УДК 519.6:629.78

ПОДГОТОВКА, ОБРАБОТКА И ВИЗУАЛИЗАЦИЯ ДАННЫХ ДЛЯ ИЗГОТОВЛЕНИЯ ГОЛОГРАММ НА ЭЛЕКТРОННО-ЛУЧЕВОЙ УСТАНОВКЕ ZBA-21

С. Ю. Сережников[1]

Разработан пакет программ для проектирования и изготовления компьютерно-синтезированных голограмм с помощью технологии электронно-лучевой литографии на установке ZBA-21. Пакет позволяет визуализировать микрорельеф, подготовленный для экспонирования, переводить данные из растрового формата в формат, используемый ZBA, а также формировать полные данные для экспонирования микрорельефа голограммы из отдельных элементов. Пакет программ используется для изготовления оригиналов голограмм в НИВЦ МГУ.

Ключевые слова: визуализация, голограммы, электронно-лучевая литография, растровый формат, векторный формат, микрорельефы, фотолитография.

Настоящая работа посвящена проблемам, возникающим при проектировании и изготовлении компьютерно-синтезированных голограмм. Такие голограммы представляют собой микрорельеф глубиной порядка $0.2-0.5$ мкм и с характерными размерами порядка 0.2 мкм. Для изготовления компьютерно-синтезированных голограмм применяется прецизионное оборудование, поскольку необходимо сформировать заданный рельеф с точностью до десятых долей микрона.

Наиболее часто применяются технологии, используемые в микроэлектронике: фотолитография и электронно-лучевая литография [1, 2]. При изготовлении оригинала голограммы с помощью электронно-лучевой литографии микрорельеф формируется электронным лучом на пластине, покрытой специальным составом, растворимость которого в проявителе меняется под действием электронного луча. Основным инструментом является прецизионный генератор изображений ZBA-21, формирующий рельеф с разрешением 0.2 мкм на пластине площадью до 10×10 см.

Процесс проектирования и изготовления голограммы с помощью этой установки происходит следующим образом.

1. На начальном этапе проектируется общая структура голограммы и определяются типы элементов и способы их расположения на пластинке. Составляются файлы исходных данных.

2. Для каждого элемента, встречающегося на голограмме, подготавливаются полные данные, описывающие его микрорельеф.

3. Далее подготовленные данные собираются воедино и переводятся в формат для ZBA. Этот этап называется сборкой топологического слоя.

4. Происходит экспонирование микрорельефа на пластинке с фоторезистом. Процесс экспонирования является долгим и дорогостоящим.

5. После проявления фоторезиста изготовление оригинала голограммы завершается.

Процесс проектировании и изготовлении оригинала голограммы обладает следующими особенностями:

— большие объемы данных: типичный размер файлов для ZBA, описывающих голограмму площадью 20×20 мм, составляет $500-1000$ Мб;

— длительность и дороговизна процесса экспонирования;

— различные способы задания и типы исходных данных, а также форматов данных для ZBA.

— обработка данных должна быть достаточно эффективной и легко контролируемой.

Перечисленные особенности вызывают необходимость предварительного просмотра и проверки исходных данных, чтобы исключить экспонирование заведомо неверного изображения.

1. Представление данных. Экспонирование микрорельефа на электронно-лучевом генераторе ZBA-21 осуществляется засвечиванием прямоугольных областей (штампов) размером $0.2 \times 0.2 - 6 \times 6$ мкм. Интенсивность экспонируемых штампов (глубина получаемого в результате рельефа) определяется временем экспозиции, которое составляет порядка $10-20$ мкс. Установка имеет главную отклоняющую систему

[1] Институт вычислительной математики РАН, ул. Губкина, 8, 117333, Москва; e-mail: ssj@pc759.cmc.msu.ru

