УДК 535.317:004.22

МОДЕЛИРОВАНИЕ ПРОЦЕССОВ ФОТОЛИТОГРАФИИ НА БАЗЕ МНОГОАГЕНТНОЙ АРХИТЕКТУРЫ

С.М. Аваков[1], А. А. Дудкин[2], А. В. Инютин[2], В.И.Кривонос[1], А. В. Отвагин[2], Р.Х. Садыхов[2]

[1] Научно-производственное республиканское унитарное предприятие «КБТЭМ-ОМО» ГНПО «Планар», г. Минск

[2] Объединенный институт проблем информатики НАН Беларуси, г. Минск

Рассмотрен алгоритм моделирования процессов фотолитографии. Исходные данные для моделирования рассматриваются как поток данных, обрабатываемый параллельной вычислительной системой. Описана система для организации параллельной обработки изображений, основанная на применении графовой модели параллельного алгоритма. Алгоритм создается из отдельных вычислительных операций (гранул) в специальном визуальном редакторе. Визуальное представление преобразуется в формат XML, который интерпретируется многоагентной системой времени выполнения на базе MPI. Система времени выполнения осуществляет динамическую оптимизацию вычислений с помощью алгоритма виртуальной ассоциативной сети. Предложенные средства позволяют выполнять быстрое проектирование параллельных алгоритмов, их анализ и адаптацию к архитектуре вычислительного кластера, а также непосредственную организацию вычислительного процесса.

Введение

Процесс фотолитографии является основным этапом переноса топологии шаблона на поверхность полупроводниковой пластины. По фотошаблонам, полученным с помощью САПР СБИС, изготавливаются стеклянные фотомаски (оригиналы топологии, одна для каждого слоя микросхемы), которые затем проецируются на фоторезист. После экспозиции на поверхности фоторезиста образуется изображение (так называемое «воздушное изображение»), интенсивность которого зависит от характеристик освещения (длины волны, когерентных свойств освещения), от характеристик маски (размер и форма элементов, комплексная функция пропускания) и от характеристик проекционного объектива (числовая апертура и форма зрачка, амплитудное пропускание и аберрации). Далее в результате сложного процесса взаимодействия света с веществом фоторезиста образуется скрытое изображение, структура которого определяется набором оптико-физических параметров фоторезиста.

В настоящее время, известен ряд программных комплексов моделирования процессов фотолитографии (ПК МПФ), реализованных как на кластерах персональных ЭВМ и рабочих станций, так и на многопроцессорных ЭВМ [1-3]. Однако высокая стоимость указанных систем ограничивает их применение. В странах СНГ и в Республике Беларусь нет аналогов ПК МПФ для многопроцессорных вычислительных комплексов.

В ОИПИ НАН Беларуси совместно с КБТМ-ОМО концерна «Планар» в рамках научно-технической программы Союзного государства «Развитие и внедрение в государствах-участниках союзного государства наукоёмких компьютерных технологий на базе мультипроцессорных вычислительных систем» («ТРИАДА») [4] разработан ПК МПФ для систем автоматического контроля оригиналов топологии СБИС, ориентированный на кластерные высокопроизводительные мультипроцессорные вычислительные системы. В настоящей работе рассматривается реализация параллельной обработки для моделирования процессов формирования изображения в фоторезисте полупроводниковой пластины при фотолитографии. В качестве объекта обработки выступают изображения оригиналов топологии - фотомасок СБИС. Цель обработки заключается в моделировании изображения в фоторезисте для последующего автоматического контроля оригиналов топологии и определении фотолитографической значимости дефектов. Дефекты топологии являются значимыми, если приводят к образованию дефектов на полупроводниковой пластине, и которые необходимо корректировать на этом этапе производства [5-6].

Задачей моделирования процессов формирования изображений является вычисление распределения интенсивности «воздушного» изображения и получение скрытого изображения по заданным оптико-физическим свойствам фоторезиста, известным характеристикам оптической системы и условиям освещения.

В настоящее время параллельная обработка информации является одной из наиболее востребованных технологий проектирования ПО. Ее широкому применению препятствует необходимость решения дополнительных задач по планированию и оптимизации структуры

параллельного приложения в процессе проектирования. Существует необходимость создания средств, облегчающих процесс разработки, анализа и планирования параллельных приложений и скрывающих конкретные механизмы реализации параллелизма, чтобы программист мог сосредоточиться на алгоритме обработки информации. Использование параллельной обработки дает возможность существенно ускорить обработку потока изображений в системе оперативного анализа ИС.

Во многих случаях разработка параллельного приложения осуществляется на базе имеющихся последовательных алгоритмов или их композиции. Вычислительные операции, входящие в состав параллельного алгоритма, часто имеют универсальный характер и могут применяться в различных задачах обработки информации. Реализация элементов алгоритма в виде переносимых операций позволяет перейти к компонентному проектированию, когда программа конструируется из крупных блоков. В настоящее время подходы компонентного проектирования активно применяются в параллельном программировании.

Наконец, процесс выполнения параллельной программы требует применения современных методов планирования и оптимизации нагрузки. Во многих случаях узлы параллельных систем имеют гетерогенные характеристики, как пространственные, так и временные. Для эффективного выполнения параллельных программ в таких системах необходимо снабжать средства организации параллельных вычислений возможностями контроля и динамического изменения конфигурации алгоритма.

Существует ряд зарубежных систем обработки данных систем технического зрения, использующих параллельную и распределенную обработку [7-10]. В основе архитектуры этих систем используются различные технологии, например CORBA [10] или многоагентный подход [7]. Мы предлагаем использовать для проектирования и организации параллельных вычислений интегрированный набор средств, включающий в себя визуальный редактор, транслятор, систему оптимизации и систему поддержки параллельных вычислений на базе MPI[11]. Использование MPI делает нашу систему широко применимой для различных параллельных компьютеров. При проектировании средств реализованы следующие возможности:

1) визуальное представление алгоритмов, основанное на графовом представлении и модели вычислений с параллелизмом задач. При этом вводится концепция вычислительной гранулы – независимой единицы вычислений, которая может разрабатываться на различных языках программирования и должна лишь реализовать конкретный интерфейс для интеграции в параллельный алгоритм;

2) поддержка переносимости, реализованная в виде библиотек вычислительных гранул, подключаемых к системе организации вычислений во время выполнения. Сами средства разработки, анализа и выполнения программ строятся на платформо-независимых языках и стандартах (C++, Java, MPI);

3) статическая и динамическая оптимизация разработанных параллельных алгоритмов с использованием алгоритма виртуальной ассоциативной сети. Этот алгоритм является разновидностью гибридного генетического алгоритма и обеспечивает быстрый поиск решения, близкого к оптимальному. Алгоритм имеет две модификации: первая применяется на этапе анализа при проектировании (статическая оптимизация), вторая – на этапе выполнения полученной программы (динамическая оптимизация);

4) назначение операций алгоритма на процессоры вычислительной системы с учетом сведений, полученных на этапе оптимизации. Информация о назначении помещается в текстовое представление алгоритма и используется системой времени выполнения для реализации расписания.

