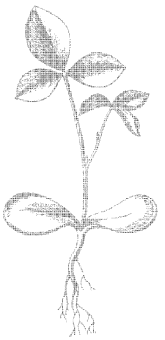
Автоматизированное проектирование СБИС на базовых кристаллах

Издательство «Радио и связь»

Массовая
библиотека
инженера
электроника

Автоматизированное проектирование СБИС на базовых кристаллах



Москва
«Радио и связь» 1988

ББК 32.973.2
А 22
УДК 621.382+627.396.6(07)

Авторы:

А. И. Петренко, В. Н. Лошаков, А. Я. Тетельбаум, Б. Л. Шрамченко

А 22 **Автоматизированное** проектирование СБИС на базовых кристаллах/А. И. Петренко, В. Н. Лошаков, А. Я. Тетельбаум, Б. Л. Шрамченко. — М.: Радио и связь, 1988. — 160 с., ил. — (Массовая б-ка инженера «Электроника»).

ISBN 5-256-00138-8

Рассматриваются вопросы автоматизации конструирования топологии больших (БИС) и сверхбольших (СБИС) интегральных микросхем, изготавливаемых на базовых кристаллах. Описываются основные подходы к решению данной проблемы и проводится подробный анализ иерархического проектирования. Обсуждаются распространенные матричные конструкции базовых кристаллов и методика проектирования СБИС, основанная на их применении. Приводятся характеристики систем автоматизированного проектирования.

Для инженерно-технических работников, занимающихся проектированием радиоэлектронной аппаратуры.

А $\frac{2403000000\text{-}009}{046(01)\text{-}88}$ 128-88 **ББК 32.973.2**

Рецензенты: доктор техн. наук А. Г. Алексенко,
кандидат техн. наук А. Н. Кармазинский

Редакция литературы по электронной технике

Производственное издание

Анатолий Иванович Петренко, Вячеслав Николаевич Лошаков, Александр Яковлевич Тетельбаум, Борис Лазаревич Шрамченко

АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЕ СБИС НА БАЗОВЫХ КРИСТАЛЛАХ

Заведующий редакцией *Ю. Н. Рысев.* Редактор *М. М. Лисина.*
Художественный редактор *Н. С. Шеин.* Обложка художника *А. С. Дзуцева.*
Технический редактор *А. Н. Золотарева.* Корректор *Т. В. Дземидович*
ИБ № 1611

Сдано в набор 15.05.87 Подписано в печать 25.08.87
Т-15698 Формат 60×90/16 Бумага тип. № 1 Гарнитура литературная Печать высокая
Усл. печ. л. 10,0 Усл. кр.-отт. 10,375 Уч.-изд. л. 11,83 Тираж 10 000 экз.
Изд. № 20960 Зак. № 101 Цена 65 к.
Издательство «Радио и связь». 101000 Москва, Почтамт, а/я 693

Московская типография № 5 ВГО «Союзучетиздат». 101000 Москва, ул. Кирова, д. 40

ISBN 5-256-00138-8

ПРЕДИСЛОВИЕ

В настоящее время во многих отраслях промышленности при разработке электронной аппаратуры все шире применяются большие интегральные микросхемы (БИС) и сверхбольшие интегральные микросхемы (СБИС). Это повышает надежность аппаратуры, снижает ее массу, габаритные размеры, потребляемую мощность и улучшает другие технико-экономические характеристики. Выпускаемые электронной промышленностью стандартные БИС в большинстве случаев не могут удовлетворить специальным требованиям авиации, связи, судостроения и других отраслей народного хозяйства к электронной аппаратуре. Это вызывает появление мелкосерийных так называемых «заказных» БИС и СБИС. Высокую стоимость заказных БИС и СБИС, определяемую преимущественно этапами проектирования и разработки, удается снизить за счет систем автоматизированного проектирования (САПР).

Наряду с развитием новых методов проектирования наблюдается тенденция стандартизации и унификации конструкций кристаллов БИС и особенно СБИС, что способствует сокращению сроков разработки и снижению стоимости их производства. Последним достижением в области унификации явились базовые кристаллы с матрицами нескоммутированных элементов (матричные или полузаказные БИС — матричные БИС). Конструкция матричной БИС представляет собой совокупность регулярно расположенных на кристалле полупроводниковых приборов (транзисторов), между которыми имеются зоны для межсоединений. Поэтому различные схемы, реализованные на базовом кристалле, отличаются только слоями коммутации. Топология СБИС на базовом кристалле строится на основе типовых элементов (схем И—НЕ, ИЛИ—НЕ, триггеров, усилителей и др.), топология которых разрабатывается заранее и хранится в системной библиотеке. Все типовые элементы стандартизованы по размерам и могут быть реализованы в произвольной области кристалла.

Применение традиционных методов автоматизации проектирования для матричных СБИС невозможно ввиду чрезмерных затрат машинного времени и оперативной памяти. В связи с этим возникают две проблемы. Во-первых, требуется получить решение в условиях, когда невозможно размещение полной информации о

проектируемом объекте в оперативной памяти. Во-вторых, необходимо решить всю задачу за приемлемый промежуток времени. Решение обоих проблем может быть получено в рамках иерархического подхода.

Учитывая последние достижения в области проектирования матричных БИС представляется целесообразным изложение основных вопросов, начиная от постановки задачи и заканчивая рассмотрением конкретных САПР БИС. Именно с этой целью и написана настоящая книга. Помимо известных результатов советских и зарубежных ученых книга содержит оригинальный материал, накопленный авторами в ходе создания средств автоматизированного проектирования матричных БИС. При этом различные этапы проектирования рассмотрены с разной степенью детализации, что объясняется, с одной стороны, относительно малой специализацией таких этапов, как схематическое и логическое моделирование, а с другой — достаточным уровнем рассмотрения ряда вопросов [1, 2].

Предисловие и гл. 4 написаны А. И. Петренко, гл. 1, 2, 3, 5— А. Я. Тетельбаумом и Б. Л. Шрамченко, гл. 6 — В. Н. Лошаковым.

1. КОНСТРУИРОВАНИЕ БОЛЬШИХ И СВЕРХБОЛЬШИХ ИНТЕГРАЛЬНЫХ МИКРОСХЕМ НА БАЗОВЫХ МАТРИЧНЫХ КРИСТАЛЛАХ

1.1. УНИФИКАЦИЯ КОНСТРУКЦИИ КРИСТАЛЛА

Характерной тенденцией развития элементной базы современной электронно-вычислительной аппаратуры является быстрый рост степени интеграции. В этих условиях актуальной становится проблема ускорения темпов разработки узлов аппаратуры, представляющих собой БИС и СБИС [1, 2]. При решении данной проблемы важно учитывать существование двух различных классов интегральных схем: стандартных (или крупносерийных) и заказных. К первым относятся схемы, объем производства которых достигает миллионов штук в год. Поэтому относительно большие затраты на их проектирование и конструирование оправдываются. Этот класс схем включает микропроцессоры, различного вида полупроводниковые устройства памяти (ПЗУ, ОЗУ, ЗУПВ), серии стандартных микросхем и др. Схемы, принадлежащие ко второму классу, при объеме производства до нескольких десятков тысяч в год, выпускаются для удовлетворения нужд отдельных отраслей промышленности. Значительная часть стоимости таких схем определяется затратами на их проектирование [3, 4].

Основным средством снижения стоимости проектирования и, главное, ускорения темпов разработки новых видов микроэлектронной аппаратуры являются системы автоматизированного проектирования (САПР). В результате совместных действий конструкторов, направленных на уменьшение сроков и снижение стоимости проектирования БИС и СБИС, появились так называемые полузаказные интегральные микросхемы, в которых топология в значительной степени определяется унифицированной конструкцией кристалла. Первые схемы, которые можно отнести к данному классу, появились в 60-х годах. Они изготавливались на унифицированном кристалле с фиксированным расположением функциональных элементов [5, 6]. При этом проектирование заключалось в назначении функциональных элементов схемы на места расположения соответствующих функциональных элементов кристалла и проведении соединений. Такой кристалл получил название базового, поскольку все фотошаблоны (исключая слои коммутации) для его изготовления являются постоянными и не зависят от реализуемой схемы. Эти кристаллы, однако, нашли ограниченное применение из-за неэффективного использования площади

кристалла, вызванного фиксированным положением функциональных элементов на кристалле.

Для частичной унификации топологии интегральных микросхем (ИС) использовалось также проектирование схем на основе набора *типовых ячеек* [7, 8]. В данном случае унификация состояла в разработке топологии набора функциональных (типовых) ячеек, имеющих стандартизованные параметры (в частности, равные размеры по вертикали). Процесс проектирования при этом заключался в размещении в виде горизонтальных линеек типовых ячеек, соответствующих функциональным элементам схемы, в размещении линеек на кристалле и реализации связей, соединяющих элементы, в промежутках между линейками. Ширина таких промежутков, называемых *каналами*, определяется в процессе трассировки. Отметим, что хотя в данном случае имеет место унификация топологии, кристалл не является базовым, поскольку вид всех фотошаблонов определяется в ходе проектирования [9].

Современные полузаказные схемы реализуются на *базовом матричном кристалле* (БМК), содержащем не соединенные между собой простейшие элементы (например, транзисторы), а не функциональные элементы как в рассмотренном выше базовом кристалле. Указанные элементы располагаются на кристалле матричным способом (в узлах прямоугольной решетки). Поэтому такие схемы часто называют матричными БИС. Как и в схемах на типовых ячейках топология набора логических элементов разрабатывается заранее. Однако в данном случае топология логического элемента создается на основе регулярно расположенных простейших элементов. Поэтому в ходе проектирования логический элемент может быть размещен в любом месте кристалла, а для создания всей схемы требуется изготовить только фотошаблоны слоев коммутации. Основные достоинства БМК, заключающиеся в снижении стоимости и времени проектирования, обусловлены [8, 10—13]: применением БМК для проектирования и изготовления широкого класса БИС; уменьшением числа детализированных решений в ходе проектирования БИС; упрощением контроля и внесения изменений в топологию; возможностью эффективного использования автоматизированных методов конструирования, которая обусловлена однородной структурой БМК.

Наряду с отмеченными достоинствами БИС на БМК не обладают предельными для данного уровня технологии параметрами и, как правило, уступают как заказным, так и стандартным схемам. При этом следует различать технологические параметры интегральных микросхем и функциональных узлов (устройств), реализованных на этих микросхемах. Хотя технологические параметры стандартных микросхем малой и средней степени интеграции наиболее высоки, параметры устройств, реализованных на их основе, оказываются относительно низкими. В качестве примера, показывающего возможности различных способов реализации устройства, в табл. 1.1 приведены данные для специализированного контроллера.

Таблица 1.1. *Данные проектирования контроллера* [11]

Показатель	Стандартные малые и средние микросхемы (МИС и СИС)	БИС на базовых матричных кристаллах	Заказные БИС
Число кристаллов	50	1 (100 логических вентилей)	1 (850 логических вентилей)
Процент использования логических вентилей, %	95	85	100
Общая площадь, см^2	323	13	13
Стоимость проектирования, тыс. дол.	10—20	20—30	50—100
Стоимость сборки, дол.	20—50	4—10	4—12
Время проектирования, недель	10—16	10—16	36—50
Общая стоимость одного контроллера, дол.:			
250	180	200	—
2500	100	90	120
25 000	80	50	60
250 000	65	30	30
$2{,}5 \cdot 10^6$	—	25	20

Целесообразность использования того или иного способа реализации устройства обычно определяется экономическими соображениями. Типичный вид зависимости стоимости Z производства устройства от объема N производства в год для различных способов реализации показан на рис. 1.1. Как видно из рисунка, использование БМК целесообразно при объеме производства от 5 до 40 тыс. штук. Результаты сравнения экономических показате-

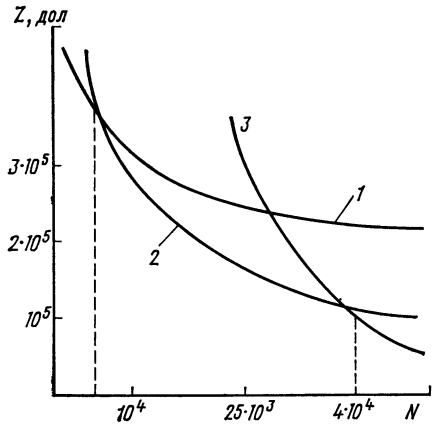


Рис. 1.1. Зависимости удельной стоимости производства от его объема:
1 — устройство на стандартных микросхемах; *2* — устройство на матричных БИС; *3* — устройство на заказных БИС

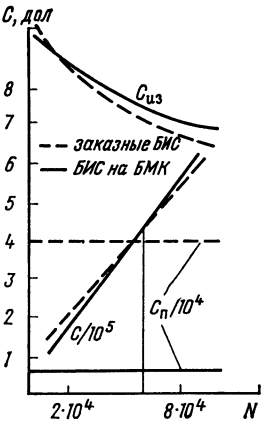


Рис. 1.2. Зависимость составляющих стоимости производства от их объема

Таблица 1.2. *Характеристики БМК на ЭСЛ-транзисторах*

Страна, фирма	Тип	Число внутренних ячеек	Число периферийных ячеек	Мощность вентиля, мВт	Время задержки, мс
США, FAIRCHILD	GE0020	20	20	30	0,3
	GE0200	200	48	20	0,75
	GE1000	1000	88	2—4	0,5—1,5
	GE1750	1750+64б*	2—4	2—4	0,5—1,5
	GE2000	2000	126	2—4	0,5—1,5
	GE4000	4000	160	2—4	0,5—1,5
США, MOTOROLA	MCA600	600	44	3,3	1,2
	MCA1200	1200	58	3,3	1,2
	MCA2500	2500	148	2	0,25
ФРГ, SIEMENS	SH100B/10	700	58	8—50	0,5
	SH100B/1	960	58	8—50	0,5
	SH100B/2	700+128б*	58	8—50	0,5
Англия, PLESSEV	SCD1XXX	75	24	10	0,55
	SCD2XXXH	300	56	10	0,55
	SCD2XXXM	300	56	3	1,5
	SCD2XXL	300	56	2,5	2
США, SIGNETICS	ACE600	600	638	64	0,5
	ACE900	878	64	3	0,5
	ACE1400	1414	144	3	0,5
	ACE1320M	100+320б*	144	3	0,5
	ACE2200	2204	144	3	0,5

* Встроенная память, содержащая указанное число бит.

Таблица 1.3. *Характеристики БМК на МДП-транзисторах*

Страна, фирма	Тип	Число внутренних ячеек	Число периферийных ячеек	Мощность вентиля, мВт	Время задержки, мс
США, TEXAS INSTRUMENTS	TAL002	240	30	1,25	5
	TAL004	500	42	1,25	5
	TAL006	540	76	0,6	2,5
	TAL008	1008	104	0,35	2,5
	TAL010	1280	88	0,35	1
	TAL020	2420	120		
США, FAIRCHILD	GT0500	650	68	1,5	2
	GT0750	1000	56	1,5	2
	GT1500	2000	140	1,5	2
Япония, FUJITSU	B200	206	26	2	6
	B500	512	60	2,3	1,8
	B2000	2018	112	0,65	0,95
Англия, FERRANTI	ULA5RA	500	38	0,3	2,5
	ULA9RA	900	48	0,3	2,5
	ULA12RA	1200	54	0,3	2,5
	ULA16RA	1600	62	0,3	2,5
	ULA18RA	1800	64	0,3	2,5
	ULA20RA	2000	72	0,3	2,5
	ULA24RA	2400	80	0,3	2,5
	ULA40RA	4000	118	0,3	2,5
	ULA100RA	10000	136	0,3	2,5

лей заказных БИС и БИС на БМК представлены на рис. 1.2, где $C_п$ — стоимость проектирования; $C_{из}$ — стоимость изготовления одной БИС; C — общая стоимость производства ($C=C_п+NC_{из}$; N — объем производства).

Учитывая, что БМК обеспечивают наиболее высокую экономическую эффективность в широком диапазоне средних объемов производства, БИС на БМК в настоящее время являются перспективной основой для разработки разнообразной электронной аппаратуры. Как видно из табл. 1.2 и 1.3, число элементов на выпускаемых кристаллах может достигать нескольких десятков тысяч. Учитывая непрерывность роста степени интеграции, ясна актуальность задачи проектирования сверхбольших интегральных схем (СБИС), содержащих 10^5 элементов на БМК и более [4, 14].

В заключение остановимся на опыте применения матричных БИС [10, 15—18]. Одно из первых применений матричные микросхемы нашли в ЭВМ серии V/6 фирмы Amdahl Corp. (США). Быстродействие центрального процессора ЭВМ 470 V/6 оказалось в 3 раза выше, чем самой производительной ЭВМ серии IBM 370. В свою очередь фирма IBM в процессорах 4331 и 4341 также использовала БИС на БМК, что повысило плотность монтажа в 15 раз. Перевод ЭВМ ЕС-1060 на матричные БИС (ЕС-1087) позволил повысить плотность компоновки в 6 раз, а быстродействие — в 3 раза. Таким образом, матричные БИС в настоящее время широко используются при разработке практически всех высокопроизводительных и мини-ЭВМ [4].

1.2. ОСНОВНЫЕ ТИПЫ БМК

Базовый кристалл представляет собой прямоугольную многослойную пластину фиксированных размеров, на которой выделяют *периферийную* и *внутреннюю области* (рис. 1.3). В периферийной области располагаются *внешние контактные площадки* (ВКП) для осуществления внешнего подсоединения и *периферийные ячейки* для реализации буферных схем (рис. 1.4). Каждая внешняя ячейка связана с одной ВКП и включает диодно-транзисторную структуру, позволяющую реализовать различные буфер-

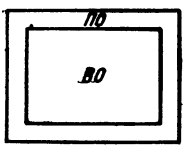


Рис. 1.3. Основные области базового матричного кристалла (ПО — периферийная, ВО — внутренняя области)

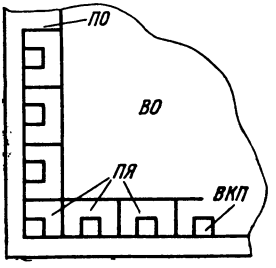


Рис. 1.4. Структура периферийной области (ПЯ — периферийная ячейка)

ные схемы за счет соответствующего соединения элементов этой структуры. В общем случае в периферийной области могут находиться ячейки различных типов. Причем периферийные ячейки могут располагаться на БМК в различных ориентациях (полученных поворотом на угол, кратный 90°, и зеркальным отражением). Под базовой ориентацией ячейки понимают положение ячейки, расположенной на нижней стороне кристалла.

Во внутренней области кристалла матричным способом располагаются *макроячейки* для реализации элементов проектируемых схем (рис. 1.5). Промежутки между макроячейками используются для электрических соединений. При матричном расположении макроячеек область для трассировки естественным образом разбивается на горизонтальные и вертикальные каналы. В свою очередь в пределах макроячейки матричным способом располагаются *внутренние ячейки* для реализации логических элементов. Различные способы расположения внутренних ячеек и макроячейках показаны на рис. 1.6. Причем наряду с размещением ячеек «встык» (рис. 1.6,*а*—*в*) применяется размещение с зазорами, в которых могут проводиться трассы электрических соединений (рис. 1.6,*г*).

Особенностью ячейки является специальное расположение выводов, согласованное со структурой макроячейки. А именно, ячейки размещаются таким образом, чтобы выводы ячеек оказались на периферии макроячейки. Так, в макроячейке (рис. 1.6,*б*) выводы каждой ячейки дублируются на верхней и нижней ее сторонах. При этом имеется возможность подключения к любому выводу с двух сторон ячейки, что создает благоприятные условия для трассировки. Последнее особенно важно при проектировании СБИС.

В макроячейке, показанной на рис. 1.6,*в*. выводы ячейки располагаются только на одной стороне, т. е. выводы ячеек верхнего ряда находятся на верхней стороне макроячейки, а нижнего — на нижней. Применение таких макроячеек позволяет сократить требуемую площадь кристалла, но приводит к ухудшению

Рис. 1.6. Примеры структур макроячеек (ВЯ — внутренние ячейки)

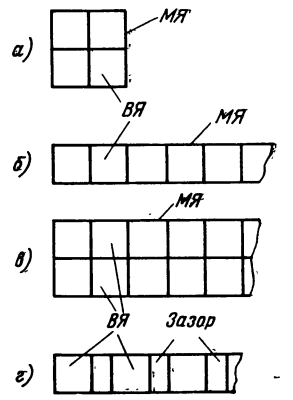
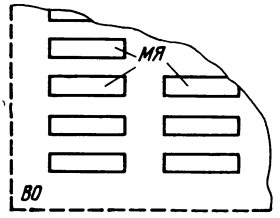


Рис. 1.5. Структура внутренней области (МЯ — макроячейка)



условий для трассировки. Поэтому данный тип макроячеек используется лишь при степени интеграции, не превышающей 100—200 вентилей на кристалл. Отметим, что в некоторых типах БМК, кроме однотипных макроячеек, во внутренней области могут присутствовать специализированные макроячейки, реализующие типовые функциональные узлы (например, запоминающее устройство).

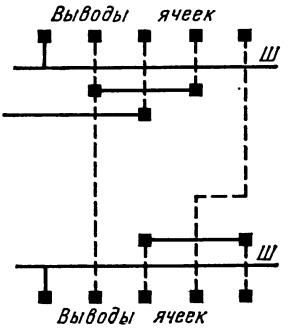


Рис. 1.7. Реализация схемы в канале без фиксированного слоя коммутации (Ш — шины питания)

Помимо ячеек, являющихся заготовками для реализации элементов, на БМК могут присутствовать *фиксированные части соединений.* К ним относятся шины (рис. 1.7) питания, земли, синхронизации и заготовки для реализации частей сигнальных соединений. Например, для макроячеек (рис. 1.6,*б*) шины питания и земли проводятся вдоль верхней и нижней сторон соответственно. Для макроячеек (рис. 1.6,*а*, *в*) шины проводятся вдоль линии, разделяющей верхний и нижний ряды ячеек, что приводит к уменьшению потерь площади кристалла. Для реализации сигнальных соединений на БМК получили распространение два вида заготовок: фиксированное расположение однонаправленных (горизонтальных или вертикальных) *участков трасс* в одном слое; фиксированное расположение участков трасс в одном слое и *контактных окон,* обеспечивающих выход фиксированных трасс во второй слой.

В первом случае для реализации коммутации проектируемой схемы не требуется разработка фотошаблона фиксированного слоя, т. е. число разрабатываемых фотошаблонов уменьшается на единицу. Во втором случае число разрабатываемых фотошаблонов уменьшается на два (не требуется также фотошаблон контактных окон). Различные способы реализации одной и той же схемы показаны на рис. 1.7—1.9, где свободная коммутация в

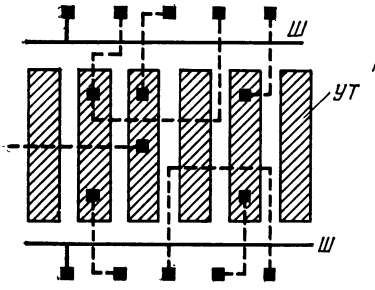


Рис. 1.8. Реализация схемы с фиксированным слоем коммутации (УТ — участки трасс)

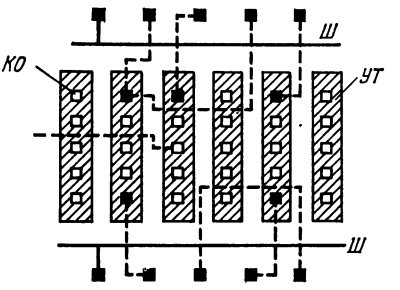


Рис. 1.9. Реализация схемы с фиксированными контактными окнами и соединениями (КО — контактные окна)

двух слоях представлена на рис. 1.7, коммутация с одним фиксированным слоем — на рис. 1.8, а коммутация с фиксированными контактными окнами — на рис. 1.9. Отметим, что в настоящее время получили распространение различные виды формы и расположения фиксированных трасс и контактных окон. Целесообразность использования того или иного вида определяется типом макроячеек, степенью интеграции кристалла и объемом производства [10].

При реализации соединений на БМК часто возникает необходимость проведения трассы через область, занятую макроячейкой. Такую трассу будем называть *транзитной*. Для обеспечения такой возможности допускается: проведение соединения через область, занятую ячейкой, проведение через зазоры (рис. 1.6,*г*) между ячейками. Первый способ может применяться, если в ячейке не реализуется элемент, или реализация элемента допускает использование фиксированных трасс и неподключенных выводов для проведения транзитной трассы.

Рассмотрим несколько примеров конструкций, применяемых в БМК. На рис. 1.10 показана ячейка базового кристалла на КМДП-транзисторах, содержащая по два последовательно соединенных *p*- и *n*-канальных транзистора (*VT1*, *VT2* и *VT3*, *VT4*). Каждая пара транзисторов образует вентиль. Для реализации элемента на такой ячейке используется один переменный фотошаблон, т. е. коммутация осуществляется в одном слое. Пример реализации схемы (рис. 1.11,*а*), выполняющей функцию $z=\overline{x \vee y}$, на приведенной выше ячейке показан на рис. 1.11,*б*. Соединения, подходящие к выводам *1*, *2*, *3*, не имеют отношения к логической схеме и предназначены для реализации транзитных трасс.

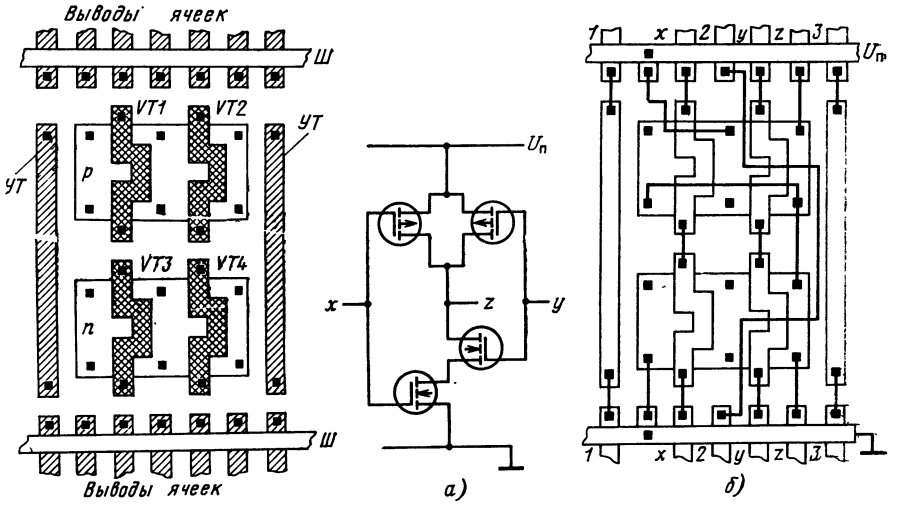


Рис. 1.10. Ячейка на КМДП-транзисторах

Рис. 1.11. Реализация логической схемы ИЛИ — НЕ на ячейке

В качестве примера конструкции БМК приведем кристаллы Л-700, И-200, И-3, поскольку в дальнейшем описываются системы автоматизированного проектирования БИС на этих кристаллах.

Общий вид кристалла Л-700 показан на рис. 1.12,*а*. Макроячейки данного кристалла представляют собой горизонтальные линейки. Поэтому иногда такой кристалл называют *линейчатым*. Ячейки для реализации логических элементов размещают вплотную друг к другу. Число макроячеек на кристалле равно 10. Каждая макроячейка содержит 80 ячеек, а ячейка — шесть транзисторов (образующих три вентиля), внешние контакты ячейки дублируются в верхнем и нижнем рядах и располагаются в обоих слоях коммутации. Пропускная способность канала для трассировки соединений между макроячейками составляет 11 проводников в горизонтальном направлении. Через каждую ячейку, на которой не реализован логический элемент, может быть проведено до трех проводников в вертикальном направлении. Фрагмент топологии БМК данного типа показан на рис. 1.13.

Другой тип макроячейки используется в кристалле И-200, общий вид которого приведен на рис. 1.12,*б*. В БМК И-200 макроячейки, содержащие по 4 ячейки, располагаются на кристалле «в матрицу» размером 6×6. Фрагмент конструкции этого типа кристалла показан на рис. 1.14. Контактные площадки макроячейки расположены на ее периферии и реализованы в одном слое (вертикальном). На рис. 1.14 они пронумерованы числами от 1 до 24. Свободная область кристалла, используемая для реализации соединений между макроячейками, состоит из *горизонтальных* и *вертикальных каналов*. Пропускная способность вертикального канала составляет 8 проводников, горизонтального 1. Кроме того, возможно проведение горизонтальных соединений через область, занятую макроячейкой, когда этому не препятствуют трассы, подходящие к контактам макроячейки. Препятствие возникает, если трасса подходит к контакту из горизонтального слоя.

Кристалл И-3 принадлежит к линейчатому типу. Однако он отличается от рассмотренного выше кристалла Л-700. Этот кристалл содержит 108 внутренних ячеек, которые характеризуются относительно высоким уровнем специализации. Каждая ячейка содержит переключатель (*П*) тока, элемент ИЛИ и шесть транзисторов (*VT*). Условное изображение такой ячейки показано на рис. 1.15,*а*. Особенность ее состоит также в том, что внешние контакты расположены в одном слое (горизонтальном) и находятся не только на периферии, но и во

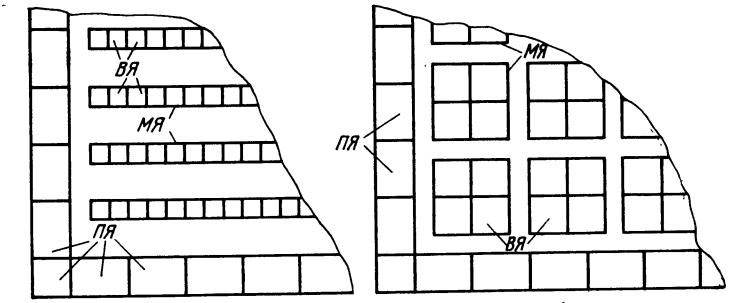


Рис. 1.12. Общий вид базового матричного кристалла

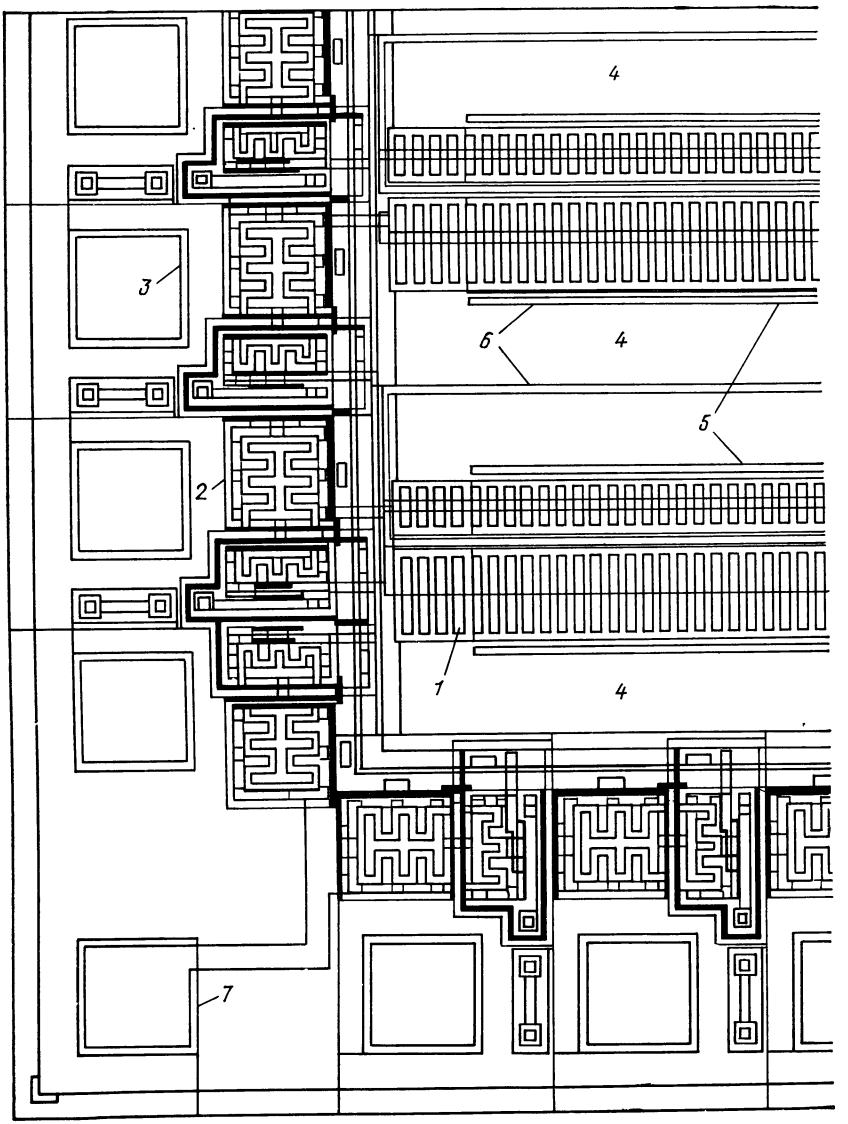


Рис. 1.13. Фрагмент топологии линейчатого базового матричного кристалла:
1 — транзисторы для реализации логических элементов в линейках; *2* — транзисторно-диодные структуры для реализации периферийных элементов; *3* — внешние контактные площадки; *4* — горизонтальные каналы; *5* — линейки ячеек; *6* — шины питания; *7* — угловые контакты для подвода стандартных уровней напряжений

внутренней области. Контакты во внутренней области используются для подключения фиксированных шин. Относительное расположение контактов ячейки показано на рис. 1.15,*б*. Пропускная способность горизонтального канала кристалла составляет 15 соединений. Вертикальные соединения можно про-

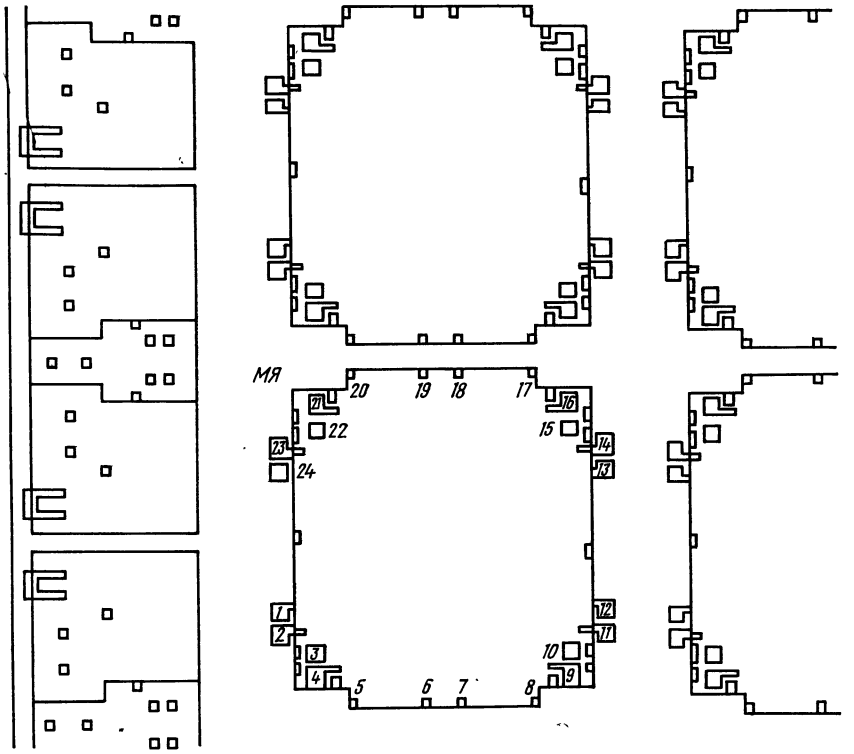


Рис. 1.14. Фрагмент конструкции БМК И-200

водить через область, занятую ячейкой (даже если на ней реализован элемент), когда этому не препятствует подключение контактов ячейки.

Таким образом, в настоящее время разработано большое многообразие типов БМК, которые имеют различные параметры. При проектировании микросхем на БМК необходимо учитывать конструктивно-технологические характеристики кристалла. К ним относятся геометрические параметры кристалла, форма и расположение макроячеек на кристалле и ячеек внутри макроячеек, рас-

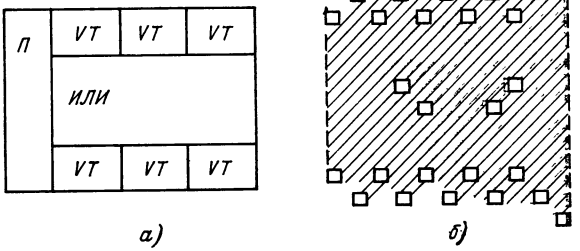


Рис. 1.15. Ячейка БМК И-3 *а)* *б)*

положение шин и способ коммутации сигнальных соединений [19, 20].

В заключение параграфа следует сказать, что задача определения структуры БМК является достаточно сложной, и в настоящее время она может решаться конструктором как вручную, так и с помощью средств автоматизации [21—25, 181]. В этом же параграфе мы познакомили читателя с некоторыми наиболее распространенными типами БМК для лучшего понимания задачи проектирования электронного устройства (или отдельного узла) на выбранном БМК. Именно последняя задача является предметом детального рассмотрения данной книги.

1.3. РЕАЛИЗАЦИЯ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ НА БМК

Выше было показано, что БМК представляет собой заготовку, на которой определенным образом размещены электронные приборы (транзисторы и др.). Следовательно, проектирование микросхемы можно было бы вести и на приборном уровне. Однако этот способ не находит распространения на практике по следующим причинам. Во-первых, возникает задача большой размерности. Во-вторых, учитывая повторяемость структуры частей кристалла и логической схемы, приходится многократно решать однотипные задачи. Поэтому применение БМК предполагает использование библиотеки *типовых логических элементов*, которая разрабатывается одновременно с конструкцией БМК. В этом отношении проектирование матричных БИС подобно проектированию печатных плат на базе типовых серий микросхем.

Таким образом, при применении БМК проектируемая схема описывается на уровне логических элементов, а каждый элемент содержится в библиотеке. Эта библиотека формируется заранее. Она должна обладать функциональной полнотой для реализации широкого спектра схем. Традиционно подобные библиотеки содержат следующие элементы: И—НЕ, ИЛИ—НЕ, триггер, входные, выходные усилители и др. Для реализации элемента используется одна или несколько ячеек кристалла, т. е. размеры элемента всегда кратны размерам ячейки. Топология элемента разрабатывается на основе конструкции ячейки и представляет собой совокупность трасс, которые совместно с имеющимися на кристалле постоянными частями реализуют требуемую функцию. Именно описание указанных соединений и хранится в библиотеке.

В качестве примера реализации логического элемента на базе БМК можно привести рис. 1.11,*б*, если схему на рис. 1.11,*а* рассматривать как схему логического элемента. Особенность данного элемента состоит в том, что он содержит три пары выводов (называемых в дальнейшем *проходами*), предназначенных для проведения транзитных трасс. Причем выводы *1* и *3* соединены с помощью фиксированных трасс, а выводы *2* соединяются трассой произвольной формы.

Рис. 1.16. Примеры форм логических элементов

В зависимости от того, на каких ячейках реализуются элементы, можно выделить *внешние* (согласующие усилители, буферные схемы и др.) и *внутренние,* или просто *логические* элементы. Если внешние элементы имеют форму прямоугольников независимо от типа кристалла, то для логических элементов существует большое разнообразие форм, которое определяется типом макроячеек. Так, для макроячейки, показанной на рис. 1.6,*а*, возможные формы элементов приведены на рис. 1.16. При этом следует иметь в виду, что каждая форма может быть реализована с поворотом относительно центра макроячейки на угол, кратный 90°. Для расширения возможностей наилучшего использования площади кристалла для каждого логического элемента разрабатываются варианты топологии, позволяющие его реализовать в различных частях макроячейки. Поскольку структура макроячейки обладает симметрией, то эти варианты топологии, как правило, могут быть получены из базового вращением относительно осей симметрии.

При проектировании на уровне элементов существенными данными являются форма логического элемента и расположение его выводов (*цоколевка*). Общий вид расположения элементов на кристаллах двух распространенных типов показан на рис. 1.17. Ясно, что для расстановки элементов на кристалле нет необходимости учитывать внутреннюю топологию элемента. Достаточно знать его размеры и положение выводов. Насколько объем информации, требуемой при конструировании, меньше полной информации об элементах, можно судить по мажоритарному элементу с управляемым входом, реализующему функцию $Y = (X1\ X2 + X1\ X3 + X2\ X3)\ C + X3\ C$. Для этого элемента принципиальная схема показана на рис. 1.18,*а*, его условное обозначение на логической схеме, а внешний контур топологии с цоколевкой —

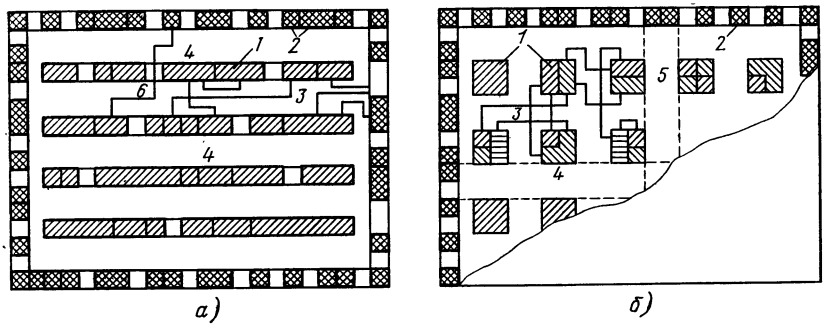


Рис. 1.17. Расположение элементов на БМК:

1 — логический элемент; *2* — периферийный элемент; *3* — межэлементная трасса; *4* — горизонтальный канал для реализации трасс; *5* — вертикальный канал; *6* — транзитная трасса

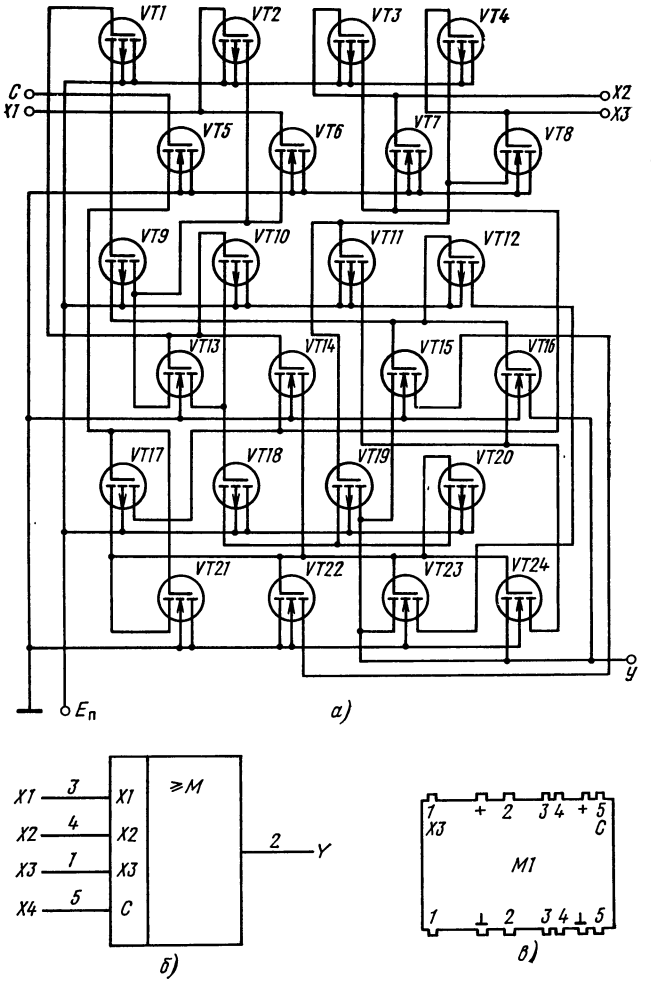


Рис. 1.18. Мажоритарный элемент

соответственно на рис. 1.18,*б* и *в*. Причем следует отметить, что это не самый крупный элемент (24 транзистора, 5 ячеек), поскольку встречаются элементы, содержащие в несколько раз больше транзисторов.

Заранее разработанная топология элемента описывается в системе координат, начало которой совпадает с некоторой точкой элемента, т. е. в относительной системе координат. Поэтому размещение элемента в любой области кристалла сводится к пересчету координат библиотечного описания топологии элемента. Таким образом, при заданной библиотеке элементов в процессе проектирования микросхемы на БМК нет необходимости знать топологию каждого элемента. Однако формирование такой би-

блиотеки требует решения специальных задач разработки и контроля топологии элементов [26—28].

Задача разработки топологии элемента обладает рядом особенностей: часть топологии, определяемая структурой ячейки, задана; размерность задачи относительно невелика; высокие требования к плотности монтажа ввиду многократного использования элемента; жесткая связь с технологией, так как проектирование ведется на приборном уровне; необходимость обеспечения заданных электрических характеристик.

Особенность контроля заключается в возможности использования методов моделирования на ЭВМ (схемотехнического и логического) для моделирования схемы логического элемента с учетом его топологической реализации, натурных испытаний и определения внешних характеристик элемента, которые учитываются разработчиком при его применении [29, 30].

В настоящее время для разработки топологии элемента, как правило, используются интерактивные методы [6, 31, 32], когда основные проектные решения принимает человек, а вычислительные средства используются как инструменты обработки информации, моделирования и контроля. Тем не менее для отдельных видов технологии имеются попытки применения и автоматических методов [8, 33]. Однако пока они не вышли за пределы лабораторных исследований. В дальнейшем процесс проектирования рассматривается в предположении, что библиотека элементов задана.

2. СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ МАТРИЧНЫХ БИС

2.1. ПОСТАНОВКА ЗАДАЧИ ПРОЕКТИРОВАНИЯ

Задача конструирования матричных БИС состоит в переходе от заданной логической схемы к ее физической реализации на основе БМК. При этом исходные данные представляют собой описание логической схемы на уровне библиотечных логических элементов, требования к его функционированию, описание конструкции БМК и библиотечных элементов, а также технологические ограничения. Требуется получить конструкторскую документацию для изготовления работоспособной матричной БИС. Важной характеристикой любой электронной аппаратуры является плотность монтажа. При проектировании матричных БИС плотность монтажа определяется исходными данными. При этом возможна ситуация, когда искомый вариант реализации не существует. Тогда выбирается одна из двух альтернатив: либо матричная БИС проектируется на БМК больших размеров, либо часть схемы переносится на другой кристалл, т. е. уменьшается объем проекти-

руемой схемы. В дальнейшем будем предполагать, что решение задачи проектирования существует.

Основным требованием к проекту является 100%-ная реализация соединений схемы, а традиционным критерием, оценивающим проект, — суммарная длина соединений. Именно этот показатель связан с такими эксплуатационными параметрами, как надежность, помехоустойчивость, быстродействие [34, 35]. В целом задачи конструирования матричных БИС и печатных плат родственны, что определяется заранее заданной формой элементов и высоким уровнем унификации конструкций. Вместе с тем имеют место следующие отличия: элементы матричных БИС имеют более сложную форму (не прямоугольную); наличие нескольких вариантов реализации одного и того же типа элемента; позиции для размещения элементов группируются в макроячейки; элементы могут содержать проходы для транзитных трасс; равномерное распределение внешних элементов по всей периферии кристалла; ячейка БМК, не занятая элементом, может использоваться для реализации соединений; число элементов матричных БИС значительно превышает значение соответствующего параметра печатных плат.

Перечисленные отличия не позволяют непосредственно использовать САПР печатных плат для проектирования матричных БИС. Поэтому в настоящее время разрабатываются новые САПР, предназначенные для проектирования матричных БИС, а также дорабатываются и модернизируются уже действующие САПР печатных плат для решения новых задач [36]. Реализация последнего способа особенно упрощается, когда в системе имеется набор программ для решения задач теории графов, возникающих при конструировании [37, 38].

Поскольку трассировка соединений на БМК ведется с заданным шагом на *дискретном рабочем поле* (ДРП), то необходимо, чтобы выводы элементов попадали в клетки ДРП. Однако внешние выводы макроячеек могут располагаться с шагом, не кратным шагу ДРП. В этом случае используется простой прием введения *фиктивных контактных площадок*, связанных с внутренними частями ячейки [39]. Например, на рис. 2.1 показан фрагмент периферии макроячейки БМК И-200. Внутренние части ячейки реализуются на сетке с клеткой размером 4×4 мкм, а вне макроячейки размеры клетки 8×11 мкм.

Для связи фиктивных контактных площадок с выводами ячейки используются фиксированные трассы, которые на рисунке представлены зачерненными областями. При этом трассировка ведется на ДРП с клеткой размером 16×22 мкм. Если трасса к макроячейке не подходит, то область фиктивной площадки остается свободной.

При разработке САПР БИС на БМК необходимо учитывать требования к системам, диктуемые спецификой решаемой задачи. К ним относятся [1, 40—42]:

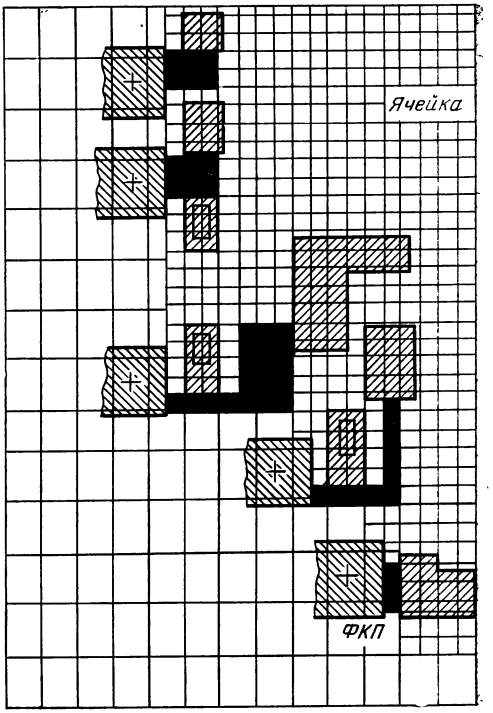


Рис. 2.1. Фрагмент периферии макроячейки базового матричного кристалла И-200 (ФКП — фиктивные контактные площадки)

1. Реализация сквозного цикла проектирования от схемы до комплектов машинных документов на изготовление, контроль и эксплуатацию матричных БИС.

2. Наличие архива данных о разработках, хранимого на долговременных машинных носителях информации.

3. Широкое применение интерактивных режимов на всех этапах проектирования.

4. Сбеспечение работы САПР в режиме коллективного пользования.

Учитывая большую размерность задачи проектирования, большинство существующих САПР матричных БИС реализовано на высокопроизводительных ЭВМ, таких как МВК «Эльбрус», ЕС-1066, СДС-7600, Cyber-203, Hitac-190 [43, 44]. Следует, однако, отметить, что в последнее время ряд зарубежных фирм применяет и мини-ЭВМ [4].

2.2. ОСНОВНЫЕ ЭТАПЫ ПРОЕКТИРОВАНИЯ

Процесс проектирования матричных БИС традиционно делится на следующие укрупненные этапы:

1. Моделирование функционирования объекта проектирования.
2. Разработка топологии.
3. Контроль результатов проектирования и доработка.
4. Выпуск конструкторской документации.

Кратко поясним необходимость каждого этапа и рассмотрим решаемые на них задачи. Поскольку матричная БИС является ненастраиваемым и не ремонтоспособным объектом, то необходимо еще на этапе проектирования обеспечить его правильное функционирование. Достижение этой цели возможно двумя способами [40]: созданием макета матричных БИС на основе дискретных элементов и его испытанием и математическим моделированием. Первый способ связан с большими временными и стоимостными затратами. Поэтому макет используется тогда, когда он специаль-

Таблица 2.1. *Характеристики САПР для проектирования устройств на матричных БИС*

Период разработки, гг.	Максимальный объем моделируемых схем, вентил.	Скорость моделирования, событий/с	Интеграция БМК, вентил.	Цикл проектирования одной матричной БИС, мес.	Число матричных БИС, проектируемых в месяц
1979—1984	$75 \cdot 10^4$	$12 \cdot 10^3$ (фирма Hitachi)	$(1—2) 10^3$	1,5 (фирма Fairchild)	4—12
1984—1990	$(10—15) 10^5$	10^5	$(1—5) 10^4$	0,5	10—30

но не разрабатывается, а уже существует (например, при переходе от реализации устройств на печатных платах к матричным БИС). Второй способ требует создания эффективной системы моделирования схем большого размера, так как при моделировании необходимо учитывать схемное окружение матричных БИС, которое по числу элементов во много раз больше самой схемы. Характеристики существующих и разрабатываемых САПР устройств на матричных БИС приведены в табл. 2.1 [4, 45].

Целесообразность применения систем моделирования можно проиллюстрировать разработкой МВК «Эльбрус-2». Проверка логической корректности разработки выполнялась с помощью интерактивной системы СЛЭМ поэлементного моделирования [46]. В то же время система СЛЭМ была жестко ориентирована на элементную базу серий И-100 и И-700 и не могла быть использована для элементной базы, включающей СБИС. Данные, полученные в ходе разработки, представлены в табл. 2.2. Более совершенной явилась система ПУЛЬС, позволяющая оперативно модернизировать элементную базу, описывать логические схемы различного уровня иерархии, независимо работать в интерактивном и пакетном режимах большому числу пользователей. Система ПУЛЬС послужила основой для создания комплексной САПР КОМПАС-82 [40] и позволила подключить к ней ряд подсистем (в том числе для проектирования топологии матричных БИС на БМК И-200, И-200М, И-3М, для генерации тестов, анализа временных соотношений и др.). В процессе проектирования моделирование выполняется на различных стадиях. Подробно задача моделирования рассматривается в гл. 4.

Таблица 2.2. *Данные о работе системы СЛЭМ*

Число сеансов в месяц:	
общее	423—660
пакетных	293—512
интерактивных	111—174
Затраченное время в месяц, ч:	
процессорное	40—97
астрономическое	326—509
Число моделируемых устройств	2—16
Максимальный объем моделируемого устройства, микросхем	$12{,}5 \cdot 10^3$ (100 тыс. вентилей)
Число исправлений в месяц	До 300

Этап разработки топологии связан с решением следуюших задач: размещение элементов на БМК, трассировка соединений, корректировка топологии. Иногда в качестве предварительного шага размещения решается специальная задача компоновки (распределения элементов по макроячейкам). В этом случае возможны различные методы решения задачи размещения. Первый метод состоит в том, чтобы после компоновки размещать группы элементов, соответствующих макроячейкам, а затем размещать элементы внутри каждой макроячейки. При этом критерий оптимальности компоновки включает составляющие, определяемые плотностью заполнения макроячеек и связностью элементов макроячейки. Достоинствами этого метода являются сокращение размерности задачи размещения и сведение исходной задачи к традиционным задачам компоновки и размещения. Возможность применения традиционных методов компоновки предопределяется тем, что условие существования реализации группы элементов в макроячейке для получивших распространение БМК легко выражается через суммарную площадь элементов и отношение совместимости пар элементов. Например, для макроячейки (рис. 1.16) суммарная площадь элементов должна быть не больше площади макроячейки и элементы формы (рис. 1.16,*б*), повернутые друг относительно друга на 90°, не должны попадать в одну макроячейку. Отметим, что так как расположение элементов внутри макроячеек существенно влияет на условия трассировки соединений между макроячейками, рассмотренный метод решения задачи размещения для некоторых типов БМК может давать сравнительно низкие результаты. Последний факт экспериментально был установлен для кристалла типа И-200 [40].

Другой метод размещения состоит в распределении элементов по макроячейкам с учетом координат макроячеек. В этом случае в ходе компоновки определяются координаты элементов с точностью до размеров макроячеек [4] и появляется возможность учета положения транзитных трасс. Более подробно данный вид размещения обсуждается в гл. 5, 6. Для матричных схем небольшой степени интеграции (до 1000 элементов на кристалле) применяются модификации традиционных алгоритмов размещения и трассировки. Возможности этих алгоритмов рассматриваются в следующих параграфах. Для СБИС на БМК необходима разработка специальных методов, примеры которых представлены в гл. 3, 5, 6.

Задача корректировки топологии возникает в связи с тем, что существующие алгоритмы размещения и трассировки могут не найти полную реализацию объекта проектирования на БМК. Возможна ситуация, когда алгоритм не находит размещение всех элементов на кристалле, хотя суммарная площадь элементов меньше площади ячеек на кристалле. Это положение может быть обусловлено как сложностью формы элементов, так и необходимостью выделения ячеек для реализации транзитных трасс. Задача определения минимального числа макроячеек для размеще-

ния элементов сложной формы представляет собой известную *задачу покрытия*. Действительно, поставим макроячейке в соответствие множество $M=\{M_1, M_2, ..., M_k\}$ всевозможных максимальных заполнений ее элементами.