(магнитную) и микроотклоняющую систему (электростатическую). Последняя перемещает луч внутри подполя (TAF) размером 200×200 мкм. Главная отклоняющая система перемещает луч внутри рабочего поля (AF) размером 3×3 мм, что соответствует выбору позиции TAF внутри AF. Перемещения свыше 3 мм осуществляются передвижением стола с пластиной, что соответствует выбору координат AF. Все AF образуют топологический слой (LA). Соответственно организован и формат представления данных: файл с данными содержит координаты AF, TAF и параметры элементарных фигур (EF): длина, ширина, интенсивность. Возможно задание повторения множеств фигур (называемых субподполями (UTAF)) или целых TAF. При этом задается либо список координат $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, в которых должны быть расположены повторяемые элементы, либо элементы располагаются на равномерной сетке (тогда задаются начальные координаты x_0, y_0, шаг повторения dx, dy и из количество nx, ny).

При разработке компьютерно-синтезированных голограмм приходится иметь дело с очень большим количеством фигур (порядка 10^5 фигур/мм^2); при общей площади голограммы порядка 20×50 мм в ней содержатся десятки миллионов фигур. Столь большие объемы данных часто невозможно обработать имеющимся стандартным программным обеспечением ZBA, рассчитанным на ЭВМ СМ-1420 и объем данных порядка 10 – 20 Мб. Дело осложняется следующим: программное обеспечение (ПО) для экспонирования ZBA устроено так, что после подготовки данных требуется их дополнительное преобразование ("компиляция" во внутренний формат, используемый ПО экспонирования); это удваивает и без того немалый (до сотен мегабайт) объем данных.

В работе рассматриваются следующие задачи, возникающие при изготовлении голограмм.

1) Визуализация данных. Необходимо обеспечить возможность предварительного просмотра топологического слоя, чтобы выявить ошибки проектирования (иначе эти ошибки выявятся только после экспонирования и проявки пластины). Кроме того, необходим контроль правильности построения элементов (например, микротекстов), что сложно проделать визуально вследствие их малых размеров.

2) Подготовка данных. Имеющуюся информацию (результаты расчета оптических элементов, параметры дифракционных решеток, изображения микротекста и т.д.) необходимо представить в виде, приемлемом для экспонирования, т.е. в виде элементарных фигур — прямоугольников. Иными словами, должны быть построены изображения всех используемых элементов в терминах элементарных фигур.

3) Сборка топологического слоя. Из файлов с отдельными элементами и описания их расположения необходимо составить описание всего топологического слоя для экспонирования.

Для решения перечисленных задач потребовалась разработка нового программного обеспечения, совместимого с имеющимся и осуществляющего как проверку корректности данных, так и само экспонирование.

2. Визуализация данных. Всего на пластинке могут содержаться десятки миллионов прямоугольников. Таких фигур на площади одного TAF обычно бывает несколько тысяч. Как правило, голограмма состоит из отдельных оптических элементов (дифракционных решеток и т.д.) размером порядка 50 – 100 мкм, среди которых часто встречаются неоднократно повторяющиеся элементы. Исходные данные представляются в виде растрового изображения, в котором каждый пиксель соответствует элементу, а его значение определяет тип элемента, который будет находиться на пластинке в данной позиции. Просматривать такие данные на основе растровых изображений на практике невозможно (например, изображение одного AF размером 3×3 мм занимает около гигабайта и на его построение требуется значительное время). Поэтому необходимо строить изображение просматриваемой области в реальном времени. Для этого нужно определить, какие части файла входят в просматриваемую область.

Исходный файл представляет собой просто поток команд без какой-либо упорядоченности по координатам или информации о границах элементов, например:

AF : 1000, 1000; TA : 0, 0;	координаты рабочего поля и подполя x, y;
UR : 0, 0, 6, 1, 3, 18;	повторяющаяся конструкция x, y, dx, dy, nx, ny;
R 0, 0, 6, 0.4; @	прямоугольник x, y, dx, dy; конец повтора;
R ...	другие прямоугольники ... ;
TW : 100, 100, 100, 200, 200, 500, 200, 600;	повторяющееся подполе x_1, y_1, x_2, y_2 ... ;
...	...

