, занимающих большой объем памяти и содержащих много избыточной информации. Преобразование трасс в векторную форму производится для более компактного их представления. При этом каждая трасса представляется в виде совокупности векторов, несущих в себе информацию о прямых участках. У каждого вектора есть координаты начала, направление и длина. Характер преобразования показан на рисунке 11.

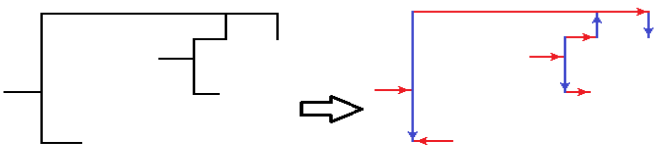

Рис. 11. Принцип преобразования трасс в векторную форму

В работе подсистемы используются файлы с двумя представлениями информации. Первые используются при передаче информации о результатах компоновки на клиентские части, вторые – для передачи результатов работы клиентов на сервер. Файлы имеют текстовый формат. В случае сохранения результатов компоновки и взаимного расположения подсхем структура информации имеет следующий вид:

`Число элементов в блоке # число контактов элементов # число цепей в схеме # разветвленность цепей # номер элемента # вес элемента # x-составляющая силы растяжения # y-составляющая силы растяжения # № цепи, подключенной к 1-му контакту # № цепи, подключенной ко 2-му контакту # ... # номер элемента # ...`

Такая структура позволяет извлечь всю необходимую информацию для создания начального размещения с учетом взаимного расположения подсхем, а также для трассировки внутренних соединений. Пример файла приведен на рисунке 12. Информация об одном элементе выделена подчеркиванием.

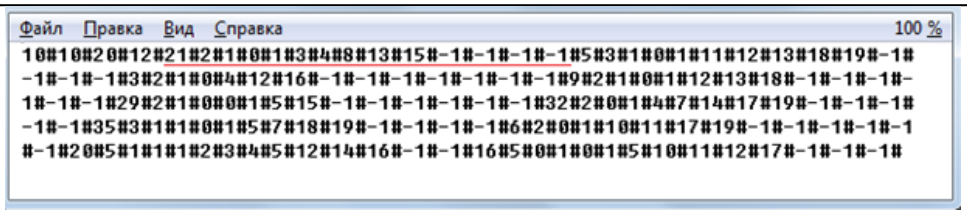

Рис. 12. Пример структуры файла, учитывающей взаимное расположение подсхем

При сохранении результатов работы клиентов структура информации имеет следующий вид:

`№ элемента в схеме # x-координата в блоке # y-координата в блоке # ... # № цепи # x-координата начала вектора # y-координата начала вектора # направление вектора # длина вектора # ... # end (окончание структуры) # процент проложенных трасс`

Координаты элементов даются в локальной системе координат подсхем и впоследствии пересчитываются в координаты схемы. Векторы являются представлением проложенных трасс. Таким образом, информация, сохраненная в файле, дает возможность полностью воссоздать результат работы клиента, содержащий размещение элементов и топологию проложенных трасс. Пример файла приведен на рисунке 13. Информация о векторе выделена подчеркиванием.

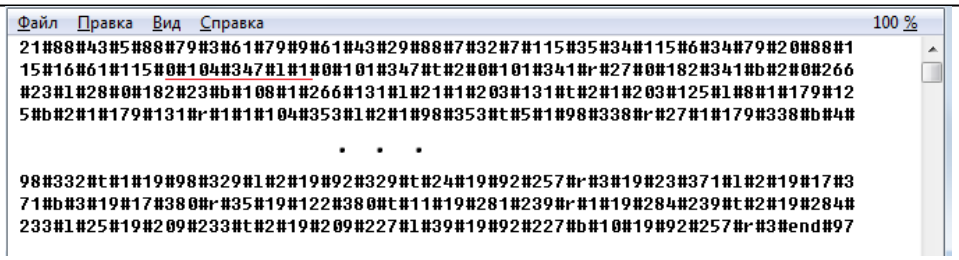

Рис. 13. Пример файла о размещении элементов и топологии проложенных трасс

Генерация исходной схемы осуществляется по таким входным параметрам, как количество элементов, количество выводов элемента, количество цепей, разветвленность цепей (математическое ожидание количества контактов, подсоединенных к цепи), отклонение разветвленности (дисперсия), а также коэффициент площади (отношение ширины каналов к размерам элемента). Расстояние между элементами принимается равным числу контактов элемента, умноженным на коэффициент площади. В результате проведенных исследований был выявлен следующий недостаток случайной генерации цепей: происходит практически однородное заполнение цепей элементами так, что становится сложно разбить схему на сильно связанные подсхемы. При этом вся схема представляет собой один сильно связанный блок. В этом случае невозможно определить влияние решения задачи компоновки и размещения на качество последующей трассировки, так как первые две задачи решаются неэффективно. Поэтому была принята следующая стратегия создания цепей: в схеме случайным образом формируется некоторое (от 3 до $n/3$, где n – количество элементов схемы) количество подсхем элементов. Далее при формировании очередной цепи с вероятностью 0,7 в нее включаются элементы только из одной подсхемы. С вероятностью 0,3 данная цепь не будет зависеть от количества подсхем, а цепи при этом равномерно распределятся между подсхемами. Данная стратегия выбрана из того соображения, что в любой реальной схеме уже неявно присутствуют сильно связанные блоки элементов. Это обусловлено степенью их функциональной законченности.