Пусть M_i содержит K_{1i} элементов формы f_{1i}, K_{2i}—f_{2i}, ..., $K_{p(i),i}$—$f_{p(i),i}$, где $p(i)$ — число различных форм элементов в M_i. Построим множество M', содержащее подмножество M_i, $i=\overline{1, k}$, причем M_i входит в M' $r(i)$ раз, где $r(i)=\max\{K_{1i}, K_{2i}, ..., K_{p(i),i}\}$. Тогда определение минимального числа макроячеек для размещения элементов сводится к задаче поиска минимального покрытия множества элементов схемы подмножествами из M'. Как известно [47], эта задача относится к классу *NP полных проблем*, что отражает ее объективную сложность. При рассмотрении конкретного типа БМК задача, как правило, упрощается. Например, для макроячейки (рис. 1.6,*а*) минимальное число $K_{\min}$ макроячеек определяется выражением

$$K_{\min}=K_1+K_2+\lceil K_3/2\rceil+\lceil K_4/2\rceil+\frac{1}{8}(K+|K|),$$

где K_1 — число элементов в схеме, занимающих полную макроячейку; K_2 — число элементов, занимающих 0,75 ее площади; K_3 — число элементов, занимающих вертикальную половину; K_4— число элементов, занимающих горизонтальную половину; $K=K_5-K_2-2[(K_3)_{\mathrm{mod}2}+(K_4)_{\mathrm{mod}2}]$; K_5 — число элементов, площадь которых составляет 0,25 площади макроячейки.

Величина $K_{\min}$ не только является нижней границей числа макроячеек БМК, но и позволяет оценить условия, в которых выполняется размещение. Чем ближе $K_{\min}$ к числу макроячеек БМК, тем труднее учитывать критерии оптимальности размещения (определяющим фактором оказывается плотность размещения).

Возможность отсутствия полной трассировки обусловлена эвристическим характером применяемых алгоритмов. Кроме того, в отличие от печатных плат навесные проводники в матричных БИС запрещены. Поэтому САПР матричных БИС обязательно включает средства корректировки топологии. Наибольшее распространение в отечественных САПР получила система «Кулон» (15 УТ—2—017), позволяющая выводить фрагменты топологии на экран графического дисплея и в интерактивном режиме корректировать ее. При корректировке желательно иметь возможность выполнения следующих операций [40, 48, 49]: выделение линий соединяемых фрагментов; изменение положения элементов и трасс с контролем вносимых изменений; автоматическая трассировка указанных соединений; контроль соответствия результатов трассировки исходной схеме. Примером реализации перечисленных операций является подсистема корректировки в САПР КОМПАС-82. Перечисленные операции показывают, что функции, выполняемые в интерактивном режиме постепенно расширяются. Уже сейчас актуальной является задача перепроектирования любого фрагмен-

та топологии. Для матричных БИС таким фрагментом может быть канал для трассировки, или макроячейка, в которой варьируется размещение элементов и др. Решение последней задачи, помимо реализации функций проектирования с заданными граничными условиями (определяемыми окружением фрагмента), требует разработки аппарата формирования подсхемы, соответствующей выделенному фрагменту.

На этапе контроля проверяется адекватность полученного проекта исходным данным [50—52]. С этой целью прежде всего контролируется соответствие топологии исходной принципиальной (логической) схеме. Необходимость данного вида контроля обусловлена корректировкой топологии, выполненной разработчиком, поскольку этот процесс может сопровождаться внесением ошибок. В настоящее время известны два способа решения рассматриваемой задачи. Первый сводится к восстановлению схемы по топологии и дальнейшему сравнению ее с исходной. Эта задача близка к проверке изоморфизма графов. Однако на практике для ее решения может быть получен приемлемый по трудоемкости алгоритм ввиду существования фиксированного соответствия между некоторыми элементами сравниваемых объектов [53, 54]. Дополнительная сложность данной задачи связана с тем, что в процессе проектирования происходит распределение инвариантных объектов (например, логически эквивалентных выводов элементов). Поэтому для логически тождественных схем могут не существовать одинаковые описания и, следовательно, требуются специальные модели, отображающие инвариантные элементы [6]. В общем случае универсальные модели для представления инвариантных элементов не известны, что и явилось одной из причин развития второго способа, согласно которому проводится повторное логическое моделирование восстановленной схемы (этот способ контроля применяется, например, в системе MATIS [55]).

Функционирование спроектированной схемы может отличаться от требуемого не только из-за ошибок, внесенных конструктором, но и в результате образования паразитных элементов. Поэтому для более полной оценки работоспособности матричных БИС при восстановлении схемы по топологии желательно вычислять значения параметров паразитных емкостей и сопротивлений и учитывать их при моделировании на логическом и схемотехническом уровнях.

Существуют причины, по которым перечисленные методы контроля не позволяют гарантировать работоспособность матричных БИС. К ним относятся, например, несовершенства моделей и методов моделирования. Поэтому контроль с помощью моделирования дополняется контролем опытного образца. Для этого на этапе проектирования с помощью специальных программ осуществляется генерация тестов для проверки готовых БИС [55]. Отметим, что при проектировании матричных БИС проведение трудоемкого геометрического контроля не требуется, так как трассировка ве-

дется на ДРП, а топология элементов контролируется при их разработке.

Заключительным этапом проектирования матричных БИС является выпуск конструкторской документации, которая содержит информацию (на соответствующих носителях) для управления технологическими станками-автоматами и сопроводительные чертежи и таблицы, состав и содержание которых регламентируются ГОСТами, а оформление — требованиями ЕСКД. Для автоматизированного выпуска графической и текстовой документации обычно разрабатывается входной язык, который позволяет [31]: компактно и наглядно описывать отдельные фрагменты документа; размещать отдельные фрагменты на площади документа; извлекать требуемую информацию из архива и включать ее во фрагменты документов; распечатывать требуемый документ.

Наличие в системе входного языка является необходимым условием перестройки на новые типы БМК и позволяет при необходимости изменять или создавать новую форму документов. Например, входной язык системы КОМПАС-82 позволил разработать комплект конструкторской документации для БИС на БМК И-200М. Особенность разработки конструкторской документации для матричных БИС состоит в меньшей трудоемкости по сравнению с другими БИС, поскольку выпуск чертежей и таблиц для фиксированных слоев БМК делается один раз на стадии проектирования БМК.

2.3. РАЗМЕЩЕНИЕ ЭЛЕМЕНТОВ НА БМК

Применяемые в настоящее время *алгоритмы размещения* элементов на БМК рассмотрим на примерах ряда эксплуатируемых систем. Поскольку в основе этих алгоритмов лежат известные методы размещения элементов на печатных платах и кристаллах, нашедшие достаточно полное отражение в литературе [7, 32, 55], будем останавливаться только на особенностях, обусловленных применением БМК.

Рассмотрение начнем с системы КОМПАС-82, так как в ней использован наиболее широкий набор алгоритмов [4]. При проектировании БМК И-200 выполняется размещение элементов, которые занимают одну или более ячеек (рис. 1.16). В процессе размещения минимизируются суммарная длина F_1 соединений и максимальное число F_2 соединений в сечении канала. Общая стратегия состоит в получении начального размещения и в последующем его улучшении с помощью ряда итерационных алгоритмов, применяемых в различных порядках. Начальное размещение выполняется в два этапа. На первом — последовательным алгоритмом по критерию F_1 элементы размещаются на неограниченном поле макроячеек. Второй этап заключается в размещении элементов на БМК с учетом расположения элементов, полученного на первом этапе. При этом центр размещения элементов на неограниченном поле располагается в центре кристалла.

Итерационные алгоритмы чередуются до тех пор, пока величина F_2 не достигнет приемлемого значения. В системе используются следующие алгоритмы: алгоритм Штейнберга для макроячеек; силовой алгоритм для макроячеек; перестановка макроячеек в сканирующем по БМК окне; алгоритм Штейнберга для элементов; силовой алгоритм для элементов; перестановки разногабаритных элементов в сканирующем окне.

Применение алгоритмов к макроячейкам подразумевает изменение положения групп элементов, соответствующих макроячейкам. Последний из перечисленных алгоритмов является оригинальным. Его суть состоит в выборе для перестановки таких групп элементов, которые могут образовать одинаковые геометрические фигуры. При этом рассматриваются всевозможные перестановки выбранных групп в окне и фиксируется лучшая из них по критерию F_2. Экспериментальные исследования показали, что данный алгоритм позволяет улучшить решение тогда, когда другим алгоритмам сделать это невозможно.

Дополнительное улучшение условий для трассировки достигается за счет рационального назначения *инвариантных компонентов*. К этому этапу относятся распределение цепей между логически эквивалентными выводами элемента (например, между входами элемента ИЛИ) и группами выводов (соответствующих логически эквивалентным элементам). Данная задача эквивалентна размещению выводов и групп выводов на заданном множестве позиций. На этом же этапе выполняется назначение периферийных элементов на периферийные ячейки.

Поскольку при реализации описанного подхода использование крупных элементов приводит к снижению плотности заполнения кристалла и некоторому ухудшению условий для трассировки, при проектировании БМК И-3 исследовалась возможность применения элементов, занимающих часть ячейки (рис. 1.15).

На БМК И-3 с учетом 64 периферийных элементов может размещаться до 928 элементов. Такой объем схемы не позволяет применять традиционные алгоритмы из-за недопустимо большого времени решения. Поэтому в САПР КОМПАС-82 размещение выполняется группами элементов, выделенными по функциональному признаку. В первую размещаемую группу входят все элементы ИЛИ. Для размещения этой группы строится модель в виде графа $G=(X, E)$, в котором вершины соответствуют элементам, а ребра строятся по следующему правилу: вершина x_i соединяется с x_j, если в схеме существует путь от выхода элемента i к входу элемента j, не проходящий через другие элементы ИЛИ. Для кристалла И-3 $|X| \leqslant 108$. Элементы первой группы размещаются в ячейках кристалла традиционными методами с использованием графа G. После размещения элементов ИЛИ размещается вторая группа, в которую входят все периферийные элементы. Эти и все последующие группы размещаются с учетом ранее размещенных элементов. Для размещения периферийных элементов используются алгоритмы Штейнберга и перестановок в сканирующем окне.

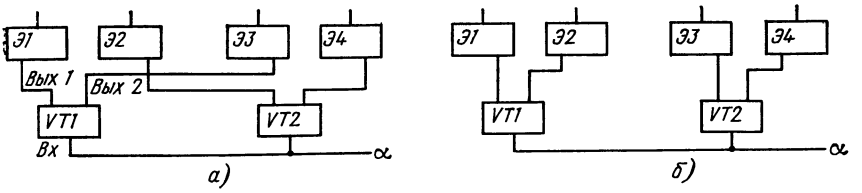


Рис. 2.2. Фрагмент схемы (*а*) и результат оптимального назначения выводов (*б*)

В третью группу входят переключатели тока, которые размещаются теми же алгоритмами. Последнюю, четвертую группу составляют транзисторы. В данном типе матричных БИС каждый транзистор имеет один вход (*Вх*) и два эквипотенциальных выхода (*Вых 1*, *Вых 2*), которые на рис. 2.2,*а* отмечены для транзистора *VT1*. Если входы некоторой группы транзисторов объединены одной цепью α, то выходы транзисторов этой группы оказываются логически эквивалентными. Поэтому до размещения транзисторов решается задача оптимального назначения эквивалентных выходов транзисторов на входы уже размещенных элементов. В результате каждому транзистору ставится в соответствие пара элементов, а оптимальным назначением считается то, для которого сумма расстояний между элементами каждой пары минимальна. Пример решения данной задачи показан на рис. 2.2, где исходная принципиальная схема с учетом положения размещенных элементов представлена на рис. 2.2,*а*, а схема после оптимального назначения — на рис. 2.2,*б*. Так как для решения данной задачи эффективные методы не известны, а размерность ее невелика (число транзисторов в группе не больше 5), задача решается методом, основанным на переборе. При числе транзисторов в группе A, равном n, необходимо исследовать $C^2{}_{2n}/2=C^{n-1}{}_{2n-1}$ вариантов распределения размещенных элементов, связанных с транзисторами группы А, на два равных по мощности подмножества. При фиксированном распределении оптимальный вариант соединений определяется решением *задачи о назначениях* размерности n. Таким образом, временна́я сложность алгоритма равна $O(n^3C^{n-1}{}_{2n-1})$.

Размещение транзисторов выполняется за три шага. На первом шаге осуществляется компоновка транзисторов по ячейкам БМК. При этом учитывается, что каждая ячейка может содержать не более 6 транзисторов. Цена назначения транзистора i в ячейку j определяется расстоянием от ячейки j до элементов, на которые нагружен транзистор i. Данная задача может рассматриваться как *транспортная* [56], в которой имеется N источников с единичной интенсивностью (транзисторы, $N\leqslant 648$) и 108 стоков (ячейки) с потреблением 6. Так как максимальное удаление транзистора от нагрузки ограничено, для каждого транзистора допустимы не более 10 ячеек. Для остальных ячеек цена назначения полагается бесконечно большой. Поэтому в матрице затрат $N\times 108$ не более,

чем 10 N конечных элементов и для ее хранения и обработки можно использовать методы работы с разреженными матрицами. Практические примеры показали, что решение данной задачи на ЭВМ БЭСМ-6 требует не более 2 мин.

На втором шаге осуществляется размещение транзисторов внутри каждой ячейки. Эта задача решается полным перебором, поскольку число вариантов размещения транзисторов в ячейке $A^k_6 = 6!/k!$, где k — число транзисторов в ячейке. При этом для оценки длин соединения используются штейнеровские деревья, близкие к оптимальным.

На завершающем, третьем шаге для каждого транзистора определяется оптимальное назначение его выходов, а для каждого элемента ИЛИ — его входов. Как и на предыдущем шаге, для решения задачи может быть использован полный перебор с тем же критерием.

Особенностью системы ИСКРА [36] является использование библиотеки модулей типовых алгоритмических решений. В этой системе для размещения элементов на БМК решаются задачи компоновки макроячеек, размещения макроячеек на кристалле и выбора ориентации элементов. Алгоритм решения каждой задачи конструируется из модулей типовых решений. Так, алгоритм компоновки базируется на модуле разрезания графа, алгоритм размещения — на модуле построения оптимального гамильтова маршрута и разрезания графа, алгоритм выборов ориентации — на модуле определения центра тяжести полного графа со взвешенными ребрами.

В системе LAMBDA [57, 58] элементы размещаются в два этапа. На первом с помощью последовательного алгоритма (SORG) формируется начальный вариант размещения; на втором — выполняется итерационное улучшение размещения (алгоритм GFDR). Критерием оптимальности размещения является суммарная длина соединений. Особенность реализации алгоритма состоит в том, что при анализе различных положений размещаемого элемента исследуются не все свободные позиции, а только те, которые попадают в ε-окрестность «идеальной» позиции. На этапе итерационного улучшения на каждом шаге осуществляется перестановка λ ($\lambda \geqslant 2$) элементов, причем каждый перемещаемый элемент устанавливается в ε-окрестности своей «идеальной» позиции. Экспериментально установлено, что с точки зрения получения наилучших результатов при небольших затратах машинного времени наиболее целесообразно использовать параметры $\varepsilon=4$, $\lambda=4$. При этом достигаются результаты, значительно превышающие результаты классического алгоритма (FDPR) парных перестановок ($\lambda=2$), который по результатам сравнительного анализа [59] обладает наилучшими характеристиками. Реализация алгоритма позволяет размещать до 1000 элементов, связанных не более чем 2000 цепями.

В последнее время получили развитие методы, в максимальной степени учитывающие специфику конструкции кристалла. Один из них нашел применение в системе AULIS [60], предназначенной для проектирования матричных БИС с макроячейками, которые показаны на рис. 1.16. Процесс размещения представляет собой последовательность решений отдельных задач. Сначала путем дихотомического деления схемы формируются группы элементов, каж-

дая из которых должна быть реализована в пределах одной макроячейки [61, 62]. При этом допускается реализация группы и в свободной периферийной ячейке.

Назначение группы в макроячейки выполняется также с помощью дихотомического деления, а размещение элементов внутри макроячеек — алгоритмом перебора. При этом оптимизируется суммарная длина соединений, которая оценивается по штейнеровским деревьям цепей. Заключительным этапом размещения является распределение инвариантных выводов. Реализация алгоритма дает возможность разрабатывать БМК, содержащие порядка 100 макроячеек.

Матричная структура базового кристалла в определенной степени учитывается и в алгоритмах, согласно которым сначала элементы распределяются по горизонтальным рядам, а затем — в каждом ряду [63, 64]. Интерес представляет итерационный алгоритм улучшения распределения элементов по рядам (столбцам). Идея состоит в том, что последовательно рассматриваются все пары соседних рядов. Для каждой пары выполняется сначала объединение элементов, а затем разбиение на две группы. Экспериментальные данные показывают, что скорость сходимости и окончательный результат практически не зависят от начального распределения. Кроме того, этот алгоритм приводит к результатам, значительно лучшим, чем результаты алгоритма минимальных сечений [4].

С целью максимального использования ресурса конструкций кристалла для реализации трасс распределение элементов по рядам и столбцам можно выполнять так, чтобы сохранялось определенное соотношение между вертикальной и горизонтальной проекциями цепи. Такой подход наиболее целесообразен при использовании БМК с линейчатой структурой, когда наиболее дефицитными оказываются горизонтальные магистрали. Реализация алгоритма парных перестановок, учитывающего названные требования, позволяет на ЭВМ VAX 11/780 проектировать кристаллы с числом вентилей до 3500 [63].

Один из способов увеличения плотности монтажа на базовом кристалле состоит в размещении элементов с учетом трассируемости соединений [65]. При этом относительное положение элементов определяется с помощью силовой модели, а в процессе определения точного размещения проверяется отсутствие циклов в графе конфликтов и загрузка каналов трассами [66].

2.4. ТРАССИРОВКА СОЕДИНЕНИЙ БМК

При трассировке соединений БМК так же, как и при трассировке печатных плат требуется реализовать в автоматическом режиме максимально возможное число соединений. В то же время результаты автоматической трассировки должны оставлять возможность для трассировки в интерактивном режиме нереализованных соединений. Поэтому на рабочем поле трассировки должны сохраняться ресурсы для дотрассировки. В современных системах это достигается за счет построения трасс *в классе простейших конфигураций* (например, с ограниченным числом поворотов и имеющих длину, близкую к минимальной [4, 40, 67]). Для максимизации числа автоматически построенных трасс необходимо предотвращать блокировку выводов элементов, к которым еще не

подведены соединения. Экспериментально установлено, что вероятность блокировки уменьшается при использовании *преимущественных направлений трасс* в слоях (в одном слое проводятся, в основном, горизонтальные отрезки, в другом — вертикальные) и построении трасс в порядке увеличения сложности [68].

Большинство современных алгоритмов трассировки БМК является модификацией известных методов трассировки печатных плат, учитывающей перечисленные требования и канальную структуру области трассировки. Так, в системе КОМПАС-82 в качестве базового используется волновой алгоритм. Для выбора порядка трассировки, уменьшающего вероятность блокировки, электрические цепи предварительно разбиваются на парные соединения применением алгоритма Прима [7]. Для оценки сложности соединений ДРП кристалла разбивается на прямоугольные области, вертикальные границы которых принадлежат границам кристалла, а горизонтальные — совпадают с горизонтальными осями симметрии горизонтальных каналов или рядов макроячеек (рис. 2.3). Вводятся четыре типа соединений: соединяемые точки принадлежат одной области; соединяемые точки лежат в соседних рядах макроячеек и в соседних областях; соединяемые точки лежат в одном ряду макроячеек и в соседних областях; прочие соединения. Введенные типы отражают сложность соединений: чем больше номер типа, тем сложнее соединение. Упорядочение соединений для трассировки выполняется по возрастанию типа. Соединения одного и того же типа упорядочиваются по длине. Экспериментально установлено [40], что данное упорядочение способствует образованию «шлейфов» трасс, позволяющих подключаться к выводам элементов без переходов.

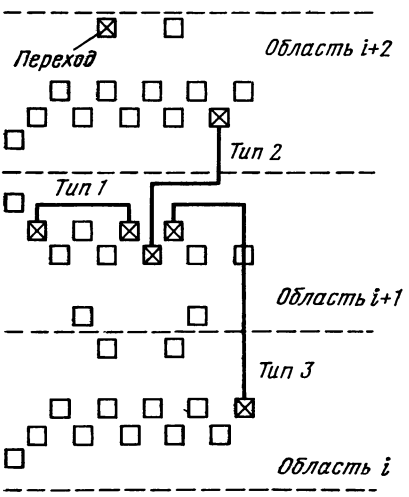


Рис. 2.3. Разбиение базового матричного кристалла на области

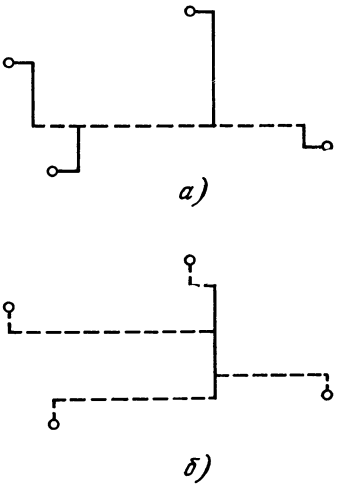


Рис. 2.4. Примеры деревьев заданной конфигурации

Трассировка БМК выполняется за несколько итераций. На каждой итерации последовательно просматриваются области, начиная с нижней. В ходе первой итерации трассируются соединения первых трех типов с числом поворотов, не превышающим 2 (простейшие конфигурации). Обоснованием такой стратегии явился опыт проектирования печатных плат и БИС, показавший, что реализация трасс сложной конфигурации на начальном этапе трассировки часто ведет к неоправданно большому расходу ресурсов ДРП и к блокировке, корректировка которой является трудоемкой, поскольку не может быть выполнена в локальной области.

На второй итерации трассируются нереализованные соединения первых трех типов с числом поворотов 3, 4. Последняя итерация включает трассировку соединений четвертого типа с ограниченным числом поворотов.

Программа, реализующая данный метод, использовалась для трассировки БМК И-200, И-3. При этом время трассировки БМК И-200 с размером ДРП 210×130 дискретов занимает 4—10 мин для схем, содержащих 300—600 парных соединений, а время трассировки БМК И-3 с размером ДРП 210×290 20—40 мин при числе соединений 700—900 [40]. Для БМК, содержащих $(5—10)\cdot10^3$ вентилей, размеры ДРП могут достигать 800×800 дискретов (например, БМК с $5\cdot10^3$ вентилей фирмы IBM описывается ДРП размером 700×700 [69]). Трассировка схем такой и бо́льшей степени интеграции требует перехода к методам с меньшей временно́й и емкостной сложностью. Обычно такие методы не используют ДРП и осуществляют поиск реализации электрических цепей в виде деревьев заданной конфигурации. Например, в методе, явившемся развитием канального алгоритма трассировки БИС для МВК «Эльбрус-2» [70], используется класс горизонтально-вертикальных деревьев (рис. 2.4). Построение деревьев ведется с учетом преимущественных направлений отрезков в слоях. На первом этапе делается попытка построения дерева, реализующего цепь, в виде, показанном на рис. 2.4,*а*, когда один сегмент (отмеченный штриховой линией) проводится в горизонтальном слое, а остальные — в вертикальном. Если такое дерево построить не удается, то делается попытка построения дерева, вид которого изображен на рис. 2.4,*б* (один сегмент в вертикальном слое).

При построении очередного дерева перебираются допустимые варианты, отличающиеся друг от друга положением отрезков, не подходящих к выводам. В качестве решения выбирается конфигурация, не имеющая пересечений с другими трассами. Экспериментально установлено, что число допустимых конфигураций дерева, в среднем, равно 1000. Для дотрассировки могут применяться как интерактивные средства, так и волновой алгоритм на ДРП, отображающем фрагмент кристалла.

Возможности описанного метода можно оценить по экспериментальным данным, полученным в ходе проектирования схемы устройства управления спецпроцессора МВК «Эльбрус-1». Эта схема содержала 1306 соединений и реализовалась на шестислойной плате, ДРП которой имело размер 274×328 дискретов. С помощью волнового алгоритма было реализовано 1223 соединения за 80 мин. Рассматриваемый алгоритм позволил построить 1139 соединений за 6,5 мин. После дотрассировки волновым алгоритмом число реализованных со-

единений увеличилось до 1215 (7% нереализованных трасс). Высокое быстродействие алгоритма, использующего фиксированные конфигурации деревьев, определяет его наибольшую эффективность при оценке результатов размещения.

Возможность использования при проектировании матричных БИС программ трассировки печатных плат подтверждает система ИСКРА. В этой системе применяется алгоритм трассировки, состоящий из трех этапов: предварительной трассировки, детальной трассировки и дотрассировки [71]. На первом этапе цепи разбиваются на парные соединения с помощью алгоритма Прима, определяется порядок трассировки цепей, из заданного набора выбираются конфигурации для реализации цепей и производится назначение парных соединений в каналы. В ходе детальной трассировки определяется форма трасс, реализующих соединения в канале, а на этапе дотрассировки делается попытка реализации непроведенных соединений с использованием трасс более сложной конфигурации как в автоматическом, так и интерактивном режимах. В результате формируется эскиз топологии (пример показан на рис. 2.5), на основании которого с помощью специальных программ осуществляется переход к топологическому чертежу (чертеж на рис. 2.6 соответствует эскизу, представленному на рис. 2.5).

С целью наиболее рационального использования ресурсов кристалла для реализации соединений во многих программах выполняется предварительное планирование (*глобальная трассировка*) трасс и *детальная трассировка* в областях БМК [72]. Причем для создания наилучших условий при детальной трассировке в ряде систем применяется глобальная перетрассировка [73, 74]. Процесс трассировки в системе LAMBDA [73] включает четыре фазы: глобальную трассировку, детальную трассировку, итерацион-

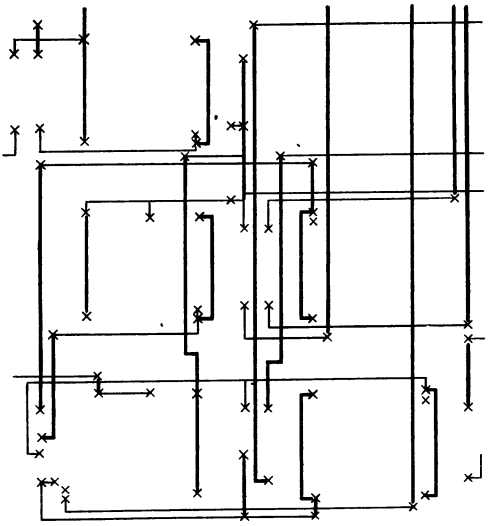


Рис. 2.5. Эскиз топологии на базовом матричном кристалле И-200

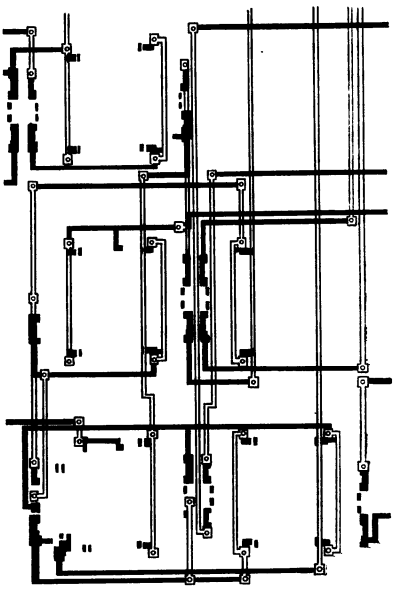


Рис. 2.6. Топологический чертеж фрагмента БИС

ную перетрассировку; интерактивную трассировку. В ходе первой фазы поле кристалла разбивается на регулярно расположенные макродискреты. Для каждого ребра e макродискрета определяется пропускная способность $c(e)$, равная максимальному числу трасс, которые могут пересекать e. Задача глобальной трассировки состоит в нахождении положения соединений с точностью до макродискретов, при котором для максимального числа ребер выполняется соотношение $f(e) \leqslant c(e)$, где $f(e)$ — число соединений, пересекающих ребро e. Для решения этой задачи последовательно осуществляется поиск минимального пути для каждого парного соединения. При этом используется следующая функция стоимости пересечения ребра, e: $l(e) = 1/(c(e) - f(e))^p$, где p — положительная константа.

Детальная трассировка выполняется с помощью *лабиринтного алгоритма* [76]. При этом делается попытка реализовать трассу в различных зонах. Сначала трасса строится в зоне, образованной из макродискретов, по которым прошла трасса на первой фазе. Если эта попытка не увенчалась успехом, то испытывается зона, которая представляет собой прямоугольник макродискретов, покрывающий соединяемые фрагменты цепи. При отрицательном исходе второй попытки испытывается зона, являющаяся прямоугольником макродискретов, покрывающим все блоки, которые инцидентны данной цепи.

На этапе *итерационной перетрассировки* [77] для каждого нереализованного соединения от обоих выводов s и t строится встречная волна (с метками S и T на рис. 2.7). Если существует дискрет, занятый трассой r (рис. 2.7,*а*), смежный с двумя дискретами, помеченными метками S и T, то удаление трассы r обеспечивает реализацию соединения (s, t). В противном случае (рис. 2.7,*б*) требуется снятие нескольких трасс, которые удаляются последовательно. Важным свойством данного метода перетрассировки является то, что если на некотором шаге существует трасса, удаление которой обеспечивает построение соединения (s, t), то алгоритм всегда находит такую трассу.

В системе MARS-M3 [78] в ходе глобальной трассировки осуществляется назначение транзитных трасс на проходы в элементах и в вертикальные каналы. Детальная трассировка основана на применении канального алгоритма с ог-

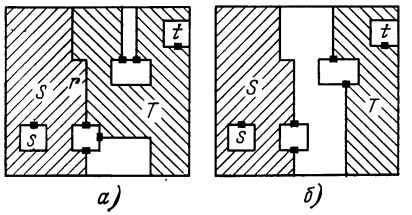


Рис. 2.7. Реализация соединения за счет снятия других соединений

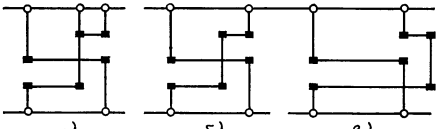


Рис. 2.8. Приемы устранения конфликта

раниченной перетрассировкой. С помощью этой программы было разработано более 1000 матричных БИС с полной автоматической реализацией трасс.

При проектировании биполярных матричных БИС, содержащих до 1500 вентилей, использовалась модифицированная программа трассировки системы LTX-2 [79]. Особенностью этапа детальной трассировки является заполнение канала. В ходе выполнения программы выравнивание плотности осуществляется за счет незначительного (примерно на 2%) увеличения длины трасс. Для БИС с 200—500 соединениями время трассировки без оптимизации занимало 2 мин, с оптимизацией 10 мин на ЭВМ VAX 11/780. После оптимизации, как правило, устранялись все участки с превышением пропускной способности.

В программе трассировки системы AULIS [60] применяется разбиение цепей на парные соединения с помощью алгоритма Прима. Полученные парные соединения упорядочиваются. В первую очередь трассируются шины питания и линейные соединения. Причем в каждой группе соединения трассируются в порядке возрастания длины. На этапе глобальной трассировки (распределения трасс по каналам) применяются простейшие конфигурации трасс. При этом целевой функцией является равномерность использования площади кристалла для трассировки. На этапе детальной трассировки делается попытка реализовать каждое соединение в ограниченном классе конфигураций трасс. На заключительном этапе нереализованные соединения трассируются с помощью волнового алгоритма без выделения слоев преимущественного направления трасс. Программа обеспечивает автоматическую трассировку 98—100% соединений.

В системе CADDAS [63] при проектировании матричных БИС средней степени интеграции (до 800 вентилей на кристалле) хорошие результаты достигались при использовании модификации волнового алгоритма. При 95%-ном заполнении ячеек БМК в большинстве случаев достигалась полная реализация соединений. Однако при увеличении числа вентилей до 1400 и более результаты трассировки оказались значительно хуже. Более того, кристалл был настолько заполнен трассами, что и в интерактивном режиме проведение не построенных трасс практически не представлялось возможным. Поэтому потребовалась разработка нового подхода к трассировке, основанного на классификации трасс и применении различных алгоритмов для построения трасс из разных классов. В зависимости от положения контактов цепи на кристалле вводятся шесть классов. Номер класса отражает сложность цепи. К первому классу относятся наиболее простые цепи (контакты расположены на одной стороне макроячейки), к шестому — наиболее сложные (контакты расположены на противоположных сторонах кристалла). Использование набора алгоритмов позволило получать полную реализацию цепей при 85%-ном заполнении ячеек больших кристаллов.

Поскольку в БМК можно выделить горизонтальные и вертикальные каналы для реализации соединений для трассировки матричных БИС часто применяются алгоритмы канального типа [80—84]. В таких алгоритмах процесс решения задачи состоит из следующих этапов: построение графа G конфликтов, поиск циклов в графе G, введение вертикальных сегментов трасс для устранения циклов, распределение горизонтальных сегментов по магистралям [8]. Устранение циклического конфликта показано на рис. 2.8. При определении положения вертикального сегмента

предпочтение отдается вертикальным магистралям, проходящим через выводы данной цепи (рис. 2.8,*а*). Если использовать такие магистрали невозможно, то сегмент располагается на свободной магистрали в *интервале цепи* (рис. 2.8,*б*). В худшем случае вертикальный сегмент располагается вне интервала цепи (рис. 2.8,*в*). Для назначения горизонтальных сегментов в горизонтальные магистрали могут использоваться метод ветвей и границ и методы ограниченного перебора [82]. В большинстве работ по канальной трассировке развивается такой подход, при котором горизонтальные участки трасс проводятся в одном слое, а вертикальные — в другом. Вместе с тем ведутся исследования трассировки канала без специализации слоев [85].

2.5. ХАРАКТЕРИСТИКИ СИСТЕМ ПРОЕКТИРОВАНИЯ МАТРИЧНЫХ БИС

По мере повышения степени интеграции матричных БИС системы их проектирования усовершенствуются: переводятся на ЭВМ более высокой производительности и пополняются новыми эффективными программами. Эволюция системы автоматизированного проектирования МВК семейства «Эльбрус» отражена в табл. 2.3 [40]. Отметим, что описанная в предыдущих параграфах система КОМПАС-82 является подсистемой САПР МВК «Эльбрус» и предназначена для разработки матричных БИС МВК «Эльбрус-3» с общей производительностью 1 млрд. операций в 1 с.

Кратко рассмотрим данные, полученные в ходе эксплуатации этой системы. При проектировании МВК «Эльбрус-3» описываемая САПР использовалась для разработки матричных БИС, являющихся аналогами печатных плат МВК «Эльбрус-2». При этом на проектирование первых 15 типов матричных БИС на БМК И-200М потребовалось 2 мес. рабочего времени и около 20 ч процессорного. В среднем, проектирование одной матричной БИС продолжалось 4—5 дней (2 дня — логическое проектирование, 2—3 дня — конструкторское). Редактирование результатов автоматического проектирования выполнялось в интерактив-

Таблица 2.3. *Характеристика САПР МВК «Эльбрус»*

Характеристика	Период, гг.		
	1980—1982	1982—1983	1983—1985
ЭВМ, используемая в САПР	БЭСМ-6	«Эльбрус-1»	«Эльбрус-2»
Быстродействие ЭВМ, млн. опер./с	1	3,5—4	25
Проектируемый объект	«Эльбрус-1»	«Эльбрус-2»	«Эльбрус-3»
Максимальный объем моделируемых схем, тыс. вентилей	1000	1000	2000
Скорость моделирования, тыс. событий/с	1	3	30
Степень интеграции БМК, вентилей	200	1500	$(5—10)\cdot 10^3$
Продолжительность цикла проектирования матричных БИС, мес.	0,5	0,5	0,5
Число типов матричных БИС, проектируемых в месяц	10	10—20	10—20

Таблица 2.4. *Характеристика спроектированных 15 типов матричных БИС на БМК И-200М*

Название матричной БИС	Использование ячеек БМК, %			Мощность, Вт	Максимальная задержка, нс	Число сигнальных цепей	Число автоматически не проведенных соединений
	внутренних	периферийных	общих				
МУН1	25	33	27	1,5	3,5	51	1
МПНК	48	37	45	3,0	5,6	96	6
МУК	66	79	69	4,6	9,2	120	9
МСВ2	22	50	28	1,9	4,2	60	0
МСВ3	29	25	27	1,8	6,2	55	4
ВХК3	31	16,5	27	1,6	4,3	46	2
ВХК4	49	50	49	2,6	3,3	60	3
МДП4	—	83	21	1,5	0,8	40	0
МКС1	21	37,5	25	1,6	3,2	55	1
МДЛ2	46	33	44	3,0	5,3	79	5
МУС2	59	33	53	3,6	4,3	116	7
МИНД	35	13	30	1,7	7,0	65	6
СВ34	31	42	33	2,3	4,2	70	5
М6К4	17	25	19	1,3	2,3	46	1
М2К8	48	25	42	2,4	4,3	73	5

Таблица 2.5. *Характеристика спроектированных 22 типов матричных БИС на БМК И-200М*

Название матричной БИС	Использование ячеек БМК, %			Мощность, Вт	Число сигнальных цепей	Число автоматически не проведенных соединений
	внутренних	периферийных	общих			
ВХР1	47	67	52	3,4	90	2
ВХР12	49	70	54	3,6	98	3
ВХР2	36	67	44	3,0	84	1
КР48	85	37	75	4,6	135	13
КСД	38	42	39	2,3	73	5
М3РК	82	50	74	4,7	122	6
М4К2	20	50	27	1,8	58	2
КДЛ1	96	25	88	5,2	157	18
МДП6	72	29	61	3,8	97	8
МКА	23	66	34	0,2	52	3
МПЦВ	49	33	45	3,0	88	4
МРП2	86	67	81	5,2	134	10
МСМК	37	62	43	4,2	116	6
МУВ7	8	33	14	1,0	38	0
МУВ9	19	36	27	1,6	53	1
МУС1	69	42	62	4,2	125	7
МУС4	38	21	33	2,4	67	3
СВ3	37	54	41	2,9	82	4
УПЧР	20	50	27	1,9	57	0
ВХКУ	43	96	56	4,0	100	6
КСР	54	50	52	3,7	108	5
МУВ78	21	52	29	1,9	60	1

ном режиме, причем работа проводилась сеансами по 2 ч за терминалом типа «Видеотон». В день проводилось, в среднем по 3 сеанса. Для дотрассировки 4—5 соединений требовалось 1—2 сеанса, а для 10 и более соединений 1—2 недели. На вывод комплекта конструкторской документации для одного типа матричных БИС расходовалось 15 мин процессорного времени. Характеристики спроектированных 37 типов матричных БИС приведены в табл. 2.4 и 2.5.

В ходе опытного проектирования была проведена разработка двух типов матричных БИС на БМК И-3М. При этом было использовано 83% элементов на кристалле, среднее число соединений равнялось 519, а время проектирования каждой матричной БИС не превысило двух недель.

Система AULIS [60] автоматического размещения и трассировки позволила сократить время проектирования матричных БИС до 1 дня (без учета времени ввода и контроля исходных данных). Расход процессорного времени ЭВМ SIEMENS 7.760 для матричной БИС, содержащей 400—2000 соединяемых выводов элементов и 80—400 ячеек, составляет 8—135 мин. Система обеспечивает автоматическую реализацию 98,6—100% соединений. На отдельных примерах она дала результаты, которые превзошли полученные вручную, хотя на неавтоматизированное проектирование затрачивалось, в среднем, по 8 недель на одну матричную БИС.

Подобные параметры имеют другие зарубежные системы: MARC [86], MASTER [4], MARS-M3 [78], LAMBDA [75], CAL-MOS и CAL-MP [87], CGAL [88], VGAUA [63].

Наряду с разработкой специализированных САПР матричных БИС с успехом проводится адаптация САПР, ориентированных на различные типы конструкций, к проектированию матричных БИС. Например, САПР ИСКРА [36] явилась развитием системы проектирования печатных плат, система АЛПАР [89, 90] была создана на основе САПР БИС на базовых ячейках.

По-видимому, наиболее ярко возможности применения САПР для проектирования матричных БИС проявились при разработке центрального процессора системы IBM/370 (одной из наиболее производительных ЭВМ) в виде одной микросхемы. Данный процессор содержит примерно 5 тыс. логических элементов (или 45 тыс. транзисторов) и около 11 тыс. соединений. Такая схема уже на порядок сложнее БИС, упоминавшихся ранее, и, по существу, относится к классу СБИС. Для реализации таких СБИС была разработана специальная конструкция БМК (большая вентильная Шотки/ТТЛ-матрица фирмы IBM [69]), относящаяся к группе линейчатых. Данный БМК, имея размеры 7×7 мм, содержит 7640 внутренних ячеек, три слоя металлизации (на последнем слое матричным способом расположены 200 контактных площадок для подключения внешних выводов, подходящие к ним соединения, а также шины питания и земли). Число выводов ячеек равно 33516, а размеры поля трассировки 800×600 дискретов.

САПР фирмы IBM реализует все этапы проектирования, перечисленные ранее. Особенность ее состоит в реализации иерархического размещения и трассировки, которая включает следующие шаги [91, 92]:

1) группирование 5 тысяч логических элементов в 114 узлов по критерию максимальной связности элементов внутри узлов;

2) последовательное размещение узлов в виде матрицы размером 12×12;

3) итерационное размещение узлов с помощью алгоритма парных перестановок по критерию минимума числа пересечений узлов соединениями (данный критерий связан с увеличением числа соединений между рядом расположенными узлами);

4) последовательное размещение элементов каждого узла в области расположения узла;

5) итерационное размещение элементов по критериям равномерности заполнения БМК трассами и элементами, минимума числа пересечения узлов соединениями и минимума суммарной длины соединений;

6) планирование трассировки (распределение проводников по каналам);

7) определение X-координат транзитных трасс в линейках элементов (по существу, на этом этапе формируются граничные условия для трассировки каждого канала);

8) трассировка соединений в каналах;

9) дотрассировка соединений с помощью алгоритма Ли в локальных областях;

10) интерактивная дотрассировка.

В связи с тем, что схема столь большого объема проектировалась впервые, параллельно применялись два способа размещения: автоматический (шаги 1—5) и интерактивный. Из трех вариантов автоматического размещения (каждый вариант потребовал около 3 ч машинного времени ЭВМ IBM 370/168) лучший был выбран для автоматической трассировки. Трассировка (шаги 6—9) заняла 1,5 ч машинного времени, после чего непроведенными остались 68 соединений, которые были реализованы на десятом шаге. При интерактивной дотрассировке использовалась распечатка результатов автоматической трассировки, потребовавшая лист размером около 4×5 м.

Интерактивное размещение предполагалось использовать в случае, если бы автоматическое размещение дало неудовлетворительные результаты. Оказалось, что автоматическая трассировка после интерактивного размещения не реализовала свыше 200 соединений. Поскольку обсуждаемая система применялась для проектирования схем различной сложности, экспериментально были получены оценки временной сложности $O(n^{1,2})$ и $O(n^{1,1})$ для размещения и трассировки соответственно, где n — число элементов матричных БИС. В целом на проектирование процессора на БМК (включая испытания опытного образца) было затрачено 9 мес., два из которых ушло на размещение, трассировку и контроль топологии. Характеристики, подобные обсуждаемым, имеет и система фирмы Texas Instruments (США), предназначенная для проектирования И²Л и ТТЛ-Шотки матричных БИС [93]. Она обес-

печивает сокращение времени проектирования в 4—10 раз для матричных БИС, содержащих до 6000 логических элементов.

Система фирмы IBM отличается от ранее рассмотренных САПР матричных БИС тем, что она позволила выйти на уровень проектирования СБИС. Существенную роль при этом сыграли введенные в систему элементы иерархического проектирования, что выразилось в двухступенчатом размещении (сначала узлов, а затем элементов в узлах) и двухступенчатой трассировке (распределение трасс по каналам и проведение соединений в каналах). Хотя публикации [69, 89, 90] по этой системе не являются исчерпывающими и в значительной мере носят рекламный характер, тем не менее они позволяют сделать вывод о том, что в процессе топологического проектирования полное описание проектируемой схемы помещалось в оперативной памяти ЭВМ. Именно это описание и использовалось при решении частных задач проектирования на выделенных подмножествах элементов и цепей. Такой способ решения позволил за счет небольших доработок имеющихся ранее программ осуществить проектирование схемы, содержащей 45 тыс. транзисторов [89]. Отметим, что элементы иерархического проектирования встречаются и в других системах (например, компоновка и размещение макроячеек, а затем — элементов в макроячейках; трассировка БМК по каналам и др.).

Вместе с тем ясно, что возможности указанного способа проектирования ограничены, поскольку степень интеграции матричных БИС растет значительно быстрее объема оперативной памяти ЭВМ, и уже сегодня актуальной является задача разработки СБИС, объем описания которой значительно превосходит объем ОЗУ самой производительной ЭВМ [94, 95, 180]. В этих условиях требуется прежде всего организовать иерархическое описание проектируемой схемы и вести проектирование уже с таким описанием. Кроме того, существенным параметром является размерность каждой решаемой задачи, так как для таких объектов, как СБИС (даже при том, что оперативной памяти достаточно) машинное время может и превысить допустимую величину. Возвращаясь к системе фирмы IBM вспомним, что в ней на размещение потребовалось времени в два раза больше, чем на трассировку, и это определялось прежде всего тем, что формирование узлов осуществлялось с помощью полного описания всей схемы. Отметим, что элементы иерархического проектирования матричных БИС, применялись также в САПР MARS-M3 [78], CGAL [88]. Принципы организации процесса проектирования при использовании иерархической структуры описания схемы и иерархическом решении основных задач реализованы в подсистеме АРТИС, предназначенной для САПР АСП-6М. Рассмотрению этих систем посвящены гл. 5—6. Теоретический анализ вопросов иерархического проектирования проводится в следующей главе.

Неотъемлемой частью интегрированных САПР БИС является *база данных*, в которой содержатся исходные и библиотечные данные, промежуточные и конечные результаты, правила проекти-

рования и проектные нормы. Связь с базой данных обычно обеспечивается с помощью системы управления базой данных (СУБД), общей для всех этапов проектирования [26, 73, 78, 96, 97]. Так, в системе CGAL [88] база данных представлена файловой структурой, в которой каждый файл рассматривается как источник или приемник промежуточной информации, а управление потоком данных осуществляется супервизором.

Большинство систем проектирования топологии матричных БИС разрабатывались как подсистемы более общих систем, включающих все основные этапы проектирования БИС. Так, система PHILO (IBM) [73] входит в состав САПР EDS, система CGAL — в UCAD [88]. При этом выходные данные топологического проектирования в зарубежных САПР, как правило, представлены в форме стандартных форматов CALMA CDSII (либо Aplicon Apple), что упрощает их дальнейшую обработку. В частности, появляется возможность использовать хорошо развитые средства интерактивного взаимодействия.

Диалог конструктора с ЭВМ остается необходимым и при создании матричных БИС. Причем использование интерактивного режима только для дотрассировки иногда оказывается малоэффективным, поскольку автоматические методы трассировки часто полностью заполняют монтажное пространство соединениями. Особенно сильно этот эффект проявляется при проектировании СБИС [63]. Значительно лучшие результаты дают средства, обеспечивающие активное вмешательство конструктора в процесс проектирования. Так, система LAMBDA позволяет в любой момент времени прервать автоматическое проектирование и с помощью графического дисплея внести корректировку, после чего система снова продолжает работу в автоматическом режиме. Во многих системах с иерархическим размещением эти же средства позволяют конструктору задать ориентировочное начальное размещение блоков и оценивают его допустимость.

Успехи, достигнутые при создании высокопроизводительных микро-ЭВМ, позволили аппаратно реализовать решение некоторых задач проектирования БИС. Сейчас достаточно широко применяются различные трассировочные машины [98, 99]. За счет распараллеливания операций удалось значительно повысить производительность вычислений. Причем время счета линейно возрастает с ростом размерности задачи (у последовательных ЭВМ этот закон квадратичен). Регулярность и детерминированность структуры БМК позволяют успешно применять эти машины для глобальной и детальной трассировки. Так, в машине фирмы IBM [98] реализован алгоритм Ли — Мура со специальной функцией штрафа за пересечение границ дискретов поля, предназначенный для глобальной трассировки БМК.

В связи с повышением степени интеграции появляется возможность реализации достаточно сложных функциональных устройств (например, микропроцессоров) на одном кристалле. При этом возникает необходимость использования на БМК помимо произ-

вольной логики блоков с регулярной структурой (таких, как ЗУПВ, ПЗУ, ПЛМ). Можно отметить два направления, по которым ведутся работы для решения данной проблемы. Первое заключается в выпуске БМК, содержащих фиксированные блоки памяти. Второе направление состоит во включении в стандартные фотошаблоны БМК специализированных областей, соответствующих блокам [73]. Топология блоков либо разрабатывается заранее и хранится в библиотеке, либо разрабатывается в ходе текущего проектирования, как это делается в системе PHILO. При этом конструкция блоков должна удовлетворять ограничениям, обеспечивающим согласование его с элементами БМК. Так, размеры блоков должны быть кратны размерам ячеек БМК, а выводы должны располагаться на границах каналов. На размещение блоков накладывается единственное ограничение — они должны располагаться вплотную к левой или правой границе внутренней области БМК. Последнее требование обеспечивает неразрывность шин питания, которые подводятся к ячейкам от центрального вертикального канала. Описанный подход позволяет использовать фрагменты топологии БМК со структурой, отличной от типовой.

Проектирование матричных БИС является комплексной проблемой, решение которой возможно совместными усилиями конструкторов БМК и разработчиков САПР. Именно поэтому в настоящее время некоторые системы содержат как подсистему проектирования топологии матричных БИС, так и подсистему конструирования БМК [100]. Данный принцип организации САПР матричных схем предполагает использование программ оценки размеров БМК для реализации схемы, оценки средней длины соединений на кристалле, а также анализ использования ресурсов кристалла применяемыми алгоритмами проектирования [5, 14, 25, 101, 178]. Используя статистические модели конструкции кристалла и процесса проектирования, в ряде случаев удается получить оценки и рекомендации, которые могут оказаться полезными при разработке матричных БИС.

3. АНАЛИЗ ИЕРАРХИЧЕСКОГО ПОДХОДА К ПРОЕКТИРОВАНИЮ

3.1. СУЩНОСТЬ ИЕРАРХИЧЕСКОГО ПОДХОДА К ПРОЕКТИРОВАНИЮ

Проектирование СБИС по трудоемкости и сложности сравнимо с разработкой таких объектов, как ЭВМ, летательный аппарат и других больших систем. Общий подход к решению проблемы такого класса состоит в разбиении исходной задачи проектирования на подзадачи. Во-первых, выделяются этапы, на кото-

рых достигается лишь некоторое подмножество целей из всей исходной совокупности. В качестве примера можно привести этап разбиения проектируемой схемы, когда минимизируется число межблочных связей. Во-вторых, объект проектирования разделяется на *фрагменты* и проектирование каждого фрагмента ведется в определенном смысле самостоятельно. Ясно, что последний принцип применим и при проектировании каждого выделенного фрагмента. Поэтому можно говорить об *иерархии фрагментов* и, следовательно, об *иерархическом подходе* к проектированию СБИС. Очень важно, что этот подход согласуется с существующим иерархическим разделением СБИС по функциональному признаку [102].

Прежде чем рассматривать иерархический подход, кратко остановимся на возможностях традиционного подхода к решению задачи. Предположим, необходимо спроектировать интегральную схему, содержащую n элементов. Пусть в процессе решения задачи применяется алгоритм с временной сложностью $O(n^{\alpha_1})$ и емкостной сложностью $O(n^{\alpha_2})$ [103]. Оценим затраты машинного времени и оперативной памяти ЭВМ при проектировании СБИС, заданной описанием на элементном уровне. Современные СБИС содержат до 10^6 элементов. В дальнейшем будем анализировать задачу с $n=10^6$. Для наиболее распространенных алгоритмов $\alpha_1=2$, $\alpha_2=1$, поэтому число T операций, выполняемых при решении задачи, удовлетворяет соотношению $T\approx k_1 n^2$, а требуемый объем памяти $V=k_2 n$. Поскольку мультипликативные составляющие k_1 и k_2, как правило, больше 10, время решения рассматриваемой задачи будет не менее 100 сут. непрерывного счета ЭВМ с быстродействием 1 млн. операций/с (ЭВМ типа ЕС 1060). Требуемый объем оперативной памяти достигает нескольких десятков мегабайт, что намного превосходит возможности современных ЭВМ.

Таким образом, возникают две проблемы. Во-первых, требуется организовать процесс решения в условиях, когда невозможно размещение в оперативной памяти ЭВМ полной информации о проектируемом объекте. Во-вторых, необходимо получить решение всей задачи за приемлемый промежуток времени. Решение обеих проблем может быть найдено в рамках иерархического подхода. Согласно данному подходу структура рассматриваемого объекта (в данном случае объекта проектирования) представляется ориентированным графом D без циклов, в котором имеется единственная вершина s без входящих ребер. Такой граф называется *корневым деревом*, а вершина s — *корнем*. Между вершинами и частями (фрагментами) объекта устанавливается взаимно однозначное соответствие. Вершина s отображает весь объект, а каждое ребро (a, b) из вершины a в b показывает, что фрагмент, соответствующий вершине b, входит во фрагмент, отображаемый вершиной a.

Вершины, не имеющие выходящих ребер, называются *листьями*, а остальные вершины — *внутренними*. Каждый лист соответствует фрагменту низшего уровня иерархии. Если из вершины a

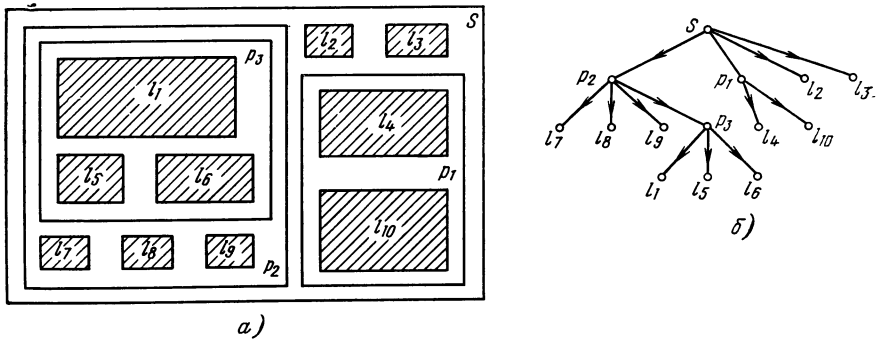


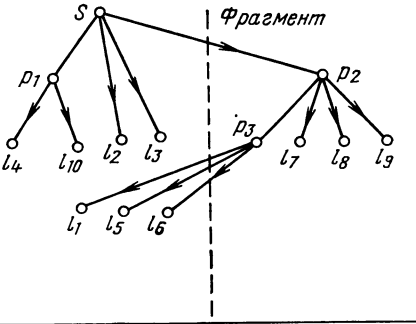
Рис. 3.1. Пример вложенных прямоугольников

идет ребро в вершину b, то a называется *отцом вершины* b, а b — *сыном вершины* a. В качестве примера простейшего объекта, имеющего иерархическую структуру, можно привести детскую игрушку — матрешку. Ей соответствует граф в виде простой цепи. Другой пример объекта, представляющего собой систему прямоугольников, показан на рис. 3.1,*а*, где прямоугольники низшего уровня заштрихованы. Граф, соответствующий данному объекту, приведен на рис. 3.1,*б*.

Процесс решения исходной задачи согласно иерархическому подходу заменяется многократным решением аналогичных задач меньшей размерности. Число таких подзадач равно числу внутренних вершин дерева D. Так, если требуется разместить прямоугольники l_1—l_{10} внутри прямоугольника s, то выполняется следующая последовательность действий. Размещаются l_1, l_5, l_6 внутри p_3; l_7, l_8, l_9, p_3 — внутри p_2; l_4, l_{10} — внутри p_1; p_2, p_1, l_2, l_3 — в s. Важно, что порядок решения указанных подзадач может быть выбран произвольно. Например, можно решать задачу «сверху вниз», когда подзадача-сын решается после подзадачи-отца. Возможна и обратная стратегия («снизу вверх») или комбинация указанных стратегий. Конкретные последовательности решения подзадач позволяют получить различные схемы вычислений, аналогичные известным (итерационную, рекурсивную, схему подстановок). При проектировании СБИС логические элементы отображаются листьями иерархического дерева, а внутренние вершины соответствуют некоторым частям схемы (например, подсхемам, выделенным по функциональному принципу).