Используется как текстовый, так и двоичный форматы данных (последний полностью аналогичен текстовому по структуре).

Так как файл представляет собой просто поток команд без какой-либо упорядоченности по координатам или информации о границах элементов, необходимо создать для этого специальные структуры данных (рис. 1).

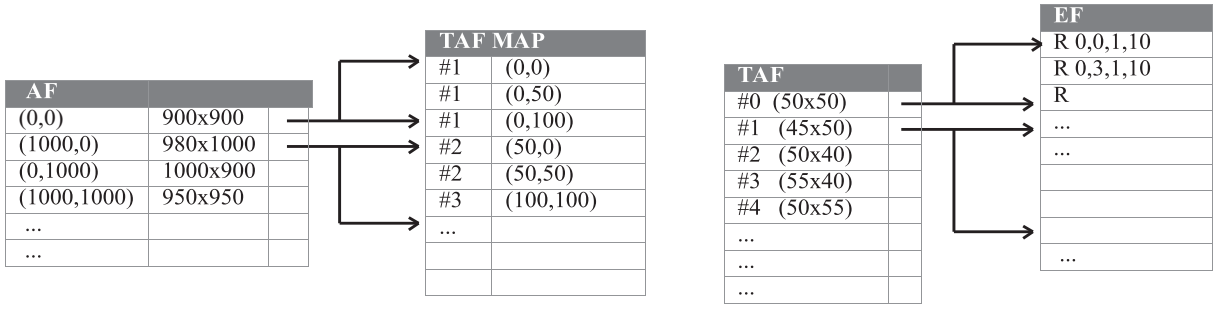

Рис. 1. Структуры данных, используемые для визуализации

В эти структуры заносится дополнительная информация о границах всех элементов. Структура EF содержит данные об отдельных прямоугольниках; для повторяющихся конструкций добавляется информация об их границах. Взяв минимум и максимум граничных значений по всем элементам в пределах одного подполя (TAF), получим ограничивающий прямоугольник для подполя. Эта информация заносится в таблицу TAF; в нее заносятся также адреса начала и конца информации о данном подполе в таблице EF. Координаты подполей и индекс подполя в таблице TAF заносятся в таблицу TAF MAP. При такой организации данных изменение координат, добавление повторяющихся TAF или удаление TAF изменяет только таблицу TAF MAP. Взяв минимум и максимум граничных значений TAF по всем TAF в пределах одного рабочего поля (AF), получим ограничивающий прямоугольник для AF. Поскольку AF немного (десятки, редко сотни), можно обойтись одной таблицей AF, в которую заносятся границы, координаты AF и адреса начала и конца информации о данном AF в таблице TAF MAP (соотношение между таблицами AF и TAF MAP аналогично соотношению между таблицами TAF и EF). После того как для массива данных построены соответствующие структуры (рис. 1), легко производить следующие операции, необходимые для визуализации в реальном времени.

1) Загрузка в оперативную память только данных, описывающих выбранную область. При этом выбираются AF, пересекающиеся с областью, затем в них выбираются TAF, пересекающиеся с областью. Это позволяет достаточно легко работать с большими массивами данных в сотни мегабайт.

2) Отсечение элементов, лежащих вне области видимости при построении изображения. Здесь используется также и информация о границах повторяемых конструкций, хранящаяся в массиве EF. При частом использовании повторяемых конструкций быстрое отсечение лежащих вне области видимости экономит много вычислений.

Кроме этого, такая структура данных позволяет легко вносить некоторые изменения (например, чтобы изменить размеры AF на всем топологическом слое, оставив все фигуры на местах, достаточно пересчитать только таблицы AF и TAF MAP).