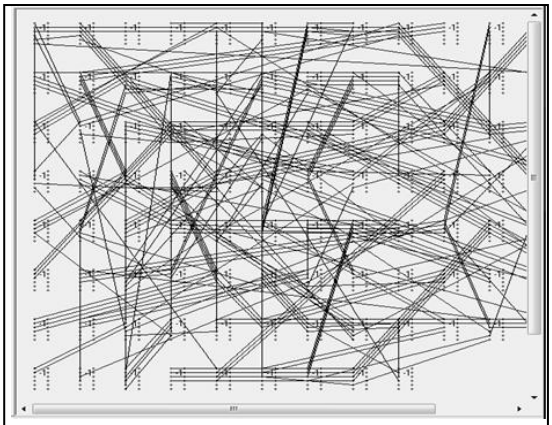

Рис. 14. Пример случайно сгенерированной схемы

Выбранная вероятность зависимости принадлежности цепи нескольким подсхемам введена априорно (из разумных практических соображений) для исключения случая формирования схемы из нескольких изолированных подсхем. Пример сгенерированной при принятой стратегии схемы приведен на рисунке 14.

Клиентская часть подсистемы

Клиент содержит в себе часть элементов сервера, а именно генераторы схем и ДРП, реализации алгоритмов конструкторского проектирования, процедуры файлового ввода вывода. Кроме того, клиент включает TCP-клиент, реализованный стандартными средствами Borland C++ Builder 6.0. Процесс проектирования на клиенте происходит автоматически в соответствии со следующими шагами: *прием файла – формирование подсхемы для обработки с применением генератора ДРП – размещение элементов – трассировка цепей – формирование выходного файла – передача файла на сервер.*

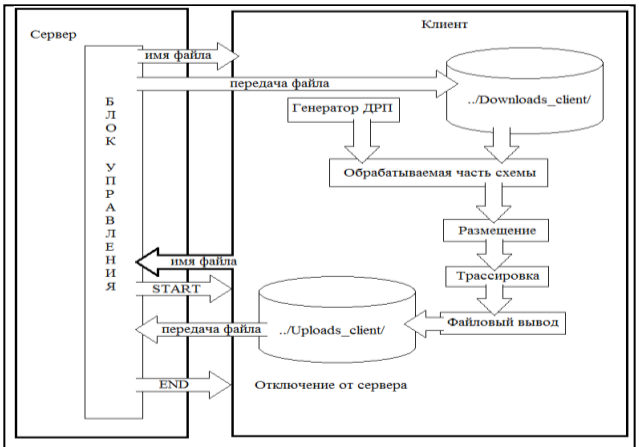

Рис. 15. Обобщенная схема клиента

Обобщенная схема клиента представлена на рисунке 15.

Результаты экспериментальных исследований

Естественной и главной целью проведения экспериментов было выявление оптимального числа подсхем (и соответственно, компьютеров-клиентов), на которые целесообразно разбивать исходную схему, чтобы обеспечить минимальное абсолютное время проектирования. При этом в качестве оценки использовалось усредненное по множеству случайных схем с одинаковыми параметрами сложности абсолютное время проектирования. В каждом опыте случайно генерировалось по 10 схем. В качестве примера на рисунке 16 приведен график, построенный на основе серии экспериментов по проектированию моделей схем, состоящих из 100 элементов с 10 контактами у каждого элемента, 50 цепями и разветвленностью цепей 15±5 контактов.

Общий характер изменения экспериментальной кривой удовлетворительно согласуется с теорией [7]. Однако при проведении нескольких серий экспериментов с одинаковыми параметрами схем вид зависимости не сохранялся. В результате анализа возможных причин был сделан вывод, что абсолютное время проектирования схемы определяется не только задаваемыми одними и теми же параметрами схемы, но в большой степени зависит от топологии случайно сгенерированных цепей. В результате этого при одинаковых входных параметрах получаются схемы, различные по сложности, и на их проектирование тратится разное время. И чем сильнее разброс по сложности схем будет в одной и той же серии экспериментов, тем больший разброс имеет место по усредненным величинам.