Сложность решения задачи, как правило, определяется числом n независимых переменных — параметром, который обычно называют *размерностью задачи*. При проектировании СБИС размерность задачи — это число логических элементов (или число листьев в иерархическом дереве). Отметим, что с элементами иерархического подхода мы уже сталкивались в гл. 1, когда от проектирования на уровне транзисторов перешли к проектированию на уровне логических элементов. При этом весь цикл проектирования разбивается на два этапа: разработка топологии логических эле-

ментов и проектирование СБИС с использованием логических элементов с фиксированной топологией. В данном случае иерархия этапов проектирования определена человеком. В дальнейшем будем рассматривать процесс проектирования, в ходе которого иерархия структуры СБИС и отдельных этапов вводится автоматически без участия разработчика аппаратуры.



Внешние ЗУ	ОЗУ
Программное обеспечение для решения задачи	Программы для решения подзадачи
Описание иерархического дерева	Описание фрагмента
Описание всех фрагментов	

Рис. 3.2. Размещение информации об объекте проектирования

Очевидно, что иерархический подход позволяет так организовать описание и хранение данных (с использованием внешних ЗУ), чтобы ограничения на доступный объем оперативной памяти не нарушались. При этом для решения каждой подзадачи в оперативную память записывается лишь небольшая часть (описание одного фрагмента) полной информации об объекте (рис. 3.2 представляет наглядную интерпретацию данной ситуации).

Разбиение исходной задачи на подзадачи сужает область поиска конечной цели и может приводить к некоторому ухудшению показателей качества объектов проектирования. Фактически, это естественная плата за расширение круга решаемых задач (в частности, по размерности). Вместе с тем, учитывая эвристический характер основных методов решения задач проектирования, следует отметить, что при иерархическом подходе появляется возможность оценить свойства объекта в целом и в результате повышается вероятность получения решения, близкого к оптимальному. Оценивая целесообразность использования принципа фрагментации, уместно вспомнить опыт разработки сложных программ, когда от программирования в машинных командах был сделан переход к программированию на алгоритмических языках. Хотя новые программы требовали большего объема машинной памяти и работали медленнее, выигрыш за счет сокращения времени разработки программ и возможности создания больших комплексов программ перекрыл все недостатки. Возможность эффективного применения принципа фрагментации подтверждается и разработкой методов проектирования, основанных на использовании типовых решений и макромоделей. Наконец, еще одно достоинство разбиения задач на подзадачи (в частности, фрагментация) заключается в получении задач, инвариантных к изменению техноло-

гического процесса производства аппаратуры, что обеспечивает универсальность метода и «живучесть» систем проектирования, созданных на его основе.

3.2. АНАЛИЗ ВРЕМЕНИ РЕШЕНИЯ ЗАДАЧИ

Предпосылкой для сокращения времени решения является переход от одной задачи большой размерности к совокупности задач малой размерности. Для получения количественных оценок изменения времени решения необходимо провести детальный анализ вычислительного процесса с учетом временной сложности применяемых алгоритмов, числа подзадач и размерности каждой подзадачи. Возможность уменьшения времени счета при использовании иерархического подхода исследуем на упрощенной задаче, когда вся необходимая для решения информация может быть размещена в оперативной памяти ЭВМ.

Пусть размерность исходной задачи n. Требуется получить минимальное время решения исходной задачи решением совокупности задач размерности x $(x<n)$, где x — искомая величина. В соответствии с постановкой задачи каждая внутренняя вершина иерархического дерева D содержит x сыновей. Такой граф называют *однородным деревом* степени x. Примеры однородных деревьев степени 3 показаны на рис. 3.3. Ясно, что однородное дерево заданной степени существует не при любом n. Определим необходимое и достаточное условие существования однородного дерева степени x с n листьями.

Предположим, такое дерево существует. Подсчитаем число V его внутренних вершин. Для этого используем многошаговый процесс преобразования дерева, удаляя на каждом шаге x листьев, являющихся сыновьями одной и той же вершины. Поскольку при этом вершина-отец удаленных листьев становится листом, то на каждом шаге уменьшение числа листьев в дереве равно x—1. Ясно, что число шагов равно числу внутренних вершин, и процесс завершается, когда дерево содержит единственную вершину, являющуюся одновременно и корнем, и листом. Поэтому

$$n-V(x-1)=1,$$

или

$$V=(n-1)/(x-1). \tag{3.1}$$

Следовательно, в однородном дереве степени x число n—1 кратно x—1. Покажем, что последнее условие является достаточным для существования однородного дерева степени x. Для этого приведем алгоритм построения дерева D_x с требуемыми параметрами. На подготовительном шаге строим граф D', состоящий из единственной вершины. Затем на каждом последующем шаге к произвольному листу D' добавляем x сыновей. Понятно, что в результате такого преобразования граф остается однородным, а число его листьев, уменьшенное на единицу, кратно x—1. Поэтому после

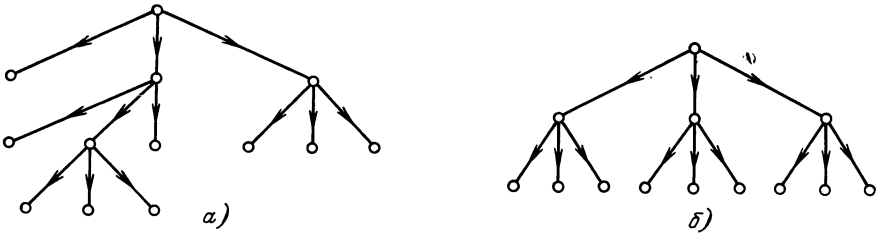


Рис. 3.3. Примеры однородных деревьев

конечного числа шагов будет построен граф $D'=D_x$. Таким образом, справедлива следующая теорема.

Теорема 3.1. Необходимым и достаточным условием существования разбиения задачи размерности n на подзадачи размерности x является кратность числа n—1 числу x—1.

Следствие 3.1. Для задачи любой размерности всегда существует ее разбиение на подзадачи размерности 2.

При произвольном n, когда n—1 не делится без остатка на x—1, можно найти наименьшее целое n' больше n, такое, что n'—1 кратно x—1. Построив дерево D_x для n' и удалив из него n'—n листьев, получим дерево для n, близкое к однородному. В дальнейшем для простоты изложения будем полагать, что n—1 кратно x—1.

Для реализации иерархического подхода к решению поставленной выше задачи необходимы два алгоритма: первый (А1) — для выделения подзадач заданной размерности x, второй (А2) — для решения задачи размерности x. Поскольку алгоритм А1 необходимо применять к исходному объекту размерности n, то сокращение времени решения всей задачи может быть получено лишь при условии, что временная сложность А1 ниже, чем А2.

Время работы А1 зависит от выбора структуры однородного дерева D_x. Согласно *принципу балансировки* [103] минимальное время работы достигается в том случае, когда на каждом шаге выделяются фрагменты равной размерности. В этом случае вычислительному процессу соответствует *симметричное дерево* D_x, пример которого показан на рис. 3.3,*б*. Характерное свойство симметричного дерева состоит в том, что каждый *путь* от корня к листу имеет одну и ту же длину l и выполняется соотношение

$$n=x^l. \tag{3.2}$$

Отметим, что однородное дерево D_x соответствует разбиению задачи, при котором создаются предпосылки для получения решений конструкторских задач, близких к оптимальным. Например, размещение объектов приблизительно равных размеров позволяет, как правило, получить результаты лучшие, чем в случае сильно отличающихся по размерам объектов. В дальнейшем будем полагать, что величины n и x связаны соотношением (3.2).

Время работы алгоритма А1 зависит не только от структуры иерархического дерева, но и от стратегии разбиения задач на подзадачи. Например, если требуется задачу размерности 16 свести к решению задач размерности 2, то можно поступить следующим образом. Сначала исходную задачу разбить на две подзадачи размерности 8. Этому шагу соответствует переход от графа на рис. 3.4,*а* к графу, изображенному на рис. 3.4,*б*. Продолжая процесс разбиения задач, последовательно получаем графы на рис. 3.4,*в* и *г*. Данная последовательность преобразования задач соответствует стратегии «сверху вниз». Возможен и другой путь получения конечного иерархического дерева (рис. 3.4,*д*—*ж*). При этом на первом шаге исходный объект разбивается на четыре

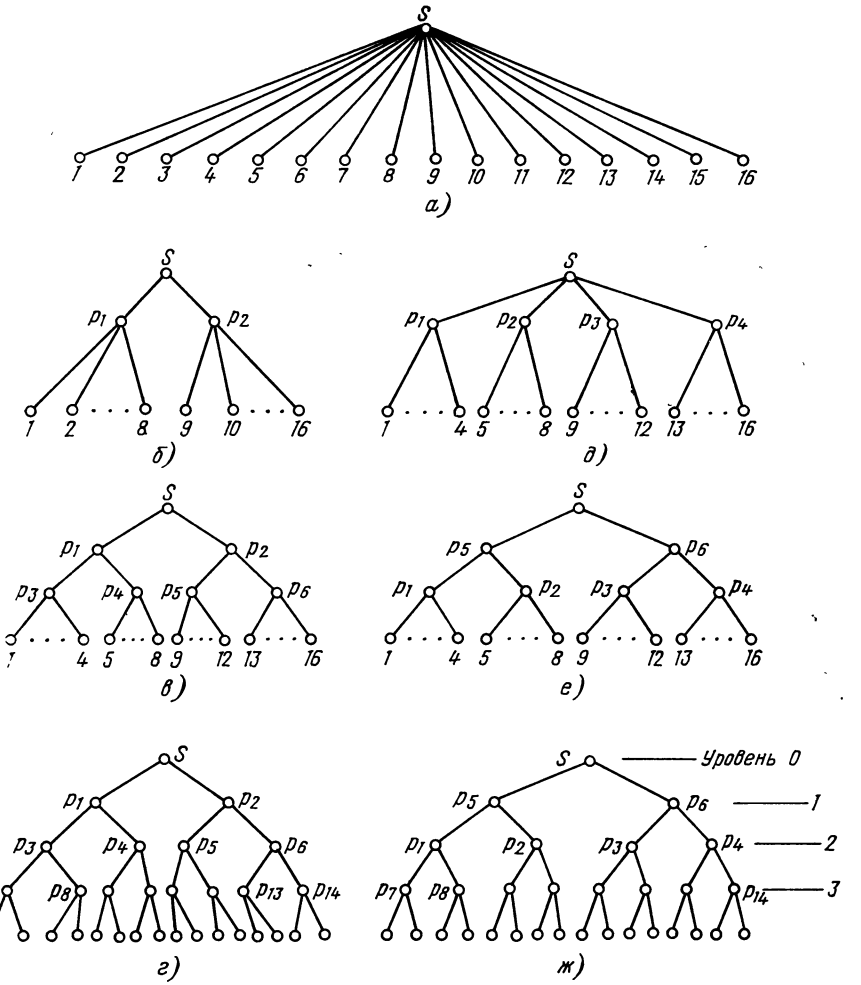


Рис. 3.4. Пример разбиения задачи на подзадачи

фрагмента p_1—p_4, содержащие по четыре элемента низшего уровня. В результате возникают пять задач размерности 4. Среди них имеются четыре задачи относительно элементов нижнего уровня и одна задача относительно фрагментов. Преобразуя каждую из указанных задач к совокупности задач размерности 2, получаем окончательный вид графа. Такой путь соответствует рекурсивной стратегии. Ясно, что возможно применение и других стратегий.

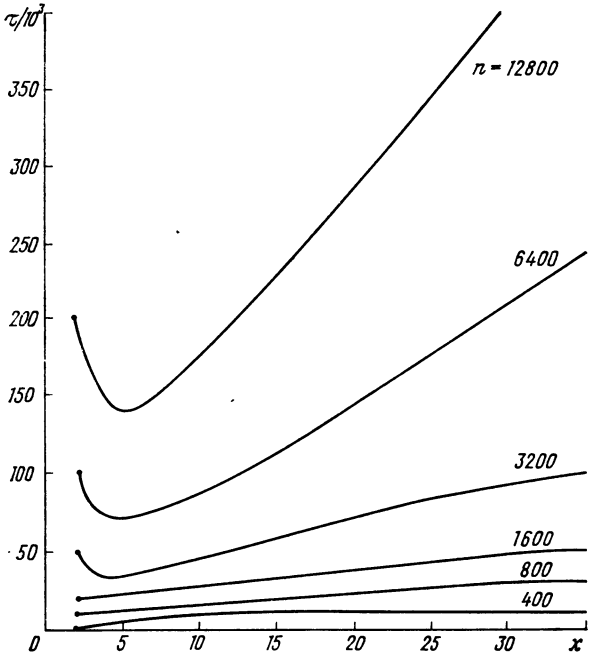


Рис. 3.5. Зависимость времени счета от параметров задачи при $\alpha=1$, $\beta=2$

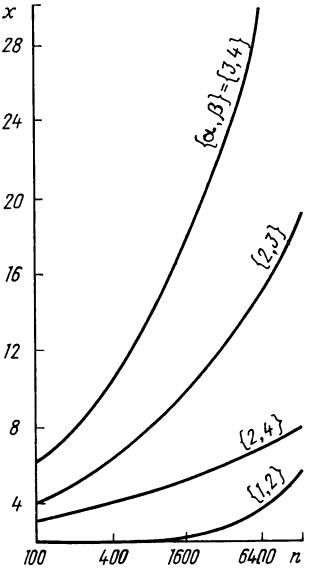


Рис. 3.6. Зависимости оптимального размера подзадач для задачи размера n при применении алгоритмов A1 и A2 с временными сложностями $O(m^\alpha)$ и $O(m^\beta)$

Таблица 3.1. *Зависимость величины $\tau/10^6$ от размерности n задачи при решении всей задачи алгоритмом A2*

β	n							
	100	200	400	800	1600	3200	6400	12800
2	0,01	0,04	0,16	0,64	2,56	10,24	41	165
3	1,0	8,0	64,0	512	$4\cdot10^3$	$26\cdot10^4$	$32\cdot10^3$	$2\cdot10^6$
4	$1\cdot10^2$	$16\cdot10^2$	$25\cdot10^3$	$4\cdot10^5$	$64\cdot10^5$	$1\cdot10^8$	$16\cdot10^8$	$25\cdot10^9$

Не останавливаясь пока на вопросе выбора наилучшей стратегии, оценим время решения задачи при разбиении задач «сверху вниз».

Предположим, что временна́я сложность алгоритма A1 вычисляется по формуле $t(A1, m) = c_1 m^\alpha$, где m — размерность задачи, решаемой алгоритмом A1; c_1 — константа; $\alpha \geqslant 1$. Тогда время работы алгоритма A1 в ходе решения всей задачи $\tau_1 = c_1 (n^\alpha + x(n/x)^\alpha + \ldots + x^l (n/x^l)^\alpha)$.

Поскольку дерево однородное, то минимальная размерность решаемой задачи разбиения равна x^2. Откуда $n/x^l = x^2$. Следовательно, $l = \log_x n - 2$ и

$$\tau_1 = c_1 n^\alpha \sum_{l=0}^{\log_x n - 2} x^{l(1-\alpha)}, \quad n \geqslant x^2.$$

Просуммировав правую часть, получим

$$\tau_1 = \begin{cases} c_1 (n^\alpha - n x^{\alpha-1})/(1 - x^{1-\alpha}) & \text{при } \alpha > 1; \\ c_1 n (\log_x n - 1) & \text{при } \alpha = 1. \end{cases}$$

Независимо от вида однородного дерева D_x алгоритм A2 применяется V раз к задаче размерности x. Если временна́я сложность алгоритма A2 вычисляется по формуле $t(A2, m) = c_2 m^\beta$, то с учетом (3.1) время работы алгоритма A2 определяется выражением

$$\tau_2 = c_2 x^\beta (n-1)/(x-1).$$

Время решения всей задачи $\tau = \tau_1 + \tau_2$.

Характер зависимости времени τ от величин n и x при $\alpha = 1$, $\beta = 2$ ($c_1 = c_2 = 1$) иллюстрируется рис. 3.5. Как видно из графиков, для каждого n можно указать размерность x подзадач, при которой время решения задачи достигает минимума. Графики зависимости оптимального x от размерности задачи для различных сочетаний α и β приведены на рис. 3.6.

Для того чтобы показать, насколько применение однородного дерева позволяет уменьшить требуемое машинное время, в табл. 3.1 приведены расчетные данные затрат времени на решение исходной задачи алгоритмом A2. Сравнивая данные последней таблицы с временем счета при оптимальных значениях размерности подзадач, получаем относительное уменьшение машинного време-

Таблица 3.2. *Выигрыш во времени при применении однородного дерева*

α	β	n							
		100	200	400	800	1600	3200	6400	12800
1	2	10	15	24	64	128	335	600	1150
1	3	$1 \cdot 10^3$	$2 \cdot 10^3$	$6{,}4 \cdot 10^3$	$5 \cdot 10^4$	$1{,}3 \cdot 10^5$	$5{,}3 \cdot 10^5$	$2 \cdot 10^6$	$8 \cdot 10^6$
1	4	$5 \cdot 10^4$	$3 \cdot 10^5$	$25 \cdot 10^5$	$2 \cdot 10^7$	$16 \cdot 10^7$	$1 \cdot 10^9$	$1 \cdot 10^{10}$	$6 \cdot 10^{10}$
2	3	65	160	300	600	1300	$3 \cdot 10^3$	$8 \cdot 10^3$	$12 \cdot 10^3$
2	4	$5 \cdot 10^3$	$2 \cdot 10^4$	$1 \cdot 10^5$	$21 \cdot 10^5$	$7 \cdot 10^6$	$7 \cdot 10^6$	$3 \cdot 10^7$	$12 \cdot 10^8$
3	4	$1 \cdot 10^2$	$2 \cdot 10^2$	$4 \cdot 10^2$	$8 \cdot 10^2$	$15 \cdot 10^2$	$3 \cdot 10^3$	$6 \cdot 10^3$	$11 \cdot 10^3$

ни (табл. 3.2). Анализируя графики на рис. 3.5, можно сделать вывод, что произвольный выбор размерности подзадач в ряде случаев приводит к значительным потерям машинного времени. Поэтому при решении практических задач параметр x целесообразно определять с учетом экспериментальных оценок трудоемкости применяемых алгоритмов.

3.3. МЕТОДИКА ПОСТРОЕНИЯ ОПТИМАЛЬНОГО ИЕРАРХИЧЕСКОГО ДЕРЕВА

До сих пор мы рассматривали возможности иерархического подхода на примере однородного дерева. В общем случае может оказаться целесообразным выделение подзадач различной размерности [104]. Будем считать в дальнейшем, что рассматриваемая нами задача z размерности n может быть решена с помощью алгоритма А2. Применяя алгоритм А1 к исходным данным, можно от задачи z перейти к совокупности позадач $z_1, z_2, \ldots, z_k$ размерности $x_1, x_2, \ldots, x_k$ $(x_1+\ldots+x_k=n)$ и задаче z' размерности k. При этом: 1) трудоемкость А1 не зависит от k; 2) каждая задача z_i $(i=\overline{1, k})$ и z' может быть решена алгоритмом А2 и допускает применение алгоритма А1.

В дальнейшем удобно будет различать задачи z_i и z'. Первую будем называть *z-задачей*, вторую — *z'-задачей*. Проведем анализ структуры *оптимального иерархического дерева* D^*, соответствующего минимальному времени решения задачи. Для этого введем понятие *мощности* $|v|$ *вершины* v. Мощность $|v|$ равна числу листьев, достижимых из v (вершина v_1 считается достижимой из вершины v_2, если существует путь из v_2 в v_1). Например, мощность листа равна единице, а $|s|=n$. Согласно принципу балансировки [103] минимальное время работы достигается в том случае, когда все сыновья одной и той же вершины имеют одинаковую мощность. Если длина пути из корня s во внутреннюю вершину v равна i, то будем говорить, что v расположена в D на i-м *иерархическом уровне* (рис. 3.4,*ж*). В дальнейшем уровень с максимальным номером (содержащий отцов листьев) будем называть *нижним*.

Теорема 3.2. В оптимальном иерархическом дереве D^* каждое поддерево оптимально.

Доказательство. Поскольку согласно иерархическому подходу все подзадачи решаются последовательно, то общее время решения равно сумме времени решения отдельных подзадач. Пусть дерево D^* содержит неоптимальное поддерево D'. Тогда $T(D^*)=T(D')+T(D^* \setminus D')$, где $T(D^*)$ — время решения всей задачи; $T(D')$ — время решения позадачи, соответствующей корню дерева D'; $T(D^* \setminus D')$ — время решения остальных подзадач.

Так как D' не оптимально, то существует $\bar{D}'$, для которого $T(\bar{D}')<T(D')$. Заменив D' на $\bar{D}'$ получим $\bar{D}^*$, для которого $T(\bar{D}^*)<T(D^*)$, что противоречит оптимальности D^*.

Теорема 3.3. Существует оптимальное иерархическое дерево D^*, в котором вершины одного и того же уровня имеют равные степени.

Доказательство. Рассмотрим произвольную вершину v i-го иерархического уровня. При заданной стратегии построения дерева D время решения подзадачи, соответствующей вершине v, определяется: мощностью $|v|$, временными сложностями алгоритмов A1 и A2 и структурой дерева $D(v)$ с корнем v. Согласно теореме 3.2 поддерево $D(v)$ оптимально. Так как мощности всех вершин уровня равны, то все поддеревья с корнями на i-м уровне могут иметь идентичную структуру. Следовательно, степени вершин i-го уровня равны.

Рассмотрим методику построения оптимального иерархического дерева D^*. В общем случае: исходя из особенностей решаемой задачи, на структуру искомого дерева могут накладываться ограничения. Будем рассматривать ограничения на минимальную размерность задачи, решаемой алгоритмом A2 (минимальная степень ρ_1 внутренней вершины D^*). В ряде случаев (и, в частности, при проектировании СБИС) целесообразно ввести дополнительное ограничение ρ_2 $(\rho_2 > \rho_1)$ на степень вершин нижнего уровня. При введенных ограничениях минимальная размерность задачи, к которой применим иерархический подход, равна $\rho_1\rho_2$.

Согласно рассматриваемому подходу трудоемкость решения задачи размерности n при заданном k удовлетворяет следующему рекуррентному соотношению: $T(n, k) = t_1(n) + kT(n/k) + T'(k)$, где $t_1(n) = c_1 n^{\alpha}$ — трудоемкость выделения k подзадач; второе слагаемое определяет трудоемкость решения k задач размерности n/k, третье — трудоемкость решения задачи z' размерности k. Трудоемкость $T(x)$ решения задачи при использовании оптимального иерархического дерева $D(x)$ определяется выражением

$$T(x) = \min_{k=\overline{\rho_1, x/\rho_2}} T(x, k).$$

Поскольку дерево $D(k)$, соответствующее задаче z', не содержит вершин нижнего уровня искомого дерева D^*, то оптимальное дерево для задачи z' должно определяться с учетом единственного ограничения ρ_1 на степени всех внутренних вершин. Следовательно,

$$T'(k) = \min_{i=\overline{\rho_1, k/\rho_1}} T'(k, i),$$

$$T'(k, i) = t_1(k) + iT'(k/i) + T'(i).$$

Приведенные соотношения позволяют последовательно, начиная с задачи минимальной размерности, определить структуру оптимального дерева и трудоемкость для задач всех размерностей вплоть до n. Данные вычисления могут быть выполнены на ЭВМ.

Методика построения дерева D^* включает два этапа. Первый этап выполняется однократно для максимальной размерности $n = n_{\max}$ и содержит следующие шаги:

1. Вычисление начальных значений функций $T(j)$ и $T'(i)$

$$T(j)=ct\ (A2,\ j);\ K(j)=0,\ j=\overline{2,\ \rho_1\rho_2-1};$$

$$T'(i)=ct\ (A2,\ i),\ K'(i)=0,\ i=\overline{2,\ \rho_1^2-1};$$

где $c=c_2/c_1$; T, T' — списки значений трудоемкости; K, K' — списки оптимальных размерностей задачи z'.

2. Выполнение процедуры RECURS $(n/2,\ T',\ K',\ \rho_1,\ \rho_1,\ T')$.

3. Выполнение процедуры RECURS $(n,\ T,\ K,\ \rho_2,\ \rho_1,\ T')$.

4. Вывод результата.

Описание процедуры RECURS $(x,\ T,\ K,\ \rho_2,\ \rho_1,\ T')$:

1. IND$=0$.

2. Для I от $\rho_1\rho_2$ до x выполнить шаги 3—16.

3. $T(I)=\infty$.

4. Для J от ρ_1 до x/ρ_2 выполнить шаги 5—11.

5. Вычислить минимальную мощность сына: $M=I/J$.

6. Если $M<\rho_2$, перейти к шагу 16.

7. Определить число $M1$ сыновей с мощностью $M+1$: $M1=I-MJ$.

8. Вычислить $T(I,\ J)=t(A1,\ I)+(J-M1)T(M)+M1\cdot T(M+1)+T'(J)$.

9. Если $T(I,\ J)\geqslant T(I)$, перейти к шагу 11.

10. Запомнить наименьшее значение трудоемкости и соответствующее число выделенных подзадач:

$$T(I)=T(I,\ J),\ K(I)=J.$$

11. Продолжение цикла по J.

12. Если IND$=1$, перейти к шагу 16.

13. Вычислить трудоемкость решения без применения иерархического подхода $TP=ct(A2,\ I)$.

14. Если $TP\geqslant T(I)$, то $IND=1$ и перейти к шагу 16.

15. $T(I)=TP$, $K(I)=0$.

16. Продолжение цикла по I.

17. Конец вычислений.

Полученные на первом этапе списки K и K' хранятся во внешней памяти и используются на втором этапе для выделения подзадач и построения оптимального дерева для конкретной задачи z. Этот процесс включает следующие шаги:

1. Занести исходную задачу z в список R разбиваемых задач. Сформировать дерево D^*, состоящее из единственной вершины z.

2. Пока $R\neq\emptyset$, для каждого элемента $r\in R$ выполнить шаги 3—10.

3. Исключить r из R.

4. Если r является z'-задачей, перейти к шагу 8.

5. Определить число подзадач для r по формуле $k=K(|r|)$, где $|r|$ — размерность задачи r.

6. Если $k=0$, перейти к шагу 2.

7. Применяя A1 к r, сформировать подзадачи $z_1,\ z_2,\ \ldots,\ z_k,\ z'$ и включить их в R. Дерево D^* дополнить вершинами $z_1,\ \ldots,\ z_k$, являющимися сыновьями вершины r. Переименовать вершину r в z'. Перейти к шагу 2.

8. Определить число подзадач для r по формуле $k=K'(|r|)$.

9. Если $k=0$, перейти к шагу 2.

10. Применяя $A1$ к r, сформировать подзадачи $z_1,\ z_2,\ \ldots,\ z_k,\ z'$ и включить их в R в виде z'-задач. Дополнить дерево D^* вершинами $z_1,\ \ldots,\ z_k,\ z'$. Соединить

z' с сыновьями z_1, z_2, ..., z_k, а отца вершины r с сыном z'. Сыновей вершины r соединить с отцами z_1, z_2, ..., z_k в соответствии с результатом алгоритма A1. Удалить r из D^* и перейти к шагу 2.

11. Конец вычислений.

В качестве примера использования описанной методики рассмотрим задачу размещения 1000 элементов. Пусть трудоемкость $A1$ ограничивается функцией $n\ln(n)$, трудоемкость $A2$ — функцией n^2, $c=1$, $\rho_1=5$, $\rho_2=10$. Результат первого этапа представлен в табл. 3.3.

На первом шаге второго этапа формируем исходные состояния дерева D^* и списка k (рис. 3.7,*а*). Согласно табл. 3.3 исходная задача разбивается на 100 подзадач размерности 10. Соответствующие D^* и R показаны на рис. 3.7,*б*. Так как $K(10)=0$, то на последующих шагах задачи $z_1, \ldots, z_{100}$ из R удаляются, а задача z' размерности 100 заменяется задачей z'_0 размерности 17 и семнадцатью задачами $z'_1, \ldots, z'_{17}$, две из которых имеют размерность 5, а остальные 6 (рис.

Таблица 3.3. *Параметры подзадач при* $t(A1, n)=n\ln(n)$, $t(A2, n)=n^2$, $c=1$, $\rho_1=5$, $\rho_2=10$

n	$K(n)$	$K'(n)$	$T(n)$	n	$K(n)$	$K'(n)$	$T(n)$
2	0	0	0,4000E 01*	85	8	15	0,1347E 04
5	0	0	0,2500E 02	90	9	15	0,1386E 04
10	0	0	0,1000E 03	95	9	16	0,1519E 04
15	0	0	0,2250E 03	100	10	17	0,1561E 04
20	0	0	0,4000E 03	150	15	30	0,2477E 04
25	0	5	0,6250E 03	200	20	40	0,3460E 04
30	0	6	0,9000E 03	250	25	50	0,4111E 04
35	0	7	0,1225E 04	300	30	60	0,4999E 04
40	0	8	0,1600E 04	350	35	70	0,5899E 04
45	0	9	0,2025E 04	400	40	80	0,6808E 04
50	5	10	0,7206E 03	450	45	90	0,7726E 04
55	5	11	0,8504E 03	500	50	100	0,8653E 04
60	6	12	0,8817E 03	600	60		0,1053E 05
65	6	13	0,1012E 04	700	70		0,1243E 05
70	7	14	0,1046E 04	800	80		0,1435E 05
75	7	15	0,1178E 04	900	90		0,1629E 05
80	8	15	0,1215E 04	1000	100		0,1825E 05

* В вычислительной математике запись 0,4000E 01 обозначает $0{,}4\cdot 10^1$.

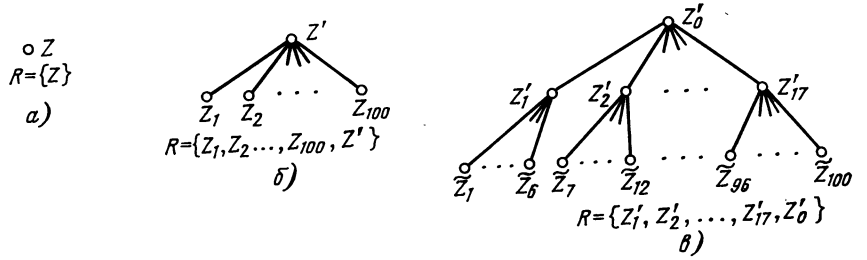


Рис. 3.7. Пример формирования оптимального дерева

3.7,*в*). При этом каждая задача $\tilde{z}_i$ ($i=1,100$) является одной из задач z_j ($j=\overline{1,100}$) в соответствии с результатом применения А1 к z'.

Выше было показано, что применение иерархического подхода позволяет значительно расширить размерность класса решаемых задач. Но его реализация на ЭВМ требует разработки дополнительного математического обеспечения.

3.4. МЕТОДИКА РЕШЕНИЯ ЗАДАЧ БОЛЬШОЙ РАЗМЕРНОСТИ

Исходные данные для задачи большой размерности всегда имеют *иерархическую структуру*, что отражает характерное свойство сложных объектов. Такая структура позволяет вводить описание объекта в ЭВМ и обрабатывать его по частям. В дальнейшем будем считать, что входной иерархической структуре соответствует дерево $D_{вх}$.

В частном случае иногда для решения конструкторских задач можно в качестве дерева D использовать дерево $D_{вх}$. При этом обеспечивается однозначное соответствие между функциональными частями объекта (вершинами $D_{вх}$) и конструктивными узлами, т. е. сохраняется конструктивная неразрывность функциональных частей. В то же время существуют ограничения, препятствующие такому использованию. Прежде всего эти ограничения вызваны существующей *иерархической структурой конструкций*, применяемых для реализации сложных объектов. Ей соответствует дерево $D_к$. Особенность $D_к$ состоит в том, что оно имеет фиксированное число k уровней, которое не зависит от проектируемого объекта, а число сыновей каждой вершины ограничено. Понятно, что число уровней в $D_{вх}$ может отличаться от этого же параметра для дерева $D_к$. Также могут сильно отличаться мощности вершин в деревьях $D_{вх}$ и $D_к$ (в случае, когда мощность вершины $v \in D_{вх}$ больше мощности вершины $w \in D_к$, функциональная часть v просто не помещается в конструктивном узле w; если же мощность v меньше мощности w, плотность монтажа в узле w оказывается низкой). Кроме того, задание однозначного соответствия между функциональной частью и элементом конструкции может привести к созданию неблагоприятных условий для физической реализации всего объекта. Последнюю ситуацию проиллюстрируем на примере линейчатого кристалла. Если некоторая функциональная часть будет полностью занимать одну линейку, то это приведет к перегрузке соседних каналов трассировки, увеличению числа транзитных трасс и удлинению связей между элементами (известно, что кратчайшие связи образуются, если элементы располагаются в области, близкой к кругу).

Все перечисленные причины показывают невозможность в общем случае использования дерева $D_{вх}$. Поэтому необходимо от исходного дерева $D_{вх}$ переходить к новому дереву D, согласованному с решаемой задачей. При этом следует учитывать требование $D^0 \subset$

$\subset D_к$, где D^0 — подграф D, содержащий все вершины верхних k уровней.

В предыдущем параграфе была рассмотрена методика построения оптимального дерева D^*, удовлетворяющего ограничениям на степень вершин при одноуровневом описании входных данных. Однако переход от исходного иерархического описания объекта к одноуровневому не является решением проблемы, так как, если одноуровневое описание превышает некоторый объем (например, объем оперативной памяти), применение алгоритма разбиения оказывается неэффективным.

Приемлемым способом в данном случае является способ локальных преобразований дерева $D_{вх}$ до получения D с заданными свойствами: 1) $D^0 \subset D_к$; 2) число вершин дерева D минимально (что соответствует минимуму числа конструктивных узлов реализации объекта). Рассмотрим одну из возможных методик, основанную на стратегии «сверху вниз» преобразования $D_{вх}$ в D.

Для преобразования структуры деревьев введем *операцию подстановки*, смысл которой поясним на примере. На рис. 3.8,*а* показан фрагмент дерева, соответствующего некоторой структуре объекта. В результате выполнения операции подстановки вершины (части объекта) v_2 сыновья v_2 становятся сыновьями вершины v (рис. 3.8,*б*). Преобразование структуры части объекта, соответствующей вершине v, схематично представлено на рис. 3.8,*в*, *г*.

Процесс преобразования начинается с рассмотрения корневой вершины $v=s$ дерева $D=D_{вх}$. Вершине v ставится в соответствие вершина $d \in D_к$, которая определяет тип конструкции для реализации всей схемы (например, панель, печатная плата, микросхема). Тип конструкции либо задан на входе, либо формально определяется с учетом параметров проектируемого объекта и характеристик конструкций.

Преобразование дерева D в вершине v состоит в переходе к новому дереву D', у которого вершина v имеет минимальное чис-

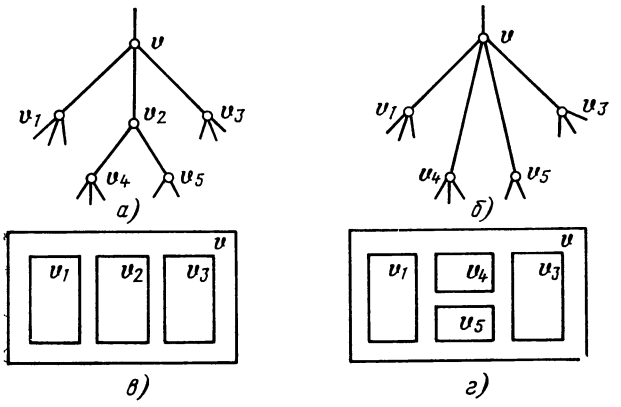


Рис. 3.8. Применение операции подстановки

ло сыновей, а мощность $|v_i|$ каждого сына v_i не превышает соответствующего параметра $|d_i|$ для сына d_i вершины d. При этом под мощностью понимается обобщенный параметр, который включает все факторы, влияющие на реализуемость. Данное преобразование выполняется в результате применения операции А3 подстановки до тех пор, пока не будет выполнено условие $|v_i| \leqslant |d_i|$ для каждого сына v и последующего применения алгоритма А1. Стратегия применения А3 и А1 определяется конкретными особенностями решаемой задачи. Учитывая эвристический характер процесса преобразования дерева, искомое решение можно и не найти. В этом случае необходимо перейти к предыдущему шагу и повторить его при скорректированном значении мощности. Полученное дерево $D=D'$ корректируется в вершине v', удовлетворяющей условиям: вершина v' еще не преобразовывалась; отец вершины v' уже преобразован; вершина v' лежит на одном из пер-

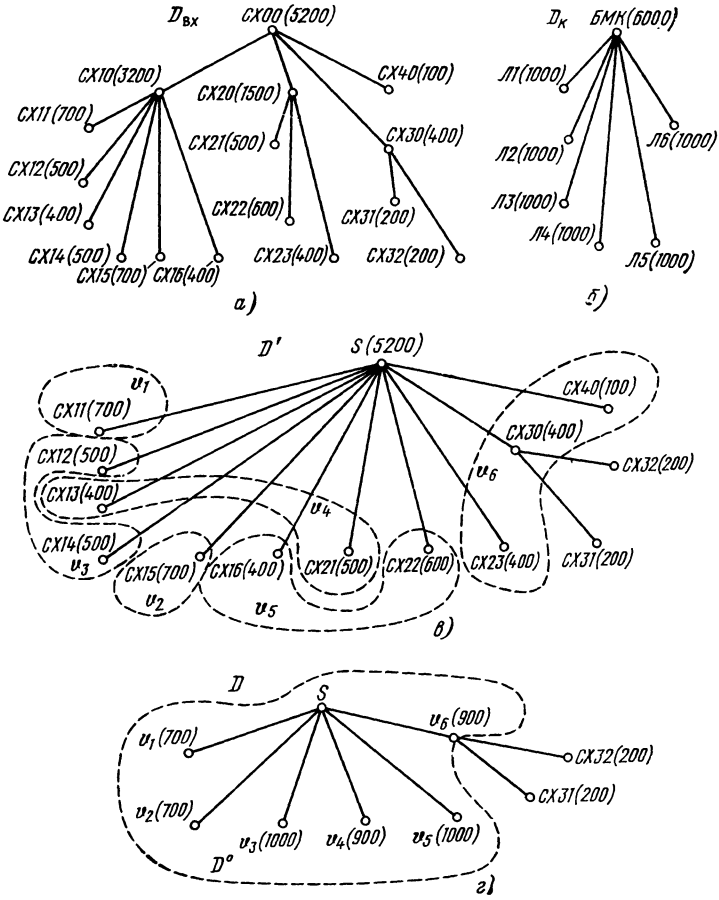


Рис. 3.9. Пример преобразования иерархического описания объекта проектирования

вых k уровней. Если такой вершины нет, то процесс построения завершается.

Процесс преобразования $D_{вх}$ проиллюстрируем упрощенным примером проектирования БИС на БМК. Пусть задано входное описание схемы, содержащей 5200 элементов, каждый из которых занимает одну ячейку. Структура входного описания отображается деревом $D_{вх}$ на рис. 3.9,*а*, где рядом с вершинами указаны имена соответствующих подсхем, а в скобках приведено число элементов в подсхемах. Требуется реализовать схему на базовом кристалле линейчатого типа, структура которого показана с помощью дерева $D_к$ (рис. 3.9,*б*). Кристалл содержит шесть линеек, каждая линейка 1000 ячеек.

После выполнения необходимого числа подстановок структура входного описания преобразуется к виду, показанному на рис. 3.9,*в*, где с помощью штриховой линии отображены результаты применения алгоритма А1. Окончательный вид дерева D, соответствующего распределению частей схемы по линейкам кристалла, приведен на рис. 3.9,*г*. Согласно полученному дереву D необходимо решать одну задачу размерности 6 (размещение линеек) и 6 задач $v_1, \ldots, v_6$ (размещение элементов внутри линеек) большой размерности. Для оптимизации времени решения задач $v_1, \ldots, v_6$ можно воспользоваться методикой, описанной в предыдущем параграфе. Особенность задачи v_6 состоит в том, что ее описание имеет иерархическую структуру. В данном случае оптимизация времени счета сводится к поиску дерева D^*, удовлетворяющего условию: суммарное время преобразования исходного дерева в D^* и применения А2 минимально. Решение сформулированной задачи в общем случае не известно. Поэтому можно воспользоваться деревом $\bar{D}^*$, доставляющим минимум времени работы А2. В предположении, что время преобразования деревьев составляет незначительную часть от времени работы А2, структуры $\bar{D}^*$ и D^* мало отличаются. Тогда процесс решения задачи состоит из следующих этапов. По методике, описанной в § 3.3, при условии $t(A1)=0$, строим дерево $\bar{D}^*$. Затем, применяя операции А3 и А1, преобразуем исходное дерево в $\bar{D}^*$ и с помощью А2 решаем выделенные задачи. Ясно, что последние преобразования применимы и в общем случае и позволяют снижать размерность задачи на любом уровне иерархического дерева.

4. МОДЕЛИРОВАНИЕ БОЛЬШИХ И СВЕРХБОЛЬШИХ ИНТЕГРАЛЬНЫХ МИКРОСХЕМ

4.1. ПРИМЕНЕНИЕ МОДЕЛИРОВАНИЯ НА РАЗЛИЧНЫХ ЭТАПАХ ПРОЕКТИРОВАНИЯ

С наступлением эры сверхбольших интегральных схем (СБИС) возрастает значение моделирования в системах автоматизированного проектирования. Особенность САПР матричных СБИС состоит в широком использовании методов *макромоделирования* и *декомпозиции*, поскольку иерархическая структура объекта проектирования содержит большое число подобных фрагментов. Причем даже на нижнем уровне находятся логические элементы, которые представляют собой довольно большие груп-

пы соединенных приборов. Важно, что электрические параметры и характеристики логических элементов определяются только при их разработке и многократно используются в ходе моделирования широкого спектра СБИС [2, 6, 105—107].

Для успешного проектирования СБИС программы САПР должны охватывать моделирование приборов, схем и логики, размещение и трассировку, верификацию и генерацию тестов. При реализации схем на основе БМК возможно появление новых задач, например перевода схем из исходного логического базиса в базис, определяемый используемым типом кристалла. Возможность эффективного решения данной задачи в автоматическом режиме иллюстрирует система LORES-2 [108], с помощью которой схемы, реализованные на печатных платах, были преобразованы в матричные БИС.

В настоящее время САПР БИС являются комплексными и обеспечивают проектирование от приборного до системного уровня [100]. Преимущество проектирования матричных схем проявляется в том, что моделирование приборов применяется лишь на этапах разработки конструкции кристалла и схем логических элементов. Можно отметить следующие особенности САПР СБИС: расширение постановки задачи моделирования, вплоть до системного уровня; увеличение объема проектных данных, относящихся к одному кристаллу; требование общей оптимизации решения с учетом факторов логического и топологического проектирования; обеспечение адекватности моделирования в условиях непрерывного развития технологии изготовления СБИС.

Для решения указаных задач необходим, как отмечалось, переход к иерархическому проектированию. Схема типового процесса проектирования СБИС приведена на рис. 4.1.

После завершения системного проектирования СБИС устанавливаются ее структура и функциональные спецификации для каждого структурного блока. Затем выполняется логическое проектирование каждого блока и полученные результаты заносятся в базу данных проектирования СБИС. Одновременно могут разрабатываться стандартные фрагменты схем с применением схемного моделирования, которые заносятся в библиотеку фрагментов в базе данных.

После детализации логической схемы проводится верификация (проверка) логики программами логического и схемного моделирования. Затем выполняются процедуры автоматического синтеза топологии. Полученные при топологическом проектировании варианты схемы после уточнения паразитных параметров оцениваются с помощью программ схемного и логического моделирования. Если вычисленные задержки сигналов превышают заданное значение, то изменяются топологические параметры и повторно выполняются процедуры топологического проектирования. Цикл повторяется до тех пор, пока не будут выполнены заданные требования. Наконец, полученные данные преобразуются в формат, необходимый для изготовления фотошаблонов. Кроме того, на осно-

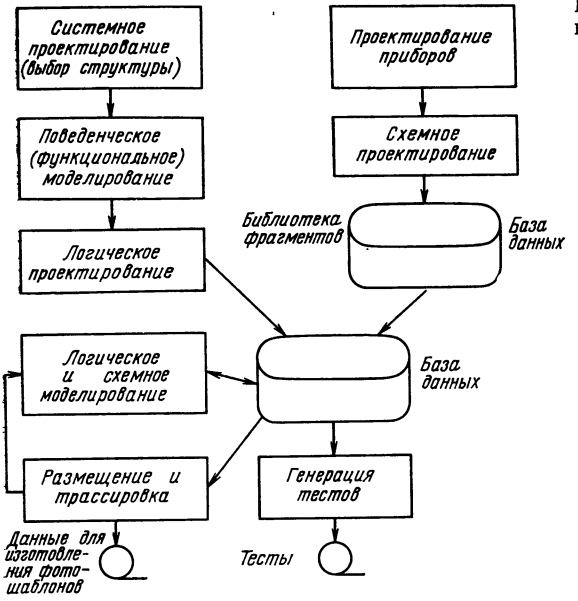


Рис. 4.1. Схема процесса проектирования СБИС

ве данных логического проектирования генерируются тестовые наборы.

На рис. 4.1 отображен процесс преимущественно нисходящего проектирования СБИС, для которого характерно последовательное разбиение больших узлов на меньшие с учетом технических ограничений. Однако на этапах обратного перехода от топологии (после размещения и трассировки) к электрической и логической схемам используется восходящее моделирование. При этом начинают с отдельных элементов и, группируя их нужным образом, формируют фрагменты, затем функциональные узлы и т. д.

Другими словами, при нисходящем проектировании функциональное поведение блока A задается с помощью соответствующего набора уравнений вход — выход, когда A считается черным ящиком. Производя нужное преобразование этого описания, соответствующая программа САПР или сам разработчик дает детальное описание, представляющее схему из функционально более простых элементов (блоков). Корректность выполненного преобразования проверяется моделированием.

При восходящем моделировании используется обратная операция. Детализированная структура A с составляющими ее блоками и соединительными цепями заменяется одним блоком A. Корректность преобразования определяется эквивалентностью поведения блока A, представленного в виде черного ящика, и структуры A.

При иерархическом проектировании СБИС используются различные уровни описаний рассматриваемого объекта:

алгоритмический уровень, на котором основное внимание уделяется поведенческому (функциональному) описанию СБИС;

структурный уровень, когда СБИС представляется в виде совокупности основных блоков и соединений между ними;

уровень системы команд, при котором определяются команды и правила их выполнения аппаратурой;

уровень межрегистровых передач, когда основными элементами системы служат регистры, а описание содержит правила передачи данных между ними;

уровень логических вентилей, на котором СБИС представляется совокупностью вентилей и триггеров, а также соединений между ними;

схемный уровень, когда элементами СБИС являются транзисторы, диоды, резисторы и т. д., а описание содержит сведения об их соединении;

уровень фотошаблонов, на котором элементы схемного уровня преобразуются в детальную топологию в том виде, в каком они будут реализованы в кристалле микросхемы.

Каждое описание функционально эквивалентно начальному, заданному алгоритмом работы СБИС (техническим заданием).

4.2. СИСТЕМНОЕ МОДЕЛИРОВАНИЕ СБИС

Как следует из рис. 4.1, разработчик СБИС, исходя из требований к характеристикам системы, продумывает ее структуру, которая на первом этапе формализации описывается на поведенческом уровне. Для этого используются специальные языки описания структур типа SDL (Стэнфордский университет), HSL (фирма Nippon Telegraph and Telephone), ISP (Университет Карнеги — Меллона), DDL (НАСА) и другие, которые можно использовать для имитации функционирования и представления структуры разрабатываемых систем с различной степенью детализации. *Поведенческая оптимизация*, проводимая с помощью статического анализа соответствующего описания системы, включает устранение дублирующих друг друга элементов структуры, проверку согласованности информационных потоков, преобразование алгоритма работы с целью развития его модульности и др. Ясно, что на выбор начальных структурных решений могут влиять сведения об имеющихся основных схемных компонентах или фрагментах.

Для цифровых СБИС удобно разделять проектируемую систему на *операционную часть* (устройство преобразования данных) и *управляющую часть* (устройство управления). Анализ поведенческого описания нужен для выделения операционной части (соединенных между собой элементов системы) и управляющей информации, т. е. сигналов, с помощью которых обрабатывается информация.

Обработка (преобразование) информации может производиться либо централизованным, либо распределенным способом. При централизованном способе все операции осуществляются единым процессором, данные для которого поступают по соответствующим информационным шинам. В распределенных же системах обработ-

ка операндов выполняется в тех частях системы, где они вырабатываются. На выбор структуры СБИС значительное влияние оказывают наличие модулей (стандартных ячеек, фрагментов и др.), уровень их представления, технические ограничения (занимаемая площадь, быстродействие, потребляемая мощность), а также требуемый алгоритм работы системы.

Из имеющихся в настоящее время средств САПР, обеспечивающих этап системного проектирования СБИС, можно отметить следующие системы, условно разделенные на пять групп:

1. Системы, преобразующие описание системы с уровня межрегистровых пересылок на логический уровень (ALERT [109], CADAT [110] и др.).

2. Системы с глобальным исследованием пространства проектных решений (ПРОЕКТ [111], УМК [112], MIMOLA [113] и др.).

3. Система IBM с локальным исследованием пространства проектных решений [114].

4. Синтезаторы логических матриц ПЛМ и ЗЛМ (APLAS [115], ABLE [116] и др.).

5. Кремниевые компиляторы БИС на основе базовой топологии и фиксированного плана кристалла [117].

Рассмотрим возможности и особенности кремниевого компилятора БИС, разработанного в Калифорнийском технологическом институте. Основным компонентом системы является ячейка (блок), которая может содержать простейшие геометрические элементы топологии и ссылки на другие ячейки. Простейшие ячейки разрабатываются вручную и представляют собой как и блоки, имеющиеся в библиотеках стандартных ячеек, обобщенные процедуры, соответствующие функциональным узлам системы (стекам, циклическим сдвигателям вместо привычных вентилей булевой алгебры). Поэтому для получения весьма подробной топологии кристалла от разработчика требуется сравнительно небольшой объем исходных данных. Каждая выбранная ячейка является программой, которая описывает топологию ячейки, трансформирует ее, вычисляет потребляемую мощность и т. д. Внешние выводы блоков являются контактными площадками ячеек и служат для подключения межблочных соединений.

На рис. 4.2 показана физическая и логическая структуры проектируемого кристалла. Элементы ядра представлены ячейками совместно с их межсоединениями. Декодер служит для выработки необходимых сигналов управления при использования микропрограммного устройства управления. Информация, заданная пользователем, включает форматы микроинструкций и длину слова, разрядность данных и спецификации шин, расположенных на кристалле, а также перечень элементов, содержащихся в ядре, совместно со значениями соответствующих параметров. Кремниевый компилятор синтезирует топологию ядра, состоящего из иерархически организованного набора ячеек одинаковой ширины, а также необходимых информационных шин. Затем компилятор генерирует управляющую часть БИС, в результате чего получается оптимизированная схема декодирования сигналов управления. Кроме того, на кристалле размещаются бу-

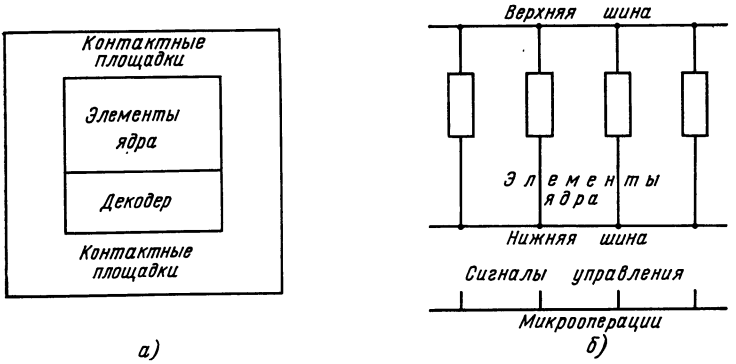


Рис. 4.2. Физическая (*а*) и логическая (*б*) структуры СБИС

феры, предназначенные для передачи сигналов к ячейкам. Затем производится минимизация длины связей между контактными площадками и выводами ячеек, расположенными по периферии кристалла. Площади кристаллов, получаемых с помощью рассматриваемой системы и на основе ручных методов проектирования, различаются на ±10%.

4.3. ЛОГИЧЕСКОЕ МОДЕЛИРОВАНИЕ СБИС

Из современных средств логического проектирования наибольшее распространение получили программы логического моделирования, языки проектирования аппаратуры и средства разработки микропрограмм. Это связано с тем, что из-за несовершенства методов прямого синтеза логической схемы по системе булевых функций в процессе логического проектирования существенную роль играет человек, проектные решения которого проверяются программами логического моделирования, позволяющими убедиться в правильном выполнении схемой логических функций до изготовления кристалла (моделирование) и после (тестирование).

На этапе логического проектирования наиболее широкое применение находят программы моделирования на уровне логических вентилей, реализующие событийный алгоритм продвижения состояний в схеме и предусматривающие *трехзначное моделирование* (1, 0, *X*) с учетом единичных, максимальных, минимальных задержек, задержек включения и выключения вентилей и задержек в соединениях [118].

С целью повышения точности логического моделирования переходят к четырехзначному [119] и семизначному [120] логическому моделированию. Среди программ моделирования рассматриваемого класса следует назвать программы TEGAS-V (США), LOGOS (Япония) [121], КОМОЛ-КФ (СССР) [118] и др. Широко используются при логическом проектировании также программы моделирования на функциональном уровне, или уровне межрегистровых передач. Они позволяют обрабатывать логические

схемы, которые на порядок больше, чем допускают программы моделирования на уровне логических вентилей. При этом затрачиваемое время в два раза меньше. Однако программы моделирования на уровне логических вентилей обеспечивают более точный учет времени задержки. В последнее время появились программы *смешанного моделирования*, в которых сочетаются возможности как функционального моделирования, так и моделирования на уровне логических вентилей. К таким программам относятся РАПИРА-2, МОДИС (СССР), MIXS (Япония) [122] и др. Учитывая распространение *программируемых логических матриц* (ПЛМ) при разработке заказных СБИС, ведутся работы по созданию САПР, которые вначале осуществляют минимизацию булевых функций, а затем автоматически отображают полученную форму на БИС ПЛМ, выполняя при этом и верификацию логики [123].

Тестирование и диагностика СБИС заключается в использовании моделей одиночных постоянных неисправностей, генерации тестовых наборов и в применении логического моделирования для определения всех неисправностей, которые можно обнаружить с помощью этих наборов. Такая процедура выполняется многократно до тех пор, пока не будет достигнут приемлемый процент обнаружения всех возможных неисправностей. Большинство средств тестирования рассчитано на комбинационные схемы, и лишь в редких случаях можно проверить последовательностные схемы. Имеется возможность преобразования последовательностных схем в комбинационные путем потактового представления состояний схем, что обеспечивает простую генерацию тестов для сложных последовательностных схем.

Полная автоматизация генерации тестов реализуется довольно редко, в большинстве случаев используются комбинации ручных и автоматических методов, реализующих *D*-алгоритм Рота, метод активизации пути и др. [124]. С ростом сложности схемного кристалла трудоемкость тестирования увеличивается экспоненциально. Сохранить трудоемкость на приемлемом уровне помогают модульность и иерархичность структуры СБИС. Главные методы обеспечения тестируемости схем — это фрагментное проектирование, когда иерархически используется алгоритм активизации блоков; метод сканируемого пути, использующий принцип последовательного сдвигового регистра для преобразования последовательностных схем в комбинационные; встроенное самотестирование.

4.4. СХЕМНОЕ МОДЕЛИРОВАНИЕ СБИС

Программы анализа электрических схем осуществляют моделирование, при котором учитывается форма электрических сигналов и нелинейность электронных схем. Необходимость такого моделирования возникает при проектировании стандартных ячеек (фрагментов) и на восходящей ветви процедур проектирования СБИС (рис. 4.1), когда в результате полного анализа переходно-

го процесса вычисляются значения задержек сигнала и уровни мощности рассеивания. Так как моделирование СБИС невозможно на приборном уровне, то задача решается методами диакоптики с использованием макромодельного (упрощенного) описания отдельных фрагментов и блоков СБИС.