В разделе 1 рассмотрен алгоритм моделирования изображения в фоторезисте. В разделе 2 рассмотрена задача параллельной обработки и описана графовая модель представления параллельного алгоритма и концепция вычислительных гранул. В разделе 3 описаны основные элементы платформы и их взаимодействие при разработке параллельных приложений. Раздел 4 более подробно описывает реализацию системы времени выполнения для организации параллельных вычислений. В разделе 5 приведен пример реализации алгоритма моделирования изображения в фоторезисте и результаты экспериментов для статической и динамической оптимизации обработки.

1. Алгоритм моделирования изображения в фоторезисте

Алгоритм моделирования изображения на поверхности фоторезиста состоит из следующих этапов [12-14]:

- вычисление зрачковой функции;
- вычисление векторной амплитуды объекта;
- вычисление передаточной матрицы проекционного объектива;
- вычисление двухмерного распределения интенсивности в заданном положении плоскости;
- вычисление двухмерного распределения интенсивности в разных положениях плоскости;
- расчет интенсивности изображения в частично – когерентном свете;
- вычисление объемного распределения интенсивности.

В предложенном алгоритме влияние векторных свойств света учитывается с помощью, так называемых векторных множителей, применяемых к зрачковой функции. Использование множителей позволило описать влияние векторного характера электромагнитных волн на качество изображения тонких периодических структур, размеры которых находятся в пределе разрешения оптических систем и существенно уменьшить время расчета воздушного изображения.

Для описания влияния передней апертуры оптической системы, необходимо учитывать множитель, который учитывает дифракцию плоской линейно поляризованной волны на входе оптической системы. При этом рассматривается влияние оптической системы на перераспределение энергии в спектре вне зависимости от направления поляризации и направления распространении волны. Другой множитель учитывает влияние входной апертуры на входе оптической системы.

На основе этих данных можно промоделировать влияние числовой входной апертуры на распространение любой Фурье-компоненты именно как векторного поля - векторная природа электромагнитных волн рассматривается на входе и на выходе оптической системы. В итоге появляется возможность вычислять векторное поле изображения не только в когерентном, но и в частично-когерентом свете.

Использование множителей позволяет описать влияние линейно-поляризованной волны на качество изображения. Для случая неполяризрованного или частично-поляризованного света необходимо использовать как электрический, так и магнитные векторы, что и реализовано в данном алгоритме.

Особенности разработанного алгоритма:

1) учет векторной природы светового поля на основе формулировки электрической и магнитной векторных амплитуд как функций от трех декартовых координат в пространстве, а также двух координат на зрачке оптической системы. Данная формулировка обеспечивает корректный учет аберраций оптической системы и влияние высоких числовых апертур на формирование изображения без существенного усложнения математического аппарата. В отличие от применяемых в настоящее время моделей предлагаемая модель является значительно более простой, что благоприятно для построения быстрого алгоритма, и вместе с тем она базируется на строгом соответствии физической природе протекающих процессов;

2) использование теории частичной когерентности для вычисления интенсивности изображения наиболее экономичным образом основываясь на системе собственных функций. Собственные функции используются для описания взаимной интенсивности различных точек изображения и моделируются через полиномы Цернике. Такой прием дает возможность значительно сократить количество интегралов по источнику света, вычисление которых остается после применения пункта 1) единственным трудоемким этапом моделирования.

Указанные принципы дополняют друг друга при разработке наиболее эффективного алгоритма вычисления объемного распределения интенсивности воздушного изображения и по отдельности не представляют собой такой ценности, какую они обретают совместно.

Алгоритм моделирования изображения в фоторезисте состоит из следующих этапов:

- вычисление прошедшей интенсивности в начале экспозиции;
- вычисление поглощенной интенсивности в начале экспозиции;
- вычисление прошедшей интенсивности в конце экспозиции;
- вычисление поглощенной интенсивности в конце экспозиции.

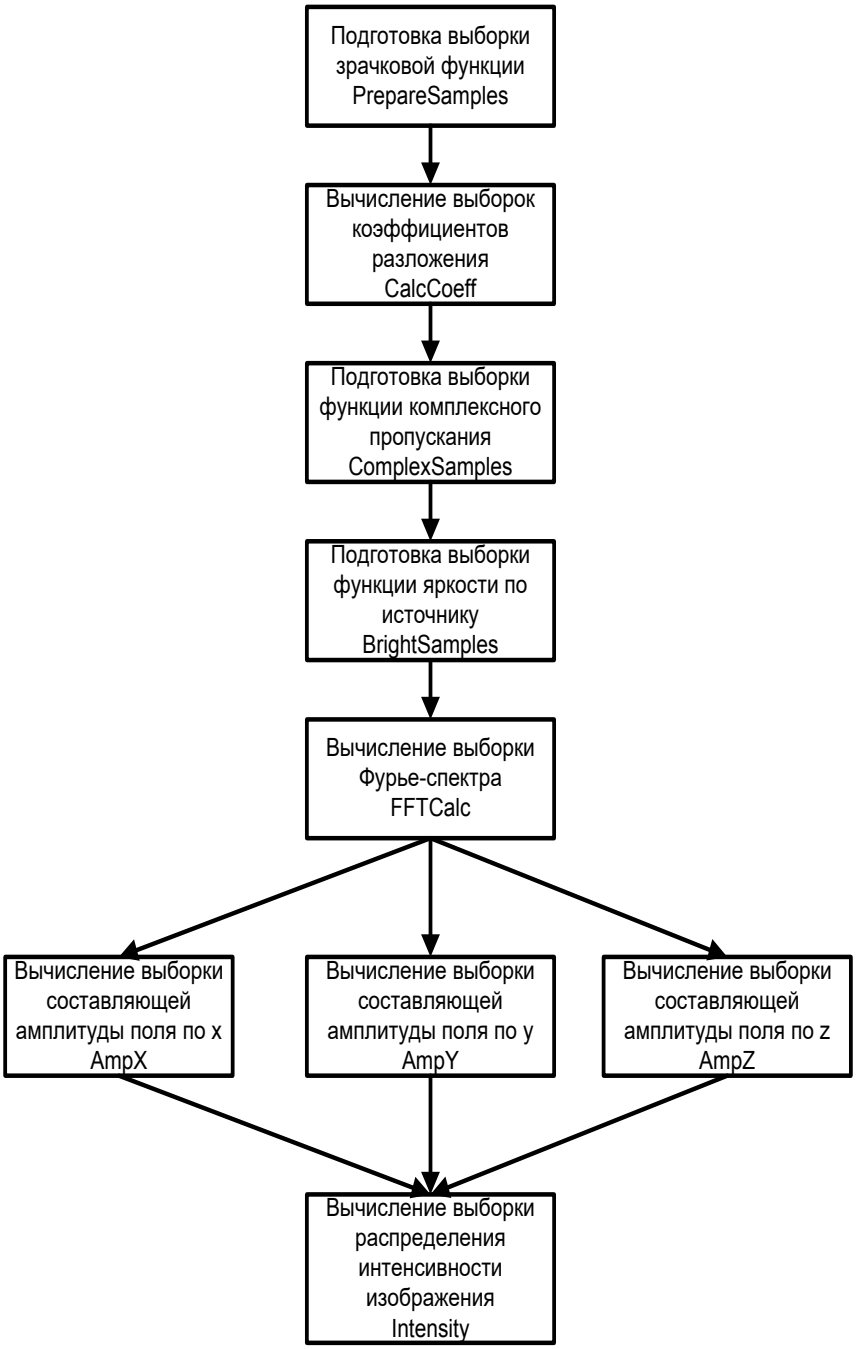


Рис. 1 - Алгоритм для построения изображения на поверхности фоторезиста

Реализация алгоритма построения и анализа изображений в фоторезисте в виде однопроцессного приложения показала, что при больших объемах исходных данных время обработки и построения модельного изображения является неприемлемо большим (свыше 1 мин./кадр). Так как все исходные объекты обрабатываются по единой программе и могут предъявляться системе поиска дефектов одновременно, целесообразным является использование параллельной вычислительной системы для одновременной обработки потока исходных данных.