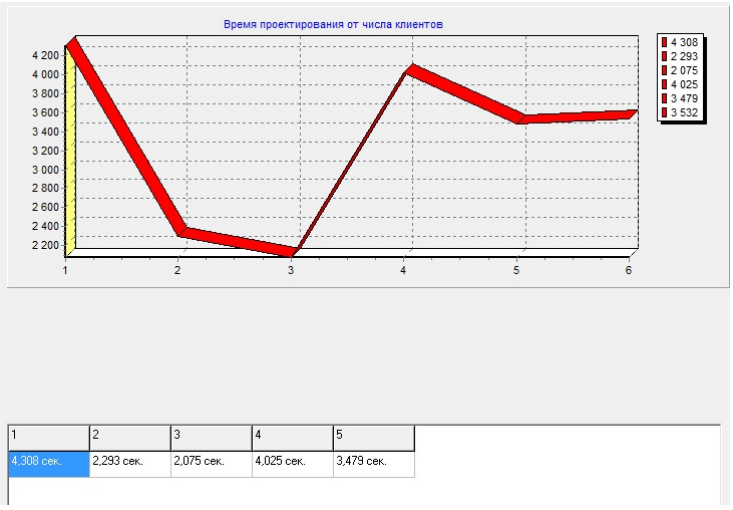

Рис. 16. Зависимость абсолютного временного выигрыша при проектировании на реальной распределенной подсистеме

Поэтому в дальнейшем оценивалось не абсолютное время проектирования на распределенной подсистеме, а время относительно сосредоточенной подсистемы. При этом по завершении каждого отдельного эксперимента вычислялась относительная величина времени работы, а по завершении серии экспериментов находилась средняя величина относительного времени в процентах.

Ввиду того, что сложность схемы определяется в основном числом элементов и загруженностью цепей, были проведены серии экспериментов для исследования поведения относительного времени проектирования при изменении загруженности схемы цепями и числа блоков разбиения при фиксированных значениях размерности схемы. При этом загруженность схемы цепями определялась процентом задействованности контактов элементов и последовательно увеличивалась и изменялась с шагом в 10 %.

Исходя из приведенных соображений, по экспериментальным данным в среде MathCAD 14 для случая 100 элементов была построена трехмерная поверхность, представляющая относительную зависимость временного выигрыша от числа подсхем, и загруженности схемы цепями (в процентах). Эта поверхность представления на рисунке 17.

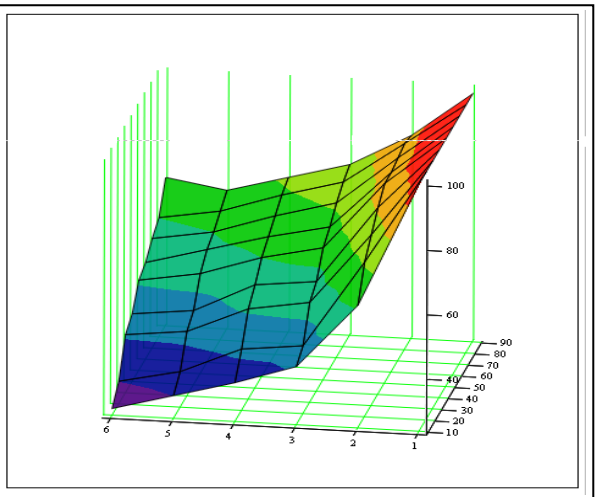

Рис.17. Трехмерная поверхность относительного временного выигрыша для 100 элементов

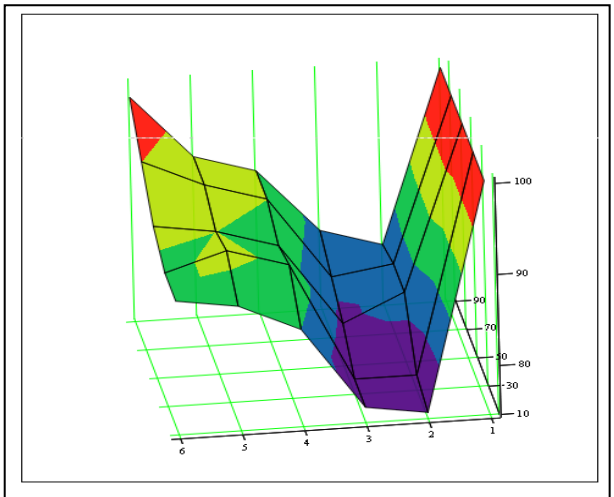

Рис.18. Трехмерная поверхность относительного временного выигрыша для 400 элементов

Аналогичная зависимость для случая 400 элементов приведена на рисунке 18, из которого видно, что при большем числе элементов в схеме наличие оптимума уже ярко выражено. Анализируя полученные поверхности, можно также заметить, что величина выигрыша и положение оптимального числа подсхем, а следовательно, и компьютеров-клиентов, в большей степени определяется размерностью схемы. Загруженность схемы цепями также влияет на размер выигрыша, но практически не влияет на положение оптимума. Кроме приведенных поверхностей, интерес представляет поверхность, отражающая динамику выигрыша в зависимости от числа подсхем, но при фиксированной загруженности исходной схемы цепями. Для этого были проведены соответствующие серии экспериментов, результаты которых представлены на рисунке 19.