Проблема размерности при моделировании больших систем, описываемых сотнями алгебро-дифференциальных уравнений, успешно решается разделением сложной задачи на ряд более простых, не вызывающих затруднений при решении. При этом различают следующие возможные подходы к решению задачи моделирования.

Модификация процедуры формирования уравнений для получения матрицы уравнений в виде блочно-диагональной (треугольной) с двойным окаймлением

$$\left[\begin{array}{ccc|c} A_1^1 & & & \\ & A_1^2 & & A_3 \\ & & \ddots & \\ & & A_1^n & \\ \hline & A_4 & & A_2 \end{array}\right] \left[\begin{array}{c} X_1^1 \\ X_2^2 \\ \vdots \\ X_1^n \\ \hline X_c \end{array}\right] = b\,, \tag{4.1}$$

где диагональные блоки $A^i{}_1$ соответствуют уравнениям отдельных частей схемы (подсхемам) с переменными $X^i{}_1$, а матрица $A^i{}_3$ учитывает влияние на каждую из подсхем связей X_c, которые, в свою очередь, через матрицу A_4 связаны с переменными подсхем $X^i{}_1$.

Уравнения вида (4.1) в однородном координатном базисе, например, можно получить, если при использовании метода узловых напряжений выделить подмножество узлов n_c, которые служат для объединения подсхем, и пронумеровать последовательно сначала внутренние узлы каждой из подсхем $n^i{}_1$, а затем узлы связи. Аналогичные результаты получают и для других координатных базисов.

Решение системы уравнений (4.1) сводится к предварительному получению и решению уравнений меньшей размерности относительно переменных связи

$$A_c\, X_c = b_c, \tag{4.2}$$

после чего переменные каждой подсхемы $X^i{}_1$ находятся из решения частных уравнений подсхем вида

$$A_1^i\, X_1^i = b_1 - A_3\, X_c. \tag{4.3}$$

Для треугольной матрицы A_1 процесс сводится к процедуре прямого хода решения

$$x_{1j} = b_{1j} - \sum_{k=1}^{j-1} a_{1,jk}\, x_{1k} - \sum_{r=1}^{m} a_{3,jr}\, x_{cr}, \tag{4.4}$$

где $a_{1,jk}$ и $a_{3,jr}$ — элементы матриц A_1 и A_3 из уравнения (4.1).

Для формирования уравнения связи (4.2) общая матрица A уравнения (4.1) записывается в виде

$$A = \hat{A} + CD - D^t D, \tag{4.5}$$

где

$$\hat{A} = \begin{array}{|c|c|}\hline A_1 & \\ \hline A_4 & I \\ \hline \end{array}; \quad C = \begin{array}{|c|}\hline A_3 \\ \hline A_2 \\ \hline \end{array}; \quad D^t = \begin{array}{|c|}\hline 0 \\ \hline I \\ \hline \end{array} \tag{4.6}$$

Далее $(m+1)$ раз (m — размерность вектора переменных связи X_c) решается система уравнений с треугольной матрицей

$$\hat{A} X^j = B^j, \; j = 0, 1, \ldots, m, \tag{4.7}$$

где $(B^j)_{j=0} = B^0 = b$ и $(B^j)_{j=1, \ldots, m} = C^j$, C^j — j-й столбец матрицы C.

Столбцы матриц A_c и b_c в уравнении (4.2) набираются процедурой

$$\left.\begin{aligned} A_c = [DX^1, DX^2, \ldots, DX^m] = [X_c^{(1)}, X_c^{(2)}, \ldots, X_c^{(m)}], \\ b_c = DX_0 = X_c^0. \end{aligned}\right\} \tag{4.8}$$

Данный подход реализован в программе CLASSIE, разработанной в Калифорнийском Университете в Беркли применительно к ЭВМ с векторными операциями типа CRAY-1 [125]. Общее время счета на этой ЭВМ при решении рассматриваемой задачи можно оценить приближенно с помощью соотношения

$$T = n_{\text{итр}} [n_{\text{я}i} t_{\text{я}i} + n_{\text{у}i} t_{\text{у}i} + n_{\text{подсх}} (n_{\text{я}i} t_{\text{я}j} + n_{\text{у}j} t_{\text{у}j})], \tag{4.9}$$

где $t_{\text{я}\,i}$, $t_{\text{я}\,j}$ — время переоценки матрицы Якоби для нелинейных параметров транзистора в уравнении связи (индекс i) и отдельных подсхем (индекс j) соответственно; $t_{\text{у}\,i}$, $t_{\text{у}\,j}$ — соответствующие времена решения одного уравнения на каждой итерации; $n_{\text{я}\,i}$, $n_{\text{я}\,j}$ — число приборов (транзисторов); $n_{\text{у}\,i}$, $n_{\text{у}\,j}$ — число уравнений; $n_{\text{итр}}$ — число итераций.

Векторный характер вычислений используемой ЭВМ, допускающий одновременную обработку по одной и той же программе нескольких массивов данных, позволяет организовать параллельное вычисление $t_{\text{я}}$ ($t_{\text{я}\,i}$ и $t_{\text{я}\,j}$), для чего транзисторы объединяются в группы по идентичным моделям или по принадлежности к отдельным подсхемам. В последнем случае распараллеливаются не только операции вычисления значений частных производных (элементов матрицы Якоби), но и операции заполнения ими матриц уравнений отдельных подсхем. Уменьшение времени решения систем линеаризованных уравнений ($t_{\text{у}\,i}$, $t_{\text{у}\,j}$) достигается формированием кодов решения для программы решения без циклов, учитывающей конкретную ненулевую структуру матриц решаемой системы уравнений.

Таблица 4.1. *Характеристики тестовых схем*

Вид схемы	Тип транзисторов	Число транзисторов	Число уравнений модели	Разреженность матриц, %
4-разрядный сумматор	БП	288	450	99,3
16-разрядный сумматор	БП	1152	1747	99,65
Фильтр на переключающихся емкостях	МОП	756	410	97,6
Цифровой фильтр	МОП	268	683	99

Экспериментальные исследования программы CLASSIE в сравнении с известной программой схемотехнического проектирования SPICE-2 [126], широко применяемой фирмами США, Европы и Японии, показали, что задачи анализа электронных схем, которые на ЭВМ типа VAX 11/780 занимают время, исчисляемое днями, на ЭВМ CRAY-1 решаются за несколько минут. В табл. 4.1 приведены полученные с помощью программы CLASSIE характеристики тестовых схем, построенных на биполярных (БП) и полевых (МОП) транзисторах [125] (см. рис. 4.3). Следует отметить, что по функциональным возможностям программа SPICE-2 соответствует отечественной программе СПАРС или ПРАМ-01 [118]. В табл. 4.2 даны временные характеристики решения для этой схемы (4-разрядного сумматора с 288 транзисторами), полученные с помощью программ SPICE-2 и CLASSIE, реализованных на одной и той же ЭВМ CRAY-1. Из приведенных данных следует, что скорость счета в программе CLASSIE выше примерно в 4 раза. К тому же, и это главное, программа CLASSIE позво-

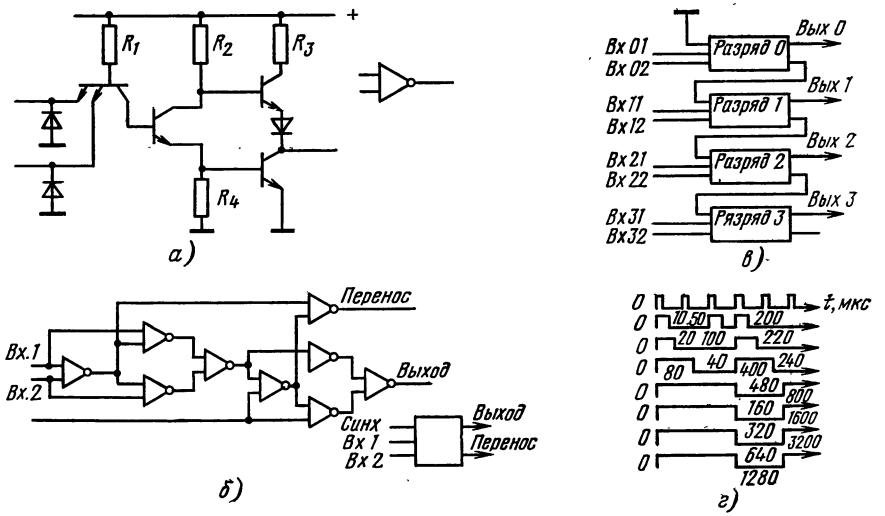


Рис. 4.3. Логический ТТЛ-элемент НЕ—И (*а*) и подсхема в виде одноразрядного сумматора (*б*); 4-разрядный сумматор (*в*) и входные сигналы при анализе переходного режима (*г*)

Таблица 4.2. *Сравнительные характеристики анализа первой тестовой схемы*

Тип программы	Параметры			
	$t_я$, мкс	$t_у$, мкс	$n_{итр}$	Скорость счета
SPICE-2	46	47	8648	1
CLASSIE	0/20*	5,4/1,2*	36219	4,2

* В числителе приведены данные, относящиеся к уравнениям связи, в знаменателе — к уравнениям подсхем. В частности, в уравнения связи параметры транзисторов не вошли, поэтому $t_{яi} \approx 0$.

лила анализировать схемы, содержащие более 1000 транзисторов на приборном уровне описания.

Решение уравнений отдельных подсхем при расчленении сложной схемы на подсхемы и учет их взаимного влияния проводятся с использованием метода эквивалентных источников. Например, если k подсхем объединены внешними узлами, то для отдельной подсхемы справедлива система уравнений

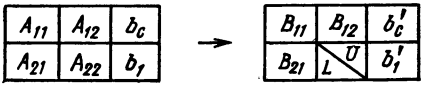
$$\begin{bmatrix} A_{11}^k & A_{12}^k \\ A_{21}^k & A_{22}^k \end{bmatrix} \begin{bmatrix} X_c^k \\ X_1^k \end{bmatrix} = \begin{bmatrix} b_c^k \\ b_1^k \end{bmatrix} + \begin{bmatrix} i^k \\ 0 \end{bmatrix}. \qquad (4.10)$$

Здесь $X^k{}_c$ — вектор узловых напряжений внешних узлов k-й подсхемы; $X^k{}_1$ — вектор внутренних переменных подсхемы, включая узловые напряжения на внутренних узлах и дополнительные токовые переменные для расширенного узлового базиса; i^k — вектор неизвестных дополнительных токовых переменных, характеризующих взаимодействие k-й подсхемы с остальной частью схемы.

Исключая вектор $X^k{}_1$ из уравнения (4.10), получаем

$$[A_{11}^k - A_{12}^k (A_{22}^k)^{-1} A_{21}^k] X_c^k = [b_c^k - A_{12}^k (A_{22}^k)^{-1} b_1^k] + i_k. \qquad (4.11)$$

Несмотря на то, что вектор i^k неизвестен, указанное исключение вектора $X^k{}_1$ легко выполняется численным методом, например с помощью метода Гаусса или LU-разложения. Действительно, если $A_{22} = LU$, то $A^{-1}{}_{22} = U^{-1}L^{-1}$, где U^{-1} и L^{-1} находятся из решения систем уравнений с соответствующей треугольной матрицей U или L и с несколькими правыми частями, соответствующими столбцам единичной матрицы 1. В целом исходная информация о модели подсхемы преобразуется к новому виду



и размещается в прежних ячейках памяти. При этом

$$B_{11} = A_{11} - A_{12} U^{-1} L^{-1} A_{21} = A_{11} - B_{12} B_{21};$$

$$B_{12} = A_{12} U^{-1};\ B_{21} = L^{-1} A_{21};\ b_c' = U^{-1} b_c,\ b_1' = L^{-1} b_1.$$

Следует отметить, что вектор X^k_c в уравнении (4.10) является составным компонентом вектора X_0, соответствующего узловым напряжениям всей схемы, рассматриваемой в качестве подсхемы нулевого уровня. Суммируя уравнения всех подсхем, получаем

$$[\sum_k B^k_{11}]\, X_0 = \sum_k [\, b^k_c - B^k_{12} b'^k_1], \qquad (4.12)$$

так как $\sum_k i^k = 0$ (в силу первого закона Кирхгофа). Решая уравнение (4.12), являющееся, по существу, уравнением для переменных связи, находим вектор X_0, а значит, и векторы X^k_c каждой подсхемы. Далее, пользуясь вторым уравнением системы (4.10), вычисляем для каждой подсхемы вектор внутренних переменных X^k_1, при этом

$$X^k_1 = (A^k_{22})^{-1} (b^k_1 - A^k_{21}\, X^k_c) \qquad (4.13)$$

$$\text{или } UX^k_1 = b'^k_1 - B^k_{21}\, X^k_c .$$

Поиск текущих значений переменных анализируемой схемы ведется итерационно. Найденные значения X^k_c и X^k_1 позволяют корректировать параметры модели подсхемы и повторять цикл вычислений, пока не будет достигнута сходимость решения. Если после некоторой итерации рассмотренного алгоритма вектор переменных $X^k = [X^k_c X^k_1]^t$ любой подсхемы не изменяет своего предыдущего значения, коэффициенты и константы уравнений (4.11) и (4.12) также не изменяются, то повторный анализ модели схемы на последующей итерации не проводится. Этим учитывается текущая неактивность (latency) отдельных подсхем, обеспечивающая экономичность и эффективность процедуры моделирования схемы в целом.

Рассмотренный подход реализован в программе STATE, разработанной Иллинойским Университетом и являющейся модификацией программы SPICE-2 [127]. Сравнительные исследования программ STATE и SPICE-2 на вычислительном комплексе DEC-10 показали, что при совпадении результатов моделирования (в пределах четырех старших разрядов) программа STATE обеспечивает, в среднем, выигрыш во времени до 50% за счет учета неактивности отдельных подсхем. Сказанное можно проиллюстрировать табл. 4.3, в которой приведены основные характеристики процедуры нахождения переходного процесса в схемах, построенных на основе ТТЛ-элемента НЕ—И, приведенного ранее на рис. 4.3,*а*. Данные первого столбца таблицы (вариант I) относятся к схеме самого ТТЛ-элемента, рассматриваемого как совокупность подсхем в виде отдельных транзисторов. Данные второго и третьего столбцов (варианты II и III) относятся к цепочкам таких элементов, используемым при объединении входов (в виде инверторов), при 5 и 8 элементах соответственно. При этом в качестве отдельных подсхем используются сами ТТЛ-элементы.

Декомпозиция систем алгебраических уравнений процедурой Гаусса — Зайделя. Для систем уравнений очень большой размерности предпочтительней вместо LU-преобразования применять итерационный метод Гаусса — Зайделя, использующий следующую

Таблица 4.3. *Характеристики процесса моделирования переходного процесса для ТТЛ-схем*

Переходной процесс	Вариант I (схема ТТЛ, используемая как инвертор)	Вариант II (цепочка из 5 инверторов)	Вариант III (цепочка из 8 инверторов)
Общее число временных итераций для подсхем	2850	2945	2770
Общее число активных временных итераций для подсхем	1516	2128	1945
Степень использования неактивности подсхем, %	46,81	27,74	29,78
Общее время счета, с	68996	97164	132338
Выигрыш в скорости решения, %	47,86	26,58	34,99

процедуру. В матрице A системы уравнений выделяются диагональная D, нижняя L и верхняя U треугольные составляющие, так что $A=D+L+U$.

Затем на каждой m-й итерации решается треугольная система уравнений относительно каждой переменной x_i, $i=1, 2, ..., n$:

$$x_i^{(m+1)} = D_{ii}^{-1}\left(b_i - L_i \hat{X}^{(m+1)} - U_i X^{(m)}\right), \qquad (4.14)$$

где L_i и U_i — i-е строки матриц L и U, а

$$\hat{X}^{(m+1)} = (x_1^{(m+1)}, \ldots, x_{i-1}^{(m+1)}, x_{i+1}^{(m)}, \ldots, x_n^{(m)})^t.$$

Нетрудно видеть, что уравнение (4.14) содержит только одну неизвестную переменную $x_i^{(m+1)}$, так как переменные $x_1^{(m+1)}$, ..., $x_{i-1}^{(m+1)}$, $x_{i+1}^{(m)}$, ..., $x_n^{(m)}$ уже определены. Этим достигается декомпозиция исходной системы уравнений на отдельные уравнения или их блоки (группы), что существенно повышает эффективность вычислений. В частности, рассмотренный подход реализован в современных программах моделирования СБИС типа DIANA и SPLICE [128, 129].

Число итераций m и, следовательно, сходимость вычислений определяются погрешностью $\varepsilon^{(m+1)} = \|x^{(m+1)} - x^{(m)}\|$, для которой справедливо соотношение $\varepsilon^{(m+1)} \leqslant \|(1+D^{-1}L)^{-1}D^{-1}U\|\varepsilon^{(m)} = \|M\|\varepsilon^{(m)}$, где $\|\ldots\|$ — евклидова норма. Отсюда следует, что итерационный процесс вычислений будет сходиться для любых начальных значений X, если все собственные значения матрицы M по модулю меньше единицы. В общем случае сходимость зависит от численных значений матриц L, D и U. Обычно быстро сходятся несколько первых итераций, а затем процесс схождения замедляется, асимптотически следуя линейному закону.

Скорость сходимости можно повысить, если упорядочением строк и столбцов исходной матрицы A предварительно привести ее к форме, максимально приближающейся к нижней треугольной матрице. Сходимость итерационного процесса можно также улучшить повышением симметричности метода Гаусса — Зайделя, ес-

ли от однонаправленного процесса вычислений перейти к двунаправленному. При этом в каждой паре смежных итераций одна выполняется в обычном «прямом» направлении, использующем нижнюю треугольную матрицу L, а вторая — в «обратном», с использованием верхней треугольной матрицы U. Тогда вместо выражения (4.14) поочередно применяют выражения:

$$x_i^{(m)} = D_{ii}^{-1}\,(b_i - L_i\,\hat{X}^{(m)} - U_i\,\check{X}^{(m-1)}),\ i = 1, 2, \ldots, n; \qquad (4.15)$$

$$x_j^{(m+1)} = D_{jj}^{-1}\,(b_j - L_j\,\check{X}^{(m)} - U_j\,\check{X}^{(m+1)}),\ j = n, n-1, \ldots, 1, \qquad (4.16)$$

где $\hat{X}^{(m)} = (x_1^{(m)}, \ldots, x_{i-1}^{(m)}, x_{i+1}^{(m+1)}, \ldots, x_n^{(m+1)})^t$;

$$\check{X}^{(m+1)} = (x_1^{(m)}, \ldots, x_{j-1}^{(m)}, x_j^{(m+1)}, \ldots, x_n^{(m+1)})^t.$$

Декомпозиция систем дифференциальных уравнений релаксационным методом Ришли. Особого упоминания заслуживает новый релаксационный метод решения уравнений большой размерности, позволяющий подобно методу Гаусса — Зайделя (4.14) осуществить декомпозицию решаемой системы нелинейных дифференциальных уравнений, в результате чего каждое уравнение системы после ее декомпозиции решается индивидуально во всем временно́м интервале от $t=0$ до $t=T$ [130].

Исходная система уравнений

$$\dot{X} = f\,(X, t),\ X\,(0) = X_0 \qquad (4.17)$$

разделяется на отдельные уравнения ($i=1, 2, \ldots, n$)

$$\dot{x}_i = f_i\,(x_1^{(m+1)}, \ldots, x_{i-1}^{(m+1)}, x_i^{(m+1)}, x_{i+1}^{(m)}, \ldots, x_n^{(m)}, t), \qquad (4.18)$$

в которых содержится только по одной неизвестной переменной $x_i^{(m+1)}$, так как остальные переменные к текущей итерации уже найдены.

Благодаря разреженности системы (4.17) для решения каждого из выделенных уравнений обычно требуется лишь небольшое число предварительно вычисленных переменных, значения которых необходимо хранить. Выделенные уравнения (4.18) решаются различными переменными шагами, определяемыми свойствами этих уравнений (жесткие или нежесткие, описывают активные или неактивные логические фрагменты объекта и др.), что способствует существенному повышению эффективности всего процесса решения. Обычно итерации при решении (4.18) продолжаются до тех пор, пока разность полученных значений переменных на двух последующих итерациях не станет меньше заданной, т. е. $\varepsilon^{(m+1)} = \max_i \|x^{(m+1)}(t) - x^{(m)}(t)\| \leqslant \varepsilon_{\text{доп}}$.

Для примера рассмотрим электронную схему динамического сдвигового регистра на МОП-транзисторах, изображенную на рис. 4.4,*а*. В предположении,

что все емкости в схеме линейны, соответствующая система уравнений модели схемы принимает вид

$$\left.\begin{aligned}&(C_1+C_2+C_3)\,\dot{v}_1-i_1(v_1)+i_2(v_1,u_1)+i_3(v_1,u_2,v_2)-\\&-C_1\dot{u}_1-C_3\dot{u}_2=0;\\&(C_4+C_5+C_6)\,v_2+C_6\dot{v}_3-i_3(v_1,u_2,v_2)-C_4\dot{u}_2=0;\\&(C_6+C_7)\,\dot{v}_3-C_6\dot{v}_2-i_4(v_3,v_2)=0,\end{aligned}\right\}\qquad(4.19)$$

где v_i — узловые напряжения ($i=1, 2, 3$); u_j — входные воздействия, $j=1, 2$ (рис. 4.4,*а*).

Применяя выражение (4.18) к уравнениям (4.19), для m-й итерации получим:

$$\left.\begin{aligned}&(C_1+C_2+C_3)\,\dot{v}_1^m-i_1(v_1^{(m)})+i_2(v_1^{(m)},u_1)+i_3(v_1^{(m)},u_2,v_2^{(m-1)})-\\&-C_1\dot{u}_1-C_3\dot{u}_2=0;\\&(C_4+C_5+C_6)\,\dot{v}_2^{(m)}-C_6\dot{v}_3^{(m-1)}-i_3(v_1^{(m)},u_2,v_2^{(m)})-C_4\dot{u}_2=0;\\&(C_6+C_7)\,\dot{v}_3^{(m)}-C_6\dot{v}_2^{(m)}-i_4(v_3^{(m)})+i_5(v_3^{(m)},v_2^{(m)})=0.\end{aligned}\right\}\qquad(4.20)$$

Схемная интерпретация системы уравнений (4.20) показана на рис. 4.4,*б*, из которого следует, что использование рассмотренного релаксационного метода разделяет исходную схему на несколько подсхем. С помощью методов численного интегрирования каждая из выделенных подсхем анализируется для всего временно́го интервала интегрирования перед тем, как перейти к анализу следующей подсхемы.

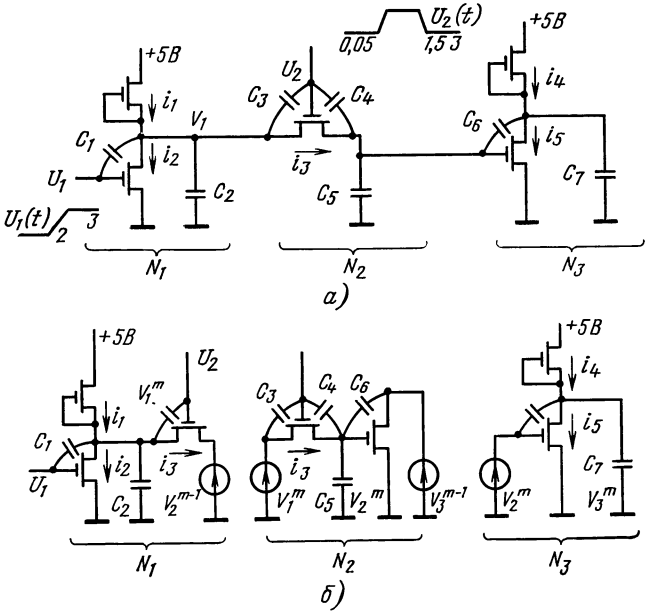


Рис. 4.4. Схема сдвигового регистра (*а*) и ее декомпозиция (*б*) для m-й итерации релаксационного метода

Если предположить, что в исходной схеме рис. 4.4,*а* имеются три подсхемы N_1, N_2 и N_3, то выделенные релаксационным методом подсхемы $\overline{N}_1$, $\overline{N}_2$ и $\overline{N}_3$ (рис. 4.4,*б*), по существу, соответствуют подсхемам N_1, N_2 и N_3 с учетом дополнительных компонентов, имитирующих их взаимные нагрузки. Таким образом, метод подсхем получает новую эффективную математическую формулировку, обеспечивающую его автоматическое применение без участия пользователя в отличие от ранее рассмотренных его вариантов, сводящихся к промежуточному нахождению уравнений переменных связи подсхем.

Рассматриваемый метод успешно реализован в программе RELAX для моделирования цифровых СБИС [130], выполненных по МОП-технологии. Исследования сходимости метода показали, что в этом применении сходимость гарантируется, если в каждом узле схемы имеется емкость, подключенная к общей шине. На рис. 4.5 приведены эпюры узловых напряжений электронной схемы (см. рис. 4.4,*а*), полученные на различных итерациях программой RELAX; для сравнения приведены решения той же задачи с помощью известной программы SPICE-2. Об эффективности релаксационного метода можно судить по следующим цифрам. Для временного анализа МОП-схемы с 131 транзисторами программа SPICE-2 расходует 818 с машинного времени на ЭВМ VAX 11/780, а программа RELAX при той же точности (достигается после 5-й итерации) — только 11,42 с; для МОП-схемы с 263 транзисторами SPICE-2 расходует 1334,8 с машинного времени, а RELAX — только 22,3 с при четырех итерациях (табл. 4.4).

Декомпозиция решения при использовании макромодельного описания (многоуровневый метод Ньютона) [131]. Предположим, что для каждой подсхемы (фрагмента СБИС) имеется символическая макромодель

$$Y = G(U), \tag{4.21}$$

где U — напряжения внешних узлов Y — токи в этих узлах. В процессе моделирования СБИС для каждой подсхемы любым из известных методов составляется уравнение вида

$$F(U, Y, X) = 0, \tag{4.22}$$

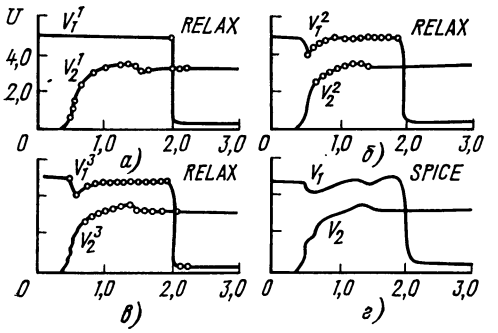


Рис. 4.5. Графики переходного процесса в сдвиговом регистре:
а — после первой итерации; *б* — после второй; *в* — после третьей; *г* — после программы SPICE-2

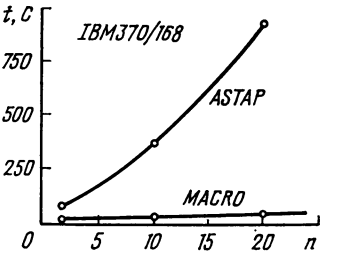


Рис. 4.6. Зависимости времени моделирования от сложности схем для программ ASTAP и MACRO

Таблица 4.4. *Сравнительные результаты работы программ* SPICE-2 *и* RELAX

Характеристика	Число узлов тестовой схемы / Число МОП-транзисторов				
	4 / 6	8 / 21	16 / 42	27 / 131	45 / 263
Время анализа по программе SPICE-2, с	21,3	121,57	211,53	818	1334,8
Время анализа по программе RELAX, с	1,08	4,38	5,85	18,42	22,30
Число итераций по программе RELAX	5	5	7	5	4
Выигрыш в быстродействии ($t_{\text{SPICE-2}}/t_{\text{RELAX}}$)	19,7	27,73	26,18	44,42	59,86

где X — внутренние переменные. При этом уравнения отдельных подсхем могут быть составлены для различных координатных базисов. Уравнение (4.22) используются на внутреннем цикле итераций и решается по методу Ньютона при постоянном U:

$$D_{XY} F(U, Y, X) [\Delta X, \Delta Y]^t = -F(U, Y, X), \tag{4.23}$$

где D_{XY} — производная по переменным X и Y.

С учетом (4.21) уравнение (4.22) можно переписать в виде

$$H(U, G(U), X) = 0. \tag{4.24}$$

Уравнение (4.24) позволяет организовать внешний итерационный цикл по методу Ньютона, когда решается уравнение

$$[D_U H(U, G, X) + D_G H(U, G, X) D_U G] \Delta U + \\ + D_X H(U, G, X) \Delta X = -H(U, G, X), \tag{4.25}$$

где, как и ранее, в общем случае D_Z — производная по переменной Z.

Рассматриваемый подход имеет следующие особенности: одни и те же символические макромодели $G(U)$ могут описывать множество одинаковых подсхем, что повышает эффективность вычислений; легко проверяется активность подсхем по оценке уравнений (4.22) и (4.24) для F и H; используются индивидуальные методы решения и выбора величины шагов для уравнений различных подсхем.

Общую процедуру моделирования сложной схемы можно представить следующей последовательностью операций:

1. Выбирается временной шаг, и рассчитываются $t_{n+1} = t_n + \Delta t$.
2. Оценивается активность i-й подсхемы по изменению F_i.

Если подсхема не активна, то

$$X_i(t_n) = X_i(t_{n-1}); \; G_i(t_n) = G_i(t_{n-1}), \; D_{G_i}^*(t_n) = D_{G_i}(t_{n-1})$$

и осуществляется переход к шагу 6.

3. Если подсхема активна, то решением уравнения (4.23) обновляется вектор

$$[X_i^{k+1}, Y_i^{k+1}] = [X_i^k, Y_i^k] + (\Delta X, \Delta Y).$$

4. Определяется по уравнению макромодели (4.21) величина

$$G_i(U) = Y_i^{k+1},$$

и вычисляется производная $D_U G_i$.

5. Проверяется индекс подсхем $i < m$, где m — число макромоделей для фрагментов схемы, и, если неравенство справедливо, повторяются шаги 2—5 на внутреннем цикле итераций по методу Ньютона.

6. Решается уравнение (4.25), и обновляется вектор

$$[U_i^{k+1}, X_i^{k+1}] = [U_i^k, X_i^k] + (\Delta U, \Delta X).$$

7. Проверяются погрешности, вычисляется величина нового временного шага Δt и осуществляется переход к шагу 1, пока не выполнится условие $t \geqslant T$, где T — общий интервал времени интегрирования.

Было показано, что рассмотренный вычислительный процесс сходится квадратично, если выполняется условие $\|\Delta X, \Delta Y\| \leqslant \leqslant \|\Delta U, \Delta X\|^2$.

Многоуровневая процедура Ньютона реализована в программе MACRO фирмы IBM [131]. Ее возможности применительно к ЭВМ IBM 370/168 можно проиллюстрировать в сравнении с известной программой ASTAP той же фирмы [132] Так, при моделировании 8-разрядного сумматора с 1600 МОП-элементами, модель которого содержит 3200 уравнений, программа ASTAP расходует часы машинного времени, в то время как программа MACRO решает эту задачу при сравнительно небольшой используемой памяти (600 кбайт) лишь за 2—4,5 мин (в зависимости от входных сигналов). На рис. 4.6 приведены сравнительные графики, показывающие зависимости времени моделирования на ЭВМ IBM370/168 переходных процессов в биполярных цифровых схемах от их сложности (числа TTJI-инверторов, подобных изображенному на рис. 4.3,*а*) для программы ASTAP и MACRO соответственно (n — число инверторов).

Декомпозиция процедуры моделирования событийным алгоритмом [133]. Сложная интегральная схема представляет собой совокупность подсхем, каждая из которых может быть описана на логическом или схемотехническом уровне. При реализации событийного алгоритма, ранее используемого лишь при логическом моделировании, в любой момент времени оценивается состояние подсхем на функциональную активность, организуется список событий и выполняются следующие процедуры:

для момента t^* выделяется множество активных подсхем A и пассивных подсхем D, которые станут активными для $t > t^*$;

экстраполируются выходные сигналы подсхем D на основании предыдущих значений переменных;

формируются уравнения подсхем A (в частности, дискретизуются и алгебраизируются с помощью численных методов);

объединяются и решаются уравнения подсхем A (с помощью методов Ньютона и LU-преобразования);

проверяется состояние подсхем D и некоторые из них переводятся в множество A;

для каждой подсхемы из множества A определяется локальная погрешность вычислений. Если $\varepsilon < \varepsilon_{доп}$, то вычисляется следующий шаг и модифицируется список событий.

Таким образом, применение событийного алгоритма позволяет эффективно использовать принцип неактивности (latency) подсхем и осуществлять декомпозицию схемы в целом, решая в каждый момент времени только уравнения активных подсхем. Такая декомпозиция схемы успешно реализована в программе SAMSON (System for Active-directed Mixed Simulation of Networks) Университета Карнеги—Меллона [132]. В отличие от предыдущих программ SAMSON относится к классу программ гибридного моделирования, таких как DIANA и SPLICE. Программа SAMSON написана на языке ПАСКАЛЬ и опробована на ЭВМ VAX 11/780.

4.5. ПРОЕКТИРОВАНИЕ ПРИБОРОВ И ПРОЦЕССОВ

Любой полупроводниковый прибор характеризуется рядом как электрических (подвижность носителей, время их жизни), так и технологических (геометрические размеры, профиль распределения примеси и т. д.) параметров. Цель моделирования прибора — рассчитать по таким параметрам пространственные и временные зависимости электростатического потенциала и квазипотенциалов Ферми для электронов и дырок. По этим трем величинам определяются, в свою очередь, векторные поля напряженности электрического поля и плотности электрического поля. Интегрированием первого из векторов по контуру между соответствующими контактами прибора и второго — по площадям этих контактов находят внешние электрические характеристики прибора.

Программы моделирования приборов в сочетании с программой моделирования технологических процессов стали обязательным инструментом проектирования приборов. Они также используются для прогноза нестабильности параметров приборов в зависимости от особенностей процесса. Программы моделирования приборов позволяют анализировать разнообразные биполярные и униполярные приборы при использовании итерационного метода Гуммеля для решения линеаризованного уравнения Пауссона и уравнений непрерывности для плотности электрического тока [134].

Сначала решается уравнение Пуассона в предположении известных квазиуровней Ферми. Полученное значение электрического потенциала подставляется в уравнения непрерывности, которые затем решаются. Этот процесс последовательно повторяется до тех пор, пока для всех неизвестных переменных не будут по-

лучены совпадающие с заданной точностью последовательные значения. Основные уравнения прибора в частных производных пространственно дискретизируются с помощью метода конечных разностей или метода конечных элементов, удобного при использовании непрямоугольных сеток. При этом производные или интегралы аппроксимируются такими выражениями, в которые входят значения неизвестных функций только в узлах сетки. Метод конечных разностей основан на локальной аппроксимации дифференциального оператора некоторым разностным оператором, а в методе конечных элементов искомое решение глобально аппроксимируется набором функций формы, которые служат пробными функциями, при этом в процессе аппроксимации применяется вариационный метод Ритца или разностный метод Галеркина.

Для решения получаемой системы линейных алгебраических уравнений применяется релаксационный или прямой метод, соответствующий структуре матрицы. Физические модели и граничные условия выбираются с учетом материала прибора (кремний, арсенид галлия) и с учетом структуры прибора (МОП, биполярный прибор). По известному профилю распределения примесей можно вычислить вольт-амперные характеристики биполярного прибора, емкость перехода и граничную частоту, а также определить все параметры транзисторов, используемые, например, в модели Гуммеля—Пуна без экспериментального определения параметров на тестовых компонентах. С помощью полученной модели транзистора и программы схемного моделирования можно оценить работоспособность проектируемого прибора в различных режимах.

Сравнение результатов двух- и трехмерного моделирования свидетельствует о том, что в области больших токов доминирующую роль играют эффекты трехмерной структуры транзистора. Моделирование биполярных приборов выполняется с помощью программ типа TRANAL [135], GALATE [134], MEDUSA [136]. Аналогично нетрудно вычислить параметры МОП-приборов, вольт-амперную характеристику, характеристики $g_m = f(u_{зат})$ и $C_{зат} = \varphi(u_{зат})$ и использовать их при анализе схем.

В соответствующих программах моделирования типа MINIMOS [137], FEDAS [138], GALENE [134] и других экономия затрат машинного времени достигается униполярной аппроксимацией, применением алгоритма Люка для решения уравнений непрерывности, метода Гуммеля, линеаризующего уравнение Пуассона, и релаксационного метода для решения разностных уравнений. Другие параметры полевых транзисторов (пороговое напряжение, его чувствительность к профилю примесей и др.) рассчитываются программой численно. Для исследования эффектов в n-канальных МОП-приборах с коротким и узким каналами (например, явления лавинного пробоя) применяются программы трехмерного моделирования [140].

В моделировании технологических процессов прослеживаются два направления. Одно из них — одномерное моделирование, обеспечивающее экономию машинного времени. Например, для расчета скорости окисления и коэффициента диффузии в зависимости от глубины обычно используется программа SUPREM

II [108]. Однако при очень малых размерах приборов точность, достигаемая этой программой, весьма ограничена. Процессы литографии, травления и осаждения моделируются с помощью программы SAMPLE [141]. Второе направление — двумерное моделирование процессов с учетом горизонтального растекания тока. Хотя проблемы, связанные с аспектами короткого канала, пока еще не решены, такие программы моделирования пригодны для проектирования субмикрометровых приборов.

В любом случае, если структура проектируемого прибора адекватно отражена в программе моделирования технологического процесса, можно путем оптимизации процесса и структуры прибора обеспечить заданные электрические характеристики приборов. Поэтому целесообразно для общей оптимизации структур приборов и технологических процессов объединять программы моделирования приборов и программы моделирования процессов.

Задача уточнения электрофизических и геометрических параметров моделей полупроводниковых приборов и углубления понимания физических процессов, происходящих в полупроводниковых материалах, не теряет своей актуальности, так как с уменьшением размеров приборов заметное влияние на их характеристики начинают оказывать все новые и новые эффекты. Вместе с тем на практике широкое распространение получили макромодели, построенные с учетом лишь основных физических явлений или внешних характеристик приборов. Причем здесь наблюдается тенденция к дальнейшему упрощению с целью снижения требуемой памяти ЭВМ и сокращения объема вычислений при проектировании СБИС.

5. АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЕ ТОПОЛОГИИ МАТРИЧНЫХ СВЕРХБОЛЬШИХ ИНТЕГРАЛЬНЫХ МИКРОСХЕМ

5.1. ОСОБЕННОСТИ ИСХОДНЫХ ДАННЫХ ДЛЯ ПРОЕКТИРОВАНИЯ МАТРИЧНЫХ СБИС

Вопросы автоматизированного проектирования топологии матричных СБИС будем рассматривать на примере подсистемы АРТИС [142—144], предназначенной для эксплуатации в составе промышленной системы АСП-6М и ориентированной на линейчатую конструкцию базового кристалла (рис. 1.13). Такое рассмотрение позволяет проводить конкретный анализ возникающих проблем и, вместе с тем, не приводит к потере общности результатов, так как большинство задач проектирования СБИС на кристаллах различных типов может быть решено родственными методами. Основной особенностью всех этих задач является большая размерность, определяющая необходимость применения иерархического подхода [145].

Так, из-за большой размерности реальные исходные данные, описывающие логическую схему СБИС, представляют собой совокупность описаний подсхем, которые были разработаны на этапе функционально-логического проектирования. В общем случае подсхема содержит логические элементы, другие подсхемы и периферийные элементы, предназначенные для электрической связи матричных БИС с частью устройства вне кристалла. (Периферийные элементы реализуются на периферийных ячейках БМК.) Таким образом, входное описание СБИС представляет собой совокупность описаний ее подсхем, входящих одна в другую, т. е. имеет иерархическую структуру [146].

Для иллюстрации особенностей иерархического представления схемы рассмотрим небольшой пример. Пусть задана схема СХ000, которая состоит из двух подсхем СХ010, СХ020 и периферийного элемента, соединенных, как показано на рис. 5.1,*а*. В свою очередь, каждая из этих подсхем также включает по две подсхемы (рис. 5.1,*б*, *в*). При этом полная структура схемы имеет вид, представленный на рис. 5.2. В приведенном примере каждой цепи подсхемы ставится в соответствие число — ее номер. Номера выводов элементов также пронумерованы. Для того чтобы воспользоваться иерархическим представлением схемы потребовалось каждой внешней цепи подсхемы поставить в соответствие дополнительное число, которое можно себе представить как номер вывода псевдоразъема подсхемы. Это дополнительное число позволяет задать взаимно однозначное соответствие между цепями внутри подсхемы и цепями, подходящими к подсхеме извне. Например, в подсхеме СХ010 (рис. 5.1,*б*) цепь *3* выходит на 1-й вывод псевдоразъема, а в схеме СХ000 (рис. 5.1,*а*) цепь *2* подходит к этому же выводу. Поэтому цепи *2* схемы СХ000 и *3* подсхемы СХ010 эквипотенциальны. В дальнейшем будем говорить о внешних выводах подсхем (на рис. 5.1 и 5.2 они представлены зачерненными квад-

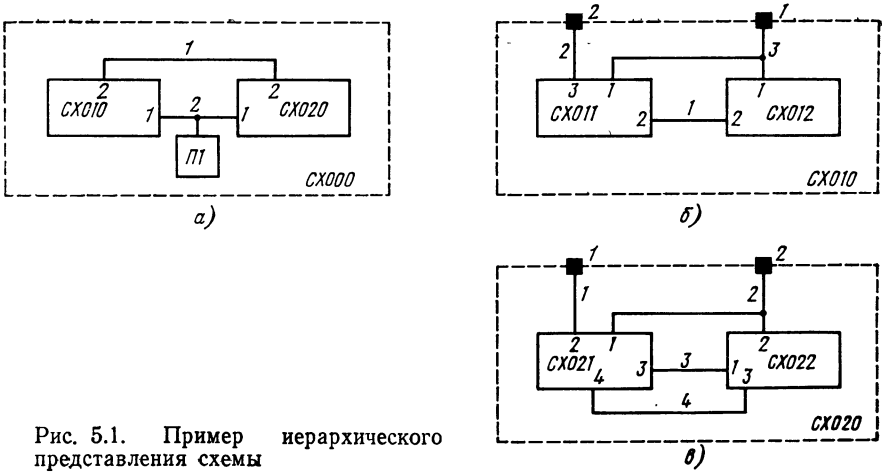


Рис. 5.1. Пример иерархического представления схемы

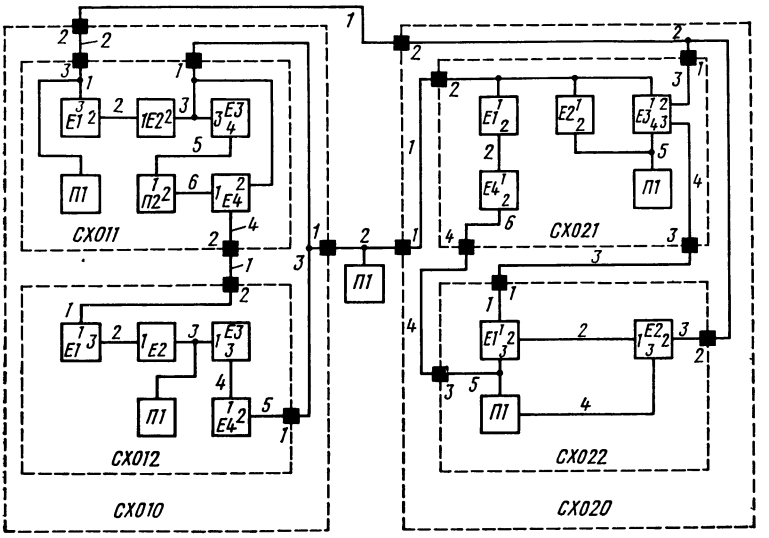


Рис. 5.2. Полная структура схемы

ратами) как о выводах обыкновенных элементов, но помня, что все это чисто абстрактное понятие.

Прежде чем перечислять основные характеристики схемы (подсхемы), введем понятие *компонента*, под которым будем подразумевать элемент или подсхему рассматриваемой схемы. При описании схемы для каждого компонента указываются *имя* и *тип*. Имя компонента в пределах схемы является уникальным (разные компоненты имеют разные имена), и определяется разработчиком. Типы различных компонентов могут совпадать; они определяются функциональным назначением компонентов. Помимо списка компонентов, содержащего имена и типы, описание схемы содержит список цепей, определяющий соединения между компонентами

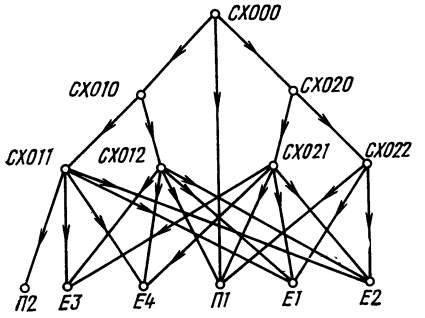


Рис. 5.3. Граф структуры схемы

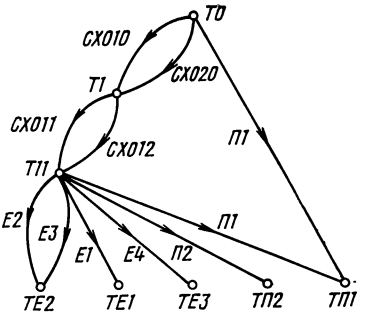


Рис. 5.4. Граф S

(включая и псевдоразъем). В качестве примера рассмотрим возможную форму описания схемы СХ010.

Список компонентов

Имя	Тип	Номер
СХ011	Т11	1
СХ012	Т12	2
Псевдо-разъем	Псевдо-разъем	3

Список цепей

Номер цепи	Номер компонента	Номер вывода
1	1	2
1	2	2
2	1	3
2	3	2
3	1	1
3	2	1
3	3	1

В приведенном описании номера компонентов и цепей либо задаются пользователем на входе системы, либо определяются автоматически в процессе трансляции входного описания. Кроме обязательно присутствующей в описании схемы информации в виде списков компонентов и цепей, пользователь может задать дополнительные требования к проекту. Например, на входе системы могут быть заданы координаты размещения некоторых элементов, группы элементов, которые желательно разместить поближе друг к другу.

Если все компоненты имеют различные имена, то структура всей схемы может быть представлена деревом. В приведенном выше примере у некоторых элементов различных подсхем имена совпадают, и структура всей схемы представляется графом, показанным на рис. 5.3. Появление одинаковых имен элементов в различных подсхемах является естественной ситуацией, возникающей в ходе проектирования по следующим причинам. Во-первых, элементы одного и того же типа могут иметь совпадающие имена. Во-вторых, при отсутствии ограничений на выбор имен в различных подсхемах имена элементов могут совпадать просто случайно.

В реальных СБИС часто используются однотипные подсхемы, которым пользователь присваивает одно и то же имя. Единственным ограничением при этом (как указывалось выше) является требование уникальности имени компонента в подсхеме. Поэтому, если схема содержит две подсхемы одного и того же типа, то им присваиваются различные имена. Можно сказать, что тип подсхемы является ее абсолютной характеристикой, полностью определяющей структуру, в то время как имя является относительной характеристикой, позволяющей различать две подсхемы, входящие в одну и ту же схему.

Граф, показанный на рис. 5.3, строился на основе отображения имен компонентов в вершины, при котором отношение вхождения переходило в отношение смежности. Такой граф отражает использование имен, но не показывает наличие однотипных компонентов. Для того чтобы полностью отобразить иерархическую структуру СБИС, воспользуемся помеченным графом $S(T, K)$, который строится согласно следующим принципам. Вершины T взаимно однозначно соответствуют типам компонентов в СБИС, и ребро $k=(T_i, T_j)$ с меткой a_{ij} принадлежит графу S, если компонент с именем a_{ij} и типом T_j входит в компонент типа T_i. Построение введенного графа S проиллюстрируем на примере. Предположим, что подсхемы СХ010 и СХ020 одного и того же типа Т1, подсхемы СХ011 и СХ012 типа Т11, а элементы Е1—Е4, П1, П2 имеют типы ТЕ1, ТЕ2, ТЕ3, ТП1, ТП2 соответственно. Тогда граф S схемы СХ000 типа Т0 имеет вид, показанный на рис. 5.4.

Отметим, что в графе S требование уникальности имен в подсхеме выражается в том, что все ребра, выходящие из одной и той же вершины, имеют различные метки.

При представлении исходных данных для проектирования матричной СБИС в виде совокупности описаний подсхем предполагалось, что все подсхемы имеют различный тип. Ясно, что многократное описание подсхем одного и того же типа является избыточным. Поэтому для проектирования СБИС, содержащих компоненты одинаковых типов, исходные данные должны содержать лишь описания каждого типа компонента. Для рассматриваемого примера (рис. 5.4) необходимо иметь описания следующих типов: Т0, Т1, Т11.

5.2. МЕТОДИКА ПРОЕКТИРОВАНИЯ ТОПОЛОГИИ

Как было показано в гл. 3, существует ряд стратегий решения задачи при иерархической структуре исходных данных. Рассмотрение возможных подходов начнем с методов, обеспечивающих минимальное время проектирования. Основной принцип при проектировании заключается в однократной разработке топологии компонента каждого типа. Реализация данного принципа возможна как в рамках стратегии «сверху вниз», так и «снизу вверх». В первом случае для каждого типа подсхемы, используя расчетные параметры топологии входящих в нее компонентов, разрабатывается топология самой подсхемы. Начинается процесс с рассмотрения самой старшей схемы, а завершается получением топологий подсхем самого нижнего уровня. Для примера на рис. 5.4 согласно данной стратегии решаются следующие три задачи конструирования:

1)разработка топологии схемы СХ000 типа Т0, в которой подсхемы СХ010, СХ020 представляются незаполненными прямоугольными областями с размерами, вычисленными на основании эмпирических формул;

2) разработка топологии схемы типа Т1;

3) разработка топологии схемы типа Т11.

Топология всей схемы образуется в результате замены незаполненных областей соответствующими топологиями подсхем.

Согласно стратегии «снизу вверх» сначала разрабатываются топологии подсхем самого нижнего уровня, а при разработке топологий подсхем более высоких уровней иерархии используются ранее полученные топологии [6, 7, 35, 147]. Обе стратегии имеют как преимущества, так и недостатки. К преимуществам обеих стратегий относятся: снижение трудоемкости проектирования, что позволяет применять более сложные методы проектирования отдельных компонентов; конструктивная неразрывность функциональных узлов, что создает предпосылки для использования библиотек топологий типовых подсхем. Общим недостатком рассматриваемых стратегий является сравнительно низкая эффективность использования площади кристалла, так как в ходе проектирования в недостаточной степени учитывается специфика конструкции кристалла и топология подсхемы не зависит от положения на кристалле области, в которой реализуется подсхема (подсхемы одного и того же типа имеют одинаковые топологии). Кроме того, различные подсхемы могут сильно отличаться по числу содержащихся в них элементов, что создает дополнительные трудности для рационального размещения элементов на БМК.

Особенность стратегии «сверху—вниз» состоит в том, что она позволяет целенаправленно распределять ресурсы конструкции кристалла и при проектировании подсхемы учитывать расположение других подсхем. Однако эта стратегия может приводить к недопустимо грубым оценкам размеров топологии подсхемы.

Достоинством стратегии «снизу вверх» является то, что она позволяет на каждом этапе знать размеры топологий всех необходимых компонентов. В то же время эти топологии получены без учета расположения на кристалле других подсхем. Поэтому в настоящее время используются комбинированные методы, основанные на сочетании принципов обеих стратегий с включением этапов итерационного улучшения решений [148].

Современные СБИС на БМК характеризуются высокой плотностью элементов. Поэтому применение описанных выше стратегий оказывается оправданным лишь в частных случаях. Для равномерного, эффективного заполнения площади кристалла целесообразно допустить, чтобы подсхемы одного и того же типа имели различную, необязательно прямоугольную, форму топологии. Обеспечить это возможно путем размещения на кристалле каждой подсхемы по частям. Следовательно, еще до размещения требуется каждую подсхему разбить на части, которые в дальнейшем будем называть *блоками*. При этом, чем меньше размер блока, тем большее многообразие форм топологии подсхемы может быть получено. Минимальным блоком является логический элемент. Однако, если заменить каждую подсхему совокупностью входящих в нее элементов, то описание всей схемы станет столь велико, что работа с ним окажется невозможной (из-за недопус-

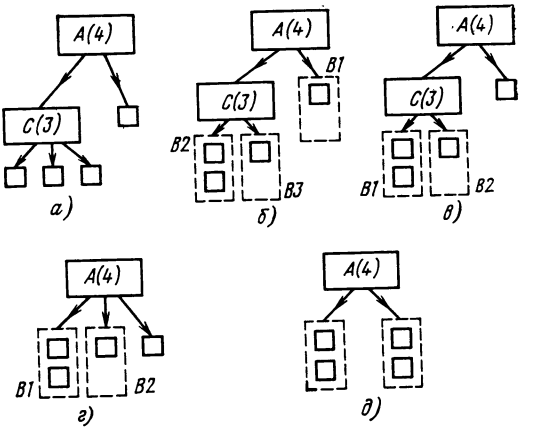
тимо большого времени решения и недостаточного объема оперативной памяти ЭВМ). Поэтому блок должен содержать такое число элементов, при котором описание всей проектируемой схемы (схемы соединений между блоками) имеет допустимые размеры. Таким образом, возникает задача преобразования исходной иерархической структуры в другую, согласованную с конструкцией кристалла, а процесс размещения приобретает иерархический характер: сначала размещаются блоки на кристалле, и затем — элементы внутри блоков. При этом в новой иерархической структуре число уровней равно двум. На первом описывается схема соединений между всеми блоками, на втором — схема соединений между элементами внутри каждого блока. Как следует из гл. 3, размер блока определяет число решаемых подзадач размещения и существенно влияет на время проектирования. Поэтому в ходе проектирования размер блока определяется с учетом доступного объема оперативной памяти и приемлемого времени счета.

Для того чтобы понятию *размер блока* придать конкретный смысл, необходимо определить форму блока. По существу, выбор формы блока зависит от опыта разработчика и определяется с учетом особенностей конструкции кристалла и применяемых алгоритмов размещения. В случае применения БМК линейчатого типа целесообразно использовать блок в форме прямоугольника, образованного элементами, расположенными в одной линейке, вплотную друг к другу. Тогда размером блока является его длина, равная сумме длин входящих в него элементов. Аналогичным образом можно определить размер каждой подсхемы исходных данных. Вычисление этих размеров может быть выполнено последовательно «снизу вверх». В процессе образования блоков (компоновки) в один блок могут попасть не только разные элементы, но и подсхемы, размер которых меньше размера блока. Таким образом, в блоки объединяются компоненты, у которых размеры меньше блока, а содержащие их подсхемы по размеру превышают блок. Исходные данные позволяют при компоновке рассматривать подсхемы в любом порядке. Однако, если их рассматривать в порядке «снизу вверх» и пользоваться подстановкой ранее скомпонованных подсхем, то появляется возможность дополнять элементами блоки, которые на предыдущих шагах оказались незаполненными. Сказанное проиллюстрируем примером.

Пусть требуется скомпоновать схему A (рис. 5.5,*а*), размер которой 4. Схема A содержит подсхему C размера 3 и единичный элемент. Результат компоновки подсхем без подстановки содержит три блока $B1, B2, B3$ (рис. 5.5,*б*). Если же подсхемы компоновать «снизу вверх», то сначала образуется два блока $B1$, $B2$ (рис. 5.5,*в*) подсхемы C. Тогда при компоновке схемы A в ее описание может быть представлен результат компоновки подсхемы C (рис. 5.5,*г*), а блок $B2$ дополнен элементом схемы A. В результате формируются всего два блока (рис. 5.5,*д*).

В процессе компоновки помимо схемы соединений между блоками формируются схемы соединений элементов внутри блоков.

Рис. 5.5. Компоновка подсхем «снизу вверх»



При этом, если блок содержит подсхемы, то схема блока формируется последовательной подстановкой входящих в него подсхем. Отметим также, что объединение элементов в блоки осуществляется в соответствии с некоторым критерием оптимизации, связанным с улучшением характеристик проекта. В качестве такого критерия может быть использована максимальная связность элементов внутри блока. При этом в особом положении оказываются периферийные элементы, которые обеспечивают выход на внешние контактные площадки, не связанные между собой. Поэтому каждый периферийный элемент целесообразно поместить в один блок. Объединять периферийный элемент с внутренними нецелесообразно, так как внутренние и периферийные элементы размещаются в различных областях кристалла. Поэтому при компоновке каждый периферийный элемент рассматривается как отдельный блок.