2. Модель представления параллельного алгоритма

Параллельная обработка информации выполняется на кластере [15], в общем случае гетерогенном. Он представляется множеством $P=\{p_1, p_2,..., p_m\}$, где p_i - отдельный компьютер (узел кластера). Каждый компьютер p_i имеет характеристику производительности s_i, которая определяет время выполнения одной условной единицы вычислений. Узлы кластера соединены коммуникационными каналами. Каждый канал между узлами p_i and p_j, обозначенный l_{ij}, имеет

характеристику b_{ij}, которая определяет ширину канала и скорость передачи данных между p_i and p_j.

При оптимизации расписания для конвейера следует учитывать не только характеристики алгоритма обработки, но и количество поступивших объектов. Сам по себе поток объектов может не быть однородным, т.е. отдельные объекты обрабатываются по различным алгоритмам. Мы называем алгоритм обработки объекта определенного типа сценарием. Сценарии обработки в общем случае содержат множество операций, каждая из которых может избирательно применяться к объектам различных типов, либо применяться с различными параметрами в зависимости от типа обрабатываемого объекта. В этом случае поток может быть детерминированным, когда его структура и характеристики объектов известны заранее, или стохастическим, когда количество и порядок следования типов объектов случайны. Средства организации и оптимизации вычислений позволяют управлять обработкой как детерминированного, так и стохастического потока.

Поток, обрабатываемый параллельным алгоритмом, представляется множеством объектов $J = \{j_1, \ldots, j_n\}$. Каждый объект обрабатывается по сценарию, определяемому типом данного кадра. Система обработки способна выполнять множество операций обработки $O = \{o_1, \ldots, o_k\}, k > m$. Сценарий обработки отдельного типа кадров содержит подмножество операций $O_j = \{o_{j_1}, \ldots, o_{j_k}\}, \bigcup_{j=1}^{n} O_j = O$. Операции в каждом сценарии упорядочены отношениями следования $o_{ja} \succ o_{jb}$, т.е. для кадра с типом j операция a выполняется перед операцией b. Каждая операция обработки o_i характеризуется сложностью вычислений $w_{o_i}^{j}$, представленной количеством единиц вычисления данной операции для определенного типа кадров j. Две операции над различными кадрами, назначенные на один и тот же процессор, не могут выполняться одновременно. Два экземпляра одной и той же операции над различными кадрами также не могут выполняться в один и тот же промежуток времени.

Все сценарии обработки представляются в форме направленных ациклических графов (НАГ). НАГ сценариев представляется кортежем $G = (V, E, W, C)$, где

– V является множеством вершин графа $v_i \in V, 1 \le i \le N$. Каждая вершина ассоциируется с операцией обработки данных. Множество вершин представляет декомпозицию параллельной программы обработки потока изображений на отдельные операции (гранулы параллелизма);

– E является множеством дуг графа $\{e_{i,j} = (v_i, v_j)\} \in E, i = \overline{1, N}, j = \overline{1, N}, i \ne j$. Дуга представляет отношение предшествования между операциями алгоритма и определяет передачу результатов обработки от источника к приемнику;

– W определяет матрицу вычислительной стоимости операций $W = \bigcup W_{o_i}^{j}$;

– C является множеством стоимости дуг, где элемент $c_{i,j} \in C$ определяет объем информации между двумя операциями обработки данных, передаваемой по дуге $e_{i,j} \in E$. Связанные дугой операции используют один формат данных, поэтому для всех сценариев и типов кадров дуги имеют равную стоимость.

Построение приложения для параллельной обработки потока данных состоит из нескольких шагов: создание НАГ сценариев, описывающего логическую структуру программы; назначение операций обработки вершинам графа и определение параметров вызова для каждого типа данных; отображение графа сценариев на топологию вычислительного кластера. Параллельная программа представляется декомпозицией $((O),(P)) \longrightarrow \bigcup_{p \in P} (O_p)$, где $\forall O_k, O_p : O_k \ne O_p \supset O_k \bigcap O_p = \varnothing$. Каждое подмножество операций O_p размещается на выбранном узле кластера.

Целью обработки потока является минимизация времени обработки N объектов данных

$$T_{opt} = \min\{\max T_N\}, \tag{1}$$

где T_N означает момент окончания обработки объекта N. Оптимизационный алгоритм использует для оценки расписания имитационную модель, эквивалентную реальному вычислительному процессу.

Пример НАГ сценариев обработки и обозначения элементов графа для параллельной сортировки трех типов данных приведен на рис. 2.

Каждая вершина НАГ сценариев обработки данных имеет уникальный индекс, используемый при определении топологии графа, а также перечень типов данных $d_1, d_2, \ldots, d_k$, которые обрабатываются этой операцией (поле Types) и соответствующие времена обработки каждого типа $t_1, t_2, \ldots, t_k$ (поле Time). Эти значения используются при имитационном моделировании и составлении расписания в случае детерминированного потока. В случае обработки стохастического потока временные характеристики определяются в ходе обработки.

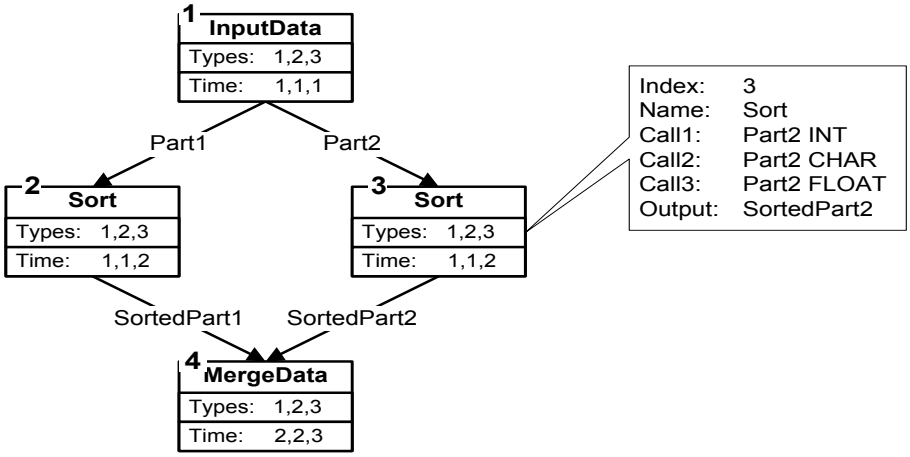


Рис.2. Пример графа сценариев операции параллельной сортировки.