При сдвиге области видимости площадь обновляемой области достаточно мала; после отсечения остается мало элементов, требующих отрисовки, что позволяет просматривать загруженную в память область без видимой задержки: она составляет несколько миллисекунд, полное построение типичного изображения 1024×768 (подобного рис. 2, б) требует порядка $50-200$ мс на Pentium-100. Площадь, изображение которой может быть загружено в данный объем памяти, сильно зависит от использования в массиве данных повторяющихся конструкций и может составлять от 0.2 до 20 и более мм^2/Мб (типично порядка 3 мм^2/Мб).

Диапазон размеров различных структур топологического слоя достаточно велик: элементарные фигуры имеют размер порядка 1 мкм, TAF — порядка 100 мкм, AF — порядка 3 мм, весь топологический слой — порядка 30×30 мм. Для просмотра различных структур нужно строить изображение в масштабе от 0.1 до 1000 мкм/пиксель. Поэтому реализовано два режима визуализации: просмотр EF или TAF. В режиме TAF для каждого подполя изображается его ограничивающий прямоугольник, сохраненный в таблице TAF. Для того чтобы сделать видимой внутреннюю структуру топологического слоя, цвета прямоугольников чередуются. На рис. 2, а показан участок голограммы, просматриваемый в режиме TAF. Довольно хорошо заметны детали исходного изображения, по которому был построен топологический слой.

3. Преобразование из растрового формата. Исходные данные для построения голограммы часто представлены в виде растровых файлов (результаты расчета оптических элементов, изображения дифракционных решеток, микротекста и других элементов). Для экспонирования данные должны быть

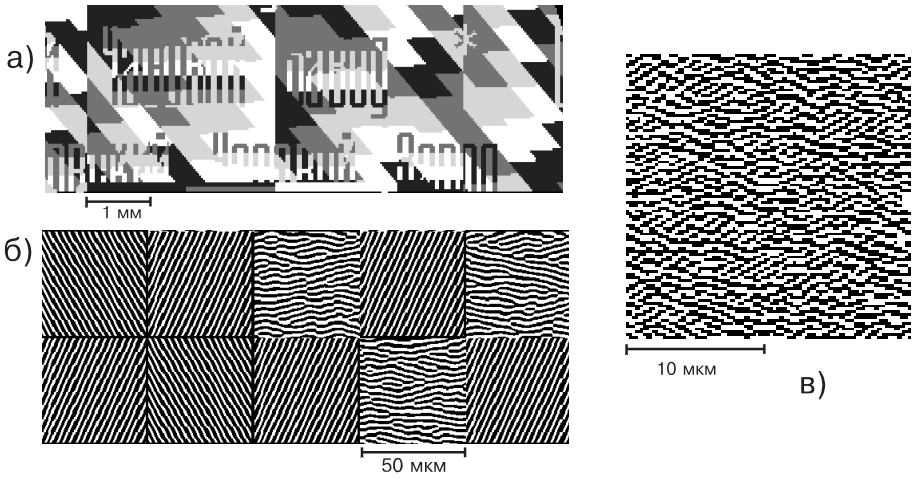

Рис. 2. Пример визуализации: а) просмотр общей структуры топологического слоя, б) просмотр мелких деталей, в) фрагмент микрорельефа оптического элемента

представлены в виде последовательности элементарных фигур — прямоугольников. Представление "один элемент растра $\leftrightarrow$ один прямоугольник" крайне неэффективно (время экспонирования увеличивается в несколько раз). Возникает задача выделения и компактного представления прямоугольников из растрового изображения. Количество прямоугольников должно быть возможно меньшим.

Объем данных на этом этапе подготовки данных может быть достаточно большим: для оптического элемента (рис. 2, в) размером $50 \times 50 - 200 \times 200$ мкм объем данных достигает $0.2 - 2$ Мб, а если задано непосредственное изображение (например микротекст размером 20×20 мм при разрешении 2×2 мкм), то потребуется до 10^8 пикселей. Поэтому необходим эффективный алгоритм, который проводит обработку достаточно быстро и минимизирует количество получающихся фигур.