Процесс формирования блоков проиллюстрируем с помощью схемы, которой соответствует граф S, показанный на рис. 5.4. Компоновка каждого типа подсхемы выполняется однократно. В результате граф S преобразуется в граф, представленный на рис. 5.6. Пример распределения элементов по блокам для подсхемы типа T11 показан на рис. 5.7. Отметим, что описание блоков по форме совпадает с описанием подсхем, но тип блоку можно не присваивать.

При работе с топологией схемы (например, при корректировке) пользователю необходимо знать всю цепочку вложенных друг в друга подсхем, младшая из которых содержит заданный элемент. Фактически это означает, что при иерархическом описании схемы полным именем физически реализованного на кристалле элемента (подсхемы) служит цепочка имен подсхем-предков. В построенном ранее графе S (рис. 5.4, 5.6) каждой вершине могут соответствовать несколько физически различных фрагментов топологии. Поэтому построим другой граф D, вершины которого взаимно однозначно соответствуют компонентам топологии. Граф

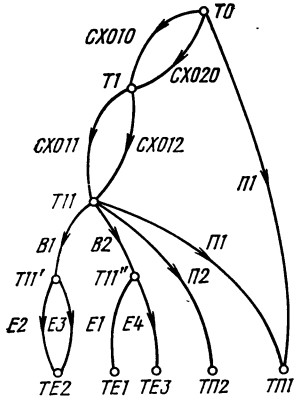


Рис. 5.6. Преобразованный граф S

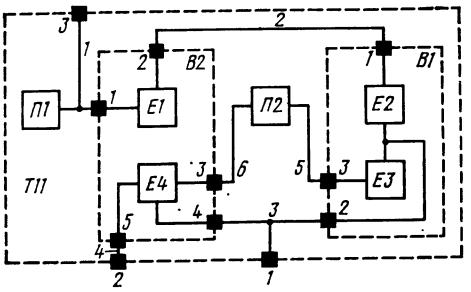


Рис. 5.7. Пример распределения элементов по блокам для подсхемы типа Т11

D представляет собой дерево, листья которого взаимно однозначно соответствуют элементам топологии, корень — всей схеме, а каждая внутренняя вершина — реализации подсхемы. Так как имена элементов и подсхем могут повторяться для физически различных объектов, вершинам графа D присваиваются индивидуальные номера (числа в скобках на рис. 5.8). Как и в предыдущих моделях, ребра графа D отображают вхождение компонентов. Ясно, что граф D может быть получен из графа S последовательным преобразованием «сверху вниз». Преобразованию подвергается каждая вершина v, в которую входят $m(m \geqslant 2)$ ребер. При этом подграф, образованный потомками вершины v, дублируется m раз, и к каждому вновь образованному подграфу S_v подключается по одному ребру из ранее входящих в v. Старшей вершине каждого подграфа S_v присваивается имя подсхемы, совпадающее с меткой входящего в нее ребра.

Имена компонентов, заданные на входе, позволяют различать их только внутри подсхемы, т. е. являются относительной характеристикой. Полные имена, являясь абсолютной характеристикой,

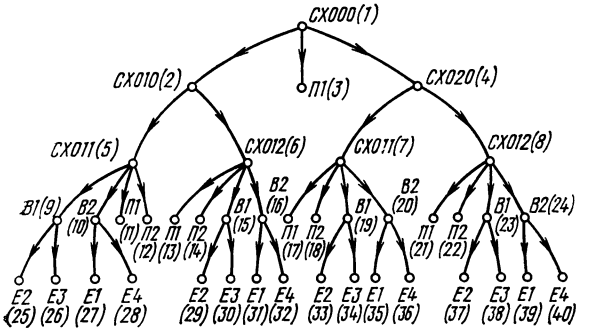


Рис. 5.8. Пример графа D

отличаются друг от друга у любых двух различных компонентов всей схемы, т. е. позволяют идентифицировать любой объект в пределах всей схемы. Граф D дает возможность легко определить полное имя любого компонента, зная номер соответствующей ему вершины. Для этого, двигаясь из исходной вершины против направления входящего в нее единственного ребра, можно найти предшествующую вершину, а продолжая процесс подобным образом, всегда приходим в корневую вершину. Имена вершин пройденного пути в порядке от корня к заданной вершине образуют полное имя. Например, вершине *12* (рис. 5.8) соответствует путь (*12*, *5*, *2*, *1*), т. е. полное имя в данном случае запишется как СХ000/СХ010/СХ011/П2. Для решения обратной задачи (определение номера вершины по заданному полному имени) требуются несколько бо́льшие вычислительные затраты. Это связано с тем, что теперь для каждой просматриваемой вершины нужно искать соответствующего сына, а число сыновей может быть велико.

Большая длина полных имен является следствием использования иерархической структуры. Сократить их можно было бы, введя требование уникальности имен подсхем. В этом случае полное имя подсхемы просто совпадало бы с ее именем на входе, а у элемента состояло бы из двух частей: входного имени элемента и содержащей его подсхемы. Однако при этом увеличивается объем исходных данных: для каждой подсхемы требуется задание имен входящих в нее подсхем.

После того как построен граф D, используя операцию подстановки, нетрудно перейти к двухуровневому представлению схемы, которому соответствует граф D_2. В этом представлении схема описывается относительно блоков, а каждый блок — относительно элементов. Граф D_2 для рассматриваемого примера показан на рис. 5.9.

Отметим, что выбор двухуровневого представления произведен исходя из эвристических соображений. При этом учитывались максимальный объем проектируемой схемы, доступный объем оперативной памяти, доступное время проектирования и простота программной реализации. Так, в подсистеме АРТИС применение двухуровневого представления позволяет проектировать матричные БИС, содержащие до 100 тыс. логических элементов, на ЭВМ типа ЕС 1060 с оперативной памятью 1М байт. Вместе с тем рас-

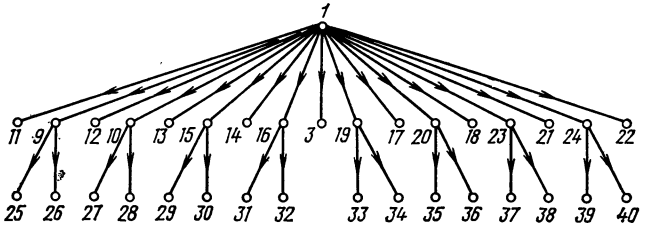


Рис. 5.9. Двухуровневое представление схемы — граф D_2

смотренные выше принципы преобразования иерархического описания применимы для получения произвольного (не обязательно двухуровневого) представления схемы.

Полученные графы D, D_2 и описания схемы и блоков позволяют в дальнейшем реализовать иерархическое проектирование топологии матричных БИС. Этот процесс включает решение задачи размещения на кристалле блоков и периферийных элементов, размещения элементов внутри блоков и трассировки соединений. Последняя задача также решается иерархически: разбиением трассировки всего кристалла на задачи построения соединений в каждом канале. В последующих параграфах данной главы описываются алгоритмы, используемые на отдельных этапах проектирования.

5.3. АНАЛИЗ ИСХОДНЫХ ДАННЫХ

Прежде чем решать задачи непосредственного проектирования, необходимо определить интегральные характеристики проектируемой матричной БИС, которые используются для выбора БМК и определения параметров формируемого иерархического представления D_2, а также позволяют сформировать набор применяемых алгоритмов решения отдельных задач и определить значения свободных параметров алгоритмов. К этим характеристикам относятся; число n логических элементов; число p периферийных элементов: суммарная длина l логических элементов; суммарное число v используемых выводов элементов; число c цепей схемы. При определении перечисленных характеристик используется граф $S=(T, K)$ (рис. 5.4), отражающий структуру входного описания схемы. Входное описание просматривается «сверху вниз». В этом случае для каждого типа j компонента легко определяется требуемое число r_j его реализаций на кристалле. А именно, справедливо выражение

$$r_j = \sum_{(i,j)\in K} r_i,$$

где сумма берется по всем ребрам (i, j), входящим в вершину j.

Рассматривая очередной тип j подсхемы, нетрудно определить число n_j логических элементов в нем, число p_j периферийных элементов, суммарную длину l_j логических элементов, суммарное число v_j используемых выводов логических и периферийных элементов, число c_j внутренних (не выходящих на псевдоразъем) цепей. Тогда интегральные характеристики СБИС определяются по общей формуле

$$w = \sum_{j\in T'} r_j w_j,$$

где w — вычисляемая характеристика ($w\in\{n, p, l, v, c\}$); T' — подмножество T, соответствующее типам подсхем. Отметим, что при определении числа цепей схемы используется свойство иерар-

хического описания схем: *для каждой цепи существует единственная подсхема, в которой эта цепь является внутренней.*

По интегральным характеристикам осуществляется выбор наиболее предпочтительного БМК из библиотеки типовых конструкций. Для выбранного кристалла на основании статистических данных, полученных в ходе предшествующего проектирования, может быть вычислена вероятность успешного завершения проектирования топологии. В частности, нулевая вероятность соответствует невозможности реализации заданной схемы на имеющемся кристалле (например, если число периферийных элементов СБИС или суммарная длина логических элементов превышает соответствующий параметр кристалла).

Рассмотрим более подробно задачу определения параметров формируемого иерархического представления. Согласно описанной выше методике проектирования для размещения всех элементов требуется сначала разместить блоки, а затем разместить элементы в блоках. Как было показано в гл. 3, размер блока может существенно влиять на время решения задачи размещения и, следовательно, на продолжительность всего цикла проектирования. При определении оптимальной структуры формируемого представления в гл. 3 учитывались вычислительные затраты как на размещение, так и на разбиение задачи на подзадачи. В рассматриваемом случае двухуровневого размещения разбиение задачи выполняется только на верхнем уровне. Время выделения блоков всей схемы включает время разбиения на блоки каждой подсхемы. Проведем анализ зависимости времени разбиения подсхемы от числа элементов в блоке.

Предположим, что используется последовательный алгоритм [32], согласно которому сначала отбираются элементы в первый блок, затем — во второй и т. д., пока все элементы не попадут в блоки. Пусть подсхема, содержащая N элементов, разбивается на k блоков по $x=N/k$ элементов в каждом. Если при включении в блок очередного элемента просматриваются все свободные элементы, то трудоемкость выделения x элементов из N определяется выражением $\tau(x,\ N)=N+(N-1)+\ldots+(N-x+1)=x(N-(x-1)/2)$. При этом трудоемкость τ формирования всех блоков равна:

$$\tau=\tau(x,\ N)+\tau(x,\ N-x)+\ldots+\tau(x,\ 2x)=N^2(1-0{,}5x^{-2})+$$
$$+N(3/2x-1{,}5x+0{,}5)+0{,}5x(x-1)-x.$$

Учитывая, что при преобразовании реальных иерархических описаний основной вклад во время формирования блоков вносит разбиение больших подсхем ($N\gg x$), то $\tau\approx N^2+N/2$. Поэтому можно считать, что время разбиения подсхем на блоки не зависит от числа элементов в блоке. Следовательно, при определении оптимального размера блока будем учитывать только время решения задач размещения.

Обозначим число логических элементов СБИС через n, а число блоков — переменной N. Если для решения задач размеще-

ния блоков и элементов внутри блоков используются алгоритмы с одной и той же оценкой трудоемкости $T(m)$ (m — число размещаемых объектов), то оценка трудоемкости решения всей задачи размещения имеет вид

$$\tau = T(N) + NT(n/N). \qquad (5.1)$$

Выражение (5.1) показывает, что в общем случае, изменяя число блоков схемы и соответственно число элементов в блоке, можно получить различные оценки трудоемкости решения. Поэтому целесообразно использовать такое значение параметра N, при котором величина τ достигает минимума. Это значение N соответствует и минимальному времени всего цикла проектирования, так как временные затраты на трассировку не зависят от размера блока, а время преобразования иерархического описания схемы в первом приближении можно считать постоянным. Для алгоритмов, используемых в подсистеме АРТИС, выполняется соотношение $T(m) \sim O(m^{\beta})$. Поэтому выражение (5.1) принимает вид

$$\tau \sim N^{\beta} + N(n/N)^{\beta}. \qquad (5.2)$$

Примеры графиков зависимости трудоемкости размещения от числа элементов в блоке при $\beta = 2$ и различных значениях n показаны на рис. 5.10. Дифференцируя правую часть (5.2) и при-

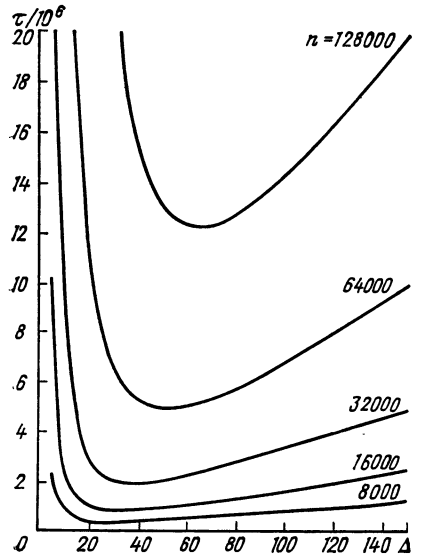


Рис. 5.10. Зависимость трудоемкости τ размещения от числа Δ элементов в блоке при вычислительной сложности алгоритма размещения $O(m^2)$ (n — число логических элементов в СБИС)

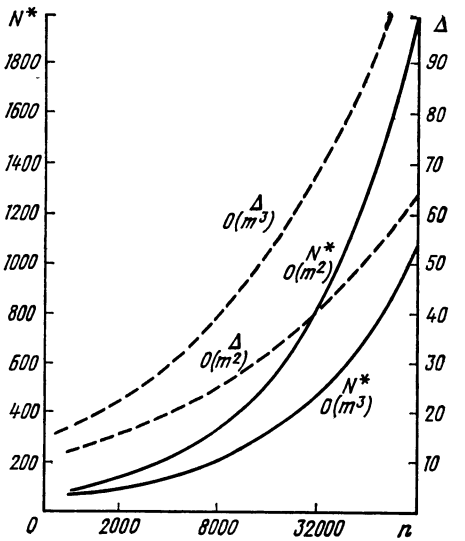


Рис. 5.11. Зависимость оптимального числа N^* блоков и размера Δ блока от числа n элементов в СБИС при вычислительной сложности алгоритма размещения $O(m^0)$, размеры логических элементов приняты единичными

равнивая нулю полученное выражение, находим значение N^*, соответствующее минимальной трудоемкости

$$N^* = (n^{\beta}(\beta - 1)/\beta)^{1/(2\beta-1)}, \beta > 1.$$

Зависимость оптимального числа блоков от числа элементов схемы показана на рис. 5.11.

Учитывая, что у элементов могут быть различные длины, а для наилучшего использования площади кристалла желательно иметь блоки приблизительно одинаковой длины, рекомендуемую длину блока определим, как $\Delta = l/N^*$, где l — суммарная длина всех элементов СБИС. Графики зависимости рекомендуемой длины блока от числа элементов схемы показаны штриховой линией на рис. 5.11.

Для того чтобы оценить снижение трудоемкости за счет реализации иерархического размещения, подставим найденное значение в выражение (5.1). Тогда имеем

$$\tau \sim n^{\beta^2/(2\beta-1)}, \qquad (5.3)$$

т. е. временна́я сложность решения задачи уменьшается. Например, при $\beta = 2$ $\tau \sim n^{1,33}$. Таким образом, формула (5.3) дает теоретическое объяснение экспериментально полученной зависимости времени размещения от числа элементов в системе фирмы IBM (см. § 2.5). Ясно, что представленные результаты исследования трудоемкости решения задач при иерархическом подходе носят, в основном, приближенный характер. Для получения более точных оценок необходимо учитывать мультипликативные постоянные трудоемкости алгоритмов, вычисленные по данным эксплуатации системы.

5.4. ФОРМИРОВАНИЕ БЛОКОВ

Формирование блоков выполняется в результате решения задачи компоновки подсхем, описания которых заданы на входе. Причем компоновке подлежат лишь подсхемы с размером, превышающим размер Δ блока, но содержащие компоненты, размер которых меньше Δ. Кроме того, ясно, что в блоки объединяются лишь компоненты с размером, меньшим Δ. Отметим также, что при формировании блоков подсхемы просматриваются последовательно в произвольном порядке.

Исходные данные для компоновки подсхемы содержат описание подсхемы и, возможно, указания пользователя о группах элементов, которые требуется разместить поближе друг к другу. Требуется сформировать блоки, описания их схем, описание исходной подсхемы относительно блоков и преобразовать граф S в соответствии с результатом компоновки. Ограничениями задачи являются: требование образования из каждого периферийного элемента отдельного блока; выполнение неравенства $\Delta_i \leqslant \Delta$ для каждого блока i, где Δ_i — суммарная длина элементов блока. В качестве критерия оптимальности компоновки целесообразно использовать максимум числа внутренних цепей блока. Такой кри-

терий можно считать оправданным, во-первых, потому, что он связан с минимизацией суммарной длины соединений, а во-вторых, при данном критерии сокращается описание всей схемы относительно блоков.

Решение задачи компоновки подсхемы состоит в последовательном построении блоков путем наращивания текущей группы компонентов. Согласно алгоритму наращивания группы на каждом промежуточном шаге включается совокупность всех компонентов, инцидентных некоторой цепи. Для каждой цепи, инцидентной компонентам наращиваемой группы E и множеству компонентов, еще не включенных в блоки, вычисляется оценка, равная числу инцидентных ей компонентов из E. Совокупность компонентов для включения в группу определяется цепью с минимальной оценкой.

Данный алгоритм отличается от традиционного последовательного тем, что он позволяет выбирать для включения в блок не один, а сразу несколько компонентов. Именно это создает предпосылки для повышения вычислительной эффективности алгоритма. (Ниже будет показано, что временная сложность рассматриваемого алгоритма равна $O(q)$, где q — число компонентов в подсхеме.) Фактически вместо элемента в данном алгоритме выбирается цепь. Поэтому важно учитывать три возможных состояния цепи: все компоненты, инцидентные цепи, могут разместиться в пределах одного блока в ходе дальнейшей компоновки; компоненты цепи в результате компоновки обязательно будут принадлежать различным блокам; все компоненты цепи уже распределены по блокам.

До компоновки в первом состоянии находится каждая цепь, удовлетворяющая условиям: суммарная длина компонентов этой цепи меньше Δ; цепь не инцидентна псевдоразъему или периферийному элементу.

В третьем состоянии находятся цепи, которые инцидентны только компонентам, не включаемым в блоки (большие подсхемы и псевдоразъем). Остальные цепи находятся во втором состоянии. В ходе компоновки номер состояния цепи может только увеличиться, он корректируется на определенных шагах алгоритма.

Алгоритм компоновки подсхемы можно представить следующей последовательностью шагов.

1. Каждому периферийному элементу поставить в соответствие отдельный блок, который пометить как периферийный.

2. Поместить все компоненты с размером, превышающим Δ, как не включаемые в блок.

3. Выбрать цепь i для продолжения компоновки. Если все цепи в третьем состоянии, то сформировать схему последнего блока (при условии, что она еще не сформирована); сформировать описание всей подсхемы относительно блоков; модифицировать граф S и завершить компоновку.

4. Включить в формируемый блок компоненты E_i, инцидентные цепи i. Если в блок включены все компоненты E_i, то цепь перевести в третье состояние и перейти к шагу 3.

5. Если в формируемый блок вошел хотя бы один компонент из E_i, то цепь i перевести во второе состояние.

6. Сформировать схему блока и перейти к шагу 3.

Прокомментируем отдельные шаги алгоритма. На шаге 3 при выборе цепи предпочтение отдается цепям, находящимся в первом состоянии. Когда формируемый блок содержит хотя бы один компонент, то из числа цепей, инцидентных компонентам из блока, выбирается та, которая инцидентна меньшему числу компонентов из E. На шаге 4 может оказаться, что в блок включается лишь часть компонентов цепи i из-за того, что суммарная длина компонентов блока достигла предельной величины. В ходе формирования схемы блока на шаге 6 корректируется состояние цепей, и если блок содержит подсхемы, то выполняется необходимое число подстановок. Отметим также, что если подсхема содержит группы элементов, которые требуется разместить на кристалле ближе друг к другу, то формирование блока начинается не с выбора цепи, а с включения в блок элементов очередной группы.

Для того чтобы эффективно реализовать алгоритм компоновки подсхемы, можно использовать списки, позволяющие избежать просмотра всех цепей на промежуточных шагах. В ходе компоновки желательно выбирать очередную цепь, находящуюся в первом или втором состоянии, переводить цепь из первого состояния во второе, а также определять состояние цепи и ее оценку с помощью постоянного числа операций, не зависящего от общего числа цепей и компонентов. Такая организация вычислений возможна при использовании структуры данных, условно показанной на рис. 5.12. Массив СОСТ определяет состояние и оценку каждой цепи, равную числу компонентов из E, которые инцидентны этой цепи. Если $\text{СОСТ}(i)>0$, то цепь i находится в первом состоянии, в противном случае — во втором, а $|\text{СОСТ}(i)|$ — оценка цепи i. Массив СЛЕД для каждой цепи i указывает следующую цепь $\text{СЛЕД}(i)$, находящуюся в том же состоянии, что i. Первая цепь, находящаяся в первом или втором состоянии, определяется соответственно как $\text{СЛЕД}(c+1)$ или $\text{СЛЕД}(c+2)$, где c — число цепей подсхемы. Для последней цепи i $\text{СЛЕД}(i)=0$. Массив ПРЕД для каждой цепи i определяет предшествующую $\text{ПРЕД}(i)$. Для первой цепи i $\text{ПРЕД}(i)=0$.

На шаге 3 алгоритма компоновки отдается предпочтение множеству C_b цепей, инцидентных компонентам формируемого блока. Поэтому помимо структуры данных (рис. 5.12), соответствующей цепям всей подсхемы, целесообразно использовать еще два массива, подобных СЛЕД и ПРЕД, но содержащих только цепи из C_b.

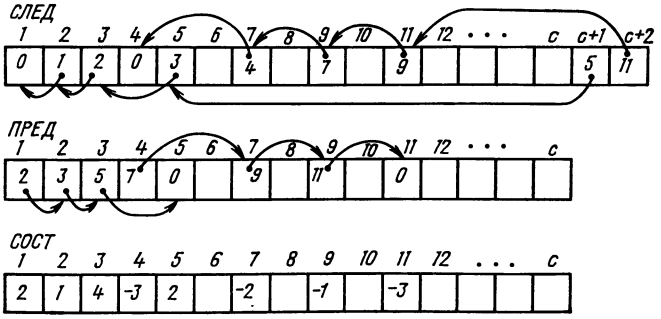


Рис. 5.12. Структура данных для формирования блоков

Проиллюстрируем использование приведенной структуры данных. Выбор первой цепи для начала формирования блока сводится к тому, что определяется значение i=СЛЕД(c+1). Если i=0, то i=СЛЕД(c+2). При i=0 компоновка завершается. Перевод цепи i из первого состояния во второе выполняется следующими операциями:

$$\text{СЛЕД (ПРЕД } (i)) = \text{СЛЕД } (i)$$
$$\text{СЛЕД } (i) = \text{СЛЕД } (c+2)$$
$$\text{СЛЕД } (c+2) = i$$
$$\text{СОСТ } (i) = -\text{СОСТ } (i),$$

а перевод цепи из любого состояния в третье выполняется одной операцией

$$\text{СЛЕД (ПРЕД } (i) = \text{СЛЕД } (i).$$

Проведем анализ временной сложности описанного алгоритма. Время решения $T=tm=tq/\Delta_e$, где t — время формирования одного блока; m — число блоков в подсхеме, содержащей q компонентов; Δ_e — среднее число компонентов в блоке ($\Delta_e=\Delta/l_0$, l_0 — средняя длина компонента в подсхеме).

Время t определяется двумя составляющими — временем корректировки состояний цепей после включения очередной совокупности компонентов и временем выбора следующей цепи для продолжения формирования блока:

$$t = \sum_{i=1}^{h} S_e\, S_c + |C_p^{(i)}|,$$

где S_c — среднее число компонентов E_i; h — число шагов с номером 3 при формирования блока ($h=\Delta_e/S_c$); S_e — среднее число цепей, инцидентных одному компоненту; $|C^{(i)}{}_p|$ — число внешних цепей формируемого блока после включения элементов E_i.

Учитывая, что число компонентов в блоке после включения очередной группы E_i определяется как iS_c, величину $|C^{(i)}{}_p|$ можно вычислить по формуле

$$|C_p^{(i)}| = (i\, S_c)^{\mu},$$

где μ — экспериментально определяемая величина, лежащая в пределах 0,75—0,5 [7] для различных типов схем.

Тогда

$$t \leqslant \Delta_e\, S_e + \Delta_e^{\mu+1}/S_c,\ T \leqslant q\,(S_e + \Delta_e^{\mu}/S_c).$$

Полученная оценка трудоемкости подтверждает вычислительную эффективность алгоритма. Вместе с тем, эта оценка показывает, что время компоновки для данного алгоритма зависит от размера блока. Поэтому при определении размера блока, доставляющего минимальное время размещения элементов на кристалле, необходимо дополнительно учитывать вычислительные затраты на компоновку. Поэтому при применении описанного алгоритма компоновки выражение (5.1) для более точного определения размера блока должно быть заменено следующим:

$$\tau = T(N) + NT(n/N) + \sum_{i \in T} q_i\,(S_e + (n/N)^{\mu})/S_c) =$$
$$= N^{\beta} + N\,(n/N)^{\beta} + n\,(S_e + (n/N)^{\mu}/S_c).$$

Найти N, при котором последнее выражение достигает минимума, можно численным методом.

5.5. ПОСТАНОВКА И РЕШЕНИЕ ЗАДАЧИ РАЗМЕЩЕНИЯ

Основным требованием при размещении элементов на кристалле является создание условий, обеспечивающих полную трассировку соединений. Для выполнения данного требования необходимо учитывать характерные свойства конструкции кристалла, оказывающие существенное влияние на результаты трассировки. Прежде всего следует отметить, что число вертикальных магистралей примерно в два раза превышает число горизонтальных. Поэтому предпочтительной является реализация соединений вертикальными сегментами. Причем с точки зрения наиболее экономного использования ресурса кристалла, оптимальными можно считать *линейные сегменты* (ЛС, рис. 5.13). Это объясняется тем, что ЛС представляет собой кратчайшую реализацию соединения, не содержит межслойных переходов, не занимает горизонтальных магистралей. Кроме того, при использовании наиболее распространенной методики двуслойной трассировки, когда все горизонтальные сегменты располагаются в одном слое, а вертикальные — в другом и точки поворота трассы не располагаются на вертикальных магистралях, проходящих в данном канале через выводы других трасс, ЛС не налагают никаких ограничений на реализацию остальных трасс. Перечисленные соображения подтверждаются и практическим опытом проектирования БМК данного типа.

Следовательно, кроме традиционного критерия оптимального размещения (минимум суммарной длины соединений), при размещении элементов на кристалле необходимо учитывать критерий максимума числа ЛС.

Рассмотрим необходимые условия выполнения полной трассировки, которые могут быть учтены на этапе размещения. Первое из них ограничивает число элементов и транзитных трасс в каждой линейке

$$\sum_{i \in E} L_i + \lceil (n_t - n_p)\, l_0/p_0 \rceil \leqslant L, \qquad (5.4)$$

где E — множество элементов, размещенных в линейке; L_i — длина элемента i; n_t — число транзитных трасс, пересекающих линейку; n_p — суммарное число проходов в элементах линейки; l_0 — длина ячейки БМК; p_0 — максимальное число транзитных трасс, проходящих через ячейку базового кристалла; обозначение $\lceil A \rceil$ определяет неотрицательное минимальное целое $B \geqslant A$; L — длина линейки.

Второе условие ограничивает число n_s трасс, пересекающих вертикальное сечение s,

$$n_s \leqslant n_{\max}, \qquad (5.5)$$

где $n_{\max}$ — суммарное число горизонтальных магистралей во всех каналах. Условие (5.5) должно выполняться для всех вертикальных сечений кристалла.

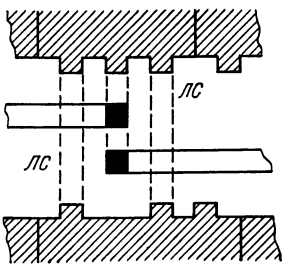


Рис. 5.13. Примеры линейных сегментов

В соответствии с методикой проектирования, описанной в § 5.3, размещение состоит из двух основных этапов: размещение блоков и размещение элементов внутри каждого блока.

Рассмотрим первый этап. Для размещения блоков на кристалле модели схемы, в которых каждая цепь представляется взвешенным полным подграфом, не соответствуют специфическим требованиям задачи. Действительно, если некоторая цепь соединяет m блоков, то представление этой цепи в модели будет требовать равномерного приближения m блоков друг к другу. Вместе с тем, размещение этих блоков в виде линейной цепочки (один блок под другим) доставляет как минимум суммарной длины трассы, так и максимум числа ЛС. Поэтому целесообразно еще до размещения так определить конфигурации цепей, чтобы создать предпосылки для получения максимального числа ЛС. Другими словами, возникает задача определения модели схемы, в которой каждая цепь представляется деревом. Если при построении такой модели максимизировать число параллельных ребер, а затем максимальные группы параллельных ребер попытаться реализовать в виде линейных сегментов, то можно ожидать, что будут созданы предпосылки для успешной трассировки. Обозначим искомый граф через $G(X, U)$, где X — множество вершин, взаимно однозначно соответствующих блокам; U — множество ребер, входящих в деревья, которые отображают цепи. Требуется так определить конфигурации деревьев (задать множество U), чтобы скелет $H(X, V)$ графа G содержал минимальное число ребер. При этом группе $u_1, \ldots, u_m$ параллельных ребер графа G соответствует одно ребро $v \in V$, соединяющее те же вершины и имеющее вес $w = m$.

Для решения данной задачи целесообразно воспользоваться эвристическим методом [32], имеющим приемлемую вычислительную сложность. Ясно, что сказанное выше относительно целесообразности соединения блоков с помощью ЛС справедливо и для элементов. Поэтому на этапе компоновки представляет интерес такой подход, когда помимо *«горизонтальных» блоков* (все элементы должны быть размещены в одной линейке) формируются и *«вертикальные блоки»*, элементы которых предполагается разместить друг под другом в различных линейках. После того как определена модель схемы, можно найти относительное расположение блоков на кристалле, используя их точечное представление. Учесть при этом свойства проектируемой схемы в целом позволяет силовой метод, согласно которому сначала размещаются периферийные блоки, а затем в результате решения статической задачи [6, 32] — остальные. Размещение периферийных элементов выполняется традиционными методами с использованием коэффициентов условной связности. Учитывая, что для рассматриваемой конструкции кристалла, реализация соединений ЛС является наиболее предпочтительной, лучшим результатом силового размещения можно считать тот, в котором бóльшее число ребер расположено вертикально. Очевидно, что положение ребра зависит от размещения периферийных блоков. Поэтому после того как

найден первоначальный вариант силового размещения, можно найти новое расположение периферийных блоков (с сохранением первоначального циклического порядка), при котором большинство ребер будет направлено вертикально или приблизительно вертикально. Так как циклический порядок периферийных блоков сохраняется, можно говорить об искомом угле поворота. Этот угол определяется как среднее значение угла отклонения ребер от вертикального направления:

$$\alpha = \sum_{i=1}^{R} w_i \alpha_i \Big/ \sum_{i=1}^{R} w_i,$$

где $R = |V|$ — число ребер графа H; w_i — вес ребра $i \in V$; α_i — угол поворота ребра i против часовой стрелки до совпадения с вертикальной прямой. Новое размещение периферийных блоков представляет собой циклический сдвиг начального варианта по часовой стрелке на число позиций, равное $M\alpha/2\pi$, где M — число периферийных ячеек кристалла. Определение координат блоков, соответствующих размещению периферийных блоков, не требует существенных вычислительных затрат, так как сводится к умножению матрицы на вектор. Описанные этапы размещения проиллюстрируем примером. Исходная схема соединений между блоками показана на рис. 5.14, ее модель в виде графа H — на рис. 5.15 (цифры указывают веса $w \neq 1$ ребер V), а результат начального силового размещения — на рис. 5.16,*а*. После перестановки периферийных блоков (угол поворота равен около 90°) в результате силового размещения (рис. 5.16,*б*) большинство ребер с максимальным весом имеют направление, близкое к вертикальному. Следующим этапом является получение матричного размещения блоков на модели кристалла, являющейся ортогональной решеткой, в узлах которой требуется разместить вершины, отображающие внутренние блоки. Число строк этой матрицы равно числу Q линеек на кристалле, а число столбцов $\lceil N/Q \rceil$, где N — число блоков. Критерием оптимальности перехода от силового размеще-

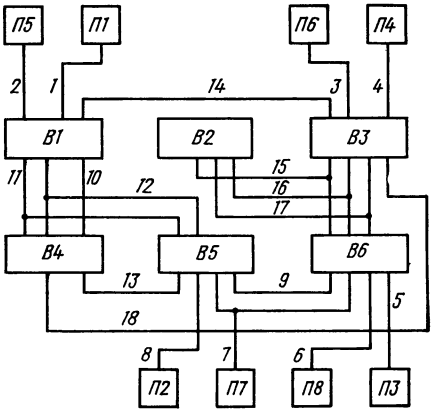


Рис. 5.14. Схема соединений блоков

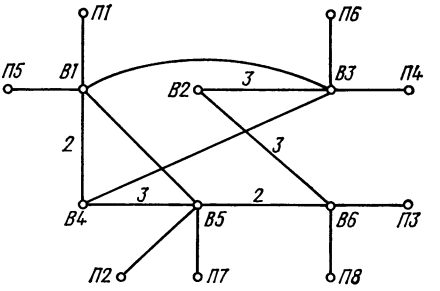


Рис. 5.15. Граф H

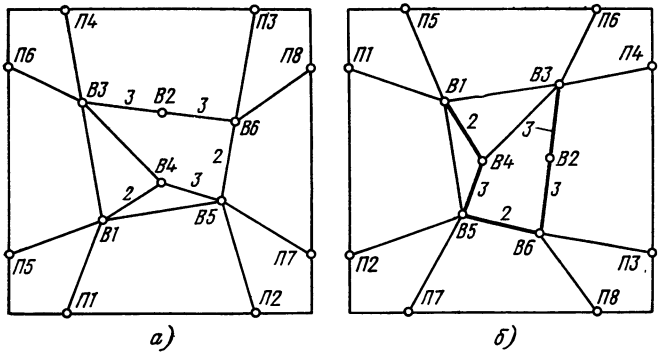


Рис. 5.16. Результат начального силового размещения (*а*) и его поворот на 90° (*б*)

ния к матричному является минимум изменения координат центров блоков и максимум вертикальных ребер.

Для решения данной задачи используется конструктивный эвристический алгоритм получения начального варианта и итерационные алгоритмы улучшения размещения. Алгоритм начального размещения базируется на последовательном принципе с установкой на очередном шаге либо одного блока, либо группы блоков, образующих «вертикальную цепочку». Так, для рассматриваемого примера (рис. 5.16,*б*) на первом шаге в первый столбец устанавливается группа блоков *B5*, *B4*, *B1*, а на втором — *B6*, *B2*, *B3* (рис. 5.17,*а*). Оптимизация полученного варианта размещения выполняется традиционными итерационными методами [19, 35] с учетом ограничений (5.4) и (5.5). Причем контроль ограничений достаточно проводить лишь в сечениях, проходящих через блоки.

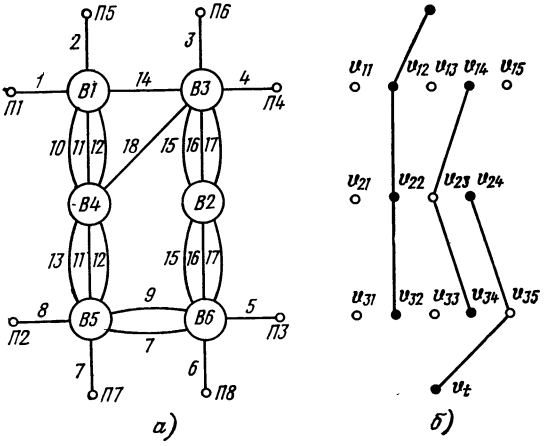


Рис. 5.17. Начальный вариант размещения (*а*) и ее модель *T* (*б*)

После того как найден улучшенный вариант размещения блоков, удовлетворяющий условиям (5.4), (5.5), выполняется этап размещения транзитных трасс. Для решения данной задачи используется модель T в виде совокупности горизонтальных рядов вершин (размещению блоков на рис. 5.17,*а* соответствует модель, показанная на рис. 5.17,*б*). Каждая вершина отображает блок (на рисунке такие вершины зачернены) или межблочный промежуток и характеризуется пропускной способностью, определяющей максимально допустимое число транзитных трасс, проходящих через эту вершину. Число рядов в модели T равно числу линеек на кристалле.

Пропускная способность вершины, отображающей блок, равна суммарному числу проходов в элементах этого блока. Пропускная способность вершины, соответствующей межблочному промежутку, зависит от конструкции кристалла. Например, для кристалла типа И-200 эта величина определяется расстоянием между соседними макроячейками, а в линейчатом кристалле суммарная пропускная способность межблочных вершин ограничена разностью между длиной линейки и суммарной длиной блоков в линейке.

Для каждой транзитной трассы известны две линейки i, j, блоки в которых она соединяет. Причем в линейке может быть несколько блоков, инцидентных этой трассе. Требуется определить для каждой транзитной трассы блоки и межблочные промежутки, через которые она проходит. Примеры возможных конфигураций транзитных трасс показаны на рис. 5.17,*б*. Критерий оптимальности размещения транзитных трасс совпадает с критерием, используемым при размещении блоков.

Пусть в линейке i вершины, инцидентные рассматриваемой транзитной цепи, лежат на отрезке $[a, b]$, а в линейке j на отрезке $[c, d]$ (рис. 5.17,*а*). Если проекции этих отрезков на горизонтальную прямую пересекаются, то транзитная трасса может состоять из единственного вертикального сегмента. На рис. 5.18,*а* три возможных положения: t_1, t_2, t_3 транзитной трассы показаны штриховой линией. Для минимизации числа использованных горизонтальных магистралей транзитные трассы разбиваются на две группы: с пересекающимися проекциями отрезков $[a, b]$ и $[c, d]$ и с непересекающимися. Сначала для каждой трассы первой

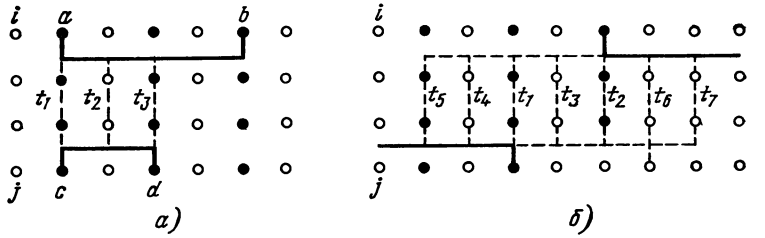


Рис. 5.18. Примеры положений транзитных трасс

группы ищется ЛС *ti*. При этом возможны следующие ситуации: 1) искомого сегмента нет; 2) существует единственный сегмент; 3) существует несколько допустимых сегментов.

В первой ситуации трасса переносится во вторую группу, в третьей выбирается лучший сегмент. При этом предпочтение отдается сегменту, который проходит через блоки. Проведение трассы через проходы в элементах способствует экономному использованию площади кристалла, так как такая трасса не занимает в линейке дополнительной площади. В случае же проведения трассы через межблочный промежуток может потребоваться дополнительная площадь, равная площади одной ячейки. Кроме того, такое проведение трасс может потребовать смещения блоков и привести к уменьшению числа ЛС. Обозначим через p_v число транзитных трасс, проведенных через вершину v. Если все допустимые сегменты проходят через межблочные промежутки, то выбирается сегмент, который проходит через вершины с величиной p, не кратной пропускной способности ячейки. Кроме того, предпочтение отдается сегменту, проходящему через вершины, которые лежат на одной вертикальной прямой с вершинами v, такими, что $p_v \neq 0$.

Для каждой трассы второй группы есть две наиболее предпочтительные конфигурации, примеры t_1, t_2 которых показаны на рис. 5.18,*б*. Трассы (например, t_3), расположенные между t_1 и t_2, также имеют минимальную длину, но на один поворот больше, в то время как трассы, расположенные левее или правее, могут иметь всего один поворот, но бóльшую длину (трассы t_4—t_7). Конфигурация трассы определяется с учетом перечисленных факторов. Если существует несколько допустимых конфигураций, лучшая выбирается по тем же правилам, которые применялись для трасс первой группы.

Все неразмещенные трассы реализуются с использованием произвольных конфигураций. При этом последовательно в каждой промежуточной линейке выбирается вершина, через которую проводится трасса, а правила выбора очередной вершины совпадают с правилами выбора вертикального сегмента трассы первой группы.

На следующем этапе определяются точные значения координат блоков и межблочных транзитных трасс на кристалле. Эта операция выполняется с использованием модели T последовательным размещением блоков в линейках таким образом, чтобы левая граница каждого блока совпадала с левой границей ячейки на кристалле.

Завершающим этапом является размещение элементов и транзитных трасс внутри каждого блока. Начальное размещение выполняется последовательным алгоритмом по критерию, совпадающему с используемым на предшествующих этапах. Для оценки критерия оптимальности необходимо знать расположение трасс, соединяющих элементы в рассматриваемой линейке и элементы, размещенные в соседних линейках. Положение трассы характери-

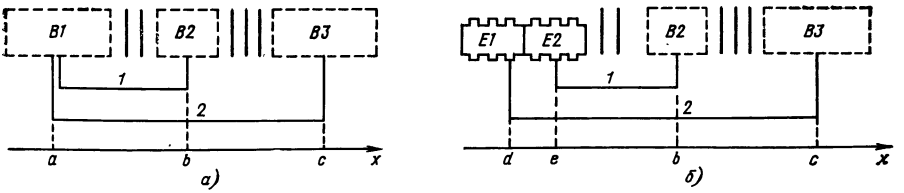


Рис. 5.19. Изменение интервалов цепей после размещения элементов в блоке

зуется интервалом горизонтальной прямой, левая граница которого совпадает с минимальной координатой вывода, инцидентного этой цепи, а правая — с максимальной. В дальнейшем будем пользоваться термином интервал цепи. В ходе размещения элементов интервалы цепей уточняются. До начала размещения элементов интервалы цепей определяются координатами центров блоков. Так, цепи *1* и *2* характеризуются интервалами $[a, b]$ и $[a, c]$ соответственно (рис. 5.19,*а*) После размещения элементов блока B1 интервалы изменяются ($[d, c]$ и $[e, b]$ на рис. 5.19,*б*). Таким образом, при размещении элементов в очередном блоке интервалы цепей полностью характеризуют положение уже размещенных элементов. Одновременно с элементами внутри блока размещаются и определенные на предшествующем этапе транзитные трассы, которые назначаются на конкретные проходы в элементах.

После начального размещения используется итерационный алгоритм парных перестановок соседних элементов блока. Рассмотрение пар только соседних элементов приводит к упрощению определения изменения критерия. Особенность алгоритма состоит в том, что при определении лучшего расположения соседних элементов выбирается лучшее назначение транзитных трасс на проходы этих элементов. Если элемент содержит логически инвариантные выводы или группы таких выводов, то может применяться дополнительная процедура оптимизации, в ходе которой осуществляется назначение соответствующих цепей на инвариантные выводы [179].

5.6. МЕТОДИКА ТРАССИРОВКИ СОЕДИНЕНИЙ

Процесс трассировки соединений на кристалле состоит в последовательной трассировке цепей в каждом канале и в области линейки, не занятой элементами. Так как выводы элементов в рассматриваемом типе конструкции БМК дублируются на верхней и нижней сторонах, то распределение элементов по линейкам еще не определяет однозначного распределения трасс и их сегментов по каналам. Это учитывается в процессе трассировки, и сегменты цепей, не реализованные в очередном канале, переносятся (когда это возможно) в соседний канал. Например, цепь может быть реализована, как показано на рис. 5.20. (Выводы *б*

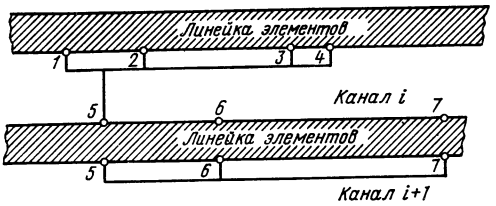


Рис. 5.20. Перенос нереализованных сегментов в следующий канал

и 7 соединить с реализованной частью цепи в канале i не удалось, и поэтому к ним подводится трасса в канале $i+1$.)

В связи с возможностью реализации некоторых соединений в одном из двух каналов в ходе трассировки целесообразно разделить все соединения на две группы: *приоритетные* и *неприоритетные*. К первой группе относятся соединения, у которых имеется вывод на верхней стороне канала. Остальные — ко второй группе.

Перед трассировкой канала формируется схема его соединений. Для этого из описания всей схемы относительно блоков выделяется подсхема соединений между блоками, расположенными вдоль рассматриваемого канала. Затем в полученную подсхему подставляются схемы соединений между элементами блоков и транзитные соединения. Трассировка соединений в канале состоит из трех этапов. На первом этапе выполняется построение трасс в ограниченном классе конфигураций. На втором этапе для ранее не реализованных цепей делается попытка построения трасс произвольной конфигурации. Наконец, на третьем этапе выполняется итерационная трассировка соединений со снятием ранее построенных и их последующей перетрассировкой. Рассмотрим кратко каждый этап.

Ограничение класса возможных конфигураций на первом этапе заключается в том, что трасса каждого парного соединения является либо линейным сегментом, либо содержит единственный горизонтальный отрезок, проекцию которого на ось X назовем интервалом соединения. (Будем считать, что интервал линейного сегмента содержит одну точку.) Перед началом трассировки каждая цепь разбивается на парные соединения. Для образования парных соединений выводы цепи упорядочиваются по возрастанию координаты X. Тогда каждые два соседних вывода в упорядоченной последовательности образуют парное соединение. Например, цепь на рис. 5.21 разбивается на три парных соединения между 1-м и 2-м выводами, между 2-м и 3-м и между 3-м и 4-м. Допустимые типы трасс парных соединений показаны на рис. 5.22. К первому типу (рис. 5.22,*а*) относятся трассы, полностью расположенные в «вертикальном» слое. Ко второму (рис. 5.22,*б*) — трассы в «горизонтальном» слое. Третий (рис. 5.22,*в*) и четвертый (рис. 5.22,*г*) типы составляют соответственно трассы с одним и двумя межслойными переходами.

С целью экономного использования при трассировке рессурса канала для соединений, интервал которых не содержит выводов

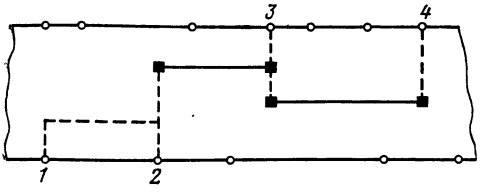


Рис. 5.21. Образование парных соединений

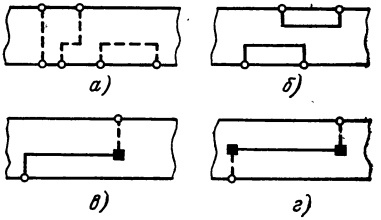


Рис. 5.22. Допустимые типы трасс парных соединений

других соединений хотя бы на одной из сторон (верхней или нижней) канала, используются трассы первого типа. Трассировка начинается с построения трасс первого типа (рис. 5.23,*а*). При этом допускается образование «параллельных трасс» (на рис. 5.23,*а* такие трассы соединяют выводы *2*, *3* и *1*, *4*). «Параллельные трассы» возникают, когда интервал одного соединения содержит интервал другой цепи, у которой все выводы расположены на одной стороне канала. В ходе дальнейшей трассировки все горизонтальные отрезки трасс проводятся в «горизонтальном слое» (рис. 5.23,*б*). Причем, если горизонтальный отрезок располагается на магистрали, ближайшей к стороне канала, то в зависимости от положения выводов могут использоваться трассы второго или третьего типа. Все соединения, не реализованные трассами первого типа, трассируются по классической схеме построения трасс в канале, рассмотренной в гл. 2. Если в результате канальной трассировки оказались нереализованными некоторые приоритетные соединения, то делается попытка их реализации за счет удаления минимального числа неприоритетных трасс.

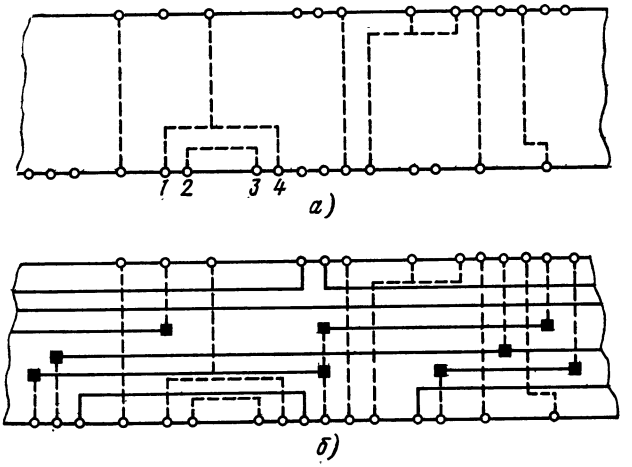


Рис. 5.23. Пример трассировки канала

Второй этап трассировки представляет собой реализацию *лабиринтного алгоритма.* В его основу положен метод поиска пути в лабиринте [76]. Так же, как и волновой метод, лабиринтный гарантирует нахождение пути между двумя точками, если путь существует. Принципиальным отличием данного метода от волнового является другая стратегия поиска цели, называемая *стратегией одностороннего ветвления* [6]. В рамках этой стратегии можно учитывать координаты цели и вводить ряд дополнительных эвристик, позволяющих значительно ускорить процесс трассировки.

Моделью канала является традиционное дискретное рабочее поле (ДРП), представляющее собой ортогональную сетку, шаг которой равен минимально допустимому расстоянию между двумя соседними проводниками. Каждая ячейка ДРП перед процессом трассировки очередной цепи может находиться в одном из следующих состояний: оба слоя заняты (ячейка, запрещенная для трассировки, межслойный переход, трассы других цепей); занят один из слоев; свободны оба слоя. Для построения очередной трассы некоторые ячейки помечаются как целевые, а ячейки, пройденные в ходе поиска цели, индексами (*путевыми координатами*), позволяющими для каждой ячейки определить предшествующую.

Процесс трассировки представляет собой последовательное построение трасс каждой цепи. Для трассировки очередной цепи она представляется совокупностью парных соединений между ее выводами. Эта задача решается с помощью алгоритма Прима [7] построения кратчайшего связывающего дерева на множестве выводов элементов, соединяемых цепью. Особенностью рассматриваемого этапа является специальная метрика

$$\rho_{(i,j)} = \begin{cases} |x_i - x_j| + |y_i - y_j|, & \text{если } x_i \neq x_j \text{ и } y_i \neq y_j; \\ |y_i - y_j|/k, & \text{если } x_i = x_j, \end{cases}$$

где x, y — координаты точек; k — коэффициент предпочтения ЛС.

В ходе трассировки парных соединений осуществляется наращивание реализованной части цепи. Для первого соединения один вывод включается в реализованную часть, а другой считается целью. Для всех последующих соединений также один вывод принадлежит реализованной части, а другой — цели. Трассировка соединения состоит в построении пути от реализованной части к цели и включает два этапа: поиск цели; проведение пути.

Основные параметры алгоритма определяются первым этапом, который подробно описывается ниже. Пока отметим лишь, что если цель достигнута, то, как и в волновом алгоритме, каждая пройденная ячейка содержит путевую координату. Поэтому на втором шаге, начиная с целевой ячейки, можно с помощью путевых координат легко восстановить искомый путь. Особенностью данного этапа является автоматическое удаление избыточных переходов, расположенных в соседних ячейках ДРП.

Трудоемкость поиска цели главным образом определяется числом пройденных в ходе поиска ячеек. Если между исходной точкой s и целевой t сущест-

вует кратчайший путь $p(s, t)$, то минимальное число пройденных ячеек при поиске цели равно длине $L(p)$ пути p. Поэтому более быстрым будет тот алгоритм, который в процессе поиска анализирует меньше ячеек, не принадлежащих p. В частности, медленная работа волнового алгоритма объясняется тем, что при поиске пути p на свободном поле число пройденных ячеек равно $2L^2(p)$.

В дальнейшем ячейку, которая с относительно высокой вероятностью может принадлежать искомому пути, будем называть *перспективной*. Для сокращения трудоемкости алгоритма важно в процессе поиска уметь из числа пройденных ячеек определять наиболее перспективную и переходить из нее в новую, также наиболее перспективную ячейку. Для оценки перспективности ячейки введем эвристическую функцию

$$F(i, j) = F(i) + K_1 \alpha_{ij} + K_2 (1 - \alpha_{ij}) + K_3 \beta_{ij} - K_4 \gamma_{ij} + K_5 (1 - \gamma_{ij})$$

где $F(i, j)$ — значение функции в ячейке j при переходе в нее из ячейки i; $F(i)$ — значение функции в ранее пройденной ячейке i;

$$\alpha_{ij} = \begin{cases} 1, \text{ если движение из ячейки } i \text{ в } j \text{ совпадает с преимущественным направлением слоя;} \\ 0 \text{ — в противном случае;} \end{cases}$$

$$\beta_{ij} = \begin{cases} 1, \text{ если при движении из } i \text{ в } j \text{ делается межслойный переход;} \\ 0 \text{ — в противном случае;} \end{cases}$$

$$\gamma_{ij} = \begin{cases} 1, \text{ если в результате перехода из ячейки } i \text{ в } j \text{ осуществляется приближение к цели;} \\ 0 \text{ — в противном случае;} \end{cases}$$

K_1—K_5 — положительные весовые коэффициенты, задаваемые на входе: K_1 — штраф за длину трассы в преимущественном направлении; K_2 — штраф за длину трассы в непреимущественном направлении; K_3 — штраф за межслойный переход; K_4 — поощрение за приближение к цели; K_5 — штраф за удаление от цели.

В соответствии со специализацией слоев вводится понятие преимущественного направления трасс в слое. Ясно, что для трассировки со специализацией слоев должно выполняться условие $K_1 + K_3 < K_2$. Поскольку при переходе в следующую точку интерес представляет движение с минимальным значением функции $F(i, j)$, то под приближением к цели подразумевается движение в преимущественном направлении с уменьшением расстояния до цели. Ясно, что такое движение дает минимально возможное значение функции $F(j)$.

Укрупненно процесс поиска цели можно представить следующей последовательностью шагов:

1. Для каждой ячейки i реализованной части цепи определить $F(i)$ как расстояние до цели и пометить эти ячейки как пройденные.

2. Из числа пройденных найти наиболее перспективную ячейку i (имеющую минимальное значение функции F). Если пройденных ячеек нет, путь не существует.

3. Найти наиболее перспективного соседа j ячейки i. Положить $F(j) = F(i, j)$ и пометить ячейку j как пройденную. Если соседа нет, то исключить ячейку i из числа пройденных, пометить ее как запрещенную и перейти к шагу 2.

4. Если j — целевая ячейка, поиск окончен.

5. Если переход из i в j не является приближением к цели, то перейти к шагу 2.

6. Положить $i=j$ и перейти к соседу j ячейки i в направлении приближения к цели. Если ячейка j занята (запрещена), перейти к шагу 2. В противном случае перейти к шагу 4.

Описанный алгоритм в процессе поиска цели на свободном поле проходит минимальное число ячеек. Пример представлен на рис. 5.24,*а*, где стрелки отображают путевые координаты, а цифры — порядок присвоения этих координат. Если же поле содержит препятствия, то поведение алгоритма зависит от соотношения между коэффициентами K_1—K_5. Например, если $K_4 \leqslant K_1$, то значение функции F в каждой последующей ячейке больше, чем в предыдущей, и после встречи с препятствием будет осуществлен переход (на шаге 2) в пройденную ячейку с минимальным значением функции F. Такая стратегия приводит к просмотру большого числа ячеек и является нецелесообразной.