Каждой вершине графа назначена операция (в поле Name указано ее название), определяющая вычислительную гранулу. Поля вида CallN содержат список параметров, используемых для вызова операции при обработке соответствующего типа данных d_N. В приведенном примере операция сортировки обрабатывает три типа данных (INT, CHAR и FLOAT), содержащиеся в массиве Part2. Поле Output определяет имя переменной, которая содержит результат работы вычислительной гранулы. Это имя также указано на соответствующей дуге, которая соединяет источник и приемник данных.

Созданный НАГ сценариев обработки представляется в форме документа XML, который используется как для визуального представления, так и для выполнения программы средствами организации вычислительного процесса.

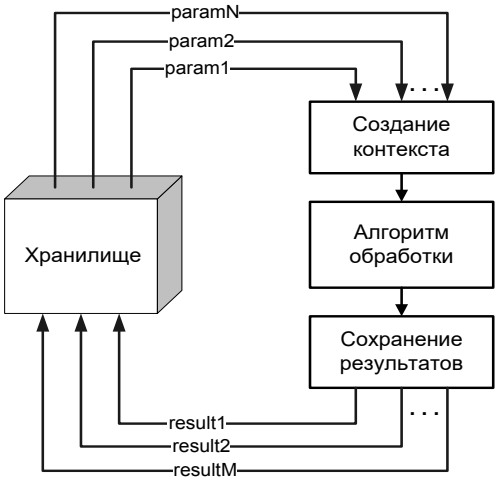


Рис. 3. Структура вычислительной гранулы и взаимодействие с хранилищем данных.

Вычислительные операции взаимодействуют со специальным механизмом хранения данных, входящим в архитектуру платформы организации параллельных вычислений. Этот механизм реализует интерфейс доступа к разделяемой памяти, в которой находятся данные и результаты операций. Вариант разделяемой памяти реализован на базе разделяемой файловой

системы, которая присутствует в большинстве современных кластеров. Механизм доступа обеспечивает запись или чтение переменной с определенным именем, однозначно определяющим ее. Реализован также механизм промежуточного хранения результатов вычислений в локальной памяти каждого вычислительного узла, на котором они были записаны или прочитаны. Это позволяет сократить затраты на получение переменной при повторном обращении к ней. Структура вычислительной гранулы и ее взаимодействие с механизмом хранения данных показаны на рис. 3.

Вычислительные гранулы объединяются в специализированные библиотеки, которые динамически подключаются при выполнении параллельного приложения. Гранула загружается из библиотеки при необходимости ее выполнения и идентифицируется по имени операции. Реализация различных классов алгоритмов обработки информации в виде библиотек вычислительных гранул позволяет расширить сферу применения системы организации параллельных вычислений и дает возможность создавать новые классы приложений.

3. Архитектура платформы организации параллельных вычислений.

Общая архитектура платформы показана на рис. 4.

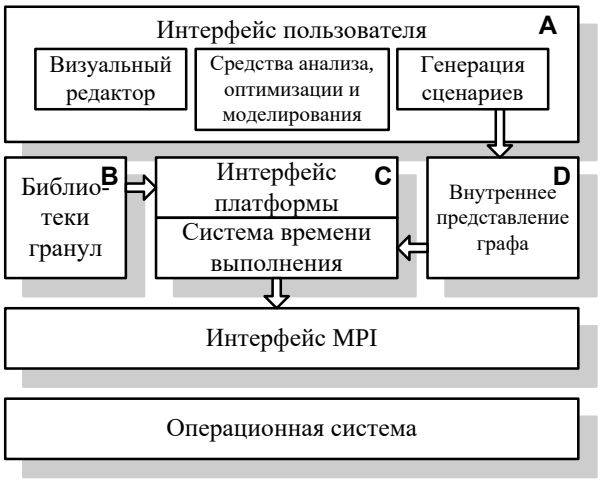


Рис. 4. Архитектура платформы организации параллельных вычислений.

Интерфейс пользователя (блок А) представляет собой средство визуального проектирования и анализа параллельной программы обработки потока данных. Это специализированный графический редактор, использующий графовую модель представления программы. Редактор позволяет создать логический граф программы, определить соответствие вершин графа операциям из библиотек вычислительных гранул (блок В), задать параметры вызова гранул для различных типов обрабатываемых данных. Для последующего анализа детерминированных потоков в редакторе задается шаблон потока в виде перечня типов данных, поступающих на вход приложения. Редактор также позволяет задать топологию и характеристики вычислительной системы, на которой будет выполняться разработанное приложение.

Средства анализа состоят из алгоритма оптимизации расписания и имитационной модели. Имитационная модель используется для оценки вариантов расписания в процессе оптимизации, а также для визуализации расписания. Модуль оптимизации позволяет визуально сравнить две модели расписания, а также корректировать модель вручную.

Существует множество алгоритмов составления расписаний для НАГ, использующих различные эвристики и алгоритмы оптимизации. Среди известных широко используются алгоритмы, основанные на списках с приоритетами [16-18]. Другим подходом являются алгоритмы кластеризации [19, 20]. Однако многие алгоритмы статической оптимизации разработаны для специальных топологий графов или используют определенные ограничения и допущения, например, нулевую задержку при передаче информации между узлами или неограниченное число процессоров.

Другой перспективной технологией поиска решений является эволюционная оптимизация. К алгоритмам этого класса относятся поиск с запретом [21], моделирование отжига и генетические алгоритмы [22]. Наиболее мощными являются генетические алгоритмы (ГА), среди которых

предложено много вариантов решения задач составления расписаний. Однако классические ГА являются технологией слепого поиска. Для ускорения поиска решения с помощью ГА и повышения его качества система оптимизации использует алгоритм виртуальной ассоциативной сети (ВАС) [23-25], относящийся к гибридным ГА.

Алгоритм ВАС основан на концепции ассоциаций между определенными операциями и выделенными процессорами. Каждая операция O и процессор P связаны виртуальной ассоциативной связью прочности $\omega_{O,P}$. Алгоритм использует для оптимизации определенную структуру ассоциативной памяти, содержащую ассоциативные связи. Эта память обучается на основе опыта, накопленного в ходе поиска решения. Алгоритм основан на представлении решений в виде популяции хромосом, как в классическом ГА. Каждая хромосома представляет вариант отображения графа сценариев на топологию системы.

Хромосомы оцениваются функцией пригодности, которая использует имитационную модель и цель оптимизации согласно (1). После этапа оценки и выбора кандидата на лучшее решение ВАС обучается на основе положительного опыта. Если выбранное решение на некотором этапе не превосходит лучшей модели, то ВАС обучается на основе предыдущего опыта. Накопление опыта позволяет реализовать направленный поиск в пространстве возможных решений. Этот поиск выполняется значительно быстрее и способен привести к лучшим решениям на ранних стадиях поиска. Алгоритм ВАС также вводит новый генетический оператор – кластеризацию, который выполняется с использованием опыта ассоциативной памяти. Этот оператор позволяет быстрее получить стабильную схему в составе хромосомы и реализовать стратегию генетического локального поиска.