Раньше преобразование растровой информации во внутреннее представление для ZBA производилось следующим образом: по всему исходному изображению выполнялся поиск целиком заполненных прямоугольников площадью $M \times N$ и их запоминание, а затем обрабатывались все более мелкие детали при уменьшении M и N до 1. Такая процедура занимала много времени (до нескольких часов). Целесообразно строить алгоритм так, чтобы обработка осуществлялась в один проход, т.е. так, чтобы к каждому пикселю изображения обращение производилось один раз (или, в худшем случае, не более чем некоторое фиксированное количество раз, не зависящее от обрабатываемых данных).

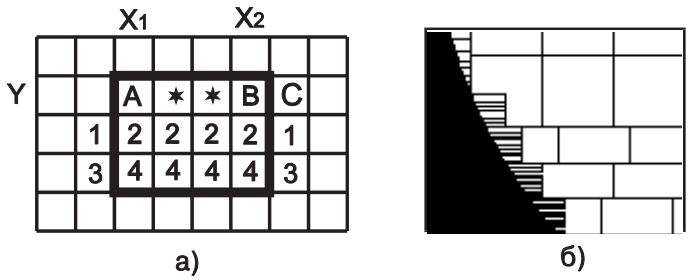

Рис. 3. Выделение прямоугольников из растрового изображения

Предлагается следующий алгоритм. Предполагается, что исходные данные заданы в растровом формате, значение каждого пикселя соответствует интенсивности, с которой его требуется экспонировать (0 — не экспонировать). Граница просматриваемой области (обычно TAF) заполняется служебными значениями FFh, чтобы в процессе обработки не производить лишних сравнений координат. Обработка производится построчно. Это дает анизотропию (повернутое на 90° или 180° изображение даст совершенно другой результат), что в зависимости от исходных данных может быть как недостатком, так и преимуществом. Предлагаемый алгоритм может быть представлен в виде последовательности следующих шагов.

Шаг 1. На строке ищется первый занятый пиксель и отмечается непрерывная область $A * *B$ (с

координатами $(X_1, Y), \ldots, (X_2, Y))$ (рис. 3, а), занятая пикселями с той же интенсивностью, что и $A(X_1, Y)$.

Шаг 2. Проверяются пиксели "1" с координатами $(X_1 - 1, Y + 1)$ и $(X_2 + 1, Y + 1)$; если хотя бы один из них свободен, то выполняется шаг 3.

Шаг 3. Если это возможно (все пиксели "2" $(X_1, Y + 1), \ldots, (X_2, Y + 1)$ заняты), то прямоугольник расширяется вниз и процесс повторяется с шага 2 (проверяются пиксели "3" и т д); в противном случае в результат записывается прямоугольник $(X_1, Y), \ldots, (X_2, Y)$.

После записи прямоугольника (начиная с позиции "C" $(X_2 + 1, Y)$) ищется следующая занятая область, затем обрабатываются строки $Y + 1$, $Y + 2$ и т.д. Поскольку граница рабочей области изображения заполнена значениями FF, то при поиске точки B на шаге 1 и при расширении прямоугольника на шаге 3 не требуется проверки на достижение границы. После обработки области на место границы возвращаются исходные значения, а граница следующей области выделяется значениями FF. Записанный прямоугольник стирается из изображения (заполняется 00) во избежание повторной его обработки.