С точки зрения экономии времени поиска целесообразно при встрече с препятствием попытаться его обойти, не сильно удаляясь от цели. Рассмотрим влияние коэффициентов функции F на выбор следующей ячейки при встрече с препятствием. Для того чтобы был возможен обход препятствия, значение F в ячейке j встречи с препятствием должно быть меньше, чем в предыдущей ячейке. Это условие обеспечивается при приближении к цели, когда $K_4 > K_1$. Тогда возможно движение не к цели (вдоль препятствия) до тех пор, пока значение F в последней достигнутой ячейке не превысит значение F в ячейке i, предшествующей j. Понятно, что при фиксированных значениях коэффициентов K_1, K_2, K_3

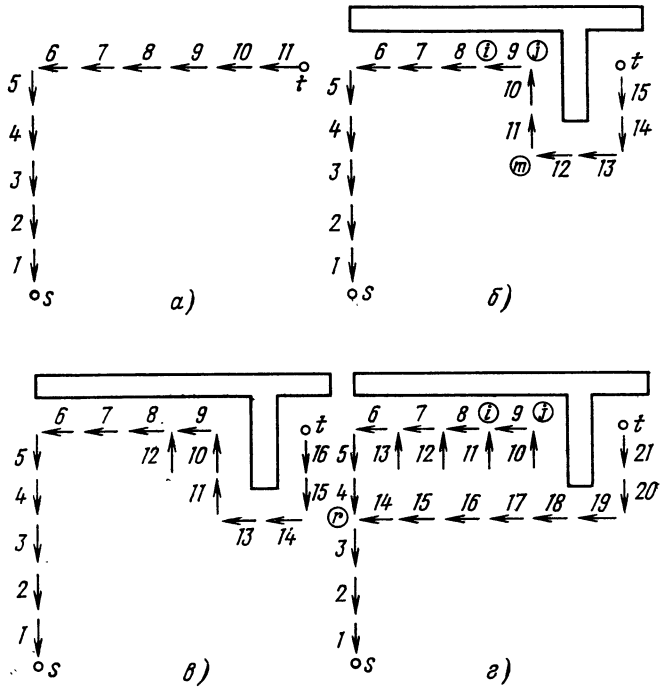


Рис. 5.24. Примеры поиска трассы лабиринтным алгоритмом

к увеличению длины обхода препятствия приводит увеличение K_4 и уменьшение K_5. Например, если соотношение коэффициентов таково, что значение функции F в ячейке m (рис. 5.24,*б*) меньше $F(i)$, то строится трасса, обходящая препятствие, при этом число пройденных ячеек минимально. Когда $F(m)>F(i)$, число пройденных ячеек увеличивается (рис. 5.24,*в*). Наконец, возможно такое соотношение коэффициентов, когда один шаг вдоль препятствия приводит к значению функции F, равному $F(r)$, где r — ячейка, достигнутая ранее i. Пример последней ситуации приведен на рис. 5.24,*г*, где показано построение трассы минимальной длины.

Для улучшения результатов трассировки, полученных лабиринтным алгоритмом, последовательным просмотром всех трасс устраняются избыточные межслойные переходы. Основным недостатком лабиринтного алгоритма является недостаточно высокий процент проведенных соединений. Главным образом этот недостаток обусловлен последовательным построением трасс, при котором положение реализованных соединений не изменяется. В то же время конструктор в ходе проведения одной трассы часто корректирует другие. Алгоритм, используемый на третьем этапе трассировки, представляет собой реализации такого режима автоматического построения трасс [149]. Эффективность подобного подхода в случае проектирования многослойных печатных плат показана в [67].

Трассировка канала организуется в виде итерационного процесса. На каждой итерации последовательно осуществляется построение всех соединений, оказавшихся нереализованными после предыдущей итерации. В ходе построения каждой цепи реализуются все ее соединения. При этом допускается удаление ранее проведенных сегментов других цепей. Если на каждой итерации сохранять один и тот же порядок построения цепей, то цепи, которые трассируются позже, имеют бóльшую вероятность полной реализации. Для устранения этого недостатка порядок трассировки цепей генерируется с помощью датчика случайных чисел. Итерационный процесс завершается при построении всех трасс либо после выполнения заданного на входе числа последовательных итераций без увеличения показателя Q качества трассировки. При этом $Q=h_1t_1+h_2t_2$, где t_1 — число реализованных приоритетных соединений, t_2 — число реализованных неприоритетных соединений; h_1, h_2 — весовые коэффициенты ($h_1>h_2$). В последнем случае окончательное решение строится путем дополнения запомненного результата лучшей итерации максимально возможным числом соединений без снятия проведенных ранее.

В основу алгоритма трассировки цепи положен волновой метод. В качестве источника волны случайным образом выбирается один из фрагментов (связных частей) цепи. Остальные фрагменты — стоки. В ходе распространения волна проходит по свободным дискретам модели платы и по дискретам, содержащим трассы других цепей. Распространение волны прекращается после достижения любого из стоков. В результате построения пути реали-

зуется новый сегмент (объединяющий фрагмент — источник и фрагмент — сток) трассируемой цепи. Если построенный путь пересекает другие трассы, то выполняется удаление пересекаемых сегментов. Затем заново выбирается источник и процесс повторяется до тех пор, пока цепь не будет реализована полностью.

Для учета широкого набора факторов, влияющих на качество трассировки, а также для обеспечения простоты адаптации алгоритма к различным технологическим ограничениям используется комплексный критерий оптимальности (стоимость пути), представляющий собой взвешенную сумму частных показателей

$$F = \lambda_1 f_1 + \lambda_2 f_2 + \ldots + \lambda_9 f_9,$$

где λ_i — весовые коэффициенты; f_1 — число удаляемых приоритетных сегментов; f_2 — число удаляемых неприоритетных сегментов; f_3 — число межслойных переходов; f_4 — число поворотов; f_5 — длина; f_6 — число пар межслойных переходов, расстояние между которыми равно длине диагонали дискрета; f_7 — длина наложения трасс в различных слоях; f_8 — длина горизонтальных (вертикальных) отрезков, расположенных в «вертикальном» («горизонтальном») слое; f_9 — число пар межслойных переходов, расположенных в соседних дискретах.

Весовые коэффициенты λ_i нормируются таким образом, чтобы критерий F на любом пути не превышал максимальной величины F_M, определяемой разрядностью машинного слова. Если теперь положить для некоторого i $\lambda_i = F_M$ и дискреты, в которых величина F не меньше F_M, не включать во фронт волны, то на построенном пути всегда будет $F < F_M$. Следовательно, частный показатель i в этом случае превращается в ограничение. Особенностью описываемого алгоритма является изменение критерия F при переходе от итерации к итерации. А именно, с целью повышения вероятности получения полной реализации всех трасс величины λ_1, λ_2 увеличиваются.

Моделью канала является трехмерная решетка, которая представлена в ЭВМ двумя трехмерными массивами A, B, описывающими дискретное рабочее поле. Массив A отражает промежуточные и конечные результаты трассировки. Элемент $A(i, j, k)$ равен номеру цепи, проходящей через дискрет с координатами (i, j) на слое k. Переход трассы из одного слоя в другой фиксируется в массиве знаком (—); $A(i, j, k) = 0$, если в соответствующем дискрете нет трассы.

Для реализации процесса распространения волны и проведения пути используется массив B. Каждый элемент этого массива отображает состояние соответствующей ячейки дискретного рабочего поля. При кодировании состояний ячеек предусматривается возможность прохождения волны по ячейкам, занятым трассами, которые были построены на предшествующих шагах. Эффективность описанной методики трассировки исследовалась путем сравнения с известными, в частности с волновым алгоритмом. По числу реализованных трасс волновой алгоритм, в среднем, давал резуль-

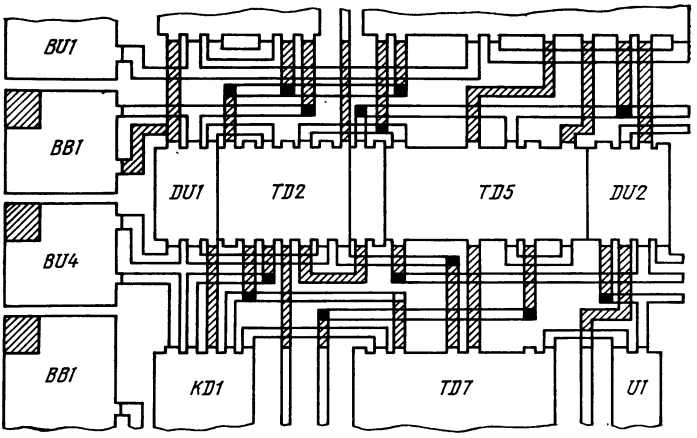


Рис. 5.25. Фрагмент топологии, разработанный с помощью подсистемы АРТИС

таты хуже на 17—23%. Вместе с тем трассы, построенные волновым алгоритмом, в основном, имели более сложную форму, что затрудняло этап корректировки. Преимущественное использование трасс простой конфигурации в подсистеме АРТИС иллюстрируется фрагментами топологии на рис. 5.25.

6. СИСТЕМА АСП-6М АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ МАТРИЧНЫХ БОЛЬШИХ И СВЕРХБОЛЬШИХ ИНТЕГРАЛЬНЫХ МИКРОСХЕМ

6.1. ОБЩАЯ ХАРАКТЕРИСТИКА СИСТЕМЫ АСП-6М

САПР АСП-6М является сквозной иерархической системой проектирования матричных БИС и СБИС линейчатого типа, проектируемых по КМОП-технологии, конструкция которых описана в гл. 1, 5. В настоящее время САПР АСП-6М включает следующие подсистемы (рис. 6.1):

1. Транслятор с языка АСП-6 иерархического описания *схемы электрической* (СхЭ), представленной в базисе типовых логических элементов (*базовых элементов* — БЭ) и тестов моделирования.
2. Иерархическое логическое моделирование.
3. Схемотехническое моделирование БЭ и фрагментов матричных БИС.
4. Интерактивное проектирование топологии БМК и БЭ.
5. Проектирование топологии матричных БИС:

а) автоматическое; б) интерактивное (в том числе коррекция топологии после этапа 5а).

6. Верификация топологии матричных БИС.

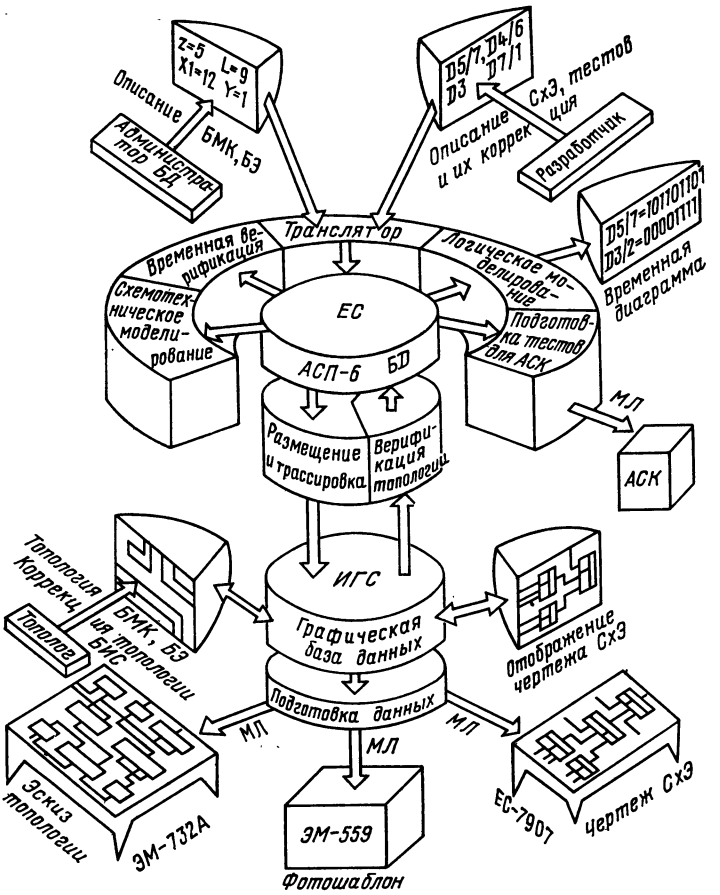


Рис. 6.1. Структурная схема САПР АСП-6М

7. Временная верификация матричных БИС.
8. Подготовка тестов для контроля матричных БИС.
9. Изготовление документации и подготовка данных для генераторов изображения по изготовлению фотошаблонов.
10. СУБД АСП-6БД.

САПР АСП-6М базируется на ЕС ЭВМ (не ниже модели ЕС-1033 с ОЗУ не менее 2 мгбайт) и *интерактивно-графической системы* (ИГС) «Кулон-1» (15УТ-4-017) [150, 151, 169] на основе мини-ЭВМ «Электроника 100-25». Связь ЕС ЭВМ с ИГС осуществляется через магнитную ленту. Ведутся работы по созданию локальной вычислительной сети САПР на базе ЕС ЭВМ, ИГС со скоростью обмена 10 мгБод. Подсистемы 1, 2, 3, 5а, 6, 7, 8, 10 реализованы на ЕС ЭВМ, подсистемы 4, 5б, 9 — на ИГС. Все подсистемы объединены в сквозную САПР на базе единых операционной системы ОС 7.0, базы данных АСП-6БД, иерархического язы-

ка описания СхЭ и тестов моделирования, промежуточных языков и форматов данных. Ввод, выдача и обработка данных в САПР АСП-6М производится в мультипрограммном режиме с использованием цифровых (на ЕС ЭВМ, ИГС) и графических (на ИГС) дисплеев.

Язык описания СхЭ — иерархический с произвольной глубиной. В процессе трансляции возможно раскрытие описания до любого уровня иерархии: для логического моделирования, временной верификации, подготовки тестов и данных для генератора изображений — до нулевого уровня (БЭ); при проектировании, верификации топологии, выпуске документации — до необходимого уровня. Подсистема логического моделирования и временной верификации обеспечивает иерархическое моделирование: моделирование матричных БИС или отдельных блоков любого уровня иерархии. Подсистемы проектирования и верификации топологии предусматривают возможность иерархического проектирования в автоматическом и интерактивном режимах и верификации блоков любого иерархического уровня (блоков 1-го уровня, состоящих из БЭ, блоков 2-го уровня, состоящих из блоков 1-го уровня и т. д.) или матричных БИС в целом при сложности последней не более 10^4 логических элементов ($\sim 10^5$ транзисторов) и производительности ЕС ЭВМ порядка 10^6—10^7 опер./с. Предусмотрена возможность применения блоков, спроектированных ранее или в текущей разработке на любом иерархическом уровне. Коррекция текстовой и графической информации производится на цифровых и графических дисплеях; выпуск текстовой документации на АЦПУ; изготовление эскизов топологии на графопостроителях ЭМ-7022 (входящих в состав ИГС) и ЭМ-732А; изготовление чертежей и схемной документации на графопостроителе ЕС-7907 («Дигиграф»). Для изготовления промежуточных фотошаблонов подготавливается информация, управляющая генераторами изображений ЭМ-559, ЭМ-5009; для функционального контроля — информация, управляющая автоматизированной системой контроля «Вестник».

Проектированию матричных БИС предшествует этап разработки БМК, включающий выбор и оптимизацию параметров *p*-, *n*-транзисторов в линейках и периферийных элементах, конструкции БМК (число и размер линеек, число магистралей в каналах для трассировки соединений, ширина проводников, размеры межслойных переходов, координаты линеек и периферийных элементов, шаг опорной сетки, технологический допуск между топологическими элементами, сопротивления и емкости металлизации слоев межсоединений и т. д.), этап проектирования топологии БМК и др. Топология слоев БМК проектируется и записывается в базу данных ИГС. Математическое описание конструкции БМК, предназначенное для автоматического проектирования топологии и ее верификации, записывается в базу данных АСП-6БД.

На основе БМК разрабатывается с использованием системы схемотехнического моделирования библиотека БЭ, содержащая более сотни БЭ на каждый тип БМК. Библиотека включает БЭ

типа И—ИЛИ—НЕ, И—НЕ, ИЛИ—НЕ, *D*-триггер, *RS*-триггер, *DR*-триггер, входные, выходные и промежуточные усилители, расположенные по периферии БМК (периферийные элементы) и другие типы сложностью до нескольких десятков транзисторов. Объединение транзисторов БЭ для реализации заданной функции производится в двух слоях металлизации и слое переходов. Топология БЭ проектируется и записывается в базу данных ИГС. С целью создания лучших условий для проектирования топологии матричных БИС в целом топология каждого БЭ представляется в виде прямоугольника. Математическое описание каждого БЭ записывается в библиотеку БЭ базы данных АСП-6БД и включает:

описание функционирования БЭ в 7-значной логике на ассемблере ЕС ЭВМ;

описание электрических характеристик: зависимость (в виде таблиц) задержки сигнала на БЭ (при переходе из 0 в 1 и из 1 в 0) от емкостной нагрузки при номинальном значении питающего напряжения и отклонениях его на $\pm 0{,}5$ В при различных значениях температуры окружающей среды;

описание конструкции БЭ: координаты контактов БЭ относительно левого нижнего угла (в единицах опорной сетки); координаты «земли», «питания»; запрещенные области рабочего поля под БЭ; длина БЭ в единицах опорной сетки и другие данные.

База данных АСП-6БД обеспечивает оперативное пополнение библиотеки новыми описаниями БМК и БЭ. САПР АСП-6М на основе указанных описаний настраивается на то или иное описание БМК и БЭ согласно имени типа БМК, содержащегося во входной информации.

Разработка БМК и БЭ производится исходя из технологических, схемотехнических, конструкторских требований, а также создания условий для автоматизации проектирования топологии матричных БИС в целом.

6.2. ЯЗЫК ИЕРАРХИЧЕСКОГО ОПИСАНИЯ СхЭ И ТЕСТОВ АСП-6

Исходной информацией для проектирования матричной БИС является иерархическое описание СхЭ [152] и тестов [153] на языке АСП-6, имя матричной БИС и тип базового кристалла, а также математического описания конструкции базового кристалла и БЭ, находящихся в базе данных ЕС ЭВМ и ИГС. Язык описания СхЭ АСП-6 разработан на основании анализа входных языков САПР БИС, СБИС, микросборок и печатных плат [148, 154—157] и опыта создания и эксплуатации систем логического моделирования АСП-55 [158], проектирования и контроля топологии матричных БИС АСП-62 [159, 160], микросборок и печатных плат АСП-53 [161]. От аналогичных разработок [148, 154—158] язык АСП-6 отличается многоцелевым назначением, широким кругом пользователей разной степени подготовленности, ориентацией на различные подсистемы САПР АСП-6М; большой сложностью опи-

сываемых схем и различной степенью регулярности их структуры. Основная особенность языка — ориентация на иерархическое описание матричных БИС высокой сложности (10^5 БЭ и более), возможность сокращенного описания матричных БИС, использующего регулярность цифровых устройств, возможность описания блоков матричных БИС различных уровней либо списком цепей либо перечнем сигналов (в пределах одного блока допускается описание одним видом). На нижних уровнях иерархического описания удобно пользоваться списком цепей, на более высоких — перечнем сигналов.

Язык АСП-6 организован в виде компактного функционально-полного ядра, окруженного множеством независимых средств сокращения описаний, что позволяет пользователям выбирать наиболее подходящее подмножество языка. Матричные БИС могут иметь многоуровневую иерархическую структуру, позволяющую описывать блоки i-го уровня через описания блоков i—1, i—2, ..., 0 уровней, где блок 0-го уровня — БЭ. Например, блок 2-го уровня может содержать счетчики, регистры, являющиеся блоками 1-го уровня, и отдельные БЭ. На различных уровнях иерархии блоки могут быть как функционально-законченными (счетчик, регистр, сумматор, вычислитель, память и т. д.), так и не являться таковыми. Блок любого уровня идентифицируется своим именем, под которым в базе данных хранится его описание. Описание блока i-го уровня включает: список цепей или перечень сигналов, спецификацию; площадь (площадь БЭ), занимаемую блоком. Кроме того, описание блока может содержать размещение входящих в него блоков низшего уровня и внешних сигналов, их координаты, электрические характеристики, описание чертежа логической схемы блока (в том числе, описание чертежа БЭ) и другие параметры.

Цепь блока i-го уровня — это совокупность электрически связанных контактов блоков низшего уровня (далее просто элементов), и, возможно, периферийных элементов (ПЭ). Цепи могут иметь имена. Описание схемы блока в виде перечня сигналов представляет собой описание входных и выходных сигналов каждого элемента (т. е. блока низшего уровня), входящих в блок i-го уровня. Спецификация — соответствие имени блока его типу. Площадь каждого блока вычисляется автоматически по описанию блока, спецификации и площади, занимаемой входящими в него элементами. Внешние цепи, т. е. цепи, содержащие внешние сигналы, обязаны иметь имена. Внешние контакты блока идентифицируются своими логическими и физическими именами. Логическое имя внешнего контакта — это имя цепи, в которую он входит. Физические имена внешних контактов определены только для матричных БИС в целом и задаются номером внешнего контакта матричных БИС.

В язык включены средства сокращения описаний, основанные на принципе умолчания, понятии массива и выделения общей части однотипных конструкций. Для повышения наглядности и сокращения описания схем блоков разрешается опускать: внешние

контакты, если логическая схема блока описывается списком цепей и внешние контакты перечислены в описании внешних цепей; типы и физические имена внешних контактов; имена внутренних цепей; внешние контакты в описании внешних цепей, если они описаны в разделе внешних контактов; описания сигналов блоков, логические имена которых совпадают с именами цепей.

Эффективным применением принципа умолчания является описание блока, в котором все контакты элементов, образующие одну цепь, имеют одинаковые логические имена, совпадающие с именем цепи. Введено понятие «Массив» — для сокращения описаний устройств, содержащих фрагменты с регулярной структурой, понятие «Массив чисел» — для описания последовательности целых чисел, образующих арифметическую прогрессию, «Массив имен» — для задания последовательности имен, начальные части (префиксы) которых одинаковы, а конечные части (суффиксы) — это числа, образующие арифметическую прогрессию. Конструкции «Массив чисел», «Массив имен» используются при описании: внешних контактов для замены нескольких описаний одним; цепей для замены нескольких описаний контактов одним описанием массива контактов; сигналов блока для замены нескольких описаний сигналов одним описанием массива сигналов и т. д.

Помимо удобства и сокращения описания иерархическое представление схем является предпосылкой уменьшения размерности задач размещения БЭ, блоков, трассировки цепей, способствует улучшению качества проектирования топологии за счет локализации элементов блока в рабочем поле БИС и уменьшения длины связей между элементами. Действительно, если блок i-го уровня является функционально-законченным образованием, производить перекомпоновку, переразмещение элементов (блоков i—1, i—2 и других уровней) между блоками i-го уровня при проектировании блока $(i+1)$-го уровня не имеет смысла. Например, перестановка элементов, принадлежащих счетчику, с элементами, принадлежащими регистру, ухудшает качество проектирования.

Приведем формальное описание языка АСП-6, где под блоком будем понимать также и матричные БИС в целом.

Синтаксис конструкций ядра языка:

⟨Описание блока⟩ :: =⟨Описание структуры блока ⟩|⟨Описание структуры элемента⟩⟨Описание блока⟩

⟨Описание структуры элемента⟩ :: =⟨Описание структуры блока⟩

⟨Описание структуры блока⟩ :: =⟨Описание списком цепей⟩|⟨Описание перечнем сигналов⟩

⟨Описание списком цепей⟩ :: =⟨Название устройства⟩ ⟨Раздел внешних контактов⟩⟨Раздел элементов⟩⟨Раздел цепей⟩ КОНЕЦ*

⟨Описание перечнем сигналов⟩ :: =⟨Название устройства⟩ ⟨Раздел внешних контактов⟩⟨Раздел сигналов⟩КОНЕЦ*

⟨Раздел внешних контактов⟩ :: =ВНЕШНИЕ* ⟨Список внешних контактов⟩

⟨Список внешних контактов⟩ :: =⟨Описание внешнего контакта⟩; |⟨Описание внешнего контакта⟩ ; ⟨Список внешних контактов⟩

⟨Описание внешнего контакта⟩ :: =⟨Логическое имя⟩=⟨Физическое имя⟩.⟨Тип внешнего контакта⟩

⟨Физическое имя⟩ :: =⟨Имя разъема⟩/⟨Номер контакта⟩|⟨Номер контакта⟩

⟨Раздел элементов⟩ :: =ЭЛЕМЕНТЫ*⟨Список элементов⟩

⟨Список элементов⟩ :: =⟨Описание элемента⟩;|⟨Описание элемента⟩;⟨Список элементов⟩

⟨Описание элемента⟩ :: =⟨Имя элемента⟩.⟨Тип элемента⟩.⟨Размещение элемента⟩

⟨Размещение элемента⟩ :: =РК=⟨Координата *X*⟩/⟨Координата *Y*⟩|РЛ==⟨Номер линейки⟩/⟨Координата на линейке⟩

⟨Раздел цепей⟩ :: =ЦЕПИ* ⟨Список цепей⟩

⟨Список цепей⟩ :: =⟨Описание цепи⟩|⟨Описание цепи⟩⟨Список цепей⟩

⟨Описание цепи⟩ :: =⟨Имя цепи⟩:⟨Список контактов⟩

⟨Список контактов⟩ :: =⟨Описание контакта⟩|⟨Описание контакта⟩.⟨Список контактов⟩

⟨Описание контакта⟩ :: =⟨Внешний контакт⟩|⟨Контакт элемента⟩

⟨Внешний контакт⟩ :: =⟨Физическое имя⟩

⟨Контакт элемента⟩ :: =⟨Имя элемента⟩/⟨Имя внешнего контакта⟩

⟨Имя внешнего контакта⟩ :: =⟨Физическое имя⟩|⟨Логическое имя⟩

⟨Раздел сигналов⟩ :: =СИГНАЛЫ* ⟨Список сигналов⟩

⟨Список сигналов⟩ :: =⟨Описание сигналов элемента⟩|⟨Описание сигналов элемента⟩;⟨Список сигналов⟩

⟨Описание сигналов элемента⟩ :: =⟨Описание элемента⟩.⟨Список сигналов элемента⟩

⟨Список сигналов элемента⟩ :: =⟨Описание сигнала⟩|⟨Описание сигнала⟩.⟨Список сигналов элемента⟩

⟨Описание сигнала⟩ :: =⟨Имя внешнего контакта⟩=⟨Имя цепи⟩

Синтаксис конструкций массивов:

⟨Массив чисел⟩ :: =(⟨Первое число⟩:⟨Последнее число⟩, ⟨Шаг⟩)

⟨Массив имен⟩ :: =⟨Начальная часть имени⟩⟨Массив чисел⟩

⟨Описание массива внешних контактов⟩ :: =⟨Массив логических имен⟩==⟨Массив физических имен⟩.⟨Назначение⟩

⟨Массив физических имен⟩ :: =⟨Массив номеров контактов⟩

⟨Описание массива элементов⟩ :: =⟨Массив имен элементов⟩.⟨Тип элемента⟩

⟨Описание массива цепей⟩ :: =⟨Массив имен цепей⟩:⟨Список массивов контактов⟩;

⟨Список массивов контактов⟩ :: =⟨Описание массива контактов⟩|⟨Описание массива контактов⟩.⟨Список массивов контактов⟩

⟨Описание массива контактов⟩ :: =⟨Массив внешних контактов⟩|⟨Массив контактов элементов⟩

⟨Массив внешних контактов⟩ :: =⟨Массив физических имен⟩

⟨Массив контактов элементов⟩ :: =⟨Имя элемента⟩/⟨Массив имен внешних контактов⟩|⟨Массив элементов⟩/⟨Имя внешнего контакта⟩

⟨Массив имен внешних контактов⟩ :: =⟨Массив физических имен⟩|⟨Массив логических имен⟩

⟨Описание массива сигналов элементов⟩ :: =⟨Описание массива элементов⟩.⟨Список массивов сигналов⟩

⟨Список массивов сигналов⟩ :: =⟨Описание сигналов массива элементов⟩|⟨Описание сигналов массива элементов⟩.⟨Список массивов сигналов⟩

⟨Описание сигналов массива элементов⟩ :: =⟨Имя внешнего контакта⟩=⟨Массив имен цепей⟩

⟨Список массива сигналов⟩ :: =⟨Массив имен внешних контактов⟩=⟨Массив имен цепей⟩

⟨Описание группы внешних контактов⟩ :: =⟨Перечень внешних контактов⟩.⟨Тип внешнего контакта⟩

⟨Перечень внешних контактов⟩ :: =⟨Логическое имя⟩=⟨Физическое имя⟩|⟨Массив логических имен⟩=⟨Массив физических имен⟩|⟨Логическое имя⟩=⟨Физическое имя⟩, ⟨Перечень внешних контактов⟩|⟨Массив логических имен⟩=⟨Массив физических имен⟩, ⟨Перечень внешних контактов⟩

⟨Описание группы элементов⟩ :: =⟨Список имен элементов⟩. ⟨Тип элемента⟩

⟨Список имен элементов⟩ ::=⟨Имя элемента⟩|⟨Массив имен элементов⟩|⟨Имя элемента⟩, ⟨Список имен элементов⟩|⟨Массив имен элементов⟩, ⟨Список имен элементов⟩

⟨Описание группы контактов элементов⟩ :: =⟨Список имен элементов⟩/⟨Список имен внешних элементов⟩

⟨Список имен внешних контактов⟩ :: =⟨Имя внешнего контакта⟩|⟨Массив имен внешних контактов⟩|⟨Имя внешнего контакта⟩, ⟨Список имен внешних контактов⟩|⟨Массив внешних контактов⟩, ⟨Список имен внешних контактов⟩

Для иллюстрации возможностей языка описания схем АСП-6 рассмотрим некоторый блок 3-го уровня ЦУ (рис. 6.2,*а*), состоящий из блоков 2-го уровня АЛУ, СОЗУ, УУ, где блок СОЗУ (рис. 6.2,*б*) состоит из блоков 1-го уровня РГ1÷РГ128, ДШ1; блок РГ(РЕГИСТР) состоит из БЭ $D1 \div D32$, ТГ1—ТГ32 (рис. 6.2,*в*). Ниже приведено описание перечнем сигналов блоков ЦУ, СОЗУ и списком цепей блока РЕГИСТР.

```
ЦУ;
    ВНЕШНИЕ*
            V(1 : 32);
            B(1 : 8);
    СИГНАЛЫ*
            У1.АЛУ;
            У2.СОЗУ;
            У3.УУ;
КОНЕЦ*
СОЗУ;
    ВНЕШНИЕ*
              A(1 : 5).ВХОД;
            X(1 : 32).ВХОД;
            Z(1 : 32).ВЫХОД;
            ТАКТ, СБРОС.ВХОД;
    СИГНАЛЫ*
            РГ(1 : 128).РЕГИСТР : ВЫБОР=R(1 : 128);
            ДШ1.ДЕШИФРАТОР : ADP(1 : 7)=A(1 : 7);
КОНЕЦ*
РЕГИСТР;
    ВНЕШНИЕ*
```

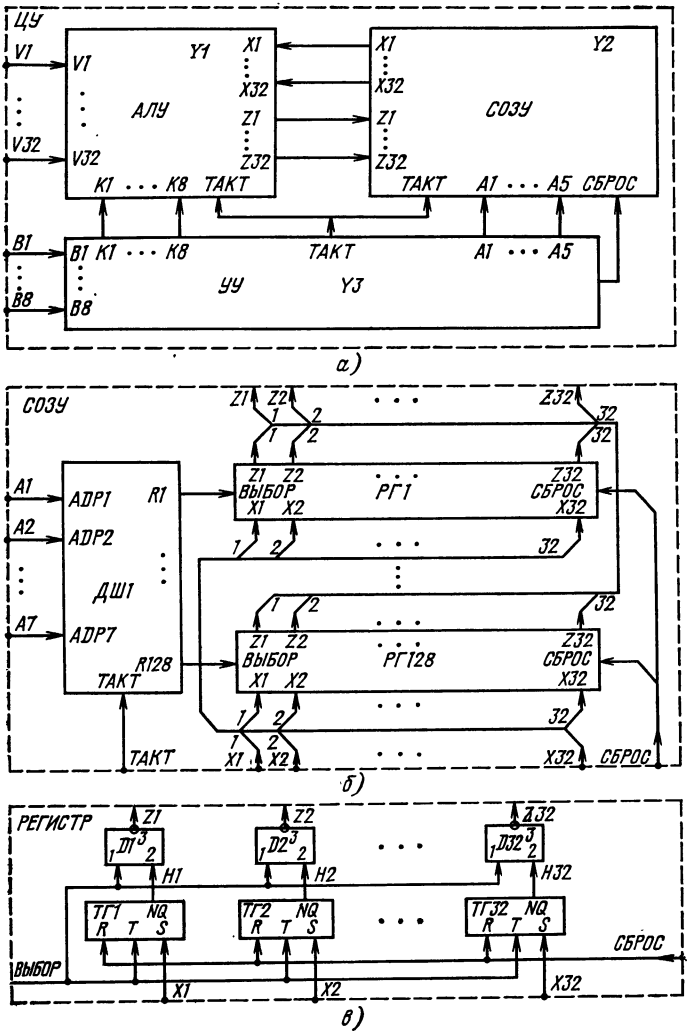


Рис. 6.2. Пример иерархического представления СхЭ

```
        Z(1 : 32).ВЫХОД;
        X(1 : 32).ВХОД;
          ВЫБОР.СБРОС.ВХОД;
ЭЛЕМЕНТЫ*
        D(1 : 32).ИНЕ;
        ТГ(1 : 32).ТРИГГЕР;
ЦЕПИ*
        СБРОС : ТГ(1 : 32)/R;
        ВЫБОР : D(1 : 32)/1.ТГ(1 : 32)/T;
```

```
            X(1 : 32) : ТГ(1 : 32)/S;
            Z(1 : 32) : D(1 : 32)/3;
            H(1 : 32) : D(1 : 32)/2.ТГ(1 : 32)/NQ;
КОНЕЦ*
```

Для сравнения и иллюстрации дополнительных возможностей языка ниже приведено описание списком цепей блока СОЗУ и перечнем сигналов блока РЕГИСТР:

```
СОЗУ;
    ВНЕШНИЕ*
            A(1 : 7).ВХОД;
            X(1 : 32).ВХОД;
            Z(1 : 32).ВЫХОД;
            ТАКТ, СБРОС.ВХОД;
    ЭЛЕМЕНТЫ*
            ДШ1.ДЕШИФРАТОР;
            РГ(1 : 128). РЕГИСТР;
    ЦЕПИ*
          A(1 : 7) : ДШ1/ADP(1 : 7);
       R(1 : 128) : ДШ1/R(1 : 128).РГ(1 : 128)/ВЫБОР;
КОНЕЦ*
РЕГИСТР;
    ВНЕШНИЕ*
          Z(1 : 32).ВЫХОД;
          X(1 : 32).ВХОД;
          ВЫБОР.СБРОС.ВХОД;
    СИГНАЛЫ*
          D(1 : 32).ИНЕ : 1=ВЫБОР;
                          2=H(1 : 32);
                          3=Z(1 : 32);
      ТГ(1 : 32).ТРИГГЕР : R=СБРОС;
                          T=ВЫБОР;
                          S=X(1 : 32); NQ=H(1 : 32);
КОНЕЦ*
```

Язык описания тестов АСП-6 логического моделирования обладает достаточно мощными изобразительными средствами, позволяющими упростить описание больших объемов информации, и в то же время достаточно прост, чтобы не препятствовать освоению системы логического моделирования начинающими пользователями. Он имеет широкий набор изобразительных средств, позволяющих структурировать описание, разбивая его на отдельные модули, сокращать описание периодических и редкоизменяющихся сигналов, а также не задавать значения отдельных параметров сигналов, принимая по умолчанию их равными стандартным. Все основные средства языка описания тестов организованы в виде функционально-полного ядра, пользуясь которым можно описать любые последовательности входных сигналов, даже не зная о существовании остальных возможностей языка, которые позволяют значительно упростить описание больших объемов ин-

формации. Дополнительные средства языка также организованы по принципу независимости любой их части от всех остальных, что позволяет применять только те возможности языка, которые облегчают пользователям решение их задач. Язык описания тестов опирается на математическую модель, определяемую следующим образом: сигналы на входах матричных БИС принимают значения из множества {0, 1, R, X, U, D, Z}, где символы 0 и 1 обозначают лог. 0 и лог. 1 соответственно; символ R — высокоомное состояние; X — неизвестное или неопределенное состояние, U — состояние R или 1; D — состояние R или 0; Z — состояние 0 или 1 или R. Все сигналы являются потенциальными; значения сигналов изменяются только в моменты времени, кратные единице измерения.

При таких соглашениях значения сигналов на группе входов $B=\{b_1, \ldots, b_n\}$ в каждый момент времени можно представить в виде вектора входных сигналов $A(t)=\{a_1(t), \ldots, a_n(t)\}$, i-я компонента которого описывает значение сигнала на входе b_i. Следовательно, вектор входных сигналов меняет значение своих компонент лишь на конечной последовательности моментов времени $T=\{t_1, \ldots, t_m\}$. Поэтому вектор входных сигналов как функцию времени можно задать, перечислив его значения $A_k=A_k(t_k)$ на интервалах $[t_k, t_{k+1}]$ и длительности интервалов $\tau_k=t_{k+1}-t_k \forall_k=\overline{1, m}$. Отсюда следует, что входные сигналы для матричных БИС можно описать в виде последовательности $W=\{w_1, \ldots, w_m\}$, k-й член которой имеет вид $w_k=(A_k, \tau_k)$ и определяет значение вектора входных сигналов на k-м интервале и длительность этого интервала. Каждый член этой последовательности далее будем называть элементарным воздействием и считать, что оно описывает ситуацию, когда на группу входов матричных БИС в течение времени τ_k подаются сигналы, равные компонентам A_k.

Средства языка, образующие его ядро, позволяют описывать входные сигналы матричных БИС в виде последовательности элементарных воздействий. Перечисляются все входы матричных БИС, для которых описываются сигналы. При этом их порядок определяет порядок следования значений сигналов в векторе входных сигналов. Однако такое описание может быть очень громоздким и трудным как для понимания, так и для поиска ошибок в нем. Для облегчения трудностей, связанных с описанием больших объемов информации, в языке имеется ряд дополнительных средств. Наиболее важные из них — средства модульного описания сигналов и средства компактного описания редкоизменяющихся и периодических сигналов. Средства модульного описания позволяют разбить множество всех входов матричных БИС на несколько отдельных групп и описывать сигналы для каждой из них в виде отдельного модуля. Структура языка не накладывает никаких ограничений на способ разбиения множества входов матричных БИС на группы, но для наибольшего сокращения описания сигналов и повышения их наглядности обычно более удобно выделять в одну

группу входы с одинаковым функциональным назначением. Модульное описание сигналов позволяет повышать наглядность описания за счет структурированности и уменьшить его объем благодаря сокращению описания числа элементарных воздействий в каждом модуле.

Средства компактного описания редкоизменяющихся сигналов удобно применять, если в каждый момент времени свои значения изменяют лишь немногие из входных сигналов. В этом случае для задания нового вектора входных сигналов можно описать лишь его изменения относительно предыдущего, указав для каждого изменения имя входа, на котором оно произошло, и новое значение сигнала для него. Данное средство дает возможность в ряде случаев резко сократить объем описания, так как позволяет описывать лишь события на входах матричных БИС.

Для описания периодических сигналов язык имеет специальную конструкцию, называемую циклическим воздействием. Указанная конструкция описывает периодические сигналы, задавая вид этих сигналов и число таких периодов.

Кроме рассмотренных средств, язык имеет еще следующие: сокращение записи последовательностей одинаковых компонент векторов; сокращение описания вектора входных сигналов, когда в нем изменяют свои значения несколько соседних компонент; запись элементарного воздействия без указания длительности, а циклического воздействия без указания числа повторений, принимая по умолчанию их равными стандартным значениям; специальный оператор установки режима трансляции, который может изменять масштаб времени, а также стандартные значения длительности и числа повторений; режим описания статических входных тестовых сигналов, т. е. таких, у которых время между подачей отдельных наборов превышает длительность переходных процессов в БИС. Основные преимущества данного языка по сравнению с аналогичными языками (система МОДИС, МОЛК [162], ЕСАП [163], СЛЭП [164]) заключается в том, что он наряду с очень простым ядром имеет достаточно мощные средства для компактного и наглядного описания больших последовательностей входных сигналов со сложной структурой.

Приведем формальное описание языка описания тестов АСП-6, где под блоком будем понимать также матричную БИС в целом.

Синтаксис ядра:

⟨Описание сигналов блока⟩ :: =[⟨Предложение описания сигналов⟩]

⟨Предложение описания сигналов⟩ :: =⟨Список имен входов⟩ ⟨Описание сигналов группы входов⟩

⟨Список имен входов⟩ :: =⟨Имя входа⟩;|⟨Имя входа⟩.⟨Список имен входов⟩

⟨Описание сигналов группы входов⟩ :: =⟨Элементарное воздействие⟩;|⟨Элементарное воздействие⟩.⟨Описание сигналов группы входов⟩

⟨Элементарное воздействие⟩ :: =⟨Вектор входных сигналов⟩/⟨Длительность⟩

⟨Вектор входных сигналов⟩ :: =⟨7-значный вектор⟩

⟨7-значный вектор⟩ :: =⟨Компонента⟩|⟨Компонента⟩⟨7-значный вектор⟩

⟨Компонента⟩ :: - 0|1|R|X|U|D|Z
⟨Длительность⟩ :: =⟨Целое без знака⟩

1. Расширение ядра:

⟨Описание сигналов блока⟩ :: = [⟨Последовательность предложений описания сигналов⟩]

⟨Последовательность предложений описания сигналов⟩ ::=⟨Предложение описания сигналов⟩|⟨Предложение описания сигналов⟩ ⟨Последовательность предложений описания сигналов⟩.

2. Средства компактного описания редкоизменяющихся сигналов:

⟨Вектор входных сигналов⟩ :: =⟨7-значный вектор⟩*⟨Изменения вектора относительно предыдущего⟩

⟨Изменение вектора относительно предыдущего⟩ :: =⟨Имя входа⟩=⟨Новое значение сигнала⟩;|⟨Имя входа⟩=⟨Новое значение сигнала⟩, ⟨Изменение вектора относительно предыдущего⟩

⟨Новое значение сигнала⟩ :: =⟨7-значный вектор⟩

3. Средство компактного описания периодических сигналов:

⟨Описание сигналов группы входов⟩ :: =⟨Воздействие⟩; |⟨Воздействие⟩ ⟨Описание сигналов группы входов⟩

⟨Воздействие⟩ :: =⟨Элементарное воздействие⟩|⟨Циклическое воздействие⟩

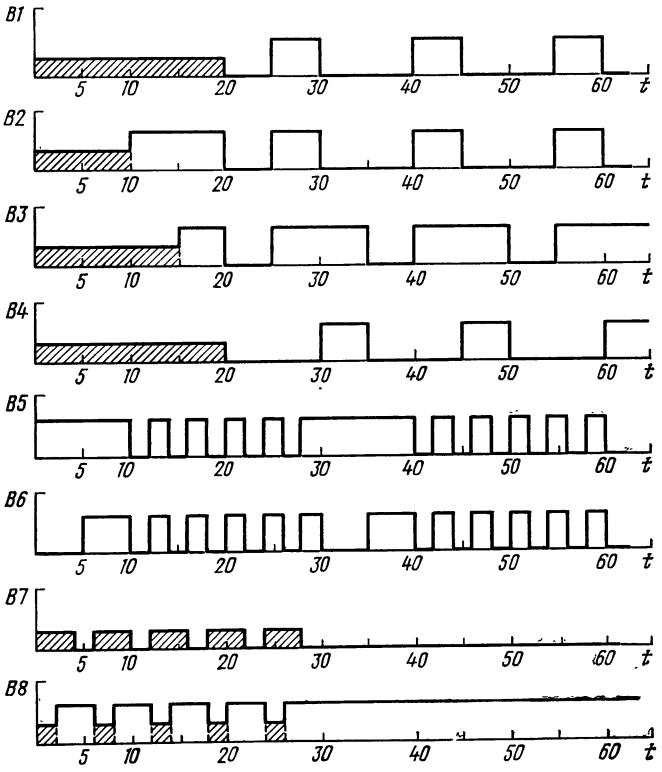


Рис. 6.3. Пример теста моделирования

⟨Циклическое воздействие⟩ :: =(⟨Описание сигналов группы входов⟩)⟨Число повторений⟩

⟨Число повторений⟩ :: =⟨Целое без знака⟩

Проиллюстрируем возможность языка описания тестов АСП-6 на примере описания теста (рис. 6.3), подаваемого на вход схемы ЦУ (рис. 6.2,*а*). Описание имеет следующий вид:

```
[B1.B2.B3.B4;
      X(4)/1Ø*B2=1/5.*B3=1/5.(ØØØØ/5.111 Ø /5. ØØ 11/5)3;
B5.B6;
      (1Ø /5.11/5.( ØØ/2.11/2)5)2;
B7.B8;
      (XX/2.*B8-1/2.*B7=Ø/2)5;]
```

6.3. ПОДСИСТЕМА ИЕРАРХИЧЕСКОГО ЛОГИЧЕСКОГО МОДЕЛИРОВАНИЯ АСП-65

Исходной информацией для системы логического моделирования АСП-65 являются описания СхЭ и тестов моделирования на языке АСП-6, для которых разработаны соответствующие трансляторы, а также описание функционирования логических элементов, находящихся в базе данных АСП-6БД. Трансляция описания с языка АСП-6 в язык загрузки осуществляется в два этапа: сначала в промежуточный язык, затем в язык загрузки. На первом этапе выполняются следующие операции: контроль синтаксических ошибок; перевод описания в промежуточный язык; обнаружение одноименных контактов в разных цепях и незадействованных контактов; замена всех вхождений блоков ненулевого уровня описаниями блоков более низкого уровня вплоть до БЭ. Промежуточный язык разработан для того, чтобы повысить эффективность реализации последних двух операций. На втором этапе выполняются следующие функции: определение типа каждого из контактов матричных БИС или блока (внутренний, внешний, входной, выходной, земля, питание и т. д.); классификация цепей по видам в зависимости от типов составляющих их контактов; составление словаря входных контактов матричных БИС; обнаружение схемотехнических ошибок и их вывод на печать или дисплей; формирование описания матричных БИС на языке загрузки. Транслятор написан на языке Ассемблер и функционирует с минимальным объемом ОЗУ в 150К байт. Трансляция иерархического пятиуровневого описания матричных БИС объемом 2000 логических элементов занимает порядка 20 с процессорного времени ЕС-1060 и требует 200К байт ОЗУ.

Транслятор описания тестов моделирования переводит описание во внутренний язык. В связи с тем, что система АСП-65 может использоваться как в пакетном, так и в интерактивном режимах, к транслятору предъявляются следующие требования: высокое быстродействие; возможность работы при малых объемах рабочего поля оперативной памяти; возможность автоматической настройки на большие объемы рабочего поля оперативной памяти с одно-

временным увеличением скорости трансляции; возможность обработки больших объемов информации даже при малых размерах рабочего поля за счет снижения скорости трансляции. Для удовлетворения этих требований при разработке транслятора были использованы следующие методы: организация словарей в виде хеш-таблиц; однопроходной синтаксический анализ без возвратов; табличное управление лексическим и синтаксическим анализатором; списковая организация рабочего поля оперативной памяти; использование внешней памяти на дисках для промежуточных результатов только в случае недостатка оперативной памяти. Для облегчения работы с системой логического моделирования транслятор имеет развитые средства диагностики, позволяющие не только обнаружить ошибки в тексте описания, но и максимально точно идентифицировать их характер, а в ряде случаев даже исправить отдельные ошибки, выдав при этом предупредительные сообщения. Такие средства диагностики значительно сокращают время отладки описания теста. Транслятор реализован на ассемблере и требует минимальный объем оперативной памяти в 110К байт. Скорость трансляции на ЕС-1060 при выделении необходимого объема оперативной памяти составляет около 10 с на 1000 векторов и зависит от объема и сложности теста.

Система событийного 7-значного логического моделирования АСП-65 [152] является развитием системы логического моделирования АСП-55, нашедшей широкое распространение среди пользователей для моделирования цифровой микроэлектронной аппаратуры. Система АСП-65 позволяет моделировать матричные БИС сложностью до 10^5 логических элементов средней степени интеграции на последовательностях входных сигналов до 10^5 векторов и обладает следующими свойствами: высоким быстродействием; возможностями обработки больших объемов информации; параллельной работой большого числа пользователей при моделировании матричных БИС или ее блоков любого уровня иерархии; развитым информационно-логическим и сервисным обеспечением, рассчитанным на пользователей не программистов; развитым контролем и диагностикой синтаксических ошибок во входном описании на языке АСП-6, низкими требованиями к объему оперативной памяти; возможностью работы в режиме коллективного пользования с использованием дисплеев. Система позволяет делать контрольные точки для перезапусков, пользоваться последовательностью из нескольких тестов для одного устройства, определять в интерактивном режиме различные условия прерывания процесса моделирования, анализировать состояние устройства в момент прерывания, прослеживать распространение сигналов и возникновение гонок, рисков и состязаний, изменять тесты и схему устройства.

Система АСП-65 имеет следующие особенности: иерархический принцип моделирования, обеспечивающий моделирование как матричных БИС в целом, так и блоков любого уровня в соответствии с описанием матричной БИС на языке АСП-6; моделирование схем с

двунаправленной передачей информации (шины данных, порты ввода-вывода, двунаправленные ключи), хранение сигналов на емкостях посредством введения «силы сигнала» с 16 градациями и высокоомного состояния R; моделирование монтажных объединений БЭ без введения псевдоэлементов; доопределение незадействованных входов БЭ логическими значениями 0, 1, X в зависимости от типа БЭ; событийное 7-значное моделирование с учетом задержки сигналов на БЭ и в межсоединениях. Каждому выходу БЭ задается задержка переключения сигнала отдельно из 0 в 1, из 1 в 0 в зависимости от величины емкостной нагрузки (определяемой числом и типом БЭ, на которые нагружен конкретный БЭ, и топологией межсоединений до этих БЭ), напряжения источника питания и температуры окружающей среды. Для переключения выходов БЭ в высокоомное состояние R назначается равная для всех выходов задержка. Для обнаружения быстрых изменений состояний входов БЭ длительностью меньше определенной величины вводится инерционная задержка. Отношение максимальной задержки элемента к минимальной в одном устройстве не должно превышать 255. Подсистема АСП-65 позволяет обнаруживать наличие рисковых ситуаций на входах элементов, к числу которых относятся некорректное одновременное изменение состояния входов и подача запрещенных комбинаций на входы элементов определенных типов (триггер, счетчик и др.). Примером первой ситуации может служить одновременная смена состояний двух входов элемента И из 0 в 1 и из 1 в 0.

Результатом работы моделирующей программы являются временны́е диаграммы сигналов для различных контактов моделируемой схемы. Все они записываются в базу данных в виде последовательностей событий. Такая форма позволяет снизить требования к объему внешней памяти, так как записывается информация только о моментах изменения сигналов. Для вывода временны́х диаграмм на дисплей или печать используется специальная программа, осуществляющая перекодировку этой информации к виду, удобному для анализа. В качестве исходных данных ей задается временно́й интервал и перечень имен контактов схемы, для которых надо получать временны́е диаграммы сигналов.

Исходная и результирующая информация для моделирования хранится в базе данных АСП-6БД. База данных повышает пропускную способность системы, обеспечивая параллельную отладку нескольких десятков схем. Запись всех результатов моделирования позволяет пользователям оперативно анализировать временны́е диаграммы сигналов во внутренних точках схем непосредственно за экраном дисплея без многократного моделирования. Развитый язык описания тестов обеспечивает моделирование матричных БИС или блоков при входных воздействиях, объем и сложность которых приближается к реальным. Это обеспечивает высокое качество отладки. Программное обеспечение системы АСП-65 реализовано на языках Ассемблер и Фортран в ОС ЕС и эффективно функционирует с рабочим полем оперативной памяти 512К байт. Выде-

ление для работы системы дополнительной оперативной памяти увеличивает ее быстродействие и диапазон объемов обрабатываемых схем.

Время моделирования схемы сложностью в M вентилей на N входных векторах $T \sim \tau MN$ (с), где τ — удельное время моделирования, т. е. время моделирования одного вентиля на одном входном векторе. Значение τ зависит от метода моделирования, производительности ЭВМ, способа описания схемы. Для системы АСП-65, реализованной на ЕС-1060, $\tau \approx 10$ мкс (около 5000 событий в 1 с). Так, при $M = 30 \cdot 10^3$ вентилей ($6 \cdot 10^3$ логических элементов) $N = 1 \cdot 10^4$, $T \approx 1$ ч (для ЕС-1060). Как показал опыт эксплуатации АСП-65, на моделирование матричных БИС сложностью в 400 БЭ (≈ 1200—1500 вентилей) требуется около 2 недель при 1—3 ч ежедневной работы за дисплеем. При этом обнаруживается до 10 ошибок в схеме.

Как следует из сказанного, язык описания схем АСП-6 и система АСП-65 ориентированы не только на моделирование матричных БИС, но и СБИС, проектируемых методом стандартных блоков.

6.4. ПОДСИСТЕМА ПРОЕКТИРОВАНИЯ ТОПОЛОГИИ МАТРИЧНЫХ БИС АСП-62

Исходной информацией для подсистемы проектирования топологий АСП-62 [159, 165—167] является описание схемы, развернутое на этапе трансляции до блоков 1-го уровня и/или БЭ, и описание конструкции БМК, БЭ, блоков 1-го уровня, находящихся в базе данных АСП-6БД. При этом длина блока 1-го уровня не должна превышать длину линейки.

Особенностью проектирования топологии матричной БИС линейчатого типа является размещение элементов, имеющих равные высоты и разные длины, с учетом транзитных соединений, проходящих через линейки, возможность подхода к контактам БЭ сверху и снизу, трассировка соединений в горизонтальных каналах и в областях, расположенных по периферии БМК. С целью учета транзитных соединений, пересекающих линейки, на этапе размещения элементов в список цепей автоматически вводятся *проходные элементы* (ПрЭ). Размер ПрЭ, входящего в библиотеку БЭ, равен одному дискрету по оси X и высоте линейки по оси Y. Проходной элемент содержит вертикальный проводник, ширина которого равна ширине вертикальных проводников, высота равна высоте линейки. Кроме того, ПрЭ содержит слой защиты, который блокирует формирование транзисторов на месте пересечения линейки с вертикальным проводником. Проектирование топологии состоит из двух этапов: размещение элементов и трассировка соединений.

Размещение элементов. Имеются два способа размещения элементов: автоматическое и автоматизированное. В автоматическом размещении реализовано два подхода: размещение при заданном

начальном закреплении внешних сигналов по периферийным элементам (алгоритм R1) и без закрепления внешних сигналов (алгоритм R2).

При проектировании новой БИС, как правило, размещение сигналов по периферийным элементам не задается. Но после возможной коррекции схемы проектирование ее производится с уже заданным размещением внешних сигналов (так как уже, возможно, спроектированы микросборки или печатные платы, на которые устанавливаются БИС).

Рассмотрим алгоритм размещения элементов, не учитывая, задано размещение внешних сигналов по периферийным элементам или нет. Указанный алгоритм (R) состоит из следующих процедур:

1. Начальная компоновка элементов в линейки последовательным алгоритмом [7] с учетом размеров элементов и линеек.

2. Размещение линеек последовательным алгоритмом друг относительно друга для достижения минимума транзитных связей в линейках.

3. Улучшение компоновки парными перестановками элементов между соседними линейками.

4. Формирование ПрЭ в линейках без конкретного их размещения внутри линеек и соответствующих соединений.

5. Размещение элементов (в том числе ПрЭ) одновременно внутри всех линеек методом дихотомического деления [7, 90].

6. Размещение БЭ внутри блока 1-го уровня с помощью алгоритма, описанного в п. 5.

7. Размещение внешних сигналов по периферийным элементам (если такое размещение не задано).

8. Разбиение списка цепей на отдельные соединения с учетом ПрЭ и формирование списка соединений в каналах.

9. Распределение сигналов соединений по инвариантным контактам БЭ по критерию минимума длины соединения.

10. Представление полученного списка соединений в координатах рабочего поля по результатам размещения элементов.