Созданная в визуальном редакторе программа преобразуется в текстовое представление на языке XML. Это представление содержит информацию для системы времени выполнения, а также отображение графа на топологию вычислительной системы. Файл XML при выполнении программы транслируется во внутреннее представление (блок D), позволяющее каждому элементу системы времени выполнения (блок C) получать необходимую информацию во время работы.

Интерфейс платформы реализует возможности по доступу к данным, хранящимся в хранилище (разделяемой памяти), а также механизмы загрузки гранул из библиотек и определения параметров гранул при вызове. При реализации гранулы используют этот интерфейс для осуществления взаимодействия в рамках платформы.

Система времени выполнения является многоагентной системой, использующей средства MPI для осуществления координации взаимодействия элементов. Ее архитектура изолирует логику поведения, определяемую сценарием обработки, от базовых средств ОС для организации и оптимизации параллельного процесса.

4. Построение и реализация системы времени выполнения для параллельных вычислений

Многоагентная система времени выполнения для организации параллельных вычислений содержит два типа программных агентов: координатор и исполнитель. Агенты реализованы как процессы MPI и используют средства MPI для обмена информацией и управления вычислениями. Количество агентов-исполнителей при выполнении параллельного приложения равно количеству физических процессоров целевой вычислительной системы плюс один агент-координатор. Взаимодействие между агентами осуществляется с помощью передачи сообщений, реализующих простой протокол взаимодействия. Взаимодействие агентов системы времени выполнения представлено на рис. 5.

Система построена по схеме «главный - подчиненный». Координатор является главным процессом, управляющим реализацией логической структуры параллельного алгоритма. Он содержит очередь дескрипторов обрабатываемых объектов. Каждый дескриптор определяет тип объекта и текущую операцию, которая должна быть выполнена. Для каждой операции создается собственный дескриптор. Очередь дескрипторов представляет собой множество операций, которые готовы к выполнению. Основная задача координатора состоит в передаче дескрипторов операций свободным исполнителям в соответствии с отображением операций графа на топологию системы. Кроме того, координатор следит за завершением операций и помещает в очередь новые дескрипторы в соответствии с топологией графа приложения.

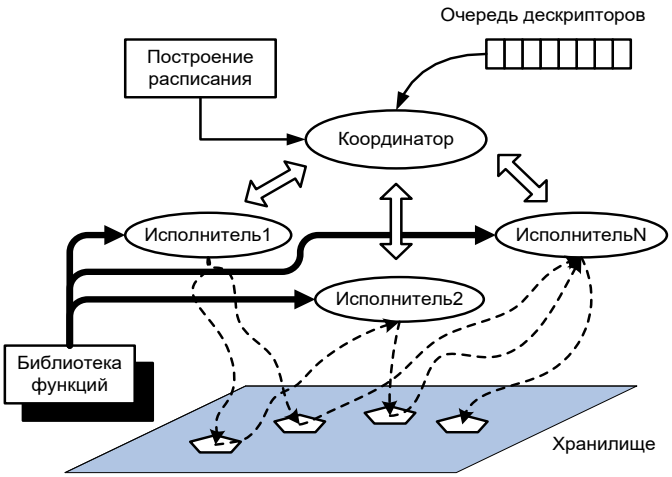


Рис. 5. Архитектура системы времени выполнения.

Агенты-исполнители являются абстракцией физических процессоров системы. Их основная задача – выполнение вычислительных гранул, которые передаются координатором. Исполнители взаимодействуют с библиотекой гранул, загружая требуемые гранулы и выполняя их на подчиненном процессоре. Взаимодействие между агентами происходит через механизмы разделяемой памяти, представленные интерфейсом хранилища. После завершения операции исполнитель посылает сообщение координатору, который отмечает момент готовности результатов операции для дальнейшего использования.

При обработке стохастических потоков данных часто возникает ситуация, когда фиксированное отображение операций на топологию вычислительной системы не обеспечивает максимальной производительности. В этом случае конфигурация приложения должна изменяться с учетом изменившихся условий функционирования. Поскольку конфигурация определяется размещением вычислительных гранул по процессорам и задается внутри координатора, то реконфигурация приложения заключается в перемещении некоторых гранул на другие процессоры. В этом случае агент-координатор преследует цель увеличения скорости обработки потока объектов и следит за ее изменением. Решение о реконфигурации принимается после снижения скорости на определенную величину Δs. При реконфигурации координатор руководствуется информацией, полученной от агентов-исполнителей.

Агент-координатор содержит алгоритм динамического управления конфигурацией параллельного приложения. Этот алгоритм подключается к агенту через специальный интерфейс диспетчера и осуществляет перераспределение операций по процессорам. Мы используем модифицированный алгоритм виртуальной ассоциативной сети, который ориентирован на непрерывное накопление информации о производительности приложения. Изменение расписания основано на информации, накопленной в процессе функционирования программы. После реконфигурации координатор использует новую схему передачи дескрипторов операций с учетом изменившегося распределения операций по процессорам.

5. Пример приложения и результаты экспериментов.

Для моделирования будем использовать топологическую структуру следующего вида (рис. 6) - квадрат размером 20*20 мкм. Черным цветом обозначена проэкспонированная часть, на которой располагается дефект вида прокол. Фоторезист UV 210 Shipley (позитивный, λ = 248 нм). Параметры процесса моделирования представлены на рис. 7. Результаты процесса моделирования для топологической структуры, показанной на рис.6, приведены на рис.8. На нем показано техмерное распределение интенсивности излучения на поверхности фоторезиста и в фоторезисте, а также поперечное сечение, проведенное посередине топологической структуры. Ось X – расстояние от центральной точки объекта, ось Y – значение интенсивности излучения.