Такое преобразование осуществляется практически в один проход: к любому байту изображения обращение происходит 1 – 3 раза: чтение при поиске, чтение при проверке, запись при стирании обработанного прямоугольника (старый алгоритм требует порядка $M \cdot N$ обращений). Имеет смысл ограничить максимальный размер формируемых прямоугольников. Без такого ограничения, например, круглая область будет разделена на узкие горизонтальные полосы, что формально оптимально в смысле количества фигур, но практически неприемлемо: ZBA формирует штампы не более 6×6 мкм, а большие прямоугольники все равно впоследствии разделяются, что многократно умножает количество фигур (при этом средние размеры прямоугольников окажутся даже меньше, чем с введенными ограничениями). Если ширина области $A * *B$ равна заданному максимуму M, то проверка пикселей "1" не проводится и прямоугольник всегда расширяется вниз пока возможно.

Таким образом, области, потенциально содержащие большие фигуры, почти никогда не разделяются меньшими фигурами (за исключением случаев, когда достигается максимальный размер), так как отделяемый прямоугольник всегда граничит с пустым пространством из-за проверки на шаге 2. Это обеспечивает хорошую эффективность алгоритма в смысле количества фигур. На рис. 3, б показан пример разбиения на прямоугольники большой круглой области, а на рис. 4 — выделенные старым и новым алгоритмом прямоугольники из одинаковых исходных фигур.

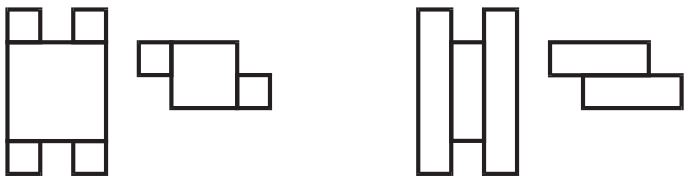

Рис. 4. Примеры выделения прямоугольников старым (а) и новым (б) алгоритмами

Дополнительный выигрыш в скорости дает обработка узких прямоугольников $(X_2 - X_1 \leqslant 8)$ отдельными специализированными подпрограммами. В результате скорость обработки составляет в среднем семь тактов процессора на пиксель и 150 – 160 тактов на каждый прямоугольник, что дает 2 – 4 млн пикселей в секунду вместе с чтением и записью данных на диск (Pentium-100). Например, обработка файла размером 7000×4000 (2.2 миллиона фигур) с получением результата размером 17 Мб требует 10 сек процессорного времени. Скорость достаточна для того, чтобы впоследствии передавать данные непосредственно на экспонирование, минуя громоздкие массивы данных (скорость экспонирования порядка 10 – 20 мкс на каждый прямоугольник).

Из нескольких тысяч прямоугольников, расположенных на площади одного TAF, примерно несколько десятков или сотен прямоугольников имеют одинаковые размеры. Для более компактного представления данных целесообразно сортировать прямоугольники по размерам и объединять одинаковые прямоугольники в повторяющиеся конструкции с указанием их размеров и списка координат. При наличии большого количества небольших прямоугольников маловероятно появление больших, поэтому в целях экономии можно ограничить максимальный размер прямоугольников до 32 (единица длины равна 0.1 мкм). Каждый прямоугольник имеет также интенсивность (всего возможно 8 различных значений). Тогда общее количество различных типов прямоугольников не превосходит $32 \cdot 32 \cdot 8 = 8192$. Таким образом, выделив буфер на несколько десятков тысяч фигур, заполнив его выделяемыми прямоугольниками и связав

в список прямоугольники одинакового типа (каждый прямоугольник будет находиться в одном из 8192 списков, изначально пустых), мы получим возможность по мере накопления достаточного количества фигур формировать конструкции (тип+список координат) и записывать их в результат, освобождая списки фигур соответствующего типа. При этом освобожденные элементы заносятся в список свободных, из которого берутся адреса для размещения новых фигур в буфере. Такая сортировка практически не требует дополнительного времени, однако уменьшает объем получающихся данных почти в два раза.

Предусмотрены режимы обработки изображений как высокого разрешения (оптические элементы (рис. 2, в), разрешение до 0.1 мкм, но изображение содержит не более одного AF), так и низкого разрешения (микротекст и т.п., может быть несколько AF, но полоса (ширина файла × высота TAF) должна помещаться в памяти). Это позволяет обрабатывать весь спектр применяемых данных.