Процедура 1 сокращает вычислительную сложность алгоритма размещения, обеспечивает формирование сильно связанных множеств элементов в каждой линейке без конкретного размещения элементов в линейках. При этом линейки заполняются разногабаритными элементами на r % ($r<100$ %) для обеспечения реализации транзитных магистралей (размещения ПрЭ). Параметр r зависит от конструкции БМК, применяемых в САПР алгоритмов размещения и трассировки и определяется статистически. Для рассматриваемых БМК и подсистемы АСП-62 $r\leqslant 80$ %. Процедуры 2, 3 уменьшают общую суммарную длину соединений и число транзитных магистралей. Поскольку размещение элементов в линейках еще не задано, то все элементы, принадлежащие одной цепи и расположенные в одной линейке, будем называть макроэлементом. Если цепь содержит несколько макроэлементов, расположенных в линейках L_i, L_j, L_k, ..., то в линейках, расположенных меж-

ду L_i, L_j, L_k, ..., формируются ПрЭ, которыми дополняется рассматриваемая цепь (процедура 4).

Процедура 5 производит размещение элементов одновременно во всех линейках. В общем виде алгоритм размещения элементов в *рабочем поле* (РП) БМК можно описать следующим образом. Пусть имеется множество элементов M_0, подлежащих размещению в РП. Результат работы алгоритма на i-м шаге представляется в виде множества фрагментов $F_i = \{f_1, \ldots, f_j, \ldots, f_{i+1}\}$ РП с указанием для каждого фрагмента f_j множества содержащихся в нем элементов M_j без указания конкретного размещения последних во фрагменте f_j, где $i = 1, 2, \ldots$ При этом

$$\bigcup_{j=1}^{i+1} f_j = f_0, \quad \bigcup_{j=1}^{i+1} M_j = M_0,$$

$$f_j \cap f_k = \varnothing, \; M_t \cap M_k = \varnothing \; \forall j, k \in \{1, 2, \ldots, i+1\},$$

где f_0 — начальное РП.

Для каждого фрагмента f_j задаются его размеры (длина) и координаты расположения в РП. На очередном шаге работы алгоритма из множества фрагментов F_i выбирается максимальный по размеру фрагмент f_j, если он содержит более одного элемента, и делится на фрагменты f_{j1} и f_{j2}. Сечение, делящее фрагмент f_j, представляет собой ломаную линию, проходящую через фрагменты линеек. Множество элементов M_j делится на два подмножества M_{j1}, M_{j2} по обе стороны от сечения. Множество элементов M_{j1} относится к фрагменту f_{j1}, множество M_{j2} — к фрагменту f_{j2}. При этом все элементы остаются в своих линейках. Критерием деления элементов M_j является минимум интервалов соединений, пересекаемых данным сечением. Далее рассчитываются размеры фрагментов из множества f_{j1}, f_{j2} для каждой линейки, координаты расположения этих фрагментов в РП. Признаком окончания работы алгоритма является состояние, при котором в каждом фрагменте f_j будет содержаться по одному элементу.

Рассмотрим более подробно процедуру разбиения элементов M_j фрагмента f_j. Слева от сечения расположим фрагмент f_{j1}, справа f_{j2}. Пусть элемент a принадлежит фрагменту f_j и входит в l цепей. Тогда на элемент a действует сила

$$p(a) = \sum_{k=1}^{l} p_k(a) = \sum_{k=1}^{l} \left(\frac{C_{k1}}{n_k - n_{k1}} - \frac{C_{k2}}{n_k - n_{k2}} \right), \qquad (6.1)$$

где $p_k(a)$ — сила, созданная цепью k, содержащая элемент a; n_{k1} (n_{k2}) — число контактов цепи k, расположенных слева (справа) от фрагмента f_j; n_k — общее число контактов цепи k; $C_{k1} = C_{k2} = 0$, если $n_{k1} = n_{k2} = 0$ или $n_{k1} > 0$, $n_{k2} > 0$; $C_{k1} = 1$, если $n_{k1} > 0$, $n_{k2} = 0$; $C_{k2} = 1$, если $n_{k1} = 0$, $n_{k2} > 0$. Тогда:

1. Рассчитаем силы, действующие на элементы фрагмента f_j.

2. Найдем элементы $a_{\max}$, $a_{\min}$, имеющие соответственно максимальные и минимальные силы притяжения к множеству M_{j1}.

3. Отнесем элемент a_{max} к множеству M_{j1}, элемент a_{min} к множеству M_{j2} фрагментов f_{j1}, f_{j2} соответственно и исключим эти элементы из множества элементов M_j.

Для оставшихся элементов выполним п.п.1—3. Процесс формирования множеств M_{j1}, M_{j2} заканчивается после разнесения последних элементов. Причем, если число элементов множества M_j нечетное, то последний элемент относится к множеству M_{j1}, если действующая на него сила положительная, и к множеству M_{j2}, если сила отрицательная.

Процедура 6 аналогична процедуре 5, только применяется к к каждому блоку 1-го уровня отдельно. При этом с целью минимизации длины обхода блоков (которые могут иметь большую длину) транзитными соединениями проходные элементы, размещенные рядом с блоком, включаются в состав БЭ блока в начале процедуры 6. Затем к ним (БЭ и ПрЭ) применяется процедура 6, что дает возможность размещать ПрЭ внутри блоков.

В процедуре 7 выполняется назначение внешних сигналов на свободные периферийные элементы с помощью венгерского алгоритма [168] по критерию минимизации общей суммарной длины соединений, если это назначение не задано или задано частично. В качестве внутреннего контакта внешней цепи, от которого определяется расстояние до j-го ПЭ, выбирается контакт цепи, ближайший к j-му ПЭ.

Процедура 8 производит разбиение цепей по алгоритму Прима на соединения из двух контактов (декомпозиция цепей) с учетом ПрЭ и формирует соединения в каждом канале, в том числе и каналах, образованных горизонтальными рядами периферийных элементов и линейками БЭ. Если размещение внешних сигналов не было задано, то после их назначения процедурой 7 и декомпозиции цепей возможно появление дополнительных транзитных соединений и соответствующих ПрЭ в линейках, так как вдоль боковых периферийных элементов имеется только 1—2 вертикальные магистрали. Размещение указанных ПрЭ по критерию минимизации длины связей осуществляется раздвижением БЭ в соответствующих линейках. Соединения, контакты которых лежат на одной линейке, приписываются к каналу с меньшим номером.

Для облегчения процесса трассировки каждое внешнее соединение разбивается на два сегмента S_{i1} и S_{i2} введением на конце линейки псевдоконтакта. Сегмент S_{i1} включает контакт линейки и псевдоконтакт, сегмент S_{i2}—псевдоконтакт и соответствующий ПЭ. При этом все псевдоконтакты, лежащие на одном конце одной и той же линейки, накладываются друг на друга и в процессе трассировки вертикальные отрезки проводников к псевдоконтактам не учитываются. В дальнейшем все фрагменты соединений и сегменты типа S_{i1} будем называть соединениями, сегменты типа S_{i2} — боковыми соединениями.

Процедура 9 выполняет переразмещение инвариантных сигналов по БЭ с целью минимизации необходимого числа магистралей в канале, длины трассы и числа пересечений. Процедура 10 по

результатам размещения формирует список соединений для каждого канала в координатах РП. Список боковых соединений в координатах РП формируется после этапа параллельной трассировки, так как на этапе размещения не определена координата Y контакта бокового соединения C, расположенного в конце канала на одной вертикали с псевдоконтактом.

Опишем теперь алгоритм R1. Нижние и верхние периферийные элементы рассматриваются как уже скомпонованные линейки L_0 и L_{t+1} соответственно, где t — число линеек БЭ. К этим и боковым периферийным элементам применяются только процедуры 4, 8, 10. Однако во всех процедурах учитываются связи с ними. Линейки L_0 и L_{t+1} используются в процедурах 1,2 алгоритма R1 как начальные условия компоновки и размещения. Если цепь содержит боковой элемент, в процедуре 4 находится ближайший к нему макроэлемент (в том числе и ПрЭ) той же цепи и строится внешнее соединение. При этом возможно появление ПрЭ в линейках, пересекаемых внешним соединением, при построении кратчайшего пути.

В процедурах 5, 6 боковые элементы используются в качестве начального размещения; на каждой итерации деления фрагмента f_j учитываются связи элементов M_j с периферийными элементами, расположенными слева и справа от f_j при подсчете сил, действующих на внутренние элементы. Причем исходными данными для процедуры деления являются множества M_1, M_2, M_3, где M_1, M_2 — множества боковых левых и правых элементов соответственно; M_3 — множество элементов матричной БИС.

В алгоритме R2 на этапе начальной компоновки в качестве начального элемента (центра «кристаллизации») выбирается элемент с максимальным числом связей и относительно этого элемента производится сначала компоновка одной линейки, затем другой и т. д. При компоновке не учитываются связи с периферийными элементами. В процедуре 2 в качестве начального размещения выбираются две линейки, имеющие максимум связей с периферийными элементами. Одну из них L_1 относят к верхнему ряду линеек БЭ, другую L_t к нижнему. Отнесем $r_1 \leqslant R_1$ внешних сигналов линейки L_1 к элементам верхнего ряда (к линейке L_0), $r_2 \leqslant R_2$ внешних сигналов линейки L_t к элементам нижнего ряда (к линейке L_{t+1}), где R_1, R_2 — число периферийных элементов линеек L_0, L_{t+1} соответственно. При начальной компоновке и размещении линеек L_0, L_1, L_t, L_{t+1} выполняются процедуры 2—6 с учетом связей только с элементами линеек L_0, L_{t+1}. При этом указанные процедуры производятся и над периферийными элементами линеек. В процедуре 5 формируются множества элементов M_1, и M_2, расположенные соответственно слева и справа от элементов РП. Процедура (5.1) нахождения элементов множеств M_1, M_2 включает следующие операции:

1. Среди всех элементов M_0 находится элемент $a_{\max}$, имеющий максимальное число соединений, и его приписывают к некоторому множеству A (в начальный момент $A = \varnothing$).

2. По формуле (6.1) рассчитываются силы, действующие на оставшиеся элементы.

3. Элемент, на который действует максимальная сила притяжения, относится к множеству A.

4. П.п. 2,3 повторяются до тех пор, пока не останется единственный элемент ($a_{\min}$). Элемент $a_{\max}$ относится к множеству M_1, элемент $a_{\min}$ — множеству M_2.

Далее выполняются операции согласно процедуре 5 при начальном разбиении элементов на множества M_1, M_2, M_3, где $M_3 = M_0 \setminus \{M_1 \cup M_2\}$.

Процедура 6 производит аналогичные операции над множеством M_3 элементов блока, если слева и справа от блока имеются множества элементов M_1, M_2. Если $M_1 = M_2 = \varnothing$, то вначале применяется процедура 5.1 к элементам блока. Если $M_1 \neq \varnothing$, $M_2 = \varnothing$, то применяются п.п. 2—4 процедуры 5.1 при условии $A = M_1$, M_0 — множество БЭ блока. Элемент $a_{\min}$ относится к множеству M_2. Аналогично, если $M_1 = \varnothing$, то $M_2 = \varnothing$. При этом $A := M_2$, и элемент $a_{\min}$ относится к множеству M_1. Далее все операции производятся в соответствии с процедурой 5. В остальном алгоритм R2 аналогичен алгоритму R.

Вычислительная сложность алгоритма размещения элементов в обоих подходах $O(N)$, где N — число элементов (блоков 1-го уровня и отдельных БЭ). Так, общее время размещения элементов и внешних сигналов развернутой до БЭ матричной БИС, содержащей около 300 БЭ в 13 линейках, порядка 1000 соединений, 1700 контактов составило около 4 мин ЭВМ ЕС-1060.

При автоматизированном размещении элементов в описании СхЭ указываются номера линеек, координаты X размещения БЭ и координаты X, Y размещения периферийных элементов. Далее выполняются п.п. 8—10 алгоритма размещения R. Декомпозиция цепей производится по минимуму значений функции $F(g)$ для каждого соединения, где $F(g)$ — функция от длины соединения g, числа соединений в максимальном сечении и суммарного числа соединений во всех сечениях соединения g. Если соединение пересекает линейки, то на свободные места этих линеек и проходные вертикальные магистрали БЭ в области соединения g размещаются ПрЭ. Размещение производится по минимуму значения функции $F(g_i)$ каждого фрагмента g_i соединения g, построенных делением соединения g соответствующими ПрЭ. Полученные ПрЭ включаются во множество контактов рассматриваемой цепи и используются при построении очередного соединения алгоритмом Прима.

Трассировка соединений осуществляется в несколько этапов.

1. Параллельная трассировка соединений матричной БИС или апертуры, содержащей несколько каналов, методом исключений [165—167].

2. Трассировка вертикальных соединений, т. е. соединений, контакты которых расположены на одной вертикали в пределах одного канала.

3. Сортировка отрезков по слоям с целью минимизации межслойных переходов.

4. Трассировка боковых и не разведенных соединений волновым алгоритмом.

5. Интерактивная трассировка соединений, не разведенных на этапах 1, 4.

На этапе 1 трассировка осуществляется проводниками, содержащими два перегиба, причем горизонтальные $\{a_i\}$ и вертикальные $\{b_{i1}, b_{i2}\}$ отрезки проводников $\{\mu_i\}$ разнесены в разные слои, межслойные переходы осуществляются в узлах опорной сетки РП, где μ_i — проводник, реализующий соединение i; b_{i1}, b_{i2} — соответ-

ственно левый и правый вертикальные отрезки проводника μ_i. В качестве контакта БЭ, расположенного на линейке элементов, принимается контакт, расположенный на нижней стороне БЭ.

Схеме соединений G поставлен в соответствие совмещенный граф интервалов $\Gamma(X, V)$, представляющий частично-ориентированный граф, где X — множество вершин графа, соответствующих множеству горизонтальных отрезков $\{a_i\}$; V — множество связей (ребер и дуг), которые строятся по следующему правилу. Пусть λ_i — множество горизонтальных магистралей РП, на которых можно размещать отрезок a_i без пересечения вертикальными отрезками b_{i1}, b_{i2} контактов других цепей и рабочего поля, занимаемого БЭ. Будем считать, что соединения i, j пересекаются, если они принадлежат различным цепям и их интервалы пересекаются. Между вершинами x_i, x_j устанавливается связь, если соединения пересекаются. Пусть соединения i, j пересекаются. Тогда вершины x_i, x_j соединяются ребром, если ни один из контактов соединения i не лежит на одной вертикали ни с одним из контактов соединения j за исключением псевдоконтактов, которые не учитываются. В противном случае вершины x_i, x_j соединяются дугой. Причем, если контакт соединения i лежит ниже контакта соединения j, то формируется дуга (x_i, x_j), если выше — дуги (x_j, x_i).

Ребро (x_i, x_j) означает, что горизонтальные отрезки проводников, реализующих соединения i, j, не могут лежать на одной горизонтальной магистрали без пересечений. Дуга (x_i, x_j) означает, что горизонтальный отрезок соединения i должен всегда лежать ниже горизонтального отрезка соединения j, иначе их вертикальные отрезки пересекутся.

Вершины графа Γ могут образовывать орциклы. Для их обнаружения применяются процедуры прямого и обратного ранжирования вершин графа. При ранжировании к элементам ранга 1 относятся вершины, не содержащие входящих в них дуг. К вершинам ранга j (R_j) относятся вершины, которые содержат хотя бы одну дугу, исходящую из вершин ранга j—1, и дуги, исходящие только из вершин низших рангов. Прямое ранжирование — это ранжирование графа $\Gamma(X, V)$, обратное ранжирование — ранжирование графа $\bar{\Gamma}(X, V)$, где $\bar{\Gamma}$ (X, V) — граф, полученный из графа $\Gamma(X, V)$ изменением направления стрелок дуг на обратное. В результате прямого и обратного ранжирования получаем список вершин прямого ранжирования $R=(R_1, R_2, \ldots, R_l)$, возможно, не пустое множество ΔX, список вершин обратного ранжирования $\bar{R}=(\bar{R}_1, \bar{R}_2, \ldots, \bar{R}_l)$ и, возможно, не пустое множество $\overline{\Delta X}$, где l — число рангов, ΔX, $\Delta\bar{X}$ — множество вершин орциклов графов $\Gamma(X, V)$, $\bar{\Gamma}(X, V)$ соответственно, а также производных от вершин орциклов, т. е. таких вершин, к которым есть путь от вершин орцикла вдоль стрелок дуг. Тогда $C=\Delta X \cap \Delta\bar{X}$ есть множество вершин орцикла графа $\Gamma(X, V)$ и графа $\bar{\Gamma}(X, V)$, что следует из способов построения указанных графов. Удаление орцикла производится разбиением одного из соединений g, вершина которого входит в орцикл, вер-

тикальным сечением D на два соединения с общим фиктивным контактом (рис. 2.8). Сечение D производится приблизительно посередине соединения g без пересечения контактов вертикальных соединений в рассматриваемом канале. По определению контакты соединения, входящего в орцикл, принадлежат различным линейкам. Пусть $y(L_j)$ — координата Y контактов БЭ, расположенных на линейке L_j; L_j — верхняя линейка, на которой лежит один из контактов соединения g, $x(D)$ — координата X сечения D. Тогда в качестве координат X, Y фиктивного контакта можно взять $x_\Phi = x(D)$, $y_\Phi = y(L_j) - 1$. Если в сечении D (в области канала) уже имеются фиктивные контакты других соединений, то сечение можно провести рядом или $y_\Phi := y(L_j) - i$, где $i = 2, 3, \ldots, n$; n — число магистралей в канале. Координаты Y фиктивных контактов должны быть разными. Полученные соединения включаются в общий список соединений. По окончании трассировки лишний вертикальный отрезок, исходящий из фиктивного контакта, удаляется. Если же и разбиение соединений невозможно, то одно из соединений орцикла удаляется и доразводится на этапах 4, 5. После коррекции списка соединений во вновь образованном графе производятся процедуры обнаружения и удаления возможных орциклов, так как коррекция списков соединений может создать новые орциклы, и т. д., пока не исключим из графа все орциклы. Нижняя оценка N необходимого числа магистралей для размещения всех горизонтальных отрезков соединений в канале или апертуре составляет $N \geqslant \max\{s, l\}$, где s — число интервалов соединений в максимальном сечении канала или апертуры; l — число рангов графа. Если N больше, чем число магистралей в канале, производится удаление некоторых соединений и доразводка их на этапах 4, 5.

На этапе 1 в отличие от [165] видоизменен вывод системы из равновесия и операции по исключению несовместимых решений определены над вершинами графа Γ. Трассировка методом исключений обеспечивает одновременное решение следующих задач: размещение соединений по каналам матричной БИС или апертуры; параллельную трассировку соединений в каналах матричной БИС или апертуры. Суть метода исключений при трассировке соединений проводниками с двумя перегибами состоит в следующем. Пусть граф Γ, соответствующий заданному множеству соединений, — граф без орциклов; $M = \{1, 2, \ldots, n\}$ — множество горизонтальных магистралей для трассировки соединений; n — число магистралей. Пометим вершины графа метками из множества M по следующему правилу: если отрезок a_i может размещаться на магистралях $\lambda_i = (\lambda_{i1}, \lambda_{i2}, \ldots, \lambda_{in_i}) \subset M$ без пересечения отрезками b_{i1}, b_{i2} контактов других соединений (кроме фиктивных и псевдоконтактов, которые можно пересекать) или запрещенных зон, то вершину x_i пометим метками λ_i, где λ_i — набор магистралей, упорядоченных в порядке возрастания их номеров. Пусть две вершины x_i, x_j имеют дугу (x_i, x_j) и эти вершины помечены метками $\lambda_i = \lambda_j = (1, 2, \ldots, m)$, где $m \leqslant n$. Из определения дуги следует, что

отрезок a_j должен лежать выше отрезка a_i, т. е. отрезок a_j не может лежать на магистрали 1, а отрезок a_i не может лежать на магистрали m. Следовательно, из набора λ_j необходимо исключить метку 1, а из набора λ_i метку m.

Пусть вершина x_i помечена метками $\lambda_i = (\lambda_{i1}, \ldots, \lambda_{in_i})$, вершина x_j метками $\lambda_j = (\lambda_{j1}, \ldots, \lambda_{jn_j})$ и вершины x_i, x_j соединены дугой (x_i, x_j). Тогда на основании сказанного из набора λ_j необходимо исключить метки, меньшие или равные метке λ_{i1} (правило 1), а из набора λ_i метки, большие или равные λ_{jn_j} (правило 2). Из определения ребра (x_i, x_j) следует, что если вершина x_i имеет одну единственную метку $\lambda_i \equiv \lambda_{ir}$, то из набора λ_j необходимо исключить эту метку. То же, если λ_j имеет одну единственную метку (правило 3).

Пусть все вершины графа помечены наборами меток согласно возможному размещению горизонтальных отрезков по магистрали. Применим правило 1 для всех вершин $A(x_i)$, смежных с x_i и имеющих дуги (x_i, x_j), где $x_j \in A(x_i)$ (процедура 1). Применим правило 2 для всех вершин $B(x_i)$, смежных с x_i и имеющих дуги (x_j, x_i), где $x_j \in B(x_i)$ (процедура 2).

Назовем прямой процедурой исключения применение процедуры 1 и правила 3 ко всем вершинам списка прямого ранжирования R, начиная с первой вершины. Назовем обратной процедурой исключения выполнение процедуры 2 и правила 3 ко всем вершинам списка обратного ранжирования $\bar{R}$, начиная с первой вершины. Назовем процедурой исключения прямую и обратную процедуры исключения.

В процессе применения к спискам R, $\bar{R}$ процедур исключения производится исключение меток согласно правилам 1—3. Поэтому процесс исключения меток носит итерационный характер. После многократного применения процедуры исключения к спискам R, $\bar{R}$ наступит момент, когда после i-й операции исключения не будет исключено ни одной метки ни у одной из вершин X. Система пришла в равновесие. При этом возможны три случая:

1. $\forall x_i \in X \ |\lambda_i| = 1$, где $|\lambda_i|$ — число меток в наборе λ_i.
2. $\exists x_i \in X$ такой, что $|\lambda_i| > 1$ и $\forall x_i \in X \ |\lambda_i| \neq 0$.
3. $\exists x_i \in X$ такой, что $|\lambda_i| = 0$.

В случае 1 размещение горизонтальных отрезков найдено (согласно значениям соответствующих меток). В случае 2 для продолжения процедуры исключения необходимо вывести систему из равновесия. Для этого выполняется следующий прием (процедура V1): среди вершин ранга $R_1 \in R$ выбирается вершина x_i (корректируемая вершина), имеющая минимальное число меток и $|\lambda_i| > 1$. При наличии нескольких таких вершин выбирается вершина с максимальной степенью. У корректируемой вершины исключаются все метки, кроме минимальной, и продолжается процедура исключения. Если все вершины ранга R_1 имеют по одной

метке, то берутся вершины ранга $R_2 \subset R$ и т. д. Выбор для коррекции вершин наименьших рангов и исключение всех меток, кроме минимальной, определяется стремлением расположить горизонтальные отрезки на магистралях с наименьшим номером с целью минимизации необходимого РП. Выбор вершины x_i с максимальной степенью определяется стремлением сократить вычислительную сложность алгоритма.

В случае 3 при заданном числе магистралей размещение не найдено. При этом из графа исключается вершина с максимальной степенью и инцидентные с ней ребра и дуги. К скорректированным и ранжированным спискам вершин R, $\bar{R}$ вновь применяются процедуры исключения, а соединение, соответствующее исключенной вершине, трассируется на этапах 4 или 5. Выбор для коррекции вершины с максимальной степенью определяется стремлением облегчить трассировку оставшихся соединений. Вычислительная сложность метода исключений $O(mn)$, где m — число соединений; n — число магистралей РП для трассировки. Так, трассировка соединений приведенной выше матричной БИС составляет около 4 мин ЭВМ ЕС-1060.

Для повышения быстродействия алгоритма можно при выводе системы из равновесия корректировать несколько вершин одновременно (процедура V2). В качестве таких вершин можно выбрать, например, все или часть вершин $E \subset R_i \in R$ в случае, если вершины рангов $R_1, \ldots, R_{i-1}$ имеют по одной метке. Упорядочим вершины множества E в порядке возрастания координаты X левых концов соответствующих горизонтальных отрезков A(E). Получим список $E = (e_1, \ldots, e_m)$. Присвоим каждой вершине $e_i \in E$, начиная с первой, минимально допустимую метку из множества λ_i меток вершины e_i, т. е. расположим соответствующий горизонтальный отрезок a_i на магистраль с минимальным номером без пересечения с отрезками из множества A(E). Процедуру присваивания минимальных меток вершинам списка E можно выполнить, применяя метод исключения к множеству вершин E. При каждом выводе систем из равновесия будем корректировать по одной вершине, начиная с вершины e_1, в порядке возрастания их номеров в списке E. Однако применение процедуры V2 может увеличить число неразведенных соединений или потребует большего числа магистралей для трассировки. Поскольку процедура исключения определена над ранжированными списками R и $\bar{R}$, то при выполнении этой процедуры из графов $\Gamma(X, V)$, $\bar{\Gamma}(X, V)$ можно исключить дуги, соединяющие вершины рангов R_i с вершинами рангов R_{i+2}, R_{i+3} и т. д., что также повышает быстродействие алгоритма, где $i \geqslant 1$. Так, из графа Γ (рис. 6.5,*б*) можно исключить дуги (x_2, x_5), (x_2, x_6), (x_4, x_6).

На рис. 6.4, 6.5 приведены примеры, иллюстрирующие трассировку соединений методом исключений с применением процедур V1, V2. Для трассировки соединений (рис. 6.4,*а*) построим списки $R = (x_1, x_2, x_3, x_4, x_5)$ и $\bar{R} = (x_4, x_5, x_1, x_2, x_3)$. В табл. 6.1 приведены изменения меток после применения прямых и обратных процедур исключения к спискам R, $\bar{R}$, где 0-й шаг — начальные метки вер-

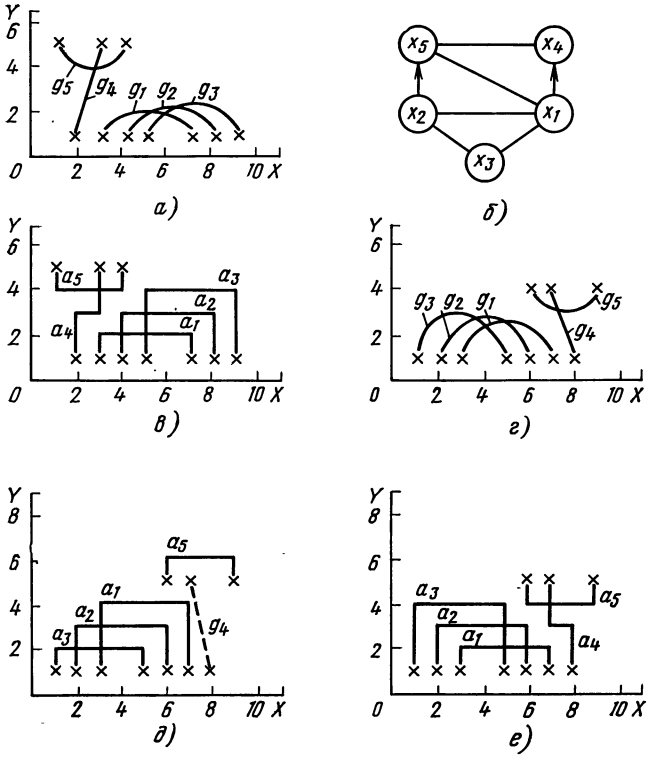


Рис. 6.4. Примеры трассировки соединений:
а, г — схемы соединений; *б* — граф Г; *в* — трассировка схемы (*а*) с применением процедур V1, V2; *д* — трассировка схемы (*г*) с применением процедуры V2; *е* — трассировка схемы (*г*) с применением процедуры V1

шины графа; *i*-й шаг — пункт процедуры исключения, соответствующий применению процедуры 1 и правила 3 к вершине $x_j \in R$ (прямая процедура исключения) и процедуры 2 и правила 3 к вершине $x_k \in \bar{R}$ (обратная процедура исключения); $j=\overline{1,5}$; $k=\overline{1,5}$. В строке «Номер шага» для каждого шага приведены вершины, к которым применяются процедура 1 или 2 и правило 3. После 1-й обратной процедуры система приходит в равновесие. Для вывода системы из равновесия применим процедуру V1: среди вершин 1-го ранга $R_1=(x_1, x_2, x_3)$ выберем вершину x_1, имеющую максимальную степень и $|\lambda_1|>1$, и оставим у нее минимальную метку ($\lambda=2$). После 2-й прямой процедуры исключения система приходит в равновесие при $|\lambda_i|=1 \forall x_i \in R$, т. е. получаем размещение горизонтальных отрезков по магистралям (рис. 6.4,*в*). Аналогичный результат получим и при выводе системы из равновесия процедурой V2. Пусть $E=(x_1, x_2, x_3)$. Тогда $\lambda_1 := 2$, $\lambda_2 := 3$, $\lambda_3 := 4$. После применения процедуры исключения получим $\lambda_4=3$, $\lambda_5=4$. При трассировке соединений (рис. 6.4,*г*) с применением процедуры V1 все соединения растрассировались в канале 1 на трех магистралях (рис. 6.4,*е*); с применением процедуры V2 три соединения растрассировались в канале 1 на трех магистралях, одно соединение в канале 2, а одно соединение вообще

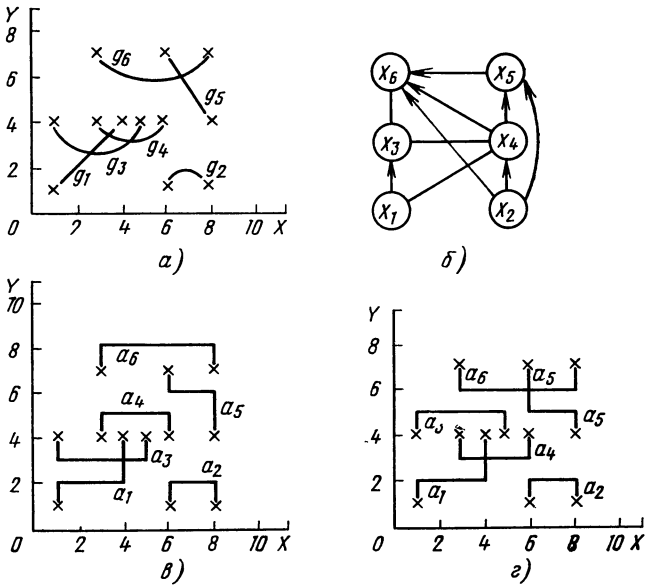


Рис. 6.5. Примеры трассировки соединений (*а*) в двух каналах с применением процедуры V2 (*в*) и с применением процедуры V1 (*г*); граф Γ (*б*) для схемы соединений (*а*)

не развилось на заданных четырех магистралях (рис. 6.4,*д*). Трассировка соединений (рис. 6.5,*а*) с применением процедуры V1 требует четыре магистрали в двух каналах (рис. 6.5,*г*), с применением процедуры V2 — пять магистралей в трех каналах (рис. 6.5,*в*).

Для сокращения времени трассировки методом исключений используется алгоритм скользящей апертуры [167]. Суть алгоритма состоит в следующем: вначале выполняется трассировка методом исключений соединений в некоторой апертуре A_1 между линейками L_0 и L_r; затем — в апертуре A_2, содержащей соединения между линейками L_1, L_{r+1}, сохраняя растрассированными соединения в канале K_1 (между линейками L_0 и L_1). После этого в апертуре A_3, содержащей соединения между линейками L_2 и L_{r+2}, оставляют растрассированными соединения в канале K_2 (между линейками L_1 и L_2) и т. д., пока не будут растрассированы соединения всех каналов. Метод скользящей апертуры позволяет уменьшить число параллельно трассируемых соединений и учесть влияние соединений, не вошедших в канал K_i, на трассировку соединений в канале K_i, с целью создания лучших условий для трассировки соединений в каналах K_{i+1}, K_{i+2} и т. д. Очевидно, чем больше величина r, тем лучше качество трассировки (меньше процент неразведенных фрагментов соединений) и тем больше время трассировки.

На этапе 2 строятся вертикальные отрезки. Здесь же для каждого бокового соединения определяются координаты x_i, y_i второго

Таблица 6.1. *Изменение меток вершин графа Г после каждого шага исключения и вывода системы из равновесия*

Вид операции		Прямая процедура			Обратная процедура		
Номер шага	0	1 x_1	2 x_2	3—5 x_3, x_4, x_5	1 x_4	2 x_5	3—5 x_1, x_2, x_3
Вершины	Начальные метки						
x_1	2, 3, 4				2,3		
x_2	2, 3, 4					2,3	
x_3	2, 3, 4						
x_4	2, 3, 4	3,4					
x_5	2, 3, 4		3,4				

Продолжение табл.

Вид операции		Вывод системы из равновесия	Прямая процедура					Обратная процедура	Результат
Номер шага	0	$\lambda_1 := 2$	1 x_1	2 x_2	3 x_3	4 x_4	5 x_5	1—5 $x_4, x_5, x_1—x_3$	
Вершины	Начальные метки								
x_1	2, 3, 4	2							2
x_2	2, 3, 4		3						3
x_3	2, 3, 4		3,4	4					4
x_4	2, 3, 4						3		3
x_5	2, 3, 4			4					4

контакта, расположенного на одной вертикали с псевдоконтактом d_i, где x_i — координата X псевдоконтакта d_i; y_i — координата Y горизонтального отрезка сегмента S_{i1} соединения i.

На этапе 3 производится разнесение горизонтальных и вертикальных отрезков по слоям с целью минимизации числа межслойных переходов и улучшения РП для доразводки соединений на этапе 4 за счет уменьшения переходных отверстий.

На этапе 4 производится трассировка боковых и не разведенных на этапе 1 соединений объемным волновым алгоритмом Ли в дискретном РП. Для каждого такого соединения i или цепи i производится построение пути или штейнеровского дерева, объединяющего фрагменты цепи i, расположенные в пределах нескольких каналов и соседних с ними периферийных элементов. Фрагмент трассировки — это множество контактов и/или проводников одной цепи, построенных на этапе 1. Волна распространяется одновре-

менно в двух слоях в пределах прямоугольника, описывающего фрагменты трассировки цепи i. В качестве начального источника распространения волны используется фрагмент K_0, расположенный ближе других к центру прямоугольника. Волна распространяется до первого встретившегося фрагмента той же цепи. Фрагмент K_0 и построенный путь являются источниками для следующей итерации распространения волны и т. д., пока не произойдет объединение всех фрагментов цепи i. Если в заданном прямоугольнике цепь не разводится, последний увеличивается влево и вправо на величину Δ, где $\Delta = 10—20$ единиц опорной сетки (задается в описании конструкции). Если и в этом случае цепь не разводится, то объединение оставшихся фрагментов цепи i осуществляется в интерактивном режиме на ИГС.

Как известно, вычислительная сложность волнового алгоритма $O(NM)$, где N — число дискретов РП всех слоев, в которых распространяется волна; M — общее число контактов всех цепей. Так, время доразводки цепи, содержащей 6 несвязных фрагментов, расположенных в прямоугольнике, охватывающем три канала, с РП $110 \times 40 \times 2$ дискретов составило около 5 с ЭВМ ЕС-1060.

На этапе 4 производится также перевод описания топологии переменных слоев, размещения, спецификации, неразведенных соединений с внутреннего языка АСП-6 в формат ИГС и запись их на магнитную ленту (МЛ) интерфейсной программой. На этапе 5 на ИГС производится чтение информации с МЛ, доразводка неразведенных на этапах 1, 4 соединений или полного проектирования топологии в интерактивном режиме и запись описания топологии переменных слоев, спецификации и размещения элементов в координатах опорной сетки на МЛ в формате ИГС. При коррекции топологии используются эскизы переменных слоев, изготовленные средствами ИГС (на графопостроителе ЭМ-7022 или ЭМ-732А), и графические дисплеи. При этом БЭ представляются прямоугольниками с условными контактами, проводники — средними линиями, межслойные переходы — своими центрами в опорной сетке. Все процедуры коррекции проводятся также в опорной сетке. В процессе коррекции допускается переразмещение элементов, сигналов среди инвариантных контактов 1-й глубины (т. е. переразмещение сигналов среди контактов схем ИЛИ либо схем И), удаление и дополнение проводников и прочее в соответствии с возможностями ИГС.

При интерактивном проектировании топологии с помощью цифрового дисплея для каждой линейки вводятся номер линейки и список имен внутренних элементов (в том числе, ПрЭ) в той последовательности, в которой предполагается их размещение в данной линейке. Для каждого внешнего элемента вводится соответствующий номер внешней контактной площадки. Результаты размещения отображаются на экране графического дисплея в виде прямоугольников.

6.5. ПОДСИСТЕМА ВЕРИФИКАЦИИ ТОПОЛОГИИ АСП-66

Поскольку в процессе доразводки или полностью интерактивного проектирования топологии возможны ошибки, на ЕС ЭВМ производится верификация топологии [160]. Исходной информацией для подсистемы АСП-66 является описание: схемы матричной БИС или блока i-го уровня на языке АСП-6, раскрытого до элементов необходимого уровня иерархии; конструкции элементов и БМК; топологии переменных слоев, спецификация и размещение в формате ИГС.

Уровень раскрытия описания и верификации может иметь любую глубину и определяется сложностью матричной БИС или блока i-го уровня (числом соединений или задействованных в схеме контактов) и производительностью ЭВМ. В системе АСП-66 раскрытие описания БИС производится до уровня БЭ и верификация производится всей БИС, что возможно ввиду малого времени верификации этой системой реальных матричных БИС (≈ 1 мин работы ЭВМ ЕС-1060 для БИС с 500 БЭ, содержащих 3000—4000 контактов, $\approx 10^4$ транзисторов).

Подсистема АСП-66 включает четыре процедуры:

1. Перевод описания топологии из формата ИГС во внутренний язык АСП-6.

2. Формирование списка *S1* в координатах опорной сетки на основании исходного описания схемы на языке АСП-6 и полученных с ИГС размещения и спецификации элементов.

3. Построение из описания топологии матричной БИС, полученного с ИГС, списка цепей *S2* в координатах опорной сетки.

4. Установление соответствия списков цепей *S1*, *S2* и выдача результатов контроля на уровне исходного описания схемы на АЦПУ или цифровые дисплеи ЕС ЭВМ.

В процедуре 1 интерфейсной программой ЕС ЭВМ производится чтение с МЛ и перевод с формата ИГС во внутренний формат АСП-6 описания топологии переменных слоев V, спецификации и размещения элементов, заданных в единицах опорной сетки, где $V = \{П, Г1, Г2, В1, В2\}$, П — множество межслойных переходов, заданных координатами их центров; Г1, Г2 (В1, В2) — соответственно множество горизонтальных (вертикальных) отрезков слоев 1 и 2, заданных средними линиями.

Процедурой 2 на основании исходного описания схемы (списка цепей, спецификации), описания конструкции БМК и элементов (включающих описание инвариантных контактов), находящихся в БД ЕС ЭВМ, и размещения элементов и спецификации, полученных с ИГС, производится сравнение спецификаций, формирование списка цепей *S1* (включающего инвариантные контакты), дополнение множества V множеством контактов К, заданных своими центрами и необходимыми элементами топологии, взятыми из описания конструкции элементов (в том числе, вертикальными отрезками БЭ, соединяющими нижние и верхние контакты БЭ, межслойными переходами на контактах). Здесь же проводится объедине-

ние в один отрезок пересекающихся отрезков одной ориентации одного слоя и упорядочение методом «вычерпывания» [103] элементов топологии внутри каждого из множеств К, П, Г1, Г2, В1, В2 по возрастанию одной из координат. Метод «вычерпывания» состоит в разбиении выбранного множества на подмножества с равными значениями одной координаты (Y — для элементов множеств К, П, Г1, Г2; X — для элементов В1, В2) и упорядочения элементов внутри подмножеств по возрастанию другой координаты (по координате X для элементов множеств К, П; по координате X левого конца для горизонтальных отрезков, по координате Y нижнего конца для вертикальных отрезков).

Исходной информацией для процедуры 3 является упорядоченное множество элементов топологии $V=\{К, П, Г1, Г2, В1, В2\}$. Пронумеруем элементы множества V целыми числами от 1 до N так, чтобы номера элементов множества К были меньше номеров других элементов. Получим множество элементов топологии $V=(v_1, v_2, \ldots, v_n)$. Поставим в соответствие элементам множества V список $L=(1, 2, \ldots, N)$ такой, что элементу $v_i \in V$ соответствует элемент $i \in L$. Пометим каждый элемент $i \in L$ меткой (адресной ссылкой) λ_i, причем начальное состояние $\lambda_i = i \ \forall i = \overline{1, N}$.

Пусть $L(i_1) = (i_1, i_2, \ldots, i_n) \subset L$ — список элементов, соответствующий элементам топологии $V(i_1) = (v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ некоторой цепи на k-м шаге формирования списка $S2$, где $i_1 > i_2 \ldots > i_n$; $\lambda_{i1} = i_2$, $\lambda_{i2} = i_3, \ldots, \lambda_{i, n-1} = i_n, \lambda_{in} = i_n$, λ_{ij} — адресная ссылка на следующий элемент списка $L(i_1)$; $n \geqslant 1$. Очевидно, что элемент i_n (корень) имеет наименьшее значение метки $\lambda_{in} = i_n$; при $n=1$ $L(i_1) = i_1$, $\lambda_{i1} = i_1$. Для нахождения элементов списка $L(i_1)$ достаточно просмотреть по адресным ссылкам список L, начиная с элемента i_1.

Рассмотрим алгоритм формирования списка $S2$. Пусть элементы $\{v_i, v_j\} \subset V$ пересекаются. Тогда:

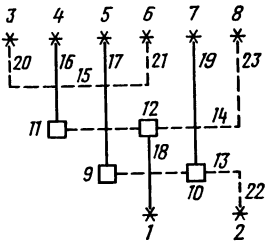
1. Находится элемент $t = \min\{i_n, j_m\}$, где i_n, j_m — корни списков $L(i)$, $L(j)$ соответственно.

2. $\lambda_l := t \ \forall l \in \{L(i) \cup L(j)\} \setminus t$. Операции по пп. 1, 2 применяются ко всем пересекающимся элементам множества V в порядке их расположения в этом множестве в следующих комбинациях: К, Г1; К, Г2; К, В1; К, В2; П, Г1; П, Г2; П, В1; П, В2, Г1, В1; Г2, В2.

3. $\lambda_i := a_i \ \forall i = \overline{1, N}$, начиная с $i=1$, где a_i — корень списка $L(i)$, полученного после выполнения пп. 1, 2. Все контакты, помеченные одной меткой, составят одну цепь, а множество таких цепей составят список $S2$. Все элементы топологии, помеченные одной меткой, будут принадлежать одной цепи.

На рис. 6.6 приведен эскиз топологии трех цепей, где ✳ — контакт; □ — межслойный переход; непрерывные и штриховые линии — проводники верхнего и нижнего слоев соответственно. Элементы топологии обозначены номерами от 1 до 23 в соответствии с принятым выше их расположением в упорядоченном множестве V и списке L. В табл. 6.2 представлен список $L=(1, 2, \ldots, 23)$, первоначальное значение меток (Λ_0), значение меток после пересечения элементов множеств К, В1; К, В2; П, Г2; П, В1; Г2, В2 (элементы других множеств не пере-

Рис. 6.6. Эскиз топологии для ее верификации: $V=\{К, П, Г_2, В_1, В_2\}$, $К=\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$, $П=\{v_9, v_{10}, v_{11}, v_{12}\}$, $Г_2=\{v_{13}, v_{14}, v_{15}\}$, $В_1=\{v_{16}, v_{17}, v_{18}, v_{19}\}$, $В_2=\{v_{20}, v_{21}, v_{22}, v_{23}\}$



секаются). Последние приведены для иллюстрации алгоритма формирования списка *S2*, хотя в действительности хранится только одна (последняя) метка в строке меток. На 1-м шаге пп. 1, 2 алгоритма пересекаются элементы v_1, v_{18} ($v_1 \cap v_{18}$). Тогда $L(1)=1$, $L(18)=18$, $t=1$, $L(1, 18)=(1, 18)$, $\lambda_{18}:=1$. На 2-м шаге $v_4 \cap v_{16}$, $L(4)=4$, $L(16)=16$, $t=4$, $L(4, 16)=(4, 16)$, $\lambda_{16}:=4$. На 3-м шаге $v_5 \cap v_{17}$, $L(5)=5$, $L(17)=17$, $t=5$, $L(5, 17)=(5, 17)$, $\lambda_{17}:=5$. На 14-м шаге $v_{10} \cap v_{19}$, $L(10)=(10, 9, 5)$, $L(19)=(19, 7)$, $t=5$, $L(10, 19)=(10, 9, 5, 19, 7)$, $\lambda_{10}:=\lambda_9:=\lambda_{19}:=\lambda_7:=5$. На 18-м шаге $v_{14} \cap v_{23}$, $L(14)=(14, 11, 1)$, $L(23)=(23, 8)$, $t=1$, $L(14, 23)=(14, 11, 1, 23, 8)$, $\lambda_{14}:=\lambda_{11}:=\lambda_1:=\lambda_{23}:=\lambda_8:=1$. На 20-м (последнем шаге) $v_{15} \cap v_{21}$, $L(15)=(15, 3)$, $L(21)=(21, 6)$, $t=3$, $L(15, 21)=(15, 3, 21, 6)$, $\lambda_{15}:=\lambda_{21}:=\lambda_6:=3$.

Пункт 3 алгоритма: $\lambda_1:=a_1=1$, так как $L(1)=1$; $\lambda_2:=a_2=2$, так как $L(2)=2$; $\lambda_3:=a_3=3$, так как $L(3)=3$; $\lambda_4:=a_4=1$, так как $L(4)=(4, 1)$; ...; $\lambda_7:=a_7=2$, так как $L(7)=(7, 5, 2)$; ...; $\lambda_{19}:=a_{19}=2$, так как $L(19)=(19, 5, 2)$ и т. д.

В результате получим пометку Λ_1 (табл. 6.2), где цепь 1 : 1, 4, 8 с меткой $\lambda_1=1$; цепь 2: 2, 5, 7 с меткой $\lambda_2=2$; цепь 3: 3, 6 с меткой $\lambda_3=3$.

Для поиска пересечений множество элементов $Г\alpha[В\alpha)$ разбивается на упорядоченные по оси $X[Y]$ подмножества $Г\alpha(X)$ $[В\alpha(Y)]$ с равными значениями координат $Y[X]$, где $\alpha \in \{1, 2\}$. Поиск пересечений элементов $v_i \in \{К, П\}$ с элементами $Г\alpha$ $[В\alpha]$ производится посредством просмотра элементов подмножества $Г\alpha(X_i)$ $[В\alpha(Y_i)]$, где $Г\alpha(X_i)$, $В\alpha(Y_i)$ — соответственно упорядоченные

Таблица 6.2. *Значение меток* $\{\lambda_i\}$ *при формировании списка S2*

L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Λ_0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
К, В1 *																4	5	1	7				
К, В2																				3	6	2	8
П, Г2										9		11	9	11									
П, В1							5		5	5	4								5				
П, В1				1							1	1											
Г2, В2					2	3		1	2				2	1	3						3		1
Λ_1	1	2	3	1	2	3	5	1	2	5	1	1	2	1	3	4	5	1	5	3	3	2	1
Λ_2	1	2	3	1	2	3	2	1	2	2	1	1	2	1	3	1	2	1	2	3	3	2	1

* В строках 2—7 приведены значения только изменяющихся меток.

подмножества элементов с координатами X_i, Y_i; X_i, Y_i — координаты элемента v_i. Поиск пересечения элемента $v_i \in \Gamma\alpha$ с элементами $B\alpha$ производится посредством выбора для каждой горизонтали v_i подмножества вертикалей $B\alpha(X)$, для которых выполняются условия $x_1 \leqslant X \leqslant x_2$, где x_1, x_2 — координаты левого и правого концов горизонтали v_i соответственно.

Процедура 4 производит установление соответствия списка *S1* списку *S2* методом хеширования [103]. Хеширование — это метод поиска, использующий преобразование ключа d_i в адрес i хеш-таблицы $H=[t_1, \ldots, t_i, \ldots, t_N]$ с помощью хеш-функции $i=h(d_i)$, где t_i — строка таблицы с адресом i. Пусть $S1=\{A_i\}$, $S2=\{B_j\}$, где $A_i=\{A_{i1}, \ldots, A_{ij}, \ldots, A_{im_l}\}$ — множество контактов цепи i (включая инвариантные контакты); $A_{ij}=\{a_{ij1}, \ldots, a_{ijk}, \ldots, a_{ijn_f}\}$ — группа контактов, инвариантных контакту j цепи i; $a_{ijk}=(X_{ijk}, Y_{ijk})$ — контакт K, инвариантный контакту j цепи i; $B=\{\lambda_l b_{l1}, \ldots, \lambda_l b_{lr}, \ldots, \lambda_l b_{ln_l}\}$ — множество контактов цепи l, меченных меткой λ_l; $B_{lr}=(x_{lr}, y_{lr})$ — контакт r цепи l. Перепишем содержимое списка в хеш-таблицу H. С помощью хеш-функции $i=h(x_{lr}, y_{lr})$ и алгоритма разрешения коллизий $j=f(i, x_{lr}, y_{lr})$ определяется адрес строки хеш-таблицы H, в которой формируется запись $\lambda_l x_{lr} y_{lr}$ (две координаты рассматриваются как одно двоичное число, соответствующее ключу d_i). Для каждой цепи, выбирая последовательно контакты a_{ijk} из списка *S1*, осуществляется с помощью хеш-функции $i=h(x_{ijk}y_{ijk})$ и функции размещения коллизий $j=f(i, x_{ijk}y_{ijk})$ поиск в хеш-таблице H координаты, у которой $x_{ijk}y_{ijk}=x_{lr}y_{lr}$. Когда координата x_{ijk}, y_{ijk} найдена, из списка *S1* вычеркиваются контакты $A_{ij} \setminus a_{ijk}$, инвариантные контакту a_{ijk}. Из групп инвариантных контактов других цепей вычеркивается контакт a_{ijk}, в хеш-таблице H стирается строка $\lambda_l x_{lr} y_{lr}$. Если при обработке контактов текущей цепи сменилось значение λ_l или для группы инвариантных контактов не найден соответствующий контакт, то в файл ошибок заносится информация о выявленной ошибке. После окончания работы со списком *S1* просматриваются все строки хеш-таблицы H и в файл ошибок производится запись для каждой обнаруженной непустой строки.

В системе АСП-66 реализована хеш-функция, которая определяет адрес i как остаток от деления двоичного числа xy на максимальное простое число, не превышающее число строк N в хеш-таблице, где $N=1,5\,M$; M — число контактов списка *S2*. Разрешение коллизий производится линейным поиском ближайшей свободной строки j к строке i.

Система АСП-66 позволяет обнаруживать следующие ошибки в топологии: разрывы в цепях или их отсутствие в топологии; замыкания между цепями, в том числе замыкания на шины питания; наличие в топологии цепей, не присутствующих в исходном описании схемы; наложение элементов друг на друга; несоответствие типа элемента, указанного в описании схемы, типу, помещенному в

топологию; несоответствие размещения периферийных элементов номерам внешних контактов.

По результатам верификации топологии производится коррекция топологии на ИГС, затем повторная верификация топологии и т. д., пока не будут исправлены все ошибки.

Как следует из построения и многолетнего опыта верификации топологии в системах проектирования микросборок, печатных плат, матричных БИС АСП-2, АСП-52 (АСП-51М), АСП-6М [160, 161], в которых реализован описанный выше алгоритм верификации топологии, вычислительная сложность последнего $O(M)$, где M — число контактов исходного описания схемы.

Как следует из описания алгоритмов, система АСП-66 может быть применена для верификации топологии блоков любого уровня иерархии, заданных в базисе более низкого уровня при наличии информации, описанной в начале данного раздела. Тем самым она позволяет производить иерархическую верификацию топологии, т. е. верификацию топологии блоков 1-го уровня, затем 2-го и т. д. и, наконец, матричной БИС в целом, представленной в базисе блоков более низкого уровня.

6.6. ВРЕМЕННАЯ ВЕРИФИКАЦИЯ ТОПОЛОГИИ

После верификации топологии производится временна́я верификация матричных БИС, содержащая следующие процедуры:

1. Вычисление емкостного и омического сопротивлений каждой цепи по результатам проектирования топологии и выдача соответствующей диагностической таблицы.

2. Моделирование схемы с учетом задержки сигналов в топологии при заданном режиме (температуре окружающей среды и напряжении источников питания).

В 1-й процедуре согласно алгоритму формирования списка *S2*, рассмотренного в § 6.5, производится восстановление топологии каждой цепи. При этом список L просматривается до конца, начиная с $M+1$-го элемента, где M — число контактов схемы. В результате все элементы топологии, помеченные одной меткой, будут принадлежать одной цепи. Например (рис. 6.6), просматривая список L с 9-го элемента, получим пометку элементов Λ_2 списка L (табл. 6.2), где элементы с меткой $\lambda_1=1$ (11, 12, 14, 16, 18, 23), с меткой $\lambda_2=2$ (9, 10, 13, 17, 19, 22), с меткой $\lambda_3=3$ (15, 20, 21) есть множество горизонтальных, вертикальных отрезков и многослойных переходов соответственно 1, 2 и 3 цепи.

Используя упорядоченное множество V (содержащее множество проводников, координаты их начала и конца, множество переходов), ширину и удельную емкость и сопротивление проводников в каждом слое, размеры межслойных переходов и их удельную емкость, взятые из описания конструкции матричных БИС, подсчитываются площади топологических элементов, емкостное и омическое сопротивления каждой цепи. Эта информация (диагностиче-

ская таблица) заносится в базу данных, выводится на АЦПУ или дисплей и используется для анализа результатов проектирования. По результатам анализа возможна коррекция топологии с последующей верификацией топологии и временной верификацией или перепроектирование матричных БИС.

Во 2-й процедуре по описанию схемы, диагностической таблице находится емкостная нагрузка на выходе каждого БЭ, определяемая емкостью топологии соответствующей цепи и входной емкостью других БЭ, входящих в цепь. По описанию электрических характеристик БЭ, приведенных в § 6.1 и емкостной нагрузке определяется задержка на каждом БЭ схемы при заданном режиме. Затем производится моделирование матричных БИС на тех же тестах с помощью подсистемы АСП-65 с учетом вычисленных задержек на БЭ. По результатам моделирования возможны коррекции, после чего производится перепроектирование с необходимого уровня.

Процедура 1 входит в состав подсистемы АСП-66, процедура 2 — в состав подсистемы АСП-65.

6.7. ПОДГОТОВКА ДАННЫХ ДЛЯ ГРАФОПОСТРОИТЕЛЕЙ, ГЕНЕРАТОРОВ ИЗОБРАЖЕНИЙ, АВТОМАТИЗИРОВАННЫХ СИСТЕМ КОНТРОЛЯ

После временной верификации на ИГС производится вычерчивание эскизов переменных слоев (как при коррекции топологии), которые анализируются, при необходимости корректируются (с последующей верификацией). При положительном результате анализа выполняется слияние файлов переменных слоев топологии матричных БИС с топологией БЭ и необходимыми топологическими элементами БМК, трансляция информации и запись ее на МЛ для генераторов изображений ЭМ-559, ЭМ-5009, изготавливающих промежуточные фотошаблоны (ПФО) в масштабе 10 : 1. В качестве послойных чертежей переменных слоев используются фоточертежи, получаемые с ПФО.

Для изготовления чертежей СхЭ с ЕС ЭВМ на ИГС передается на МЛ описание схемы (список цепей и спецификация) в формате ИГС. С помощью графических средств ИГС и системы АСКОР [169], библиотеки описаний графики элементов (БЭ, блоков любого уровня иерархии), описания базы чертежа и описания СхЭ производится в автоинтерактивном режиме проектирование чертежа СхЭ с учетом требований ЕСКД и изготовление его на графопостроителе ЕС-7907 («Дигиграф»). При этом выполняется ручное размещение элементов (чтобы обеспечить читаемость схемы) и автоматическая трассировка соединений, объединение в жгут, доразводка с использованием оператора «резиновый жгут», обеспечивающего сохранность и отображение на дисплее элементов и соединений при любых переразмещениях элементов и трасс, соответствие изготовленного чертежа СхЭ и топологии.

Библиотека описаний графики элементов в ИГС может содержать графическое представление БЭ и блоков любого уровня иерархии, что обеспечивает проектирование и изготовление чертежей СхЭ любых уровней иерархии. Параллельно с изготовлением чертежа СхЭ производится трансляция и запись на МЛ теста моделирования для автоматизированной системы контроля (АСК) матричных БИС «Вестник». Эта процедура входит в состав подсистемы АСП-65.