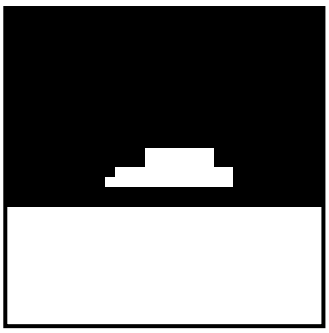


Рис. 6. Пример топологической структуры для тестирования

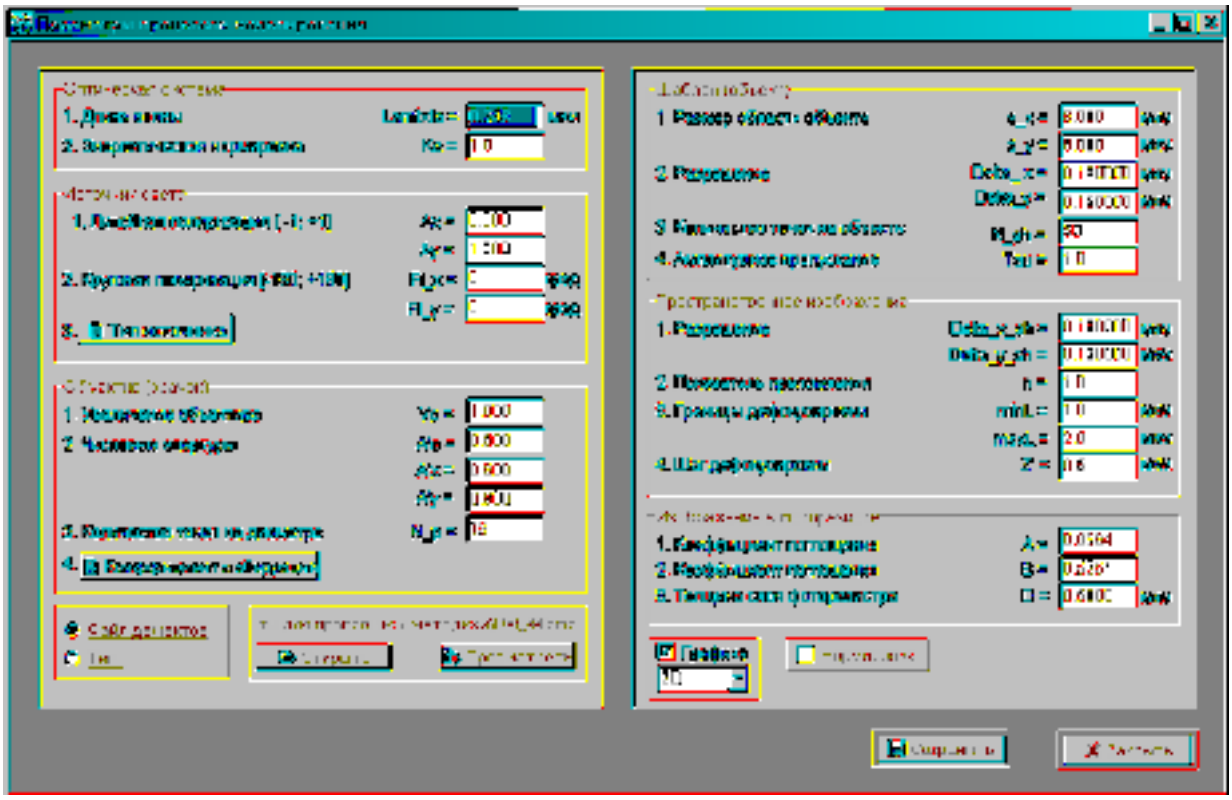


Рис. 7. Параметры процесса моделирования

Проверка результата функционирования разработанного ПК МПФ проводилась путем сравнения с результатами работы программы SIGMA C на изображениях топологии оригинала содержащего дефекты стандарта SEMI-P22-0699. Дефекты получены на установке автоматического контроля ЭМ-6329. Результаты моделирования совпали, а, следовательно, ПК МПФ функционирует правильно.

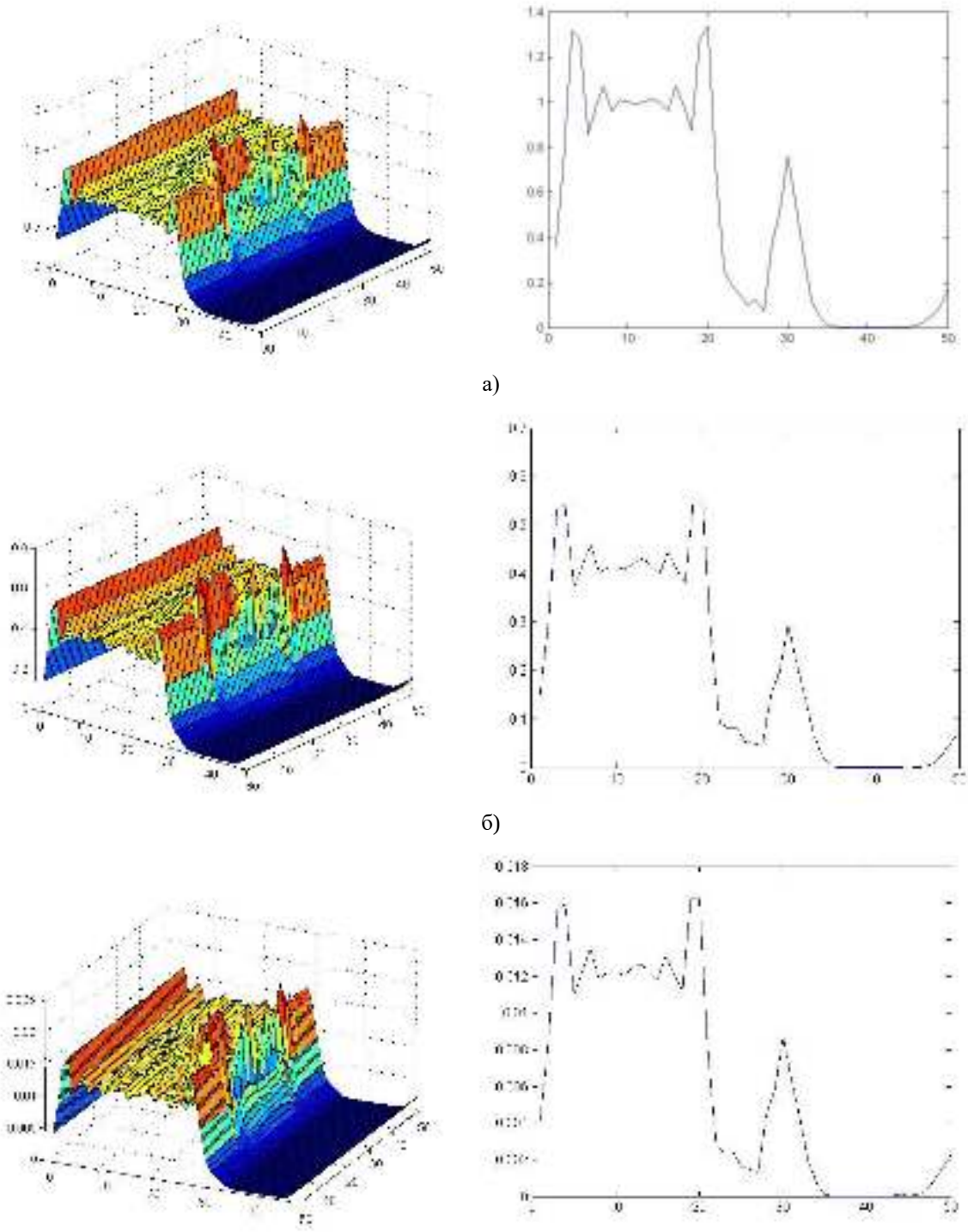
При исследовании параллельного приложения были проведены эксперименты по измерению производительности в зависимости от количества использованных процессоров, а также в зависимости от исходных данных (размера исследуемого фотошаблона).

Первая группа экспериментов показывает зависимость скорости обработки 50 объектов данных от количества процессоров, выделенных для запуска приложения.