4. Сборка топологического слоя. После подготовки всех элементов необходимо сформировать полную топологию изображения, т.е. скомпоновать описания всех использованных элементов. Обычно возникает следующая задача. Имеется исходный файл, в котором значение каждого пикселя $F[i, j]$ соответствует типу элемента, размещаемого в области голограммы $iw \leqslant x < (i + 1)w$, $jh \leqslant y < (j + 1)h$ размером $w \times h$. Кроме того, имеются файлы с данными для каждого элемента. Обычно это растровые данные — результаты расчета оптических элементов или изображения дифракционных решеток. Оптические элементы, такие как “скрытое изображение” [3] должны быть периодически отложены на всей предназначенной для них площади. Их размер может превышать размер пикселя (для простоты он выбирается кратным размеру пикселя $W_i = m_i \cdot w$; $H_i = n_i \cdot h$ для i-ого элемента). Требуется сформировать полностью представление топологического слоя для экспонирования. Поскольку многие пиксели содержат одинаковые данные, нужно собирать их в повторяющиеся конструкции во избежание чрезмерного увеличения объема данных.

Предварительно растровые изображения элементов представляются в виде элементарных фигур. При этом размер TAF выбирается равным $w \times h$. Элемент оказывается разбитым на $m_i \times n_i$ подполей, и при просмотре пикселей со значением i координатам (x, y) соответствует подполе i-ого элемента ($x \bmod m_i$, $y \bmod n_i$). Таким образом подполя элемента попадают при обработке на свои места, образуя непрерывное изображение (например, фрагменты на рис. 2, б справа — подполя одного оптического элемента). Изображение обрабатывается последовательно для каждого AF, а внутри AF последовательно для каждого типа элемента i, где $m_i > 1$ и $n_i > 1$, а затем для всех элементов с $m_i = n_i = 1$. Аналогично сортировке прямоугольников, описанной в [4, 5], организуется буфер и $m_i \cdot n_i$ списков, а в список с номером (k, l) добавляются координаты (x, y), если $x \bmod m_i = k$ и $y \bmod n_i = l$. По мере накопления достаточно большого количества элементов в списках в результат записываются повторяющиеся конструкции (список координат, данные подполя) и списки освобождаются. Элементы размером в один пиксель собираются вместе и к ним применяется аналогичная процедура: в список с номером i (количество списков здесь равно количеству элементов с $m_i = n_i = 1$) заносятся координаты (x, y), если $F[x, y] = i$. В настоящем варианте программы эта операция организована довольно эффективно и формирование выходного файла происходит со скоростью 3–4 Мб/сек (фактически со скоростью записи на диск).

Описанные выше алгоритмы визуализации и обработки данных при проектировании компьютерно-синтезированных голограмм реализованы в виде пакета программ. Разработанный пакет используется в НИВЦ МГУ для изготовления оригиналов компьютерно-синтезированных голограмм и позволяет решать поставленные задачи достаточно эффективно.

СПИСОК ЛИТЕРАТУРЫ

1. *Глазков И.М., Райхман Я.А.* Генераторы изображений в производстве интегральных микросхем. Минск: Наука и техника, 1981.
2. Электронно-лучевая литография в изготовлении микроэлектронных приборов. М.: Радио и связь, 1984.
3. *Гончарский А.В., Попов В.В., Степанов В.В.* Введение в компьютерную оптику. М.: Изд-во МГУ, 1991.
4. *Van Renesse R.L.* Optical Document Security. London: Artech House, 1997.
5. *Михальчук В.М., Ровдо А.А, Рыжиков С.В.* Микропроцессоры 80x86, Pentium. Архитектура, функционирование, программирование, оптимизация кода. Минск: Битрикс, 1994.

Поступила в редакцию
15.03.2002