6.8. ПОДСИСТЕМА УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ АСП-6БД

Подсистема АСП-6БД предназначена для надежного хранения проблемных программ системы АСП-6М, описаний устройств (набора матричных БИС и блоков различных уровней иерархии) в десятки и сотни мегабайт на различных этапах проектирования и оперативного доступа к ним. Описание матричных БИС включает:

1. БЭ, содержащие описания: а) функционирования БЭ в многозначной логике; б) электрических характеристик БЭ; в) конструкции БЭ; г) схемы БЭ на уровне компонентов (для схемотехнического моделирования БЭ или блока); д) графического представления БЭ для вычерчивания эскиза топологии и СхЭ.
2. Блоки всех уровней иерархии, содержащие описания: а) схемы блока (список цепей и спецификация); б) размещения элементов и внешних сигналов по внешним контактам блока, трассировку соединений; в) конструкции блока (координаты внешних контактов, размеры блока и др.).
3. Описание СхЭ матричной БИС.
4. Описание тестов моделирования матричной БИС.
5. Описание конструкций матричной БИС.
6. Описание результатов проектирования (временные диаграммы, размещение элементов, трассировка соединений, диагностические таблицы, тесты для АСК) и верификации топологии блоков различных уровней и матричной БИС в целом и другие данные.

Данные по пп.1, 2а, 3, 4, 5 являются исходными. Данные по пп. 2б, 2в, 6 формируются в процессе выполнения соответствующих прикладных программ системы АСП-6М (например, размещение элементов и внешних сигналов по контактам, трассировка соединений блока i-го уровня иерархии, размеры блока). Однако некоторые из них или все данные могут быть заданы как исходные.

База данных (БД) является ядром САПР, объединяющим отдельные проблемные программы в единую систему. Использование централизованной БД упрощает их взаимодействие и обмен данными, а включение в нее сведений о пользователях САПР, датах начала и завершения этапов проектирования, плановых сроках позволяет автоматизировать контроль процесса проектирования матричных БИС. Описания устройств в БД АСП-6М организованы в виде совокупности неструктурированных массивов переменной длины. Этим БД АСП-6М существенно отличается от БД инфор-

мационно-поисковой системы (ИПС) и АСУ, в которых информация детально структурирована. Математическое обеспечение БД оперирует массивами как объектами, не имеющими внутренней структуры, хотя их объем может достигать нескольких мегабайт. В отдельных случаях большие массивы могут обрабатываться последовательным образом, причем структурирование информации осуществляется проблемными программами САПР. Массивы БД содержат произвольную информацию в символьном, шестнадцатеричном или каком-либо ином представлении. Математическое обеспечение БД не накладывает ограничений на синтаксический или сематический характер информации.

В [170] показано, что выбранный способ организации данных в наибольшей степени отвечает требованиям САПР БИС и СБИС, содержащим сложные проблемные программы. В соответствии с алгоритмами САПР АСП-6М к отдельным структурным единицам информации (описания контактов, БЭ, блоков и т. п.) происходит 10^9—10^{10} обращений при выполнении программы. Возможный путь достижения приемлемого быстродействия проблемных программ САПР — обработка данных непосредственно в оперативной памяти ЭВМ в отличие от ИПС и АСУ, которые обращаются к СУБД за каждым элементом данных. Именно это явилось причиной организации БД АСП-6М в виде массивов, готовых для загрузки в оперативную память ЭВМ.

Устройства, проектируемые с помощью АСП-6М, представлены в БД заголовками описаний. Заголовок имеет фиксированный формат и содержит наименования устройства, сведения о типе информации, ее объеме, датах ее ввода в БД, частоте обращений к данным и другие характеристики. Между заголовками имеются связи для описания вхождений блоков друг в друга. Они позволяют быстро определить, на каких блоках отразится внесение изменений в заданный блок. В БД включены описания всех пользователей САПР. Они связаны с заголовками описаний устройств связями, устанавливающими, кем разрабатываются устройства и кто имеет доступ к информации.

Основная информация об устройствах организована в виде массивов. Массив состоит из заголовка фиксированного формата и информационной части произвольной длины. Заголовок содержит сведения о типе информации, ее объеме, датах создания и модификации. В заголовке определено поле (для записи из проблемных программ) дополнительных данных о массиве. Их состав зависит от типа информации, хранимой в массиве. В каждом массиве хранятся данные, относящиеся только к одному из проектируемых устройств. Вся информация об устройстве распределена по массивам так, чтобы каждый массив содержал данные одного типа и мог использоваться проблемными программами как в совокупности с другими, так и независимо от них. Внутренняя организация массивов может иметь сложную структуру со ссылками на данные других массивов.

Часть массивов в БД связана непосредственно с заголовками описаний устройств, информацию о которых они содержат. Это массивы, тип которых допускает наличие одного однотипного массива для каждого устройства (например, словари элементов и внешних контактов, списки цепей и т. д.). Остальные массивы содержат такую информацию, что возможно наличие нескольких однотипных массивов, относящихся к одному устройству (например, описания тестовых воздействий, результаты моделирования, описания контрольных точек для моделирования и т. п.). В БД все однотипные массивы, соответствующие одному устройству, объединены в одно множество и связаны с заголовком этого множества, который содержит его общие характеристики (число массивов, их суммарный объем, частоту обращений, дату последнего обращения и т. п.) и связан с заголовком описания устройства. Двухуровневая иерархическая организация связи информационных массивов с заголовками описаний устройств позволяет структурировать информацию об устройстве и обеспечивает доступ к массивам.

Программное обеспечение БД АСП-6М состоит из модулей доступа и автономных программ обслуживания БД. Модули доступа подключаются к проблемной программе на этапе редактирования связей. Обращение к ним производится по оператору CALL с параметрами. Модули доступа выполняют следующие операции над информационными массивами: чтение из БД в оперативную память, загрузка из оперативной памяти в БД, удаление из БД, последовательное чтение из БД в оперативную память, последовательная загрузка из оперативной памяти в БД. Возможность последовательного чтения и загрузки позволяет проблемной программе работать с массивами, объем которых превышает размеры доступной оперативной памяти. В этом случае проблемная программа сама задает длину очередной загружаемой или считываемой части массива, исходя из его внутренней структуры и наличия свободной оперативной памяти.

Программы обслуживания предназначены для контроля за состоянием БД, выполнения сервисных функций и отладки проблемных программ САПР, работающих с БД. Они позволяют тестировать БД на отсутствие в ней сбоев и ошибок, распечатывать оглавление БД, переписывать массивы из одной БД в другую через промежуточный набор данных, удалять массивы из БД, загружать массивы в БД с дисплеев, выводить на печать содержимое массивов БД.

Математическое обеспечение БД АСП-6М имеет ряд особенностей, обусловленных ее использованием в качестве основы для САПР коллективного пользования. Оно позволяет работать с БД в мультипрограммном режиме одновременно нескольким проблемным программам, обеспечивая необходимую целостность данных. Каждая проблемная программа может инициировать любое число операций последовательной загрузки (чтения) массивов, не завершая загрузку (чтение) других массивов. Это позволяет формировать или просматривать одновременно несколько массивов, объем

которых превышает размеры доступной оперативной памяти. Благодаря динамическому распределению памяти в БД, при инициации операции последовательной загрузки массива не требуется задавать его объем. Память, освободившаяся в БД в результате удаления массивов, используется для загрузки вновь создаваемых массивов. Поэтому БД не требует периодической реорганизации и может эксплуатироваться в условиях интенсивного обновления информации. Эффективность использования памяти в БД практически не зависит от числа и размеров информационных массивов.

СУБД АСП-6БД реализована на ЕС ЭВМ в среде ОС.7.0 на базе сетевой СУБД СЕТОР [171]. Несмотря на то, что СУБД СЕТОР ориентирована в основном на создание БД АСУ и ИПС, она обладает хорошими возможностями для создания МО, способного работать с массивами переменной длины. Они основаны на особенностях физической организации данных в СУБД СЕТОР. Хеширование записей в основных файлах обеспечивает быстрый поиск их по типу, а списочная организация зависимых файлов позволяет реализовывать сколь угодно длинные цепочки страниц массивов и распределять память в БД динамическим образом. Простота и эффективность организации связей между записями основных и зависимых файлов позволяет создавать сложные структуры данных, обладающие необходимыми характеристиками. Отметим основные особенности СУБД СЕТОР, определившие ее выбор в качестве основы для СУБД АСП-6БД: поддержка мультипрограммного режима работы проблемных программ, малые требования к объему оперативной памяти (ядро СУБД без буферов ввода-вывода занимает около 40 кбайт); простота освоения и эксплуатации; устойчивость к сбоям ЭВМ и ошибкам в проблемных программах; наличие средств журнализации и восстановления БД; простота сопряжения с другими системами, в частности с диалоговой системой коллективного пользования CMS, выбранной для обеспечения интерактивного режима работы в САПР АСП-6М; развитые средства модификации структуры БД путем расширения форматов записей и введения новых файлов и связей; возможность выбора для проблемных программ любого алгоритмического языка (Ассемблер, Фортран, PL/1 и т. д.).

Описания пользователей САПР АСП-6БД и заголовки описаний устройств, массивов и множеств однотипных массивов реализованы в виде записей соответствующих основных файлов. Информационные части массивов реализованы списками по 2048 байт на странице. Соответствие между заголовками и информационными частями массивов представлены связью файла заголовков с файлом страниц.

СУБД СЕТОР не допускает прямых связей между основными файлами БД. Поэтому для реализации связей между описаниями пользователей, заголовками описаний устройств, массивами и множествами однотипных массивов в БД введен вспомогательный зависимый файл-связка. Его записи не содержат информационных полей, а играют роль связок. Каждая такая запись связана с за-

писями двух основных файлов. Считается, что между записями основных файлов установлена связь, если они связаны с одной и той же записью-связкой. Этот же механизм решает проблему связей типа «многие ко многим», для реализации которых СУБД СЕТОР не имеет специальных средств.

Модули доступа и обслуживающие программы СУБД АСП-6БД реализованы на языке Ассемблер с использованием средств *языка манипулирования данными* (ЯМД) СУБД СЕТОР. Модули доступа к БД обеспечивают независимость проблемных программ от особенностей ЯМД СУБД СЕТОР и размера страниц информационных частей массивов. Это представляет в будущем возможность быстрой настройки прикладных программ на новые СУБД. Замена СУБД потребует коррекции только модулей доступа к БД.

Опыт работы с АСП-6БД подтвердил правильность выбранных принципов и эффективность БД на этапах разработки и эксплуатации проблемных программ САПР. Математическое обеспечение БД не требует больших объемов оперативной памяти. Ядро СУБД СЕТОР с буферами ввода-вывода занимает 80К байт. Проблемной программе, взаимодействующей с БД, требуется 7К байт для связи с ядром СУБД СЕТОР и 1—2К байт на каждый модуль доступа. Математическое обеспечение БД представляет быстрый доступ проблемных программ к информационным массивам. Чтение из БД массива объемом в 10К байт на ЭВМ ЕС-1060 выполняет за 0,01—0,04 с.

6.9. ОПЫТ ЭКСПЛУАТАЦИИ САПР АСП-6М

В табл. 6.3 даны параметры БМК линейчатого типа. В табл. 6.4 приведены временные характеристики процедур САПР АСП-6М при проектировании на БМК В3 матричной БИС, имеющей следующие характеристики:

Число БЭ (число вентилей)	462 (1200)
Число цепей	427
Число соединений	1284
Общее число контактов в цепях	1724
Число внешних контактов	57

Таблица 6.3. *Типы и параметры БМК*

Параметр БМК	Тип БМК		
	В1	В2	В3
Число условных вентилей	547	832	1750
Число линеек	8	10	13
Число магистралей во внутренних каналах	10	10	11
Число магистралей в горизонтальных периферийных каналах	6	6	10
Число магистралей в вертикальных периферийных каналах	1	1	2
Число внешних контактов	62	62	62

Таблица 6.4. *Временные характеристики процедур САПР АСП-6М*

Процедура	Время ЕС-1060
Моделирование на 312 векторах	10 с
Размещение БЭ и внешних сигналов по внешним контактам	6 мин
Трассировка соединений методом исключений с применением процедуры V1 и процент (число) неразведенных фрагментов соединений	16 мин 0,5% (6)
Трассировка соединений методом исключений с применением процедуры V2 и процент (число) неразведенных фрагментов соединений	1 мин 1% (11)
Верификация топологии	38 с

Под фрагментом соединения табл. 6.4 понимается часть соединения, расположенного между контактами (в том числе, контактами ПрЭ) двух соседних линеек. Процент неразведенных фрагментов соединения указан относительно числа соединений. Время моделирования дано с учетом трансляции описания схемы и тестов; процент использования транзисторов в линейках (процент заполнения кристалла) составляет 61%. Для матричных БИС аналогичной сложности (с 50—70%-ным заполнением) цикл описания, ввода описания схемы и тестов, моделирования составляет 1—2 недели при ежедневных 1—3 ч сеансах работы; цикл проектирования и верификации топологии с интерактивной доразводкой и коррекцией топологии, изготовлением эскизов топологии, подготовкой информации на МЛ для генератора изображений, выпуском и оформлением документации (листинги и МЛ описания схемы после проектирования топологии, тесты для АСК, МЛ описания топологии) составляет 1—2 недели. При этом параллельно проектируется 10—20 матричных БИС. При необходимости цикл проектирования отдельных БИС может быть сокращен до нескольких дней при создании соответствующих условий. Все процедуры в системе АСП-6М производятся в мультипрограммном режиме с широким использованием дисплеев.

На рис. 6.7 приведен фрагмент эскиза топологии матричных БИС БМК В3, где БЭ представлены в виде прямоугольников с условными контактами и проводниками в средних линиях.

Как следует из описания алгоритмов и опыта эксплуатации САПР АСП-6М, вычислительная сложность алгоритмов $O(N)$, где N — число условных вентилей матричных БИС; она может быть использована для проектирования матричных БИС сложностью до $\sim$10 тыс. БЭ ($\sim$30 тыс. вентилей, 120—150 тыс. транзисторов) на ЭВМ типа ЕС-1060, ЕС-1066, ЕС-1087 (с производительностью до 10^7 опер./с). При этом необходимо широко использовать иерархическое описание и выполнять раскрытие описания при трансляции для размещения элементов до блоков 1-го уровня. Это позволит производить 3-уровневое иерархическое размещение (БЭ, блок 1-го уровня, линейка). Для трассировки соединений и верифика-

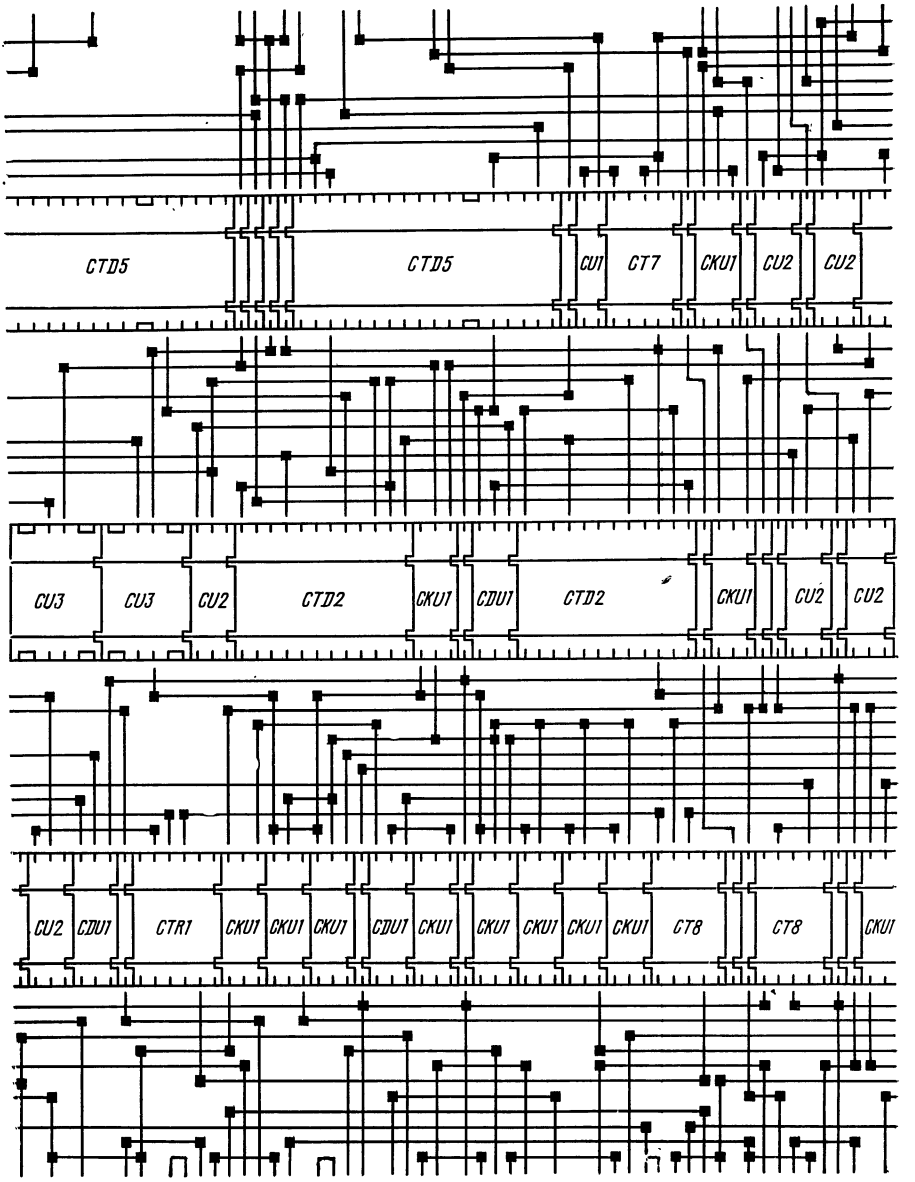


Рис. 6.7. Фрагмент эскиза топологии матричных БИС линейчатого типа

ции топологии раскрытие описания осуществляется до БЭ и указанные процедуры производятся для матричных БИС в целом.

Для проектирования более сложных матричных БИС необходимо применение многоуровневого проектирования. С этой целью ведутся работы по созданию многоуровневой подсистемы проек-

тирования топологии (размещение разногабаритных блоков i-го уровня, трассировка соединений между блоками i-го уровня), так как во всех остальных подсистемах САПР АСП-6М реализован принцип иерархического проектирования. Для размещения блоков i-го уровня реализуется автоинтерактивный метод, обеспечивающий как автоматическое, так и интерактивное размещение блоков. Вследствие высокой скорости параллельной трассировки соединений методом исключений с использованием скользящей апертуры допускается для трассировки соединений раскрытие описания блока i-го уровня до БЭ и трассировка соединений между контактами БЭ, если производительность ЭВМ это позволяет. При этом качество трассировки лучше по сравнению с иерархической трассировкой. Для трассировки соединений между разногабаритными блоками i-го уровня реализован метод касательных [172—177], обеспечивающий высокое быстродействие при построении ортогонального *минимального* (или близкого к нему) *штейнеровского дерева* (МШД) на двухслойном рабочем дискретном поле практически не ограниченных размеров. Вычислительная сложность последнего $O(M)$, где M — число контактов в описании блока $(i+1)$-го уровня и не зависит от числа дискретов и размера рабочего поля. Для ЭВМ ЕС-1060 время трассировки методом касательных $T \approx 0{,}1M$ с на рабочем поле $10^5 \times 10^5$ дискретов и более. Метод касательных базируется на двух процедурах:

1. Для очередной цепи i строится сокращенное рабочее поле $\overline{\text{РП}}_i \subset \text{РП}_i$, содержащее, по крайней мере, одно ортогональное МШД из множества ортогональных МШД цепи i полного (не сокращенного) рабочего поля РП_i.

2. В $\overline{\text{РП}}_i$ строится МШД (или близкое к нему) цепи i.

Процедуры 1, 2 применяются для всех цепей. Основными проблемами такой постановки задачи являются:

1. Создание идеологии построения корректно сокращенного рабочего поля. Задача усложняется из-за отсутствия практически приемлемых алгоритмов нахождения МШД.

2. Разработка машинного алгоритма, такого, чтобы время построения $\overline{\text{РП}}_i$ было существенно меньше синтеза в нем МШД.

3. Разработка машинного алгоритма синтеза МШД в $\overline{\text{РП}}_i$, такого, чтобы суммарное время трассировки было существенно меньше, чем в волновых алгоритмах, и не зависело от размера РП.

Указанные проблемы решены в [172—174]: в [172] определены необходимые и достаточные условия построения корректно сокращенного $\overline{\text{РП}}_i$, в [173] приведен конструктивный алгоритм синтеза корректно сокращенного $\overline{\text{РП}}_i$, в [174] описан алгоритм синтеза МШД в последнем. Ранее метод касательных был реализован в САПР АСП-2, АСП-52 [175—177] для автоматического проектирования топологий двухслойных микросборок и печатных плат.

Следует отметить, что все подсистемы, кроме АСП-62, могут быть использованы с небольшой доработкой для проектирования

БИС на основе других типов БМК, а также для проектирования БИС, СБИС методом стандартных блоков. Система АСП-6М является составной частью комплексной САПР АСП-6, включающей системы проектирования двухслойных и многослойных микросборок, печатных плат и гибридных ИС. В настоящее время ведутся работы по адаптации и включению подсистемы АРТИС, описанной в гл. 5, в систему АСП-6М, что позволит сделать последнюю более гибкой и универсальной.

СПИСОК ЛИТЕРАТУРЫ

1. **Баталов Б. В., Норенков И. П.** Система автоматизированного проектирования сверхБИС//Микроэлектроника. — 1980. — № 9. — С. 401—412.
2. **Кармазинский А. Н.** Синтез принципиальных схем цифровых элементов на МДП-транзисторах. — М.: Радио и связь, 1983. — 256 с.
3. **Автоматизация** проектирования больших и сверхбольших интегральных схем/А. И. Петренко, В. М. Курейчик, А. Я. Тетельбаум и др.//Зарубежная радиоэлектроника. — 1981. — № 6. — С. 47—66.
4. **Автоматизация** проектирования топологии БИС на базовых матричных кристаллах/А. И. Петренко, А. Я. Тетельбаум, Б. Л. Шрамченко, Н. В. Луганский//Зарубежная радиоэлектроника. — 1985. — № 8. — С. 26—40.
5. **Применение** вычислительных машин для проектирования цифровых устройств/Под ред. Н. Я. Матюхина. — М.: Сов. радио, 1968. — 256 с.
6. **Автоматизация** конструирования БИС/А. И. Петренко, П. П. Сыпчук, А. Я. Тетельбаум и др. — Киев: Вища школа, 1983. — 312 с.
7. **Селютин В. А.** Машинное конструирование электронных устройств. — М.: Сов. радио, 1977. — 384 с.
8. **Селютин В. А.** Автоматизированное проектирование топологии БИС. — М.: Радио и связь, 1983. — 112 с.
9. **Абрайтис Л. Б.** Подсистема автоматизации проектирования топологии БИС ячеечного типа//Управляющие системы и машины. — 1974. — № 5. — С. 49—54.
10. **Состояние и перспективы развития базовых кристаллов цифровых БИС на МДП-транзисторах/М. В. Алюшин, А. Н. Кармазинский, Ю. А. Кулагин, К. К. Салгус. — М.: ЦНИИ «Электроника», 1981. (Обзоры по электронной технике. Сер. 3. Микроэлектроника). Вып. 5. — 54 с.**
11. **Крегер Д., Тозун О.** Автоматизированное проектирование обостряет конкуренцию приборов со стандартами.//Электроника. — 1980. — Т. 33, № 15. — С. 28—34.
12. **Vernon I.** A ULA is More than Silicon//Proc. 5th SSCC. — 1979. — P. 51.
13. **Hurst S.** Universal Logic Gates for Custom Disign I. C. Requirements//Microel. Reliab. — 1980. — Vol. 19. — P. 457.
14. **Файзулаев Б. Н.** Влияние межэлементных связей кристалла на быстродействие логических БИС//Вопросы радиоэлектроники. Сер. ЭВТ. — 1982. — Вып. 16. — С. 91—94.
15. **Кордеро У.** Совершенно новая инфраструктура процессора 4341//Электроника. — 1979. — Т. 52, № 23. — С. 24—33.
16. **Жуковский В. А., Кушнер Ю. К., Бубенников А. Н.** Биполярные матричные БИС — элементно-конструктивная база высокопроизводительных ЦВМ четвертого поколения//Зарубежная радиоэлектроника. — 1979. — № 11. — С. 3—21.
17. **Иванов Ю. П., Черников В. П.** Базовый кристалл сверхбыстродействующих БИС — перспективная элементная база арифметических устройств ЭВМ IV поколения//Электронная техника. Сер. 3. Микроэлектроника. — 1979. — № 3. — С. 11—19.
18. **Выбор** элементной базы быстродействующих микропроцессорных БИС/ И. И. Шагурин, Ю. П. Иванов, Г. П. Мозговой, В. Г. Немудров//Микроэлектроника и полупроводниковые приборы: Сб. статей/Под ред. А. А. Васенкова. — М., 1977. — Вып. 2. — С. 65—80.

19. **Автоматизация** проектирования вычислительных структур/А. В. Каляев, А. Н. Мелихов, В. М. Курейчик, В. Ф. Гузик, В. А. Калашников. — Ростов н/Д: Изд-во Ростовского ун-та, 1983. — 224 с.
20. **Конструирование** и расчет БГИС, микросборок и аппаратуры на их основе/ Под ред. Б. Ф. Высоцкого. — М.: Радио и связь, 1981. — 211 с.
21. **Файзулаев Б. Н., Павлычев В. А., Драбкин В. А.** Оценка средней длины связей в логических цепях ЭВМ//Вопросы радиоэлектроники. Сер. ЭВТ. — 1982. — Вып. 16. — С. 95—99.
22. **Батищев Д. И.** Поисковые методы оптимального проектирования. М.: Сов. радио, 1975. — 216 с.
23. **Морозов К. К., Одиноков В. Г., Курейчик В. М.** Автоматизированное проектирование конструкций радиоэлектронной аппаратуры. — М.: Радио и связь, 1983. — 280 с.
24. **Harada N. A** New Average Interconnection Lenght Prediction Method for Masterstice LSI//Proc. ISCAS. — 1982. — P. 760—763.
25. **Gamal A. E.** Two-dimensional Stochastic Model for Interconnections in Master Slice Intergrated Circuits//IEEE Trans. — 1981. — Vol. CAS-28. — P. 127—137.
26. **Вермишев Ю. Х.** Вычислительные процессы машинного проектирования радиоэлектронной аппаратуры//Обмен опытом в радиопромышленности. — 1975. — № 6. — С. 24—28.
27. **Рубцов В. П., Захаров В. П., Жижко В. А.** Автоматизация проектирования больших интегральных схем. — Киев: Техника, 1980. — 233 с.
28. **Автоматизированная** система проектирования фотошаблонов на базе ЭВМ БЭСМ-6/Г. Г. Казеннов, Б. В. Баталов, В. Е. Щербаков, Л. В. Еремина// Микроэлектроника: Сб. статей/Под ред. А. А. Васенкова. — М.: 1976. — Вып. 9. — С. 22—26.
29. **Норенков И. П.** Введение в автоматизированное проектирование технических устройств и систем. — М.: Высшая школа, 1980. — 309 с.
30. **Глориозов Е. Л., Ссорин В. Г., Сыпчук П. П.** Введение в автоматизацию схемотехнического проектирования. — М.: Сов. радио, 1976. — 223 с.
31. **Вермишев Ю. Х.** О структурах графической и текстовой информации в САПР//Обмен опытом в радиопромышленности. — 1977. — Вып. 3. — С. 22—23.
32. **Петренко А. И., Тетельбаум А. Я.** Формальное конструирование ЭВА. — М.: Сов. радио, 1979. — 258 с.
33. **Тетельбаум А. Я., Шрамченко Б. Л.** Методы машинного проектирования электронных устройств//Зарубежная радиоэлектроника. — 1977. — № 2. — С. 24—50.
34. **Морозов К. К., Одиноков В. Г.** Использование ЭЦВМ при конструировании некоторых узлов радиоэлектронной аппаратуры. — М.: Сов. радио, 1972. — 154 с.
35. **Абрайтис Л. Б., Шейнаускас Р. И., Жилявичюс В. А.** Автоматизация проектирования ЭВМ. — М.: Сов. радио, 1978. — 272 с.
36. **Зайцева Ж. Н., Бодрягин В. И., Фомин К. Г.** Нисходящая технология проектирования адаптируемой САПР на базе типовых алгоритмических решений//Вопросы радиоэлектроники. Сер. ЭВТ. — 1981. — Вып. 6. — С. 3.
37. **Зайцева Ж. Н., Сабитов В. Н.** Практические приемы организации программы динамической структуры на языке ПЛ/1 для транслятора Г//Вопросы радиоэлектроники. Сер. ЭВТ. — 1979. — Вып. 6. — С. 17—22.
38. **Бершадский А. М.** Применение графов и гиперграфов для автоматизации РЭА и ЭВА. — Саратов: Изд-во Саратов. ун-та, 1983. — 120 с.
39. **Бодрягин В. И., Зайцева Ж. Н., Мазанько Н. С.** Автонастройка монтажного поля базового кристалла матричной БИС в системе автоматизированного проектирования//Вопросы радиоэлектроники. Сер. ЭВТ. — Вып. 10. — 1982. — С. 37—42.
40. **Разработка** системы автоматизированного проектирования большого числа типов логических СБИС для МВК «Эльбрус»/Г. Г. Рябов, В. Е. Вулихман, А. Н. Воздвиженский и др. — М.: Изд-во ИТМ и ВТ АН СССР, 1982. — 169 с.

41. **Песков М. И.** Опыт разработки и внедрения системы автоматизации проектирования//Обмен опытом в радиопромышленности. — 1971. — № 7. — С. 71—78.
42. **Песков М. И.** Организация разработки и эксплуатации системы автоматизированного проектирования многослойных печатных плат.//Вычислительная техника/Каунас. политехн. ин-т. — 1973. — Т. 6. — С. 5—10.
43. **Вакка Э., Линкольн Н.** Суперкомпьютер превосходит самого себя, проектируя своего преемника//Электроника. — 1981. — Т. 54, № 13. — С. 43—44.
44. **Миллер Д., Рубан Д.** Быстродействующая эффективная система структурного логического проектирования//Электроника. — 1981. — Т. 54, № 23.
45. **Ohna Y., Miyoshi M., Sata K.** Logic Verification Systems for Very Lagre Computers Using LST//Proc. 16th DAC, 1979. — P. 367—374.
46. **Лакшин Г. Л., Коротаев Ю. С.** Система логического моделирования устройств МВК «Эльбрус». — М.: Изд-во ИТМ и ВТ АН СССР, 1979. — 97 с.
47. **Карп Р. М.** Сводимость комбинаторных проблем: Пер. с англ.//Кибернетический сб. — 1975. — Вып. 12. — С. 16—38.
48. **Анисимов В. И., Перков Н. К., Соколов В. В.** Организация пакета прикладных программ для автоматизированного проектирования электронных схем//Разработка, эксплуатация и развитие САПР РЭА. — М.: МДНТП, 1978. — С. 97—99.
49. **Деньдобренко Б. Н., Малика А. С.** Автоматизация конструирования РЭА. — М.: Высшая школа, 1980. — 382 с.
50. **Баталов Б. В., Магрупов Т. М.** Определение соответствия топологии большой интегральной схемы принципиальной электрической схеме//Микроэлектроника, 1974. — Т. 3, вып. 4. — С. 305—311.
51. **Muehldorf E., Savkar A.** LSI Logic Testing an Overwiew//IEEE Trans., 1981. — Vol. C-30, N 1. — P. 1—16.
52. **Galiay I.** Physical Versus Logical Fault Models in MOS LSI Circuits: Impact on their Testability//IEEE Trans. — 1980. — Vol. C-29, N 6. — P. 527—531.
53. **Панферов В. П., Мухин Ю. А.** Метод контроля соответствия двух принципиальных схем//Электронная техника. Сер. 19. Микроэлектронные устройства. — 1978. — Вып. 1(7). — С. 54—58.
54. **Kubo N., Shiracova I., Ozaki H.** A Fast Algorithm for Testing Grapf Izomorphism//Proc. ISCAS. — 1979. — P. 641—644.
55. **Kawamura M., Hirabayashi K.** MACTIS — a Mask Checking Timing Simulator//IEEE Trans. — 1980. — Vol. CAS-27, N 12. — P. 1276—1278.
56. **Юдин Д. Б., Гольштейн Е. Г.** Задачи и методы линейного программирования. — М.: Сов. радио, 1961.
57. **Yoshizawa K.** Automatic Layout Algorithms for Master Slice LSI//Proc. ISCAS. — 1979. — P. 470—473.
58. **Goto S.** An Automatic Layout Design System for Master Slice LSI Chip// Proc. ISCAS. — 1982. — P. 631—635.
59. **Hanan M., Wolff P. K., Agujle B. I.** A Study of Placement Techniques//J. Des. Aut. and Fault Tol. Comput. — 1976. — Vol. 1, N 1. — P. 65—82.
60. **Opitz F., Volker A.** AULIS Automatic Placement and Routing of Gate Arrays//Proc. ECCCTD. — 1983. — P. 331—337.
61. **Абрайтис Л. Б.** Алгоритм для определения максимально связанных наборов элементов//Автоматика и вычислительная техника. — 1980. — № 5. — С. 53—59.
62. **Методы** разбиения/К. К. Морозов, А. Н. Мелихов, Л. С. Берштейн и др.; Под ред. К. К. Морозова. — М.: Сов. радио, 1978. — 136 с.
63. **Smith D.** The Variable Geometry Automated Universal Array Layout System//IEEE Trans. — 1984. — Vol. CAD-3, N 1. — P. 20—26.
64. **Kambe T.** A Placement Algorithm for Polycell LSI and its Evaluation//Proc. 19th DAC. — 1982. — P. 655—662.
65. **Moullion M.** Computer Aided Layout for Single Layer Arrays//Proc. ECCTD.— 1983. — P. 338—342.
66. **Kawamoto T., Kajitani Y.** Minimum Width Routing of the 2-Row 2-Layer Polycell — Layout//Proc. ISCAS. — 1979. — P. 685—688.

67. **Петренко А. И., Тетельбаум А. Я., Забалуев Н. Н.** Топологические алгоритмы трассировки. МПП. — М.: Радио и связь, 1983. — 178 с.
68. **Базилевич Р. П.** Алгоритмические методы гибкой трассировки межсоединений. — Киев: Институт Кибернетики АН УССР, 1979. — 52 с.
69. **Дэнски А.** Уменьшение нагрузочных сопротивлений вентилей — способ снижения задержек в вентильных матрицах//Электроника. — 1980, № 22. — С. 38—41.
70. **Рябов Г. Г., Лакшин Г. Л.** Размещение и трассировка в классе горизонтально-вертикальных деревьев. — М.: Изд-во ИТМ и ВТ АН СССР, 1978. — 64 с.
71. **Зайцева Ж. Н., Алферова М. Ф.** Метод параллельной трассировки на платах любого конструкторского уровня//Вопросы радиоэлектроники. — 1979. — Вып. 6. — С. 51—57.
72. **Хонг С. Ю., Наир Р.** Трассировочные машины — новый инструмент для физического проектирования СБИС//ТИИЭР. — 1983. — Т. 71, № 1. — С. 70—80.
73. **Donze F.** PHILO — a VLSI Design System//Proc. 19th DAC. — 1982. — P. 163—169.
74. **Tsukiyama S.** An Algorithm of Global Routing for Master Slice LSI//Proc. ISCAS. — 1982. — P. 1009—1012.
75. **Matsyda T.** Lambda: A Quick, Low Cost Layout Design System for Master — Slice LSIs//Proc. 19th DAC. — 1982. — P. 802—808.
76. **Korn R.** An Efficient Variable — Cost Maze Router//Proc. 19th DAC. — 1982. — P. 425—431.
77. **Рябов Л. П., Темницкий Ю. Н.** Автоматическое редактирование печатного монтажа//Обмен опытом в радиопромышленности. — 1978. — Вып. 4—5. — С. 86—88.
78. **Tanaca C.** An Integrated Computer Aided Design System for Gate Array/ Masterslices Pt 2. The Layout Design System Mars-m3//Proc. 19th. DAC. — 1982. — P. 812—819.
79. **Wiesed M.** Automatic Routing of Gate Arrays//Proc. ECCTD. — 1983. — P. 328—330.
80. **Широ Г. Э.** Методы трассировки печатных плат и КМОП БИС//Электронная техника. Сер. 10, Микроэлектронные устройства. — 1978. — Вып. 2 (8). — С. 3—11.
81. **Широ Г. Э.** Трассировка попарно-многослойных печатных плат с использованием стандартных параллельных моделей//Электронная техника. Сер. 10. Микроэлектронные устройства. 1980. — Вып. 2(20). — С. 77—86.
82. **Yoshimura T.** An Efficient Channel Router//Proc. 21th DAC. — 1984. — P. 38—44.
83. **Deutsch R.** A Dogleg «Optimal» Channel Router with Completion Enhancements//Proc. 18th DAC. — 1981. — P. 762—768.
84. **Rivest R., Fiduccia Ch.** A «Greedy» Channel Router//Proc. 19th DAC. — 1982. — P. 418—424.
85. **Marec-Sadowsca M., Kuh E.** A New Approach to Channel Routing//Proc. ICCC. — 1982. — P. 764—766.
86. **Судо Ц.** Системы автоматизированного проектирования СБИС в Японии// ТИИЭР. — 1983. — Т. 71, № 1. — С. 159—178.
87. **Sansen W.** Design Automation Software Towards MOS/VLSI//Proc. ICCC. — 1980. — Vol. 1. — P. 98—102.
88. **Todd I.** CGAL — A Mutly Technology Gate Array Layout System//Proc. 19th. DAC. — 1982. — P. 792—801.
89. **Петрухин В. П.** Программная система АЛПАР для конструирования макротопологии БИС на основе библиотечного набора элементов//Вычислительная техника/Каунас. политехн. ин-т. — 1978. — Т. 11. — С. 46—48.
90. **Петрухин В. П.** Оптимизация размещения при конструировании БИС на основе библиотечного набора элементов//Электронная техника. Сер. 10. — 1978. — Вып. 4. — С. 70—76.
91. **Дэвис К.** Матрица логических вентилей — процессор «системы 370»//Электроника. — 1980. — № 22. С. 29—34.

92. **Фойер Н.** Программа автоматизированной трассировки связей кристалла с 5 тыс. логических вентилей//Электроника. — 1980. — № 22. — С. 35—38.
93. **Hightower D., Alexander F.** A Mature I^2L/STL Gate Array Layout System// Proc. ISCAS. — 1980. — P. 149—155.
94. **Chu K., Sharma R.** A Technology Independent MOS Multiplier Generator// Proc. 21th DAC. — 1984. — P. 90—97.
95. **Гранди Д. Л., Бранис Дж.** Биполярная КИД-технология, открывающая перспективы создания матричных СБИС на 100 тыс. вентилей//Электроника. — 1983. — № 14. — С. 55—61.
96. **Ohuo V.** Integrated Design Automation System for Custom and Gate Array VLSI Design//Proc. ICCC. — 1982. — P. 512—515.
97. **Анисимов В. И.** Подсистема проектирования печатных плат конструктивных узлов электронной аппаратуры: Реализация на АРМ-Р//Вычислительная техника/Каунас. политехн. ин-т. — 1982. — Т. 15. — С. 21—22.
98. **Nair R.** Global Wiring on a Wire Routing Machine//Proc. 19th DAC. — 1982. — P. 224—231.
99. **Breuer M., Shamsa K.** A Hardware Router//J. Digital Systems. — 1981. — Vol. 4, N 4. — P. 393—408.
100. **Смит.** Кремниевый компилятор, сокращающий продолжительность разработки СБИС//Электроника. — 1983. — № 23. — С. 15—17.
101. **Donath W. E.** Placement and Average Interconnection Lengths of Computer Logic//IEEE Trans. — 1979. — Vol. CAS-26, N 4. — P. 272—277.
102. **Тетельбаум А. Я.** Иерархический подход к проектированию сверхбольших интегральных схем//Электронная техника. — Сер. 10, Микроэлектронные устройства. — 1981. — Вып. 6(30). — С. 29—32.
103. **Ахо А., Хопкрофт Дж., Ульман Дж.** Построение и анализ вычислительных алгоритмов. — М.: Мир, 1979. — 536 с.
104. **Пономарев М. Ф., Коноплев Б. Г., Фомичев А. В.** Базовые матричные кристаллы: Проектирование специализированных БИС на их основе. — М.: Радио и связь, 1985. — 80 с.
105. **Уланов А. П.** Автоматизация разработки программ контроля матричных БИС//Теоретические и прикладные вопросы разработки и эксплуатации САПР РЭА. (Тезисы докл. Всесоюзной конф.) — М.: МАИ, 1986. — С. 101—102.
106. **Даттон Р., Хансен С.** Моделирование процессов изготовления интегральных схем//ТИИЭР. — 1981. — Т. 69, № 10. — С. 145—163.
107. **Берисфорд Р.** Достижения в области заказных БИС и СБИС, расширяющие возможности разработчиков систем//Электроника. — 1983. — Т. 56, № 5. — С. 36—50.
108. **Петренко А. И.** Средства моделирования сверхбольших интегральных схем в процессе их проектирования//Зарубежная радиоэлектроника, 1983. — № 12. — С. 10.
109. **Friedman T., Yang S.** Quality of Design From an Automatic Logic Generator//IBM J. Res. Dev. — 1985. — Vol. 3, N 5. — P. 415—422.
110. **Hill F., Peterson G.** Digital Systems: Hardware Organization and Design. — N. Y.: Wiley, 1978. — 428 p.
111. **Глушков В. М., Летичевский А. А., Капитонова Ю. В.** Автоматизация проектирования ЭВМ. — Киев: Наука, 1979. — 168 с.
112. **Hafer L., Parcer A.** Automated Synthesis of Digital Hardware//IEEE Trans.— 1982. — Vol. C-31, N 2. — P. 93—109.
113. **Zimmerman G.** Computer-aided Design of Control Structures for Digital Computers of Control Structures//Proc. ICCC. — 1980. — P. 781—784.
114. **Darringer J. A.** Experiments in Logic Synthesis//Proc. ICCC. — 1980. — P. 234—237.
115. **Kang S., Weich S.** Automatic PLA Synthesis from a DDL-P Description// Proc. 18th DAC. — 1981. — P. 391—397.
116. **Goates G., Patil S.** ABLE — a Lisp-based Layout Modeling Language with User Definable Models for SLA Design//Proc. 18th DAC. — 1981. — P. 322—329.
117. **Mead C., Conway L.** Introduction to VLSI Systems. — Boston: Addison-Wesley, 1980. — 457 p.

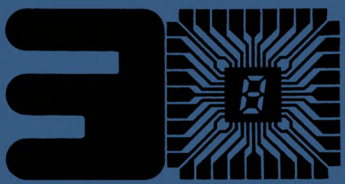
118. **Петренко А. И.** Основы автоматизации проектирования. — Киев: Техника, 1982. — 295 с.
119. **Fujinami Y.** A Logic Simulation System for MOS LSI Circuits with Bidirectional Elements//Monograph of Technical Groap on Circuits and Systems of of Inst. Electron. Commun. Eng. Jap. — 1982. — P. 117—128.
120. **Watanabe J.** Seven Value Logic Simultation for MOS LSI Circuits//Proc. ICCC. — 1980. — P. 941—944.
121. **Szygenda S., Thompson E.** Digital Logic Simulation in a Time Based Table Driver Environments//IEEE Trans. — 1975. — Vol. C-24, N 7. — P. 659—667.
122. **Sasaci T.** MIXS: a Mixed Level Simulator for Large Digital System Logic Verification//Proc. 17th DAC. — 1980. — P. 626—633.
123. **Miyashita H.** PLACAD : PLA Design Automation Systems//Proc. NCIECE. — 1981. — P. 410.
124. **Уильямс Т., Паркер К.** Проектирование контролепригодных устройств// ТИИЭР — 1983. — Т. 71, № 1. — С. 122—139.
125. **Vladimirescu A., Pederson D.** Performance Limits of the CLASSIC. Circuit Simulation Program//Proc. ISCAS. — 1982. — P. 1229—1232.
126. **Nagel L.** SPICE-2: a Computer Program to Simulate Semiconductor Circuits. — Berkley: Univ. of California, 1975. — 142 p.
127. **Yang P., Hajj I., Trick T.** SLATE: a Curcuit Simulation Program with Latency Appraximation and Node Tearing//Proc. ICCC. — 1980. — P. 353—355.
128. **De Man H.** DIANA: A Mixed-Mode Simulator with a Hardware Description Language for Hierarchical Design of VLSI//Proc. ICCC. — 1980. — P. 356—360.
129. **Newton A.** Technique for the Simulation of Large-Scale Integrated Circuits// IEEE Trans. — 1979. — Vol. CAS-26. — N 8. — P. 741—749.
130. **Lelarasmee E., Ruehli A., Sangiovanni-Vincentelli A.** The Waveform Relaxion Method For Time-Domain Analysis of Large-Scale Intergreted Circuits//IEEE Trans. — 1982. — Vol. CAD-1, N 2. — P. 131—145.
131. **Rabbat N., Hsieh H.** A Latent Macromodule — Approach to Large-Scale Sparse Networks//IEEE Trans. — 1976. — Vol. CAS-22, N 8. — P. 745—752.
132. **Хэчтел Г., Санджованни — Винченечелли А.** Обзор методов моделирования третьего поколения//ТИИЭР. — 1981. — Т. 69, № 10. — С. 11—110.
133. **Sakallan K., Director S.** An Event Driven Approach for Mixed Gate and Circuit Level Simulation//Proc. ISCAS. — 1982. — P. 1194—1197.
134. **Энгль В., Диркс Х., Майперцхлен Б.** Моделирование полупроводниковых приборов//ТИИЭР. — 1983. — Т. 71, № 1. — С. 14—42.
135. **Yoshii A.** A Three-Dimensional Analysis of Semiconductor Devises//IEEE Trans. — 1982. — Vol. ED-29, N 2. — P. 184—189.
136. **Engl W., Laur R., Dirks H.** MEDUSA — a Simulator for Modular Circuits// IEEE Trans. — 1982. — Vol. CAD-1, N 1. — P. 83—95.
137. **Selberherr S., Fichtner W., Potzl H.** MINI MOS — a Programm Package to Facilitate MOS Design and Analysis//Numerical Analysis of Semiconductor Devices/B. T. Broun and J. Miller, Eds. — Dublin: Boole Press, 1979. — P. 214—221.
138. **Yokoyama K., Yosmi A., Horigudi S.** Tresholl Sensitivity Minimization of Short-Channel MOSFETs by Computer Simulation//IEEE Trans. — 1980. — Vol. ED-27, N 9. — P. 1505—1514.
139. **Buturda E.** Three-Dimentional Finite Element Simulation of Semiconductor Devices//Proc. ISSCC. — 1980. — P. 76—77.
140. **Oldhat W.** A general simulator for VLSI lithography and etching process// IEEE Trans. — 1980. — Vol. ED-27, N 8. — P. 1455—1459.
141. **Баталов Б. В., Егоров Ю. Б., Русаков С. Г.** Основы математического моделирования больших интегральных схем на ЭВМ. — М.: Радио и связь, 1982. — 168 с.
142. **Петренко А. И., Тетельбаум А. Я., Шрамченко Б. Л.** АРТИС — подсистема иерархического проектирования топологии СБИС на базовом кристалле// Вычислительная техника/Каунас. политехн. ин-т, — 1982. — Т. 15. — С. 32—33.
143. **Петренко А. И., Тетельбаум А. Я., Шрамченко Б. Л.** Методические указания по использованию системы АРТИС при решении задач автоматическо-

то проектирования СБИС. — Киев: Изд-во Киевск. политехн. ин-та, 1983. — 56 с.
144. **Петренко А. И., Тетельбаум А. Я., Шрамченко Б. Л.** Автоматизированная система АРТИС конструирования СБИС на базовом кристалле//Автоматизация конструкторского проектирования в радиоэлектронике и вычислительной технике. — Вильнюс: Изд-во Литовск. гос. ун-та, 1983. — С. 42—51.
145. **Тетельбаум А. Я., Шрамченко Б. Л., Луганский Н. В.** Метод снижения размерности задачи конструирования СБИС на базовом кристалле//Тезисы докл. 10-й научн.-техн. конф. «Автоматизация конструкторского проектирования РЭА и ЭВА». — Пенза: ПДНТП, 1983. — С. 3—5.
146. **Тетельбаум А. Я., Луганский Н. В.** Автоматизированное проектирование СБИС на базовом кристалле//Тезисы докл. Всесоюз. научн.-техн. конф. «Автоматизация проектирования ЭВА и Систем». — Ереван. — 1983. — Т. 2. — С. 96—97.
147. **Базилевич Р. П.** Декомпозиционные и топологические методы автоматизированного конструирования электронных устройств. — Львов: Вища школа, 1981. — 168 с.
148. **Ниссен К.** Методология и средства иерархического проектирования СБИС// ТИИЭР. — 1983. — Т. 72, № 1. — С. 81—94.
149. **Тетельбаум А. Я., Шрамченко Б. Л.** Иерархическая итерационная трассировка двусторонних печатных плат//Автоматизация конструкторского проектирования в радиоэлектронике и вычислительной технике/Каунас. политехн. ин-т. — 1984. — Т. 4. — С. 17—29.
150. **Применение** мини-ЭВМ «Электроника 100-25» в автоматизированных системах проектирования/В. Е. Межов, Н. А. Ратмиров, И. Л. Талов, Б. Л. Толстых//Электрон. пром-сть. — 1978. — Вып. 10. — С. 37—40.
151. **Программное** обеспечение системы 15УТ-4-017/В. Е. Межов, Н. А. Ратмиров, И. Л. Талов, Б. Л. Толстых//Электрон. пром-сть. — 1979. — Вып. 6. — С. 21—23.
152. **Система** иерархического моделирования БИС и СБИС на базе ЕС ЭВМ/ А. А. Коломасов, В. П. Золотов, В. Н. Лошаков, Г. Н. Брусникин//Математическое и машинное моделирование в микроэлектронике. — Вильнюс: Изд-во АН ЛитССР, 1985. — С. 25—29.
153. **Золотов В. П.** Язык описания входных сигналов для системы логического моделирования//Электронная техника. Сер. 10. Микроэлектронные устройства. — 1982. — Вып. 2(32). — С. 37—39.
154. **Рябов Г. Г., Лакшин Г. Л.** Логическое моделирование в КАСПИ-ЭВМ// Машинное моделирование. — М.: МДНТП, 1983. — С. 62—66.
155. **Szygenda S. A., Lekkos A. A.** Integrated Techniques for Functional and Gate Level Digital Logic Simulation//Proc. 10th DAW. — 1973. — P. 159—172.
156. **Functional** Simulation in the LAMP System/S. G. Chappel, F. R. Menon, J. E. Pellegrin, A. M. Schowe//Proc. DAC. — 1976. — P. 42—47.
157. **Van Cleemput W. M., Slutr E. A.** Initial Design Consideration for a Hierarchical IC Design System//Proc. AACCSC. — 1977. — P. 334—341.
158. **Коломасов А. А., Золотов В. П.** Логическое моделирование в системе автоматизации проектирования//Электронная техника. Сер. 10. Микроэлектронные устройства. — 1982. — Вып. 2(32). — С. 31—32.
159. **Лошаков В. Н., Брусникин Г. Н.** Автоматизация проектирования сверхбольших матричных БИС//Теоретические и прикладные вопросы разработки, внедрения и эксплуатации систем автоматизированного проектирования радиоэлектронной аппаратуры. — М.: Изд-во МАИ, 1983. — С. 35—37.
160. **Брусникин Г. Н., Замордуев С. А.** Алгоритмы контроля топологии матричных БИС на соответствие электрической схеме//Электронная техника. Сер. 10. Микроэлектронные устройства. — 1984. — Вып. 6. — С. 34—36.
161. **Системы** машинного проектирования топологий изделий электронной техники//В. Н. Лошаков, Г. Э. Широ, Л. Б. Осипов, Г. Н. Брусникин//Электронная техника. Сер. 10. Микроэлектронные устройства. — 1977. — Вып. 6. — С. 73—81.
162. **Штейн М. Е., Штейн Б. Е.** Методы машинного проектирования цифровой аппаратуры. — М.: Сов. радио, 1973. — 294 с.

163. **Юрин О. Н.** Единая система автоматизации проектирования ЭВМ. — М.: Сов. радио, 1978. — 175 с.
164. **Рябов Г. Г., Лакшин Г. Л.** Логическое поэлементное моделирование в комплексной автоматизированной системе проектирования высокопроизводительных ВС//Разработка, эксплуатация и развитие систем автоматизированного проектирования РЭА. — М.: МДНТП, 1978. — С. 26.
165. **Лошаков В. Н.** Параллельная трассировка соединений проводниками второго порядка//Электронная техника. Сер. 11. Комплексная микроминиатюризация радиоэлектронных устройств и систем. — 1975. — Вып. 4. — С. 52—65.
166. **Лошаков В. Н.** Алгоритм размещения линеек контактных площадок//Электронная техника. Сер. 11. Комплексная микроминиатюризация радиоэлектронных устройств и систем. — 1975. — Вып. 4. — С. 67—74.
167. **Лошаков В. Н.** Метод сокращения операций в алгоритме параллельной трассировки соединений и некоторые оценки//Электронная техника. Сер. 11. Комплексная микроминиатюризация радиоэлектронных устройств и систем. — 1976. — Вып. 1. — С. 51—58.
168. **Теория** и методы автоматизации проектирования вычислительных систем/ Под ред. М. Брейера. — М.: Мир, 1977. — С. 221—223.
169. **Унифицированные** интерактивные средства проектирования изделий электронной техники/Б. Л. Толстых, И. Л. Талов, В. Н. Харин, В. Е. Межов, Ю. Н. Черняев. — М.: Радио и связь, 1984. — 147 с.
170. **Roberts K. A., Baker T. E., Jerome D. M.** A Vertically Organized Computer—Aided Design Data Base//Proc. 18th DAC. — 1981. — P. 319—323.
171. **Бойко В. В., Саванков В. М.** Проектирование информационной базы автоматизированной системы на основе СУБД. — М.: Финансы и статистика. — 1980. — С. 46—75.
172. **Лошаков В. Н.** Сокращение рабочих полей больших размерностей методом касательных при трассировке соединений. Описание и обоснование метода//Электронная техника. Серия 10. Микроэлектронные устройства. — 1978. — Вып. 2. — С. 36—50.
173. **Лошаков В. Н.** Машинный алгоритм сокращения рабочих полей методом касательных при трассировке соединений//Электронная техника. Сер. 10. Микроэлектронные устройства. — 1978. — Вып. 2. — С. 51—66.
174. **Лошаков В. Н.** Система автоматизации проектирования больших гибридных интегральных схем с применением ЭВМ//Электрон. пром-сть. — 1970. № 2. — С. 45—49.
175. **Лошаков В. Н., Петрович Н. И.** Атоматизация проектирования межсоединений гибридных БИС//Электрон. пром-сть. — 1972. — № 4. — С. 16—21.
176. **Лошаков В. Н., Петрович Н. И.** Некоторые оценки машинного проектирования межсоединений гибридных БИС//Обмен опытом в радиоэлектронной промышленности. — 1973. — Вып. 3. — С. 48—50.
177. **Брусникин Г. Н., Лошаков В. Н., Широ Г. Э.** Об опыте машинного проектирования топологий гибридных больших интегральных схем//Электронная техника. Сер. 3. Микроэлектроника. — 1975. — Вып. 2(56). — С. 73—78.
178. **Sastry S., Parker A. C.** Stochastic Models for Wireability Analysis of Gate Arrays//IEEE Trans. — 1986. — Vol. CAD-5, N 1. — P. 52—65.
179. **Terai M.** A Method of Improving the Terminal Assignment in the Channel Routing for Gate Arrays//IEEE Trans. — 1985. — Vol. CAD-4, N 1. — P. 329—336.
180. **Owens R. M.** A System for Designing, Simulating, and Testing High Performance VLSI Signal Processors//IEEE Trans. — 1986. — Vol. CAD-5, N 3. — P. 420—428.
181. **Iosupovici A.** A Class of Array Architectures for Hardware Grid Routers// IEEE Trans. — 1986. — Vol. CAD-5, N 2. — P. 245—255.

ОГЛАВЛЕНИЕ

65 к.



Автоматизированное проектирование СБИС на базовых кристаллах

Издательство «Радио и связь»