Таблица 1. Время вычисления для разного количества процессоров

Количество объектов	Количество процессоров			
	1	2	3	4
50	77,8	44,67	41,63	24,48
100	154,92	89,23	83,19	49,51
200	310,11	178,47	166,29	101,42

Как видно из результатов, общее ускорение обработки составляет примерно 3 раза для случая 4 процессоров. Этот результат обусловлен тем, что параллельное приложение имеет нелинейную структуру с различной продолжительностью операций, поэтому его параллелизм также нелинеен. Тем не менее, полученное ускорение позволяет существенно повысить скорость обработки фотошаблонов.



а)

б)

в)

Рис.8. Результаты моделирования: а) распределение интенсивности излучения на поверхности фоторезиста - «воздушное» изображение; б)-в) распределение интенсивности излучения в слое фоторезиста: прошедшая интенсивность, начало экспозиции и поглощенная интенсивность, конец экспозиции

Вторая группа экспериментов показывает зависимость производительности параллельного приложения от размера входных данных (количество обрабатываемых объектов фиксировано и равно 100, количество процессоров 4).

Таблица 2. Время вычисления для различных входных данных

Количество объектов	Количество точек зрачка		
	50	100	200
100	49,51	156,36	534.56

Судя по результатам, алгоритм показывает практически линейную масштабируемость. При увеличении количества точек зрачка в 2 раза по координатам X и Y объем данных для обработки увеличивается в 4 раза. При этом время обработки возрастает в 3,16 раза для 100 точек, и в 3,42 раза для 200 точек (относительно времени для 100 точек). Это свидетельствует о хорошей масштабируемости алгоритма, поскольку время работы практически линейно зависит от поступающего объема данных.

Результаты экспериментов по тестированию сценария расчета характеристик воздушного изображения приведены в таблице 3 (время обработки потока объектов в секундах).

Таблица 3. Расчет характеристик воздушного изображения

Количество объектов	Количество процессоров			
	1	2	3	4
20	223.206	168.795	178.342	187.471
50	566.994	427.355	442.9	461.272
100	1150.89	865.339	880.74	902.452

Из результатов видно, что оптимальным является время работы на 2 процессорах. Очевидно, это связано с большими объемами коммуникаций между задачами. В этом случае для быстрого доступа к данным требуется высокая степень их локальности.

Сценарий протестирован на последовательности 20 изображений, при этом получены следующие результаты времени выполнения (таблица 4).

Таблица 4. Результаты тестирования сценария построения изображения в фоторезисте

Количество изображений	Количество процессоров			
	1	2	3	4
20	621,35	320,231	263,04	204,36

Результаты свидетельствуют о высокой степени параллелизма сценария, позволяющей достичь существенного ускорения обработки даже для относительно коротких потоков моделируемых изображений.

Для оценки разработанной платформы организации параллельных вычислений были проведены две серии экспериментов: для детерминированных потоков, имеющих фиксированное количество объектов с определенной последовательностью типов и для стохастических потоков, когда исходные объекты и их типы генерировались случайным образом. Эксперименты проведены на суперкомпьютере СКИФ К-500, находящемся в ОИПИ НАН Беларуси [26].

Экспериментальные потоки данных имели нерегулярную структуру и генерировались случайным образом. Рис. 9 содержит результаты статической оптимизации для детерминированных потоков, представленные в виде сравнения продолжительности обработки потоков расписаниями, полученными алгоритмами классического ГА и ВАС. На диаграмме указаны результаты для различного количества процессоров (сокращение пр.), участвовавших в обработке.

Во второй серии экспериментов для стохастических потоков использовались полученные в первой серии оптимальные статические расписания, которые сравнивались с расписаниями, реализованными динамически изменяемой конфигурацией приложения. Рис. 10 содержит результаты оптимизации, показывающие изменение времени обработки для статического (С) и динамического (Д) алгоритма ВАС.

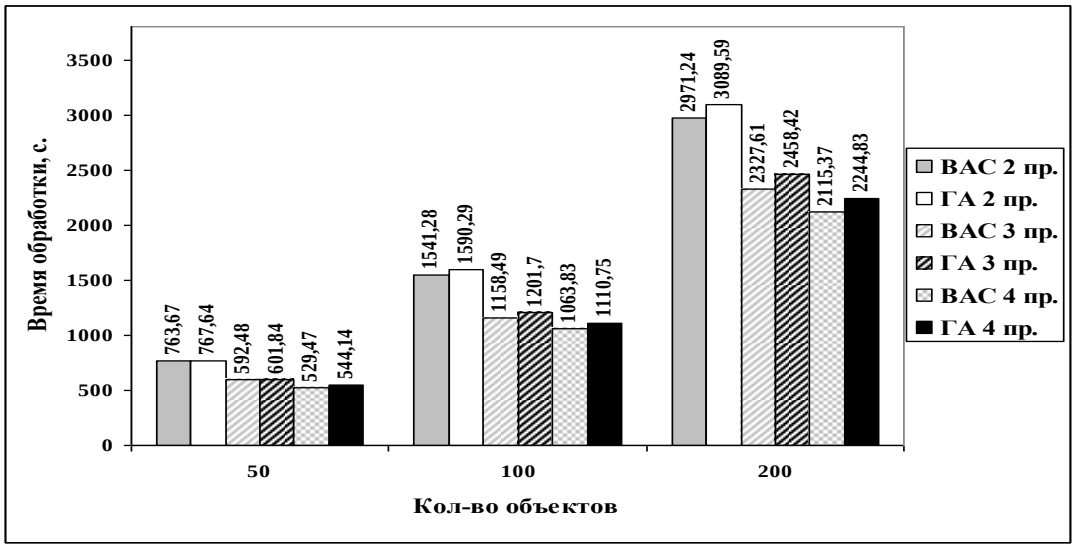


Рис. 9. Улучшение показателей для статической оптимизации

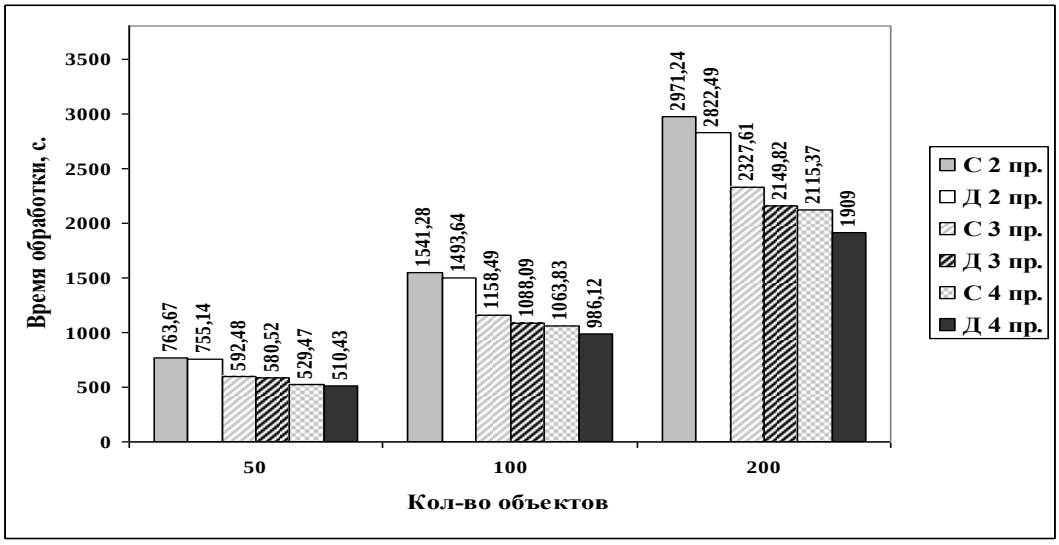


Рис. 10. Улучшение показателей для динамической оптимизации

Результаты первой серии экспериментов свидетельствуют о том, что алгоритм виртуальной сети находит лучшие расписания, при этом производительность алгоритма возрастает с увеличением пространства поиска. Алгоритм виртуальной сети имеет меньшую вычислительную сложность и находит решения быстрее, чем классический ГА.