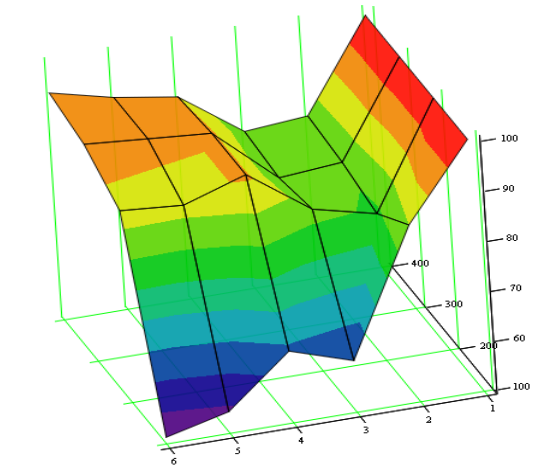

Рис. 19. Динамика выигрыша в зависимости от числа подсхем

В заключение отметим, что

- выигрыш на схемах с малым количеством элементов достигает трехкратного уменьшения времени проектирования;
- на схемах с малым количеством элементов оптимальное число компьютеров-клиентов смещается в сторону больших значений (по сравнению с более сложными схемами);
- выигрыш по времени обратно пропорционален загруженности схемы цепями, однако с ростом размерности схемы влияние загруженности на выигрыш ослабевает;
- при увеличении размерности схемы оптимальное число компьютеров-клиентов смещается в сторону меньших значений и составляет 3 клиента;
- выигрыш при увеличении размерности схемы уменьшается достаточно резко, затем стабилизируется в районе 1,3 раза.

В результате проведенных экспериментов был выяснен характер зависимости выигрыша от размерности схемы. Такой характер может быть объяснен с точки

зрения сложности применяемого алгоритма трассировки. В программе используется волновой алгоритм, время работы которого возрастает прямо пропорционально площади ДРП, причем зависимость квадратичная. Соответственно, чем большего размера схема разбивается на подсхемы, тем больше в общем случае будет доля работы дотрассировки на сервере относительно клиентов в силу разности площадей ДРП у клиентов и сервера. И следовательно, уменьшается доля работы, которую можно распараллелить. Именно из-за этой причины уменьшается выигрыш. На малых же схемах разница в площадях, с которыми работают компьютеры-клиенты и сервер, невелика, и поэтому растет выигрыш и смещается оптимум разбиения в сторону увеличения числа подсхем. Одним из путей решения этой проблемы, по мнению авторов, является применение алгоритмов трассировки, слабо зависящих от площади ДРП.

Литература

1. Нестеров Ю.Г., Папшев Ю.С. Выбор программно-технического комплекса САПР / Разработка САПР: В 10 кн.: практ. пособие; [под ред. А.В. Петрова]. М.: Высш. школа, 1990. Кн. 6. 159 с.

2. Глушань В.М. Использование GRID-технологий для построения подсистемы автоматизированного проектирования СБИС // Конгресс по интеллект. системам и информ. технологиям «AIS&IT'11»: сб. тр. в 4 т. М.: Физматлит, 2011.

3. Курейчик В.М., Глушань В.М., Щербаков Л.И. Комбинаторные аппаратные модели и алгоритмы в САПР. М.: Радио и связь, 1990. 216 с.

4. Глушань В.М., Иванько Р.В. Регрессивная и прогрессивная концепции развития САПР электронных схем // Интеллектуальные системы (AIS'06); Интеллектуальные САПР (CAD-2006): тр. Междунар. науч.-технич. конф. М.: Физматлит, 2006. Т.2.

5. Лаврик П.В. Концептуальные подходы параллельной обработки информации в контексте реализации распределенной САПР. // Конгресс по интеллект. системам и информ. технологиям «AIS&IT'10»: сб. тр. в 4 т. М.: Физматлит, 2010. Т. 3. С. 112–118.

6. Глушань В.М., Лаврик П.В. Имитационная модель распределенной САПР электронных схем / Свид. о гос. регистр. прогр. для ЭВМ № 2010612909. Заявка №2010611080, заявл.3.03.2010; зарег. 28.04. 2010.

7. Глушань В.М., Иванько Р.В., Лаврик П.В., Орлов Н.Н. Обобщение некоторых результатов исследования когнитивной модели распределенной САПР // Изв. ВолгГТУ. 2008. № 5.