Результаты второй серии экспериментов показывают, что алгоритм виртуальной сети, управляя системой динамической оптимизации, существенно улучшает показатели производительности в случае обработки стохастических потоков изображений.

Заключение

Реализация параллельных приложений в рамках специализированных компонентных архитектур дает возможность пользователям существенно ускорить процессы создания, анализа и оптимизации программ. Архитектура средств обработки потоков данных, основанная на использовании многоагентных систем, может быть легко адаптирована для многих приложений, предполагающих параллельную обработку. Использование компонентного подхода определяет гибкую структуру параллельного приложения, позволяющую адаптировать вычислительный процесс к условиям функционирования. Средства проектирования легко расширяются новыми операциями, алгоритмами и типами данных для реализации приложении обработки информационных потоков из различных предметных областей.

Список литературы

1. Spence, C. Full-chip Lithography Simulation and Design analysis – how OPC is changing IC Design / C. Spence // Proc. SPIE. – 2005. – Vol. 21, iss. 10. – P. 1-14.
2. Poortinga, Eric R. Comparing software and hardware simulation tools on an embedded-attenuated PSM / Eric R. Poortinga [et al.] [Electronic resource]. – 2007. – Mode of access: http://www.micromagazine.com/archive/00/06/poortinga.html. – Date of access: 12.07.2008.
3. Optolith - 2D Optical Lithography Simulator [Electronic resource]. – 2005. – Mode of access: http://www.silvaco.com/products/vwf/athena/optolith/optolith_datasheet.html. – Date of access: 11.07.2008.
4. Программа Союзного государства "ТРИАДА" [Electronic resource]. – 2006. – Mode of access: http://supercomp.basnet.by/app_triada_ru.html. – Date of access: 12.07.2008.
5. Аваков, С.М. Новые методы и высокопроизводительные алгоритмы детектирования дефектов для модульной платформы автоматического контроля оригиналов топологии СБИС / С.М. Аваков // Инженерный вестник. – 2006. – № 1 (21) / 5. – С. 88-97.
6. Аваков, С.М. Автоматический контроль топологии планарных структур / С.М. Аваков. – Минск: ФУАинформ, 2007. – 168 с.
7. D. Argiro, S. Kubica, M. Young, and S. Jorgensen. Khoros: An integrated development environment for scientific computing and visualization. Whitepaper, Khoral Research, Inc., 1999.
8. M. Zikos, E. Kaldoudi, S. Orphanoudakis. DIPE: A Distributed Environment for Medical Image Processing // Proceedings of MIE'97, Porto Carras, Sithonia, Greece, May 25-29, 1997. - pp. 465-469.
9. M. Guld, B. Wein, D. Keysers, C. Thies et al. A distributed architecture for content-based image retrieval in medical applications // Proceedings of the 2nd International Workshop on Pattern Recognition in Information Systems. – 2002. - pp. 299--314.
10. J. Wickel, P. Alvarado, P. Dörfler et al. Axiom — a modular visual object retrieval system // M. Jarke, J. Koehler, and G. Lakemeyer, editors, KI 2002: Advances in Artificial Intelligence. - LNAI 2479. - Springer, 2002. - p. 253–267.
11. W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface*. - MIT Press. - 1995.
12. Sheppard C.J.R., Torok P. Approximate forms for diffraction integrals in high numerical aperture focusing // Optik. – 1997. - Vol. 105. - No. 2. - P. 77-82
13. Voznesensky N.B., Belozubov A.V. Polarization effects on image quality of optical systems with high numerical apertures. Proc. SPIE, 1999, Vol.3754, p.366-373.
14. Вознесенский Н.Б., Белозубов А.В., Вознесенская Н.Н., Виноградова Г.Н. Описание векторного электромагнитного поля в двойном дипольном приближении. Оптический журнал, Том 69, № 3, март, 2002, стр. 5-10.
15. K. Hwang, Z. Xu. Scalable Parallel Computing – Technology, Architecture, Programming. - McGraw-Hill, USA, 1998.
16. O. Beaumont, A. Legrand, Y. Robert. - Static scheduling strategies for heterogeneous systems. // Computing and Informatics. – Vol. 21. – 2002. – pp. 413-430.
17. Y.-K. Kwok, I. Ahmad. - Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors // ACM Computing Surveys. - Vol.31. - №. 4. – 1999. - pp. 406-471.
18. T. Hagras, J. Janecek. A Fast Compile-Time Task Scheduling Heuristic for Homogeneous Computing Environments // International Journal of Computers and Their Applications. - Vol. 12. - №. 2. – 2005. - p. 76-82.
19. Gerasoulis, T. Yang. A comparison of clustering heuristics for scheduling directed acyclic graphs onto multiprocessors // Journal of Parallel and Distributed Computing. - №4 (16). – 1992. - p. 276-291.
20. M.A. Palis, J.-C. Liou, and D.S.L. Wei. Task Clustering and Scheduling for Distributed Memory Parallel Architectures. // Trans. Parallel and Dist. Systems. - Vol. 7. - №. 1. - Jan. 1996. - pp. 46-55.
21. S. Porto, A. C. Ribeiro. A Tabu Search Approach to Task Scheduling on Heterogeneous Processors under Precedence Constraints // International Journal of High-Speed Computing. - №2 (7).- 1995. - p. 45-71.
22. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы.: Учебное пособие / Под ред. В.М.Курейчика. - М.: Физматлит, 2004.

23. Y. M. Yufik, T. B. Sheridan. Virtual Networks: New framework for operator modeling and interface optimization in complex supervisory control systems // A Rev. Control. - Vol. 20. - p. 179-195.

24. Р. Х. Садыхов, А. В. Отвагин. Алгоритм поиска решений на основе модели виртуальной сети в системах параллельной обработки //Автоматика и вычислительная техника. – №1. – Рига, Латвия. – 2001. – С. 25-33.

25. R. Kh. Sadykhov, A. V. Otwagin. Algorithm for optimization of parallel computation on the basis of genetic algorithms and model of a virtual network // Proceedings of the International Workshop on Discrete-Event System Design DESDes'01, Przytok, Poland, June 27-29, 2001. - p.121-126.

26. Абрамов С.М., Айламазян А.К., Анищенко В.В., Парамонов Н.Н., Танаев В.С., Чиж О.П. Основные принципы создания и применения перспективных моделей семейства суперкомпьютеров "СКИФ" // Вестник связи. - №4. – Минск, 2002. - стр. 52-55